
Software Carpentry Workshop: Shell, Git, R

University of Oxford, Feb 19-20, 2024, 9:00 am - 5:00 pm GMT

Privacy note. This document will be anonymised after the course; it does have backup copies

- Workshop webpage: https://wmotion.github.io/2024-02-19-oxford/
- This document https://pad.carpentry.org/2024-02-19-oxford

If you do not want to show the different colours over the text, click on the settings wheel in the right hand corner of this screen and untick 'Authorship colours'.
You can also add your name to the chat on the right hand side of the screen if you would like to ask any questions.

1. Unix Shell

1.1 Lesson notes

If you are lagging behind, see the sequence of commands typed on the screen here:
https://raw.githubusercontent.com/wmotion/teach.git.gitautopush/main/unixshell.log
(The link and its content will be edited after the lesson)
[Apologies - the issues with technology this morning means that the link above has not worked - please look at the Unix lesson (https://swcarpentry.github.io/shell-novice/index.html) and other useful links below for more information]

Useful commands: (in **bold**)
pwd (print working directory - what is the path of the directory that I am currently working in)
ls (list files - list all the files in the directory that I am working in)
cd (change directory - change my directory to following path)
mkdir (make directory - create a directory in my current location)

nano [filename] (create a text file in nano)
rm (remove - remove this file or directpry - use with caution, theres no recycle bin in the Bash shell)
wc (word count - number of characters in a particular file)
***** (match everything in lieu of the *, e.g. *ls *.pdf* shows all files in that directory that are pdfs; it also matches nothing in the sense that a hypothetical oddity like *.pdf* would be caught)
cat (concatenate or join together - it prints the contents of files one after another. If there is only one file then it will only print one file).
sort (arranges the files listed in the folder but whatever parameter you define in either numeric or alphabteical order)
head (list the few lines of information within a specific file)
tail (list the last few lines of information within a specific file)
| (the vertical slash aka *pipe* takes the output of the previous command [to the left of the pipe] and adds it to the input of the next command [to the right of the line])
echo (prints text into the terminal)
bash (a call to run a text file as a bash command)
grep (for "global regular expression print" - a powerful command-line tool to search for text patterns in files, variables, and the output of commands; look at it as a sort of ctrl+F launched from outside the file you search in).
find (to find files and directories to search within a specific place)

For loops episode - https://swcarpentry.github.io/shell-novice/05-loop.html

NOTE: Everything after the hashtag do not need to be typed. This is just extra comments so you understand what the commands in bash do.

```
$ cd ../creatures/ #change directory to creatures
$ head -n 5 basilsk.dat minotaur.dat unicorn.dat #first 5 lines of each of these files
$ for number 1 2 3
> do #do the function below
> echo number
> done #closes the loop
```

examples of for loops to explain the difference between a variable name and a variable value

```
$ for number in a b c
> do
> echo number
> done

$ for number in 1 2 3
> do
> echo $number
> done
OUTPUT
1
2
3
```

```
$ for quid in 10 100 100 nothing
> do
> echo $quid
> done
OUTPUT
10
100
1000
nothing

$ for quid in 10 100 100 nothing
> do
> echo quid
> done
OUTPUT
quid
quid
quid
quid

$ head -n 5 *.dat
$ for filenames in *.dat
> do
> echo $filenames
> head -n 5
> done
```

Shell script episode - **https://swcarpentry.github.io/shell-novice/06-script.html**

```
$ cd ../alkanes/
$ ls
$ head -n 15 octane.pdb
$ head -n 15 octane.pdb | tail -n 5

$ nano middle.sh
```

- [Within Nano]
- head -n 15 octaine.pdb | tail -n 5
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ cat middle.sh # this command shows the content of the script]
$ bash middle.sh  # to run the script

$ nano middle.sh
```

- [Within Nano]
- head -n 15 "$1" | tail -n 5
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ cat middle.sh
$ bash middle.sh
$ bash middle.sh octane.pdb
$ bash middle.sh methane.pdb

$ nano middle.sh
```

- [Within Nano]
- head -n 15 "$2" "$1" | tail -n "$3"
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ cat middle.sh
$ bash middle.sh octane.pdb 15 5 # Shows last 5 lines in first 15 lines of that file
$ bash middle.sh octane.pdb 10 3 # Shows last 3 lines in the first 10 lines of that file
```

Findings things episode - https://swcarpentry.github.io/shell-novice/07-find.html

```
$ cd ../writing/
$ ls
$ cat haiku.txt
$ grep not haiku.txt

$ grep The haiku.txt
$ grep -w The haiku.txt
$ grep -w "is not" haiku.txt
$ grep -w -n "is not" haiku.txt
$ grep -wn "is not" haiku.txt

$ grep -r Yesterday .
$ wc -1 littlewomen.txt  # Number of lines within this text file
$ grep -rn Yesterday

$ ls
$ find .  # finding files within the current directory
$ find . -type d
$ find . -type f
# next: find all files names in this directory that ends with .txt]
$ find . name *.txt  #  this find the single file in the current directory
$ find . -name "*.txt"  # this finds all files in  the current directory and below
```

1.2 Useful resources

- This course
  - Lesson + exercises: http://swcarpentry.github.io/shell-novice/
  - Exercises (selected from the curriculum)
    - Scripts (aka batch files, macros, …) https://docs.google.com/presentation/d/1OcRyJygB7M40ARjJR931zk8WlooFisyJE

            6DRF7K_o-Y/edit#slide=id.p
        - Loop$ with for … in ….; do … ; done https://docs.google.com/presentation/d/1Mvm3WQsM1FxvguNkHTxn74g6TVn0j9li3RlTLEL9-5I/edit#slide=id.p
        - Shell wildcard charac?er* https://docs.google.com/presentation/d/1JcEAVjSBi4tStJ_ZPydpY_M-XVMJjoIUnVsXp-Kr7vs/edit#slide=id.p
        - Move, rename, copy, and delete files https://docs.google.com/presentation/d/1XJrbPtWibZlrwT2TY3WFko5K5e_p33w2Hm08KXcYt1c/edit#slide=id.p
    - Software Carpentries Shell Reference https://swcarpentry.github.io/shell-novice/reference.html
    - More on the Shell: https://carpentries-incubator.github.io/shell-extras/
- Official Bash manual: https://www.gnu.org/software/bash/manual/
- Five reasons why researchers should learn to love the command line https://www.nature.com/articles/d41586-021-00263-0
- Extra (general) readings
    - D. Ince. The Computer. Very Short Introductions. Oxford University Press
    - **S. Dasgupta. Computer Science. Very Short Introductions. Oxford University Press**
    - Fortunato & Galassi (2021) The case for free and open source software in research and scholarship. https://doi.org/10.31235/osf.io/ye7sx

1.3 Feedback

1.3.1 What worked out well (and what you would keep)

1. very practical, easy coding to follow, good advice and tips
2. Found the course very easy to follow, very systematic and coherent.
3. practical examples very good + v. easy to follow-along
4. Good explanations from the basics with context
5. Easy to follow along, and helpful examples
6. engaging, easy to follow, solid explanation -- perfect for beginners like myself. Thank you so much!
7. The practicals and step by step was good
8. Well instructed, fell easy to follow and well organised

- **A1-8**. Thanks. I will strive to keep it going like this

1.3.2 What didn't work out well (and how you would improve it)

F1. Timings > longer course!

- **A1**. Agree, undoubtedly. Three lessons in three days of 6 hours could possibly work out optimally, but I do not even know if the Carpentries organization mandates the hosts to follow a standard format. To be enquired. Instructors have to manage time, content and participants, indeed

F2. Coffee / tea would be nice!

- **A2**. Hopefully the host will read this too :-)

F3. Would be nice to practice interacting with remote servers, setting up ssh + configs, writing more complicated shell scripts, but maybe there's a more advanced course that I don't know about!

- **A3**: One step in that direction at the Carpentries is https://carpentries-incubator.github.io/shell-extras/. I suggest at any rate to follow a hop-skip-jump approach and go through this lesson's material first.

F4. When something goes a little wrong and helper needed, then easy to get a bit behind with what's going on in the rest of the session (I also agree with point 4, now I've seen it!)

- **A4**: If I could have used my laptop, you could have benefitted from the split screen of day 1. Also we were writing the commands in this document for that purpose. We have not communicated this cleary enough, sorry. At any rate, please do not hesitate to speak up and mention when it is going too fast for you; we might able to accomodate on the spot and one stitch in time may save nine. This can help others too (see F10)!

F5. Would enjoy the opportunity to do a test run where we do an exercise on our own based on a set of instructions and then cross-check answers, I would feel more confident to work with the script independently.

- **A5**. Fair enough. This lesson given in this time frame allows for a few exercises, the less so the more interaction occurs which is just as valuable. On the upside, the curriculum offers plenty of exercises to internalize the lesson as one should.

F6. I agree with point 3. Also, could be helpful if the sessiosn can be separated into mac users and windows users so we can avoid the confusion when commands sometimes don't work

- **A6**. This is not doable alas and, after all, knowing that different operating systems configure certain settings differently (and that this may have a little social cost) is something worth being aware of.

F7. Can be a bit difficult to follow/overwhelming when multiple people are talking at the same time (as the instructions)

- **A7**. Point taken during the class. I hope it went better. I will underline this proactively in my next lessons

F8. Would be good to look at the data before playing with it in bash to see what it looks like. When extracting/moving/manipulating data frames it's easier to understand if you know what the first df looks like.

- **A8**. Point taken. It does not take much time and provides helpful context.

F9. Would be great to have an independent exercise and also a little better understanding of the applications, e.g. how this can help workflows

- **A9**. See A5

F10. sometimes too quick and I can't remember all the codes to get to a specific file to perform command

- **A10**. See A4

2. Git for version control

2.1 Lesson notes

Useful commands (some more information can be found here https://git-scm.com/docs)
git (overall command name)
git init (initlises a empty git repository)
git status (Show the working tree status)
git config (Get and set repository options)
--global (in git config: applies to all git functions)
git commit (Records changes to the repository)
git log (show commit logs - see commit above)
git diff (Show changes between commits - HEAD provides shortcut for latest commit)
git checkout (Switch branches or restores working tree files)
touch (creates files)

```
$ cd../
$ ls
$ mkdir git-lesson
$ cd git-lesson
$ ls
$ git # this command on its own displays lots of help information
$ git  --version
$ git version
$ ls -a
$ git init # creates a hidden empty git repository
$ ls -a
$ ls
$ ls -a .git # best not to do anything with this as it can cause problems
```

Edit file and stage (add to the index)

```
$ nano lines.txt
```

- [In nano type the following]
- first line
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ cat lines.txt
$ git status
$ git add lines.txt
```

Commit the change

```
$ git status
$ git config --global --list
$ git config --global --list | sort

# if you dont have user.email and user.name after the last line of code do:
$ git config --global user.name "[Your name here]"
$ git config --global --list # to check to see if the user name was put in correctly
$ git config --global user.email "[Your email address here]"
$ git config --global --list # to check to see if the user email was put in correctly

# more handy configurations once we are here (one-off)
$ git config --global core.autocrlf true #windows [use this if you are running windows]
$ git config --global core.autocrlf input #mac-or-linux [for mac or linux users]

# resume with the flow
$ git status
$ git commit -m "Commit lines.txt" lines.txt
$ git status
$ git log # provides a log of your recent commits, including commit unique identifer]
```

Repeat the cycle and show differences

```
$ git status
$ nano lines.txt # Lets edit an existing file]
```

- [In nano _add_ the text below]
- second line
- third line
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ git status

$ git diff # shows the difference with versions 'somewhere' in repository
$ git add
$ git diff  # difference was with respect to index/stage
$ git status
$ git commit -m "Add second and third line" lines.txt # commited new version with message
$ git status
$ git diff
$ git log
$ git log --oneline  # only brings up message that was included alongside the commit]
```

DIY

```
$ nano lines.txt
```

- [In nano add the text below]
- fourth line
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ git status
$ git diff
$ git commit -m "Add fourth line" lines.txt # commited new version with message
$ git status
$ git diff
$ git log
```

Committing without a message in the command line, and extended messages

```
$ nano lines.txt
```

- [In nano capitalize the third line]
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ git diff
$ git add
$ git config --global core.editor nano  # one more configuration so we use all the same editor
$ git commit lines.txt # this command now will open up nano so you add a commit message - this is a slightly different way of writing -m "[some text]" as shown above
```

- [In nano add the text below]
- Capitalize the third line
- I did this because I wanted to show one line dissappearing and one new line added in the git diff. This is an instructional move.
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ git status
$ git diff
$ git log
$ git log --oneline
```

Exploring the history

```
$ git diff
$ git diff [Insert reference here from last git log on first line]
$ git diff [Insert reference here from last git log on second line]
$ git diff HEAD # HEAD is the shortcut pointing to the latest commit]
$ git diff HEAD~1 # HEAD is the shortcut pointing to the latest commit, ~1 means the one before the last commit
$ git diff HEAD HEAD~1 lines.txt # the differences between the last and second to last commit for the file lines.txt
$ git diff HEAD~3 HEAD~1 lines.txt
$ git diff HEAD~1 HEAD~3 lines.txt
```

Recall a commit in your working tree

```
$ git log --oneline
$ cat lines.txt
$ git checkout
$ git checkout HEAD~3 lines.txt  # this restores the previous version of the file from
```

```
three commits ago
$ cat lines.txt
$ git status
$ git checkout HEAD lines.txt
$ cat lines.txt
```

Committing directories

```
# working with directories now
$ git status
$ mkdir my folder
$ git status
$ ls
$ touch myfolder/emptyfile.txt
$ ls myfolder
$ cat myfolder/emptyfile.txt
$ git status
$ git status -u
$ git diff
$ git add myfolder/emptyfile.txt]
$ git diff
$ git status
$ git commit -m "add directory with empty file"
$ git log --oneline
```

How to ignore files and directories

```
$ mkdir data
$ touch data/a.dat data/b.dat bigarchive.zip
$ ls -R
$ git status -u
$ nano .gitignore
```

- [In nano add the text below]
- .zip
- data/
- [Ctrl O to save; return to confirm file name; Ctrl X to exit and be back in the shell]

```
$ git status -u
$ git add .gitignore
$ git commit -m "Add gitignore" .gitignore
$ git log --oneline
```

2.2 GitHub accounts

Setting up a link between Github account and your local computer:

# this is preparatory to work with GitHub accounts

```
$ ls -a
$ git branch -M master main
$ git log --oneline
```

Setting up encrypted connections from your computer

```
$ ssh -T git@github.com  # test connection (does it exist already?)
$ ls ~/.ssh
$ echo ~
$  ssh-keygen -t ed25519 -C "[Email address that you used to start your Github account]"
```

[Save this file to the location that the shell has suggested - just please enter until the command is exhausted - we do not use passphrases here for expediency, but they are important]

```
# inspect the location that the shell has suggested
$ cat ~/.ssh/id_ed25519  # this is the private key (like your PIN)
$ cat ~/.ssh/id_ed25519.pub  # this is the public key (like your IBAN)
```

Give public key to GitHub

Go to your Github account and login - https://github.com/login
Click on icon in top right hand corner
Got to Settings
At the left hand side of the new page, click on Add GPG and SSH keys
Add new SSH key
Under Title write: 'SCW Oxford not so secure' because we did not use a passphrase after ssh-keygen
Paste in **public key** from your bash - copy results from last line of code
Confirm new SSH key must be added
Return to Bash

```
# test connection
$ ssh -T git@github.com
```

- [Write yes if prompted with first-time question]

```
$ ssh -T git@github.com [it should now just say welcome]
```

Create repository on GitHub

Go to your Github account and login - https://github.com/login
Click on icon Repositories in top menu and press create new
Fill in repository name **2402-scw-oxf** and description and save
Choose whether it has to be public or private
Do not add a README file this time, since this would complicate the lesson
Copy the quick set-up link using ssh show in the highlighted bar

Connect the remote repository on GitHub with local repository in terminal

$ git remote add origin **git@github.com:[your GitHub username]/2402-scw-**

oxf.git

```
$ git remote -v
$ git push  # fail
$ git push origin main  # pass
$ git log --oneline
```

please check what happened in the web page of the remote repo (refresh first)
back to terminal

Transfer information between local and remote repository

```
$ nano lines.txt
```

- [add one more line]

```
$ git commit -m 'add line in local repository' lines.txt
$ git log
$ git log --oneline
$ git push origin main
$ git remote -v

$ # pull = fetch + merge, as reminder for the future
$ git fetch  # silent command, expected
$ git pull  # fail
$ git pull origin main  # passes without much action
```

Edit the file on GitHub and retrieve the changes in the local repository

go to GitHub
go to file lines.txt
go to edit (pen icon at the right hand side)
add another line to the lines.txt
press ctrl+S or click on green icon *Commit these changes*

- commit message: lines.txt: add one more line in the remote
- description: we are going to pull this change from the local repository
- save as main branch, still

To check in terminal:

```
$ git status
$ git log --oneline
$ git fetch
$ git log --oneline
$ git pull origin main  # there is action
$ cat lines.txt
$ git log --online
```

Create a conflict with simultaneous changes in the local and the remote

```
$ nano lines.txt # add line 'last line in the local repository'
$ git commit -m 'add last line in local repo' lines.txt
$ git log --oneline
```
# DO NOT PUSH THE COMMIT

Go to GitHub
Add line 'last line in remote repository'
Commit as before

Back in the terminal:
```
$ git push origin main  # this fails; asks to make a pull
$ git pull origin main  # fails too
$ git config pull.rebase false  # type this if the error message suggests an action like that; else ignore this one line
$ git pull origin main  # fails because of a "conflict" of lines
$ git status  # seek advice
$ git diff
$ nano lines.txt
```

- # fix manually the conflict = make your choice and remove the decorations Git added

```
$ git add lines.txt # to mark resolution
$ git status  # seek advice
$ git commit  # to finish the merge (complete the default commit message as appropriate)
$ git status
$ git push origin main  # conflict has been resolved, the transfer works now
$ cat lines.txt
$ git log --oneline
```

2.3 Useful resources

- Software Carpentry:
  - this course: https://swcarpentry.github.io/git-novice/index.html
  - reference, glossary and cheat-sheets: https://swcarpentry.github.io/git-gitnovice/reference.html
- Git official website: https://www.git-scm.com/
  - Short videos (4-8'): https://www.git-scm.com/videos
  - Reference manual for commands: https://www.git-scm.com/docs
  - General features: https://www.git-scm.com/about
  - ProGit book (505 pages, open access, with full and partial translations in other languages): https://www.git-scm.com/book
  - Please explore the website.
- Cheat sheets in several languages: https://training.github.com/
- Wikipedia: https://en.wikipedia.org/wiki/Git
- Blog: *How to undo almost everything in Git* (J. Wehner, 8 June 2015) https://github.blog/2015-06-08-how-to-undo-almost-anything-with-git/
- Create a ssh config file (to avoid always working in localadmin) https://betterprogramming.pub/a-step-by-step-walkthrough-to-create-your-first-ssh-config-file-f01267b4eacb

- Applications
    - Git repositories for the Linux kernel: https://git.kernel.org/
    - Git on GitHub: https://github.com/git/
- Cartoons:
    - https://xkcd.com/1296
    - https://xkcd.com/1597

2.4 Feedback

2.4.1 What worked out well (and what you would keep)

1. Run through coding with examples of connecting to git via ssh very heflpul!
2. ...great
3. Very systematic, enjoyed being to work out git.
4. Can follow along on screen easily enough, thanks!
5. I found the illustration of the private and public key very useful.

- **A1-5**. Thanks. I will strive to keep it going like this

2.4.2 What didn't work out well (and how you would improve it)

F1. More time!

- **A1**. See A1 in §1.3.2

F2. ...Yes more time

- **A2**. See A1

F3. Difficult to catch up if someone falls behind a little.

- **A3**. Acknowledged. See A4 in §1.3.2

F4. An introduction to context would be great. Understanding why this is useful and what is important about it? Not sure what the best use for this would be and why git is the best format for this.

- **A4**. Point taken but there was no time for an introduction or a wrap-up. For Git we worked to build up a bottom-up approach by typing along, gathering evidence and bringing that to coherence. I appreciate that people who code less do not foresee the immediate potential of Git. The curriculum provides more context though, and hope this makes do and mend for the in-class lesson

F5. Not 100% clear on when to be using git/exactly what for - as point above says, a bit more context needed I think!

- **A5**. See A4

F6. A bit more time I think on this section.

- **A6**. See A1 and A7

F7. I appreciate that we had limited time, but I would have liked more time with Git (perhaps at the expense of less time on Unix shell commands).

- **A7**. As we all felt, more time would have benefitted us all. Sacrificing Unix Shell for Git would have disappointed others, though. In the end we managed to give a feel of potentials ahead for all three lessons. Please refer to the curriculum to find more of relevant clues in Git. Take note that the Git lesson as covered by the Carpentries leaves out many important issues (like branching), so please manage your expectations accordingly.

3. Programming with R

3.1 Lesson notes

Data for this lesson - r-novice-inflammation-data.zip - https://swcarpentry.github.io/r-novice-inflammation/data/r-novice-inflammation-data.zip

As I progress through the epsiodes I will write and create scripts, which I will share with everyone on this etherpad.

# Meanwhile Giordano and Gurauv put all the R commands here as Nicky types along, E&OE!

- start RStudio
- create a new project in an exisiting directory (`r-novice-inflammation-data`)
  - it should contain a `data` folder
  - save scripts as you write them, in case R crashes

```
?read.csv  # pull the help page on this function
read.csv(file="data/inflammation-01.csv",header=FALSE)  # F stands for FALSE too
#  press ctrl + return to run the command in the line the pointer is on

? is an assignment operatore (keyboard shortcuts Alt + minus for Windows; Option
+ minus for Mac)

# use variable to calculate
1 + 1
weight_kg * 2.2
weight + 100

# change variable
weight_kg <- 57.5

# create a new variable
weight_lb <- weight_kg * 2.2

# change the value of exisiting function
weight_kg <- 55
weight_lb  # show the value in memory
```

```r
# read file into R
dat <- read.csv(file="data/infammation-01.csv", header=FALSE)
dat

# explore the dataset
head(dat)  # the topmost rows, not unlike the Unix Shell
class(dat)  # type of data brought in
dim(dat)  # numbers of rows and columns
view(dat)  # open a dedicated view into the dataframe (click on the little x to close)

# explore the dataset (subsetting)
dat[1, 1]  # value of first row, first column
dat[30, 20]

# print multiple values in one go
dat[c(1, 5, 20), c(15, 25)]   # use the function c() for combine

# print multiple value based on column name
dat[c(1, 5, 20), c("V15", "V25")]   # these column names have been set by R

# call up all information in row 20
dat[20, ]  # omit column value to show all columns (that is, the row)

# call up all information in column 25
dat[, 25]

# groups of data
1:5  # prints a range of numbers
dat[1:10, 25]  # first ten rows of column 25
dat[10, 5:15]

# assign a name to subset of data
patient_1 <- dat[1, ]  # this create a new variable

# maximum inflammation for patients 1 and 2
max(patient_1)
max(dat[2,])

# maximum inflammation on a day
max(dat[,30])

# other summary functions
min(patient_1)
mean(patient_1)  # `mean` will throw an error because patient is a data frame
mean(as.numeric(patient_1))
mean(dat[1, ])  # as verification
median(dat[,7])
sd(dat[,7])
```

```
# summary functions in one command
summary(dat[ ,5:10])

# use the apply() function
?apply  # take note of the meaning of 'margin' for rows or columns

# inflammation for every patient
avg_patient_inflammation <- apply(dat, MARGIN=1, mean)
# inflammation for every day
avg_day_inflammation <- apply(x=dat, MARGIN=2, FUN=mean)  # specifying each variable's name (optional if the expected order is maintaned)
avg_day_inflammation <- apply(dat, 2, mean)  # the same
avg_day_inflammation <- apply(dat, mean, 2)  # throws an error owing to mismatch with expected order

# plot inflammation data
plot(avg_patient_inflammation)
plot(avg_day_inflammation)

# question: call up more information about the argument (str for structure)
str(avg_patient_inflammation)
str(avg_day_inflammation)

# question: use reserved names for variable (not recommended, may work, may not work)
plot <- 155
plot(plot)

# lunch break; we resume at 13

# Episode 2 - Talking functions

# function definition
fahrenheit_to_celsius < - function(temp_F) {
    •  temp_C <- (tempF - 32)    5 / 9
    •  return(temp_C)
}

fahrenheit_to_celsius(32)  # convert freezing point of water
fahrenheit_to_celsius(212)  # convert boiling point of water

# function to convert Celsius degrees to Kelvin
celsius_to_kelvin <- function(temp_C) {
    •  temp_K <- temp_C + 273.15
    •  return (temp_K)
}
```

celsius_to_kelvin(0) # freezing point of water
celsius_to_kelvin(100) # boiling point thereof

# nested functions - the first way
~
fahrenheit_to_kelvin <- function (temp_F) {

- temp_C <- fahrenheit_to_celsius(temp_F)
- temp_K <- celsius_to_kelvin(temp_C)
- return(temp_K)

}

fahrenheit_to_kelvin(32)   # freezing point

# nested functions - the second way
celsius_to_kelvin(fahrenheit_to_celsius(32))

# create a new function to analyse out inflammation dataset
# min, max, mean

min_day_inflammation <- apply(dat, MARGIN=2, min)
plot(min_day_inflammation)

max_day_inflammation <- apply(dat, MARGIN=2, max)
plot(max_day_inflammation)

plot(max_avg_inflammation) # we have that already

analyse <- function (filename) {

- dat < - read.csv(file=filename, header=FALSE)
- # code for mean value
- avg_day_inflammation <- apply(dat, MARGIN=2, mean)
- plot(avg_day_inflammation)
- # code for min value
- min_day_inflammation <- apply(dat, MARGIN=2, min)
- plot(min_day_inflammation)

- # code for max value
- max_day_inflammation <- apply(dat, MARGIN=2, max)
- plot(max_day_inflammation)
- # we don 't use a return function here; this function is content with plotting

}

analyse("data/inflammation-02.csv")
analyse("data/inflammation-03.csv")

```
# for loops
# example using character data
best_practice <- c("Let", "the", "computer", "do", "the", "work")
best_practice

# function to print these words (unwieldy)
print_words <- function(sentence) {

    •  print(sentence[1])
    •  print(sentence[2])
    •  print(sentence[3])
    •  print(sentence[4])
    •  print(sentence[5])
    •  print(sentence[6])

}
print_words(best_practice)

# used this function with less words
best_practice[-6]
best_practice

# use a shortened version in the function
print_words(best_practice[-6])  # fails

# create a function using for loops
print_words <- function(sentence){

    •  for (word in sentence) {
           •  print(word)
    •  }

}

print_words(best_practice)
print_words(best_practice[-6])  # works

# list files in a directory, by file extension
list.files(path='data', pattern='.csv')

# same, listing files by file name
list.files(path='data', pattern='inflammation')

# same, outputting the file path
list.files(path='data', pattern='.csv', full.name=TRUE)

# use for loop to run `analyse` function through a list of files
# the pattern argument contains regular expressions: see link in §3.2
filenames <- list.files(path='data', pattern = "inflammation-[0-9]{2}.csv",
full.name=TRUE)
```

filenames <-filenames[1:3] # create the list the for iterates over

for (f in filenames) {

- print(f)
- analyse(f)

}

# digression on extracting parts of a string
library(tidyverse)
?str_extract
# look for package stringr
# look for cheatsheets, they can be really helpful
# end of digression

# we resume at 14:40

# create a function to analyse a group of files
analyse_all <- function(folder="data", pattern) {

- # Runs the function analyse in each file in the given folder
- # and with the required pattern
- filenames <- list.files(path=folder, pattern=pattern, full.name=TRUE)
- for (f in filenames) {
    - analyse(f)
- }

}

analyse_all(folder="data", "inflammation")

# questions and digressions
# we changed the plotting so that they display a main title
# end of digression

# saving plots to pdfs
pdf("inflammation-01.pdf")
analyse("data/inflammation-01.csv")
dev.off() # sign off

# questions and digressions
# how to bundle functions in one file and load that file
# end of digression

# exploring the if-else construct
num <- 42
if (num > 100) {

- print("greater")

} else {

- print("not greater")

```
}
print("done")

# apply if else to plots based on data
plot_dist(dat[,10], threshold=10)  # day 10 data

# exercise - function for drawing boxplot or stripchart
?boxplot
?stripchart

# the batter of the current notetaker is low
# continued ..
plot_dist <- function(x, threshold) {
  if (length(x) > threshold) {
    boxplot(x)
    } else {
      stripchart(x)
      }
}

plot_dist(dat[,10], threshold = 10)
plot_dist(dat[1:5, 10], threshold = 10)

output <- NULL
is.NULL(output)
!is.NULL(output)

# edit the analyse function to incorporate an if else statement
analyse <- function(filename, output=NULL) {
  # Plots the average, min and max inflammation over time
  # Input - filename - character string of the location fo the csv file
  # Output - charater string of the pdf file that we are saving to
    if (!is.null(output)) {
       pdf(output)
    }
    dat <- read.csv(file = filename, header = FALSE)
    avg_day_inflammation <- apply(dat, 2, mean)
    plot(avg_day_inflammation)
    min_day_inflammation <- apply(dat, 2, min)
    plot(min_day_inflammation)
    max_day_inflammation <- apply(dat, 2, max)
    plot(max_day_inflammation)
    if (!is.null(output)) {
    dev.off()
    }
}
```

```
analyse("data/inflammation-01.csv")

# rerun code with filenames
analyse("data/inflammation-02.csv", output = "inflammation-02.pdf")

# create a directory
dir.create("results")

# point the pdf output to this directory
analyse("data/inflammation-02.csv", output = "results/inflammation-02.pdf")

# incorporate pdf export into analyse_all function
f <- "inflammation.csv"
sub("csv","pdf",f)

# apply this to the file path function
file.path("results", sub("csv","pdf",f))

filenames <- list.files(path="data", pattern = "inflammation-[0-9]{2}.csv",
full.name=TRUE)

analyse_all <- function(pattern) {
  # Directory name of the folder contained in the data
  data_dir <- "data"
  # Directory name of the folder for the results
  results_dir <- "results"
  filenames <- list.files(path = data_dir, pattern = pattern, full.names = FALSE)
  # Runs the function analyse for each file in the current working directory
  # that fits the pattern
  for (f in filenames) {
     pdf_name <- file.path(results_dir, sub("csv","pdf",f))
     analyse(file.path(data_dir, f), output = pdf_name)
    }
}

analyse_all("inflammation-[0-9]{2}.csv")
```

3.2 Useful resources

- Software Carpentry
  - This course - https://swcarpentry.github.io/r-novice-inflammation/index.html
  - Basic operations - https://swcarpentry.github.io/r-novice-inflammation/reference.html
- Website - R - https://www.r-project.org/
- Website - R Studio (Posit) website - https://posit.co/products/open-source/rstudio/
- Website - Tidyverse - https://www.tidyverse.org/ [A really useful suite of packages]
- Website - Regular expressions as used in R - https://cran.r-project.org/web/packages/stringr/vignettes/regular-expressions.html
- Book - R for Data Science - https://r4ds.had.co.nz/

- Book - Hands on Programming with R - https://rstudio-education.github.io/hopr/
- Blog - Connecting RStudio to Github - https://happygitwithr.com/rstudio-git-github.html
- Blog - Six Reasons why you should learn R for Data Science - https://www.dataquest.io/blog/three-mighty-good-reasons-to-learn-r-for-data-science/
- Book chapter - Starting your R project - https://bookdown.org/daniel_dauber_io/r4np_book/starting-your-r-projects.html [includes some good practice for folder structures in R]
- Other Carpentries courses for R
    - Data Analysis and Visualization with R for Social Scientists https://datacarpentry.org/r-socialsci/
    - Introduction to R for Geospatial Data - https://datacarpentry.org/r-intro-geospatial/
    - Introduction to Geospatial Raster and Vector Data with R - https://datacarpentry.org/r-raster-vector-geospatial/
    - Data Analysis and Visualization in R for Ecologists - https://datacarpentry.org/R-ecology-lesson/

Troubleshooting

If you recieve an error when you load packages saying that RTools is not installed, you may need to install it seperately outside of RStudio.
To do that go here https://cran.r-project.org/bin/windows/Rtools/rtools43/rtools.html and download the installer to download the latest version (Currently 4.3)
PLease accept all deafults when downloading this package.
Make sure that R and Rstudio are closed when you downlaod RTools and once installed start a new session in RStudio to reinatlise. Then try the line of code 'has_rtools(debug = FALSE)' - this will check to see if RTools is saved on your computer (if it returns TRUE thats good).
Link to documentation - https://search.r-project.org/CRAN/refmans/pkgbuild/html/has_rtools.html#:~:text=has_rtools()%20determines%20if%20Rtools,find%20out%20where%20it's%20installed.
Once that is all complete try again to downloade the packages in R.

This is a useful step by step if you need to download RTools - https://jtleek.com/modules/01_DataScientistToolbox/02_10_rtools/#1

Other useful resources (Q&A, etc)

- RTools not linked to RStudio https://stackoverflow.com/questions/19885381/rtools-not-being-detected-by-r
- Define all functions in one .R file, call them from another .R file. How, if possible? https://stackoverflow.com/questions/13548266/define-all-functions-in-one-r-file-call-them-from-another-r-file-how-if-pos

3.3 Feedback - Morning (only R)

3.3.1 What worked out well (and what you would keep)

2. Coding along
3. Very systematic, and easy to follow. Grateful for all the resources made available and the notes taken on the ongoing sessions.
4. Presenters are clear and engaging :)
5. Good examples, great idea to use a "real" dataset people can look at

3.3.2 What didn't work out well (and how you would improve it)

1. More time! Fell a little behind during the GIT catchup and didn't really follow through to the end, but not comfortable asking for help due to lack of time.
A. See the answer on line 188 - there is great deal on content in Carpentries courses that is more than can be covered in 1-2 days. Wd could consider organising a longer course (4-5 days) although it is difficult to ensure that all leaners can attend all days. Hopefully we have provided enough context and introduction that learners can follow through the remaining lessons by themselves. Please do contact us if you have any questions!
2. Timekeeping -- I came for the R session only, but the previous day spilled over. I guess the solution is to cut something from the first day?
A. Unfortunetely due to technical issue the first day we overran slightly into day 2. This couldnt be helped I'm afriad but hopefully we provided enought content on day 2 as well.
3. ...

3.4 Feedback - Afternoon

3.4.1 What worked out well (and what you would keep)

1. Able to keep attention (despite intense ADHD problems)! Very well explained.
2. ...Attempting to solve coding problems , liked seeding problem solving in real time
3. The progressive approach, I liked the small steps that built to a reasonably powerful method
4. Great course,
5. well paced
6. Engrossing, good to have the live experience

3.4.2 What didn't work out well (and how you would improve it)

1. Could have used an introduction to some basic concepts (had to look up "vectors" and "strings" because of unfamiliarity with coding and softwares for quant
A. Some of the other R carpemntries course do provide some of this context (I'd recommend https://datacarpentry.org/R-ecology-lesson/index.html)
2. As so much of modern programming is 'standing on the shoulders of giants', I would like to see more on using 3rd party packages. How to install them, import them, how the namespacing works, etc.
A. I agree - unfortunely there was not enough time in one day to expand beyond base R, which provides the foundation for learning R. Hopefully the organisers will support some further R Carpentries courses in the future.
3. If I were being picky, a room with windows would be nice :)
A. Fair enough! Especially with the unseasonable weather we had :) Other than that I hope the teaching

locations were suitable!
4. Longer on R! Signposting to courses for next level? e.g. supported work with our own datasets?
A. This could have been a longer course and I hope the organisers take this interest on board and offer some other R courses in the furture. RROx may also be able to offer supported work and specific requirements!