

1 Basic linux commands

Here will be briefly explained some basic linux commands. For information about more commands, feel free to consult with the internet.

1.1 Getting around

| Command | Explanation |
|-----------------------------------|---|
| <code>ls</code> | list contents of current directory |
| <code>ls <i>dirname</i></code> | list contents of directory named <i>dirname</i> |
| <code>cd <i>dirname</i></code> | change current directory to <i>dirname</i> |
| <code>cd ~</code> | go to home directory (default directory) |
| <code>cd ..</code> | go one directory up |
| <code>pwd</code> | print name of current directory |
| <code>mkdir <i>dirname</i></code> | make a new directory named <i>dirname</i> |
| <code>man <i>command</i></code> | show manual for <i>command</i> |

1.2 File manipulation

| Command | Explanation |
|---|---|
| <code>cp <i>filename1 filename2</i></code> | copy file |
| <code>cp -r <i>dirname1 dirname2</i></code> | copy directory |
| <code>mv <i>name1 name2</i></code> | move file or directory (can be used for renaming) |
| <code>rm <i>filename</i></code> | remove file (you won't be able to recover removed file) |
| <code>rm -r <i>dirname</i></code> | remove directory and its contents |
| <code>more <i>filename</i></code> | command for paging through text file one screenful at a time |
| <code>less <i>filename</i></code> | similar as more. Better for viewing large text files |
| <code>cat <i>filename1 filename2 filenameN</i></code> | concatenate text files in print output to terminal window |
| <code>head <i>filename</i></code> | print first 10 lines of a text file to terminal window |
| <code>head -n 20 <i>filename</i></code> | print first 20 lines of a text file to terminal window |
| <code>tail</code> | opposite of <code>head</code> |
| <code>wc <i>filename</i></code> | show the number of lines, words and characters in a text file |
| <code>cut -f 2 <i>tabDelimitedFilename</i></code> | extract 2nd column from a tab delimited file |
| <code>cut -f 3 -d , <i>comaSeperatedFilename</i></code> | extract 3rd column from a coma seperated file |
| <code>nano <i>filename</i></code> | open <i>filename</i> in text editor <code>nano</code> |

1.3 Archiving and unarchiving of files

Note that for the `tar` utility, option `c` stands for compress, `x` - for uncompress or extract, `z` - for dealing with `tar.gz`, and `j` - for dealing with `tar.bz2`

1.3.1 Compressing

| Command | Explanation |
|--|--|
| <code>tar -cvf filename.tar filename</code> | compress file to .tar format |
| <code>tar -zcvf filename.tar.gz filename</code> | compress file to .tar.gz format |
| <code>tar -jcvf filename.tar.bz2 filename</code> | compress file to .tar.bz2 format |
| <code>zip filename.zip filename</code> | compress file to .zip format |
| <code>gzip filename</code> | compress file to .gz format |

1.3.2 Uncompressing

| Command | Explanation |
|---|---|
| <code>tar -xvf filename.tar</code> | uncompress from .tar format |
| <code>tar -zxvf filename.tar.gz</code> | uncompress from .tar.gz format |
| <code>tar -jxvf filename.tar.bz2</code> | uncompress from .tar.bz2 format |
| <code>unzip filename.zip</code> | uncompress from .zip format |
| <code>gzip -d file.gz</code> | uncompress from .gz format |

1.4 Input/Output redirection

| Command | Explanation |
|--|---|
| <code>command > filename</code> | Output of <i>command</i> is saved to <i>filename</i> , overwriting it |
| <code>command >> filename</code> | Output of <i>command</i> is appended at the end of <i>filename</i> |
| <code>command < filename</code> | <i>command</i> reads input from <i>filename</i> |
| <code>command1 command2</code> | <i>command2</i> takes the output of <i>command1</i> and produces result |

1.5 Filters

| Command | Explanation |
|--|--|
| <code>grep text filename</code> | Prints every line in <i>filename</i> containing <i>text</i> |
| <code>sed 's/red/green/' filename</code> | Prints every line in <i>filename</i> substituting word <i>red</i> with word <i>green</i> |

1.6 Pattern matching

TODO:describe pattern usage.

| Pattern | Explanation |
|----------------|---------------------------------|
| <code>*</code> | matches zero or more characters |
| <code>?</code> | matches one character |

1.7 Miscellaneous

| Command | Explanation |
|----------------------------------|---|
| <code>echo text</code> | display a line of text |
| <code>history</code> | view your command line history |
| <code>wget someWebAddress</code> | download contents of <i>someWebAddress</i> to current directory |

2 Setup of the working environment

Let's create directories for our data:

```
mkdir programs
mkdir ngs_work
mkdir binaries
```

Many of the open source tools are deposited in the <https://github.com> repository. To download software from <https://github.com> easily, we will use a tool called `git`. `git` is already preinstalled on our servers, however, on your own Ubuntu servers you can install it by typing:

```
sudo apt-get install git
```

3 *De-novo* assembly of sequenced reads

3.1 Installation of *de-novo* assembler

For *de-novo* assembly we will use `mira` assembler. You can download it from <http://sourceforge.net/projects/mira-assembler/>. Click on Files → MIRA → stable. Rightclick on `mira_4.0.2_linux-gnu_x86_64_static.tar.bz2` and Copy link address. To download it on our linux server, we will be using command `wget`. In linux terminal type:

```
cd ~/programs
wget -O mira.tar.bz2
```

and paste the copied location. The program will be downloaded in our `~/programs` directory. The downloaded software is archived in `.tar.bz2` format, therefore we need to extract it from archive. To extract it from archive, type in terminal:

```
tar -jxvf mira.tar.bz2
```

A new folder named `mira_4.0.2_linux-gnu_x86_64_static` will appear. This is our extracted software. MIRA is already precompiled for us, so we just need to find the compiled binary files in the folder and copy them in appropriate directory:

```
cd mira_4.0.2_linux-gnu_x86_64_static
cd bin
cp mira ~/binaries
```

cd

MIRA needs a manifest file for performing *de-novo* assembly. We will create a basic manifest file. Our manifest file will consist of 5 entries. From MIRA's manual, these entries are:

- **project** - name of our assemblies project. The project name will be used by MIRA in project's directory naming
- **job** - tells the assembler whether
 1. we want to perform *de-novo* assembly or map reads against reference genome
 2. genomic DNA or transcripts were sequenced
 3. we want accurate (slow) or draft (fast) assembly
- **readgroup** - tells assembler which reads can be pooled together when assembling reads from multiple sequencing technologies
- **data** - tells the assembler where are our reads
- **technology** - tells the assembler what sequencing technology was used for generating reads

To create manifest file, we will use text editor **nano**. To start the text editor, type

nano

and the editor will open. Now to create the manifest file, in the text editor type:

```
project=readsfordenovo_Assembly
job=denovo,genome,draft
readgroup
data=readsfordenovo.fastq
technology=iontor
```

To save the text file hit **Ctrl o** enter the name of the file (e.g. **readsfordenovo.mnfst**), hit enter to save and **Ctrl x** to quit nano. To launch MIRA, type:

```
~/binaries/mira readsfordenovo.mnfst
```

If you wish to gain finer control of some aspects of the assembling process, then, please, do refer to the MIRA's manual.

4 Building variant calling pipeline

We have *IonTorrent* targeted resequencing data from human chromosomes 1., 2. and 19. Our task is to find all nonsynonymous and stop mutations present in the data and to automatize this process by building data analysis pipeline. To accomplish this task we can divide our work in following subtasks:

- Installation of relevant tools
- Obtaining of reference genome
- Read mapping against reference genome
- Variant calling
- Variant annotation
- Filtering of nonsynonymous and stop mutation variants
- Pipeline building

4.1 Obtaining of reference genome

We begin our task by obtaining reference genome. We will download reference genome from database made by University of California, Santa Cruz. The web address of the database is <http://genome.ucsc.edu/>