हर कदम, हर डगर
किसानों का हमसफर
भारतीय कृषि अनुसंधान परिषद

Agrisearch with a human touch

ऑनलाइन प्रशिक्षण कार्यक्रम

ट्रांसक्रिप्टोमिक डेटा विश्लेषण

Online Training Programme

Transcriptomic Data Analysis

Sponsoring Project

DBT - Establishment of Centre for Bioinformatics and Computational Biology in Agriculture-BIC at ICAR-Indian Agricultural Statistics Research Institute

E-Manual

September 23-29, 2025

Course Adviser : Dr. Rajender Parsad

Course Director: Dr. Girish Kumar Jha

Course Coordinators: Dr. Mohammad Samir Farooqi

Dr. Sudhir Srivastava

Dr. Sneha Murmu

Division of Agricultural Bioinformatics

ICAR-Indian Agricultural Statistics Research Institute

Library Avenue, PUSA, New Delhi - 110012

https://iasri.res.in/

आमुख

जैव सूचना विज्ञान वस्तुतः जीव विज्ञान, कंप्यूटर विज्ञान और सांख्यिकी का अंतःविषय क्षेत्र है। पिछले दो दशकों के दौरान जैविक विज्ञान के क्षेत्र में बृहद डेटा (आंकड़ा) उत्पन्न किया गया जिसमें सबसे पहले जीवों के जिनोम अनुक्रमण की विषय में जानकारी प्राप्त की गई। इसके उपरान्त इन प्राप्त जानकारियों को उच्च प्रयोगात्मक तकनीक से जैव प्रौद्योगिकी अनुसंधान प्रयोगशालाओं में किये गये प्रयोगों तथा इसके प्रभावों की गतिशीलता का अध्ययन किया जा रहा है। जैविक अनुसंधान के क्षेत्र में विभिन्न जैवसूचना विज्ञान तकनीकों/ टूल्स के प्रयोग, डेटा की संचयन एवं पुनःप्राप्ति, विश्लेषण, एनोटेसन और परिणाम के अपनी सम्पूर्णता में जैविक प्रणालियों को बेहतर ढंग में समझने में सहायक है। इन तकनीकियों के द्वारा आनेवाले समय में कृषि जैव सूचना हेतु आवश्यक सामग्री, उपकरण एवं तकनीकों के विकास को बढ़ाने में सहायक हो रहा है। इस प्रशिक्षण का उद्देश्य एन.ए.आर.ई.एस. में कार्यरत वैज्ञानिक/ तकनीकी/ संविदात्मक अनुसंधान कर्मचारी वर्ग के लिए ट्रांसक्रिप्टोमिक डेटा विश्लेषण और कृषि में इसके अनुप्रयोगों का एक अवलोकन प्रदान करना है।

यह प्रशिक्षण कार्यक्रम मुख्य रूप से ट्रांसक्रिप्टोमिक डेटा विश्लेषण पर केंद्रित है। यह प्रशिक्षण निम्नलिखित उद्देश्यों के साथ तैयार किया गया है:

- जैव सूचना विज्ञान के टूल्स और तकनीकियों का उपयोग करके प्रतिभागियों को ट्रांसक्रिप्टोमिक डेटा विश्लेषण की सिद्धांतों से परिचित कराना
- उपयुक्त उदाहरण डेटा के साथ व्याख्यानों और प्रदर्शनों के माध्यम से उपरोक्त को स्पष्ट करना

इस प्रशिक्षण कार्यक्रम में (1) लिनक्स और आर का परिचय, (2) ओमिक्स डेटा में टेक्स्ट एनालिटिक्स, (3) एन.जी.एस. डेटा जनरेशन: तकनीक, मुद्दे और चुनौतियां, (4) एन.जी.एस. डेटा: प्री-प्रोसेसिंग, असेंबली और क्वांटिफिकेशन, (5) आर.एन.ए.-सेक डेटा का एनोटेशन, (6) बल्क और एकल-कोशिका आर.एन.ए.-सेक डेटा विश्लेषण तथा (7) माइक्रो आर.एन.ए. की पहचान और टारगेट प्रेडिक्शन, को शामिल किया गया है।

इस कार्यक्रम के आयोजन के समर्थन और सहयोग के लिए हम डॉ. राजेंद्र प्रसाद (निदेशक महोदय, भा.कृ.अनु.प.-भा.कृ.सां.अ.सं.) और डॉ. गिरीश कुमार झा (प्रधान प्रभाग, कृषि जैव सूचना विज्ञान) का आभार व्यक्त करते हैं।

लेखकगण

PREFACE

Bioinformatics is an interdisciplinary field comprising of biology, statistics and computer sciences. 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 better understanding of biological systems in their entirety. This will further lead to development of tools and techniques for sustainable agriculture. The aim of this training is to provide an overview of transcriptomic data analysis and its applications in agriculture to Scientific/ Technical/ Contractual Research Personnel working in NARES. This training programme is mainly focused on transcriptomic data analysis.

This training has been formulated with following objectives:

- To familiarize the participants with the concepts of transcriptomic data analysis using bioinformatics tools and techniques
- To illustrate the above through lectures and demonstrations with suitable example data

In this training programme, lectures related to (1) Introduction to Linux and R, (2) Text Analytics in Omics Data, (3) NGS Data Generation: Techniques, Issues and Challenges, (4) NGS Data: Pre-processing, Assembly and Quantification, (5) Annotation of RNA-Seq Data, (6) Bulk and Single-Cell RNA-Seq Data Analysis, and (7) microRNA Identification and Target Prediction, have been included.

We express our gratitude to Dr. Rajender Parsad (Director, ICAR-IASRI) and Dr. Girish Kumar Jha (Head, Division of Agricultural Bioinformatics) for the support and cooperation in organizing this program.

Authors

CONTENTS

Overview of 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*

Introduction to R

Neeraj Budhlakoti, Sudhir Srivastava, D. C. Mishra
ICAR-IASRI, New Delhi

Introduction

R is a powerful programming language for statistical analysis. This software is an implementation of S programming language which was designed by John Chambers at Bell Labs. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. It is currently developed by R Development Core Team. The name `R' is from the first names of first two authors and partly due to its inheritance from `S'. Recently R has become one of world's popular statistical analysis software, because of the following reasons:

(i) R is free, open source and capable of almost any statistical analysis including the most recently developed statistical methodologies.

(ii) R provides very good graphical facilities.

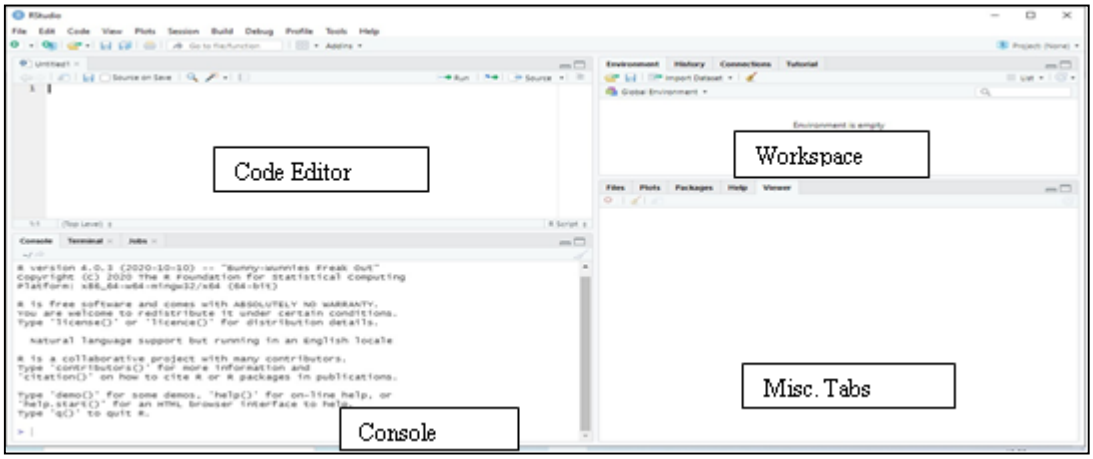
(iii) R is easily extensible through new contributions from statisticians and researchers around the globe, which also makes R quite different from other popular statistical analysis software. In fact, R community is highly active in terms of new contributions in the form of packages to R.

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

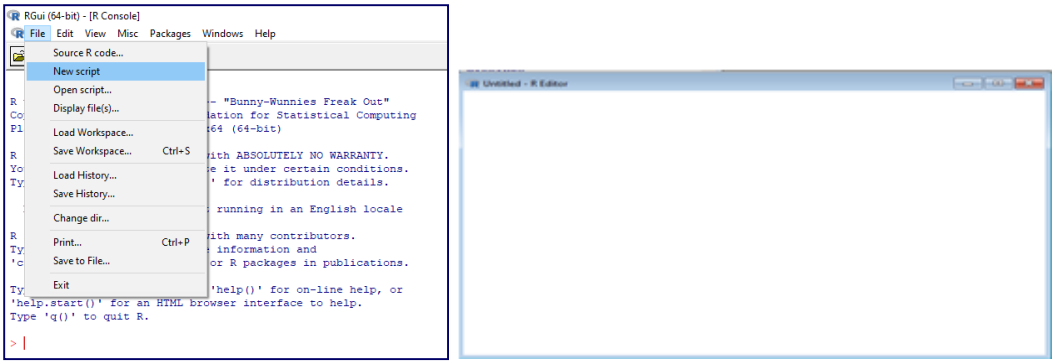
RStudio is a free, open-source IDE (integrated development environment) for R. It can be downloaded from https://www.rstudio.com/products/rstudio/download/. One must install R before installing RStudio. The interface is organized so that the user can clearly view graphs, data tables, R code, and output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



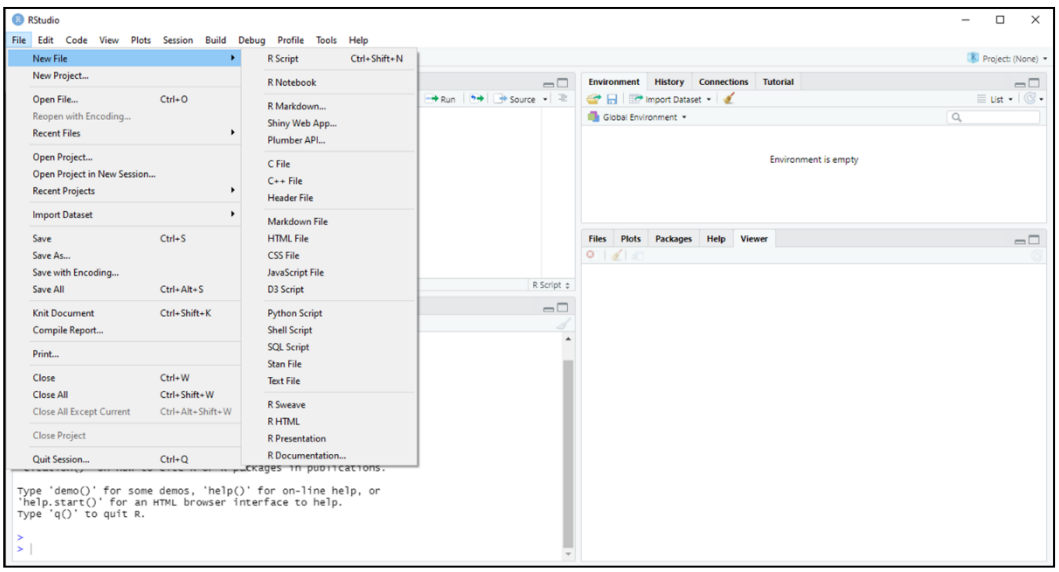
R Studio Interface

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.

Getting and installing R

To be able to use R, it needs to be installed in computer. R is available for free download from any one of the mirror sites of Comprehensive R Archive Network (CRAN) in http://cran.r-project.org/. For downloading, it is better to select a mirror located nearer to you. R is available for installation in Windows/Macintosh/Unix platforms. To install R in a given machine, first double-click the downloaded file R.exe, then select language as `English'. R setup wizard window will appear. Select on `Next' and accept most of the default settings during the installation.

To start R, click on start menu → all programs → R → R 3.4.1 and a screen as shown in Fig. 1 appears. The white blank screen is called R Console and this is the place where all R codes are written and outputs appear, unless outputs are directed to some external files. There will be a toolbar at the top of the Console and a few menus in the R window. To know what the buttons on the toolbar does, hold your mouse on the button for some time, a description of the button will appear. There is an R editor which can be used to write and edit R codes. R editor window is just like a text editor with facilities for select, cut, copy, paste, typing text, deleting text etc. This window opens by clicking on File → New Script in the menu bar. The codes written in R editor window needs to be passed to the R console for execution by clicking on `Run line or selection button' on the toolbar in the R editor window.

Downloading and installing a package

To use an R package, download the package from CRAN and then install and load it in an R session. A package can be downloaded from within R or from outside R. On a Windows machine which is connected to internet, a package can be installed by clicking on Packages ! Install packages(s) from the menu bar. This will open a list of mirrors. Select a mirror and then, from the available list of packages in the website, select the desired packages. They will be installed into R.

One can also install the package from command line and then package can be loaded using following command.

> install.packages("agricolae")

> library(agricolae)

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

R has an inbuilt help facility similar to the man facility of UNIX. To get more information on any specific named function, for example solve, the command is

```
> help(solve)
```

An alternative is

```
> ?solve
```

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 syntacticmeaning including if, for and function.

```
> help("[[")
```

Either form of quote mark may be used to escape the other, as in the string "It’s important". Our 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. On UNIX, subsequent help requests are sent to the HTML-based help

system. The ‘Search Engine and Keywords’ link in the page loaded by help.start() is particularly useful as it is contains a high-level concept list which searches though available functions. It can be a great way to get your bearings quickly and to understand the breadth of what R has to offer. The help.search command (alternatively ??) allows searching for help in various ways. For example,

> ??solve

Try ?help.search for details and more examples.

The examples on a help topic can normally be run by

> example(topic)

Windows versions of R have other optional help systems: use

> ?help

Data types in R

There is a > symbol in the Console. The commands are typed after this symbol and then the Enter button needs to be pressed. When a command is written in the Console and the Enter button is pressed, R reads the commands and returns some results or some error message on the Console.

R is an object oriented language and therefore, all data types in R are some kind of object. Objects may be variables, vectors, matrices, arrays, character strings, functions, or more general structures built from such components. During an R session, objects are created and stored by name. Once can use the command

> objects()

to display the names of the objects which are currently stored within R. The collection of objects currently stored is called the workspace. One can remove objects using the function rm(). For example, the following code removes objects x and y form the workspace.

> rm(x, y)

An object created during an R session can be saved in a file for use in future R sessions. The entire workspace of an R session and the history of all the commands used during the session can also be saved. Some commonly encountered objects are discussed below.

Numbers

The most basic way to store a number is to make an assignment of a single number:

```
a <- 3
```

The "<-" tells R to take the number to the right of the symbol and store it in a variable whose name is given on the left. You can also use the "=" symbol. When you make an assignment R does not print out any information. If you want to see what value a variable has just type the name of the variable on a line and press the enter key:

```
> a
[1] 3
```

This allows you to do all sorts of basic operations and save the numbers:

```
> b <- sqrt(a*a+3)
> b
[1] 3.464102
```

Strings: One can not only limited to just storing numbers. We can also store strings. A string is specified by using quotes. Both single and double quotes will work:

```
> a <- "hello"
> a
[1] "hello"
> b <- c("hello","there")
> b
[1] "hello" "there"
> b[1]
[1] "hello"
```

Vectors

Simplest object in R is a vector. A vector is a collection of elements. For example, creates a vector of 5 numbers.

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

Matrices

A matrix object also is a collection of elements but it has two dimensions. They can also be numeric, character or logical in nature. Following is an example of creating a matrix.

```
> x=matrix(c("a","b","c","d"),nrow=2)
> x
[,1] [,2]
[1,] "a" "c"
[2,] "b" "d"
```

Matrix facilities: As noted above, a matrix is just an array with two subscripts. However it is such an important special case it needs a separate discussion. R contains many operators and functions that are available only for matrices. For example t(X) is the matrix transpose function, as noted above. The functions nrow(A) and ncol(A) give the number of rows and columns in the matrix A respectively.

The operator %*% is used for matrix multiplication. An n by 1 or 1 by n matrix may of course be used as an n-vector if in the context such is appropriate. Conversely, vectors which occur in matrix multiplication expressions are automatically promoted either to row or column vectors, whichever is multiplicatively coherent, if possible, (although this is not always unambiguously possible, as we see later).

If, for example, A and B are square matrices of the same size, then

```
> A * B
```

is the matrix of element by element products and

> A %*% B

is the matrix product. If x is a vector, then

> x %*% A %*% x

is a quadratic form. The function crossprod() forms "crossproducts", meaning that crossprod(X, y) is the same as t(X) %*% y but the operation is more efficient.

Forming partitioned matrices with cbind() and rbind(): As we have already seen informally, matrices can be built up from other vectors and matrices by the functions cbind() and rbind(). Roughly cbind() forms matrices by binding together matrices horizontally, or column-wise, and rbind() vertically, or row-wise.

In the assignment

> X <- cbind(arg_1, arg_2, arg_3, ...)

the arguments to cbind() must be either vectors of any length, or matrices with the same column size, that is the same number of rows. The result is a matrix with the concatenated arguments arg 1, arg 2, . . . forming the columns. If some of the arguments to cbind() are vectors they may be shorter than the column size of any matrices present, in which case they are cyclically extended to match the matrix column size (or the length of the longest vector if no matrices are given).

The function rbind() does the corresponding operation for rows. In this case any vector argument, possibly cyclically extended, are of course taken as row vectors. Suppose X1 and X2 have the same number of rows. To combine these by columns into a matrix X, together with an initial column of 1s we can use

> X <- cbind(1, X1, X2)

The result of rbind() or cbind() always has matrix status. Hence cbind(x) and rbind(x) are possibly the simplest ways explicitly to allow the vector x to be treated as a column or row matrix respectively.

Arrays

Arrays are multi-dimensional generalization of vectors and matrices. A two-dimensional array is a matrix. Arrays can have more than two dimensions. Arrays can be constructed from vectors by the array function, which has the form

> Z <- array(data_vector, dim_vector)

For example, if the vector h contains 24 or fewer, numbers then the command

> Z <- array(h, dim=c(3,4,2))

Factors

Factor objects are used to specify categorical or classificatory or grouping variables. For example, males and females are two levels of a variable gender. Then gender can be thought of a factor object.

```
> gender=c("M", "F", "M")
> gender=as.factor(gender)
> levels(gender)
[1] "F" "M"
```

Factor variables are particularly useful in analysis of variance and in linear model with grouping variables.

Lists

A list is a collection of objects where each object can be of different type. For example, a list can have first object as a vector, second object as a matrix and third object as a data frame.

```
> mylist=list(x=c(10,20,30),y=matrix(1:6,nrow=3))
> mylist
$x
[1] 10 20 30
$y
[,1] [,2]
[1,] 1 4
[2,] 2 5
```

[3,] 3 6

> mylist[[1]]

Data frames

A data frame is a two dimensional object. But unlike matrices, different columns of data frame can be different types, for example some columns can be numeric, some columns can be character, some columns can be factors. Here a column generally refers to a variable.

> age=c(20,25,28,30,26)

> weight=c(50,53,54,55,51)

> mydata=data.frame(age,weight)

> mydata

age weight

1 20 50

2 25 53

3 28 54

4 30 55

5 26 51

The data.frame() function is used to create a data frame.

Functions

Functions in R are a kind of objects which takes one or more inputs and produces some result(s) as output. R has a number of in-built functions. R also provides facility to create new functions by users. R has huge number of in-built functions. As a simple example, to obtain the mean and variance of a set of numbers 10, 13, 21, 34, 51, 32, 45, 32, 17, 29, 41, 52, the following code can be used.

> x=c(10,13,21,34,51,32,45,32,17,29,41,52)

> mean(x)

[1] 31.41667

> var(x)

[1] 200.9924

Here, c(), mean() and var() are in-built functions of R. The function c() assigns those numbers to the object x. The commands mean(x) and var(x) computes the mean and variance of an object x. Here, x is the input, also called argument, to the function mean() and var().

Reading data from files

Large data objects will usually be read as values from external files rather than entered during an R session at the keyboard. R input facilities are simple and their requirements are fairly strict and even rather inflexible. There is a clear presumption by the designers of R that you will be able to modify your input files using other tools, such as file editors or Perl1 to fit in with the requirements of R. Generally this is very simple. If variables are to be held mainly in data frames, as we strongly suggest they should be, an entire data frame can be read directly with the read.table() function. There is also a more primitive input function, scan(), that can be called directly. For more details on importing data into R and also exporting data, see the R Data Import/Export manual.

Reading data from a spreadsheet

To read an entire data frame directly, the external file will normally have a special form. The first line of the file should have a name for each variable in the data frame. Each additional line of the file has as its first item a row label and the values for each variable.

```
> mydata2=read.table("rainfall.txt",header=TRUE,sep=",")
```

Reading data from Excel sheet

```
> library(xlsx)
> read.xlsx("myfile.xlsx", sheetName = "Sheet1")
```

Reading data from a particular Web page

```
> webdata=read.table("http://data.princeton.edu/wws509/datasets/effort.dat")
> webdata
```

Some Hands on with R

Examining the distribution of a set of data

Given a (univariate) set of data we can examine its distribution in a large number of ways. The simplest is to examine the numbers. Two slightly different summaries are given by summary function.

```
> attach(faithful)
> summary(eruptions)
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.600 2.163 4.000 3.488 4.454 5.100
> fivenum(eruptions)
[1] 1.6000 2.1585 4.0000 4.4585 5.1000
> stem(eruptions)
> hist(eruptions)
## make the bins smaller, make a plot of density
> hist(eruptions, seq(1.6, 5.2, 0.2), prob=TRUE)
> lines(density(eruptions, bw=0.1))
> rug(eruptions) # show the actual data points
```

Fitting Linear models

The basic function for fitting ordinary multiple models is lm(), and a streamlined version of the call is as follows:

```
> fitted.model <- lm(formula, data = data.frame)
```

For example

```
> fm2 <- lm(y ~ x1 + x2, data = production)
```

would fit a multiple regression model of y on x1 and x2 (with implicit intercept term).

Fitting Generalized linear model

Since the distribution of the response depends on the stimulus variables through a single linear function only, the same mechanism as was used for linear models can still be used to specify the linear part of a generalized model. The family has to be specified in a different way. The R function to fit a generalized linear model is glm() which uses the form

> fitted.model <- glm(formula, family=family.generator, data=data.frame)

Linear equations and inversion

Solving linear equations is the inverse of matrix multiplication. When after

> b <- A %*% x

only A and b are given, the vector x is the solution of that linear equation system. In R,

> solve(A,b)

solves the system, returning x (up to some accuracy loss).

Eigenvalues and eigenvectors

The function eigen(Sm) calculates the eigenvalues and eigenvectors of a symmetric matrix Sm. The result of this function is a list of two components named values and vectors. The assignment

> ev <- eigen(Sm)

will assign this list to ev. Then ev$val is the vector of eigenvalues of Sm and ev$vec is the matrix of corresponding eigenvectors. Had we only needed the eigenvalues we could have used the assignment:

> evals <- eigen(Sm)$values

evals now holds the vector of eigenvalues and the second component is discarded. If the expression

> eigen(Sm)

is used by itself as a command the two components are printed, with their names. For large matrices it is better to avoid computing the eigenvectors if they are not needed by using the expression

> evals <- eigen(Sm, only.values = TRUE)$values

Generating Plots

One of the most frequently used plotting functions in R is the plot() function. This is a generic function: the type of plot produced is dependent on the type or class of the first argument.

plot(x, y)

plot(xy)

If x and y are vectors, plot(x, y) produces a scatterplot of y against x. The same effect can be produced by supplying one argument (second form) as either a list containing two elements x and y or a two-column matrix.

plot(x) If x is a time series, this produces a time-series plot. If x is a numeric vector, it produces a plot of the values in the vector against their index in the vector. If x is a complex vector, it produces a plot of imaginary versus real parts of the vector elements.

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.

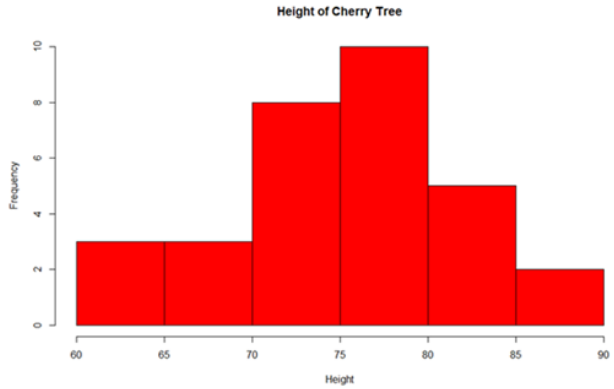
> data(trees)

***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.

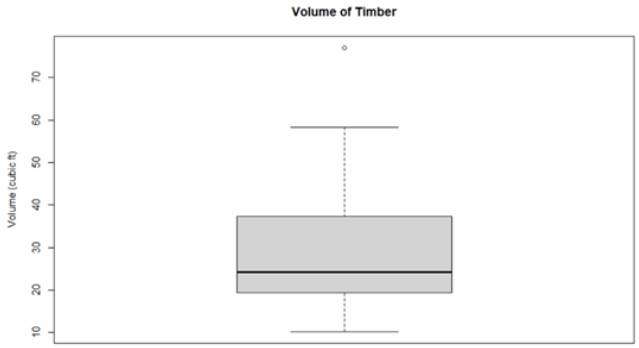
***Changing colors*:** Use the col statement

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))

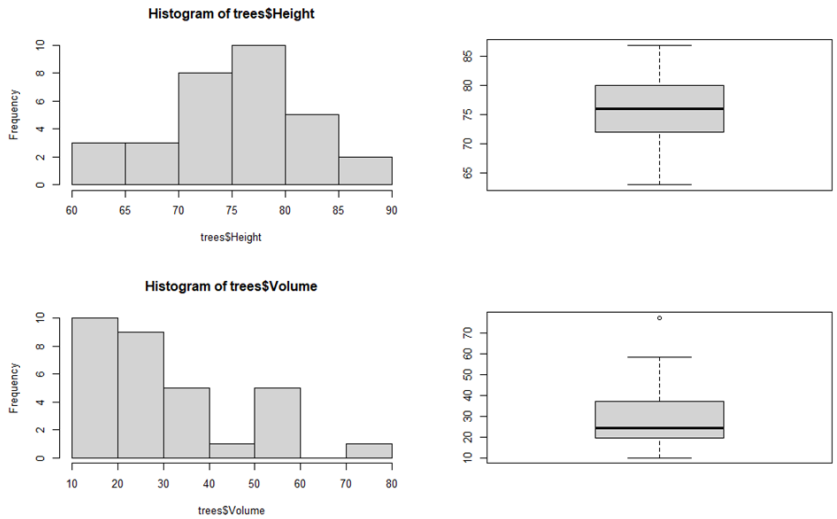
> hist(trees$Height)

> boxplot(trees$Height)

> hist(trees$Volume)

> boxplot(trees$Volume)

> par(mfrow=c(1,1))



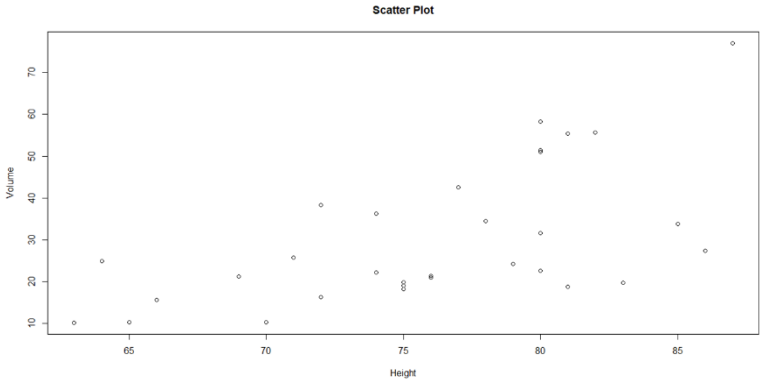
- Use layout()

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.

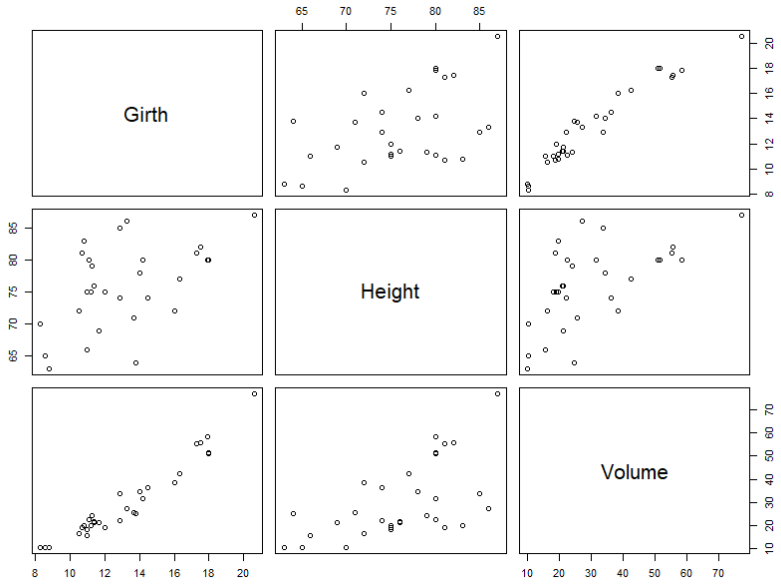
> plot(trees$Height, trees$Volume, xlab="Height", ylab="Volume", main="Scatter Plot", pch=20)



parameter pch stands for ‘plotting character’.

> pairs(trees)

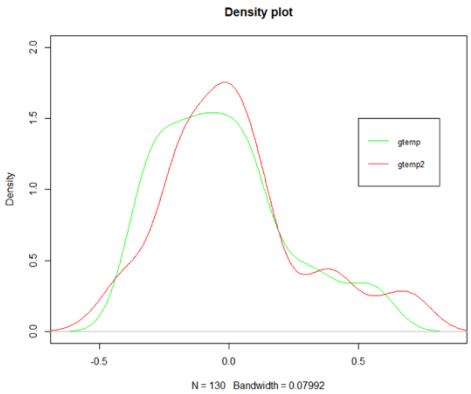
A matrix of scatterplots is produced.



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)
```



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

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/

Text Analytics in Omics Data

K.K. Chaturvedi and M.S. Farooqi

Introduction

Text mining is about extracting the information and knowledge from the given corpora of text or text related sources. Text mining is the process of compiling, organizing, and analyzing large set of documents to discover undiscovered knowledge from the available texts which will be useful for analysts and decision makers. In simple words, it can be defined as discovery of undiscovered knowledge from the available text. Text mining is one of the upcoming important areas of data mining to find or search useful and novel information from the available text data or documents.

Data mining defined as identifying the data sets, selection of features, preparing data and analyzing the trends and distribution to discover novel information or knowledge. In addition to this, the text mining refers to identify the documents or set of documents, extracting features, select the useful or important features using suitable data reduction techniques, preparing data sets and then analyze the trends and distribution to uncover the hidden information/knowledge using syntactic and semantic way.

Text mining is solely depend on the language and can be achieved by using Natural Language processing techniques. Textual data contains group of sentences or paragraphs or words that determine its purpose and use. The analysis of textual contents is difficult because of not containing any numeric values as it is desirable in any analytical technique. The text data need to be processed and converted into numeric form to analyze it. The basic application areas starts from feedback processing, opinion mining, text summarization, news report analysis, spam filtering and identification, document similarity, document retrieval, biological data repositories etc. The text is usually contains many relevant as well as irrelevant information lying in the document. The irrelevant or un-useful information may be discarded during the pre-processing of the documents or report. The text mining is two stage processes, i.e. pre-processing of documents and application of mining techniques.

Omics refers to a field of study in biology ending in -omics, such as genomics, proteomics or metabolomics. The suffix -ome is used to study in genome, proteome or metabolome respectively. Text mining in omics data is also refers as biomedical text mining (also known as BioNLP). In BioNLP, text mining is applied to biomedical and molecular biology texts and literature. It is a rather recent research field on the edge of natural language processing, bioinformatics, medical informatics and computational linguistics.

Pre-processing of Text Data

The standard steps for pre-processing of any textual document contains following steps:

Tokenization: Tokenization represents the process of converting a stream of characters into sequence of tokens. The token can be a sentence or a paragraph or a line or a word or an alphabet. Tokenization refers to separate the token from the given text. Here, we consider the token as a word or a term. We remove all punctuation marks, non-printable characters and other meaningless symbols as they may not contribute in the classification task. All capital letters are also replaced by small case letters.

Stop Word Removal: Stop words are those words which are commonly used in the texts but do not carry useful meaning. A list of English stop words, i.e. prepositions, conjunctions, articles, verbs, nouns, pronouns, adverbs, adjectives, etc. can be downloaded from

www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words. In our study, we have used SMART data set as a stop word list. We have also created our own user defined list as a dictionary of stop words which contains some other language characters, numbers and other special characters.

Stemming: A process of converting the derived word to their base word is called stemming. The base word is known as stem. As we know that each individual term can be expressed in many forms and carry special meaning. For example, the terms "computerized", "computerize", "computarization" and "computation" share the same base stem as "computer". Instead of considering all terms as individual terms, it can be replaced by a single term. In this way, the weightage can be increased for such terms and numbers of unnecessary terms can be avoided. Standard Porter stemming algorithm (Porter, 1980) can be utilized for word stemming.

Feature Reduction: Most frequently and less frequently occurring terms may be removed from the dataset. They may not be able to discriminate the documents or may lead to overfit/ underfit the model. Generally, in classification, the term which has less than three or more than forty/fifty occurrences will generally be removed from the dataset as most of the data mining algorithm may not be able to handle large feature sets.

Information Gain or InfoGain: It is an entropy based measure used for feature selection. Information gain is helpful in determining the importance or relevance of the attribute or feature or term or token. Information gain is used to rank all the terms in the data set. Selecting the top few terms helps in removing the non-relevant features in the data set. This is also helpful in determining the effect of number of terms to show any significant improvement in the performance of machine learning techniques.

Representation of Textual Data

Text data is generally available in unstructured format. To make it analytical friendly, there is a strong need to make it structured. The textual data can be divided into multiple attributes, i.e. title, heading, sections, subsections, paragraphs, etc. There are many ways to make the textual reports structured. Each textual document contains tokens and there are large numbers of documents available. This can be represented as document*term matrix where the rows are considered as documents and columns are considered as terms or tokens or words or features (Salton and Buckley, 1988). This matrix can be populated in either of the following ways:

Binary Representation: The matrix will be filled up by zero or one for the absence or presence of a term in the document. Such type of representation will be helpful in probability based model such as Naive Bayes classification where presence or absence of a term is considered. This type of representation is biased towards a larger document.

$$W_i = \begin{cases} 1, & if\ term\ appears\ in\ a\ document \\ 0, & else \end{cases}$$

Term Frequency Representation: In this type of representation, the frequency of a term in the document will be counted and stored in the matrix. This type of representation will give importance to the term in a particular document as in Multinomial Naive Bayes Classification. This will be normalized with the length of the document to avoid the bias towards the larger documents.

$$TF_i = Occurrences\ of\ a\ term\ in\ the\ document$$

This can be normalized, if we divide the TF (Term-Frequency) by total number of terms in the document

$$Norm\ (TF_i) = \frac{TF_i}{n}$$

Where n is the total number of terms in the document.

TF*IDF representation: TF*IDF stands for "term frequency (TF) times inverse document frequency (IDF)". It is generally used to determine the importance of a term in the complete dataset or document set. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. The inverse document frequency is obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient. Such type of representation is used in ranking of terms and selecting top few terms.

$$W_i = TF_i * IDF_i$$

$$\text{Where IDF}_\text{i} = \log\left(\frac{\text{N}}{\text{DF}_\text{i}}\right)$$

Where DFi is the document frequency and being defined as the appearance of a particularterm in the number of documents and N is the total number of documents.

Mining of Textual Data

Once the data table has been prepared, the data mining techniques can be applied based on the desirable task. Similar to data mining, there are many ways to handle this data using statistical, data mining, machine learning and soft computing techniques to discover and infer new knowledge with the existing literature. The categories of various tasks in text mining are data exploration, classification, clustering, association and visualization. General statistics can be seen with respect to frequency distribution and transformation of data into multiple ways. The level of natural language processing are summarized in fig. 1.

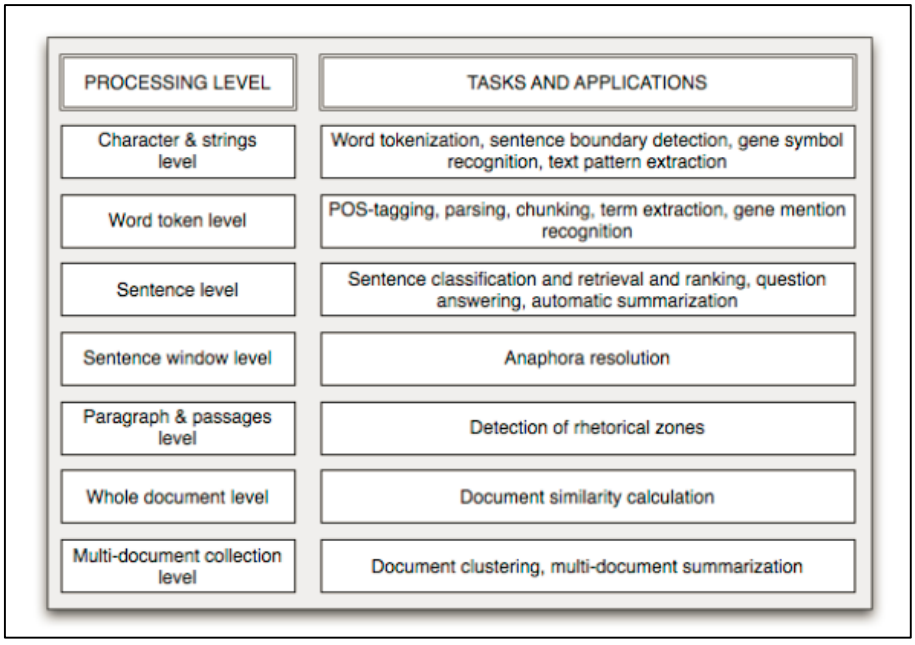


Fig. 1: Processing levels of NLP

Applications of Text Mining

The text mining is being applicable in wide areas of research started from bibliometric analysis to social media analytics. It is also applicable in government, research, and business as there is lot of documentations in the form of reports, articles, proceedings, compilations etc. are being made.

Developments in this area have been related to the identification of biological entities (named entity recognition), such as protein and gene names as well as chemical compounds and drugs [1] in free text, the association of gene clusters obtained by microarray experiments with the biological context provided by the corresponding literature, automatic extraction of protein interactions and associations of proteins to functional concepts (e.g. gene ontology terms). Even the extraction of kinetic parameters from text or the subcellular location of proteins have been addressed by information extraction and text mining technology. Information extraction and text mining methods have been explored to extract information related to biological processes and diseases. The applications can be classified as follows

- Spam filtering
- Reports and records management
- Sentiment Analysis Tools
- Opinion Mining
- Social media analytics for National Security/Intelligence
- Literature mining especially Life Sciences
- Feedback mining
- Creating recommendations
- Customer service support
- Document labeling and categorization
- Fraud detection by investigating the claims
- Fighting cyberbullying or cybercrime
- Duplicate and clone detection
- Plagiarism detection
- Search/Information Access
- Biology (Fig. 2)

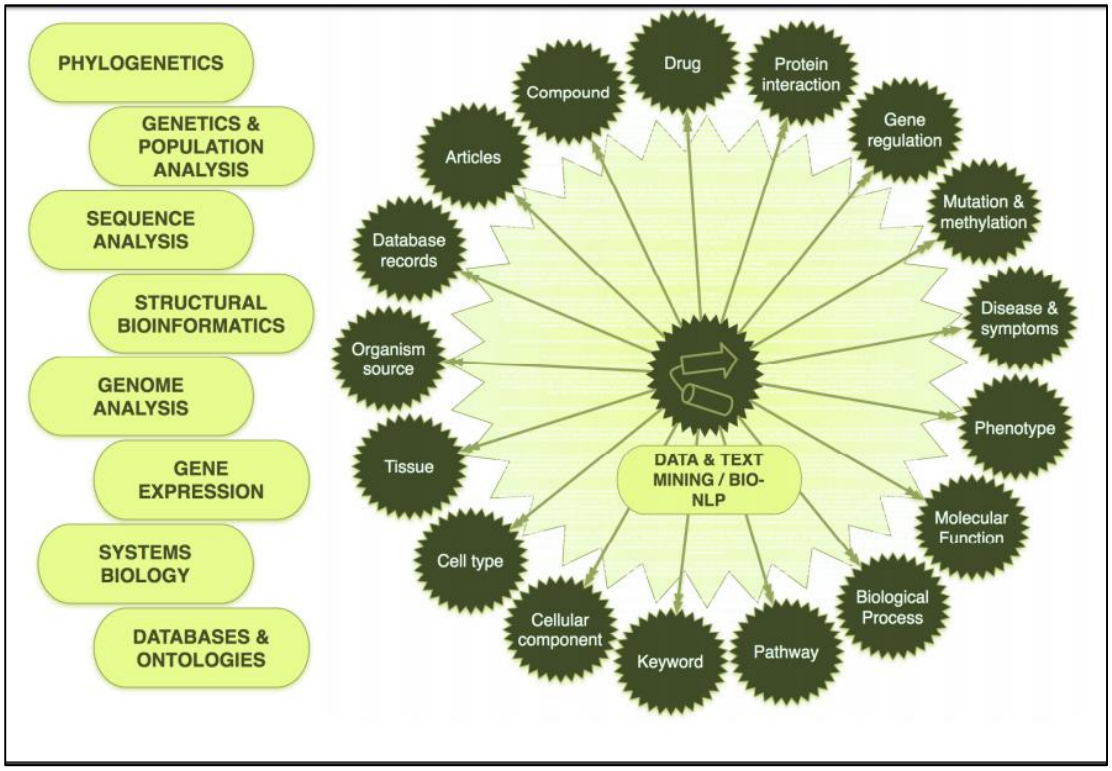


Fig. 2: Application areas of Omics domain

Sources for Literature

There are large number of resources are available to access various datasets from large number of resources. The data types are diversified and need to be processed and integrated. The data is varied from literature to the experimental processed data. The data are accessible from PubMed/Entrez, Medline, KDD Challenge, Google Scholar, MedMiner, CrossRef Search etc.

- Life sciences -> generates heterogeneous data types (sequence, structure and others)
- Natural language used for communicating scientific discoveries.
- Natural language texts amenable for direct human interpretation
- Natural language not only in scientific articles, but also patents, reports, newswire, database records, controlled vocabularies (GO terms)
- Functional information & annotations directly or indirectly derived from the literature (curation and electronic annotation).
- Databases are generally only capable of covering a small fraction of the biological context information that can be encountered in the literature.
- Contextual information of experimental results (cell line, tissue, conditions).
- User demands of better information access (beyond keyword searches)
- Rapid growth of information, manual information extraction not efficient.
- Integration of information from full text articles, databases and genomic studies

The process of text analytics and curation process is summarized in fig. 3. The process starts with manual curation, extraction of features, annotation and prediction with revisions using biological annotation.

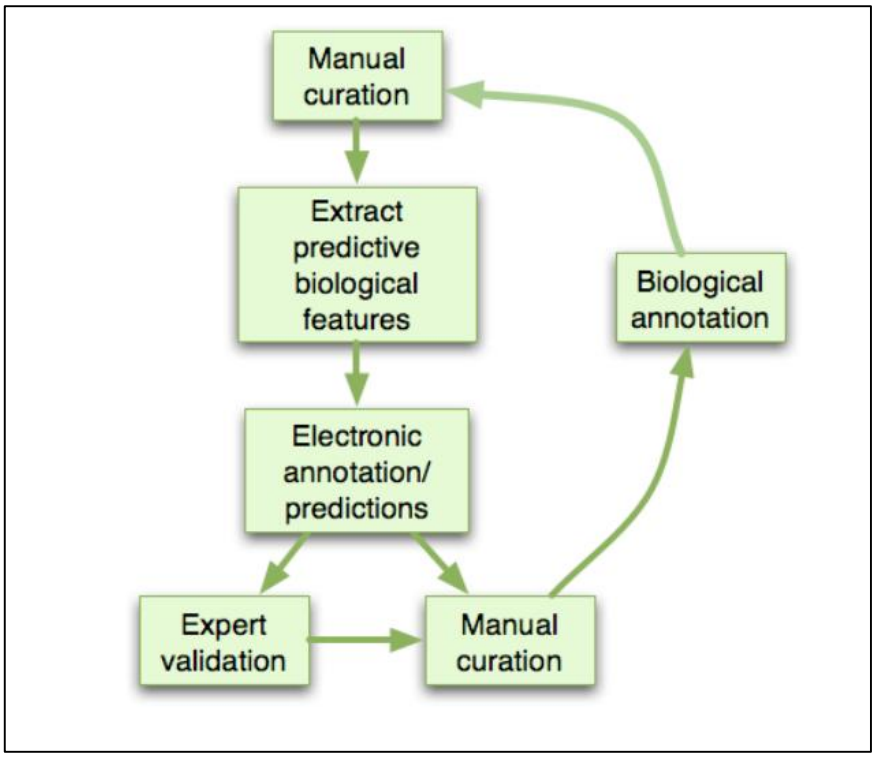


Fig. 3: Process of Text Analytics and Curation

Challenges in Text Mining

There are many challenges are being faced by the text mining community. Rapid and continuous updation of information, unstructured text data, use of more than one language in literature/documents, handling of linguistic text are making the mining task difficult to infer any new knowledge and patterns. Few of the important challenges are

- Information is in unstructured textual form.
- Not readily accessible to be used by computers.
- Dealing with huge collections of documents
- Problems in handling of synonyms/Antonyms/PoS tagging etc.
- Handling of dictionary and identification of primary term
- Other coding or language characters

Tools for Text Mining

Large number of tools are available to do the text mining. The tools are categorized in open source tools and commercial tools as follows

The Open Source category of tool contains Carrot2 used for text and search results clustering framework, GATE (General Architecture for Text Engineering) an open-source toolbox for natural language processing and language engineering, Python based Gensim for large-scale

topic modelling and extraction of semantic information from unstructured text, OpenNLP for natural language processing, Natural Language Toolkit (NLTK) a suite of libraries and programs for symbolic and statistical natural language processing (NLP), Orange having text mining add-on, R provides a framework for text mining applications in the package tm, KNIME as Text Processing extension, PLOS Text Mining Collection etc.

The Closed source or commercial category of tools comprises with IBM SPSS - provider of Modeler (previously called IBM SPSS Modeler and IBM SPSS Text Analytics) contains advanced NLP-based text analysis capabilities (multi-lingual sentiment, event and fact extraction), Mathematica provides built in tools for text alignment, pattern matching, clustering and semantic analysis, RapidMiner with its Text Processing Extension for data and text mining, SAS Text Miner and Teragram for commercial text analytics, natural language processing, and taxonomy software used for Information Management, STATISTICA Text Miner as an optional extension to STATISTICA Data Miner for Predictive Analytics Solutions, Megaputer Intelligence derives actionable knowledge from large volumes of text and structured data including natural language processing (NLP), machine learning, sentiment analysis, entity extraction, clustering, and categorization. The enterprise feedback and text analytics solutions developed as Luminoso for natural language processing (NLP), machine learning and artificial intelligence research at MIT Media Lab. These tools enables the end user in learning, understanding, measuring and acting on large number of reports, literatures, documents, feedback and many more to infer the novel knowledge and innovative ideas.

Conclusion

The text mining/analytics is a very important topic in the current day of research in every domain varied from literature mining to source code analysis. The availability of digital contents in variety of platforms with the help of open source or commercial tools make somewhat easy to process these large set of documents and reports. The spam filtering, biological literature mining, information retrieval has been improved significantly in last few years. The text mining area has a large number of application areas in Agriculture as well.

NGS Data Generation: Techniques, Issues and Challenges

Dwijesh Chandra Mishra

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:

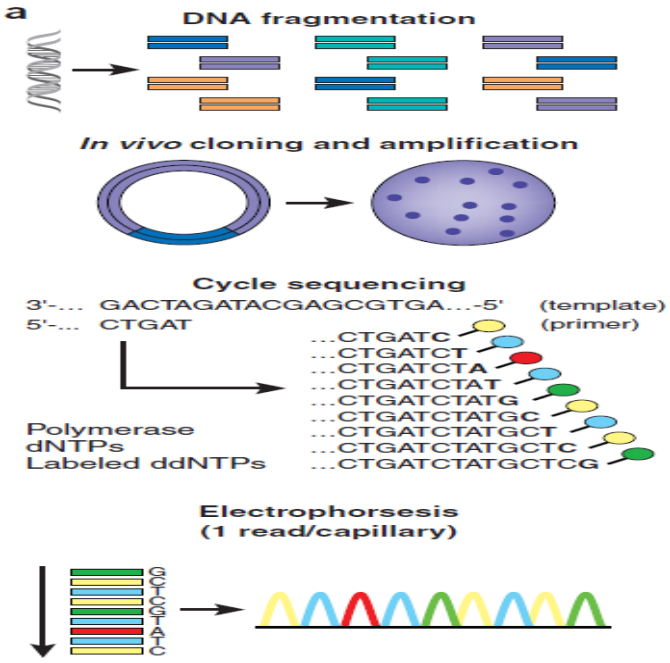
- Kchouk M, Gibrat JF, Elloumi M. (2017). Generations of Sequencing Technologies: From First to Next Generation. Biol Med (Aligarh) 9: 395. doi:10.4172/0974-8369.1000395.
- H.P.J. Buermans, J.T. den Dunnen. (2014). Next generation sequencing technology: Advances and applications, Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease, 1842(10): 1932-1941. https://doi.org/10.1016/j.bbadis.2014.06.015.

NGS Data: Pre-processing, Assembly and Quantification

Dwijesh Chandra Mishra, Sanjeev Kumar and Neeraj Budhlakoti

Sanger Sequencing

- DNA is fragmented
- Cloned to a plasmid vector
- Cyclic sequencing reaction
- Separation by electrophoresis
- Readout with fluorescent tags



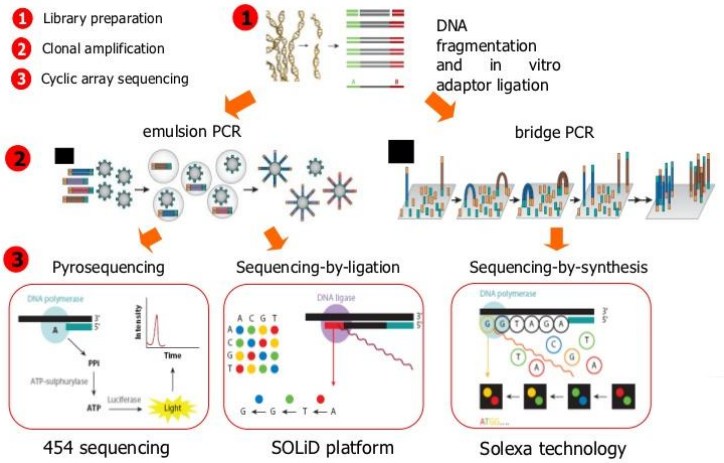
Sanger Vs NGS

- '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™ System: 2007
- Helicos Heliscope™ : 2009
- Pacific Biosciencies SMRT: 2010

General Experimental Procedure

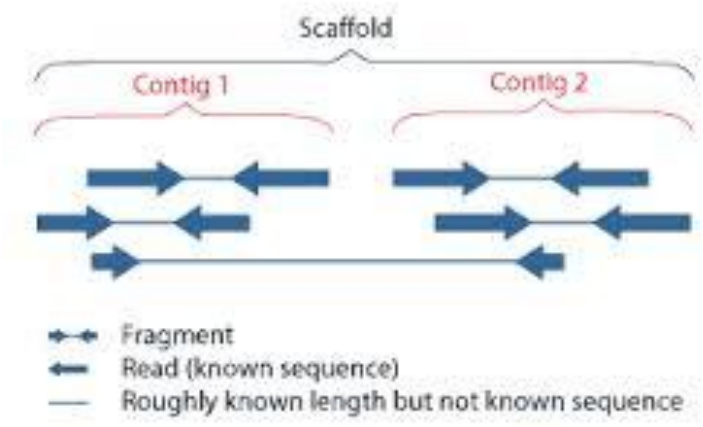


Sequencing Technology at a Glance

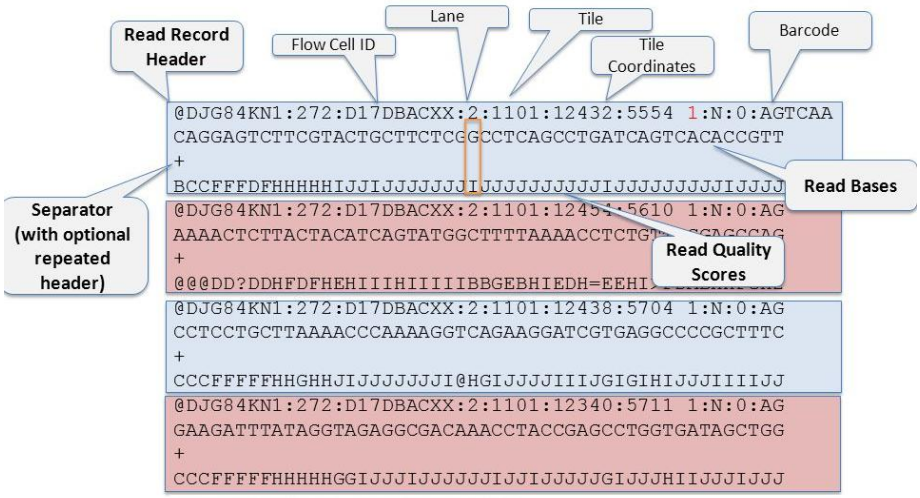
Method	Read length	Accuracy	Time per run	Cost per 1 million bases	Advantages	Disadvantages
Chain termination (Sanger sequencing)	400 to 900 bp	99.9%	20 minutes to 3 hours	Rs 144000	Long individual reads. Useful for many applications.	More expensive and impractical for larger sequencing projects.
Pyrosequencing (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



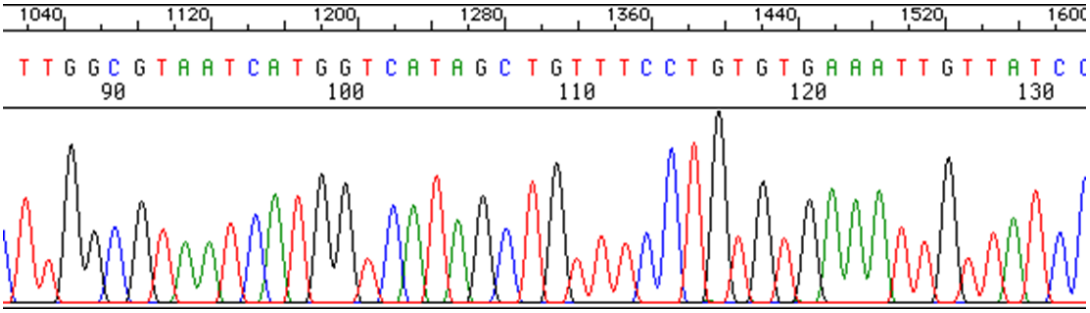
NOTE: for paired-end runs, there is a second file with one-to-one corresponding headers and reads.

(Passarelli, 2012)

Before Assembly

Fragment readout

- DNA characters in a fragment are determined from chromatogram
- Base call is a DNA character that is determined from chromatogram



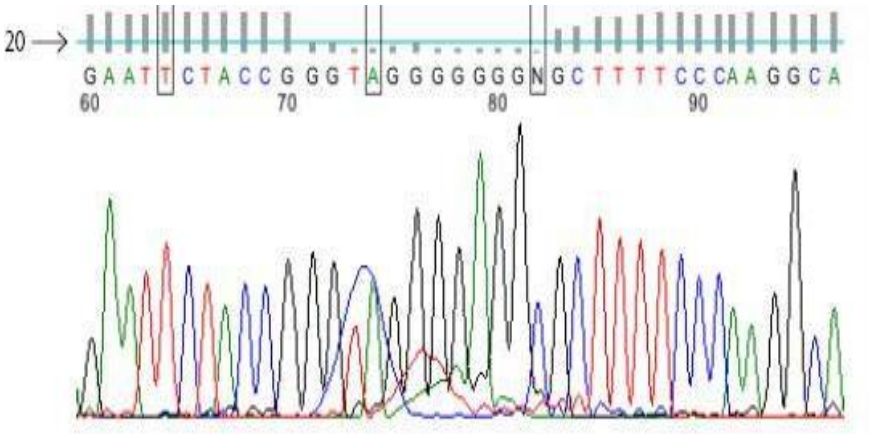
Fragment readout

- Phred Score- determine the quality value of a base

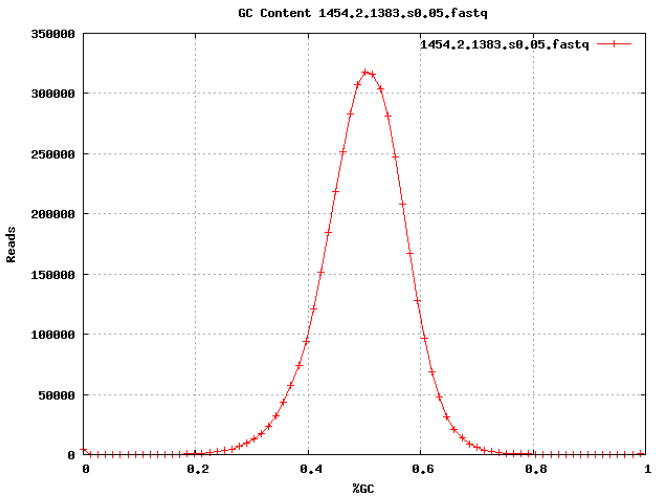
$$q = -10 \times log_{10}(\mathrm{p})$$

where p is the estimated error probability for the base

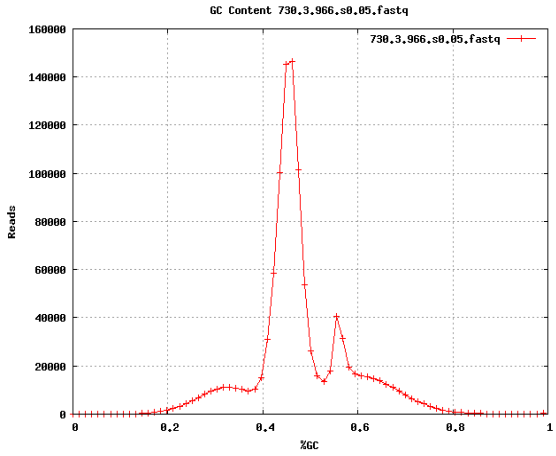
- if Phred assigns a quality score of 30 to a base, the chances that this base is called incorrectly are 1 in 1000
- The most commonly used method is to count the bases with a quality score of 20 and above
- Phred Score



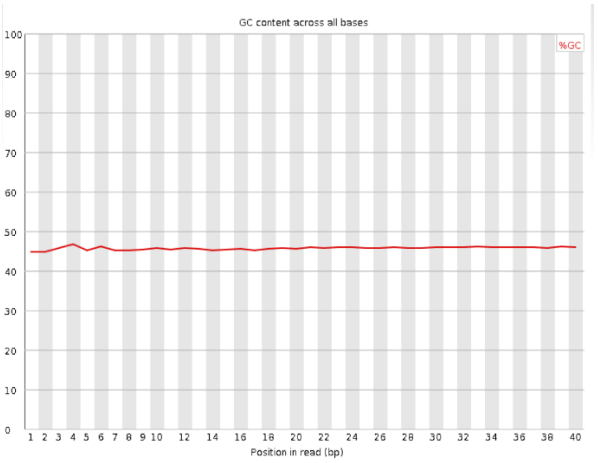
Genome Properties



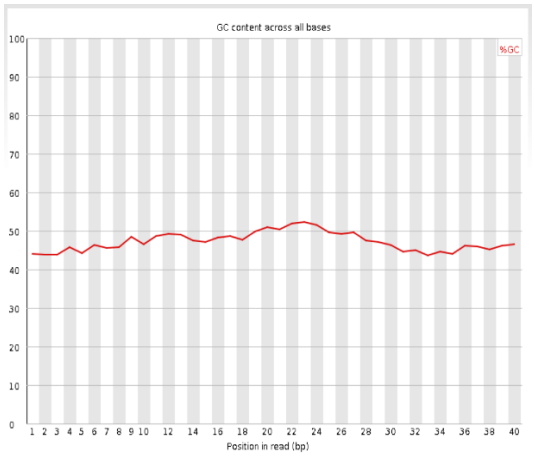
PASS



FAIL

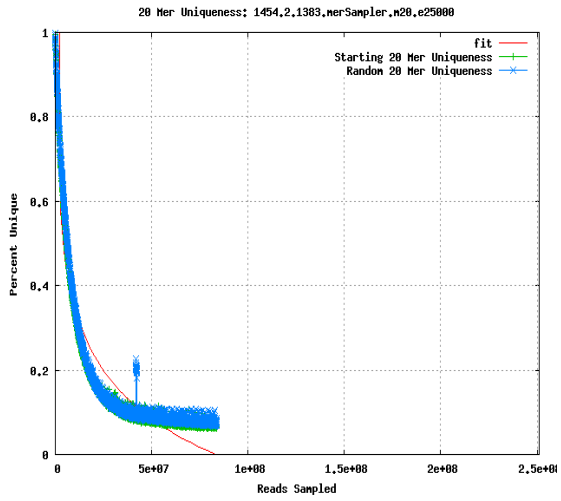


PASS

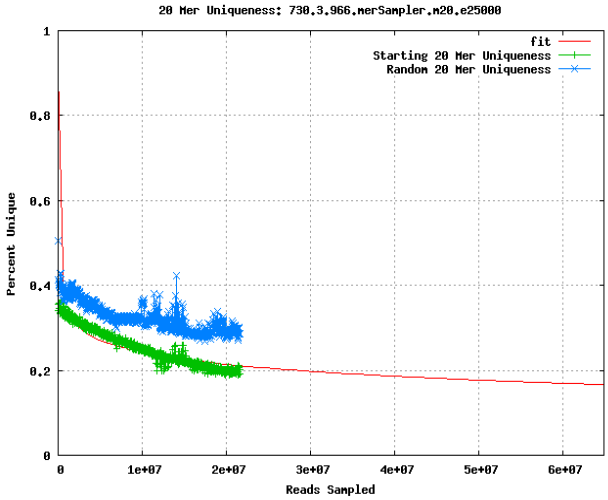


FAIL

Library Quality

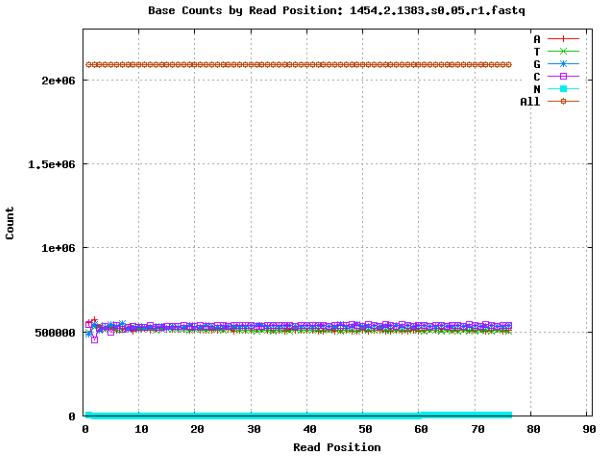


PASS

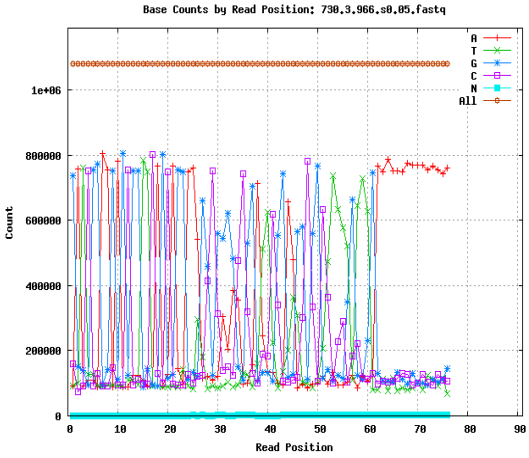


FAIL

Run Quality

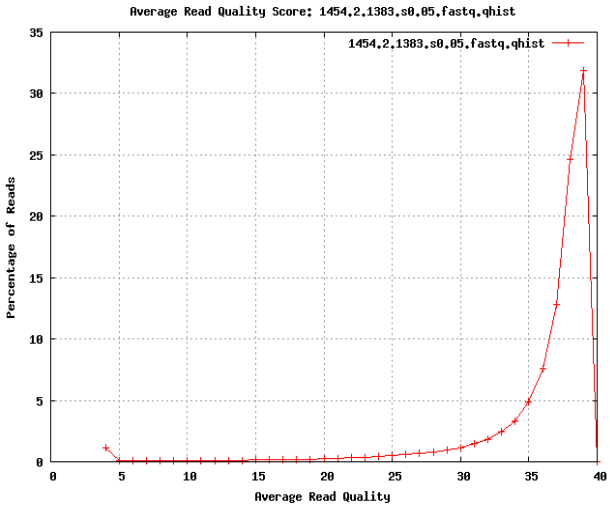


PASS

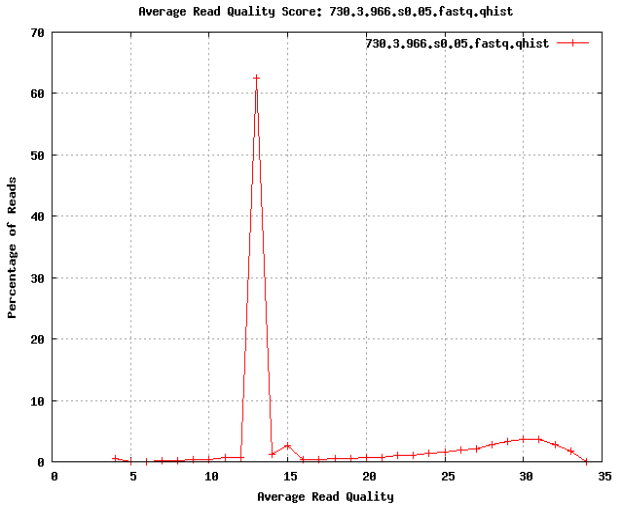


FAIL

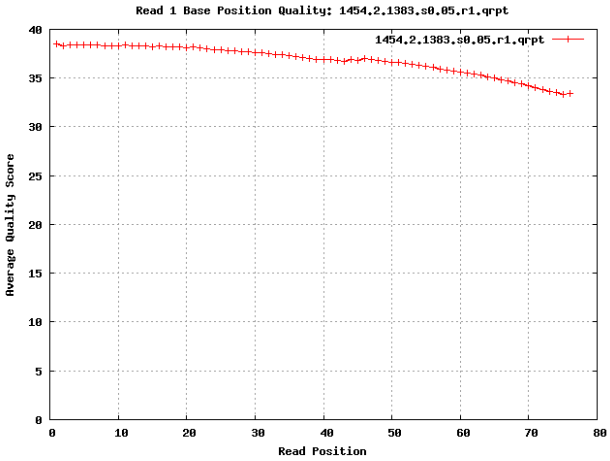
Read Quality



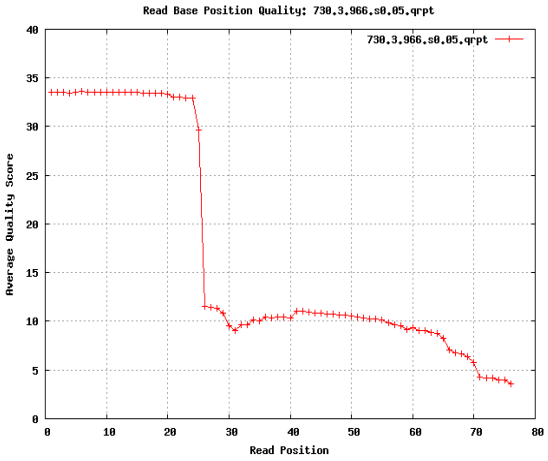
PASS



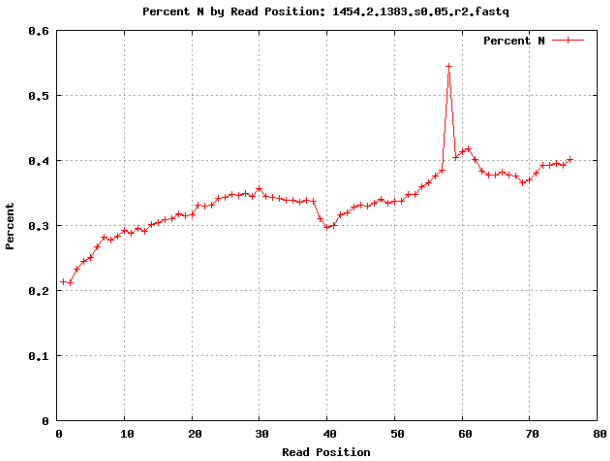
FAIL



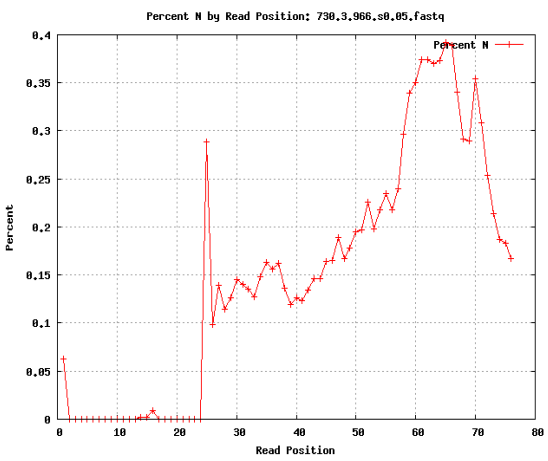
PASS



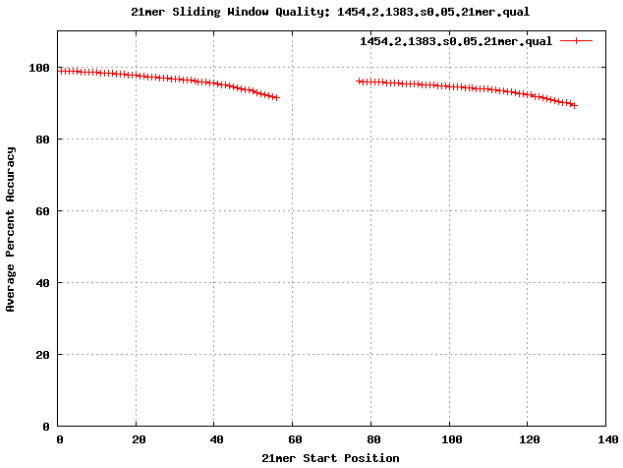
FAIL



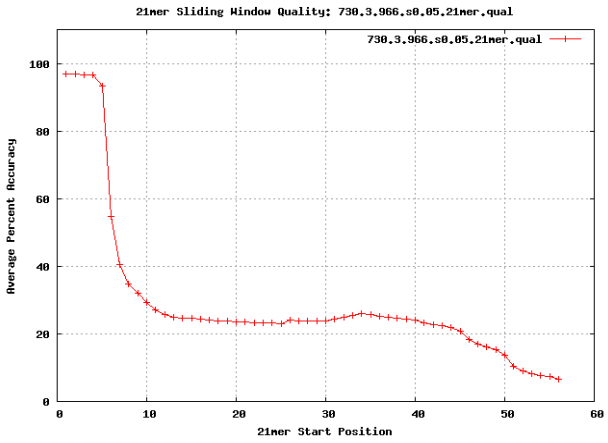
PASS



FAIL



PASS



FAIL

Trimming

- Trimming low-quality sequences

  -removal of reads containing poor quality base calls

- Trimming vector sequences

  -removal of reads containing vector sequences

Annotation of RNA-Seq Data

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

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

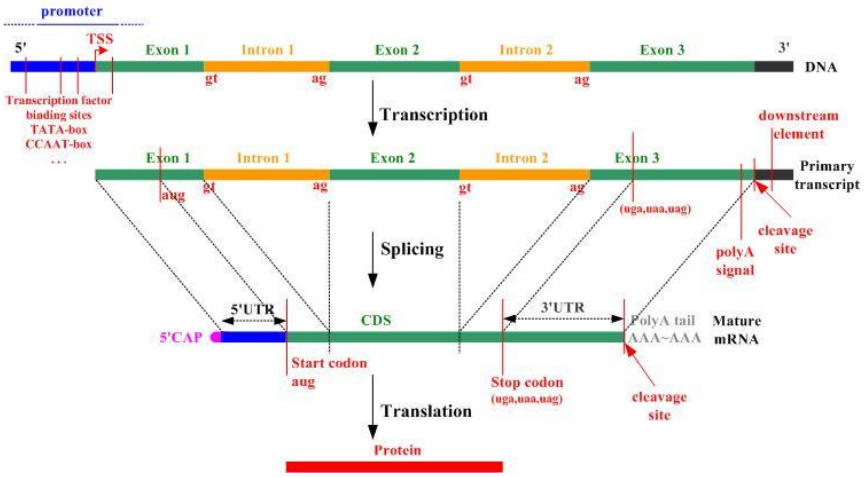


Fig. 1: Representative Diagram of Protein Coding Eukaryotic Gene

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

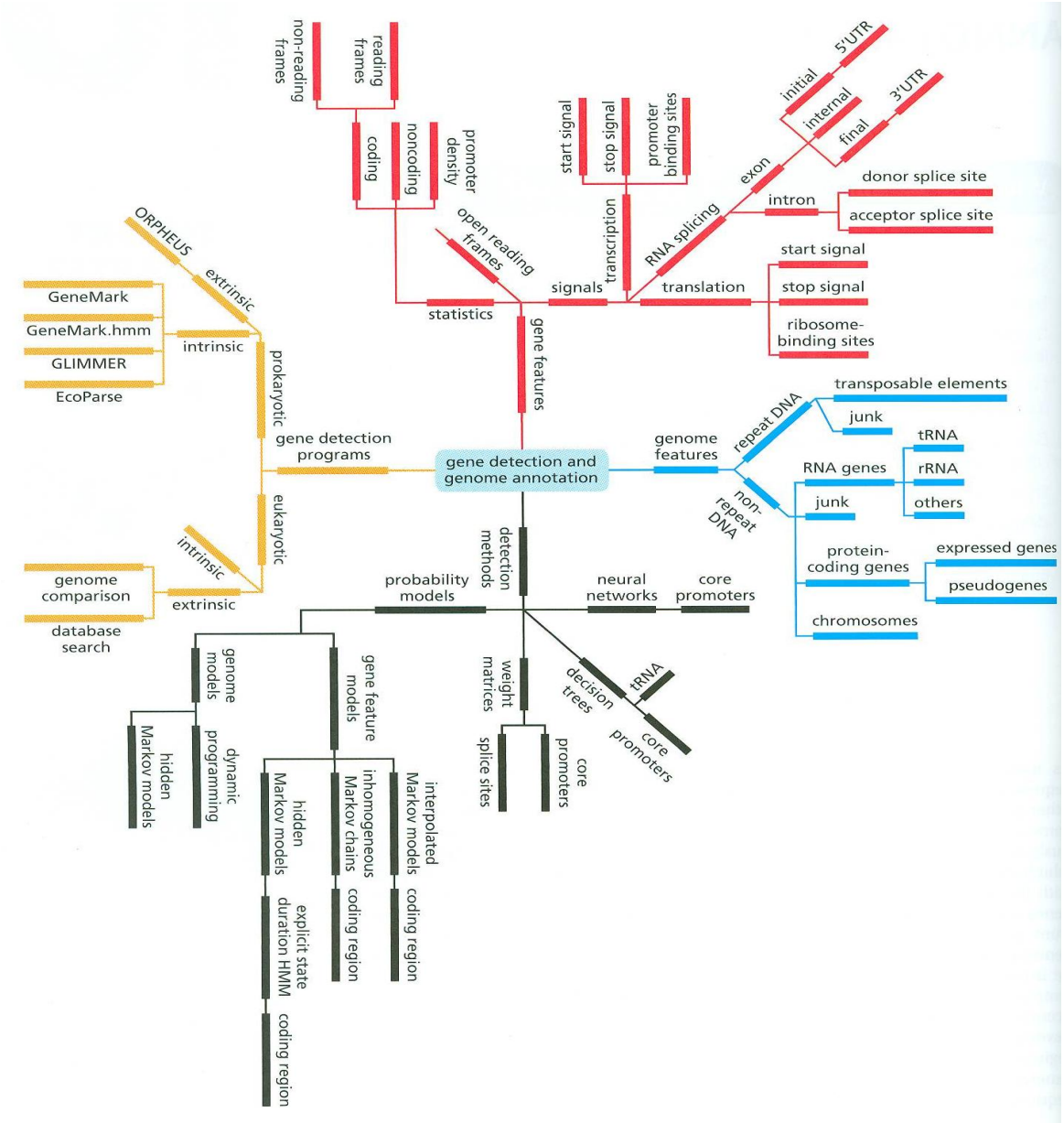
Sequence similarity search is a conceptually simple approach that is based on finding similarity in gene sequences between ESTs (expressed sequence tags), proteins, or other genomes to the input genome. This approach is based on the assumption that functional regions (exons) are more conserved evolutionarily 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,



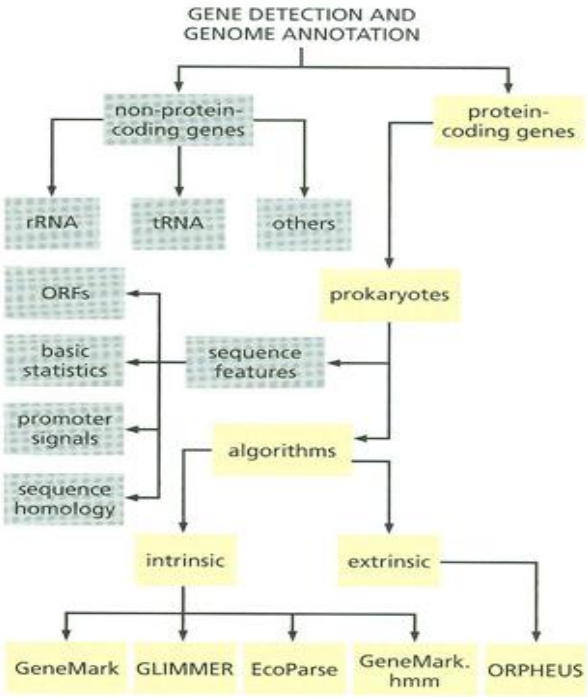
linear discriminant analysis, Linguist methods, Hidden Markov Model and Neural Network.

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).

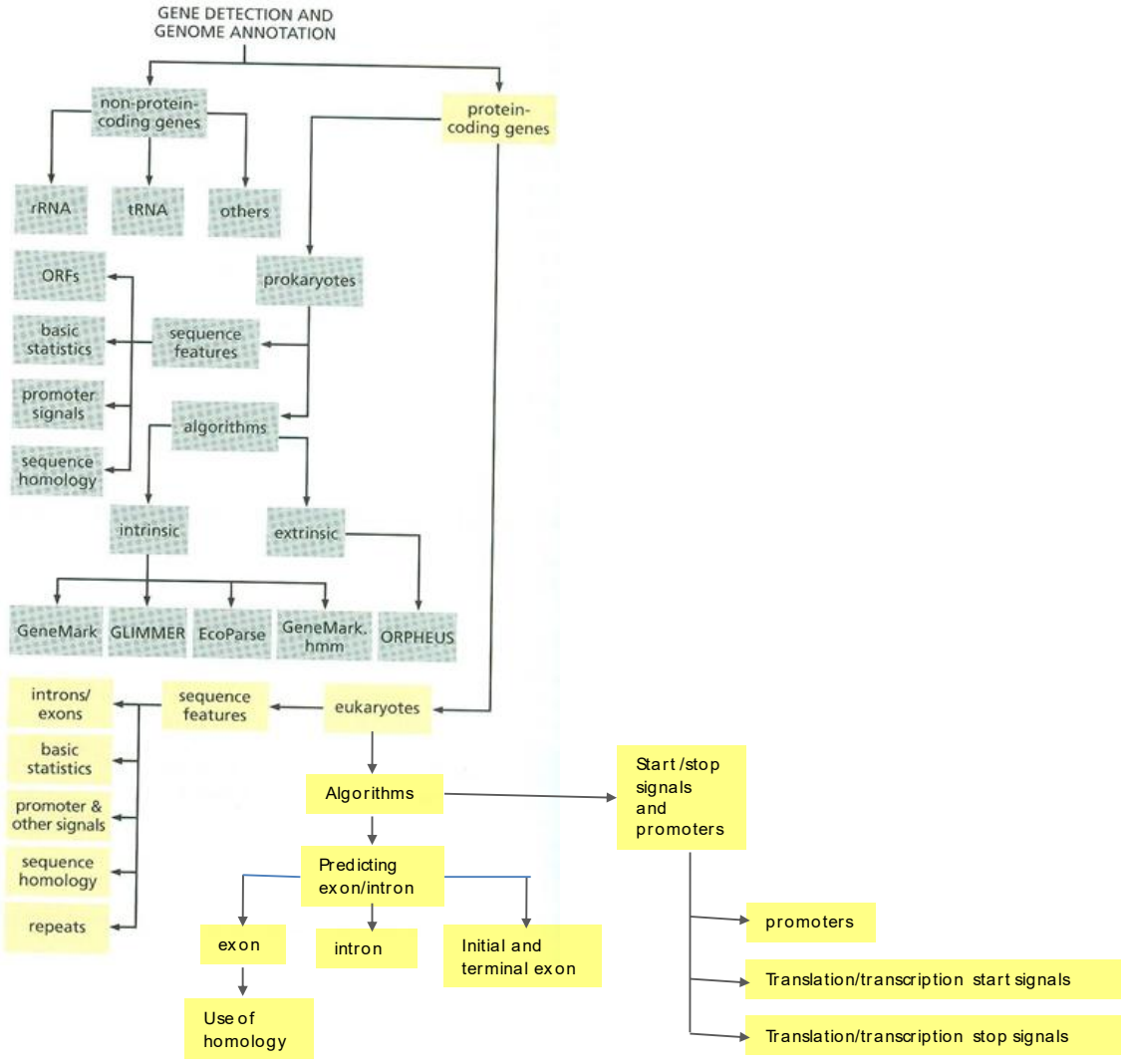


Fig. 4: Flow Diagram of Eukaryotic Gene Discovery

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 0^{th} order to 8^{th}. 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)

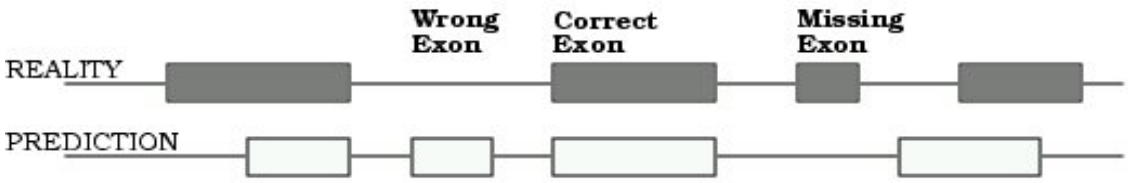


Fig. 6: Real and Predicted Exons

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.

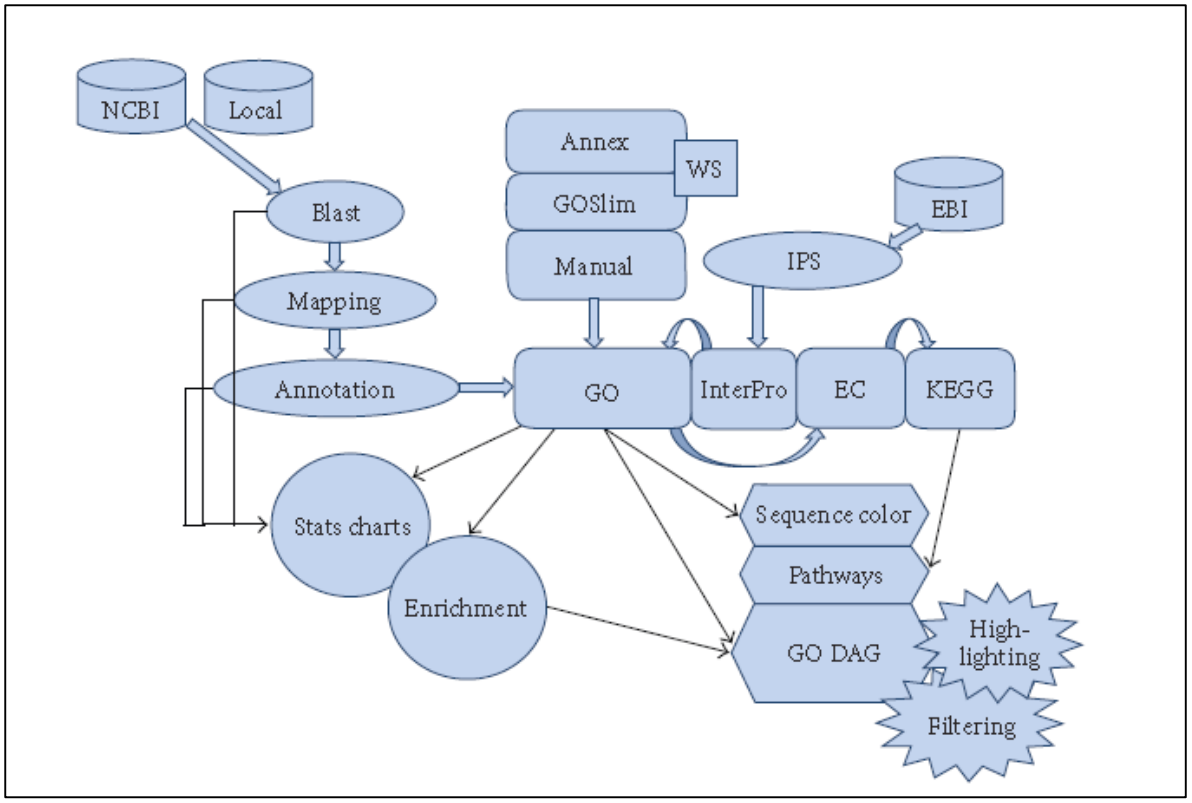


Fig. 7: Schematic Representation of Blast2GO Application.

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.

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

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.

Annotation of RNA-Seq Data (Practical)

Sneha Murmu

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.

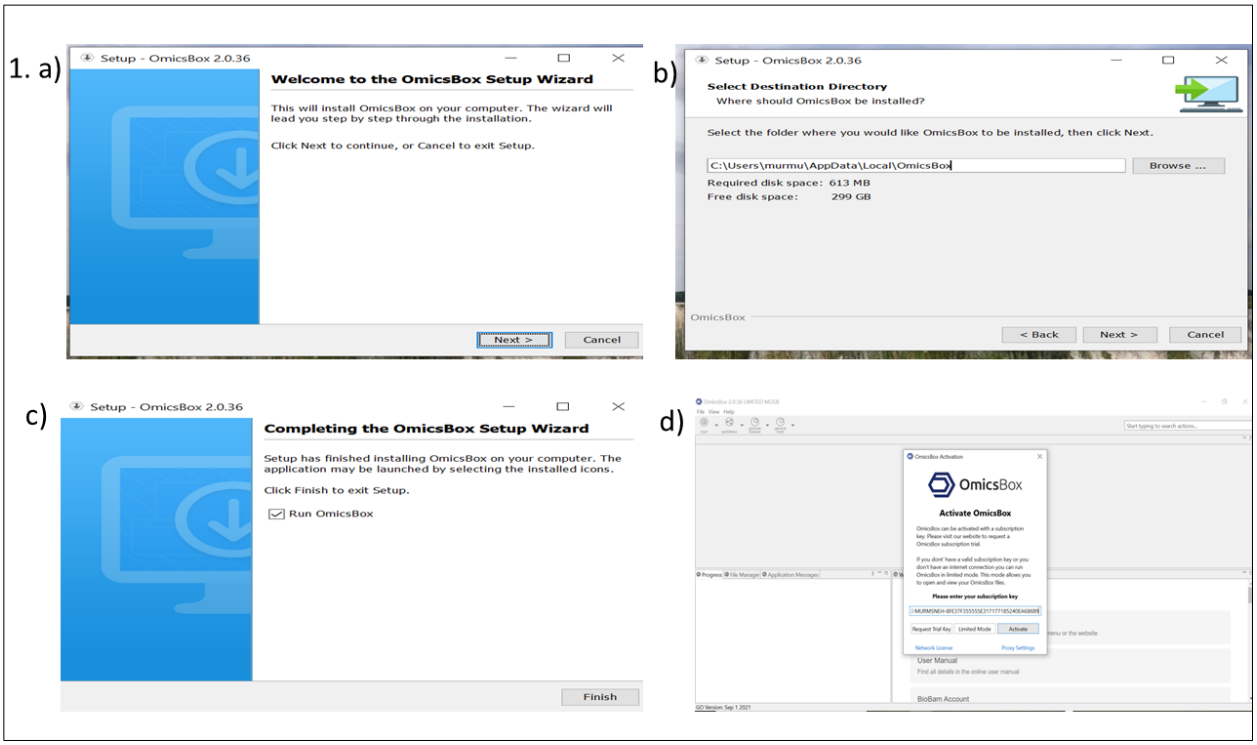


Figure 1: Installation steps of Blast2GO in Windows system.

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.

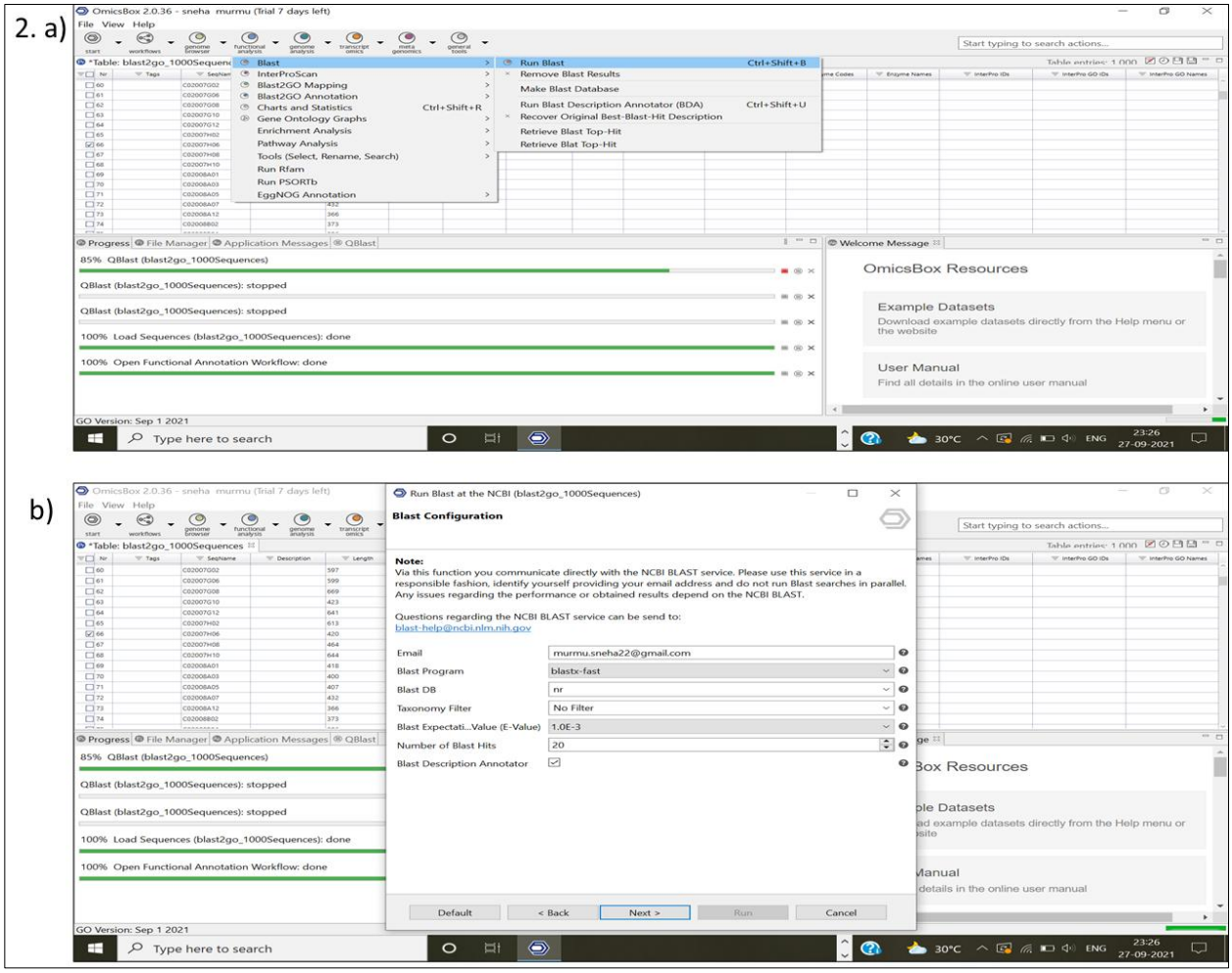


Figure 2: BLAST search.

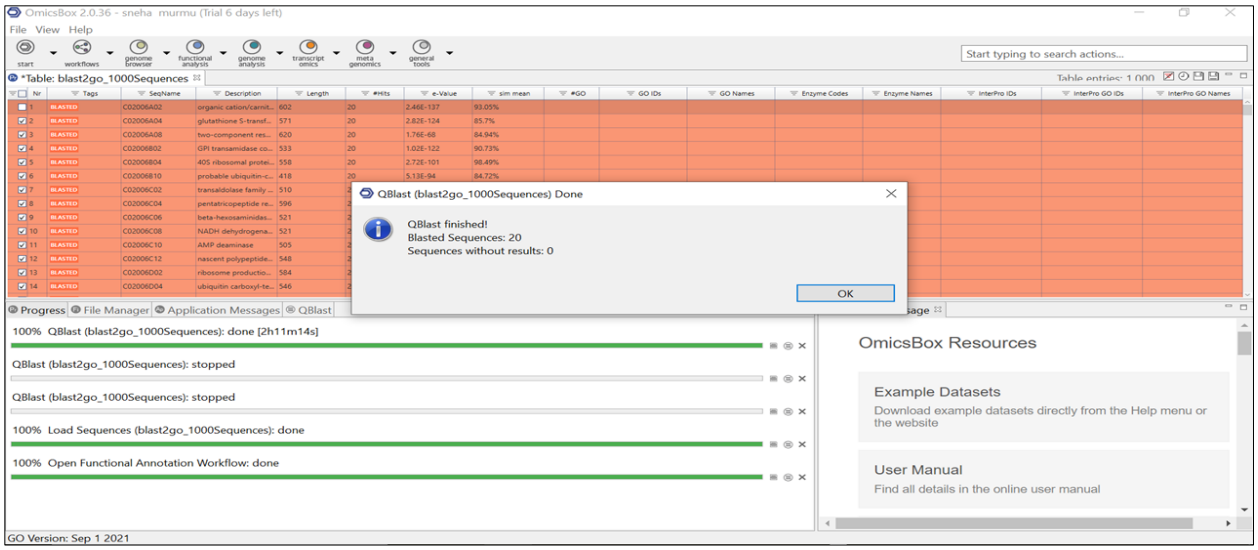


Figure 3: BLAST result.

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).

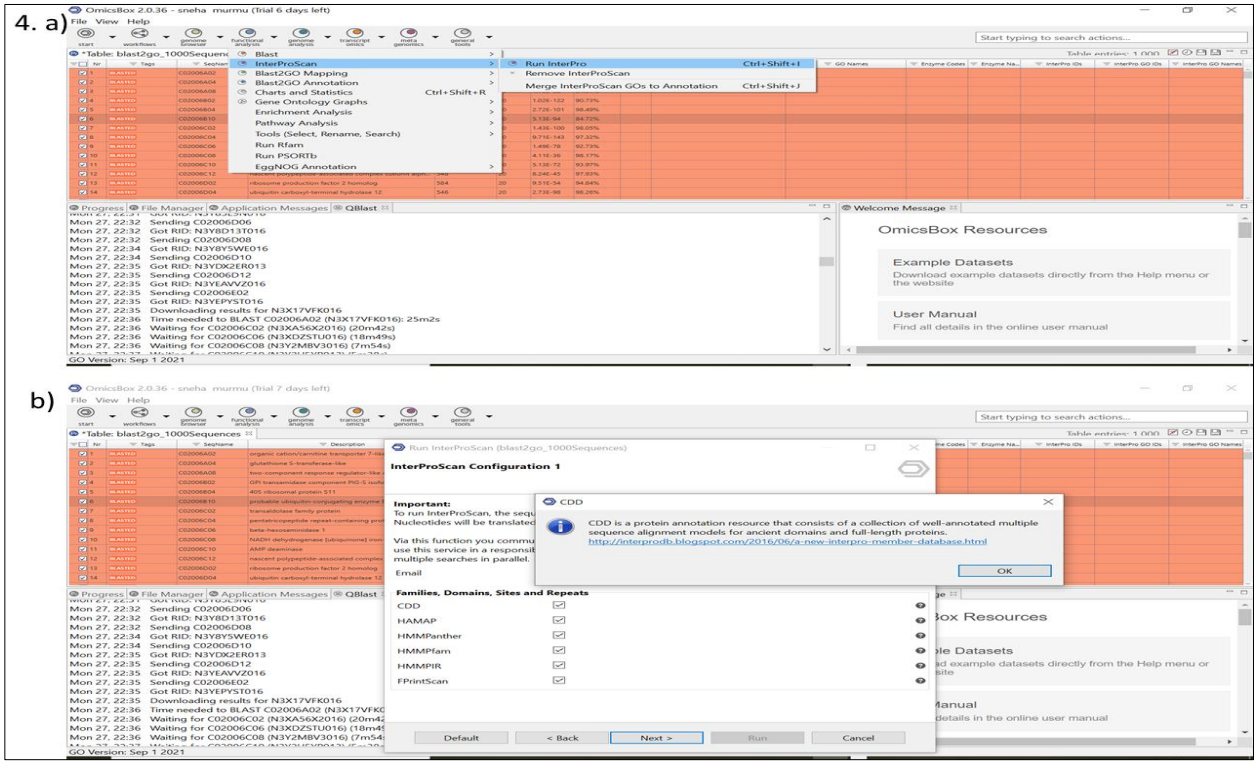


Figure 4: InterProScan search.

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.

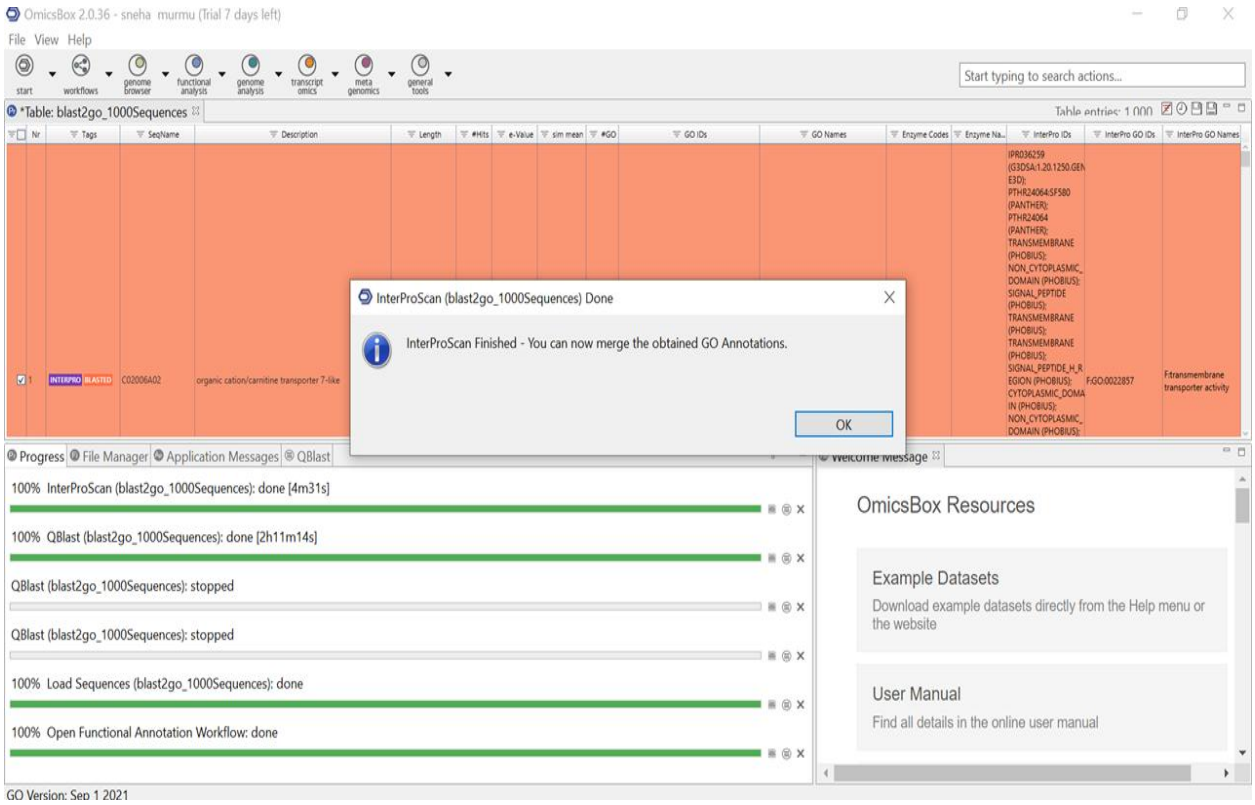


Figure 5: InterProScan result.

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.

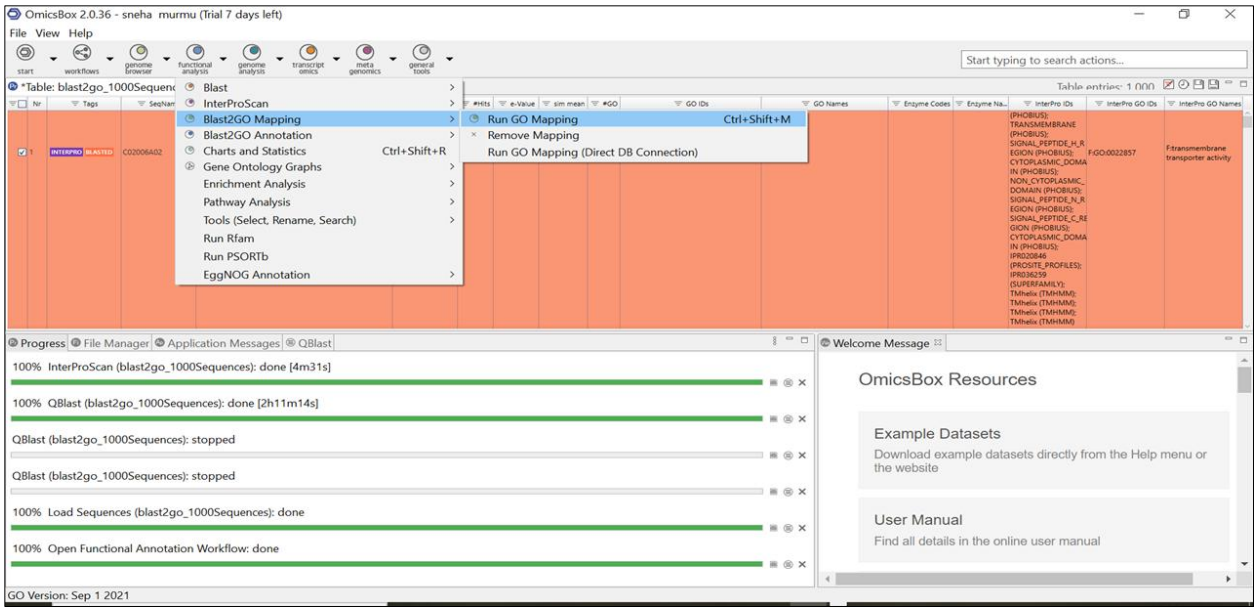


Figure 6: Mapping.

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.

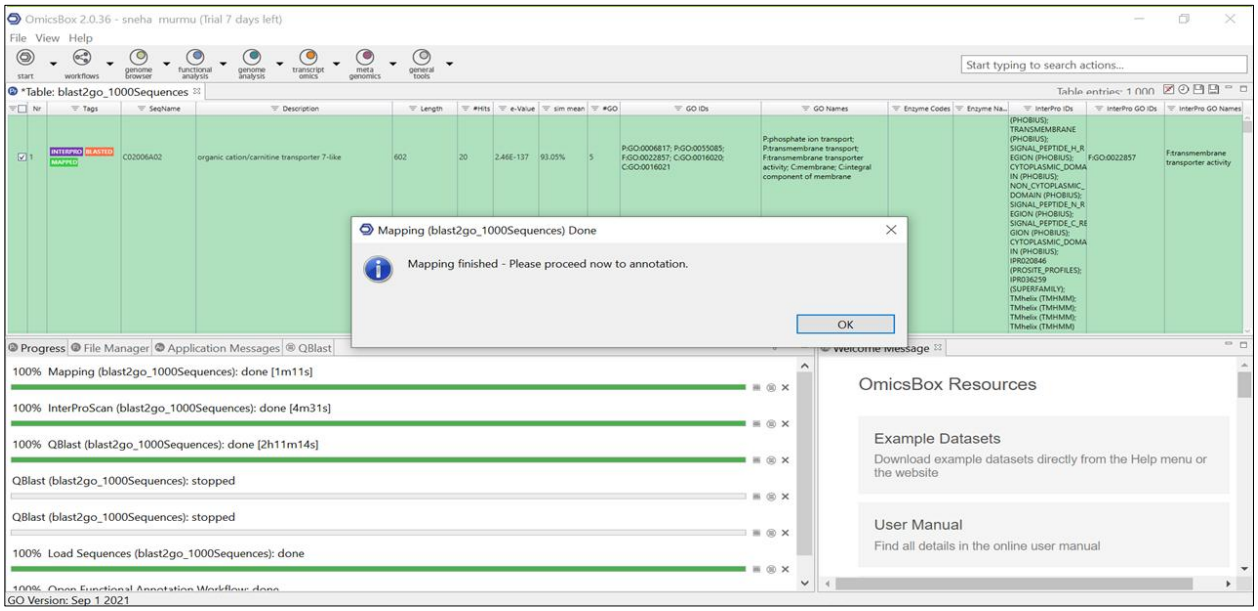


Figure 7: Mapping result.

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.

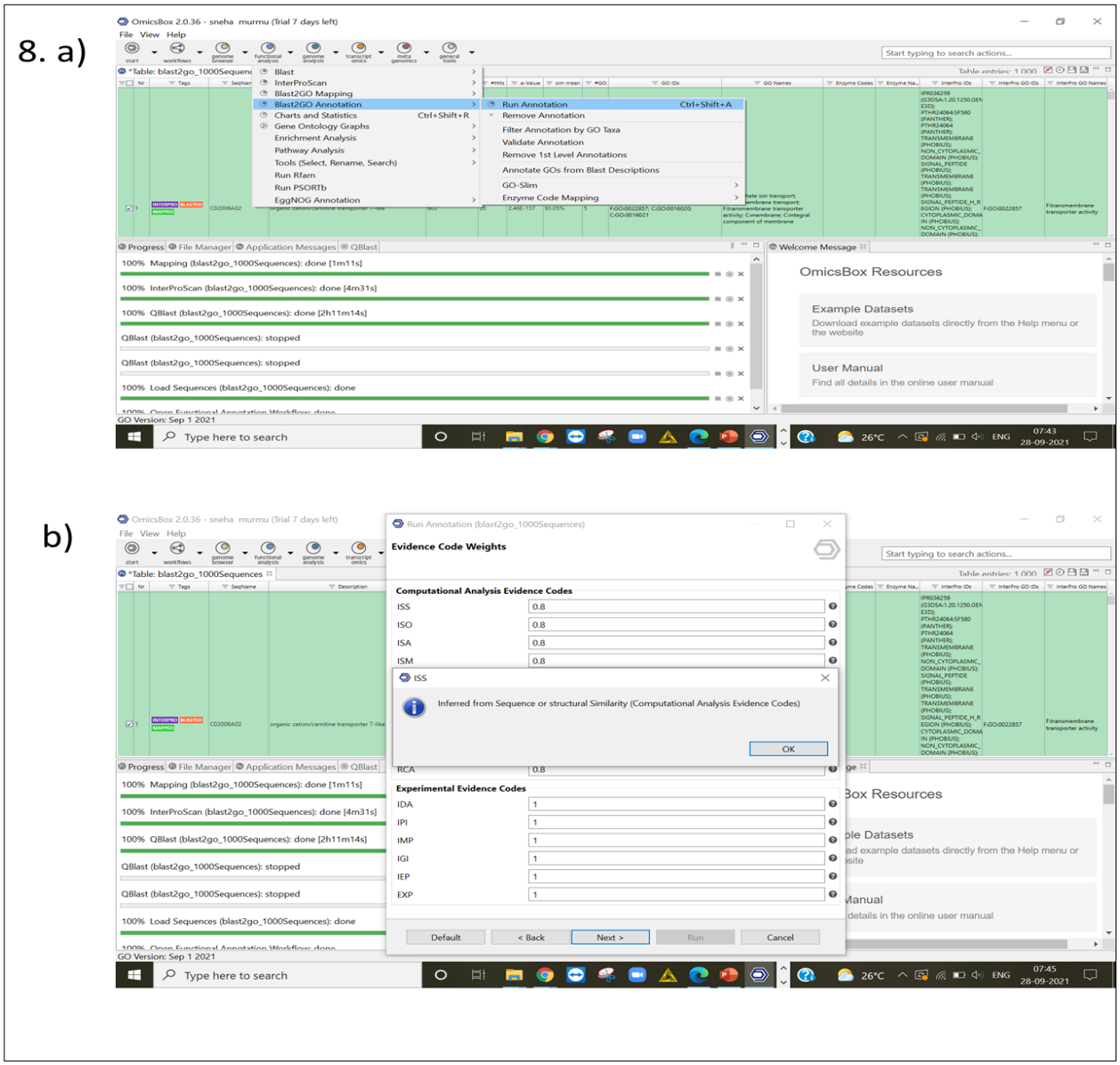


Figure 8: Annotate.

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.

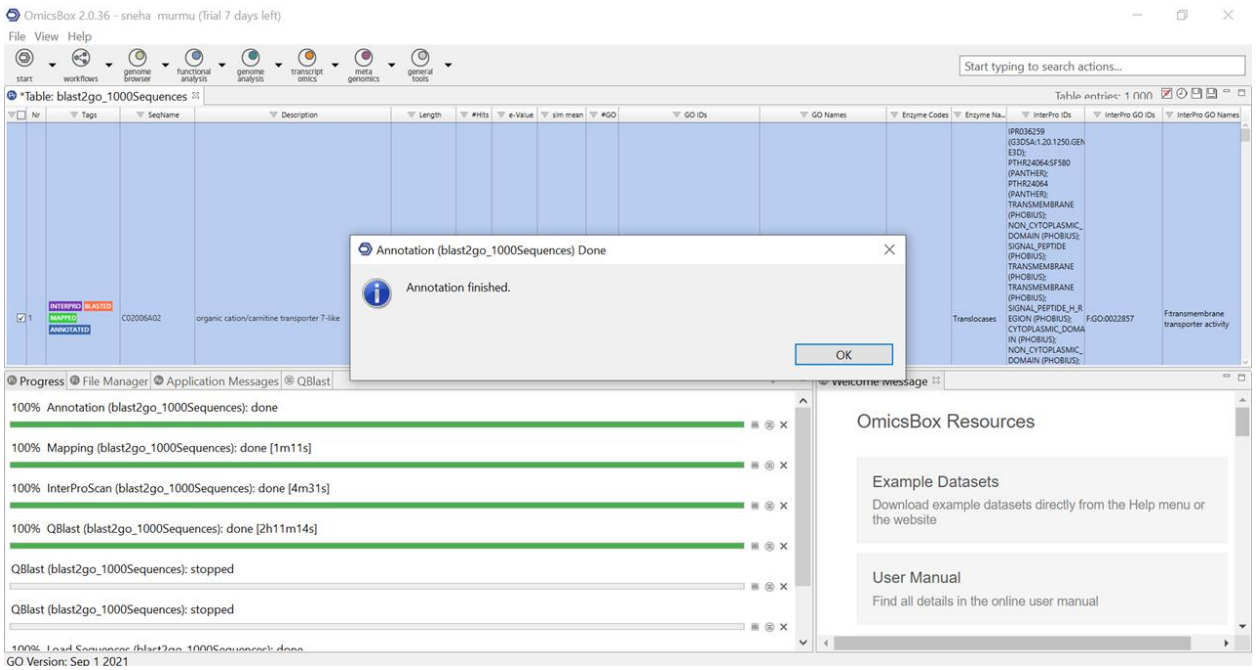


Figure 9: Annotate result.

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.

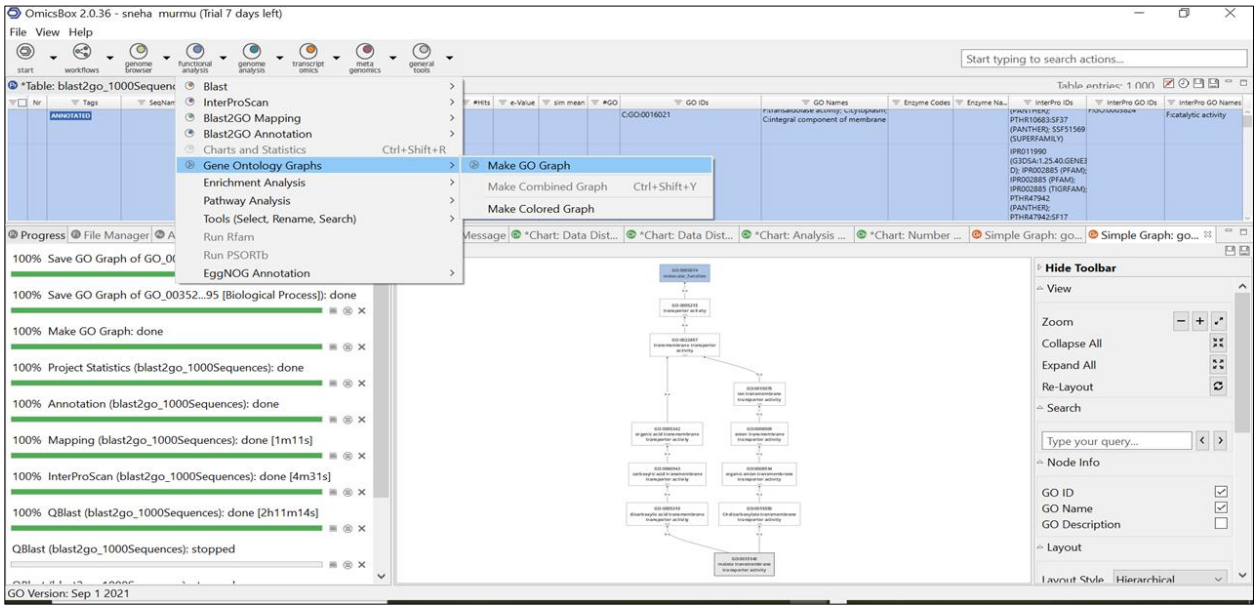


Figure 10. Generate GO graph.

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).

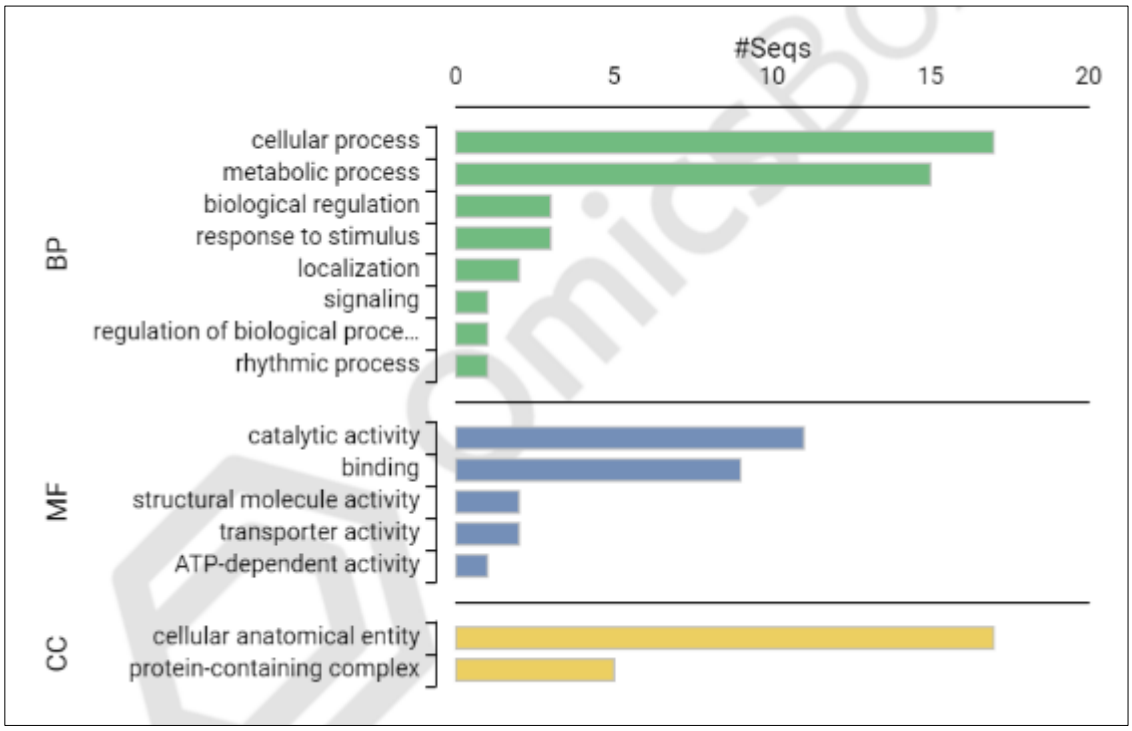


Figure 11: GO graph.

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).

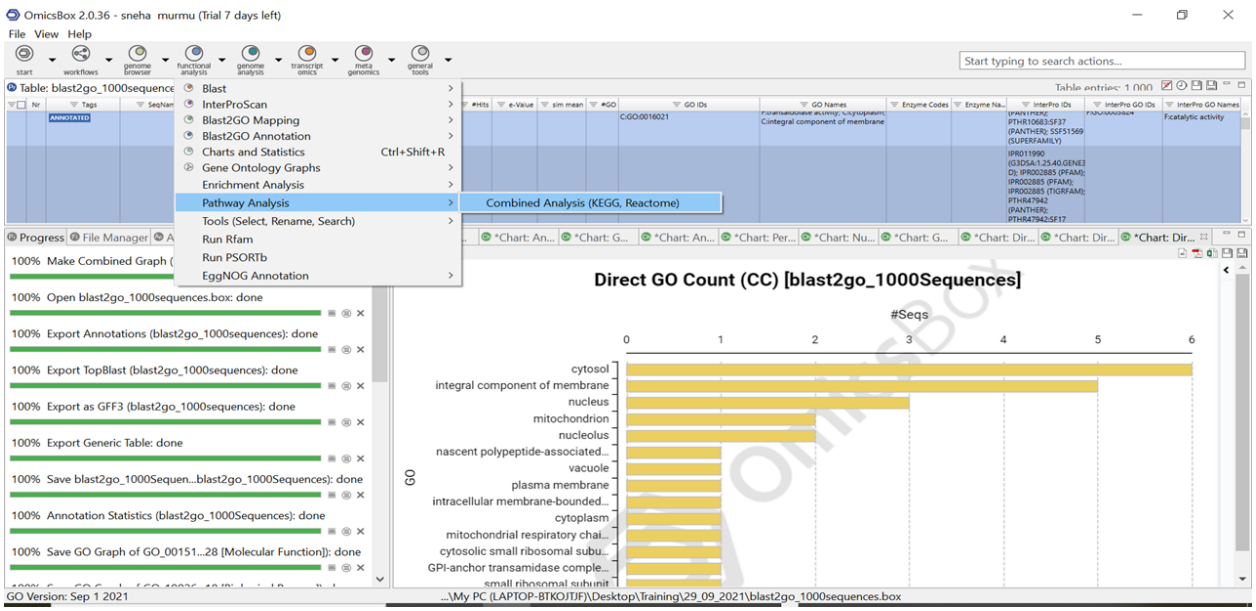


Figure 12. Run Pathway Analysis.

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).

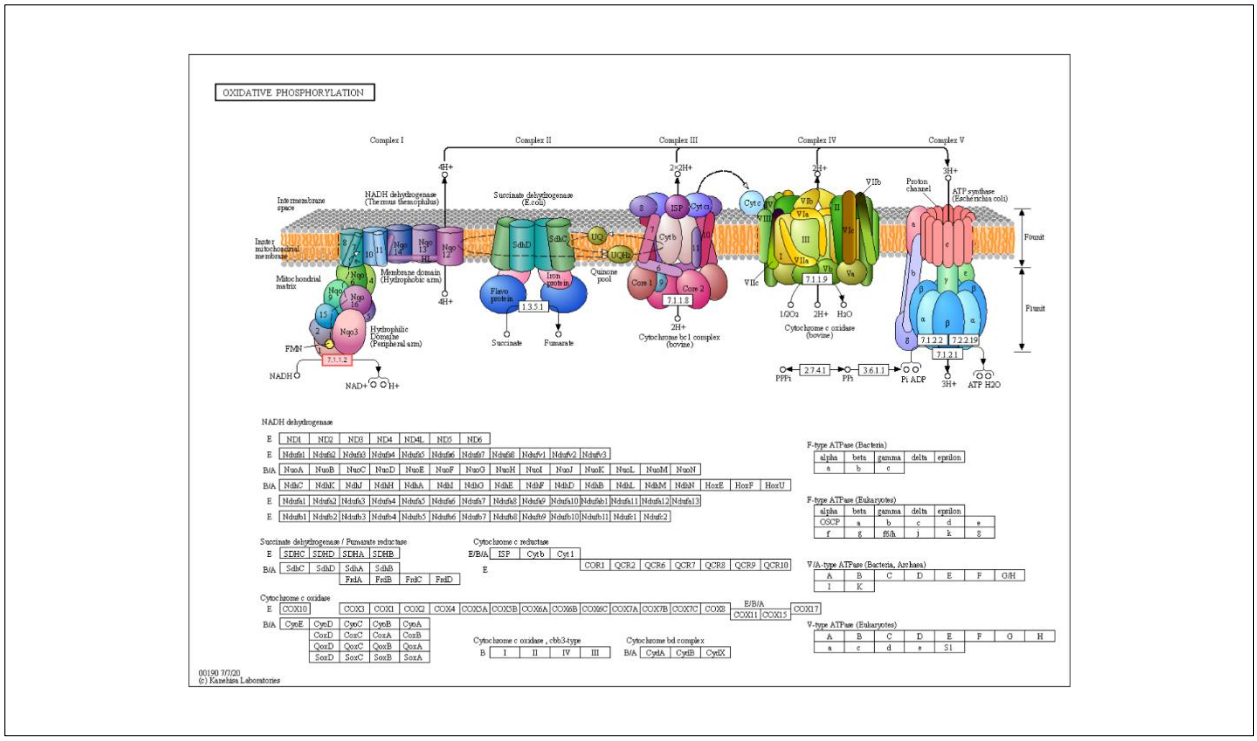


Figure 13. Pathway graph.

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.

Overview of RNA-Seq Data Analysis

Mohammad Samir Farooqi and Sudhir Srivastava

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.

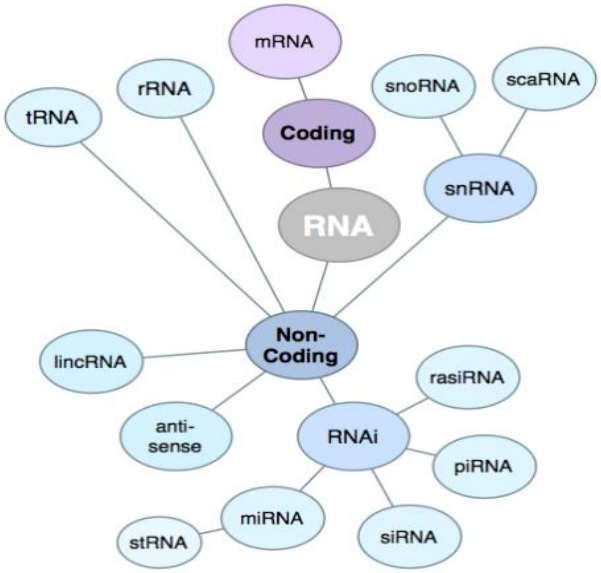


Fig. 1: Different types of RNA

(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).**

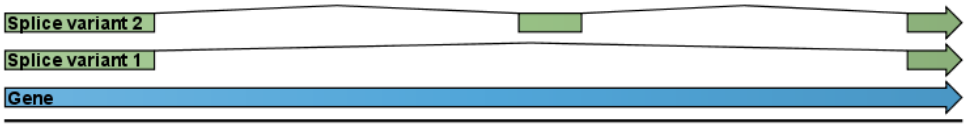


Fig. 2(a): A simple gene with three exons and two splice variants.

The given example is a simple gene with three exons and two splice variants. The transcripts are extracted as shown in **Fig. 2(b).**

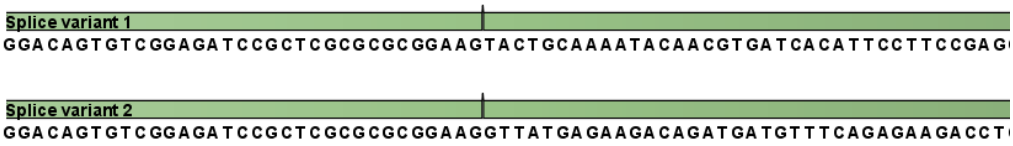


Fig. 2(b): All the exon-exon junctions are joined in the extracted transcript.

Next, the reads are mapped against all the transcripts plus the entire gene [see **Fig. 2(c)**].

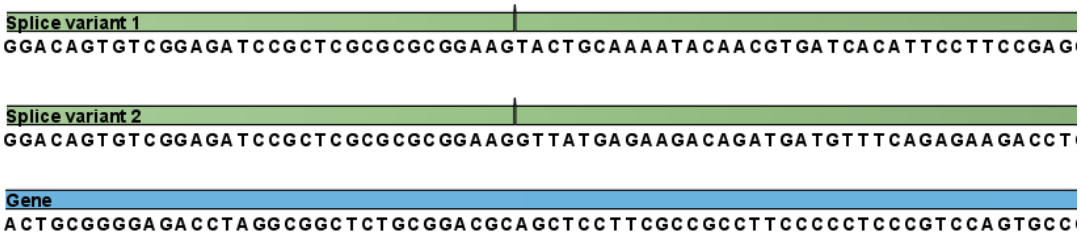


Fig. 2(c): The reference for mapping: all the exon-exon junctions and the gene.

(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.

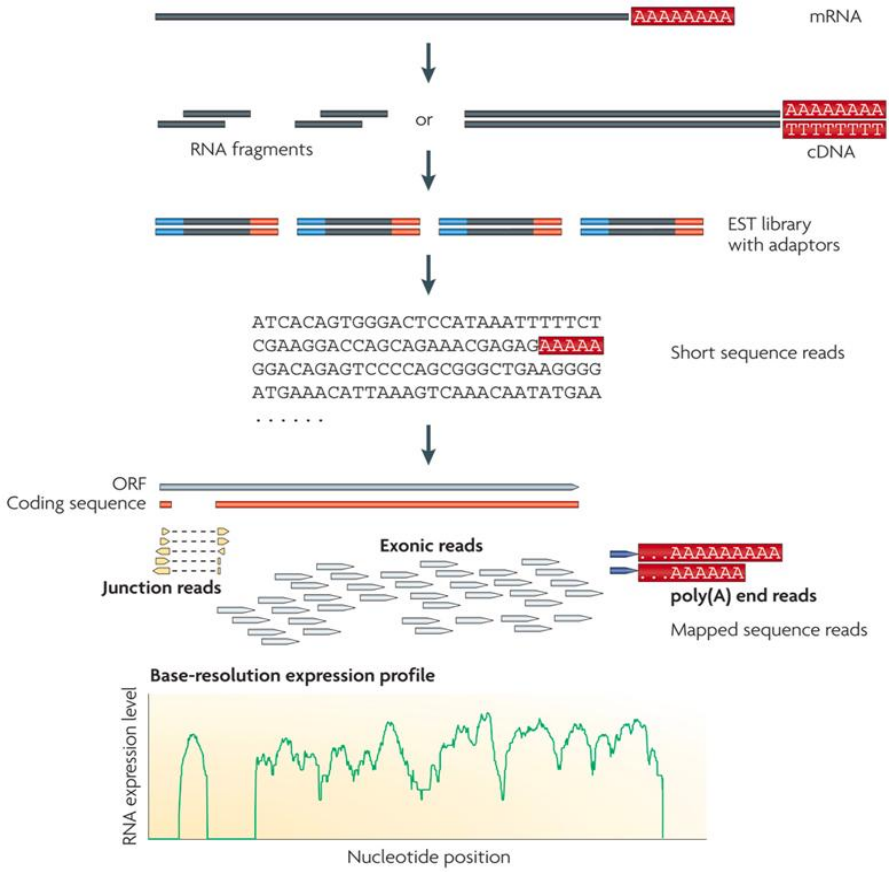


Fig. 3: General RNA-seq experiment. mRNA is converted to cDNA, and fragments from that library are used to generate short sequence reads. Those reads are assembled into contigs which may be mapped to reference sequences (Wang et al., 2009).

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**.

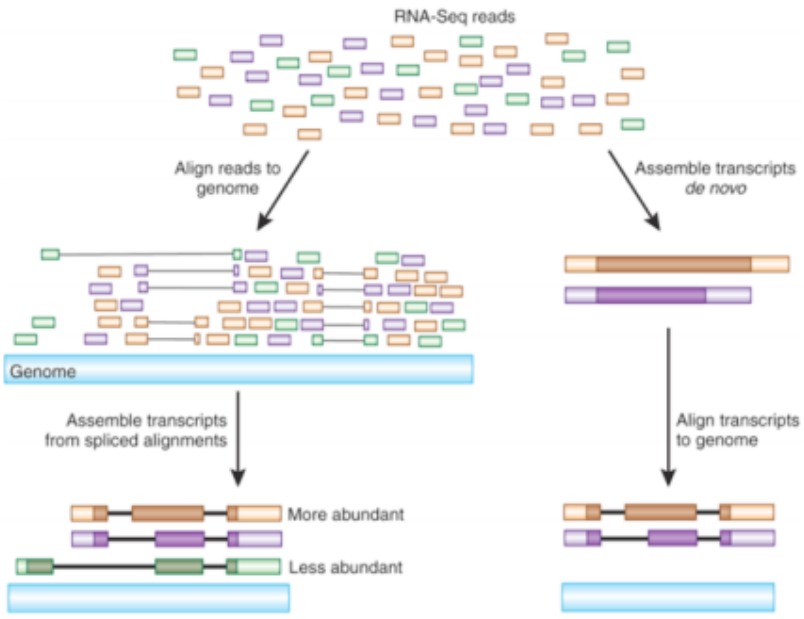


Fig. 4: RNA-seq workflow (Adapted from Advancing RNA-Seq analysis Brian J. Haas and Michael C. Zody Nature Biotechnology 28, 421-423 (2010)

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.

Transcriptomic Data Analysis with R

Soumya Sharma[1]

[1]**ICAR-IASRI**

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, $\log_2$FoldChange 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)
```

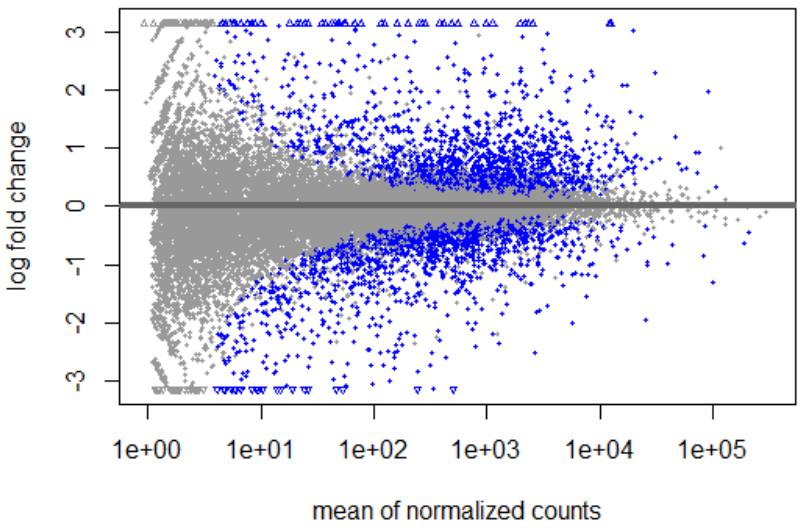


Fig 1: MA plot showing significantly upregulated and downregulated genes as blue dots.

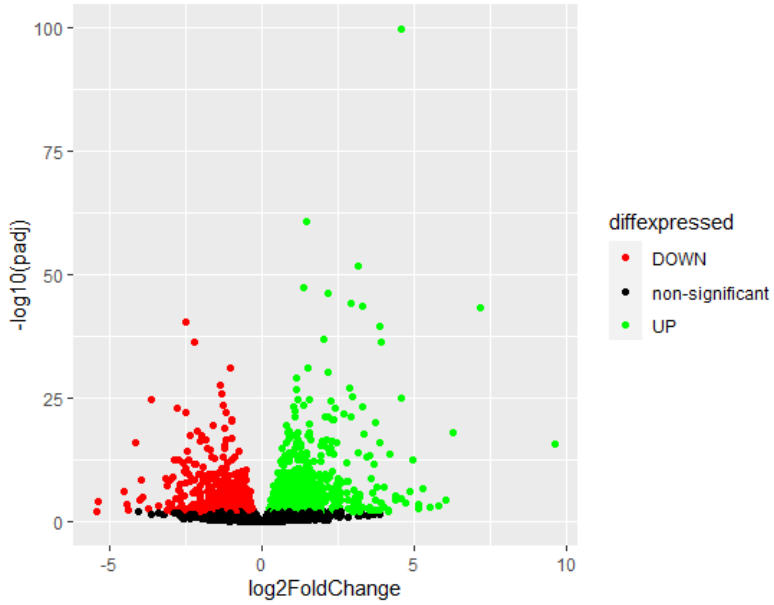


Fig 2: Volcano plot representing upregulated genes as green, downregulated genes as red and non-significant genes as black dots.

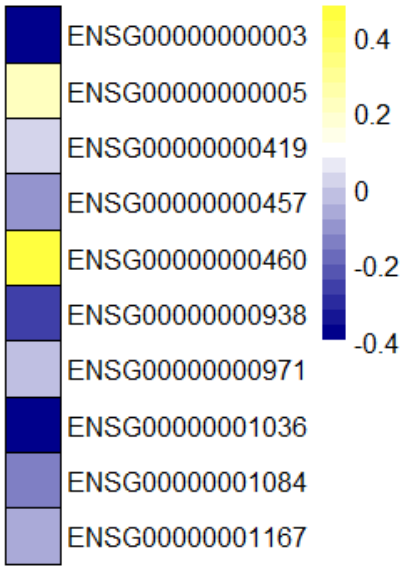


Fig 3: Heatmap representing the expression levels of first 10 genes in terms of log2FoldChange values in a scale of -0.4 to 0.4 where, blue colour represents downregulated genes, yellow represents upregulated genes and expression levels of remaining genes are represented by gradation of colour between blue and yellow.

Refrences:

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.

Single-Cell RNA-Seq Data Analysis

Sudhir Srivastava and Mayank Rashmi

Introduction

Over the past two decades, the development of advanced high-throughput technologies has transformed research across diverse fields, including biomedicine, agriculture, and environmental sciences, by enabling investigations at the cellular and subcellular levels. The term ‘omics’ refers to the comprehensive studies of the genome, transcriptome, epigenome, proteome, and metabolome of a given sample using high-throughput approaches. Single-cell sequencing is a powerful approach that profiles individual cells to study their genome, transcriptome, epigenome, proteome, and other omics layers. Traditional sequencing, also known as bulk sequencing, analyzes the DNA or RNA from a group of cells, providing an average signal across the population. While useful for identifying general patterns in large groups of cells, bulk sequencing cannot capture the unique differences between individual cells. As a result, bulk sequencing is less suitable for studying complex systems with diverse cell types or for detecting rare cell populations and subtle cellular variations. Single-cell sequencing is a newer technology that enables the exploration of genomics, transcriptomics, and other omics layers at single-cell resolution, allowing researchers to distinguish cell populations, track cellular responses, and uncover evolutionary relationships among cells. To date, a number of technologies have been proposed for single-cell transcriptomic studies, and these are differentiated by at least one of the aspects, like cell isolation, cell lysis, reverse transcription, amplification, transcript coverage, strand specificity, and unique molecular identifiers (UMIs). UMI detects and quantifies the availability of unique transcripts. Single-cell RNA Sequencing (scRNA-Seq) allows the comparison of transcriptomes at the level of individual cells. Its major application is to assess transcriptional similarities and differences within a cell population. This enables the identification of rare cell types, characterization of cellular heterogeneity, analysis of developmental trajectories, and investigation of disease-associated transcriptional changes. scRNA-Seq was studied for the first time in 2009 by Tang and their group. Several other next-generation sequencing (NGS)-based assays have been updated for single-cell methods in addition to single-cell RNA-seq. These include assays for genomics, proteomics, and epigenetics, particularly single-cell ATAC-sequencing, which is frequently carried out in combination with scRNA-Seq. Various scRNA-Seq platforms and techniques differ

in terms of transcript coverage (3'/5' tag-based vs. whole-transcript) and throughput (number of cells), and are used to analyse differently. Single-cell RNA sequencing helps in quantification of mRNA expression levels in each cell. The Transposase-Accessible Chromatin Assay for Single-cell Assay utilizing Sequencing (scATAC) describes the openness of cis-regulatory elements in neighboring genes. When scRNA-Seq and scATAC data are analyzed together, they can reveal gene regulatory linkages associated with cellular heterogeneity and enhance the critical genetic information from other omics. Single-cell DNA sequencing (scDNA-seq) technologies offer an unusual opportunity to examine individual cell clonality and the sequence of mutations, both factors that have a big impact on therapeutic results by analyzing DNA mutations and copy number variations.

General Workflow of Single-cell Sequencing

Single-cell sequencing is a process that isolates a single cell for sequencing, then studies molecular mapping, cell heterogeneity, epigenetic change, and immune infiltration. Firstly, isolate the single cells from a tissue sample of interest that includes a few steps, like micro-dissection, microfluidic platforms, and droplet-based methods. Next to perform a list of single tasks in a way that preserves cellular mRNA. mRNA molecule captured by using poly(T) sequence primers that bind to mRNA poly(A) tails. Then convert poly (T)-primed mRNA into cDNA using reverse transcription. This reverse transcribed cDNA is usually amplified by PCR or in vitro transcription. A cDNA library is prepared by inserting index nucleotide barcodes that identify each library. After that pooled the cDNA libraries and made the sequencing libraries by NGS techniques.

There are many commercial kits and reagents are now available for the entire wet-lab procedure for scRNA-Seq. The steps from the beginning (lysing cells) to the end (sequencing) are depicted below in Figure 1. The workflow for scRNA-Seq data generation, from the initial step of cell lysis to the final step of sequencing, is depicted in Figure 1. Some of them are based on the switching mechanism at 5′ end of RNA template (SMARTer) procedure for mRNA capture, reverse transcription, and cDNA amplification. More recently, a number of droplet-based platforms like Chromium from 10x Genomics, ddSEQ from Bio-Rad Laboratories, InDrop from 1CellBio, and μEncapsulator from Dolomite Bio/Blacktrace Holdings are commercially available.

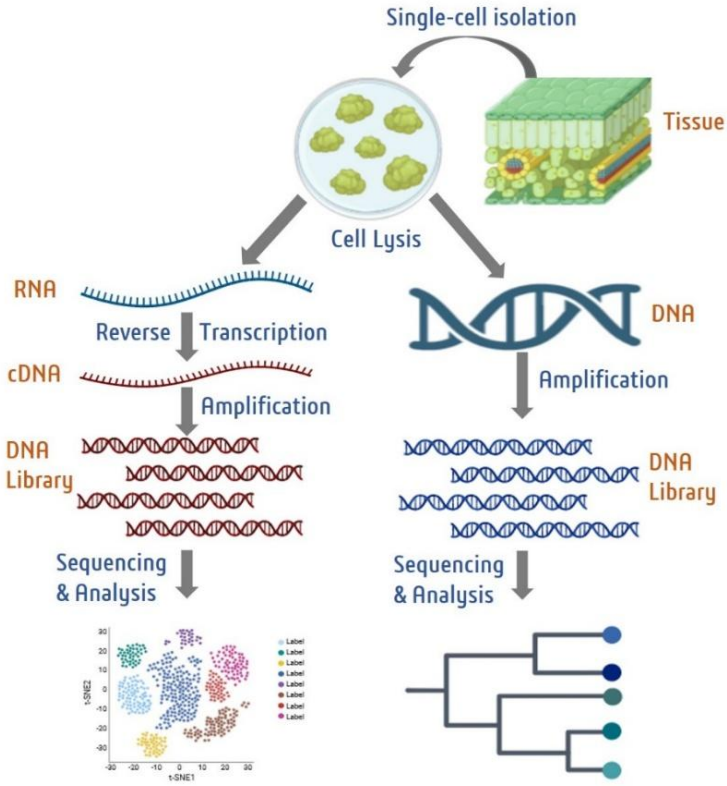


Figure 1. Basic steps of single-cell sequencing

A comparison of different scRNA-Seq technology and experimental protocols is given below in Table 1.

Table 1: A comparison of different scRNA-seq technology and experimental protocols

Platform / Protocol	**Capture Strategy**	**Transcript Coverage**	**Throughput**	**Strengths**	**Limitations**
Smart-Seq / Smart-Seq2	Plate-based (FACS)	Full-length	Hundreds	High sensitivity, detects splice variants	Low throughput, expensive per cell
Fluidigm C1	Microfluidic chip (single-cell capture)	Full-length	Hundreds	Automated handling, high-quality cDNA	Limited cell number, expensive chips
MATQ-seq	Plate-based / low-input	Full-length	Tens-Hundreds	Very sensitive, works for low RNA input	Low throughput, specialized protocol
Drop-seq	Droplet microfluidics	3′-end	Thousands	High throughput, low cost	Lower sensitivity, more dropouts

Experimental design plays a pivotal role in scRNA-Seq studies. Before selecting a protocol, it is essential to carefully assess key factors that may affect data quality, cost, and the biological insights obtained. First, the number of cells to be sequenced per experiment must be considered. This largely depends on the overall heterogeneity of the sample and the expected proportion of specific cell types, based on prior knowledge. An online estimator developed by the Satija Lab provides guidance on how many cells should be sampled in an scRNA-seq experiment to reliably capture a minimum number of cells from each cell type within the dataset (https://satijalab.org/howmanycells/). Second, cell size is an important factor. Each platform has its limitations in handling cells of different sizes. Smaller cells (typically < 25 μm in diameter) can usually be processed with minimal damage, whereas larger or irregularly shaped cells, such as adult cardiomyocytes and neurons, pose greater challenges. In such cases, single-nucleus RNA sequencing (snRNA-Seq) can serve as a practical alternative. Third, minimizing technical biases remains critical. Although new analytical methods continue to improve bias correction, distinguishing true biological variation from technical noise is still challenging. Therefore, careful experimental design that reduces confounding factors is essential.

Single-Cell Data Analysis

Sequenced data of scRNAs can be generated experimentally or retrieved from public repositories such as GEO, SRA, scRNASeqDB, and PlantscRNAdb, etc. Depending on the library preparation method used, the RNA sequences (reads or tags) may be derived either from the 3′ ends or 5′ ends of the transcripts (e.g., 10X Genomics, CEL-seq2, Drop-seq, inDrops) or from full-length transcripts (e.g., Smart-seq). Sequenced data is further analyzed by bioinformatics pipeline using various software and tools. The basic steps of scRNA-Seq data analysis are discussed below.

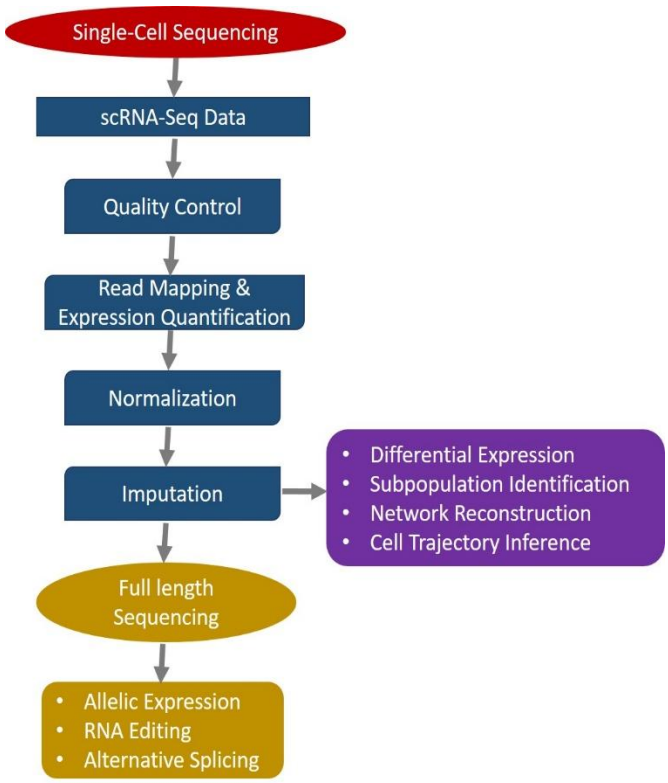


Figure 2. Basic steps of scRNA-Seq data analysis

- **Preprocessing raw scRNA-Seq data:** This step involves various steps such as formatting reads, demultiplexing samples, quality control, trimming, mapping, and quantification.

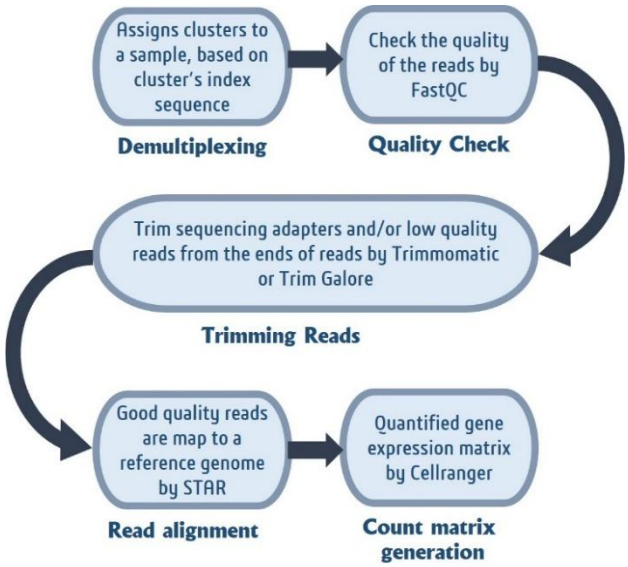


Figure 3. Processing of scRNA sequencing data

- **Demultiplexing:** Raw single-cell transcriptome sequencing data are typically generated in either FASTQ or BCL format, depending on the sequencing platform. Since only FASTQ files can be directly used for quality control, BCL files must first be converted into FASTQ format using appropriate tools. For example, the cellranger mkfastq pipeline, which wraps the bcl2fastq software, is commonly used for this conversion. A simple CSV sample sheet containing at least three columns (lane, sample, and index) must be provided along with the BCL file path. Once converted, the resulting FASTQ files can be assessed for quality using tools such as FastQC.
- **Quality control**: In scRNA-Seq experiments, some low-quality data are generated from cells that are broken, dead, or contaminated with multiple cells. Such low-quality cells can negatively impact downstream analyses and lead to misinterpretation of results. FastQC is a widely used tool for general quality checks and noise removal, while SinQC and Scater are commonly applied for scRNA-Seq–specific quality control.
- **Trimming**: This step removes sequencing adapters and low-quality bases from the ends of reads to improve data reliability. After trimming, data quality is reassessed to ensure clean input for downstream analysis. Commonly used trimming tools for scRNA-seq include Trimmomatic, Cutadapt, TrimGalore, fastp, *etc*.
- **Mapping**: High-quality reads are aligned to a reference genome to identify their gene of origin. STAR is widely used for scRNA-Seq, as it efficiently finds the longest matching sequence within the reference genome. Alternative aligners such as HISAT2, Kallisto, and Salmon are also employed, depending on the analysis goals and computational requirements. Unlike traditional alignment, Kallisto and Salmon use pseudoalignment, which rapidly assigns reads to transcripts without fully mapping each base, thereby reducing computational time and memory usage. Unique Molecular Identifiers (UMIs), which are short molecular tags added to transcripts prior to sequencing, help correct for amplification bias. Reads sharing the same UMI, gene, and cell barcode are collapsed into a single count, ensuring more accurate quantification.

- **Quantification/ Count matrix formation:** The number of reads or UMIs mapping to each gene is counted for every cell. In the resulting count matrix, genes are represented as rows and cells as columns, with each entry indicating the quantified expression level of a gene in a specific cell. For 10X Genomics data, the count matrix is typically generated using the Cell Ranger pipeline. Other commonly used quantification tools include STARsolo, Kallisto|bustools, Salmon/Alevin, HTSeq, Drop-seq Tools, *etc*. which provide flexibility for different scRNA-Seq platforms and analysis needs.

	Cell1	Cell2	Cell3	.....	CellN
Gene1	4	8	22	.	13
Gene2	7	2	3	.	6
Gene3	1	3	9	.	0
.....	.	.	.	.	.
.....	.	.	.	.	.
.....	.	.	.	.	.
GeneM	30	0	17	.	5

Figure 4. Count matrix representation

- **Batch Effect Correction:** The large data is generated in different scales of time, and sometimes data is produced by different laboratories using various protocols, library preparation, and sequencing platforms. So, there is some technical and biological variability, as well as systematic errors are defined as batch effects. The two batch correction methods are MNN (mutual nearest neighbor), used for similar cells in different batches, and kBET (k-nearest neighbor batch effect test), based on the $\chi 2$ method.
- **Normalization and Clustering:** Normalization of scRNA-Seq data is a critical step to remove technical biases (e.g., sequencing depth, capture efficiency) and enable meaningful biological comparisons. Common normalization approaches include log-normalization, scaling by library size, and advanced methods such as SCTransform (Seurat) or scran. Imputation is often applied to address missing values or dropouts, with tools such as MAGIC, SAVER, scImpute, and ALRA helping to recover gene expression signals. After normalization and imputation, dimensionality reduction and clustering are performed to explore cell-to-cell variability. Linear methods such as Principal Component Analysis (PCA) are used for initial dimensionality reduction, while non-linear approaches like t-SNE (t-distributed stochastic neighbor embedding) and UMAP (Uniform Manifold

Approximation and Projection) are widely used for visualization and clustering of cell populations.

- **Marker Identification and Analysis**: Marker identification is a key step in scRNA-Seq analysis to discover disease-relevant genes, understand transcriptional dynamics, and characterize both protein-coding and noncoding RNAs at the single-cell level. The key goals of the scRNA-Seq data analysis are as follows:

✓ Identify cell subpopulations (which are often distinct cell types) within a specific condition or tissue to unravel the heterogeneity of cells

Methods/Tools: Clustering (Seurat, Scanpy, SC3, SIMLR), trajectory inference (Monocle, Slingshot, PAGA).

✓ Find the significantly differentially expressed genes between distinct subpopulations or groups of cells

Methods/Tools: Seurat (FindMarkers), DESeq2, edgeR, MAST, limma-trend

✓ Alternative splicing: For the alternative splicing, generally five basic modes are recognized, including exon-skipping (cassette exon), mutually exclusive exons, alternative donor site, alternative acceptor site, and intron retention.

Methods/Tools: Seurat (FindMarkers), DESeq2, edgeR, MAST, limma-trend.

✓ Detection of RNA-editing dynamics

Tools: REDItools, RNAEditor, RES-Scanner, GIREMI.

✓ Identification of equally expressed genes between parental and maternal genomes by allelic expression analysis

Tools: WASP, ASEReadCounter (GATK), MBASED, scBASE.

✓ Gene regulatory network inference: The goal is to reveal transcription factor–target relationships and regulatory modules.

Tools: REDItools, RNAEditor, RES-Scanner, GIREMI

Advantages of Single-Cell Sequencing

- Genetic material at the single-cell level can be studied with the help of single-cell sequencing. This offers a deeper understanding of the variations inside cells.
- Rare cell types that are not identified by bulk sequencing can be found using it. This is helpful in areas like immunology and cancer studies.

- In a particular tissue, it offers extensive information about individual cells.
- By analyzing the cellular diversity and composition of every cell in a tissue sample, it can be employed to investigate complicated biological systems.
- It helps to improve our knowledge of complexities in tissues and various cell types.

Limitations of Single-Cell Sequencing

- The high cost of single-cell sequencing restricts its use in extensive research.
- Cells may experience stress during the isolation process that influences their viability and characteristics.
- Single-cell sequencing data is noisier than bulk sequencing data, which makes it less reliable.
- A large amount of data is missing or has zero values because single-cell sequencing frequently fails to identify every gene in every cell. It is possible that certain genes exist in a cell but are not detected by sequencing. Data analysis may be challenging due to this sparsity.
- Because it is gathered from a single cell, the data may not be accurate. It might be challenging to determine which measures are accurate and which are the result of technological mistakes.
- Accurately detecting genetic changes in individual cells can be challenging due to the potential for errors and biases introduced by the amplification step.

Applications of Single-Cell Sequencing

- Single-cell sequencing has uses in developmental biology, neurology, diabetes, and cancer research, among other scientific areas.
- Finding genetic differences within tumor cells is helpful in the study of cancer. It helps in understanding the nature of cancer cells and how they function. It is also helpful in the development of targeted therapies.
- Immunology uses it to investigate various immune cells and how they contribute to various diseases.
- In developmental biology, it supports the study of cellular differentiation and the modifications that take place throughout processes like embryogenesis.

- Through the identification of microbial species and their genetic composition, as well as functions in various processes such as antibiotic resistance, single-cell sequencing is also helpful in the study of microbes.
- It is also helpful for understanding the causes of diseases and for clinical diagnostics. It facilitates the identification of unusual or unique cell types and is also helpful for the development of treatment methods for various diseases.

Conclusion

Over the past 13 years, numerous scRNA-Seq protocols have been developed to enhance our understanding of cellular expression variability and dynamics. These methods enable the study of highly heterogeneous samples, including clinical specimens that have undergone fixation or freezing, thereby broadening the scope of experimental applications. Advanced computational analyses of scRNA-Seq data have driven the growth of single-cell transcriptomics, providing powerful tools for both biological and clinical research. By resolving gene expression at single-cell resolution, scRNA-Seq offers deep insights into cellular heterogeneity, transcriptional dynamics, and the regulatory mechanisms underlying diverse biological processes and disease states.

References

Andrews, T. S., Kiselev, V. Y., McCarthy, D., & Hemberg, M. (2021). Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. *Nature protocols*, **16(1)**, 1–9. https://doi.org/10.1038/s41596-020-00409-w

Buttner, M., Miao, Z., Wolf, F. A., Teichmann, S. A., and Theis, F. J. (2019). A test metric for assessing single-cell RNA-seq batch correction. *Nat. Methods* 16, 43–49. doi: 10.1038/s41592-018-0254-1.

Haghverdi, L., Lun, A. T. L., Morgan, M. D., and Marioni, J. C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.* 36, 421–427. doi: 10.1038/nbt.4091.

He, J., Lin, L., & Chen, J. (2022). Practical bioinformatics pipelines for single-cell RNA-seq data analysis. *Biophysics reports*, **8(3)**, 158–169. https://doi.org/10.52601/bpr.2022.210041

Ilicic, T., Kim, J. K., Kolodziejczyk, A. A., Bagger, F. O., McCarthy, D. J., Marioni, J. C., et al. (2016). Classification of low quality cells from single-cell RNA-seq data. *Genome Biol.* 17:29. doi: 10.1186/s13059-016-0888-1.

Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F., & Luo, Y. (2022). Single-cell RNA sequencing technologies and applications: A brief overview. *Clinical and translational medicine*, **12(3)**, e694. https://doi.org/10.1002/ctm2.694

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., et al. (2009). mRNA-Seq whole-transcriptome analysis of a single cell. *Nat. Methods*, 6, 377–382. doi: 10.1038/nmeth.1315.

Important Links

https://satijalab.org/seurat/

https://satijalab.org/howmanycells/

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/

https://github.com/madsen-lab/valiDrops

https://github.com/alexdobin/STAR?tab=readme-ov-file

https://github.com/alexdobin/STAR/blob/master/docs/STARsolo.md

https://www.scrna-tools.org/

https://github.com/seandavi/awesome-single-cell

https://github.com/JiekaiLab/scDIOR

Seurat - Guided Clustering Tutorial

https://satijalab.org/seurat/articles/pbmc3k_tutorial#setup-the-seurat-object

miRNA Identification and Target Prediction

Anu Sharma and Sarika Sahu

Introduction

According to Central Dogma of Biology, there is one DNA in a nucleus which is transcribed into one mRNA and mRNA is translated into protein. However, some years back it was apparent that some of the genes generate other RNAs and these RNAs are non-translated but they get altered at the RNA level and bind with other RNAs thereby inhibiting the protein formation (Fig. 1 and Fig. 2). A large group of these non-coding RNAs are microRNA.

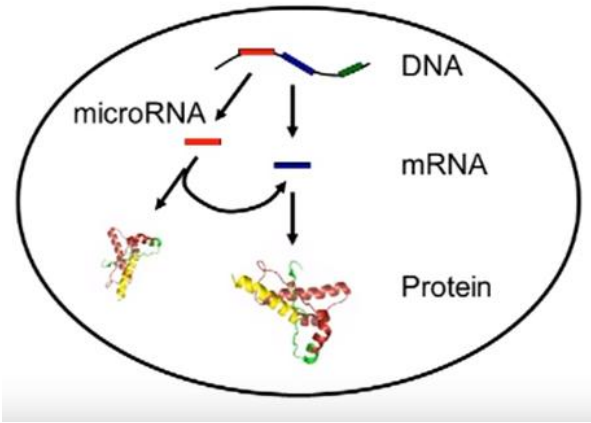


Fig.1: Gene Expression Pathway

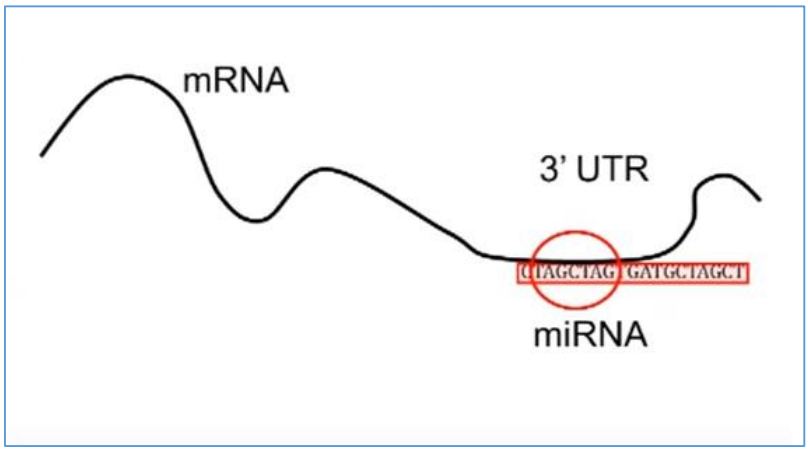


Fig.2: miRNA Regulation

The first described microRNA, lin-4 was cloned and characterised as a translational repressor of developmental timing from Caenorhabditis. Elegans[1-2]. The transcript of this gene was highly unusual as it was non-coding, and produced extremely small transcripts (22nt) from hairpin structured RNA precursors. Second microRNA, let-7 was also cloned from C.elegans [3].

MicroRNAs primarily function as translational repressors by binding to complementary target sequences in the 3' UTR (untranslated region) of mRNA.

The miRNAs are involved in numerous cellular processes in each stage of growth, development, proliferation, differentiation, stress, diseases, transgene suppression, signalling pathway and defence against the viruses [4-8]. Another important property of miRNAs is their conservation among different organisms [9-11]. Different approaches used for miRNA identification includes, gene cloning technology and bioinformatics strategies. Gene cloning is a conventional method to identify the new miRNA accurately, even though it has disadvantages, such as problem of finding miRNAs with low expression, difficulty in cloning, degradation of RNA during sample separation etc. [12]. Rapid development in the field of bioinformatics leads to number of computational programs and other tools to successfully predict the miRNA [13, 14]. Biogenesis of microRNAs is explained in section 2.

MicroRNA Biogenesis

Mature microRNAs are short, single-stranded RNA molecules approximately 22 nucleotides in length. MicroRNAs are sometimes encoded by multiple loci, some of which are organized in tandemly co-transcribed clusters. MicroRNA genes are transcribed by RNA polymerase II as large primary transcripts (pri-microRNA) that are processed by a protein complex containing the RNase III enzyme Drosha, to form an approximately 70 nucleotide precursor microRNA (pre-microRNA). This precursor is subsequently transported to the cytoplasm where it is processed by a second RNase III enzyme, DICER, to form a mature microRNA of approximately 22 nucleotides (Figure 1). The mature microRNA is then incorporated into a ribonuclear particle to form the RNA-induced silencing complex, RISC, which mediates gene silencing. This process is shown in Fig. 3.

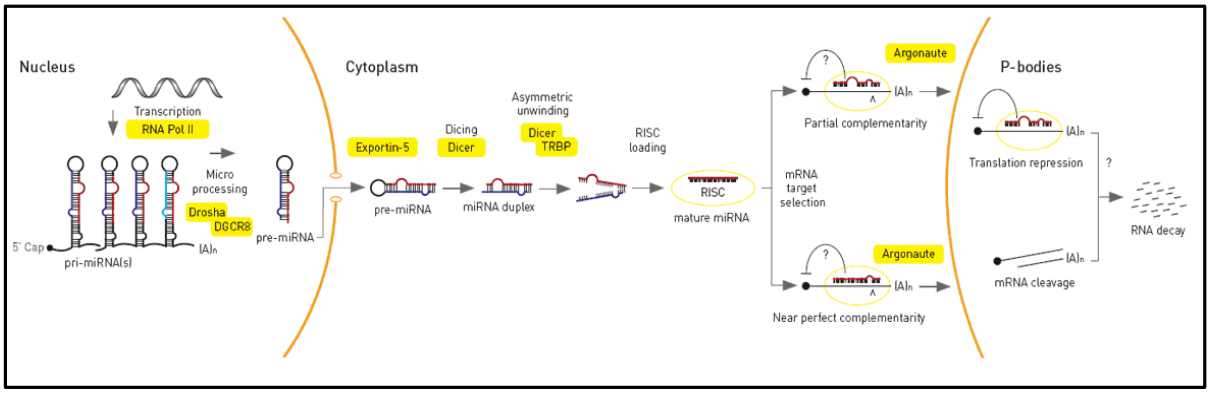


Fig.3: miRNA Biogenesis

Next section describes in detail the computational approach for miRNA prediction and target identification.

1) Computational Prediction and Target Identification

Procedure for computational identification and target prediction is given below:

Step1->Data Preparation*:* search miRNAs homologues from available online databases and download the genomic sequence of the organism under study.

Step1->homology search: Perform a BLASTn search for mRNA sequences in the whole genome assembly sequence of an organism with the e-value <= 0.01 and other default parameters including low complexity filter. Three criteria may be used for screening BLAST

results (1) more than 95% identity between each potential miRNA and the corresponding miRNA in the reference set (known as miRNA homologue); (2) the length difference between each potential miRNA and the corresponding miRNA in the reference set should not more than three bases; (3) the length of the potential miRNA should be between 19-24 nucleotides.

Step2->Secondary structure validation: Extract Pre-miRNA sequences using a sliding window of about 100 nucleotides (nt) in size (moving in increments of approximately 10 nt) from the region ~80 nt upstream of the beginning of the mature miRNA to ~80 nt downstream of the miRNA. Submit the extracted miRNA precursor sequences to Mfold (http://www.bioinfo.rpi.edu/applications/mfold/rna/form1.cgi) for checking of the fold-back secondary structure.

The following steps were considered for screening the candidate miRNA homologs:

i. The RNA sequence folding into an appropriate stem-loop hairpin secondary structure that contains the ~22 nt mature miRNA sequence located in one arm of the hairpin structure;

ii. The predicted mature miRNAs with no more than 6 mismatches with the opposite miRNA sequence in the other arm;

iii. maximum size of 7 nt for a bulge in the miRNA sequence;

iv. miRNA precursors with secondary structures free energy change (ΔG) less than or equal to -37kcal/mol;

v. The A+U content of pre-miRNA within 30-70% are considered;

vi. no loop or break in miRNA sequences should be there.

These criteria significantly reduced false positives.

Step3->Identification of putative candidate miRNA sequences

This step involves the distinction between the real pre-miRNAs from other hairpin sequences with similar stem-loops (pseudo pre-miRNAs) using suitable tools e.g. *MiPred*

Step4-> Target prediction and Functional Annotation

Two set of sequences were taken for target prediction. Firstly, miRanda software was used for target prediction with inputs as predicted miRNAs and 3′UTR sequences of organism. The values for miRanda parameters may be selected as follows: Smith–Waterman hybridization default alignment score[16] greater than or equal to 80, minimum free energy (MFE) of miRNA::mRNA[17] less than or equal to –20kcal/mol and the other parameters with default values.

Secondly, mRNA sequences of are blasted with miRNA sequences using two approaches (1) BLASTN and (2) psRNATarget http://plantgrn.noble.org/psRNATarget/?function=3[18]. miRNA-target alignments may be screened on the basis of three empirical rules (position count starting from 5′ end of the miRNA): (1) no mismatches at positions 2 to 7 (seed region); (2) not more than one G : U pairing allowed in the seed region; and (3) not more than one gap allowed in the alignment. Functional annotation of predicted miRNAs may be done with GO.

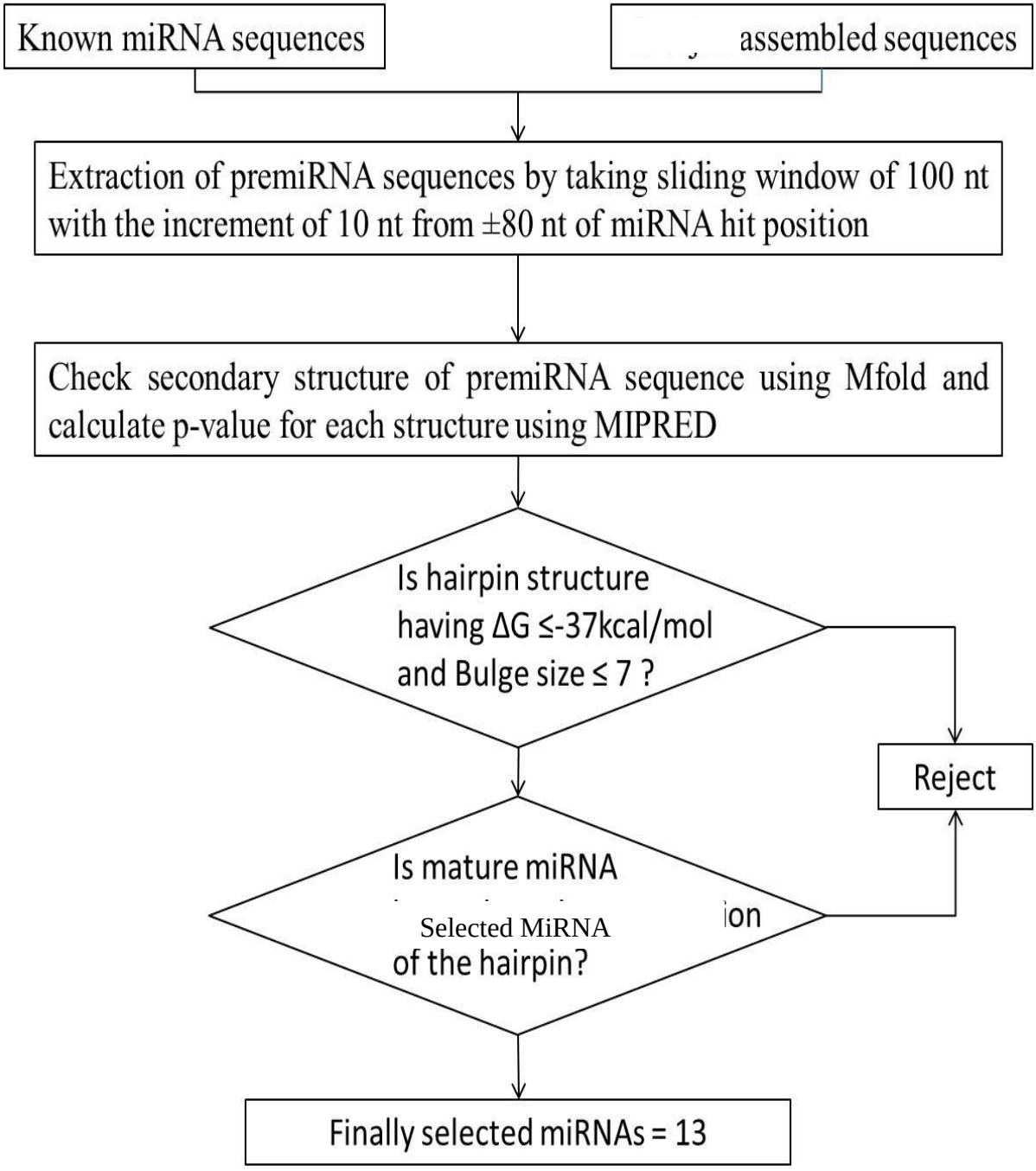


Fig. 4: An overview of different steps involved in microRNA prediction

2) Online MicroRNAs resources

Table 1. Selected online resources [15]

Application name	Type	Web address
miRBase	miRNA sequence and annotation archive	http://www.mirbase.org
smirnaDB	miRNA expression database	http://www.mirz.unibas.ch/cloningprofiles
microRNA.org — miRNA Expression	miRNA expression database	http://www.microrna.org/microrna/getExprForm.do
miRNEYE	miRNA expression database	http://mirneye.tigem.it
S-MED//CC-MED	miRNA expression database	http://www.oncomir.umn.edu/SMED & http://www.oncomir.umn.edu/colon/basic_search.php
TargetScan	miRNA target prediction	http://www.targetscan.org
DIANA-microT-CDS	miRNA target prediction	http://www.microrna.gr/microT-CDS
microRNA.org	miRNA target prediction	http://microrna.org
miRDB	miRNA target prediction	http://www.mirdb.org
RNA22	miRNA target prediction	http://cm.jefferson.edu/rna22v1.0-homo_sapiens
TargetMiner	miRNA target prediction	http://www.isical.ac.in/~bioinfo_miu/targetminer20.htm
PicTar	miRNA target prediction	(http://pictar.mdc-berlin.de)
DIANA-TarBase	Manually curated validated mirna target database	http://www.microrna.gr/tarbase
miRTarBase	Manually curated validated mirna target database	http://mirtarbase.mbc.nctu.edu.tw
miRecords	Manually curated validated mirna target database	http://mirecords.biolead.org
StarBase	Sequence based miRNA experiment database	http://starbase.sysu.edu.cn
DIANA-miRPath	miRNA pathway analysis	http://www.microrna.gr/miRPathv2
miTalos	miRNA signaling pathway analysis	http://mips.helmholtz-muenchen.de/mitalos
GeneTrail	Gene set analysis tool	http://genetrail.bioinf.uni-sb.de

miRGator	Integrated system for functional investigation of miRNAs	http://mirgator.kobic.re.kr
miR2Disease	Database of miRNA related diseases	http://www.mir2disease.org
HMDD	Database of miRNA related diseases	http://202.38.126.151/hmdd/mirna/md
PhenomiR	Database of miRNA related phenotypes	http://mips.helmholtz-muenchen.de/phenomir
miRenvironment	Database of validated miRNA–environment interactions	http://202.38.126.151/hmdd/tools/miren.html
wapRNA	RNA-Seq analysis	http://waprna.big.ac.cn
DSAP	RNA-Seq analysis	http://dsap.cgu.edu.tw
miRanalyzer	RNA-Seq analysis	http://bioinfo2.ugr.es/miRanalyzer/miRanalyzer.php
miRCat	RNA-Seq analysis	http://srna-tools.cmp.uea.ac.uk/animal/cgi-bin/srna-tools.cgi?rm=input_form&tool=mircat

miRBase: The microRNA Sequence Database

The miRBase Sequence database is the primary repository for published microRNA (miRNA) sequence and annotation data. miRBase provides a user-friendly web interface for miRNA data, allowing the user to search using key words or sequences, trace links to the primary literature referencing the miRNA discoveries, analyze genomic coordinates and context, and mine relationships between miRNA sequences. miRBase also provides a confidential gene-naming service, assigning official miRNA names to novel genes before their publication. The methods outlined in this chapter describe these functions. miRBase is freely available to all at http://microrna.sanger.ac.uk/.

Demonstration of miRNA Identification and Target Prediction

Source of data: Transcriptome data and Small RNA sequencing.

To identity or predict the miRNA from any of the above-mentioned source data. However, with advent of sequencing technology, the small-RNA sequencing data is more authenticate for miRNA identification.

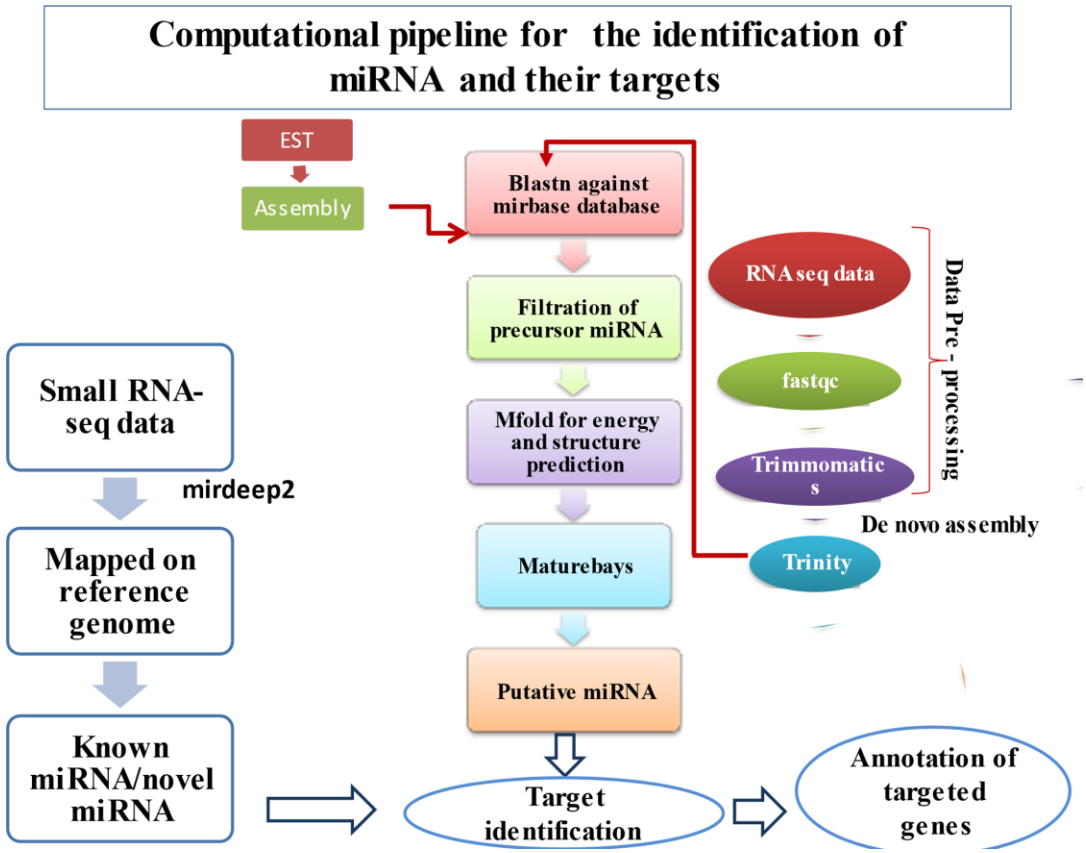


Fig. 1: workflow for the identification of miRNA and their targets

Methodology:

Identification of miRNA from transcriptome data.

Step 1: Check the quality of the data and filter the low quality reads.

Step 2: To assemble the raw transcript into the contigs and check the quality of assembly.

Step 3: Homology method: search the homology between known miRNA sequence from miRbase data base in the transcript with the parameters: e-value and word-length by using blastn program.

Step 4: Filters the blastn result based on similarity and e-value.

Step 5: for the precursor miRNA (pre-miRNA) 100 upstream and 100 downstream sequences along with aligned region retrieves as pre-cursor miRNA.

Step 6: to check the stability of secondary structure of pre-miRNA on the basis of MFE (minimum fold energy), mfold program run and filter the stable pre-miRNA on the basis of MFE<-30 kcal/mol

Step 7: To find out the mature miRNA sequence on the stable pre-miRNA with MatureBayes program.

Step 8: psRNAtarget server/miRanda is use for the target prediction of miRNA.

Step 9: Annotation of targeted genes of miRNA using various tools like DAVID, AgriGo.

Identification of miRNA from small-RNA seq data:

Step 1: Check the quality and pre-processes the reads. Filter the low-quality reads.

Step 2: Run mirdeep2/miRanalyzer for the identification of miRNA (both Raw reads and refence genome is required).

Step 3: Run mfold the prediction of stable miRNA

Step 4: Identification of differentially expressed miRNA (statistically).

Step 5: Target identification of predicted miRNA using MIRANDA and psRNAtarget server for animal and plant respectively.

Step 6: Functional annotation of targeted genes of miRNA using tools like DAVID, Agrigo,

Step 7: Study of gene regulatory network by cytoscape

References

1. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). Basic local alignment search tool. J Mol Biol 215(3): 403-410.
2. Ambros V (2004). The functions of animal microRNAs. Nature 431: 350-355.
3. Aukerman MJ, Sakai H (2003). Regulation of flowering time and floral organ identity by a microRNA and its APETALA2-Like target genes. Plant Cell 15: 2730-2741.
4. Barozai MYK, Irfan M, Yousaf R., Ali I, Qaisar U, et al. (2008). Identification of micro-RNAs in cotton. Plant Physiol Biochem 46(8-9): 739-51.
5. Barozai MYK (2012). The novel 172 sheep (Ovis aries) microRNAs and their targets. Mol. Biol. Rep. 39(5): 6259- 6266.
6. Bartel DP (2004). MicroRNAs: genomics, biogenesis, mechanism, and function. Cell 116: 281–297.
7. Fuliang X, Taylor PF, Zhang B (2010). Identification and characterization of microRNAs and their targets in the bioenergy plant switchgrass (Panicum virgatum). Planta 232: 417-434.

8. Griffiths-Jones S (2006). miRBase: the microRNA sequence database. Methods Mol Biol 342: 129-138.

9. Lee, R.C., Feinbaum, R.L and Ambros, V. (1993). The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. Cell, 75, pp. 843-854.

10. Reinhart,B.J , Slack, F.J, Basson, M, Bettinger, J.C, Pasquinelli, A.E, Rougvie, A.E., Horvitz, H.R. Ruvkun, G. (2000). The 21 nucleotide let-7 RNA regulates developmental timing in Caenorhabditis elegans. Nature, 403, pp. 901-906.

11. Shi Y, Zhao X, Hsieh J, Wichterle H, Impey S, et al. (2010). MicroRNA regulation of neural stem cells and neurogenesis. J Neurosci 30: 14931–14936.

12. Vlachos I.S., Hatzigeorgiou, A.G. (2013). Online resources for miRNA analysis. Clinical Biochemistry. 46:879–900

13. Weber MJ (2005). New human and mouse microRNA genes found by homology search. FEBS J 272: 59-73.

14. Wightman, B, Ha, I, Ruvkun, G. (1993). Post transcriptional regulation of the heterochronic gene lin-14 by lin-4 mediates temporal pattern formation in C. elegans. Cell, 75, pp. 855-862.

15. Zhang BH, Pan XP, Cox SB, Cobb GP, Anderson TA (2006). Evidence that miRNAs are different from other RNAs. Cell Mol. Life Sci. 63: 246-254.

@ 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 endeavor 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

Md. Samir Farooqi, Sudhir Srivastava, Sneha Murmu, Rinku Verma and Girish Kumar Jha (2025). Transcriptomic Data Analysis, Online Training Programme under the Project **DBT-Establishment of Centre for Bioinformatics and Computational Biology in Agriculture-BIC at ICAR-Indian Agricultural Statistics Research Institute**, E-Manual, ICAR-Indian Agricultural Statistics Research Institute, New Delhi.