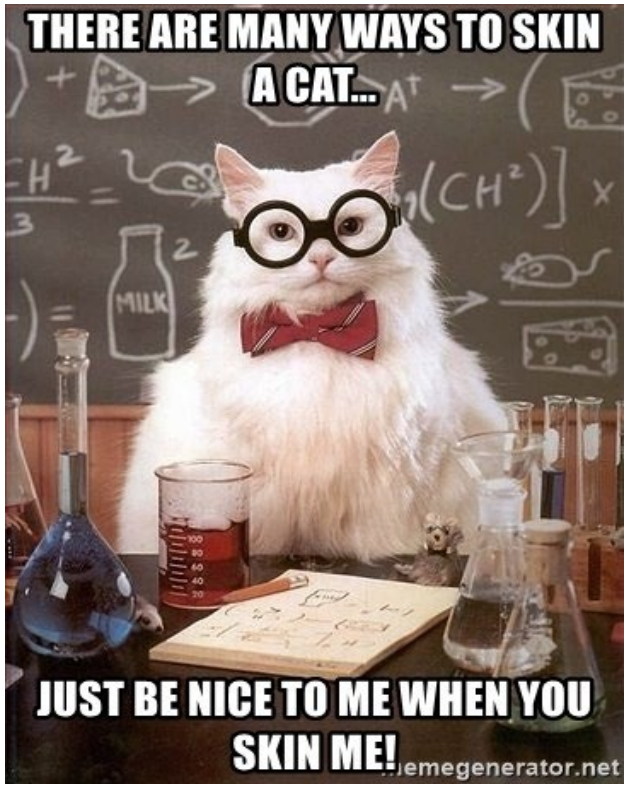
latr-meetups

# PROJECT SET-UP AND WORKFLOWS

The critical first step!

# latr-meetups

- Many of the principles and ideas today are not specific to r – they can be used across any programming\\statistical software, or even more broadly to computing in general



latr-meetups

RStudio®

# R STUDIO PROJECTS

- A place for everything in your project to live
- Like a storage container
- An R Studio project creates a ‘working directory’

latr-meetups

# HOW TO SET UP YOUR R PROJECT

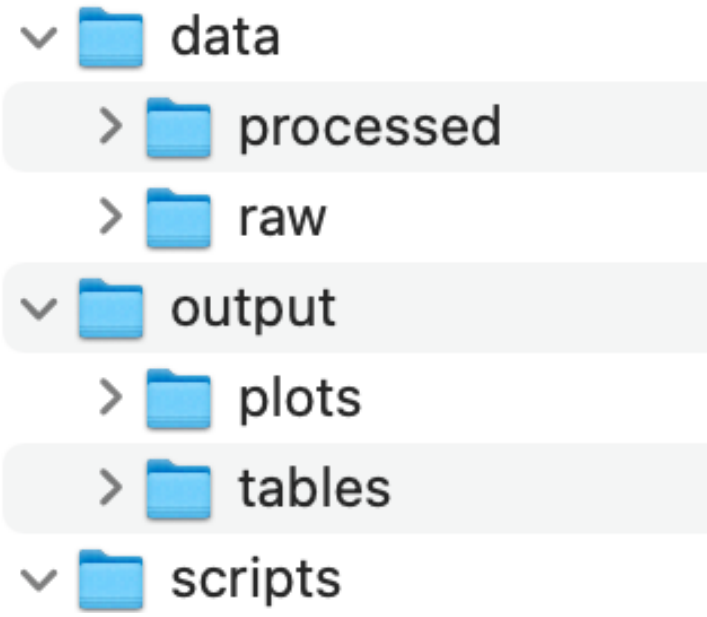
- Show example

# THE NEXT STEP – ORGANIZING YOUR DIRECTORY

- We generally have 3 different aspects of a project:
  - Inputs
    - Data (csv files, spreadsheets, individual's data, databases etc)
  - Outputs
    - Results
    - Tables
    - Plots\\graphs
    - Reports\\Papers\\Web content etc
  - Scripts to convert the input to an output

# ORGANISING YOUR DIRECTORY

- Project title
  - Data
    - Raw
    - Processed
  - Output
    - Tables
    - Plots
  - Scripts



# ORGANISING YOUR R SCRIPTS

- Breaking down your process into separate (repeatable) steps
- 1. Loading your data
- 2. Cleaning your data
- 3. Analysing your data

latr-meetups

# ORGANISING YOUR SCRIPTS

**1_load.r –** load your packages\\libraries\\data and scripts

**2_functions.r** – save any special functions you are using or have written

**3_clean.r** – transform the raw data into a useable format ready for analysis (usually spend about 80% of your time here…)

**4_do.r** – where the good stuff happens – analysis, summarising, outputs

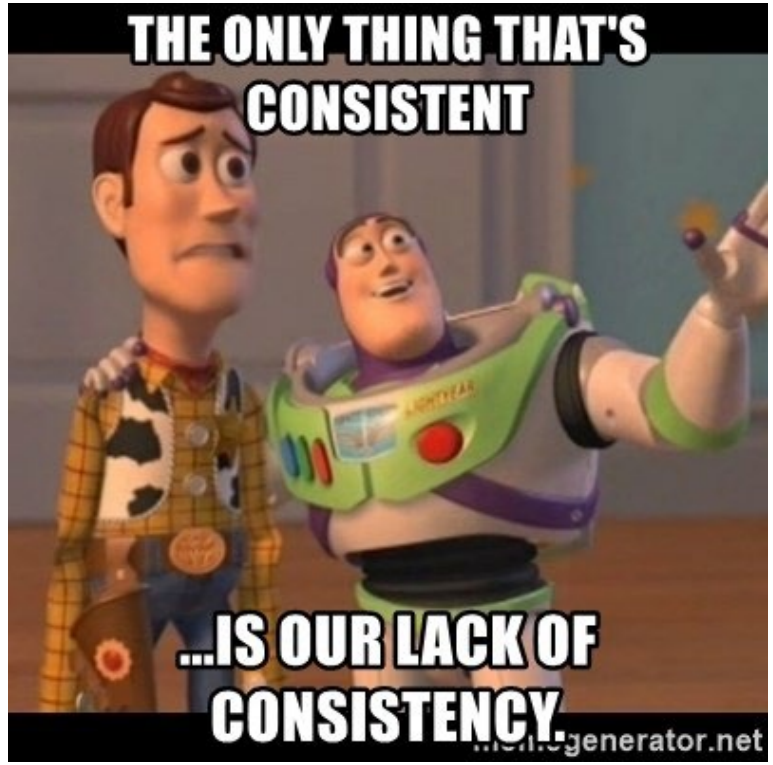
# ORGANISING YOUR SCRIPTS

- Show example in R studio

# NAMING CONVENTIONS AND VERSION CONTROL

- If nothing else **BE CONSISTENT**

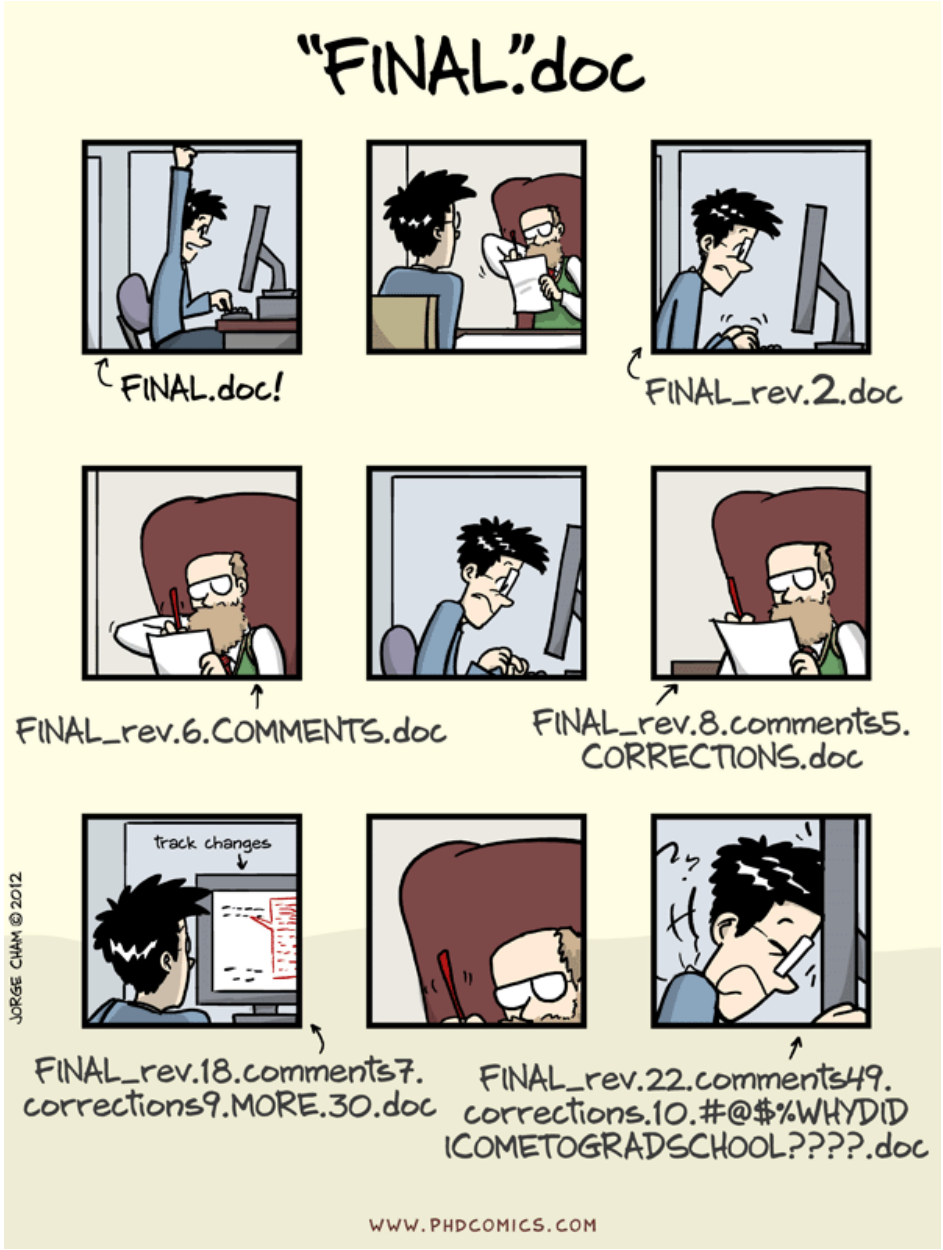
- Save yourself from headaches, overwrites, future problems, data loss, grumpy colleagues and grumpy you!



# NAMING

- Stick to all lower case
- Replace spaces with "_"
- Avoid special symbols ($, @, %, #, &, *, (, ), !, \\)
- Simple and short explanatory names
- Name things consistently
  - E.g. l_knee_1, l_knee_2, and not: ~~left_knee_1, left_knee1~~

# NEVER, EVER, EVER BE TEMPTED TO CALL SOMETHING "FINAL" – IT LIKELY WON'T BE!



latr-meetups

# VERSION CONTROL

- Great habit to get into
- Common methods include a ‘semantic’ number system
- E.g. v1.0, 1.1, 1.2, 2.0
- Major changes\\revisions = whole number
- Minor changes\\revisions = decimal number
- Date can also be added - but be consistent!!
  - ISO 8601 Standard is YYYY-MM-DD i.e. 2022-06-28

# VERSION CONTROL WITH R

- Data – the raw data should NEVER be edited, but having the version\\date is very important incase you need to update later
- By breaking up your project into the different parts you often won't need to use version naming with your scripts
- For advanced version control there are automated\\semi-automated processes that track your changes as you make them – beyond the scope of today but something to think about!

# THE LAST PART – PLANNING YOUR CODE!

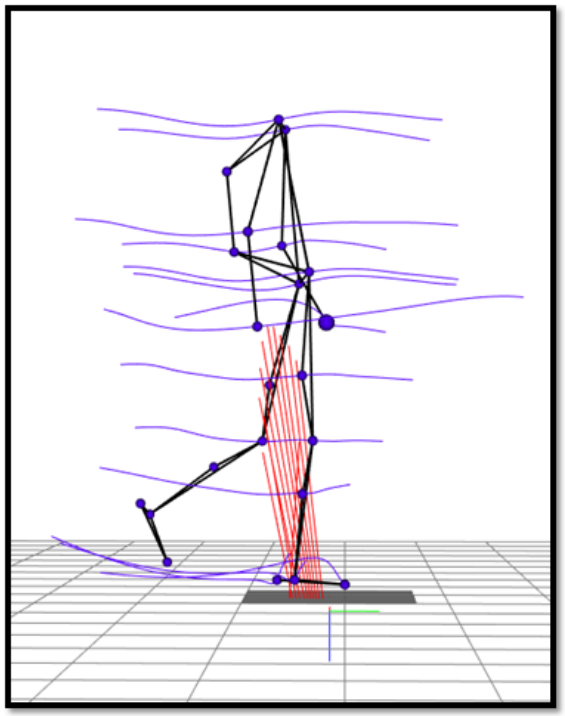
- Once your workflow is set up it's time to start coding!
  - … but first more planning…

- It can be really helpful to first dot point\\write out what steps you will need to take to get from start to finish.
- This is also the first steps to annotate your code – a very important step for reproducibility

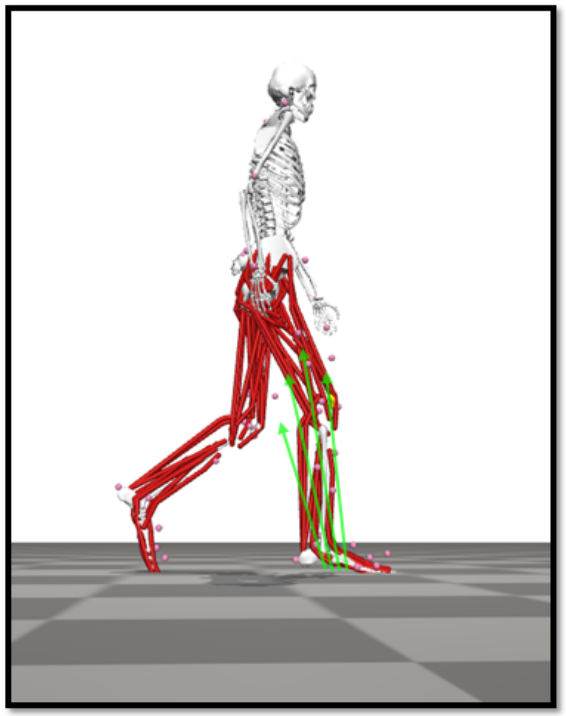
# PLANNING YOUR CODE

- Show example in R studio

# PROJECT SET UP: EXAMPLE

Gait experiment

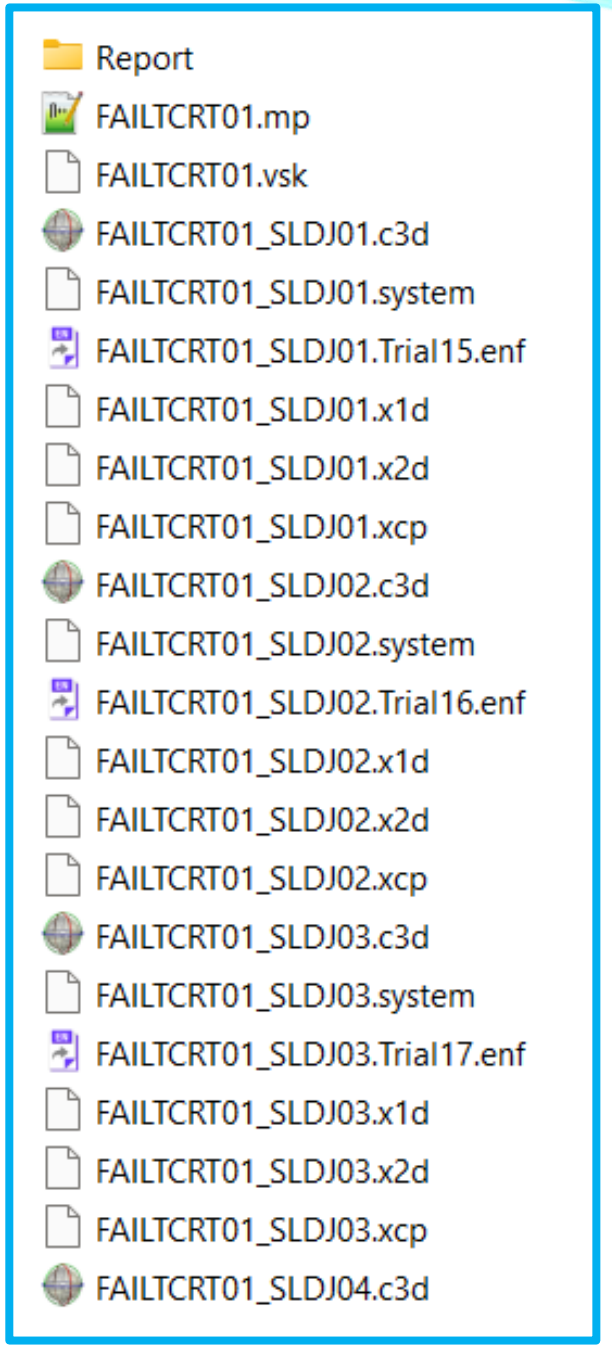


Musculoskeletal modelling

OpenSim

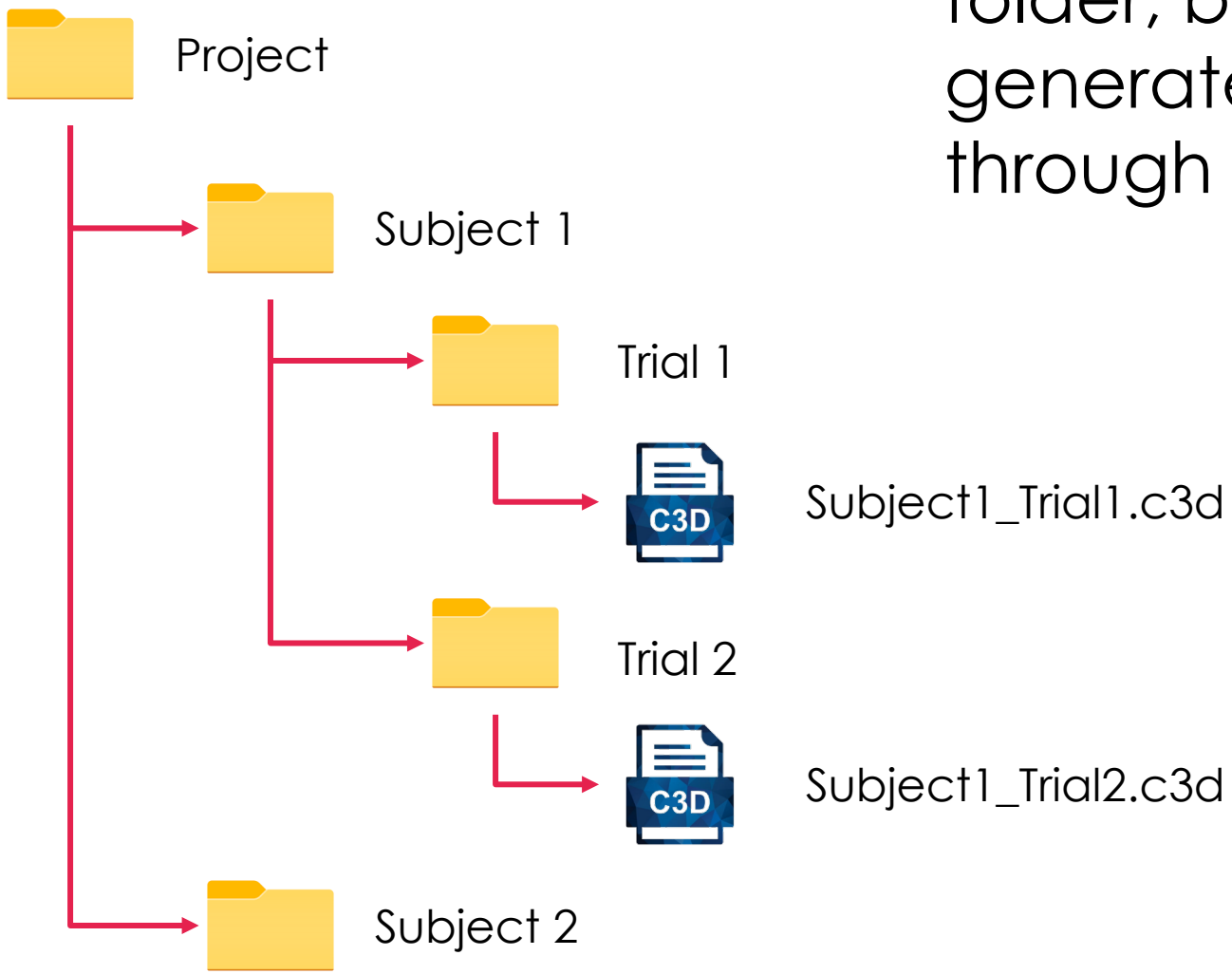
# Data files from 1 subject, 1 session

- For modelling, I generally only need C3D files.



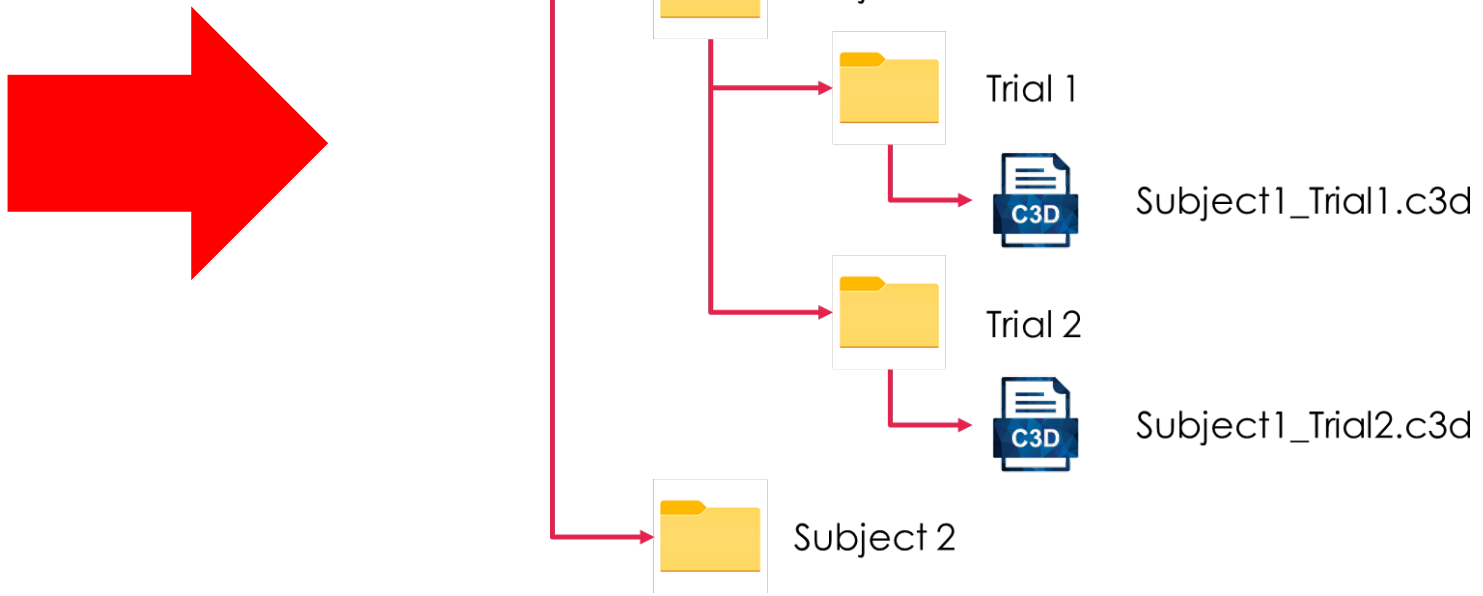

+ 50 more files…

- I need each C3D file in its own folder, because each C3D file will generate many more files as it runs through the modelling pipeline.



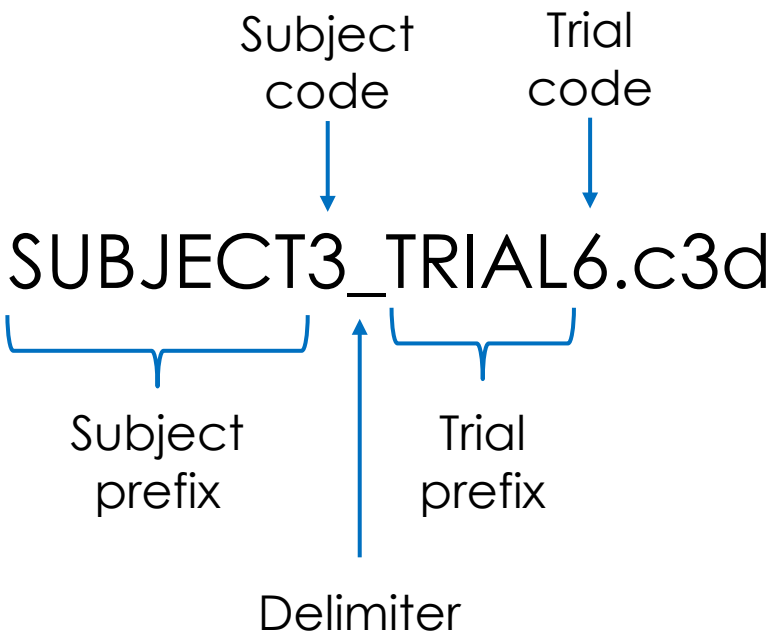
# latr-meetups

- Report
- FAILTCRT01.mp
- FAILTCRT01.vsk
- FAILTCRT01_SLDJ01.c3d
- FAILTCRT01_SLDJ01.system
- FAILTCRT01_SLDJ01.Trial15.enf
- FAILTCRT01_SLDJ01.x1d
- FAILTCRT01_SLDJ01.x2d
- FAILTCRT01_SLDJ01.xcp
- FAILTCRT01_SLDJ02.c3d
- FAILTCRT01_SLDJ02.system
- FAILTCRT01_SLDJ02.Trial16.enf
- FAILTCRT01_SLDJ02.x1d
- FAILTCRT01_SLDJ02.x2d
- FAILTCRT01_SLDJ02.xcp
- FAILTCRT01_SLDJ03.c3d
- FAILTCRT01_SLDJ03.system
- FAILTCRT01_SLDJ03.Trial17.enf
- FAILTCRT01_SLDJ03.x1d
- FAILTCRT01_SLDJ03.x2d
- FAILTCRT01_SLDJ03.xcp
- FAILTCRT01_SLDJ04.c3d



To automate with scripting, file naming must be consistent and follow a systematic convention.

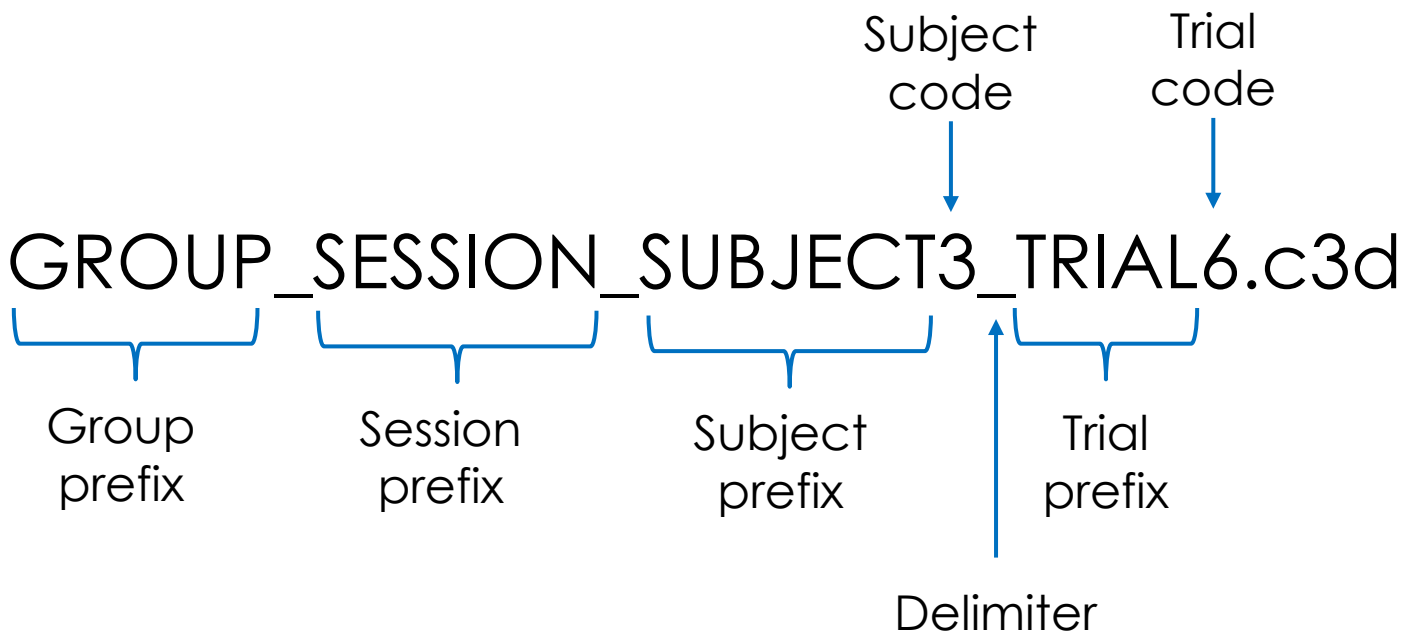
**Mandate this in your experimental protocols!**




FAILTCRT09_WALK11.c3d

FAILT11_SDP09.c3d

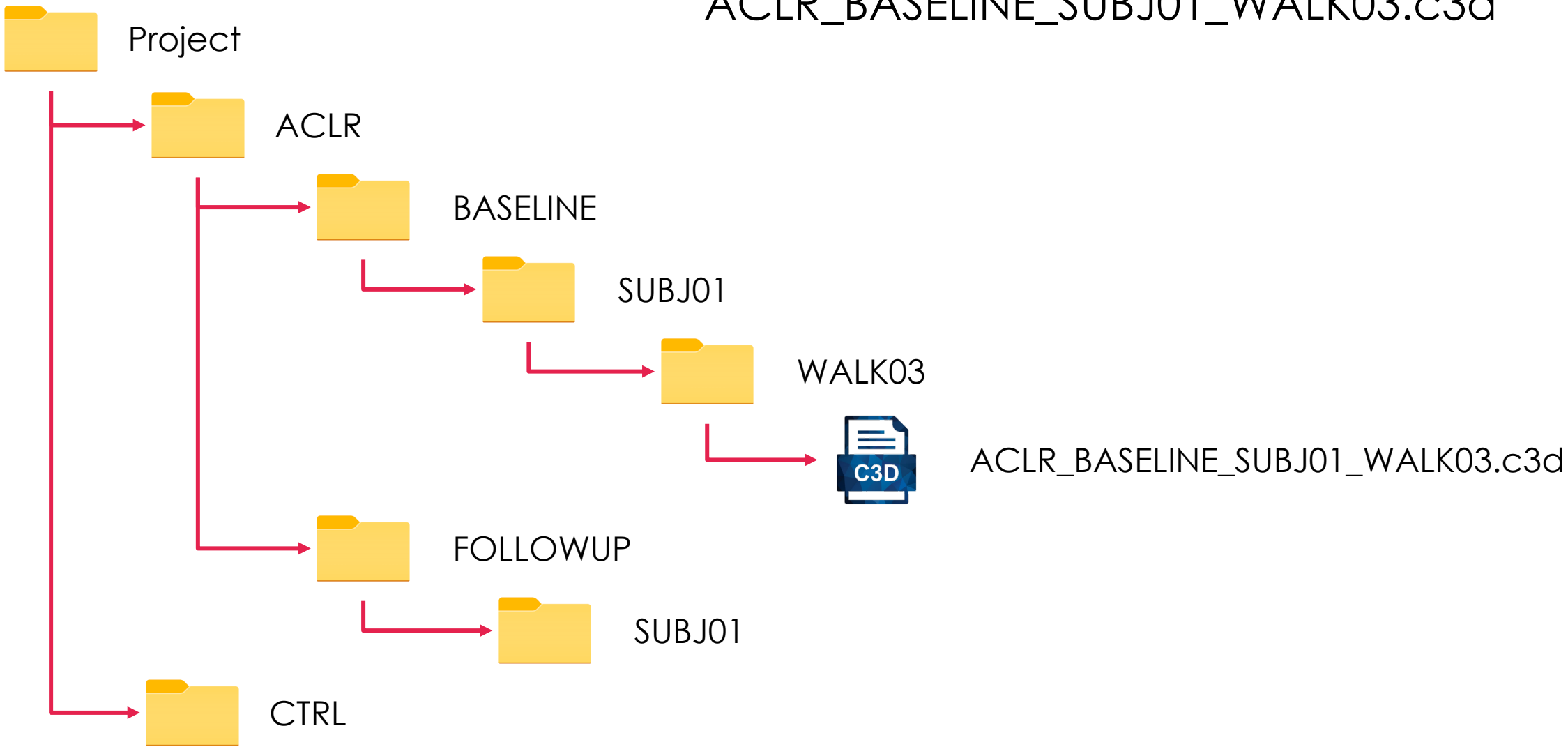
Sometimes the file names need to encode more information.




ACLR_BASELINE_SUBJ01_WALK03.c3d

CTRL_FOLLOWUP_SUBJ01_SLDJ03.c3d

ACLR_BASELINE_SUBJ01_WALK03.c3d



# 1. Get the list of all the C3D files in the raw database

```
# root folder
rootfolder <- "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\inputDatabase\\"

# find all the C3D files in the root folder, including subdirectories
fileslist <- list.files(rootfolder, pattern=".c3d", recursive=TRUE, full.names=TRUE)
```

## Output: a list of full paths to the files

```
…
…
…
[762] "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\inputdatabase\\FAILT89\\New Session\\FAILT89_SLDJ06.c3d"
[763] "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\inputdatabase\\FAILT89\\New Session\\FAILT89_SLDJ07.c3d"
[764] "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\inputdatabase\\FAILT89\\New Session\\FAILT89_SLDJ08.c3d"
[765] "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\inputdatabase\\FAILT89\\New Session\\FAILT89_SLDJ09.c3d“
…
…
…
```

# 2. Split the C3D file name into its components

```
# just get the file names, discard the path to the filename
filenames <- fileslist %>% basename()
```

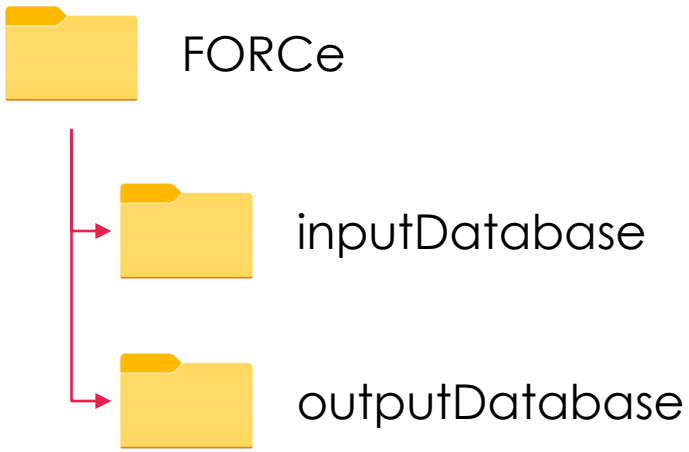
The %>% operator is called a pipe. Piping is a convenience feature of R that allows you to string together a number of functions by using the output of one function as the first input to the next. This can help with readability of code.

## Output: a list containing file names without the folder path

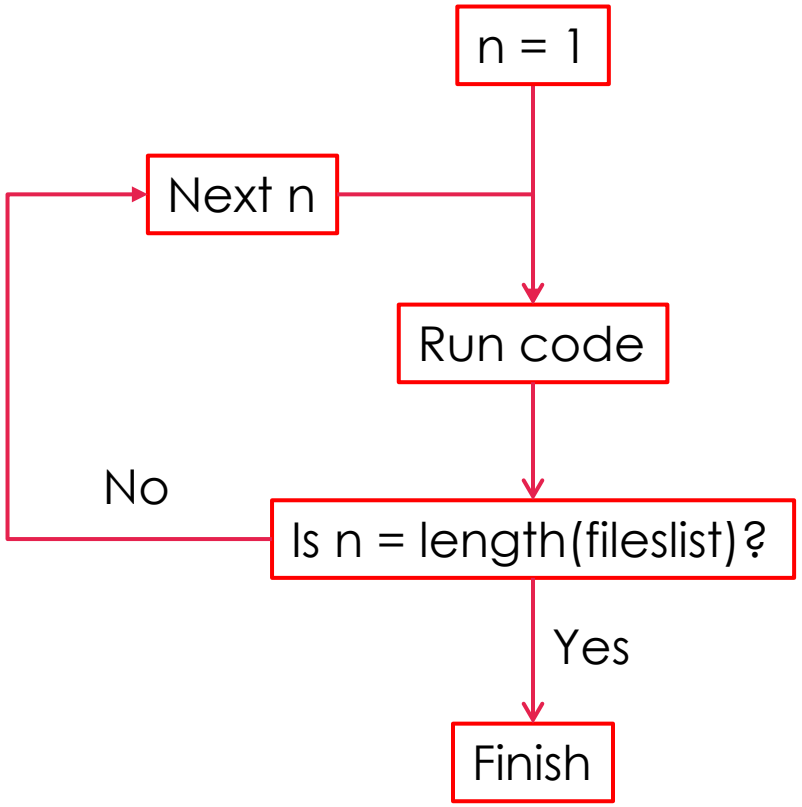
```
[1] "FAILT01_SLDJ02.c3d"   "FAILT01_SLDJ03.c3d"   "FAILT01_SLDJ04.c3d"   "FAILT01_SLDJ05.c3d"
[5] "FAILT01_SLDJ06.c3d"   "FAILT01_SLDJ07.c3d"   "FAILT01_Walk01.c3d"   "FAILT01_Walk02.c3d"
[9] "FAILT01_Walk03.c3d"   "FAILT02_SLDJ06.c3d"   "FAILT02_SLDJ08.c3d"   "FAILT02_SLDJ09.c3d"
…
…
…
```

## 3. Create the root folder to the new database

```
# root folder path
rootfolder <- "C:\\Users\\Prasanna\\Documents\\data\\FORCe\\outputDatabase"



# create it if it doesn't exist
if (!dir.exists(rootfolder)) {
  dir.create(rootfolder)
}
```



3. Iterate through the list of files and create the new database (note: we need to use both `fileslist` and `filenames`)

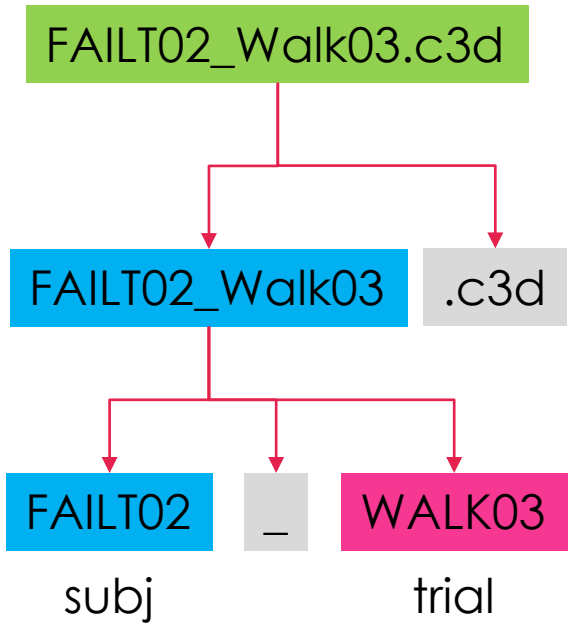



```
# loop through the list of file names
for (n in 1:length(fileslist)) {


  # CODE TO BE EXECUTED ON EACH LOOP GOES HERE


}
```

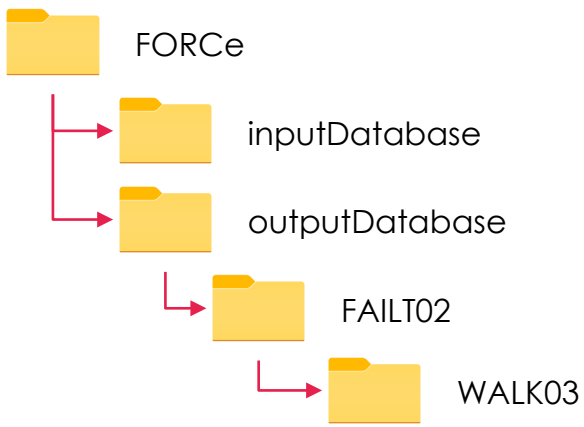
```
# loop through the list of file names
for (n in 1:length(fileslist)) {

  # split the file name into tokens based on the delimiter
  filetoks <- filenames[n] %>% str_split(pattern="\\.c3d") %>%
                  sapply("[[", 1) %>% str_split(pattern="_")

  # get the subject and trial, convert to upper case for consistency
  subj = toupper(filetoks[[1]][1])
  trial = toupper(filetoks[[1]][2])

  # create the subject folder if it doesn't exist
  subjfolder = file.path(rootfolder, subj)
  if (!dir.exists(subjfolder)) {
    dir.create(subjfolder)
  }

  # create the trial folder if it doesn't exist
  trialfolder = file.path(subjfolder, trial)
  if (!dir.exists(trialfolder)) {
    dir.create(trialfolder)
  }

  # copy the C3D file from the raw database to the new database
  newfilepath = file.path(trialfolder, toupper(filenames[n]))
  file.copy(fileslist[n], newfilepath)

}
```

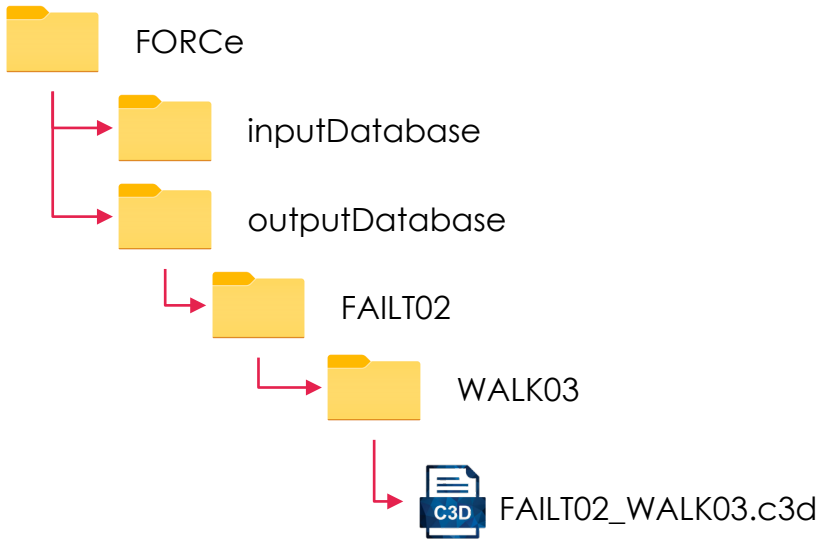
```
# loop through the list of file names
for (n in 1:length(fileslist)) {

  # split the file name into tokens based on the delimiter
  filetoks <- filenames[n] %>% str_split(pattern="\\.c3d") %>%
                  sapply("[[", 1) %>% str_split(pattern="_")

  # get the subject and trial, convert to upper case for consistency
  subj = toupper(filetoks[[1]][1])
  trial = toupper(filetoks[[1]][2])

  # create the subject folder if it doesn't exist
  subjfolder = file.path(rootfolder, subj)
  if (!dir.exists(subjfolder)) {
    dir.create(subjfolder)
  }

  # create the trial folder if it doesn't exist
  trialfolder = file.path(subjfolder, trial)
  if (!dir.exists(trialfolder)) {
    dir.create(trialfolder)
  }

  # copy the C3D file from the raw database to the new database
  newfilepath = file.path(trialfolder, toupper(filenames[n]))
  file.copy(fileslist[n], newfilepath)

}
```

```
# loop through the list of file names
for (n in 1:length(fileslist)) {

  # split the file name into tokens based on the delimiter
  filetoks <- filenames[n] %>% str_split(pattern="\\.c3d") %>%
                  sapply("[[", 1) %>% str_split(pattern="_")

  # get the subject and trial, convert to upper case for consistency
  subj = toupper(filetoks[[1]][1])
  trial = toupper(filetoks[[1]][2])

  # create the subject folder if it doesn't exist
  subjfolder = file.path(rootfolder, subj)
  if (!dir.exists(subjfolder)) {
    dir.create(subjfolder)
  }

  # create the trial folder if it doesn't exist
  trialfolder = file.path(subjfolder, trial)
  if (!dir.exists(trialfolder)) {
    dir.create(trialfolder)
  }

  # copy the C3D file from the raw database to the new database
  newfilepath = file.path(trialfolder, toupper(filenames[n]))
  file.copy(fileslist[n], newfilepath)

}
```



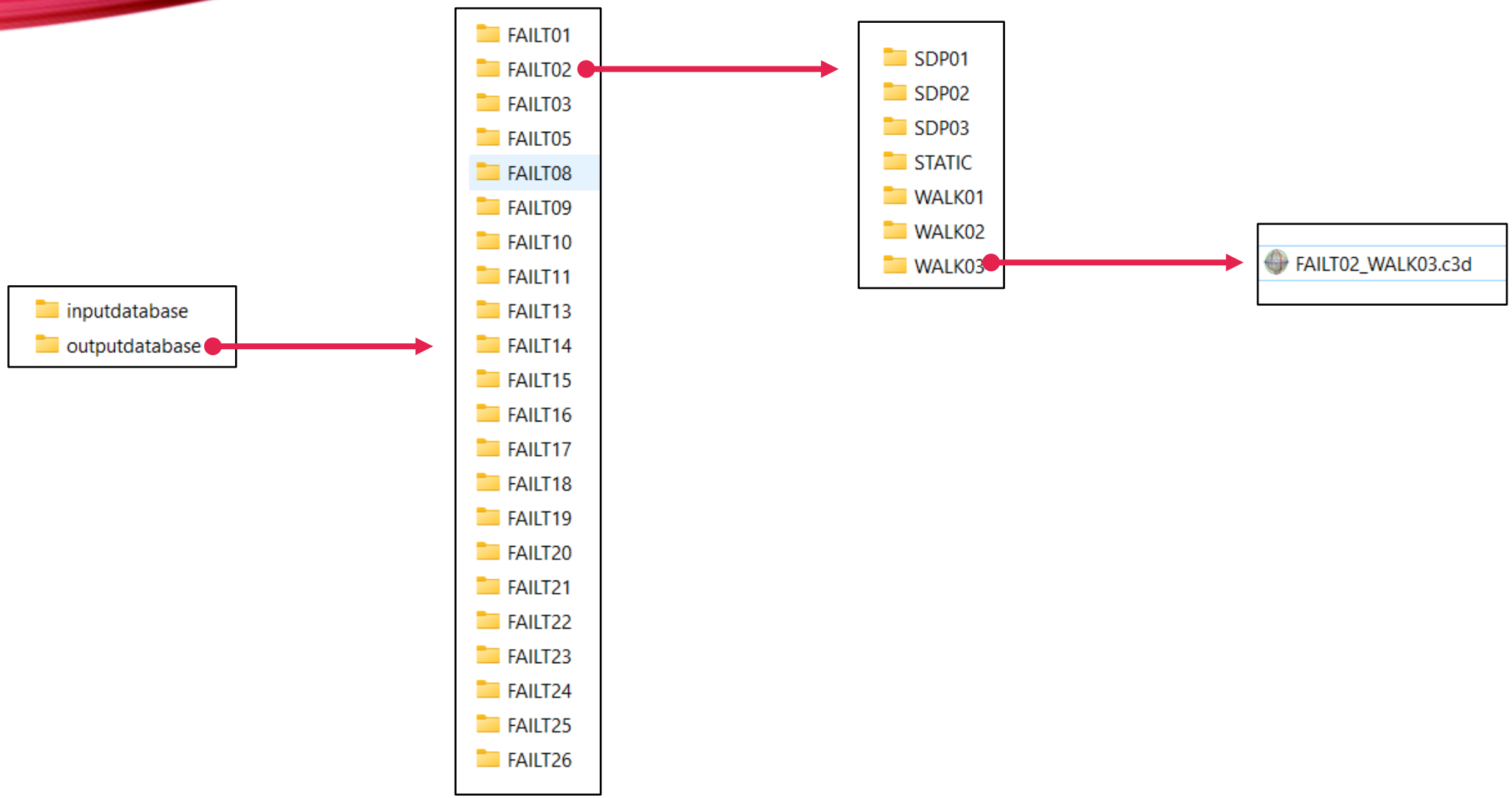
```
# loop through the list of file names
for (n in 1:length(fileslist)) {

  # split the file name into tokens based on the delimiter
  filetoks <- filenames[n] %>% str_split(pattern="\\.c3d") %>%
                  sapply("[[", 1) %>% str_split(pattern="_")

  # get the subject and trial, convert to upper case for consistency
  subj = toupper(filetoks[[1]][1])
  trial = toupper(filetoks[[1]][2])

  # create the subject folder if it doesn't exist
  subjfolder = file.path(rootfolder, subj)
  if (!dir.exists(subjfolder)) {
    dir.create(subjfolder)
  }

  # create the trial folder if it doesn't exist
  trialfolder = file.path(subjfolder, trial)
  if (!dir.exists(trialfolder)) {
    dir.create(trialfolder)
  }

  # copy the C3D file from the raw database to the new database
  newfilepath = file.path(trialfolder, toupper(filenames[n]))
  file.copy(fileslist[n], newfilepath)

}
```






```
# loop through the list of file names
for (n in 1:length(fileslist)) {

  # split the file name into tokens based on the delimiter
  filetoks <- filenames[n] %>% str_split(pattern="\\.c3d") %>%
                  sapply("[[", 1) %>% str_split(pattern="_")

  # get the subject and trial, convert to upper case for consistency
  subj = toupper(filetoks[[1]][1])
  trial = toupper(filetoks[[1]][2])

  # create the subject folder if it doesn't exist
  subjfolder = file.path(rootfolder, subj)
  if (!dir.exists(subjfolder)) {
    dir.create(subjfolder)
  }

  # create the trial folder if it doesn't exist
  trialfolder = file.path(subjfolder, trial)
  if (!dir.exists(trialfolder)) {
    dir.create(trialfolder)
  }

  # copy the C3D file from the raw database to the new database
  newfilepath = file.path(trialfolder, toupper(filenames[n]))
  file.copy(fileslist[n], newfilepath)

}
```

C:\Users\Prasanna\Documents\data\FORCe



Try it yourself!

Play around with the code using your own data:

https://github.com/latr-meetups/latr/blob/main/code/build_database/build_database.R

(Locate the Raw button on the right of the screen,
Press "Alt" on your keyboard and left-click on your mouse at the same time to download)

**ASK LATR** – BYO PROBLEM FOR US TO GO THROUGH!