

---

# **NGS course 2014 Documentation**

***Release 1.0***

**Libor Morkovsky**

**Jul 27, 2017**



---

## Contents

---

<b>1</b>	<b>UNIX primer</b>	<b>3</b>
<b>2</b>	<b>Doing useful stuff</b>	<b>7</b>
<b>3</b>	<b>Scrimer</b>	<b>9</b>
<b>4</b>	<b>Graphical tools</b>	<b>11</b>
<b>5</b>	<b>MetaCentrum</b>	<b>13</b>
<b>6</b>	<b>Visualizing your data</b>	<b>15</b>
<b>7</b>	<b>Supplementary material</b>	<b>17</b>
<b>8</b>	<b>Indices and tables</b>	<b>21</b>



Contents:



# CHAPTER 1

---

## UNIX primer

---

You need to connect to the system, type in commands, keep stuff running while you're disconnected and then pick up your results.

### Connecting to the system

To connect from MS Windows we'll use PuTTY. PuTTY is a simple window that sends everything you type to the remote computer and displays anything the remote computer sends back.

**Warning:** Clipboard works differently in PuTTY! When you select text PuTTY assumes you want to copy it - so it is automatically copied to clipboard. To paste text you need to press right mouse button.

**Warning:** Trying to use windows shortcuts - especially Ctrl-C kills your current running program.

Run putty and enter following information:

```
Host: localhost
Port: 2222
```

### The shell

What you see now is the shell. Shell is a program for entering commands. Your shell is *bash*. Bash is to shell what MS Word is to text editor. You can choose your shell if you need, but most people use bash.

## Multiple windows

You're all used to work with multiple windows (in MS Windows;). You can have them in (remote) Linux as well.

```
screen
```

---

**Note:** The additional benefit is that you can log off, and your programs keep running.

---

Screen is controled after you press the master key - `ctrl-a`. The next key you press is a command to screen.

To create a new window, press `ctrl-a c` (create). To flip among your windows press `ctrl-a space` (you flip windows often, it's the biggest key available). To detach screen - "keep your programs running and go home" - press `ctrl-a d` (detach).

Coming back to work you need to connect to your screen (`-r` is for restore).

```
screen -r
```

## Moving around

You need to type these commands in bash - keep your eye on the prompt - the beginning of the line where you type. Different programs present different prompts.

```
pwd      # prints current directory path
cd        # changes current directory path
ls        # lists current directory contents
ll        # lists detailed contents of current directory
mkdir     # creates a directory
rm        # removes a file
rm -r     # removes a directory
cp        # copies a file/directory
mv        # moves a file/directory
locate    # tries to find a file by name
ln -s     # create symbolic link
```

## Getting help

Call me or Vaclav to get any help ;)

Once you know the name of the command that does what you need, all the details are easily accessible using `man`. To get all possible help about finding text do:

```
man grep
```

To find the name of the command that does what you need, use google:

```
linux search for string
```



## Viewing files

`less` is the command:

```
less /data/slavic1/00-reads/GSVZDOM02.fastq
```

Toggle line wrapping by typing `-S<enter>`.

Search for sequence `ACGT` by typing `/ACGT<enter>`. Press `n` (next) to

Exit `less` by typing `q`.

## Chaining commands

You know how to display whole file (`less`). What if you want to display just specific information from the file?

Change the directory, so we don't have to type so much (press `<tab>` often to spare some typing in bash):

```
cd /data/slavic1
```

View only first 100 lines:

```
<00-reads/GSVZDOM02.fastq head -100 | less
```

View only sequence names (they all start with `@`):

```
<00-reads/GSVZDOM02.fastq grep ^@ | less
```

We can see that the simple assumption was not correct - not only sequence names start with `@`. Let's display every fourth line - names are only on fourth lines:

```
<00-reads/GSVZDOM02.fastq awk '(NR % 4 == 1)' | less
```

Writing to file instead of looking at it is easy:

```
<00-reads/GSVZDOM02.fastq awk '(NR % 4 == 1)' > test-file

# check if the data is there ;)
less test-file

# get rid of the file
rm test-file
```

Chaining is not limited to two commands. I need first 1000 sequence names without the `@`:

```
<00-reads/GSVZDOM02.fastq awk '(NR % 4 == 1)' | cut -c2- | head -1000 > second-test
less second-test
```

## Getting out of it all

```
exit # quits current session
```



## CHAPTER 2

---

### Doing useful stuff

---

#### Moving files around

In the `/data` directory you've got sample data with some precomputed results for the case some computations fail. You don't want to overwrite those, so you will create a 'clean' directory with only the input data.

Using links you can access to the same data from different locations:

**Warning:** Linux uses case sensitive filesystems - File is not file.

```
# create a sandbox directory
cd /data
mkdir slavici_sandbox
cd slavici_sandbox

# link the data from the original directory
ln -s ../slavici/00-reads
ln -s ../slavici/01-genome

# readgroups is small, we can copy it
cp ../slavici/20-smalt/readgroups.txt

# check if there is everything we need
ll
ll 00-reads
```

#### Installing software

There is a canonical software install procedure in UNIX. It can be summarized as

```
wget -O - http://some.site.com/package.tar.gz | tar xvz
cd package
less R<tab> # usually README or README.txt, type capital R!
./configure
make
sudo make install
```

Looks easy .. ? Some packages do not have `configure` file, you just skip the `configure` step then. And some - usually biological - packages are just weird. Then you have to look for information in `README.*` or `INSTALL.txt`.

Let's try to install two packages - `Pipe viewer` and `vcflib`. `Pipe viewer` is a nice tool you can use to watch the progress of your operations. It is distributed in standard `.tar.gz` form. `Vcflib` lives at GitHub - this is where a lot of current open source software resides nowadays.

`Pipe viewer`: go to google, enter `pipe viewer`. Click the `ivarch.com` link. Look for downloads. Right click the `pv-1.5.2.tar.gz` link, select `copy address` or something similar (depends on your browser). Go to PuTTY, type `wget -O -<space>` and right-click your mouse. Then type "`| tar xvz <enter>`"

```
# go to a directory with software
cd ~/sw

# this is a spoiler, you should create the first line yourself
# and do not copy it here
wget -O - http://www.ivarch.com/programs/sources/pv-1.5.2.tar.gz | tar xvz
cd pv<tab>
./configure
make
sudo make install

# test pv
</dev/zero pv > /dev/null
```

`vcflib`: go to google, type `vcflib`, choose the GitHub link. Find `clone url`. Click the clipboard button. Go to PuTTY, type `git clone<space>` and right-click your mouse in PuTTY window. Press `<enter>`.

```
cd ~/sw
git clone --recursive https://github.com/ekg/vcflib.git
cd vcflib
make
```

`vcflib` does not have `make install`. We need to copy the binaries to `$PATH` manually.

```
sudo cp bin/* /usr/local/bin
```

## CHAPTER 3

---

### Scrimer

---

Scrimer is a pipeline for designing primers from transcriptome. Most of the steps are general to NGS data analysis.

Go to:

```
http://scrimer.rtf.d.org
```

In the documentation you will find commands to perform individual steps of NGS analysis.

The data in your virtual machine are filtered to a size where the steps won't take too long - so you should be able to run them all during the session.

### Looking at quality checks

### Steps to take

You should try to understand what the commands do (you don't have to understand *how* they do it).

Go to `/data/slavici_sandbox` directory in your virtual machine.

Set the number of CPUs that can be used (there's only one in the VM):

```
CPUS=1
```

The following headings can be found in the **Scrimer manual**. Follow instructions there, with additional info given here.

### Remove cDNA synthesis adaptors

Set the adaptor sequences, that were used during the library preparation. They will be removed from the sequences.

```
# primers used to synthesize cDNA
# (sequences were found in .pdf report from the company that did the normalization)
PRIMER1=AAGCAGTGGTATCAACGCAGAGTTTGTCTTTTCTTTTCTTTTNN
PRIMER2=AAGCAGTGGTATCAACGCAGAGTACGCGGG
PRIMER3=AAGCAGTGGTATCAACGCAGAGT
```

Run the QC on the raw data and on the trimmed data - so you can see the difference. QC reports produced by `fastqc` are html pages. You need to get them to your machine, because your Linux does not have any graphical display (only text).

Use WinScp to copy them to your machine, after you generate them.

### Map reads to reference assembly

Follow all the instructions on the page.

### Detect and choose variants - Call variants with FreeBayes

Skip the line following `# the rest, ignore order.`

### IGV - Integrative Genomics Viewer

Use WinScp to copy your data from the virtual machine. Run IGV. Load the data to IGV and look around.

### Galaxy

The same data that you see in IGV can be visualized online by uploading to Galaxy server. I uploaded the data beforehand, so we don't upload 10x 500 MB at once.

To see the loaded data, go to:

```
https://usegalaxy.org/u/liborm/v/ngs-course-2014
```

To get the data to Galaxy interface, go to:

```
https://usegalaxy.org/u/liborm/h/ngs-course-2014
```

To see Galaxy, go to:

```
http://usegalaxy.org
```





The best thing is that now you know almost everything to use MetaCentrum. It is the same as using PuTTY to access a local virtual machine.

### Key differences

- you have to register
- you don't use localhost:2222, but something like skirit.ics.muni.cz:22
- you cannot use `sudo`
- you need to allocate computers using `qsub` command
- your data is somewhere else than in `/data`
- you can have 100 cores instead of 1



## CHAPTER 6

---

### Visualizing your data

---

We'll use RStudio that is installed in the virtual machine. To start RStudio go to (in your browser):

```
http://localhost:8787
```

Look around the program - if you ever user R withou RStudio, the difference is big!

Get the data:

```
# download it here to your machine
https://owncloud.cesnet.cz/public.php?service=files&t=aab865a16555adc995b50e33b148318a

# use WinScp to copy it to virtual machine

# unpack the data
unzip data_viz.zip
```

Then use RStudio to navigate to the folder where you unpacked the data. Open `multires_profiles.R`.

Use package manager to install `gtools` library. (It's easy.)

Run the scripts, and see GC profiles of various genomes.

You can suggest other types of plots on this kind of data - we can try to create them.

### Extra UNIX excersise

If you want to plot the same profiles for `/data/slavici`, there is a script `base_counts.py` that can sum it for you. But I wrote it in a hurry for a certain type of data - one gzipped chromosome per file. You can try to convert `luscinia_small.fasta` to this format and run the script.



### Course materials preparation

#### VirtualBox image

Download Debian net install image - use i386 so there is as few problems with virtualization as possible. Not all machines can virtualize x64.

<https://www.debian.org/CD/netinst/>

#### Create new VirtualBox machine

- Linux/Debian (32 bit)
- 1 GB RAM - this can be changed at the users machine, if enough RAM is available
- 12 GB HDD as system drive (need space for basic system, gcc, rstudio and some data)
- name the machine 'node'
- users: root:debian, user:user
- setup port forwarding - 22 to 22 (ssh) - 8787 to 8787 (rstudio server)

Log in as root:

```
apt-get install sudo
usermod -a -G sudo user
```

Login as user:

```
# colrize prompt - uncomment force_color_prompt=yes
# add ll alias - uncomment alias ll='ls -l'
# fast sort and uniq
# export LC_ALL=C
```

```
# maximal width of man
# export MANWIDTH=120
nano ~/.bashrc
. ~/.bashrc

# everyone likes git and screen
sudo apt-get install git screen

# add important stuff to python
sudo apt-get install python-dev python-pip python-virtualenv

# make a vbox snapshot here 'usable system'

# install CloudBioLinux into virtual environment (not to pollute whole system)
mkdir sw
virtualenv py-cbl
. py-cbl/bin/activate

# cloudbiolinux installation (https://github.com/chapmanb/cloudbiolinux)
git clone git://github.com/chapmanb/cloudbiolinux.git
cd cloudbiolinux
python setup.py build
python setup.py install

# fix problem with distribution (new debian wheezy not yet supported?)
sudo bash -c "echo DISTRIB_CODENAME=wheezy >> /etc/os-release"

# choose a minimal flavor for installing
fab -f fabfile.py -H localhost -p user install_biolinux:flavor=ngs_pipeline_minimal

# to ease problem debugging
sudo apt-get install strace
sudo updatedb

# fix /usr/local/lib missing in search path
sudo bash -c "echo /usr/local/lib >> /etc/ld.so.conf.d/local.conf"
# rebuild the cache
sudo ldconfig
```

#### R Studio server:

```
# install rstudio server
# https://www.rstudio.com/ide/download/server.html
sudo apt-get install gdebi-core
wget http://download2.rstudio.org/rstudio-server-0.98.507-amd64.deb
# get old openssl
wget http://ftp.de.debian.org/debian/pool/main/o/openssl/libssl0.9.8_0.9.8o-4squeeze14_amd64.deb
dpkg -i libssl0.9.8_0.9.8o-4squeeze14_amd64.deb

# update R to some decent version
# http://cran.r-project.org/bin/linux/debian/README.html
sudo bash -c "echo 'deb http://mirrors.nic.cz/R/bin/linux/debian wheezy-cran3/' >> /etc/apt/sources.list"
sudo apt-key adv --keyserver keys.gnupg.net --recv-key 381BA480
sudo apt-get update
sudo apt-get install r-base
sudo R
```

```
> update.packages(.libPaths(), checkBuilt=TRUE, ask=F)

# add some packages by hand
curl http://ftp.gnu.org/gnu/parallel/parallel-latest.tar.bz2|tar xvj
cd parallel-20140422/
./configure
make && sudo make install
```

## Prepare data

Create a subset of nightingale data on other machine:

Transfer them to VirtualBox:

```
sudo mkdir /data
sudo chown user:user /data
```

## Create documentation

This was not done in the virtual machine, but belongs to the course preparation...

```
mkdir ngs-course-2014
cd ngs-course-2014

# use default answers to all the questions
sphinx-quickstart

# track the progress with git
git init
git commit -a -m "empty docs and slide"
```

## Spare parts

If instaling from remote machine: Use fabricant to install cloudbiolinux: need the 127.0.0.1 otherwise it does not use ssh

```
fab -f fabfile.py -H 127.0.0.1 --port=2222 -u user -p user install_
  ↳biolinux:flavor=ngs_pipeline_minimal

# full install - does not work
fab -f fabfile.py -H localhost install_biolinux
```





## CHAPTER 8

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`