Online Training on
Omics Data Analysis: Genome to Proteome
ऑनलाइन प्रशिक्षण
ओमिक्स डेटा विश्लेषण: जीनोम से प्रोटीओम तक

Sponsoring Project

ICAR Consortium Research Platform on Genomics

October 09-18, 2023

E-Manual

Dr. Girish Kumar Jha, Head (DABin), Course Director
Dr. Sudhir Srivastava, Course Coordinator
Dr. Sneha Murmu, Course Coordinator

Division for Agricultural Bioinformatics
ICAR-Indian Agricultural Statistics Research Institute
Library Avenue, PUSA, New Delhi - 110012

http://cabgrid.res.in/cabin/; https://iasri.icar.gov.in/

आमुख

भा.कृ.अनु.प.-भारतीय कृषि सांख्यिकी अनुसंधान संस्थान देश में कृषि सांख्यिकी, संगणक अनुप्रयोग और जैव सूचना विज्ञान के विषयों में कार्य करने वाला एक प्रमुख संस्थान है। संस्थान परीक्षण अभिकल्पना, नमूनाकरण तकनीक, सांख्यिकीय आनुवंशिकी, पूर्वानुमान तकनीक, जैव सूचना विज्ञान और संगणक अनुप्रयोग पर विशेष जोर देने के साथ कृषि सांख्यिकी में अनुसंधान, शिक्षण और प्रशिक्षण कार्यक्रम आयोजित करने में कार्यरत है। संस्थान बहुत सक्रिय रूप से सलाहकार सेवा प्रदान कर रहा है, जिसने संस्थान को राष्ट्रीय कृषि अनुसंधान और शिक्षा प्रणाली (एन.ए.आर.ई.एस) और राष्ट्रीय कृषि सांख्यिकी प्रणाली (एन.ए.एस.एस) दोनों में अपनी उपस्थिति महसूस कराने में सक्षम हुआ है। संस्थान ने कृषि अनुसंधान के लिए उपयोगी सांख्यिकीय सॉफ्टवेयर पैकेज विकसित करने में अग्रणी भूमिका निभाई है।

पिछले दो दशकों के दौरान जैविक विज्ञान में बृहद डेटा उत्पन्न किया गया जिसमें सबसे पहले जीवों के जीनोम अनुक्रमण की विषय मेयइन जानकारी प्राप्त की गयी। इसके उपरांत इन प्राप्त जानकारियों को उच्च प्रयोगात्मक तकनीक से जैव प्रोद्योगिकी अनुसंधान प्रयोगशालाओं में किए गए प्रयागों तथा इसके प्रभावों की गतिशीलता का अध्ययन किया जा रहा है। जैविक अनुसंधान के छेत्र में विभिन्न जैव सूचना विज्ञान तकनीकों/ टूल्स के प्रयोग, डेटा की संचयन एवं पुन:प्राप्ति, विश्लेषण, एनोटेशन और परिणाम के अपनी सम्पूर्णता में जैविक प्रणालियों को बेहतर ढंग में समझने में सहायक है। इसमे टिकाव कृषि के लिए टूल्स और तकनीकों के विकास को बढ़ावा मिलेगा। इस प्रशिछन का उद्देश्य ओमिक्स डेटा विश्लेषण और कृषि में इसके अनुप्रयोगों का अवलोकन प्रदान करना है। इससे सहभागियों को अनुसंधान, शिक्षण और प्रशिक्षण में अपनी क्षमताओं को उन्नत करने में मदद मिलेगी।

इस प्रशिक्षण में मुख्य रूप से जीनोमिक्स, ट्रांसक्रिपटॉमिक्स, मेटाजीनोमिक्स और प्रोटिओमिक्स डेटा विश्लेषण से संबंधित मॉड्यूल शामिल हैं। कृषि जैव सूचना विज्ञान से संबंधित अवधारणाओं, मुद्दों और समाधानों पर विशेष जोर दिया जाएगा। इस प्रशिक्षण कार्यक्रम में विभिन्न व्याख्यान शामिल किए गए हैं: जैव सूचना विज्ञान का परिचय, सुपर-कंप्यूटिंग सुविधा अशोका; लिनक्स और आर प्रोग्रामिंग भाषाओं की मूल बातें; जैविक डेटाबेस; अनुक्रम विश्लेषण; एनजीएस डेटा विश्लेषण का परिचय; जीनोम असेंबली और एनोटेशन; ट्रांसक्रिपटॉमिक्स डेटा, मेटागेनोमिक्स और नॉन-कोडिंग आरएनए डेटा का विश्लेषण; जीनोम-वाइड एसोसिएशन अध्ययन और जीनोमिक चयन; प्रोटीन संरचना प्रेडिक्शन; आणविक गतिशीलता और सिमुलेशन; प्रोटिओमिक्स एक्सप्रेशन डेटा विश्लेषण; पोस्ट-ट्रांसलेशनल संशोधन.

हम इस अवसर पर संस्थान के संकाय सदस्यों को धन्यवाद देना चाहते हैं जिन्होंने इस

PREFACE

The ICAR-Indian Agricultural Statistics Research Institute is a premier Institute in the disciplines of Agricultural Statistics, Computer Applications and Bioinformatics in the country. The Institute has been engaged in conducting research, teaching and organizing training programmes in Agricultural Statistics with special emphasis on Experimental Designs, Sampling Techniques, Statistical Genetics, Forecasting Techniques, Bioinformatics and Computer Applications. The Institute has been very actively pursuing advisory service that has enabled the institute to make its presence felt both in National Agricultural Research and Education System (NARES) and National Agricultural Statistics System (NASS). The Institute has taken a lead in developing Statistical Software Packages useful for Agricultural Research.

During the last two decades enormous sequence data have been generated in biological science, firstly with the onset of sequencing the genomes of living organisms and, secondly, rapid application of high throughput experimental techniques in laboratory research. Application of various bioinformatics tools in biological research enables storage, retrieval, analysis, annotation, and visualization of results and promotes a better understanding of biological systems in their entirety. This will further lead to the development of tools and techniques for sustainable agriculture. The aim of this training is to provide an overview of omics data analysis and its applications in agriculture. This would help them upgrade their capabilities in research, teaching, and training.

This training mainly consists of modules related to genomics, transcriptomics, metagenomics, and proteomics data analysis. Special emphasis will be laid on concepts, issues, and solutions related to agricultural bioinformatics. Various lectures were included in this training programme: Introduction to Bioinformatics, Super-Computing Facility ASHOKA; Basics of Linux and R Programming Languages; Biological Databases; Sequence Analysis; Introduction to NGS Data Analysis; Genome Assembly and Annotation; Analysis of Transcriptomics Data, Metagenomics and Non-coding RNA Data; Genome-Wide Association Studies and Genomic Selection; Protein Structure Prediction; Molecular Dynamics and Simulation; Proteomics Expression Data Analysis; Post-Translational Modifications.

We would like to take this opportunity to thank the faculty of the Institute who spared their valuable time in making this course meaningful and successful that helped in bringing out this manual in time. We are also thankful to the various ICAR Institutes, National Research Centres, Bureaus and State Agricultural Universities for deputing their employees in this training programme. We are grateful to Dr. Rajender Parsad, Director, ICAR-IASRI and Dr. Girish Kumar Jha, Head, Division of Agricultural Bioinformatics, ICAR-IASRI, Dr. Anu Sharma (PI of CRP Genomics Project at ICAR-IASRI) and Dr. Vindhya Mohindra (PI of CRP Genomics Project from lead center ICAR-NBFGR) for the support and cooperation in organizing this program. We are thankful to each one who supported directly or indirectly for preparing this training manual.

Authors

CONTENTS

Introduction to Bioinformatics

Girish Kumar Jha, Sneha Murmu, Soumya Sharma and Ritwika Das

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

History

- 1950-70s:

In the early 1950s, there was still controversy surrounding DNA's role as the carrier of genetic information. DNA's genetic role was firmly established in 1952 through the Hershey-Chase experiment. While the double-helix structure of DNA was revealed in 1953, it took more years to decipher the genetic code and develop DNA sequencing methods. Meanwhile, significant progress was made in protein analysis, especially with the publication of insulin's amino acid sequence in the late 1950s. This achievement spurred the development of protein sequencing methods, like the Edman degradation method, which allowed for automated sequencing of more than 15 protein families. However, a challenge with protein sequencing was assembling the complete sequence for large proteins, leading to the early development of bioinformatics software to address this issue.

Margaret Dayhoff, often referred to as the "mother and father of bioinformatics," was a physical chemist who recognized the potential of applying computational methods to biology and medicine. She collaborated with physicist Robert S. Ledley and together, in the late 1950s, they developed COMPROTEIN, one of the earliest bioinformatics software, for determining protein primary structure using Edman peptide sequencing data. They used this software to tackle the challenge of assembling complete sequences for large proteins, which was a significant computational problem. Dayhoff contributed to simplifying the handling of protein sequence data by developing the one-letter amino acid code, which is still in use today.

> The Birth of Sequence Databases:

Dayhoff and Eck's 1965 "Atlas of Protein Sequence and Structure" was the first biological sequence database. It contained 65 protein sequences, providing a basis for early computational analysis. Researchers began to consider the idea that protein sequences might reveal evolutionary history, similar to how language evolves, where the arrangement of letters conveys meaning.

> The Concept of Orthology:

Emile Zuckerkandl and Linus Pauling introduced the term "Paleogenetics" in 1963 to explore the evolutionary aspects of biomolecular sequences. They observed that orthologous proteins from different species showed varying degrees of similarity, correlating with their evolutionary divergence. Orthology, defined by Walter M. Fitch in 1970, described homology resulting from speciation events. This observation led to the hypothesis that orthologous proteins evolved from a common ancestor, and their sequences could be used to predict ancestral sequences and trace evolutionary history.

> Challenges in Sequence Alignment:

Initial efforts in sequence-based phylogenetic studies focused on closely related proteins that could be assessed visually for homology. However, for more distant or unequal-length protein sequences, visual comparison was impractical and often led to errors.

In 1970, Needleman and Wunsch developed the first dynamic programming algorithm for pairwise protein sequence alignments. Multiple sequence alignment (MSA) algorithms emerged in the early 1980s, addressing the challenge of aligning numerous sequences of different lengths more efficiently. In 1987, Da-Fei Feng and Russell F. Doolitle developed a practical approach to multiple sequence alignment (MSA) known as "progressive sequence alignment." Their method involved several steps:

- Performing a Needleman–Wunsch alignment for all possible sequence pairs.
- Extracting pairwise similarity scores from each of these pairwise alignments.
- Using these similarity scores to construct a guide tree, which represents the relationships between sequences.
- Aligning the sequences in a stepwise manner, starting with the two most similar sequences and then progressively adding the next most similar sequences according to the guide tree.

In 1988, the popular MSA software CLUSTAL was developed as a simplification of the Feng–Doolittle algorithm. CLUSTAL has remained in use and continued to be maintained up to the present day. This software made MSA more accessible and efficient, allowing researchers to align multiple sequences effectively.

➢ A Mathematical Framework for Amino Acid Substitutions (1978):

Margaret Dayhoff, Schwartz, and Orcutt developed the first probabilistic model of amino acid substitutions. The model was based on 1572 point accepted mutations (PAMs) in the phylogenetic trees of 71 protein families. They created a 20x20 asymmetric substitution matrix containing probability values based on observed amino acid mutations. This matrix introduced the concept of substitutions as a measurement of evolutionary change, shifting from the previous concept of evolutionary distance based on the least number of changes.

✚ Paradigm Shift from Protein to DNA Analysis (1970-1980):

Francis Crick's sequence hypothesis confirmed that DNA encodes information for proteins. DNA sequencing methods, including Maxam-Gilbert (1976) and Sanger's "plus and minus" method (1977), made DNA sequencing more accessible. The Sanger chain termination method (1977) remains in use today. DNA sequences could potentially provide information about all proteins in an organism. Manual tasks like comparisons, calculations, and pattern matching were more efficiently performed by computers.

➢ Development of Sequence Analysis Software (1979):

Roger Staden's software (1979) was one of the first to analyze Sanger sequencing reads. The software could search for overlaps, verify, edit, and join sequence reads, and annotate and manipulate sequence files. It introduced additional characters ("uncertainty codes") to record basecalling uncertainties in sequence reads. Staden's Package is still developed and maintained today.

➢ Using DNA Sequences in Phylogenetic Inference:

Early phylogenetic trees were reconstructed from protein sequences with a focus on maximum parsimony. Parsimony methods assumed minimal evolutionary changes but could fail with moderate to large changes. DNA sequences provided additional information, such as synonymous mutations. Joseph Felsenstein introduced maximum likelihood (ML) methods for phylogenetic tree inference from DNA sequences. ML estimation involved finding the tree with the highest probability of evolving the observed data. Bioinformatics tools and

statistical methods based on ML and Bayesian statistics have been developed and are still in use today.

> ➢ Overcoming Technical Limitations in the Late 1970s:

The late 1970s faced technical limitations that needed addressing to broaden computer use in DNA analysis. The subsequent decade played a pivotal role in addressing these issues and advancing the field.

> ➢ Molecular Methods for Targeting and Amplifying Specific Genes:

Genes are less abundant and cannot be individually sequenced, as they are contiguous on DNA molecules and present in low copies per cell. A solution emerged when Jackson, Symons, and Berg (1972) used restriction endonucleases and DNA ligase to cut and insert circular SV40 viral DNA into lambda DNA. E. coli cells were transformed with this construct, and the inserted DNA was replicated and amplified in the host organism. This experiment pioneered the isolation and amplification of genes independently from their source organism. Concerns about ethical issues led to a moratorium on the use of recombinant DNA, and guidelines were established during the 1975 Asilomar conference.

> ➢ Invention of Polymerase Chain Reaction (PCR):

The polymerase chain reaction (PCR) was a significant development that allows DNA amplification without cloning procedures. The first description of "repair synthesis" using DNA polymerase was in 1971 by Kjell Kleppe et al. The invention of PCR is credited to Kary Mullis for his substantial optimizations, including the use of thermostable Taq polymerase and the development of the thermal cycler. Mullis patented the process and gained recognition for inventing PCR. Both gene cloning and PCR are widely used in DNA library preparation, critical for obtaining sequence data.

🔸 DNA Sequencing and Bioinformatics in the 1980s:

The late 1970s saw the emergence of DNA sequencing, along with enhanced DNA manipulation techniques. DNA sequencing and manipulation led to increased availability of sequence data. Access to computers and bioinformatics software also grew during the 1980s, facilitating the analysis of sequence data.

🔸 1990-2000: Genomics, Structural Bioinformatics, and the Information Superhighway

> ➢ Dawn of the Genomics Era:

In 1995, the first complete genome sequencing of a free-living organism (Haemophilus influenzae) was achieved by The Institute for Genomic Research (TIGR), led by J. Craig Venter. The Human Genome Project, initiated in 1991 by the U.S. National Institutes of Health, aimed to sequence the human genome and cost $2.7 billion over 13 years. Celera Genomics led a private effort to sequence the human genome in competition with the publicly funded Human Genome Project, achieving it at one-tenth of the cost due to different experimental strategies.

> ➢ Challenges in Early Genomics:

Sequencing genomes was costly and time-consuming; for example, sequencing a human genome with 2018 technology would cost $1000 and take less than a week, but older methods were much slower. Specialized software was needed to handle the massive amount of

sequencing data. Several Perl-based software tools were developed in the mid to late 1990s for assembling whole-genome sequencing reads.

➢ Emergence of the Internet:

The rise of the World Wide Web (WWW) in the mid-1990s revolutionized communication and enabled the creation of online bioinformatics resources. Nucleotide sequence databases like EMBL and GenBank became accessible online in the early 1990s. The National Center for Biotechnology Information (NCBI) made its website and tools, including BLAST, available online in 1994. Major databases such as Genomes (1995), PubMed (1997), and Human Genome (1999) were established and are still in use today.

➢ Structural Bioinformatics:

Advances allowed computers to predict protein secondary and tertiary structures with varying degrees of certainty. Molecular dynamics simulations became possible, although they required significant computational resources. The use of graphics processing units (GPUs) and supercomputers aided in making molecular dynamics simulations more accessible.

🞣 2000-2010: High-Throughput Bioinformatics

➢ Second-Generation Sequencing:

Second-generation sequencing (next-generation sequencing or NGS) began with the '454' pyrosequencing technology. These technologies enabled the sequencing of thousands to millions of DNA molecules in a single machine run.

➢ Biological Big Data:

The drop in DNA sequencing costs and the adoption of massively parallel sequencing resulted in exponential growth in sequence data in public databases. Sequencing data has exceeded the exabyte (10^{18}) level. New repository infrastructure for model organisms and general genomic databases emerged to store, organize, and make data accessible. The Genomic Standards Consortium was established in 2005 to define the minimum information required for genomic sequences.

Aim

- Data acquisition and database development
  To organize data in a way that allows researchers to access existing information and to submit new entries as they are produced.

- Tool development
  To develop tools and resources that aid in the analysis of data.

- Data analysis
  To use different tools to analyze the data and interpret the results in a biologically meaningful manner

Branches

There are several branches of Bioinformatics (Figure 1). Some of them are explained below.
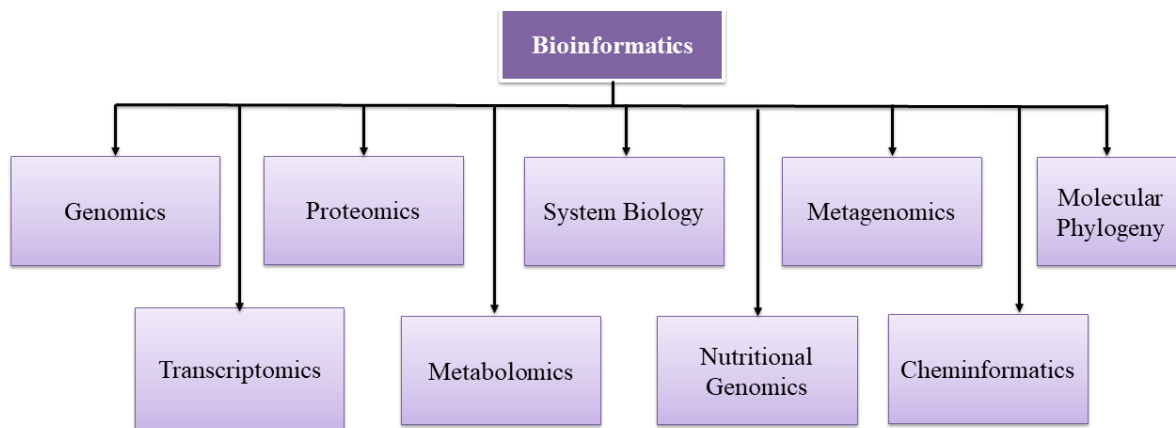
Figure 1: Branches of Bioinformatics

- Genomics

  Genomics is a fundamental field in bioinformatics that focuses on the study of an organism's entire genetic material, which is stored in its DNA (or RNA for some viruses). This genetic material, often referred to as the genome, contains all the information needed to build and maintain an organism. Genomics aims to understand and analyze the structure, function, evolution, and variations in the genome. Here are the key components of genomics in bioinformatics:

  - Sequencing: Genomic research often begins with DNA sequencing. This process involves determining the order of nucleotides (A, T, C, G) in a DNA molecule. There are various sequencing technologies, such as Sanger sequencing and next-generation sequencing (NGS), which allow scientists to read and decode the genetic information.

  - Genome Assembly: The raw sequencing data obtained is fragmented into smaller pieces, and the bioinformatics part of genomics involves assembling these pieces to create a complete genome. Genome assembly algorithms help organize and connect these sequences to form a coherent picture of the genome.

  - Functional Annotation: Once the genome is assembled, the next step is to identify and annotate the functional elements. This includes finding genes (coding regions), regulatory sequences, non-coding regions, and other structural components. Bioinformatics tools predict the locations of genes and their functions based on sequence similarity, conserved motifs, and other features.

  - Comparative Genomics: Genomic sequences of different organisms, both within the same species and across species, are compared to identify similarities and differences. Comparative genomics helps in understanding evolutionary relationships, studying gene conservation, and discovering genes responsible for specific traits or diseases.

  - Structural Genomics: Structural genomics focuses on determining the three-dimensional structures of proteins and other macromolecules encoded by the genome. This is crucial for understanding protein functions and interactions. Techniques like X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy are used in structural genomics.

- Functional Genomics: This field aims to understand the functions of genes and their products. Functional genomics methods, such as transcriptomics (studying gene expression), proteomics (studying proteins), and metabolomics (studying metabolites), provide insights into how genes are expressed and how they influence an organism's biology.

- Phylogenomics: Phylogenomics combines genomics and phylogenetics to study evolutionary relationships among species. It uses genomic data to reconstruct phylogenetic trees and understand the evolutionary history of different organisms.

- Genomic Variation: Genomic variation studies focus on identifying variations in the genome, such as single nucleotide polymorphisms (SNPs), insertions, deletions, and copy number variations. These variations can be associated with diseases and traits.

Transcriptomics

Transcriptomics is a branch of bioinformatics and genomics that focuses on the study of transcriptomes, which are the complete sets of RNA transcripts produced in a cell, tissue, or organism. These RNA transcripts, often referred to as messenger RNA (mRNA), provide critical information about which genes are actively being expressed and to what extent in a specific biological sample. Understanding transcriptomes is vital for unraveling the molecular mechanisms underlying various biological processes and diseases. Here are the key components of transcriptomics in bioinformatics:

- Data Generation: Transcriptomics begins with the generation of RNA sequencing (RNA-Seq) data. RNA-Seq is a high-throughput technology that allows researchers to identify and quantify the RNA molecules present in a biological sample. It provides information about gene expression levels, alternative splicing, and the presence of non-coding RNAs, among other things.

- Data Preprocessing: The raw RNA-Seq data typically contains errors, biases, and artifacts. Preprocessing involves cleaning and quality-checking the data to ensure its reliability. This step includes tasks like adapter removal, read alignment to the reference genome or transcriptome, and removal of duplicate reads.

- Gene Expression Quantification: After preprocessing, bioinformaticians quantify gene expression levels. This step involves determining the number of RNA fragments that map to each gene, which serves as a measure of the gene's activity. Different algorithms and tools are available for this purpose.

- Differential Expression Analysis: One of the key objectives in transcriptomics is to identify genes that are differentially expressed under different experimental conditions. This analysis helps researchers understand how gene expression is altered in response to various stimuli, diseases, or genetic mutations. Statistical methods are used to compare expression levels between conditions.

- Functional Analysis: Transcriptomics data can be further analyzed to gain insights into the biological functions and pathways affected by changes in gene expression. Tools and databases, such as gene ontology (GO) analysis and pathway enrichment analysis, help in understanding the roles of differentially expressed genes.

- Alternative Splicing Analysis: In addition to quantifying gene expression, transcriptomics also allows for the study of alternative splicing events. Alternative splicing can generate multiple mRNA isoforms from a single gene, expanding the functional diversity of the proteome.

- Long Non-Coding RNA (lncRNA) Analysis: Transcriptomics can reveal the presence and differential expression of long non-coding RNAs, which play crucial roles in gene regulation and various cellular processes.

Proteomics

Proteomics is a branch of bioinformatics and biology that focuses on the large-scale study of proteins. It involves the comprehensive analysis of the structure, function, and expression of all the proteins in a biological system, such as a cell, tissue, or organism. Proteins are crucial molecules in living organisms, responsible for performing various biological functions, and understanding their properties and behaviors is essential for gaining insights into complex biological processes. Here are some key aspects of proteomics in bioinformatics:

- Protein Identification and Characterization: Proteomics involves identifying and characterizing proteins. This can include determining the amino acid sequence, post-translational modifications (e.g., phosphorylation, glycosylation), and three-dimensional structures of proteins.

- Protein Expression and Quantification: Proteomic studies aim to measure the relative abundance of proteins in different biological conditions. This can help researchers understand how proteins are regulated and expressed under various circumstances, such as disease states or drug treatments.

- Protein-Protein Interactions: Proteins rarely function in isolation; they often work together in complexes. Proteomics helps in identifying protein-protein interactions, which are crucial for understanding cellular processes and signaling pathways.

- Functional Annotation: Assigning biological functions to proteins is a fundamental goal of proteomics. This may involve studying the role of proteins in specific pathways, cellular processes, and disease mechanisms.

- Biomarker Discovery: Proteomics plays a vital role in biomarker discovery for diseases. By comparing protein profiles in healthy and diseased samples, researchers can identify potential biomarkers for early disease diagnosis or monitoring treatment responses.

- Mass Spectrometry and Other Techniques: Mass spectrometry is a common technology used in proteomics. It allows the precise measurement of protein masses and has the capability to identify and quantify thousands of proteins simultaneously. Other techniques, like gel electrophoresis and antibody-based assays, are also used in proteomic studies.

↓ Metabolomics

Metabolomics is a subfield of bioinformatics that focuses on the comprehensive analysis of small molecules, known as metabolites, in biological systems. Metabolites include a wide range of compounds such as sugars, amino acids, lipids, organic acids, and other small molecules that play crucial roles in various biochemical processes within living organisms. Metabolomics aims to identify, quantify, and analyze these metabolites to gain insights into an organism's metabolism and understand its biochemical pathways, which can be essential for both basic research and practical applications. Following are the key concepts of metabolomics in bioinformatics:

- Data Generation: Metabolomics data is generated through various analytical techniques, such as mass spectrometry (MS), nuclear magnetic resonance spectroscopy (NMR), and liquid or gas chromatography. These techniques allow researchers to detect and quantify a wide range of metabolites present in a biological sample.

- Data Preprocessing: Metabolomics datasets can be large and complex, and preprocessing is a crucial step in data analysis. It involves data cleaning, alignment, normalization, and the removal of any technical variation or noise. This step ensures that the data is suitable for subsequent analysis.

- Metabolite Identification: One of the primary goals of metabolomics is to identify the metabolites detected in the sample. Bioinformatics tools and databases play a critical role in matching experimental data to known metabolite profiles. This process often involves spectral databases, reference libraries, and computational algorithms to make accurate identifications.

- Quantitative Analysis: Metabolomics data also provides quantitative information about the abundance of metabolites in a sample. Researchers can compare the concentration of specific metabolites across different samples or conditions to understand the metabolic changes.

- Statistical and Multivariate Analysis: Bioinformatics tools are used to analyze metabolomics data statistically. Techniques like principal component analysis (PCA), partial least squares-discriminant analysis (PLS-DA), and hierarchical clustering can reveal patterns and trends in the data, helping researchers identify biomarkers or distinguish between sample groups.

- Pathway Analysis: Metabolomics data can be integrated with other omics data, such as genomics and proteomics, to gain a more comprehensive understanding of the biological systems. Pathway analysis tools help researchers map metabolites onto known metabolic pathways, identifying key pathways and their interactions.

- Biomarker Discovery: Metabolomics is often applied to discover biomarkers, which are specific metabolites associated with a particular disease or condition. Identifying biomarkers can be valuable in disease diagnosis, prognosis, and treatment monitoring.

↓ System Biology
Systems biology is an interdisciplinary field in bioinformatics that focuses on understanding complex biological systems by studying how individual components, such

as genes, proteins, and metabolites, interact and function as a whole. It aims to provide a comprehensive and integrated view of biological processes to better explain and predict the behavior of living organisms. Following are the key aspects of systems biology in bioinformatics:

- Holistic Approach: Systems biology takes a holistic approach to biology, looking beyond the individual components. It considers the interactions, feedback loops, and dependencies among genes, proteins, and other molecules in biological systems.

- Data Integration: It involves integrating data from various sources, such as genomics, transcriptomics, proteomics, and metabolomics, to create a comprehensive picture of biological processes. This integration is often achieved through computational methods.

- Mathematical and Computational Modeling: Systems biology heavily relies on mathematical and computational modeling techniques. These models simulate biological processes and provide a framework for understanding and predicting system behavior. Examples of modeling techniques include differential equations, agent-based models, and network analysis.

- Network Analysis: Biological networks, such as protein-protein interaction networks and metabolic pathways, are a central focus of systems biology. Network analysis helps uncover relationships and patterns within complex biological systems.

- Dynamic Processes: Systems biology often deals with dynamic processes. It explores how biological systems change over time in response to various stimuli, environmental conditions, or genetic variations.

- Hypothesis Generation and Testing: Systems biology generates hypotheses about how biological systems work. These hypotheses can then be tested through experiments, helping to refine the models and improve our understanding of the system.

- Biomedical Applications: Systems biology has practical applications in medicine and drug discovery. It can be used to study complex diseases, identify potential drug targets, and optimize treatment strategies.

- Quantitative Biology: A quantitative approach is a hallmark of systems biology. It involves measuring and quantifying various biological components and processes, often using high-throughput technologies.

+ Nutritional Genomics

Nutritional genomics, often referred to as nutrigenomics, is a branch of genomics that focuses on the interaction between nutrition and genes. It aims to understand how an individual's genetic makeup influences their response to specific nutrients, foods, and dietary patterns. Nutritional genomics plays a significant role in agriculture by helping improve crop production and the nutritional quality of food. Following are the key points how it applies to agriculture in the context of bioinformatics:

- Genomic Sequencing of Crops: One of the key aspects of nutritional genomics in agriculture is the genomic sequencing of crop plants. Advances in bioinformatics and

genomics have made it possible to sequence the entire genomes of various crops, such as rice, wheat, and maize. This provides a comprehensive understanding of the genes and genetic variations present in these crops.

- Identification of Nutritional Genes: Bioinformatics tools are used to identify genes related to the nutritional content of crops. This includes genes that influence the levels of essential nutrients like vitamins, minerals, and proteins. By identifying these genes, researchers can target specific genetic traits for crop improvement.

- Marker-Assisted Breeding: Nutritional genomics, coupled with bioinformatics, facilitates marker-assisted breeding programs. Researchers can identify genetic markers associated with desirable nutritional traits in crops. This helps in the selection and breeding of crop varieties with improved nutritional content.

- Customized Diets for Livestock: Nutritional genomics also plays a role in livestock agriculture. By understanding the genetic makeup of animals, farmers can tailor their diets to optimize growth, health, and the nutritional quality of animal products, such as meat and dairy.

- Optimizing Soil and Crop Interactions: Understanding the genetic factors that influence a crop's ability to absorb nutrients from the soil is crucial for sustainable agriculture. Bioinformatics helps in studying these interactions and optimizing nutrient uptake for crop growth.

- Resilience to Environmental Stress: Nutritional genomics can help in developing crop varieties that are resilient to environmental stress, such as drought or nutrient-poor soil. By understanding the genetic basis of stress responses, crops can be engineered to thrive under challenging conditions.

- Personalized Nutrition: In the context of agriculture, personalized nutrition refers to tailoring crop choices and farming practices based on the nutritional needs of specific regions or populations. Nutritional genomics can help identify which crops are best suited for a particular area, taking into account genetic factors.

🞧 Metagenomics

Metagenomics is a powerful field within bioinformatics that has significant implications for agriculture. It involves the study of genetic material collected directly from environmental samples, such as soil, water, or plant tissues. In the context of agriculture, metagenomics has several applications:

- Soil Health and Microbiome Analysis: Metagenomics is used to analyze the soil microbiome, which includes bacteria, fungi, and other microorganisms. Understanding the diversity and functional potential of these microorganisms is crucial for assessing soil health. Healthy soils are essential for crop growth and productivity. Metagenomics helps in identifying beneficial microbes, understanding their roles in nutrient cycling and disease suppression, and designing strategies for sustainable agriculture.

- Plant-Microbe Interactions: Metagenomics enables the study of interactions between plants and the microorganisms in the rhizosphere (the soil zone around plant roots).

These interactions play a vital role in nutrient uptake, disease resistance, and overall plant health. By analyzing the metagenome of the rhizosphere, researchers can gain insights into the beneficial or pathogenic microorganisms present and their impact on crop growth.

- Crop Pathogen Detection: Metagenomics can be used to identify and characterize pathogens in agricultural environments. By analyzing metagenomic data from infected plant samples, researchers can detect the presence of harmful pathogens, such as viruses, bacteria, or fungi. This information is valuable for disease management and quarantine measures.

- Biological Control: Metagenomics can assist in identifying natural enemies of agricultural pests. Beneficial microorganisms or nematodes can be detected and used for biological pest control strategies, reducing the reliance on chemical pesticides.

- Microbial-Based Crop Enhancements: Metagenomics helps in the discovery and development of microbial-based products that can enhance crop growth, nutrient uptake, and stress resistance. These products, such as biofertilizers or biostimulants, are environmentally friendly alternatives to traditional agricultural inputs.

- Monitoring Ecosystem Changes: Metagenomics can be used to monitor changes in agricultural ecosystems over time. This includes tracking shifts in microbial populations due to changes in land use, cropping systems, or climate conditions. Understanding these changes can guide more sustainable agricultural practices.

- Resilience to Climate Change: As climate change impacts agriculture, metagenomics can provide insights into how plant-microbe interactions may be affected. This information is essential for developing crop varieties and management strategies that can adapt to changing environmental conditions.

- Waste Management: In livestock farming, metagenomics can be used to manage waste, such as manure. By understanding the microbial communities in waste, strategies for reducing environmental contamination and converting waste into bioenergy or other valuable products can be developed.

Cheminformatics

Cheminformatics is a specialized field within bioinformatics that deals with the storage, retrieval, and analysis of chemical information and data, particularly in the context of biological and agricultural applications. In the context of agriculture, cheminformatics plays a crucial role in various aspects of crop management, agricultural research, and biotechnology. Here's how cheminformatics is applied in bioinformatics to benefit agriculture:

- Pesticide and Fertilizer Development: Cheminformatics is used to design and develop new pesticides and fertilizers. Researchers can use databases of chemical structures and properties to predict the effectiveness and safety of these agrochemicals. This helps in reducing the environmental impact and improving crop yields.

- Chemical Safety: Cheminformatics tools are used to assess the safety of chemicals used in agriculture. This includes predicting the toxicity of pesticides and assessing their impact on non-target organisms, such as beneficial insects and pollinators.

- Drug Discovery for Plant Health: Bioinformatics and cheminformatics can be used to discover compounds that protect plants from diseases. This is essential in reducing the need for chemical pesticides. Identifying compounds that enhance plant immunity or inhibit pathogens is a common application.

- Plant Breeding: In modern agriculture, cheminformatics plays a role in crop improvement. For instance, researchers can use chemical profiling to identify compounds responsible for desirable traits in crops, such as nutritional content or disease resistance. This information can guide traditional breeding programs or genetic engineering efforts.

- Metabolomics: Cheminformatics tools are crucial in metabolomics, which involves studying the chemical processes occurring within organisms, including plants. Metabolomics data can be used to understand how plants respond to environmental changes, stress, and disease, helping in crop management and breeding.

- Herbicide Design: Cheminformatics assists in designing herbicides that selectively target weeds while sparing crop plants. Understanding the chemical properties and interactions of herbicides with plant biology is key to developing effective and environmentally friendly weed control solutions.

- Molecular Docking: Cheminformatics and molecular docking techniques are used to study how chemicals interact with biological molecules like plant proteins and enzymes. This information is valuable in understanding how chemicals can influence plant processes and can be used in the development of targeted agrochemicals.

- Environmental Impact Assessment: Cheminformatics can be used to assess the environmental impact of agricultural chemicals. This includes predicting their persistence in soil and water, their potential to leach into groundwater, and their impact on non-target organisms.

Computational resources

Databases and algorithms are essential components of bioinformatics, a multidisciplinary field that combines biology, computer science, and data analysis. They play a crucial role in managing, analyzing, and interpreting biological data, making it easier for researchers to extract meaningful information from large datasets. An overview of databases and algorithms commonly used in bioinformatics are as follows:

Databases in Bioinformatics:

Genomic Databases: These contain DNA and RNA sequences from various species. Examples include GenBank, Ensembl, and RefSeq. Genomic databases provide a wealth of genetic information used in sequence analysis, gene annotation, and comparative genomics.

Protein Databases: These store information about proteins, including sequences, structures, and functional annotations. Popular protein databases include UniProt, Protein Data Bank

(PDB), and Pfam. Researchers use these databases to study protein structure, function, and evolution.

Gene Expression Databases: These house data related to gene expression levels in different tissues, conditions, or experimental settings. The Gene Expression Omnibus (GEO) and ArrayExpress are examples of repositories for gene expression data.

Metabolic Pathway Databases: These contain information about biochemical pathways and the interactions between molecules in metabolic processes. KEGG and Reactome are widely used for pathway analysis.

✦ Algorithms

In bioinformatics, several key algorithms and methods are used for tasks related to sequence analysis and phylogenetics. These algorithms are fundamental for understanding the relationships between biological sequences, such as DNA, RNA, and proteins. Here, I'll provide an overview of pairwise and multiple sequence alignment, substitution matrices, and phylogenetic tree reconstruction algorithms:

1. Pairwise Sequence Alignment:
Pairwise sequence alignment is used to identify regions of similarity between two biological sequences. This can be helpful for comparing sequences for structural and functional analysis.
Needleman-Wunsch Algorithm: This algorithm performs global alignment, meaning it compares the entire sequences and finds the optimal alignment by maximizing a similarity score.
Smith-Waterman Algorithm: It's used for local sequence alignment, which finds the best-matching subsequence within the sequences.

2. Substitution Matrices:
Substitution matrices are used to score the substitution of one amino acid or nucleotide with another in sequence alignments. They provide a measure of evolutionary relatedness between sequences.
PAM (Point Accepted Mutation) Matrices: Developed by Margaret Dayhoff, PAM matrices describe the probability of specific amino acid substitutions over a fixed evolutionary distance.
BLOSUM (Blocks Substitution Matrix) Matrices: These matrices are used in protein sequence alignment and are based on observed substitutions within closely related sequences.

3. Phylogenetic Tree Reconstruction:
Phylogenetic tree reconstruction is used to infer evolutionary relationships and construct a tree that represents the divergence of species or sequences over time.
Neighbor-Joining (NJ): A distance-based method that constructs a tree by iteratively joining the closest neighbors.
Maximum Parsimony: This method seeks the tree that requires the fewest evolutionary changes (mutations) to explain the observed sequence data.
Maximum Likelihood: A likelihood-based approach that estimates the probability of observing the given sequences under different tree topologies.
Bayesian Inference: Uses a Bayesian framework to estimate the posterior distribution of tree topologies and model parameters.

Challenges

- Traditional bioinformatics methods heavily rely on reference databases, limiting analysis to known sequences and structures.
- These methods struggle to predict novel patterns, making them less effective in underexplored biological areas.
- Rapid improvements in high throughput sequencing technologies have given rise to heterogeneous and enormous amounts of omics data making it a big data problem.
- Moreover, there has been a shift in data types, transitioning from conventional structured data to a more diverse range of architectures, including unstructured, semi-structured, and heterogeneous formats, each with distinct characteristics.
- There is a need for advanced computational techniques such as Artificial Intelligence (AI) to leverage various data types, including sequences, images, and unstructured text, facilitating the integration of diverse biological information.

Big Data

In bioinformatics, as in other fields, the concept of "Big Data" is characterized by the "5 Vs," which describe key aspects of the data challenges faced. These Vs are Volume, Velocity, Variety, Veracity, and Value.

- Volume: Big data in bioinformatics originates from various sources, including genomics (DNA sequencing), transcriptomics (RNA sequencing), proteomics (protein data), metabolomics (small molecule data), structural biology (protein structures), and more. Additionally, data sources include literature, and data from high-throughput experiments. Genomic data, in particular, has seen a dramatic increase in the form of DNA and RNA sequences, with millions of sequences available in public databases. This volume continues to expand rapidly.

- Variety: Biological data comes in diverse formats, such as sequences, alignments, 3D structures, images, clinical records, and omics data. Integrating and analyzing these various data types poses challenges.

- Velocity: The speed at which new biological data is generated is incredibly high, especially with the advent of high-throughput sequencing technologies. Keeping up with the pace of data generation is a significant challenge for bioinformaticians.

- Veracity: Veracity relates to the accuracy, quality, and reliability of data. In bioinformatics, ensuring the veracity of data is crucial since errors or inaccuracies can lead to incorrect scientific conclusions.

- Value: The value of big data in bioinformatics is the benefit that can be derived from it. It involves extracting meaningful insights, making discoveries, and ultimately improving healthcare, agriculture, and various biological research fields.

Artificial Intelligence in Bioinformatics

Artificial intelligence (AI) was formally defined at the Dartmouth conference in 1956. It quickly entered a period of rapid development and innovation, becoming known as the "golden age" of AI. The field of AI encompasses a wide array of content, and one of its crucial branches is machine learning (ML). ML is a methodology for achieving AI and includes a range of mathematical tools and algorithms. Although ML initially achieved remarkable progress, it faced a significant setback in the 1960s due to theoretical limitations.

It wasn't until the introduction of the backpropagation algorithm in the 1980s that ML experienced a resurgence in activity and widespread application. Subsequently, deep learning (DL) emerged from artificial neural networks (ANN) within the realm of machine learning and has been a driving force behind the current era of deep learning since 2006.

Over the last decade, AI has found extensive use in omics studies, thanks to the accumulation of large-scale omics data and the growing need for big data analysis. Machine learning, as a subset of AI, focuses on acquiring insights and establishing patterns from data through computational models and algorithms. Its goal is to enhance system performance through computation and learning from experiences. Machine learning has diverse applications, spanning natural language processing, computer vision, data mining, and more. Various machine learning algorithms serve distinct purposes, including clustering, classification, regression, association rule mining, dimension reduction, and others. Based on the nature of the data and training strategies, machine learning is categorized into three primary types: supervised, unsupervised, and reinforcement learning.

Supervised learning deals primarily with regression and classification problems, while unsupervised learning focuses on clustering. Reinforcement learning, on the other hand, involves learning from new experiences through trial-and-error. The field boasts a variety of traditional machine learning algorithms such as generalized regression, decision trees, naive Bayes, support vector machines (SVM), K-means clustering, and more.

Deep learning, a critical branch of machine learning, originated from artificial neural networks and was formally introduced in 2006. It has since experienced rapid and substantial development. Deep learning encompasses a multidisciplinary approach, merging elements of statistics, optimization, algorithms, programming, distributed computing, and other fields. By constructing models with multiple hidden layers, deep learning allows for the discovery of intricate relationships within data, improving the accuracy of classification and prediction. This evolution has had a significant impact on various fields, making it a fundamental component of the broader AI landscape.

In the forthcoming sections, various applications of ML in different omics have been discussed.

🞧 Machine learning in genomics:

In the field of genomics and genome research, machine learning has become a crucial tool for various applications. These applications encompass diverse aspects of genomics, from predicting 3D genome structures to genome annotation, transcription regulation, effects of genetic variants, and even genome editing (Figure 2).

1. Reconstruction of 3D Genome Structure:
Understanding the spatial organization of the eukaryotic genome is essential for elucidating chromosomal activities within the cell. Experimental techniques, such as chromosome conformation capture (3C)-based technologies, provide insights into 3D genome organization but have limitations in resolution and cost. Therefore, machine learning methods have been developed to complement experimental studies. These methods are categorized based on their training data, including genomic sequences, 3C-based interactions, chromatin states derived from epigenetic modifications, or hybrid data. They aim to predict various aspects of 3D genome structure, including reconstruction, compartmentalization, topologically associating domains (TADs), and chromatin loops.

2. Computational Modeling of Epigenomic and Chromatin States:
Epigenomic modifications play a crucial role in genome regulation. Machine learning approaches have been employed to interpret and predict the effects of epigenetic modifications, DNA methylation, histone modifications, and chromatin states. These methods generate features from epigenetic data and leverage deep learning techniques to understand and predict epigenomic changes.

3. Genome Annotation and Transcription Regulation:
Machine learning is applied to the annotation of the genome, including the identification of protein-coding genes, non-coding RNAs, microRNAs, transcript splicing isoforms, regulatory elements, protein-binding sites, and cis-regulatory binding modules. It goes beyond simple identification to elucidate their functions and interactions. This is essential for understanding the roles of different genomic elements in gene regulation.

4. Identifying the Effects of Genetic Variants:
Genetic variants, especially those in non-coding regions, can significantly impact gene expression and phenotypes. Machine learning models have been developed to classify and predict the pathogenicity of genetic variants. These models help identify functional effects of non-coding variants and their contributions to diseases.

5. Machine Learning in Genome Editing:
The advent of genome editing technologies, such as CRISPR, has opened new possibilities in genome engineering. Machine learning is applied to design guide RNAs for CRISPR-based editing, predict cleavage tendencies, evaluate off-target effects, and identify optimal editing locations. These methods contribute to more precise and efficient genome editing.
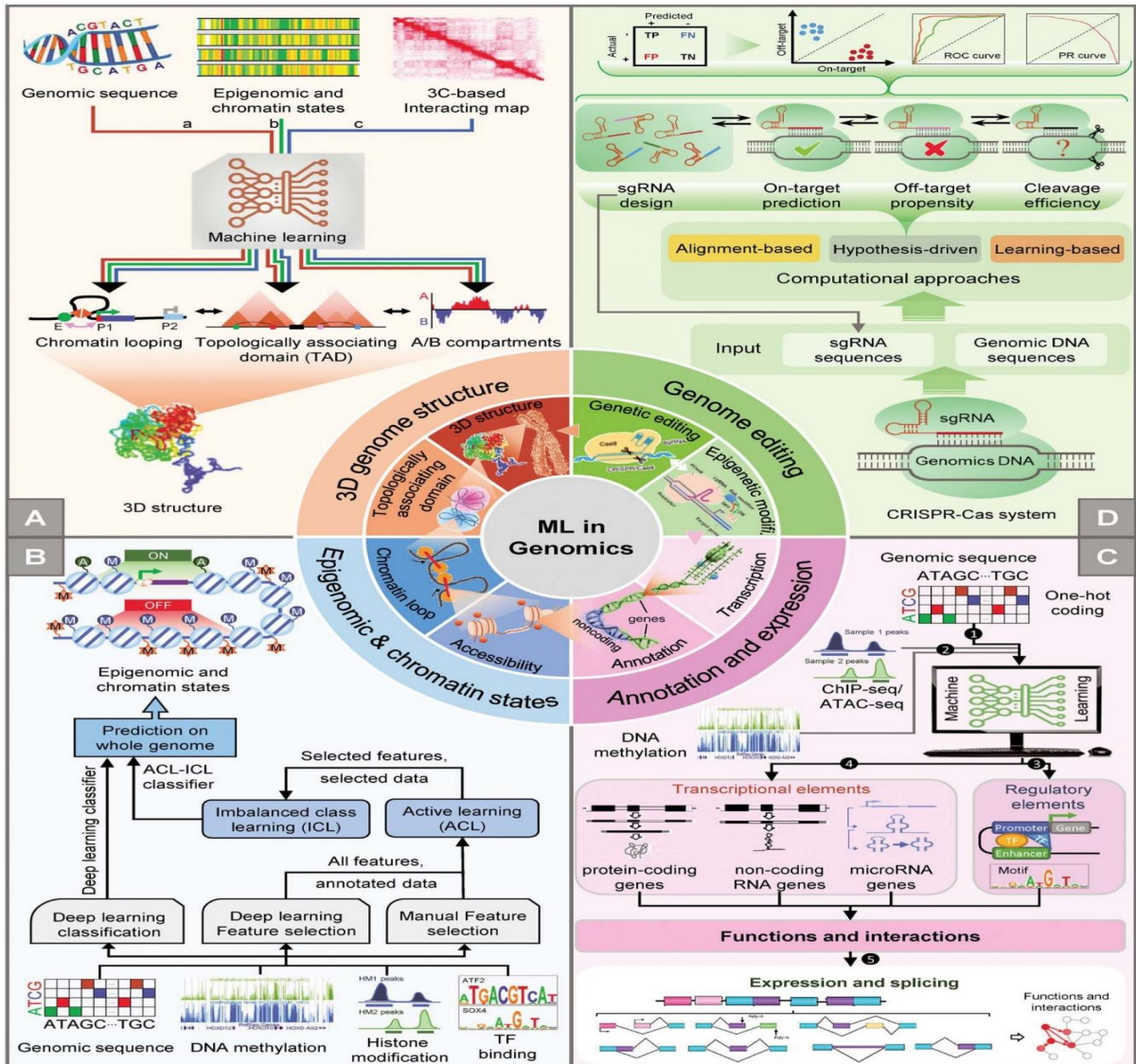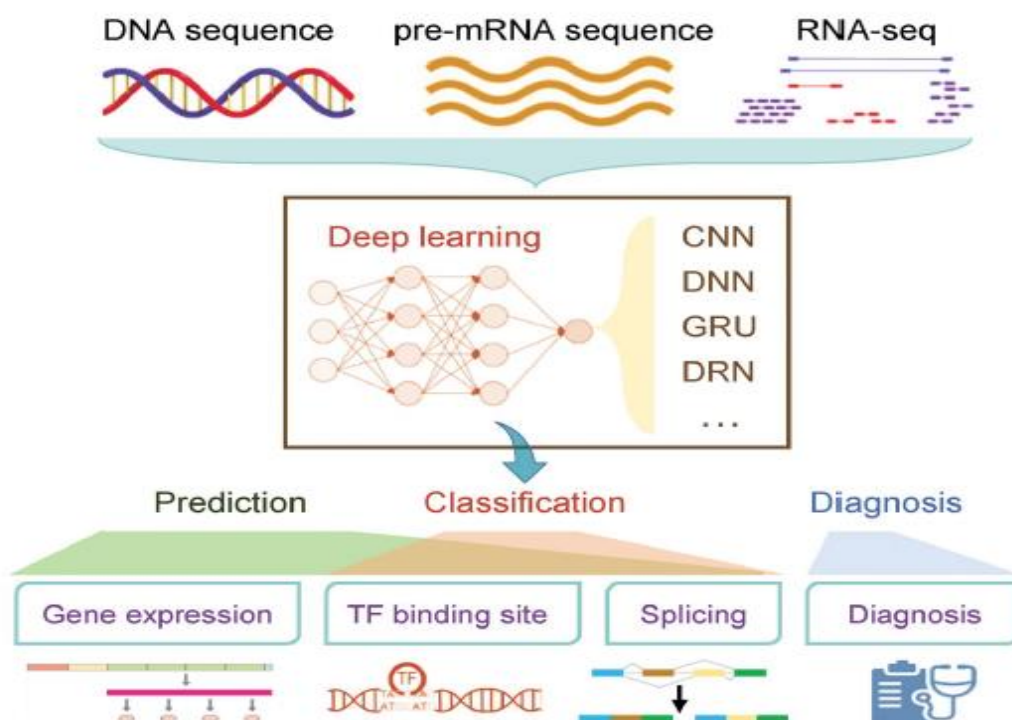
Figure 2: Schematics representation of the application of AI in genomics (Li et al., 2022)

➕ Machine learning in transcriptomics
Machine learning, particularly deep learning, has made significant contributions to various aspects of transcriptomics, including the prediction and understanding of gene expression, splicing patterns, and transcription factor binding sites (Figure 3).

1. Prediction of Gene Expression:
Machine learning, especially deep learning, has proven highly effective in predicting gene expression levels based on genetic and epigenetic information. For instance, deep neural networks (DNNs) have been used to build models like D-GEX, which can predict target gene expression based on landmark genes. Histone modifications, which play a vital role in gene regulation, have also been leveraged for gene expression prediction using models like DeepChrome, demonstrating the superior performance of deep learning compared to traditional machine learning methods.

2. Prediction and Classification of Splicing:
Splicing, which determines how the genome is transcribed, influences the diversity of transcriptomes and proteomes. Aberrant splicing can lead to diseases, making it a crucial

area of study. Deep learning methods, including deep neural networks, are employed to predict and classify splicing patterns based on RNA-seq data, genomic sequences, and epigenetic features. These models accurately predict splicing outcomes in different biological contexts and contribute to our understanding of splicing regulation.

3. Prediction of Transcription Factor Binding Sites:
Transcription factors (TFs) are central to gene regulation, and their binding sites on DNA are essential for controlling gene expression. Machine learning, particularly deep learning, has been applied to identify TF-binding sites more accurately and efficiently. Models like PIQ and DeepBind have demonstrated the ability of deep learning to predict TF-binding sites. These models improve the accuracy of prediction, especially in comparison to traditional methods based on position weight matrices (PWMs).

4. Auxiliary Diagnosis Using Transcriptomics:
Machine learning plays a significant role in aiding disease diagnosis, particularly in the medical field. Artificial neural networks (ANNs) can analyze gene expression data to enhance the accuracy and efficiency of disease classification and diagnosis. Machine learning models combined with gene expression data are used for various medical applications, including predicting myopathy subtypes, drug-induced liver injury, and diagnoses related to mental and neurological diseases. In the context of cancer, machine learning assists in cancer classification, predicting molecular subtypes, early diagnosis, prognosis, and recurrence prediction. The integration of multiple cohort datasets is a promising avenue for improving auxiliary diagnosis, although the challenge of limited data remains.

Figure 3: Schematics representation of the application of AI in transcriptomics (Li et al., 2022)

- Machine learning in proteomics

Machine learning is playing a pivotal role in the field of proteomics, where it aids in efficiently processing and analyzing vast amounts of proteomic data (Figure 4). Specifically, machine learning methods are significantly impacting proteomics in various areas, as outlined below.

1. Biomass Spectrometry

Mass spectrometry (MS) is an indispensable tool for studying protein structures and components. However, the processing of MS data has often lagged behind the development of MS instruments. Machine learning, particularly deep learning, is stepping in to address the challenges posed by high-dimensional and sparse proteomic data. Deep learning models are being harnessed for tasks like de novo sequencing, peptide property prediction, and mass spectrometry imaging analysis. For instance, DeepNovo, a deep learning-based model, is enhancing the accuracy of de novo peptide sequencing. Moreover, DeepRT employs deep learning to predict peptide retention times, a critical factor in liquid chromatography-mass spectrometry tandem analysis. Machine learning has the potential to significantly enhance the retrieval and analysis of peptide data, thereby advancing our understanding of proteome characterization.

2. Screening of Protein Biomarkers

Biomarkers are vital for disease screening, diagnosis, and therapy guidance. Traditional statistical methods often face limitations in biomarker discovery due to classification boundaries and variable correlations. Machine learning methods, both supervised and unsupervised, offer more flexibility in this context. Researchers have been combining machine learning with proteomic techniques, such as mass spectrometry, to identify disease-specific protein markers. For example, a study utilized a deep belief network (DBN) to screen for protein diagnostic markers in Alzheimer's disease, yielding a marker group with high diagnostic accuracy. While machine learning holds immense promise in biomarker discovery, challenges like overfitting and model interpretability need to be addressed.

3. Nucleic Acid–Binding Protein Prediction

Identifying proteins that bind to nucleic acids is essential for understanding various biological processes. Traditionally, this identification has been hampered by accuracy and scalability issues. However, with the availability of high-throughput measurements, such as protein binding microarrays and SELEX, machine learning has emerged as a highly accurate predictor of nucleic acid–binding properties in proteins. Tasks include DNA-binding domain recognition and predicting protein-DNA/RNA docking interactions. Despite the success, challenges remain, particularly in reducing cross-prediction between DNA and RNA-binding residues.

4. Predicting Protein–Protein Interactions (PPIs)

PPIs are a critical domain where machine learning is revolutionizing our understanding of protein functions. While public databases offer some PPI data, they often lack specificity and comprehensiveness. Combining experimental methods with machine learning is proving effective in predicting PPIs accurately. Various machine learning algorithms, including random forests, support vector machines, and Bayesian probabilistic inference, are being used for PPI prediction. Deep learning has also found application in predicting PPIs through methods like domain-based ensemble models. Accurate identification of PPIs is instrumental in comprehending a wide range of physiological activities.

5. Protein Post-Translational Modification (PTM)

PTM prediction is yet another area significantly benefiting from machine learning methods. PTMs, such as phosphorylation and glycosylation, play vital roles in regulating protein function. Machine learning models have been developed to predict PTM sites with high accuracy. For example, Musite predicts phosphorylation sites, while GlycoEP identifies N-, O-, and C-linked glycosylation sites. Additionally, web servers like MusiteDeep employ convolutional neural networks (CNNs) for predicting multiple PTM sites simultaneously, offering advantages in accuracy and speed. Furthermore, tools like SAPH-ire TFx assist in the identification of functional PTM sites from large-scale datasets.
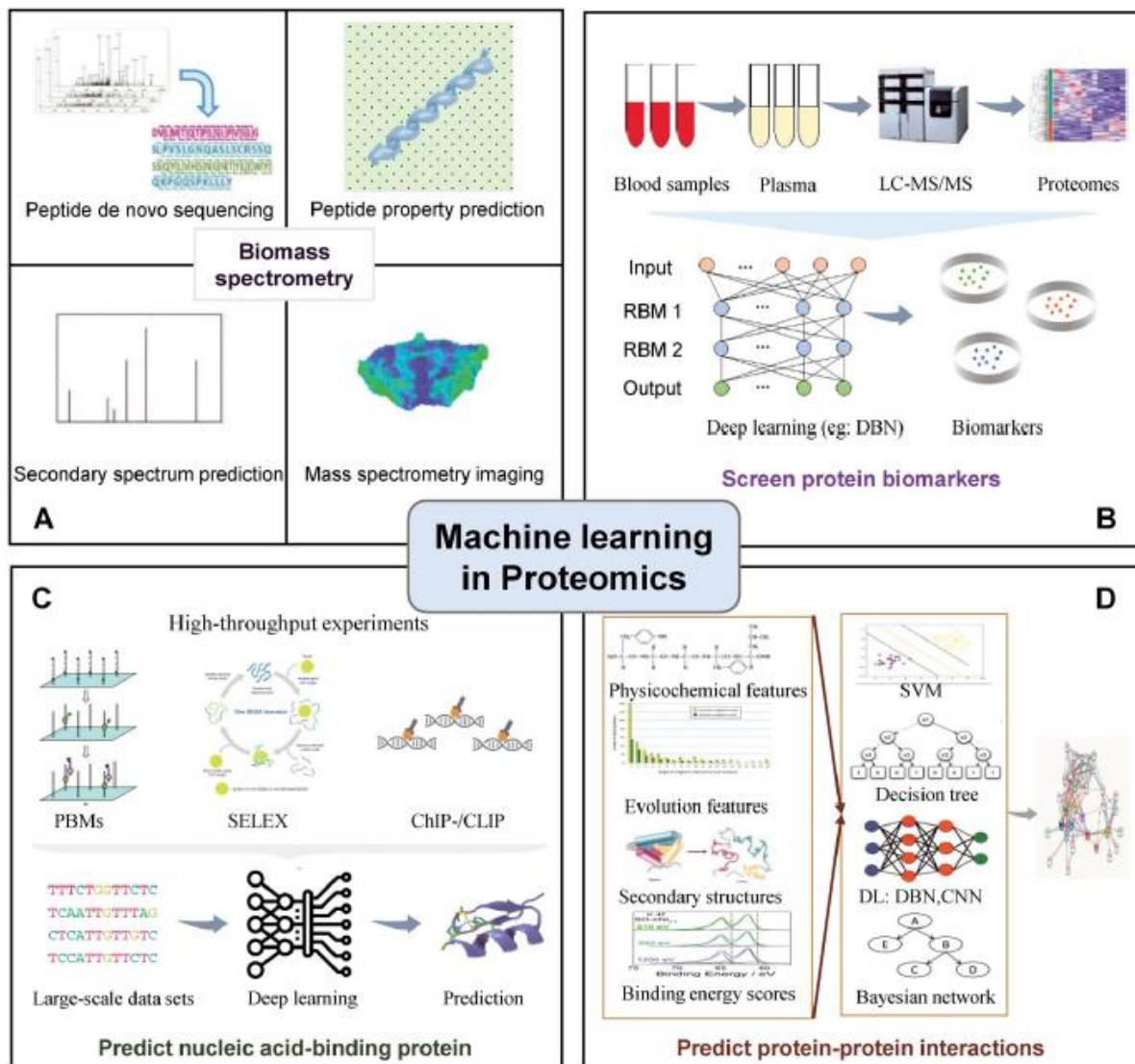
Figure 4: Schematics representation of the application of AI in proteomics (Li et al., 2022)

➕ Machine learning in metabolomics

Metabolomics, akin to genomics and proteomics, focuses on quantitatively analyzing all metabolites in organisms to uncover their relationships with physiological and pathological changes. It's a valuable technology for diagnosing diverse diseases characterized by metabolic variations. Traditional methods often struggle with the

sparsity of large-scale metabolomic data obtained through mass spectrometry, chromatography, and nuclear magnetic resonance. This challenge has led to an increased interest in machine learning algorithms. In the field of metabolomics, various machine learning techniques are being employed, contributing to advancements in data processing, metabolic phenotype stratification, and metabolic modeling (Figure 5).

1. Data Processing and Analysis:
Machine learning has significantly enhanced the processing and analysis of metabolomic data. These algorithms excel in pattern recognition and multivariate classification, assisting in classifying data based on complex patterns. Traditional methods like partial least squares discriminant analysis (PLS-DA), as well as support vector machines (SVM), have been employed for this purpose. SVM has gained prominence in metabolomics due to its high prediction and classification accuracy. Deep learning, a subset of machine learning, has also been applied in metabolomics for processes like estimating the detection probability of specific peaks. Deep learning, through methods like deep neural networks (DNNs), aids in eliminating false-positive peaks, enhancing the quality of metabolomic data. Tandem mass spectrometry (MS/MS) is used to identify unknown metabolites. The application of deep learning, such as the DeepMASS framework, helps effectively identify these unknown metabolites. Additionally, machine learning methods are being used to automate quality control and quality assurance processes in data processing.

2. Stratification of Metabolic Phenotypes:
Machine learning, particularly deep learning, is revolutionizing the stratification of metabolic phenotypes. This approach characterizes the metabolic profiles and processes of individuals based on the presence, content, and ratios of specific metabolites. Deep learning techniques have demonstrated success in capturing the intricate metabolic characteristics present in the data. For example, deep neural networks combined with t-distribution random neighborhood embedding have revealed the metabolic heterogeneity in human colorectal cancer. Deep learning frameworks are also employed in classifying the estrogen receptor status of breast cancer, surpassing other machine learning methods in prediction accuracy. Novel methods combining deep neural networks enhance metabolic phenotype stratification and metabolite selection, offering high classification accuracy.

3. Genome-Scale Construction of Metabolic Models:
Machine learning is also playing a vital role in constructing genome-scale metabolic models (GEMs). GEMs encompass the metabolic reactions of a specific organism's genome and serve as a platform for metabolic flux modeling. The modeling process involves constraint-based quantitative modeling, integrating biochemical and genetic information. Machine learning optimizes model parameters, tests various input conditions, and enhances biomarker recognition, quantifying metabolite flux, and predicting metabolic genes. Applications extend to determining predictors of metabolic-related drug side effects, generating collision cross-section values of small molecules, and identifying early metabolic disease markers. Despite these advancements, challenges persist, including experimental limitations, small sample sizes, interpretability issues, and a lack of comprehensive reference data.
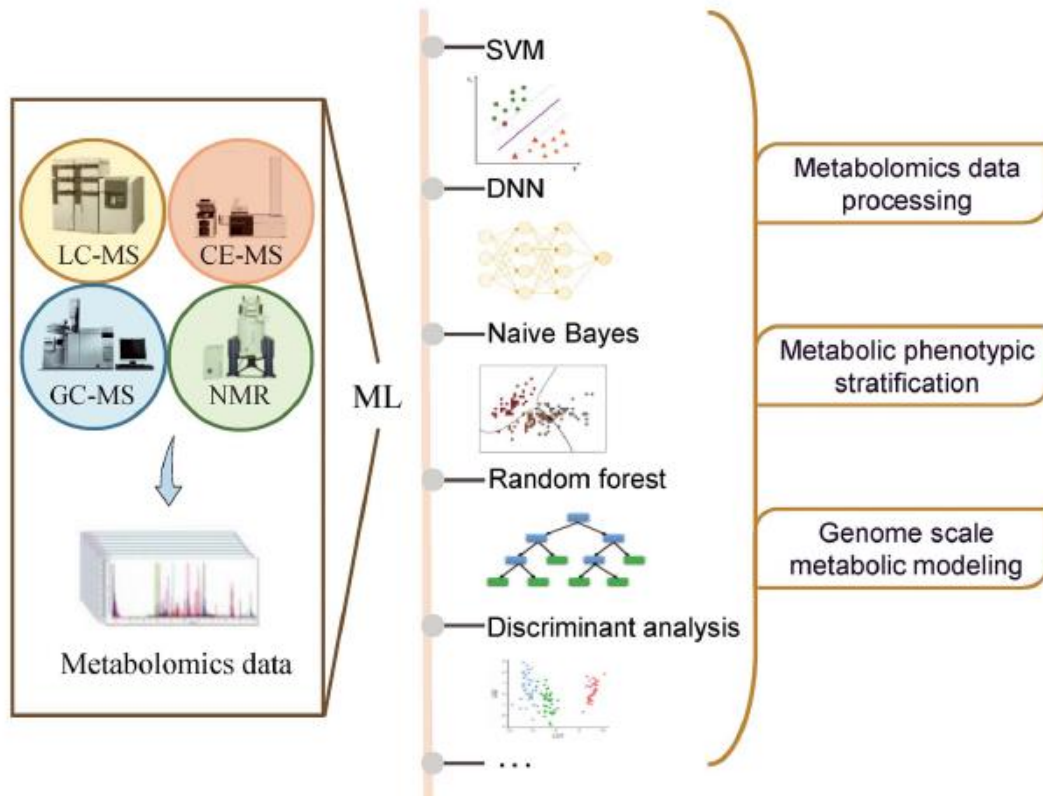
Figure 5: Schematics representation of the application of AI in metabolomics (Li et al., 2022)

References

Gauthier, J., Vincent, A. T., Charette, S. J., & Derome, N. (2019). A brief history of bioinformatics. Briefings in bioinformatics, 20(6), 1981-1996.

Jawdat, D. (2006, April). The era of bioinformatics. In 2006 2nd International Conference on Information & Communication Technologies (Vol. 1, pp. 1860-1865). IEEE.

Li, R., Li, L., Xu, Y., & Yang, J. (2022). Machine learning meets omics: applications and perspectives. Briefings in Bioinformatics, 23(1), bbab460.

Mochida, K., & Shinozaki, K. (2011). Advances in omics and bioinformatics tools for systems analyses of plant functions. Plant and Cell Physiology, 52(12), 2017-2038.

ASHOKA: Functioning and Activities

K.K. Chaturvedi, U.B. Angadi and Jai Bhagwan

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

First HPC systems were vector-based systems (e.g. Cray) named 'supercomputers' because they were an order of magnitude more powerful than commercial systems. The 'supercomputer', a large systems are just scaled up versions of smaller systems. High performance computing can mean high flop count per processor and totalled over many processors working on the same or related problems. This can have faster turnaround time, more powerful system, scheduled to first available system(s) and using multiple systems simultaneously. The HPC is any computational technique that solves a large problem faster than possible using single, commodity systems, Custom-designed, high-performance processors, Parallel computing, Distributed computing and Grid computing.

Parallel computing is a single system with many processors working on the common task. The Distributed computing is configured as many systems loosely coupled by a scheduler to work on related problems and Grid Computing is defined as many systems tightly coupled by software and networks to work together on single problems or on related problems.

Parallel computer is a computer that contains multiple processors where each processor works on its section of the problem and allowed to exchange information with other processors.

Two big advantages of parallel computers are performance and memory. Parallel computers enable us to solve problems that benefit from or require, fast solution, require large amounts of memory and both.

As per the Moore's Law 'predicts' that single processor performance doubles every 18 months, eventually physical limits on manufacturing technology will be reached as in figure 1.
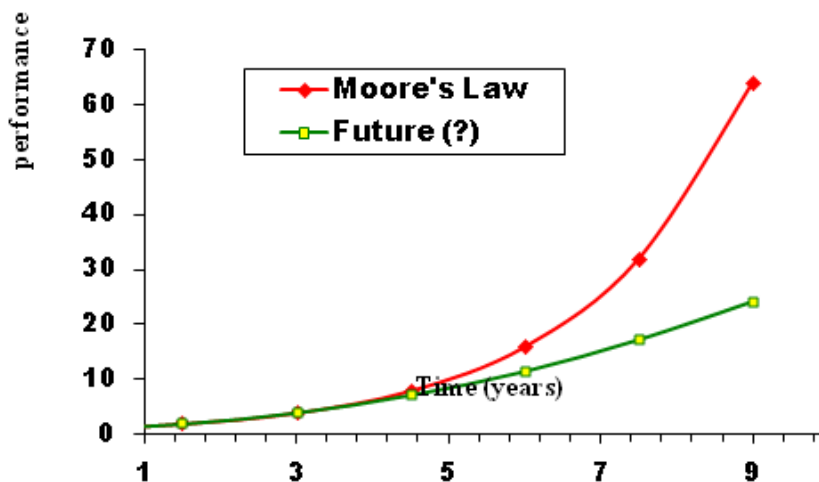
Fig. 1: Moore's Law towards performance of the system

There are two types of parallel computers by their memory model namely shared memory and distributed memory. All processors have access to a pool of shared memory (Figure 2-A) while each processor has its own local memory in distributed memory (Figure 2-B).
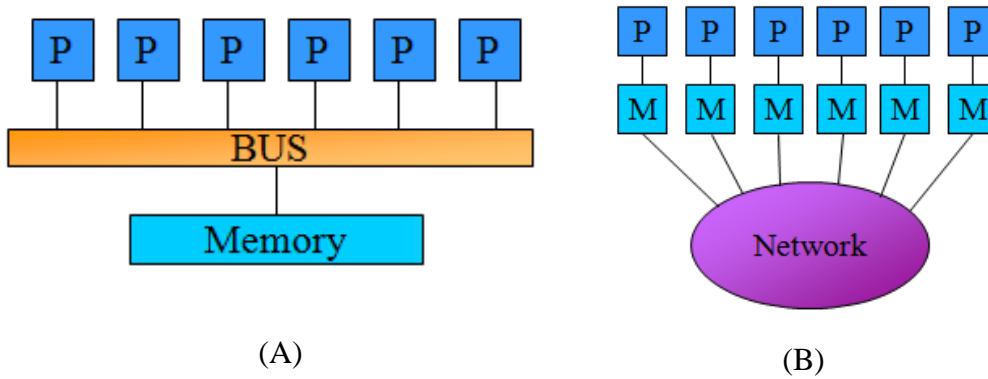
Fig. 2: Shared Memory and distributed memory system

Shared memory have two types of architecture i.e., Uniform memory access (UMA) and Non-uniform memory access (NUMA). Each processor has uniform access to memory in UMA and also called as symmetric multiprocessors, or SMPs (Figure 3-A). Time for memory access depends on location of data in NUMA as local access is faster than non-local access but it is easy to scale up than SMPs (Figure 3-B).
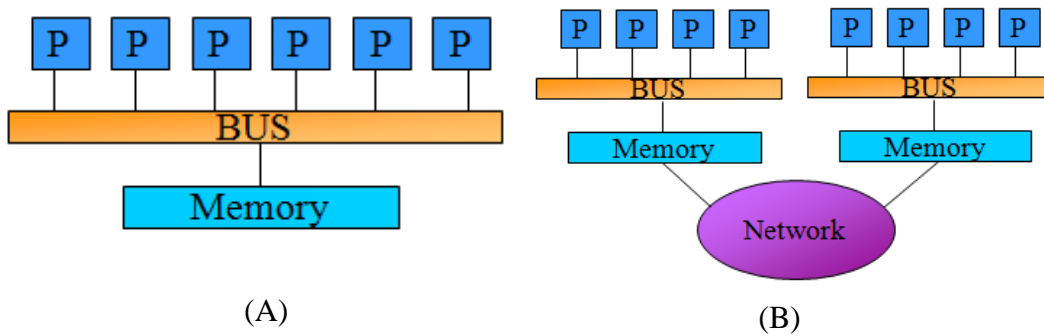

Fig. 3: Shared Memory with UMA and NUMA

The distributed memory is two types namely Massively Parallel Processor (MPP) and cluster. MPP is tightly integrated, single system image and cluster is an individual computers connected by specialized software and connected using interconnect network. Distributed memory is shown in figure 4.
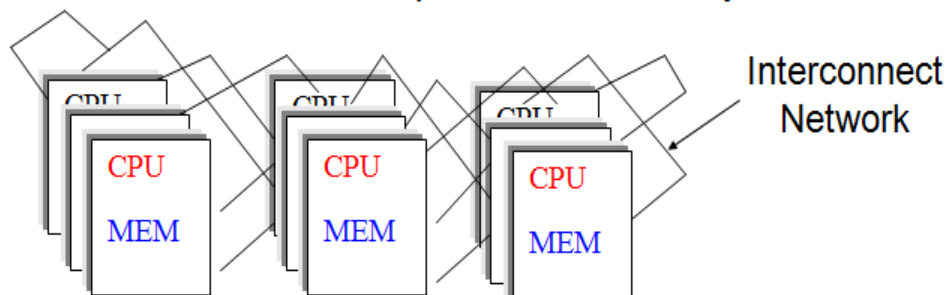
Fig. 4: Distributed Memory

Both types of memory systems have processors, memory and network/interconnect.

Terminology

Clock period (cp): The minimum time interval between successive actions in the processor. It is measured in nanoseconds (~1-5 for fastest processors) which is inverse of frequency (MHz).

Instruction: An action executed by a processor, such as a mathematical operation or a memory operation.

Register: A small and extremely fast location for storing data or instructions in the processor.

Functional Unit (FU): A hardware element that performs an operation on an operand or pair of operations. Common FUs are ADD, MULT, INV, SQRT, etc.

Pipeline: A Technique enables multiple instructions to be overlapped during execution.

Superscalar: Multiple instructions are possible per clock period.

Flops: Floating point operations per second.

Cache: A Fast memory in the processor which keep instructions and data close to functional units so processor can execute more instructions more rapidly.

SRAM: Static Random Access Memory (RAM). Very fast (~10 nanoseconds), made using the same kind of circuitry as the processors, so speed is comparable.

DRAM: Dynamic RAM. Longer access times (~100 nanoseconds), but hold more bits and are much less expensive (10x cheaper).

Memory hierarchy: The hierarchy of memory in a parallel system, from registers to cache to local memory to remote memory.

Networks Latency: How long does it take to start sending a "message"? Measured in microseconds.

Networks Processors: How long does it take to output results of some operations, such as floating point add, divide etc., which are pipelined?)

Networks Bandwidth: What data rate can be sustained once the message is started? Measured in Mbytes/sec or Gbytes/sec

Types of Clusters/Processors

Symmetric Multiprocessors (SMPs) connect processors to global shared memory using either bus or crossbar. It provides simple programming model, but has problems with buses can become saturated and crossbar size must increase with number of processors. Problem grows with number of processors, limiting maximum size of SMPs. Programming models are easier since message passing is not necessary. The techniques are auto-parallelization via compiler options, loop-level parallelism via compiler directives, OpenMP, and pthreads.

In MPP, each processor has its own memory and is not shared globally but the processors adds another layer to memory hierarchy (remote memory). The processor/memory nodes are connected by interconnect network using many possible topologies. The processors must pass data via messages so the communication overhead can be minimized. Many vendors have custom interconnects that provide high performance for their MPP system such as Gigabit Ethernet, Fast Ethernet, etc.

Clusters are similar to MPPs with processors and memory. The processor performance must be maximized and memory hierarchy needs remote memory as no shared memory for message passing to avoid the communication overhead.

Clusters are different from MPPs as commodity processors including interconnect and OS with multiple independent systems and separate I/O systems. The advantages of clusters are inexpensive, fastest processors first, potential for true parallel I/O and high availability while the disadvantages are less mature software (programming and system), more difficult to manage (changing slowly), lower performance interconnects (not as scalable to large number).

Distributed Memory Programming provides message passing using MPI, MPI-2 and active/one-sided messages.

There are two types of parallelism i.e., data and task. Each processor performs the same task on different sets or sub-regions of data in data parallelism. Each processor performs a different task in task parallelism. Most parallel applications fall somewhere on the continuum between these two extremes.

Example of data parallelism in a bottling plant, there are several 'processors', or bottle cappers, applying bottle caps concurrently on rows of bottles.

Example of task parallelism in a restaurant kitchen, there are several chefs, or 'processors', working simultaneously on different parts of different meals. A good restaurant kitchen also demonstrates load balancing and synchronization--more on those topics later.

A common form of parallelism used in developing applications was Master-Worker parallelism where a single processor is responsible for distributing data and collecting results (task parallelism) and all other processors perform same task on their portion of data (data parallelism).

According to Flynn's Taxonomy, the computing systems are classified into the following broad categories:

- SISD: Single Instruction and Single Data
- SIMD: Single Instruction and Multiple Data
- MISD: Multiple Instruction and Single Data
- MIMD: Multiple Instruction and Multiple Data

The purpose of High-performance computing (HPC) platform is to provide the access to the compute resources remotely. The user can login remotely and submit compute their jobs either from the command line or through the GUI based interface provided to them. The computing systems are connected together through a high bandwidth data transfer and made available to the users in a queue-based job submission system. There are many open-source and commercial software packages installed.

At IASRI, New Delhi

The National Agricultural Bioinformatics Grid in ICAR consists of an advanced HPC infrastructure at IASRI, New Delhi and moderate HPC facilities at the domain centres for undertaking research in the field of agricultural bioinformatics. Clusters are collections of computers that are connected together. The special sets of software are used to configure HPC environment. This set up has been named as Advanced Supercomputing Hub for Omics Knowledge in Agriculture (ASHOKA). The importance of HPC is rapidly growing because more and more scientific and technical problems are being studied on the huge data sets which require very high computational power as well. HPC offers environment for biologists,

scientists, analysts, engineers and students to utilize the computing resources in making vital decisions, to speed up research and development, by reducing the execution time.

The following HPC infrastructure are set up under NAIP project NABG which are as follows in the form of clusters, network and storage.

Types of Clusters

    a. 256 Nodes Linux Based Cluster with two masters
    b. 16 Nodes Windows Based Cluster with one master
    c. 16 Nodes GPGPU Based Linux Cluster with one master
    d. 16 Nodes Linux based SMP system
    e. 16 Nodes Linux Based Cluster at each of the five domains with one master

Types of Networks

    a. High bandwidth network with low latency (Q-logic QDR InfiniBand switch)
    b. Gigabit network for cluster administration and management
    c. ILO3 Management Network

Types of Storage

    a. Parallel File System (PFS) for computational purpose
    b. Network Attached Storage (NAS) for user Home Directory
    c. Archival Storage for back up.

The hardware configuration of the Head/Master node is as follows

Server Name	:	HP ProLiant DL380-G7 Server
Type of Processor	:	Intel Xeon X5675 3.07Ghz
Number of Processors	:	2
Core per Processor	:	6
Total memory (RAM)	:	96GB
Memory per Core	:	8GB
Hard Disk	:	6*600GB SAS
OS	:	RHEL 6.2 (Linux)

The hardware configuration of each compute node is as follows

Server Name	:	HP ProLiant SL390-G7 Server
Type of Processor	:	Intel Xeon X5675 3.07 Ghz
Number of Processors	:	2
Core per Processor	:	6
Total memory (RAM)	:	96G
Memory per Core	:	8GB
Hard Disk	:	300GB SAS
OS	:	RHEL 6.2 (Linux)

Measuring Performance

The memory is measured in terms of bytes i.e., Kilo (2^{10} or 10^3), Mega (2^{20} or 10^6) , Giga (2^{30} or 10^9) – Tera (2^{40} or 10^{12}), Peta (2^{50} or 10^{15}) , Exa (2^{60} or 10^{18})

The computational performance is measured in Flop/s (Flop/s = floating point operations per second) i.e., Mega Flops, Tera Flops, Peta Flops etc.

One can calculate peak performance of the cluster using standard formula i.e. Cluster Performance = (Number of nodes) * (number of CPUs per node) * (number of cores per CPU) * (CPU speed in GHz) * (CPU instruction per cycle)

The grid has been established using the following network diagram as in figure 5.
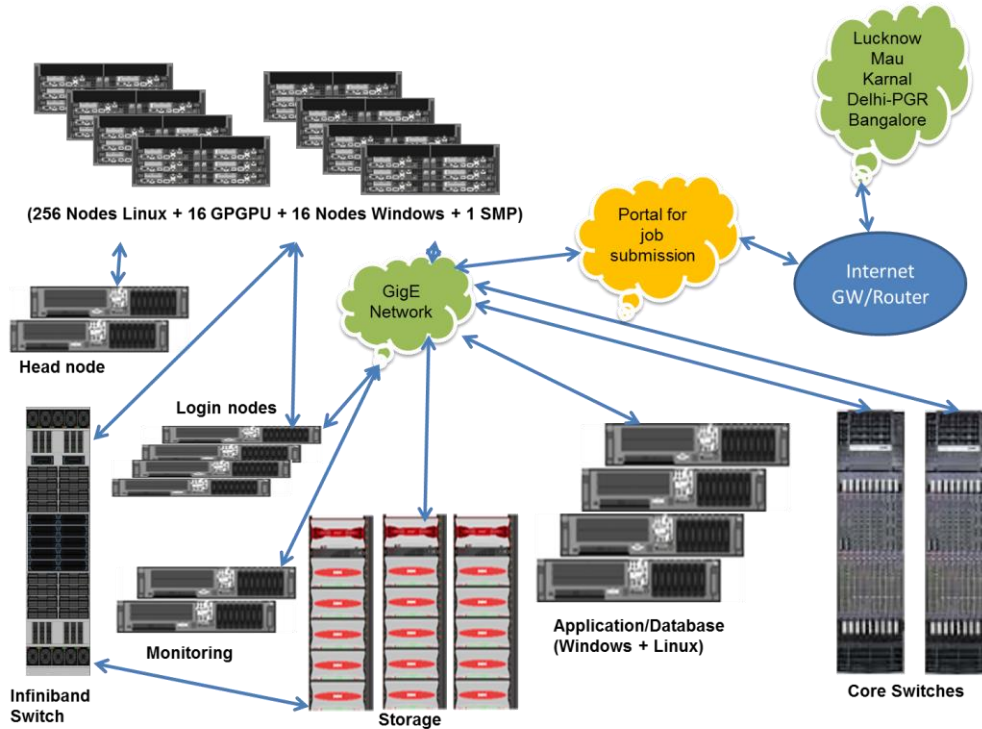
Fig. 5: Network diagram of NABG Grid

The hardware and software specifications of the SMP is as follows

Server Name	:	HP ProLiant DL 980 G7
Type of Processor	:	Intel Xeon E7- 2830 2.13GHz
Number of Processors	:	8
Core per Processor	:	8
Total memory (RAM)	:	1.5 TB
Hard Disk	:	396 GB
OS	:	RHEL 6.2

A switched fabric computer network communications link, is being used in HPC and enterprise data centre with InfiniBand interconnect switch. The InfiniBand architecture specification defines a connection between processor nodes and high performance I/O nodes such as storage devices as in figure 6.
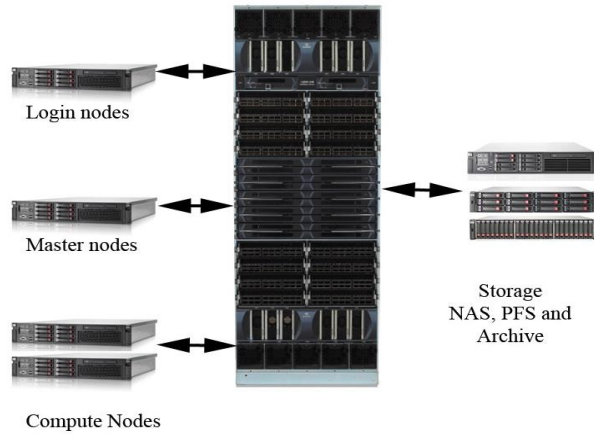
Fig. 6: InfiniBand interconnect switch

Main purpose of Ethernet network in the cluster is to provide services like cluster management, cluster monitoring, compute node deployment and many other things in figure 7.
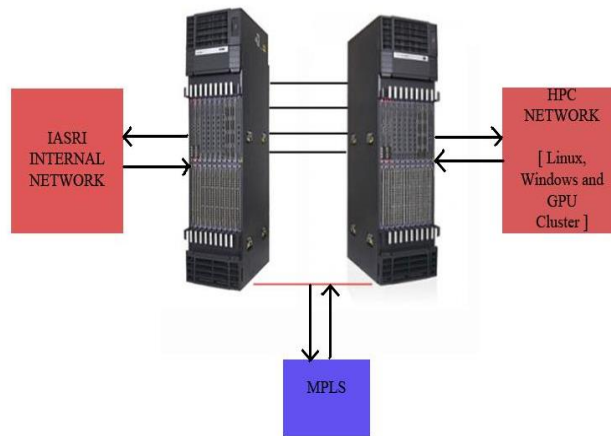
Fig. 7: InfiniBand interconnect switch

Different types of file system are configured for storing user's data, running parallel jobs and archiving the important data. There are three types of storage (i) Network Attached Storage (NAS), (ii) Parallel File System (PFS) and (iii) Archival Storage.

The following challenges in bioinformatics are exists which essentially require the grid based architecture.

- Protein folding & structure prediction
- Homology search
- Multiple alignment
- Genomic sequence analysis
- Gene finding
- Gene expression data analysis
- Drug discovery
- Phylogenetic inference
- Computational genomics, proteomics
- Computational evolutionary biology

Introduction to Linux Basics

S. B. Lal

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

The Linux operating system is basically a variant of the UNIX operating system, and Linux has probably all that UNIX offers and more. It is a multi-user, multitasking, network operating system which also has a user friendly Graphical User Interface (GUI).

Every desktop computer uses an operating system. The most popular operating systems are Windows, Mac OS, UNIX, Linux.

What is an Operating System?

An operating system is the first piece of software that the computer executes when a system is turned on. The operating system loads itself into memory and begins managing the resources available in the computer. It provides those resources to other applications that the user wants to run. Typical services that an operating system provides include:

A task scheduler - The task scheduler is able to allocate the execution of the CPU to a number of different tasks. Some of those tasks are the different applications that the user is running, and some of them are operating system tasks.

A memory manager - The memory manager controls the system's RAM and normally creates a larger virtual memory space using a file on the hard disk.

A disk manager - The disk manager creates and maintains the directories and files on the disk. When a file is needed, the disk manager makes it available from the disk.

A network manager - The network manager controls all data moving between the computer and the network.

Other I/O services manager - The OS manages the keyboard, mouse, video display, printers, etc.

Security manager - The OS maintains the security of the information in the computer's files and controls who can access the computer.

An operating system normally also provides the default user interface for the system. The standard "look" of Windows 98 includes the Start button, the task bar, etc. The Mac OS provides a completely different look and feel for Macintosh computers.

To understand why Linux has become so popular, it is helpful to know a little bit about its history.

Background on Linux

Linux, a UNIX-like operating system, is based on Minix and has been invented by Linus Benedict Torvalds in 1991. The following is an excerpt of a newsgroup, called "comp.os.minix" where Linus posted this text on 08/01/91: "...As I mentioned a month ago, I'm working on a free version of a Minix-look-alike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be, depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02... but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it."

Linux is a free version of UNIX that continues to be developed by the cooperative efforts of volunteer groups of programmers, primarily on the Internet, who exchange code, report bug, and fix problems in an open-ended environment. As a result, the world now has a powerful, robust, and full-featured operating system that continues to change and grow.

In other words, Linux is little bit harder to manage than something like Windows, but offers more flexibility and configuration options.

Linux is licensed under the GPL (General Public license) from the GNU organization, under which the kernel is provided with the source code, and is available for free. As a result, Linux is considered to be more secure and stable than closed source or proprietary systems like Windows because anyone can analyse the source code written in the C language and find bugs or add new features. One important point that should be noted is that even though the source is free, anyone is allowed to sell it for profit.

Linux is known as an *open source* operating system and also called *free software* because everything about Linux is accessible to the public and is freely available to anyone. Since the Linux source code is available, anyone can copy, modify, and distribute this software. This allows for various companies such as SuSE, Red Hat, Caldera and others to sell and distribute Linux; however, at the same time, these companies must keep their Linux distribution code open for public inspection, comment, and changes. Despite of the command-line origins of Linux, these distributing companies are working to make the Graphical User Interface (GUI).

The GNU General Public License

To make software free, you need a license that defines the rights and the limits, that have to be regarded by the open source developer that wants to obtain, edit and eventually redistribute your source code. Because of that exists the GNU GPL (General Public License). Of course, there are also other licenses, but today's most open source programs are distributed under this popular license.

The GNU project was started in 1984 and "GNU is recursive acronym for "GNU's Not Unix"; The Free Software Foundation, which stands for the freedom, the security and the protection of free source code therefore founded this kind of license, designed to protect open source code. GNU is also founder and maintainer of many software packages for the Linux operating system, such as basic tools and file system software.

Is Linux Right for you?

It depends on you and what you would like to do. Linux is not an all-purpose operating system and it would probably be more suited for some people and not so pleasing for others. If you are a person using your computer for some entertainment at home and are satisfied with your Windows system there are no compelling reasons for switching over to Linux, but you do have a choice now. There are several other reasons to consider Linux. Linux is not just a simple operating system. It is an entire server and desktop environment, equipped with add-ons, GUI tools and interfaces, and supplementary programs.

You can use Linux at home and even in college to understand the commands and even the internal workings of UNIX systems.

Distributions

When people use the name Linux they are probably referring to a particular distribution of Linux. There are several software packages provided for Linux over the Internet but selecting and downloading one is a complicated task not necessarily manageable for new users who want to try out Linux. This is exactly where a distribution kicks in.

A distribution is a set of software packages that are tested and provided on CD by a company for a small fee just like Windows. The advantages of using distributions are the support and manuals, as well as the fact that Linux can be specialized for use in a particular area. For example, if you would like using Linux for embedded systems a distribution may offer just the right amount of required software, leaving out optional things like the graphical user interface. So you get what you want instead of a general package for all users.

The mainstream distributions, which are seemingly popular, are RedHat, SuSE, Caldera and Debian. Among these distributions RedHat seems to be most widespread.

Caldera is probably more suited for those who are already using Windows. SuSE is a German based distribution known for its large number of bundled packages and support. Debian is unique because its not owned by a company and it's a non-profit volunteer-based distribution developed solely by users.

Getting Started with Linux

Once the installation is complete, the system will reboot and start up with Linux. There are a series of messages on the screen while booting of the system regarding the hardware enabled, services started etc. After a while, the system will display a login: prompt. You can now log in.

Some systems are configured to start graphical mode with a box in the middle containing both login: and Password: prompts. Press *[CTRL]-[ALT]-[F1]* to switch to the virtual console (text login screen), where you can log in to the system in the usual way.

Accounts and Privileges

Linux is a multi-user system, meaning that many users can use one Linux system simultaneously, from different terminals. So to avoid confusion, each user's workspace must be kept separate from the others.

Even if a particular Linux system is a stand-alone personal computer with no other terminals physically connected to it, it can be shared by different people at different times, making the separation of user workspace is important.

This separation is accomplished by giving each individual user an *account* on the system. You need an account in order to use the system; with an account you are issued an individual workspace to use, and a unique *username* that identifies you to the system and to other users. It is the name along with the password by which the system will recognize the user.

Logging into the System

To begin a session on a Linux system, you need to *log in*. Do this by entering your username at the login: prompt on your terminal, and then entering your password when asked.

Every Linux system has its own name, called the system's *hostname*; a Linux system is sometimes called a *host*, and it identifies itself with its hostname at the login: prompt. It's important to name your system -- like a username for a user account, a hostname gives name to the system you are using (and it becomes especially important when putting the system on a network). The system administrator usually names the system when it is being initially configured (the hostname can be changed later; its name is kept in the file `/etc/hostname'). The name of the terminal you are connecting from is displayed just after the hostname.

To log in to the system, type your username (followed by) at the login: prompt, and then type your password when asked (also followed by); for security purposes, your password is not displayed on the screen when you type it.

Once you've entered your username and password, you are "logged in" to the system. You can then use the system and run commands.

As soon as you log in, the system displays the contents of `/etc/motd', the "Message of the Day" file. The system then displays the time and date of your last login, and reports whether or not you have electronic mail waiting for you. Finally, the system puts you in a *shell*---the environment in which you interact with the system and give it commands. Bash is the default shell on most Linux systems.

The dollar sign (`$') displayed to the left of the cursor is called the *shell prompt*; it means that the system is ready and waiting for input. By default, the shell prompt includes the name of the current directory.

Logging Out of the System

To end your session on the system, type *logout* at the shell prompt. This command logs you out of the system, and a new login: prompt appears on your terminal.

- To log out of the system

  $ *logout*

You can also logout by just pressing *Ctrl+d*.

Logging out of the system frees the terminal you were using and ensures that nobody can access your account from this terminal.

Console Basics

A Linux *terminal* is a place to put input and get output from the system, and usually has at least a keyboard and monitor.

When you access a Linux system by the keyboard and monitor that are directly connected to it, you are said to be using the *console* terminal.

Linux systems feature *virtual consoles*, which act as separate console displays that can run separate login sessions, but are accessed from the same physical console terminal. Linux systems are configured to have seven virtual consoles by default. When you are at the console terminal, you can switch between virtual consoles at any time, and you can log in and use the system from several virtual consoles at once.

Switching Between Consoles

To switch to a different virtual console, press **[ALT]-[F*n*],** where *n* is the number of the console to switch to.

- To switch to the fourth virtual console, press **[ALT]-[F4].**

You can also cycle through the different virtual consoles with the left and right arrow keys. To switch to the next-lowest virtual console, press [ALT]-[←]and to the next-highest virtual console, press **[ALT]-[→].**

- To switch from the fourth to the third virtual console, press   **[ALT]-[←]**

The seventh virtual console is reserved for the X Window System. If X is installed, this virtual terminal will never show a login: prompt, but when you are using X, this is where your X session appears. If your system is configured to start X immediately, this virtual console will show an X login screen.

You can switch to a virtual console from the X Window System using [CTRL] in conjunction with the usual [ALT] and function keys. This is the only console manipulation keystroke that works in X.

- To switch from X to the first virtual console, press: [CTRL]-[ALT]-[F1]

Running a Command

A *command* is the name of a tool that performs a certain function along with the options and arguments. Commands are case sensitive.

To run the hostname command just type the command in front of prompt ($)

> $ *hostname*

Options always begin with a hyphen character, `-', which is usually followed by one alphanumeric character. Always separate the command, each option, and each argument with a space character.

Long-style options begin with two hyphen characters (`--').

For example, many commands have an option, `--version', to output the version number of the hostname.

> $ *hostname --version*

Sometimes, an option itself may take an argument. For example, hostname has an option for specifying a file name to use to read the hostname from, `-F'; it takes as an argument the name of the file that hostname should read from. To run hostname and specify that the file `host.info' is the file to read from

> $ *hostname -F host.info*

Changing Your Password

To change your password, use the passwd command. It prompts you for your current password and a new password to replace it with. You must type it exactly the same way both times, or passwd will not change your password.

> $ passwd username

Listing Your Username

Use whoami to output the username of the user that is logged in at your terminal.

> $ *whoami*

Listing Who Is on the System

Use who to output a list of all the users currently logged in to the system. It outputs a minimum of three columns, listing the username, terminal location, and time of login

for all users on the system. A fourth column is displayed if a user is using the X Window System.

```
$ who
abc    tty1    Oct 20 20:09
def    tty2    Oct 21 14:37
def    ttyp1   Oct 21 15:04 (:0.0)
$
```

The output in this example shows that the user abc is logged in on tty1 (the first virtual console on the system), and has been on since 20:09 on 20 October. The user def is logged in twice -- on tty2 (the second virtual console), and ttyp1, which is an X session with a window location of `(:0.0)'.

Listing the Last Times a User Logged In

Use last to find out who has recently used the system, which terminals they used, and when they logged in and out.

```
$ last abc
```

Listing System Activity

When you run a command, you are starting a *process* on the system, which is a program that is currently executing. Every process is given a unique number, called its *process ID*, or "PID."

Use ps to list processes on the system. By default, ps outputs 5 columns: process ID, the name of the terminal from which the process was started, the current status of the process (including `S' for *sleeping*, meaning that it is on hold at the moment, `R' meaning that it is running, and `Z' meaning that it is a process that has already died), the total amount of time the CPU has spent on the process since the process started, and finally the name of the command being run.

Listing Your Current Processes

Type *ps* with no arguments to list the processes you have running in your current shell session.

```
$ ps

 PID TTY STAT TIME COMMAND

 193   1 S    0:01 -bash

 204   1 S    0:00 ps

$
```

Listing All of a User's Processes

To list all the running processes of a specific user, use ps and give the username to list as an argument with the `-u' option.

```
$ ps -u abc
```

Listing All Processes on the System

To list all processes running by all users on the system, use the `aux' options.

```
$ ps aux
```

Listing Processes by Name or Number

To list processes whose output contains a name or other text to match, list all processes and pipe the output to grep. This is useful for when you want to see which users are running a particular program or command.

To list all the processes whose commands contain reference to an `sbin' directory in them

       $ ps aux | grep sbin

To list any processes whose process IDs contain a 13 in them

       $ ps aux | grep 13

To list the process, which corresponds to a process ID, give that PID as an argument to the `-p' option (PID is 344)

       $ ps -p 344

Finding the System Manual of a Command

Use the man command to view a page in the system manual. As an argument to man, give the name of the program whose manual page you want to view.

       $ man ps

Use the up and down arrow keys to move through the text. Press [Q] to stop viewing the manual page and exit man.

Working with Shell

S*hell* is a program that reads your command input and runs the specified commands. The shell environment is the most fundamental way to interact with the system -- you are said to be in a shell from the very moment you've successfully logged in to the system.

The `$' character preceding the cursor is called the *shell prompt*; it tells you that the system is ready and waiting for input.

If your shell prompt shows a number sign (`#') instead of a `$', this means that you're logged in with the superuser, or root, account. Beware: the root account has complete control over the system; one wrong keystroke and you might accidentally break it something awful. You need to have a different user account for yourself, and use that account for your regular use.

Every Linux system has at least one shell program, and most have several. The standard shell on most Linux systems is bash("Bourne again shell").

Running a List of Commands

To run more than one command on the input line, type each command in the order you want them to run, separating each command from the next with a semicolon (`;'). For example, to clear the screen and then log out of the system

       $ clear; logout

Redirecting Input and Output

The shell moves text in designated "streams." The *standard output* is where the shell streams the text output of commands -- the screen on your terminal, by default. The

standard input, typically the keyboard, is where you input data for commands. You can redirect these streams -- to a file, or even another command -- with *redirection*.

Redirecting Input to a File

To redirect standard input to a file, use the `<' operator. To do so, follow a command with < and the name of the file it should take input from. For example, to redirect standard input for ls -l to file `listing'

> $ ls -l < listing

Redirecting Output to a File

Use the `>' operator to redirect standard output to a file. If you redirect standard output to an existing file, it will overwrite the file, unless you use the `>>' operator to *append* the standard output to the contents of the existing file. For example, to append the standard output of *ls -l* to an existing file `commands'

> $ *ls -l>> commands*

Redirecting Output to another Command's Input

Piping is to connect the standard output of one command to the standard input of another. You do this by specifying the two commands in order, separated by a vertical bar character, `|' (also called as a "pipe"). Commands built in this fashion are called *pipelines*.

For example, it's often useful to pipe commands that display a lot of text output to more for perusing text.To pipe the output of apropos bash shell shells to less

> $ *ls –l | more*

Managing Jobs

The processes you have running in a particular shell are called your *jobs*. You can have more than one job running from a shell at once, but only one job can be active at the terminal, reading standard input and writing standard output. This job is the *foreground* job, while any other jobs are said to be running in the *background*.

The shell assigns each job a unique *job number*. Use the job number as an argument to specify the job to commands. Do this by giving the job number preceded by a `%' character.

Suspending a Job

Type *Ctrl+z* to suspend or stop the foreground job. This is useful when you want to do something else in the shell and return to the current job later. The job stops until you either bring it back to the foreground or make it run in the background.

For example, if you are finding a file at Linux partition from root (/), typing *Ctrl+z* will suspend the find program and return you to a shell prompt where you can do something else. The shell outputs a line giving the job number (in brackets) of the suspended job, the text `Stopped' to indicate that the job has stopped, and the command line itself, as shown here:

> [1]+ Stopped   find / -name abc

In this example, the job number is 1 and the command that has stopped is `find / -name abc'. The `+' character next to the job number indicates that this is the most recent job.

If you have any stopped jobs when you log out, the shell will tell you this instead of logging you out:

> $ *logout*
>
> There are stopped jobs.
>
> $

At this point you can list your jobs, stop any jobs you have running and then log out.

Putting a Job in the Background

New jobs run in the foreground unless you specify otherwise. To run a job in the background, end the input line with an ampersand (`` `&' ``). This is useful for running non-interactive programs that perform a lot of calculations. To run the command find / -name abc > shell-commands as a background job

> $ *find / -name abc > shell-commands &*
>
> [1] 6575
>
> $

The shell outputs the job number (in this case, 1) and process ID (in this case, 6575), and then returns to a shell prompt. When the background job finishes, the shell will list the job number, the command, and the text `` `Done' ``, indicating that the job has completed successfully:

> [1]+ Done   find / -name abc >shell-commands

To move a job from the foreground to the background, first suspend it and then type *bg* (for "background").

- For example, to start the command *find / -name abc > shell-commands* in the foreground, suspend it, and then specify that it finish in the background, you would type:

> $ *find / -name abc > shell-commands*
>
> *Ctrl+z*
>
>
> [1]+ Stopped   find / -name abc >shell-commands
>
> $ *bg*
>
> [1]+ find / -name abc &
>
> $

If you have suspended multiple jobs, specify the job to be put in the background by giving its job number as an argument. TFor example, to run job 4 in the background

> $ *bg %4*

Putting a Job in the Foreground

Type *fg* to move a background job to the foreground. By default, fg works on the most recent background job. For example, to bring the most recent background job to the foreground

> $ *fg*

To move a specific job to the foreground when you have multiple jobs in the background, specify the job number as an option to fg. To bring job 3 to the foreground

$ *fg %3*

Listing Your Jobs

To list the jobs running in the current shell, type *jobs*.

$ *jobs*

[1]- Stopped   find / -name abc >shell-commands

[2]+ Stopped   find / -name abc >bash-commands

$

This example shows two jobs--- *find / -name abc > shell-commands* and *find / -name abc > bash-commands*. The `+' character next to a job number indicates that it's the most recent job, and the `-' character indicates that it's the job *previous* to the most recent job. If you have no current jobs, jobs returns nothing.

Stopping a Job

Typing *Ctrl+c* interrupts the foreground job before it completes, exiting the program. To interrupt *cat*, a job running in the foreground

$ *cat*

Ctrl+c

$

Use kill to interrupt ("kill") a background job, specifying the job number as an argument. To kill job number 2

$ *kill %2*

Command History

Your command *history* is the sequential list of commands you have typed, in the current or previous shell sessions. The commands in this history list are called *events*.

By default, bash remembers the last 500 events, but this number is configurable.

Your command history is stored in a text file in your home directory called `.bash_history'; you can view this file or edit it like you would any other text file.

Viewing Your Command History

Use history to view your command history. To view your command history

$ *history*

1 who

2 apropos shell >shell-commands

3 apropos bash >bash-commands

4 history

$

This command shows the contents of your command history file, listing one command per line prefaced by its *event number*. Use an event number to specify that event in your history. To search your history for the text `find'

> $ *history | grep find*

Specifying a Command from Your History

You can specify a past event from your history on the input line, in order to run it again.

The simplest way to specify a history event is to use the up and down arrow keys at the shell prompt to browse your history. The up arrow key takes you back through past events, and the down arrow key moves you forward into recent history. When a history event is on the input line, you can edit it as normal, and type to run it as a command; it will then become the newest event in your history.

To run a history event by its event number, enter an exclamation mark (`!') followed by the event number (1).

> $ *!1*

Biological Databases

K. K. Chaturvedi

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Bioinformatics is the field of science in which biology, physics, chemistry, mathematics. Statistical and computer science, information and communication technology become a single discipline. It is emerging field that application of computer to **collection, organization, storing, maintaining, accessing, sharing,** analysis, interpretation and presentation of biological data (nucleotide and amino acids sequences, protein domains, protein structures) which helps to accomplishing life science research.

The potential flood of sequence data and the rapidly evolving database technologies empowered researchers to establish international DNA data banks in the early 1980s. Today, we have massive sequence data in the public biological databases due to concerted effort at a number of molecular biology laboratories throughout the world, and the internet and computer technologies. At the beginning, the main concern of bioinformatics was the creation and maintenance of database to store nucleotide and amino acid sequences with wen based interfaces user can access existing data and submitting new data to the database. Hence, database creation and maintenance is major components in bioinformatics. Now, emphasis has shifted to decipher the functional, structural and evolutionary clues encoded in the languages of biology, in which sequences is represented by as sentence, motifs and patterns are by words and nucleotides and amino acids are by letters. However, database design and management is core area in bioinformatics.

Data represents facts or value of results and relations between them have the capacity to represent information (Figure 1). Patterns of relationship between information have the capacity to represent knowledge. Each data is assigned to one data type, which indicates possible relationship with other data. For example; text, integer, float/double, character, time, date and binary.

A **database** is a collection of data organized in the way which can be easily, stored, accessed and managed. Database system is amalgamation of database, database management system and users. (Fig. 1)

Types of Database models

In mid of 1960 the "database" word was first introduced with direct-access-storage. Charles Bachman has introduced Integrated Data Store (IDS), founded, the group "Database Task Group" responsible for the creation and standardization of COBOL. In 1971 the DTG within CODASYL (Conference on Data Systems Languages) delivered standard for database, which generally became known as the "Codasyl approach", this led to network database. Same period IBM was developed IMS (Information Management System), which is similar to Codasyl approach and used hierarchical model of data. Edgar Codd worked at IBM in San Jose, California and he was unhappy with the above two models. He wrote a number of papers those illustrated a new approach based on relational algebra for construction of database that led to a well accepted Relational Model of Data for Large Shared Data Banks. This based on concept relational algebra. There are three main types of database models; 1) Network Model, 2)

Hierarchy Model, and 3) Relational Model. Main objective of these models is integration of data, which is process of combining data of different sources under single query interface.
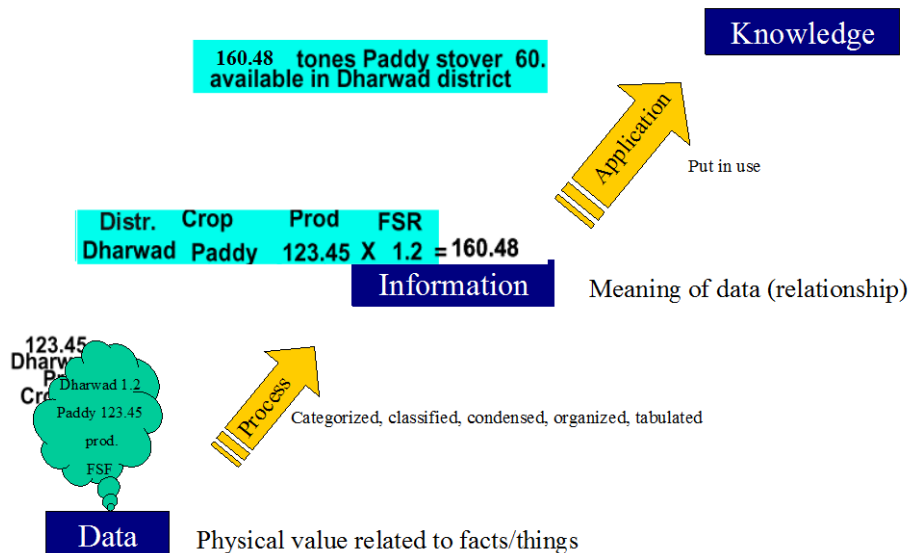
Fig. 1: Data to knowledge

Network Database Model

This model visualizes data in a flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice.

Hierarchical database model

This model is a data model in which the data is organized into a reverse tree-like structure. In this data can be represented as parent and child relationships by 1 to many relationships that each parent can have many children, but each child has only one parent. All attributes of a specific record are listed under an entity type.

Relational Database Model

In this model, database structure is represented in terms of tuples (rows), grouped into relations (tables) and values in each columns of tuple are represented as attributes values (data) and identified solely by the attribute name (Field).

Major Components and Architecture of Database System
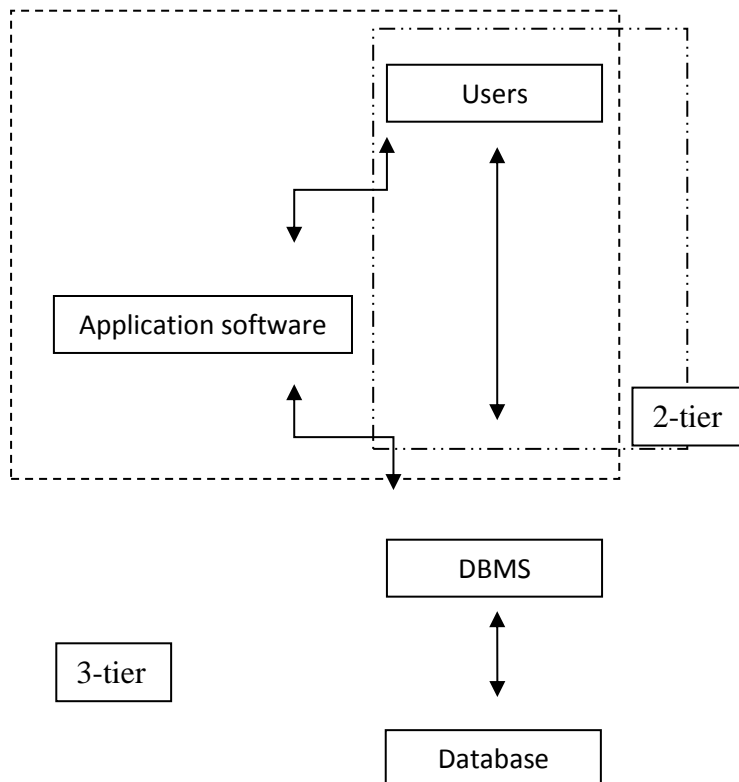
Fig. 2: Architecture of Database

- Users: DB Administrator, Developer and end-user.

- Application: Application software to any specific domain.

- DBMS: Software for creation, insertion, deletion and modification.

- Database: Collection of data

Database architecture logically divided in to two types

- 2 - tier: End-user < -- > DBMS; Here end-user/client can directly communicate with database server.

- 3- tier: End-user < -- > Application Software < -- > DBMS; Here end-user/client will communicate with database server through application tools.

Basic Concept of DataBase Management System (DBMS)

Database Management Systems (DBMS) is specially designed applications software that designed to interact with the user, other applications and database(s) to capture and analyse data. The DBMS have facilities to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SAP, MS Access, FoxPro, IBM DB2/TeraByte, etc. Now database have generally portable across different DBMS by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database.

Major functions of DBMS

- Data definition: Defining new data structures, removing and modifying the existing structure.

- Update: Inserting, modifying, and deleting data.

- Retrieval: Obtaining information for end-user queries or for applications.

- Administration: Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

Benefits of DBMS

- Segregation of work to end-users

- Easy editing, maintenance and retrieval

- Minimizing data duplication

- Reducing time in development and maintenance

- Data security

- Multiple user accessing

- Backup and recovery

Relational Database Management System (RDBMS)

A Relational database Management System (RDBMS) is a database management system to manage relational database based on relation database model as discussed above, which is introduced by E. F. Codd. In this data is represented in terms of tuples (rows) Relational database is collection of tables, table is consist of rows usually called as records and columns called as field or attributes, and columns are identified by unique name. Table is most simplest and fundamental unit of data storage. Each table has its own primary key (one or more fields), which ensures that uniqueness of each record with set of fields. The keys are very important part of relational database. They are used to establish and identify relationship between tables. The RDBMS supports Structured Query Language (SQL).

Normalization

Normalization is a systematics pre-process of decomposing tables to eliminate data redundancy. This will help to easy insertion, updation and deletion. Normalization rule are divided into following form

- First Normal Form: Row cannot contain repeating group of data.

- Second Normal Form: Remove partial dependency between columns

- Remove transitive functional dependency

- Boyce and Codd Normal Form: This deals with certain anomaly that is not handled by3NF.

Entity-Relationship (E-R) Diagram

ER diagram is visual diagrammatic representation of data with standard symbols and notation, which describes how data is related to each other (Fig. 3).

Major symbols and notations
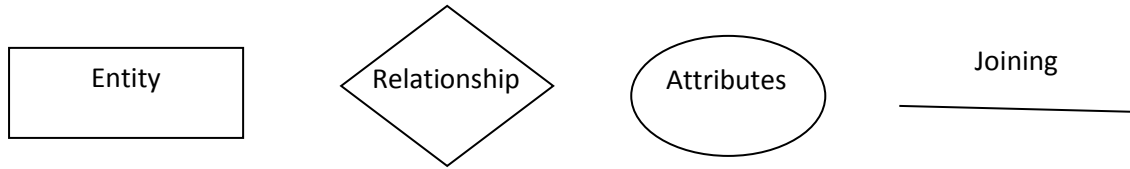
Entity may be any object, person, place and etc. Attributes are features or characteristics. For Example livestock census statistics is shown in table 1.

Table 1: Livestock data before normalization

State	State Capital	Dist	Dist Head Qrts	Year	Animal	Category	Population	Population (000)
Karnataka	Bangalore	Dharwad	Dharwad	2007	Cattle	< 1 year	14355	14.356
Karnataka	Bangalore	Dharwad	Dharwad	2007	Cattle	1-2.5 year	24675	24.675
Karnataka	Bangalore	Dharwad	Dharwad	2007	Cattle	>2.5 year	44355	44.355
Karnataka	Bangalore	Uttar Kannada	Karwar	2007	Cattle	< 1 year	45255	45.255
Karnataka	Bangalore	Uttar Kannada	Karwar	2007	Cattle	1-2.5 year	56555	56.555
Karnataka	Bangalore	Uttar Kannada	Karwar	2007	Cattle	>2.5 year	1836	1.836

The ER diagram for the table 1 is shown in Fig. 4.
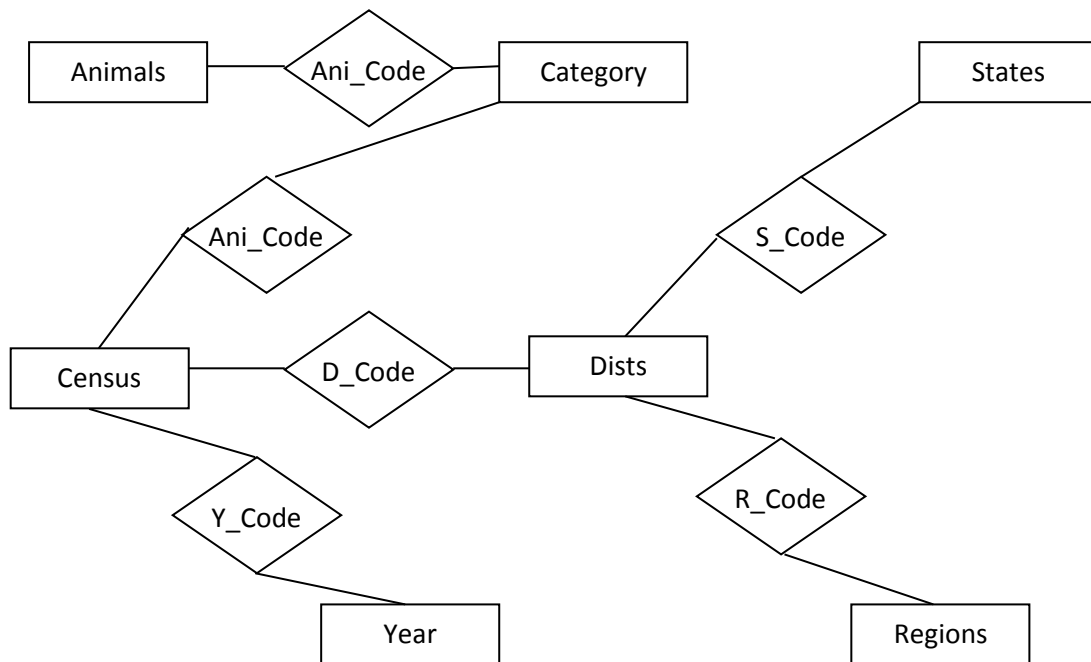
Fig. 4: ER-Diagram

The relationships of the tables are shown in Fig. 5.
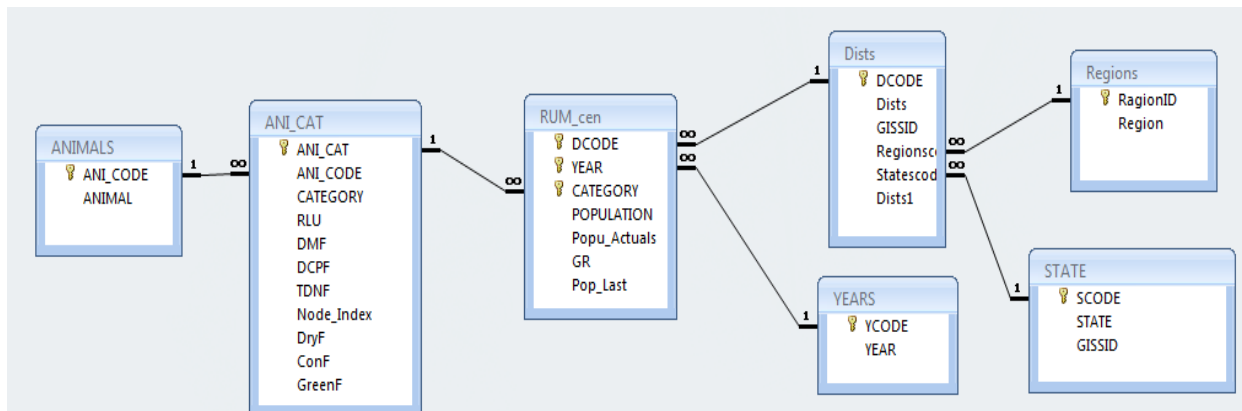
Fig. 5: Relationship diagram from MS Access

Structured Query Language (SQL)

SQL is a tool for communicate with database. SQL is a plat form independent common language is used to perform all types of data operation such as data defining, storing and managing in RDBMS database concept. Now, all RDBMS software employs this language as standard database language. Some of the sample commands are mentioned in table 2.

Table 2: Sample of SQL commands

Command	Description	Syntax
Data Definition		
create	To create new table or database	CREATE TABLE "tablename" ("column1_name" "data type", "column2_name" "data type", ". . . ")
alter	For alteration	ALTER TABLE table_name ADD column_name datatype; ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE table_name MODIFY COLUMN column_name datatype;
drop	To drop a table	DROP TABLE "tablename"
rename	To rename a table	RENAME TABLE tbl_name TO new_tbl_name;
Data Manipulation		
Insert	To insert a new row	INSERT INTO tablename" (column1,... column_last) VALUES (value1, ... value_last);
update	To update existing row	UPDATE "tablename" SET "columnname" = "newvalue" [,"nextcolumn" = "newvalue2"...] WHERE "columnname" OPERATOR "value" [AND|OR "column" OPERATOR "value"];
delete	To delete a row	DELETE FROM "tablename" WHERE "columnname" OPERATOR "value" [AND|OR "column" OPERATOR "value"];
Transaction control		
commit	To permanently save	COMMIT;
rollback	To undo change	ROLLBACK;
savepoint	To save temporarly	SAVEPOINT SAVEPOINT_NAME;
Data query		
select		SELECT[ALL| DISTINCT] *column1* [,*column2*] FROM *table1* [,*table2*] [WHERE "conditions"] [GROUP BY "column-list"] [HAVING "conditions] [ORDER BY "column-list" [ASC | DESC]]

Biological Database

Life science is a field which generates an enormous amount of un-integrated data. Biological databases are collection of life sciences data, information and knowledge collected from different sources such as scientific experiments, published literature, high-throughput

experiment, and computational & statistical analyses in form text, numbers, videos, images and diagrams. These data are broadly classified into four categories based type of data such as literature, sequences, structures and micro-array data. Also area wise classified into Genomics, Proteomics, Metabolomics, and Micro-array (gene expression) and Phylogenetics.

Primary Genomic Databases

- GenBank (National Center for Biotechnology Information) url: http://www.ncbi.nlm.nih.gov/genome

- DNA Data Bank of Japan (National Institute of Genetics) url: http://www.ddbj.nig.ac.jp/

- European Nucleotide Archive (European Bioinformatics Institute) url: http://www.ebi.ac.uk/ena/

Primary Protein Databases

- Uniprot (Universal Protein Resources) url:www.uniprot.org

- PDB url: www.rcsb.org/pdb/

Metabolomics databases

- META Cyc url: http://metacyc.org/

- KEGG: url : http://www.genome.jp/kegg/pathway.html

- Plant Metabolic Network (PMN) url: http://www.plantcyc.org/

Phylogenetics databases

- PhylomeDB url: http://phylomedb.org

- TreeBASE url: http://treebase.org

Microarray Database

- EMBL-EBI microarray database array express url: http://www.ebi.ac.uk/arrayexpress/

- Stanford University database url: http://smd.princeton.edu/

- Gene expression Omnibus (GEO) (NLM) url: http://www.ncbi.nlm.nih.gov/geo/

- ExpressDB - Harvard url: http://arep.med.harvard.edu/ExpressDB/

Similarly many bioinformatics databases such as Compound-Specific Databases, Comprehensive Metabolomic Database, drug database, RNA database, SNP database, Microsatellites, Literature database, Crystallographic database, NMR spectra database, Carbohydrate structure databases, Protein-protein interactions database, Signal transduction pathway databases, primer databases, Taxonomic databases and etc.

Introduction to NGS Data

Dwijesh Chandra Mishra

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

DNA sequencing is a biochemical method in order to determine the correct order of nucleotide bases in a DNA macromolecule by using sequencing machines. Earlier sequencing was based on a single type of method that is Sanger sequencing. In 2005, Next Generation Sequencing (NGS) Technologies emerged and changed the view of the analysis and understanding of living beings. Over the last two decade, considerable progress has been made on new sequencing methods. NGS is modern high-throughput DNA sequencing technologies. They are faster, cheaper, rapid and parallel. They require much less template preparation than the Sanger sequencing technique.

First Generation Sequencing

1. Dideoxy method or chain termination method:

This method is developed by Sanger and Coulson in 1977. In this method one strand of the double stranded DNA is used as template to be sequenced. This sequencing is based on using chemically modified nucleotides called dideoxy-nucleotides (ddNTPs). The dideoxynucleotides are used in elongation of DNA complementary strand, once incorporated into the DNA strand they prevent the further elongation. The sequencing reaction is carried out in four test tubes which consist of various components besides the templates. These components are a small stretch of DNA sequence called primer, DNA polymerase enzyme, a mixture of four deoxy nucleotide triphosphate (A, T, G, and C) and one of the dideoxy nucleotide, i.e. either ddATP, ddTTP, ddGTP or ddCTP labeled with radioactive substances or non-radioactive substances like dig or biotin. The synthesis of new DNA strand continues in the presence of DNA polymerase enzyme until a dideoxynucleotide is added in the complementary DNA strand which results in the generation of different sized DNA fragments, ending with labeled ddNTPs. After the reaction is complete the reaction mixture of all the four tubes are loaded adjacent to each other on a polyacrylamide sequencing gel. The four lanes specific to ddATP, ddCTP, ddGTP and ddTTP produce fragments of varying length upon electrophoresis and autoradiography. The position of bands in the gel is used to directly read DNA sequences from bottom to top. The automated version of this method uses ddNTPs that are labeled with different color fluorescent dyes so that all four reactions can be run in a single tube.

2. Chemical cleavage method of DNA sequencing:

This method is developed by Maxam and Gilbert in 1977. This method uses chemicals to break DNA molecules of specific bases, thus creating fragments of different sizes. DNA molecule to be sequenced is radiolabeled. The sequencing reaction is devised into four tubes along with a fifth reference tube.

Chemicals	Reaction
Dimethyl sulphate	Alters guanine at N7 position by methylation
Acid	Alters either adenine or guanine
Hydrazine	Alters either thymine or cytosine
Hydrazine + NaCl	Alters cytosine
NaOH	Reference

For removing altered basepairs from the sequencing reaction, piperidine is added in each tube. Piperidine breaks the DNA molecules at the sugar residue from the point of altered nucleotide thus making different sized fragments of DNA. The mixture of DNA fragments are separated on high resolution polyacrylamide gels by loading the contents of all the four tubes in adjacent lanes. After electrophoresis, the gels are exposed to x-ray film for developing autoradiographs of the DNA bands from which sequence is read.

Second Generation Sequencing

The first generation of sequencing especially Sanger sequencing was extensively used for three decades, however, the cost and time was a major drawback. In 2005, second generation sequencing technologies came into the market which eliminates the limitations of the first generation sequencing. The basic characteristics of second generation sequencing technology are: (1) The generation of many millions of short reads in parallel, (2) The speed up of sequencing the process compared to the first generation, (3) The low cost of sequencing and (4) The sequencing output is directly detected without the need for electrophoresis. Short read sequencing approaches divided under two wide approaches: sequencing by ligation (SBL) and sequencing by synthesis (SBS).

1. 454/Roche:

Roche/454 sequencing appeared on the market in 2005, using pyrosequencing technique which is based on the detection of pyrophosphate, released after each nucleotide incorporation in the new synthetic DNA strand (http://www.454.com). The pyrosequencing technique is a sequencing-by-synthesis approach. DNA samples are randomly fragmented and each fragment is attached to a bead whose surface carries primers that have oligonucleotides complementary to the DNA fragments so each bead is associated with a single fragment. Then, each bead is isolated and amplified using PCR emulsion which produces about one million copies of each DNA fragment on the surface of the bead. The beads are then transferred to a plate containing many wells called picotiter plate (PTP) and the pyrosequencing technique is applied which consists in activating of a series of downstream reactions producing light at each incorporation of nucleotide. By detecting the light emission after incorporation of each nucleotide, the sequence of the DNA fragment is deduced. The use of the picotiter plate allows hundreds of thousands of reactions occur in parallel, considerably increasing sequencing throughput. The latest instrument launched by Roche/454 called GS FLX+ that generates reads with lengths of up to 1000 bp and can produce ~1Million reads per run. The Roche/454 is able to generate relatively long reads which are easier to map to a reference genome. The main errors detected of sequencing are insertions and deletions due to the presence of homopolymer regions. Indeed, the identification of the size of homopolymers should be determined by the intensity of the light emitted by pyrosequencing. Signals with too high or too low intensity lead to under or overestimation of the number of nucleotides which causes errors of nucleotides identification.

2. Illumina/ Solexa:

Illumina technology is sequencing by synthesis approach and is currently the most used technology in the NGS market. During the first step, the DNA samples are randomly fragmented into sequences and adapters are ligated to both ends of each sequence. Then, these adapters are fixed themselves to the respective complementary adapters, the latter are hooked on a slide with many variants of adapters (complementary) placed on a solid plate. During the second step, each attached sequence to the solid plate is amplified by PCR bridge amplification that creates several identical copies of each sequence. A set of sequences made from the same original sequence is called a cluster. Each cluster contains approximately one million copies of the same original sequence. The last step is to determine each nucleotide in the sequences, Illumina uses the sequencing by synthesis approach that employs reversible terminators in which the four modified nucleotides, sequencing primers and DNA polymerases are added as a mix, and the primers are hybridized to the sequences. Then, polymerases are used to extend the primers using the modified nucleotides. Each type of nucleotide is labeled with a fluorescent specific in order for each type to be unique. The nucleotides have an inactive 3'-hydroxyl group which ensures that only one nucleotide

is incorporated. Clusters are excited by laser for emitting a light signal specific to each nucleotide, which will be detected by a coupled-charge device (CCD) camera and Computer programs will translate these signals into a nucleotide sequence. The process continues with the elimination of the terminator with the fluorescent label and the starting of a new cycle with a new incorporation.

3. ABI/SOLiD:

Sequencing by Oligonucleotide Ligation and Detection (SOLiD) is a NGS sequencer Marketed by Life Technologies (http:// www.lifetechnologies.com). In 2007, Applied Biosystems (ABI) has acquired SOLiD and developed ABI/SOLID sequencing technology that adopts the ligation (SBL) approach. The ABI/SOLiD process consists of multiple sequencing rounds. It starts by attaching adapters to the DNA fragments, fixed on beads and cloned by PCR emulsion. These beads are then placed on a glass slide and the 8-mer with a fluorescent label at the end is sequentially ligated to DNA fragments, and the color emitted by the label is recorded. Then, the output format is color space which is the encoded form of the nucleotide where four fluorescent colors are used to represent 16 possible combinations of two bases. The sequencer repeats this ligation cycle and each cycle the complementary strand is removed and a new sequencing cycle starts at the position n-1 of the template. The cycle is repeated until each base is sequenced twice. The recovered data from the color space can be translated to letters of DNA bases and the sequence of the DNA fragment can be deduced.

4. Ion Torrent:

Life Technologies commercialized the Ion Torrent semiconductor sequencing technology in 2010 (https//www.thermofisher.com/us/en/home/brands/ion-torrent.html). It is similar to 454 pyrosequencing technology but it does not use fluorescent labeled nucleotides like other second-generation technologies. It is based on the detection of the hydrogen ion released during the sequencing process. First, emulsion PCR is used to clonally amplify adapter ligated DNA fragments on the surface of beads. The beads are subsequently distributed into micro-wells where sequencing by synthesis reaction occurs. Ion torrent chip consists of a flow compartment and solid state pH sensor micro-arrayed wells that are manufactured using processes built on standard complementary metal oxide semiconductor (CMOS) technology. The release of H+ during extension of each nucleotide is detected as a change in the pH within the sensor wells. Since there is no detectable difference for H+ released from a A, T, G or C bases, the individual dNTPs are applied in multiple cycles of consecutive order. The speed of sequencing is 2-8 hrs depending on the machine and chip used. Error rate for substitutions is ~0.1%, similar to Illumina. Homopolymer repeats more than 6bp lead to increased error rates.

Third Generation Sequencing

The second generation sequencing technologies generally require PCR amplification step which is a long and expensive procedure. Also, it became clear that the genomes are very complex with many repetitive areas that second generation sequencing technologies are incapable to solve them and the relatively short reads made genome assembly more difficult. Third generation sequencing technologies are remedy to these problems. These third generations of sequencing have the ability to cover a low sequencing cost and easy sample preparation without the need PCR amplification. The execution time reduces significantly than second generation sequencing technologies. The most widely used third generation sequencing technology approach is SMRT (Pacific Biosciences) and Oxford Nanopore sequencing.

1. Pacific biosciences SMRT sequencing

Pacific Biosciences (http//www.pacificbiosciences.com/) developed the first genomic sequencer using SMRT approach and it's the most widely used third-generation sequencing technology. Template preparation involves ligation of single stranded hairpin adapters onto the ends of digested DNA or cDNA molecules, generating a capped template called SMRT-bell. This technology works with single molecule

detection which does not require any amplification step. By using a strand displacing polymerase, the original DNA molecule can be sequenced multiple times, thereby increasing accuracy.

DNA synthesis occurs in zeptoliter sized chambers, called zero-mode waveguides (ZMWs). These ZMW are small reaction wells that each ideally contains one complex consisting of template molecule, sequencing primer and DNA polymerase bound to the bottom of the ZMW. The fluorescent signals of the extended nucleotides are recorded in real time at 75 frames per second for the individual ZMWs. This is achieved by powerful optical system that illuminates the individual ZMWs with red and green laser beamlets from the bottom of the SMRT cell and a parallel confocal recording system to detect the signal from the fluorescent nucleotides. When a nucleotide complementary to the template is bound in position by the polymerase within the illumination zone of the zmw, the identity of the nucleotide is recorded by its fluorescent label. Each SMRT cell produces ~50k reads and upto 1 gb of data in 4 hrs. The average read length is >14 kb. This technology has a high error rate of approximately 11%. This is useful for denovo assembly of small bacterial and viral genomes as well as large genome finishing.

2. Oxford nanopore sequencing

This technology is also based on single molecule strategy. This relies on the transition of DNA or individual nucleotides through a small channel called protein nanopore. A nanopore is a nanoscale hole made of proteins or synthetic materials. A sequencing flow cell comprises hundreds of independent micro-wells, each containing a synthetic bilayer perforated by biological nanopores. Sequencing is accomplished by measuring characteristic changes in ionic current that are induced as the bases are threaded through the pore by a molecular motor protein. Library preparation is minimal, involving fragmentation of DNA and ligation of adapters. The first adapter is bound with a motor enzyme as well as a molecular tether, whereas second adapter is a hairpin oligonucleotide that is bound by a second motor protein. This library design allows sequencing of both strands of DNA from a single molecule, which increases accuracy. The variation in ionic current is recorded progressively on a graphic model and then interpreted to identify the sequence. MinION is released in 2014 which generate longer reads, ensure better resolution structural genomic variants and repeat content.it is a mobile single –molecule nanopore sequencing measures connected by a USB 3.0 port of a laptop computer. PromethION is bigger than MinION, equivalent to 48 MinIONs.

References:

- Kchouk M, Gibrat JF, Elloumi M. (2017). Generations of Sequencing Technologies: From First to Next Generation. Biol Med (Aligarh) 9: 395. doi:10.4172/0974-8369.1000395.

- H.P.J. Buermans, J.T. den Dunnen. (2014). Next generation sequencing technology: Advances and applications, Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease, 1842(10): 1932-1941. https://doi.org/10.1016/j.bbadis.2014.06.015.

Genome Assembly

Dwijesh Chandra Mishra, Sanjeev Kumar, Sudhir Srivastava and Neeraj Budhlakoti

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Sanger Sequencing

- DNA is fragmented

- Cloned to a plasmid vector

- Cyclic sequencing reaction

- Separation by electrophoresis

- Readout with fluorescent tags
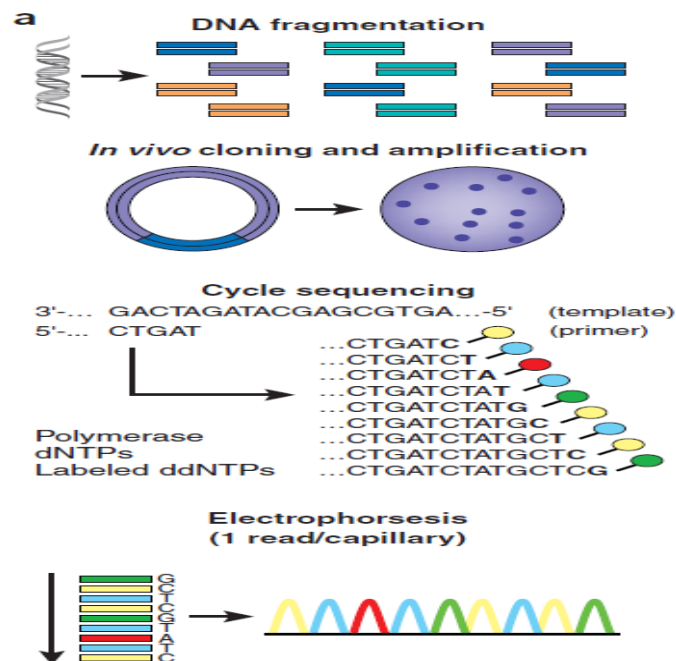
Sanger Vs NGS

- 'Sanger sequencing' has been the only DNA sequencing method for 30 years but…

- …hunger for even greater sequencing throughput and more economical sequencing technology…

- NGS has the ability to process millions of sequence reads in parallel rather than 96 at a time (1/6 of the cost)

NGS Platforms: Different sequencing techniques used for next generation sequencing are:

- Roche/454 FLX: 2004

- Illumina Solexa Genome Analyzer: 2006

- Applied Biosystems SOLiD^{TM} System: 2007

- Helicos Heliscope^{TM} : 2009

- Pacific Biosciencies SMRT: 2010

General Experimental Procedure
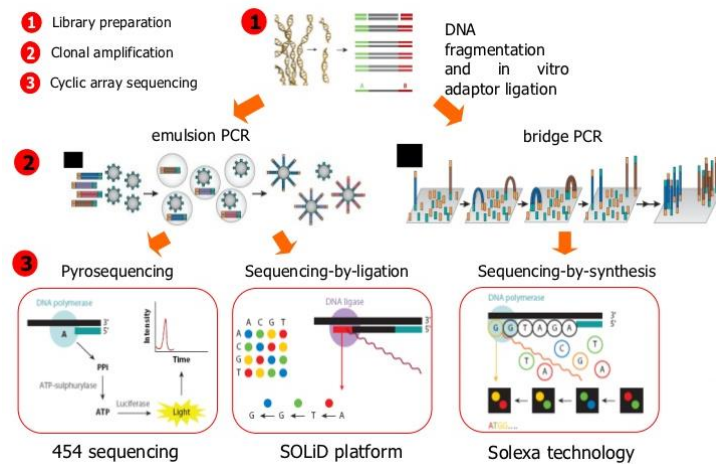
Sequencing Technology at a Glance

Method	Read length	Accuracy	Time per run	Cost per 1 million bases	Advantages	Disadvantages
Chain termination (Sanger sequencing)	400 to 900 bp	99.9%	20 minutes to 3 hours	Rs 144000	Long individual reads. Useful for many applications.	More expensive and impractical for larger sequencing projects.
Pyrosequencing (454)	700 bp	99.9%	24 hours	Rs 600	Long read size. Fast	Runs are expensive. Homopolymer errors.
Sequencing by synthesis (Illumina)	50 to 300 bp	98%	1 to 10 days, depending upon sequencer and specified read length	Rs 3 to 9	Potential for high sequence yield, depending upon sequencer model and desired application.	Equipment can be very expensive. Requires high concentrations of DNA.
Sequencing by ligation (SOLiD sequencing)	50+35 or 50+50 bp	99.9%	1 to 2 weeks	Rs 78	Low cost per base.	Slower than other methods. Have issue sequencing palindromic sequence.
Single-molecule real-time sequencing (Pacific Bio)	10,000 bp to 15,000 bp avg. (14,000 bp);	87%	30 minutes to 4 hours	Rs 7.8–36	Longest read length. Fast.	Moderate throughput. Equipment can be very expensive.

Reads, Contigs and Scaffolds

- Reads are what you start with (35bp-800bp)

- Fragmented assemblies produce contigs that can be kilobases in length

- Putting contigs together into scaffolds is the next step
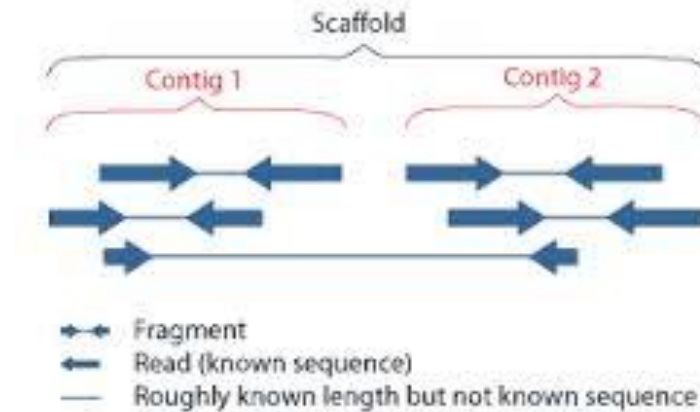
FASTQ Format
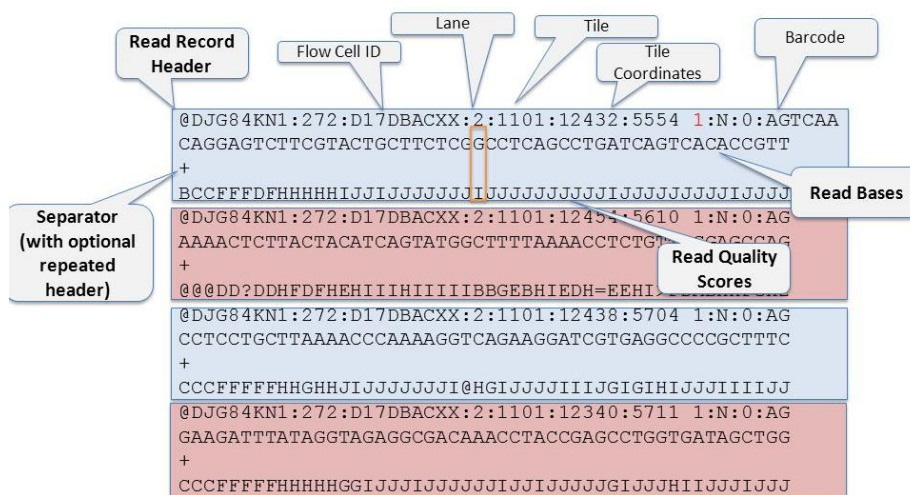
Before Assembly

Fragment readout

- DNA characters in a fragment are determined from chromatogram

- Base call is a DNA character that is determined from chromatogram
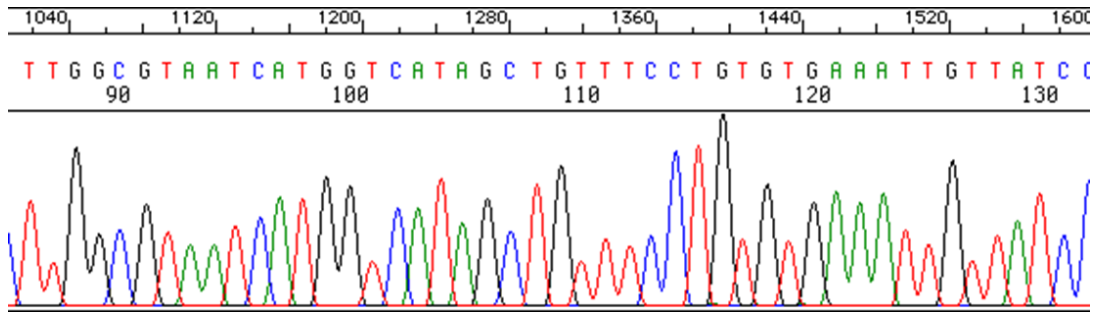
Fragment readout

- Phred Score- determine the quality value of a base

$$q = -10 \times log_{10}(p)$$

where p is the estimated error probability for the base

- if Phred assigns a quality score of 30 to a base, the chances that this base is called incorrectly are 1 in 1000

- The most commonly used method is to count the bases with a quality score of 20 and above

- Phred Score
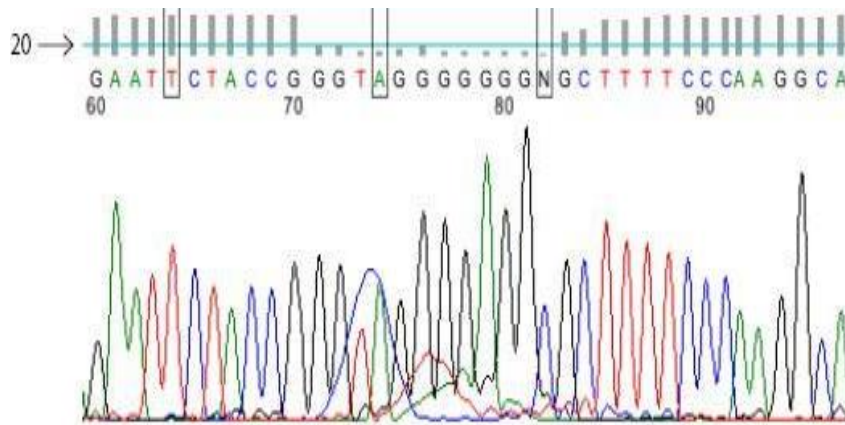
Genome Properties

PASS

FAIL

GC content across all bases

PASS

GC content across all bases

FAIL

Library Quality

20 Mer Uniqueness: 1454.2.1383.merSampler.m20.e25000

PASS

20 Mer Uniqueness: 730.3.966.merSampler.m20.e25000

FAIL

Run Quality

Base Counts by Read Position: 1454.2.1383.s0.05.r1.fastq

PASS

Base Counts by Read Position: 730.3.966.s0.05.fastq

FAIL

Read Quality

Average Read Quality Score: 1454.2.1383.s0.05.fastq.qhist

PASS

Average Read Quality Score: 730.3.966.s0.05.fastq.qhist

FAIL

Read 1 Base Position Quality: 1454.2.1383.s0.05.r1.qrpt

PASS

Read Base Position Quality: 730.3.966.s0.05.qrpt

FAIL

Percent N by Read Position: 1454.2.1383.s0.05.r2.fastq

PASS

Percent N by Read Position: 730.3.966.s0.05.fastq

FAIL

21mer Sliding Window Quality: 1454.2.1383.s0.05.21mer.qual

PASS

21mer Sliding Window Quality: 730.3.966.s0.05.21mer.qual

FAIL

Trimming

- Trimming low-quality sequences

  -removal of reads containing poor quality base calls

- Trimming vector sequences

  -removal of reads containing vector sequences

Sequence Analysis

S. B. Lal

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

1. Introduction

Since the development of high-throughput methods for production of gene and protein sequences during 90s, the rate of addition of new sequences to the databases increases very rapidly. However, comparing sequences with known functions with these new sequences is one way of understanding the biology of that organism from which the new sequence comes. Thus, sequence analysis can be used to study of the similarities between the compared sequences. Now a days, there are many tools and techniques that provide the sequence comparisons (sequence alignment) and analyze the alignment to understand the biology.

Sequence analysis in molecular biology and bioinformatics is an automated, computer-based examination of characteristic fragments, e.g. of a DNA strand. It basically includes relevant topics:

1. The comparison of sequences in order to find similarity and dissimilarity in compared sequences (sequence alignment)

2. Identification of gene-structures, reading frames, distributions of introns, exons and regulatory elements

3. Finding and comparing point mutations or the single nucleotide polymorphism (SNP) in organism in order to get the genetic marker.

4. Revealing the evolution and genetic diversity of organisms.

5. Functional annotation of genes.

Sequence alignment is a way to identify regions of similarity in DNA, RNA, or protein sequences that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Aligned sequences of nucleotide or amino acid residues are typically represented as rows within a matrix. If two sequences share a common ancestor for the alignment, mismatches can be interpreted as point mutations and gaps as indels (that is, insertion or deletion mutations). Thus, a letter or a stretch of letters may be paired up with dashes in the other sequence to signify such an insertion or deletion. Homologous sequences may have different length, which is generally explained through insertions or deletions in sequences. Since an insertion in one sequence can always be seen as a deletion in the other one frequently uses the term "indel". In sequence alignments of proteins, the degree of similarity between amino acids sequence can be interpreted as a rough measure of how conserved a particular region or sequence motif is among lineages. The absence of substitutions, or the presence of only very conservative substitutions (that is, the substitution of amino acids whose side chains have similar biochemical properties) in a particular region of the sequence, suggest that this region has structural or functional importance. Although DNA and RNA nucleotide bases are more similar to each other than are amino acids, the conservation of base pairs can indicate a similar functional or structural role.

Very short or very similar sequences can be aligned by hand. However, most interesting problems require the alignment of lengthy, highly variable or extremely numerous

sequences that cannot be aligned solely by human effort. Computational methods need to be developed for the alignment of a large pair of sequences. Computational approaches are of two categories: *global alignments* and *local alignments*. Global alignment is a form of global optimization that "forces" the alignment to span the entire length of all query sequences. Global alignment will be applied when the sequences are of similar lengths. Local alignments identify regions of similarity within long sequences. Local alignments are often preferable, but it consumes more time to calculate because of the additional challenge of identifying the regions of similarity in the local regions. Number of algorithms is being applied for the sequence alignment, including optimizing methods like dynamic programming, and heuristic algorithms or probabilistic methods designed for large-scale database search.

```
AAB24882    TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881    -------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                               ****: .***:  * *:** * :****.:* *******..


AAB24882    PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881    HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
             **** *:***********:***:**.: .**************     :  *.: :
```

Fig. 1 Sample of sequence Alignment text based representations

In sequence alignment of graphical representations, sequences are written in rows so that aligned residues appear in successive columns. While in text formats, aligned columns containing identical or similar characters are indicated with a system of conservation symbols. An asterisk or pipe symbol is used to represent the similarity of these two columns, a colon for conservative substitutions and a period for semi-conservative substitutions.

Many sequence visualization techniques use a color coding scheme to display information about the properties of the individual sequence elements. In DNA and RNA sequences, each nucleotide is represented by a specific color. In protein alignments, color is used to indicate amino acid properties in determining the conservation of a given amino acid substitution.

2. Pair-wise Alignment

Pair-wise sequence alignment methods are used to find the best-matching pairs of two sequences. The three primary methods of pair-wise alignments are dot-matrix, dynamic programming and word methods. One way of quantifying the utility of a pair-wise alignment is the 'maximum unique match', or the longest subsequence that occurs in both query sequence.

a) Dot-Matrix Method: The two sequences are written along the top row and leftmost column of a two-dimensional matrix and a dot is placed at any point where the characters in the appropriate columns match. We try to draw lines diagonally. The dot plots of very closely related sequences will appear as a single line along the matrix's main diagonal (Fig. 2). The dot-matrix approach produces a simple way of alignments for small sequences with the similar regions but time-consuming to analyze large sequences.
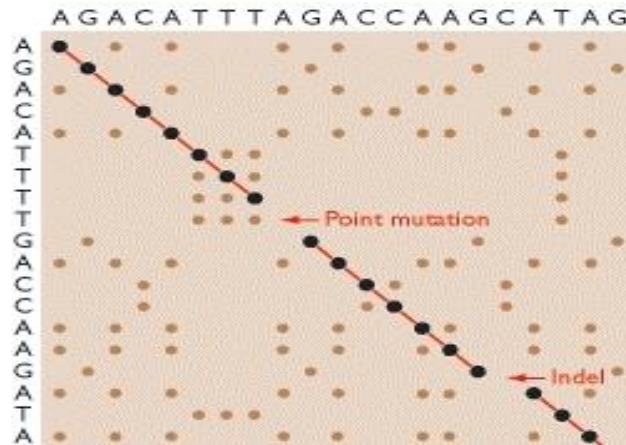
Fig. 2: The dot matrix technique for sequence alignment

There are many problems with dot plots such as noise, lack of clarity, difficulty extracting match summary statistics. Dot-plots are limited to two sequences only.

b) Dynamic Programming: Dynamic programming can be applied to produce global and local alignments. This can be done by applying Needleman-Wunsch algorithm for global alignment and Smith-Waterman algorithm for the local alignments. In general, alignments use a substitution matrix to assign scores for matches or mismatches, and a gap penalty for matching an in one sequence with a gap in the other.

DNA and RNA alignments may use a different scoring matrix, but in practice often simply assign a positive match score, a negative mismatch score, and a negative gap penalty. Dynamic programming can be useful in aligning nucleotide to protein sequences. The framesearch method produces a series of global or local pair-wise alignments between a query nucleotide sequence and a search set of protein sequences, or vice versa. The BLAST and EMBOSS provide basic tools for creating alignments of the sequences.

c) Word Method: Word or *k*-tuple methods are heuristic methods but are not guaranteed to find an optimal alignment solution. These methods are especially useful in large-scale database searches Word methods are best known for their implementation in the database search tools FASTA and BLAST family. Word methods identify a series of short, non-overlapping subsequences ("words") that are matched to candidate database sequences. The relative positions of the word in the two sequences being compared are subtracted to obtain an offset; this will indicate a region of alignment if multiple distinct words produce the same offset.

In the FASTA method, the user defines a value *k* to use as the word length with which to search the database. The method is slower but more sensitive for lower values of *k*, which are preferred for searching a very short query sequence. The BLAST family of search methods provides a number of algorithms optimized for particular types of queries. BLAST was developed to provide a faster alternative to FASTA without sacrificing accuracy. BLAST uses a word search of length *k*, but evaluates only the most significant word matches. Most BLAST implementations use a fixed default word length that is optimized for the query and database. Web based implementations are available such as EMBL FASTA and NCBI BLAST.

3. Global and Local Alignment

Global Alignment

Global alignments, which attempt to align every residue of each sequence, when the size of the sequences are similar or of equal size. A general global alignment technique is based on dynamic programming i.e., Needleman-Wunsch algorithm. This can be easily understood with the following two sequences aligned globally as follows

G A A T T C A G T T A   (sequence #1)
G G A T C G A   (sequence #2)

In simple dynamic programming principle, we construct a matrix. The matrix will be filled by inserting 0 or 1 where ever there is a mismatch or match. We also penalize the gaps with 0 as a simple case. Following steps are needed for construction of the matrix

    i.   Initialization

    ii.   Matrix fill (scoring)

    iii. Traceback (alignment)

i. Initialization

The first step is to create a matrix with M + 1 columns and N + 1 rows where M and N are the sizes of the sequences to be aligned.

With the given sequences, length of sequence #1 = 11 and length of sequence #2 is 7. The size of the matrix will be 12*8 (11+1 * 7+1). The first row and first column of the matrix can be initially filled with 0 because we assume assumes there is no gap opening or gap extension penalty as shown in fig. 3.

Fig. 3. Initial matrix with two sequences

ii. Matrix Fill

One possible way of filling the matrix is to find the maximum global alignment score by starting from the upper left hand corner of the matrix and find the maximal score $M_{i,j}$ for each position in the matrix.

For each position, $M_{i,j}$ is defined to be the maximum score at position i,j i.e.,

$M_{i,j}$ = MAXIMUM[

    $M_{i-1, j-1} + S_{i,j}$ (match/mismatch in the diagonal),

    $M_{i,j-1} + w$ (gap in sequence #1),

    $M_{i-1,j} + w$ (gap in sequence #2)**]**

In fig. 4, $M_{i-1,j-1}$ will be red, $M_{i,j-1}$ will be blue and $M_{i-1,j}$ will be green. The score at position 1,1 in the matrix can be calculated. Since the first residue in both sequences is a G i.e., a match, so score $S_{1,1} = 1$. We assumed the gap penalty as 0.

Thus, $M_{1,1} = MAX[M_{0,0} + 1, M_{1,0} + 0, M_{0,1} + 0] = MAX [1, 0, 0] = 1$.

A value of 1 is then placed in position 1,1 of the scoring matrix.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1										
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Fig. 4. Sample fill of the entry $M_{1,1}$

Now the element $M_{1,2}$, the value is the max of 0 (for a mismatch), 0 (for a vertical gap) or 1 (horizontal gap). The rest of element of first row can be filled up similarly. At this point, there is a G in both sequences (light blue). Thus, the value for the cell at row 1 column 8 is the maximum of 1 (for a match), 0 (for a vertical gap) or 1 (horizontal gap). The value will again be 1 as in fig. 5

		G	A	A	T	T	T	C	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1										
A	0	1										
T	0	1										
C	0	1										
G	0	1										
A	0	1										

Fig. 5. Sample fill of the entry whene there is a collosion of two cells for $M_{1,8}$

Now similarly at column 2. The location at row 2 will be assigned the value of the maximum of 1(mismatch), 1(horizontal gap) or 1 (vertical gap). So its value is 1.

After filling in all of the values the score matrix is shown in fig. 6:

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	3	4	4	4	4	4
G	0	1	2	2	3	3	3	4	4	5	5	5
A	0	1	2	3	3	3	3	4	5	5	5	6

iii. Traceback Step

After the matrix fill step, find the the maximum alignment score for the two test sequences. The traceback step determines the actual alignment(s) that result in the maximum score. Note that with a simple scoring algorithm such as one that is used here, there are likely to be multiple maximal alignments.

The traceback step begins in the matrix that leads to the maximal score. In this case, there is a 6 in that location. Traceback takes the current cell and looks to the neighbor cells that could be direct predecessors. This means that it looks to the neighbor to the left (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor above it (gap in sequence #1). The algorithm for traceback chooses as the next cell in the sequence one of the possible predacessors. In this case, the neighbors are marked in red. They are all also equal to 5 as in fig 7.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5	5
A	0	1	2	3	3	3	4	5	5	5	5	6

Fig. 7. Traceback process start where the score is maximum

Since the current cell has a value of 6 and the scores are 1 for a match and 0 for anything else, the only possible predecessor is the diagonal match/mismatch neighbor. If more than one possible predecessor exists, any can be chosen. The corresponding row and column can be crossed out as in fig. 8. This gives us a current alignment of

(Seq #1)   A

           |

(Seq #2)   A

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	
G	0	1	1	1	1	1	1	1	1	1	1	
G	0	1	1	1	1	1	1	1	2	2	2	
A	0	1	2	2	2	2	2	2	2	2	2	
T	0	1	2	2	3	3	3	3	3	3	3	
C	0	1	2	2	3	3	4	4	4	4	4	
G	0	1	2	2	3	3	4	4	5	5	5	
A												6

Fig. 8. Traceback steps and crossing of the row and column

Now, look at the current cell and determine which cell is its direct predecessor. In this case, it is the cell with the red 5 as in fig. 9. The alignment as described in the above step adds a gap to sequence #2 , so the current alignment is

(Seq #1)   T A

                  |

(Seq #2)   _ A

Once again, the direct predecessor produces a gap in sequence #2.

Fig. 9. Traceback steps and crossing of the row and column

After this step, the current alignment is

(Seq #1)   T T A

                  |

             _ _ A

Continuing on with the traceback step, we eventually get to a position in row 0 and column 0, which tells us that traceback is completed as in fig. 10.

Fig. 10. Final matrix with the traceback steps

One possible maximum alignment is

G A A T T C A G T T A

| | | | | |

G G A _ T C _ G _ _ A

Local Alignment

Local alignments are more useful for dissimilar sequences that may contains regions of similarity or similar sequence motifs within their larger sequence context. The Smith-Waterman algorithm is a general local alignment method based on dynamic programming. A local alignment searches for regions of local similarity between two sequences and need not include the entire length of the sequences. This can be done by reading a scoring matrix that contains values for every possible residue or nucleotide match or mismatch. The Smith-Waterman algorithm is a member of the class of algorithms that can calculate the best score and local alignment in the order of m*n steps, where 'm' and 'n' are the lengths of the two sequences. Local alignment methods only report the best matching areas between two sequences while there may be a large number of alternative local alignments which do not score as highly as the best alignment done by this algorithm.

Consider the two DNA sequences to be globally aligned are:

ACACACT (x=7, length of sequence 1)

AGCACAC (y=7, length of sequence 2)

It also follows three steps

i.   Initialization

ii. Matrix fill (scoring)

iii. Traceback (alignment)

Let us assume the simple scoring scheme as

- $S_{i,j} = 2$ if there is a match

- $S_{i,j} = -1$ if there is a mismatch

- $w = -1$ as gap penalty

i.   <u>Initialization</u>

The first step in the global alignment dynamic programming approach is to create a matrix with M + 1 columns and N + 1 rows where M and N correspond to the size of the sequences to be aligned. In this example, we assume that there is no gap opening or gap extension penalty. The first row and first column of the matrix can be initially filled with 0 as in fig. 11.

		A	C	A	C	A	C	T
	0	0	0	0	0	0	0	0
A	0							
G	0							
C	0							
A	0							
C	0							
A	0							
C	0							

Fig. 11. Initial matrix with first row and first column element as 0

ii. Matrix Fill

One way to fill the matrix is to find the maximum global alignment score by starting from the upper left hand corner in the matrix and get the maximal score $M_{i,j}$ for each position in the matrix. In order to find $M_{i,j}$ for any i,j it is minimal to know the score for the matrix positions to the left, above and diagonal to i, j. In terms of matrix positions, it is necessary to know $M_{i-1,j}$, $M_{i,j-1}$ and $M_{i-1, j-1}$.

For each position, $M_{i,j}$ is defined to be the maximum score at position i,j; i.e.

$M_{i,j}$ = MAXIMUM[

$M_{i-1, j-1} + S_{i,j}$ (match/mismatch in the diagonal),

$M_{i,j-1} + w$ (gap in sequence #1),

$M_{i-1,j} + w$ (gap in sequence #2)]

Using this information, the score at position 1,1 in the matrix can be calculated. Since the first residue in both sequences is A, $S_{1,1} = 2$, and by the assumptions stated at the beginning, w = 0. Thus, $M_{1,1}$ = MAX[$M_{0,0} + 2$, $M_{1, 0} -1$, $M_{0,1} -1$] = MAX [2, -1, -1] = 2.

A value of 2 is then placed in position 1,1 of the scoring matrix as in fig. 12. And subsequently the whole matrix is filled in the same way.

		A	C	A	C	A	C	T
	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0
G	0	1	1	1	1	1	1	0
C	0	0	3	2	3	2	3	2
A	0	2	2	5	4	5	4	3
C	0	1	4	4	7	6	7	6
A	0	2	3	6	6	9	8	7
C	0	1	4	5	8	8	11	10

Fig. 12. Final filled matrix

iii. Traceback

After the matrix fill step, the maximum alignment score for these two test sequences is 11. The traceback step determines the actual alignment(s) for the maximum score. It is not mandatory that the last cell has the maximum alignment score.

The traceback step begins with the position that leads to the maximal score. In this case, there is 11 in that location.

Trace back takes the current cell and looks to the neighbor cells that could be direct predecessors. This means it looks to the neighbor to the left (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor above it (gap in sequence #1) as in fig. 13. The algorithm for trace back chooses as the next cell in the sequence one of the possible predecessors. This continues till cell with value 0 is reached.

		A	C	A	C	A	C	T
	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0
G	0	1	1	1	1	1	1	0
C	0	0	3	2	3	2	3	2
A	0	2	2	5	4	5	4	3
C	0	1	4	4	7	6	7	6
A	0	2	3	6	6	9	8	7
C	0	1	4	5	8	8	11	10

Fig. 13. Traceback Step

The only possible predecessor is the diagonal match/mismatch neighbor. If more than one possible predecessor exists, any can be chosen. This gives us a current alignment of

(Seq #1)   C

             |

(Seq #2)   C

So now we look at the current cell and determine which cell is its direct predecessor. In this case, it is the cell with the red 9 as in fig. 14.

(Seq #1)   C A

             | |

(Seq #2)   C A

		A	C	A	C	A	C	T
	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0
G	0	1	1	1	1	1	1	0
C	0	0	3	2	3	2	3	2
A	0	2	2	5	4	5	4	3
C	0	1	4	4	7	6	7	6
A	0	2	3	6	6	9	8	7
C	0	1	4	5	8	8	11	10

Fig. 14. Traceback step with the correct arrows

Continuing with the traceback step, we eventually get a position in column 0 or row 0 which tells us that traceback is completed as in fig. 15.
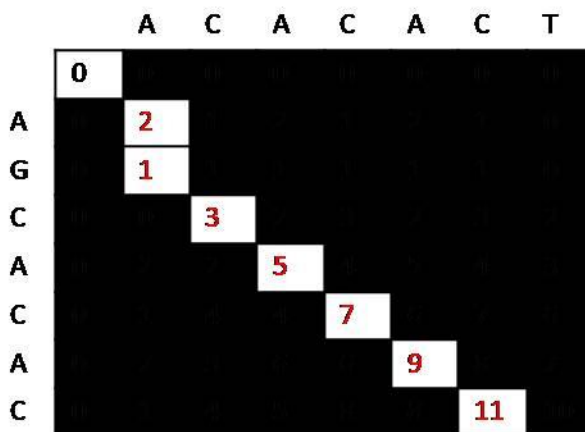
Fig. 15. Final Traceback Matrix

The possible maximum alignment is:

AG C A C A C

 | | | | | |

A _ C A C A C

There is a combination of these two methods which is called hybrid methods, also known as semiglobal or "glocal" methods. This method attempts to find the best possible alignment that includes the start and end of one or the other sequence. This can be especially useful when the downstream part of one sequence overlaps with the upstream part of the other sequence. In this case, neither global nor local alignment is entirely appropriate.

4. Significance of Sequence Alignment

Sequence alignments are useful in bioinformatics for identifying sequence similarity, producing phylogenetic trees, and developing homology models of protein structures. However, the biological relevance of sequence alignments is not always clear. Alignments are often assumed to reflect a degree of evolutionary change between sequences descended from a common ancestor; however, it is formally possible that convergent evolution can occur to produce apparent similarity between proteins that are evolutionarily unrelated but perform similar functions and have similar structures.

In database searches such as BLAST, statistical methods can determine the likelihood of a particular alignment between sequences or sequence regions arising by chance with the given the size and composition of the database being searched. These values can vary significantly depending on the search space. In particular, the likelihood of finding a given alignment by chance increases, if the database consists only of sequences from the same organism as the query sequence. Repetitive sequences in the database or query can also distort both the search results and the assessment of statistical significance. BLAST automatically filters such repetitive sequences in the query to avoid apparent hits that are statistical artifacts.

The choice of a **scoring function** that reflects biological or statistical observations about known sequences is important to producing good alignments. Protein sequences are frequently aligned using substitution matrices that reflect the probabilities of given character-to-character substitutions. A series of matrices called PAM matrices (Point

Accepted Mutation matrices, originally defined by Margaret Dayhoff and sometimes referred to as "Dayhoff matrices") explicitly encode evolutionary approximations regarding the rates and probabilities of particular amino acid mutations. Another common series of scoring matrices, known as BLOSUM (Blocks Substitution Matrix), encodes empirically derived substitution probabilities. Variants of both types of matrices are used to detect sequences with differing levels of divergence, thus allowing users of BLAST or FASTA to restrict searches to more closely related matches or expand to detect more divergent sequences. Gap penalties account for the introduction of a gap - on the evolutionary model, an insertion or deletion mutation - in both nucleotide and protein sequences, and therefore the penalty values should be proportional to the expected rate of such mutations. The quality of the alignments produced therefore depends on the quality of the scoring function.

5. Sequence Databases

The repositories for the genomic sequences are

National Center for Biotechnology Information (**NCBI**) is part of the United States National Library of Medicine (NLM), a branch of the National Institutes of Health. The NCBI is located in Bethesda, Maryland and was founded in 1988 through legislation sponsored by Senator Claude Pepper. The NCBI houses genome sequencing data in GenBank and an index of biomedical research articles in PubMed Central and PubMed, as well as other information relevant to biotechnology. All these databases are available online through the Entrez search engine. The NCBI is directed by David Lipman, one of the original authors of the BLAST sequence alignment program and a widely respected figure in Bioinformatics. The NCBI has had responsibility for making available the GenBank DNA sequence database since 1992 as shown in fig. 16. GenBank coordinates with individual laboratories and other sequence databases such as those of the European Molecular Biology Laboratory (EMBL) and the DNA Data Bank of Japan (DDBJ). Since 1992, NCBI has grown to provide other databases in addition to GenBank. NCBI provides Online Mendelian Inheritance in Man, the Molecular Modeling Database (3D protein structures), dbSNP a database of single-nucleotide polymorphisms, the Unique Human Gene Sequence Collection, a Gene Map of the human genome, a Taxonomy Browser, and coordinates with the National Cancer Institute to provide the Cancer Genome Anatomy Project.
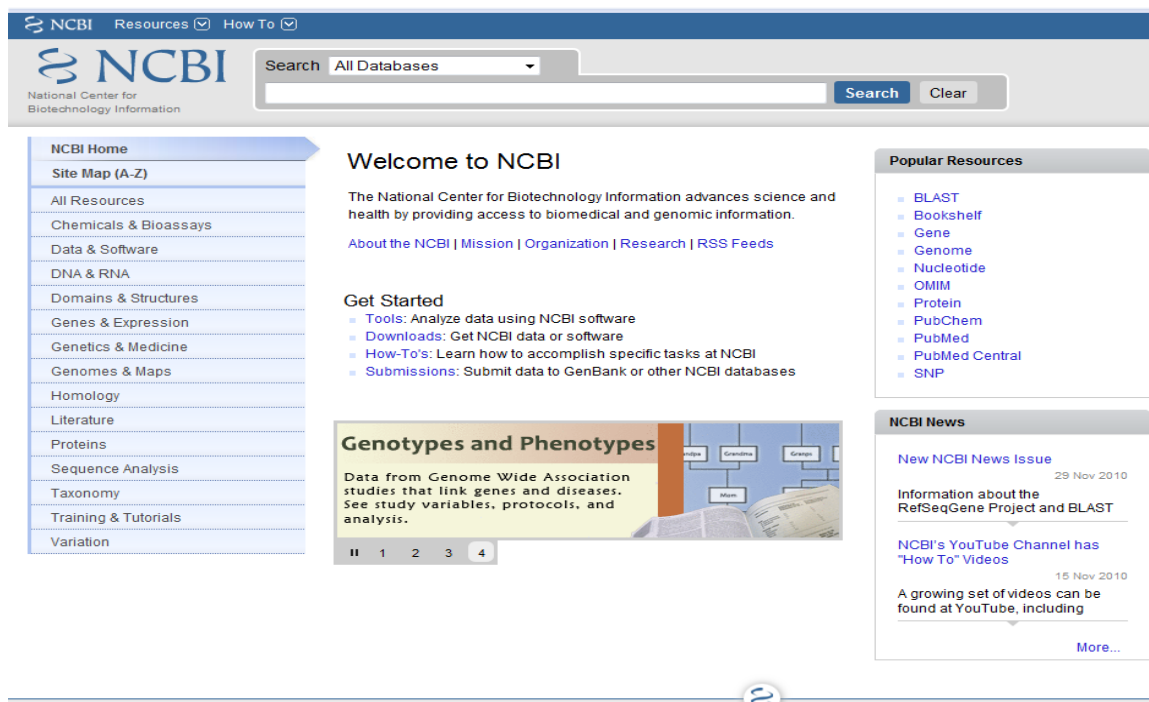
Fig. 16. NCBI portal

The NCBI assigns a unique identifier (Taxonomy ID number) to each species of organism. The NCBI has software tools that are available by WWW browsing or by FTP. For example, BLAST is a sequence similarity searching program. BLAST can do sequence comparisons against the GenBank DNA database in less than 15 seconds. The **NCBI Bookshelf** is a collection of freely available, downloadable, on-line versions of selected biomedical books. The Bookshelf has various titles covering aspects of molecular biology, biochemistry, cell biology, genetics, microbiology, a couple of disease states from a molecular and cellular point of view, research methods, and virology. Some of the books are online versions of previously published books, while others, such as Coffee Break (book), are written and edited by NCBI staff. The Bookshelf is a complement to the Entrez PubMed repository of peer-reviewed publication abstracts in that Bookshelf contents provide established perspectives on evolving areas of study and a context in which many disparate individual pieces of reported research can be organized.

European Molecular Biology Laboratory (**EMBL**) is a molecular biology research institution supported by 20 European countries and Australia as associate member state. The EMBL was created in 1974 and is a non-profit organisation funded by public research money from its member states. Research at EMBL is conducted by approximately 85 independent groups covering the spectrum of molecular biology. The Laboratory operates from five sites: the main Laboratory in Heidelberg, and Outstations in Hinxton (the European Bioinformatics Institute (**EBI**)), Grenoble, Hamburg, and Monterotondo near Rome as in fig. 17. Each of the sites has a research specific field. At EBI, the research is oriented towards computational biology and bioinformatics. At Grenoble and Hamburg the research is in the field of structural biology. At Monterotondo the research is focused mainly on mouse models for clinical research. At the headquarters in Heidelberg, there are big departments in Cell Biology and Gene Expression as well as smaller complementing the aforementioned research fields.

Fig. 17. EMBL portal

The cornerstones of EMBL's mission are: to perform basic research in molecular biology and molecular medicine, to train scientists, students and visitors at all levels, to offer vital services to scientists in the member states, to develop new instruments and methods in the life sciences, and to actively engage in technology transfer. EMBL's international PhD Programme has a student body of about 170. The Laboratory also sponsors an active Science and Society programme. Many scientific breakthroughs have been made at EMBL, most notably the first systematic genetic analysis of embryonic development in the fruit fly by Christiane Nüsslein-Volhard and Eric Wieschaus, for which they were awarded the Nobel Prize for Medicine in 1995.

DNA Data Bank of Japan (DDBJ) is a DNA data bank. It is located at the National Institute of Genetics (NIG) in the Shizuoka prefecture of Japan. It is also a member of the International Nucleotide Sequence Database Collaboration or INSDC. It exchanges its data with European Molecular Biology Laboratory at the European Bioinformatics Institute and with GenBank at the National Center for Biotechnology Information on a daily basis. Thus these three databanks contents the same data at any given time. DDBJ began data bank activities since 1986 at NIG and it boasts to be the only nucleotide sequence data bank in Asia. Although DDBJ mainly receives its data from Japanese researchers, however it can accept data from a contributor belonging to any other country as in fig. 18.
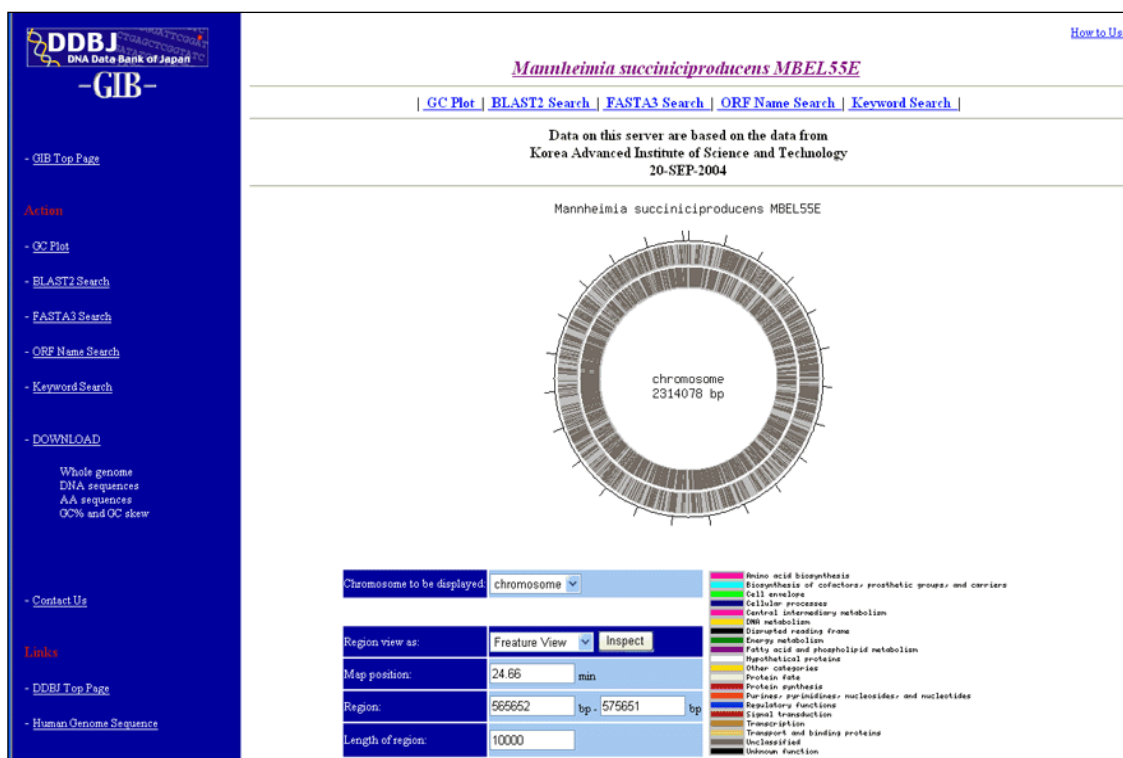
Fig. 18. DDBJ Portal

DDBJ is primarily funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT). DDBJ has an international advisory committee which consists of nine members, 3 members each from Europe, US, and Japan. This committee advice DDBJ about its maintenance, management and future plans once a year. Apart from this DDBJ also has an international collaborative committee which advises on various technical issues related to international collaboration and consists of working-level participants.

6. Softwares Used in Sequence Alignment

S. No.	Name	Function	Website Link
1	ALIGN	Sequence Analysis	http://www.ebi.ac.uk/Tools/emboss/align
2	CENSOR	Sequence Analysis	http://www.ebi.ac.uk/Tools/censor/
3	CLUSTALW2	Sequence Analysis	http://www.ebi.ac.uk/Tools/clustalw2/
4	CpG Plot/ CpGreport	Sequence Analysis	http://www.ebi.ac.uk/Tools/emboss/ cpgplot/
5	Genewise	Sequence Analysis	http://www.ebi.ac.uk/Tools/Wise2/
6	Kalign	Sequence Analysis	http://www.ebi.ac.uk/Tools/kalign
7	MAFFT	Sequence Analysis	http://www.ebi.ac.uk/Tools/mafft/
8	MUSCLE	Sequence Analysis	http://www.ebi.ac.uk/Tools/muscle/

9	Pepstats/ Pepwindow/Pepinfo	Sequence Analysis	http://www.ebi.ac.uk/Tools/emboss/ pepinfo/
10	PromoterWise	Sequence Analysis	http://www.ebi.ac.uk/Tools/Wise2/ promoterwise.html
11	SAPS	Sequence Analysis	http://www.ebi.ac.uk/Tools/saps/
12	T-coffee	Sequence Analysis	http://www.ebi.ac.uk/Tools/t-coffee/
13	Transeq	Sequence Analysis	http://www.ebi.ac.uk/Tools/emboss/transeq/
14	COBALT	Sequence Analysis	http://www.ncbi.nlm.nih.gov/tools/ cobalt/
15	Genome Workbench	Sequence Analysis	http://www.ncbi.nlm.nih.gov/projects/ gbench/
16	ORF Finder	Sequence Analysis	http://www.ncbi.nlm.nih.gov/gorf/gorf/ html
17	Primer - BLAST	Sequence Analysis	http://www.ncbi.nlm.nih.gov/tools/ primer-blast
18	ProSplign	Sequence Analysis	http://www.ncbi.nlm.nih.gov/sutils/static/pr osplin/prosplign.html
19	Splign	Sequence Analysis	http://www.ncbi.nlm.nih.gov/sutils/ splign/
20	VecScreen	Sequence Analysis	http://www.ncbi.nlm.nih.gov/VecScreen/Ve cScreen.html
21	Sequence Analysis	Sequence analysis	http://www.informagen.com/SA/
22	SeWeR	Sequence analysis	http://www.bioinformatics.org/sewer/
23	Motif Search	Sequence analysis	http://nbc11.biologie.uni-kl.de/framed/left/menu/auto/right/ motifsearch2/ index.pl
24	DNA Translator	Sequence analysis	http://nbc11.biologie.uni-kl.de/framed/left/menu/auto/right/JDT/
25	Non coding RNA Gene Finder	Sequence analysis	http://nbc11.biologie.uni-kl.de/framed/left/menu/auto/right/ ncRnaGeneFinder/index.pl
26	TransTerm	Sequence analysis	http://nbc11.biologie.uni-kl.de/framed/left/menu/auto/right/ transterm/

27	QRNA	Sequence analysis	http://nbc11.biologie.unikl.de/framed/left/menu/auto/right/qrna/
28	Clustalformatter 5	Sequence analysis	http://nbc11.biologie.uni-kl.de/framed/left/menu/auto/right/ClustalFormatter/
29	BioEdit	Sequence Alignment Editor	http://www.mbio.ncsu.edu/BioEdit/bioedit.html
30	FASTA	Sequence Similarity Search	http://www.ebi.ac.uk/Tools/fasta/
31	HMMER	Homology of protein	http://hmmer.janelia.org/
32	JAligner	Pairwise seq. alignment	http://jaligner.sourceforge.net/
33	JSTRING	Java Search for Tandem Repeats IN Genomes	http://bioinf.dms.med.uniroma1.it/JSTRING/
34	NCBI BLAST	Aligning Sequences	http://blast.ncbi.nlm.nih.gov/Blast.cgi
35	Gene Runner/ Motif Runner	Motif based sequence analysis	http://www.generunner.net/
36	GoCore	Protein Seq. Alignment & Analysis	http://www.helsinki.fi/project/ritvos/GoCore/
37	MAFFT	Multiple alignment	http://mafft.cbrc.jp/alignment/server/index.html
38	MAUVE	Multiple alignment	http://gel.ahabs.wisc.edu/mauve/
39	MEME Suite	Motif based sequence analysis	http://meme.nbcr.net/
40	CORAL (CDTree)	Aligning Core Conserved Regions	http://www.ncbi.nlm.nih.gov/Structure/cdtree/cdtree.shtml
41	BlastAlign	Align N Seq. with large INDELs	http://www.bioafrica.net/blast/BlastAlign.html
42	ARB software	Sequence DB Handling and Data Analysis	http://www.arb-home.de/

43	Automated Codon Usage Analysis Software - ACUA	Nucleotide Analysis	http://www.bioinsilico.com/acua
44	AnnHyb	Nucleotide Analysis	http://www.bioinformatics.org/annhyb/
45	SOAP2	Short read Alignment	http://soap.genomics.org.cn/
46	ACT (Artemis Comparison Tool)	DNA Sequence Comparison	http://www.sanger.ac.uk/resources/software/act/
47	WU-BLAST	Multiple Sequence Alignment	www.ebi.ac.uk/Tools/blast2/
48	CLUSTALW2	multiple sequence alignment	http://www.ebi.ac.uk/Tools/clustalw2/

References

www.wikipedia.org/

cnx.org/content/m11026/latest/

www.ncbi.nlm.nih.gov/

www.ebi.ac.uk/embl/

www.ddbj.nig.ac.jp/

Genome Annotation

Sanjeev Kumar, D. C. Mishra, Sneha Murmu and Jyotika Bhati

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Until the genome revolution, genes were identified by researchers with specific interests in a particular protein or cellular process. Once identified, these genes were isolated, typically by cloning and sequencing cDNAs, usually followed by targeted sequencing of the longer genomics segments that code for the cDNAs. Once an organism's entire genome sequence becomes available, there is strong motivation for finding all the genes encoded by a genome at once rather than in a piecemeal approach. Such catalogue is immensely valuable to researchers, as they can learn much more from the whole picture than from a much more limited set of genes. For example, genes of similar sequence can be identified, evolutionary and functional relationships can be elucidated, and a global picture of how many and what types of genes are present in a genome can be seen. A significant portion of the effort in genome sequencing is devoted to the process of *annotation*, in which genes, regulatory elements, and other features of the sequence are identifies as thoroughly as possible and catalogued in a standard format in public databases so that researchers can easily use the information. Functional genomics research has expanded enormously in the last decade and particularly the plant biology research community. Functional annotation of novel DNA sequences is probably one of the top requirements in functional genomics as this holds, to a great extent, the key to the biological interpretation of experimental results.

Computational Gene Prediction

Computational gene prediction is becoming more and more essential for the automatic analysis and annotation of large uncharacterized genomic sequences. In the past two decades, many algorithms have been evolved to predict protein coding regions of the DNA sequences. They all have in common, to varying degree, the ability to differentiate between gene features like Exons, Introns, Splicing sites, Regulatory sites etc. Gene prediction methods predicts coding region in the query sequences and then annotates the sequences databases.

Gene Structure and Expression

The gene structure and the gene expression mechanism in eukaryotes are far more complicated than in prokaryotes. In typical eukaryotes, the region of the DNA coding for a protein is usually not continuous. This region is composed of alternating stretches of *exons* and *introns*. During transcription, both exons and introns are transcribed onto the RNA, in their linear order. Thereafter, a process called *splicing* takes place, in which, the intron
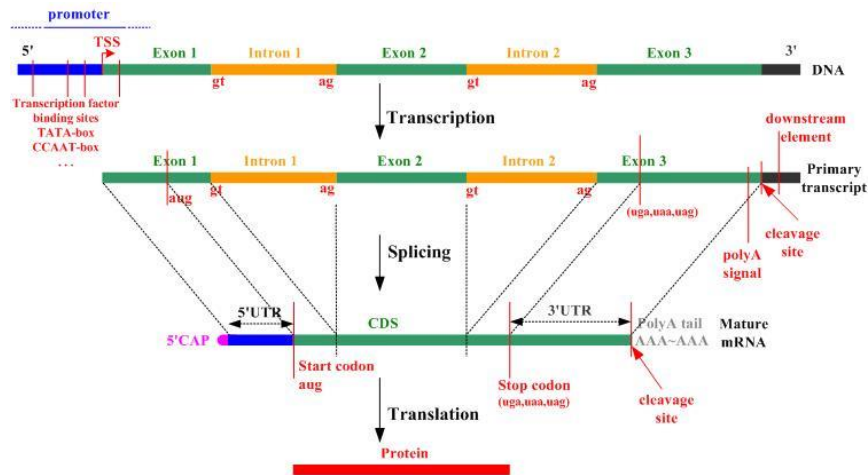
Fig. 1: Representative Diagram of Protein Coding Eukaryotic Gene

sequences are excised and discarded from the RNA sequence. The remaining RNA segments, the ones corresponding to the exons are ligated to form the mature RNA strand. A typical multi-exon gene has the following structure (as illustrated in Fig. 1). It starts with the promoter region, which is followed by a transcribed but non-coding region called *5' untranslated region (5' UTR)*. Then follows the initial exon which contains the start codon. Following the initial exon, there is an alternating series of introns and internal exons, followed by the terminating exon, which contains the stop codon. It is followed by another non-coding region called the *3' UTR*. Ending the eukaryotic gene, there is a polyadenylation (polyA) signal: the nucleotide Adenine repeating several times. The exon-intron boundaries (i.e., the splice sites) are signalled by specific short (2bp long) sequences. The 5'(3') end of an intron (exon) is called the *donor* site, and the 3'(5') end of an intron (exon) is called the *acceptor* site. The problem of gene identification is complicated in the case of eukaryotes by the vast variation that is found in gene structure.

Gene Prediction Methods

There are mainly two classes of methods for computational gene prediction (Fig. 2). One is based on sequence similarity searches, while the other is gene structure and signal-based searches, which is also referred to as Ab initio gene finding.

Sequence Similarity Searches

Sequence similarity search is a conceptually simple approach that is based on finding similarity in gene sequences between ESTs (expressed sequence tags), proteins, or other genomes to the input genome. This approach is based on the assumption that functional regions (exons) are more conserved evolutionarily than non-functional regions (intergenic or intronic regions). Once there is similarity between a certain genomic region and an EST, DNA, or protein, the similarity information can be used to infer gene structure or function of that region. EST-based sequence similarity usually has drawbacks in that ESTs only correspond to small portions of the gene sequence, which means that it is often difficult to predict the complete gene structure of a given region. Local alignment and global alignment are two methods based on similarity searches. The most common local alignment tool is the BLAST family of programs, which detects sequence similarity to known genes, proteins, or ESTs. The biggest limitation to this

type of approaches is that only about half of the genes being discovered have significant homology to genes in the databases.

Ab initio Gene Prediction Methods

The second class of methods for the computational identification of genes is to use gene structure as a template to detect genes, which is also called *ab initio* prediction. *Ab initio* gene predictions rely on two types of sequence information: signal sensors and content sensors. Signal sensors refer to short sequence motifs, such as splice sites, branch points, poly pyrimidine tracts, start codons and stop codons. Exon detection must rely on the content sensors, which refer to the patterns of codon usage that are unique to a species, and allow coding sequences to be distinguished from the surrounding non-coding sequences by statistical detection algorithms.

Many algorithms are applied for modeling gene structure, such as Dynamic Programming,
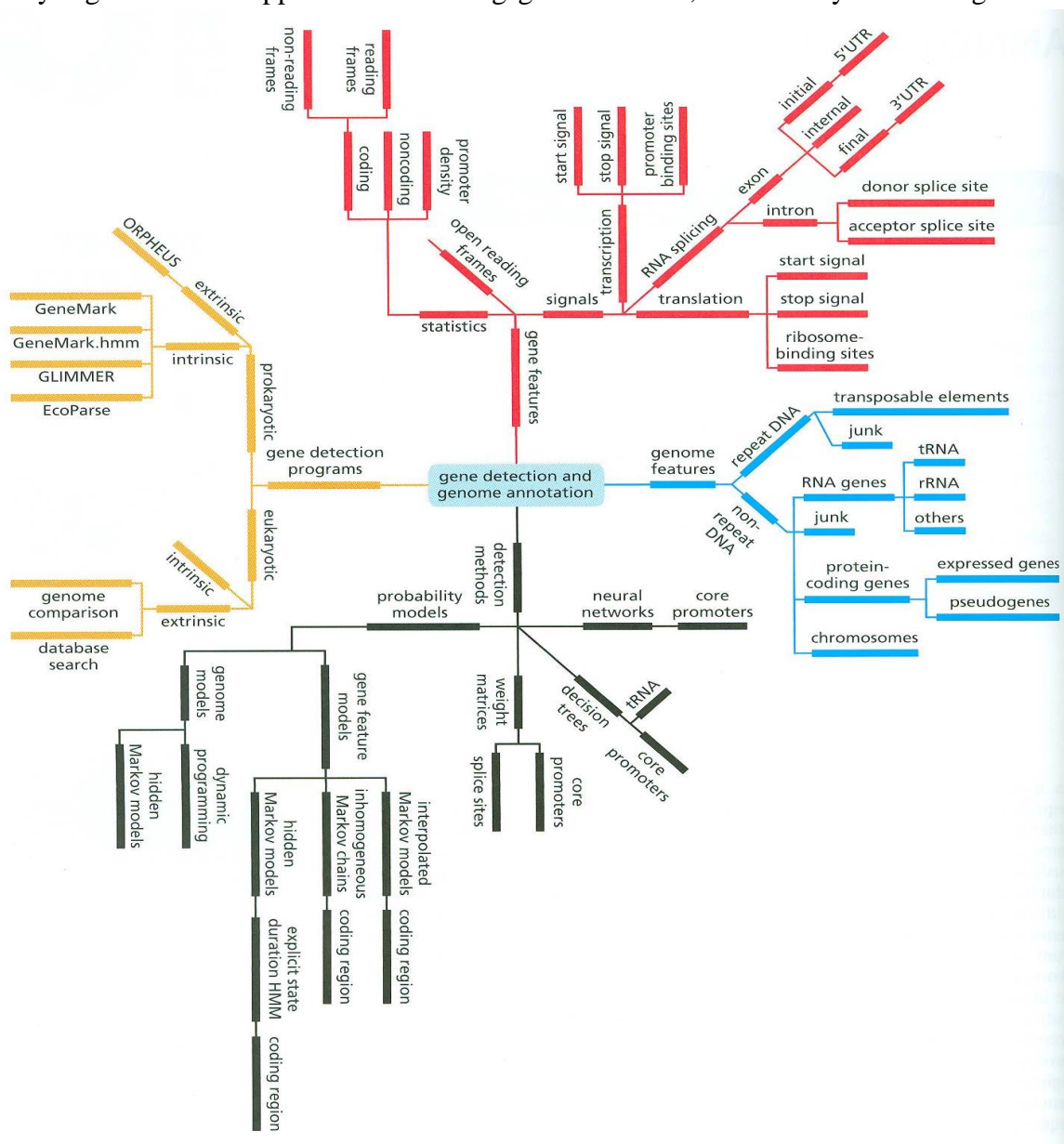
linear discriminant analysis, Linguist methods, Hidden Markov Model and Neural Network.

Based on these models, a great number of *ab initio* gene prediction programs have been developed.

Fig. 2: Diagrammatic Representation of Gene Prediction and Annotation

Gene Discovery in Prokaryotic Genomes

Discovery of genes in Prokaryote is relatively easy, due to the higher gene density typical of prokaryotes and the absence of introns in their protein coding regions. DNA sequences that encode proteins are transcribed into mRNA, and the mRNA is usually translated into proteins without significant modification. The longest ORFs (open reading frames) running from the first available start codon on the mRNA to the next stop codon in the same reading frame generally provide a good, but not assured prediction of the protein coding regions. Several methods have been devised that use different types of Markov models in order to capture the compositional differences among coding regions, "shadow" coding regions (coding on the opposite DNA strand), and noncoding DNA. Such methods, including ECOPARSE, the widely used GENMARK, and Glimmer program, appear to be able to identify most protein coding
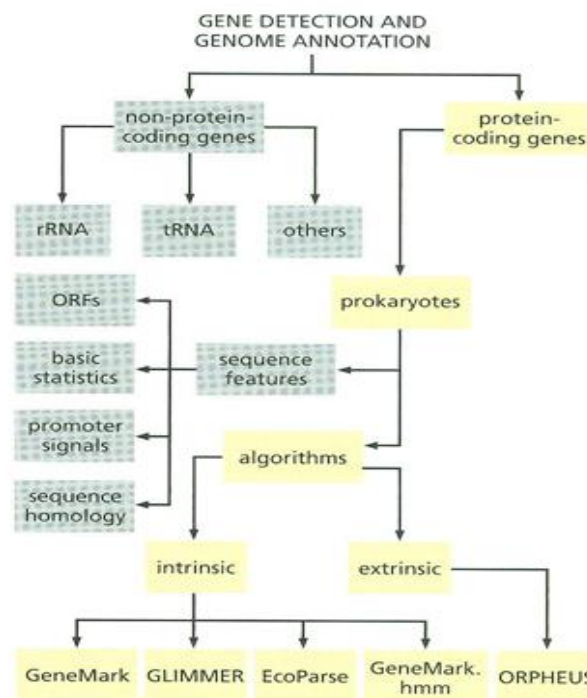
genes with good performance (Fig. 3).

Fig. 3: Flow Diagram of Prokaryotic Gene Discovery

Gene Discovery in Eukaryotic Genome

It is a quite different problem from that encountered in prokaryotes. Transcription of protein coding regions initiated at specific promoter sequences is followed by removal of noncoding sequences (introns) from pre-mRNA by a splicing mechanism, leaving the protein encoding exons. Once the introns have been removed and certain other modifications to the mature RNA have been made, the resulting mature mRNA can be translated in the 5` to 3` direction, usually from the first start codon to the first stop codon. As a result of the presence of intron sequences

in the genomic DNA sequences of eukaryotes, the ORF corresponding to an encoded gene will be interrupted by the presence of introns that usually generate stop codons (Fig.4).
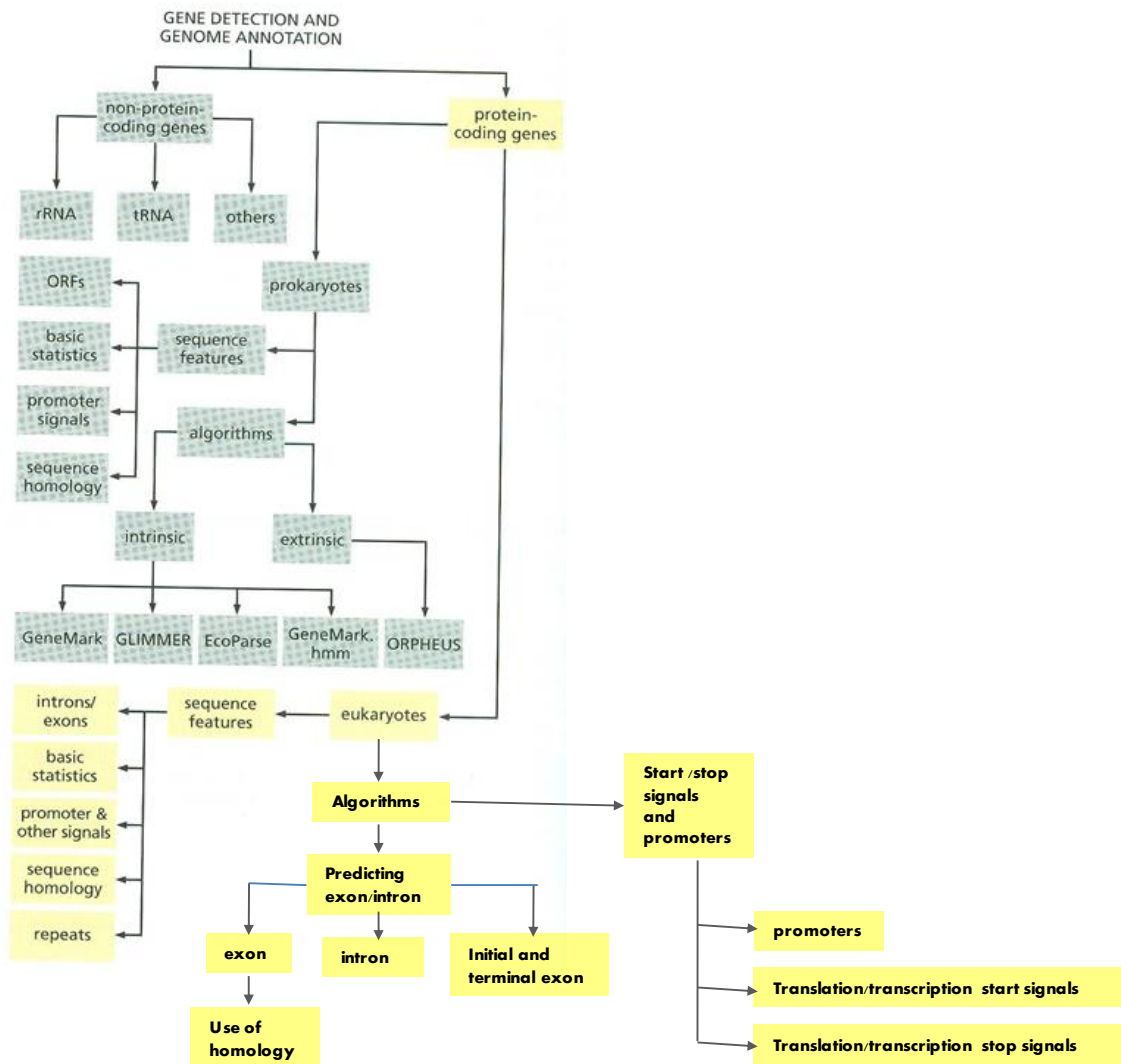
Fig. 4: Flow Diagram of Eukaryotic Gene Discovery

Gene Prediction Program

There are two basic problems in gene prediction: prediction of protein coding regions and prediction of the functional sites of genes. Gene prediction program can be classified into four generations. The first generation of programs was designed to identify approximate locations of coding regions in genomic DNA. The most widely known programs were probably TestCode and GRAIL. But they could not accurately predict precise exon locations. The second generation, such as SORFIND and Xpound, combined splice signal and coding region identification to predict potential exons, but did not attempt to assemble predicted exons into complete genes. The next generation of programs attempted the more difficult task of predicting complete gene structures. A variety of programs have been developed, including GeneID, GeneParser, GenLang, and FGENEH. However, the performance of those programs remained rather poor. Moreover, those programs were all based on the assumption that the input sequence contains exactly one complete gene, which is not often the case. To solve this

problem and improve accuracy and applicability further, GENSCAN and AUGUSTUS were developed, which could be classified into the fourth generation.

GeneMark

GeneMark uses a Markov Chain model to represent the statistics of the coding and noncoding frames. The method uses the dicodon statistics to identify coding regions. Consider the analysis of a sequence x whose base at the ith position is called x_i. The Markov chains used are fifth order, and consist of a terms such as $P(a/x_1x_2x_3x_4x_5)$, which represent the probability of the sixth base of the sequence x being given a given that the previous five bases in the sequence x where $x_1x_2x_3x_4x_5$, resulting in the first dicodon of the sequence being $x_1x_2x_3x_4x_5a$. These terms must be defined for all possible pentamers with the general sequence $b_1b_2b_3b_4b_5$. The values of these terms can be obtained of analysis of data, consisting of nucleotide sequence in which the coding regions have been actually identified. When there are sufficient data, they are given by

$$P(\frac{a}{b_1b_2b_3b_4b_5}) = \frac{n_{b_1b_2b_3b_4b_5a}}{\sum_{a=A,C,G,T} n_{b_1b_2b_3b_4b_5a}}$$

where, $n_{b_1b_2b_3b_4b_5a}$ is the number of times the sequence $b_1b_2b_3b_4b_5a$ occurs in the training data. This is the maximum likelihood estimators of the probability from the training data.

Glimmer

The core of Glimmer is Interpolated Markov Model (IMM), which can be described as a generalized Markov chain with variable order. After GeneMark introduces the fixed-order Markov chains, Glimmer attempts to find a better approach for modeling the genome content. The motivational fact is that the bigger the order of the Markov chain, the more non-randomness can be described. However, as we move to higher order models, the number of probabilities that we must estimate from the data increases exponentially. The major limitation of the fixed-order Markov chain is that models from higher order require exponentially more training data, which are limited and usually not available for new sequences. However, there are some oligomers from higher order that occur often enough to be extremely useful predictors. For the purpose of using these higher-order statistics, whenever sufficient data is available, Glimmer IMMs.

Glimmer calculates the probabilities for all Markov chains from 0th order to 8th. If there are longer sequences (e.g. 8-mers) occurring frequently, IMM makes use of them even when there is insufficient data to train an 8-th order model. Similarly, when the statistics from the 8-th order model do not provide significant information, Glimmer refers to the lower-order models to predict genes.

Opposed to the supervised GeneMark, Glimmer uses the input sequence for training. The ORFs longer than a certain threshold are detected and used for training, because there is high probability that they are genes in prokaryotes. Another training option is to use the sequences with homology to known genes from other organisms, available in public databases. Moreover, the user can decide whether to use long ORFs for training purposes or choose any set of genes to train and build the IMM.

GeneMark.hmm

GeneMark.hmm is designed to improve GeneMark in finding exact gene starts. Therefore, the properties of GeneMark.hmm are complementary to GeneMark. GeneMark.hmm uses GeneMark models of coding and non-coding regions and incorporates them into hidden Markov model framework. In short terms, Hidden Markov Models (HMM) are used to describe the transitions from non-coding to coding regions and vice versa. GeneMark.hmm predicts the most likely structure of the genome using the Viterbi algorithm, a dynamic programming algorithm for finding the most likely sequence of hidden states. To further improve the prediction of translation start position, GeneMark.hmm derives a model of the ribosome binding site (6-7 nucleotides preceding the start codon, which are bound by the ribosome when initiating protein translation). This model is used for refinement of the results.

Both GeneMark and GeneMark.hmm detect prokaryotic genes in terms of identifying open reading frames that contain real genes. Moreover, they both use pre-computed species-specific gene models as training data, in order to determine the parameters of the protein-coding and non-coding regions.

Orpheus

The ORPHEUS program uses homology, codon statistics and ribosome binding sites to improve the methods presented so far by using information that those programs ignored. One of the key differences is that it uses database searches to help determine putative genes, and is thus an extrinsic method. This initial set of genes is used to define the coding statistics for the organism, in this case working at the level of codon, not dicodons. These statistics are then used to define a larger set of candidate ORFs. From this set, those ORFs with an unambiguous start codon end are used to define a scoring matrix for the ribosome-binding site, which is then used to determine the 5` end of those ORFs where alternative start are present.

EcoParse

EcoParse is one of the first HMM model based gene finder, was developed for gene finding in *E.coli*. It focuses on the uses the codon structure of genes. With EcoParse a flora of HMM based gene finder, usuing dynamic programming and the viterbi algorithm to parse a sequence, emerged.

Evaluation of Gene Prediction Programs

In the field of gene prediction accuracy can be measured at three levels

a.   Coding nucleotides (base level)

b.   Exon structure (exon level)

c.   Protein product (protein level)

At base level gene predictions can be evaluated in terms of *true positives (TP)* (predicted features that are real), *true negatives* (TN) (non-predicted features that are not real), *false positives (FP)* (predicted features that are not real), and *false negatives (FN)* (real features that were not predicted) Fig. 5. Usually the base assignment is to be in a coding or non coding segment, but this analysis can be extended to include non coding parts of genes, or any functional parts of the sequences.

Fig. 5: Four Possible Comparisons of Real and Predicted Genes

Sensitivity (Sn): The fraction of bases in real genes that are correctly predicted to be in genes is the sensitivity and interpreted as the probability of correctly predicting a nucleotide to be in a given gene that it actually is.

$$Sn = \frac{TP}{TP + FN}$$

Specificity (Sp): The fraction of those bases which are predicted to be in genes that actually are is called the specificity and interpreted as the probability of a nucleotide actually being in a gene given that it has been predicted to be.

$$Sp = \frac{TP}{TP + FP}$$

Care has to be taken in using these two values to assess a gene prediction program because, as with the normal definition of specificity, extreme results can make them misleading.

Approximate correlation coefficient (AC) has been proposed as a single measure to circumvent these difficulties. This defined as AC=2(ACP-0.5), where

$$ACP = \frac{1}{n}\left(\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN}\right),$$

At the exon level, determination of prediction accuracy depends on the exact prediction of exon start and end points. There are two measures of sensitivity and specificity used in the field, each of which measures a different but useful property.

The sensitivity measures used are

$S_{n1} = CE/AE$ and $Sn2 = ME/AE$

The specificity measures used are

$S_{p1} = CE/PE$ and $S_{p2} = WE/PE$

Where,

AE = No of actual exons in the data

PE = No of predicted exons in the data

CE = No of correct predicted exons

ME = No of missing exons (rarely occurs)

WE = No of wrongly predicted exons (Figure-5)
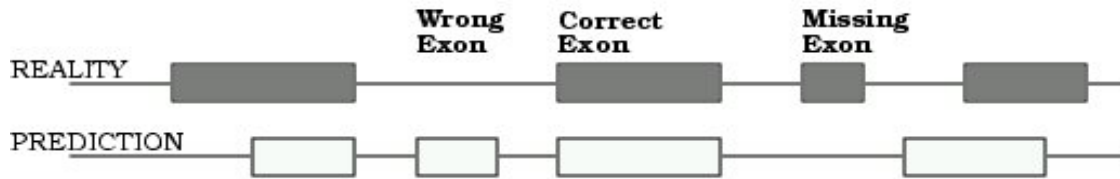
Fig. 6: Real and Predicted Exons

Gene Ontology

The gene ontology (GO, http:www.geneontology.org) is probably the most extensive scheme today for the description of gene product functions but other systems such as enzyme codes, KEGG pathways, FunCat, or COG are also widely used. Here, we describe the Blast2GO (B2G, www.blast2go.org) application for the functional annotation, management, and data mining of novel sequence data through the use of common controlled vocabulary schemas. The main application domain of the tool is the functional genomics of nonmodel organisms and it is primarily intended to support research in experimental labs. Blast2GO strives to be the application of choice for the annotation of novel sequences in functional genomics projects where thousands of fragments need to be characterized. Functional annotation in Blast2GO is based on homology transfer. Within this framework, the actual annotation procedure is configurable and permits the design of different annotation strategies. Blast2GO annotation parameters include the choice of search database, the strength and number of blast results, the extension of the query-hit match, the quality of the transferred annotations, and the inclusion of motif annotation. Vocabularies supported by B2G are gene ontology terms, enzyme codes (EC), InterPro IDs, and KEGG pathways.

Fig.7 shows the basic components of the Blast2GO suite. Functional assignments proceed through an elaborate annotation procedure that comprises a central strategy plus refinement functions. Next, visualization and data mining engines permit exploiting the annotation results to gain functional knowledge. GO annotations are generated through a 3-step process: blast, mapping, annotation. InterPro terms are obtained from InterProScan at EBI, converted and merged to GOs. GO annotation can be modulated from Annex, GOSlim web services and manual editing. EC and KEGG annotations are generated from GO. Visual tools include sequence color code, KEGG pathways, and GO graphs with node highlighting and filtering options. Additional annotation data-mining tools include statistical charts and gene set enrichment analysis functions.
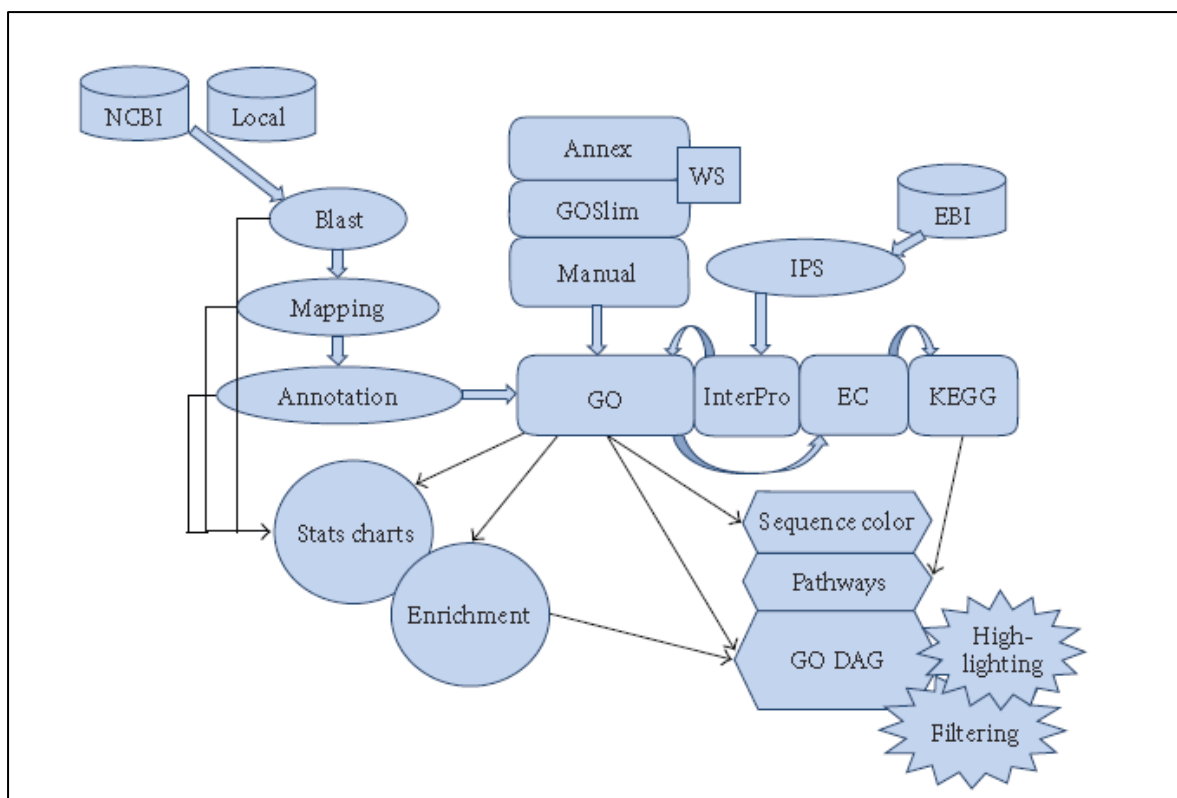
Fig. 7: Schematic Representation of Blast2GO Application.

The Blast2GO annotation procedure consists of three main steps: blast to find homologous sequences, mapping to collect GO terms associated to blast hits, and annotation to assign trustworthy information to query sequences.

Blast Step

The first step in B2G is to find sequences similar to a query set by blast. B2G accepts nucleotide and protein sequences in FASTA format and supports the four basic blast programs (blastx, blastp, blastn, and tblastx). Homology searches can be launched against public databases such as (the) NCBI nr using a query-friendly version of blast (QBlast). This is the default option and in this case, no additional installations are needed. Alternatively, blast can be run locally against a proprietary FASTA-formatted database, which requires a working www-blast installation. The Make Filtered Blast-GO-BD function in the Tools menu allows the creation of customized databases containing only GO annotated entries, which can be used in combination with the local blast option. Other configurable parameters at the blast step are the expectation value (e-value) threshold, the number of retrieved hits, and the minimal alignment length (hsp length) which permits the exclusion of hits with short, low e-value matches from the sources of functional terms. Annotation, however, will ultimately be based on sequence similarity levels as similarity percentages are independent of database size and more intuitive than e-values. Blast2GO parses blast results and presents the information for each sequence in table format. Query sequence descriptions are obtained by applying a language processing algorithm to hit descriptions, which extracts informative names and avoids low content terms such as "hypothetical protein" or "expressed protein".

Mapping Step

Mapping is the process of retrieving GO terms associated to the hits obtained after a blast search. B2G performs three different mappings as follows.

a. Blast result accessions are used to retrieve gene names (symbols) making use of two mapping files provided by NCBI (geneinfo, gene2accession). Identified gene names are searched in the species-specific entries of the gene product table of the GO database.

b. Blast result GI identifiers are used to retrieve UniProt IDs making use of a mapping file from PIR (Non-redundant Reference Protein database) including PSD, UniProt, Swiss-Prot, TrEMBL, RefSeq, GenPept, and PDB.

c. Blast result accessions are searched directly in the DBXRef Table of the GO database.

Annotation Step

This is the process of assigning functional terms to query sequences from the pool of GO terms gathered in the mapping step. Function assignment is based on the gene ontology vocabulary. Mapping from GO terms to enzyme codes permits the subsequent recovery of enzyme codes and KEGG pathway annotations. The B2G annotation algorithm takes into consideration the similarity between query and hit sequences, the quality of the source of GO assignments, and the structure of the GO DAG. For each query sequence and each candidate GO term, an annotation score (AS) is computed (see Figure 8). The AS is composed of two terms. The first, direct term (DT), represents the highest similarity value among the hit sequences bearing this GO term, weighted by a factor corresponding to its evidence code (EC). A GO term EC is present for every annotation in the GO database to indicate the procedure of functional assignment.

$$DT = \max(similarity \times EC_{weight})$$
$$AT = (\#GO - 1) \times GO_{weight}$$
$$AR : lowest.node(AS(DT + AT) \geq threshold)$$

Fig. 8: Blast2GO Annotation Rule

ECs vary from experimental evidence, such as inferred by direct assay (IDA) to unsupervised assignments such as inferred by electronic annotation (IEA). The second term (AT) of the annotation rule introduces the possibility of abstraction into the annotation algorithm. Abstraction is defined as the annotation to a parent node when several child nodes are present in the GO candidate pool. This term multiplies the number of total GOs unified at the node by a user defined factor or GO weight (GOw) that controls the possibility and strength of abstraction. When all ECw's are set to 1 (no EC control) and the GOw is set to 0 (no abstraction is possible), the annotation score of a given GO term equals the highest similarity value among the blast hits annotated with that term. If the ECw is smaller than one, the DT decreases and higher query-hit similarities are required to surpass the annotation threshold. If the GOw is not equal to zero, the AT becomes contributing and the annotation of a parent node is possible if multiple child nodes coexist that do not reach the annotation cutoff. Default values of B2G annotation parameters were chosen to optimize the ratio between annotation coverage and annotation accuracy. Finally, the AR selects the lowest terms per branch that exceed a user-defined threshold.

Blast2GO includes different functionalities to complete and modify the annotations obtained through the above-defined procedure. Enzyme codes and KEGG pathway annotations are generated from the direct mapping of GO terms to their enzyme code equivalents. Additionally, Blast2GO offers InterPro searches directly from the B2G interface. B2G launches sequence queries in batch, and recovers, parses, and uploads InterPro results. Furthermore, InterPro IDs can be mapped to GO terms and merged with blast-derived GO annotations to provide one integrated annotation result. In this process, B2G ensures that only the lowest term per branch remains in the final annotation set, removing possible parent-child relationships originating from the merging action.

References

1. Conesa, S. Gotz, J. M. Garcia-Gomez, J. Terol, M. Talon, and M. Robles, "Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research," Bioinformatics, vol. 21, no. 18, pp. 3674–3676, 2005.

2. Conesa and S. Gotz, "Blast2GO: A Comprehensive Suite for Functional Analysis in Plant Genomics," International Journal of Plant Genomics, vol. 2008, 2008.

3. H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa, "KEGG: Kyoto Encyclopedia of Genes and Genomes," Nucleic Acids Research, vol. 27, no. 1, pp. 29–34, 1999.

4. J.D. Watson, R.M. Myers, A.A. Caudy and J.A. Witkowski, "Recombinant DNA: Genes and Genomes - A Short Course," 3rd Ed., 2007.

5. M. Ashburner, C. A. Ball, J. A. Blake, et al., "Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium," Nature Genetics, vol. 25, no. 1, pp. 25–29, 2000.

6. Ruepp, A. Zollner, D. Maier, et al., "The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes," Nucleic Acids Research, vol. 32, no. 18, pp. 5539–5545, 2004.

7. R. L. Tatusov, N. D. Fedorova, J. D. Jackson, et al., "The COG database: an updated version includes eukaryotes," BMC Bioinformatics, vol. 4, p. 41, 2003.

8. Schomburg, A. Chang, C. Ebeling, et al., "BRENDA, the enzyme database: updates and major new developments," Nucleic Acids Research, vol. 32, Database issue, pp. D431–D433, 2004.

9. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," Journal of Molecular Biology, vol. 215, no. 3, pp. 403–410, 1990.

10. S. Myhre, H. Tveit, T. Mollestad, and A. Lægreid, "Additional Gene Ontology structure for improved biological reasoning," Bioinformatics, vol. 22, no. 16, pp. 2020–2027, 2006.

Hands-on Session for Genome Annotation

Sneha Murmu

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Genome annotation is the process of identifying functional elements within a genome, such as genes, regulatory regions, and repeat elements. The goal of genome annotation is to create an accurate and comprehensive description of the genome's structure and function. This can be a time-consuming process, but it is essential for understanding how genes and other functional elements work together to control an organism's biology.

One powerful tool for genome annotation is Blast2GO (Conesa et al., 2005). Blast2GO is a commercial bioinformatics software suite that provides comprehensive functional annotation of nucleotide and protein sequences. It combines powerful sequence similarity search algorithms, such as BLAST (Altschul et al., 1997) and HMMER (Finn et al., 2011), with functional annotation tools, such as InterProScan (Zdobnov et al., 2001) and Gene Ontology (GO) mapping, to provide a detailed functional analysis of genomic and transcriptomic data.

Blast2GO works by first performing a sequence similarity search, typically using BLAST, to identify sequences with homology to known sequences in public databases. The resulting hits are then annotated using a variety of functional annotation tools, including InterProScan, which identifies conserved protein domains and functional motifs, and GO mapping, which assigns GO terms based on the functional categories of annotated genes.

Blast2GO also includes tools for statistical analysis and data visualization, allowing users to explore functional trends and patterns in their data. It can be used to analyze a wide range of genomic and transcriptomic data sets. One of the strengths of Blast2GO is its user-friendly interface, which allows even non-experts to perform complex functional annotation analyses. Blast2GO is also highly customizable, allowing users to tailor the annotation process to their specific needs and research questions.

Here are the four broad steps involved in genome annotation using Blast2GO:

- ↓ Sequence quality control and assembly: Before annotating a genome, it is important to ensure that the quality of the sequencing data is high and that the genome has been properly assembled. This may involve trimming low-quality sequences, filtering out contaminants, and performing de novo assembly or mapping to a reference genome.

- ↓ Sequence similarity search: The first step in genome annotation is to identify sequences with homology to known sequences in public databases. This is typically done using BLAST or a similar tool. The resulting hits can provide clues about the function and evolutionary relationships of the sequences in question.

- ↓ Functional annotation: Once sequences have been identified using a sequence similarity search, functional annotation tools can be used to identify functional domains and motifs, assign Gene Ontology terms, and perform other types of functional analysis. Blast2GO includes a number of annotation tools, including InterProScan, which searches for conserved domains and motifs in protein sequences, and GO mapping, which assigns Gene Ontology terms based on the functional categories of annotated genes.

- ↓ Data analysis and visualization: Once the sequences have been annotated with functional information, the data can be analyzed and visualized in a variety of ways. Blast2GO includes tools for statistical analysis and data visualization. The results of the analysis can be exported in a variety of formats for further analysis.

Installation of Blast2GO:

Following are the general steps to install Blast2GO:

1. System requirements: Check that your computer meets the system requirements for Blast2GO. Blast2GO is compatible with Windows, macOS, and Linux operating systems, and requires at least 8 GB of RAM.

2. Download Blast2GO: Visit the Blast2GO website (https://www.blast2go.com/) and download the appropriate installation file for your operating system. You may need to create an account and purchase a license, depending on your intended use of the software.

3. Install Blast2GO: Double-click the downloaded installation file and follow the on-screen instructions to install Blast2GO (as depicted in Figure 1). You may need to provide administrator permissions, depending on your operating system and security settings.

4. Configure Blast2GO: Once Blast2GO is installed, you will need to configure it to work with your specific computing environment. This may include setting preferences for sequence databases, annotation tools, and other settings.

5. Activate license: If you have purchased a license for Blast2GO, you will need to activate it before you can use the software. This typically involves entering a license key or activating the license through an online portal.

Once Blast2GO is installed and configured, you can begin using it to analyze and annotate your genomic or transcriptomic data.
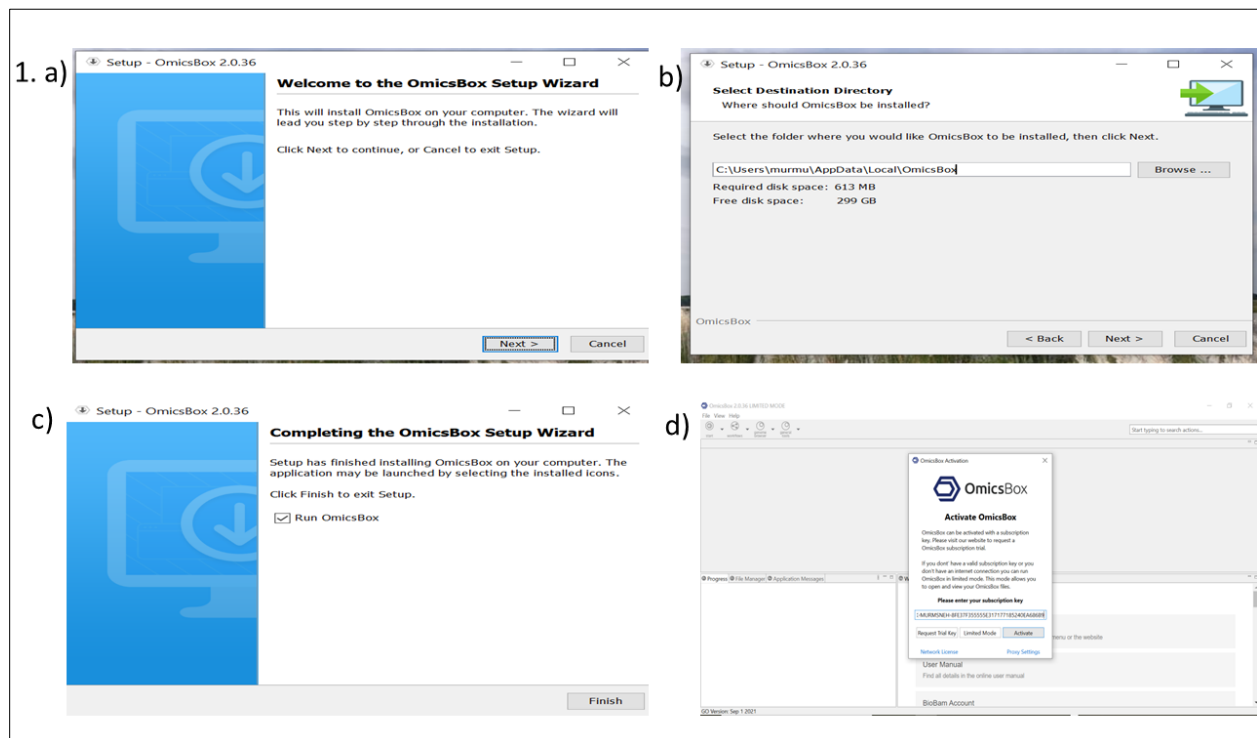
Figure 1: Installation steps of Blast2GO in Windows system.

Stepwise guide to perform annotation using Blast2GO

1. Open Blast2GO: Launch Blast2GO on your computer.

2. Load sequences: Load your sequence file(s) into Blast2GO. This can be done by clicking on "Load data" in the main menu and selecting the appropriate file type (e.g., FASTA).

3. Run **BLAST** search: In the main menu, click on "Run BLAST" and select the appropriate database for your search (e.g., NCBI non-redundant protein database) as shown in Figure 2. You can choose to run a BLASTP (protein query against protein database) or a BLASTX

(nucleotide query against protein database) search. You can also set various search parameters, such as the e-value threshold and the maximum number of hits to return.

4. View BLAST results: Once the BLAST search is complete, you can view the results in the BLAST results table (as shown in Figure 3). The table will show the sequence ID, the best hit, the e-value, the bit score, and other relevant information. You can sort the table by various columns to help you identify the best hits.

5. Import BLAST results: To import the BLAST results into the Blast2GO annotation pipeline, select the sequences you want to annotate and click on "Import selected hits". This will import the BLAST results and link them to the appropriate sequences in the annotation pipeline.
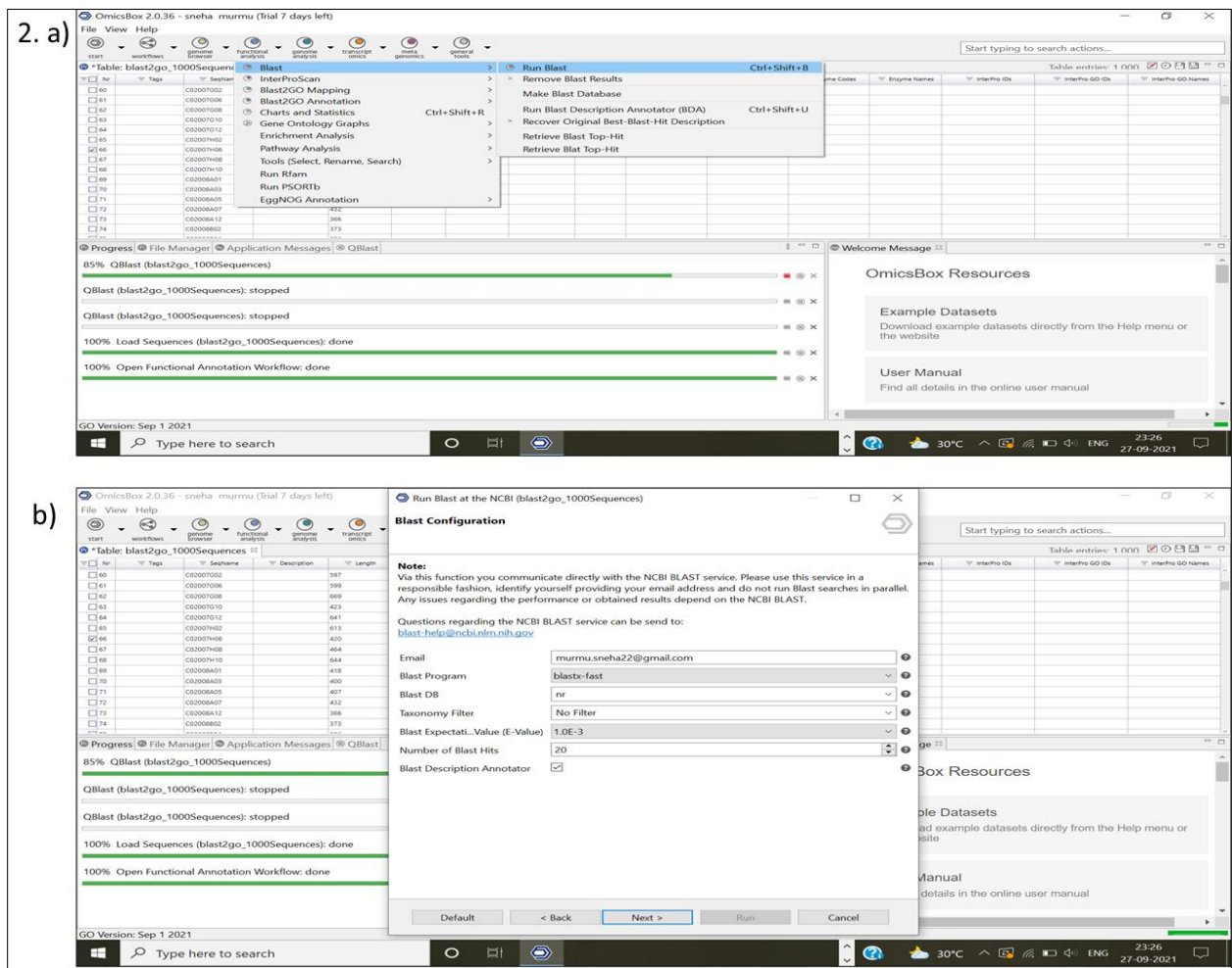
Figure 2: BLAST search.
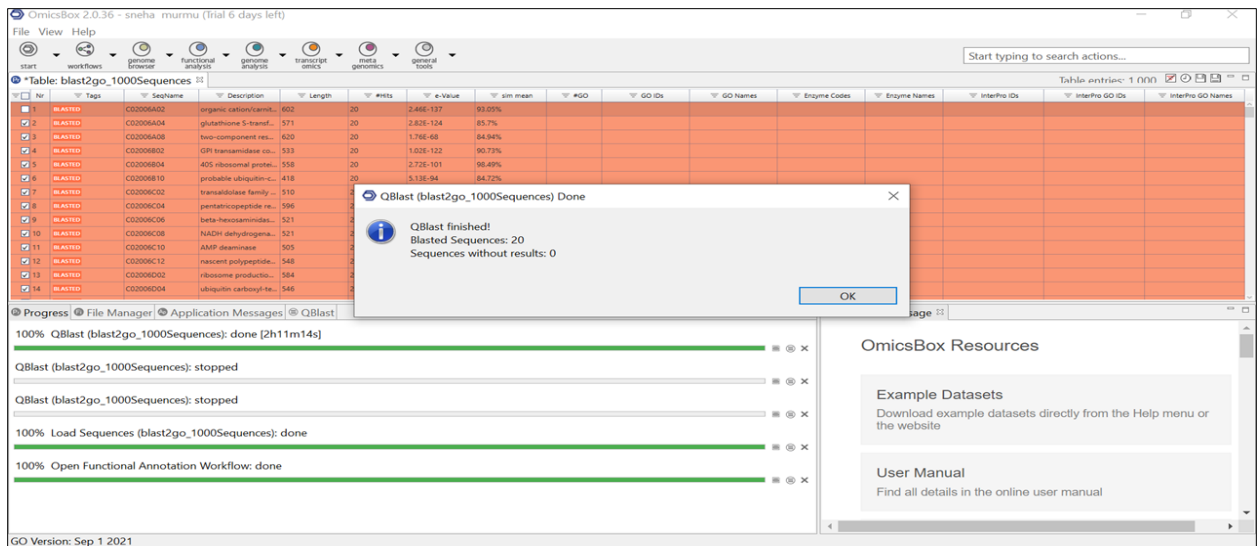
Figure 3: BLAST result.

6. Run **InterProScan**: In the main menu, click on "Run InterProScan" and select the appropriate database for your search (e.g., InterPro database). You can choose to run the search on protein or nucleotide sequences (Figure 4a).

7. Set search parameters: You can set various search parameters, such as the e-value threshold, the maximum number of sequences to align, and the type of analysis to perform (e.g., Pfam, Prosite, SMART, etc.) (Figure 4b).
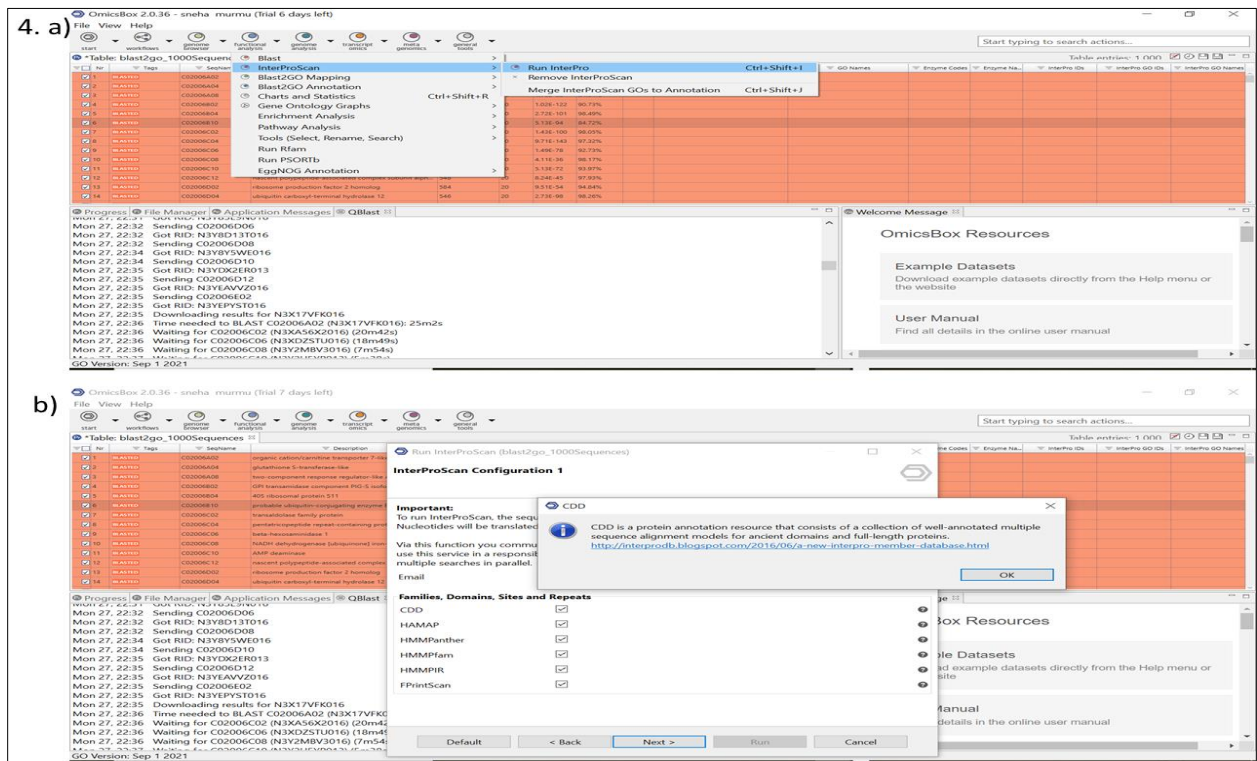
Figure 4: InterProScan search.

8. View InterProScan results: Once the InterProScan search is complete, you can view the results in the InterProScan results table. The table will show the sequence ID, the best match, the e-value, the score, and other relevant information (Figure 5). You can sort the table by various columns to help you identify the best matches.
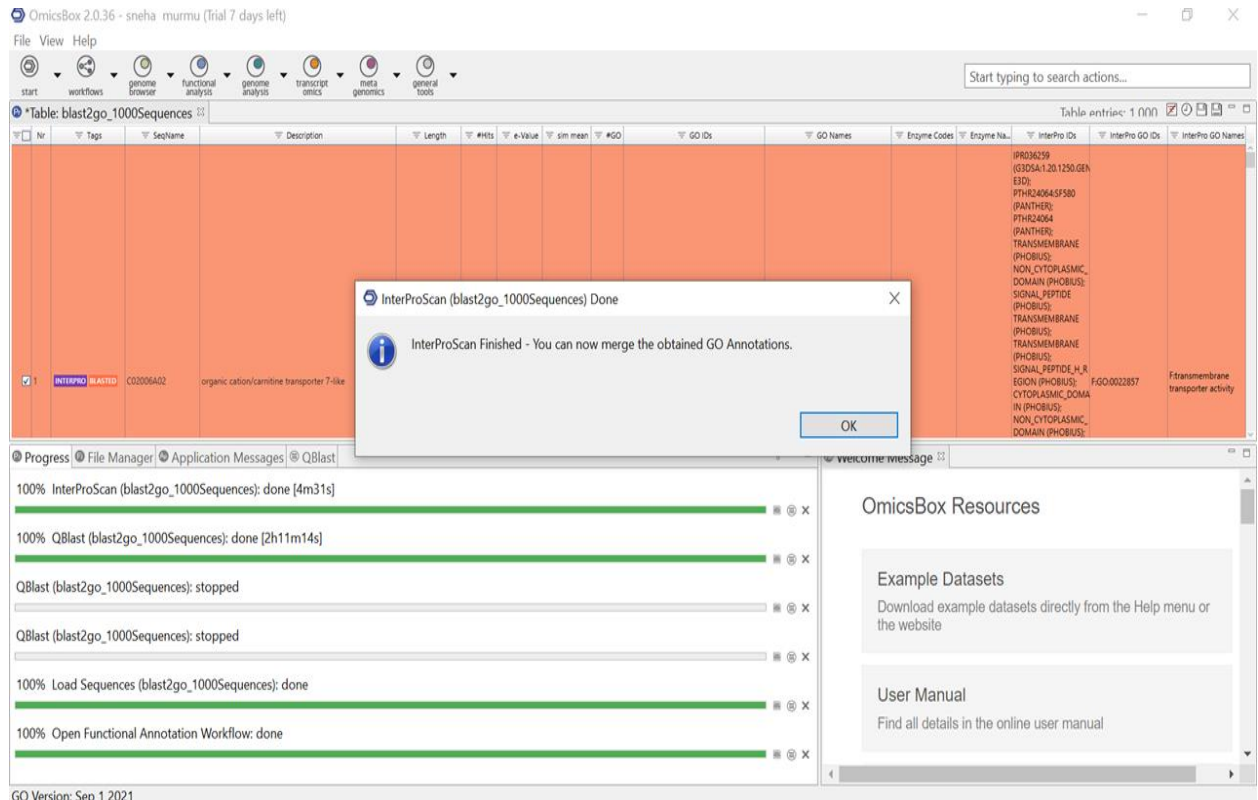
Figure 5: InterProScan result.

9. Import InterProScan results: To import the InterProScan results into the Blast2GO annotation pipeline, select the sequences you want to annotate and click on "Import selected hits". This will import the InterProScan results and link them to the appropriate sequences in the annotation pipeline.

10. Perform **mapping**: Once the BLAST results have been imported, you can use the Blast2GO mapping tools to map your sequences to Gene Ontology (GO) terms (Figure 6). This involves using the BLAST results to transfer functional annotations from similar sequences to your own sequences.
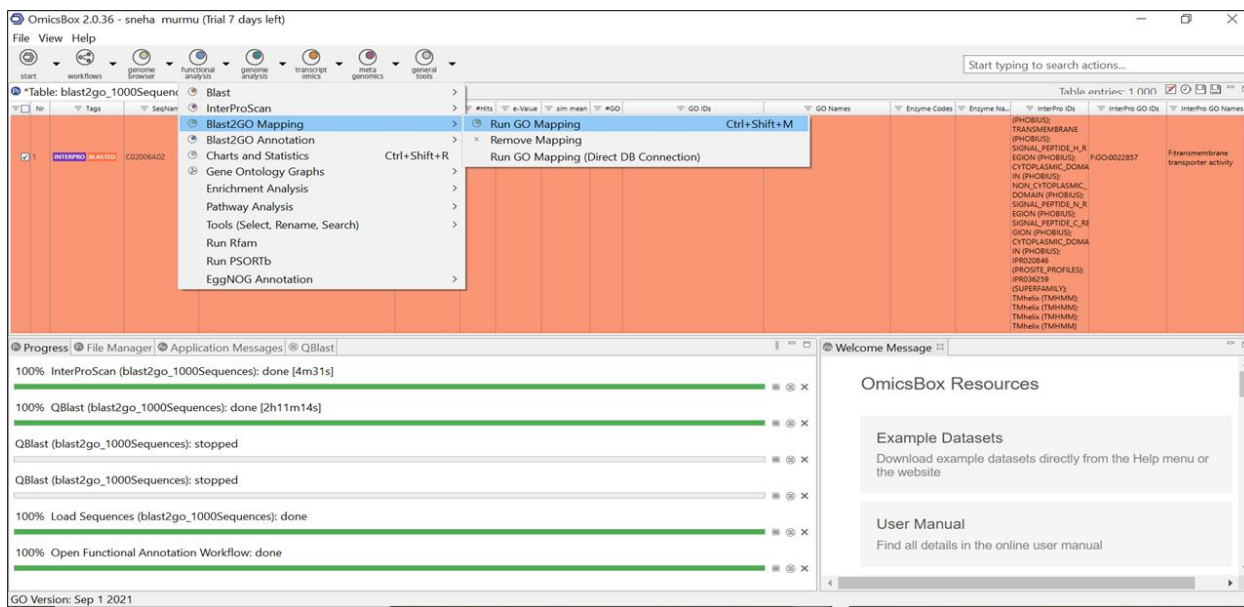
Figure 6: Mapping.

11. Edit mappings: You can edit the mappings manually, by adding or removing GO terms, or by changing the evidence codes. You can also remove or filter out low-confidence mappings, based on various criteria such as the e-value, the similarity score, or the GO term specificity.

12. Export mapping results: Once your sequences have been mapped, you can export the results in a variety of formats, such as tab-delimited text files or FASTA files (Figure 7). These results can be used for further analysis.
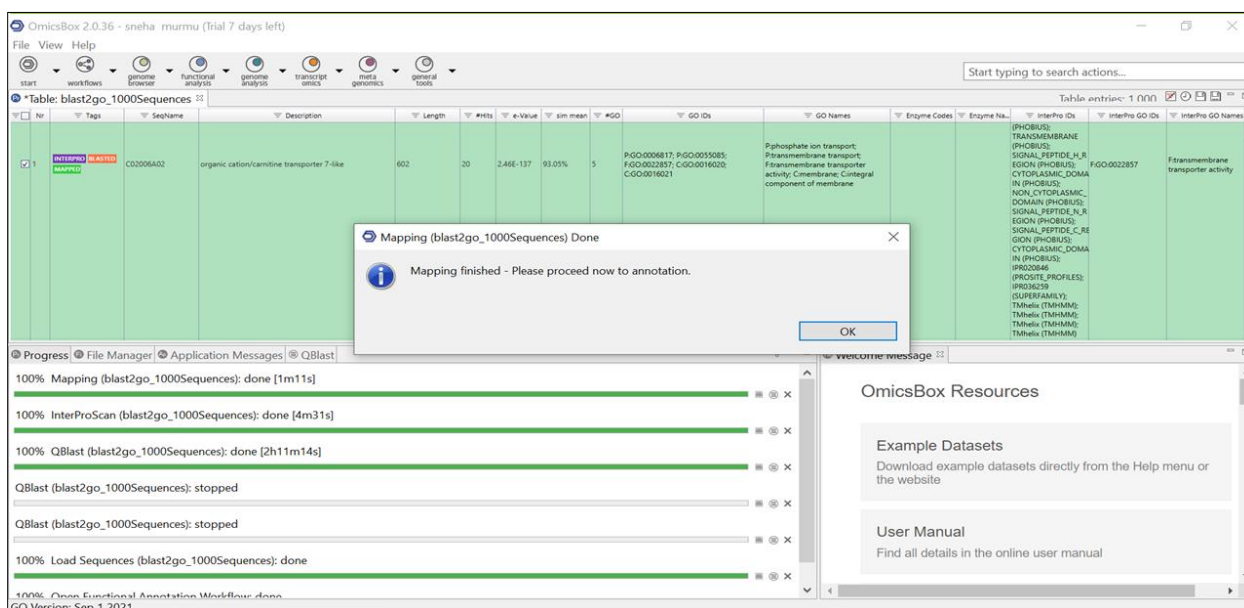
Figure 7: Mapping result.

13. **Annotate** sequences: Once the InterProScan results have been imported, you can use the Blast2GO annotation tools to assign functional information to your sequences (Figure 8). This may include mapping Gene Ontology (GO) terms, performing enrichment analysis, and performing other types of functional analysis.
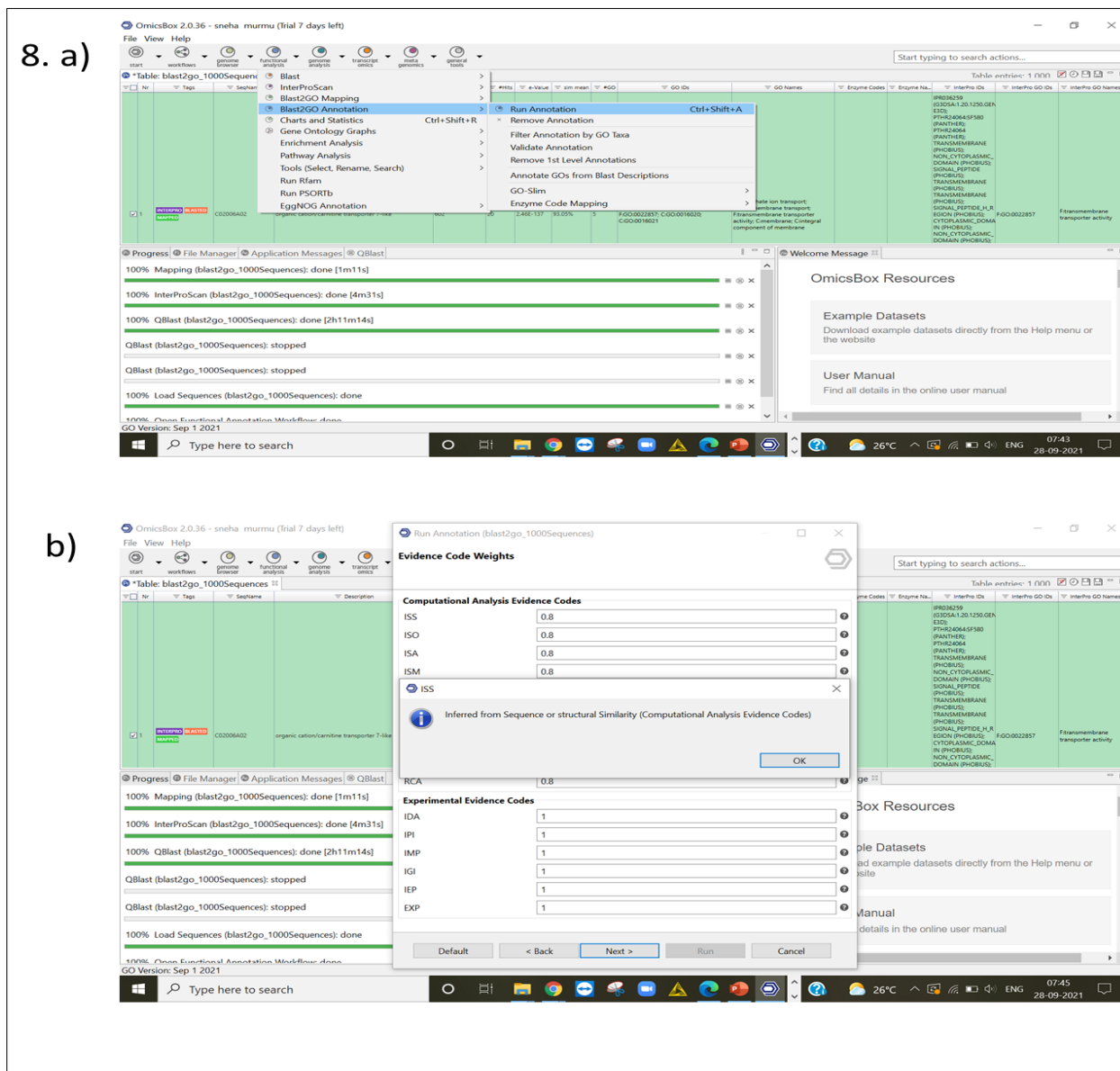
Figure 8: Annotate.

14. Export annotation results: Once your sequences have been annotated, you can export the results in a variety of formats, such as tab-delimited text files or FASTA files. These results can be used for further analysis, visualization, or sharing with collaborators.
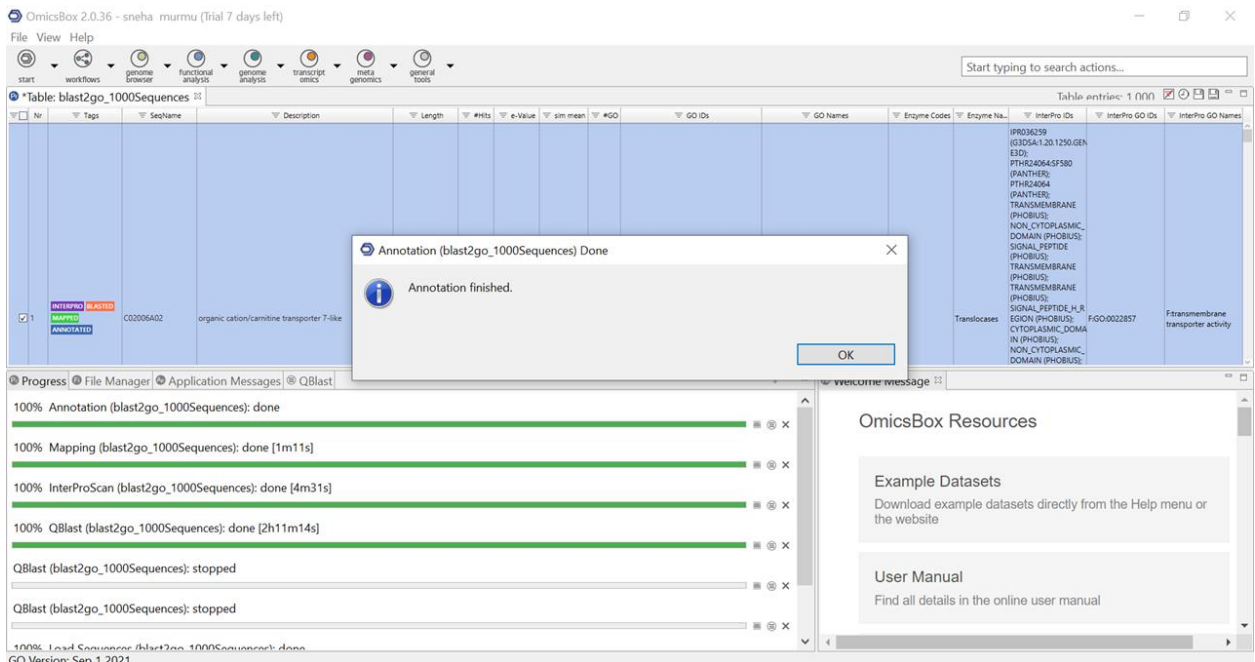
Figure 9: Annotate result.

15. Generate Gene Ontology (GO) graph: To create a GO graph in Blast2GO, click on "Graphs" in the main menu and select "GO Graph" (Figure 10). This will generate a graphical representation of the GO terms assigned to your sequences, based on the hierarchical structure of the Gene Ontology.
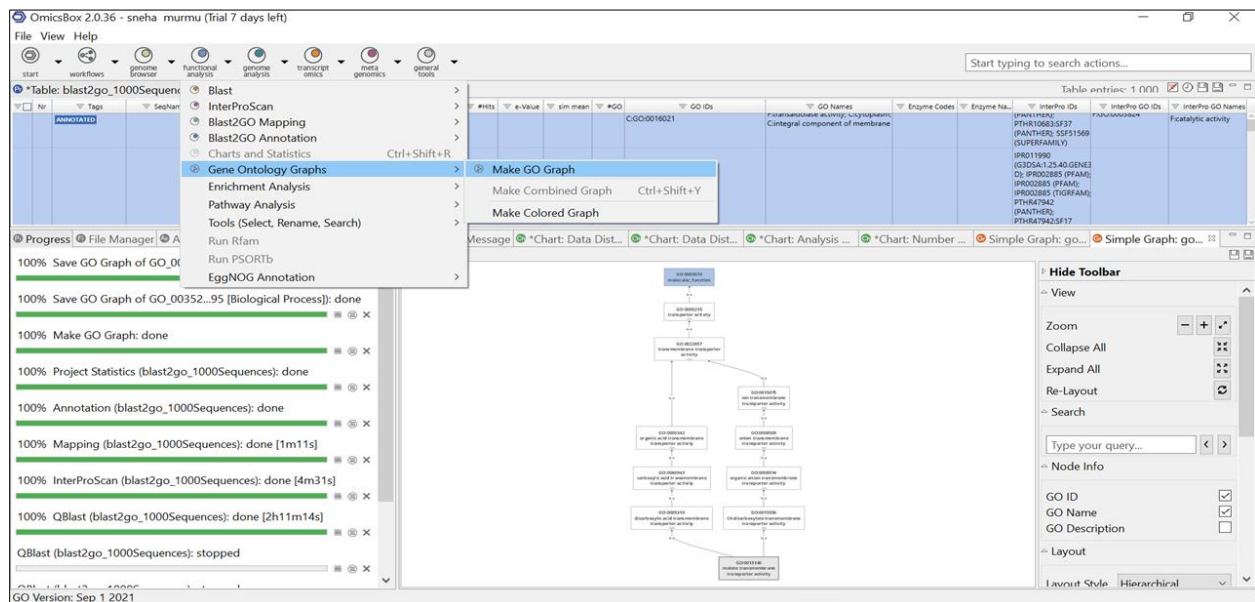
Figure 10. Generate GO graph.

16. Customize GO graph: You can customize the appearance of the GO graph by changing the colors, font sizes, or layout. You can also filter the GO terms based on various criteria such as

the level in the hierarchy, the number of sequences assigned to the term, or the statistical significance of the enrichment.

17. Analyze GO graph: Once you have generated a GO graph, you can use it to analyze the functional annotations of your sequences. This can include identifying overrepresented or underrepresented GO terms, comparing the GO profiles of different datasets or treatments, or visualizing the relationships between different biological processes, molecular functions, or cellular components (Figure 11).
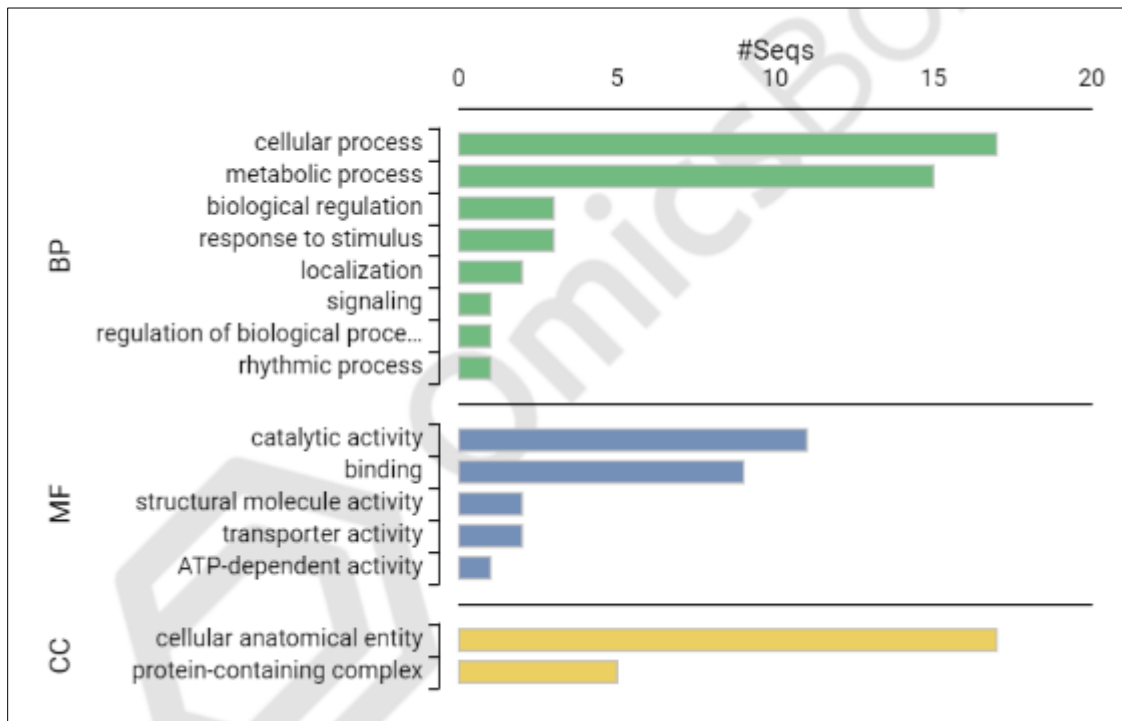
Figure 11: GO graph.

18. Export GO graph: Once you have customized and analyzed your GO graph, you can export it in a variety of formats, such as PNG, PDF, or SVG. These graphs can be used for presentations, publications, or further analysis with other tools or software.

19. Perform pathway analysis: To perform pathway analysis in Blast2GO, you need to use the KEGG (Kyoto Encyclopedia of Genes and Genomes) pathway database. In the main menu, click on "Annotation" and select "Pathway annotation". This will open the pathway annotation dialog box (Figure 12).
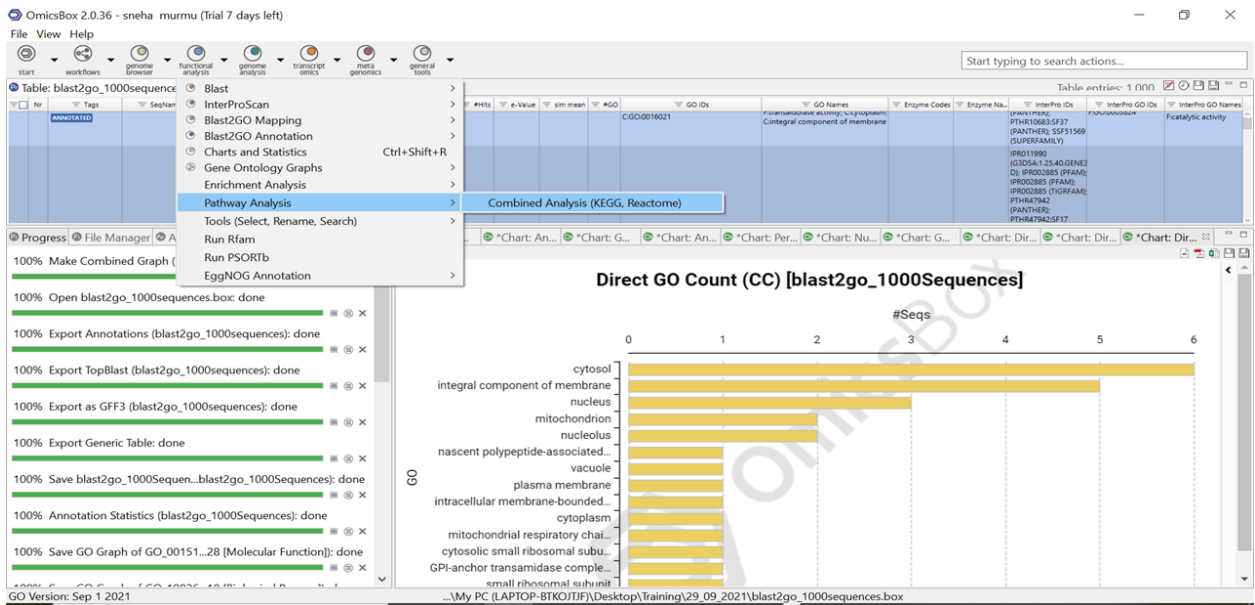
Figure 12. Run Pathway Analysis.

20. Select pathway database: In the pathway annotation dialog box, select the "KEGG" database and click on "Start". Blast2GO will download and install the latest version of the KEGG database on your computer.

21. Run pathway analysis: Once the KEGG database is installed, you can use the Blast2GO pathway analysis tools to identify the KEGG pathways that are enriched in your sequences. This involves comparing the frequency of KEGG pathway terms in your sequences to the frequency of these terms in a reference dataset, such as the entire KEGG database.

22. Filter and visualize pathways: Once the pathway analysis is complete, you can use the Blast2GO pathway analysis tools to filter and visualize the enriched pathways. This can involve setting statistical thresholds, such as the false discovery rate (FDR) or the p-value, or selecting specific pathways based on their relevance to your research question.

23. Analyze pathways: Once you have identified the enriched pathways, you can use the Blast2GO pathway analysis tools to analyze the functional annotations and gene products associated with these pathways. This can include identifying the key enzymes or regulators, comparing the pathway profiles of different datasets or treatments, or visualizing the relationships between different metabolic or signaling pathways (Figure 13).
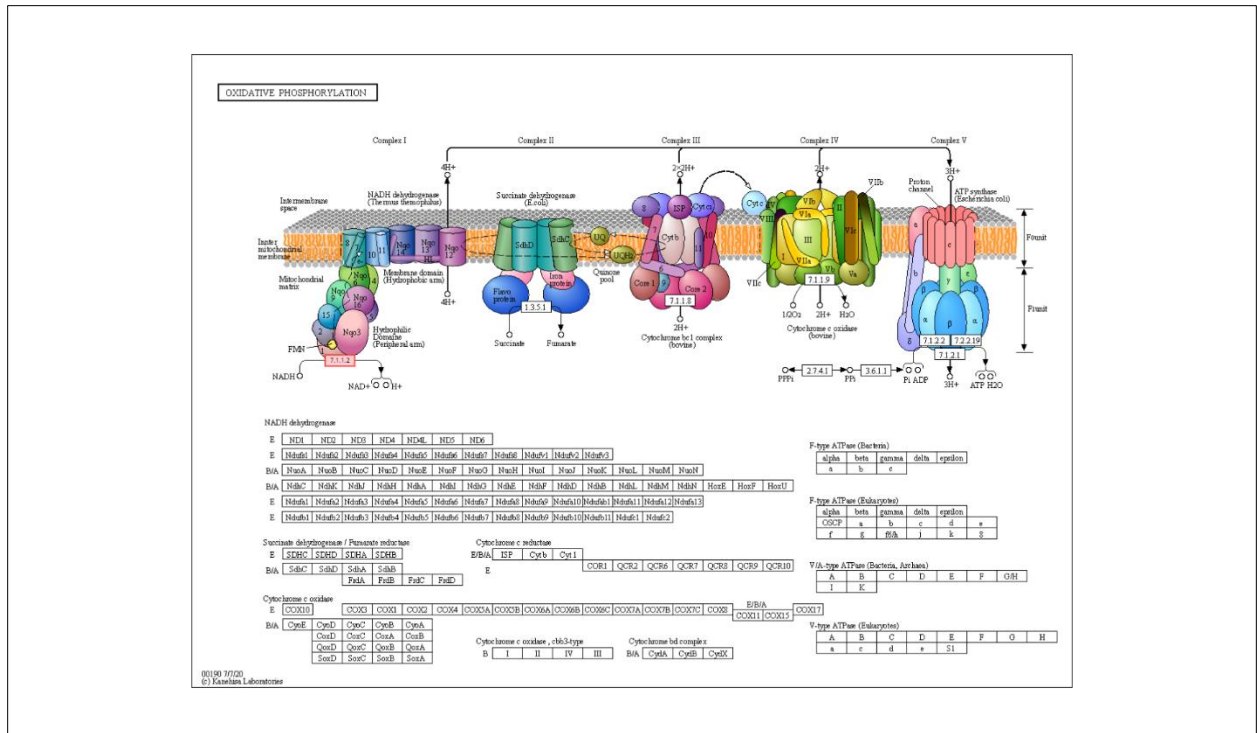
Figure 13. Pathway graph.

24. Export pathway data: Once you have customized and analyzed your pathway data, you can export it in a variety of formats, such as Excel, CSV, or XML. These data can be used for further analysis with other tools or software, or for visualizing and communicating the results of your pathway analysis.

References

Conesa, A., Götz, S., García-Gómez, J. M., Terol, J., Talón, M., & Robles, M. (2005). Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. Bioinformatics, 21(18), 3674-3676.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acids research, 25(17), 3389-3402.

Finn, R. D., Clements, J., & Eddy, S. R. (2011). HMMER web server: interactive sequence similarity searching. Nucleic acids research, 39(suppl_2), W29-W37.

Zdobnov, E. M., & Apweiler, R. (2001). InterProScan–an integration platform for the signature-recognition methods in InterPro. Bioinformatics, 17(9), 847-848.

R for Bioinformatics

Sudhir Srivastava

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

R is a programming language that allows for advanced statistical computing and graphics. It was created by the statisticians Ross Ihaka and Robert Gentleman. It is supported by the R Core Team and the R Foundation for Statistical Computing. The language is very powerful for writing programs. Output may be limited based on the function, but even small code can generate wonderful graphics. It is very sensitive to syntax, case, punctuation used, even spacing. R is open source and free on the Internet. R is used among statisticians, computer scientists and bioinformaticians for data analysis and developing statistical software. The official R software environment is an open-source free software environment within the GNU package, available under the GNU General Public License. It is written primarily in C, Fortran, and R itself (partially self-hosting). Precompiled executables are provided for various operating systems. R has a command line interface as well as multiple third-party graphical user interfaces such as RStudio (an integrated development environment) and Jupyter (a notebook interface).

Working in R and RStudio

R can be installed in Linux, Unix, Windows and Mac platforms from www.r-project.org. For downloading R, please visit https://cloud.r-project.org/.

The R GUI

RStudio is a free, open-source IDE (integrated development environment) for R. It can be downloaded from https://www.rstudio.com/products/rstudio/download/. One must install R before installing RStudio. The interface is organized so that the user can clearly view graphs, data tables, R code, and output all at the same time.
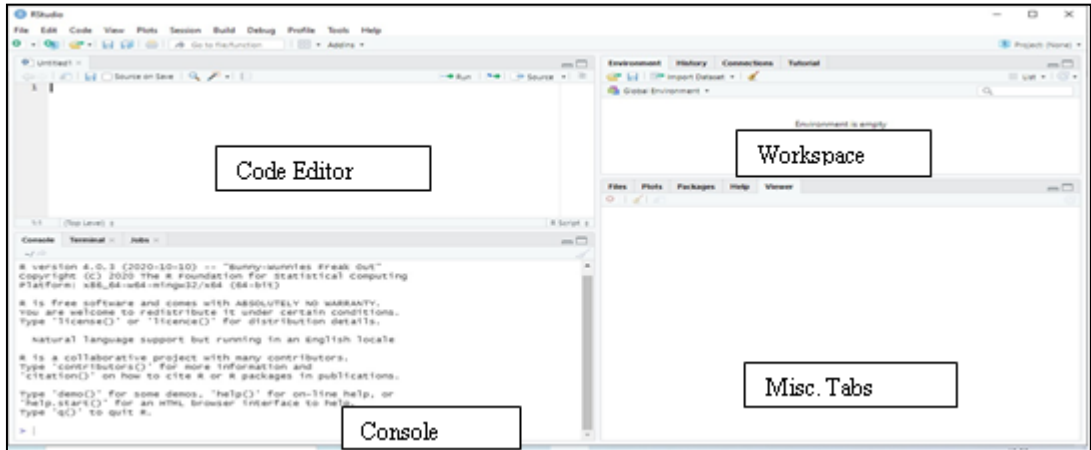
R Studio Interface

There are various ways for working in R:

- Work directly from the R editor to type in your script and execute the script completely (batch) or line-by-line (highlight and execute)

- Write script in an external editor (Notepad or software that interfaces with R) and execute in R by copy/paste or highlighting

- Beyond the native R GUI, external GUI can work with R to help in writing scripts, selecting functions, procedures, statistical tests, or graphics
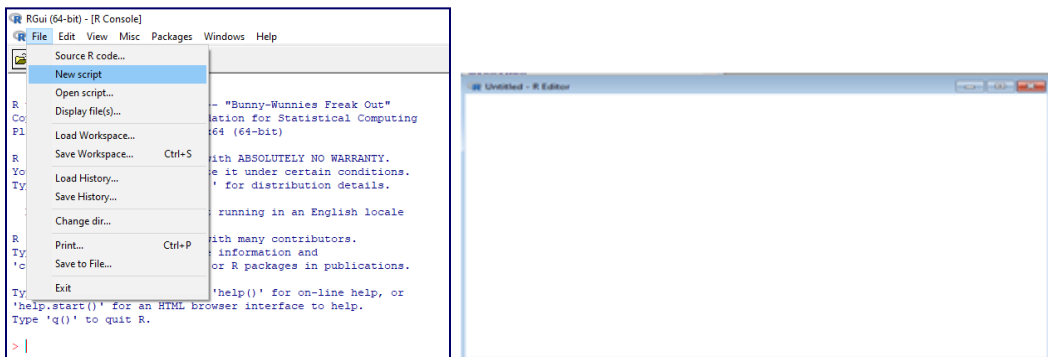
Getting started: R
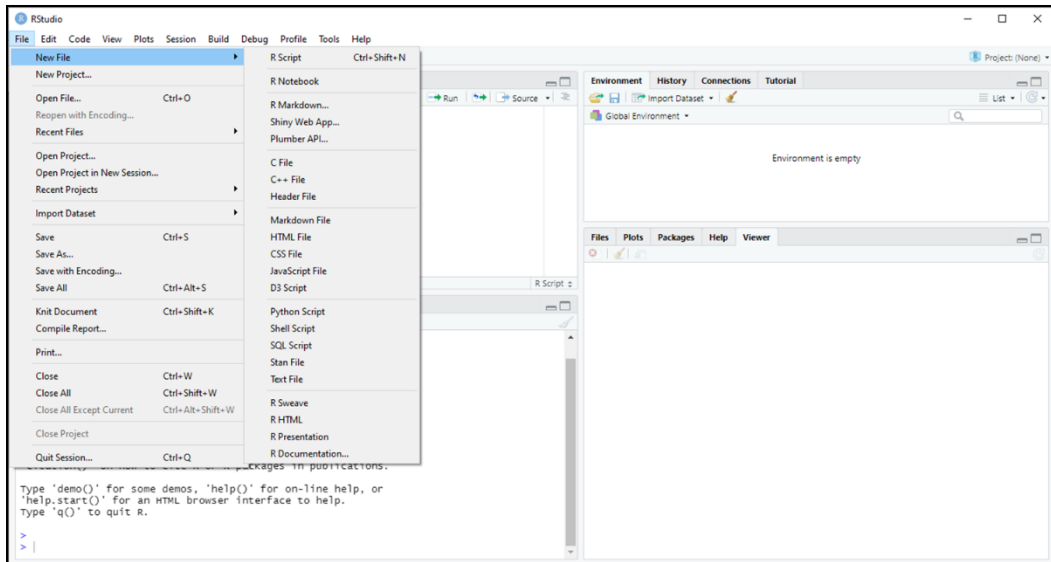
Getting started: RStudio

R is an expression language with a very simple syntax. It is case sensitive as are most UNIX based packages. For example, A and a are different symbols and refer to different variables. The set of symbols which can be used in R names depends on the operating system and country within which R is being run (technically on the locale in use). Normally all alphanumeric symbols are allowed (and in some countries this includes accented letters) plus '.' and '_', with the restriction that a name must start with '.' or a letter, and if it starts with '.' the second character must not be a digit. Elementary commands consist of either expressions or assignments. If an expression is given as a command, it is evaluated, printed (unless specifically made invisible), and the value is lost. An assignment evaluates an expression and passes the value to a variable but the result is not automatically printed. Commands are separated either by a semi-colon (';'), or by a newline. Elementary commands can be grouped together into one compound expression by braces ('{' and '}'). Comments can be put almost anywhere, starting with a hashmark ('#'), everything to the end of the line is a comment. If a command is not complete at the end of a line, R will give a different prompt, by default + on second and subsequent lines and continue to read input until the command is syntactically complete.

R Workspace

R workspace is temporary space on your CPU's RAM that "disappears" at the end of R session. It includes any user-defined objects (vectors, matrices, data frames, lists, functions). All data, analyses, output are stored as objects in the R workspace. This workspace is not saved on disk unless you tell R to do so. This means that your objects are lost when you close R and not save the objects, or worse when R or your system crashes on you during a session. When you close the RGui or the R console window, the system will ask if you want to save the workspace image. If you select to save the workspace image then all the objects in your current R session are saved in a file ".RData". ".RData" is a binary file located in the working directory of R, which is by default the installation directory of R. During your R session, you can also explicitly save the workspace image.

Go to the 'Session' menu and then select 'Save Workspace as'

> save.image("example1.Rdata")

If you have saved a workspace image and you start R the next time, it will restore the workspace. So all your previously saved objects are available again.

Go to the 'Session' menu and then select 'Load Workspace'.

> load.image("example1.Rdata")

- Windows uses a \ (left slash) to delineate locations in CPU:

  C:\Users\hp\Documents

- R uses / (right slash) to delineate locations in CPU:

  C:/Users/hp/Documents

- An alternative to R's / (single right) is \\ (two left) slashes:

  C:\\Users\\hp\\Documents

- There is no issue in the MAC OS/Linux as they have retained the / (right slash) as the basis for directory delineation

- Print the current working directory

  > getwd()

- List the objects in the current workspace

  > ls()

- Change to my directory

  > setwd(mydirectory)

- Display last 25 commands

  > history()

- Display all previous commands

  > history(max.show=Inf)

- Saving R workspace

  > x <- 5 # object x; x is assigned value 5

  > y <- 10 # object y; y is assigned value 10

  > z <- x+y # object z (addition of numbers x and y); z is assigned the value x+y

  > save(x, y, file = "example1_xy.RData") # save two specified objects x and y

  > save.image(file = "example1.RData") # save entire workspace

- Removing objects R workspace: Use rm()

  > ls()

   [1]   "x" "y" "z"

  > rm(x, y) # removes objects x and y

  > ls()

  [1] "z"

- Use load() to add previously saved objects or workspaces to your current R session.

  > load(file = "example1.RData")

  > ls()

  [2] "x" "y" "z"

Getting help with functions and features

To get more information on any specific named function, use help() function or ? help operator.

> help(lm) or > help("lm")

> ?lm

For a feature specified by special characters, the argument must be enclosed in double or single quotes, making it a "character string". This is also necessary for a few words with syntactic meaning including if, for and function.

> help("[[")

The convention is to use double quote marks for preference.

On most R installations help is available in HTML format by running help.start() which will launch a Web browser that allows the help pages to be browsed with hyperlinks. The help.search command (alternatively ??) allows searching for help in various ways.

> help.search("lm")

> ??lm

The examples on a help topic can normally be run by

> example(lm)

Windows versions of R have other optional help systems: Use ?help for further details.

R Datasets

R comes with a number of sample datasets that you can experiment with. One has to type data( ) to see the available datasets. The results will depend on which packages you have loaded. For getting details on a sample dataset, type help(datasetname). Example: > help("AirPassengers")

R Packages

One of the strengths of R is that the system can easily be extended. The system allows you to write new functions and package those functions in a so called `R package' (or `R library'). The R package may also contain other R objects, for example data sets or documentation. There is a lively R user community and many R packages have been written and made available on CRAN for other users. For

example, there are packages for statistics, bioinformatics and many more. To attach package to the system you can use the menu or the library function.

- Via the menu in RGui: Select the 'Packages' menu and select 'load package...', a list of available packages on your system will be displayed. Select one and click 'OK'.
- Via the library function: > library( )

Data Management

Everything in R is an object. An object is simply a data structure that has some methods and attributes. The data elements in any R object has attributes. These attributes describe the nature of the elements. Object attributes are modes, class and types.

- **Modes:** logical (TRUE, FALSE), numeric, character (string), complex (complex number)
- **Type** (e.g. vectors can be character, numeric, logical or complex)
- **Class:** Describes object type and mode of object or element that is specified.

Objects in R:

- **Scalar**: a single number (1x1 vector)
- **Vector**: all elements of the same type (Type: logical, character, numeric or complex)
- **List**: can contain objects of different types
- **Matrix**: table of vectors, where all elements are numeric (or complex)
- **Data frame**: table of number and/or character vectors. Can contain lists, too.

Data objects in R can exist in many different modes, classes, and types. mode( ) function returns the mode of an object. Some object classes like arrays and matrices require all elements to be of the same mode. A vector can have only mode type of elements. It can have only numeric, character, logical or complex elements. Other objects (data frames, lists) allow for different modes to exist, i.e. objects within data frames and lists can be of different modes. Class describes object type and mode of object or element that is specified. class( ) function returns class of an object. Examples: "vector", "data.frame", "numeric", "factor"

```
> z <- 0:9
> z
 [1] 0 1 2 3 4 5 6 7 8 9
```

```
> digits <- as.character (z)
> digits
[1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
> d <- as.integer (digits)
> d
[1] 0 1 2 3 4 5 6 7 8 9
> class (z)
[1] "integer"
> class (digits)
[1] "character"
> class (d)
[1] "integer"
```

Vector Arithmetic

<- the arrow is the assignment symbol, used to assign a value or function to a symbol or object. The '=' operator can be used as an alternative.

```
> 5+10
[1] 15
> x <- 5 # object x; x is assigned value 5
> y <- 10 # object y; y is assigned value 10
> z <- x+y # object z; z is assigned the value x+y
> z # Display z
[1] 15
> sqrt(z)
[1] 3.872983
 > ls() # List objects
 [1] "x" "y" "z"
```
Here, x, y and z are scalar objects, each having a single value.

Assignment statement using c() *function*

```
> x <- c(9.5, 10.8, 2.5, 3.9, 19.6)
> x
[1]  9.5 10.8  2.5  3.9 19.6
> assign("x", c(9.5, 10.8, 2.5, 3.9, 19.6))
> x
```

```
[1]  9.5 10.8  2.5  3.9 19.6
> c(9.5, 10.8, 2.5, 3.9, 19.6) -> x
> x
[1]  9.5 10.8  2.5  3.9 19.6
> 1/x
[1] 0.10526316 0.09259259 0.40000000 0.25641026 0.05102041
> y <- c(x, 1, 0, 1, x)
> y
 [1]  9.5 10.8  2.5  3.9 19.6  1.0  0.0  1.0  9.5 10.8  2.5  3.9 19.6
```

The elementary arithmetic operators:
- +, -, *, / and ^
- log, exp, sin, cos, tan, sqrt
- max and min select the largest and smallest elements of a vector respectively.
- range is a function whose value is a vector of length two, namely c(min(x), max(x)).
- length(x) is the number of elements in x.
- sum(x) gives the total of the elements in x.
- prod(x) gives the product.

```
> x <- c(1:10)
> x
[1]  1  2  3  4  5  6  7  8  9 10
> x [x>6]
[1]  7  8  9 10
> x [(x>6) | (x<4)]
[1]  1  2  3  7  8  9 10
> x <- seq (1,10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> rev (x) # reverse order
 [1] 10  9  8  7  6  5  4  3  2  1
> x <- (1:4)^2
> x
[1]  1  4  9 16
```

Missing values

Arithmetic functions on missing values yield missing values.

```
> x <- c(1, 5, 4, NA, 6)
> x
[1]  1  5  4 NA  6
> mean(x)
[1] NA
> mean(x, na.rm = TRUE)
[1] 4
```

The function is.na(x) gives a logical vector of the same size as x with value TRUE if and only if the corresponding element in x is NA.

```
> is.na(x)
[1] FALSE FALSE FALSE  TRUE FALSE
```

Impossible values (e.g., dividing by zero) are represented by the symbol NaN (Not a Number).

```
> 5/0
[1] Inf
> 0/0
[1] NaN
> Inf - Inf
[1] NaN
```

is.na(xx) is TRUE both for NA and NaN values.

is.nan(xx) is only TRUE for NaNs.

```
> color <-c("red", "green", "blue")
> color # the values of character variable color are red, green and blue
[1] "red"   "green" "blue"
> cat(color) # remove quotation marks
red green blue
> cat(color[1])
red
```

Assign names to the Elements

```
> x <- c(Delhi="red", Mumbai="green", Kolkata="blue")
> x
 Delhi  Mumbai Kolkata
 "red" "green"  "blue"
```

```
> names(x)
[1] "Delhi"   "Mumbai"  "Kolkata"
> fruit <- c(2, 3, 6)
> names(fruit) <- c("orange", "apple", "banana")
> fruit
orange  apple banana
    2      3      6
> fruit[c("apple","orange")]
 apple orange
    3      2
> Fruit <- c(orange=2, apple=3, banana=6)
> Fruit
orange  apple banana
      2      3      6
```

All elements of a vector must have the same type. If you concatenate vectors of different types, they will be converted to the least "restrictive" type.

```
> c(2, "car")
[1] "2"   "car"
```

Logical values are converted to 0 / 1 OR "TRUE"/ "FALSE".

```
> c(FALSE, 5)
[1] 0 5
> c(FALSE, "red")
[1] "FALSE" "red"
```

***Background in Vector Arithmetic*:** Vector addition required the vectors to be the same length (dimension).

```
x <- c(9, 2)
> x
[1] 9 2
> y <- c(5, 1)
> y
[1] 5 1
> x + 5
[1] 14  7
> x + y
```

```
[1] 14  3
> x - y
[1] 4 1
> x*y
[1] 45  2
> 2*x+y+5
[1] 28 10
> x/y
[1] 1.8 2.0
```

*Concatenate – **c()***
c(x, y)
```
> z <- c(6, 4, 1, 0)
> z
[1] 6 4 1 0
> x <- c(6, 4)
> x
[1] 6 4
> y <- c(1, 0)
> y
[1] 1 0
> z <- c(x, y)
> z
[1] 6 4 1 0
```

Generating regular sequences – seq()
```
> x1 <- 1:10
> x1
 [1]  1  2  3  4  5  6  7  8  9 10
> x2 <- seq(1, 10)
> x2
 [1]  1  2  3  4  5  6  7  8  9 10
> x3 <- seq(1, 10, by = 2)
> x3
[1] 1 3 5 7 9
> x4 <- seq(10, 22, length = 5)
```

```
> x4
[1] 10 13 16 19 22
> x5 <- seq(length = 31, from = -5, by = 3)
> x5
 [1] -5 -2  1  4  7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70
73
[28] 76 79 82 85
```

Generating regular sequences – **rep()**

Replicate or repeat

```
   > x6 <- rep(3, 5)
   > x6
   [1] 3 3 3 3 3
   > x7 <- 1:3
   > x7
   [1] 1 2 3
   > x8 <- rep(x7, times = 5) # put five copies of x7 end-to-end in x8
   > x8
    [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
   > x9 <- rep(x7, each = 5) # repeats each element of x7 five times before moving
   on to the next
   > x9
    [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

Summaries and Subscripting

```
> x <- c(1, 3, 4, 7, 11, 32)
> x[1:3]
[1] 1 3 4
> x[c(1:3, 6)]
[1]  1  3  4 32
> x[-(1:4)]
[1] 11 32
> mean(x) # Mean
[1] 9.666667
> m1 <- sum(x)/length(x)
> m1
```

```
[1] 9.666667
> var(x) # Variance
[1] 131.8667
> sum((x-m1)^2)/(length(x)-1)
[1] 131.8667
> sd(x) # Standard deviation
[1] 11.48332
> sqrt(sum((x-m1)^2)/(length(x)-1))
[1] 11.48332
> summary(x) # Summary
  Min. 1st Qu.  Median   Mean 3rd Qu.   Max.
 1.000  3.250   5.500  9.667  10.000 32.000
> summary(x[1:4]) # Summary
  Min. 1st Qu.  Median   Mean 3rd Qu.   Max.
 1.00   2.50    3.50   3.75   4.75   7.00
```

Matrices

Matrices or more generally arrays are multi-dimensional generalizations of vectors.
In fact, they are vectors that can be indexed by two or more indices.

```
> X <- matrix(1:12, nrow = 3, ncol = 4)
> X
   [,1] [,2] [,3] [,4]
[1,]   1    4    7   10
[2,]   2    5    8   11
[3,]   3    6    9   12
> dim(X)
[1] 3 4
> Y <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
> Y
   [,1] [,2] [,3] [,4]
[1,]   1    2    3    4
[2,]   5    6    7    8
[3,]   9   10   11   12
```
Assigning names to rows and columns
```
> rownames(X) <- c("A", "B", "C")
```

```
> X
  [,1] [,2] [,3] [,4]
A   1   4   7   10
B   2   5   8   11
C   3   6   9   12
> colnames(X) <- c("X1", "X2", "X3", "X4")
> X
  X1 X2 X3 X4
A 1 4 7 10
B 2 5 8 11
C 3 6 9 12
```

Accessing elements of a matrix

```
> X
  X1 X2 X3 X4
A 1 4 7 10
B 2 5 8 11
C 3 6 9 12
> X[,1]
A B C
1 2 3
> X[1,]
X1 X2 X3 X4
 1  4  7 10
> X[2, 3]
[1] 8
```

Adding additional rows or binding matrices – rbind()
Adding additional columns or binding matrices – cbind()

```
> X <- matrix(1:12, nrow = 3, ncol = 4)
> X
     [,1] [,2] [,3] [,4]
[1,]   1    4    7   10
[2,]   2    5    8   11
[3,]   3    6    9   12
> rbind(X, c(5, 1, 2, 6))
     [,1] [,2] [,3] [,4]
[1,]   1    4    7   10
```

```
[2,]   2   5   8   11
[3,]   3   6   9   12
[4,]   5   1   2    6
> cbind(X, c(3, 4, 9))
    [,1] [,2] [,3] [,4] [,5]
[1,]   1   4   7   10   3
[2,]   2   5   8   11   4
[3,]   3   6   9   12   9
```

Transpose – t(); Determinant – det(); Inverse – solve()
```
> X <- matrix(c(1, 3, 8, 12), nrow = 2, byrow = TRUE)
> X
    [,1] [,2]
[1,]   1    3
[2,]   8   12
> t(X) # Transpose of matrix
    [,1] [,2]
[1,]   1    8
[2,]   3   12
> det(X) # Determinant of matrix
[1] -12
> solve(X) # Inverse of matrix
        [,1]        [,2]
[1,] -1.0000000  0.25000000
[2,]  0.6666667 -0.08333333
```

List and Data Frame
An R list is an object consisting of an ordered collection of objects known as its
components.
```
> Lst <- list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))
> Lst
$name
[1] "Fred"
$wife
[1] "Mary"
$no.children
```

```
[1] 3
$child.ages
[1] 4 7 9
> length(Lst) # Length
[1] 4
> names(Lst) # Names
[1] "name"     "wife"      "no.children" "child.ages"
 > Lst$no.children
[1] 3
> Lst[[3]]
[1] 3
```

A data frame object in R has similar dimensional properties to a matrix but it may contain categorical data, as well as numeric (mixed modes). The standard layout is to put data for one observation across a row and variables as columns. Columns can be thought of as vectors, being either numeric or character. Columns can have column names, similar to variable names. Column names can be of any length, consisting of letters, numbers and a period (.) if desired. Underscores are not allowed. Column names must start with a letter. Columns (vectors) in a data.frame must be of the same length. On one level, as the notation will reflect, a data frame is a list. Each component corresponds to a variable, i.e., the vector of values of a given variable for each sample. Therefore, a data frame is like a list with components as columns of table. Lists have columns of the same lengths.
A list can be made into a data.frame:

✓ Components must be vectors (numeric, character, logical) or factors.
✓ All vectors and factors must have the same lengths.

Matrices and even other data frames can be combined with vectors to form a data frame if the dimensions match up.

```
> students <- data.frame(gender = c("F", "M","F"), ht = c(170, 188.5, 168.3), wt = c(91.8,90, 82.6))
> students
  gender   ht  wt
1     F 170.0 91.8
2     M 188.5 90.0
3     F 168.3 82.6
> students[1, 2]  # Identify the row 1, col 2 element in object Students
```

[1] 170
> names(students) # Identify the column names in object Students
[1] "gender" "ht"   "wt"
> rownames(students) <- c("S1", "S2", "S3") # Apply row names to object Students
> students
   gender   ht   wt
S1   F 170.0 91.8
S2   M 188.5 90.0
S3   F 168.3 82.6

Lists
Lists combine a collection of objects into a larger composite object.
> intake.pre <- c(23,35,34,13,46, 45,34)
> intake.post <- c(56,57,36,58,36,67,32)
> mylist <- list(before=intake.pre, after=intake.post)
> mylist
$before
[1] 23 35 34 13 46 45 34
$after
[1] 56 57 36 58 36 67 32
> mylist[1]
$before
[1] 23 35 34 13 46 45 34
> mylist[[1]]
[1] 23 35 34 13 46 45 34
> dat <- data.frame(intake.pre, intake.post)
> dat
  intake.pre intake.post
1   23   56
2   35   57
3   34   36
4   13   58
5   46   36
6   45   67
7   34   32
> dat$intake.pre

```
[1] 23 35 34 13 46 45 34
> dat$intake.pre[3]
[1] 34
> dat$intake.pre[c(1,3)]
[1] 23 34
> dat$intake.pre[-3]
[1] 23 35 13 46 45 34
```

Factor

Factors are the data objects which are used to categorize the data and store it as
levels. They can store both strings and integers. They are useful in the columns
which have a limited number of unique values such as gender (Male, Female), etc.

factor(x = character(), levels, labels = levels, ordered = is.ordered(x))

```
> gender <- c("male","male","female","female","male","female","male")
> gender
[1] "male"  "male"  "female" "female" "male"  "female" "male"
> class(gender)
[1] "character"
> gender <- factor(gender)
> gender
[1] male  male  female female male  female male
Levels: female male
> class(gender)
[1] "factor"
```

Two-way Layout

Consider our two-way layout problem, where we produced the indicator variables
using rep(). A better way to do this is using the function *gl*, which will generate
factors.

```
> clevels <- gl(3,8)
> clevels
 [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
+ 3
Levels: 1 2 3
> rlevels <- gl(4,2,length=24)
> rlevels
```

[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4

+ 4

Levels: 1 2 3 4

Use the function expand.grid to produce a data frame with the desired factors.

```
> reps
[1] 1 2
> colLevels <- 1:3
> colLevels
[1] 1 2 3
> rowLevels <- 1:4
> rowLevels
[1] 1 2 3 4
> height = seq(60, 80, 10)
> height
[1] 60 70 80
> weight = seq(100, 200, 50)
> weight
[1] 100 150 200
> sex = c("Male","Female")
> sex
[1] "Male"   "Female"
```

Generating Random Numbers

As a language for statistical analysis, R has a comprehensive library of functions for generating random numbers from various statistical distributions.

Example: Generate 5 random integers between 1 and 10

```
> set.seed (100) # function in R used to reproduce results
> sample (1:10, 5) # sampling without
 replacement is the default
[1] 10  7  6  3  1
> sample (1:10, 5, replace = TRUE)
[1] 10  7  6  6  4
> sample (c("H","T"),5, replace = TRUE)
[1] "H" "T" "T" "H" "H"
> runif (5, 0, 1) # generating between 0 and 1, excluding 0 and 1
[1] 0.6902905 0.5358112 0.7108038 0.5383487 0.7489722
```

> rnorm (5, 1, 3) # generating random numbers from normal dist with (1,3)
[1] 0.3950981 3.2195215 1.3701385 0.9120499 -0.1665627

Importing Data

> mydata <- read.table ("mydata.txt", header=TRUE) # From Text file
> head(mydata)

	Height	Weight	Sex
1	60	100	Male
2	70	100	Male
3	80	100	Male
4	60	150	Male
5	70	150	Male
6	80	150	Male

> mydata <- read.table ("mydata.csv", header=TRUE) # From CSV file
> mydata <- read.delim ("mydata.csv") # Importing file with a separator character
> mydata <- read.delim2("mydata.csv")

Importing from Excel: Importing from 1st worksheet
We will require a package named 'xlsx'.
> library(xlsx)
Warning message:
package 'xlsx' was built under R version 4.0.5
> mydata <- read.xlsx("mydata.xlsx", 1)

Importing SPSS
library(foreign)
mydata <- read.spss("mydata.sav", to.data.frame=TRUE,
use.value.labels=FALSE)

Importing SAS files
library(sas7bdat)
mydata <- read.sas7bdat("mydata.sas7bdat")

Importing Minitab files
library(foreign)
mydata <- read.mtp("mydata.mtp")

Descriptive Statistics

Descriptive statistics investigates the variables separately. Various descriptive statistics can be computed by using in-built R functions as given below.

Name of function	Use of function
mean	calculates the mean of an input
median	calculates the median of an input
var	calculates the variance of an input
sd	calculates the standard deviation of an input
IQR	calculates the interquartile range of an input
min	calculates the minimum value of an input
max	calculates the maximum of an input
range	returns a vector containing the minimum and maximum of all given arguments
summary	returns a vector containing a mixture of the above functions (minimum, first quartile, median, mean, third quartile, maximum)

```
> data(trees)
> head(trees)
  Girth Height Volume
1  8.3    70   10.3
2  8.6    65   10.3
3  8.8    63   10.2
4 10.5    72   16.4
5 10.7    81   18.8
6 10.8    83   19.7

> summary(trees)
    Girth          Height      Volume
 Min.   : 8.30  Min.   :63  Min.   :10.20
 1st Qu.:11.05  1st Qu.:72  1st Qu.:19.40
 Median :12.90  Median :76  Median :24.20
 Mean   :13.25  Mean   :76  Mean   :30.17
 3rd Qu.:15.25  3rd Qu.:80  3rd Qu.:37.30
 Max.   :20.60  Max.   :87  Max.   :77.00
```

```
> mean(trees$Height)
[1] 76
> sd(trees$Height)
[1] 6.371813
> range(trees$Height)
[1] 63 87
```

Graphics

Histogram plots the frequencies that data appears within certain ranges.

`> data(trees)`

*Add a title***:** The "main" statement will give the plot an overall heading.

*Add axis labels***:** Use "xlab" and "ylab" to label the X and Y axes, respectively.

*Changing colors***:** Use the col statement

hist(trees$Height, main="Height of Cherry Tree", xlab="Height",
ylab="Frequency", col="red")
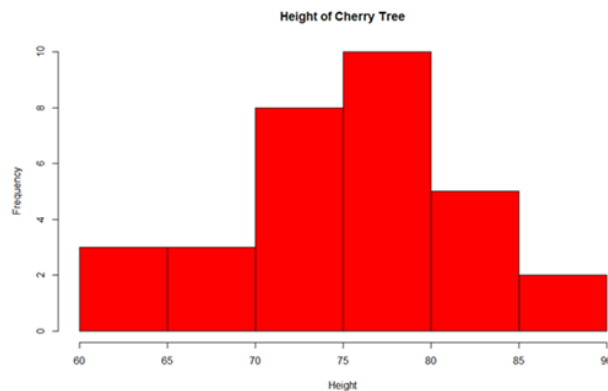
A boxplot provides a graphical view of the median, quartiles, maximum, and minimum of a data set.

`> boxplot(trees$Volume,main='Volume of Timber', ylab='Volume (cubic ft)')`
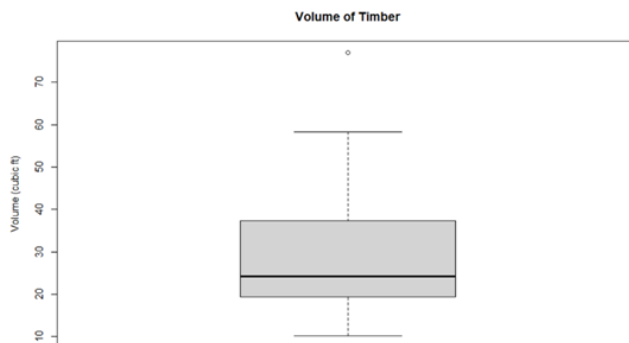
Partitioning the Graphics Window

A useful facility before beginning is to divide a page into smaller pieces so that more than one figure can be displayed graphically.

par: used to set or query graphics parameters

par(mfrow=c(2,2))

This will create a window of graphics with 2 rows and 2 columns.
The windows are filled up row-wise.
Use mfcol instead of mfrow to fill up column-wise.

> data(trees)
> par(mfrow=c(2,2))
> hist(trees$Height)
> boxplot(trees$Height)
> hist(trees$Volume)
> boxplot(trees$Volume)
> par(mfrow=c(1,1))
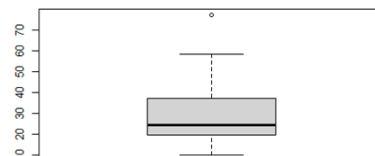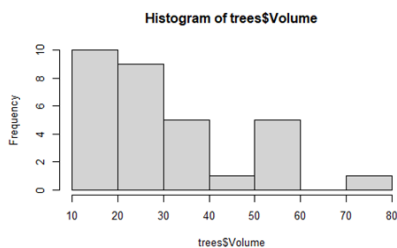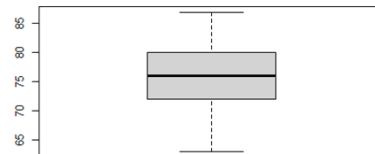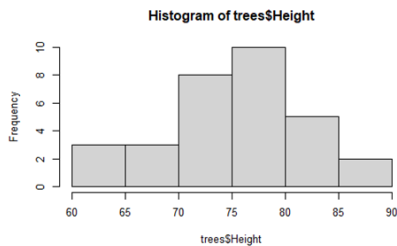
- Use layout()
Example: layout(matrix(1:4,2,2)) will partition the window into 4 equal parts
One can view the layout with layout show (n = 4)

A **scatter plot** provides a graphical view of the relationship between two sets of numbers.

> plot(trees$Height, trees$Volume, xlab="Height", ylab="Volume", main="Scatter Plot", pch=20)
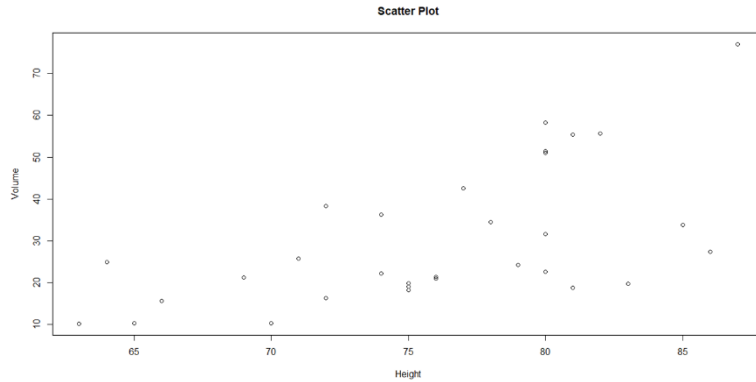
parameter pch stands for 'plotting character'.

> pairs(trees)

A matrix of scatterplots is produced.
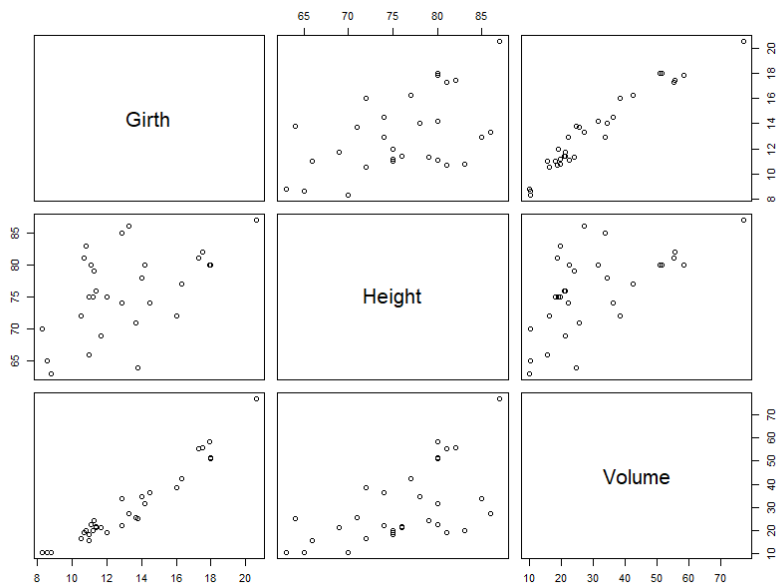
Density plot is a representation of the distribution of a numeric variable that uses a kernel density estimate to show the probability density function of the variable.

In R Language we use the density() function which helps to compute kernel density estimates.

```
> plot(density(gtemp), ylim=c(0, 2), col = "green",main = "Density plot")
> lines(density(gtemp2), col="red")
> legend(0.5,1.5, cex=0.8, c("gtemp", "gtemp2"), col=c("green", "red"), lty=1:1)
```
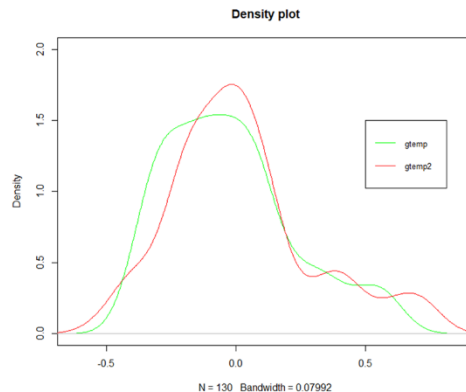

Writing functions

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions such as seq(), mean(), max(), sum(), etc. The user can create their own functions.

General form of the function:

```
func_name <- function(arg1, arg2, ...) {
Function body
}
```

func_name is the name of actual name of function.

The argument can be any type of object (like a scalar, a matrix, a data frame, a vector, a logical, etc)

Local vs global environment

It's not necessarily to use return() at the end of your function. The reason you return an object is if you've saved the value of your statements into an object inside the function. In this case, the objects in the function are in a local environment and won't appear in your global environment.

```
fun1 <- function(x){
  2*x+3
}
```

```
> fun1(4)
[1] 11

fun2 <- function(x){
  y <- 2*x+3
}
> fun2(4)
> print(y)
Error in print(y) : object 'y' not found
```

We can return the value of y using return(y) at the end of the function.

```
fun2_1 <- function(x){
  y <- 2*x+3
  return(y)
}
> fun2_1(4)
[1] 11

fun3 <- function(x, y){
  z1 <- 2*x+y
  z2 <- x+2*y
  z3 <- 2*x+2*y
  z4 <- x/y
  return(c(z1, z2, z3, z4))
}
> fun3(1, 2)
[1] 4.0 5.0 6.0 0.5
```

If we need to return multiple objects from a function, we can use list() to list them together. To extract objects from output, use [[ ]] operator.

```
fun4 <- function(x, y){
  m1 <- mean(x)
  m2 <- mean(y)
  sd1 <- sd(x)
  sd2 <- sd(y)
  cor.xy <- cor(x, y)
```

```
  xy <- cbind(x, y)
  list(m1, m2, sd1, sd2, cor.xy, xy)
}

> x <- c(1, 4, 8, 11, 20, 23)
> y <- c(2, 6, 3, 8, 21, 29)
> fun4(x, y)
[[1]]
[1] 11.16667
[[2]]
[1] 11.5
[[3]]
[1] 8.750238
[[4]]
[1] 10.96814
[[5]]
[1] 0.9471335
[[6]]
     x  y
[1,]  1  2
[2,]  4  6
[3,]  8  3
[4,] 11  8
[5,] 20 21
[6,] 23 29
```

for loops
-The for loop is used when iterating through a list.
-The basic structure of the for loop:
```
for(index in list){
 commands
}
```

```
cars <- c("Toyota", "Ford", "Chevy")
for(I in cars) {
  print(i)
```

```
}
[1] "Toyota"
[1] "Ford"
[1] "Chevy"
```

while loop

The while loop is used when you want to keep iterating as long as a specific
condition is satisfied. The basic structure of the while loop:

```
while(condition) {
  commands
}
i <- 3
while(i <= 6) {
 i <- i+1
 print(i)
}
[1] 4
[1] 5
[1] 6
[1] 7
```

Ifelse function

The ifelse function is very handy because it allows the user to specify the action
taken for the test condition being true or false. Like the if statement the ifelse
function can be included in any function or loop.

The basic structure of the ifelse function:

```
Ifelse(test, action.if.true, action.if.false)

> x <- seq(1:10)
> ifelse(x < 6, "T", "F")
[1] "T" "T" "T" "T" "T" "F" "F" "F" "F" "F"
```

R Packages for Bioinformatics

R packages are extensions to the R statistical programming language. R packages
contain code, data, and documentation in a standardised collection format that can
be installed by users of R. A large number of R packages are freely through CRAN

(the Comprehensive R Archive Network; https://cran.r-project.org/) and Bioconductor set of R packages (www.bioconductor.org). Some well-known bioinformatics R packages are the Bioconductor set of R packages (www.bioconductor.org). Bioconductor is a free, open source and open development software project for the analysis and comprehension of genomic data.

R Packages for analysis of biological sequence analysis and retrieval of genomic data
- seqinr
- tidysq
- biomartr
- rentrez

R packages for sequence alignment
- Biostrings
- msa
- msaR
- ggmsa
- AlignStat

R Packages for differential gene expression analysis of microarray data
- amda
- maGUI
- maEndToEnd
- limma
- GEOlimma

R packages for differential gene expression analysis of RNA-Seq data
- edgeR
- DESeq2
- ideal
- DEvis

R Packages for protein structure analysis
- Bio3D
- Rpdb

- XLmap

R packages for protein-protein interaction graphs
- graph
- RBGL
- Rgraphviz
- crosstalkr
- igraph

R Packages for proteomics data analysis
- RforProteomcs
- protti
- Proteus
- DanteR
- MSstats
- MSqRob
- DAPAR

R Packages for metagenomics data analysis
- MicrobiomeExplorer
- matR
- MegaR

R Packages for GWAS and genomic selection
- statgenGWAS
- GWASTools
- BlueSNP
- rrBLUP
- lme4GS
- BWGS
- GSelection
- learnMET
- GAPIT

Demonstration of an R package "GAPIT: Genomic Association and Prediction Integrated Tool"

GAPIT implemented a series of methods for Genome Wide Association (GWAS) and Genomic Selection (GS). The GWAS models include

- General Linear Model (GLM)
- Mixed Linear Model (MLM or Q+K)
- Compressed MLM (CMLM)
- Enriched CMLM
- SUPPER
- Multiple Loci Mixed Model (MLMM)
- FarmCPU
- BLINK

The GS models include

- gBLUP
- Compressed BLUP
- SUPER BLUP

GAPIT is an R package which can be freely downloaded from http://www.r-project.org or http://www.rstudio.com.

There are two sources to install GAPIT package.

Zhiwu Zhang Lab website

source("http://zzlab.net/GAPIT/GAPIT.library.R")

source("http://zzlab.net/GAPIT/gapit_functions.txt")

GitHub

install.packages("devtools")

devtools::install_github("jiabowang/GAPIT3",force=TRUE)

library(GAPIT3)

Help manual: https://zzlab.net/GAPIT/gapit_help_document.pdf

Import data from Zhiwu Zhang Lab

myY <- read.table("http://zzlab.net/GAPIT/data/mdp_traits.txt", head = TRUE)

myGD=read.table(file="http://zzlab.net/GAPIT/data/mdp_numeric.txt",head=T)

myGM=read.table(file="http://zzlab.net/GAPIT/data/mdp_SNP_information.txt", head=T)

GWAS

myGAPIT=GAPIT(

  Y=myY[,c(1,2,3)], #fist column is ID

  GD=myGD,

```
GM=myGM,
PCA.total=3,
model=c("FarmCPU", "Blink"),
Multiple_analysis=TRUE)
```

References

Giorgi, F. M., Ceraolo, C. and Mercatelli, D. (2022). The R Language: An Engine for Bioinformatics and Data Science. *Life (Basel, Switzerland)*, **12(5)**, 648. https://doi.org/10.3390/life12050648

Ihaka, R. and Gentleman, R (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, **5**, 299–314. doi: 10.1080/10618600.1996.10474713

W. N. Venables, D. M. Smith and the R Core Team. *An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics,* Version 4.2.2 (2022-10-31), URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

https://en.wikipedia.org/wiki/R_(programming_language)

https://en.wikipedia.org/wiki/Bioconductor

https://www.cran.r-project.org/

https://www.bioconductor.org/

Overview of Genomic Selection Methods

Neeraj Budhlakoti and D. C. Mishra

ICAR- Indian Agricultural Statistics Research Institute, New Delhi, India

Abstract

Since the inception of the theory and conceptual framework of genomic selection (GS), extensive research has been done on evaluating its efficiency for utilization in crop improvement. Though marker-assisted selection has proven its potential for improvement of qualitative traits that are controlled by one to few genes with large effects, its role in improving quantitative traits that are controlled by several genes with small effects is limited. In this regard, GS that utilizes genomic-estimated breeding values of individuals obtained from genome-wide markers to choose candidates for the next breeding cycle is a powerful approach to improve quantitative traits. In the past 20 years, GS has been widely adopted in animal breeding programs globally because of its potential to improve selection accuracy, minimize phenotyping, reduce cycle time and increase genetic gains. Improved statistical models that leverage the genomic information to increase the prediction accuracies are critical for the effectiveness of GS-enabled breeding programs.

Keywords: *GEBVs, GS, LD, MAS, QTL, SNP.*

Introduction

As it is known earlier selection based on phenotypic data has been successfully used in past. As abundance of DNA and marker data, trend slightly shifted to marker assisted selection (MAS). MAS is an indirect selection process where a trait of interest is selected, not based on the trait itself, but on a marker linked to it. MAS has been shown to be efficient and effective for traits that are associated with one or a few major genes with large effect but does not perform as well when it is used for selection of polygenic traits (Bernardo 2008).As most economic traits are influenced by many genes, tracking a small number of these through DNA markers will only explain a small proportion of the genetic variance. In addition, individual genes are likely to have small effects and so a large amount of data is needed to accurately estimate their effects. To overcome these difficulties, Meuwissen et al. (2001) proposed a variant of MAS that they called genomic selection. The key features of this method are that markers covering the whole genome are used so that potentially all the genetic variance is explained by the markers and the markers are assumed to be in linkage disequilibrium (LD)

with the Quantitative trait loci (QTL), so that the number of effects per QTL to be estimated is small.

Any successful GS program, starts with forming a training population in such a way that individuals/lines/variety are genotyped for genomic markers distributed over entire genome and should be representative of whole population. The training individuals are further subjected to extensive phenotyping for underlying trait of interest. The information of individual genotype and phenotype is used for identification and building of suitable statistical model using phenotype as a response and genotype as independent variable whereas part of training data can also be used for validation of fitted model. Genomic Estimated Breeding Values (GEBVs) of the individuals of the breeding population (where only information of genotyped individuals is available with no phenotypic records) is being calculated using their genotyped information where marker effect are estimated from developed model. Ultimately individuals/line/variety from the breeding population can be selected based on superiority of their estimated value of GEBVs.
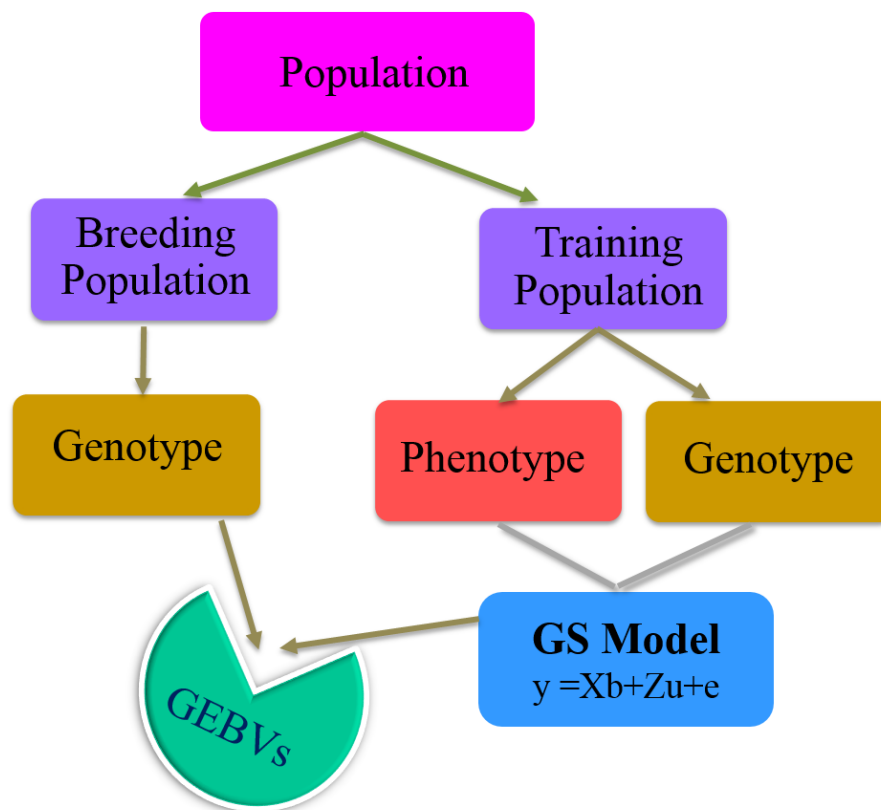
Fig. 1: Basic schema of genomic selection process

The major limitation to the implementation of genomic selection has been the large number of markers required and the cost of genotyping these markers are very high. Recently both these limitations have been overcome in most livestock and plant species following the sequencing of the livestock genomes, the subsequent availability of hundreds of thousands of single nucleotide polymorphisms (SNP), and dramatic improvements in development of SNP genotyping technology. Various regression methods have been developed for predicting phenotype. Methods are based on analysis of data consist of genotype and phenotype information. These methods are primarily based on linear models, which are easy to interpret and able to fit to the data without over fitting. However, the relationship between breeding value and genetic markers is likely to be more complex than a simple linear relationship, particularly when large numbers of SNPs are fitted simultaneously in the model. To answer these issues, model-free or so-called nonparametric methods which side-step linearity and require lesser genetic assumptions have gained more attention (Gianola et al, 2006).

Statistical model for Genomic Selection

Process of selecting the suitable individuals in GS starts with a simple linear model sometime also called as least squares regression or ordinary least squares regression (OLS).

$$Y = 1_n \mu + X\beta + \varepsilon$$

where, $\mathbf{Y} = n \times 1$ vector of observations; μ is the mean; $\boldsymbol{\beta} = p \times 1$ vector of marker effects; $\varepsilon = n \times 1$ vector of random residual effects; $\boldsymbol{X} =$ design matrix of order $n \times p$ (where each row represents the genotype/individuals/lines (n) and column corresponds to marker (p)), $\varepsilon \sim N(0, \sigma_e^2)$.

One major problem in linear models using several thousands of genome-wide markers is that number of markers (p) exceed the number of observations (n) i.e. genotype/individuals/lines and this creates the problem of over-parameterization (large 'p' and small 'n' problem (p>>n)). Using a subset of the significant markers can be an alternative for dealing with large 'p' and small 'n' problem. Meuwissen et al. (2001) used a modification of the least squares regression for GS. They performed least squares regression analysis on each maker separately with following model

$$Y = X_j \beta_j + \varepsilon$$

where,

$$X_j = j^{th} \text{column of the design matrix of marker}$$
$$\beta_j = \text{genetic effect of } j^{th} \text{ marker}$$

Marker with significant effects are selected using the log likelihood of this model and those are further used for estimation of breeding values. However, it has to be noted that some crucial or key information may be lost by selection based on subset of markers.

Hence, an efficient solution for the over-parameterization problem in linear models is using ridge regression (RR), which is a penalized regression-based approach (Meuwissen et al., 2001). It also solves the problems of multicollinearity at the same time (i.e. correlated predictors e.g. SNP or markers). RR shrinks the coefficients of correlated predictors equally towards zero and solves the regression problem using ℓ2 penalized least squares. Here, the goal is to derive an estimator of parameter β with smaller variance than the least square estimator. Similar to RR, least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996; Usai et al., 2009) is other variant of penalized regression, which uses the ℓ_1 penalized least squares criterion to obtain a sparse solution. LASSO sometime may not work well highly correlated predictors (e.g. SNPs in high linkage disequilibrium) (Ogutu et al., 2012). The elastic net (ENET) is an extension of the lasso that is robust to extreme correlations among the predictors (Friedman et al., 2010) and it is a compromise between ℓ1 penalty (lasso) and ℓ2 penalty (ridge regression) (Zou and Hastie, 2005).

The RR model considers that each marker contribute to equal variance, which is not the case for all traits. Therefore, the variance of the markers based on the trait genetic architecture has to be modeled. For this purpose, several Bayesian models have been proposed where it is assumed that there is some prior distribution of marker effects. Further, inferences about model parameters are obtained on the basis of posterior distributions of the marker effects. There are several variants of Bayesian models for genomic prediction such as Bayes A, Bayes B, Bayes Cπ and Bayes Dπ (Meuwissen et al., 2001; Habier et al., 2011) and other derivatives e.g. Bayesian LASSO, Bayesian ridge regression (BRR). Besides the marker-based models, the best linear unbiased prediction (BLUP), is one of the most commonly used genomic prediction method. There are many variants of BLUP available for this purpose e.g. genomic BLUP (GBLUP), single-step GBLUP (ssGBLUP), ridge regression BLUP (RRBLUP), GBLUP with linear ridge kernel regression (rrGBLUP), of which is GBLUP is very frequently used. While the BLUP has been used in other plant and animal breeding studies traditionally for various purposes (Henderson et al., 1959), the GBLUP uses the genomic relationships calculated using markers instead of the conventional pedigree-based BLUP which uses the pedigree relationships to obtain the GEBVs of the lines or individuals (Meuwissen et al., 2001).

The genomic prediction models discussed so far perform well for traits with additive genetic architecture but their performance becomes very poor in case of epistatic genetic architectures. Hence, Gianola et al. (2006) first used nonparametric and semiparametric methods for modeling complex genetic architecture. Subsequently, several statistical methods were implemented to model both main and epistatic effects for genomic selection (Xu, 2007; Cai et al., 2011; Legarra and Reverter, 2018). There are several nonparametric methods have been studied in relation to genomic selection e.g. NW (Nadaraya-Watson) estimator (Gianola et al., 2006), RKHS (Reproductive Kernel Hilbert Space) (Gianola et al., 2006), SVM (support vector machine) (Maenhout et al., 2007; Long et al., 2011), ANN (Artificial Neural Network) (Gianola et al., 2011) and RF (Random Forest) (Holliday et al., 2012) among them nonparametric methods SVM, NN and RF are based on machine learning approach.

Methods discussed earlier in this section are based on genomic information where information is available for single-trait i.e. single-trait genomic selection (STGS). As performance of STGS based methods may be affected significantly in case of pleiotropy i.e., one gene linked to multiple traits. A mutation in a pleiotropic gene may have an effect on several traits simultaneously. It was also observed that low heritability traits can borrow information from correlated traits and consequently achieve higher prediction accuracy can be achieved. Also STGS based methods considers the information of each trait independently. Hence we may lose crucial information which may ultimately result in poor genomic prediction accuracy. Now-a-days we are also getting data on multiple traits, so multi-trait genomic selection (MTGS) based methods may provide more accurate GEBVs and subsequently the higher prediction accuracy. Several MTGS based methods have been studied in relation to GS e.g. Multivariate mixed model approach (Jia and Jannink, 2012; Klápště et al., 2020), Bayesian multi-trait model (Jia and Jannink, 2012; Cheng et al., 2018), MRCE (Multivariate Regression with Covariance Estimation)(Rothman et al., 2010), cGGM (conditional Gaussian Graphical Models) (Chiquet et al., 2017). Jia et al. (2012) presented three multivariate linear models (i.e., GBLUP, Bayes A, and Bayes Cπ) and compared them to uni-variate models and a detailed comparison of various STGS and MTGS based methods has also been studied by Budhlakoti et al. (2019). A brief structure of different STGS and MTGS based methods used in GS studies are given in Fig. 2.
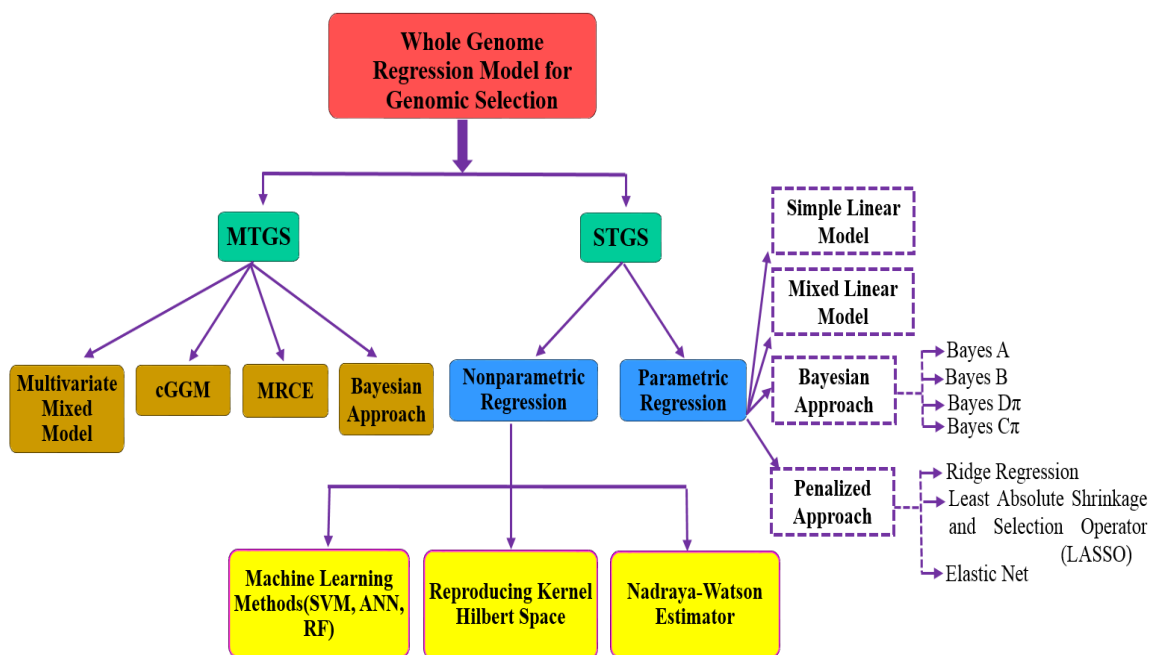
Fig. 2: Overall summary of the most commonly used models in Genomic Selection

Tools and packages to implement Genomic Selection

Several tools and packages have been developed for the evaluation of genomic prediction and implementation of GS, some of which are discussed below.

Tools/Packages	Description	URL	Reference
GMStool	It is a genome-wide association study (GWAS)-based tool for genomic prediction using genome-wide marker data	https://github.com/ JaeYoonKim72/GM Stool	Jeong et al. (2020)
rrBLUP	R package based on BLUP models its and other derivatives	https://CRAN.R-project.org/ package=rrBLUP	Endelman, (2011)
BWGS	It has a wide choice of totally 15 parametric and nonparametric statistical models for estimation of GEBV for selection candidates.	https://CRAN.R-project.org/package =BWGS	Charmet et al. (2020)
BGLR	This package is an extension of the BLR package (Perezand Campos, 2014) and can be used to implement several Bayesian models	https://CRAN.R-project.org/package =BGLR	Perez and Campos, (2014)
GenSel	Used for estimation of molecular marker–based breeding values of animals for trait under evaluation	https://github.com/ austin-putz/GenSel	Fernando and

			Garrick, (2009)
lme4GS	This package can be used for fitting mixed models with covariance structures with user defined parameter	https://github.com/perpdgo/lme4GS	Caamal-Pat et al. (2021)
GSelection	Package comprises of a set of functions to select the important markers and estimates the GEBV of selection candidates using an integrated model framework	https://CRAN.R-project.org/package=GSelection	Majumdar et al. (2019)
STGS	It is a comprehensive package which gives a single-step solution for genomic selection based on most commonly used statistical methods (i.e., RR, BLUP, LASSO, SVM, ANN, and RF).	https://CRAN.Rproject.org/package=STGS	Budhlakoti et al. (2019a)
MTGS	MTGS is a comprehensive package which gives a single-step solution for genomic selection using various MTGS-based methods (MRCE, MLASSO, i.e., multivariate LASSO, and KMLASSO, i.e., kernelized multivariate LASSO).	https://CRAN.R-project.org/package=MTGS	Budhlakoti et al. (2019)

Issues and challenges in genomic selection

Genomic selection is a powerful tool for plant and animal breeding, but it also presents a number of challenges and issues. Some of the key challenges and issues in genomic selection include:

1. Data quality and quantity: Genomic selection requires large amounts of high-quality genomic data. However, obtaining this data can be challenging, especially in species with complex genomes or limited genomic resources.

2. Genetic diversity: Genomic selection works best when there is a large amount of genetic diversity in the population. However, in some species, there may be limited genetic diversity, which can limit the effectiveness of genomic selection.

3. Phenotyping: In order to train genomic selection models, accurate and consistent phenotypic data is required. However, phenotyping can be time-consuming, expensive, and difficult to standardize.

4. Trait heritability: The effectiveness of genomic selection depends on the heritability of the trait being selected. Some traits may have low heritability, making it difficult to accurately predict their values using genomic data.

5. Statistical model used: The choice of statistical model used in genomic selection is important because it can impact the accuracy of the predictions and the efficiency of the analysis. Some of the key concerns related to the type of statistical model used in genomic selection include:

    i. Overfitting: Overfitting can occur when a model is too complex for the data, leading to high accuracy in the training set but poor performance on new data. This can be a concern in genomic selection, particularly when using models with a large number of parameters or when the sample size is small.

    ii. Model assumptions: Different statistical models have different assumptions about the data, and violating these assumptions can lead to biased or inaccurate predictions. For example, linear regression assumes that the residuals are normally distributed and homoscedastic, and violating these assumptions can lead to poor performance.

    iii. Scalability: Some statistical models are computationally intensive and may not be feasible for very large datasets. This can be a concern in genomic selection, particularly as the amount of genomic data continues to grow.

    iv. Interpretability: Some statistical models are more interpretable than others, which can be important for understanding the biological basis of the trait being predicted. For example, linear regression models can provide insight into which genomic regions are associated with the trait, while more complex models may be more difficult to interpret.

    v. Incorporation of external information: Some statistical models can incorporate external information, such as gene annotation or pathway information, to improve predictions. However, the quality and relevance of this external information can impact the performance of the model.

6. Integration with traditional breeding: Genomic selection is most effective when it is integrated with traditional breeding methods. However, this can be challenging, especially in species with long breeding cycles or complex genetic architectures.

Conclusion and perspectives

Genomic selection has improved genetic gains in plant and animal breeding research over the past two decades. Advances in cheaper next-generation sequencing technologies have resulted in the availability of high-density SNP genotyping chips and completely sequenced crop and animal genomes, boosting the predictive ability of a genomic selection model. However, there is still scope for improvement in the methodology of genomic selection, such as imputation of missing genotypic value and implementation of GxE interaction, to successfully implement it in breeding programs. Regular updating of the training set and evaluation under controlled conditions is necessary for better performance. To achieve fruitful outcomes, a structured program is needed that includes human resource development, advanced data recording methodologies, and trait phenotyping.

Reference

Bernardo, R. (2008). Molecular markers and selection for complex traits in plants: Learning from the last 20 years. Crop Science 48, 1649–1664. doi:10.2135/CROPSCI2008.03.0131.

Budhlakoti, N., Mishra, D. C., Rai, A., Lal, S. B., Chaturvedi, K. K., and Kumar, R. R. (2019). A Comparative Study of Single-Trait and Multi-Trait Genomic Selection. Journal of Computational Biology 26, 1100–1112. doi:10.1089/CMB.2019.0032.

Budhlakoti, N, Mishra, D. C., Rai, A. and Chaturvedi, K.K. (2019a) Package 'STGS', 1-11.

Budhlakoti, N., Mishra, D. C., and Rai, A. (2019b). Package 'MTGS', 1–6.

Caamal-Pat, D., Pérez-Rodríguez, P., Crossa, J., Velasco-Cruz, C., Pérez-Elizalde, S., and Vázquez-Peña, M. (2021). lme4GS: An R-Package for Genomic Selection. Frontiers in Genetics 12, 982. doi:10.3389/FGENE.2021.680569/BIBTEX.

Cai, X., Huang, A., and Xu, S. (2011). Fast empirical Bayesian LASSO for multiple quantitative trait locus mapping. BMC Bioinformatics 12, 1–13. doi:10.1186/1471-2105-12-211/FIGURES/5.

Charmet, G., Tran, L. G., Auzanneau, J., Rincent, R., and Bouchet, S. (2020). BWGS: A R package for genomic selection and its application to a wheat breeding programme. PLOS ONE 15, e0222733. doi:10.1371/JOURNAL.PONE.0222733.

Cheng, H., Kizilkaya, K., Zeng, J., Garrick, D., and Fernando, R. (2018). Genomic prediction from multiple-trait Bayesian regression methods using mixture priors. Genetics 209, 89–103. doi:10.1534/GENETICS.118.300650/-/DC1.

Chiquet, J., Mary-Huard, T., St´, S., and Robin, S. (2017). Structured regularization for conditional Gaussian graphical models. Statistics and Computing 27, 789-804.

Endelman, J. B. (2011). Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. The Plant Genome 4, 250–255. doi:10.3835/PLANTGENOME2011.08.0024.

Fernando, R. and Garrick, D. (2009). GenSel- User Manual for a portfolio of Genomic Selection related Analyses. (http://taurus.ansci.iastate.edu/Site/Welcome_files/GenSel%20 Manual%20v2.pdf)

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of statistical software 33, 1.

Gianola, D., Fernando, R. L., and Stella, A. (2006). Genomic-Assisted Prediction of Genetic Value With Semiparametric Procedures. Genetics 173, 1761. doi:10.1534/GENETICS.105.049510.

Gianola, D., Okut, H., Weigel, K. A., and Rosa, G. J. M. (2011). Predicting complex quantitative traits with Bayesian neural networks: a case study with Jersey cows and wheat. BMC Genet. 12, 87. doi:10.1186/1471-2156-12-87.

Habier, D., Fernando, R. L., Kizilkaya, K., and Garrick, D. J. (2011). Extension of the bayesian alphabet for genomic selection. BMC Bioinformatics 12, 1–12. doi:10.1186/1471-2105-12-186/FIGURES/2.

Henderson, C. R., Kempthorne, O., Searle, S. R. and von Krosigk, C. M. (1959). The estimation of environmental and genetic trends from records subject to culling. Biometrics, 15: 192.

Holliday, J. A., Wang, T., and Aitken, S. (2012). Predicting Adaptive Phenotypes From Multilocus Genotypes in Sitka Spruce (Picea sitchensis) Using Random Forest. doi:10.1534/g3.112.002733.

Jeong, S., Kim, J. Y., and Kim, N. (2020). GMStool: GWAS-based marker selection tool for genomic prediction from genomic data. Scientific Reports 10, 1–12. doi:10.1038/s41598-020-76759-y.

Jia, Y., and Jannink, J. L. (2012). Multiple-Trait Genomic Selection Methods Increase Genetic Value Prediction Accuracy. Genetics 192, 1513. doi:10.1534/GENETICS.112.144246.

Klápště, J., Dungey, H. S., Telfer, E. J., Suontama, M., Graham, N. J., Li, Y., et al. (2020). Marker Selection in Multivariate Genomic Prediction Improves Accuracy of Low Heritability Traits. Front. Genet. 11, 499094. doi:10.3389/FGENE.2020.499094/FULL.

Legarra, A., and Reverter, A. (2018). Semi-parametric estimates of population accuracy and bias of predictions of breeding values and future phenotypes using the LR method 01 Mathematical Sciences 0104 Statistics. Genetics Selection Evolution 50, 1–18. doi:10.1186/S12711-018-0426-6/FIGURES/3.

Long, N., Gianola, D., Rosa, G. J. M., and Weigel, K. A. (2011). Application of support vector regression to genome-assisted prediction of quantitative traits. Theor. Appl. Genet. 123, 1065–1074. doi:10.1007/S00122-011-1648-Y.

Maenhout, S., De Baets, B., Haesaert, G., and Van Bockstaele, E. (2007). Support vector machine regression for the prediction of maize hybrid performance. Theor. Appl. Genet. 115, 1003–1013. doi:10.1007/s00122-007-0627-9.

Majumdar, S. G., Rai, A., and Mishra, D. C. (2019). Package 'GSelection', 1–14.Available at: https://rdrr.io/cran/GSelection/man/GSelection-package.html.

Meuwissen, T. H. E., Hayes, B. J., and Goddard, M. E. (2001). Prediction of total genetic value using genome-wide dense marker maps. Genetics 157, 1819–1829. doi:10.1093/GENETICS/157.4.1819.

Ogutu, J. O., Schulz-Streeck, T., and Piepho, H. P. (2012). Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. BMC Proc. 6, S10. doi:10.1186/1753-6561-6-S2-S10.

Perez, P., and Campos, G. (2014). BGLR : A Statistical Package for Whole Genome Regression and Prediction. Genetics 198, 483–495.

Rothman, A. J., Levina, E., and Zhu, J. (2010). Sparse Multivariate Regression With Covariance Estimation. J. Comput. Graph. Stat. 19, 947. doi:10.1198/JCGS.2010.09188.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of Royal Statistical Society, 58: 267–288.

Usai, M. G., Goddard, M. E., and Hayes, B. J. (2009). LASSO with cross-validation for genomic selection. Genet. Res. (Camb). 91, 427–436. doi:10.1017/S0016672309990334.

Xu, S. (2007). An Empirical Bayes Method for Estimating Epistatic Effects of Quantitative Trait Loci. Biometrics 63, 513–521. doi:10.1111/J.1541-0420.2006.00711.X.

Zou, H., and Hastie, T. (2005). Regularization and variable selection via the elastic net. J. R. Stat. Soc. B 67, 301–320.

Introduction to Genome-Wide Association Studies

Soumya Sharma

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Genome-wide association study (GWAS) is a research strategy to find genetic variations that are statistically linked to a disease or a particular trait. The approach involves scanning the genomes of a large number of individuals in search of genetic variants that are more prevalent in persons with a particular disease or trait than in people without the disease or trait. These genomic variants are often utilised to look for neighbouring variants that are directly responsible for the disease or trait once they have been found.

Linkage disequilibrium (LD) between the markers being studied and the functional polymorphisms of the causal genes is the basis for GWAS. On the chromosome, loci that are physically close to one another are separated by recombination less frequently than loci that are farther apart. Gametic-phase disequilibrium, often known as LD, is the nonrandom connection of alleles at two loci. The SNPs close to the causal locus may have strong LD with the functional polymorphisms and hence be linked to the desired trait. These relationships are discovered through genome-wide association studies, which also highlight the genomic areas that contain the significant SNPs and the relevant genes.

Genome-wide association study (GWAS) attempts to predict association of specific traits (phenotype) with genetic variants (genotype) by statistical analysis at population level. Phenotypic information can be obtained by systematically measuring the phenotype (physical and physiological traits) that can be influenced by various genetic and environmental factors. Individual genotyping is usually done with microarrays for common variations or next-generation sequencing technologies like WES or WGS for rare variants. Due to the current expense of next-generation sequencing, microarray-based genotyping is the most frequently used approach for retrieving genotypes for GWAS. However resequencing the entire genome has the ability to uncover almost all genetic variations. This genotypic information along with phenotypic data can be analysed to identify the genetic markers (SNPs, SSRs etc.), QTLs or candidate genes associated with a specific trait.

The input files for GWA studies usually include the genotype file i.e., marker information and the phenotype file i.e., trait information and also coded family relations between individuals. Following the data input, producing reliable GWAS results requires meticulous quality control.

Testing for associations

The biometrical model underpins the genetic association theory. Depending on whether the phenotype is continuous (such as plant height, grain yield etc.) or binary (such as the presence or absence of disease), linear or logistic regression models are typically employed in GWAS to test for associations. To account for stratification and eliminate confounding effects from demographic characteristics, covariates such as age, sex, and ancestry are added, with the caveat that this may impair statistical power for binary traits in ascertained samples. Adding an additional individual-specific random effect term to linear or logistic mixed models to account for genetic relatedness among individuals might improve statistical power for genome discovery and boost control for stratification at the expense of increased complexity. Adding an additional individual-specific random effect term to linear or logistic mixed models to account for genetic relatedness between people might boost statistical power for genome discovery and increase control for stratification at the cost of more processing resources. When doing a GWAS, it's important to remember that genotypes of genetic variants that are physically close together aren't independent because they are in linkage disequilibrium; this test dependency should be taken into account as well.

The following equation depicts the linear regression model for testing the association between a marker and a trait:

$$Y \sim X\alpha + Z_s\beta_s + e$$
$$e \sim N(0, \sigma_e^2 I)$$

where, for each individual, Y is a vector of phenotype values, X is a matrix assigning records to phenotypes fixed effect, α is a corresponding vector of fixed effects sizes (e.g., the mean, population structure effects, and age), Z_s is a vector of genotype values for all individuals at genetic variations, β_s is the corresponding fixed effect size of genetic variants, σ_e^2 measures residual variance and I is an identity matrix.

The underlying assumption is that if the marker will have effect on trait only if it is in linkage disequilibrium with an unseen QTL. The null hypothesis for the study asserts that marker has no effect on the trait, while the alternative hypothesis states that it does have an effect on the trait (as it is in LD with a QTL). If F > $F_{\alpha;v1;v2}$ where F is the F statistic obtained from the data

and $F_{\alpha;\nu1;\nu2}$ is the value from a F distribution at α level of significance and $\nu1$, $\nu2$ degrees of freedom, the null hypothesis is rejected.

There are numerous statistical models to find associations between marker loci and a variety of traits, ranging from simple to highly complex. Accurate decoding of complex traits in diverse population requires more comprehensive statistical models which takes care of false positives arising from family relatedness and population structure, at the same time also keeps in check the number of false negatives due to over correction. Confounding effects due to population structure and kinship among individuals is taken into account by using these covariates in the statistical model. STRUCTURE (Pritchard et al., 2000), PCA (Price et al., 2006), and a discriminant analysis of principal components (DAPC) (Jombart et al., 2010) are methods for determining population organisation by using genetic markers. False positives arising due to common ancestry and family relatedness can be addressed by incorporating kinship matrix into the statistical model. One of the most often used methods for estimating family relatedness among individuals in a diverse population is identity-by-state (Loiselle et al., 1995).

Inclusion of population structure and a kinship matrix as covariates in mixed linear models (MLM) to reduce false positives is a widely used approach. Many MLM-based approaches have been presented since Yu et al. (2006) published the first MLM of association mapping (Zhang et al., 2010; Wang et al., 2014). All of these models are called single-locus models as they do a unidimensional genome scan by examining one marker at a time and then iterate the process for each marker in the dataset. But the true genetic model of complex traits that are governed by multiple loci at the same time cannot be explained by single locus models. Multilocus association mapping models have been suggested as a solution to this problem since they consider the input from all loci at the same time (Wang et al., 2016). One more constraint of MLM based models is increase in number of false negatives due to overfitting which may lead to omission of certain potentially valuable association (Liu et al., 2016). False negatives may arise during multiple comparison adjustments for evaluating statistical significance. Bonferroni correction (Holm, 1979) and false discovery rate (FDR) (Benjamini and Hochberg, 1995) are two commonly used multiple comparison approaches in association mapping for determining the significant threshold. Highly conservative standards can result in a high rate of false negatives. As a result, selection of a proper model and threshold are critical steps in detecting true trait associated markers that may be located inside or in high LD with genes that govern trait variation, while minimizing both false-positive and false-negative associations.

Statistical models for GWAS

Some popular models for GWAS include:

(1) analysis of variance (ANOVA)

(2) general linear model with principle component analysis (GLM + PCA) (Price et al., 2006),

(3) MLM with principle component analysis and Kinship matrix for family relatedness estimates (GLM+PCA+K) (Yu et al., 2006)

(4) compressed MLM (Zhang et al., 2010)

(5) enriched compressed MLM (Li et al., 2014)

(6) settlement of MLM under progressively exclusive relationship (SUPER) (Wang et al., 2014)

(7) multiple loci MLM (MLMM) (Segura et al., 2012)

(8) fixed and random model circulating probability unification (FarmCPU) (Liu et al., 2016). Models from (1) to (6) are single locus models, while (7) and (8) are multilocus models.

Among these popular models of GWAS, the GLM and MLM are said to have a better control of false positives than ANOVA (Price et al., 2006; Yu et al., 2006). The GLM with PCA model is supposed to lower the number of false positives caused by population structure alone (Price et al., 2006). The kinship matrix is included in the MLM with PCA and K model, which is intended to reduce false positives caused by family relatedness (Yu et al., 2006). By controlling false positives, the MLM model is said to perform better than the GLM model alone (Yu et al., 2006). The benefit of MLM model in controlling false positives disappears when complex qualities are connected with population structure with considerable genetic divergence, The MLM approach does a good job of controlling P-value inflation, but it also produces false negatives, making it difficult to identify actual correlations (Zhang et al., 2010). The compressed MLM model (CMLM), which clusters individuals into groups and fits genetic values of groups as random effects in the model, was created to address this challenge (Zhang et al., 2010). When compared to traditional MLM methods, the CMLM method boosts statistical power (Zhang et al., 2010). Another option for dealing with P-value deflation caused by MLM is to adopt a SUPER model, in which just the linked genetic markers are utilised as pseudo–Quantitative Trait Nucleotides (QTNs) to determine kinship, rather than all of the markers (Wang et al., 2014). When a pseudo QTN is associated with the testing marker, it is not included in the kinship analysis. Between the pseudo QTNs and the testing marker, the SUPER model applies an LD threshold. When compared to using total kinship from all

markers, this strategy improves statistical power. FarmCPU is a multilocus model that was created to reduce false positives while keeping false negatives to a minimum (Liu et al.,2016). To partially minimise the confusion between testing markers and kinship, the FarmCPU model use a modified MLM method called multiple loci linear mixed model (MLMM), which combines many markers simultaneously as covariates in a stepwise MLM. When compared to other models, this model is said to improve statistical power, computing efficiency, and the capacity to control false positives and false negatives (Liu et al., 2016).

Single-locus models, such as the general linear model (GLM) and the mixed linear model (MLM) require multiple tests that undergo a Bonferroni correction (Bradbury et al., 2007) for multiple comparison adjustments. The typical Bonferroni correction is often too conservative, which results in many important loci associated with the target traits being eliminated because they do not satisfy the stringent criterion of the significance test. The multi-locus models are better alternatives for GWASs because they do not require the Bonferroni correction, and thus more marker-trait associations may be identified. Recently, several new multi-locus GWAS models, such as multi-locus RMLM (mrMLM, Wang et al., 2016), fast multi-locus random-SNP-effect EMMA (FASTmrEMMA, Wen et al., 2017), and Iterative modified-Sure Independence Screening EM-Bayesian LASSO (ISIS EM-BLASSO, Tamba et al., 2017), have been developed.

Representation of GWAS Results

GWAS results are typically represented as two types of p-value plots: genome-wide association plots (Manhattan plots) and quantile-quantile (QQ) plots. In Manhattan plot marker loci are represented as chromosomes and position on the chromosome in genomic order on x-axis and negative logarithm of their p values ($-\log_{10}P$) on y-axis (Fig1). The Manhattan plot resembles the Manhattan skyline because clusters of significant P values tend to ascend to the top due to local correlation of the genetic variants brought on by linkage.
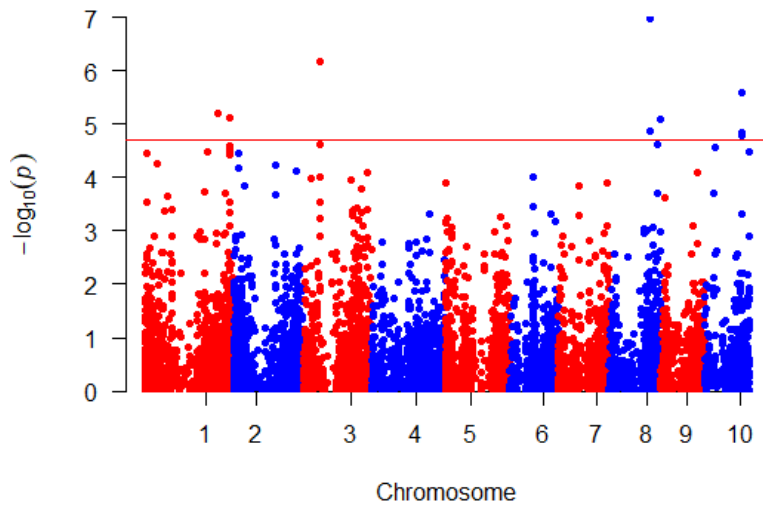
Fig 1: An illustration of a Manhattan plot depicting several strongly associated loci to the trait

Quantile-quantile plots (QQ plots) are widely used to display the proportion of significant results in relation to the projected number of significant results at a specific P value (Fig 2). The figure unambiguously demonstrated that, at levels more than P 0.001, more significant SNP were discovered in their analysis than would have been expected by chance.
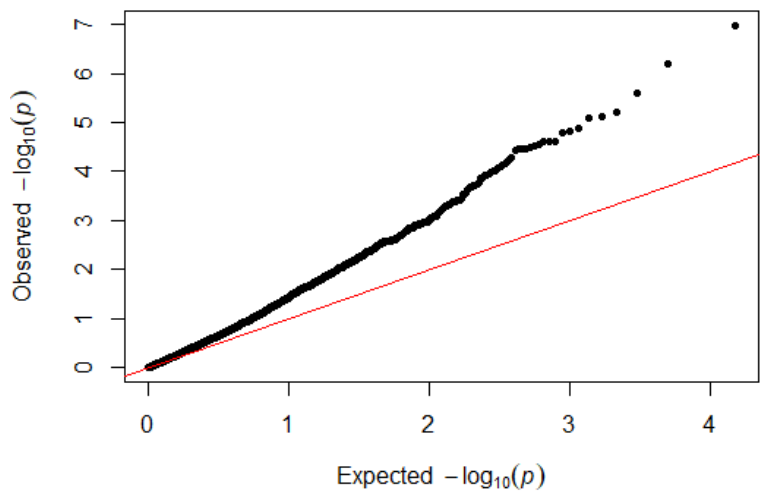
Fig 2: Quantile-quantile (QQ) plot. Comparison of GWAS P-values (black dotted line) to those expected for a null distribution (red line).

References:

Benjamini, Y., Hochberg, Y. (1995). Controlling the false discovery rate, A practical and powerful approach to multiple testing. *J. R. Stat. Soc. Series. B. Stat. Methodol.* 57, 289–300. doi: 10.1111/j.2517-6161.1995.tb02031.x

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., et al. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nat. Genet.* 38 (8), 904–909. doi: 10.1038/ng1847

Zhang, Z., Ersoz, E., Lai, C. Q., Todhunter, R. J., Tiwari, H. K., Gore, M. A., et al. (2010). Mixed linear model approach adapted for genome-wide association studies. *Nat. Genet.* 42, 355–360. doi: 10.1038/ng.546

Kaler, A. S., Gillman, J. D., Beissinger, T., & Purcell, L. C. (2020). Comparing different statistical models and multiple testing corrections for association mapping in soybean and maize. Frontiers in plant science, 10, 1794.

Yu, J., Pressoir, G., Briggs, W. H., Vroh, B. I., Yamasaki, M., Doebley, J. F., et al. (2006). A unifed mixed-model method for association mapping that accounts for multiple levels of relatedness. *Nat. Genet.* 38, 203–208. doi: 10.1038/ng1702

Liu, X., Huang, M., Fan, B., Buckler, E. S., Zhang, Z. (2016). Iterative usage of fixed and random effect models for powerful and efficient genome-wide association studies. *PLoS Genet.* 12 (2), e1005767. doi: 10.1371/journal.pgen.1005767

Li, M., Liu, X., Bradbury, P., Yu, J., Zhang, Y.-M., Todhunter, R. J., et al. (2014). Enrichment of statistical power for genome-wide association studies. *BMC Biol.* 12, 73. doi: 10.1186/s12915-014-0073-5

Wang, Q., Tian, F., Pan, Y., Buckler, E. S., Zhang, Z. (2014). A SUPER powerful method for genome wide association study. *PLoS ONE* 9, e107684. doi: 10.1371/journal.pone.0107684

Segura, V., Vilhjálmsson, B. J., Platt, A., Korte, A., Seren, Ü., Long, Q., et al. (2012). An efficient multi-locus mixed-model approach for genome-wide association studies in structured populations. *Nat. Genet.* 44, 825–830. doi: 10.1038/ng.2314

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* 6, 65–70.

Wen, Y. J., Zhang, H., Ni, Y. L., Huang, B., Zhang, J., Feng, J. Y., et al. (2018). Methodological implementation of mixed linear models in multi-locus genome-wide association studies. *Brief. Bioinform.* 19, 700–712. doi: 10.1093/bib/bbw145

Tamba, C. L., Ni, Y. L., Zhang, Y. M. (2017). Iterative sure independence screening EM-Bayesian LASSO algorithm for multi-locus genome-wide association studies. *PLoS Comput. Biol.* 13, e1005357. doi: 10.1371/journal.pcbi.1005357

Practical Exercise of GWAS
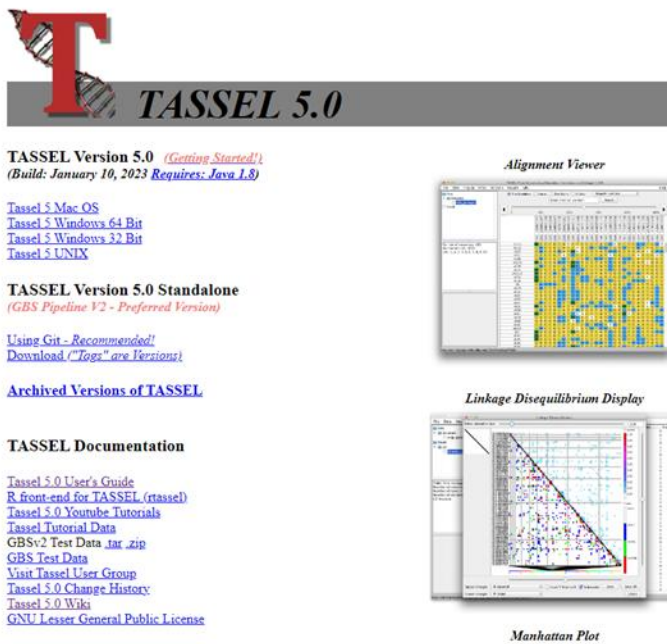
Soumya Sharma

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

TASSEL also known as Trait Analysis by aSSociation, Evolution and Linkage is a powerful statistical software to conduct association mapping such as General Linear Model (GLM) and Mixed Linear Model (MLM). The GUI (graphical user interface) version of TASSEL is very well built for anyone who does not have a background or experience in working in command line. The following section demonstrates how to prepare input files and run association analysis in TASSEL in stepwise manner.

1. Download and install TASSEL software

Download and install the latest version of the TASSEL software at this link: https://www.maizegenetics.net/tassel

2. Preparing the Input files

Phenotype file

Phenotype file can be prepared as shown below in the figure below

Please remember if your data has covariates such as sex, age or treatment, then, please categories them with header name factor.

Genotype file

TASSEL supports various genotype file formats such as VCF (variant call format), .hmp.txt, and plink. We are using the hmp.txt version of the genotype file for this demonstration. The below screenshot of the hmp.txt genotype file.
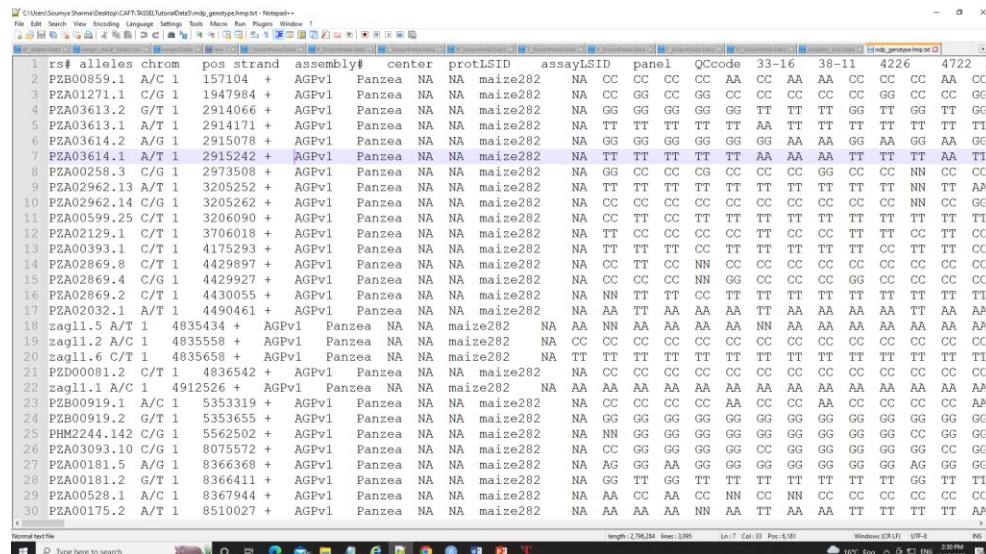
3. **Importing phenotype and genotype files**

Import the files by following the steps shown below.

Start Tassel -> go to "file" menu -> select "open" -> specify the "folder" where files are located -> choose the "files" to open holding CTRL button -> click on "open"
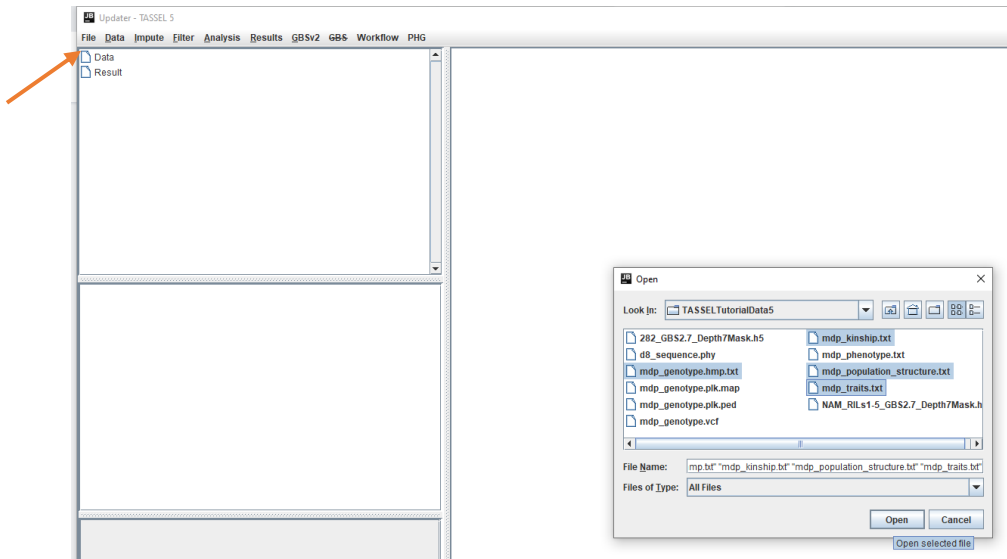
4. Phenotype distribution plot

It is always a wise idea to look at the phenotype distribution by plotting to check for any outliers.

Select the "phenotype" file -> go to "Results" -> go to "Charts" -> select graph type as "Histogram" -> select the trait under "Series 1"
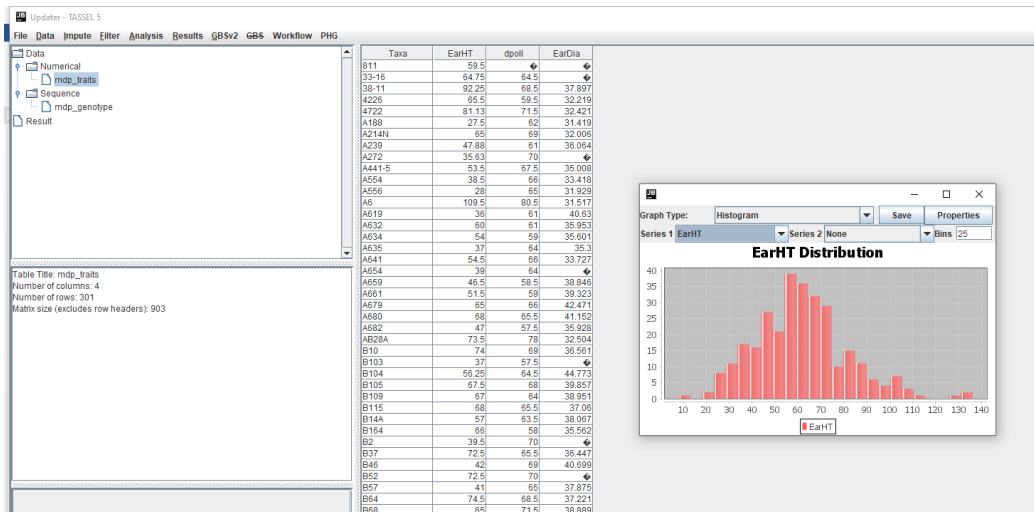
5. Genotype summary analysis

Next crucial step is to look at the genotype data by simply following the steps shown.

Select genotype data -> go to "Data" menu -> click "Geno Summary"

The output will be as shown in the figure below. The arrow depicts missing genotypic data to see if it requires to be imputed.
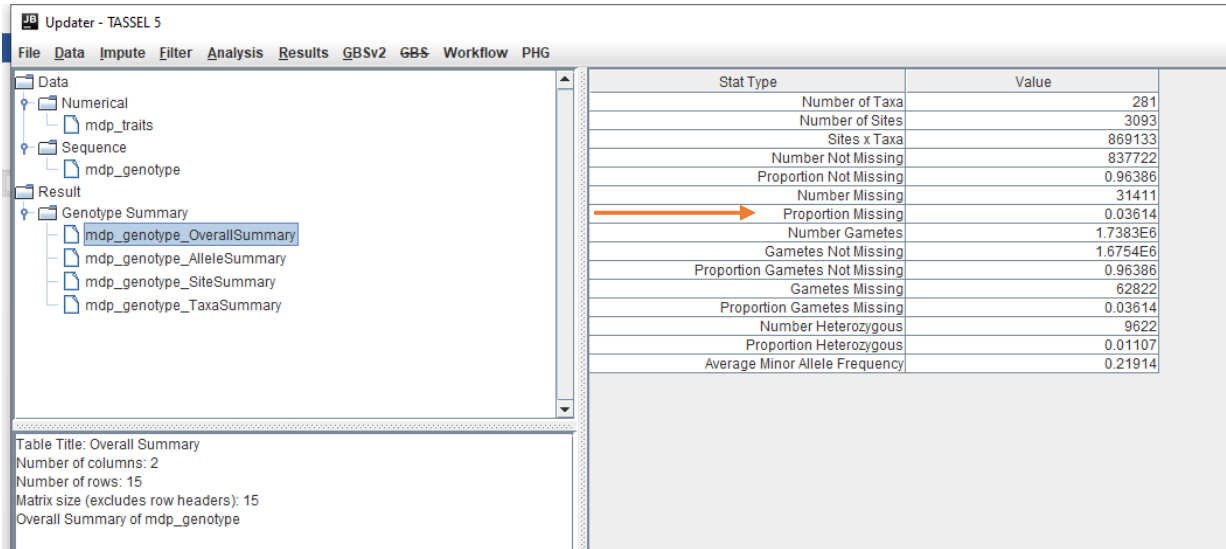
Minor allele frequency distribution

Select genotype _SiteSummary -> go to "Results" -> click on "Charts" -> select "Minor Allele Frequency" under "Series 1"
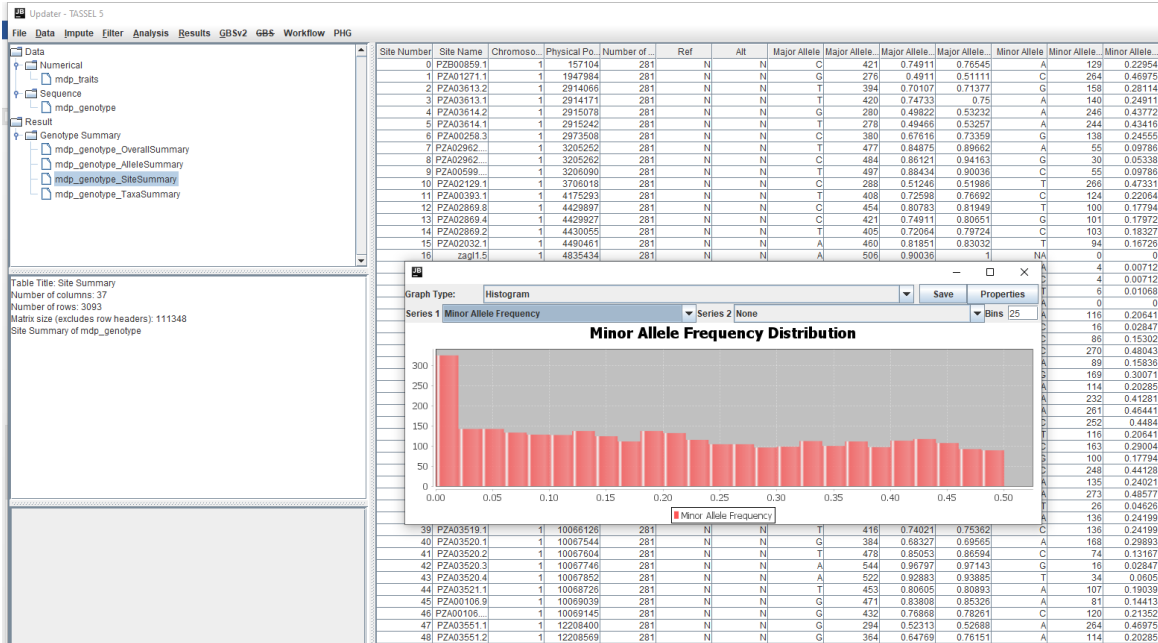
Proportion of heterozygous in the samples to check for selfed samples.

Select genotype_TaxaSummary -> go to "Results" -> click on "Charts" -> select "Proportion Heterozygous" under "Series 1"
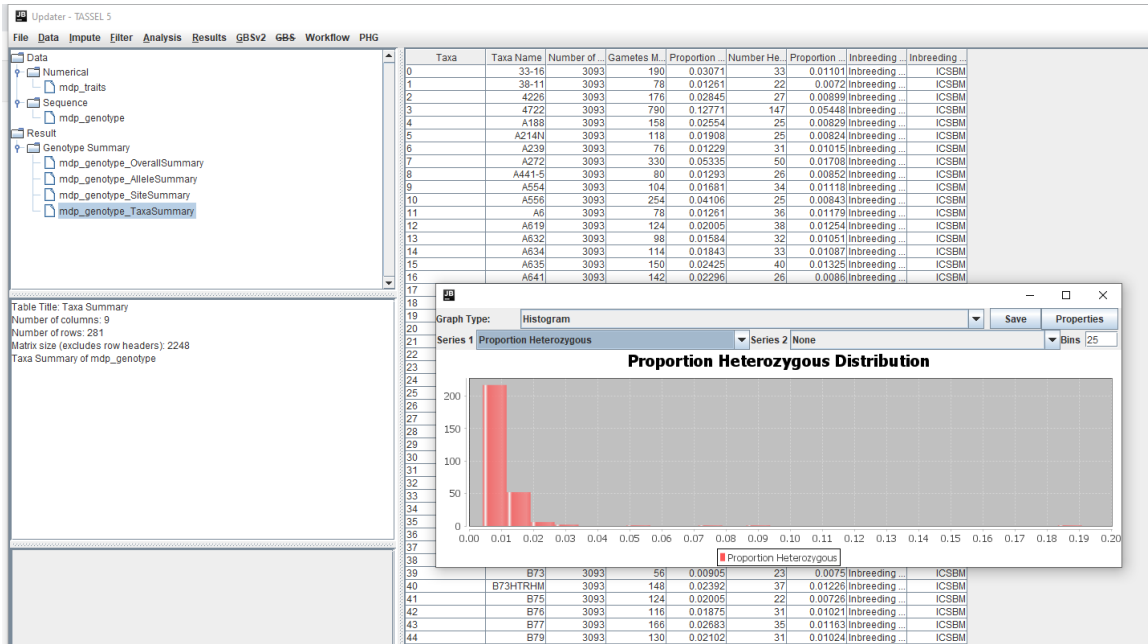
6. Imputation of missing values

Select genotype file -> go to "impute" -> click on "LD KNNi imputation" -> set parameters ->click "okay"
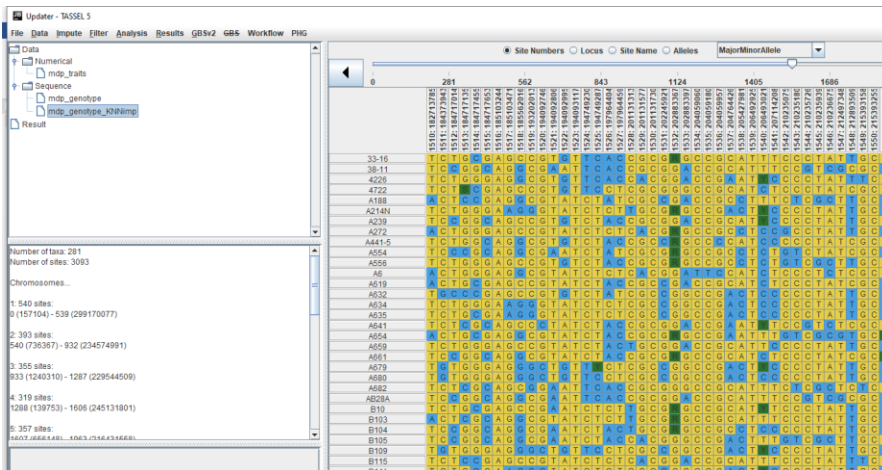
7. Filter Markers based on Minor allele frequency (MAF)

Steps to filter markers based on Minor allele frequency (MAF) are shown below:

0.05 Minor allele Frequency (set filter thresholds for rare alleles)

Select genotype file -> go to "filter" -> click on "Filter Genotype Table Sites" -> set parameters -> click "OK"
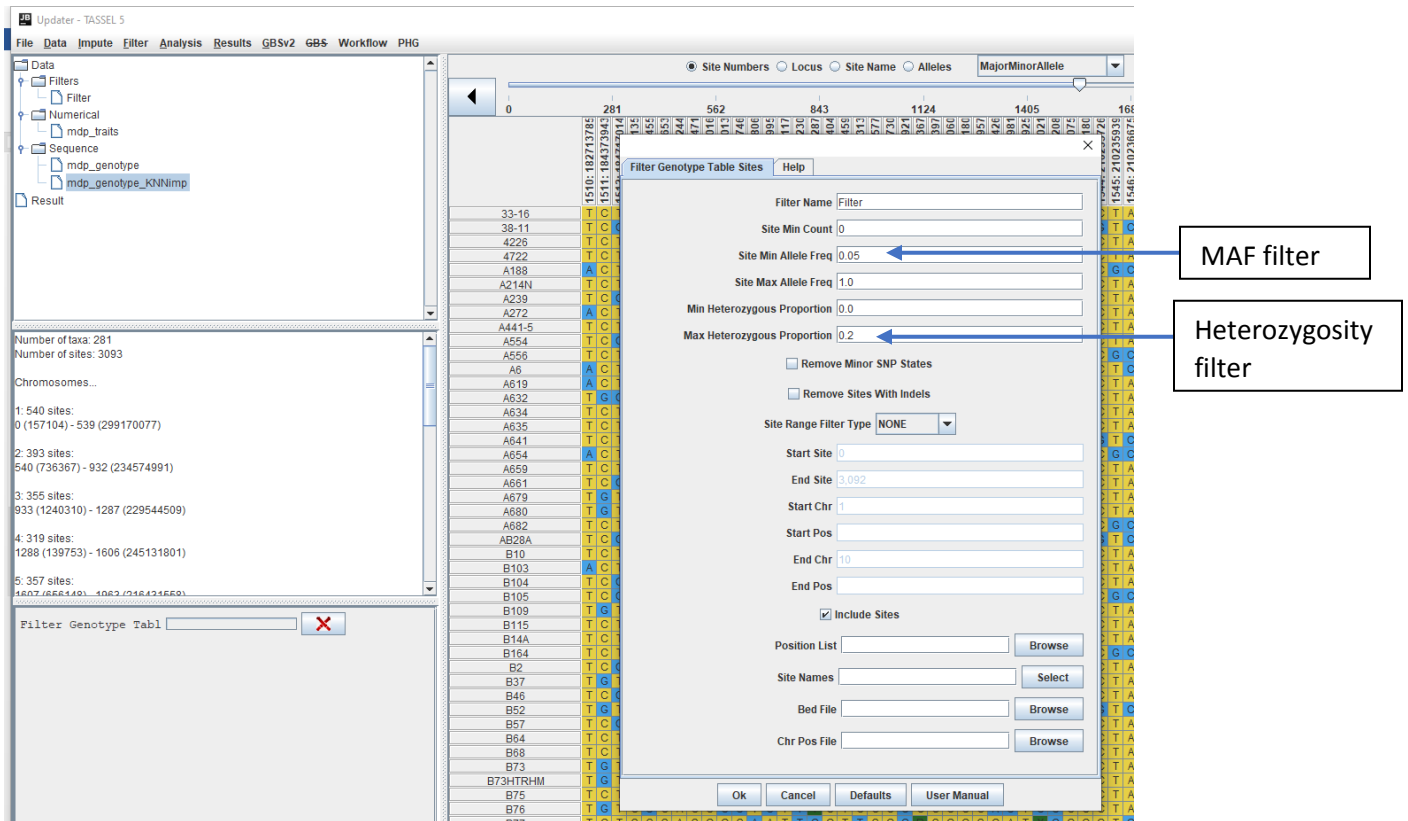
Conduct GWAS analysis

8. Principal component analysis (PCA)

PCA output can be used as the covariate in the GLM or MLM to correct for population structure. Please follow the steps shown below:

Select genotype file -> go to "Analysis" -> go to "Relatedness" -> click on "PCA"-> set parameters -> click "ok"

9. Intersecting the files

Intersect the genotype, phenotype and PCA files by following the steps below:

Select genotype, phenotype and PCA files simultaneously by holding 'CTRL' button -> go to "Data" -> click on "Intersect join"

10. Running General Linear Model (GLM)

Run the GLM analysis by selecting the intersected files following the steps below:

Select the intersect joined file "mdp_traits + PC_mdp_genotype + mdp_genotype" -> go to "Analysis" -> go to "association" -> click on "GLM" -> set parameters -> click "ok"

The output of the GLM analysis is produced under the Result node. The GLM association test can be evaluated by plotting Q-Q plot and the Manhattan plot as shown below.

Select the association analysis output file -> go to "Results" -> click on "Manhattan plot"-> select the trait
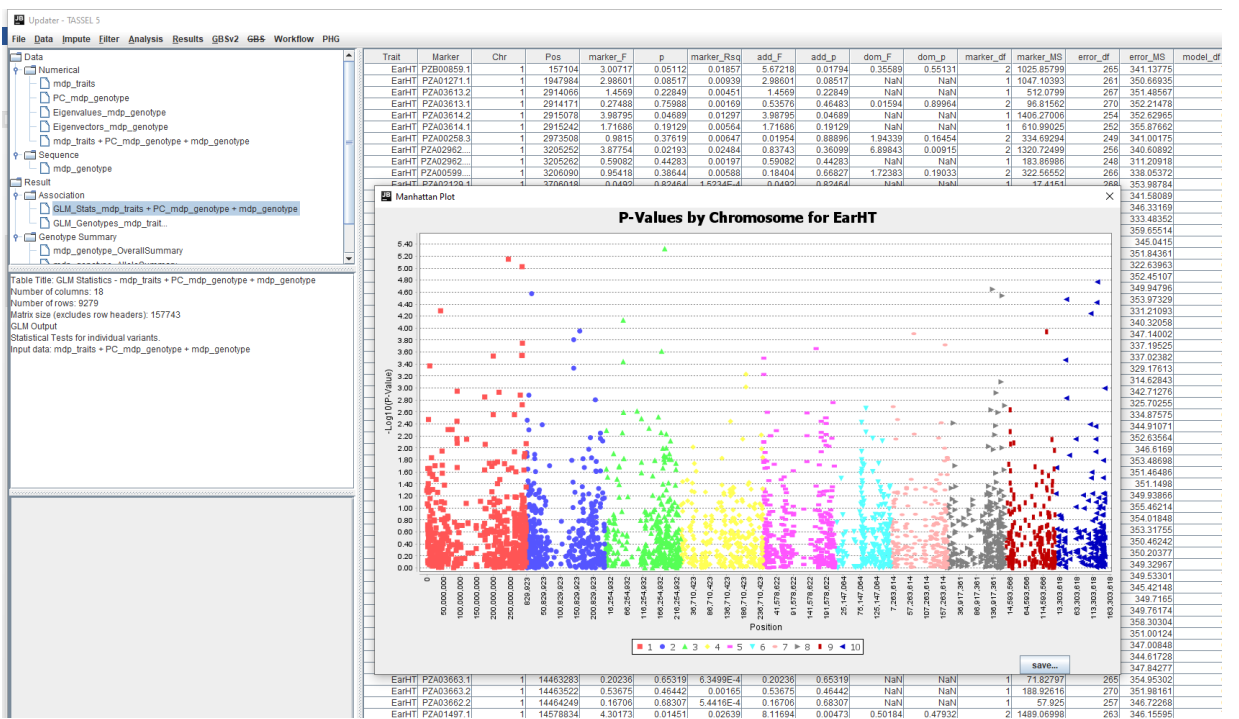
Select the association analysis output file -> go to "Results" -> click on "QQ plot"-> select the trait -> click "okay"
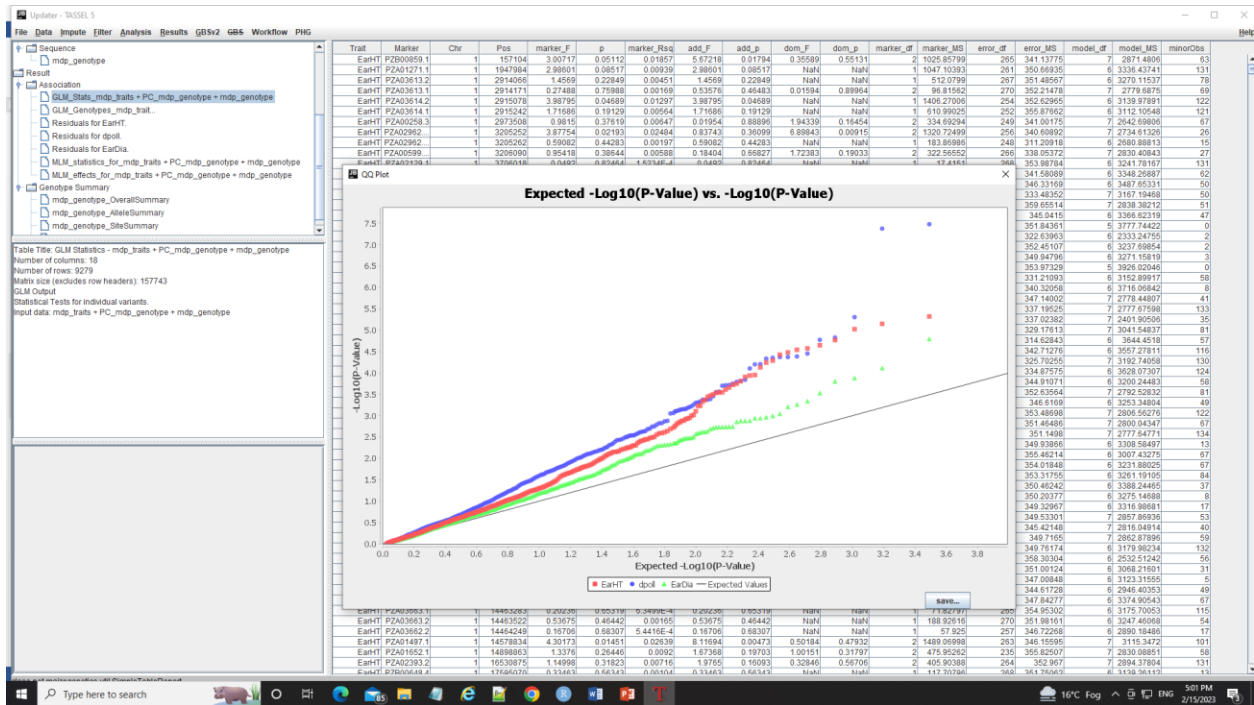
11. Mixed Linear Model (MLM)

Calculating Kinship matrix

Follow the below steps to calcuate the kinship matrix:

Select genotype file -> go to "Analysis" ->go to "Relatedness" -> click on "kinship" -> set parameters -> click "ok"

Running Mixed Linear Model (MLM)

MLM model includes the PCA and the kinship matrix i.e. MLM (PCA+K).

Therefore, once the Kinship matrix has been calculated, MLM can be now be conducted by following below steps:

Select the intersect joined file "mdp_traits + PC_mdp_genotype + mdp_genotype" and kinship file simultaneously by holding 'CTRL' button -> go to "Analysis" -> go to "Association" -> click on "MLM" -> set parameters -> click "okay"

Plot the output (MLM stats file in the Results branch following the steps shown for GLM).

12. Exporting results

One may export the results in .txt format by the following the below steps:

Select the file -> go to "File" -> click on " Save As" ->browse the folder to save the file -> name the file ->click "okay"

13. Plotting GWAS results in R using qqman package

The R code to plot GWAS result using QQMAN package is below:

```
library(qqman)

library(dplyr)

# import TASSEL results

# note

TASSEL_MLM_Out <- read.table("mlm_out.txt", header = T, sep = "\t")

# Number of traits

head(unique(TASSEL_MLM_Out$Trait))

# note: for each plot trait name must be specificed

# first trait as example (i.e., EarHT)

Trait1 <-  TASSEL_MLM_Out %>% filter(.$Trait == "EarHT")

# Bonferroni correction threshold

nmrk <- nrow(Trait1)

(GWAS_Bonn_corr_threshold <- -log10(0.05 / nmrk))

# Manhattan plot

(Mann_plot <- manhattan(
```

```r
  TASSEL_MLM_Out,
  chr = "Chr",
  bp = "Pos",
  snp = "Marker",
  p = "p",
  col = c("red", "blue"),
  annotateTop = T,
  genomewideline = GWAS_Bonn_corr_threshold,
  suggestiveline = F
)
)
# QQ plot
QQ_plot <- qq(TASSEL_MLM_Out$p)
# Manhattan and Q-Q plot arranged in 1 rows and 2 columns
old_par <- par()
par(mfrow=c(1,2))
(Mann_plot <- manhattan(
  TASSEL_MLM_Out,
  chr = "Chr",
  bp = "Pos",
  snp = "Marker",
  p = "p",
  col = c("red", "blue"),
  annotateTop = T,
  genomewideline = GWAS_Bonn_corr_threshold,
  suggestiveline = F,
  main = "EarHT" # trait name
)
```

)

(QQ_plot <- qq(TASSEL_MLM_Out$p, main = "EarHT"))

The output plot will be as shown below:
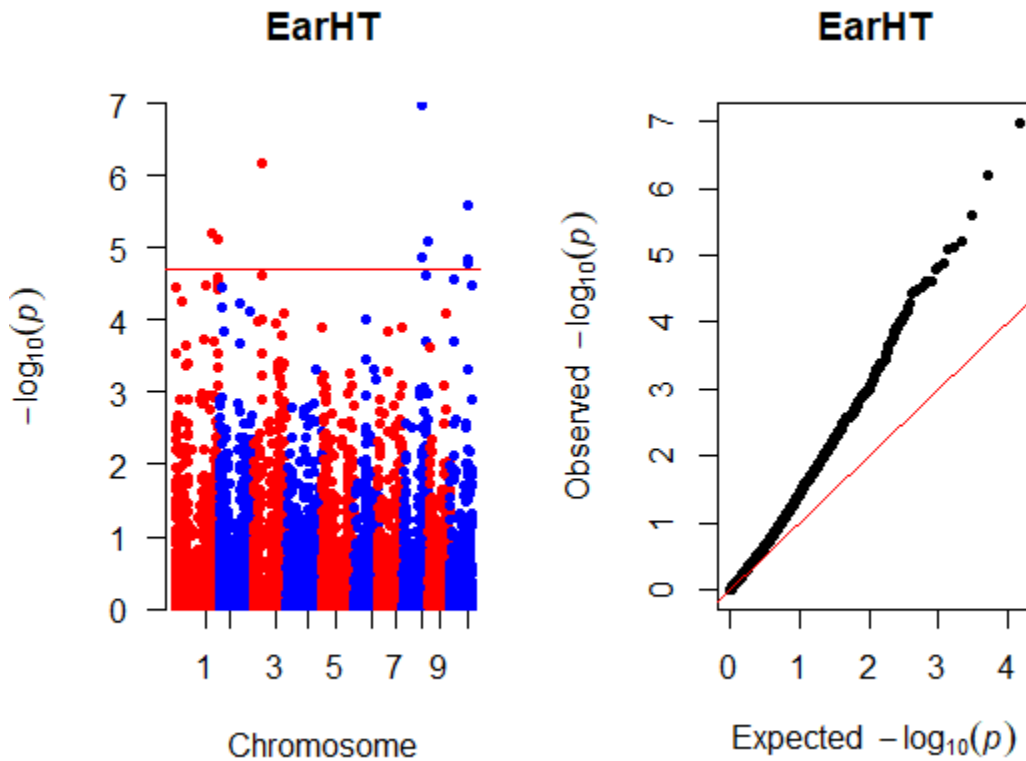
Reference:

Bradbury, P. J., Zhang, Z., Kroon, D. E., Casstevens, T. M., Ramdoss, Y., & Buckler, E. S. (2007). TASSEL: software for association mapping of complex traits in diverse samples. *Bioinformatics*, *23*(19), 2633-2635.

Transcriptomic Data Analysis

Mohammad Samir Farooqi and Sudhir Srivastava

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

The advent of Next-Generation Sequencing (NGS) technology has transformed genomic studies. One important application of NGS technology is the study of the *transcriptome*, which is defined as the complete collection of all the RNA molecules in a cell. Various types of RNA that have been classified so far are shown in **Fig. 1**. All of these molecules are called *transcripts* since they are produced by process of transcription.
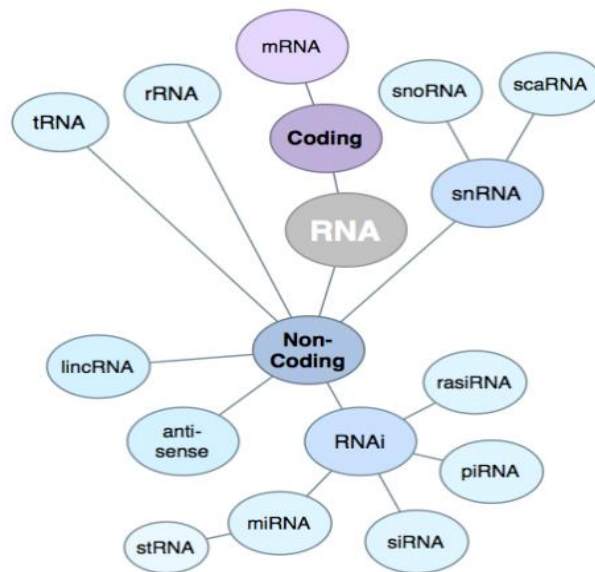
Fig. 1: Different types of RNA

(Image source: http://scienceblogs.com/digitalbio/2011/01/08/next-gene-sequencing)

Understanding the transcriptome is essential for interpreting the functional elements of the genome and revealing the molecular constituents of cells and tissues, and also for understanding development and disease [1]. The main purpose of transcriptomics are: to catalogue all species of transcript, including mRNAs, non-coding RNAs and small RNAs; to determine the transcriptional structure of genes, in terms of their start sites, 5′ and 3′ ends, splicing patterns and other post-transcriptional modifications; and to quantify the changing expression levels of each transcript during development and under different conditions.

The study of transcriptome is carried out through sequencing of RNAs. *RNA sequencing (RNA-Seq)* is a powerful method for discovering, profiling, and quantifying RNA transcripts [2]. RNA-Seq uses NGS datasets to obtain sequence reads from millions of individual RNAs. The RNA-Seq analysis is performed in several steps: First, all genes are extracted from the reference genome (using annotations of type *gene*). Other annotations on the gene sequences are preserved (e.g.CDS information about coding sequences etc). Next, all annotated

transcripts (using annotations of type *mRNA*) are extracted [3]. If there are several annotated splice variants, they are all extracted. An example is shown in below **Fig. 2(a).**
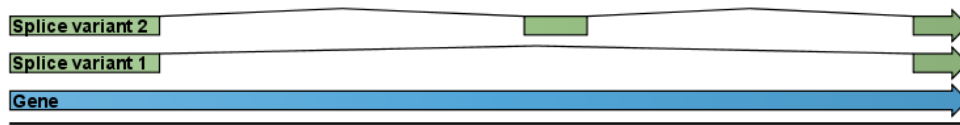
Fig. 2(a): A simple gene with three exons and two splice variants.

The given example is a simple gene with three exons and two splice variants. The transcripts are extracted as shown in **Fig. 2(b).**
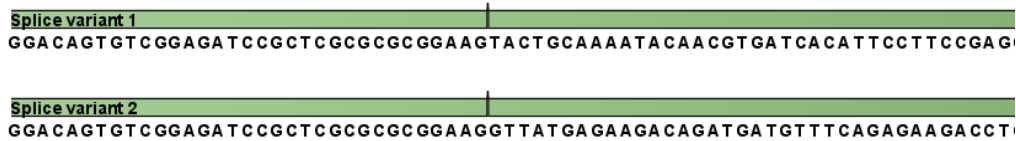
Fig. 2(b): All the exon-exon junctions are joined in the extracted transcript.

Next, the reads are mapped against all the transcripts plus the entire gene [see **Fig. 2(c)**].
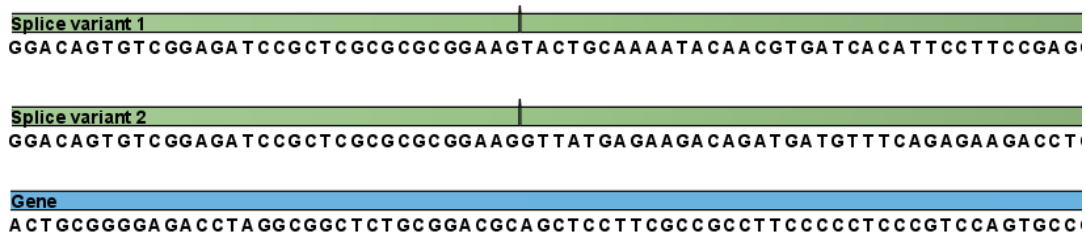
Fig. 2(c): The reference for mapping: all the exon-exon junctions and the gene

(Image source: CLC Genomic workbench tutorials)

From this mapping, the reads are categorized and assigned to the genes and expression values for each gene and each transcript are calculated and putative exons are then identified.

RNA Sequencing Experiment

In a standard RNA-seq experiment, a sample of RNA is converted to a library of complementary DNA fragments and then sequenced on a high-throughput sequencing platform, such as Illumina's Genome Analyzer, SOLiD or Roche 454 [4]. Millions of short sequences, or reads, are obtained from this sequencing and then mapped to a reference genome (**Fig. 3**). The count of reads mapped to a given gene measures the expression level of this gene. The unmapped reads are usually discarded and mapped reads for each sample are assembled into gene-level, exon-level or transcript-level expression summaries, depending on the objectives of the experiment. The count of reads mapped to a given gene/exon/transcript measures the expression level for this region of the genome or transcriptome.

One of the primary goals for most RNA-seq experiments is to compare the gene expression levels across various treatments. A simple and common RNA-seq study involves two treatments in a randomized complete design, for example, treated versus untreated cells, two different tissues from an organism, plants, etc. In most of the studies, researchers are

particularly interested in detecting gene with differential expressions (DE). A gene is declared differentially expressed if an observed difference or change in read counts between two experimental conditions is statistically significant, i.e. if the difference is greater than what would be expected just due to random variation [5]. Detecting DE genes can also be an important pre-step for subsequent studies, such as clustering gene expression profiles or testing gene set enrichments.
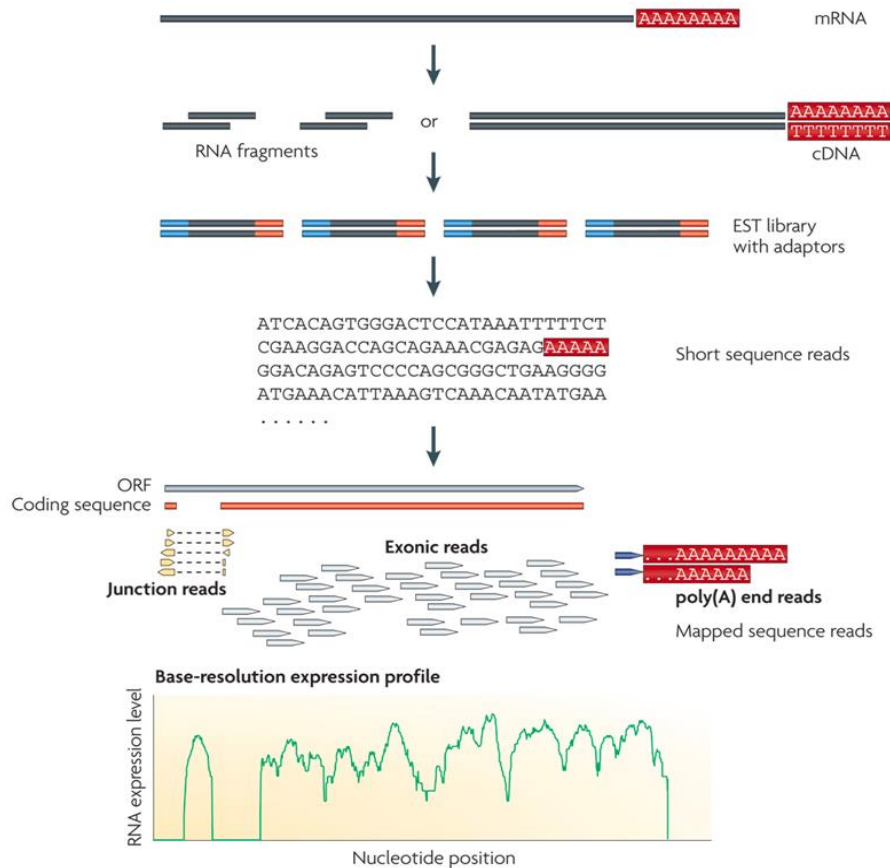
Fig. 3: General RNA-seq experiment. mRNA is converted to cDNA, and fragments from that library are used to generate short sequence reads. Those reads are assembled into contigs which may be mapped to reference sequences (Wang et al., 2009)

Analysing RNA-Seq data

RNA-seq experiments must be analyzed with robust, efficient and statistically correct algorithms. Fortunately, the bioinformatics community has been striving hard at work for incorporating mathematics, statistics and computer science for RNA-seq and building these ideas into software tools. RNA-seq analysis tools generally fall into three categories: (i) those for read alignment; (ii) those for transcript assembly or genome annotation; and (iii) those for transcript and gene quantification. Some of the open source softwares available for RNA-seq analysis are as follows:

- **Data preprocessing**
    - Fastx toolkit
    - Samtools
- **Short reads aligners**

- Bowtie, TOPHAT, Stampy, BWA, Novoalign, etc
  - **Expression studies**
    - Cufflinks package
    - R packages (DESeq, edgeR, *more...*)
  - **Visualisation**
    - CummeRbund, IGV, Bedtools, UCSC Genome Browser, etc.

Besides there are commercially data analysis pipelines like GenomeQuest, CLCBio etc available for researchers to use. The most commonly used pipeline is to identify protein coding genes by aligning RNA-Seq data to annotate data from sources like RefSeq. After generating the alignments, the number of aligning sequences is counted for each position. Since each alignment represents a transcript, the alignments allow to count the number of RNA molecules produced from every gene.

Using NGS technology, RNA-Seq enables to count the number of reads that align to one of thousands of different cDNAs, producing results similar to those of gene expression microarrays [6]. Sequences generated from an RNA-Seq experiment are usually mapped to libraries of known exons in known transcripts. RNA-Seq can be used for discovery applications such as identifying alternative splicing events, allele-specific expression, and rare and novel transcripts [7]. The sequencing output files (compressed FASTQ files) are the input for secondary analysis. Reads are aligned to an annotated reference genome, and those aligning to exons, genes and splice junctions are counted. The final steps are data visualisation and interpretation, consisting of calculating gene- and transcript-expression and reporting differential expression. A general Bioinformatics workflow to map transcripts from RNA-seq data is shown in **Fig. 4**.
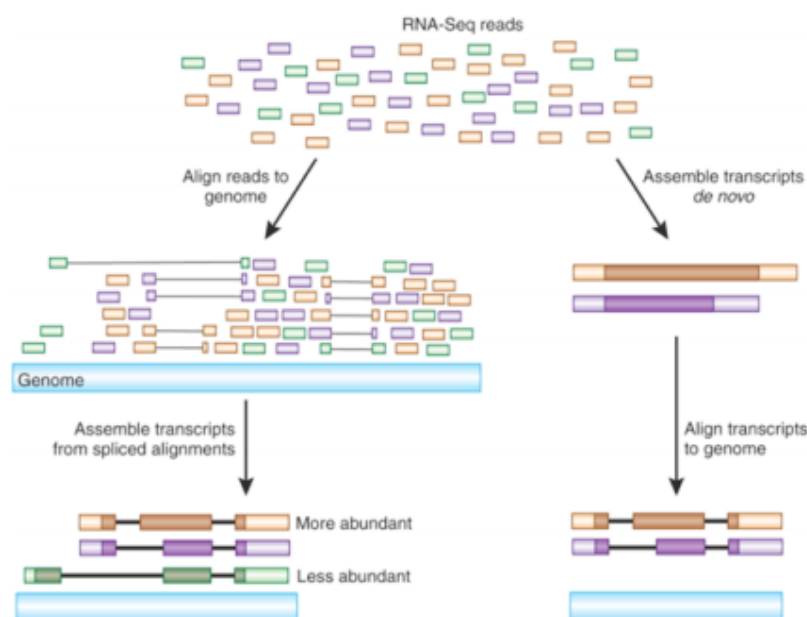
Fig. 4: RNA-seq workflow (Adapted from Advancing RNA-Seq analysis Brian J. Haas and Michael C. Zody Nature Biotechnology 28, 421-423 (2010)

RPKM (Reads per KB per million reads)

RNA-Seq provides quantitative approximations of the abundance of target transcripts in the form of counts. However, these counts must be normalized to remove technical biases inherent in the preparation steps for RNA-Seq, in particular the length of the RNA species and the sequencing depth of a sample. The most commonly used is RPKM (Reads Per Kilobase of exon model per Million mapped reads). The RPKM measure of read density reflects the molar concentration of a transcript in the starting sample by normalizing for RNA length and for the total read number in the measurement [8]. RPKM is mathematically represented as:

$$\text{RPKM} = \frac{total\ exon\ reads}{mapped\ reads\ (millions)\ \text{X}\ exon\ length\ (KB)}$$

Total exon reads

This is the number of reads that have been mapped to a region in which an exon is annotated for the gene or across the boundaries of two exons or an intron and an exon for an annotated transcript of the gene. For eukaryotes, exons and their internal relationships are defined by annotations of type mRNA.

Exon length

This is calculated as the sum of the lengths of all exons annotated for the gene. Each exon is included only once in this sum, even if it is present in more annotated transcripts for the gene. Partly overlapping exons will count with their full length, even though they share the same region.

Mapped reads

The total gene reads for a gene is the total number of reads that after mapping have been mapped to the region of the gene. A gene's region is that comprised of the flanking regions, the exons, the introns and across exon-exon boundaries of all transcripts annotated for the gene. Thus, the sum of the total gene reads numbers is the number of mapped reads for the sample.

Applications of RNA-seq

This technique can be used to:

- Measure gene expression

- Transcriptome assembly, gene discovery and annotation

- Detect differential transcript abundances between tissues, developmental stages, genetic backgrounds, and environmental conditions

- Characterize alternative splicing, alternative polyadenylation, and alternative transcription.

Future Directions

Although RNA-Seq is still in the infancy stages of use, it has clear advantages over previously developed transcriptomic methods. Compared with microarray, which has been the dominant approach of studying gene expression in the last two decades, RNA-seq technology has a wider measurable range of expression levels, less noise, higher throughput,

and more information to detect allele-specific expression, novel promoters, and isoforms [9]. For these reasons, RNA-seq is gradually replacing the array-based approach as the major platform in gene expression studies. The next big challenge for RNA-Seq is to target more complex transcriptomes to identify and track the expression changes of rare RNA isoforms from all genes. Technologies that will advance achievement of this goal are pair-end sequencing, strand-specific sequencing and the use of longer reads to increase coverage and depth. As the cost of sequencing continues to fall, RNA-Seq is expected to replace microarrays for many applications that involve determining the structure and dynamics of the transcriptome.

References

1. https://www.genome.gov/13014330

2. Wang Z., Gerstein M., Synder M. (2009). Rna-seq: a revolutionary tool for transciptomics, Nat Rev Genet 10(1): 57–63.

3. http://scienceblogs.com/digitalbio/2011/01/08/next-gene-sequencing-results-a/

4. Shendure J, Ji H (2008) Next-generation RNA sequencing. Nature Biotechnology 26: 2514-2521

5. Anders S, Huber W (2010). Differential expression analysis for sequence count data. Genome Biol. 11:R106.
   Illumina, Inc,. (2011). Getting started with RNA-Seq Data Analysis. Pub. No. 470-2011-003.

6. Illumina, Inc,. (2011). RNA-Seq Data Comparison with Gene Expression Microarrays. A cross-platform comparison of differential gene expression analysis. Pub. No. 470-2011-004

7. Yaqing Si (2012). Statistical analysis of RNA-seq data from next-generation sequencing technology. PhD thesis. Iowa State University, Ames, Iowa.

8. Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by rna-seq. Nat Methods, 5(7):621-628.

9. Wang L., Si Y., Dedow L.K., Shao Y., Liu P., Brutnell T.P. (2010). A low-cost library construction protocol and data analysis pipeline for Illumina-based strand-specific multiplex RNA-seq. PLoS One 6(10):e26426.

10. Brian J. H. and Michael C. Z. (2010). Advancing RNA-Seq analysis Nature Biotechnology 28, 421-423.

Hands-on Session for Transcriptomic Data Analysis

Soumya Sharma

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Identification of differentially expressed genes from the RNA-Seq data is an important area of bioinformatics data analysis. There are several packages available in R to carry out the differential gene expression analysis, like **DESeq2** (Love et al., 2014), **edgeR** (Robinson et al., 2010), **limma** (Smyth et al., 2005) *etc.* After preprocessing and quantification of reads in RNA-Seq data, we get a matrix of read counts of each gene in every sample. Then we can use the "**DESeq2**" package to identify differentially expressed genes. Here, we demonstrate the differential gene expression analysis with R using a sample dataset available in the R package **airway** (Himes et al., 2014) in following steps.

    i)   Download the sample dataset from the "**airway**" package. The package contains 2 data files. One file contains read counts of 64102 genes in 8 samples obtained from the RNA-Seq experiment on 4 primary human airway smooth muscle cell lines treated with 1 micromolar dexamethasone for 18 hours. Another file contains sample-wise metadata information, *viz.*, treated or untreated. Import the count matrix and metadata file into RStudio.

R code to collect sample dataset from "airway" package:

```
# installing Bioconductor packages
if (!requireNamespace("BiocManager", quietly=TRUE))

  install.packages("BiocManager")

  BiocManager::install("airway")

library(airway)

data(airway)

airway

sample_info <- as.data.frame(colData(airway))

sample_info <- sample_info[,c(2,3)]

sample_info$dex <- gsub('trt', 'treated', sample_info$dex)

sample_info$dex <- gsub('untrt', 'untreated', sample_info$dex)

names(sample_info) <- c('cellLine', 'dexamethasone')
```

Get the samplewise metadata file

write.table(sample_info, file = "/sample_info.csv", sep = ',', col.names = T, row.names = T, quote = F)

Get the matrix of read counts for each gene in every sample

countsData <- assay(airway)

write.table(countsData, file = "/counts_data.csv", sep = ',', col.names = T, row.names = T, quote = F)

ii)   Then we have to load the package "**DESeq2**" to perform the subsequent differential gene expression analysis. We have to create a DESeqDataSet object and then run the 'DESeq()' function to perform the said analysis.

Differential gene expression analysis using the "DESeq2" package in R

BiocManager::install("DESeq2")

library(DESeq2)

read in counts data

counts_data <- read.csv('/counts_data.csv')

read in sample info

colData <- read.csv('/sample_info.csv')

making sure the row names in colData matches to column names in counts_data

all(colnames(counts_data) %in% rownames(colData))

are they in the same order?

all(colnames(counts_data) == rownames(colData))

dds <- DESeqDataSetFromMatrix(countData = counts_data, colData = colData, design = ~ dexamethasone)

dds

#pre-filtering: removing rows with low gene counts

keeping rows that have at least 10 reads total

keep <- rowSums(counts(dds)) >= 10

dds <- dds[keep,]

set the factor level

dds$dexamethasone <- relevel(dds$dexamethasone, ref = "untreated")

```
# --------Run DESeq ---------------------

dds <- DESeq(dds)

res <- results(dds)

res

summary(res)

res0.01 <- results(dds, alpha = 0.01) # When padj = 0.01

summary(res0.01)
```

Here, we are trying to find the genes which are differentially expressed in Dexamethasone treated conditions as compared to untreated conditions. Hence, the reference level is set as 'untreated'. After the analysis, the result contains base means, log_2FoldChange values, p-values, adjusted p-values, *etc.* for each gene. If at 1% level, the adjusted p-value for a gene is found as > 0.01, it means the result has been obtained purely by chance, *i.e.*, a non-significant result. Otherwise, that gene is differentially expressed if the adjusted p-value is < 0.01. In the latter case, if the log2FoldChange value is > 0, the gene is upregulated and if it is < 0, then that gene is downregulated. Thus, we can find out differentially expressed genes using R.

 iii) Visualization of differentially expressed genes in R. After identifying differentially expressed genes, we can visualize the result in terms of various plots such as MA plot, volcano plot, heatmap, *etc.* Several R packages are available to develop these plots. MA plot can be generated using the 'plotMA()' function. We can use the "**ggplot2**" package to develop volcano plot. Similarly, R package "**heatmap2", "pheatmap**" *etc.* are useful to create heatmaps. MA plot (fig 1), volcano plot (fig 2) and heatmap (fig 3) created from the result of the previous analysis.

R code to visualize the result of differential gene expression analysis
```
# MA plot

plotMA(res)

# Volcano plot

library(ggplot2)

library(tidyverse)

df<-as.data.frame(res)
```

df$diffexpressed <- "non-significant"

if log2Foldchange > 0 and padj < 0.01, set as "UP"

df$diffexpressed[df$log2FoldChange > 0 & df$padj < 0.01] <- "UP"

if log2Foldchange < 0 and padj < 0.01, set as "DOWN"

df$diffexpressed[df$log2FoldChange < 0 & df$padj < 0.01] <- "DOWN"

ggplot(df, aes(log2FoldChange, -log10(padj), col= diffexpressed))+geom_point()+scale_color_manual(values = c("red", "black", "green"))

Developing Heatmap of first 10 genes for better demonstration

library(pheatmap)

library(RColorBrewer)

breaksList = seq(-0.4, 0.5, by = 0.04)

rowLabel = row.names(counts_data[1:10,])

pheatmap(df$log2FoldChange[1:10], color = colorRampPalette(c("dark blue", "white", "yellow"))(25), breaks = breaksList, border_color = "black", cellheight = 25, cellwidth = 25, cluster_rows = F,cluster_cols = F, fontsize = 12, labels_row = rowLabel)
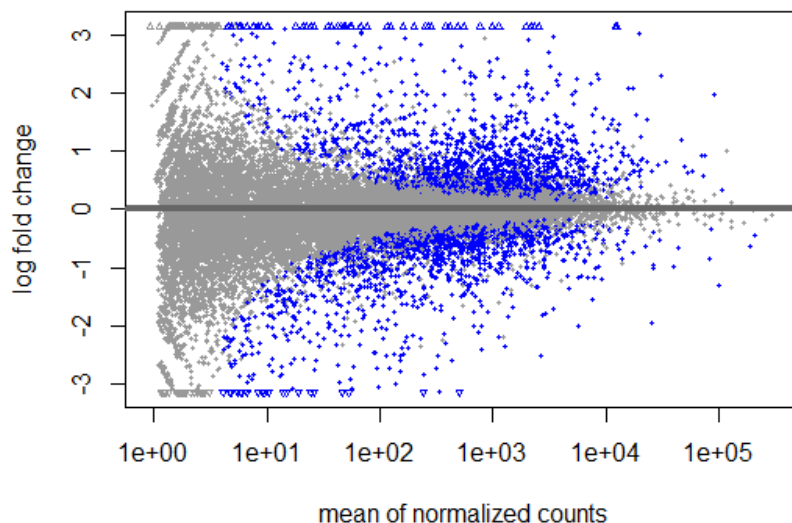
Fig 1: MA plot showing significantly upregulated and downregulated genes as blue dots.
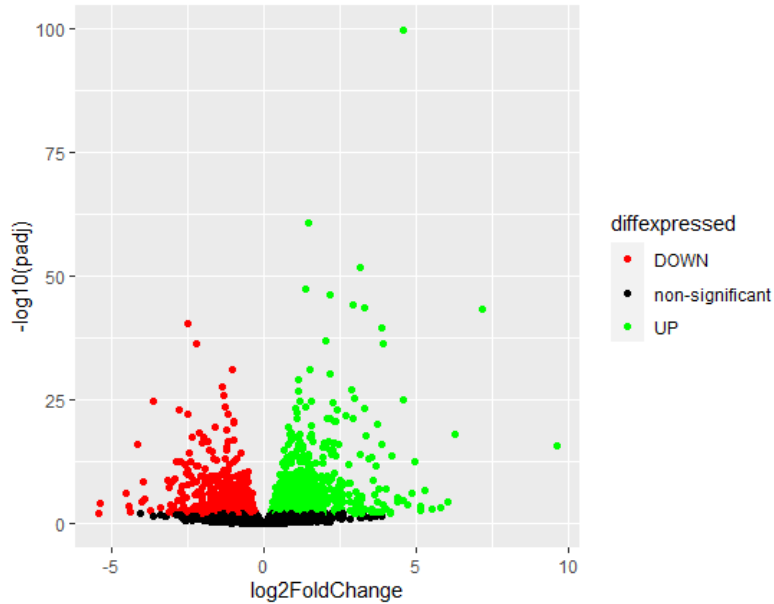
Fig 2: Volcano plot representing upregulated genes as green, downregulated genes as red and non-significant genes as black dots.
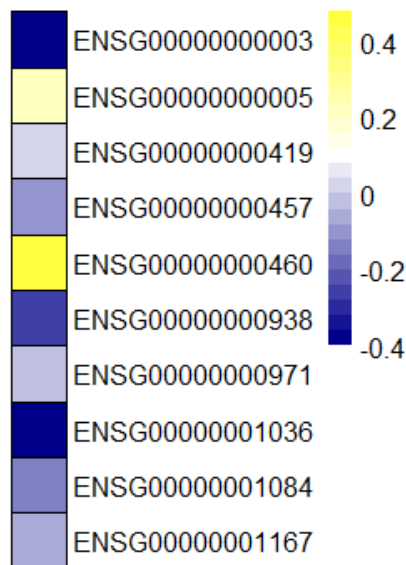
Fig 3: Heatmap representing the expression levels of first 10 genes in terms of log2FoldChange values in a scale of -0.4 to 0.4 where, blue colour represents downregulated genes, yellow represents upregulated genes and expression levels of remaining genes are represented by gradation of colour between blue and yellow.

References:

Himes, B. E., Jiang, X., Wagner, P., Hu, R., Wang, Q., Klanderman, B., & Lu, Q. (2014). RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells. *PloS one*, *9*(6), e99625.

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology*, *15*(12), 1-21.

Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics*, *26*(1), 139-140.

Smyth, G. K. (2005). Limma: linear models for microarray data. *Bioinformatics and computational biology solutions using R and Bioconductor*, 397-420.

Analysis of Non-Coding Sequencing Data

Sarika Sahu

ICAR- Indian Agricultural Statistics Research Institute, New Delhi

Non-coding RNAs (ncRNAs) are RNA molecules that do not code for proteins. They are transcribed from DNA and can be categorized into two main types: long non-coding RNAs (lncRNAs) and small non-coding RNAs (sncRNAs). While sncRNAs are shorter than 200 nucleotides, lncRNAs are usually longer than 200 nucleotides. Non-coding RNAs have been found to play important roles in a variety of cellular processes, including gene expression, cell differentiation, and development.

One of the most well-studied classes of sncRNAs are microRNAs (miRNAs). miRNAs are single-stranded RNA molecules that are about 21-25 nucleotides long. They play important roles in post-transcriptional regulation of gene expression by targeting mRNAs for degradation or translational repression. This means that miRNAs can control the amount of protein that is produced from a particular gene. miRNAs have been implicated in a variety of biological processes, including cell proliferation, differentiation, and apoptosis. Dysregulation of miRNA expression has been linked to various diseases, such as cancer, neurological disorders, and cardiovascular disease.

Another type of sncRNA is the small interfering RNA (siRNA). Like miRNAs, siRNAs are about 21-25 nucleotides long and are involved in gene regulation by inducing degradation of specific mRNAs. However, siRNAs are usually exogenously introduced into cells for therapeutic purposes or for use in research. They can be used to specifically target and silence disease-causing genes or to study gene function in experimental systems.

Piwi-interacting RNAs (piRNAs) are a class of sncRNAs that interact with a family of proteins known as Piwi proteins. piRNAs are typically longer than miRNAs or siRNAs and are

expressed primarily in the germ cells of animals. They play important roles in protecting the genome from transposable elements (mobile genetic elements that can cause mutations) by inducing their silencing or degradation. piRNAs have also been implicated in other processes such as epigenetic regulation and germ cell development.

In addition to sncRNAs, lncRNAs have also been found to play important roles in various biological processes. They are involved in gene regulation at multiple levels, including transcription, splicing, and chromatin remodeling. lncRNAs can interact with DNA, RNA, and proteins to modulate gene expression. Dysregulation of lncRNA expression has been implicated in a variety of diseases, such as cancer, cardiovascular disease, and neurological disorders.

One example of a lncRNA is Xist, which is involved in X chromosome inactivation in female mammals. Xist is expressed from one of the two X chromosomes in female cells and coats the same chromosome it is transcribed from, leading to silencing of most genes on that chromosome. Another example is HOTAIR, which is involved in regulating gene expression during development and has been found to be dysregulated in various types of cancer.

In conclusion, non-coding RNAs are a diverse group of RNA molecules that play important roles in a variety of cellular processes. While sncRNAs like miRNAs and siRNAs are involved in post-transcriptional regulation of gene expression, piRNAs are involved in transposon silencing in germ cells. lncRNAs, on the other hand, are involved in gene regulation at multiple levels and have been implicated in various diseases. With the continued development of new technologies for studying RNA, we can expect to uncover many more functions and roles for these fascinating molecules in the future.

Long non-coding RNAs (lncRNAs) are a diverse class of RNA molecules that have been found to play important roles in gene regulation and other biological processes in many different organisms, including plants. In this discussion, we will explore the current understanding of lncRNAs in plants, their functions, and their potential applications in agriculture.

Plant lncRNAs are typically longer than 200 nucleotides and are transcribed from intergenic regions, introns, and other non-coding regions of the genome. They can be classified into several different categories based on their genomic origin and structure, including natural antisense transcripts (NATs) and long intergenic non-coding RNAs (lincRNAs).

One of the most well-studied plant lncRNAs involved in growth and development is COOLAIR, a natural antisense transcript (NAT) of the FLOWERING LOCUS C (FLC) gene in Arabidopsis thaliana. FLC is a key regulator of flowering time, and the expression of COOLAIR promotes FLC mRNA decay, leading to earlier flowering. COOLAIR is also involved in regulating the expression of other genes related to plant development, such as genes involved in the biosynthesis of gibberellins, a class of plant hormones that promote stem elongation and other growth processes.

Another lncRNA involved in the regulation of flowering time is IPS1 (Induced by Phosphate Starvation 1) in Arabidopsis. IPS1 is a lincRNA that is induced by phosphate starvation and negatively regulates the expression of miR399, a microRNA that targets a gene involved in phosphate homeostasis. The downregulation of miR399 by IPS1 promotes the expression of genes involved in phosphate uptake and transport, leading to earlier flowering.

LINC5 is another lincRNA involved in the regulation of flowering time in Arabidopsis. LINC5 is specifically expressed in the shoot apical meristem, where it interacts with the transcription factor WUSCHEL (WUS) to promote its expression. WUS is a key regulator of stem cell maintenance and differentiation in the shoot apical meristem, and the expression of LINC5 is

required for normal shoot development. Similarly, in rice, a lincRNA called LDMAR is involved in the regulation of lateral root development. LDMAR is specifically expressed in lateral root primordia and promotes the expression of genes involved in lateral root development. Knockdown of LDMAR leads to a reduction in the number of lateral roots, indicating its importance in this process.

In addition to their roles in plant growth and development, lncRNAs have also been implicated in stress responses. For example, a lincRNA called COLDAIR in Arabidopsis is involved in the regulation of the COLD-REGULATED (COR) genes in response to cold stress. COLDAIR interacts with a transcription factor called CBF1 to promote the expression of COR genes, which are involved in protecting plants from freezing damage.

LincRNAs, on the other hand, are transcribed from intergenic regions of the genome and can interact with DNA, RNA, and proteins to modulate gene expression. They can act as scaffolds for the assembly of regulatory complexes, as well as serve as guides for chromatin-modifying enzymes. In rice, a lincRNA called NERICA1 is involved in promoting nodulation in response to symbiotic bacteria by interacting with chromatin-modifying enzymes to regulate gene expression.

Plant lncRNAs have also been found to play important roles in stress responses, such as drought, salt, and cold stress. For example, in Arabidopsis, a lincRNA called COLDAIR is involved in regulating the expression of COLD-REGULATED (COR) genes in response to cold stress. COLDAIR interacts with a transcription factor called CBF1 to promote the expression of COR genes, which are involved in protecting plants from freezing damage.

The roles of plant lncRNAs in development have also been extensively studied. In maize, a lincRNA called Zm401 is involved in regulating the expression of key genes during the transition from vegetative growth to reproductive development. Zm401 interacts with a

chromatin-modifying complex to regulate the expression of genes involved in flowering and other developmental processes.

One study identified 285 lncRNAs in potato leaves and tubers and analyzed their expression patterns during potato development. The researchers found that many lncRNAs were differentially expressed in different tissues and developmental stages, indicating their potential roles in regulating potato growth and development.

Another study investigated the role of a potato lncRNA called lncRNA1604 in response to potato virus Y (PVY) infection. The researchers found that lncRNA1604 was induced in response to PVY infection and was involved in regulating the expression of genes involved in defense responses. Knockdown of lncRNA1604 resulted in increased susceptibility to PVY infection, indicating its role in potato resistance to viral infections.

In addition to their roles in development and stress responses, lncRNAs in potato have also been implicated in other biological processes. For example, a recent study identified a potato lncRNA called StTILLING1 that was involved in regulating the production of starch in potato tubers. Knockdown of StTILLING1 resulted in reduced starch content and altered starch granule morphology, indicating its role in starch synthesis.

Overall, the study of lncRNAs in plants is still in its early stages, and much remains to be learned about their functions and mechanisms of action. However, the identification of lncRNAs involved in growth and development processes in plants provides new insights into the regulatory networks underlying these processes and offers new targets for crop improvement and genetic engineering.

Circular RNAs (circRNAs) are a relatively new class of lncRNAs that are formed by back-splicing events, in which a downstream splice acceptor is joined to an upstream splice donor. circRNAs can act as sponges for microRNAs (miRNAs) and other RNA-binding proteins, thereby regulating gene expression. In tomato, a circRNA called ciRs-7 is involved in regulating fruit ripening by sequestering miR-7, which targets several genes involved in fruit ripening. Some of the known functions of circRNAs in plants include regulating gene expression at both the transcriptional and post-transcriptional levels, modulating alternative splicing, and participating in stress responses. For example, a circRNA called circRNA_022653 has been shown to regulate the expression of the transcription factor WRKY40 in response to salt stress in Arabidopsis thaliana.

In addition, circRNAs have been implicated in plant development, particularly in the regulation of flowering time. A circRNA called circFTO has been found to play a role in the photoperiodic flowering pathway in Arabidopsis, by regulating the expression of a key flowering-time regulator called CONSTANS.

.References

1. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology* 215, 403–410. doi:10.1016/S0022-2836(05)80360-2.

2. Baulcombe, D. (2004). RNA silencing in plants. *Nature* 431, 356–363. doi:10.1038/nature02874.

3. Bader GD, Hogue CW. An automated method for finding molecular complexes in large protein interaction networks. BMC bioinformatics. 2003 Dec;4(1):1-27.

4. Bolger, A. M., Lohse, M., and Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* 30, 2114–2120. doi:10.1093/bioinformatics/btu170.

5. Denman, R. B. (1993). Using RNAFOLD to predict the activity of small catalytic RNAs. *BioTechniques* 15, 1090–5.

6. Dong, P., Wang, H., Fang, T., Wang, Y., and Ye, Q. (2019). Assessment of extracellular antibiotic resistance genes (eARGs) in typical environmental samples and the transforming ability of eARG. *Environment International* 125, 90–96. doi:10.1016/j.envint.2019.01.050.

7. Du L, Zhang C, Liu Q, Zhang X, Yue B. Krait: an ultrafast tool for genome-wide survey of microsatellites and primer design. Bioinformatics. 2018 Feb 15;34(4):681-3.

8. Fujita, Y., Fujita, M., Satoh, R., Maruyama, K., Parvez, M. M., Seki, M., *et al*. (2005). AREB1 Is a Transcription Activator of Novel ABRE-Dependent ABA Signaling That Enhances Drought Stress Tolerance in *Arabidopsis* . *The Plant Cell* 17, 3470–3488. doi:10.1105/tpc.105.035659.

9. Gkirtzou K, Tsamardinos I, Tsakalides P, Poirazi P. MatureBayes: a probabilistic algorithm for identifying the mature miRNA within novel precursors. PloS one. 2010 Aug 6;5(8):e11843.

10. Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., *et al*. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity

platform for reference generation and analysis. *Nature Protocols* 8, 1494–1512. doi:10.1038/nprot.2013.084.

11. Hill, R. M., and Rawate, P. D. (1982). Evaluation of food potential, some toxicological aspects, and preparation of a protein isolate from the aerial part of *amaranthus* (pigweed). *Journal of Agricultural and Food Chemistry* 30, 465–469. doi:10.1021/jf00111a014.

12. Jain P., Hussain S., Nishad J., Dubey H., Bisht D.S., Sharma T.R., Mondal T.K. (2021) Identification and functional prediction of long non-coding RNAs of rice (Oryza sativa L.) at reproductive stage under salinity stress. *Molecular Biology Reports*. 10.1007/s11033-021-06246-8 .

13. Jain P., Sharma V., Dubey H., Singh P.K., Kapoor R., Kumari M., Singh J., Pawar D., Bisht D., Solanke A.U., Mondal T.K., Sharma T.R. (2017) Identification of long non-coding RNA in rice lines resistant to Rice blast pathogen *Magnaporthe oryzae.* Bioinformation. 13:249-55.

14. Jeyaraj, A., Liu, S., Zhang, X., Zhang, R., Shangguan, M., and Wei, C. (2017). Genome-wide identification of microRNAs responsive to Ectropis oblique feeding in tea plant (Camellia sinensis L.). Scientific Reports 7, 13634. doi:10.1038/s41598-017-13692-7

15. Ramírez Gonzales L, Shi L, Bergonzi SB, Oortwijn M, Franco- Zorrilla JM, Solano-Tavira R, Visser RG, Abelenda JA, Bachem CW. Potato CYCLING DOF FACTOR 1 and its lncRNA counterpart StFLORE link tuber development and drought response. The Plant Journal. 2021 Feb;105(4):855-69.

16. Loewen G, Jayawickramarajah J, Zhuo Y, Shan B. Functions of lncRNA HOTAIR in lung cancer. Journal of hematology & oncology. 2014 Dec;7(1):1-0.

17. Zhang G, Diao S, Zhang T, Chen D, He C, Zhang J. Identification and characterization of circular RNAs during the sea buckthorn fruit development. RNA biology. 2019 Mar 4;16(3):354-61.

18. Meng X, Li X, Zhang P, Wang J, Zhou Y, Chen M. Circular RNA: an emerging key player in RNA world. Briefings in bioinformatics. 2017 Jul 1;18(4):547-57.

Overview of Metagenomics Data Analysis

Md. Samir Farooqi and Sudhir Srivastava

ICAR- Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Metagenomics is the study of overall genomes present in any environment without the need for prior individual identification or amplification. It encompasses microbial communities sampled directly from their natural environment, without prior culturing. Community genomics, environmental genomics, and population genomics are synonyms for the same approach. Metagenomics term was first used by Jo Handelsman *et al.* and first appeared in publication in 1998. The field initially started with the cloning of environmental DNA, followed by functional expression screening and was then quickly complemented by direct random shotgun sequencing of environmental DNA. The idea of cloning DNA directly from environmental samples was first proposed by Pace in 1991. There has been remarkable progress in this field of research due to recent advances in Next Generation Sequencing (NGS) technologies. Since over 99.8% of microbes in some environments are still far from culturing in the media, metagenomics offers a path to the study of microbial community structure, phylogenetic composition, species diversity and abundance, metabolic capacity and functional diversity.

Metagenomics helps in knowing about the functional gene composition of the microbial communities and thus gives more information about the phylogenetic surveys, which are more often based on the diversity of one gene like 16s rRNA gene. It gives genetic information on potentially novel biocatalysts or enzymes, genomic linkages between function and phylogeny for uncultured organisms, and evolutionary profiles of community function and structure. So it acts as novel tool for generating novel hypothesis of microbial function.

Majority of microorganisms have not been cultivated in the laboratory, and almost all of our knowledge of microbial life is based on organisms raised in pure culture. Metagenomics provides an additional set of tools to study uncultured species. Metagenomics entails extraction of DNA from a community so that all of the genomes of organisms in the community are pooled. These genomes are usually fragmented and cloned into an organism that can be cultured to create 'metagenomic libraries', and these libraries are then subjected to analysis based on DNA sequence or on functions conferred on the surrogate host by the metagenomic DNA.

For a typical sequence-based metagenome project one need to go through sampling and processing, sequencing technology, assembly, binning, annotation, experimental design, statistical analysis, and data storage and sharing.

These steps are described as follow:

Sampling and Processing

DNA extracted should represent all cell present in the sample and sufficient amount of high-quality nucleic acids must be obtained for subsequent library production and sequencing. Also processing requires specific protocols for each sample type. The physical and chemical structure of each microbial community affects the quality, size, and amount of microbial DNA that can be extracted.

Sequencing Technology

High-throughput sequencing technologies has improved the capabilities of metagenomic studies to a greater strength but at the same time, it has led to generation of huge and big data sets that largely require high end algorithms and computational tools for data analysis and storage. Metagenome sequencing, also called shotgun sequencing, refers to sequencing DNA fragments extracted from microbial populations. Over the past few years metagenomic shotgun sequencing has gradually shifted from classical Sanger sequencing technology to next-generation sequencing (NGS). However, Sanger sequencing is still best because of its low error rate, long read length (> 700 bp) and large insert sizes (e.g. >30 Kb for fosmids or bacterial artificial chromosomes (BACs)). The only drawback associated is the labor intensive cloning process.

Bioinformatics Approach

Metagenomic projects running worldwide pose several levels of challenges with respect to the processing, analyzing and storing huge data being accumulated. Some of the major computational challenges include the assembly of the whole data, phylogenetic surveys, gene finding and comparative metagenomic analysis for the metabolic pathways.

The data generated by metagenomics experiments are both enormous and inherently noisy. Collecting, curating, and extracting useful biological information from datasets as well as pre-filtering steps in which low-quality sequences and sequences of probable eukaryotic origin (especially in metagenomes of human origin) are removed.

Assembly

DNA sequence data from genomic and metagenomic projects are essentially the same, but genomic sequence data offers higher coverage while metagenomic data is usually highly non redundant. Furthermore, the increased use of second-generation sequencing technologies with short read lengths means that much of future metagenomic data will be error-prone. Taken in combination, these factors make the assembly of metagenomic sequence reads into genomes difficult and unreliable. Mis-assemblies are caused by the presence of repetitive DNA sequences that make assembly especially difficult because of the difference in the relative abundance of species present in the sample. Mis-assemblies can also involve the combination of sequences from more than one species into chimeric contigs.

Two strategies can be employed for metagenomics samples:

i)   Reference-based assembly (co-assembly)

ii)   De novo assembly

Reference-based assembly can be done with software packages such as Newbler (Roche), AMOS (http://sourceforge.net/projects/amos/), or MIRA. It works well, if the metagenomic dataset contains sequences where closely related reference genomes are available. De novo assembly typically requires larger computational resources. Tools based on the de Bruijn graphs was specifically created to handle very large amounts of data. Machine requirements for the de Bruijn assemblers Velvet or SOAP are still significantly higher than for reference-based assembly (co-assembly), often requiring hundreds of gigabytes of memory in a single machine and run times frequently being days.

In metagenomics single reads have generally lower quality and hence lower confidence in accuracy than do multiple reads that cover the same segment of genetic information. Therefore, merging reads increases the quality of information. So in a complex community with low sequencing depth or coverage, it is unlikely to actually get many reads that cover the same fragment of DNA. Hence assembly may be of limited value for metagenomics. Hence there is a need for metagenomic assembly to obtain high-confidence contigs that enable the study of, e.g., major repeat classes.

Binning

Taxonomic binning is another problem in metagenomics analysis. Sequence binning refers to the separation of sequences into taxon specific groups. A binning step may be part of the assembly process of metagenomic data or may be used for separating the genomes of a few members in order to study the biological processes carried by each one of them. Various algorithms have been developed, which employ two types of information contained within a given DNA sequence.

i) First compositional binning makes use of the fact that genomes have conserved nucleotide composition (e.g. a certain GC or the particular abundance distribution of k-mers).

ii) Secondly, the unknown DNA fragment might encode for a gene and the similarity of this gene with known genes in a reference database can be used to classify and hence bin the sequence.

Important considerations for using any binning algorithm are the type of input data available and the existence of a suitable training datasets or reference genomes. In general, composition-based binning is not reliable for short reads, as they do not contain enough information. It can however be improved, if training datasets (e.g. a long DNA fragment of known origin) exist and that is used to define a compositional classifier. These "training" fragments can either be derived from assembled data or from sequenced fosmids and should ideally contain a phylogenetic marker (such as rRNA gene) that can be used for high-resolution, taxonomic assignment of the binned fragment.

Annotation

For annotation of metagenomics two approaches are used for annotation of coding regions in the assembled contigs. First, if assembly has produced large contigs and reconstructed genomes are the objective of the study then it is preferable to use existing pipelines for genome annotation, such as RAST or IMG. For this, minimal contigs length of 30,000 bp or longer are required. Second, annotation can be performed on the entire community and relies on unassembled reads or short contigs. Here the tools for genome annotation are significantly less useful than those specifically developed for metagenomic analyses.

Experimental Design and Statistical Analysis

For the reduction of sequencing cost and a much wider appreciation of the utility of metagenomics to address fundamental questions in microbial ecology require proper experimental designs with appropriate replication and statistical analysis. The data from multiple metagenomic shotgun-sequencing projects can be reduced to tables, where the columns represent samples and the rows indicate either a taxonomic group or a gene function (or groups thereof) and the fields containing abundance or presence/absence data. As metagenomic data often contain many more species or gene functions then the number of samples taken, so appropriate corrections for multiple hypothesis testing have to be implemented (e.g. Bonferroni correction for t-test based analyses).

Sometimes variation between sample types can be due to true biological variation and technical variation and this should be carefully considered when planning the experiment. One should kept

in mind that many microbial systems are highly dynamic, so temporal aspects of sampling can have a substantial impact on data analysis and interpretation. Taking multiple samples and then pooling them will lose all information on variability and hence will be of little use for statistical purposes. Ultimately, good experimental design of metagenomic projects will facilitate integration of datasets into new or existing ecological theories. One of the ultimate aims of metagenomics is to link functional and phylogenetic information to the chemical, physical, and other biological parameters that characterize an environment.

Sharing and Storage of Data

Data sharing is important for the genomic research, there is a requirement for whole new level of organization and collaboration to provide metadata and centralized services (e.g., IMG/M, CAMERA and MG-RAST) as well as sharing of both data and computational results. Once this has been achieved, researchers will be able to download intermediate processed results from any one of the major repositories for local analysis or comparison. A suite of standard languages for metadata is currently provided by the Minimum Information about any (x) Sequence checklists (MIxS). MIxS is an umbrella term to describe MIGS (the Minimum Information about a Genome Sequence), MIMS (the Minimum Information about a Metagenome Sequence) and MIMARKS (Minimum Information about a MARKer Sequence) and contains standard formats for recording environmental and experimental data. The latest of these checklists, MIMARKS builds on the foundation of the MIGS and MIMS checklists, by including an expansion of the rich contextual information about each environmental sample.

The US National Center for Biotechnology Information (NCBI) is mandated to store all metagenomic data, however, the sheer volume of data being generated means there is an urgent need for appropriate ways of storing vast amounts of sequences. As the cost of sequencing continues to drop while the cost for analysis and storing remains more or less constant, selection of data storage in either biological (i.e. the sample that was sequenced) or digital form in (de-) centralized archives might be required. Ongoing work and successes in compression of (meta-) genomic data, help in the storage of digital information cost-efficiently.

Applications of Metagenomics

Among the enormous applications of metagenomics the most important ones include environmental studies, human health, identification of novel microbes, genes, pathways and mechanisms of their survival, biodegradation of sewage, ocean pollutants, plastics, garbage, energy generation and bio-fuels and biotechnological and industrial implications of the huge meta-sequence data coming out from the unseen microbial communities.

Community Metabolism

In many bacterial communities, natural or engineered (such as bioreactors), there is significant division of labor in metabolism (Syntrophy), during which the waste products of some organisms are metabolites for others. Eg. in methanogenic bioreactor.

Metatranscriptomics

Metagenomics allows researchers to access the functional and metabolic diversity of microbial communities, but it cannot show which of these processes are active. The extraction and analysis of metagenomic mRNA (the metatranscriptome) provides information on the regulation and expression profiles of complex communities apart from its technical difficulties (e g. the short half-life of mRNA).

Viruses

Metagenomic sequencing is particularly useful in the study of viral communities. As viruses lack a shared universal phylogenetic marker (as 16S RNA for bacteria and *archaea*, and 18S RNA for *eukarya*), the only way to access the genetic diversity of the viral community from an environmental sample is through metagenomics. Viral metagenomes (also called viromes) should thus provide more and more information about viral diversity and evolution.

Advantages of Metagenomics in Different Areas

Metagenomics has the potential to advance knowledge in a wide variety of fields. It can also be applied to solve practical challenges in medicine, engineering, agriculture, sustainability and ecology.

Agriculture

As one gram of soil contains around 10^9-10^{10} microbial cells which comprise about one gigabase of sequence information. They perform a wide variety of ecosystem services necessary for plant growth, including fixing atmospheric nitrogen, nutrient cycling, disease suppression, and sequester iron and other metals. Metagenomic approaches can contribute to improved disease detection in crops and livestock and the adaptation of enhanced farming practices which improve crop health by harnessing the relationship between microbes and plants.

Biotechnology

Recent progress in mining the rich genetic resource of non-culturable microbes has led to the discovery of new genes, enzymes, and natural products. The application of metagenomics has allowed the development of fine chemicals, agrochemicals and pharmaceuticals *etc.*

Ecology

Metagenomics can provide valuable insights into the functional ecology of environmental communities. *eg*. Breaking down of defecations helps to release the nutrients in the faeces into a bioavailable form that can be taken up into the food chain.

Environmental remediation

Metagenomics can improve strategies for monitoring the impact of pollutants on ecosystems and for cleaning up contaminated environments. Increased understanding of how microbial communities cope with pollutants improves assessments of the potential of contaminated sites to recover from pollution and increases the chances of bioaugmentation or biostimulation trials to succeed.

Medicine

Metagenomic sequencing of human microbiome helps to determine the core human microbiome. It also helps to understand the changes in the human microbiome that can be correlated with human health, and to develop new technological and bioinformatics tools to support these goals.

Biofuels

Biofuels are fuels derived from biomass conversion, as in the conversion of cellulose contained in corn stalks, switchgrass, and other biomass into cellulosic ethanol. Metagenomic approaches helps

in the analysis of complex microbial communities thus allowing the targeted screening of enzymes with industrial applications in biofuel production, such as glycoside hydrolases.

Conclusion

Metagenomics has changed the way microbiologists approach many problems, redefined the concept of a genome, and accelerated the rate of gene discovery. The potential for application of metagenomics to human benifit seems endless. Metagenomics gives genetic information on potentially novel biocatalysts or enzymes, genomic linkages between function and phylogeny for uncultured organisms and evolutionary profile of community function and structure. It can also be complemented with metatranscriptomic or metaproteomic approaches to describe expressed activities. Metagenomics is also a powerful tool for generating novel hypotheses of microbial functions, remarkable discoveries of proteorhodopsin-based photoheterotrophy or ammonia-oxidizing Archaea. One of the primary goals of metagenomics projects is to perform a comparative analysis of microbial communities residing in diverse ecological niches. Assessing such differences can not only yield valuable insights into the inherent structure of these microbial communities, but can also identify genes/proteins/organisms that may confer specific functional characteristics to a given environment. Insights gained from such comparative studies are expected to have immense potential in several important areas of biological research, ranging from healthcare (e.g., disease diagnostics, detection of pathogenic contamination and characterization of novel pathogens), industrial biotechnology (bio-prospecting) and bio-remediation studies.

References

1. Chen, K.; Pachter, L. (2005). "Bioinformatics for Whole-Genome Shotgun Sequencing of Microbial Communities". *PLoS Computational Biology*, 1 (2): e24, doi:10.1371/journal.pcbi.0010024

2. Field D, Amaral-Zettler L, Cochrane G, et al., (2011). The Genomic Standards Consortium: Minimum information about a marker gene sequence (MIMARKS) and minimum information about any (x) sequence (MIxS) specifications. *PLoS Biol*, 9(6):e1001088.

3. Gilbert J.A., Field D., Huang Y., Edwards R., Li W., Glina P. and Joint I. (2008). Detection of large numbers of novel sequences in the metatranscriptomes of complex marine microbial communities. *PLoS ONE,* 3: e3042.

4. Glass EM, Wilkening J, Wilke A, Antonopoulos D, Meyer F, 2010(1). Using the metagenomics RAST server (MG-RAST) for analyzing shotgun metagenomes. *Cold Spring Harb Protocol*, pdb prot5368.

5. Huson DH, Auch AF, Qi J, Schuster SC, (2007). MEGAN analysis of metagenomic data. *Genome Research*, 17(3):377-386.

6. Kristiansson E, Hugenholtz P, Dalevi D, (2009). ShotgunFunctionalizeR, An Rpackage for functional comparison of metagenomes. *Bioinformatics*, 25(20):2737-2738.

7. Markowitz VM, Ivanova NN, *et al.* (2008). IMG/M: a data management and analysis system for metagenomes. *Nucleic Acids Res*, 36 Database: D534-538.

8. Morris R. M., Nunn B. L., Frazar C., Goodlett D. R., Ting Y. S., Rocap G. (2010). Comparative metaproteomics reveals ocean-scale shifts in microbial nutrient utilization and energy transduction. *ISME Journal,* 4: 673–685.

9. Rho M, Tang H, Ye Y, (2010). FragGeneScan: predicting genes in short and error prone reads. *Nucleic Acids Research*, 38(20):e191.

10. Thomas *et al*., (2012). Metagenomics - a guide from sampling to data analysis. *Microbial Informatics and Experimentation* 2:3.

11. Quast, C., Pruesse, E., Yilmaz, P., Gerken, J., Schweer, T., Yarza, P., & Glöckner, F. O. (2012). The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucleic acids research, 41(D1): D590-D596.

12. Z L Sabree, M R Rondon, and J Handelsman, University of Wisconsin-Madison, Madison, WI, USA (2009). Metagenomics. *Elsevier Inc.*

Metagenomics Data Analysis using QIIME 2

Anu Sharma

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

1. Introduction

QIIME 2 is a completely reengineered microbiome bioinformatics platform based on the popular QIIME platform, which it has replaced. QIIME 2 facilitates comprehensive and fully reproducible microbiome data science, improving accessibility to diverse users by adding multiple user interfaces.
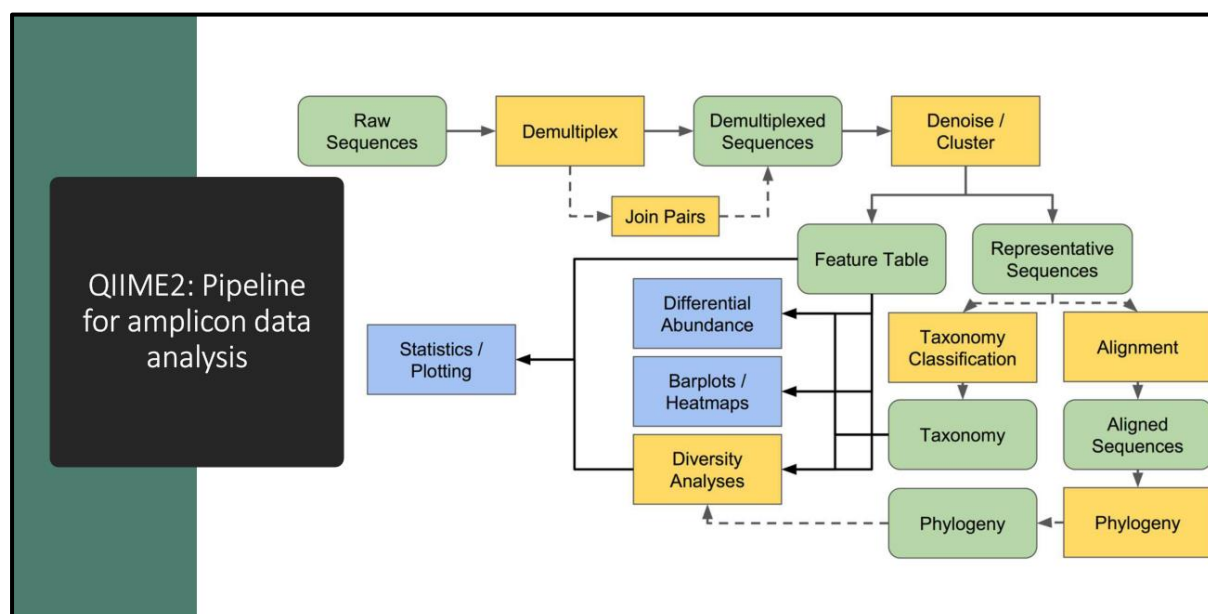

Fig. 1: Pipeline for amplicon data analysis

Key features:

- Integrated and automatic tracking of data provenance
- Semantic type system
- Plugin system for extending microbiome analysis functionality
- Support for multiple types of user interfaces (e.g. API, command line, graphical)

2. Data files: QIIME 2 artifacts

Data produced by QIIME 2 exist as QIIME 2 artifacts. A QIIME 2 artifact contains data and metadata. The metadata describes things about the data, such as its type, format, and how it was generated (provenance). A QIIME 2 artifact typically has the .qza file extension when stored in a file.

Since QIIME 2 works with artifacts instead of data files (e.g. FASTA files), data can be imported at any step in an analysis, though typically it start by importing raw sequence data. QIIME 2 also has tools to export data from an artifact. By using QIIME 2 artifacts instead of simple data files, QIIME 2 can automatically track the type, format, and provenance of data for

researchers. Using artifacts instead of data files enables researchers to focus on the analyses they want to perform, instead of the particular format the data needs to be in for an analysis.

2.1 Data files: visualizations

Visualizations are another type of data generated by QIIME 2. When written to disk, visualization files typically have the .qzv file extension. Visualizations contain similar types of metadata as QIIME 2 artifacts, including provenance information. Similar to QIIME 2 artifacts, visualizations are standalone information that can be archived or shared with collaborators.

In contrast to QIIME 2 artifacts, visualizations are terminal outputs of an analysis, and can represent, for example, a statistical results table, an interactive visualization, static images, or really any combination of visual data representations. Since visualizations are terminal outputs, they cannot be used as input to other analyses in QIIME 2.

2.2 Semantic types

Every artifact generated by QIIME 2 has a semantic type associated with it. Semantic types enable QIIME 2 to identify artifacts that are suitable inputs to an analysis. For example, if an analysis expects a distance matrix as input, QIIME 2 can determine which artifacts have a distance matrix semantic type and prevent incompatible artifacts from being used in the analysis (e.g. an artifact representing a phylogenetic tree). Semantic types also help users avoid semantically incorrect analyses. For example, a feature table could contain presence/absence data (i.e., a 1 to indicate that an OTU was observed at least one time in a given sample, and a 0 to indicate than an OTU was not observed at least one time in a given sample). However, if that feature table were provided to an analysis computing a quantitative diversity metric where OTU abundances are included in the calculation (e.g., weighted UniFrac), the analysis would complete successfully, but the result would not be meaningful.

This guide assumes that QIIME 2 have been installed using one of the procedures in the install documents at https://docs.qiime2.org/2022.8/install/.

3.   Obtaining and importing data

```
wget \
  -O 'emp-single-end-sequences.zip' \
  'https://docs.qiime2.org/2021.11/data/tutorials/moving-pictures-usage/emp
-single-end-sequences.zip'

unzip -d emp-single-end-sequences emp-single-end-sequences.zip
```

```
qiime tools import \
  --type 'EMPSingleEndSequences' \
  --input-path emp-single-end-sequences \
  --output-path emp-single-end-sequences.qza
```

4. Demultiplexing sequences

To demultiplex sequences we need to know which barcode sequence is associated with each sample. This information is contained in the sample metadata file. You can run the following commands to demultiplex the sequences (the demux emp-single command refers to the fact that these sequences are barcoded according to the Earth Microbiome Project protocol, and are single-end reads). The demux.qza QIIME 2 artifact will contain the demultiplexed sequences.

```
qiime demux emp-single \
  --i-seqs emp-single-end-sequences.qza \
  --m-barcodes-file sample-metadata.tsv \
  --m-barcodes-column barcode-sequence \
  --o-per-sample-sequences demux.qza \
  --o-error-correction-details demux-details.qza
```

After demultiplexing, it's useful to generate a summary of the demultiplexing results. This allows you to determine how many sequences were obtained per sample, and also to get a summary of the distribution of sequence qualities at each position in your sequence data.

```
qiime demux summarize \
  --i-data demux.qza \
  --o-visualization demux.qzv
```

5. Sequence quality control and feature table construction

QIIME 2 plugins are available for several quality control methods, including DADA2, Deblur, and basic quality-score-based filtering. In this tutorial we present this step using DADA2. These steps are interchangeable, so you can use whichever of these you prefer. The result of both of these methods will be a FeatureTable[Frequency] QIIME 2 artifact, which contains counts (frequencies) of each unique sequence in each sample in the dataset, and a FeatureData[Sequence] QIIME 2 artifact, which maps feature identifiers in the FeatureTable to the sequences they represent.

```
qiime dada2 denoise-single \
  --i-demultiplexed-seqs demux.qza \
  --p-trim-left 0 \
  --p-trunc-len 120 \
  --o-representative-sequences rep-seqs.qza \
  --o-table table.qza \
  --o-denoising-stats stats.qza

qiime metadata tabulate \
  --m-input-file stats.qza \
  --o-visualization stats.qzv
```

6. FeatureTable and FeatureData summaries

```
qiime feature-table summarize \
  --i-table table.qza \
  --m-sample-metadata-file sample-metadata.tsv \
  --o-visualization table.qzv
qiime feature-table tabulate-seqs \
  --i-data rep-seqs.qza \
  --o-visualization rep-seqs.qzv
```

7. Generate a tree for phylogenetic diversity analyses

```
qiime phylogeny align-to-tree-mafft-fasttree \
```

```
  --i-sequences rep-seqs.qza \
  --output-dir phylogeny-align-to-tree-mafft-fasttree
```

8. Alpha and beta diversity analysis

```
qiime diversity core-metrics-phylogenetic \
  --i-phylogeny phylogeny-align-to-tree-mafft-fasttree/rooted_tree.qza \
  --i-table table.qza \
  --p-sampling-depth 1103 \
  --m-metadata-file sample-metadata.tsv \
  --output-dir diversity-core-metrics-phylogenetic
```

9. Taxonomic analysis

```
wget \
  -O 'gg-13-8-99-515-806-nb-classifier.qza' \
  'https://docs.qiime2.org/2021.11/data/tutorials/moving-pictures-usage/gg-13-8-99-515-806-nb-classifier.qza'

qiime feature-classifier classify-sklearn \
  --i-classifier gg-13-8-99-515-806-nb-classifier.qza \
  --i-reads rep-seqs.qza \
  --o-classification taxonomy.qza

qiime metadata tabulate \
  --m-input-file taxonomy.qza \
  --o-visualization taxonomy.qzv

qiime taxa barplot \
  --i-table table.qza \
  --i-taxonomy taxonomy.qza \
  --m-metadata-file sample-metadata.tsv \
  --o-visualization taxa-bar-plots.qzv
```

References:

1. https://docs.qiime2.org/2022.8/tutorials/moving-pictures-usage/
2. https://docs.qiime2.org/2022.8/concepts/#data-files-qiime-2-artifacts
3. Mehrbod Estaki,Lingjing Jiang,Nicholas A. Bokulich,Daniel McDonald,Antonio González,Tomasz Kosciolek,Cameron Martino,Qiyun Zhu,Amanda Birmingham,Yoshiki Vázquez-Baeza,Matthew R. Dillon,Evan Bolyen,J. Gregory Caporaso,Rob Knight (2020). QIIME 2 Enables Comprehensive End-to-End Analysis of Diverse Microbiome Data and Comparative Studies with Publicly Available Data. Current Protocols in Bioinformatics. *Current Protocols in Bioinformaticse100, Volume 70*, Published in Wiley Online Library (wileyonlinelibrary.com).doi: 10.1002/cpbi.100

Statistical Analysis of Metagenomics Data

Ritwika Das

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

❑ **Statistical Analysis of Metagenomic Profiles**

Taxonomic and functional differences between metagenomic samples can highlight the influence of ecological factors on patterns of microbial life in a wide range of habitats. Statistical hypothesis tests help to distinguish ecological influences from sampling artifacts, but knowledge of only the p-value is insufficient to make inferences about biological relevance. Biological relevance of a feature requires consideration of effect sizes and their associated confidence intervals. Interpretation of statistical results can also benefit from transforming raw p-values to superior interpretations and by allowing interactive filtering that permits focusing on features with specific statistical properties.

p-value indicates the probability of an observed difference occurring simply by chance. Features in a profile with p-values below 0.05 are termed as statistically significant and can reasonably be assumed to be enriched in one of the metagenomes due to ecological or taxonomic differences as opposed to being the result of a sampling artifact. Fisher's exact test uses hypergeometric distribution to efficiently calculate the exact p-value without the requirement of all possible permutation of sequences in a pair of metagenomic samples. The chi-square test and G-test are well-known large sample approximations to Fisher's exact test. Barnard's test is computationally prohibitive for the majority of features in a typical metagenomic profile. So, we need to decide between an approximation to Barnard's exact test (*e.g.*, bootstrapping) and Fisher's exact test.

A typical metagenomic profile consists of several hundred features. When performing multiple hypothesis tests, it is useful to modify the p-values so that they reflect a particular interpretation. If we wish to examine a list of features where the probability of observing one or more false positive is less than a specified probability, we can use a correction method. Commonly applied correction methods include Bonferroni, Holm-Bonferroni and Šidák (Abdi, 2007). Alternatively, during exploratory analysis, we may be willing to accept a specific percentage of false positives. This can be achieved using the Benjamini–Hochberg false discovery rate (FDR) procedure (Benjamini and Hochberg, 1995) or the Storey FDR approach (Storey and Tibshirani, 2003). These approaches complement each other while performing an exploratory analysis. The list of significant features obtained without any multiple test correction method gives us an initial global look at those features which may be differentially abundant between our samples. An FDR approach can be used to refine this initial list and to make the number of expected false positives explicit. Finally, a correction technique can be applied to focus our attention to only those features where the observed enrichment or depletion is highly unlikely to be a sampling artifact.

❑ **Effect Size and Confidence Intervals**

To assess if a feature is of biological relevance, we should consider the magnitude of the observed difference (*i.e.*, an effect size statistic). An arbitrarily small effect can be statistically significant if the sample sizes are sufficiently large. So, biological significance of a feature must be supported by effect size statistics as well as p-values.

Table 1: Contingency table summarising data for a feature of interest

	Sample 1	Sample 2	
Sequences in feature	x_1	x_2	$R_1 = x_1 + x_2$
Sequences in other features	y_1	y_2	$R_2 = y_1 + y_2$
Total assigned sequences	$C_1 = x_1 + y_1$	$C_2 = x_2 + y_2$	$N = C_1 + C_2$

Table 2: Effect size statistics of a feature of interest

Effect size statistic	Equation
Difference between proportions	$DP = p_1 - p_2$
Ratio of proportions	$RP = p_1/p_2$
OR	$OR = (x_1/y_1)/(x_2/y_2)$

$p_1 = x_1/C_1$, $p_2 = x_2/C_2$; RP is often referred to as relative risk.

The most intuitive effect size statistic is the difference between proportions (DP) of sequences assigned to a given feature in the two samples. Ratio of proportions (RP) is also a measure that provides complementary information to the DP. Consideration of multiple effect size statistics is often essential while assessing biological relevance as features can have a small (or, large) DP, but a large (or, small) RP. The odds ratio (OR) has many desirable mathematical properties. However, RP is preferred over OR due to the difficulty in interpretation of the latter.

Confidence interval (CI) indicates the range of effect size values that have a specified probability of being compatible with the observed data. A 95% CI gives a lower and upper bound in which the true effect size will be contained 19 times out of 20. There is a close relationship between p-values and CI. CI that encompasses the identity effect size (*e.g.*, DP = 0 or RP = OR = 1) will have a p-value > (1 – the coverage of the CI) (*i.e.*, a p-value ≥ 0.05 for a 95% CI). If the identity effect size is outside the CI, the p-value will be ≤ 0.05 for a 95% CI. Critically, CI provides a mean to infer the biological relevance of a feature even when it is marginally statistically significant.

❑ **Software: STAMP (Parks *et al*., 2010)**

❑ **Concept of STAMP**
STAMP is a open source software package for analyzing various metagenomic profiles, *viz.,* taxonomic profiles indicating the number of marker genes assigned to different taxonomic units or functional profiles indicating the number of sequences assigned to different subsystems or pathways. A user-friendly, graphical interface permits easy exploration of statistical results and generation of publication quality plots for inferring biological relevant features present in a metagenomic profile. STAMP facilitates statistical hypothesis tests to identify features (*e.g.*, taxa or metabolic pathways) that differ significantly between

1. Pairs of profiles (Two Sample)
2. Sets of profiles organized into two groups (Two Groups)
3. Sets of profiles organized into multiple groups (Multiple Groups)

❑ **Software Installation**
STAMP is implemented in Python and can be installed in any operating system, *i.e.*, Windows/ MacOS/Linux. Source codes and executable binary file can be downloaded from the following link:

https://github.com/dparks1134/STAMP/releases/tag/v2.1.3
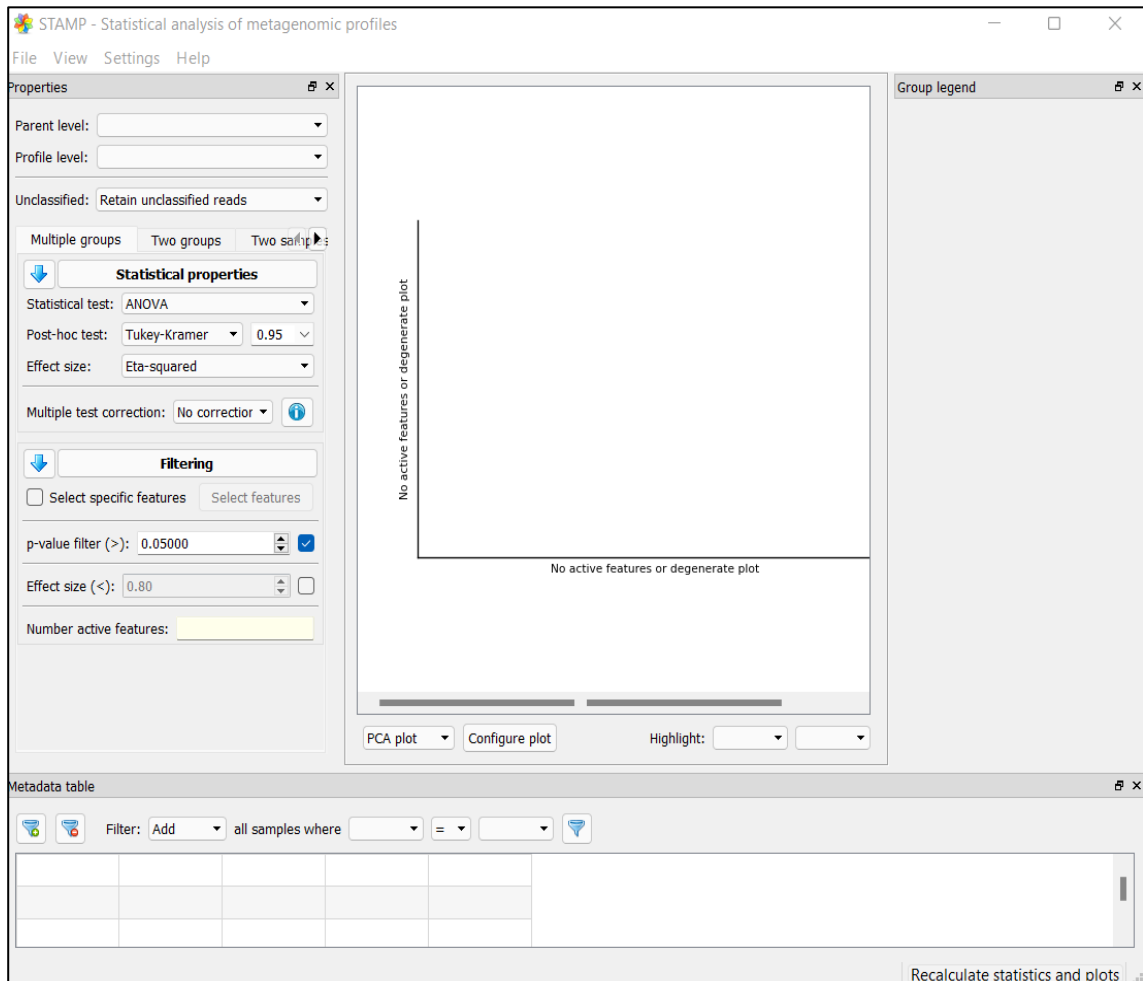
Upon installation of the software, some example datasets also get downloaded in the installation folder. Here, profile and metadata for the dataset *EnterotypeArumugam* is used for the demonstration of this software.

❑ **Input files**
STAMP requires 2 input files:
1. Metagenomic profile file
2. Metadata file

1. **Metagenomic profile file**:
   STAMP can analyze both taxonomic and functional profiles. User defined input files should be text files in tab-separated values (TSV) format. It can contain hierarchical profile information for two or more samples. The first row of the file contains headers for each column. First few columns indicate the hierarchical structure of a feature in an arrangement of the highest to the lowest level. There are no restrictions on the depth of the hierarchy but it must form a strict tree structure. Reads that have an unknown classification at any point in the hierarchy should be marked as unclassified (case insensitive). The parent of a classified child in the hierarchy must also be classified. Other columns contain abundance values of features in different samples.

STAMP can analyze taxonomic or functional profiles obtained from MG-RAST software in *.tsv* format. First column of this MG-RAST profile is the *metagenome* column. To perform statistical analysis using STAMP, MG-RAST profile needs to be converted into a STAMP compatible profile (*.spf*) using: File → Create STAMP profile from... → MG-RAST profile

Similarly, taxonomic and functional profiles from BIOM, Rita, CoMet and mothur can also be analyzed using STAMP. It can directly process abundance profiles for multiple samples obtained from the JGI IMG/M web portal. COG profiles from IMG/M do not contain information about which COG category or higher level class a COG belongs to. STAMP can add this information using: Append COG categories to IMG/M profile.

2. **Metadata file**:
   STAMP requires additional data associated with each sample to perform statistical analysis of metagenomic samples organized in two or more groups. These additional information are provided in a metadata file in *.tsv* format. First column of this file indicates Sample Ids. Other columns provide information about various grouping categories and corresponding values.

Grouping categories

```
   Enterotypes.metadata.tsv
 1  Sample Id   Enterotype   Nationality Clinical Status Gender   Project Clinical Status [filtered]  Nationality [filtered]  Gender [filtered]
 2  AM-AD-1 Unclassified    american    healthy F   gill06  na  na  na
 3  AM-AD-2 Unclassified    american    healthy M   gill06  na  na  na
 4  AM-F10-T1   Enterotype 3 - twin american    obese   F   turnbaugh09 na  na  na
 5  AM-F10-T2   Enterotype 3    american    obese   F   turnbaugh09 obese   na  F
 6  DA-AD-1 Enterotype 2    danish  healthy F   MetaHIT healthy danish  F
 7  DA-AD-2 Enterotype 3    danish  healthy M   MetaHIT healthy danish  M
 8  DA-AD-3 Enterotype 3    danish  obese   F   MetaHIT obese   danish  F
 9  DA-AD-4 Enterotype 2    danish  obese   M   MetaHIT obese   danish  M
10  ES-AD-1 Enterotype 1    spanish CD  F   MetaHIT CD  spanish F
11  ES-AD-2 Enterotype 2    spanish healthy M   MetaHIT healthy spanish M
12  ES-AD-3 Enterotype 2    spanish UC  F   MetaHIT UC  spanish F
13  ES-AD-4 Enterotype 3    spanish healthy F   MetaHIT healthy spanish F
14  FR-AD-1 Enterotype 3    french  healthy M   MicroObes   healthy french  M
15  FR-AD-2 Enterotype 3    french  healthy M   MicroObes   healthy french  M
16  FR-AD-3 Enterotype 1    french  healthy M   MicroObes   healthy french  M
17  FR-AD-4 Enterotype 3    french  healthy M   MicroObes   healthy french  M
18  FR-AD-5 Enterotype 3    french  obese   M   MicroObes   obese   french  M
19  FR-AD-6 Enterotype 2    french  obese   M   MicroObes   obese   french  M
20  FR-AD-7 Enterotype 3    french  obese   M   MicroObes   obese   french  M
21  FR-AD-8 Enterotype 3    french  obese   M   MicroObes   obese   french  M
22  IT-AD-1 Enterotype 3    italian elderly F   MicroAge    elderly italian F
23  IT-AD-2 Enterotype 3    italian elderly M   MicroAge    elderly italian M
24  IT-AD-3 Enterotype 3    italian elderly F   MicroAge    elderly italian F
25  IT-AD-4 Enterotype 2    italian elderly M   MicroAge    elderly italian M
26  IT-AD-5 Enterotype 3    italian elderly M   MicroAge    elderly italian M
27  IT-AD-6 Enterotype 3    italian elderly F   MicroAge    elderly italian F
28  JP-AD-1 Enterotype 1    japanese    healthy M   kurokawa07  healthy japanese    M
29  JP-AD-2 Enterotype 3    japanese    healthy F   kurokawa07  healthy japanese    F
30  JP-AD-3 Enterotype 3    japanese    healthy M   kurokawa07  healthy japanese    M
31  JP-AD-4 Enterotype 1    japanese    healthy F   kurokawa07  healthy japanese    F
32  JP-AD-5 Enterotype 3    japanese    healthy M   kurokawa07  healthy japanese    M
33  JP-AD-6 Enterotype 1    japanese    healthy F   kurokawa07  healthy japanese    F
34  JP-AD-7 Enterotype 1    japanese    healthy M   kurokawa07  healthy japanese    M
35  JP-AD-8 Enterotype 1    japanese    healthy M   kurokawa07  healthy japanese    M
36  JP-AD-9 Enterotype 1    japanese    healthy F   kurokawa07  healthy japanese    F
37  JP-IN-1 Infant  japanese    healthy F   kurokawa07  na  na  na
38  JP-IN-2 Infant  japanese    healthy M   kurokawa07  na  na  na
39  JP-IN-3 Infant  japanese    healthy M   kurokawa07  na  na  na
40  JP-IN-4 Infant  japanese    healthy F   kurokawa07  na  na  na
```

If metadata file is not provided, STAMP assumes all samples contained in a single group and performs only "Two Sample" tests.

❑ **Analyzing Metagenomic Profiles:**
Upload both profile file and metadata file to the STAMP software to perform various statistical analysis for multiple groups/ two groups/ two samples.

❖ **Statistical Analysis for Multiple Groups**
Statistical properties can be set through the Properties window. It helps to set a number of properties related to performing statistical tests:

- **Parent Level**: The proportion of sequences assigned to a feature will be calculated relative to the total number of sequences assigned to its parent category. By default, it is set as *Entire sample*.
- **Profile Level**: The hierarchical level at which statistical tests will be performed. It facilitates analysis of metagenomic profile at different depths of the hierarchy.
- **Unclassified**: Unclassified sequences can be handled in 3 ways: a) retained in the profile (Retain unclassified reads), removed from the profile (Remove unclassified reads), or removed from consideration except when calculating a profile (Use only for calculating frequency profiles).
- **Statistical Properties**: The statistical test, post-hoc test along with the confidence interval width, effect size, and multiple test correction method to use can be specified in this section. A list of methods provided in STAMP for analyzing multiple groups is given in Table 3.

- **Filtering**: This section provides a number of filters for identifying features that satisfy a set of criteria (*i.e.*, desired p-value and effect size).

Table 3: Multiple groups statistical techniques available in STAMP

Statistical hypothesis tests	Comments	References
ANOVA	An analysis of variance (ANOVA) is a method for testing whether or not the means of several groups are all equal. It can be seen as a generalization of the t-test to more than two groups.	Bluman, 2007
Kruskal-Wallis H-test	A non-parametric method for testing whether or not the median of several groups are all equal. It considers the rank order of each sample and not the actual proportion of sequences associated with a feature. This has the benefit of not assuming the data is normally distributed. Each group must contain at least 5 samples to apply this test.	Bluman, 2007
Post-hoc tests		
Games-Howell	Used to determine which means are significantly different when an ANOVA produces a significant p-value. This post-hoc test is designed for use when variances and group sizes are unequal. It is preferable to Tukey-Kramer when variances are unequal and group sizes are small, but it more computationally expensive.	
Scheffé	A general post-hoc test for considering all possible contrasts unlike the Tukey-Kramer method which considers only pairs of means. Currently, STAMP only considers pairs of means so the Tukey-Kramer method is preferred. In general, this test is highly conservative.	
Tukey-Kramer	Used to determine which means are significantly different when an ANOVA produces a significant p-value. It considers all possible pairs of means while controlling the familywise error rate (*i.e.*, accounting for multiple comparisons). In general, we recommend using the Games-Howell post-hoc test when reporting final results and the Tukey-Kramer method for exploratory analysis since it is less computationally intensive. The Tukey-Kramer may also be preferred as it is more widely used and known amongst researchers.	Bluman, 2007
Welch's (uncorrected)	Simple performs Welch's t-test on each possible pair of means. No effort is made to control the familywise error rate.	
Multiple test correction methods		
Benjamini-Hochberg FDR	Initial proposal for controlling false discovery rate instead of the familywise error. Step-down procedure.	Benjamini and Hochberg, 1995
Bonferroni	Classic method for controlling the familywise error. Often criticized as being too conservative.	Adbi, 2007
Šidák	Less common method for controlling the familywise error rate. Uniformly more powerful than Bonferroni, but requires the assumption that individual tests are independent.	Adbi, 2007
Storey's FDR	Recent method used to control the false discovery rate. More powerful than the Benjamini-Hochberg method. Requires estimating certain parameters and is more computationally expensive than the Benjamini-Hochberg approach.	Storey and Tibshirani, 2003 Storey *et al.*, 2004

❖ **Graphical exploration of results:**
Statistical analysis results can be graphically represented with the help of various plots. The Group legend window helps to select the particular grouping category for which we want to explore the results.
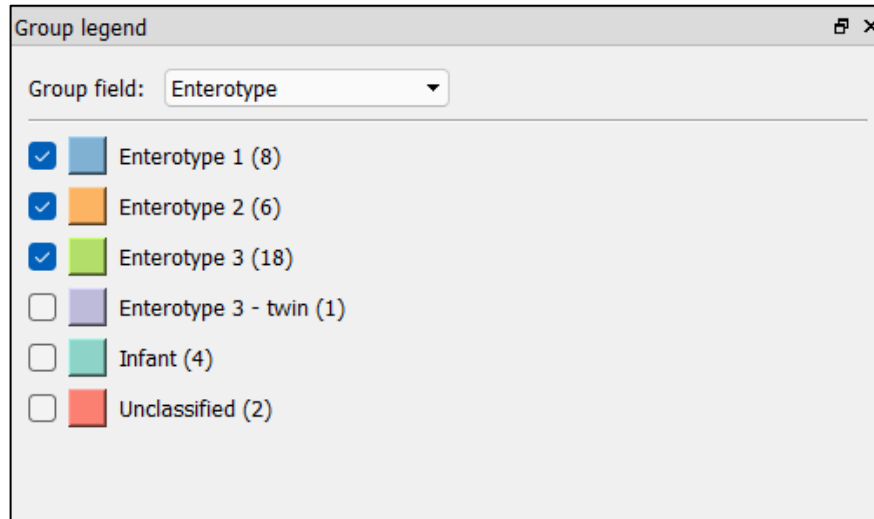
The following plots can be generated for exploring the analysis results of multiple groups:

• **PCA plot**: Principal component analysis (PCA) plot of the samples. Clicking on a marker within the plot indicates the sample represented by the marker. Markers of different colours belong to different groups.
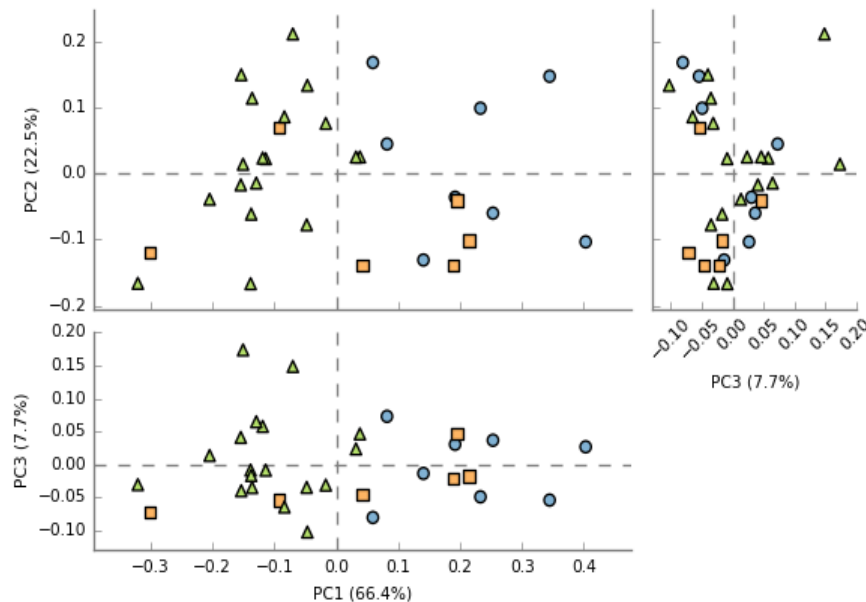
- **Heatmap plot**: It represents the proportion of sequences assigned to each feature in every sample. Dendrograms can be shown along the sides of the heatmap and are used to cluster both the features and samples.
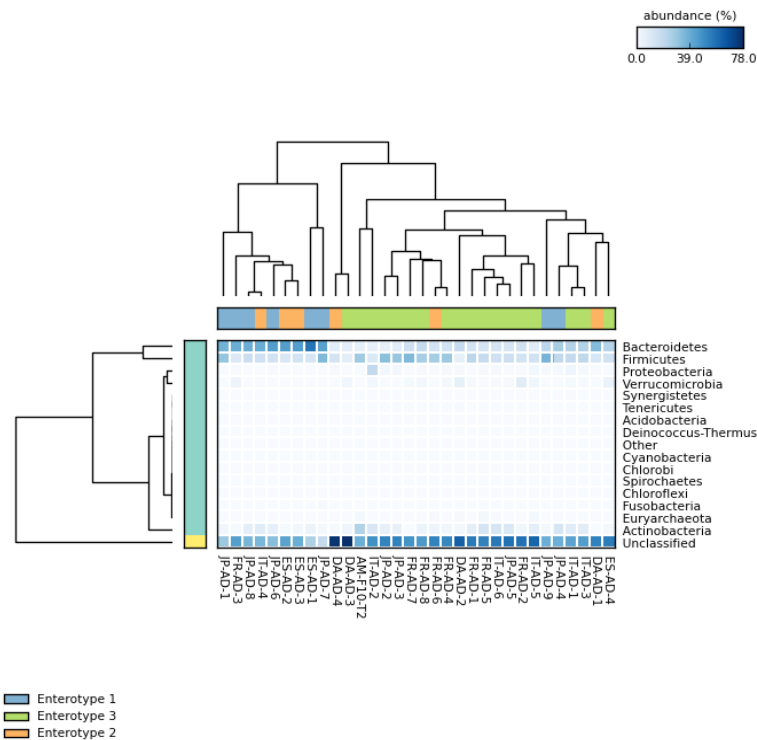
- **Bar plot**: Bar plot represents the proportion of sequences assigned to a particular feature in every sample.

Feature	Eta-squared	p-value	Corrected p-value
Acidobacteria	-1.000	1.000	1.000
Actinobacteria	0.133	0.125	0.125
Bacteroidetes	0.582	3.20e-6	3.20e-6
Chlorobi	0.025	0.692	0.692
Chloroflexi	0.035	0.594	0.594
Cyanobacteria	0.063	0.392	0.392
Deinococcus-Thermus	0.025	0.692	0.692
Euryarchaeota	0.110	0.183	0.183
Firmicutes	0.065	0.376	0.376
Fusobacteria	0.102	0.210	0.210
Other	0.140	0.113	0.113
Proteobacteria	0.050	0.475	0.475
Spirochaetes	0.025	0.689	0.689
Synergistetes	0.031	0.636	0.636
Tenericutes	0.033	0.614	0.614
Unclassified	0.450	1.73e-4	1.73e-4
Verrucomicrobia	0.168	0.070	0.070

- **Box plot**: It is similar to a bar plot. Box plot provides a more concise summary of the distribution of sequence proportions of a feature in various groups. The box-and-whiskers graphics show the median of the data as a line, the mean of the data as a star, the 25th and 75th percentiles of the data as the top and bottom of the box, and uses whiskers to indicate the most extreme data point within 1.5*(75th – 25th percentile) of the median. Data points outside of the whiskers are shown as crosses.

Feature	Eta-squared	p-value	Corrected p-value
Acidobacteria	-1.000	1.000	1.000
Actinobacteria	0.133	0.125	0.125
Bacteroidetes	0.582	3.20e-6	3.20e-6
Chlorobi	0.025	0.692	0.692
Chloroflexi	0.035	0.594	0.594
Cyanobacteria	0.063	0.392	0.392
Deinococcus-Thermus	0.025	0.692	0.692
Euryarchaeota	0.110	0.183	0.183
Firmicutes	0.065	0.376	0.376
Fusobacteria	0.102	0.210	0.210
Other	0.140	0.113	0.113
Proteobacteria	0.050	0.475	0.475
Spirochaetes	0.025	0.689	0.689
Synergistetes	0.031	0.636	0.636
Tenericutes	0.033	0.614	0.614
Unclassified	0.450	1.73e-4	1.73e-4
Verrucomicrobia	0.168	0.070	0.070

- **Post hoc plot**: Upon rejection of the null hypothesis, post hoc tests are performed to identify which pairs of groups are differing significantly from each other. Post hoc plot shows the results of such a test. It provides p-value and effect size measure for each pair of groups for a particular feature.
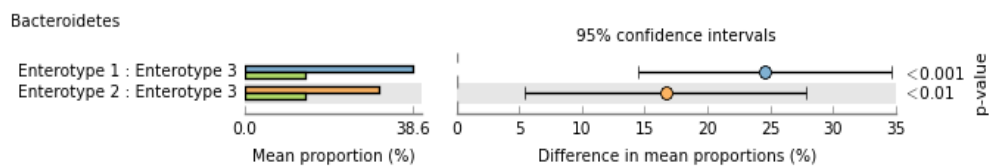
Each of these plots provides a number of customization options. To customize a plot, click the Configure plot button below the plot. Plots can also be sent to a new window using the Send plot to window command under the View menu. This allows multiple plots to be viewed at once. Plots can be saved in raster (PNG) and vector (PDF, PS, EPS, SVG) formats (File → Save plot).

❖ **Statistical Analysis for Two Groups**

To analyze a pair of groups, click on the Two groups tab in the Properties window. In the Profile section, we have to specify which pair of groups will be analyzed. Data points of these 2 groups will be represented by 2 different colours. Groupings are determined by the value of the Group field present in the Group legend window. Here, the filtering section provides a large number of filters for identifying features that satisfy a set of criteria.

Sequence filter removes features that have been assigned fewer than the specified number of sequences. Parent sequence filter does the filtering of sequence counts within parental categories. Effect size filters remove features with small effect sizes. Here, two different effect size statistics are used. It allows one to filter features based on both absolute (*i.e.*, difference between proportions) and relative (*i.e.*, ratio of proportions) measure of effect size.

A list of methods for statistical analysis of metagenomic profiles present in two groups is given in Table 4.

Table 4: Two groups statistical techniques available in STAMP

Statistical hypothesis tests	Comments	References
t-test (equal variance)	Student's t-test which explicitly assumes the two groups have equal variance. When this assumption can be made, this test is more powerful than Welch's t-test.	Bluman, 2007
Welch's t-test	A variation of Student's t-test that is intended for use when the two groups cannot be assumed to have equal variance.	Bluman, 2007
White's non-parametric t-test	Non-parametric test proposed by White *et al.* for clinical metagenomic data. This test uses a permutation procedure to remove the normality assumption of a standard t-test. In addition, it uses a heuristic to identify sparse features which are handled with Fisher's exact test and a pooling strategy when either group consists of less than 8 samples. See White *et al.*, 2009 for details. For large datasets this test can be computationally expensive. It may help to reduce the number of replicates performed which can be set in the Preferences->Settings dialog.	White *et al.*, 2009
Confidence interval methods		
DP: t-test inverted	Only available when using the equal variance t-test. Provides confidence intervals by inverting the equal variance t-test.	
DP: Welch's inverted	Only available when using Welch's t-test. Provides confidence intervals by inverting Welch's t-test.	
DP: bootstrap	Only available when using White's non-parametric t-test. Provides confidence intervals using a percentile bootstrapping method. If White's non-parametric t-test defaults to using Fisher's exact test, confidence intervals are obtained using the Asymptotic with CC approach (see Table 3).	
Multiple test correction methods		
Benjamini-Hochberg FDR	Initial proposal for controlling false discovery rate instead of the familywise error. Step-down procedure.	Benjamini and Hochberg, 1995
Bonferroni	Classic method for controlling the familywise error. Often criticized as being too conservative.	Adbi, 2007
Šidák	Less common method for controlling the familywise error rate. Uniformly more powerful than Bonferroni, but requires the assumption that individual tests are independent.	Adbi, 2007
Storey's FDR	Recent method used to control the false discovery rate. More powerful than the Benjamini-Hochberg method. Requires estimating certain parameters and is more computationally expensive than the Benjamini-Hochberg approach.	Storey and Tibshirani, 2003 Storey *et al.*, 2004

❖ **Graphical exploration of results:**

Similar to multiple groups, here, bar plot, box plot, PCA plot and heatmap plot can be generated to explore the result of statistical analysis for two groups.

208
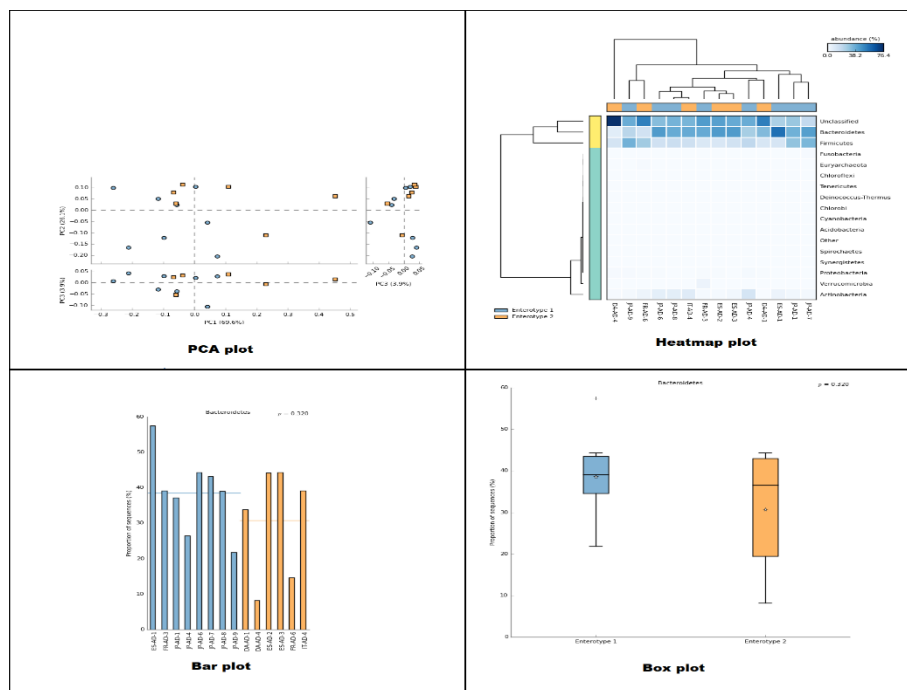
Other plots:

- **Scatter plot**:
  It indicates the mean proportion of sequences within each group which are assigned to each feature. This plot is useful for identifying features that are clearly enriched in one of the two groups. The spread of the data within each group can be shown in various ways (*e.g*., standard deviation, minimum and maximum proportions).
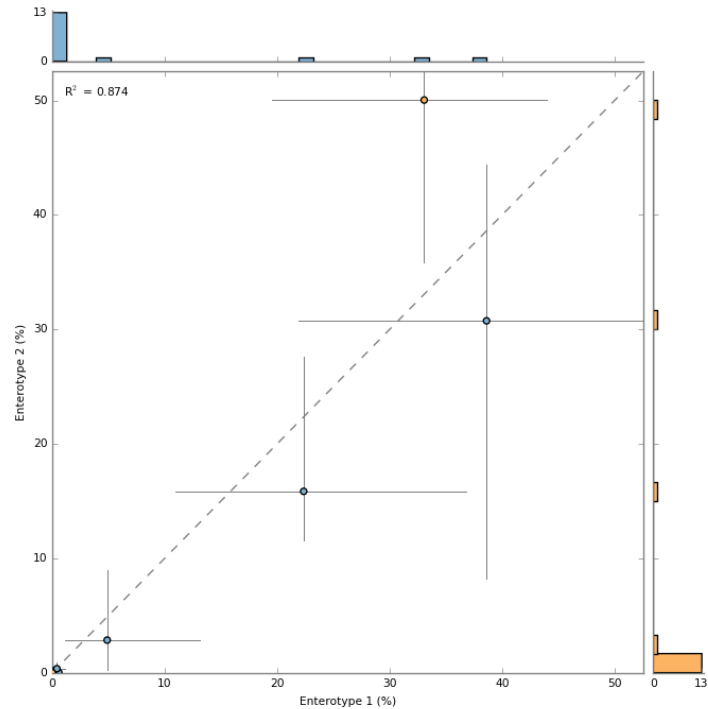
- **Extended error bar plot**:
  It indicates the difference in mean proportion between two groups along with the associated confidence interval of this effect size and the p-value of the specified statistical test. In addition, a bar plot indicates the proportion of sequences assigned to a feature in each group of samples.
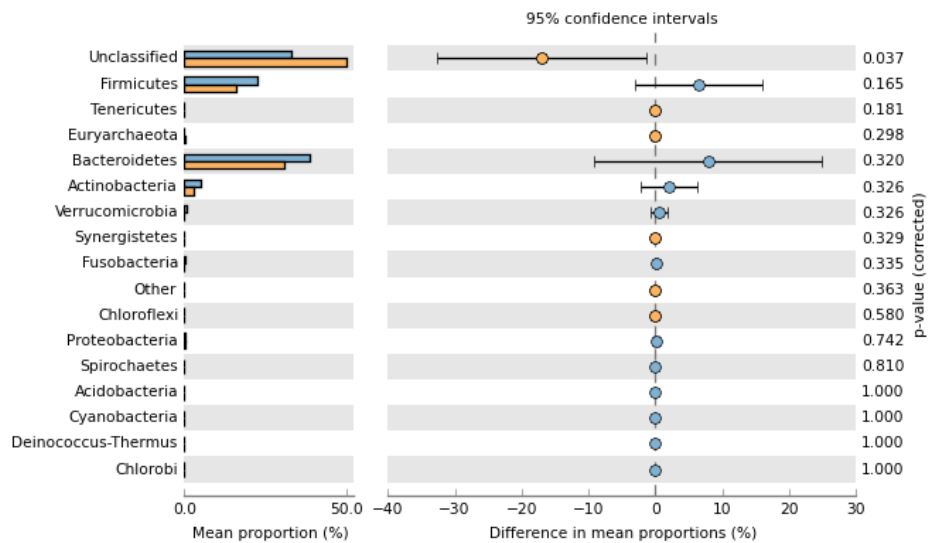
❖ **Statistical Analysis for Two Samples**
To analyze a pair of samples, click on the Two samples tab in the Properties window. The Profile section is used to specify which pair of samples will be analyzed. Data points (features) belonging to these 2 samples will be represented by 2 different colours.

Similar to the previous analyses, various statistical properties and filtering criteria can be explicitly mention for the analysis of metagenomic profiles belonging to two different samples.
A list of statistical techniques for the analysis of metagenomic profiles belonging to two different samples is given in Table 5.

Table 5: Two samples statistical techniques available in STAMP

| Statistical hypothesis tests	Comments	References
Bootstrap	A rough non-parametric approximation to Barnard's exact test. Assumes sampling with replacement.	Manly, 2007
Chi-square	Large sample approximation to Fisher's exact test. Generally liberal compared to Fisher's.	Cochran, 1952 Agresti, 1992
Chi-square with Yates'	Large sample approximation to Fisher's exact test which has been corrected to account for the discrete nature of the distribution it is approximating. Generally conservative compared to Fisher's.	Yates, 1934
Difference between proportions	Z-test. Large sample approximation to Barnard's exact test.	Agresti, 1990
Fisher's exact test[1]	Conditional exact test where p-values are calculated using the 'minimum-likelihood' approach. Computationally efficient even for large metagenomic samples. Widely used and understood.	Agresti, 1990 Rivals et al., 2007
G-test	Large sample approximation to Fisher's exact test. Often considered more appropriate than the Chi-square approximation. Generally liberal compared to Fisher's.	Agresti, 1990
G-test with Yates'	Large sample approximation to Fisher's exact test which has been corrected to account for the discrete nature of the distribution it is approximating. Generally conservative compared to Fisher's.	Yates, 1934
G-test (w/Yates') + Fisher's	Applied Fisher's exact test if any entry in the contingency table is less than 20. Otherwise, the G-test with Yates' continuity correction is used. For clarity, we recommend reporting final results using just Fisher's exact test. However, it is far more efficient to explore the data using this hybrid statistical test.	Agresti, 1990 Rivals et al., 2007 Yates, 1934
Hypergeometric[1]	Conditional exact test where p-values are calculated using the 'doubling' approach. More computationally efficient than the 'minimum-likelihood' approach, but the latter approach is more commonly used by statistical packages (i.e., R and StatXact). Our results suggest the doubling approach is generally more conservative than the minimum-likelihood approach.	Rivals et al., 2007
Permutation	Approximation to Fisher's exact test. Assumes sampling without replacement.	Manly, 2007
Confidence interval methods		
DP: Asymptotic	Standard large sample method.	Newcombe, 1998
DP: Asymptotic with CC	As above, with a continuity correction to account for the discrete nature of the distribution being approximated.	Newcombe, 1998
DP: Newcombe-Wilson	Method recommended by Newcombe in a comparison of seven asymptotic approaches.	Newcombe, 1998
OR: Haldane adjustment	Standard large sample method with a correction to handle degenerate cases.	Bland, 2000; Lawson, 2004; Agresti, 1999
RP: Asymptotic	Standard large sample method.	Agresti, 1990
Multiple test correction methods		
Benjamini-Hochberg FDR	Initial proposal for controlling false discovery rate instead of the familywise error. Step-down procedure.	Benjamini and Hochberg, 1995
Bonferroni	Classic method for controlling the familywise error. Often criticized as being too conservative.	Adbi, 2007
Šidák	Less common method for controlling the familywise error rate. Uniformly more powerful than Bonferroni, but requires the assumption that individual tests are independent.	Adbi, 2007
Storey's FDR	Recent method used to control the false discovery rate. More powerful than the Benjamini-Hochberg method. Requires estimating certain parameters and is more computationally expensive than the Benjamini-Hochberg approach.	Storey and Tibshirani, 2003 Storey et al., 2004

❖ **Graphical exploration of results:**
Similar to the statistical analysis for two groups, here, bar plot, scatter plot and extended error bar plot can be generated to explore the result of statistical analysis of metagenomic profiles belonging to two different samples.
Other plots:
- **Profile bar plot**: It is a grouped bar plot indicating the proportion of sequences assigned to each feature in the two selected samples. It is recommended for investigating higher hierarchical levels of a profile where the number of features is relatively small. Confidence intervals for each proportion are calculated using the Wilson score method.
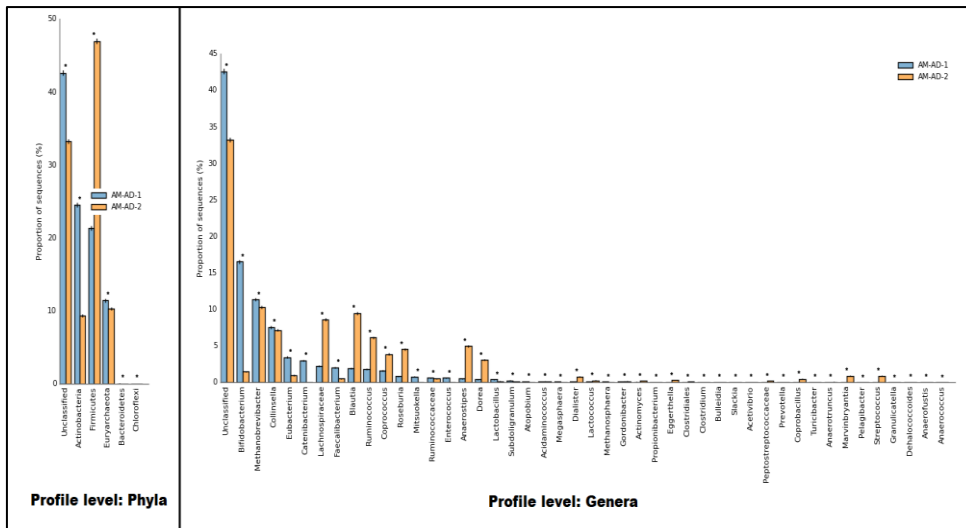
- **Sequence histogram**: It gives a general overview of the number of sequences assigned to each feature in both the samples.
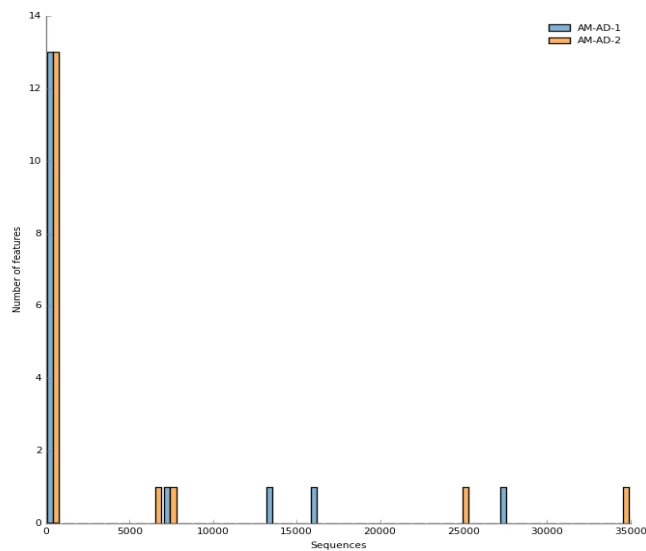
- **Multiple comparison plots**: It can be used to analyze the results of applying a multiple test correction technique, *e.g.*, Benjamini-Hochberg FDR.

Multiple test correction method: Benjamini-Hochberg FDR
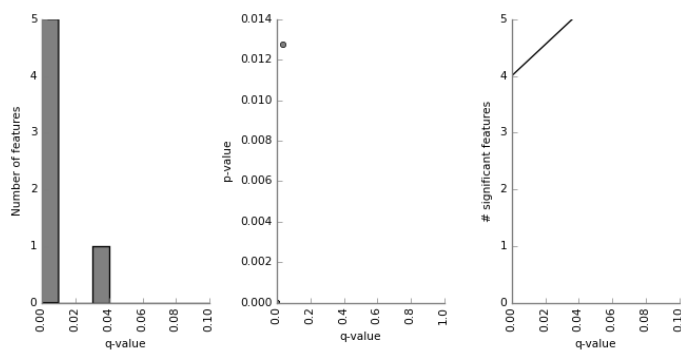
- **p-value histogram**: It shows the distribution of p-values and corrected p-values (*i.e.*, number of features corresponding to a particular p-value) in a metagenomic profile.
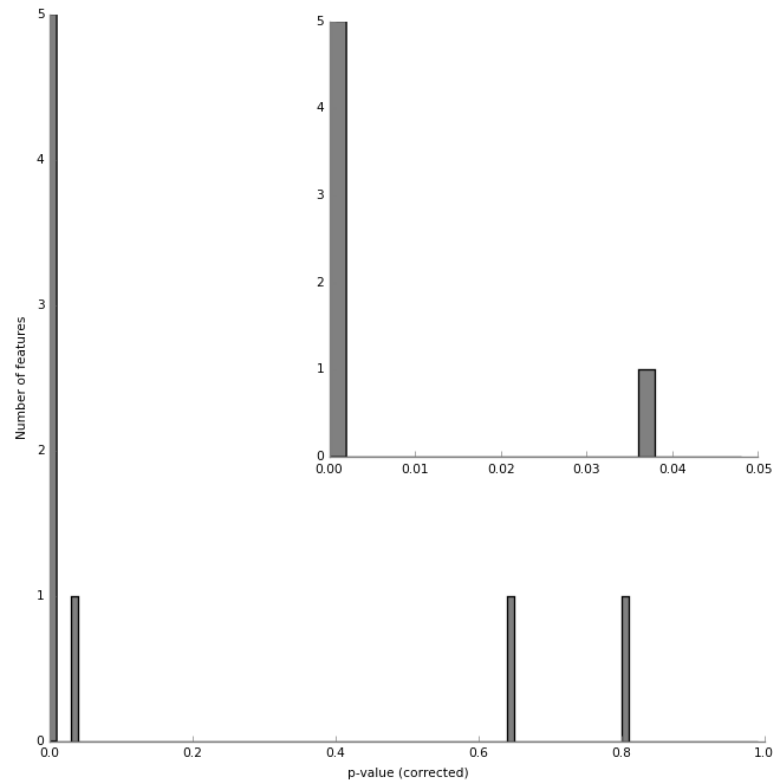
✓ **References:**

Abdi, H. (2007). *Encyclopedia of Measurement and Statistics*. Sage, Thousand Oaks, CA.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, **57**, 289 – 300.

Parks, D. H. and Beiko, R. G. (2010). Identifying biologically relevant differences between metagenomic communities. *Bioinformatics*, **26**, 715 – 721.

Storey, J. D. and Tibshirani, R. (2003). Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences of the United States of America*, **100**, 9440 – 9445.

Protein Structure Prediction

Sudhir Srivastava and Sneha Murmu

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

1. Introduction

Protein structure prediction is the prediction of the three-dimensional structure of a protein from its amino acid sequence, *i.e.*, the prediction of its secondary, tertiary, and quaternary structure from its primary structure. It is one of the most important goals in structural biology. It is important to know the 3D structure of a protein for better understanding of its function. There are a number of approaches for protein structure prediction and the performance of current methods is assessed in the biennial CASP experiment (Critical Assessment of Techniques for Protein Structure Prediction).

2. Proteins

Proteins are regular, linear polymers composed of amino acids. Proteins are fundamental to life. Like other biological macromolecules such as polysaccharides and nucleic acids, proteins participate in virtually every process within cells. They have a remarkable variety of functions. For instance, they act as enzymes, catalyzing most biochemical reactions. Proteins also have structural or mechanical functions, such as actin and myosin in muscle and the proteins in the cytoskeleton, which form a system of scaffolding that maintains cell shape. Other proteins are important in cell signaling, immune responses, cell adhesion, and the cell cycle. Proteins are also necessary in animals' diets, since animals cannot synthesize all the amino acids they require. Through the process of digestion, animals break down ingested protein into free amino acids that are then used in metabolism. In short, proteins are of central importance to almost every biological process.

2.1. Amino Acids: An amino acid, also called residue in proteins, is composed of a carboxyl group, an amino group and a side chain (Fig.1).
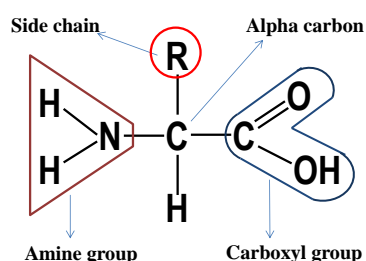
Fig. 1. General structure of an amino acid

There are 20 amino acids which form the basis for every natural protein and these differ only in the side chain atoms as shown in Fig. 2. The amino acids are linked in proteins by the

formation of peptide bonds between amino and carboxyl groups of two adjacent residues as given in Fig. 3.
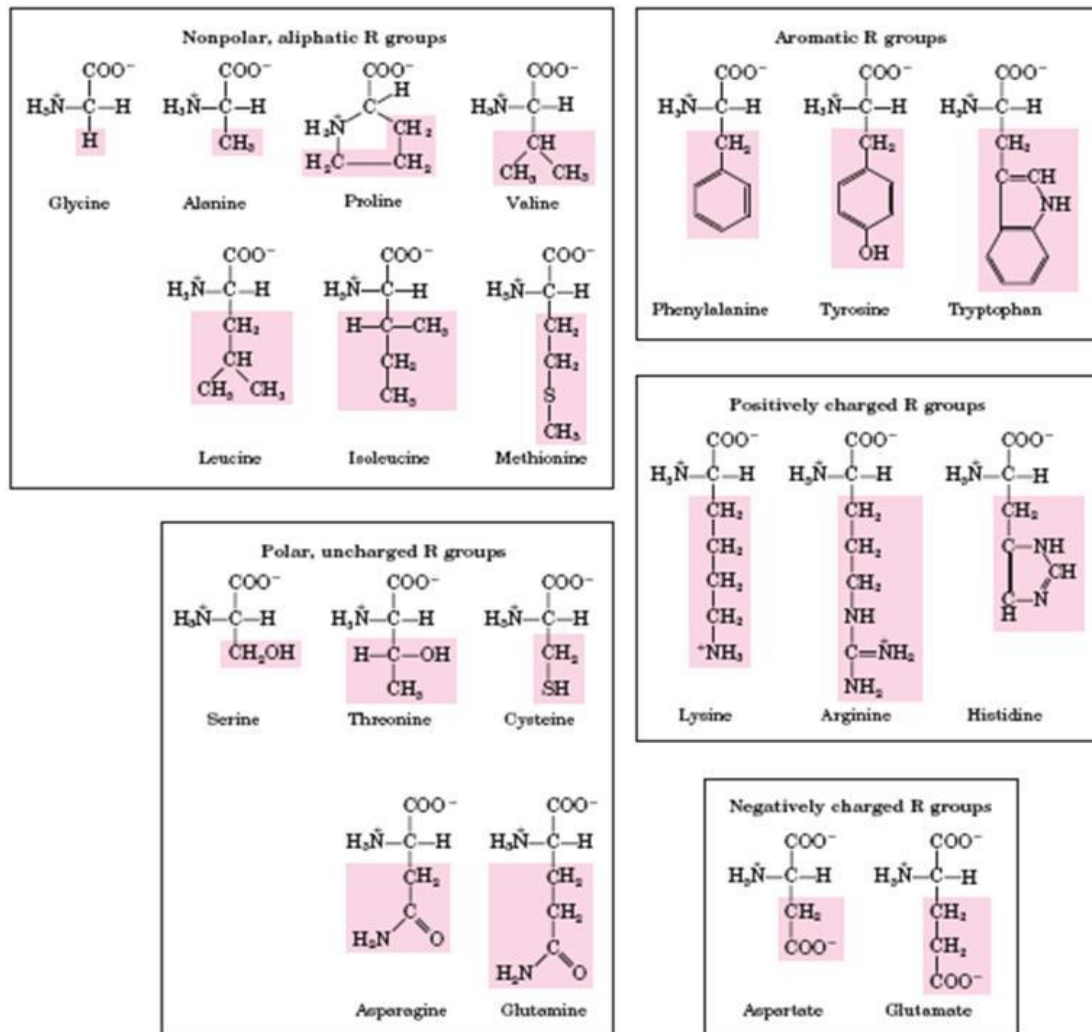
Fig. 2. The 20 common amino acids of proteins
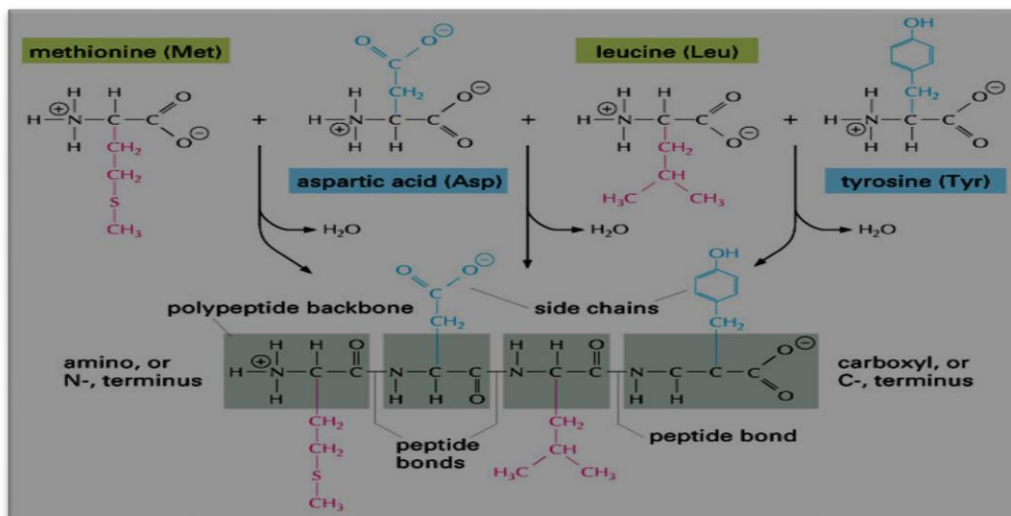
Fig. 3. Peptide bond formation

2.2. Levels of Protein Structure

Protein structures are defined in the following manner:

(a) Primary structure: It is the amino acid sequence forming the polypeptide chain.

(b) Secondary structure: It is defined by the local structural patterns, α-helix, β-strand (groups of β-strands assemble into β-sheet), turn and interconnecting loop.

(c) Tertiary structure: The 3D conformation of the protein is the tertiary structure. This is the functional form of the protein.

(d) Quaternary structure: The aggregation and complex formation of protein having two or more polypeptide subunits defines the quaternary structure.

(a) Primary structure   (b) Secondary structure   (c) Tertiary structure   (d) Quaternary structure
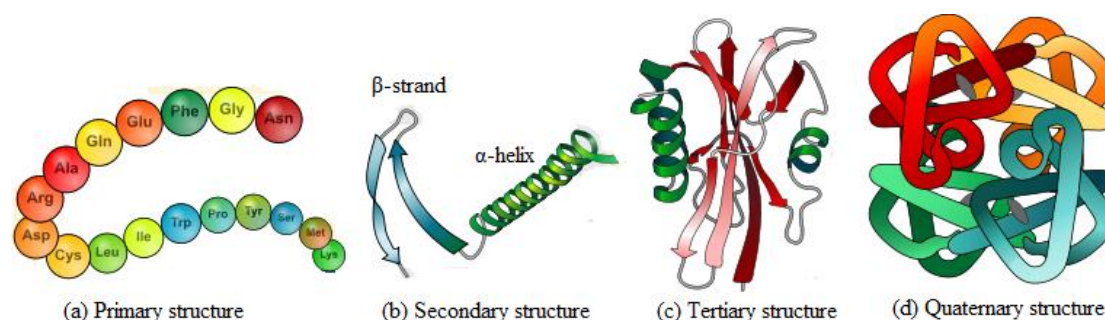
Fig. 4. Levels of structures in proteins

3. Protein Structure Databases and Visualization Tools

The major protein structure bases are as follows:

- Protein Data bank (PDB, http://www.rcsb.org/pdb/): It is a very large global repository for the processing and distribution of 3D macromolecular structure data primarily determined using experimental tools and techniques such as X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, cryoelectron microscopy, *etc*. It provides access to the structural data as well as methods to visualize the structures.

- NCBI Molecular Modeling Database (MMDB, http://www.ncbi.nlm.nih.gov/Structure/MMDB/mmdb.shtml): It includes experimentally resolved structures of proteins, RNA, and DNA, derived from the PDB, with value-added features such as explicit chemical graphs, computationally identified 3D domains that are used to identify similar 3D structures, *etc*.

Some important protein structure visualization databases and tools are as follows:

- Cn3-D (http://www.ncbi.nlm.nih.gov/Structure/CN3D/cn3d.shtml)
- Rasmol, Chemscape Chime & Protein Explorer (http://www.umass.edu/microbio/rasmol)
- SWISS PDBViewer (http://us.expasy.org/spdbv/)

4. Primary Structure Analysis

There are various tools for predicting the physical properties using the sequence information. Some of the important tools are as follows:

(a) Compute pI/Mw (http://ca.expasy.org/tools/pi_tool.html): This tool calculates the isoelectric point and molecular weight of an input sequence. The sequence can be input in FASTA format.

(b) Peptide Mass (http://www.expasy.org/tools/peptide-mass.html): It cleaves one or more protein sequence from the SWISS-PROT and/ or TrEMBL databases or a user entered protein sequence with a chosen enzyme and computes the mass of the generated peptides. It also returns theoretical isoelectric point and mass values for the proteins of interest.

(c) ProtParam (http://ca.expasy.org/tools/protparam.html): It allows the computation of various physical and chemical parameters for a given protein stored in SWISSPROT or TrEMBL or for a user entered sequence.

(d) SAPS (http://www.isrec.isb-sib.ch/software/SAPS_form.html): Statistical Analysis of Protein Sequences (SAPS) is a tool to evaluate a wide variety of protein sequence properties by using statistical criteria. It is a program that provides extensive statistical information in any given sequence.

5. Secondary Structure Prediction and Analysis

A number of methods for predicting the secondary structure are available and some of the important methods are discussed below:

(a) Chou-Fasman Method: The Chou-Fasman algorithm is one of the most widely used methods for the prediction of protein secondary structure. This method depends on assigning a set of prediction values to a residue and then applying an algorithm to the conformational parameters and positional frequencies. The conformational parameters for each amino acid are calculated by considering the relative frequency of a given amino acid within a protein, its occurrence in a given type of secondary structure, and the fraction of residues occurring in that type of structure. These parameters are measures of a given amino acid's preference to be found in helix, sheet or coil. This method is nearly 50-60 % accurate in predicting the secondary structures.

(b) GOR Method: The GOR method was developed by three scientists, Garnier, Osguthorpe and Robson. This method assumes that amino acids up to 8 residues on each side influence the secondary structure of the central residue. The accuracy of GOR when checked against a set of 267 proteins of known structure is 64%. This implies that 64% of the amino acids are correctly predicted as being helix, sheet or coil. The algorithm uses a sliding window of 17 amino acids. All possible pairs of amino acids in this window are checked for their

information content as to predicting the structure of the central amino acid by comparing them to a set of 266 other proteins of known structure. The method works better for helix than for sheet because sheet is dependent on longer range interactions between non-adjacent sequence fragments. GOR under predicts the number of β strands and usually one can predict 36.5% of the β strands correctly.

(c) Nearest Neighbour Method: This method is based on the hypothesis that short homologous sequences of amino acids have the same secondary structure tendencies. A list of short sequence fragments is made by sliding a window of length n along a set of approximately 100-400 training sequences of known structure but minimal sequence similarity. For example, in SIMPA96 (http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page= npsa_simpa96.html), one of the implementations of this method, n is 13 and there are 300 proteins. The secondary structure of the central amino acid in each training window is recorded and a sliding window of the same size is then selected from the query sequence.

NNSSP (http://bioweb.pasteur.fr/seqanal/interfaces/nnssp-simple.html) is another program that predicts the secondary structure combining the nearest neighbour and multiple sequence alignment approaches.

(d) Hidden Markov Models (HMMs): HMMs may be used to predict the secondary structure of a protein of a given structural class as used in the structural classification databases. Each HMM is trained with the sequence of the proteins in that structural class. The models are use with a query sequence to predict both the class and secondary structure of the protein. Pfam (http://pfam.sanger.ac.uk/Software/Pfam/search.shtml) uses the HMM approach.

(e) Neural Networks: Most of the effective structure prediction models extract patterns from databases of known protein structures. A neural network comprises a particular tool for pattern recognition and classification. The simplest layered feed-forward neural network consists of a layer of input units and a layer of output units. Signals are transmitted from input to output layer (feed forward) via the connections.

HNN (http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=npsa_nn.html) is Hierarchical Neural Network based program that gives secondary structure prediction. nnPredict (www.cmpharm.ucsf.edu/%7Enomi/nnpredict.html) predicts the secondary structure type for each residue in an amino acid sequence. The basis of the prediction is a two-layer, feed-forward neural network.

(f) Multiple Alignments Based Self-Optimisation Method: SOPMA (http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=npsa_sopma.html) is a secondary structure prediction program that uses multiple alignments. SOPMA correctly predicts 69.5% of amino acids for a three-state description of a secondary structure (alpha-helix, beta-sheet

and coil) in a whole database containing 126 chains of non-homologous (less than 25% identity) proteins.

Some tools for secondary structure prediction of proteins: JPred, NetSurf, GOR, MetaPP, PREDATOR, Predict Protein, PSIPRED, SymPred, PSSPred

6. Three-Dimensional Structure Prediction

With recent advances in large-scale sequencing technologies, an exponential growth in protein sequence information has been seen. Protein structures are primarily determined by using X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy, but these methods are time consuming, expensive, and not feasible for all proteins. A number of different approaches for protein structure prediction have been developed. These range in scope from adapting a solved structure to match the sequence of an unknown one, to the attempt to fold a protein from first principles. Success varies and different methods should be used for different proteins.

6.1. Methods of Protein Structure Prediction

Based on the degree of similarity between the unknown structure (target) and structures from the database (templates), one can broadly distinguish three approaches:

(a) Comparative (or homology) modeling: Comparative modeling builds the target structure based upon a homologous structure. Stretches of the polypeptide chain presumably differing between the two structures have to be edited, but otherwise the structure can be more or less copied. For this approach to work, there needs to be a significant sequence similarity detected between the target and one or more sequences in the database. This approach relies heavily on good sequence comparison and alignment methods. Under these circumstances, it is the method of choice, as it will produce fewer errors as compared to other methods.

Steps involved in homology modeling are as follows:

- Identify a set of template proteins (with known structures) related to the target protein. This is based on sequence homology (BLAST, FASTA) with sequence identity of 30% or more.
- Align the target sequence with the template proteins. This is based on multiple alignment. Identify conserved regions.
- Build a model of the protein backbone, taking the backbone of the template structures (conserved regions) as a model.
- Model the loops. In regions with gaps, use a loop-modeling procedure to substitute segments of appropriate length.
- Add side chains to the model backbone.
- Evaluate and optimize entire structure.

Homology modeling tools: SWISS-MODEL, 3DJIGSAW, EasyModeller, Easypred3D, FoldX, GeneSilico, Geno3D, HHPred, Modeller, PRIME, ROBETTA, *etc.*

(b) Fold recognition: Fold recognition exploits knowledge from the fact that the number of naturally occurring protein folds is limited. It is therefore likely that a sequence with no significant sequence similarity may still be similar to structures in the database. Two different sub-categories exist in fold recognition. Threading is closest to *ab initio* methods. The target sequence is placed on the 3D coordinates of protein structures in a fold library ("threaded"). The folds with the substituted amino acid sequence are then evaluated with an energy function. Profile or mapping methods instead try to extend the capabilities of sequence comparison algorithms to detect weak sequence homologies. In addition, and depending on the method used, information like predicted secondary structure and/ or predicted surface accessibility is incorporated to improve the results. Once an alignment with a known structure is found, the same methods as in comparative modeling may be used to produce a model.

Threading/fold recognition tools: 3D PSSM, I-TASSER, MUSTER, RAPTOR, *etc.*

(c) *Ab initio* modeling: Ab initio methods attempt to construct a model structure based on the physico-chemical properties of the amino acid chain. No knowledge on known structures is required. Calculations are based on complex energy functions, which encapsulate the information about atomic forces. An optimization method is used to guide the process of selecting promising structures throughout construction. In addition, secondary structure predictions can be incorporated to produce better structures. Despite recent improvements, the ab initio methods still produce more erroneous models for all but the most difficult structures.

***ab initio* modeling tools:** EVfold, QUARK, I-TASSER, ROBETTA, PEP-FOLD, Bhageerath, *etc.*

Role of Artificial Intelligence in Protein Structure Prediction

Over the past few years in the field of Artificial Intelligence (AI), there have been a surge of learnable programs that excel at challenging problems that are too complicated for humans to write down an explicit solution. AlphaFold, developed by DeepMind, based on deep learning is one such game-changing program in the field of structural biology (Jumper et al., 2021). It employs the power of AI to predict the 3D structure of proteins directly from their amino acid sequences, achieving unprecedented accuracy. This program learns from a vast dataset stored in the Protein Data Bank (PDB), which contains information about known protein structures. The AlphaFold process begins with a complex step called feature extraction. Here, the program looks for similar protein sequences to the one it aims to predict. These sequences are found in genetic databases, and they are used to create what's known as a Multiple Sequence Alignment (MSA). This MSA provides essential information about the protein's evolutionary

relationships, helping AlphaFold understand its structure. The MSA, along with additional features, is then transformed into a two-dimensional (2D) array and fed into a deep residual convolutional network, known as ResNet. The primary goal of ResNet is to generate a "distogram." A distogram is a 2D matrix that represents the distances between all pairs of amino acids in the 3D structure of the folded protein. It essentially serves as a map of distances within the protein. One advantage of using a distogram is its invariance to rotations and translations of the protein. In simple terms, it doesn't matter how the protein is oriented; the distances between amino acids remain the same, simplifying the learning process.

AlphaFold goes further by not merely predicting distances but by providing a distribution over distances. This means that for each pixel in the 2D matrix, AlphaFold predicts a distribution across 64 possible distance bins, offering valuable insights into the model's confidence in these distances. The final step is converting this predicted distogram distribution into the actual 3D shape of the protein. This is accomplished using a gradient descent approach. A physical model of the protein's chain is constructed, which is parameterized by the phi and psi angles between consecutive amino acid residues. This backbone geometry is then refined iteratively using a scoring function that combines the neural network's predictions with a set of physically simulated forces. This process determines the most probable conformation of the protein.

6.2. An Example of Protein Structure Modeling

Steps in Protein structure modeling are as follows (Fig.5):

- Open NCBI (http://www.ncbi.nlm.nih.gov).
- Select protein database and type Accession number of the target protein and then search. For example, type ADB44002 in the search box.
- Download the protein sequence in FASTA format and save it.
- Perform multiple sequence alignments using BLAST server at NCBI against PDB database (Select blastp).
- Select suitable homologous template for the given protein sequence with highest bit score and lowest e-value having similarity greater than 30%.
  In the given example, the Chain A of the "Crystal Structure of The Poplar Glutathione Peroxidase 5 In the Reduced Form" (2P5Q_A) has been selected as a template.
- Download the PDB file of the selected template from Protein Data Bank (http://www.rcsb.org/pdb/).
- Open Accelrys Discovery Studio 3.5; go to File and open Protein Sequence Window.
- Paste sequence of the target protein in the sequence window.
  >gi|283827717|gb|ADB44002.1| peroxidase [Mangifera indica]

DFTVKDAKGNDMDLSIYKGKVLVIVNVASRCGLTNSNYTELSQLYQKYKDQGLE
ILAFPCNQFGAQEPGSNEQIVEFACTRFKAEYPIFDKVDVNGDKAAPIYKFLKSSK
GGLFGDSIKWN

- Then align sequence to the template using sequence-template alignment tool.
- Upload the structure of template and run alignment.
- After aligning click on homology modelling and Generate 3D structure models.
- Verify the quality of a structure model

(i)

(ii)

(iii)

(iv)

(v)
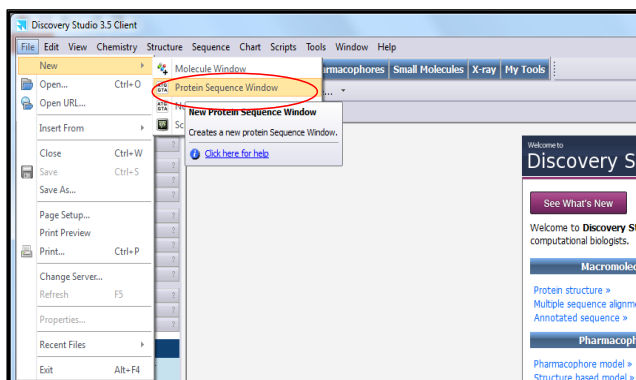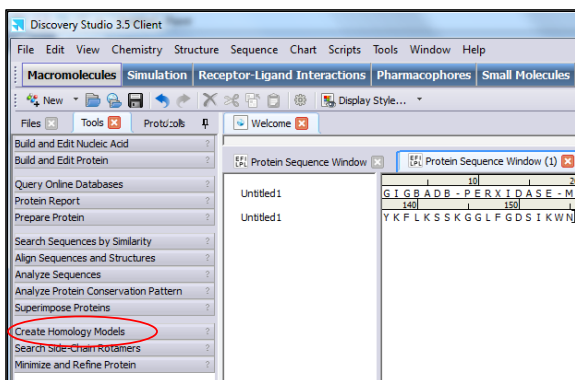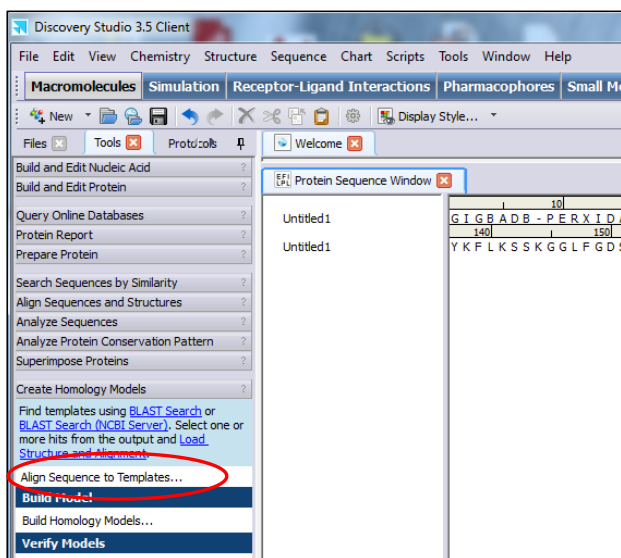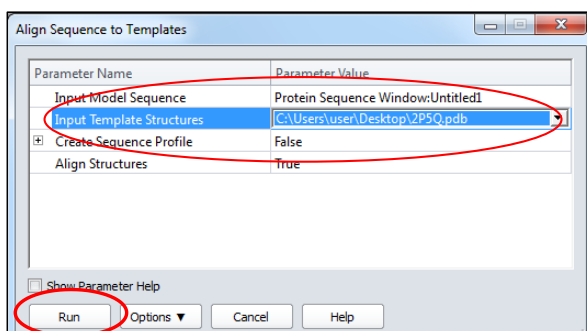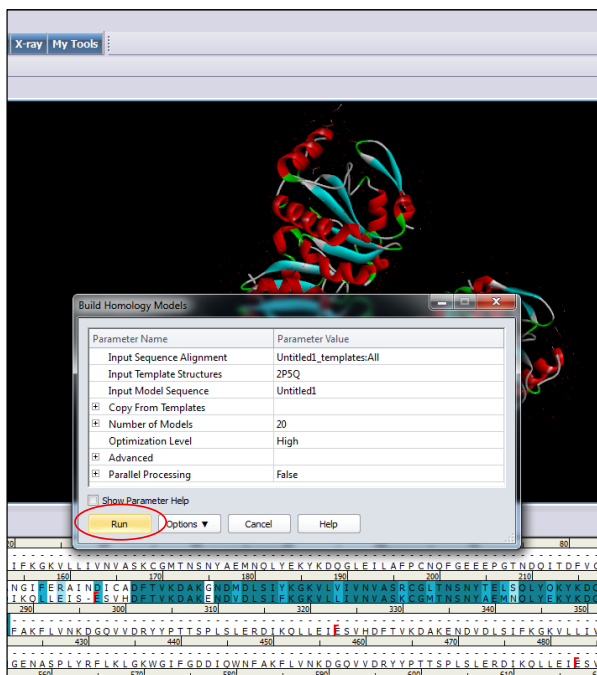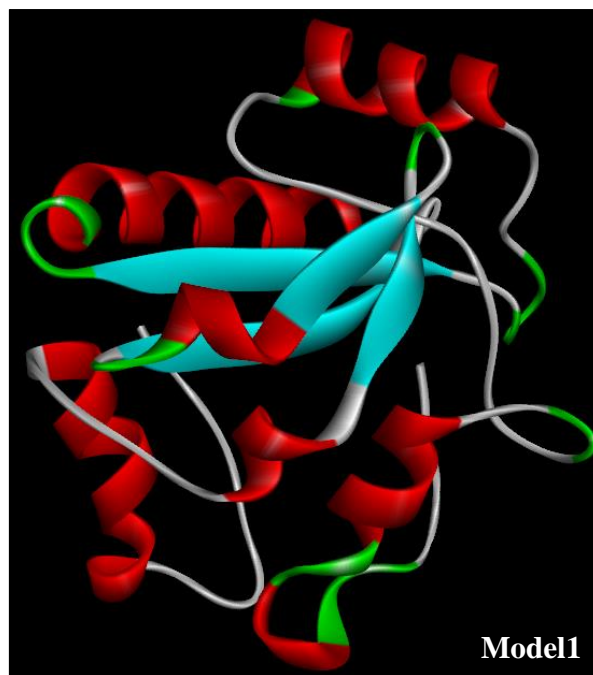
Model1

(vi)
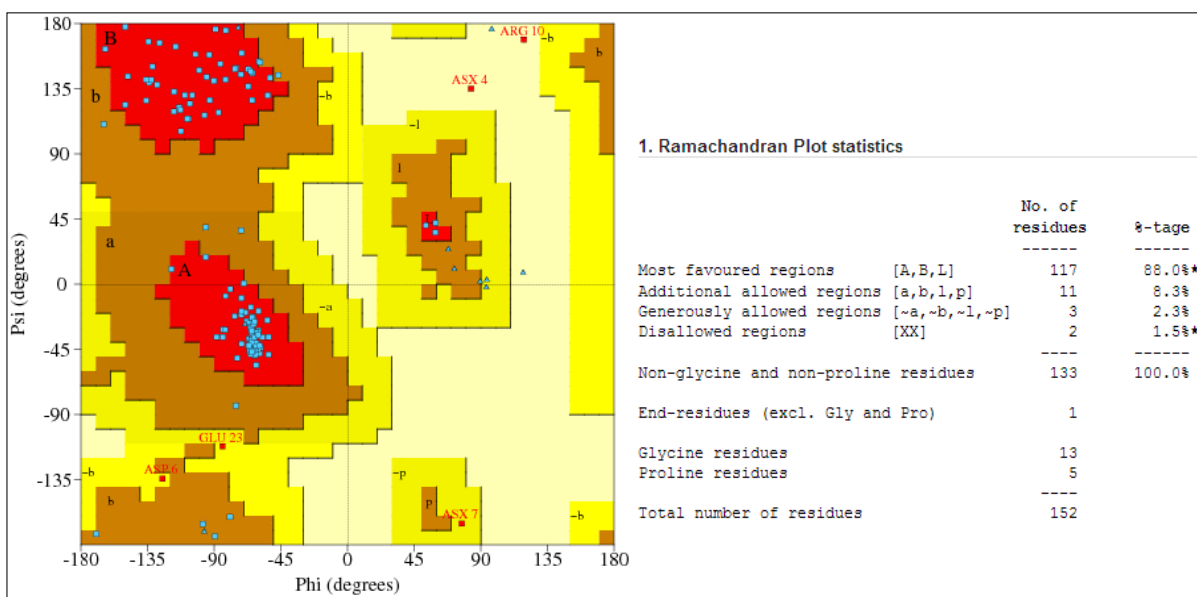
(vii)

Fig. 5. (i) to (vi) Different steps of homology modeling and (vii) Ramachandran plot

6.3. Model Validation tools

To obtain an accurate model, the quality assessment of the model is one of the important steps in protein structure prediction. The accuracy of a model determines its usefulness for specific applications. The quality assessments of predicted protein structure can be done by using QMEAN server and Structure Analysis and Verification Server.

QMEAN server is available at http://swissmodel.expasy.org/qmean/. The QMEAN (Qualitative Model Energy ANalysis) score is a composite score consisting of a linear combination of six structural descriptors: C_beta interaction energy, all atom pairwise energy, solvation energy, torsion angle energy, secondary structure agreement and solvent accessibility agreement.

Structure Analysis and Verification Server is available at http://nihserver.mbi.ucla.edu/SAVES/. There are five tools available at this server. PROCHECK is used to check the stereochemical quality of a protein structure by analyzing residue-by-residue geometry and overall structure geometry. WHAT_CHECK is used for extensive checking of many stereochemical parameters of the residues in the model. ERRAT checks the overall quality factor of non-bonded atomic interactions. VERIFY_3D determines the compatibility of an atomic model and PROVE calculates the volumes of atoms in macromolecules.

References

Accelyris Discovery Studio 3.5.

Lesk, A. M. (2005). Introduction to Bioinformatics. 2nd Editon, Oxford University Press.

Mount, D.W. (2004). Bioinformatics: Sequence and Genome Analysis. 2nd edition, Cold

   Spring Harbor Laboratory Press, New York, USA.

Nelson, D. L., Lehninger, A. L. and Cox, M. M. (2008). Lehninger Principles of Biochemistry. 5th edition, W.H. Freeman, New York, USA.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), 583-589.

Molecular Dynamics and Simulation

Sneha Murmu, U B Angadi and Sudhir Srivastava

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Molecular dynamics (MD) simulation is a computational technique used to study the behavior of atoms and molecules over time. It is based on the laws of classical mechanics, which describe how particles move and interact with each other under the influence of forces. In an MD simulation, the positions, velocities, and accelerations of the atoms or molecules are calculated at each time step, and the system is evolved forward in time.

The basic principle of MD simulation is based on the integration of Newton's second law of motion, which states that the force acting on an object is proportional to its mass times its acceleration. In MD, the forces acting on each atom or particle are calculated using a force field, which describes the interactions between the atoms or particles in the system. The force field is typically based on empirical or theoretical models, which consider the van der Waals forces, electrostatic interactions, and bonded interactions such as covalent bonds, hydrogen bonds, and torsional angles. The motion of the atoms or particles is then simulated using numerical integration of Newton's equations of motion. This process involves calculating the position and velocity of each atom or particle at each time step, based on the forces acting on it, and then updating the forces based on the new positions and velocities.

MD simulations can provide detailed information on the structure, dynamics, and thermodynamics of a system. They can be used to study the behavior of molecules, proteins, and materials in different environments, such as solvents, membranes, or under mechanical stress. MD simulations can also be used to predict the behavior of systems under different conditions or to explore the effects of mutations or drug interactions on protein structures.

Force Fields

Force fields are critical components of molecular dynamics (MD) simulations. They provide a mathematical description of the interatomic or intermolecular forces that govern the behavior of the simulated system. Force fields specify the potential energy and its corresponding force as a function of the coordinates of the atoms or molecules, which is used to calculate the motion

of the system over time. They are mathematical models that include parameters for the bond stretching, bond bending, torsion, and non-bonded interactions between atoms (Figure 1). The accuracy of the force field determines the accuracy of the MD simulations.

There are two primary types of force fields used in molecular dynamics simulations: classical and quantum mechanical. Classical force fields are most commonly used in biomolecular simulations and are based on a set of mathematical functions and empirical parameters to describe the interactions between atoms. These force fields are computationally efficient and can simulate systems up to millions of atoms. Quantum mechanical force fields, on the other hand, consider the electronic structure of atoms and molecules and are computationally more intensive but can provide higher accuracy in describing the system.

A functional form for a force field (also called Potential Energy Function) that can be used to model single molecule or assemblies of atoms and / or molecules is as shown below:

$$\psi(\mathbf{r}^N) = \sum_{bonds} \frac{k_i}{2}(l_i - l_{i,0})^2 + \sum_{angles} \frac{k_i}{2}(\theta_i - \theta_{i,0})^2 + \sum_{torsions} \frac{V_n}{2}(1 + cos(n\omega - \gamma)) +$$

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6} + \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}\right]\right) \qquad \text{... Equation 1}$$

$\psi(r^N)$ denotes the potential energy, which is a function of the positions (r) of N particles (usually atoms).

The first term in the Equation 1 models the interaction between pairs of bonded atoms, here modelled by a harmonic potential that gives the increase in energy as the bond length l_i deviates from the reference value $l_{i,0}$. The second component is a summation over all valence angles in the molecule, modelled using a harmonic potential. A valence bond angle is the angle formed between three atoms A-B-C in which A and C are both bonded to B. The third component is a torsional potential that models how the energy changes as a bond rotates. The fourth component is the non-bonded term. It is calculated between all pairs of atoms (i and j) that are in different molecules or that are the same molecule but separated by at least three bonds (1, n relationship where n ≥ 4). In a simple force field, the non-bonded term is modelled using a Coulomb potential term for electrostatic interactions and a Lennard-Jones potential for van der Waals interactions.

The first three are the components of covalent (or bonded) contribution and the last one is the component of non-covalent (or non-bonded) contribution.

A simple form of the above equation:

A potential function or force field calculates the molecular system's potential energy (E) in a given conformation as a sum of individual energy terms,

$$E = E_{Covalent} + E_{Non-covalent} \qquad \qquad \text{... Equation 2}$$

where, $E_{Covalent} = E_{bond} + E_{angle} + E_{dihedral}$

$E_{Non-covalent} = E_{electrostatic} + E_{van\ der\ Waals}$
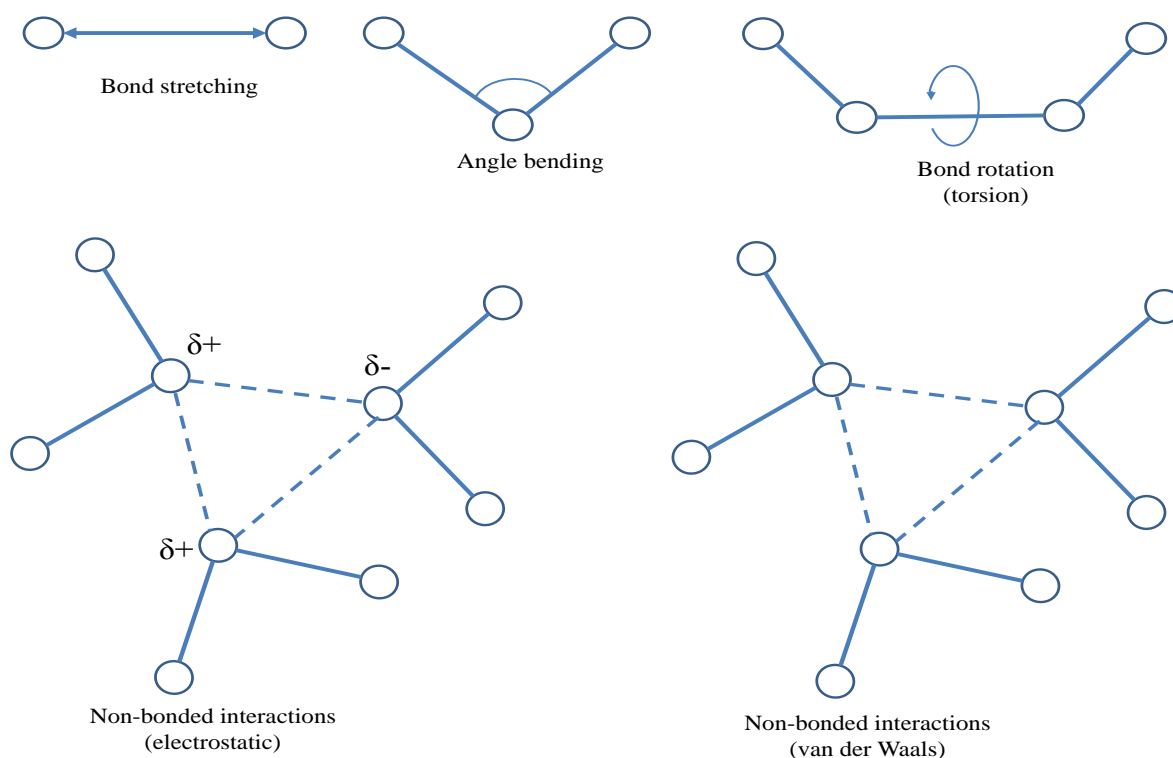
Figure 1: Schematic representation of bonded (upper row) and non-bonded (lower row) components contributing to a molecular mechanics force field.

There are several different force fields that have been developed over the years, each with its own strengths and limitations. Here are some examples:

CHARMM (Brooks et al., 2009): The Chemistry at Harvard Macromolecular Mechanics (CHARMM) force field is widely used for biomolecular simulations. It includes parameters for all of the major types of interactions, including covalent bonds, angles, dihedrals, van der Waals forces, and electrostatics. It is known for its accuracy in reproducing protein structures and dynamics.

AMBER (Case et al., 2010): The Assisted Model Building with Energy Refinement (AMBER) force field is also widely used in biomolecular simulations. It includes parameters for bond stretching, bond bending, torsion, and non-bonded interactions, and is known for its accuracy in reproducing experimental structures and dynamics.

OPLS (Damm et al., 1997): The Optimized Potentials for Liquid Simulations (OPLS) force field was originally developed for liquid simulations, but has also been used in biomolecular simulations. It includes parameters for bond stretching, bond bending, torsion, and non-bonded interactions, and is known for its accuracy in reproducing thermodynamic properties of liquids.

GROMOS (Scott et al., 1999): The Groningen Molecular Simulation (GROMOS) force field is widely used in simulations of small molecules and peptides. It includes parameters for bond stretching, bond bending, torsion, and non-bonded interactions, and is known for its accuracy in reproducing thermodynamic properties of small molecules.

Conclusion

In summary, the principle of molecular dynamics simulation is based on the integration of classical mechanics, which involves calculating the positions, velocities, and forces of all atoms or particles in a system as a function of time. MD simulations can provide detailed information on the structure, dynamics, and thermodynamics of a system and can be used to study a wide range of molecular and material systems.

************************************Practical************************************

The purpose of this hands-on is to provide an introduction to the fundamental commands needed to set up, run, and analyze MD simulations using a suitable simulation tool. GROMACS which is one of the most popular Molecular Dynamics (MD) simulation software, will be used for the practical session. Before starting with the steps of typical MD simulation, let us have a quick look on how to install GROMACS in linux (here, Ubuntu).

Installation

To install GROMACS, we need the following software installed on our system:

i.   C & C++ Compiler which comes built-in with Ubuntu.

ii. CMake – A linux software to make binaries

iii. BuildEssential – It is a reference for all the packages needed to compile a package.

iv. FFTW Library: a library used by Gromacs to compute discrete Fourier transform

v.   DeRegressionTest Package

Following are commands to install above mentioned pre-requisites:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install cmake

sudo apt-get install build-essential

wget http://gerrit.gromacs.org/download/regressiontests-5.1.1.tar.gz

tar xvzf regressiontests-5.1.1.tar.gz

sudo apt-get install libfftw3-dev

wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-5.1.1.tar.gz

tar xvzf gromacs-5.1.1.tar.gz

cd gromacs-5.1.1/

mkdir build

cd build

sudo cmake .. -DGMX_BUILD_OWN_FFTW=OFF -DREGRESSIONTEST_DOWNLOAD=OFF -DCMAKE_C_COMPILER=gcc -DREGRESSIONTEST_DOWNLOAD=ON

make

make check

sudo make install

source /usr/local/gromacs/bin/GMXRC

If the execution of above commands was successful, the installation is complete. You may check the version of your Gromacs with a command to make sure the installation finished as expected.

gmx pdb2gmx --versionource /usr/local/gromacs/bin/GMXRC

MD Simulation protocol

Following steps are involved in simulating a protein structure.

- Create initial state
    i.   Generate topology of protein
    ii. Add box and solvation to the system
    iii. Add ions to the solved system

- Introduction to the interaction potentials
    iv. Energy minimization

- Predict how the particles move
    v.   Equilibration of system
    vi. MD Production run

Now, we will see how to perform each step in more details. For the purpose of demonstrating simulation of protein, a small protein structure of ubiquitin (PDB code 1UBQ) was downloaded from RCSB PDB.

1. Generate topology

The obtained protein structure must be checked for the following things:

- Remove the water molecules if present
- Non-standard residues like heteroatoms must be removed
- Residues with missing atoms must be fixed beforehand

If water molecules are present, we can simply use the grep command to search for "HOH" in the PDB file and then remove them. The following command can be used for removing water molecules:

grep -v HOH 1UBQ.pdb > 1UBQ_clean.pdb

The next step is to use the pdb2gmx module of GROMACS. The pdb2gmx module generates three files:

☐   The topology for the molecule.

☐ A position restraint file.

☐ A post-processed structure file.

The topology (topol.top by default) contains all the information necessary to define the molecule within a simulation. This information includes nonbonded parameters as well as bonded parameters. The following command was used to execute pdb2gmx:

gmx pdb2gmx -f 1UBQ_clean.pdb -o 1UBQ_processed.gro -water spce

The structure is processed by pdb2gmx, and we are prompted to choose a force field. We will use the all-atom OPLS force field, so '15' was typed at the command prompt

The force field will contain the information that will be written to the topology.

2. Solvation

To simulate proteins and other molecules we need to define the box dimensions around the protein and fill in the box with solvent. The box was defined using the following command:

gmx editconf -f 1UBQ_processed.gro -o 1UBQ_newbox.gro -c -d 1.0 -bt cubic

-c : centers the protein in the box

-d 1.0 : places the protein at least 1.0 nm from the box edge

-bt cubic : The box type is defined as a cube

Specifying a solute-box distance of 1.0 nm will mean that there are at least 2.0 nm between any two periodic images of a protein. This distance will be sufficient for just about any cut off scheme commonly used in simulations.

The box is filled with solvent (water) by using the command below:

gmx solvate -cp 1UBQ_newbox.gro -cs spc216.gro -o 1UBQ_solv.gro -p topol.top

-cp : this parameter takes as input the configuration of the protein which is contained in the output file obtained from the previous step

-cs : configuration of the solvent is part of the standard GROMACS installation. We are using spc216.gro, which is a generic equilibrated 3-point solvent model.

3. Adding Ions

Neutralizing a system is a practice carried out for obtaining correct electrostatic values during the simulation. This is done because under periodic boundary and using PME electrostatics - the system has to be neutral. Therefore, we are adding ions to neutralization purpose only. The tool for adding ions within GROMACS is called genion which reads through the topology and replace water molecules with the ions that the user specifies. The input is called a run input file, which has an extension of. tpr. The .tpr file contains all the parameters for all of the atoms in the system.ed by the GROMACS grompp module (GROMACS pre-processor).

Assemble .tpr file with the following command:

gmx grompp -f ions.mdp -c 1UBQ_solv.gro -p topol.top -o ions.tpr

Now we have an atomic-level description of our system in the binary file ions.tpr. We will pass this file to genion:

gmx genion -s ions.tpr -o 1UBQ_solv_ions.gro -p topol.top -pname NA -nname CL -neutral

-s : input file given as structure/state file (.tpr file)

-pname and -nname : define the positive and negative ion names

-neutral : add only the ions necessary to neutralize the net charge on the protein by adding the correct number of negative ions (in this case will add 8 Cl- ions to offset the +8 charge on the protein)

4. Energy minimization (EM)

EM is done to ensure there that the system has no steric clashes or inappropriate geometry.

First, we need to assemble structure, topology, and simulation parameters into a binary input file (.tpr file):

gmx grompp -f minim.mdp -c 1UBQ_solv_ions.gro -p topol.top -o em.tpr

Here, minim.mdp is the file containing information regarding molecular dynamics parameter. It is not inherently present in the GROMACS distribution; hence it needs to be created before the execution of above command. An mdp file contain following parameters,

; minim.mdp - used as input into grompp to generate em.tpr

; Parameters describing what to do, when to stop and what to save

integrator = steep   ; Algorithm (steep = steepest descent minimization)

emtol   = 1000.0   ; Stop minimization when the maximum force < 1000.0 kJ/mol/nm

emstep   = 0.01   ; Minimization step size

nsteps   = 50000   ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions

nstlist   = 1   ; Frequency to update the neighbor list and long range forces

cutoff-scheme = Verlet   ; Buffered neighbor searching

ns_type   = grid   ; Method to determine neighbor list (simple, grid)

coulombtype   = PME   ; Treatment of long range electrostatic interactions

rcoulomb   = 1.0   ; Short-range electrostatic cut-off

rvdw   = 1.0   ; Short-range Van der Waals cut-off

pbc   = xyz   ; Periodic Boundary Conditions in all 3 dimensions

Next, we have to invoke mdrun to carry out the EM:

gmx mdrun -v -deffnm em

The output em.edr file contains all of the energy terms that GROMACS collects during EM. We can analyze any .edr file using the GROMACS energy module:

gmx energy -f em.edr -o potential.xvg

At the prompt, type "10 0" to select Potential (10); zero (0) terminates input. The average of Epot is shown, and a file called "potential.xvg" is written. To plot this data, we need the Xmgrace plotting tool.

5. Equilibration

Since the objective of MD simulation is to study the dynamics of a particular system, we have to suit the *in-silico* environment of our simulation system as close as possible to the real system (e.g. experimental job in wet laboratory). Therefore, in equilibration step we optimize the

temperature to 300K since we assumed that we do the experimental job at room temperature, and pressure value at 1 atm.

Equilibration will be carried out in two steps. First, an NVT (constant Number of atoms, Volume, and Temperature) simulation will be performed in order to bring the system to the target temperature. Second, an NPT (constant Number of atoms, Pressure, and Temperature) simulation will be performed to allow the system to find the correct density.

5. a) Temperature Equilibration

We will call grompp and mdrun just as we did at the EM step and run the following two commands:

gmx grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.tpr

gmx mdrun -deffnm nvt

To analyze the temperature progression, using energy we use the command given below:

gmx energy -f nvt.edr -o temperature.xvg

Type "16 0" at the prompt to select the temperature of the system and exit and the temperature.xvg can be plotted by Xmgrace tool.

5. b) Pressure Equilibration

We had included the -t flag to include the checkpoint file from the NVT equilibration. This file contains all the necessary state variables to continue our simulation. To conserve the velocities produced during NVT, we must include this file. The coordinate file (nvt.gro) is the final output of the NVT simulation.

gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p topol.top -o npt.tpr

gmx mdrun -deffnm npt

To analyze the pressure progression, again by using energy:

gmx energy -f npt.edr -o pressure.xvg

Type "18 0" at the prompt to select the pressure of the system and exit. 'pressure.xvg' file will be created which can be plotted through Xmgrace.

To take a look at density as well using energy, we need to enter "24 0" at the prompt while running the following command:

gmx energy -f npt.edr -o density.xvg

6. Production MD

After running the two equilibration phases, the system is now well equilibrated at desired temperature and pressure. To run the production MD, we will make use of the checkpoint file to grompp and run a 1 ns MD simulation:

gmx grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top -o md_0_1.tpr

To execute mdrun:

gmx mdrun -deffnm md_0_1

Analysis

GROMACS comes equipped with many analysis tools, a complete list of which can be found in the manual. Here you will be exposed to a few useful analysis tools: 'rms', 'rmsf', and 'gyrate. But first, it is useful to learn how to process the trajectory file to only keep the components of interest. Use *trjconv*, which is a post-processing tool to strip out coordinates, correct for periodicity, or manually alter the trajectory (time units, frame frequency, etc). trjconv accounts for any periodicity in the system.

gmx trjconv -s md_0_1.tpr -f md_0_1.xtc -o md_0_1_noPBC.xtc -pbc mol –center

Select 1 ("Protein") as the group to be centered and 0 ("System") for output. Downstream analyses will be conducted on this "corrected" trajectory.

For checking the structural stability GROMACS has a built-in utility for RMSD calculations called rms. Root mean square deviation (RMSD) is used for measuring the difference between the backbones of a protein from its initial structural conformation to its final position. The command to plot rmsd graph is as follows:

gmx rms -s md_0_1.tpr -f md_0_1_noPBC.xtc -o rmsd.xvg -tu ns

When prompted choose 4 ("Backbone") for both the least-squares fit and the group for RMSD calculation.

The radius of gyration of a protein is a measure of its compactness. If a protein is stably folded, it will likely maintain a relatively steady value of Rg. If a protein unfolds, its Rg will change over time. The command to plot radius of gyration graph is as follows:

gmx gyrate -s md_0_1.tpr -f md_0_1_noPBC.xtc -o gyrate.xvg

When prompted choose group 1 (Protein) for analysis.

With this, we have now completed molecular dynamics simulation of a protein with GROMACS, and analyzed some of the results.

References

Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., & Berendsen, H. J. (2005). GROMACS: fast, flexible, and free. Journal of computational chemistry, 26(16), 1701-1718.

Case, D. A., Darden, T. A., Cheatham, T. E., Simmerling, C. L., Wang, J., Duke, R. E., ... & Kollman, P. A. (2008). Amber 10 (No. BOOK). University of California.

Brooks, B. R., Brooks III, C. L., Mackerell Jr, A. D., Nilsson, L., Petrella, R. J., Roux, B., ... & Karplus, M. (2009). CHARMM: the biomolecular simulation program. Journal of computational chemistry, 30(10), 1545-1614.

Damm, W., Frontera, A., Tirado–Rives, J., & Jorgensen, W. L. (1997). OPLS all- atom force field for carbohydrates. Journal of computational chemistry, 18(16), 1955-1970.

Scott, W. R., Hünenberger, P. H., Tironi, I. G., Mark, A. E., Billeter, S. R., Fennen, J., ... & Van Gunsteren, W. F. (1999). The GROMOS biomolecular simulation program package. The Journal of Physical Chemistry A, 103(19), 3596-3607.

Introduction to Proteomics Data Analysis

Sudhir Srivastava, Sneha Murmu, Dwijesh Chandra Mishra, U. B. Angadi and K. K. Chaturvedi

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Introduction

Proteins are important large biomolecules or macromolecules performing a wide variety of functions. The word "proteome" is defined as the entire set of proteins translated and/ or modified within a living organism. The word "proteome" was coined by Marc Wilkins in 1994 in a symposium on "2D Electrophoresis: from protein maps to genomes" held in Siena in Italy while he was a Ph.D. student at Macquarie University. An organism's genome is more or less constant whereas proteome is not constant. Proteomes differs from cell to cell and from time to time. That's why proteomics is more complicated when compared to genomics.

Proteomics more generally refers to large-scale liquid chromatography (LC) coupled with mass spectrometry (MS) [LC-MS] based discovery studies designed to address both quantitative and qualitative aspects of the proteome research (Figure 1).
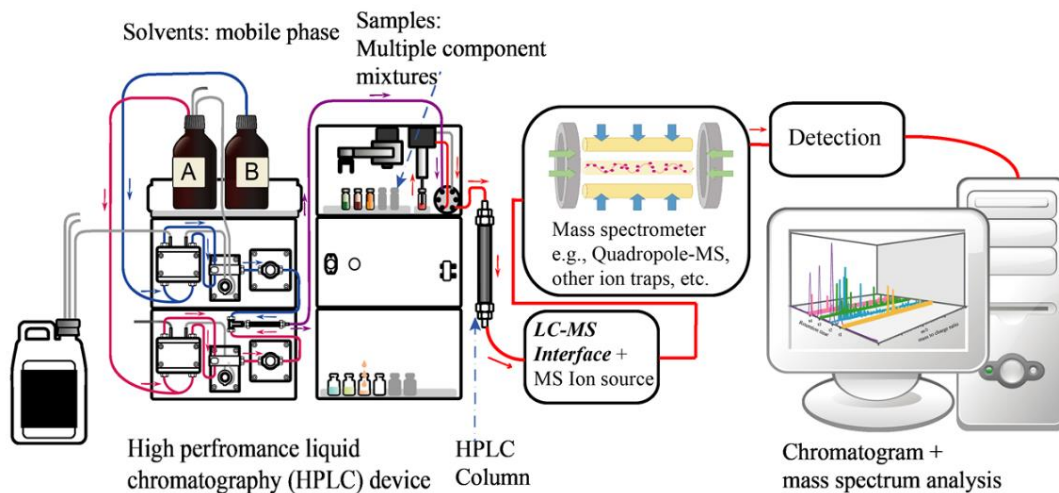
Figure 1. Liquid chromatography coupled with mass spectrometry [LC-MS]

Source: https://upload.wikimedia.org/wikipedia/en/f/f9/Liquid_chromatography_tandem_Mass_spectrometry_diagram.png

Now proteomics has emerged as a powerful tool across various fields such as biomedicine mainly applied to diseases, agriculture, and animal sciences. It is important for studying different aspects of plant functions such as identification of candidate proteins involved in the defensive

response of plants to biotic and abiotic stresses, effect of global climate changes on crop production, etc. In animal sciences, proteomics studies play important role in studying physiology, immunology, reproduction and lactational biology. The practical application of proteomics includes expression proteomics, structural proteomics, biomarker discovery, interaction proteomics, protein networks, etc.

Basics Steps of Proteomics Data Analysis

The proteomic abundance (expression) data are usually generated using high throughput technologies usually involving MS. LC-MS is used in proteomics as a method for identification and quantification of peptides and proteins in complex mixtures. There are two basic proteomics approaches, namely bottom-up and top-down. The most common proteomics approach is the bottom-up in which proteins in a sample are enzymatically digested into peptides and subjected to chromatographic separation, ionization and mass analysis. Conversely, top-down proteomics addresses the study of intact proteins and consequently is most often used to address purified or partially purified proteins. There are various steps involved in quantitative proteomics data analysis, viz., peptide and protein identification, protein abundance quantification, data cleaning, data normalization, handling of missing values by using imputation techniques, data visualization and interpretation, statistical analysis of proteomics data, etc.

Peptide and protein identification

There are two major approaches for determining the sequence of peptides.

(i) Searching against fragmentation spectra databases

(ii) de novo peptide sequencing

Some of the software/ tools for peptide and protein identification are listed below:

Category	Name	Description
Searching against fragmentation spectra databases	Andromeda (part of Mascot)	A peptide search engine based on probabilistic scoring
	Mascot	Probability-based database searching algorithm

	SEQUEST	Identifies collections of tandem mass spectra to peptide sequences that have been generated from protein sequence databases
	X!Tandem/X!!Tandem	Searches tandem mass spectra with peptide sequences in database
de novo peptide sequencing	PEAKS	Performs de novo sequencing for each peptide, confidence scores on individual amino acid assignments with manually assisted mode and automated de novo sequencing on an entire LC run processed data
	SHERENGA	Performs de novo peptide sequencing via tandem mass spectrometry
	PECAN	Library free peptide detection for data-independent acquisition of tandem mass spectrometry data

Quantification of feature abundance

The quantification of features (peptides or proteins) may be either label-free or labelled (metabolic, enzymatic, or chemical) to detect differences in feature abundances among different conditions. In label-free quantification, MS ion intensity (peak area) and spectral counting of features are the major approaches. In this article, we have considered MS ion intensity data obtained from label-free bottom-up proteomics experiments.

Software/Tools for label-based quantitative proteomics:

- MaxQuant
- Proteome Discoverer (Thermo Scientific)
- XPRESS

Software/Tools for label-free quantitative proteomics:

- MaxLFQ - Label free quantification module available in MaxQuant
- emPAI - Exponentially modified protein abundance index
- Mascot Distiller (Matrix Science)

Problem of missing values and heterogeneity in proteomics data

Various approaches exist for proteomics data analysis in which the first step is to summarize the intensities of all features using a quantitative summary followed by logarithmic transformation to approximate it to normal distribution. In spite of availability of various tools/methods, there are various challenges in analyzing proteomics data such as missing value (MV) and data heterogeneity. There are various drawbacks of the methods which can be studied by examining the statistical properties of these methods.

The variations in the biological data or technical approaches to data collection lead to heterogeneity for the samples under study. The data set usually consists of biological replicates only or both biological and technical replicates. Biological variability arises from genetic and environmental factors and it is intrinsic to all organisms. The technical approaches include sample collection and storage, sample preparation, extraction, LC separation and MS detection.

The data set is called balanced when it contains an equal number of subjects/ samples in each group, and the features have no missing observations. However, this is not always the condition. Sometimes the data can be unbalanced having unequal number of subjects, or missing observations, or both. MVs in proteomics data can occur due to biological and/or technical issues. These are of three types of MVs: (i) missing completely at random (MCAR) in which MVs are independent of both unobserved and observed data; (ii) missing at random (MAR) if conditional on the observed data, the MVs are independent of the missing measurements; and (iii) missing not at random (MNAR) when data is neither MCAR nor MAR. The data with missing observations can be analyzed either by excluding the features having missing observations, by using statistical methods that can handle unbalanced data, or by using imputation methods. If the features having missing observations are excluded, then there is loss of information from the experiment. Therefore, the use of methods that can handle MVs, such as imputation methods, are generally preferred. However, the use of imputation methods may lead to wrong interpretation and these methods are questionable in statistical terms.

Statistical analysis of proteomics abundance data

Differential abundance analysis is carried out to detect significant features in two or more conditions such as normal versus different disease conditions. However, data normalization is necessary before performing further analysis. There are various transformation and/ or

normalization methods such as logarithmic transformation, quantile normalization, variance stabilizing normalization, median scaling normalization, etc. In case of missing values, the user has to impute the data using imputation techniques such as singular value decomposition, *k*-nearest neighbor, maximum likelihood estimation, etc. The statistical approaches/ tests such as t-test, moderated t-test, ANOVA, linear mixed model, etc. can be used for detecting significant features. A general workflow of label-free quantitative proteomics data is given below:
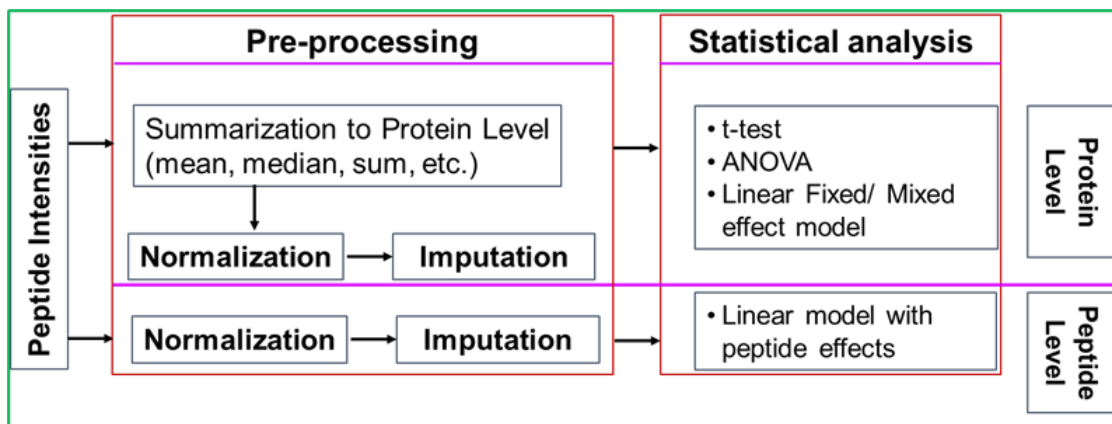
Figure 2. A general workflow of label-free quantitative proteomics data

Various methods of normalizing proteomics expression data are given below:

- Variance stabilizing normalization (VSN)
- Quantile normalization (quantile)
- Median normalization (median)
- EigenMS normalization (EigenMS)
- Local regression normalization (LoessF, LoessCyc)

Various imputation methods can be categorized into the following:

(i) Imputation by a single value:

- Half of global minimum intensity among peptides - the minimal observed intensity value among all peptides
- Half of minimal intensity of individual peptide
- Random tail imputation

(ii) Local-similarity-based imputation methods:

- *K*-nearest neighbors (KNN)

- Local least-squares (LLS) imputation

- Regularized expectation maximization (REM) algorithm

(iii) Global-structure-based imputation methods

- Probabilistic principal component analysis (PPCA)

- Bayesian principal component analysis (BPCA) algorithm

There are various tools and packages available for proteomics abundance data analysis such as DanteR, MSstats, RepExplore, PANDA-view, MSqRob, PANDA, DAPAR, ProStaR etc. Some of the important tools are discussed below:

(i) DanteR: Taverner *et al.* (2012) developed DanteR, a graphical R package that features extensive statistical and diagnostic functions for quantitative proteomics data analysis, including normalization, imputation, hypothesis testing, interactive visualization and peptide-to-protein rollup.

(ii) MSstats: Choi *et al.* (2014) developed an R package "MSstats" for statistical relative quantification of proteins and peptides in MS based proteomics. It (version 2.0) supports label-free and label-based experimental workflows and data-dependent, targeted and data-independent spectral acquisition. It performs differentially abundance/ expression analysis of features (peptides or proteins) based on linear mixed models.

(iii) RepExplore: Glaab and Schneider (2015) developed a web server "RepExplore" to analyse the proteomics and metabolomics data with technical and biological replicates. The analysis is based on previously published statistical methods, which have been applied successfully to biomedical omics.

(iv) PANDA-view: Chang *et al.* (2018) developed an easy-to-use tool "PANDA-view" for both statistical analysis and visualization of quantitative proteomics data and other -omics data. There are various kinds of analysis methods such as normalization, MV imputation, statistical tests, clustering and principal component analysis, an interactive volcano plot.

(v) MSqRob: Goeminne *et al.* (2018) provided a tutorial on analysis of quantitative proteomics data. The tutorial discussed the key statistical concepts to design proteomics experiments and analyse label-free MS based quantitative proteomics data using their free and open-source R package MSqRob.

(vi) PANDA: Chang *et al.* (2019) developed a comprehensive and flexible tool named PANDA for proteomics data quantification. The tool supports both label-free and labeled quantifications and it is compatible with existing peptide identification tools and pipelines with considerable flexibility.

(vii) DAPAR & ProStaR: Wieczorek et al. (2017) developed software tools, DAPAR and ProStaR that can perform the statistical analysis of label-free XIC-based quantitative discovery proteomics experiments. DAPAR is an R package that contains various functions such as filtering, normalization, imputation of missing values, aggregation of peptide intensities, differential abundance analysis of proteins, etc. ProStaR is a user-friendly graphical interface that allows access to the DAPAR functionalities through a web browser.

Conclusion

In this article, we have given the basic introduction of proteomics, various steps of proteomics data analysis, problem of MVs and heterogeneity in proteomics data and different methods for analysis of proteomics data. This article will be useful for the researchers working in the field of proteomics and bioinformatics. Furthermore, the methods for proteomics data analysis can further be used for analyzing the expression data obtained from similar experiments (e.g., microarray and metabolomics data).

References

Anderson NL, Anderson NG (1998). Proteome and proteomics: new technologies, new concepts, and new words. *Electrophoresis*, **19(11)**, 1853-61.

Ceciliani F, Eckersall D, Burchmore R, Lecchi C. (2014). Proteomics in veterinary medicine: applications and trends in disease pathogenesis and diagnostics. *Vet Pathol.*, **51(2)**:351-62. doi: 10.1177/0300985813502819.

Chang C, et al. (2018). PANDA-view: An easy-to-use tool for statistical analysis and visualization of quantitative proteomics data. *Bioinformatics*.

Choi M, et al. (2014). MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments. *Bioinformatics*, **30(17)**. 2524-6.

Glaab E, Schneider R (2015). RepExplore: addressing technical replicate variance in proteomics and metabolomics data analysis. *Bioinformatics*, **31(13)**, 2235-7.

Goeminne LJE, Gevaert K, and Clement L (2018). Experimental design and data-analysis in label-free quantitative LC/MS proteomics: A tutorial with MSqRob. *J Proteomics*, **171**, 23-36.

Karpievitch YV, Dabney AR, and Smith RD (2012). Normalization and missing value imputation for label-free LC-MS analysis. *BMC Bioinformatics*, **13 Suppl 16**, S5.

Rubin DB (1976). Inference and missing data. *Biometrika*, **63(3)**, 581–92.

Taverner T., et al. (2012). DanteR: an extensible R-based tool for quantitative analysis of -omics data. *Bioinformatics*, **28(18)**, 2404–2406. doi:10.1093/bioinformatics/bts449.

Wasinger, VC, Cordwell, SJ, Cerpa-Poljak, A, Yan, JX, Gooley, AA, Wilkins, MR, Duncan, MW, Harris, R, Williams, KL, Humphery-Smith, I (1995). Progress with gene-product mapping of the Mollicutes: *Mycoplasma genitalium*. *Electrophoresis*, **16 (1)**, 1090-1094. doi:10.1002/elps.11501601185

Wieczorek, S., Combes, F., Lazar, C., Giai Gianetto, Q., Gatto, L., Dorffer, A., Hesse, A.-M., Couté, Y., Ferro, M., Bruley, C., & Burger, T. (2017). DAPAR & ProStaR: software to perform statistical analyses in quantitative discovery proteomics. *Bioinformatics* (Oxford, England), **33(1)**, 135-136. https://doi.org/10.1093/bioinformatics/btw580

https://en.wikipedia.org/wiki/Proteomics

https://en.wikipedia.org/wiki/List_of_mass_spectrometry_software

Over-view of Post-Translational Modifications

Monendra Grover

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Posttranslational modifications (PTMs) of proteins greatly expand proteome diversity, increase functionality, and allow for rapid responses, all at relatively low costs for the cell. PTMs play key roles in plants through their impact on signaling, gene expression, protein stability and interactions, and enzyme kinetics. Following a brief discussion of the experimental and bioinformatics challenges of PTM identification, localization, and quantification (occupancy), a concise overview is provided of the major PTMs and their (potential) functional consequences in plants, with emphasis on plant metabolism. Classic examples that illustrate the regulation of plant metabolic enzymes and pathways by PTMs and their cross talk are summarized. Recent large-scale proteomics studies mapped many PTMs to a wide range of metabolic functions. Unraveling of the PTM code, i.e. a predictive understanding of the (combinatorial) consequences of PTMs, is needed to convert this growing wealth of data into an understanding of plant metabolic regulation.

The primary amino acid sequence of proteins is defined by the translated mRNA, often followed by N- or C-terminal cleavages for preprocessing, maturation, and/or activation. Proteins can undergo further reversible or irreversible posttranslational modifications (PTMs) of specific amino acid residues. Proteins are directly responsible for the production of plant metabolites because they act as enzymes or as regulators of enzymes. Ultimately, most proteins in a plant cell can affect plant metabolism (e.g. through effects on plant gene expression, cell fate and development, structural support, transport, etc.). Many metabolic enzymes and their regulators undergo a variety of PTMs, possibly resulting in changes in oligomeric state, stabilization/degradation, and (de)activation (Huber and Hardin, 2004), and PTMs can facilitate the optimization of metabolic flux. However, the direct in vivo consequence of a PTM on a metabolic enzyme or pathway is frequently not very clear, in part because it requires measurements of input and output of the reactions, including flux through the enzyme or pathway.

PTMs can occur spontaneously (nonenzymatically) due to the physical-chemical properties of reactive amino acids and the cellular environment (e.g. pH, oxygen, reactive oxygen species [ROS], and metabolites) or through the action of specific modifying enzymes

PTMs are also determined by neighboring residues and their exposure to the surface. The 20 amino acids differ strongly in their general chemical reactivity and their observed PTMs . In particular, Cys and Lys can each carry many types of PTMs, whereas the N-terminal residue of proteins is also prone to multiple PTMs, ranging from N-terminal cleavage to N-terminal lipid modifications (acylation), acetylation, and ubiquitination . In addition to these PTMs that occur in vivo and presumably have physiological significance, several PTMs are often generated during sample preparation due to exposure to organic solvents (e.g. formic acid leading to the formylation of Ser, Thr, and N termini), (thio) urea (N-terminal or Lys carbamylation), reducing agents and oxygen, unpolymerized acrylamide (Cys propionamide), and low or high pH (cyclization of N-terminal Gln or Glu into pyro-Glu;). A large-scale proteomics study of Arabidopsis (Arabidopsis thaliana) leaf extracts did address the frequency of PTMs that do not require specific affinity enrichment based on a data set of 1.5 million tandem mass spectrometry (MS/MS) spectra acquired at 100,000 resolution on an LTQ-Orbitrap instrument followed by error-tolerant searches and systematic validation by liquid chromatography retention time . This revealed, for example, that modification of Met and N-terminal Gln into oxidized Met and pyro-Glu, respectively, showed by far the highest modification frequencies, followed by N-terminal formylation, most likely induced during sample analysis, as well as deamidation of Asn/Gln (approximately 1.2% of all observed Asn/Gln). Several of these nonenzymatic PTMs (in particular deamidation, oxidation, and formylation) can also occur in vivo and, therefore, cannot be simply dismissed as artifacts but need to be considered as potential regulators.

Since many PTMs are reversible, specific residues can also alternate between PTMs (e.g. dependent on cellular conditions, protein configuration [folding], or protein-protein interactions), and one PTM can influence the generation of other PTMs. This can result in an explosion of possible proteoforms and in cross talk between PTMs occurring on the same protein. Cross talk between PTMs on the same protein can coordinately determine the activity, function, and/or interactions of a protein. Finally, cross talk also exists between PTMs located on interacting proteins. Time-resolved and quantitative determination of combinatorial PTMs is challenging, and understanding of the biological outcomes is only in its infancy. Prominent examples of PTM cross talk are Lys ubiquitination and acetylation or Lys ubiquitination and phosphorylation . Phosphorylation can also directly promote substrate proteolysis by caspase (peptidase) during apoptosis. Recent biochemical and proteomics studies suggested that most if not all enzymes of the Calvin-Benson cycle undergo redox

regulation through selective redox PTMs, including reversible disulfide bond formation, glutathionylation, and nitrosylation, with an interplay between these PTMs dependent on (sub)cellular conditions . Moreover, the regulators carrying out these PTMs (e.g. thioredoxins, glutaredoxins, etc.) themselves can also undergo some of these PTMs, making for a complex network of PTMs

The identification, localization, and quantification of different combinations of PTMs on the same protein can sometimes be better solved by so-called top-down or middle-down proteomics, as opposed to the more common bottom-up proteomics (. or chemical cleavage) prior to MS analysis. In contrast, in top-down proteomics, proteins are not digested into smaller fragments but rather injected and analyzed by a specialized mass spectrometer in its entirety. In middle-down proteomics, the intact proteins are cleaved into just a few fragments by limited proteolysis prior to injection into the mass spectrometer. Top-down and middle-down proteomics are not high throughput and are best carried out on either purified proteins or protein mixtures of low complexity. Classic examples of studies using top-down, middle-down, but also bottom-up proteomics on proteins with different PTMs involve histones) and the p53 tumor suppression protein.

References

Agetsuma M, Furumoto T, Yanagisawa S, Izui K (2005) The ubiquitin-proteasome pathway is involved in rapid degradation of phosphoenolpyruvate carboxylase kinase for C4 photosynthesis. Plant Cell Physiol **46**: 389–398.

Akter S, Huang J, Waszczak C, Jacques S, Gevaert K, Van Breusegem F, Messens J (2015) Cysteines under ROS attack in plants: a proteomics view. J Exp Bot **66**: 2935–2944.

Alban C, Tardif M, Mininno M, Brugière S, Gilgen A, Ma S, Mazzoleni M, Gigarel O, Martin-Laffon J, Ferro M, et al. (2014) Uncovering the protein lysine and arginine methylation network in Arabidopsis chloroplasts. PLoS One **9**: e95512.

Altelaar AF, Munoz J, Heck AJ (2013) Next-generation proteomics: towards an integrative view of proteome dynamics. Nat Rev Genet **14**: 35–48.

Bailey KJ, Gray JE, Walker RP, Leegood RC (2007) Coordinate regulation of phosphoenolpyruvate carboxylase and phospho*enol*pyruvate carboxykinase by light and CO_2 during C_4 photosynthesis. Plant Physiol **144**: 479–486.

Balmer Y, Vensel WH, Tanaka CK, Hurkman WJ, Gelhaye E, Rouhier N, Jacquot JP, Manieri W, Schürmann P, Droux M, et al. (2004) Thioredoxin links redox to the regulation of fundamental processes of plant mitochondria. Proc Natl Acad Sci USA **101**: 2642–2647.

Balsera M, Uberegui E, Schürmann P, Buchanan BB (2014) Evolutionary development of redox regulation in chloroplasts. Antioxid Redox Signal **21**: 1327–1355.

Banerjee A, Sharkey TD (2014) Methylerythritol 4-phosphate (MEP) pathway metabolic regulation. Nat Prod Rep **31**: 1043–1055.

Barberon M, Zelazny E, Robert S, Conéjéro G, Curie C, Friml J, Vert G (2011) Monoubiquitin-dependent endocytosis of the iron-regulated transporter 1 (IRT1) transporter controls iron uptake in plants. Proc Natl Acad Sci USA **108**: E450–E458.

Bartel B, Citovsky V (2012) Focus on ubiquitin in plant biology. Plant Physiol **160**: 1.

Bartsch O, Mikkat S, Hagemann M, Bauwe H (2010) An autoinhibitory domain confers redox regulation to maize glycerate kinase. Plant Physiol **153**: 832–840.

Berr A, Shafiq S, Shen WH (2011) Histone modifications in transcriptional activation during plant development. Biochim Biophys Acta **1809**: 567–576.

Bigeard J, Rayapuram N, Pflieger D, Hirt H (2014) Phosphorylation-dependent regulation of plant chromatin and chromatin-associated proteins. Proteomics **14**: 2127–2140.

Biggar KK, Li SS (2015) Non-histone protein methylation as a regulator of cellular signalling and function. Nat Rev Mol Cell Biol **16**: 5–17.

Bonissone S, Gupta N, Romine M, Bradshaw RA, Pevzner PA (2013) N-terminal protein processing: a comparative proteogenomic analysis. Mol Cell Proteomics **12**: 14–28.

Borner GH, Lilley KS, Stevens TJ, Dupree P (2003) Identification of glycosylphosphatidylinositol-anchored proteins in Arabidopsis: a proteomic and genomic analysis. Plant Physiol **132**: 568–577.

Boyle PC, Martin GB (2015) Greasy tactics in the plant-pathogen molecular arms race. J Exp Bot **66**: 1607–1616.

Bracha-Drori K, Shichrur K, Lubetzky TC, Yalovsky S (2008) Functional analysis of Arabidopsis postprenylation CaaX processing enzymes and their function in subcellular protein targeting. Plant Physiol **148**: 119–131.

Brzezowski P, Richter AS, Grimm B (2015) Regulation and function of tetrapyrrole biosynthesis in plants and algae. Biochim Biophys Acta **1847**: 968–985.

Carlson SM, Gozani O (2014) Emerging technologies to map the protein methylome. J Mol Biol **426**: 3350–3362.

Catherman AD, Skinner OS, Kelleher NL (2014) Top down proteomics: facts and perspectives. Biochem Biophys Res Commun **445**: 683–693.

Cavazzini D, Meschi F, Corsini R, Bolchi A, Rossi GL, Einsle O, Ottonello S (2013) Autoproteolytic activation of a symbiosis-regulated truffle phospholipase A2. J Biol Chem **288**: 1533–1547.

Černý M, Skalák J, Cerna H, Brzobohatý B (2013) Advances in purification and separation of posttranslationally modified proteins. J Proteomics **92**: 2–27.

Chalkley RJ, Bandeira N, Chambers MC, Clauser KR, Cottrell JS, Deutsch EW, Kapp EA, Lam HH, McDonald WH, Neubert TA, et al. (2014) Proteome informatics research group (iPRG)_2012: a study on detecting modified peptides in a complex mixture. Mol Cell Proteomics **13**: 360–371.

Chalkley RJ, Clauser KR (2012) Modification site localization scoring: strategies and performance. Mol Cell Proteomics **11**: 3–14.

Champion A, Kreis M, Mockaitis K, Picaud A, Henry Y (2004) Arabidopsis kinome: after the casting. Funct Integr Genomics **4**: 163–187.

Chastain CJ, Failing CJ, Manandhar L, Zimmerman MA, Lakner MM, Nguyen TH (2011) Functional evolution of C(4) pyruvate, orthophosphate dikinase. J Exp Bot **62**: 3083–3091.

Chen YB, Lu TC, Wang HX, Shen J, Bu TT, Chao Q, Gao ZF, Zhu XG, Wang YF, Wang BC (2014) Posttranslational modification of maize chloroplast pyruvate orthophosphate dikinase reveals the precise regulatory mechanism of its enzymatic activity. Plant Physiol **165**: 534–549.

Choudhary C, Weinert BT, Nishida Y, Verdin E, Mann M (2014) The growing landscape of lysine acetylation links metabolism and cell signalling. Nat Rev Mol Cell Biol **15**: 536–550.

Christian JO, Braginets R, Schulze WX, Walther D (2012) Characterization and prediction of protein phosphorylation hotspots in Arabidopsis thaliana. Front Plant Sci **3**: 207.

Cieśla J, Frączyk T, Rode W (2011) Phosphorylation of basic amino acid residues in proteins: important but easily missed. Acta Biochim Pol **58**: 137–148.

Cox J, Mann M (2011) Quantitative, high-resolution proteomics for data-driven systems biology. Annu Rev Biochem **80**: 273–299.

Czyzewicz N, Yue K, Beeckman T, De Smet I (2013) Message in a bottle: small signalling peptide outputs during growth and development. J Exp Bot **64**: 5281–5296.

Daloso DM, Müller K, Obata T, Florian A, Tohge T, Bottcher A, Riondet C, Bariat L, Carrari F, Nunes-Nesi A, et al. (2015) Thioredoxin, a master regulator of the tricarboxylic acid cycle in plant mitochondria. Proc Natl Acad Sci USA **112**: E1392–E1400.

de Boer AH, van Kleeff PJ, Gao J (2013) Plant 14-3-3 proteins as spiders in a web of phosphorylation. Protoplasma **250**: 425–440.

DeHart CJ, Chahal JS, Flint SJ, Perlman DH (2014) Extensive post-translational modification of active and inactivated forms of endogenous p53. Mol Cell Proteomics **13**: 1–17.

Denison FC, Paul AL, Zupanska AK, Ferl RJ (2011) 14-3-3 proteins in plant physiology. Semin Cell Dev Biol **22**: 720–727.

Dietz KJ, Hell R (2015) Thiol switches in redox regulation of chloroplasts: balancing redox state, metabolism and oxidative stress. Biol Chem **396**: 483–494.

Dinh TV, Bienvenut WV, Linster E, Feldman-Salit A, Jung VA, Meinnel T, Hell R, Giglione C, Wirtz M (2015) Molecular identification and functional characterization of the first Nα-acetyltransferase in plastids by global acetylome profiling. Proteomics **15**: 2426–2435.

di Pietro M, Vialaret J, Li GW, Hem S, Prado K, Rossignol M, Maurel C, Santoni V (2013) Coordinated post-translational responses of aquaporins to abiotic and nutritional stimuli in Arabidopsis roots. Mol Cell Proteomics **12**: 3886–3897.

Dix MM, Simon GM, Wang C, Okerberg E, Patricelli MP, Cravatt BF (2012) Functional interplay between caspase cleavage and phosphorylation sculpts the apoptotic proteome. Cell **150**: 426–440.

Dong L, Ermolova NV, Chollet R (2001) Partial purification and biochemical characterization of a heteromeric protein phosphatase 2A holoenzyme from maize (Zea mays L.) leaves that dephosphorylates C4 phosophoenolpyruvate carboxylase. Planta **213**: 379–389.

Duncan KA, Huber SC (2007) Sucrose synthase oligomerization and F-actin association are regulated by sucrose concentration and phosphorylation. Plant Cell Physiol **48**: 1612–1623.

Elortza F, Mohammed S, Bunkenborg J, Foster LJ, Nühse TS, Brodbeck U, Peck SC, Jensen ON (2006) Modification-specific proteomics of plasma membrane proteins: identification and characterization of glycosylphosphatidylinositol-anchored proteins released upon phospholipase D treatment. J Proteome Res **5**: 935–943.

Elrouby N, Coupland G (2010) Proteome-wide screens for small ubiquitin-like modifier (SUMO) substrates identify Arabidopsis proteins implicated in diverse biological processes. Proc Natl Acad Sci USA **107**: 17415–17420.

Engineer CB, Ghassemian M, Anderson JC, Peck SC, Hu H, Schroeder JI (2014) Carbonic anhydrases, EPF2 and a novel protease mediate CO_2 control of stomatal development. Nature **513**: 246–250.

Fedorova M, Bollineni RC, Hoffmann R (2014) Protein carbonylation as a major hallmark of oxidative damage: update of analytical strategies. Mass Spectrom Rev **33**: 79–97.

Fedosejevs ET, Ying S, Park J, Anderson EM, Mullen RT, She YM, Plaxton WC (2014) Biochemical and molecular characterization of RcSUS1, a cytosolic sucrose synthase phosphorylated in vivo at serine 11 in developing castor oil seeds. J Biol Chem **289**: 33412–33424.

Ferrández-Ayela A, Micol-Ponce R, Sánchez-García AB, Alonso-Peral MM, Micol JL, Ponce MR (2013) Mutation of an Arabidopsis NatB N-alpha-terminal acetylation complex component causes pleiotropic developmental defects. PLoS One **8**: e80697.

Finkemeier I, Laxa M, Miguet L, Howden AJ, Sweetlove LJ (2011) Proteins of diverse function and subcellular location are lysine acetylated in Arabidopsis. Plant Physiol **155**: 1779–1790.

Gao ZP, Chen GX, Yang ZN (2012) Regulatory role of Arabidopsis pTAC14 in chloroplast development and plastid gene expression. Plant Signal Behav **7**: 1354–1356.

Geigenberger P. (2011) Regulation of starch biosynthesis in response to a fluctuating environment. Plant Physiol **155**: 1566–1577.

Geigenberger P, Kolbe A, Tiessen A (2005) Redox regulation of carbon storage and partitioning in response to light and sugars. J Exp Bot **56**: 1469–1479.

Giglione C, Fieulaine S, Meinnel T (2015) N-terminal protein modifications: bringing back into play the ribosome. Biochimie **114**: 134–146.

Graciet E, Lebreton S, Gontero B (2004) Emergence of new regulatory mechanisms in the Benson-Calvin pathway via protein-protein interactions: a glyceraldehyde-3-phosphate dehydrogenase/CP12/phosphoribulokinase complex. J Exp Bot **55**: 1245–1254.

Grimaud F, Rogniaux H, James MG, Myers AM, Planchot V (2008) Proteome and phosphoproteome analysis of starch granule-associated proteins from normal maize and mutants affected in starch biosynthesis. J Exp Bot **59**: 3395–3406.

Guerra DD, Callis J (2012) Ubiquitin on the move: the ubiquitin modification system plays diverse roles in the regulation of endoplasmic reticulum- and plasma membrane-localized proteins. Plant Physiol **160**: 56–64.

Haag F, Buck F (2015) Identification and analysis of ADP-ribosylated proteins. Curr Top Microbiol Immunol **384**: 33–50.

Hang R, Liu C, Ahmad A, Zhang Y, Lu F, Cao X (2014) Arabidopsis protein arginine methyltransferase 3 is required for ribosome biogenesis by affecting precursor ribosomal RNA processing. Proc Natl Acad Sci USA **111**: 16190–16195.

Havelund JF, Thelen JJ, Møller IM (2013) Biochemistry, proteomics, and phosphoproteomics of plant mitochondria from non-photosynthetic cells. Front Plant Sci **4**: 51.

Hemsley PA. (2014) Progress in understanding the mechanisms and functional importance of protein-membrane interactions in plants. New Phytol **204**: 741–743.

Hemsley PA. (2015) The importance of lipid modified proteins in plants. New Phytol **205**: 476–489.

Hemsley PA, Weimar T, Lilley K, Dupree P, Grierson C (2013a) Palmitoylation in plants: new insights through proteomics. Plant Signal Behav **8**: 8.

Hemsley PA, Weimar T, Lilley KS, Dupree P, Grierson CS (2013b) A proteomic approach identifies many novel palmitoylated proteins in Arabidopsis. New Phytol **197**: 805–814.

Heyl A, Brault M, Frugier F, Kuderova A, Lindner AC, Motyka V, Rashotte AM, Schwartzenberg KV, Vankova R, Schaller GE (2013) Nomenclature for members of the two-component signaling pathway of plants. Plant Physiol **161**: 1063–1065.

Hodges M, Jossier M, Boex-Fontvieille E, Tcherkez G (2013) Protein phosphorylation and photorespiration. Plant Biol (Stuttg) **15**: 694–706.

@ Disclaimer

The information contained in this reference manual has been taken from various web resources. The information is provided by "ICAR-IASRI" and whilst we endeavour to keep the information up-to-date and correct, we make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability, or availability with respect to the website or the information, products, services, or related graphics contained in the reference manual for any purpose. Any reliance you place on such information is therefore strictly at your own risk. In no event will we be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from loss of data or profits arise out of or in connection with the use of this manual. We have no control over the nature, content and availability of those sites. The inclusion of any links does not necessarily imply a recommendation or endorse the views expressed within them.

@ Citation

Srivastava Sudhir, Murmu Sneha, Jha Girish Kumar (2023). Omics Data Analysis: Genome to Proteome, Training under ICAR-Consortium Research Platform on Genomics, E-Manual, ICAR-Indian Agricultural Statistics Research Institute, New Delhi.