I should be able to:

* Use the shell for version control
* Managing cloud services (like deploying your own shiny server etc.)
* Execute shell commands in R & RMarkdown
* Execute R commands and scripts in the shell

# The shell

## Function documentation

| **Command** | **Description** |
| --- | --- |
| man | Display manual pages for a command |
| whatis | Single line description of a command |

## Shell variables

**echo $USER:** Shows the value of USER.Graphical user interface, text, application

Description automatically generated

| Command | R |
| --- | --- |
| whoami | Sys.info() / whoami::whoami() |
| date | Sys.date() / Sys.time() |
| cal | Display calendar |
| clear | Ctrl + L |

## Copying, pasting, moving, and deleting files

| Command | R |
| --- | --- |
| pwd | getwd() / here::here() |
| ls | dir() / list.files() / list.dirs() / fs::dir\_ls() / dir\_info() |
| cd | setwd() |
| mkdir | dir.create() / fs::dir\_create() |
| rmdir | fs::dir\_delete() |

|  |  |
| --- | --- |
| Command | R |
| touch | file.create() / fs::file\_create() / fs::file\_touch() |
| cp | file.copy() / fs::file\_copy() / fs::dir\_copy() |
| mv | file.rename() / fs::file\_move() |
| rm  find . -name '\*.html' -delete | file.remove() / fs::file\_delete() |
| diff |  |

**pwd**: Prints the absolute path of your current working directory.

**Ctrl + c:** Ends any running instruction.

**set | grep HISTFILESIZE:** how many old commands are stored in your command history. What is its value?

**ls <path>:** Shows what you have in a specific path or working directory.

If our current working directory is /home/repl

* Absolute path: /home/repl/seasona
* Relative path: seasonal

It also has -R to check all sub folders and -F to check add / at every directory and \* to any program.

**ls:** Shows a list of files and directories in the current or specified directory.

| Command | Description |
| --- | --- |
| ls | List directory contents |
| ls -l | List files one per line |
| ls -a | List all files including hidden files |
| ls -la | Display file permissions, ownership, size & modification date |
| ls -lh | Long format list with size displayed in human readable format |
| ls -lS | Long format list sorted by size |
| ls -ltr | Long format list sorted by modification date |

find: Can be used for searching files and directories.

find Script-output/ -name '\*.xlsx' -delete

| Command | Description |
| --- | --- |
| find | Find files or directories under the given directory; recursively |
| find [folder] -name '\*.txt' | Find files by extension |
| find -type d -iname | Find directories matching a given name, in case-insensitive mode |
| find -type d -name | Find directories matching a given name, in case-sensitive mode |

**cd:** move around in the filesystem.

| Command | Description |
| --- | --- |
| cd . | Navigate into directory |
| cd .. | Go up one level |
| cd - | Go to previous working directory |
| cd ~ | Change directory to home directory |
| cd / | Change directory to root directory |

We can combine directory instructions by splitting each path by “/”.

**cp o mv:** Copy and paste a file in a location file with a specific name in a specific location. We can use -r option to copy or move an entire directory.

Rename a file

cp Esteban/original.txt Marleni/duplicate.txt

But it can also copy o move many files to a specific location

cp seasonal/autumn.csv seasonal/winter.csv backup

| Command | Description |
| --- | --- |
| mv | Move or rename files/directories |
| mv -f | Do not prompt for confirmation before overwriting files |
| mv -i | Prompt for confirmation before overwriting files |
| mv -n | Do not overwrite existing files |
| mv -v | Move files in verbose mode |

**rm:** Delete files. It also has the option -r to erase folders too.

| Command | Description |
| --- | --- |
| rm | Remove files/directories |
| rm -r | Recursively remove a directory & all its subdirectories |
| rm -rf | Forcibly remove directory without prompting for confirmation or showing error messages |
| rm -i | Interactively remove multiple files, with a prompt before every removal |
| rm -v | Remove files in verbose mode, printing a message for each removed file |

**rmdir:** Delate empy folders.

**mkdir:**  Creates a empty directory with certain name.

mkdir directory\_name

**touch:** modifies file timestamps which is information associated with file modification.

**-c:** do not create any file

**-t:** use [[CC]YY]MMDDhhmm[.ss] instead of current time

Where:

CC – First two digits of a year

YY – Second two digits of a year

MM – Month of the Year (01-12)

DD – Day of the Month (01-31)

hh – Hour of the day (00-23)

mm – Minutes of the hour (00-59)

Let’s use as example my date of birth.

touch -c -t 199605181330 Database1.accdb

stat Database1.accdb

**diff:** stands for difference. It is used to compare files line by line and display differences.

| Command | Description |
| --- | --- |
| diff | Compare files & directories |
| diff -w | Compare *files*; ignoring white spaces |
| diff -y | Compare *files*; showing differences side by side |
| diff -u | Compare *files*; show differences in unified format |
| diff -r | Compare *directories* recursively |
| diff -rq | Compare *directories*; show the names of files that differ |

diff -w imports\_olsrr.txt imports\_blorr.txt

results: 4a5 indicates after line 4 in file 1, add line 5 from file 2 to make both the files identical.

diff -w imports\_blorr.txt imports\_olsrr.txt

5d4 indicates delete line 5 from file 1 to match both the files at line4.

## Checking files content

| **Command** | **Description** |
| --- | --- |
| echo | Display messages |
| cat | Print contents of a file |
| head | Prints first ten lines of a file by default |
| tail | Prints last ten lines of a file by default |
| more | Open a file for interactive reading, scrolling & searching |
| less | Open a file for interactive reading, scrolling & searching |
| sort | Sort a file in ascending order |

**echo:** command prints text to the terminal.

echo Great Truth > release.txt (saves a file)

| Command | Description |
| --- | --- |
| echo | Display messages |
| echo -n | Print message without trailing new line |
| echo &gt; file | Write message to a file |
| echo &gt;&gt; file | Append message to a file |
| echo -e | Enable interpretation of special characters |

**head:** Show the first 10 rows of a csv file. We can control the number of rows with **-n** option.

head -n 3 seasonal/summer.csv

| Command | Description |
| --- | --- |
| head | Output the first parts of a file |
| head -n | Output the first n lines of a file |
| head -c | Output the first c bytes of a file |
| head -n -x | Output everything but the last x lines of a file |
| head -c -x | Output everything but the last x bytes of a file |

**tail:** Show all rows of csv but first 6 rows (lines).

tail -n +7 seasonal/spring.csv

| Command | Description |
| --- | --- |
| tail | Display the last part of a file |
| tail -n num | Show the last num lines of a file |
| tail -n +num | Show all contents of the file starting from num line |
| tail -c num | Show last num bytes of a file |
| tail -f | Keep reading file until Ctrl + C |
| tail -F | Keep reading file until Ctrl + C; even if the file is rotated |

**cut:** Selects columns position based of number position of the file. We use “-f” list the needed columns separating each one with “,” for individual values or “-“ for ranges, al we use -d to type the columns separator (this doesn’t takes spaces as a possibility).

cut -d, -f 1 seasonal/spring.csv

**paste:** We can bind columns of different files by using this function. We can use the d , as delimitator. It is important to say that last few rows might have the wrong number of columns.

paste -d , seasonal/autumn.csv seasonal/spring.csv

**cat:** Let’s check the content of a text file.

cat couse.txt

cat release\_names\_18.txt release\_names\_19.txt > release\_names\_18\_19.txt (Concatenate files)

| **Command** | **Description** |
| --- | --- |
| cat | Print & concatenate files |
| cat &gt; | Concatenate several files into the target file |
| cat &gt;&gt; | Append several files into the target file |
| cat -n | Lines ID |

**less:** Shows part of the file divided by pages. We can use the spacebar to go to the next page :n to go to the next file opened and :q to exits the document.

less seasonal/spring.csv seasonal/summer.csv

| Command | Description |
| --- | --- |
| less | Open a file for interactive reading, scrolling & searching |
| space | Page down |
| b | Page up |
| G | Go to the end of file |
| g | Go to the start of file |
| / | Forward search |
| ? | Backward search |
| n | Go to next match |
| N | Go to previous match |
| q | Quit |

**grep** (**g**lobal **r**egular **e**xpression **p**rint)**:** Processes text line by line, and prints any lines which match a specified pattern. It is a powerful tool for matching a regular expression against:

* Stream of input
* Text in a file
* Multiple files

It has these options:

* **-c:** print a count of matching lines rather than the lines themselves
* **--color:** will highlight the matched strings
* **-RI:** Search recursively for an exact string
* **-H:** To print the filename for each match
* **-h:** do not print the names of files when searching multiple files
* **-Hn:** Print file name & corresponding line number for each match
* **-i:** ignore case (e.g., treat "Regression" and "regression" as matches)
* **-l:** print the names of files that contain matches, not the matches
* **-n:** print line numbers for matching lines
* **-v:** invert the match, i.e., only show lines that don't match
* **-E:** Use extended regular expression

grep molar seasonal/autumn.csv

**wc:** ("word count") prints the number of characters (-c), words (-w), and lines (-l) in a file.

grep 2017-07 seasonal/spring.csv | wc -l

**sort:** puts data in order. By default it does this in ascending alphabetical order, but the flags :

| Command | Description |
| --- | --- |
| sort | Sort lines of text files |
| sort -r | Sort a file in descending order |
| sort --ignore-case | Ignore case while sorting |
| sort -b | Tells it to ignore leading blanks |
| sort -f | Tells it to fold case (i.e., be case-insensitive). |
| sort -n | Use numeric order for sorting |
| sort -u | Preserve only unique lines while sorting |

**uniq:** removes adjacent duplicated lines. The flag “-c” also counts the number of times a value is repeated.

**xargs:** Extents the functions supported by the | operator.

**-t:** prints the command executed

**-I:** works like the ‘.’ in the %>%

**ls \*.qmd | grep -P -v "^(index|\d{2})" | xargs -I{} quarto render {}**

**-d:** changes the delimiter from \n to any other

xargs -d “\t” function

-P: Specifies the number of processes per iteration.

-n: Specifies the number of processes to print each time

**seq 5 | xargs -n 1 -P 1 bash -c 'echo $0; sleep 1'**

## Repeat commands

**!function:** Runs the last typed commend that used that function.

**history:** Shows all recent typed commands.

**!number:** Runs the same lines ran based on history function.

## Saving files

**nano newfile.sh/txt:** Create or edit a text file.

Text, letter

Description automatically generated

Note that in Nano, "copy and paste" is achieved by navigating to the line you want to copy, pressing **CTRL + K** to cut the line, then **CTRL + U** twice to paste two copies of it.

**>:** We can save the result of any function by using > and a new direction.

head -n 5 seasonal/summer.csv > top.csv

> result.txt head -n 3 seasonal/winter.csv

**|:** Works as %>% in R.

head -n 5 seasonal/summer.csv | tail -n 3

**\*:** We can use this wildcard to select many files. (regular expressions)

cut -d , -f 1 seasonal/\*.csv (all csv in a folder)

cut -d , -f 1 seasonal/\* (all files in a folder)

**?:** matches a single character, so 201?.txt will match 2017.txt or 2018.txt, but not 2017-01.txt.

**[...]:** matches any one of the characters inside the square brackets, so 201[78].txt matches 2017.txt or 2018.txt, but not 2016.txt.

**{...}**: matches any of the comma-separated patterns inside the curly brackets, so {\*.txt, \*.csv}

## Loops

bash headers.sh

$@: all of the command-line parameters given to the script

for f in seasonal/\*.csv; do echo $f; head -n 2 $f | tail -n 1; done

# Print the first and last data records of each file.

for filename in $@

do

head -n 2 $filename | tail -n 1

tail -n 1 $filename

done

## Data Transfer

| **Command** | **Description** | **R** |
| --- | --- | --- |
| wget | Download files from the web | download.file() |
| curl | Transfer data from or to a server | curl::curl\_download() |
| hostname | Name of the current host | R.utils::getHostname.System() |
| ping | Ping a remote host | pingr::ping() |
| nslookup | Name server details | curl::nslookup() |

**wget:** It will download contents of a URL and files from the internet.

| Command | Description |
| --- | --- |
| wget url | Download contents of a url |
| wget -o file url | Download contents of url to a file |
| wget -c | Continue an incomplete download |
| wget -P folder\_name -i urls.txt | Download all urls stored in a text file to a specific directory |
| wget --limit-rate | Limit download speed |
| wget --tries | Limit number of retries |
| wget --quiet | Turn off output |
| wget --no-verbose | Print basic information |
| wget --progress-dot | Change progress bar type to dot |
| wget --timestamping | Check if the timestamp of the file has changed before downloading |
| wget --wait | Wait between retrievals |

**curl:** Command will transfer data from or to a server.

| Command | Description |
| --- | --- |
| curl url | Download contents of a url |
| curl url -o file  curl url > file | Download contents of url to a file |
| curl url &gt; file | Download contents of url to a file |
| curl -s | Download in silent or quiet mode |

## Sudo

**S**uper **U**ser **DO** is a prefix of commands that only superuser or root users are allowed to run.

| **Command** | **Description** |
| --- | --- |
| dpkg --list | List installed packages |
| sudo apt-get update | Update packages |
| sudo apt-get install | Install packages |
| sudo apt-get remove | Remove packages (retain configuration, plugins and settings |
| sudo apt-get purge | Remove packages including personalized settings |
| sudo apt-get autoremove | Remove any dependencies no longer in use |

## File Compression

**tar**: creates, maintains, modifies, and extracts files that are archived in the *tar* format. Tar stands for tape archive and is an archiving file format. The **vf** option is common for all the above operations while the following are specific.

|  |  |
| --- | --- |
| * t for listing * x for extracting | * c for creating * r for adding files |

While dealing with tar.gz archives we will use z in addition to vf and the above options.

| Command | Description | **R** |
| --- | --- | --- |
| tar -tvf | List an archive  tar -tvf release\_names.tar | utils::untar('archive.tar', list = TRUE) |
| tar -tvfz | List a gzipped archive | utils::untar('archive.tar.gz', list = TRUE) |
| tar -xvf | Extract an archive  tar -xvf release\_names.tar | utils::untar('archive.tar') |
| tar -xvfz | Extract a gzipped archive | utils::untar('archive.tar.gz') |
| tar -cvf | Create an uncompressed tar archive  tar -cvf pkg\_names.tar pkg\_names.txt | utils::tar('archive.tar') |
| tar -cvfz | Create a tar gzipped archive | utils::tar('archive.tar', compression = 'gzip') |
| tar -rvf | Add a file to an existing archive  **tar -rvf release\_names.tar release\_names\_18.txt release\_names\_19.txt** |  |
| tar -rvfz | Add a file to an existing gzipped archive |  |

**gzip**, **gunzip,** and **zcat:** commands are used to compress or expand files in the GNU GZIP format i.e. files with .gz extension.

| **Command** | **Description** | **R** |
| --- | --- | --- |
| gzip | Compress a file  gzip release\_names.txt | utils::tar(compression = 'gzip' / R.utils::gzip() |
| gzip -d | Decompress a file  gzip -d sept\_15.csv.gz | utils::untar() / R.utils::gunzip() |
| gzip -c | Compress a file and specify the output file name  gzip -c release\_names.txt > releases.txt.gz | utils::untar(exdir = filename) |
| zip -r | Compress a directory  zip -r packproj.zip dir1 dir2 | zip::zip() |
| zip | Add files to an existing zip file  zip myproject.zip bash.sh | zip::zipr\_append() |
| unzip | Extract files from a zip files  unzip zip\_example.zip | zip::unzip() |
| unzip -d | Extract files from a zip file and specify the output file name  unzip zip\_example.zip –d newfolder | zip::unzip(exdir = dir\_name) |
| unzip -l | List contents of a zip file  unzip -l main\_project.zip | zip::zip\_list() |

## System Info

**uname**: command is used to view important information about the system.

| **Command** | **Description** |
| --- | --- |
| uname | Print details about the current machine and the operating system running on it |
| uname -mp | Hardware related information; machine & processor |
| uname -srv | Software related information; operating system, release number and version |
| uname -n | Nodename of the system |
| uname -a | Print all available information system |

**free:** Display free, used, swap memory in the system.

**df:** Display file system disk space usage.

**exit:** Exit the terminal.

**shutdown:** Shutdown the system.

| **Command** | **Description** | **R** |
| --- | --- | --- |
| time | See how long a command takes to execute | Sys.sleep() |
| sleep | Delay activity in seconds |  |
| sleep 1m | Delay activity in minutes |  |
| sleep 1h | Delay activity in hours |  |
| history | Display command history list with line numbers | history() |
| history -c | Clear the command history list |  |

## Running R from brash

Graphical user interface, text, application, email

Description automatically generated