Handling important NGS data formats in UNIX

Practical training course

NGS Workshop in Nove Hrady 2014

Vaclav Janousek, Libor Morkovsky

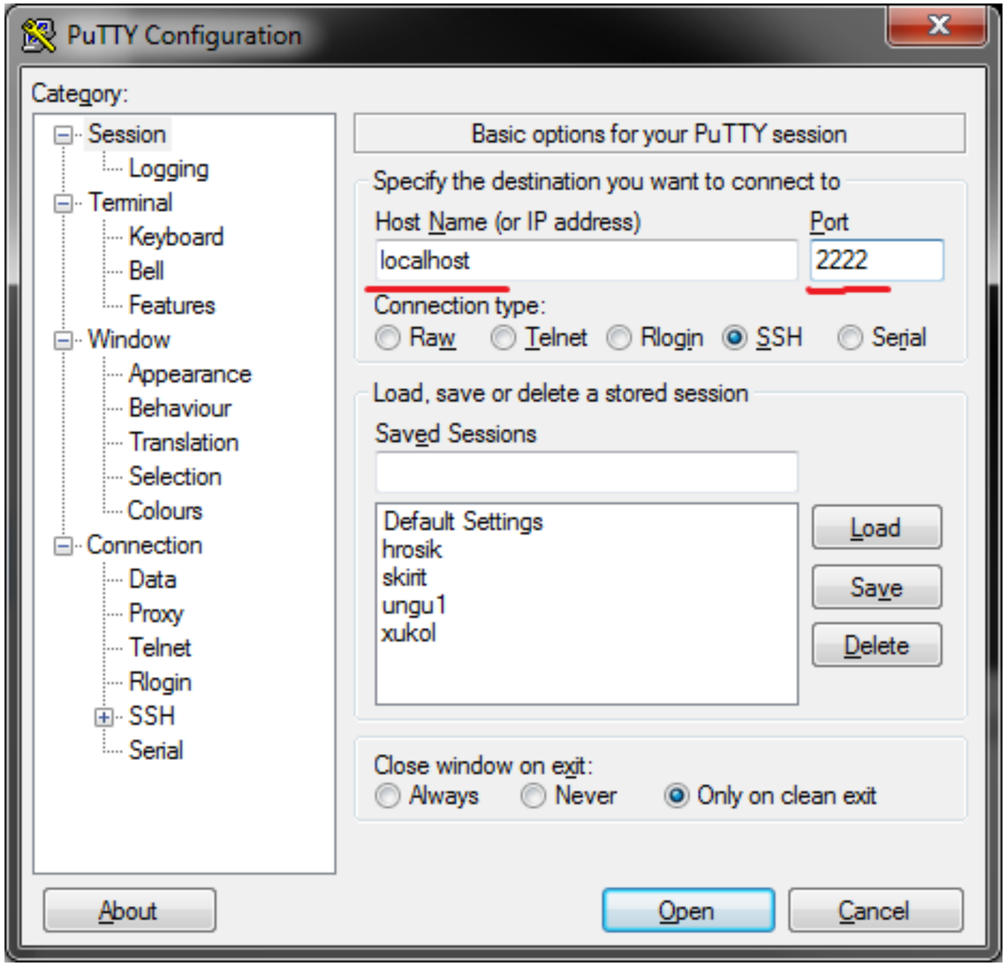
http://ngs-course-nhrady.readthedocs.org

(Exercises & Reference Manual)

Structure of today’s course

- Basic orientation in UNIX
- Software installation
- FASTQ exercise
- GFF, VCF & BED exercise

MS Windows


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): localhost

Port: 2222

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

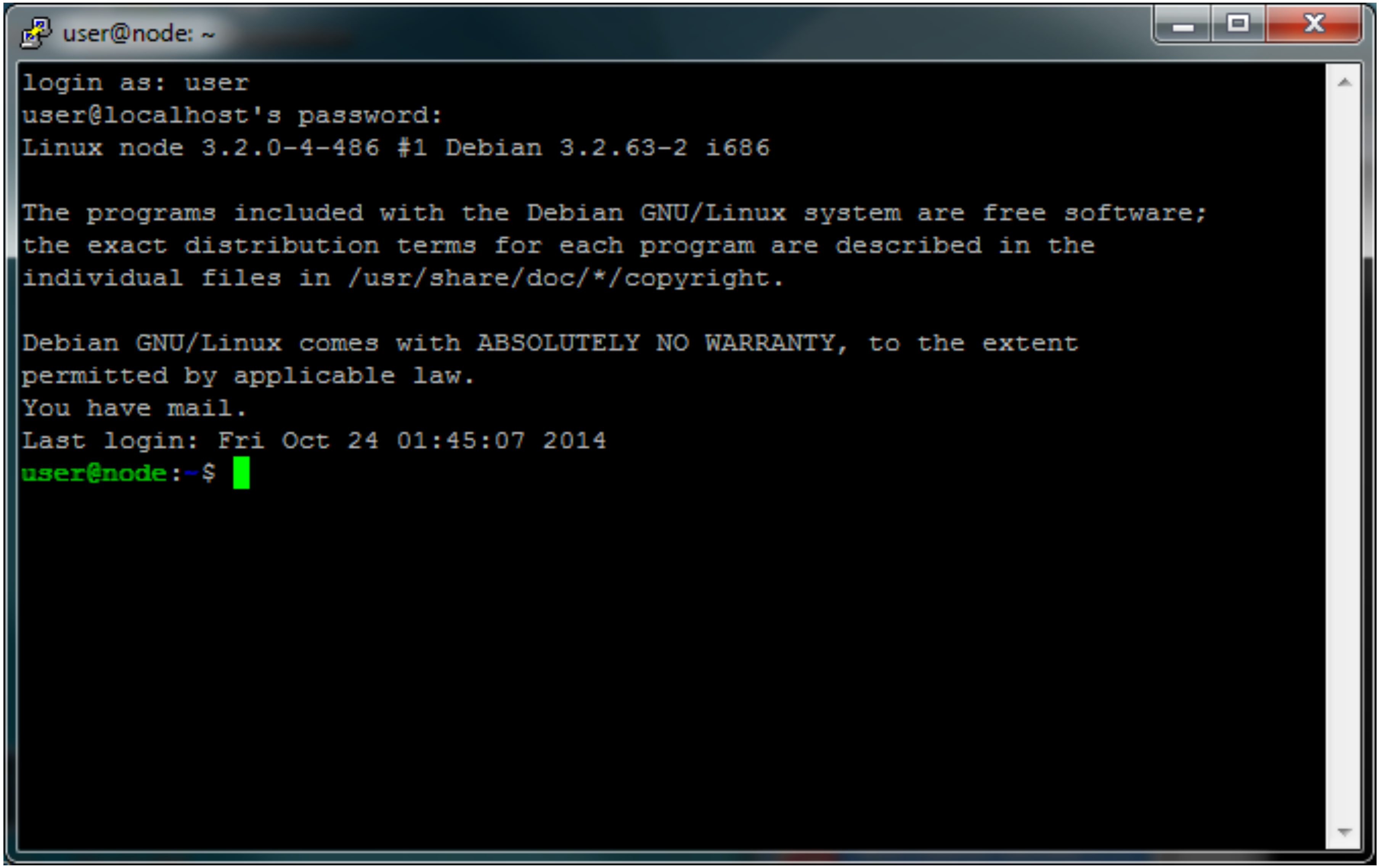
Saved Sessions

Default Settings
hrosik
skirit
ungu1
xukol

Close window on exit: Always Never [✓ Only on clean exit]

Mac OS, Linux

```
ssh -p 2222 user@localhost
```



```
login as: user
user@localhost's password:
Linux node 3.2.0-4-486 #1 Debian 3.2.63-2 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have mail.
Last login: Fri Oct 24 01:45:07 2014
user@node:~$
```

```
sudo apt-get install htop
```

Multiple windows

```
screen
```

- Once in there
  - ctrl+a c … new window (try run htop in one of them)
  - ctrl+a space … switch between windows
  - ctrl+a d … detach (i.e. get off the screen mode)

```
screen -r # get back to screen
screen -ls # list actually running screen sessions
```

Move around commands

- Figure out what these commands do…

```
pwd
ls -ash
cd directory
cd ..
cd
```

Move around commands

- Figure out what these commands do…

```
pwd # path to current directory
ls -ash # lists all files/directories in current directory
cd directory # go to given directory
cd .. # go one directory up
cd # go to home directory
```

Prepare data

- You want to have the data we are going to work with today in data directory in your home directory…

- We have to find the data:

```
locate fastq
locate gff
locate vcf
```

Symbolic links

- The data are in some general directory accessible to all potential users but you want to have them in your own 'data' directory:

```
mkdir data # create directory data
cd data # go to your new data directory
ln -s /data/00-reads 00-reads
ln -s /data/01-genome 01-genome
ln -s /data/02-variants 02-variants
ls -l # check it out
```

Software installation

- Let’s get bedtools

Installation of bedtools

- We need to get source files, compile them and move them to location where the system can find them

```
cd sw # go to sw directory
git clone https://github.com/arq5x/bedtools2 # get
source code from github
cd bedtools2
make # compile binaries
cd bin
sudo cp * /usr/local/bin # copy binaries to place
where the system can find them
```

FASTQ exercise

- Explore FASTQ format:

```
cd data
ls
less -SN 00-reads/00GS60IET02.RL1.fastq
```

Explore FASTQ file: less

Key	Command
Space bar	Next page
b	Previous page
Enter key	Next line
/<string>	Look for string
<n>G	Go to line <n>
h	Help
q	Quit

Explore FASTQ file:

- FASTQ format:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACT
CACAGTTTA
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>
CCCCCCC65
```

How many reads?

- globbing + `grep` + pipe + `wc`:

```
grep "^@[0-9A-Z]*$" 00-reads/*.fastq | wc -l
```

How many reads?

- What’s globbing?

```
ls 00-reads/*.fastq
```

How many reads?

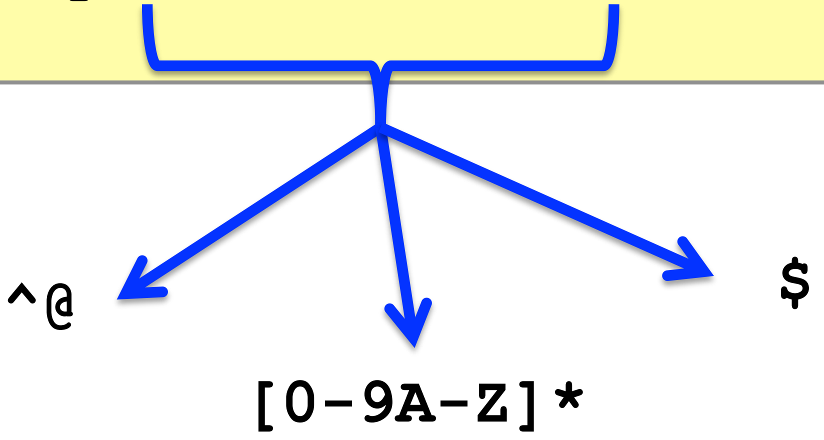
- `grep`
  - matches strings and patterns in text:

```
grep hello file.txt # search for 'hello' in
file.txt
```

How many reads?

- `grep` pattern specification of ID line in FASTQ file:

```
grep "^@[0-9A-Z]*$" 00-reads/*.fastq
```



How many reads?

- pipe (|) + `wc -l`

```
grep "^@[0-9A-Z]*$" 00-reads/*.fastq | wc -l
```

Pipe sends output of the first command to the input of the second one

Summary stat of read lenghts

- Whole pipeline in one line:

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 )
{print $0} }' 00-reads/*.fastq  | tr '\n@' '\t
\n' | tail -n +2 | awk -F $'\t' 'BEGIN{OFS=FS}
{ print $1,length($2)}' | tabtk num -c 2
```

globbing + `awk | tr | tail | awk | tabtk`

Summary stat of read lenghts

- What’s `awk`?
  - complex data manipulation
  - simple programming language

```
awk ’BEGIN{do something}{do something}END{do something}’ f1.txt > f2.txt
```

Summary stat of read lenghts

- `awk`:
  - extract first and second line for each read

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print $0} }' 00-
reads/*.fastq  | head
```

$$\underbrace{\texttt{if((NR+3) \% 4 == 0 || (NR+2) \% 4 == 0)}}_{\text{condition if()}}\underbrace{\texttt{\{print \$0\}}}_{\text{operation to do if condition is TRUE}}$$

Summary stat of read lenghts

- `awk`:
  - extract first and second line for each read

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print $0} }' 00-
reads/*.fastq  | head
```

$$\texttt{if(}\ \underbrace{\texttt{(NR+3) \% 4 == 0}}_{\text{is it first line of FASTQ?}}\ \underset{\text{OR}}{\texttt{||}}\ \underbrace{\texttt{(NR+2) \% 4 == 0}}_{\text{is it second line of FASTQ?}}\ \texttt{)\{print \$0\}}$$

Summary stat of read lenghts

- awk:
  - extract first and second line for each read

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print $0} }' 00-
reads/*.fastq  | head
```

```
if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print $0}
```

awk built-in variable NR (number of record)

modulo => rest of division

Summary stat of read lenghts

- Output

```
@SEQ_ID1
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCAC
@SEQ_ID2
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCAC
@SEQ_ID3
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCAC
```

Summary stat of read lenghts

- `tr` + `tail`:
  - each record = one line

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print $0} }' 00-
reads/*.fastq  | tr '\n@' '\t\n' | tail -n +2 | head
```

```
… | tr '\n@' '\t\n' | tail -n +2 | head
```

replace newlines for TABs &
replace @ for newlines

get rid of first empty
line

Summary stat of read lenghts

- Output

```
SEQ_ID1  GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAA
SEQ_ID2  GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAA
SEQ_ID3  GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAA
```

Summary stat of read lenghts

- awk again:
  - Get lengths of reads

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print
$0} }’ 00-reads/*.fastq  | tr '\n' '\t' | tr '@' '\n' |
tail -n +2 | awk -F $'\t' 'BEGIN{OFS=FS}{ print
$1,length($2)}' | head
```

$$\texttt{... | awk }\underbrace{\texttt{-F \$'\textbackslash t'}}_{\text{set TAB as input field delimiter}}\ \underbrace{\texttt{'BEGIN\{OFS=FS\}}}_{\text{pass TAB as output field delimiter}}\underbrace{\texttt{\{ print \$1,length(\$2)\}}}_{\text{print first column (ID) and length of second one (length or reads)}}\texttt{' | head}$$

Summary stat of read lenghts

- Output

```
SEQ_ID1  456
SEQ_ID2  567
SEQ_ID3  123
```

Summary stat of read lengths

- `tabtk` again:
  - useful utility to deal with tables

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print
$0} }' 00-reads/*.fastq  | tr '\n@' '\t\n' | tail -n +2 |
awk -F $'\t' 'BEGIN{OFS=FS}{ print $1,length($2)}' | tabtk
num -c 2
```

```
… | tabtk num -c 2
```

Summary stat of read lengths

- `tabtk` again:
  - useful utility to deal with tables

```
awk '{ if( (NR+3) % 4 == 0 || (NR+2) % 4 == 0 ){print
$0} }' 00-reads/*.fastq  | tr '\n@' '\t\n' | tail -n +2 |
awk -F $'\t' 'BEGIN{OFS=FS}{ print $1,length($2)}' | tabtk
num -c 2
```

```
197160  236.627     40      721
```

Find primers in FASTQ files

- shell variables + `grep` + `less`:

```
PRIMER1="AAGCAGTGGTATCAACGCAGAGTACGCGGG"
PRIMER2="AAGCAGTGGTATCAACGCAGAGT"
grep --color=always $PRIMER1 00-reads/*.fastq | less -RS
```

Find primers in FASTQ files

- shell variables:

```
PRIMER1="AAGCAGTGGTATCAACGCAGAGTACGCGGG"
PRIMER2="AAGCAGTGGTATCAACGCAGAGT"
echo $PRIMER1
```

Find primers in FASTQ files

- `grep` + `less`:

```
grep --color=always $PRIMER1 00-reads/*.fastq | less -RS
```

GFF, VCF, BED exercise

- Look for SNPs and INDELs identified using reads in 5' UTRs.
  1. Get 5' UTRs from the GFF annotation file and convert it to the BED format
  2. Get SNPs and INDELs from VCF file and convert it to the BED format
  3. Get counts of SNPs and Indels in 5' UTRs (use BEDTools)

Get 5' UTRs from GFF to BED

- Whole pipeline in one line:

```
grep 5utr 01-genome/luscinia_small.gff3 | tr
'; ' '\t' | sed 's/Name=//' | awk -F $'\t'
'BEGIN{OFS=FS}{print $1,$4-1,$5,$10}' > 01-
genome/utrs.bed
```

```
grep | tr | sed | awk
```

Get 5' UTRs from GFF to BED

- GFF:

```
chr1    virtual_genome  mRNA    1       878     1       +   .   ID=contig45913
chr1    liftover        exon    36      147     94      +   .   source=gmap_taeGut1;Name=contig45913;Target=chr1 16243 16354 +;ID=
chr1    liftover        exon    38      147     94      +   .   coords=gmap_taeGut1;Name=ENSTGUT00000004895
chr1    liftover        exon    148     273     94      +   .   source=gmap_taeGut1;Name=contig45913;Target=chr1 17853 17978 +;ID=
chr1    liftover        exon    274     380     96      +   .   source=gmap_taeGut1;Name=contig45913;Target=chr1 24662 24768 +;ID=
chr1    liftover        exon    274     380     96      +   .   coords=gmap_taeGut1;Name=ENSTGUT00000004895
chr1    liftover        exon    381     562     95      +   .   source=gmap_taeGut1;Name=contig45913;Target=chr1 25776 25957 +;ID=
chr1    liftover        exon    381     562     95      +   .   coords=gmap_taeGut1;Name=ENSTGUT00000004895
chr1    mvz-annot       5utr    1       20      1       +   .   color=#00cc00;Name=CLIC6 5'UTR
chr1    mvz-annot       3utr    723     728     1       +   .   color=#00cc00;Name=CLIC6 3'UTR
```

Get 5’ UTRs from GFF to BED

- grep 5’ UTRs:

```
grep 5utr 01-genome/luscinia_small.gff3 | head
```

Get 5' UTRs from GFF to BED

- GFF => BED

```
chr1 mvz-annot   5utr   1      20     1   +   .   color=#00cc00;Name=CLIC6 5'UTR
chr1 mvz-annot   5utr   11955  12128  1   +   .   color=#00cc00;Name=MORC3 5'UTR
chr1 mvz-annot   5utr   31756  31950  1   +   .   color=#00cc00;Name=PIGP 5'UTR
```

```
chr1    1       20      CLIC6
chr1    11955   12128   MORC3
chr1    31756   31950   PIGP
```

Get 5’ UTRs from GFF to BED

- extract gene names:

```
grep 5utr 01-genome/luscinia_small.gff3 | tr '; '
'\t′ | sed 's/Name=//' | head
```

Get 5’ UTRs from GFF to BED

- make BED:

```
grep 5utr 01-genome/luscinia_small.gff3 | tr '; '
'\t’ | sed 's/Name=//' | awk -F $'\t' 'BEGIN{OFS=FS}
{print $1,$4-1,$5,$10}' > 01-genome/utrs.bed
```

```
… | awk -F $'\t' 'BEGIN{OFS=FS}{print $1,$4-1,$5,$10}’ | …
```

BEDTools expect zero based coordinates

Get SNPs & Indels from VCF

- whole pipeline:

```
grep -hv ^# 02-variants/*.vcf | awk -F $'\t'
'BEGIN{OFS=FS}{ if(length($4)==1){ print $1,($2-1),
($2+length($4)-1),"SNP"}else{ print $1,($2-1),
($2+length($4)-1),"INDEL"} }' > 02-variants/
variants.bed
```

`grep | awk`

Get SNPs & Indels from VCF

- VCF:

```
##INFO
##INFO
...
##FORMAT
...
# HEADER
DATA: chrom  position  ID  REF  ALT
DATA...
```

Get SNPs & Indels from VCF

- get data rows from VCF:

```
grep -hv ^# 02-variants/*.vcf | head
```

-v

-h

^#

Get SNPs & Indels from VCF

- awk:

```
grep -hv ^# 02-variants/*.vcf | awk -F $'\t'
'BEGIN{OFS=FS}{ if(length($4)==1){ print $1,($2-1),
($2+length($4)-1),"SNP"}else{ print $1,($2-1),
($2+length($4)-1),"INDEL"} }' | head
```

```
… | awk -F $'\t' 'BEGIN{OFS=FS}{ if(length($4)==1)
{ print $1,($2-1),($2+length($4)-1),"SNP"}
else{ print $1,($2-1),($2+length($4)-1),"INDEL"} }'
| …
```

Get SNPs & Indels from VCF

- awk:

```
grep -hv ^# 02-variants/*.vcf | awk -F $'\t'
'BEGIN{OFS=FS}{ if(length($4)==1){ print $1,($2-1),
($2+length($4)-1),"SNP"}else{ print $1,($2-1),
($2+length($4)-1),"INDEL"} }' | head
```

```
… { if(length($4)==1){
            print $1,($2-1),($2+length($4)-1),"SNP”
          }else{
            print $1,($2-1),($2+length($4)-1),"INDEL”
    }
}'  | …
```

We multiply ‘position’ column
to two (start, end)

Get SNPs & Indels from VCF

- BED:

```
chr1    291    292    SNP
chr1    360    361    SNP
chr1    385    392    INDEL
...     ...    ...    ...
```

Get counts of SNPs & Indels in 5'UTRs

- Whole pipeline:

```
bedtools intersect -a utrs.bed -b
variants.bed -wa -wb | cut -f 4,8 |  sort -
k2,2 | bedtools groupby -g 2 -c 1 -o count
```

```
bedtools intersect | cut | sort | bedtools
                     count
```

Get counts of SNPs & Indels in 5'UTRs

- Associate the two BED files (based on physical position in genome):

```
bedtools intersect -a 01-genome/utrs.bed -b
02-variants/variants.bed -wa -wb | head
```

```
bedtools intersect -a utrs.bed -b variants.bed -wa -wb | …
```

Get counts of SNPs & Indels in 5’UTRs

- Cut out columns:

```
bedtools intersect -a 01-genome/utrs.bed -b
02-variants/variants.bed -wa -wb | cut -f
4,8 | head
```

```
… | cut -f 4,8 | …
```

Get counts of SNPs & Indels in 5’UTRs

- Cut out columns:

```
bedtools intersect -a 01-genome/utrs.bed -b 02-
variants/variants.bed -wa -wb | cut -f 4,8 |
sort -k2,2 | bedtools groupby -g 2 -c 1 -o count
```

```
… | sort -k2,2 | bedtools groupby -g 2 -c 1 -o count
```

data has to be sorted
before it goes to ‘groupby’

Get counts of SNPs & Indels in 5'UTRs

- Cut out columns:

```
bedtools intersect -a 01-genome/utrs.bed -b 02-
variants/variants.bed -wa -wb | cut -f 4,8 |
sort -k2,2 | bedtools groupby -g 2 -c 1 -o count
```

```
… | sort -k2,2 | bedtools groupby -g 2 -c 1 -o count
```

-k specifies range of columns

column based on which we group

column based on which we apply statistics

statistics

Get counts of SNPs & Indels in 5’UTRs

- Cut out columns:

```
bedtools intersect -a 01-genome/utrs.bed -b 02-
variants/variants.bed -wa -wb | cut -f 4,8 |
sort -k2,2 | bedtools groupby -g 2 -c 1 -o count
```

```
INDEL       148
SNP         159
```