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
`ls`	list contents of current directory
`ls` *dirname*	list contents of directory named *dirname*
`cd` *dirname*	change current directory to *dirname*
`cd ~`	go to home directory (default directory)
`cd ..`	go one directory up
`pwd`	print name of current directory
`mkdir` *dirname*	make a new directory named *dirname*
`man` *command*	show manual for *command*

1.2 File manipulation

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

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

Command	Explanation
`tar -cvf` *filename.tar filename*	**compress** file to **.tar** format
`tar -zcvf` *filename.tar.gz filename*	**compress** file to **.tar.gz** format
`tar -jcvf` *filename.tar.bz2 filename*	**compress** file to **.tar.bz2** format
`zip` *filename.zip filename*	**compress** file to **.zip** format
`tar -xvf` *filename.tar*	**uncompress** from **.tar** format
`tar -zxvf` *filename.tar.gz*	**uncompress** from **.tar.gz** format
`tar -jxvf` *filename.tar.bz2*	**uncompress** from **.tar.bz2** format
`unzip` *filename.zip*	**uncompress** from **.zip** format

1.4 Input/Output redirection

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

1.5 Filters

Command	Explanation
`grep` *text filename*	Prints every line in *filename* containing *text*
`sed 's/`*red*`/`*green*`/'` *filename*	Prints every line in *filename* substituting word *red* with word *green*

1.6 Pattern matching

TODO:describe pattern usage.

Pattern	Explanation
`*`	matches zero or more characters
`?`	matches one character

1.7 Miscellaneous

Command	Explanation
`echo` *text*	display a line of text

2 Setup of the working environment

Let's create directories for our data:

```
mkdir programs
mkdir NGS_data
mkdir bin
```

Many of the open source tools are deposited in the *github.com* repository. To download software from 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
```

We will need a tool named samtools and we will download it with *git*. To install samtools, type:

```
cd programs
git clone https://github.com/samtools/samtools
git clone https://github.com/samtools/htslib
cd samtools
make
```

To test whether samtools compiled correctly, type:

```
./samtools
```

If you see the program's interface, then the program was compiled correctly. Move compiled binary file to a directory where we will store our compiled software.

```
cp samtools ~/bin
cd
```

Installation of IonTorrent mapping software *tmap*:

```
cd programs
git clone git://github.com/iontorrent/TMAP.git
cd TMAP
git submodule init
git submodule update
sh autogen.sh
./configure
make
```

Move the compiled binary file to our bin folder:

```
cp tmap ~/bin
```

3 Building variant calling pipeline

We have *IonTorrent* targeted resequencing data from human chromosomes 1., 2. and 19. and our task is to find all nonsynonymous and stop mutations present in the data.

- Obtaining reference genome
- Read mapping against reference genome
- Variant calling
- Variant annotation
- Filtering of nonsynonymous and stop mutation variants

3.1 Obtaining reference genome