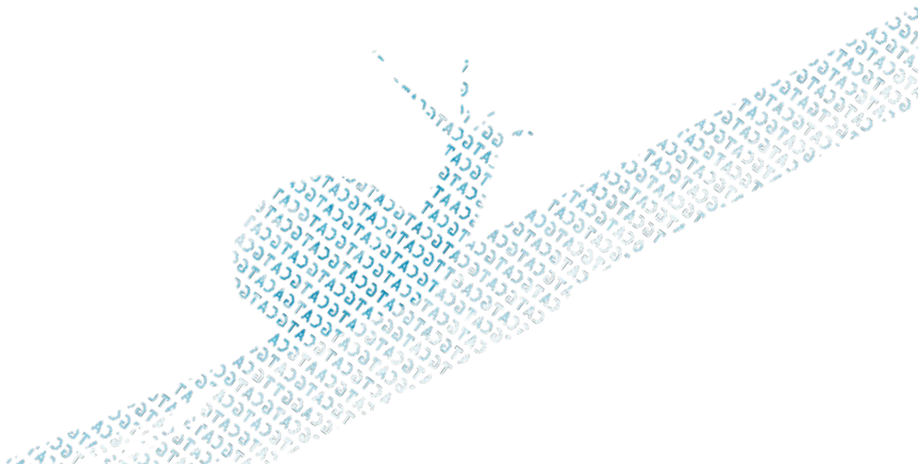
Genetic Diversity Centre Zurich

Genetic Diversity: Analysis

Warmup

Monday 21. June 2021

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Terminal

What do the three terminal commands have in common and what are the differences?

```
$ cp file1.txt file2.txt

$ mv file1.txt file2.txt

$ cat file1.txt > file2.txt
```

GDC Genetic Diversity Centre Zurich

```
$ cp file1.txt file2.txt
```

> Creates a copy of file1.txt called file2.txt but does not change the original file. Both files are identical in content.

```
$ mv file1.txt file2.txt
```

> Renames (or moves) file1.txt to new, file2.txt. file1.txt is lost.

```
$ cat file1.txt > file2.txt
```

> Reads the content of file1.txt and redirects the output to file2.txt. The source file is not change. This might take a while depending on the file size. Careful with file that are not simple text format (e.g. pictures).

What do the different cat commands do?

```
$ cat f1.txt f5.txt f11.txt > New.txt
$ cat f?.txt > New.txt
$ cat f*.txt > New.txt
```

```
$ cat f1.txt f5.txt f11.txt > New.txt
```

> Reads text files f1, f5, and f11 and redirects content to a new file. The command concatenates the content of the three files into a new file.

```
$ cat f?.txt > New.txt
```

> Similar as before but it would concatenate only files that would match the search pattern: starting with f, ending with .txt and only one chracter in between (e.g. f1.txt, f5.txt).

```
$ cat f*.txt > New.txt
```

> Same as the forst but it would concatenate all files that start with f and end with .txt. In our case f1.txt, f3.txt, and f11.txt.

Can you make this cascade of commands work?

```
$ touch log.txt

$ echo -n "N(seq): " > log.txt

$ grep ">" -c seq.fa > log.txt

$ echo -n "n(lines): " > log.txt

$ wc -l seq.fa > log.txt

$ echo " " > log.txt

$ cat log.txt
```

Can you make this cascade of commands work?

```
$ touch log.txt
$ echo -n "N(seq): " >> log.txt
$ grep ">" -c seq.fa >> log.txt
$ echo -n "n(lines): " >> log.txt
$ wc -l seq.fa >> log.txt
$ echo " " >> log.txt
$ cat log.txt
```

What is the difference between the two solutions?

```
$ grep ">" file.fa > header.list

$ grep "Daphnia" header.list > daphnia.list

$ sort daphnia.list > daphnia.sorted.list
```

```
$ grep ">" file.fa | grep "Daphnia" |\

sort > daphnia.sorted.list
```

This is a step-by-step protocol. Two intermediate files are created.

```
$ grep ">" file.fa > header.list

$ grep "Daphnia" header.list > daphnia.list

$ sort daphnia.list > daphnia.sorted.list
```

No intermediate files are generated and the outputs are passed on to the next command?

```
$ grep ">" file.fa | grep "Daphnia" |\

sort > daphnia.sorted.list
```

Challenge #7.2: What is the problem with the following command? Can you correct it?

```
$ # cat sequence*.fa >> sequence_all.fa # ✖✖✖ Do not use!
```

Challenge #7.2: What is the problem with the following command? Can you correct it?

```
$ # cat sequence*.fa >> sequence_all.fa # ✖✖✖ Do not use!
```

sequence*.fa includes e.g. sequence1.fa, sequence_1.fa, sequence123.fa, but also **sequence_all.fa**

The output is part of the input! It will loop with itself and run until you forcefully stop it (e.g. [ctr]+[c]) or you hard disk is full.

Challenge #7.2: What is the problem with the following command? Can you correct it?

```
$ cat sequence*.fa >> sequence_all.fasta
```

```
$ cat sequence*.fa >> all_sequence.fa
```

```
$ cat sequence*.fa >> out/sequence_all.fa
```

R / RStudio

Can you see a problem with these R commands?

```
a <- 3
b <- 5
c <- a + b
```

A5

c is a base function and it is already used to combine values into a vector or list.

```
x <- 1:5
y <- LETTERS[1:4]
data <- c(x, y)
```

Try not to overwrite (base) functions!

What is the output from these lines of R code?

```
> x <- c(10,20,50)
> y <- c(10,20)
> x / y
```

GDC Genetic Diversity Centre Zurich

A6

The output comes with a warning because the two object differ in length.

```
> x <- c(10,20,50)
> y <- c(10,20)
> x / y
[1] 1 1 5
Warning message:
In x/y : longer object length is not a multiple of shorter
object length
> suppressWarnings(x/y)
1 1 5
```

$$\begin{array}{ccc} 10 & 10 & 1 \\ 20 & 20 & 1 \\ 50 & 10 & 5 \end{array}$$

What should be the first and last line of an R script and why?

```
1 > ?
…
end > ?
```

Zurich Genetic Diversity Centre

A7

```
rm(list = ls())
```

```
project.log < - sessionInfo()
write("project.log")
```

Why do I need libraries in R?
What does the R command "library" do?
What libraries should I load?

```
library(tidyverse)
library("ggplot2")
library("vegan")
library("phyloseq")
library("microbiome")
library("ape")
library("scales")
library("randomForest")
library("knitr")
library("plyr")
library("dplyr")
library("dbplyr")
library("plotly")
library("reshape")
```

Load only libraries you really need! If you are only using a function from a library, you do not have to attach it. As long as the package is installed, you can source the function specifying the package. This way you avoid conflicts!

```
ggplot2::annotate()
ggtern::annotate()
```

What is the meaning of conflicts and how can I solve the problem?

```
> tidyverse_conflicts()
── Conflicts ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
x plotly::arrange()   masks dplyr::arrange(), plyr::arrange()
x readr::col_factor() masks scales::col_factor()
x purrr::compact()    masks plyr::compact()
x dplyr::count()      masks plyr::count()
x purrr::discard()    masks scales::discard()
x dplyr::failwith()   masks plyr::failwith()
x plotly::filter()    masks dplyr::filter(), stats::filter()
x dplyr::id()         masks plyr::id()
x dbplyr::ident()     masks dplyr::ident()
x dplyr::lag()        masks stats::lag()
x plotly::mutate()    masks dplyr::mutate(), plyr::mutate()
x plotly::rename()    masks dplyr::rename(), plyr::rename()
x dbplyr::sql()       masks dplyr::sql()
x plotly::summarise() masks dplyr::summarise(), plyr::summarise()
x dplyr::summarize()  masks plyr::summarize()
```

Detach packages that colide with other packages and use the functions with a package specifier.

```
> detach("package:scales", unload = TRUE)
> scales::col_factor()
```