Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

---

Course content-instructor assignment:
Shell: Annajiat (SWC-shell in GitBash), Renato
git: Raphaela, Holger
Python: Renato, Raphaela
R: François, Ece

Instructors:
- Holger Dinkel, UTC+2 shell/git/python
- Renato Alves (GitHub @unode), UTC+2 PYTHON
- Annajiat Alim Rasel (GitHub @annajiat), UTC+6 shell, git or python (R)
- Raphaela Heil (GitHub @RaphaelaHeil), UTC+2 shell/python (git)
- François Michonneau francois@carpentries.org UTC+2 (GitHub @fmichonneau; Twitter @fmic_), R, shell, git
- Ece Kartal UTC+2 R

Helpers:
- Can Demircan (**python**)
- Lion Schulz (**R**, some Python, git)
- Vanessa Scharf (**git**, **shell**, python)
- Marius Görner (**Python**, some Git, some Unix shell)
- Jan Kevin Schlüsener (**python**, **git**, shell) (Github @jkschluesener)
- Lisanna Paladin (**Python**, git, shell) (GitHub @lisanna)

Links:
Course page: https://hdinkel.github.io/2021-06-14-mpikyb-online/
SC lessons: https://software-carpentry.org/lessons/

Zoom Details for the workshop:

---

https://us02web.zoom.us/j/85341337558?pwd=SWp4RGUwenVkTFBINWpRVWZJVnVaZz09
Meeting-ID: 853 4133 7558
Kenncode: 048532

DAY 1 - Shell

http://swcarpentry.github.io/shell-novice/
Automating a task and thus saving some minutes, can add up significantly over time (see https://xkcd.com/1205/). Tools for automating tasks: autohotkey (WIN) or automator (MAC)

Other useful tools: selenium, apache jmeter

Data used can be downloaded here:
http://swcarpentry.github.io/shell-novice/data/data-shell.zip
https://swcarpentry.github.io/shell-novice/reference.html
http://swcarpentry.github.io/shell-novice/setup.html

Commands used so far:
ls, clear, alias/unalias, cd, man, info, touch, mkdir, nano, echo, wc, sort

Examples for making alias (shortcuts) for commands, directories, etc:
alias ll='ls -l' -> we can use ll instead of ls -l
To delete an alias: unalias ll
Making alias for directories: *alias pypath=~/Documents/PYTHON/* or *pypath=~/Documents/PYTHON/* and use it as *cd "${pypath}"*
Notes:
1. be carefull not to go too crazy with aliases, you can "overwrite" existing commands and thus create problems that are not easily seen
2. they only affect the current shell unless the alias is stored in a shell specific file.

You could define an "undo" command by using an alias :D
alias rm=please_put_this_file_in_the_trash_bin_instead

Navigation

/c/users/user_name
to mean
c:\users\user_name

Ctrl + L (keyboard shortcut for "clear")
Ctrl + c (cancel)
mv [old] [new] (to rename files)

Press tab to automatically fill a command, press it twice to list all possible solutions if multiple auto-filling solutions are possible

path to the desktop on windows(assuming username is d)
/c/users/d/Desktop
~/Desktop

Trubleshooting

command --help
man ls (does not work in Windows Git Bash; just Mac and Linux)
info ls (does not work in Windows Git Bash; just Mac and Linux)
navigating help page with keyboard shortcuts:

- space: next page
- B: previous page
- "q" (quit help)
- | less (for shorter version); example "ls --help | less"

When searching online an error message, clean it from information specific to your setting (example: the specific path to your folders)

Useful Links:

- https://man7.org/linux/man-pages/index.html
- https://ss64.com/ Useful website for "help"
- https://explainshell.com/ Explains commands provided as input
- http://www.shellcheck.net, which will check shell scripts (both uploaded and typed in) for common errors.
- https://stackoverflow.com/questions/tagged/shell
- https://codeblog.jonskeet.uk/2010/08/29/writing-the-perfect-question/
- https://tldp.org/LDP/abs/html/ (Linux) Advanced Bash-Scripting Guide
- https://tldp.org/guides.html Linux Documentation collection
- https://notepad-plus-plus.org/ Editor for Windows

Suggestion: always have a look at the documentation to check which options/functions are available

Commands history

The "Up" arrow goes back to your last command, You can navigate your command history through the "Up" and "Down" arrows

Exploring More ls Flags

You can also use two options at the same time. What does the command ls do when used with the -l option? What about if you use both the -l and the -h option?
(what does ls -l or ls -h do?)

ls -h: human readable format
ls -l: use a long listing format

username/file/folder/variable
short yet meaningful
alphabets
if necessary, followed by _ (underscore) and numbers

avoid:
john's computer - 1
renaming username to "john" didn't solve the problem

handing space
my documents
my\ documents
"my documents"

change to home folder
cd
cd ~
cd /c/users/user_name

Changing shell prompt
export backup_prompt=$PS1

- export PS1='\w \n'

or
export PS1='\w \n$ '

http://swcarpentry.github.io/shell-novice/03-create/index.html

Create/edit a file:
touch draft.txt
nano draft.txt
echo > draft.txt

Ctrl + "o" (allows saving of file name, after verifying that it is correct)
(functions like "%" in MATLAB; allows notes for the human; will not be read by machine)

If you are lost, "explorer ." opens current folder (Windows)
"open ." (Mac)
"nautilus ." (Linux)
"dolphin ."
"konqueror ."
"thunar ." (depending on your favorite file manager)
"nautilus" (Ubuntu)

Simply changing the extension does not change the file type; e.g., .txt to .mp4

http://swcarpentry.github.io/shell-novice/04-pipefilter/index.html
https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

I/O (input/output) redirection

command < input_file.txt > output_file.txt

different ways to run:
command
command < input_file.txt
command > output_file.txt

Pipes
command1 | command2 | command3

Shell scripts (list.sh):
command1
command2
command3
....
command1000

run using
bash **list.sh**
or
./list.sh
or
source list.sh

Out put everything from a schell script to a file: bash steps.sh >> log.txt

https://www.go-fair.org/fair-principles/

Exercise:
Consider the file data-shell/data/animals.txt. After these commands, select the answer that corresponds to the file animals-subset.txt:

```
$ head -n 3 animals.txt > animals-subset.txt
$ tail -n 2 animals.txt >> animals-subset.txt
```

1. The first three lines of animals.txt
2. The last two lines of animals.txt
3. The first three lines and the last two lines of animals.txt
4. The second and third lines of animals.txt

Correct answer (please add + after the number below):
1
2
3 +++++++
4

Speed:
Faster +++++++
Same++++
Slower
First half same, second half slower ++
Faster the begnning, second part was good

DAY 2 - git/github

Q: How do/did you tackle multiple versions of documents/code until now? (add your own strategies / add "+")

- naming/renaming++++++++++++
- MSWord versioning++
- regular Backup/Timemachine+++
- Git/Github ++++
- GitKraken +
- Gitlab++

git config --global user.email firstname.lastname@wherever.com
git config --global user.email githubusername@users.noreply.github.com [for a private email]
git config --global core.editor "nano -w"
git config --global core.autocrlf input [for Linux/Mac]
git config --global core.autocrlf true [for windows]

git config --list
git
git init [creates a new repository inside the current directory]

- don't make a repository inside repository
- to remove a repository, remove the .git file (see it with ls -a)
- Git only tracks files by default not directories.

Steps for saving changes:

- git add file_name
    - for adding all the changes: 'git add --all' or 'git add *'
    - for adding multiple files: git add file1 file2
- git commit -m 'message'
- git remote add origin git@github.com:githublink

Commands for checking status:

- git log: info on an exact commit
    - git log --oneline: compress message
- git status: what has been changed
- git diff: comapre different versions
    - default is last changes
    - use 'git diff --staged' compare to latest commits and what is staged
    - git diff HEAD~n file_name: compare to n times going back

Recovering files:

- git checkout Head~n file_name: recovering the file from latest commits (e.g., HEAD is latest or n-steps before latest)

Ignoring files:

- make a .gitignore file and add stuff that you want to ignore
- force git to track ignored files: git add -f file_name

Collaborating:

- Add already existing repo to your local computer: git clone <url> name_for_repo
- git pull before making new changes or during git conflict (we might need to also edit the files after git pull if there were more conflicts)
- git push to upload new commits to shared repository
- history checking: https://coderefinery.github.io/git-intro/10-archaeology/

Open science

- 'The Turing Way' -project
- get DOI for codes on Git
- Finding the right license for your code: https://choosealicense.com/

Recovering old versions
1
2+
3
4
5++++++++++

Speed

Faster:
Same: ++++++nan+++++++
Slower:

"Final.doc" http://swcarpentry.github.io/git-novice/fig/phd101212s.png

- “owner”: invite a collaborator to your repository
- “collaborator”:
    - *clone* the repository you have been invited to --> git clone <owner’s url> <name>-planets
    - make a new file and add a fun fact about your favourite planet
    - commit and push it to the owner’s repository
- “owner”: get the changes with git pullswitch roles and repeat

What is your favorite editor?
vim++
nano
Pycharm +
vscode ++
RStudio ++
atom+
(Notepad ++) ++
emacs w/ evil mode for sweet vim keybindings
Kate
Microsoft Word ++

what tools/platform did you use to collaborate

- etherpad ++
- google docs+ +++++++++
- overleaf ++++
- slack++++++++++
- Basecamp +
- onedrive++
- google colab+ +
- glip/ring central
- Dropbox+ ++
- Trello ++
- Notion

- Nextcloud +

Do you have access to a local gitlab instance at your institute?
yes +++
no ++
I dont know + ++++++

useful Links

- https://coderefinery.github.io/git-intro/10-archaeology/
- https://git-scm.com/book/en/v2
- https://rogerdudler.github.io/git-guide/index.de.html (German)
- https://the-turing-way.netlify.app/reproducible-research/vcs.html
- https://dvc.org/ (data version control)
- https://openneuro.org/

DAY 3 - Introduction to Python

Lesson: https://swcarpentry.github.io/python-novice-inflammation/ (Programming with Python)

Data: https://swcarpentry.github.io/python-novice-inflammation/data/python-novice-inflammation-data.zip
Code: https://swcarpentry.github.io/python-novice-inflammation/code/python-novice-inflammation-code.zip

Create a folder (in Desktop or elsewhere) and unpack both .zip files inside of the new folder.

Expected result:

- new_folder/code/
- new_folder/data/

Markdown vs Code cells

Shortcuts available to add cells above/below, to run markdown, to move cursor etc. (press Esc before):

- a/b : add a cell above/below
- c/x/v: copy/cut/paste
- m/r/y: markdown/raw/code

Tab completion available + shift tab for contextual information

useful Links:

- https://pep8.org/ how to style your code / python style guide

- https://matplotlib.org/stable/gallery/images_contours_and_fields/interpolation_methods.html
    - https://matplotlib.org/stable/gallery/index.html
- https://swcarpentry.github.io/python-novice-inflammation/03-matplotlib/index.html#plot-scaling
- https://seaborn.pydata.org
- https://plotnine.readthedocs.io/en/stable/# (uses grammar of graphics (ggplot2) in python)
- https://numpy.org/doc/stable/user/numpy-for-matlab-users.html

How to see variables in Notebook:

- globals() for global variables disctionary
- locals() for local variables dictionary
- "line magic" --> %whos

" " or ' ' are equaly exchangeable for defining strings

Shift + TAB -> help or help(function)

Slicing/indexing/dimensions (e.g., for numpy arrays):

- 0:n : start from the first element of the array to n-1 th element
- shape: get number of rows or columns
- for numpy functions like "np.mean"
    - axis = 0: over rows
    - axis = 1: over columns
- -n : counting backwards
- stacking (sticking arrays to each other): np.vstack, np.hstack

https://swcarpentry.github.io/python-novice-inflammation/02-numpy/index.html#slicing-strings

```
import numpy as np
import matplotlib.pyplot as plt

# numpy -> np
# matplotlib.pyplot -> plt

data = np.loadtxt(fname="data/inflammation-01.csv", delimiter=",")

# Plotting:
fig = plt.figure(figsize=(10, 3))  # imperial -> inches

axes1 = fig.add_subplot(1, 3, 1)
axes2 = fig.add_subplot(1, 3, 2)
axes3 = fig.add_subplot(1, 3, 3)

axes1.set_ylabel("average")
axes1.plot(np.mean(data, axis=0))

axes2.set_ylabel("max")
```

```
axes2.plot(np.max(data, axis=0))

axes3.set_ylabel("min")
axes3.plot(np.min(data, axis=0))

fig.tight_layout()

plt.savefig("inflammation.png")
plt.show()
```

Lists:

- .append: add one element
- .extend: add multiple elements
- .pop(n): returns the item present at the given index (n) and this item is also removed from the list.
- .reverse: reverse an array

Glob: a module used to retrieve files/pathnames matching a specified pattern.
import glob
filenames = sorted(glob.glob('data/inflammation-.csv'))*

***False* elements in Python:**

- empty string
- empty list
- zero

Indexing a np.array: Look up 'Mask'

How was the Speed

Faster:
Same: ++++++++++
Slower:

DAY 4 - Introduction to Python

Arrays and lists behave differently when matematical operations are applied to them

dir(numpy)
dir(<yourlist>) --> list methods

use help function to get descriptions of what a method does

Conditionals <, >, <= and >= can be applied to array, not to lists

https://swcarpentry.github.io/python-novice-inflammation/07-cond/index.html#sorting-a-list-into-buckets

os.chdir to change your working directory

loops in Python and conditionals

indentation (readability)

Functions --> return (explicit or implicit), inputs, default inputs

Be friendly to your future self, use comments wisely (do not answer the what question, but the what) and name variables in a meaningful way

```
import numpy as np
import matplotlib.pyplot as plt

# numpy -> np
# matplotlib.pyplot -> plt

def visualize(filename):
    output_file = filename.replace(".csv", ".png")

    if output_file == filename:
        print("Output is the same as input")
        return

    data = np.loadtxt(fname=filename, delimiter=",")

    fig = plt.figure(figsize=(10, 3))  # imperial -> inches

    axes1 = fig.add_subplot(1, 3, 1)
    axes2 = fig.add_subplot(1, 3, 2)
    axes3 = fig.add_subplot(1, 3, 3)

    axes1.set_ylabel("average")
    axes1.plot(np.mean(data, axis=0))

    axes2.set_ylabel("max")
    axes2.plot(np.max(data, axis=0))

    axes3.set_ylabel("min")
    axes3.plot(np.min(data, axis=0))

    fig.tight_layout()

    plt.savefig(output_file)
    plt.show()
```

```
# <--- no return
# return None
```

Errors

Debugging:

- Spyder
- Use IPython embed() and run a script from a normal terminal https://ipython.readthedocs.io/en/stable/api/generated/IPython.terminal.embed.html
- for Jupyter: https://blog.jupyter.org/a-visual-debugger-for-jupyter-914e61716559
- use PyCharm as editor (community edition is free, or get a student liecense for professional edition with institutional email)
- visual studio

Python from command line:

- python script_name.py
- giving input to the script:
    - put in the script:
        - import sys
        - sys.argv[n] : n = 0 is the script name, n>0 are inputs
    - run the script as: python script_name.py input1 input2
- there is also a module called "argparse", we can use for scripts.

Parallel processing:

- multiprocessing
- joblib (external library to be installed)

DAY 4 - Introduction to R

Links

- https://github.com/swcarpentry/r-novice-gapminder/
- https://kdestasio.github.io/post/r_best_practices/
- https://github.com/swcarpentry/good-enough-practices-in-scientific-computing/blob/gh-pages/good-enough-practices-for-scientific-computing.pdf
- RStudio cheatsheets: https://www.rstudio.com/resources/cheatsheets/

Change some stuff in Tools-> global options (disable automatic restoring of the saved environment upon every startup)

Esc -> quiting in console
up key -> calling the previous code

some default avaialble functions: sin, log, log10, log2, exp

"==" isequal
"!=" isdifferent
R is sensitive to inequality only until a certain decimal

Variables:

- assigning variable: x <- value, .x <- value (it's a hidden variable)
    - it's also possible to assign variables with = but it's better to reserve it for other special uses (eg. when defining arguments of a function)
    - use of hidden variables: https://search.r-project.org/CRAN/refmans/crunch/html/hide.html
- ls() -> shows all variables
- ls(all.names = TRUE) -> lists also hidden variables
- rm(x): removes the x variable
- rm(list = ls()): removes all the variables

list of shortcut keys: Tools -> keyborad shortcuts help
Tab key -> automaticcally completes the variable

calling a function without brackets shows its source code
Keyboard shortcut: "Alt" + "-" return " <- "

What will be the value of each variable after each statement in the following program?

```
mass <- 47.5
age <- 122
mass <- mass * 2.3
age <- age - 20
```

Run the code from the previous challenge, and write a command to compare mass to age. Is mass larger than age?

Clean up your working environment by deleting the mass and age variables.

Install packages

- `install.packages("gapminder")`
- `install.packages("ggplot2")`
- `install.packages("dplyr")`
- `install.packages("tidyr")`
- synching packages between collaborators: https://rstudio.github.io/renv/articles/renv.html

Download dataset:
https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd/data/gapminder_data.csv

It is useful to get some general idea about the dataset, directly from the command line, before loading it into R. Understanding the dataset better will come in handy when making decisions on how to load it in R. Use the command-line shell to answer the following questions:

wc -l

1. What is the size of the file?

- ls -lh

1. How many rows of data does it contain?

- wc -l

1. What kinds of values are stored in this file?

- head

Getting help:

- ?function_name
- help(package = 'ggplot2')
- vignette(package = 'ggplot2')

How was the Speed?

Faster: ++
Same: ++++++
Slower:

RStudio cheatsheets: https://www.rstudio.com/resources/cheatsheets/

```
cats <- data.frame(
  coat = c("calico", "black", "tabby"),
  weight = c(2.1, 5.0, 3.2),
  likes_string = c(1, 0, 1)
)
write.csv(cats, "data/feline-data.csv", row.names = FALSE)

cats <- read.csv(file = "data/feline-data.csv", stringsAsFactors = TRUE)
```

useful functions on vectors: tail, head, length, typeof, class

Exercise:

Start by making a vector with the numbers 1 through 26. Multiply the vector by 2, and give the resulting vector names A through Z (hint: there is a built in vector called LETTERS)

Exercise

* Select only the rows for the continent Asia, and store the result in the variable gapminder_asia
* Select only the rows where the gdp per capita is greater than 20000

```
gapminder_asia <- gapminder[gapminder$continent=='Asia',]
gapminder_final <- gapminder_asia[gapminder_asia$gdpPercap>20000,]
```

How was the Speed?
Faster:++++
Same:++++++
Slower

DAY 5 - Introduction to R

Links:

- https://github.com/swcarpentry/r-novice-gapminder/

To load the gapminder data from yesterday
gapminder <- read.csv("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd/data/gapminder_data.csv", stringsAsFactors = TRUE)

To download: https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd/data/gapminder_data.csv

Ex1: Modify the example so that the figure shows how life expectancy has changed over time:

```
ggplot(data = gapminder, mapping= aes(x= gdpPercap, y= lifeExp)) +
  geom_point()

  ggplot(data=gapminder,mapping = aes(x=year, y=lifeExp))+
  geom_point()
```

Ex2: Modify the code from the previous challenge to **color** the points by the “continent” column.

Ex3: Switch the order of the point and line layers from the previous example. What happened?

```
  ggplot(data= gapminder, mapping=aes(x=year, y=lifeExp, by=country)) +
  geom_line(mapping=aes(color=continent)) + geom_point()
```

Ex4:
*Modify the color and size of the points on the point layer in the previous example.

* Modify your solution so that the points are now a different shape and are colored by continent with new

trendlines.

```
ggplot(data = gapminder, mapping= aes(x= gdpPercap, y= lifeExp)) +
  geom_point()+ scale_x_log10() +geom_smooth(method=lm, size= 2)
```

ggplot is quite customizable beyond those themes - you can essentially make the plot look like you want to (e.g. http://xkcd.r-forge.r-project.org/)

Extra point for all on windows machines: Your plots will look a bit pixel-y. That's because R doesn't use antialiasing on Windows by default. If you want to make your plots a bit 'smoother' you can just run:

```
trace(grDevices::png, quote({
  if (missing(type) && missing(antialias)) {
    type <- "cairo-png"
    antialias <- "subpixel"
  }
}), print = FALSE)
```

Ex5: Generate boxplots to compare life expectancy between the different continents during the available years.

Ex6: Write a data-cleaning script file that subsets the gapminder data to include only data points collected since 1990.

S:

```
write.table(gapminder[gapminder$year>1990,],
        file="results/only_from_1990_second_v.csv", sep=",", row.names = FALSE)
```

How was the speed?
Faster+
Same+++++++++
Slower

Data Manipulation

dplyr package - to filter, combine, summarize data

Exp 1: a data frame that has the African values for lifeExp, country and year, but not for other Continents. How many rows does your data frame?

```
year_country_lifeExp_Africa <- gapminder %>%
                        filter(continent == "Africa") %>%
                        select(year, country, lifeExp)
```

Ex2: Calculate the average life expectancy per country. Which has the longest average life expectancy and which has the shortest average life expectancy?

How was the speed?

Faster
Same++++++++
Slower

Afternoon Session: Rmarkdown
tidyverse: https://www.tidyverse.org/learn/

Ex1: Create a new R Markdown document. Delete all of the R code chunks and write a bit of Markdown (some sections, some italicized text, and an itemized list).
Load gapminder : gapminder <- read.csv("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd/data/gapminder_data.csv", stringsAsFactors = TRUE)

Ex2: Use chunk options to control the size of a figure and to hide the code.
pdf Markdown outoput :tinytex::install_tinytex()

If you are curious about writing your thesis in RMarkdown:
https://ourcodingclub.github.io/tutorials/rmarkdown-dissertation/

Further useful resources/links:

- https://py.checkio.org
- https://www.biostars.org/ (although google gets you most of it)
- https://www.biostarhandbook.com/
- http://pythontutor.com/visualize.html#mode=edit
- https://datacarpentry.org/semester-biology/exercises/#Python
- https://www.pythonforbiologists.org/
- https://www.programmingforbiologists.org/
- http://hplgit.github.io/bioinf-py/doc/pub/html/index.html#
- https://coderefinery.org/lessons/ in particular the version control related topics which cover more advanced topics, such as branching
- some fun for later this year: https://adventofcode.com/
- https://www.data-to-viz.com/ (https://www.data-to-viz.com/)
- https://www.datacamp.com/ (https://www.datacamp.com/)
- https://www.tidyverse.org/ (https://www.tidyverse.org/)
- https://rstudio-education.github.io/tidyverse-cookbook/index.html (https://rstudio-education.github.io/tidyverse-cookbook/index.html)
- https://learnxinyminutes.com/docs/python/

Code challenges:

- https://gitexercises.fracz.com/
- http://rosalind.info/ (geared towards bioinformatics)
- https://pythonprinciples.com/challenges/
- https://www.hackerrank.com/
- https://www.codewars.com/
- http://projecteuler.net/
- https://learnxinyminutes.com/docs/r/

Quotes/Tips from your instructors:

- **Renato**:
  - "Be kind to your future self"
  - "Don't be a stranger to learning. You'll be doing it your whole life"
- **Annajiat**:
  - Watch how others work or work in pairs like in pair programming <https://en.wikipedia.org/wiki/Pair_programming>.
- **Raphaela**:
  - "Just keep swimming"
- **Ece**:
  - "play with real data, then you will enjoy more"
- **Holger**:
  - "Learn to touch-type!" = use all your fingers, not just two! This is the single most important computer skill you can learn (until alexa gets smart enough) to make you more productive (eg. type as fast as you think)!

Where to go from here?

how would you like to continue learning?

- meet regularly?
- have us setup a zoom room for you (when? how often?)
- try pair-programming
- do code-reviews (using github)
- ....?

probably best to have one, two, three "champions" to volunteers to drive this forward. Volunteers? Add your names here or get in touch

Troubleshooting Area:

Whenever you encounter a problem/error please paste the error message here, together with your name so we can help you (we will probably delete it after the error has been fixed, unless it applies to others as well)

---

git add mars.txt
warning: LF will be replaced by CRLF in mars.txt.
The file will have its original line endings in your working directory
this is related to the settings above (autocrlf) what it means is that the operating-system dependent file endings (CRLF) are replaced. This is done, so your colleagues/collaborators can work with the same files without problems.
This is just a warning, not an error.
(please add your name if you want/need more dedicated help/feedback) Thank you!

---

masih: How can I have access to the data files?
The links are above, see day 3

---

ping: how to set up work directory in python?
what do you mean with work directory?
create a directory?
Ah, thanks. So how to change it?
are you in windows?Mac i use Anaconda
you can just go the folder (using shell) which you want as you wd and start up python/jupyter. Ok. thanks a lot!
if you are using the anaconda navigator you have to use it to navigate to your wd, as stated below (I think?)

you can navigate the folder structure on the left of JupyterLab

you wd is the directory from which you start python/jupyter
If you use Anaconda Navigator, it will be your home directory, then you change from there. Otherwise, as above stated

---

tab-completion issues:
One possible fix: use the magic: %config Completer.use_jedi = False (https://stackoverflow.com/a/64554305)

---

when I run library(ggplot2), I get this error msg
Error in library(ggplot2) : es gibt kein Paket namens ‘ggplot2’
Could you try running install.packages("ggplot2")?

When I add "df_print: paged" and press knit I get this error:
Error in yaml::yaml.load(..., eval.expr = TRUE) :
  Scanner error: mapping values are not allowed in this context at line 3, column 12
Calls: <Anonymous> ... parse_yaml_front_matter -> yaml_load -> <Anonymous>
Execution halted
Any ideas what's the problem?
Can you post you complete head? I made the mistake of adding "df_print: paged" manually
This was the head:
---
title: "learn_rmarkdown"
output:
  html_document
    df_print: paged
author:
  Roxana
---

1) I try to make two plots in consecutive chunks:
Error in parse_block(g[-1], g[1], params.src, markdown_mode) :
  Duplicate chunk label 'plotting', which has been used for the chunk:
ggplot(data = gapminder, aes(x=year,y=lifeExp,color=continent))+geom_line()
Calls: <Anonymous> ... process_file -> split_file -> lapply -> FUN -> parse_block
Execution halted

2) I put as input:
```{r load library, global options}
knitr::opts_chunk$set(message = FALSE,fig.path = "results",warning=FALSE,
                      fig.width = 15,echo = FALSE)

library(ggplot2)
library(dplyr)
```
But I still get the messages in the html document

Could you try without "load library" as below
```{r global options}
knitr::opts_chunk$set(message = FALSE,fig.path = "results",warning=FALSE,
                      fig.width = 15,echo = FALSE)
```
I tried and still get the messages
Ideally you should separate the global options and load library into separate chunks. That should work
Right! Now it works. Thanks!