Best Practice for Programming in R

ARUSHI GARG

FEBRUARY 2023

WWW.ARUSHIGARG.COM

ARUSHIGARG14@GMAIL.COM

TWITTER.COM/ARUSHIGRG

MASTODON: @ARUSHIG@SCIENCES.SOCIAL

Setting expectations

❑ Primarily aimed at academics, students or those starting out to code.

If any of you has, however, briefly been exposed to a more formal way of programming, due to work in industry or package development, it might be that many of the major points here are already known to you and redundant

❑ Major focus on simple quick things that you can do to drastically change the way your code looks and improve use, sharing and readability

❑ Primary aim is to improve readability of your code. This is not meant to teach how to code better ☺

❑ R, particularly R Studio perspective. Principles can transfer to other languages, but practices would need to be modified – in some cases – drastically

Invitation

- ❑ Open up a piece of your own code as you go through this document.
- ❑ As we go through examples of what to do and what not to do, try to see what you could have done differently in your own code
- ❑ If you have questions, feel free to interrupt.

My Background

PhD candidate in Psycho- / Neurolinguistics

DONDERS INSTITUTE

Radboud University

IN·DEI·NOMINE·FELICITER

Master in Cognitive Science

Institute of Cognitive Science

UNIVERSITÄT OSNABRÜCK

Software Developer (SAP ABAP)

accenture

SAP®

So What!!?

So What!!?

How I felt about coding practices in academic research when I first moved here from industry

Software Industry & Coding Practices

- Company or project standards
  - Code formatting (including things as specific as indentation)
  - Naming conventions
  - Modularity conventions
  - Version control
  - Thorough documentation
    - Technical specifications
    - Functional specifications
  - Self-reviews; Self-tests
  - Peer-reviews (not like in academia, but review of the code)
  - Rigourous testing at different levels; technical as well as functional testing

Academia & Coding Practices

- Company or project standards
  - Code formatting (incl things as speci indentation)
  - Naming conventio
  - Modularity conventi
  - Version control
  - Thorough documentatio
    - Technical specifications
    - Functional specificatio
  - Self-reviews; Self-test
  - Peer-reviews (not like in academia, but review of the code)
  - Rigourous testing at different levels; technical as well as functional testing

Well, we are doing just fine without any standards!

OR ARE WE???!!!

Needs of the software industry ≠ Needs of academia

- Different landscape
    - Larger teams
        - You might be working on a piece of code today that someone else will work on tomorrow.
        - During the testing phase, a completely different team might be responsible for debugging and fixing it.
        - During maintenance and support, another company or project team might handle it
- Different aim
    - Software is a product or a service
        - If it is broken – the customer would leave
    - Product needs to be used many times in different scenarios

BUT!

Or “HOWEVER!” If we are being fancy

Academia does have some needs

- We need to make sure our results are reliable (or “Crap! I hope I don’t have to retract my paper”)
- Reproducibility
- Efficiency of coding
- Sharing of scripts (or “I now need to spend 2 days to fix my script so that I can send it to her”)
- Open Science
- “What does this code from last year do!”

Academia actually has quite many needs

- “I don't know what the difference between "Study1_analysis", "Study1_analysis_final", Study1_analysis_adjusted is “ or “ which was the script with the right results!”
- “I don’t understand the logic I used behind this code block” or “Why did I multiply x by 2 here?”
- Error resolution & Debugging
- Reusability
- Uniformity & Consistency
- Automatisation

Needs of the software you are using

Needs of R

≠ Needs of JAVA

≠ Needs of Neurobs Presentation

≠ Needs of ABAP

≠ Needs of MATLAB

≠ Needs of Python

Our focus is on R and RStudio

❑ Similar principles might apply to Python (but not the same)

❑ But other languages and software might differ radically

Preview of what is coming up

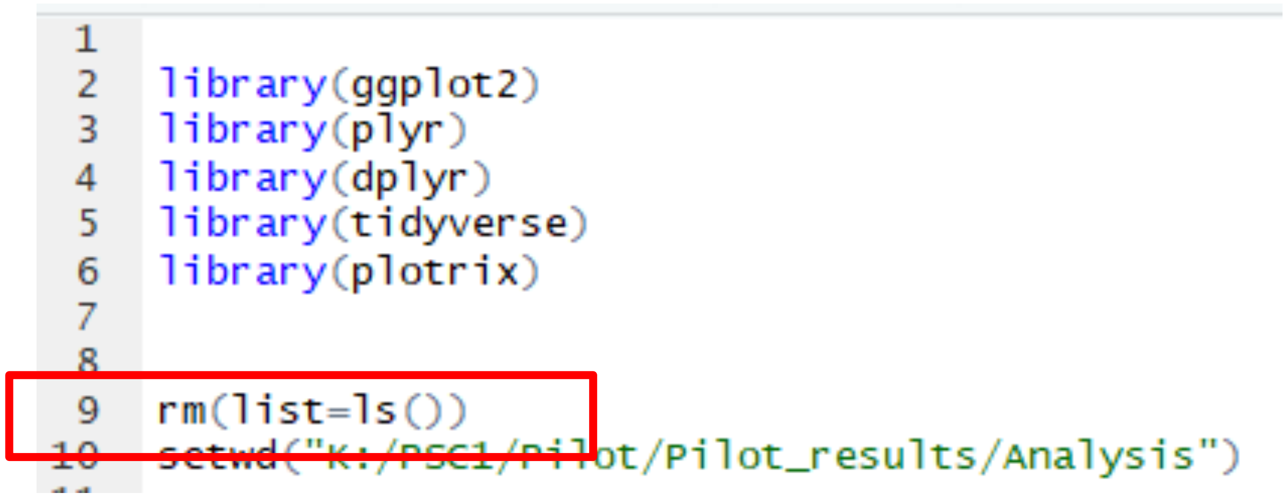
- ❑ Clearing workspace
- ❑ RStudio project functionality
- ❑ Code headers, code folding, section headers
- ❑ Library declarations
- ❑ Version Control
- ❑ Commenting practices
- ❑ Naming conventions
- ❑ Hard coding vs. parameter coding
- ❑ Reducing visual chaos
- ❑ Modularity
- ❑ User-defined functions
- ❑ Bonus mention: pipe functions

So, what can we do?

LET'S BEGIN WITH THE SIMPLER, FASTER CHANGES WE CAN MAKE

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

- Stop doing this!

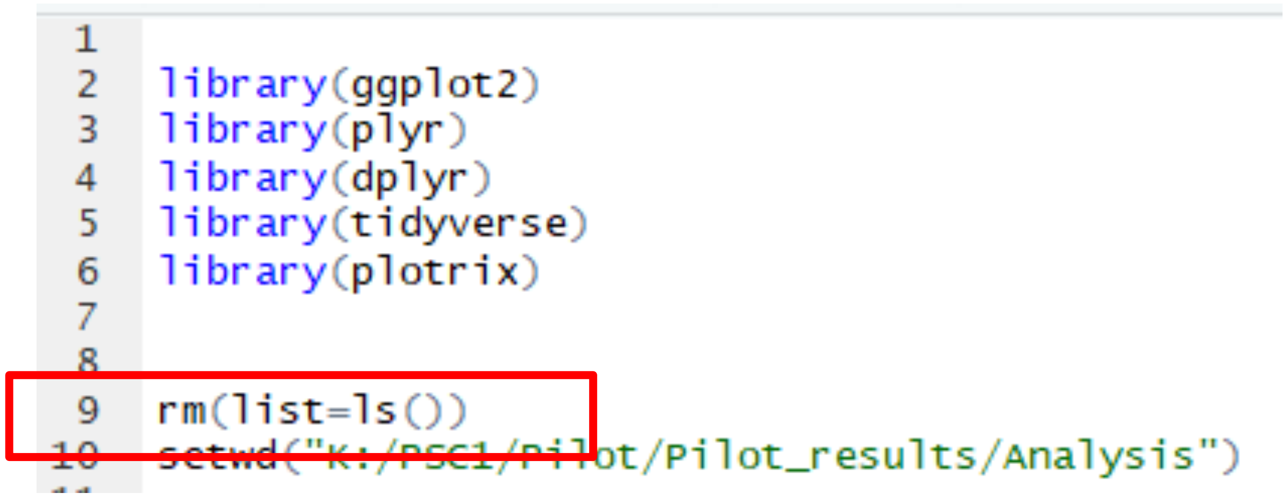


```

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

- Stop doing this!

- Instead use:
    - R Studio project functionality
    - New Session

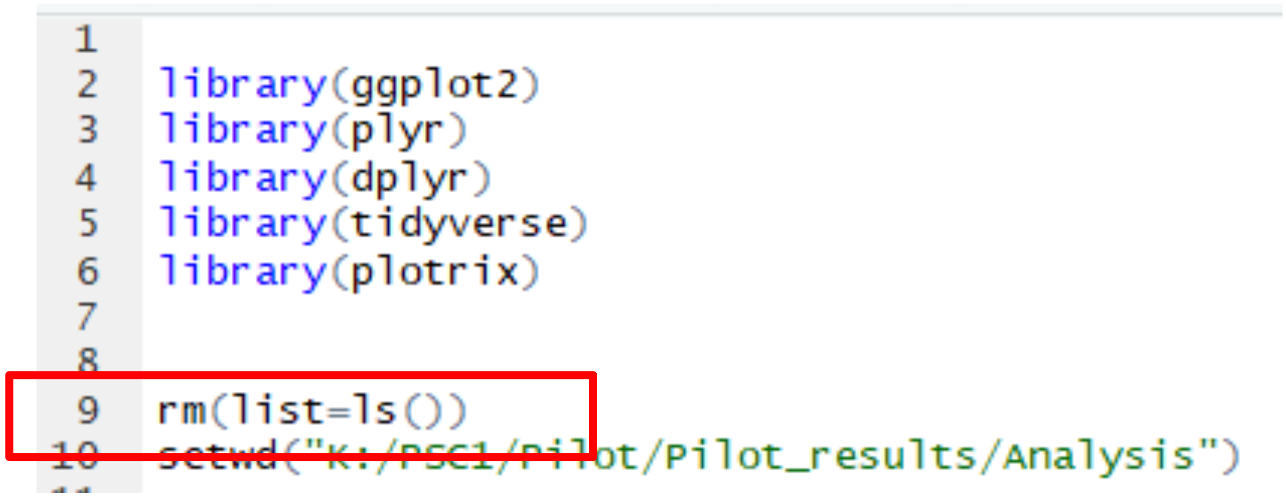


```

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

❑ Stop doing this!

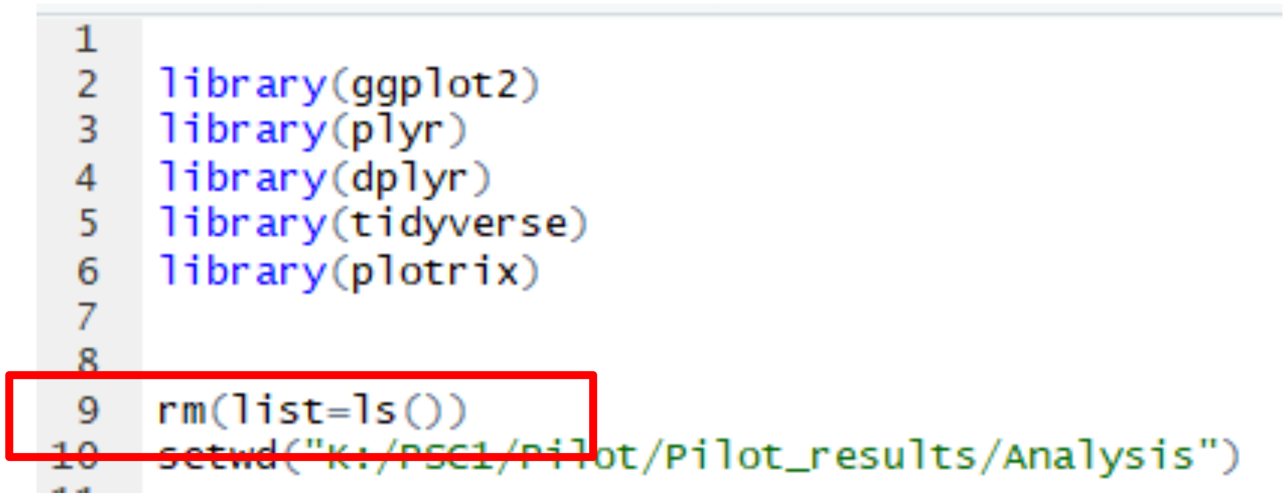
Why?!!! – you may ask



```
library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

- ❑ Stop doing this!



```

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

This seems like an extreme idea, I get it!

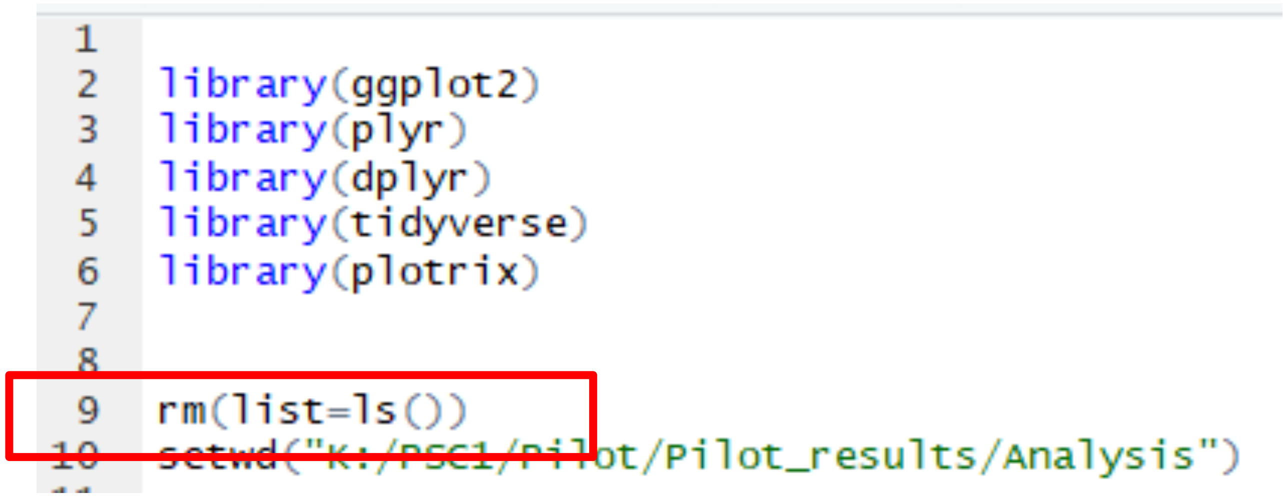
I even got back comments after my first presentation from a couple of people who highly resisted the idea and could not believe that I would propose running a script without first making sure that the workspace is not free of conflicting variables.

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

- ❑ Stop doing this!

How can someone recommend running a script in workspace that’s not empty?

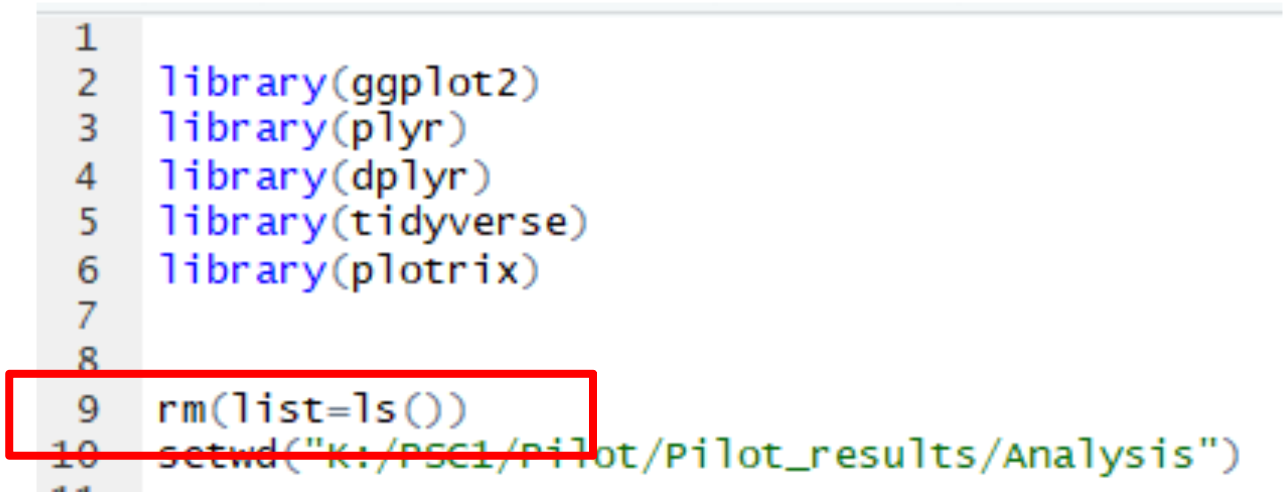
That’s not what I am recommend. You need a clean workspace. You just get to it in a different way!



```

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

Let me show you what happens when you run a script with this command at the beginning.

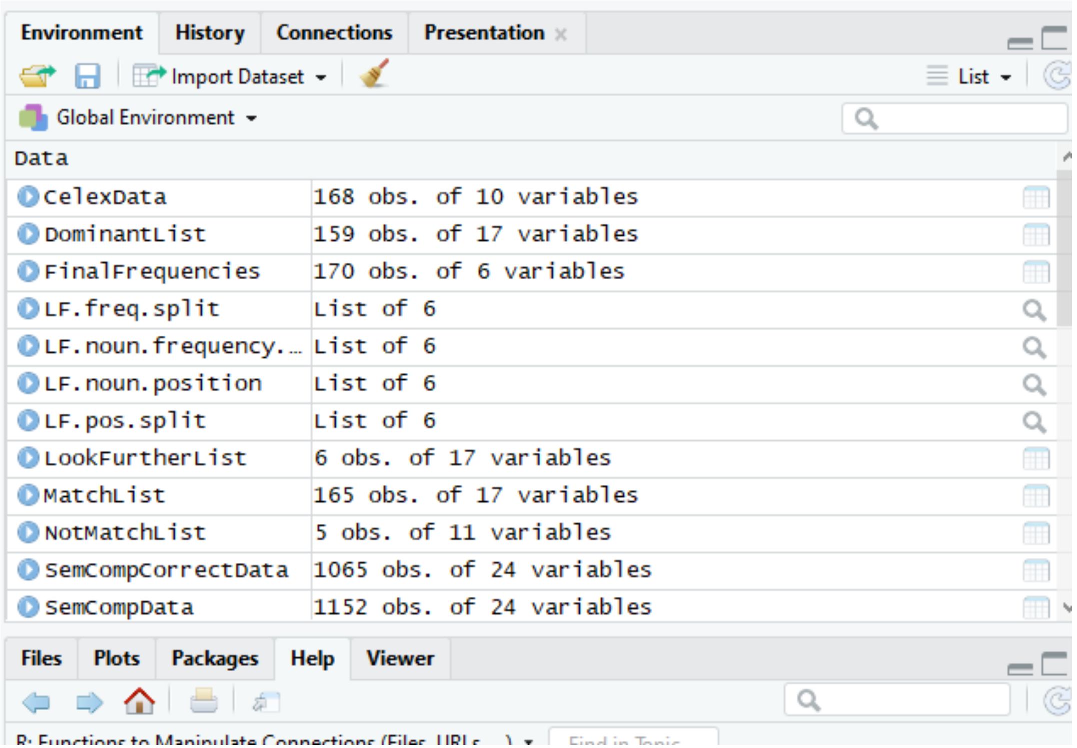


```

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```

Stop clearing the workspace (rm(list =ls()) at the beginning of your code

Let’s say that I am working on a complicated project with a complicated piece of code and I have all of these variables in the workspace

I take a break or something urgent comes up and my workspace is left like this



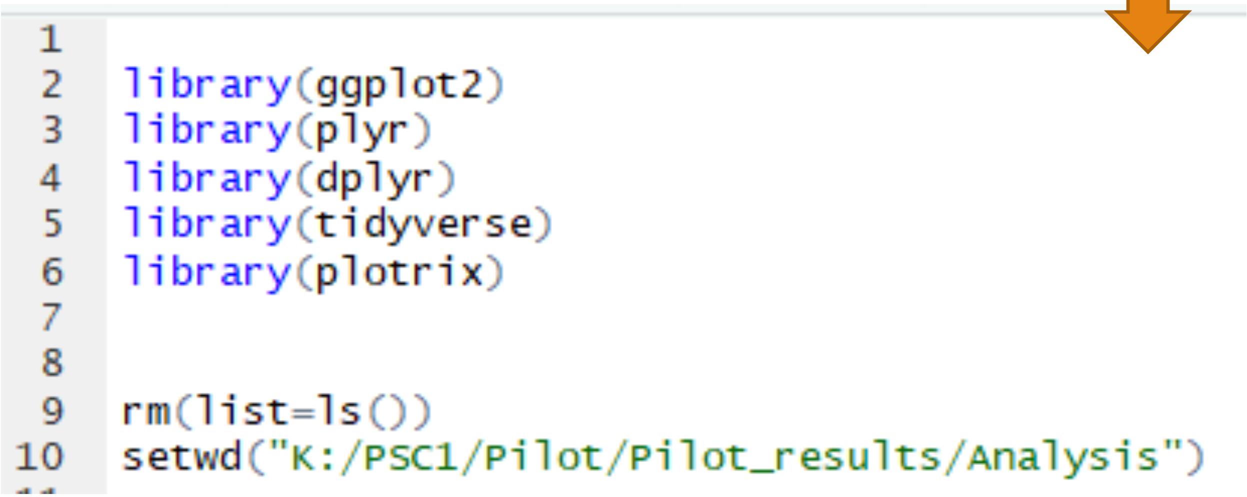
```
Data
CelexData              168 obs. of 10 variables
DominantList           159 obs. of 17 variables
FinalFrequencies       170 obs. of 6 variables
LF.freq.split          List of 6
LF.noun.frequency.…    List of 6
LF.noun.position       List of 6
LF.pos.split           List of 6
LookFurtherList        6 obs. of 17 variables
MatchList              165 obs. of 17 variables
NotMatchList           5 obs. of 11 variables
SemCompCorrectData     1065 obs. of 24 variables
SemCompData            1152 obs. of 24 variables
```

Original workspace

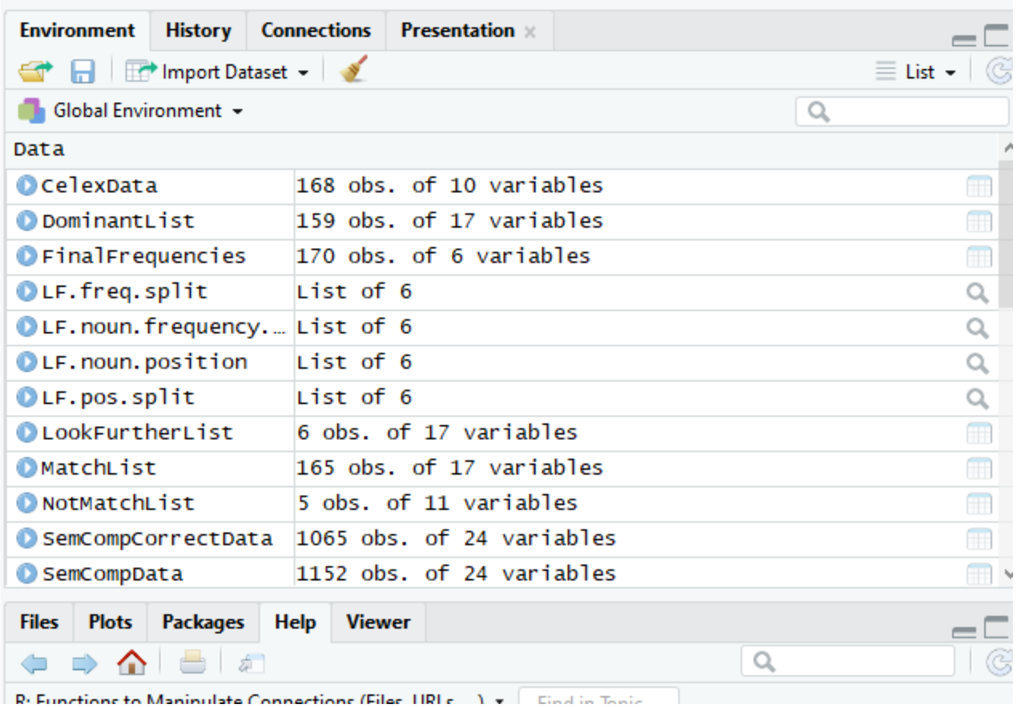
Stop clearing the workspace at the beginning of your code

In the meanwhile, a colleague has shared a script with me that I was waiting for, which solves a problem I am facing in another project

In the spirit of charging forward and excited about finally having a solution to my problem, I run the script, which looks like



```r

library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSC1/Pilot/Pilot_results/Analysis")
```



Data	
celexData	168 obs. of 10 variables
DominantList	159 obs. of 17 variables
FinalFrequencies	170 obs. of 6 variables
LF.freq.split	List of 6
LF.noun.frequency.…	List of 6
LF.noun.position	List of 6
LF.pos.split	List of 6
LookFurtherList	6 obs. of 17 variables
MatchList	165 obs. of 17 variables
NotMatchList	5 obs. of 11 variables
SemCompCorrectData	1065 obs. of 24 variables
SemCompData	1152 obs. of 24 variables

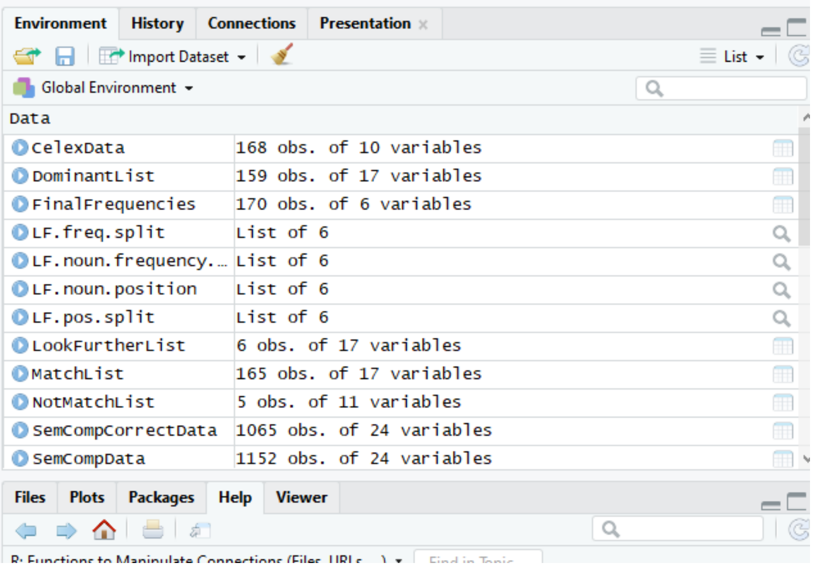
Original workspace

Stop clearing the workspace at the beginning of your code

What happens next?

Stop clearing the workspace at the beginning of your code

What happens next?

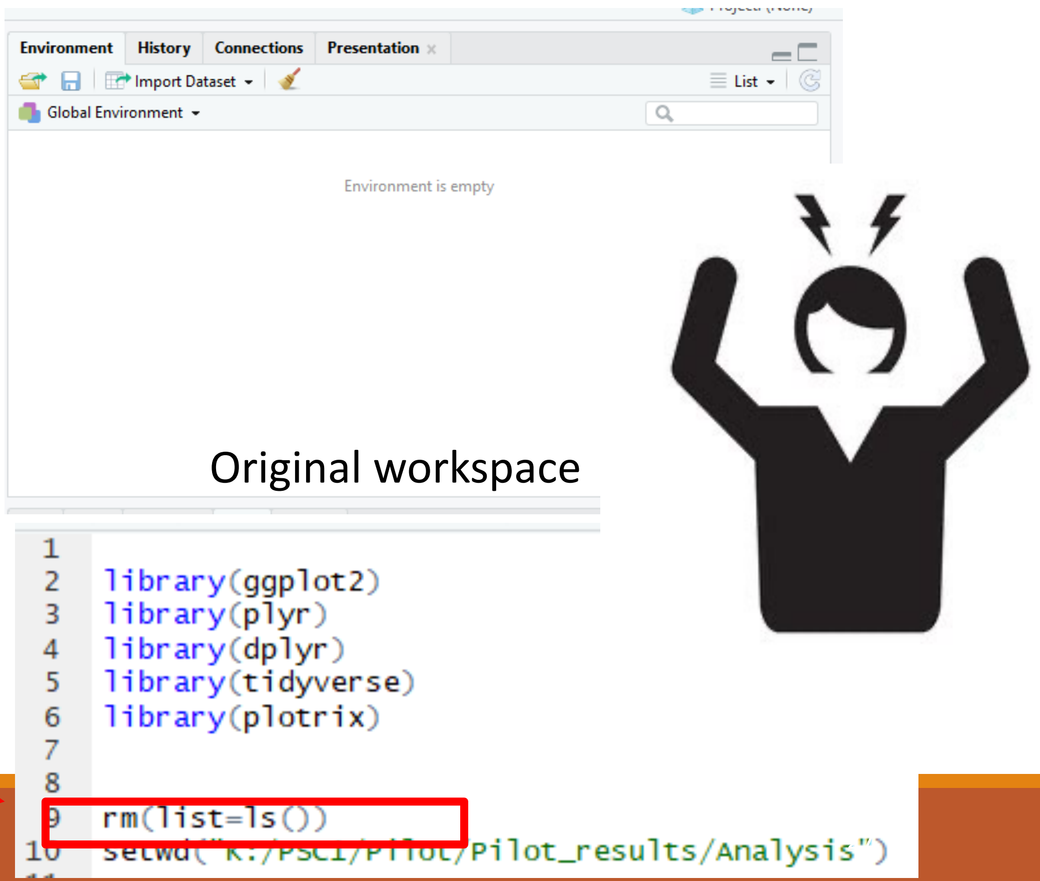


Data

celexData	168 obs. of 10 variables
DominantList	159 obs. of 17 variables
FinalFrequencies	170 obs. of 6 variables
LF.freq.split	List of 6
LF.noun.frequency...	List of 6
LF.noun.position	List of 6
LF.pos.split	List of 6
LookFurtherList	6 obs. of 17 variables
MatchList	165 obs. of 17 variables
NotMatchList	5 obs. of 11 variables
SemCompCorrectData	1065 obs. of 24 variables
SemCompData	1152 obs. of 24 variables

Original workspace

All my precious variables, values and dataframe are gone! Because of that line



Environment is empty

Original workspace

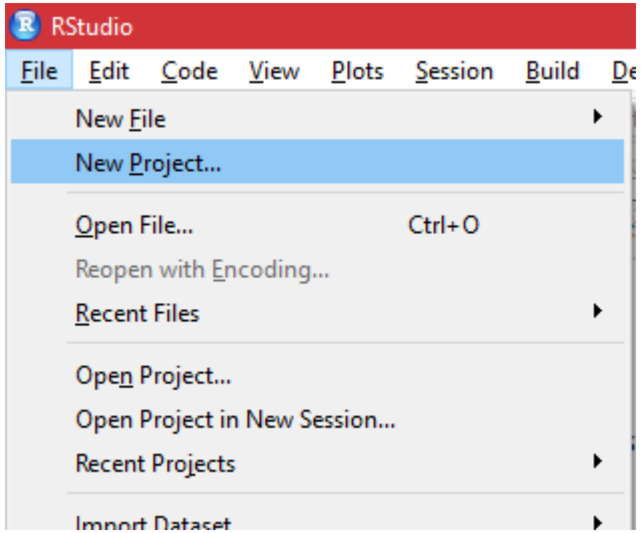
```
library(ggplot2)
library(plyr)
library(dplyr)
library(tidyverse)
library(plotrix)


rm(list=ls())
setwd("K:/PSCI/Pilot/Pilot_results/Analysis")
```

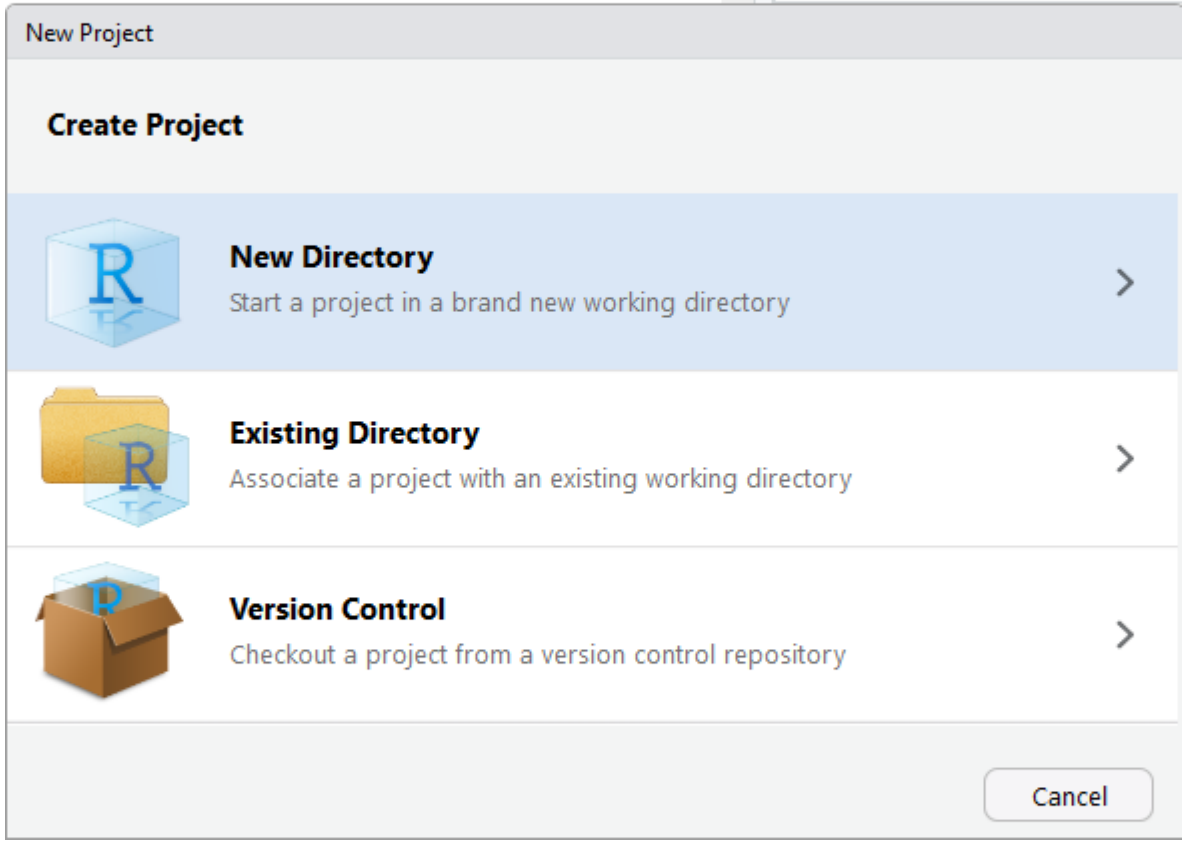
Stop clearing the workspace at the beginning of your code

Instead, use R Studio’s project functionality

Use of RStudio project functionality

- ❑ One of the most impactful changes




New Project

Create Project

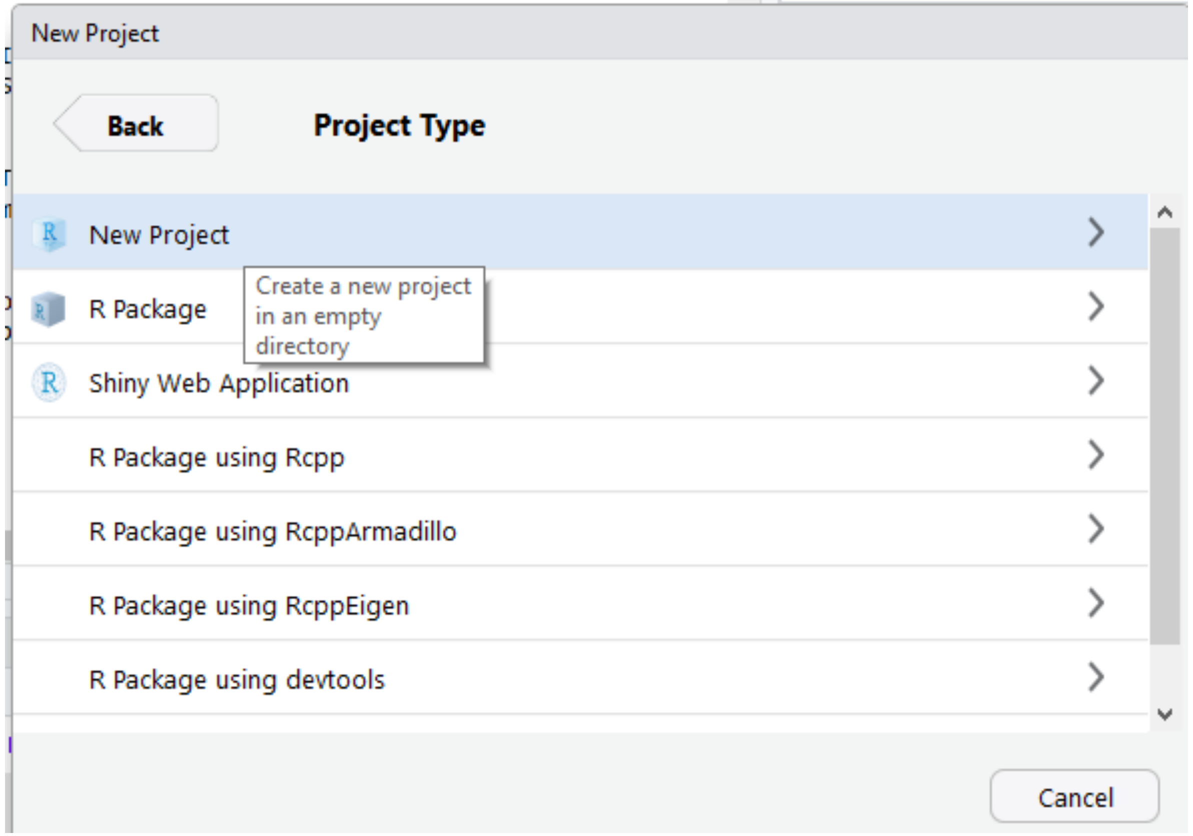
New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

Use of RStudio project functionality

- One of the most impactful changes



New Project

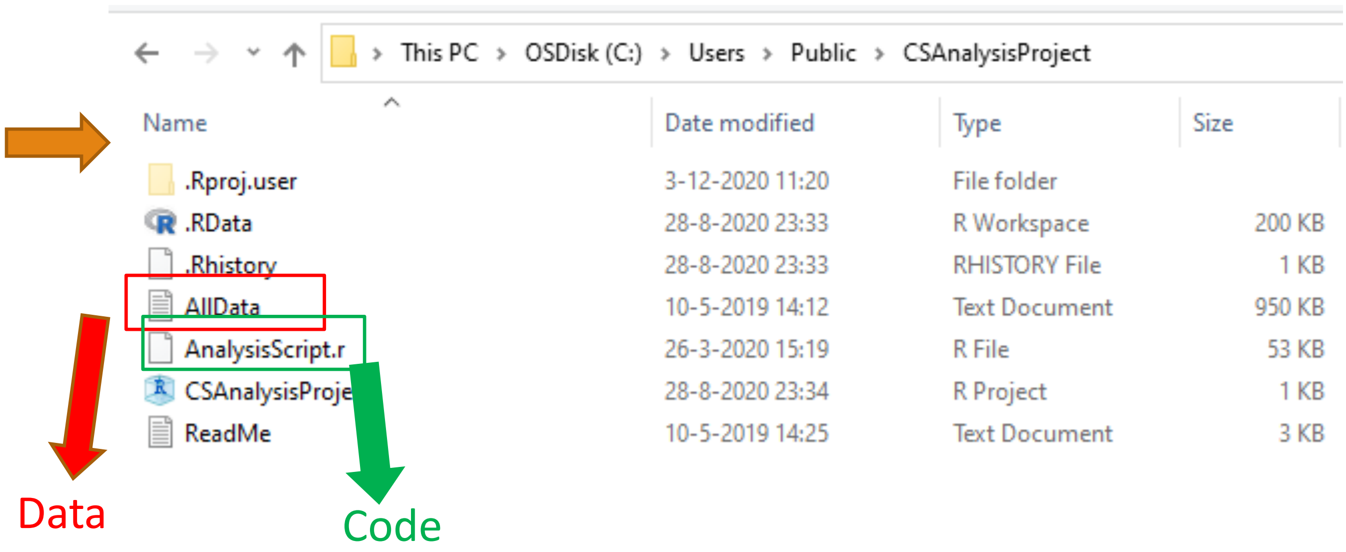
Back

Project Type

New Project
Create a new project in an empty directory
R Package
Shiny Web Application
R Package using Rcpp
R Package using RcppArmadillo
R Package using RcppEigen
R Package using devtools
Cancel

Use of RStudio project functionality

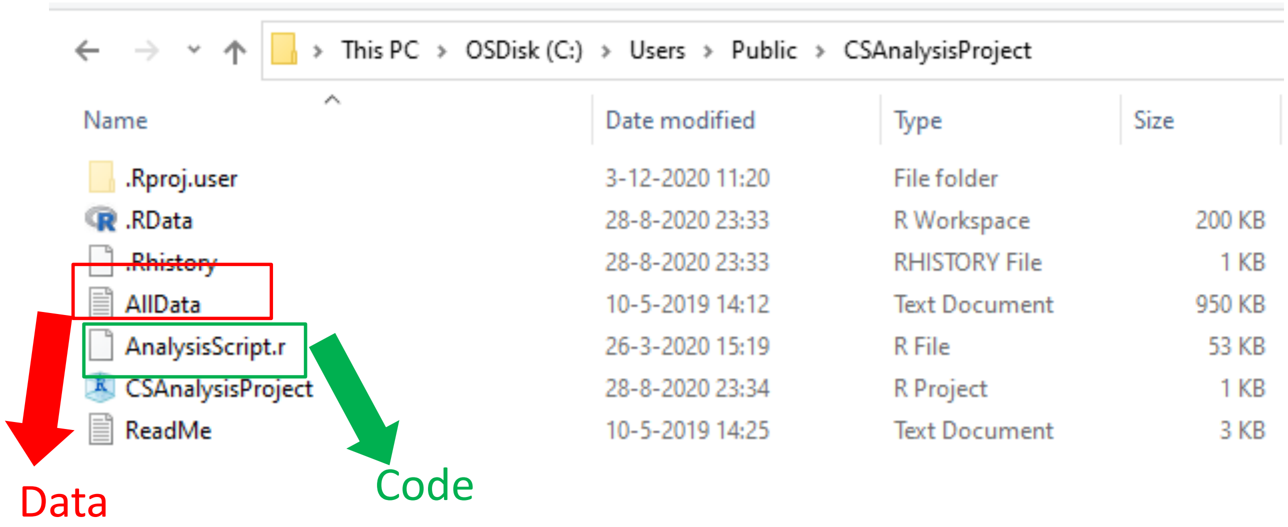
- ❑ One of the most impactful changes
- ❑ Data is in the same folder as the code



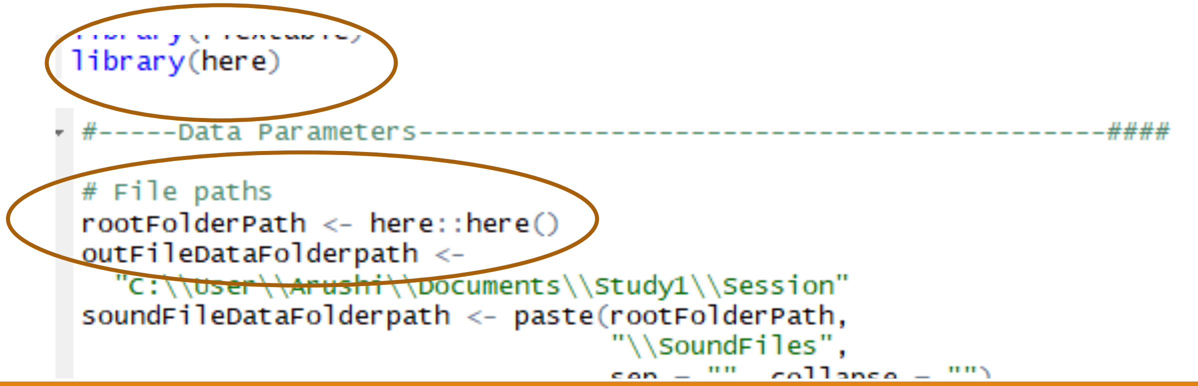
Name	Date modified	Type	Size
.Rproj.user	3-12-2020 11:20	File folder	
.RData	28-8-2020 23:33	R Workspace	200 KB
.Rhistory	28-8-2020 23:33	RHISTORY File	1 KB
AllData	10-5-2019 14:12	Text Document	950 KB
AnalysisScript.r	26-3-2020 15:19	R File	53 KB
CSAnalysisProje	28-8-2020 23:34	R Project	1 KB
ReadMe	10-5-2019 14:25	Text Document	3 KB

Use of RStudio project functionality

- ❑ One of the most impactful changes
- ❑ Data is in the same folder as the code
- ❑ Easily shareable. Allows sharing and linking of data and code simultaneously
- ❑ here:here()
  - ❑ C:/Users/Public/CSAnalysisProject



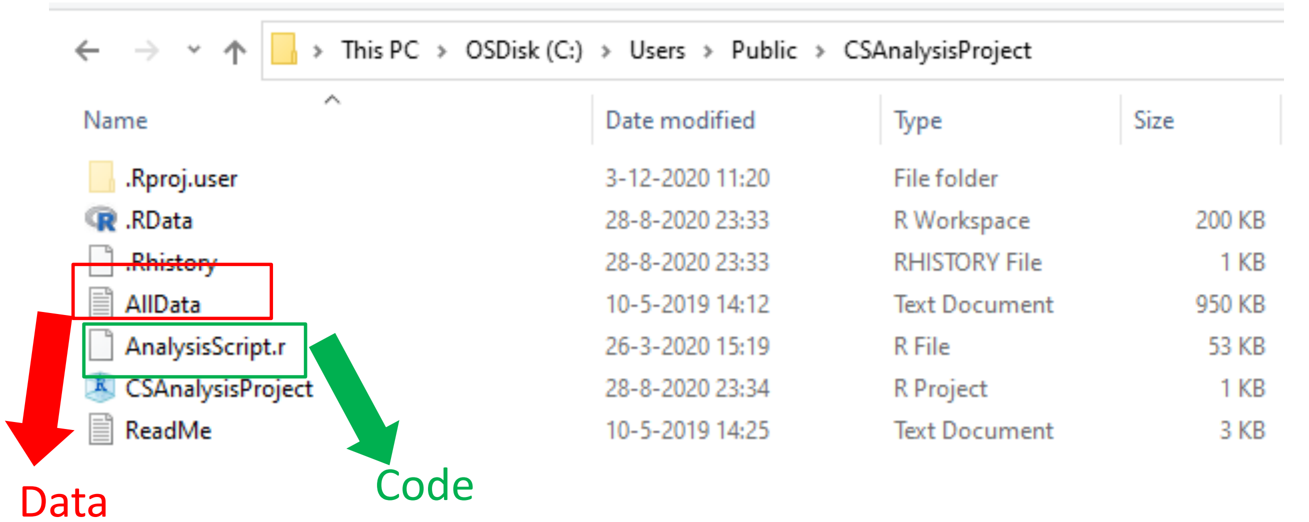
Name	Date modified	Type	Size
.Rproj.user	3-12-2020 11:20	File folder	
.RData	28-8-2020 23:33	R Workspace	200 KB
.Rhistory	28-8-2020 23:33	RHISTORY File	1 KB
AllData	10-5-2019 14:12	Text Document	950 KB
AnalysisScript.r	26-3-2020 15:19	R File	53 KB
CSAnalysisProject	28-8-2020 23:34	R Project	1 KB
ReadMe	10-5-2019 14:25	Text Document	3 KB



```
library(flextable)
library(here)

#-----Data Parameters-------------------------------------------####

# File paths
rootFolderPath <- here::here()
outFileDataFolderpath <-
  "C:\\User\\Arushi\\Documents\\Study1\\Session"
soundFileDataFolderpath <- paste(rootFolderPath,
                                 "\\SoundFiles",
                                 sep = "", collapse = "")
```

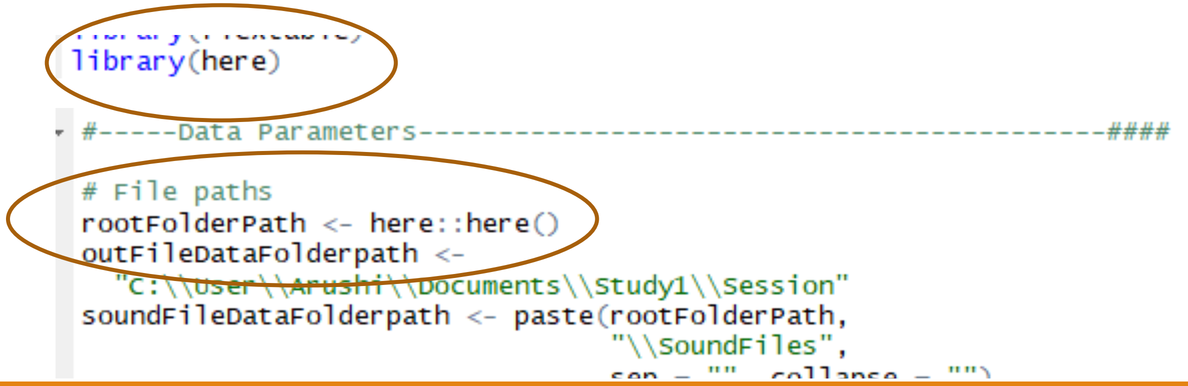
Use of RStudio project functionality

- ❑ One of the most impactful changes
- ❑ Data is in the same folder as the code
- ❑ Easily shareable. Allows sharing and linking of data and code simultaneously
- ❑ here:here()
  - ❑ C:/Users/Public/CSAnalysisProject
  - ❑ points to the directory which has the .Rproj file



This PC › OSDisk (C:) › Users › Public › CSAnalysisProject

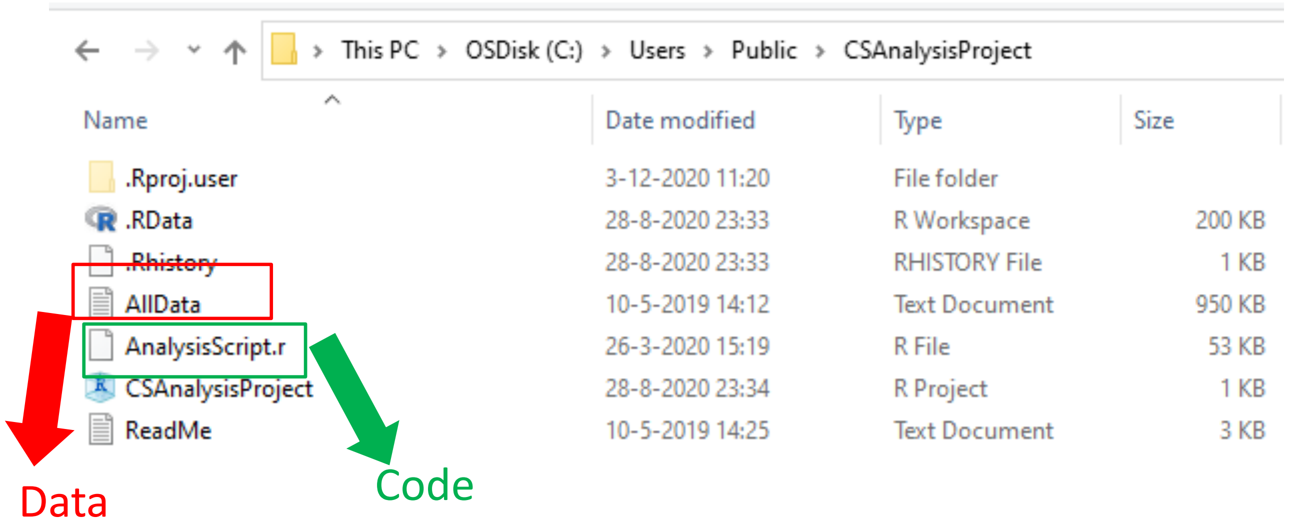
Name	Date modified	Type	Size
.Rproj.user	3-12-2020 11:20	File folder	
.RData	28-8-2020 23:33	R Workspace	200 KB
.Rhistory	28-8-2020 23:33	RHISTORY File	1 KB
AllData	10-5-2019 14:12	Text Document	950 KB
AnalysisScript.r	26-3-2020 15:19	R File	53 KB
CSAnalysisProject	28-8-2020 23:34	R Project	1 KB
ReadMe	10-5-2019 14:25	Text Document	3 KB



```
library(flextable)
library(here)

#-----Data Parameters--------------------------------------------####

# File paths
rootFolderPath <- here::here()
outFileDataFolderpath <-
  "C:\\User\\Arushi\\Documents\\Study1\\Session"
soundFileDataFolderpath <- paste(rootFolderPath,
                                 "\\SoundFiles",
                                 sep = "", collapse = "")
```

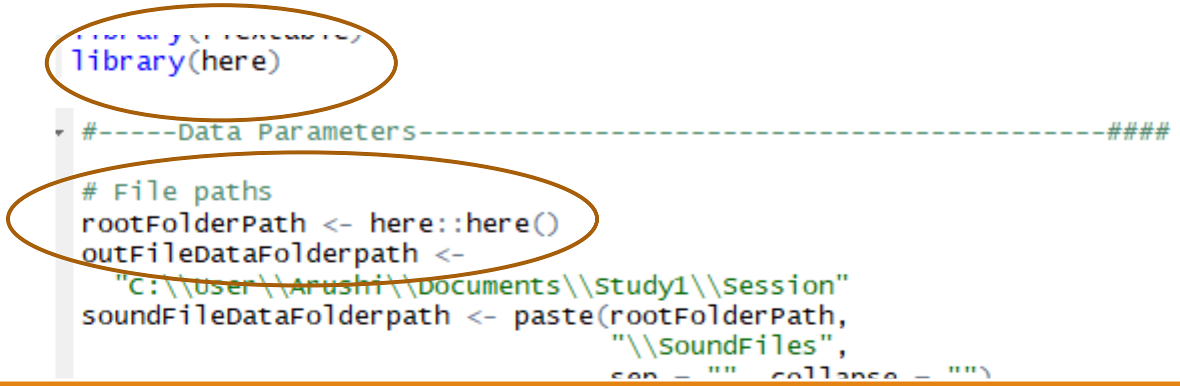
Use of RStudio project functionality

- ❑ One of the most impactful changes
- ❑ Data is in the same folder as the code
- ❑ Easily shareable. Allows sharing and linking of data and code simultaneously
- ❑ here:here()
  - ❑ C:/Users/Public/CSAnalysisProject
  - ❑ points to the directory which has the .Rproj file



This PC › OSDisk (C:) › Users › Public › CSAnalysisProject

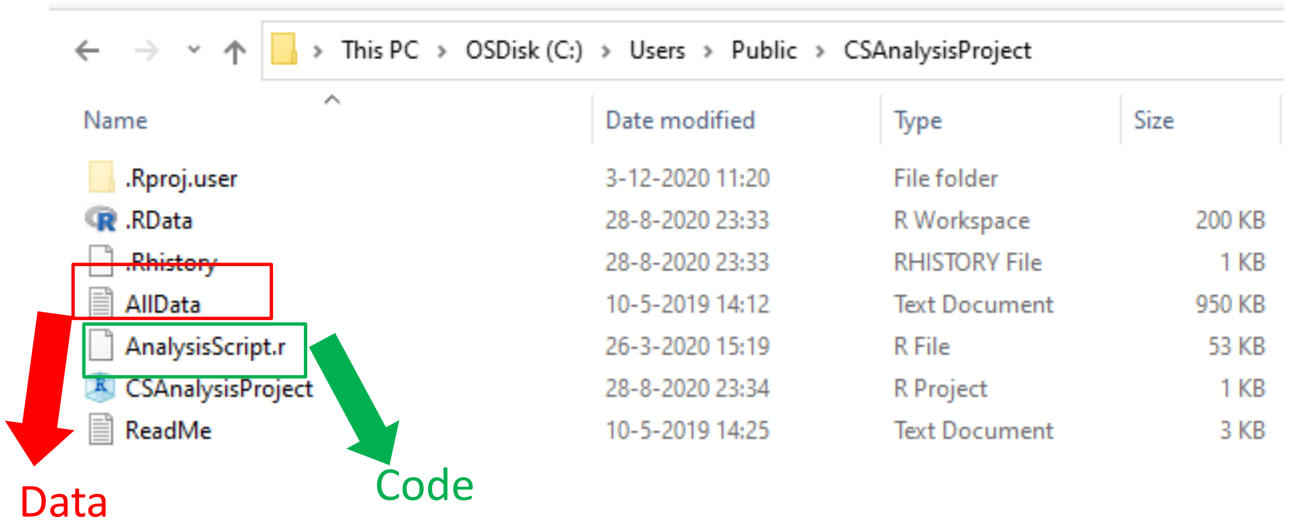
Name	Date modified	Type	Size
.Rproj.user	3-12-2020 11:20	File folder	
.RData	28-8-2020 23:33	R Workspace	200 KB
.Rhistory	28-8-2020 23:33	RHISTORY File	1 KB
AllData	10-5-2019 14:12	Text Document	950 KB
AnalysisScript.r	26-3-2020 15:19	R File	53 KB
CSAnalysisProject	28-8-2020 23:34	R Project	1 KB
ReadMe	10-5-2019 14:25	Text Document	3 KB



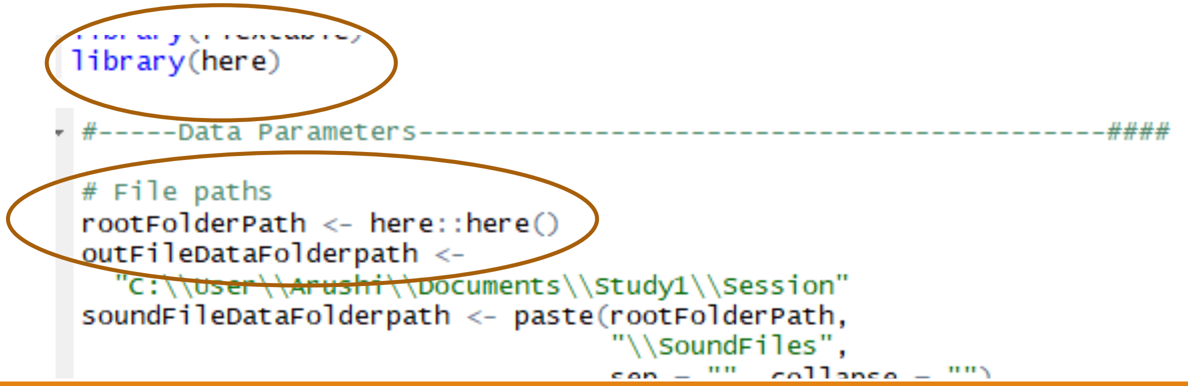
```
library(flextable)
library(here)

#-----Data Parameters-------------------------------------------####

# File paths
rootFolderPath <- here::here()
outFileDataFolderpath <-
  "C:\\User\\Arushi\\Documents\\Study1\\Session"
soundFileDataFolderpath <- paste(rootFolderPath,
                                 "\\SoundFiles",
                                 sep = "", collapse = "")
```

Use of RStudio project functionality

- ❑ One of the most impactful changes
- ❑ Data is in the same folder as the code
- ❑ Easily shareable. Allows sharing and linking of data and code simultaneously
- ❑ here:here()
  - ❑ C:/Users/Public/CSAnalysisProject
  - ❑ points to the directory which has the .Rproj file
  - ❑ Thus, the folder path specified in the code does not need to be changed when running the code in someone else's computer which has a different directory or folder organisation.



Name	Date modified	Type	Size
.Rproj.user	3-12-2020 11:20	File folder	
.RData	28-8-2020 23:33	R Workspace	200 KB
.Rhistory	28-8-2020 23:33	RHISTORY File	1 KB
AllData	10-5-2019 14:12	Text Document	950 KB
AnalysisScript.r	26-3-2020 15:19	R File	53 KB
CSAnalysisProject	28-8-2020 23:34	R Project	1 KB
ReadMe	10-5-2019 14:25	Text Document	3 KB



```
library(flextable)
library(here)

#-----Data Parameters-------------------------------------------####

# File paths
rootFolderPath <- here::here()
outFileDataFolderpath <-
  "C:\\User\\Arushi\\Documents\\Study1\\Session"
soundFileDataFolderpath <- paste(rootFolderPath,
                                 "\\SoundFiles",
                                 sep = "", collapse = "")
```

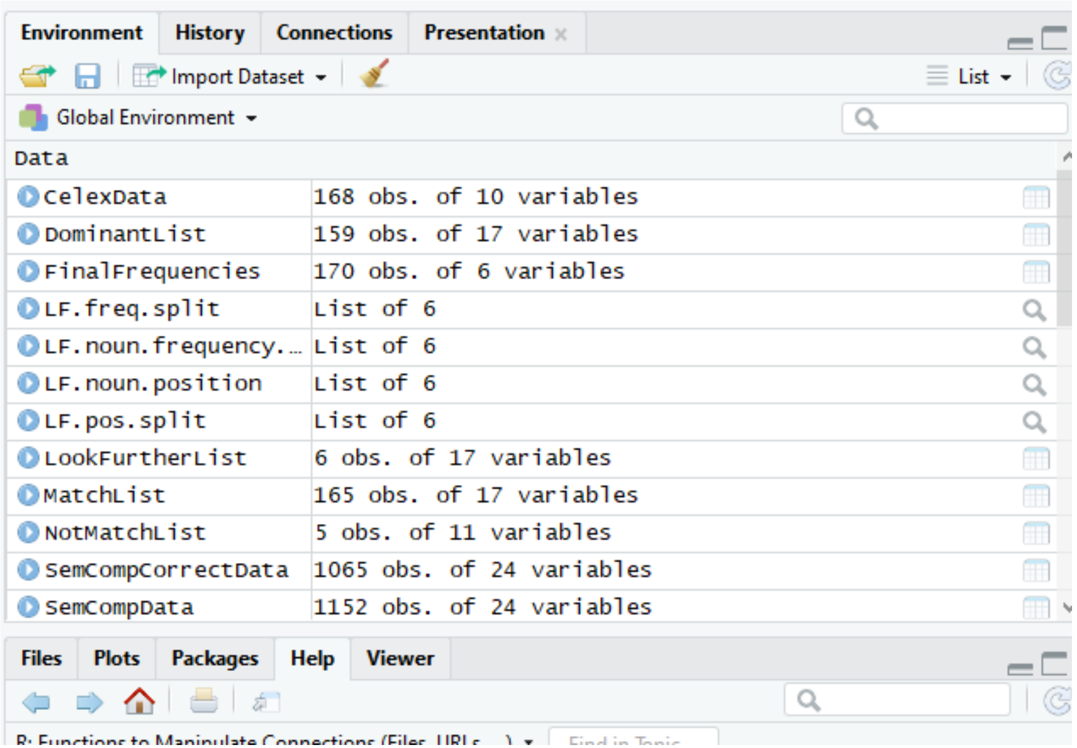
Use of RStudio project functionality

Let me show you what happens when you use R Studio project functionality instead of clearing the workspace

Use of RStudio project functionality

Same scenario: complicated project, important variables in the workplace;

a colleague sends code related to a different project that I was waiting on eagerly. However, this colleague is smart and uses project functionality



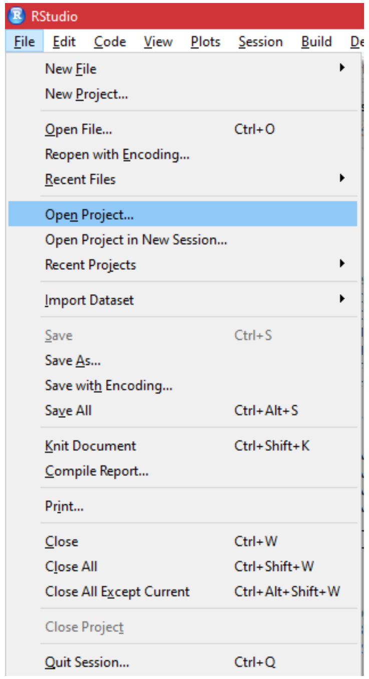
Data

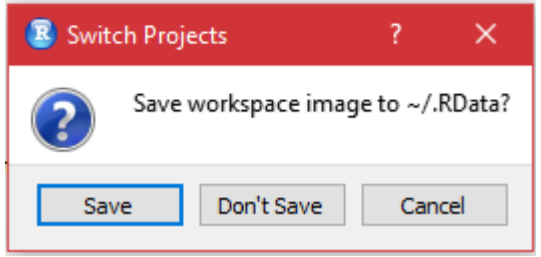
CelexData	168 obs. of 10 variables
DominantList	159 obs. of 17 variables
FinalFrequencies	170 obs. of 6 variables
LF.freq.split	List of 6
LF.noun.frequency.…	List of 6
LF.noun.position	List of 6
LF.pos.split	List of 6
LookFurtherList	6 obs. of 17 variables
MatchList	165 obs. of 17 variables
NotMatchList	5 obs. of 11 variables
SemCompCorrectData	1065 obs. of 24 variables
SemCompData	1152 obs. of 24 variables

Original workspace

Use of RStudio project functionality

However, this colleague is smart and use project functionality. They instruct me to use projects. So, I do it

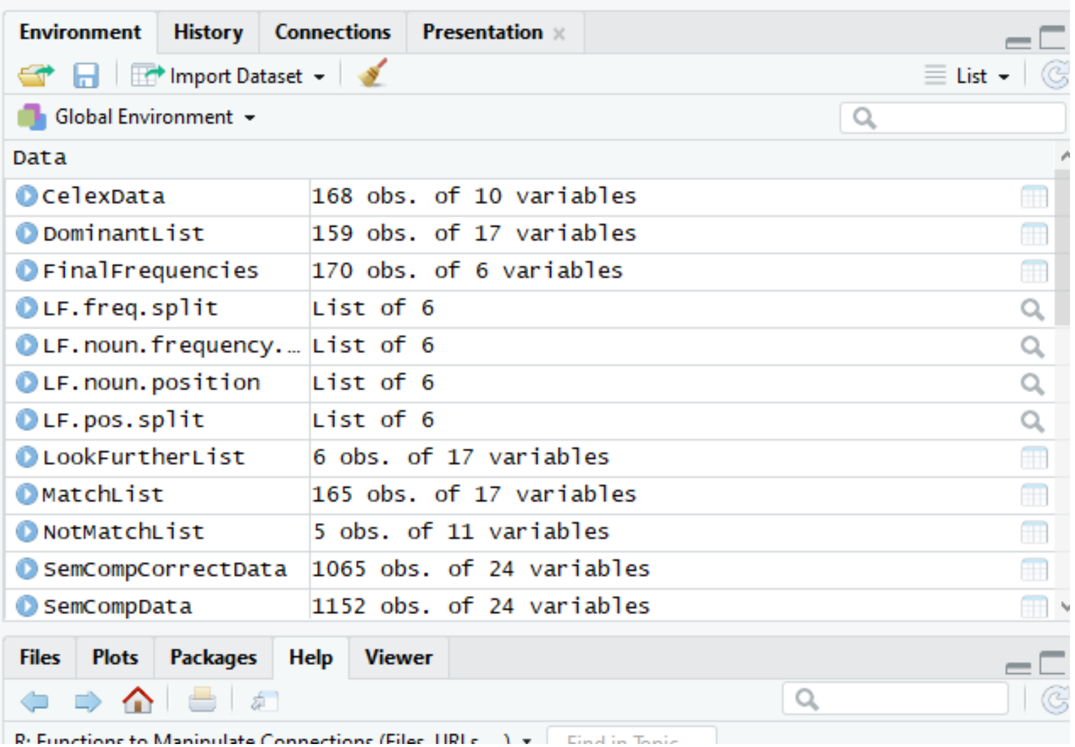




Save workspace image to ~/.RData?

When I open a project, I am prompted to save the current workspace

Saving this workspace image prevents loss of data!!!

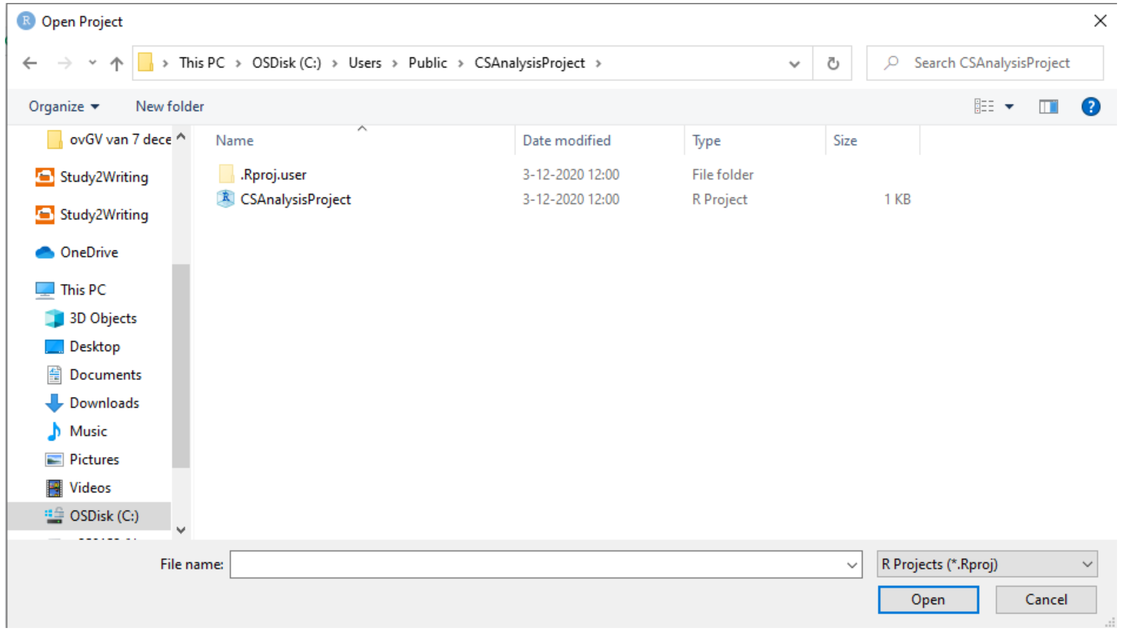


Data	
celexData	168 obs. of 10 variables
DominantList	159 obs. of 17 variables
FinalFrequencies	170 obs. of 6 variables
LF.freq.split	List of 6
LF.noun.frequency.…	List of 6
LF.noun.position	List of 6
LF.pos.split	List of 6
LookFurtherList	6 obs. of 17 variables
MatchList	165 obs. of 17 variables
NotMatchList	5 obs. of 11 variables
SemCompCorrectData	1065 obs. of 24 variables
SemCompData	1152 obs. of 24 variables

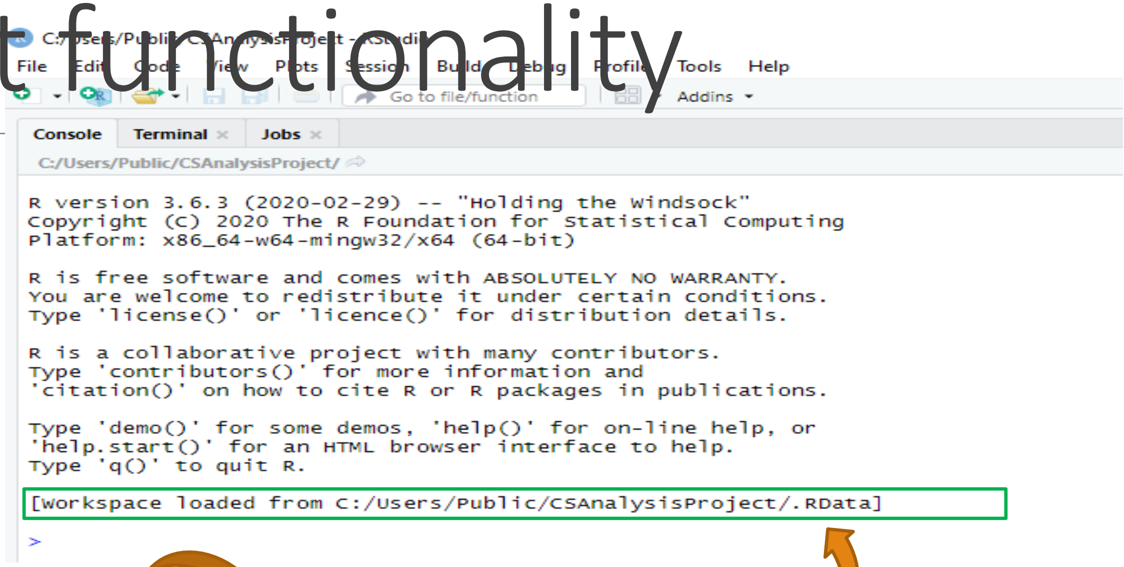
Original workspace

Use of RStudio project functionality

Choose the project and open the directory. I run a couple of things and the project workspace is now loaded.

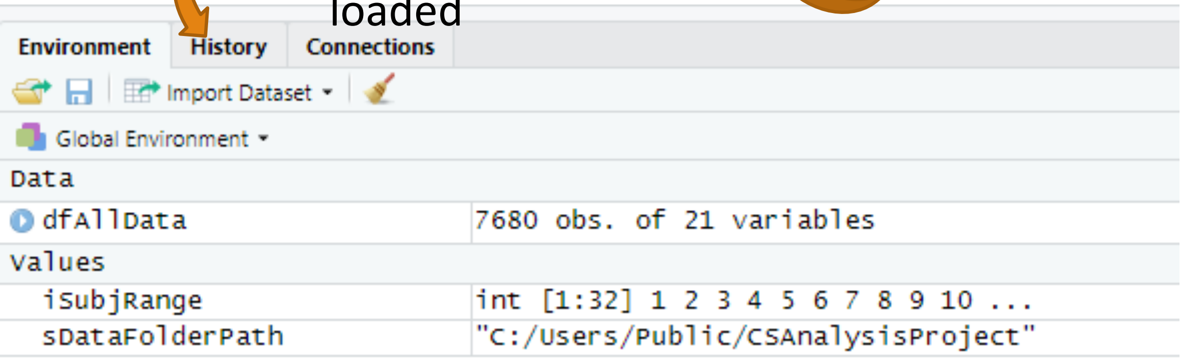


Name	Date modified	Type	Size
.Rproj.user	3-12-2020 12:00	File folder	
CSAnalysisProject	3-12-2020 12:00	R Project	1 KB



```
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/Public/CSAnalysisProject/.RData]

>
```

Saved workspace is loaded

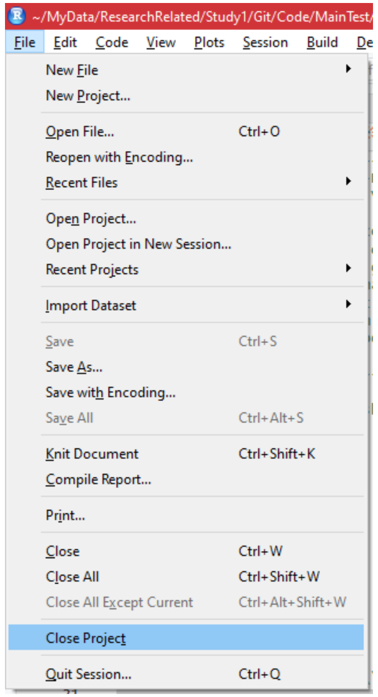


```
Data
dfAllData        7680 obs. of 21 variables
Values
  iSubjRange     int [1:32] 1 2 3 4 5 6 7 8 9 10 ...
  sDataFolderPath "C:/Users/Public/CSAnalysisProject"
```

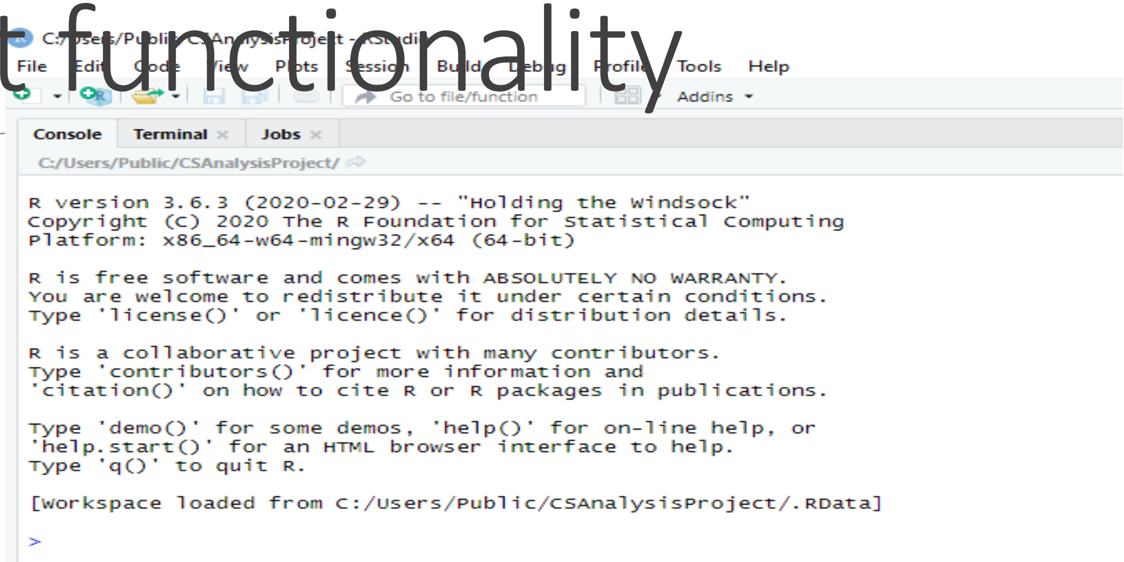
Project workspace

Use of RStudio project functionality

Now if I need to switch back to the original workspace, I close project

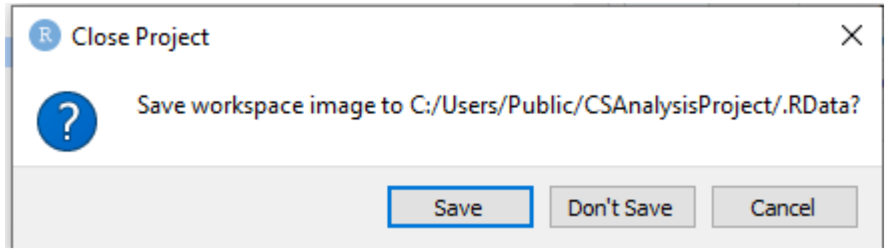


I am now prompted to save the project workspace



```
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/Public/CSAnalysisProject/.RData]

>
```

Project workspace

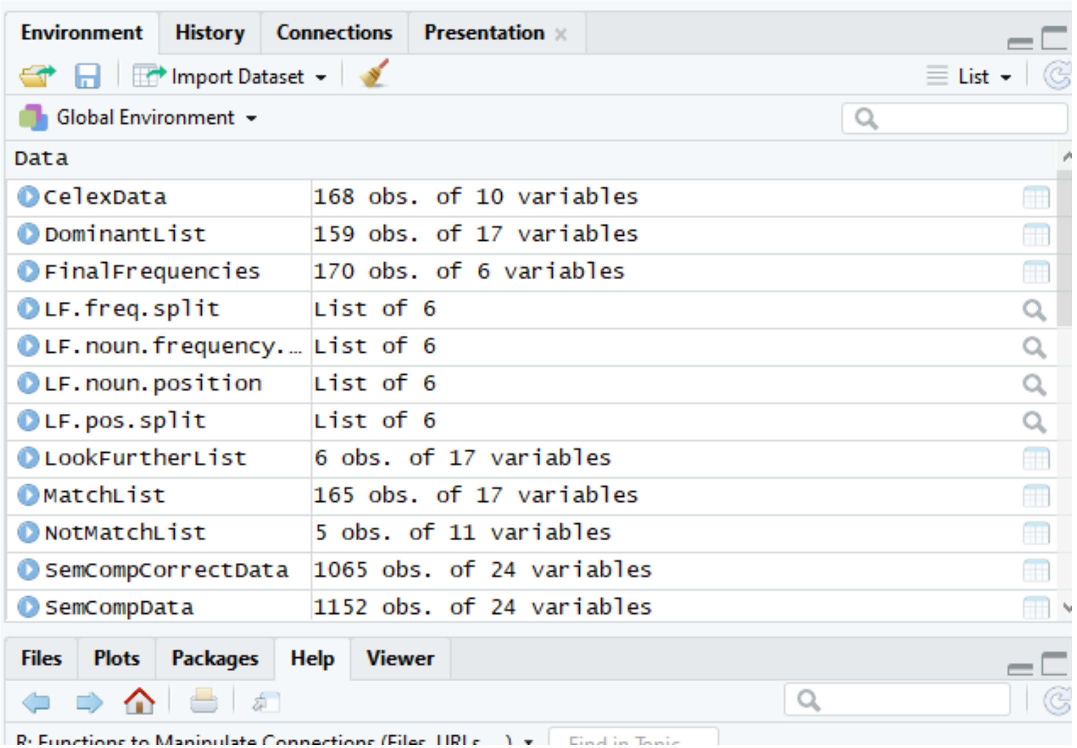


Save workspace image to C:/Users/Public/CSAnalysisProject/.RData?

Saving the project workspace image prevents loss of data from the project analysis and can be helpful in pausing analyses midway

Use of RStudio project functionality

And I can get back to my original workspace without any loss of data!!!



Data

CelexData	168 obs. of 10 variables
DominantList	159 obs. of 17 variables
FinalFrequencies	170 obs. of 6 variables
LF.freq.split	List of 6
LF.noun.frequency.…	List of 6
LF.noun.position	List of 6
LF.pos.split	List of 6
LookFurtherList	6 obs. of 17 variables
MatchList	165 obs. of 17 variables
NotMatchList	5 obs. of 11 variables
SemCompCorrectData	1065 obs. of 24 variables
SemCompData	1152 obs. of 24 variables

Original workspace

Aside: advantages of here package

- **getwd()** not optimal: It returns different results depending on file types and directory structures
  - path needs to be rewritten according to directory structure or different OS

- **relative paths** are not useful if your current file is a sub-folder in the main project folder – and – you want to reference a file in another sub folder

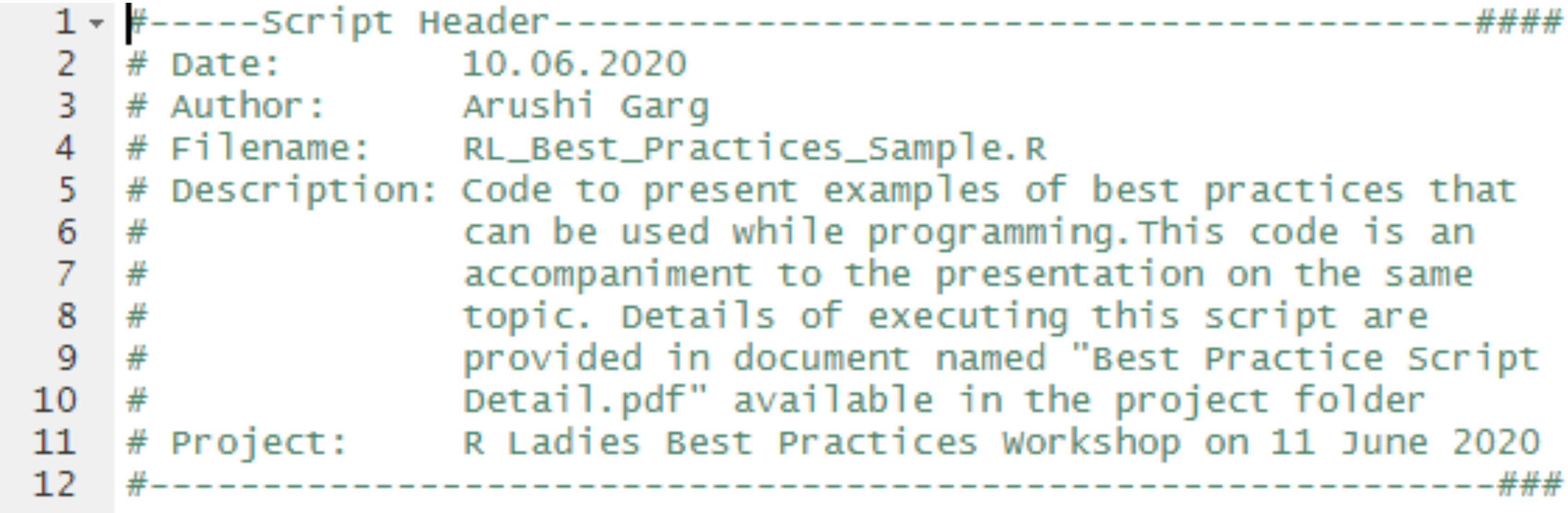
- folder path specified in the code does not need to be changed when running the code in someone else's computer which has a different directory name or even different OS

- creates file paths corresponding to requirement of current OS

- more details: https://malco.io/2018/11/05/why-should-i-use-the-here-package-when-i-m-already-using-projects/

Code Headers

- Utilise code folding features!



```
#-----Script Header-----------------------------------------####
# Date:        10.06.2020
# Author:      Arushi Garg
# Filename:    RL_Best_Practices_Sample.R
# Description: Code to present examples of best practices that
#              can be used while programming.This code is an
#              accompaniment to the presentation on the same
#              topic. Details of executing this script are
#              provided in document named "Best Practice Script
#              Detail.pdf" available in the project folder
# Project:     R Ladies Best Practices Workshop on 11 June 2020
#------------------------------------------------------------###
```

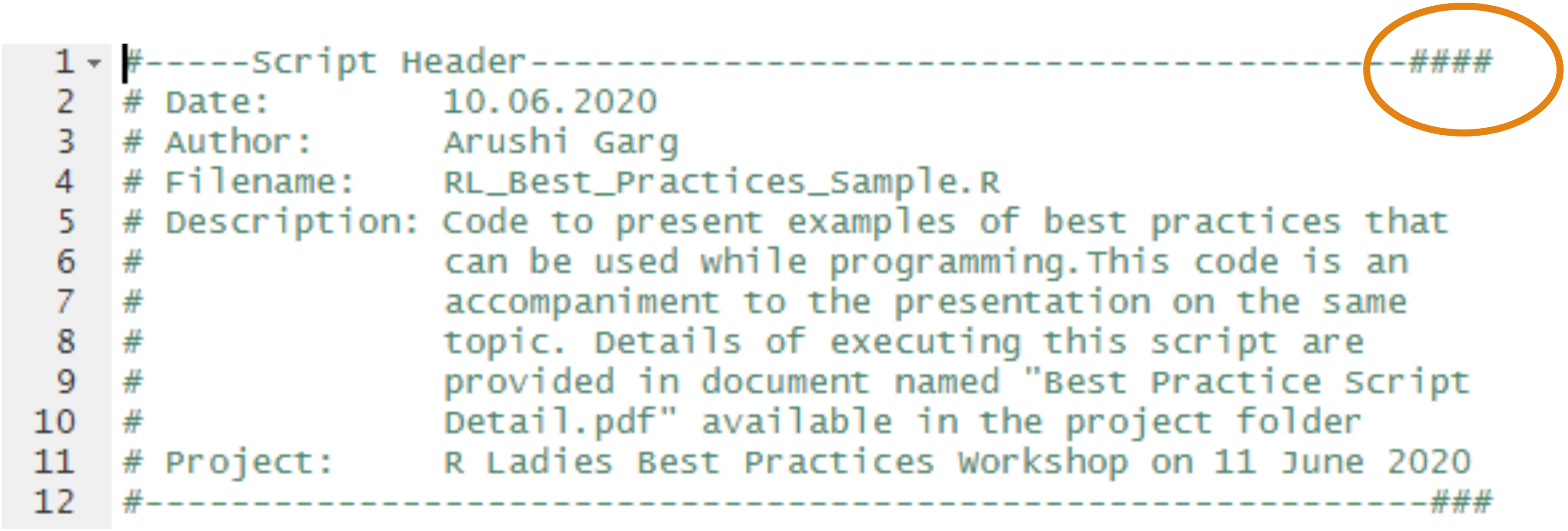
❑ Useful for identifying purpose of script.

❑ If script is shared with someone, it can be informative with regards to the original author and project

Code Folding

For the uninitiated

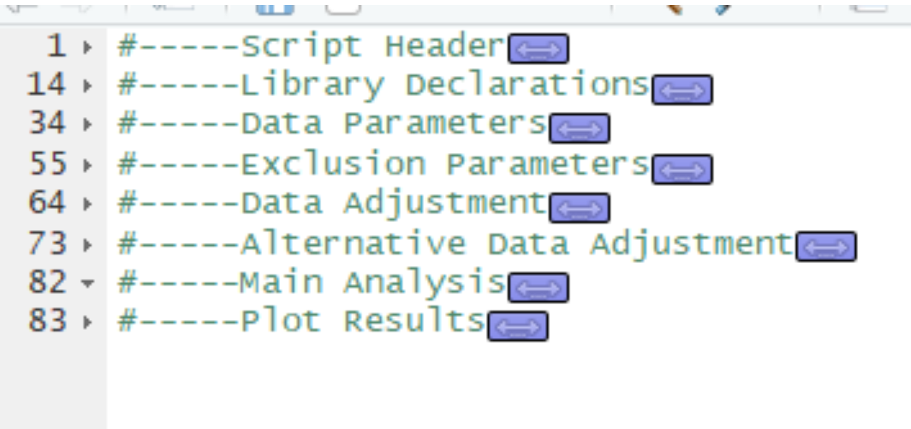
- R Studio feature
- A comment line that ends in 4 “-”, “=“ or “#”
- Find more info on Rstudio help or website!
- Also works in Rmarkdown code chunks



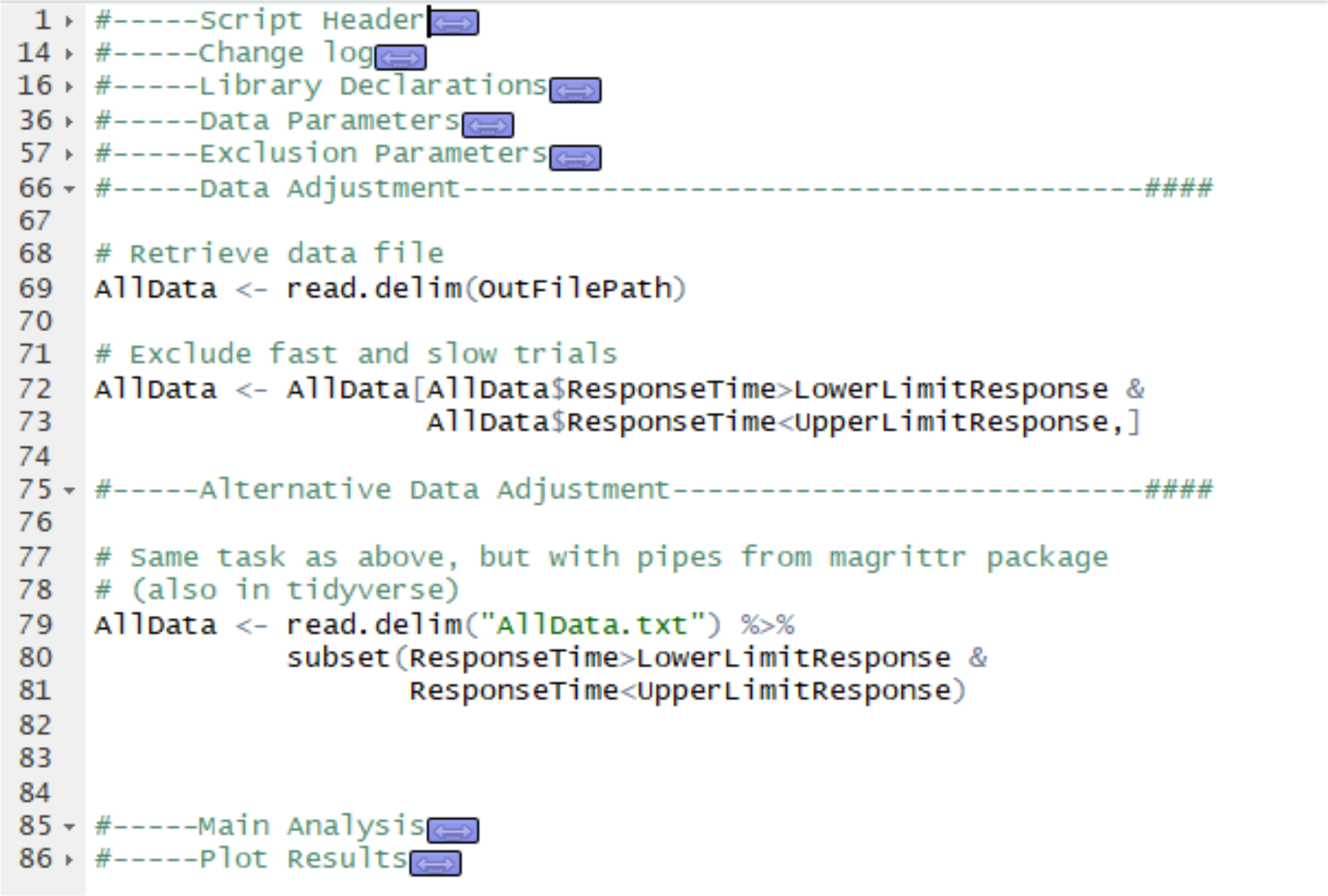
```
#-----Script Header--------------------------------------------####
# Date:          10.06.2020
# Author:        Arushi Garg
# Filename:      RL_Best_Practices_Sample.R
# Description: Code to present examples of best practices that
#                can be used while programming.This code is an
#                accompaniment to the presentation on the same
#                topic. Details of executing this script are
#                provided in document named "Best Practice Script
#                Detail.pdf" available in the project folder
# Project:       R Ladies Best Practices Workshop on 11 June 2020
#--------------------------------------------------------------###
```

Section Headers

- Utilise code folding features!



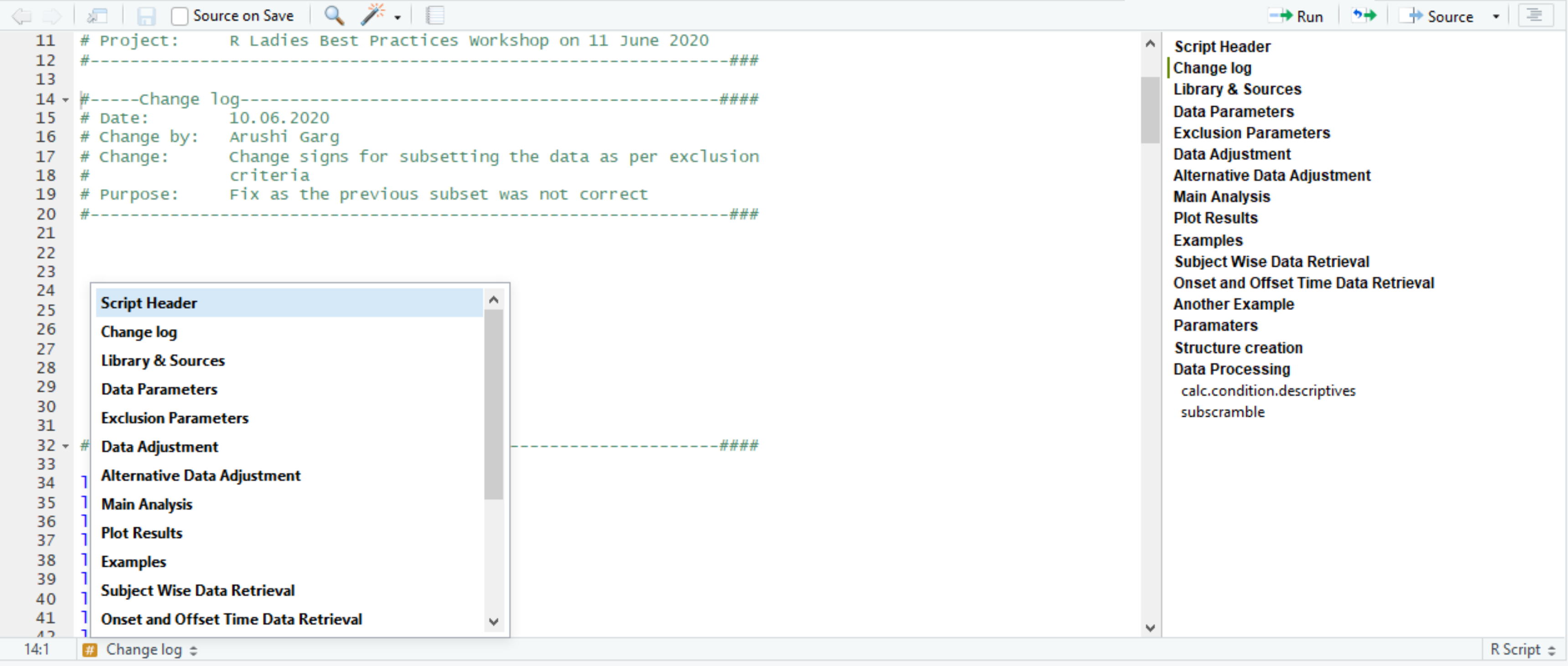
```
#-----Script Header
#-----Library Declarations
#-----Data Parameters
#-----Exclusion Parameters
#-----Data Adjustment
#-----Alternative Data Adjustment
#-----Main Analysis
#-----Plot Results
```



```
#-----Script Header
#-----Change log
#-----Library Declarations
#-----Data Parameters
#-----Exclusion Parameters
#-----Data Adjustment---------------------------------------####

# Retrieve data file
AllData <- read.delim(OutFilePath)

# Exclude fast and slow trials
AllData <- AllData[AllData$ResponseTime>LowerLimitResponse &
                   AllData$ResponseTime<UpperLimitResponse,]

#-----Alternative Data Adjustment---------------------------####

# Same task as above, but with pipes from magrittr package
# (also in tidyverse)
AllData <- read.delim("AllData.txt") %>%
           subset(ResponseTime>LowerLimitResponse &
                  ResponseTime<UpperLimitResponse)



#-----Main Analysis
#-----Plot Results
```

Section Headers

- Utilise code folding features!

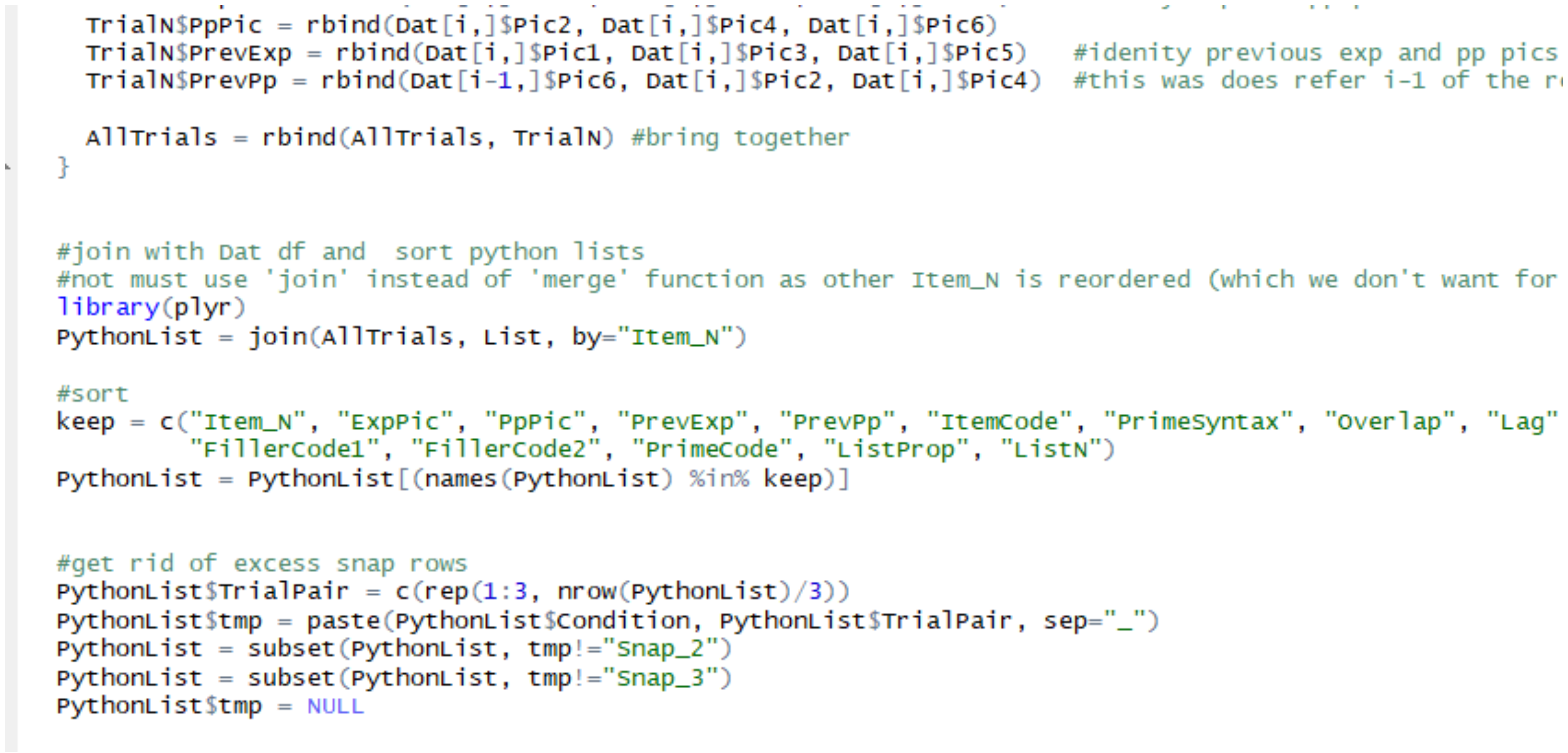


```
# Project:     R Ladies Best Practices Workshop on 11 June 2020
#-----------------------------------------------------------------###

#-----Change log-------------------------------------------------####
# Date:        10.06.2020
# Change by:   Arushi Garg
# Change:      Change signs for subsetting the data as per exclusion
#              criteria
# Purpose:     Fix as the previous subset was not correct
#-----------------------------------------------------------------###
```

- ❑ Can be helpful to navigate using the menu on the right side or at the bottom

Library declarations: What NOT to do

Why?



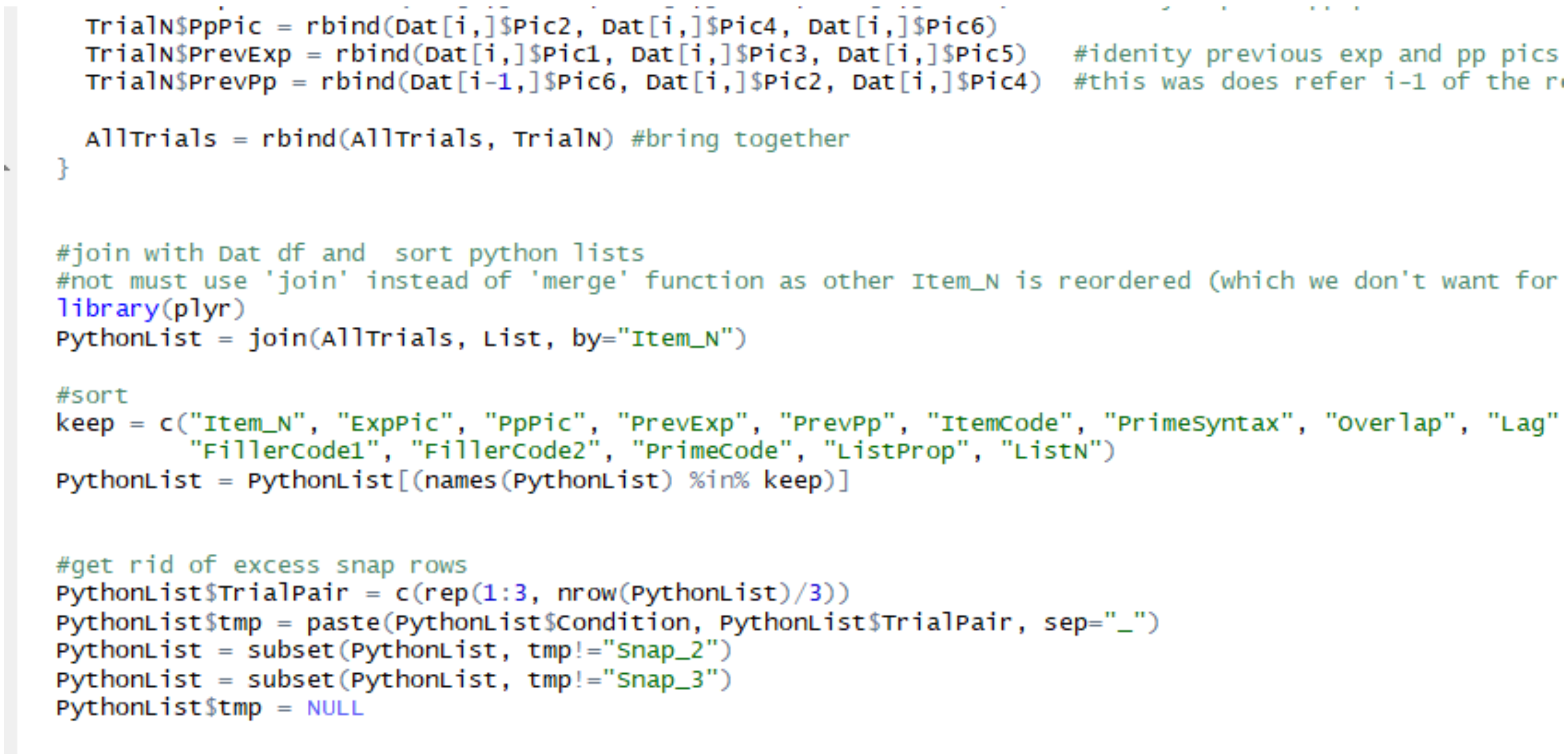
```
    TrialN$PpPic = rbind(Dat[i,]$Pic2, Dat[i,]$Pic4, Dat[i,]$Pic6)
    TrialN$PrevExp = rbind(Dat[i,]$Pic1, Dat[i,]$Pic3, Dat[i,]$Pic5)   #idenity previous exp and pp pics
    TrialN$PrevPp = rbind(Dat[i-1,]$Pic6, Dat[i,]$Pic2, Dat[i,]$Pic4)  #this was does refer i-1 of the r

    AllTrials = rbind(AllTrials, TrialN) #bring together
  }


  #join with Dat df and  sort python lists
  #not must use 'join' instead of 'merge' function as other Item_N is reordered (which we don't want for
  library(plyr)
  PythonList = join(AllTrials, List, by="Item_N")

  #sort
  keep = c("Item_N", "ExpPic", "PpPic", "PrevExp", "PrevPp", "ItemCode", "PrimeSyntax", "Overlap", "Lag"
           "FillerCode1", "FillerCode2", "PrimeCode", "ListProp", "ListN")
  PythonList = PythonList[(names(PythonList) %in% keep)]


  #get rid of excess snap rows
  PythonList$TrialPair = c(rep(1:3, nrow(PythonList)/3))
  PythonList$tmp = paste(PythonList$Condition, PythonList$TrialPair, sep="_")
  PythonList = subset(PythonList, tmp!="Snap_2")
  PythonList = subset(PythonList, tmp!="Snap_3")
  PythonList$tmp = NULL
```

- ❑ library or requirement statements should be avoided within the code

Library declarations: What NOT to do

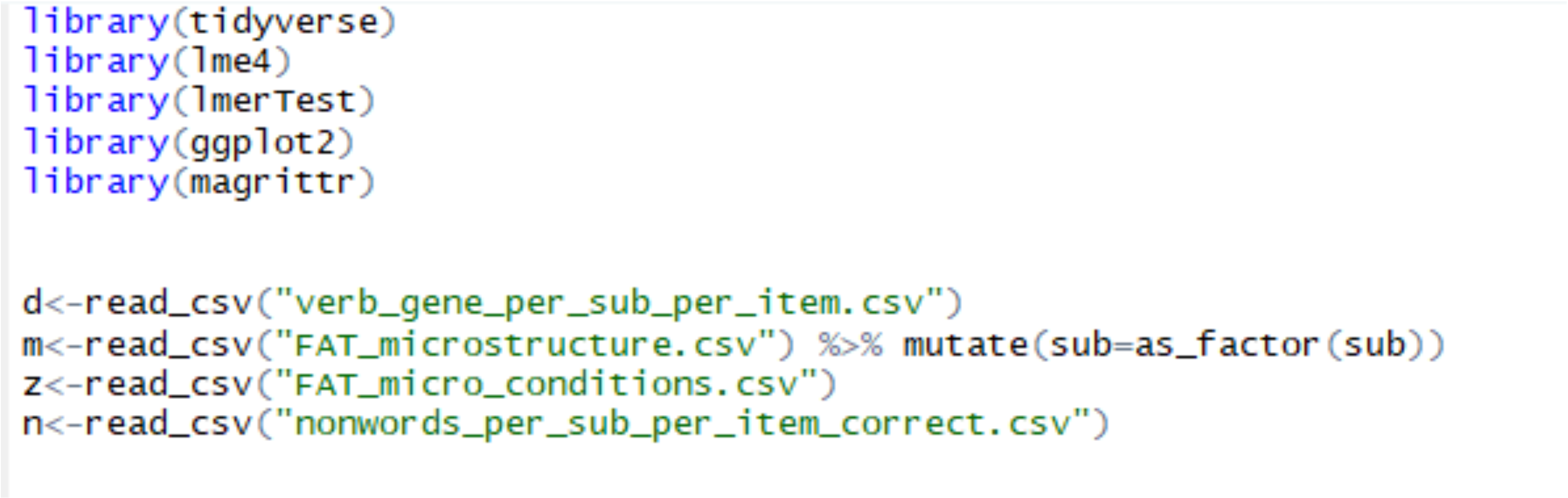
Why?

- Packages are basic requirements; should be visible at the top
- You might not have something available, and only realise after 2 hours (or 2 days!) of execution of the code that came before it.



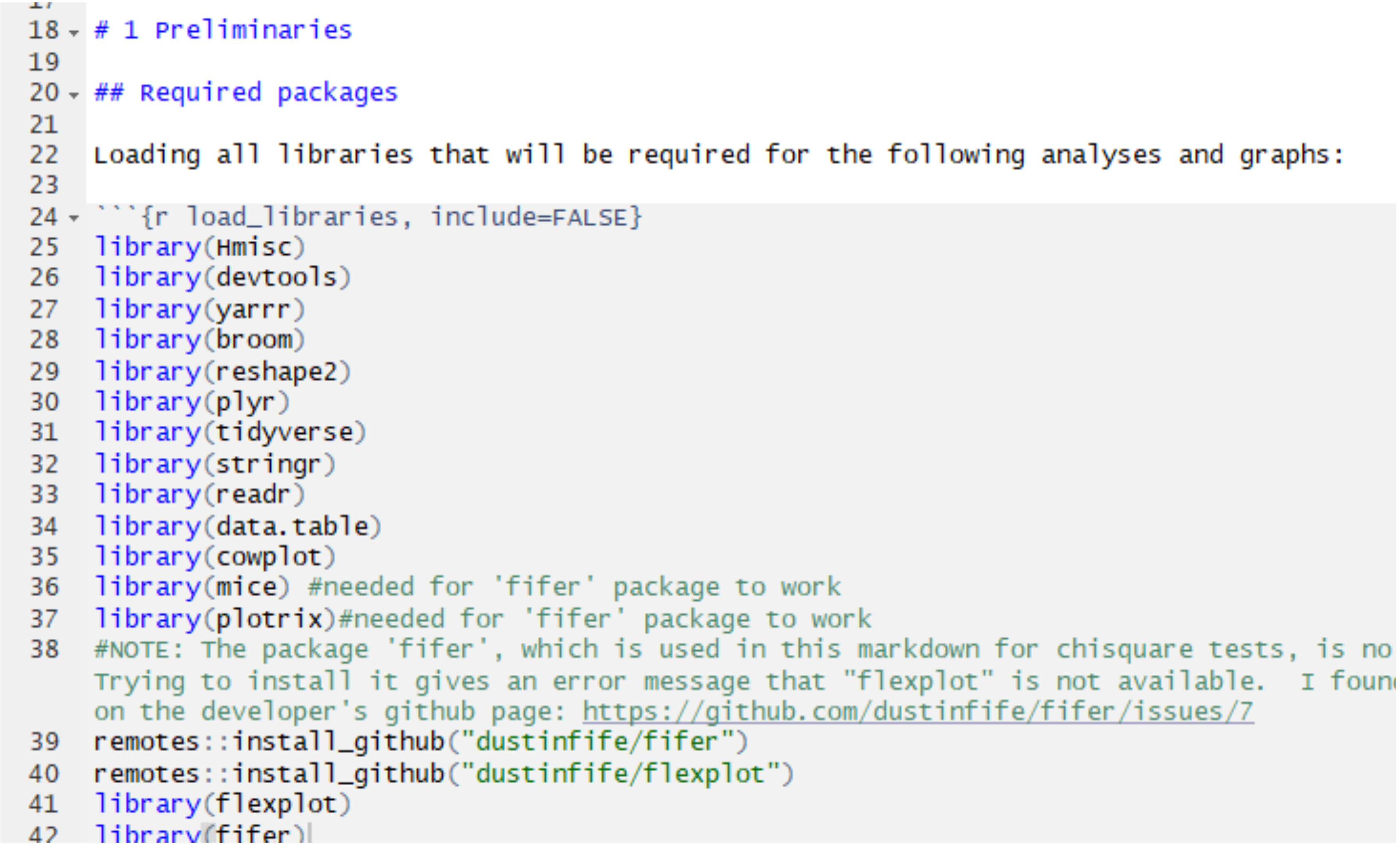
```
    TrialN$PpPic = rbind(Dat[i,]$Pic2, Dat[i,]$Pic4, Dat[i,]$Pic6)
    TrialN$PrevExp = rbind(Dat[i,]$Pic1, Dat[i,]$Pic3, Dat[i,]$Pic5)   #idenity previous exp and pp pics
    TrialN$PrevPp = rbind(Dat[i-1,]$Pic6, Dat[i,]$Pic2, Dat[i,]$Pic4)  #this was does refer i-1 of the r

    AllTrials = rbind(AllTrials, TrialN) #bring together
  }


#join with Dat df and  sort python lists
#not must use 'join' instead of 'merge' function as other Item_N is reordered (which we don't want for
library(plyr)
PythonList = join(AllTrials, List, by="Item_N")

#sort
keep = c("Item_N", "ExpPic", "PpPic", "PrevExp", "PrevPp", "ItemCode", "PrimeSyntax", "Overlap", "Lag"
         "FillerCode1", "FillerCode2", "PrimeCode", "ListProp", "ListN")
PythonList = PythonList[(names(PythonList) %in% keep)]


#get rid of excess snap rows
PythonList$TrialPair = c(rep(1:3, nrow(PythonList)/3))
PythonList$tmp = paste(PythonList$Condition, PythonList$TrialPair, sep="_")
PythonList = subset(PythonList, tmp!="Snap_2")
PythonList = subset(PythonList, tmp!="Snap_3")
PythonList$tmp = NULL
```

Library declarations: Better



```
library(tidyverse)
library(lme4)
library(lmerTest)
library(ggplot2)
library(magrittr)


d<-read_csv("verb_gene_per_sub_per_item.csv")
m<-read_csv("FAT_microstructure.csv") %>% mutate(sub=as_factor(sub))
z<-read_csv("FAT_micro_conditions.csv")
n<-read_csv("nonwords_per_sub_per_item_correct.csv")
```

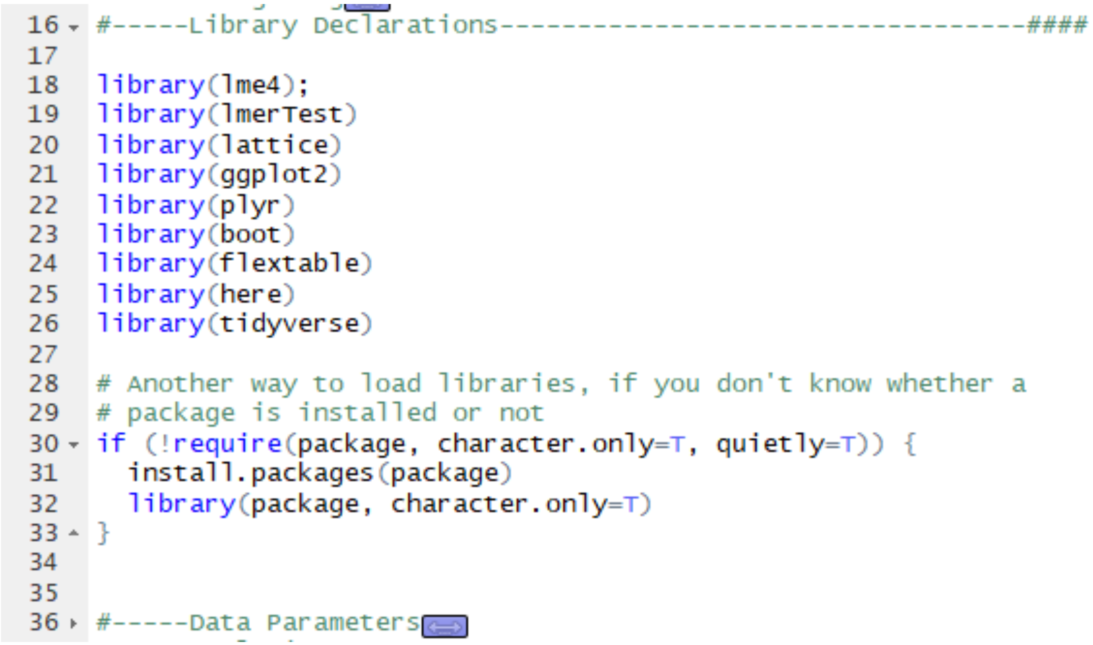
Library declarations: Better



~~~
# 1 Preliminaries

## Required packages

Loading all libraries that will be required for the following analyses and graphs:

```{r load_libraries, include=FALSE}
library(Hmisc)
library(devtools)
library(yarrr)
library(broom)
library(reshape2)
library(plyr)
library(tidyverse)
library(stringr)
library(readr)
library(data.table)
library(cowplot)
library(mice) #needed for 'fifer' package to work
library(plotrix)#needed for 'fifer' package to work
#NOTE: The package 'fifer', which is used in this markdown for chisquare tests, is no
Trying to install it gives an error message that "flexplot" is not available.  I foun
on the developer's github page: https://github.com/dustinfife/fifer/issues/7
remotes::install_github("dustinfife/fifer")
remotes::install_github("dustinfife/flexplot")
library(flexplot)
library(fifer)
~~~

Library declarations : good

Make sure to remove any package calls that you are not using in the code



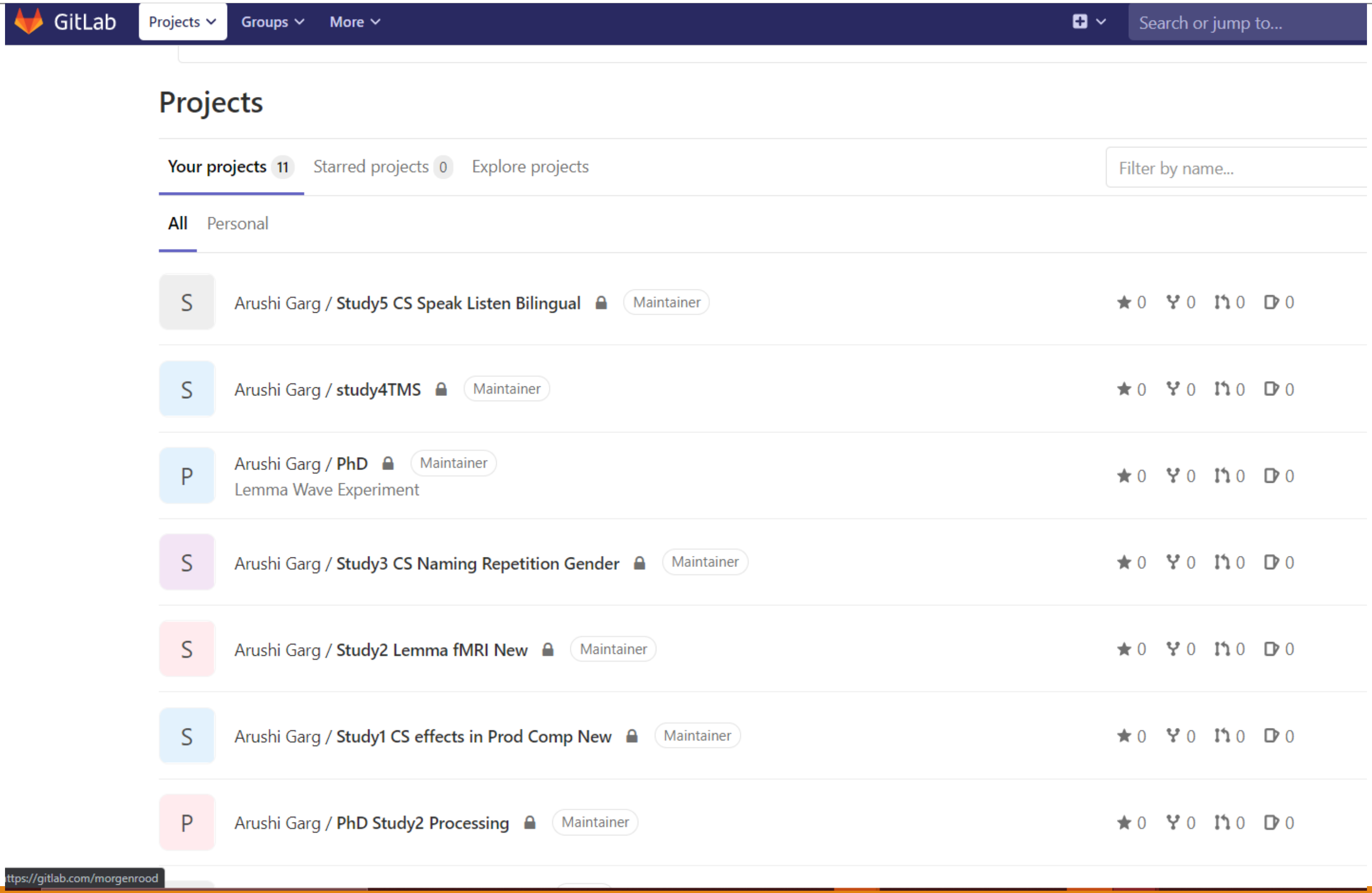
```r
#-----Library Declarations----------------------------------####

library(lme4);
library(lmerTest)
library(lattice)
library(ggplot2)
library(plyr)
library(boot)
library(flextable)
library(here)
library(tidyverse)

# Another way to load libraries, if you don't know whether a
# package is installed or not
if (!require(package, character.only=T, quietly=T)) {
  install.packages(package)
  library(package, character.only=T)
}


#-----Data Parameters
```

Library declarations are best at the top of the code within a separate section fold of their own

Version Control

- ❑Critical in industry
- ❑ Almost nobody practices it in academia
  - ❑unless they develop software that is deployed for someone outside their group
- ❑Can be VERY useful for the development of your analysis
  - ❑– since you will tend to change things here and there in your scripts and then forget about it and then wonder what your last working version was.
- ❑Several ways of practicing version control
  - ❑ Version Control Software
  - ❑ Separate File Names
  - ❑ Within Code Change Logs (+ Documentation)

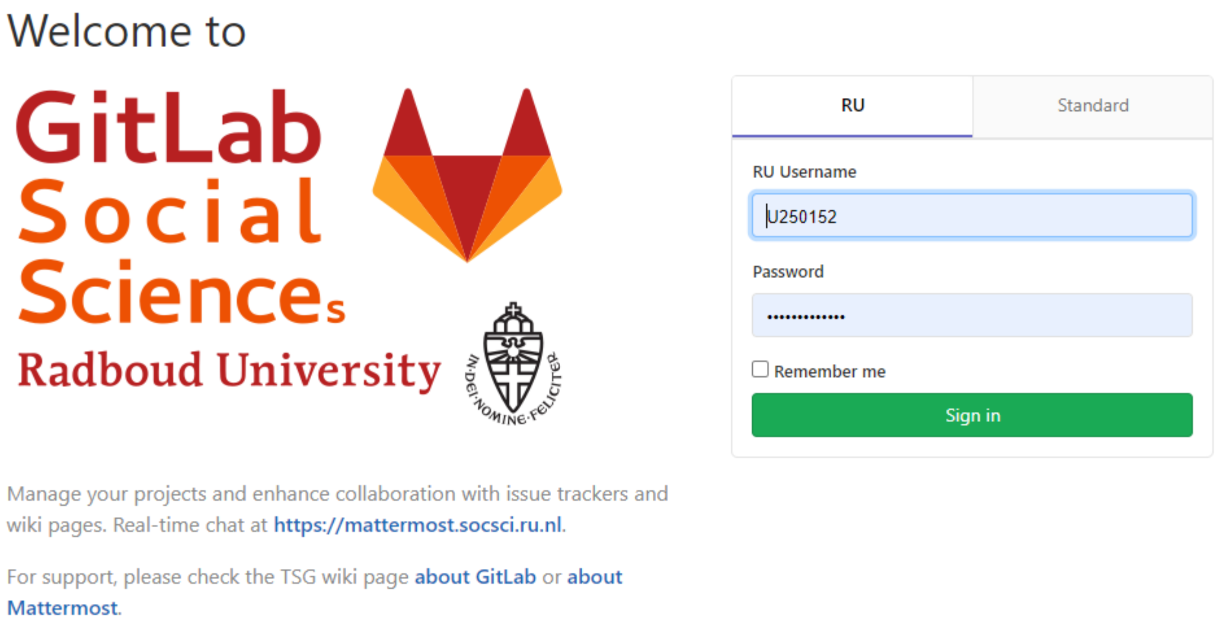
Version Control - Using Version Control Software (GIT or SVN)



Projects

S	Arushi Garg / Study5 CS Speak Listen Bilingual	Maintainer	★ 0	0	0	0
S	Arushi Garg / study4TMS	Maintainer	★ 0	0	0	0
P	Arushi Garg / PhD Lemma Wave Experiment	Maintainer	★ 0	0	0	0
S	Arushi Garg / Study3 CS Naming Repetition Gender	Maintainer	★ 0	0	0	0
S	Arushi Garg / Study2 Lemma fMRI New	Maintainer	★ 0	0	0	0
S	Arushi Garg / Study1 CS effects in Prod Comp New	Maintainer	★ 0	0	0	0
P	Arushi Garg / PhD Study2 Processing	Maintainer	★ 0	0	0	0

Version Control - Using Version Control Software (GIT or SVN)



Welcome to

GitLab Social Sciences Radboud University

Manage your projects and enhance collaboration with issue trackers and wiki pages. Real-time chat at https://mattermost.socsci.ru.nl.

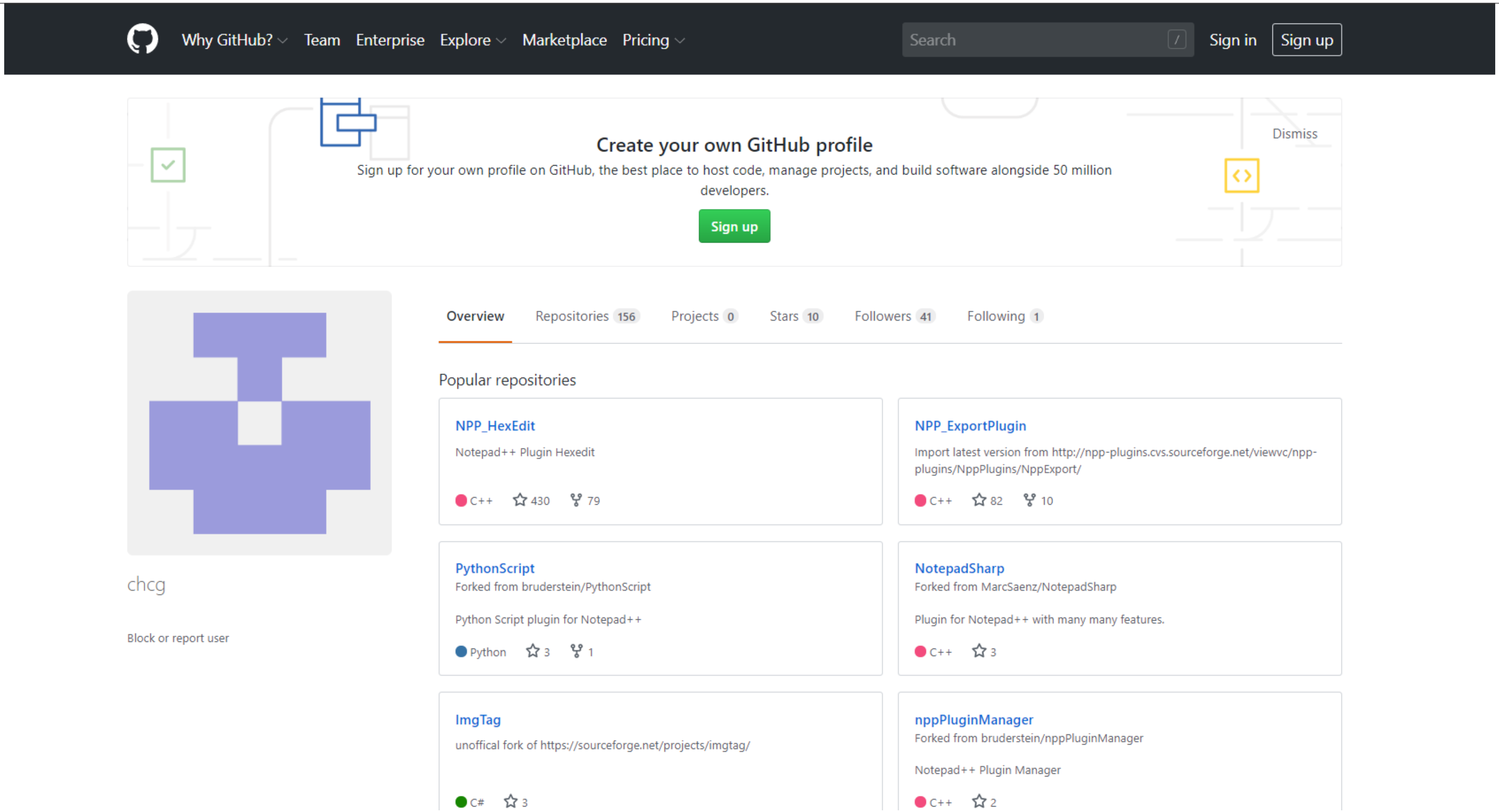
For support, please check the TSG wiki page about GitLab or about Mattermost.

RU Username: U250152

Password: •••••••••••••

□ Remember me

Version Control - Using Version Control Software (GIT or SVN)



Create your own GitHub profile

Sign up for your own profile on GitHub, the best place to host code, manage projects, and build software alongside 50 million developers.

chcg

Block or report user

Popular repositories

NPP_HexEdit
Notepad++ Plugin Hexedit
● C++ ☆ 430 79

NPP_ExportPlugin
Import latest version from http://npp-plugins.cvs.sourceforge.net/viewvc/npp-plugins/NppPlugins/NppExport/
● C++ ☆ 82 10

PythonScript
Forked from bruderstein/PythonScript
Python Script plugin for Notepad++
● Python ☆ 3 1

NotepadSharp
Forked from MarcSaenz/NotepadSharp
Plugin for Notepad++ with many many features.
● C++ ☆ 3

ImgTag
unoffical fork of https://sourceforge.net/projects/imgtag/
● C# ☆ 3

nppPluginManager
Forked from bruderstein/nppPluginManager
Notepad++ Plugin Manager
● C++ ☆ 2

Version Control - Using Version Control Software (GIT or SVN)

ADVANTAGES

- ❑ Saved code history; revert to any version
- ❑ Branch your project; try a different analysis
- ❑ Choose level of technicality; can function minimally with basic commands
- ❑ Sync across devices
- ❑ Great for collaboration
  - ❑ Easy to share code
  - ❑ Easy to share changes after sharing preliminary

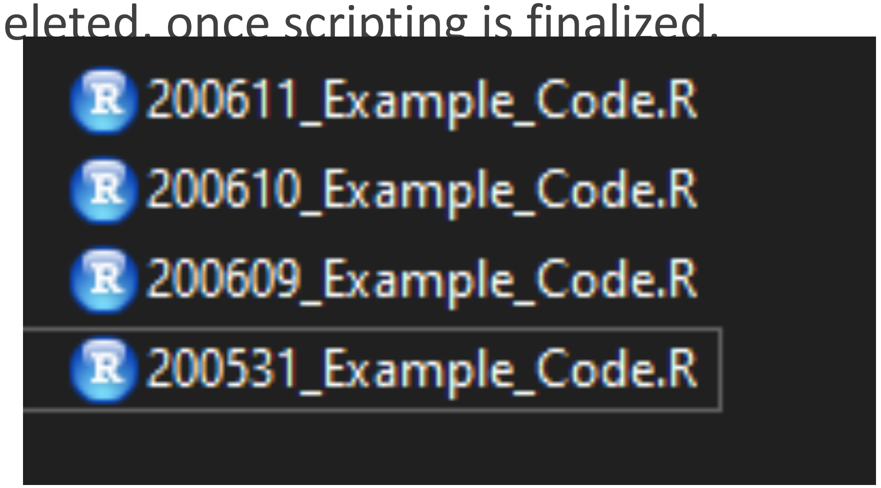
Version Control - Using Version Control Software (GIT or SVN)

DISADVANTAGES

- ❑ Requires initial investment of time and effort
- ❑ Bad if you forget to push and/or commit changes
  - ❑– only to remember it later when you don’t remember what, why or how of the changes you made
- ❑ Useless, unless good, informative change messages used
- ❑ Ever expanding repository

Version Control – Separate File Names

- Use date before the file name (or at the end, before the file extension) to mark the latest version of your file
  - Most helpful date format: YYYYMMDD or YYMMDD (222200611 or 220611)
- Important change script header log to document change
- Can be used in combination with the next way of version control
- Older versions can be deleted, once scripting is finalized.



```
200611_Example_Code.R
200610_Example_Code.R
200609_Example_Code.R
200531_Example_Code.R
```

Version Control – Separate File Names

Advantages

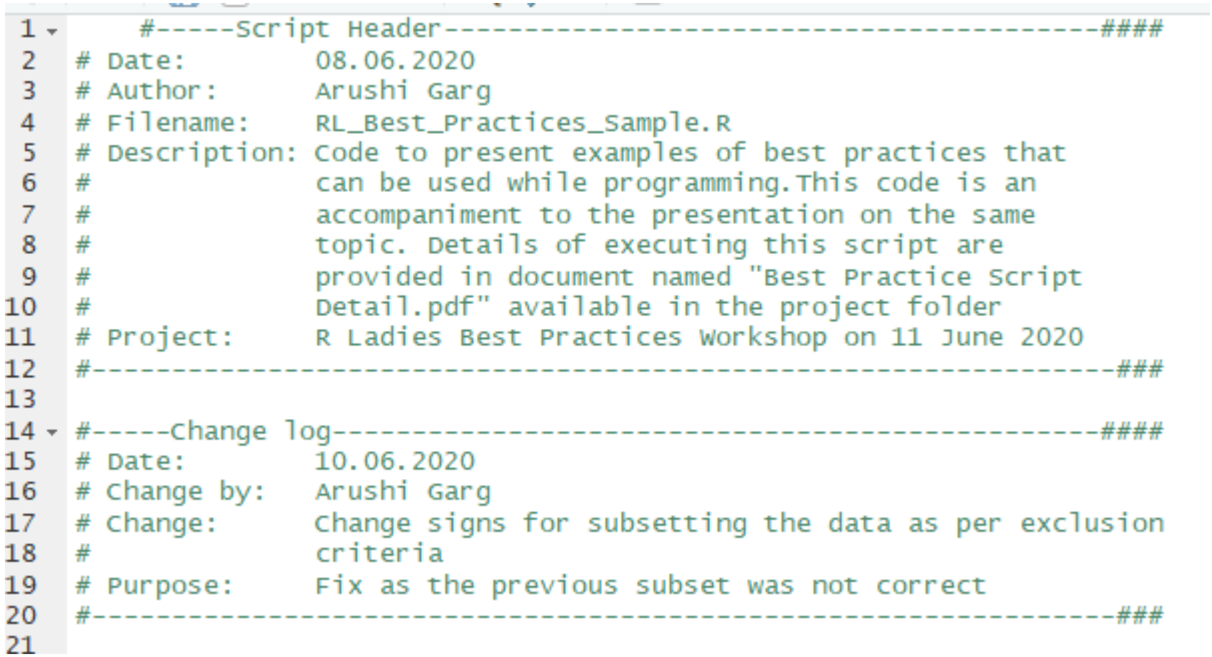
- ❑ No learning curve; you can start doing it straightaway
- ❑ Maintain history of previous changes
- ❑ Easy to revert to a previous version by simply picking a previous file
- ❑ No code syncing

Version Control – Separate File Names

Disadvantages

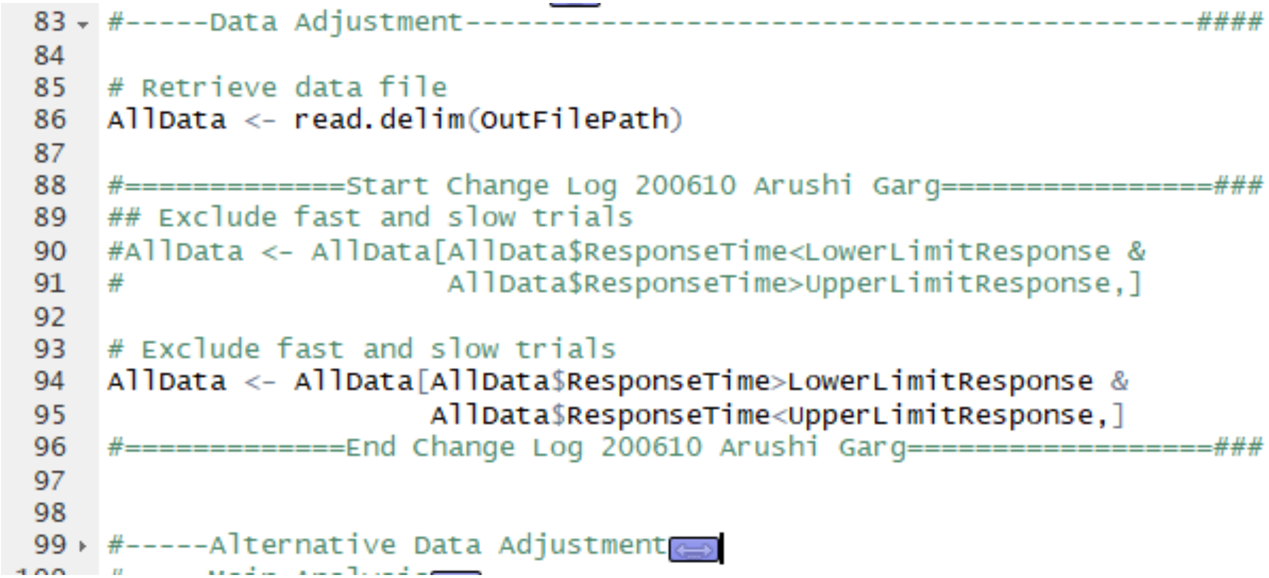
- ❑ Not good for collaboration
- ❑ Hard to see which version has what changes; you have to open each file
- ❑ Let’s say you are coding on a Wednesday. You don’t like changes you made on Tuesday and so you went back to the Monday version. It is hard to keep track without opening individual files and checking that the Wednesday version is not a continuation of Tuesday but of Monday.
- ❑ Disk storage size can keep on increasing, with every new file
- ❑ Important to remember to include change in header!
- ❑ Tedious to compare changes across versions
- ❑ Accidental saves of old file with new changes are likely

Version Control - Within Code Change Logs (+ Documentation)



```
   #-----Script Header------------------------------------------####
# Date:        08.06.2020
# Author:      Arushi Garg
# Filename:    RL_Best_Practices_Sample.R
# Description: Code to present examples of best practices that
#              can be used while programming.This code is an
#              accompaniment to the presentation on the same
#              topic. Details of executing this script are
#              provided in document named "Best Practice Script
#              Detail.pdf" available in the project folder
# Project:     R Ladies Best Practices Workshop on 11 June 2020
#-----------------------------------------------------------------###

#-----Change log----------------------------------------------####
# Date:        10.06.2020
# Change by:   Arushi Garg
# Change:      Change signs for subsetting the data as per exclusion
#              criteria
# Purpose:     Fix as the previous subset was not correct
#-----------------------------------------------------------------###
```

Below the script header, there is a section called the Change Log. All of the different versions of the code are logged there



```
#-----Data Adjustment-------------------------------------------####

# Retrieve data file
AllData <- read.delim(OutFilePath)

#=============Start Change Log 200610 Arushi Garg================###
## Exclude fast and slow trials
#AllData <- AllData[AllData$ResponseTime<LowerLimitResponse &
#                    AllData$ResponseTime>UpperLimitResponse,]

# Exclude fast and slow trials
AllData <- AllData[AllData$ResponseTime>LowerLimitResponse &
                   AllData$ResponseTime<UpperLimitResponse,]
#=============End Change Log 200610 Arushi Garg==================###


#-----Alternative Data Adjustment
```

The corresponding changes are marked in the main code with comment lines

Version Control - Within Code Change Logs (+ Documentation)

ADVANTAGES

- No initial investment of time and effort to set it up and understand the basic commands and concepts
- No multiple files needed
- This does not require anything outside of your code to maintain the versions
- No conflicts with other files
- You see all your changes and versions in the same file.
- Easy to see what changes happened when.
- Commented out pieces of old code can be deleted once code is finalised (or a new final version can be made without the change log or changes at the end)

Version Control - Within Code Change Logs (+ Documentation)

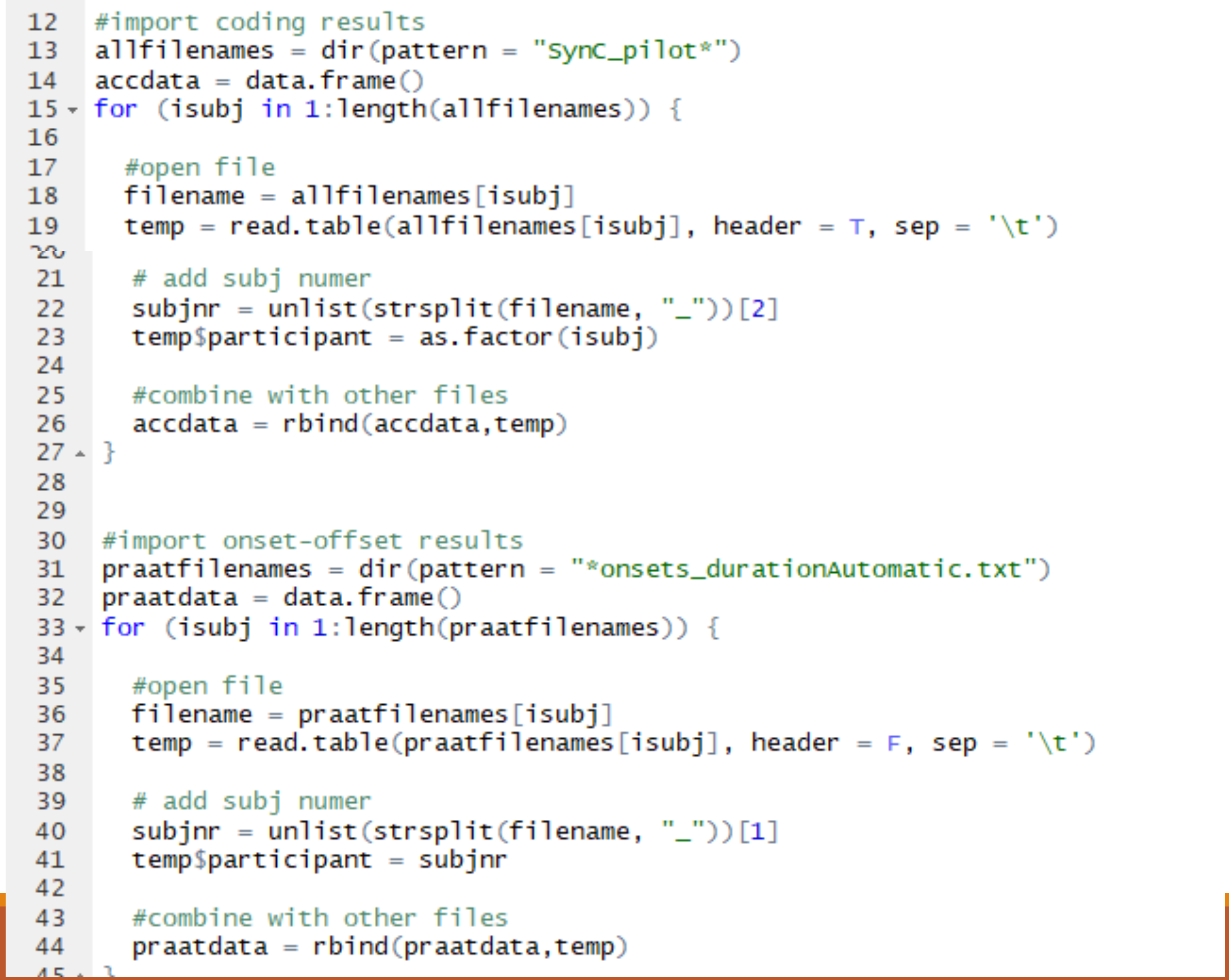
DISADVANTAGES

- ❑ Not an elegant solution; obsolete – people with more software experience can resist change logs
- ❑ Difficult to share changes to your code
- ❑ Falls midway between the other two options for ease in collaboration

Commenting

- ❑ Appropriate and thorough commenting is imperative. It helps with:
  - ❑ Readability of the code
  - ❑ Sharing; others can understand and verify steps easily
  - ❑ Revisiting your code even after years
  - ❑ Making changes and adjustment of code
- ❑ Advice on commenting ranges from *commenting on every line* to *commenting as less as possible*, while maintaining readability
- ❑ Comments should establish balance between under- and over-explaining
- ❑ Generally recommended to answer “why” question, rather than “what”
- ❑ Should be helpful beyond what the code is telling; otherwise it is just clutter
- ❑ Don’t leave all the heavy lifting to comments -> name variable and functions to be self-explanatory

Commenting: What NOT to do



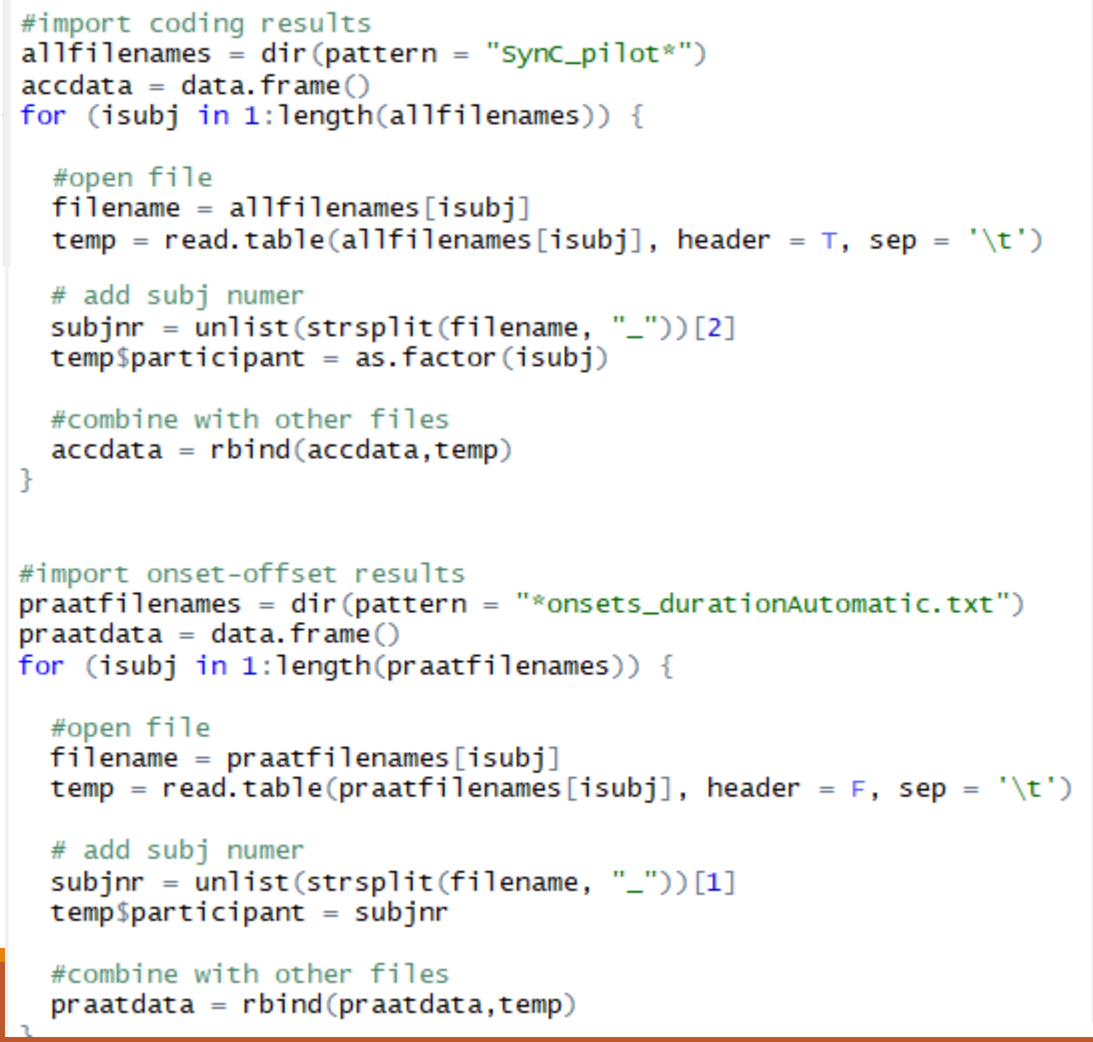
```r
#import coding results
allfilenames = dir(pattern = "SynC_pilot*")
accdata = data.frame()
for (isubj in 1:length(allfilenames)) {

  #open file
  filename = allfilenames[isubj]
  temp = read.table(allfilenames[isubj], header = T, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[2]
  temp$participant = as.factor(isubj)

  #combine with other files
  accdata = rbind(accdata,temp)
}


#import onset-offset results
praatfilenames = dir(pattern = "*onsets_durationAutomatic.txt")
praatdata = data.frame()
for (isubj in 1:length(praatfilenames)) {

  #open file
  filename = praatfilenames[isubj]
  temp = read.table(praatfilenames[isubj], header = F, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[1]
  temp$participant = subjnr

  #combine with other files
  praatdata = rbind(praatdata,temp)
}
```

Anyone who knows basic R knows what’s happening in those lines!

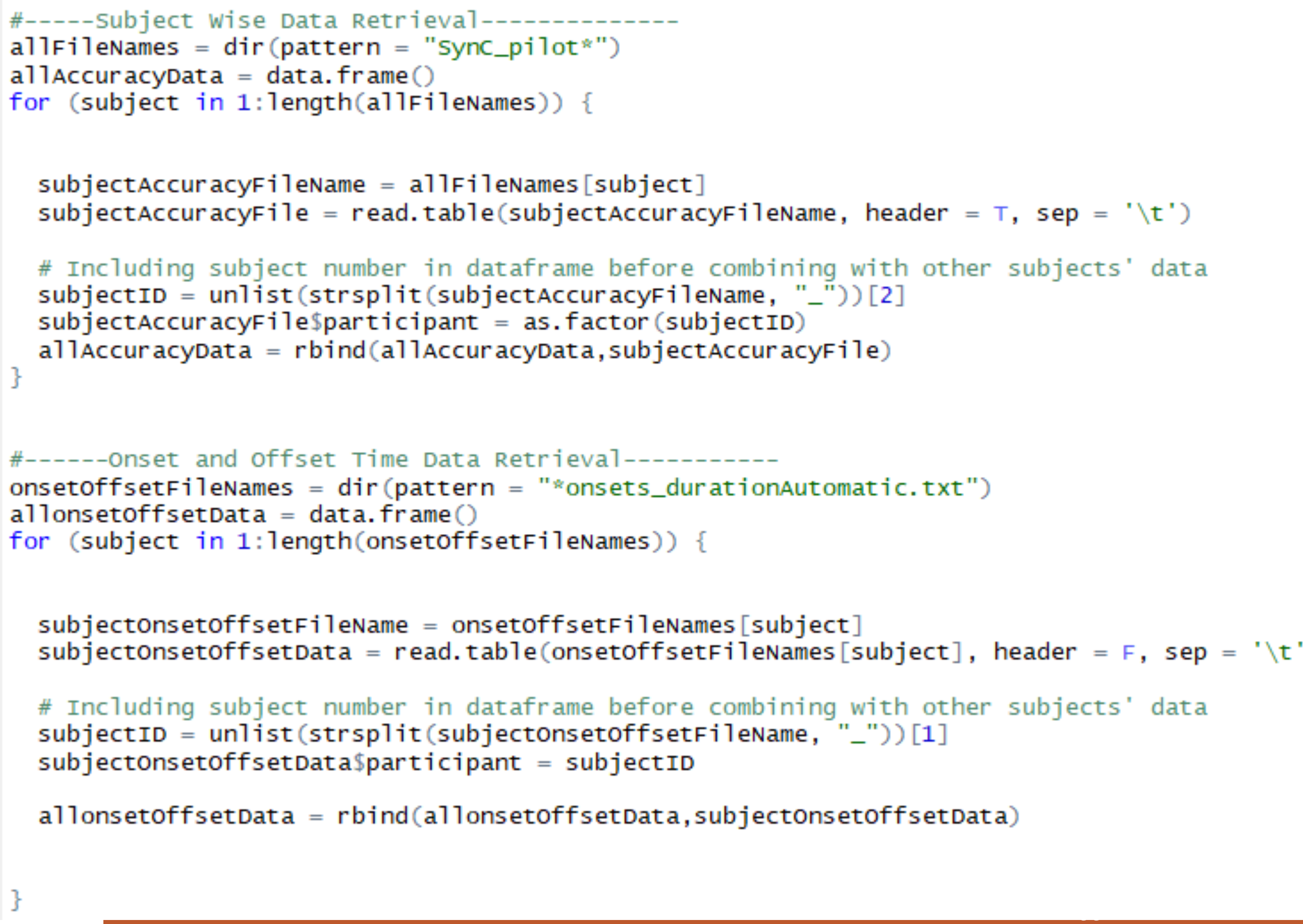
Comments should be helpful beyond what the code is clearly saying

Define the purpose!

Commenting: What could be done instead

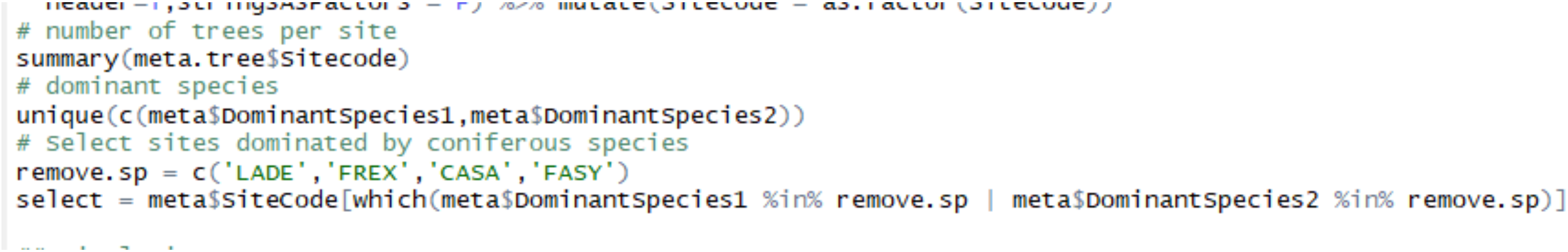


```
#import coding results
allfilenames = dir(pattern = "SynC_pilot*")
accdata = data.frame()
for (isubj in 1:length(allfilenames)) {

  #open file
  filename = allfilenames[isubj]
  temp = read.table(allfilenames[isubj], header = T, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[2]
  temp$participant = as.factor(isubj)

  #combine with other files
  accdata = rbind(accdata,temp)
}


#import onset-offset results
praatfilenames = dir(pattern = "*onsets_durationAutomatic.txt")
praatdata = data.frame()
for (isubj in 1:length(praatfilenames)) {

  #open file
  filename = praatfilenames[isubj]
  temp = read.table(praatfilenames[isubj], header = F, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[1]
  temp$participant = subjnr

  #combine with other files
  praatdata = rbind(praatdata,temp)
}
```



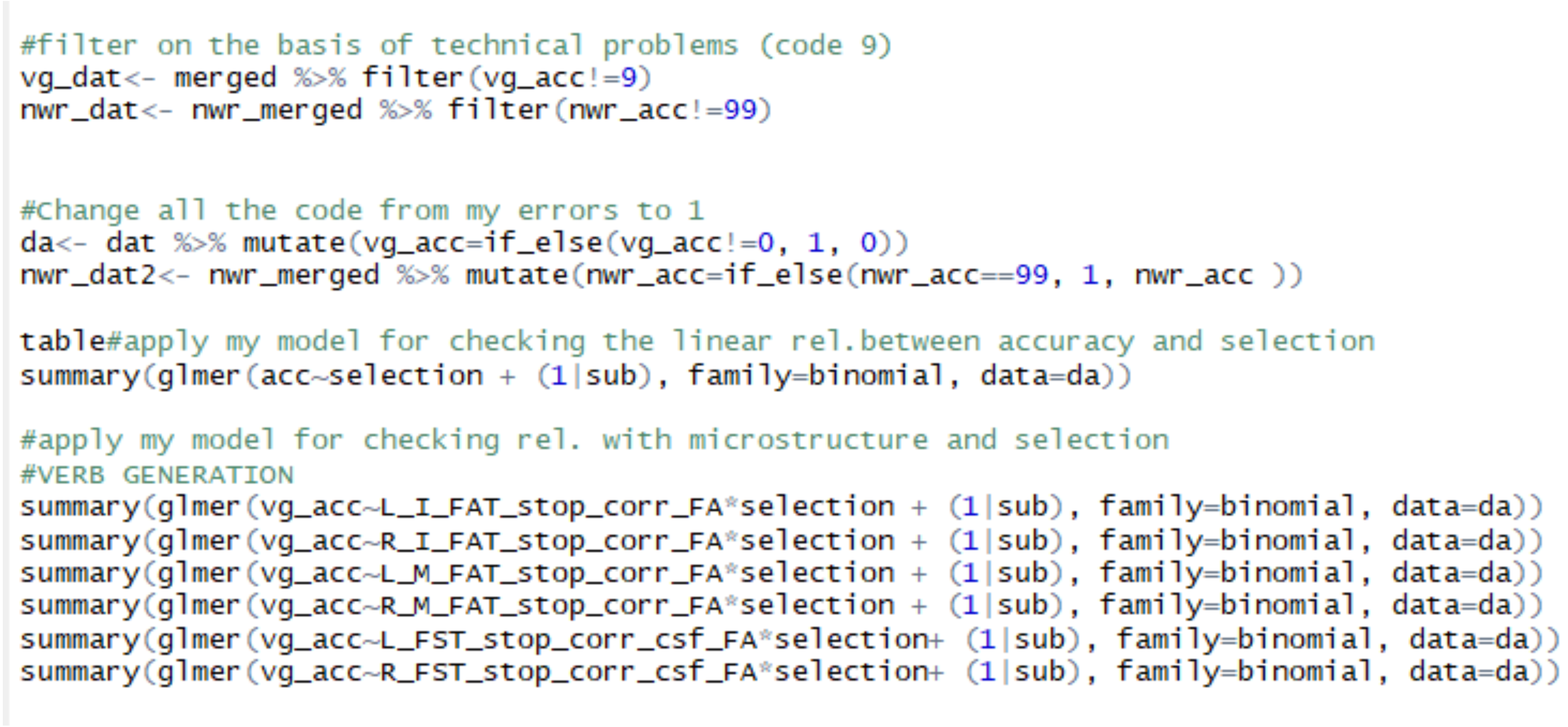
```
#-----Subject Wise Data Retrieval--------------
allFileNames = dir(pattern = "SynC_pilot*")
allAccuracyData = data.frame()
for (subject in 1:length(allFileNames)) {


  subjectAccuracyFileName = allFileNames[subject]
  subjectAccuracyFile = read.table(subjectAccuracyFileName, header = T, sep = '\t')

  # Including subject number in dataframe before combining with other subjects' data
  subjectID = unlist(strsplit(subjectAccuracyFileName, "_"))[2]
  subjectAccuracyFile$participant = as.factor(subjectID)
  allAccuracyData = rbind(allAccuracyData,subjectAccuracyFile)
}


#------Onset and Offset Time Data Retrieval-----------
onsetOffsetFileNames = dir(pattern = "*onsets_durationAutomatic.txt")
allOnsetOffsetData = data.frame()
for (subject in 1:length(onsetOffsetFileNames)) {


  subjectOnsetOffsetFileName = onsetOffsetFileNames[subject]
  subjectOnsetOffsetData = read.table(onsetOffsetFileNames[subject], header = F, sep = '\t'

  # Including subject number in dataframe before combining with other subjects' data
  subjectID = unlist(strsplit(subjectOnsetOffsetFileName, "_"))[1]
  subjectOnsetOffsetData$participant = subjectID

  allOnsetOffsetData = rbind(allOnsetOffsetData,subjectOnsetOffsetData)


}
```

Commenting: What NOT to do



```
  header=T,stringsAsFactors = F) %>% mutate(SiteCode = as.factor(SiteCode))
# number of trees per site
summary(meta.tree$Sitecode)
# dominant species
unique(c(meta$DominantSpecies1,meta$DominantSpecies2))
# Select sites dominated by coniferous species
remove.sp = c('LADE','FREX','CASA','FASY')
select = meta$SiteCode[which(meta$DominantSpecies1 %in% remove.sp | meta$DominantSpecies2 %in% remove.sp)]
```

Commenting: What NOT to do



```
#filter on the basis of technical problems (code 9)
vg_dat<- merged %>% filter(vg_acc!=9)
nwr_dat<- nwr_merged %>% filter(nwr_acc!=99)


#Change all the code from my errors to 1
da<- dat %>% mutate(vg_acc=if_else(vg_acc!=0, 1, 0))
nwr_dat2<- nwr_merged %>% mutate(nwr_acc=if_else(nwr_acc==99, 1, nwr_acc ))

table#apply my model for checking the linear rel.between accuracy and selection
summary(glmer(acc~selection + (1|sub), family=binomial, data=da))

#apply my model for checking rel. with microstructure and selection
#VERB GENERATION
summary(glmer(vg_acc~L_I_FAT_stop_corr_FA*selection + (1|sub), family=binomial, data=da))
summary(glmer(vg_acc~R_I_FAT_stop_corr_FA*selection + (1|sub), family=binomial, data=da))
summary(glmer(vg_acc~L_M_FAT_stop_corr_FA*selection + (1|sub), family=binomial, data=da))
summary(glmer(vg_acc~R_M_FAT_stop_corr_FA*selection + (1|sub), family=binomial, data=da))
summary(glmer(vg_acc~L_FST_stop_corr_csf_FA*selection+ (1|sub), family=binomial, data=da))
summary(glmer(vg_acc~R_FST_stop_corr_csf_FA*selection+ (1|sub), family=binomial, data=da))
```

Commenting: examples from the Internet

```
function addSetEntry(set, value) {
/* Don't return `set.add` because it's not chainable in IE 11. */
set.add(value);
return set;
}
```

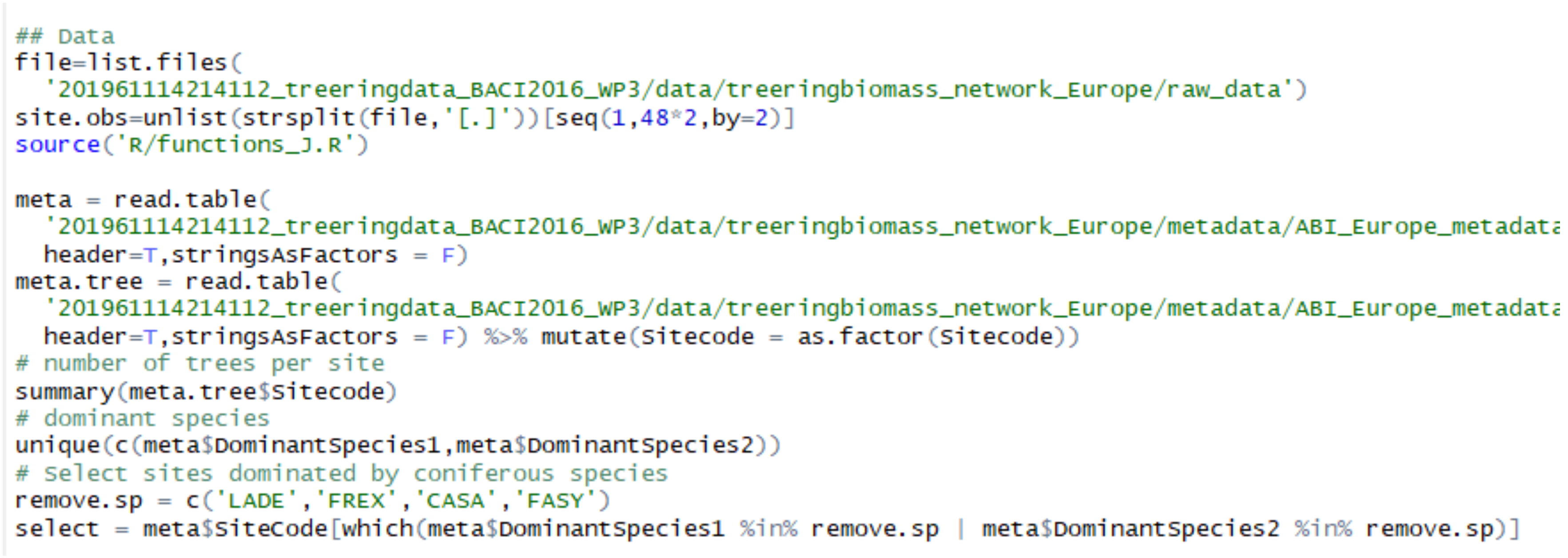
```
/* don't use the global isFinite() because it returns true for null values*/
Number.isFinite(value)
```

Note that these are not examples from R itself

Naming Conventions

Things to avoid:

- Inconsistency in naming
- Names that do not reveal exact purpose of variable/function/file
- Cryptic abbreviations

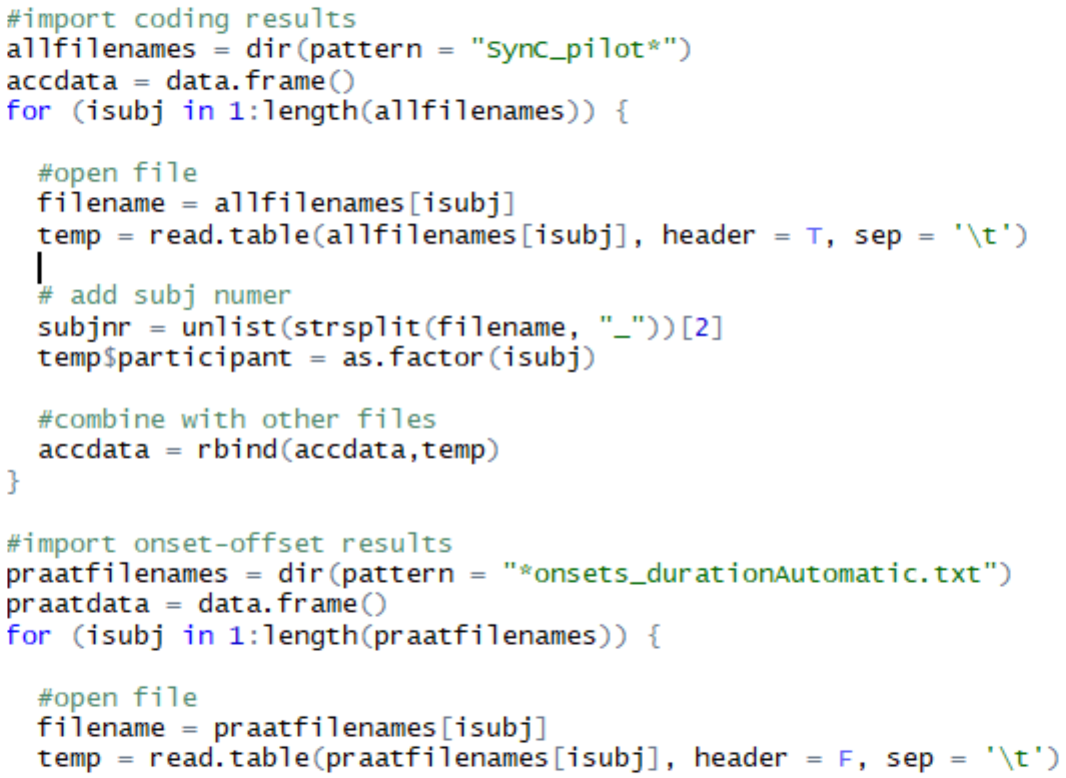


```
## Data
file=list.files(
  '201961114214112_treeringdata_BACI2016_WP3/data/treeringbiomass_network_Europe/raw_data')
site.obs=unlist(strsplit(file,'[.]'))[seq(1,48*2,by=2)]
source('R/functions_J.R')

meta = read.table(
  '201961114214112_treeringdata_BACI2016_WP3/data/treeringbiomass_network_Europe/metadata/ABI_Europe_metadata
  header=T,stringsAsFactors = F)
meta.tree = read.table(
  '201961114214112_treeringdata_BACI2016_WP3/data/treeringbiomass_network_Europe/metadata/ABI_Europe_metadata
  header=T,stringsAsFactors = F) %>% mutate(Sitecode = as.factor(Sitecode))
# number of trees per site
summary(meta.tree$Sitecode)
# dominant species
unique(c(meta$DominantSpecies1,meta$DominantSpecies2))
# Select sites dominated by coniferous species
remove.sp = c('LADE','FREX','CASA','FASY')
select = meta$SiteCode[which(meta$DominantSpecies1 %in% remove.sp | meta$DominantSpecies2 %in% remove.sp)]
```

Naming Conventions

Things to avoid:

- Inconsistency in naming
- Names that do not reveal exact purpose of variable/function/file
- Cryptic abbreviations

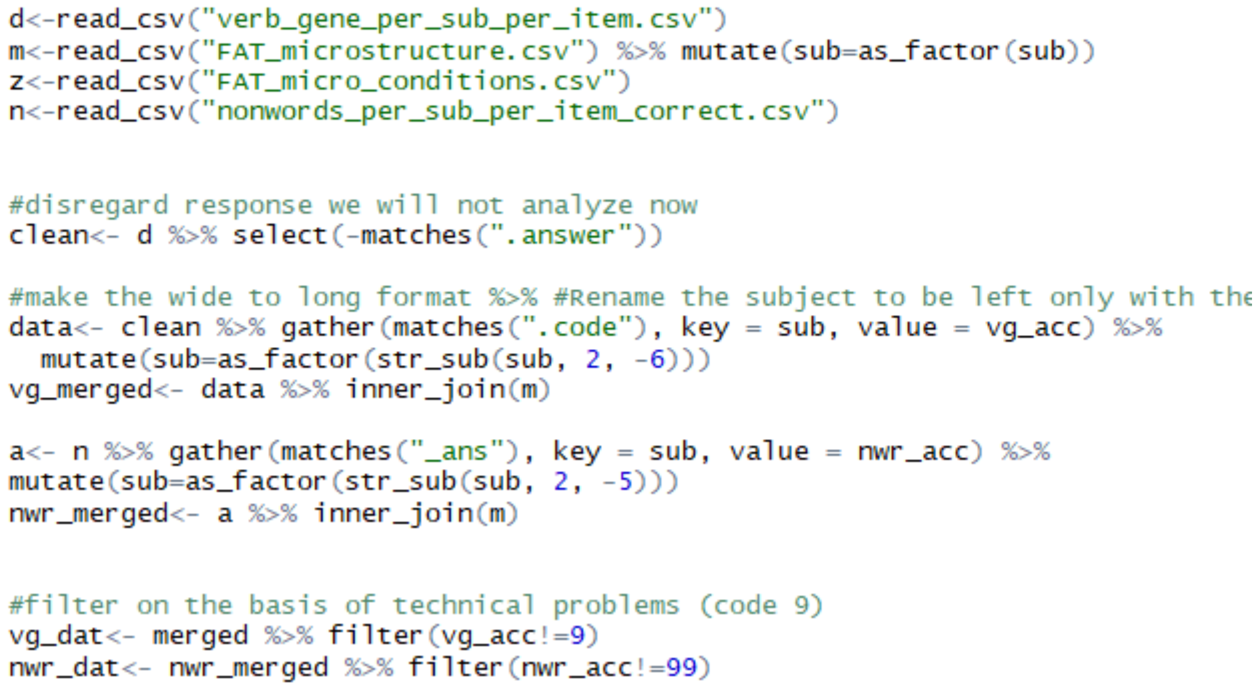


```
#import coding results
allfilenames = dir(pattern = "SynC_pilot*")
accdata = data.frame()
for (isubj in 1:length(allfilenames)) {

  #open file
  filename = allfilenames[isubj]
  temp = read.table(allfilenames[isubj], header = T, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[2]
  temp$participant = as.factor(isubj)

  #combine with other files
  accdata = rbind(accdata,temp)
}

#import onset-offset results
praatfilenames = dir(pattern = "*onsets_durationAutomatic.txt")
praatdata = data.frame()
for (isubj in 1:length(praatfilenames)) {

  #open file
  filename = praatfilenames[isubj]
  temp = read.table(praatfilenames[isubj], header = F, sep = '\t')
```

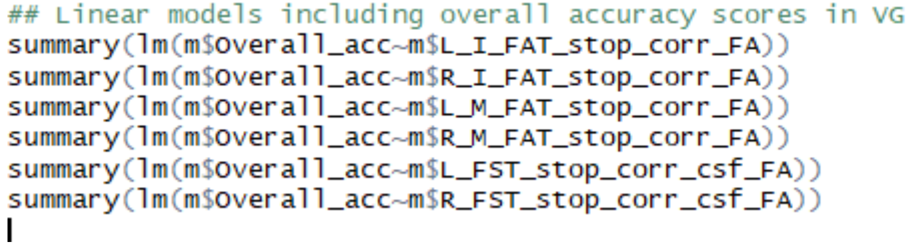
Naming Conventions

Things to avoid:

- Inconsistency in naming
- Names that do not reveal exact purpose of variable/function/file
- Cryptic abbreviations



```
d<-read_csv("verb_gene_per_sub_per_item.csv")
m<-read_csv("FAT_microstructure.csv") %>% mutate(sub=as_factor(sub))
z<-read_csv("FAT_micro_conditions.csv")
n<-read_csv("nonwords_per_sub_per_item_correct.csv")


#disregard response we will not analyze now
clean<- d %>% select(-matches(".answer"))

#make the wide to long format %>% #Rename the subject to be left only with the
data<- clean %>% gather(matches(".code"), key = sub, value = vg_acc) %>%
  mutate(sub=as_factor(str_sub(sub, 2, -6)))
vg_merged<- data %>% inner_join(m)

a<- n %>% gather(matches("_ans"), key = sub, value = nwr_acc) %>%
mutate(sub=as_factor(str_sub(sub, 2, -5)))
nwr_merged<- a %>% inner_join(m)


#filter on the basis of technical problems (code 9)
vg_dat<- merged %>% filter(vg_acc!=9)
nwr_dat<- nwr_merged %>% filter(nwr_acc!=99)
```

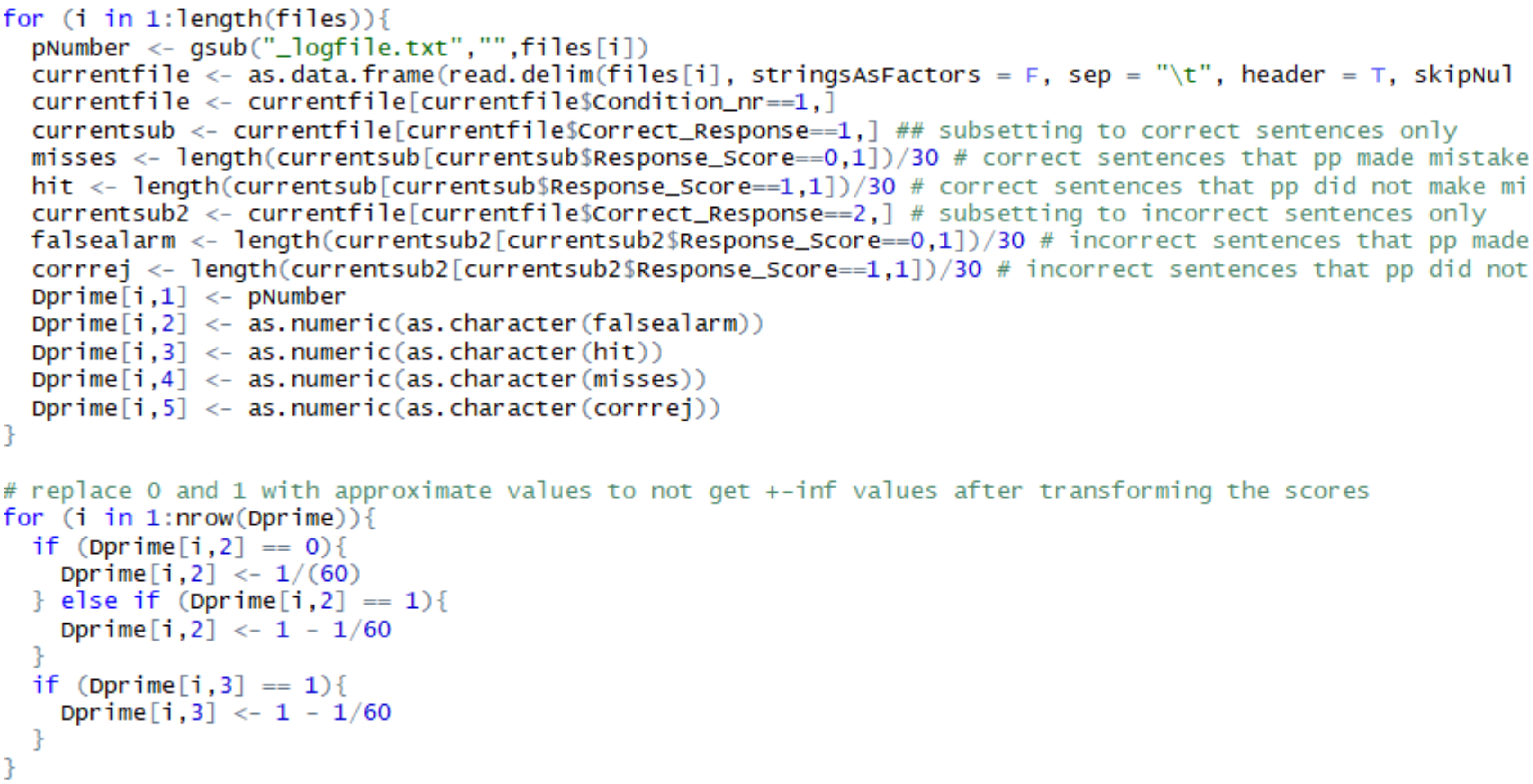


```
## Linear models including overall accuracy scores in VG
summary(lm(m$Overall_acc~m$L_I_FAT_stop_corr_FA))
summary(lm(m$Overall_acc~m$R_I_FAT_stop_corr_FA))
summary(lm(m$Overall_acc~m$L_M_FAT_stop_corr_FA))
summary(lm(m$Overall_acc~m$R_M_FAT_stop_corr_FA))
summary(lm(m$Overall_acc~m$L_FST_stop_corr_csf_FA))
summary(lm(m$Overall_acc~m$R_FST_stop_corr_csf_FA))
```

Naming Conventions

Things to avoid:

- Inconsistency in naming
- Names that do not reveal exact purpose of variable/function/file
- Cryptic abbreviations



```
for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep = "\t", header = T, skipNul
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting to correct sentences only
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct sentences that pp made mistake
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sentences that pp did not make mi
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting to incorrect sentences only
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # incorrect sentences that pp made
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorrect sentences that pp did not
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after transforming the scores
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
  if (Dprime[i,3] == 1){
    Dprime[i,3] <- 1 - 1/60
  }
}
```

Naming Conventions

- ❑Important for readability, reusability, modularity of the code
- ❑The domain of R is very inconsistent when it comes to naming conventions (read: Rasmus Bååth, 2012)
- ❑ Varied styles across companies
- ❑ Internal packages have different styles from each other
- ❑ Google's advice radically differs from R internal packages

Naming Conventions

- Case based
  - alllowercase
  - lowerCamelCase
  - UpperCamelCase
- Separator based
  - period.separated
  - underscore_separated

- Variables are nouns. e.g.:
  - *subjectNumber*
  - *meanRTPlot*
  - *allDataFilePath*
- Functions are verbs. e.g.:
  - *retrieveData* (& not *dataRetriever*)
  - *calcFourierTransform*

- Names should be self-explanatory
- Balance between explaining and being concise
  - Modularity helps with that
- Possible to choose a different style according to purpose
  - Variables - loweCamelCase
  - Functions – period.separated
  - File Names – underscore_separated

Good variable and function names reduce need for comments and improve readability and comprehension of the code

Naming Conventions

- Make your choice according to your purpose
  - Are you coding individually?
  - Or in a team?
  - Are you coding for an individual project
  - Or are you developing an R extension or package?

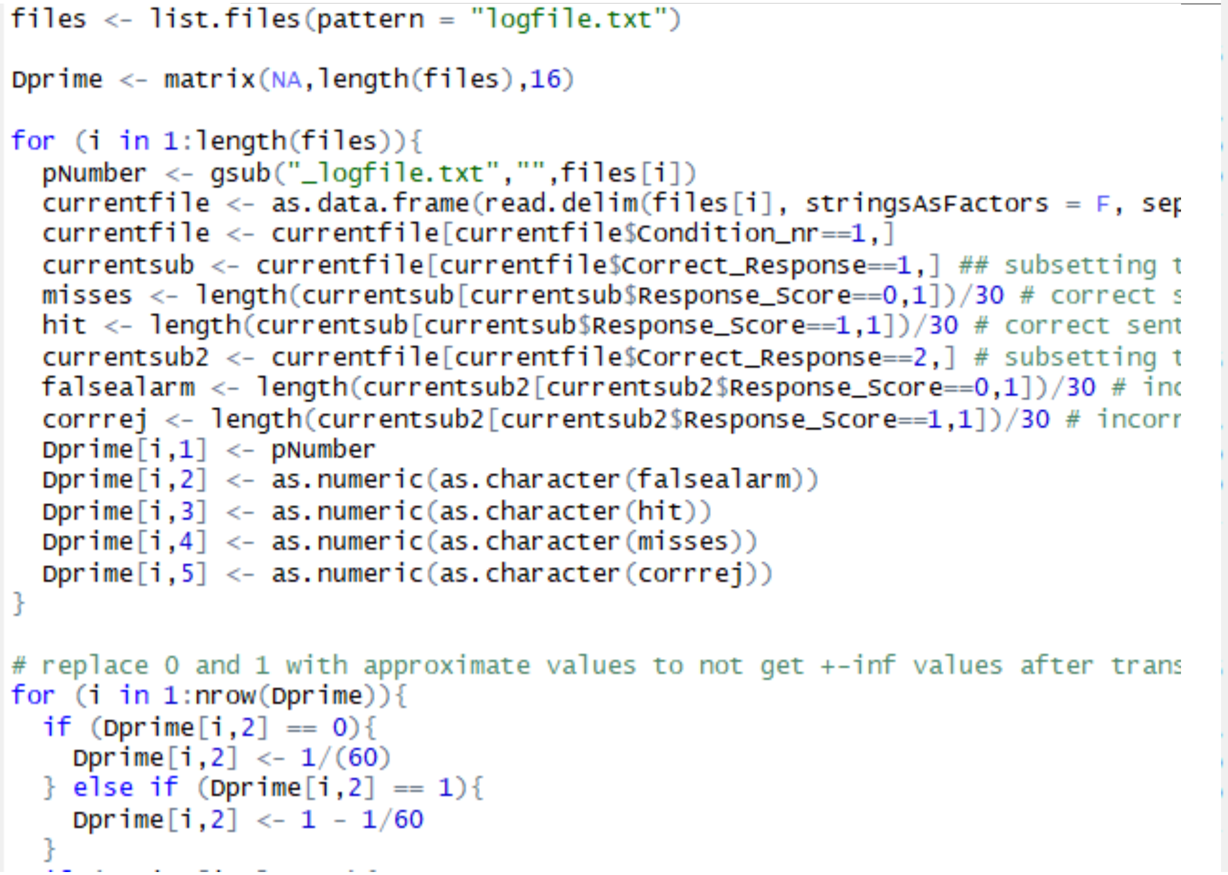
Naming Conventions

- ❑ Coding individually for an individual project
  - ❑ Consistency
  - ❑ Choose freely but maintain choice across projects
- ❑ Coding individually for a package or extension
  - ❑ Choice should be driven by conventions in existing packages
  - ❑ Rasmus Baath 2012 is a good source for this; You can also do your own analysis like he does
- ❑Coding in a team
  - ❑ Choice driven by what everyone is comfortable with and can maintain

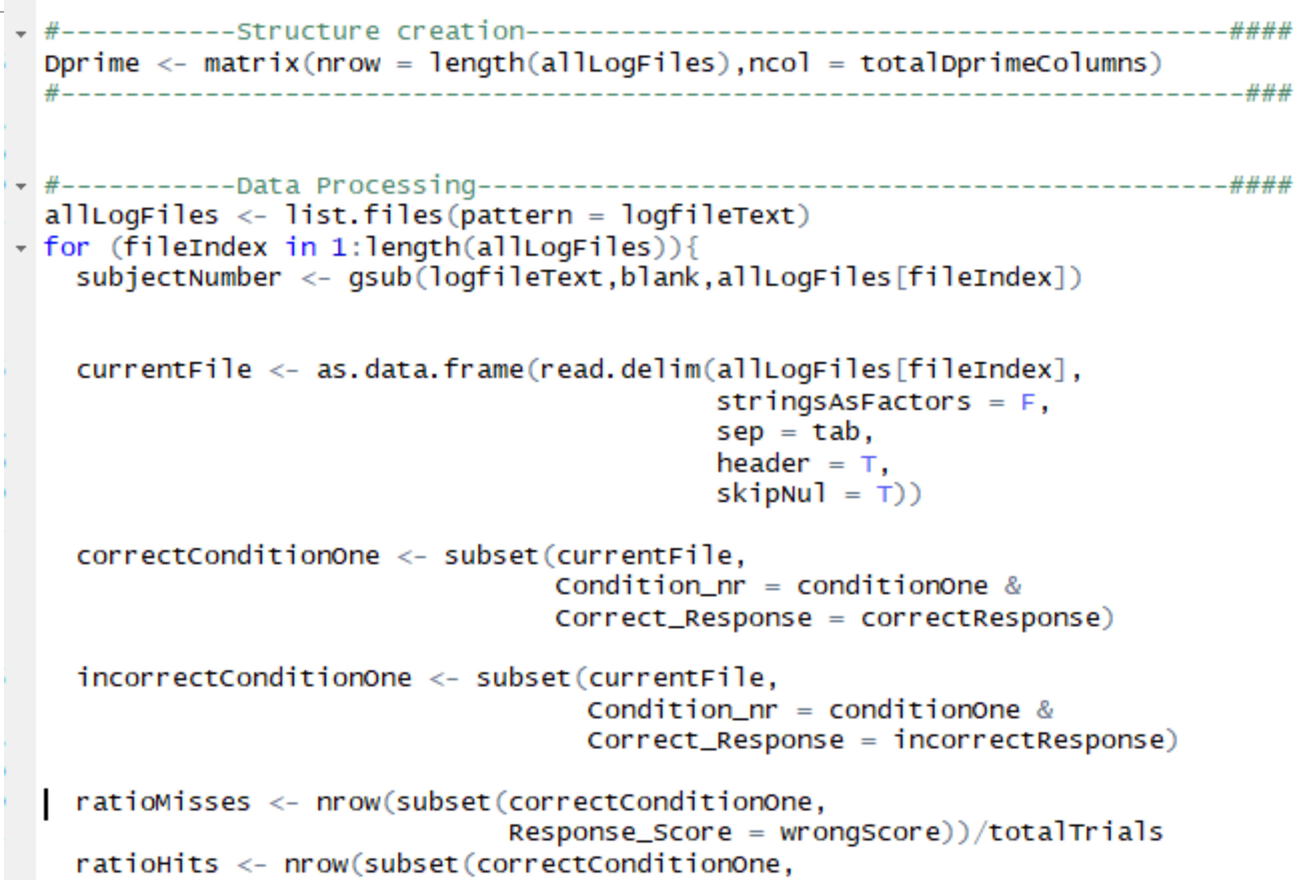
Naming Conventions

- ❑ For other languages
  - ❑ R is quite a recent language and data types of variables are transformable.
  - ❑ For some other languages/softwares e.g. Neurobs Presentation, Java, C++
    - ❑ data types are fixed
    - ❑ Need to be declared before being called
    - ❑ Difference between variable and constants
    - ❑ In such cases it is a good idea to denote the datatype in the variable name. e.g.:
      - ❑ ivSubjectNumber -> i shows its an integer, v shows it's a variable
      - ❑ scTaskOne -> s shows it's a string, c shows it's a constant
    - ❑ Declarations should be made separately (like library and parameter declarations)

Naming Conventions Comparison



```r
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting t
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct s
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sent
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting t
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # inc
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorr
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

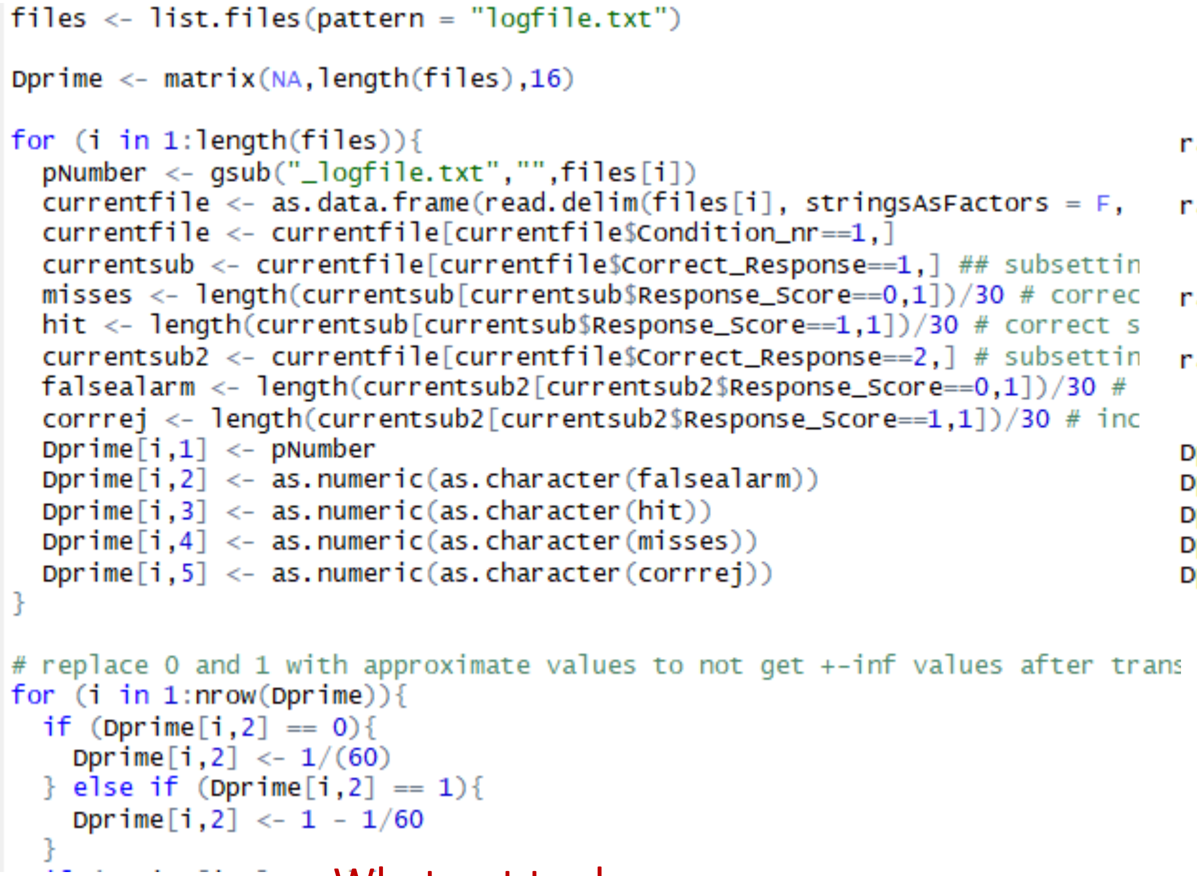


```r
#-----------Structure creation------------------------------------------####
Dprime <- matrix(nrow = length(allLogFiles),ncol = totalDprimeColumns)
#----------------------------------------------------------------------------###


#-----------Data Processing----------------------------------------------####
allLogFiles <- list.files(pattern = logfileText)
for (fileIndex in 1:length(allLogFiles)){
  subjectNumber <- gsub(logfileText,blank,allLogFiles[fileIndex])


  currentFile <- as.data.frame(read.delim(allLogFiles[fileIndex],
                                          stringsAsFactors = F,
                                          sep = tab,
                                          header = T,
                                          skipNul = T))

  correctConditionOne <- subset(currentFile,
                                Condition_nr = conditionOne &
                                Correct_Response = correctResponse)

  incorrectConditionOne <- subset(currentFile,
                                  Condition_nr = conditionOne &
                                  Correct_Response = incorrectResponse)

  ratioMisses <- nrow(subset(correctConditionOne,
                             Response_Score = wrongScore))/totalTrials
  ratioHits <- nrow(subset(correctConditionOne,
```

What not to do

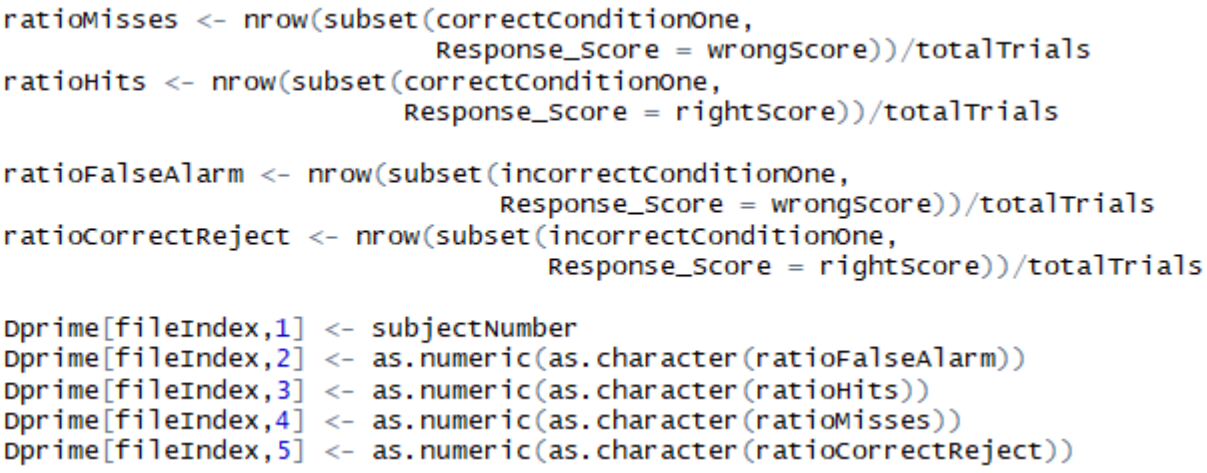
How it can be improved

Naming Conventions Comparison



```
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F,
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsettin
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correc
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct s
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsettin
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 #
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # inc
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

What not to do

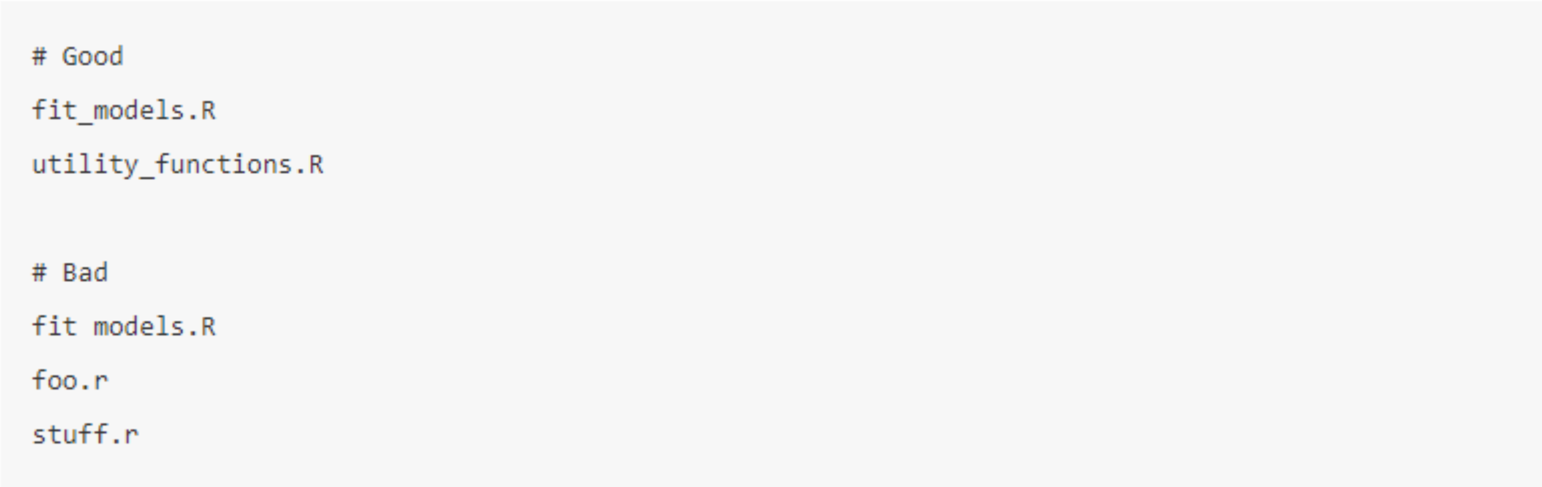


```
ratioMisses <- nrow(subset(correctConditionOne,
                           Response_Score = wrongScore))/totalTrials
ratioHits <- nrow(subset(correctConditionOne,
                         Response_Score = rightScore))/totalTrials

ratioFalseAlarm <- nrow(subset(incorrectConditionOne,
                               Response_Score = wrongScore))/totalTrials
ratioCorrectReject <- nrow(subset(incorrectConditionOne,
                                  Response_Score = rightScore))/totalTrials

Dprime[fileIndex,1] <- subjectNumber
Dprime[fileIndex,2] <- as.numeric(as.character(ratioFalseAlarm))
Dprime[fileIndex,3] <- as.numeric(as.character(ratioHits))
Dprime[fileIndex,4] <- as.numeric(as.character(ratioMisses))
Dprime[fileIndex,5] <- as.numeric(as.character(ratioCorrectReject))
```

How it can be improved

Special Note: File Naming Conventions

- Avoid special characters or spaces in file names
- Stick to letters, numbers and underscore



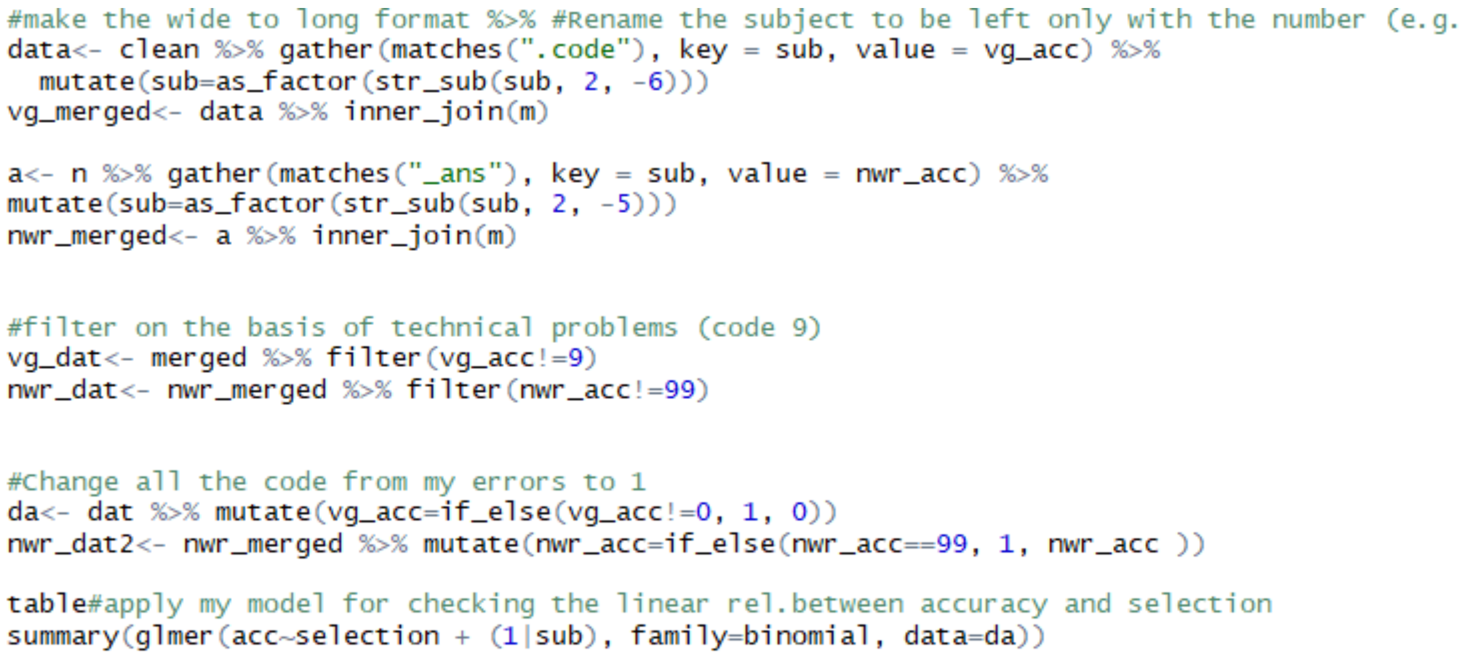
```
# Good
fit_models.R
utility_functions.R

# Bad
fit models.R
foo.r
stuff.r
```

Example source: https://style.tidyverse.org/files.html#names

Avoid hard coding; Use Parameters

What NOT to do

- In RStudio, blue usually reflects hard-coded values

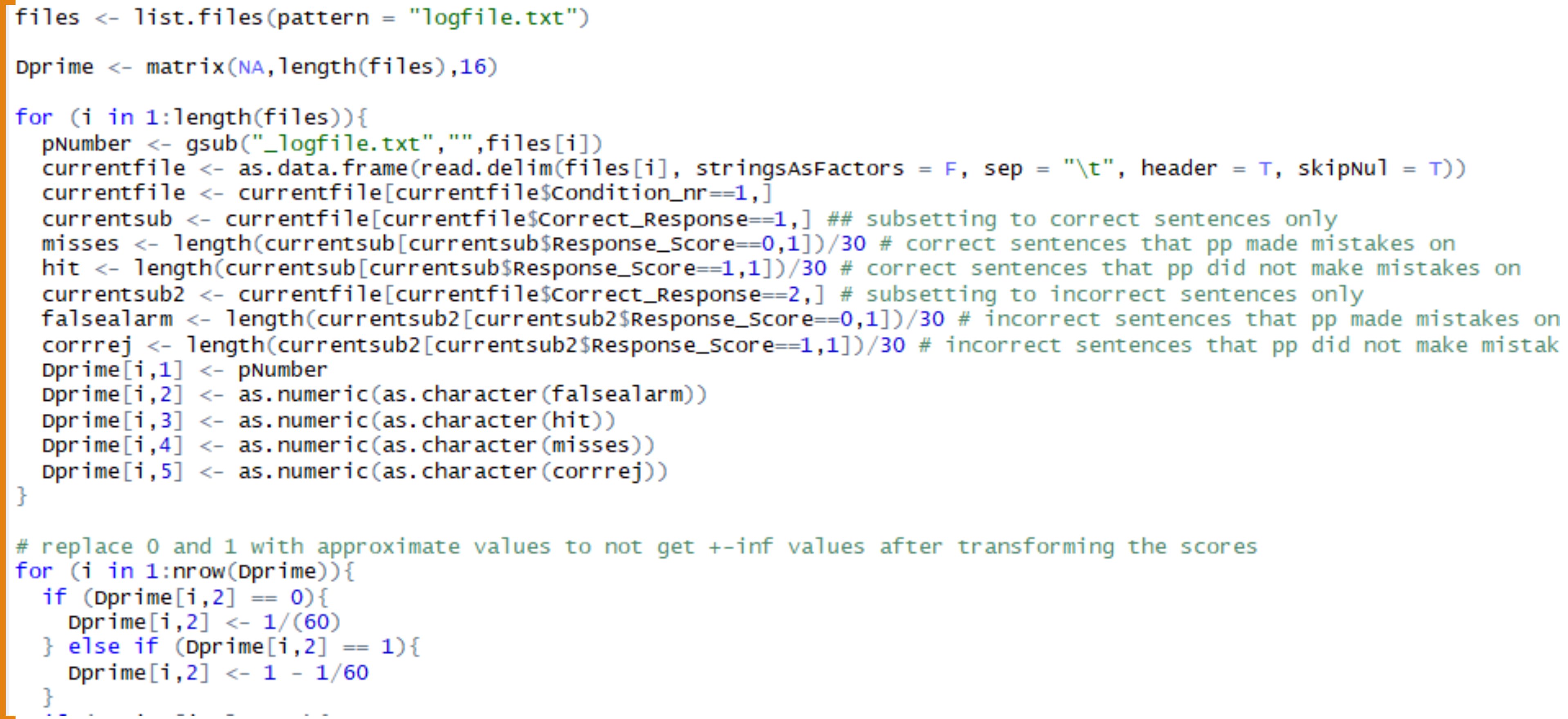


```
#make the wide to long format %>% #Rename the subject to be left only with the number (e.g.
data<- clean %>% gather(matches(".code"), key = sub, value = vg_acc) %>%
  mutate(sub=as_factor(str_sub(sub, 2, -6)))
vg_merged<- data %>% inner_join(m)

a<- n %>% gather(matches("_ans"), key = sub, value = nwr_acc) %>%
mutate(sub=as_factor(str_sub(sub, 2, -5)))
nwr_merged<- a %>% inner_join(m)


#filter on the basis of technical problems (code 9)
vg_dat<- merged %>% filter(vg_acc!=9)
nwr_dat<- nwr_merged %>% filter(nwr_acc!=99)


#Change all the code from my errors to 1
da<- dat %>% mutate(vg_acc=if_else(vg_acc!=0, 1, 0))
nwr_dat2<- nwr_merged %>% mutate(nwr_acc=if_else(nwr_acc==99, 1, nwr_acc ))

table#apply my model for checking the linear rel.between accuracy and selection
summary(glmer(acc~selection + (1|sub), family=binomial, data=da))
```

Avoid hard coding; Use Parameters

What NOT to do

- In RStudio, blue usually reflects hard-coded values



```r
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep = "\t", header = T, skipNul = T))
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting to correct sentences only
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct sentences that pp made mistakes on
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sentences that pp did not make mistakes on
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting to incorrect sentences only
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # incorrect sentences that pp made mistakes on
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorrect sentences that pp did not make mistak
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after transforming the scores
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

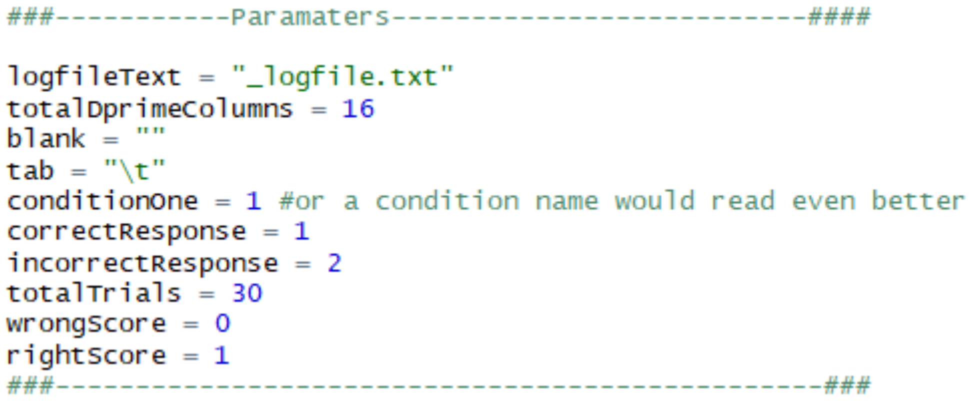
Avoid hard coding; Use Parameters

How it can be better:

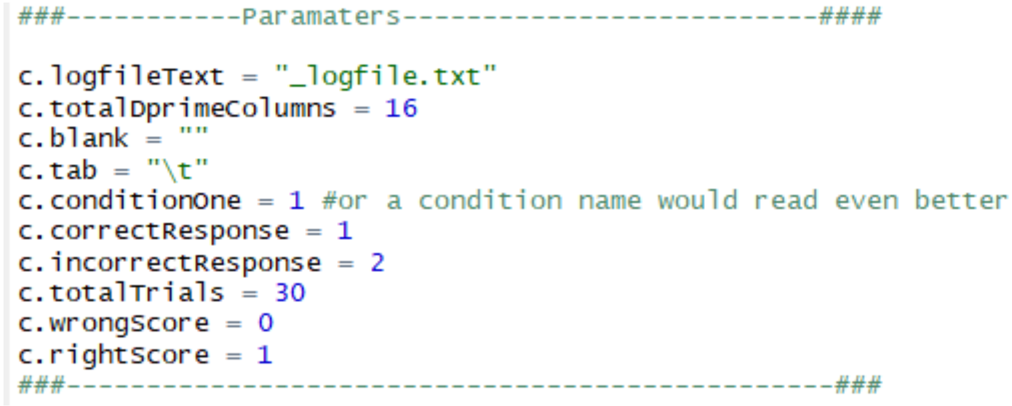
❑ Define parameters on top (Note that these can use constant identifier to remind yourself to not change them)

Aside: Note use of both period.separated and loweCamelCase convention -> this is a special scenario where I merged them for clarity.

(Only good if followed consistently!)



```
###-----------Paramaters-------------------------####

logfileText = "_logfile.txt"
totalDprimeColumns = 16
blank = ""
tab = "\t"
conditionOne = 1 #or a condition name would read even better
correctResponse = 1
incorrectResponse = 2
totalTrials = 30
wrongScore = 0
rightScore = 1
###--------------------------------------------###
```



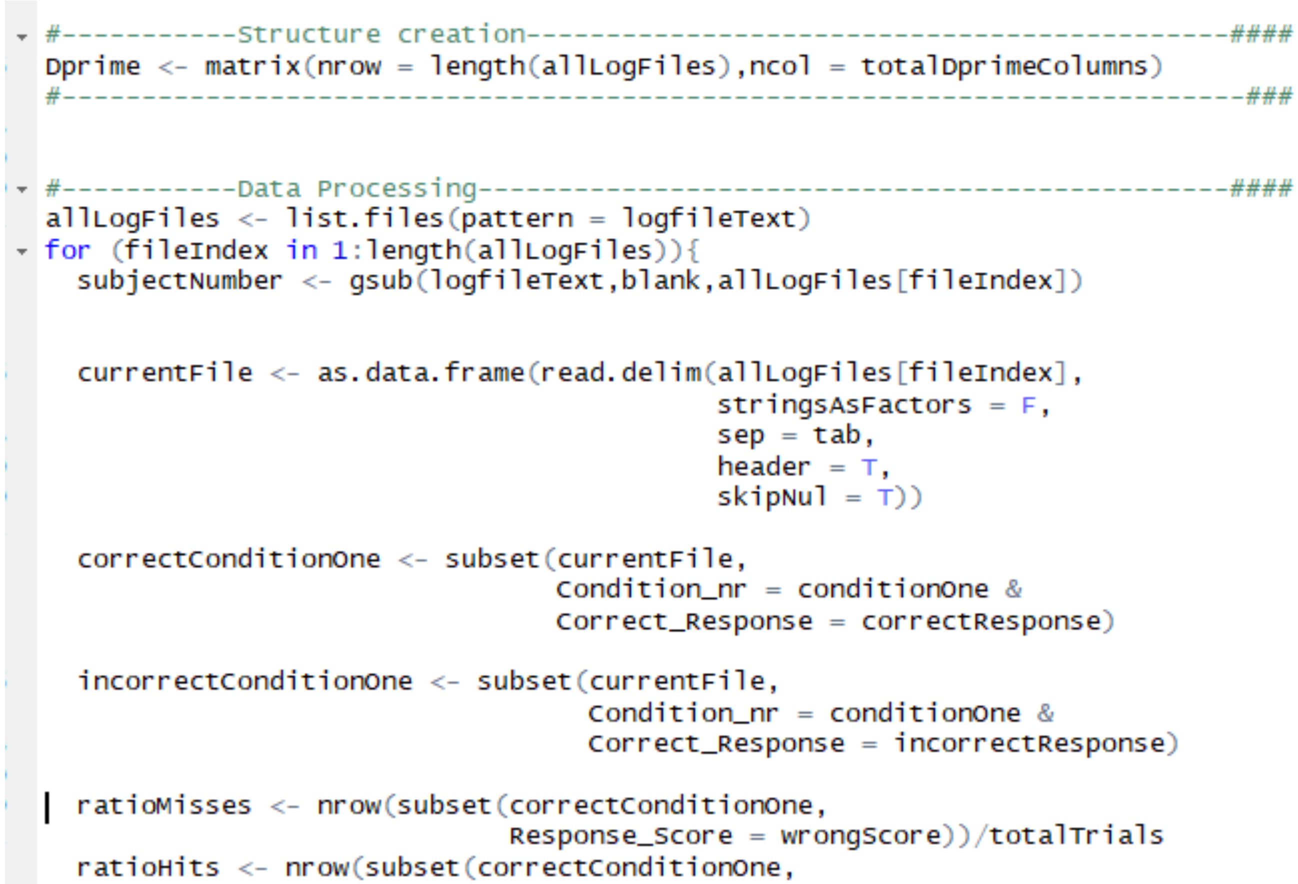
```
###-----------Paramaters-------------------------####

c.logfileText = "_logfile.txt"
c.totalDprimeColumns = 16
c.blank = ""
c.tab = "\t"
c.conditionOne = 1 #or a condition name would read even better
c.correctResponse = 1
c.incorrectResponse = 2
c.totalTrials = 30
c.wrongScore = 0
c.rightScore = 1
###--------------------------------------------###
```

“c.” prefix is meant to reflect to the coder that they are parameters or constants that shouldn’t be changed within the rest of the code

Avoid hard coding; Use Parameters

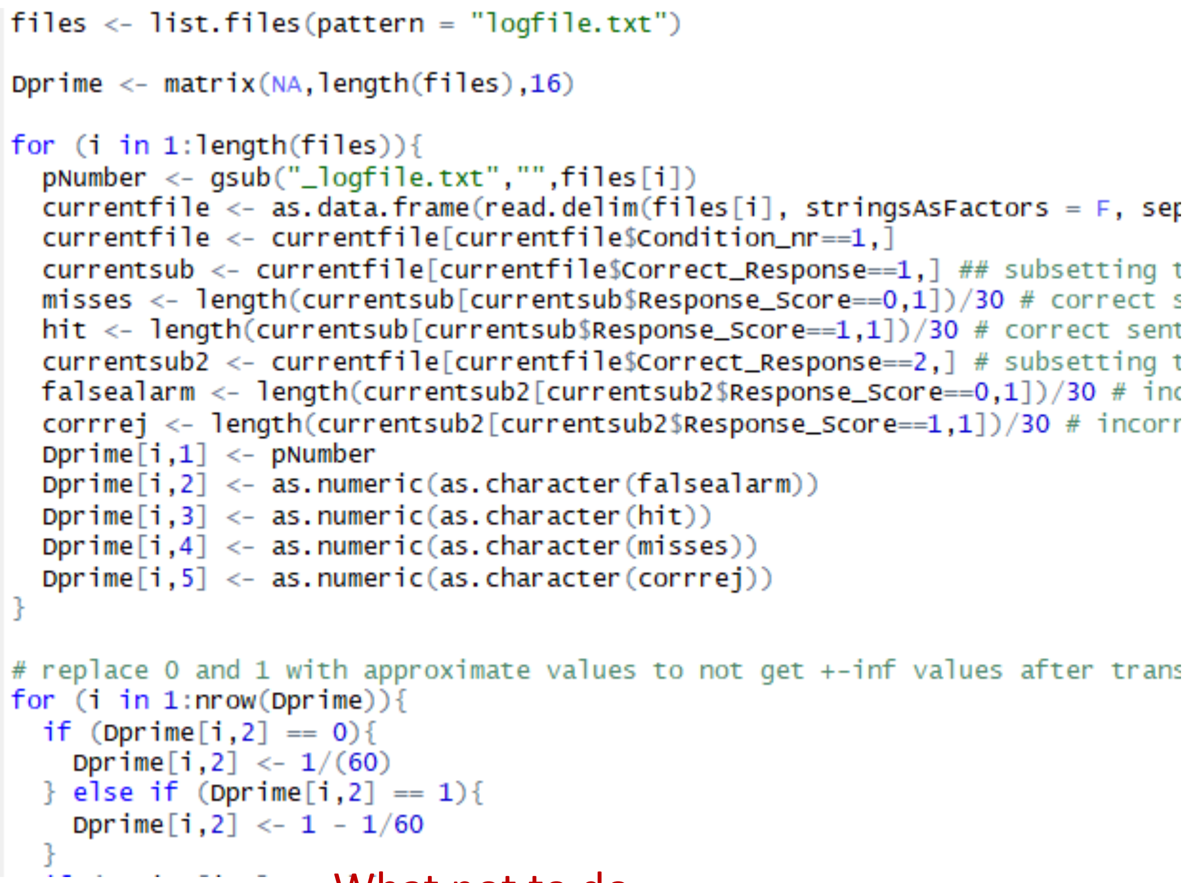
How it can be better:

❑ Use defined parameters in code



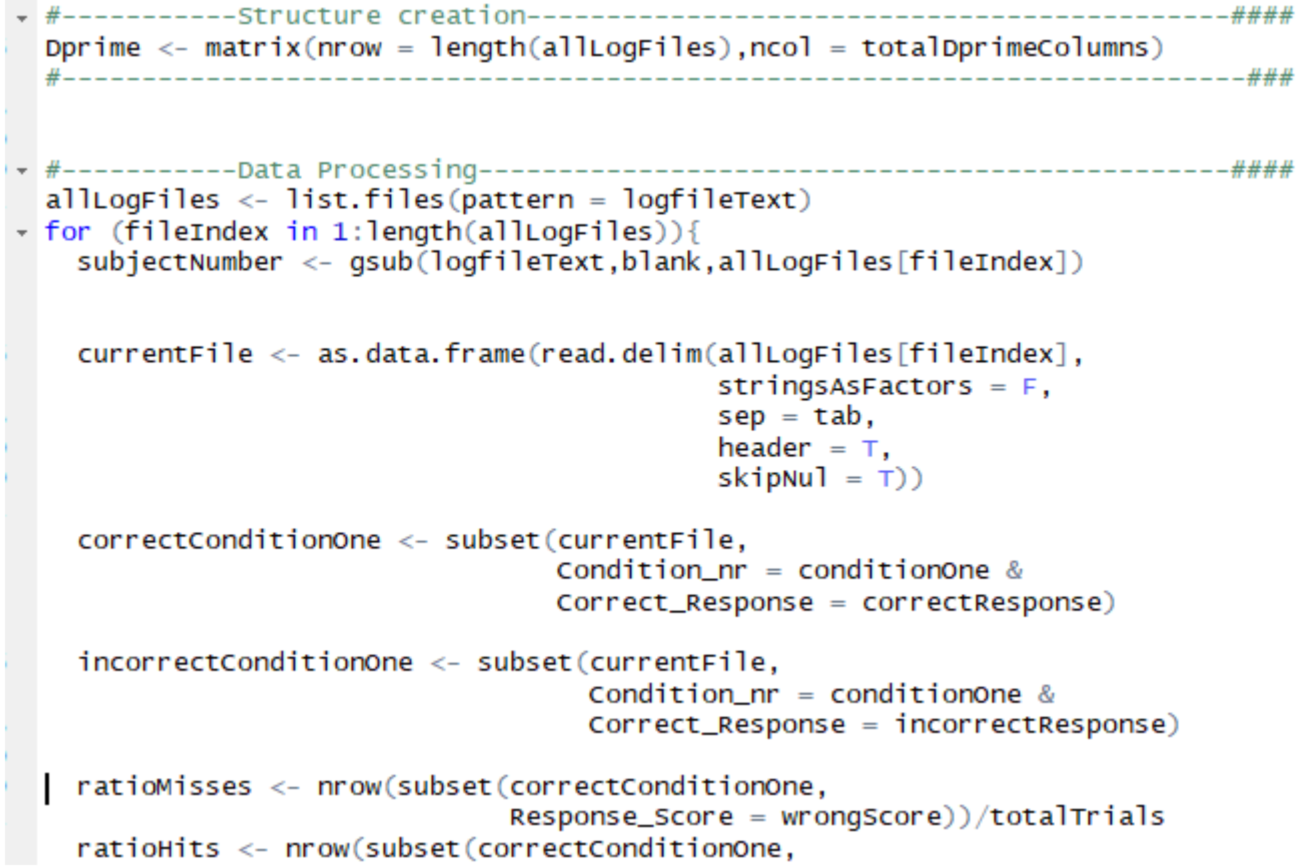
```
#-----------Structure creation-------------------------------------------####
Dprime <- matrix(nrow = length(allLogFiles),ncol = totalDprimeColumns)
#-------------------------------------------------------------------------###


#-----------Data Processing----------------------------------------------####
allLogFiles <- list.files(pattern = logfileText)
for (fileIndex in 1:length(allLogFiles)){
  subjectNumber <- gsub(logfileText,blank,allLogFiles[fileIndex])


  currentFile <- as.data.frame(read.delim(allLogFiles[fileIndex],
                                          stringsAsFactors = F,
                                          sep = tab,
                                          header = T,
                                          skipNul = T))

  correctConditionOne <- subset(currentFile,
                                Condition_nr = conditionOne &
                                Correct_Response = correctResponse)

  incorrectConditionOne <- subset(currentFile,
                                  Condition_nr = conditionOne &
                                  Correct_Response = incorrectResponse)

  ratioMisses <- nrow(subset(correctConditionOne,
                             Response_Score = wrongScore))/totalTrials
  ratioHits <- nrow(subset(correctConditionOne,
```

Avoid hard coding; Use Parameters



```r
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting t
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct s
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sent
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting t
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # inc
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorr
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

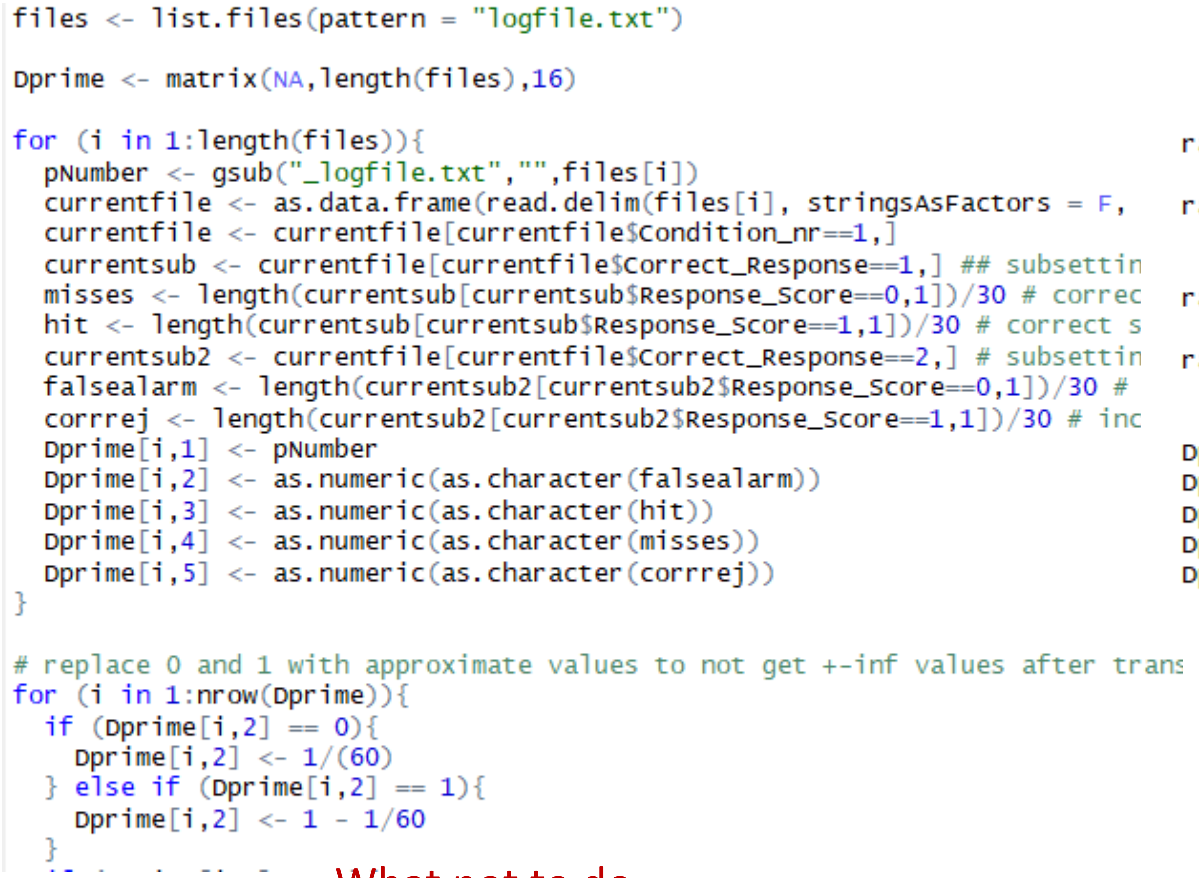
What not to do



```r
#-----------Structure creation-------------------------------------------####
Dprime <- matrix(nrow = length(allLogFiles),ncol = totalDprimeColumns)
#-----------------------------------------------------------------------###


#-----------Data Processing---------------------------------------------####
allLogFiles <- list.files(pattern = logfileText)
for (fileIndex in 1:length(allLogFiles)){
  subjectNumber <- gsub(logfileText,blank,allLogFiles[fileIndex])


  currentFile <- as.data.frame(read.delim(allLogFiles[fileIndex],
                                          stringsAsFactors = F,
                                          sep = tab,
                                          header = T,
                                          skipNul = T))

  correctConditionOne <- subset(currentFile,
                                Condition_nr = conditionOne &
                                Correct_Response = correctResponse)

  incorrectConditionOne <- subset(currentFile,
                                  Condition_nr = conditionOne &
                                  Correct_Response = incorrectResponse)

  ratioMisses <- nrow(subset(correctConditionOne,
                             Response_Score = wrongScore))/totalTrials
  ratioHits <- nrow(subset(correctConditionOne,
```

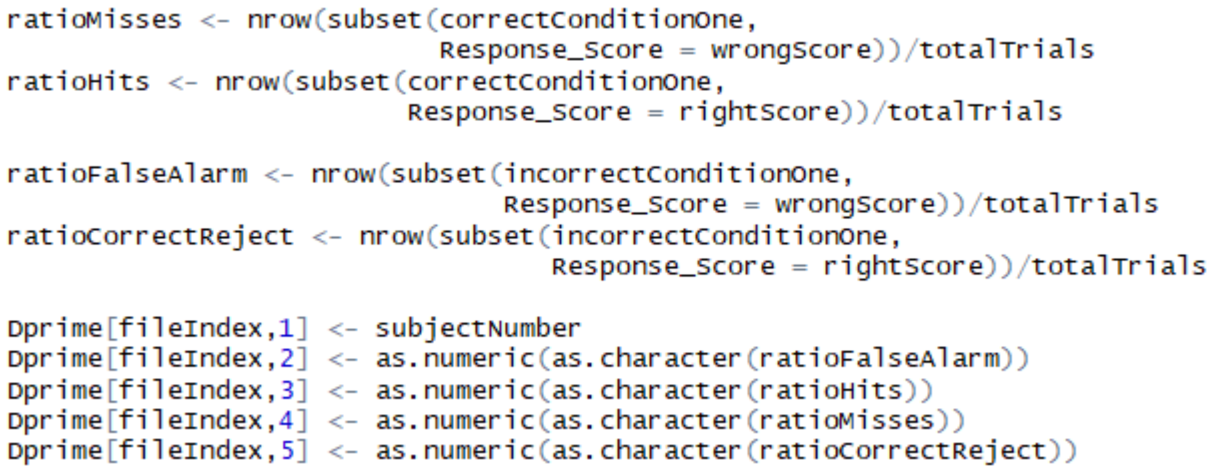
How it can be improved

Avoid hard coding; Use Parameters



```
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F,
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsettin
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correc
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct s
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsettin
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 #
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # inc
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

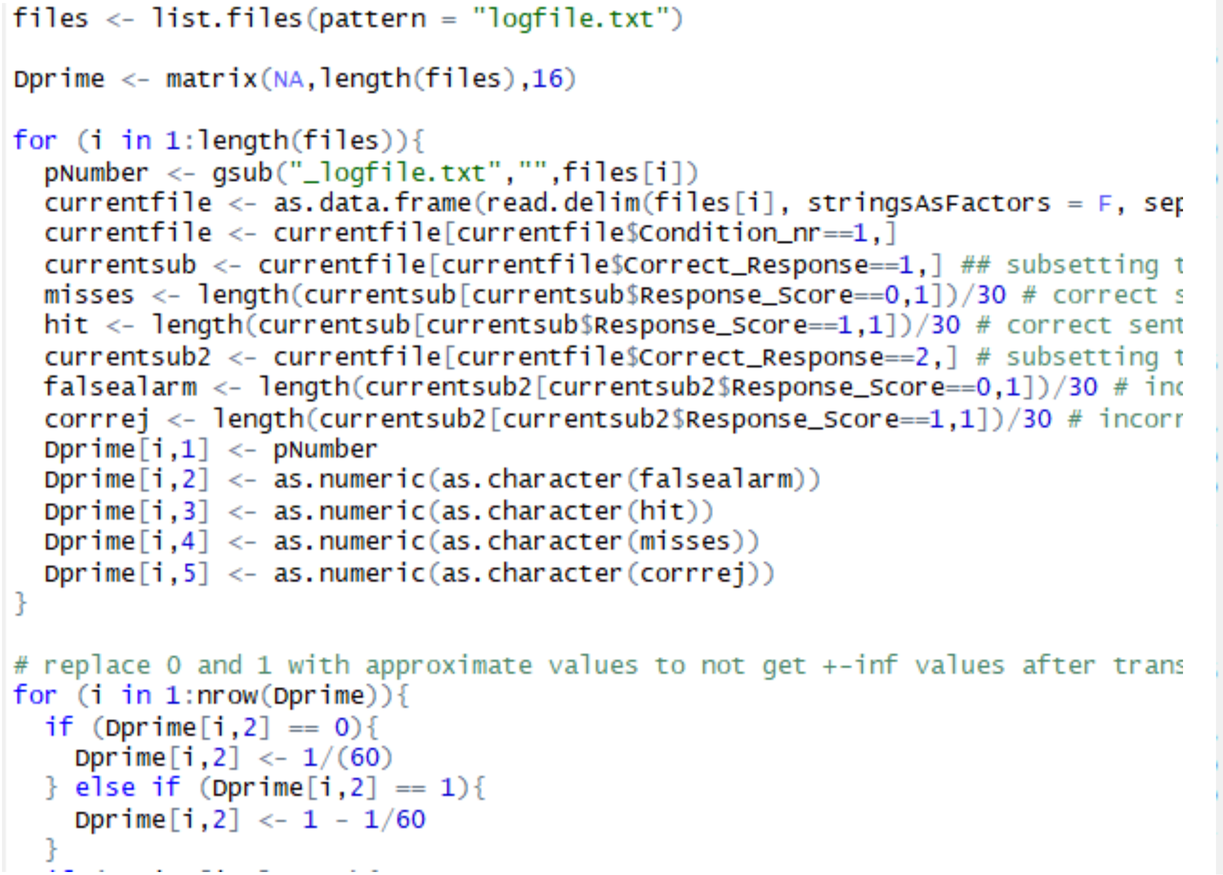
What not to do



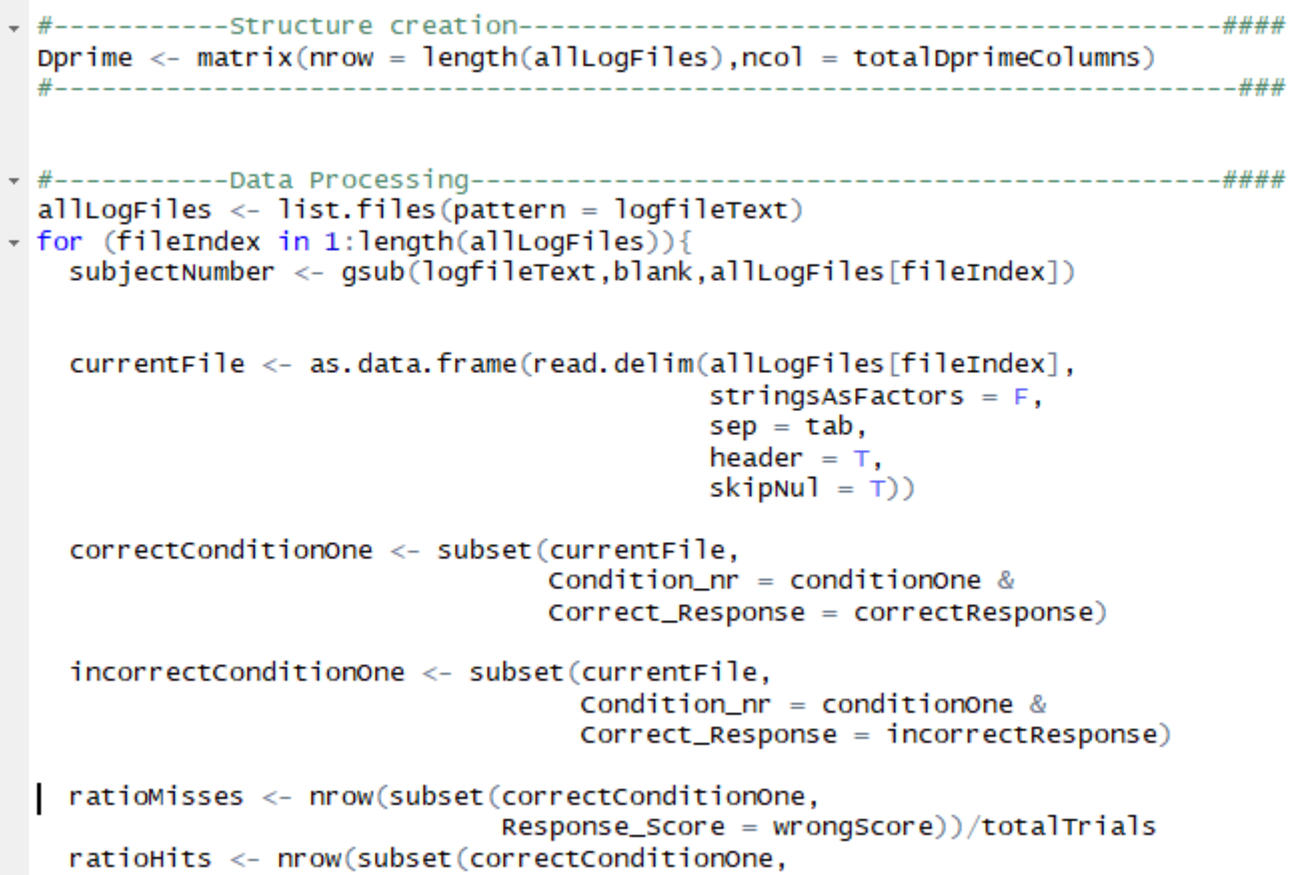
```
ratioMisses <- nrow(subset(correctConditionOne,
                           Response_Score = wrongScore))/totalTrials
ratioHits <- nrow(subset(correctConditionOne,
                         Response_Score = rightScore))/totalTrials

ratioFalseAlarm <- nrow(subset(incorrectConditionOne,
                               Response_Score = wrongScore))/totalTrials
ratioCorrectReject <- nrow(subset(incorrectConditionOne,
                                  Response_Score = rightScore))/totalTrials

Dprime[fileIndex,1] <- subjectNumber
Dprime[fileIndex,2] <- as.numeric(as.character(ratioFalseAlarm))
Dprime[fileIndex,3] <- as.numeric(as.character(ratioHits))
Dprime[fileIndex,4] <- as.numeric(as.character(ratioMisses))
Dprime[fileIndex,5] <- as.numeric(as.character(ratioCorrectReject))
```

How it can be improved

Reduce visual chaos:Break up code using separators and code folding

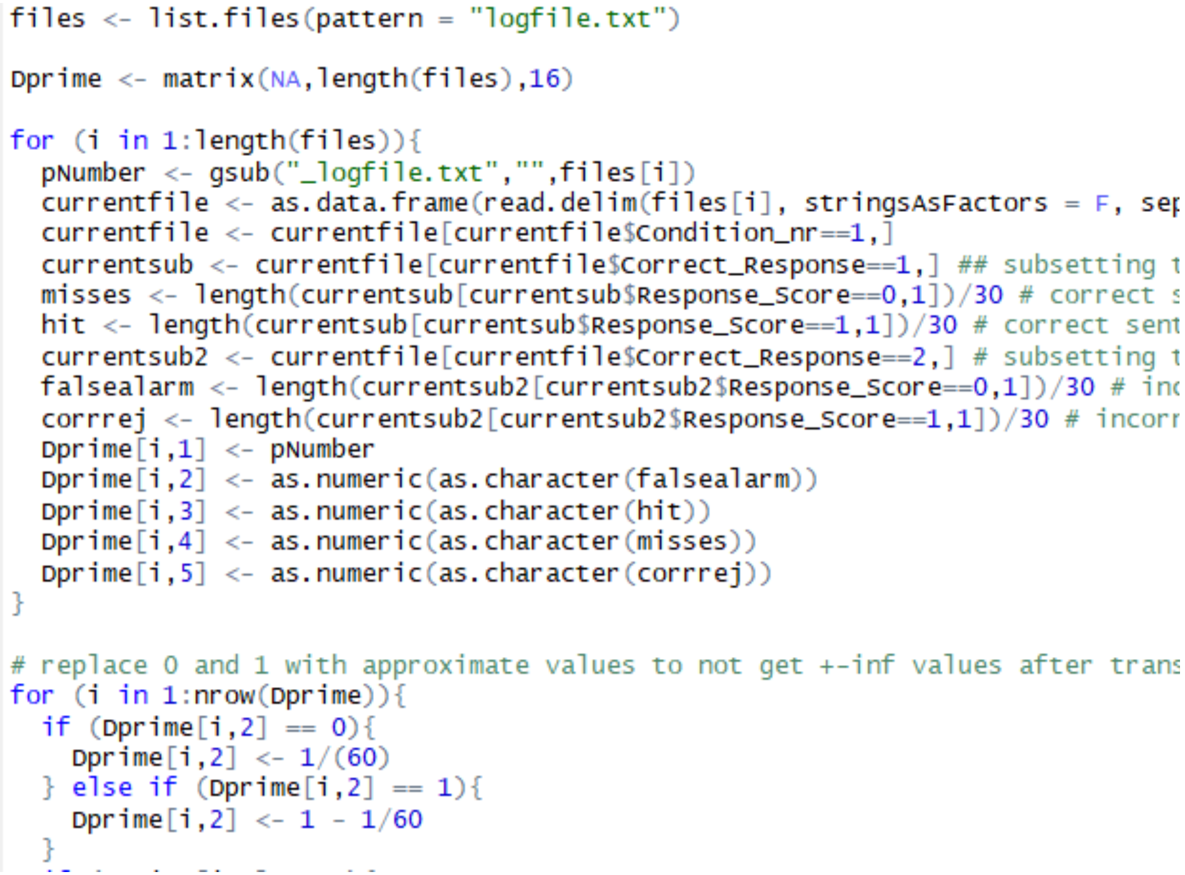


```
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting t
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct s
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sent
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting t
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # inc
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorr
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

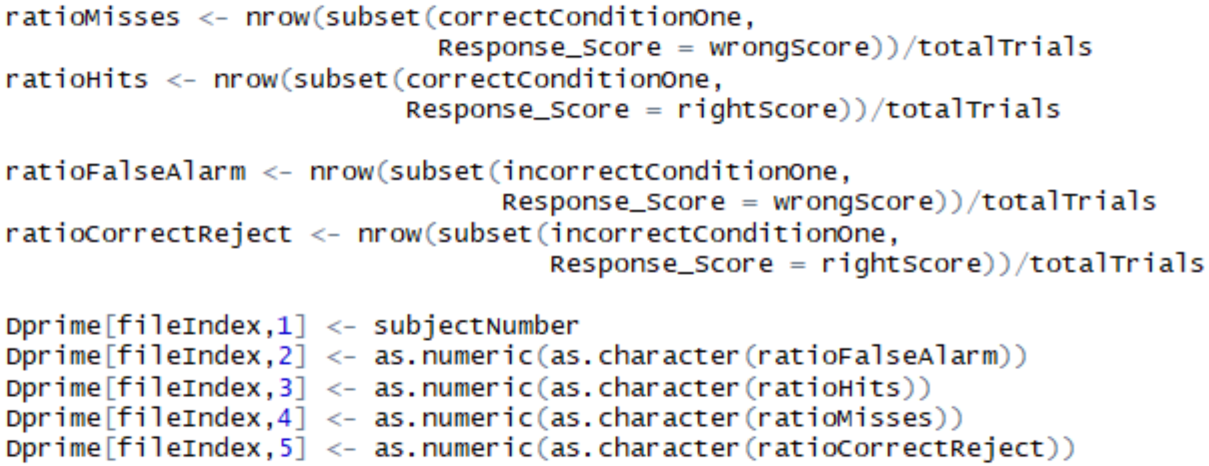


```
#-----------Structure creation--------------------------------------------####
Dprime <- matrix(nrow = length(allLogFiles),ncol = totalDprimeColumns)
#--------------------------------------------------------------------------###


#-----------Data Processing-----------------------------------------------####
allLogFiles <- list.files(pattern = logfileText)
for (fileIndex in 1:length(allLogFiles)){
  subjectNumber <- gsub(logfileText,blank,allLogFiles[fileIndex])


  currentFile <- as.data.frame(read.delim(allLogFiles[fileIndex],
                                          stringsAsFactors = F,
                                          sep = tab,
                                          header = T,
                                          skipNul = T))

  correctConditionOne <- subset(currentFile,
                                Condition_nr = conditionOne &
                                Correct_Response = correctResponse)

  incorrectConditionOne <- subset(currentFile,
                                  Condition_nr = conditionOne &
                                  Correct_Response = incorrectResponse)

  ratioMisses <- nrow(subset(correctConditionOne,
                             Response_Score = wrongScore))/totalTrials
  ratioHits <- nrow(subset(correctConditionOne,
```

Reduce visual chaos: Break up code using whitespaces and indenting

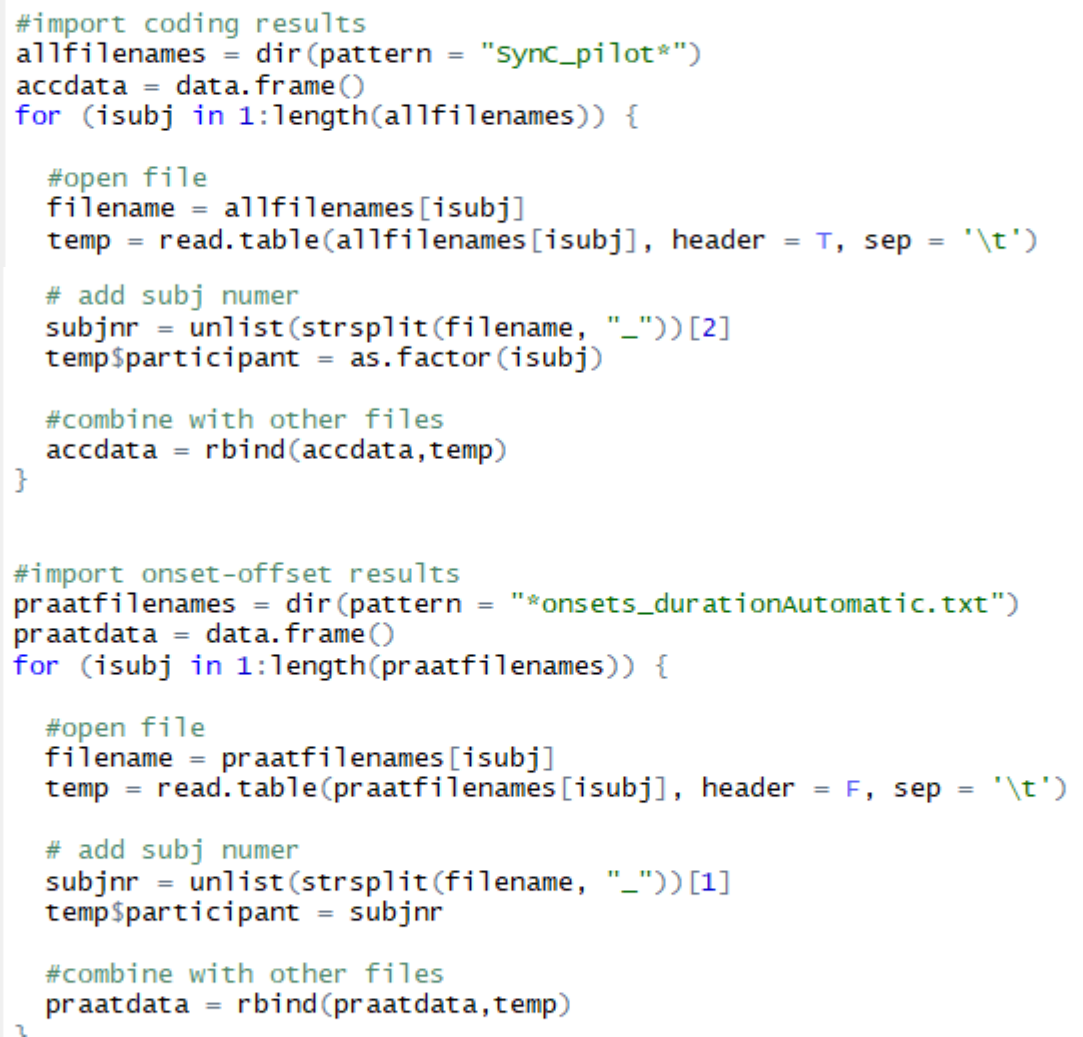


```r
files <- list.files(pattern = "logfile.txt")

Dprime <- matrix(NA,length(files),16)

for (i in 1:length(files)){
  pNumber <- gsub("_logfile.txt","",files[i])
  currentfile <- as.data.frame(read.delim(files[i], stringsAsFactors = F, sep
  currentfile <- currentfile[currentfile$Condition_nr==1,]
  currentsub <- currentfile[currentfile$Correct_Response==1,] ## subsetting t
  misses <- length(currentsub[currentsub$Response_Score==0,1])/30 # correct s
  hit <- length(currentsub[currentsub$Response_Score==1,1])/30 # correct sent
  currentsub2 <- currentfile[currentfile$Correct_Response==2,] # subsetting t
  falsealarm <- length(currentsub2[currentsub2$Response_Score==0,1])/30 # inc
  corrrej <- length(currentsub2[currentsub2$Response_Score==1,1])/30 # incorr
  Dprime[i,1] <- pNumber
  Dprime[i,2] <- as.numeric(as.character(falsealarm))
  Dprime[i,3] <- as.numeric(as.character(hit))
  Dprime[i,4] <- as.numeric(as.character(misses))
  Dprime[i,5] <- as.numeric(as.character(corrrej))
}

# replace 0 and 1 with approximate values to not get +-inf values after trans
for (i in 1:nrow(Dprime)){
  if (Dprime[i,2] == 0){
    Dprime[i,2] <- 1/(60)
  } else if (Dprime[i,2] == 1){
    Dprime[i,2] <- 1 - 1/60
  }
```

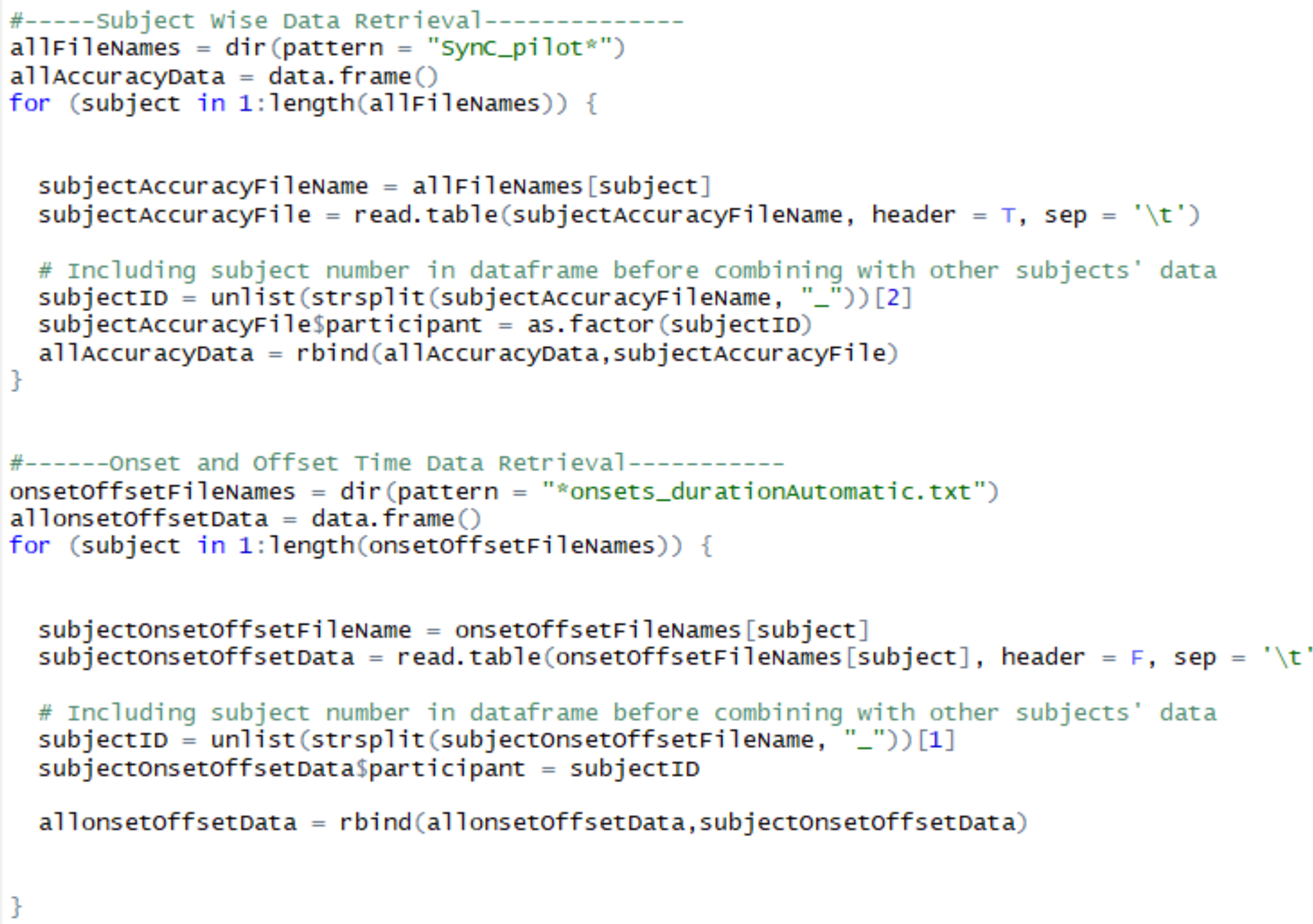


```r
ratioMisses <- nrow(subset(correctConditionOne,
                           Response_Score = wrongScore))/totalTrials
ratioHits <- nrow(subset(correctConditionOne,
                         Response_Score = rightScore))/totalTrials

ratioFalseAlarm <- nrow(subset(incorrectConditionOne,
                               Response_Score = wrongScore))/totalTrials
ratioCorrectReject <- nrow(subset(incorrectConditionOne,
                                  Response_Score = rightScore))/totalTrials

Dprime[fileIndex,1] <- subjectNumber
Dprime[fileIndex,2] <- as.numeric(as.character(ratioFalseAlarm))
Dprime[fileIndex,3] <- as.numeric(as.character(ratioHits))
Dprime[fileIndex,4] <- as.numeric(as.character(ratioMisses))
Dprime[fileIndex,5] <- as.numeric(as.character(ratioCorrectReject))
```

Reduce visual chaos: Reduce commenting & group lines aligned to same purpose



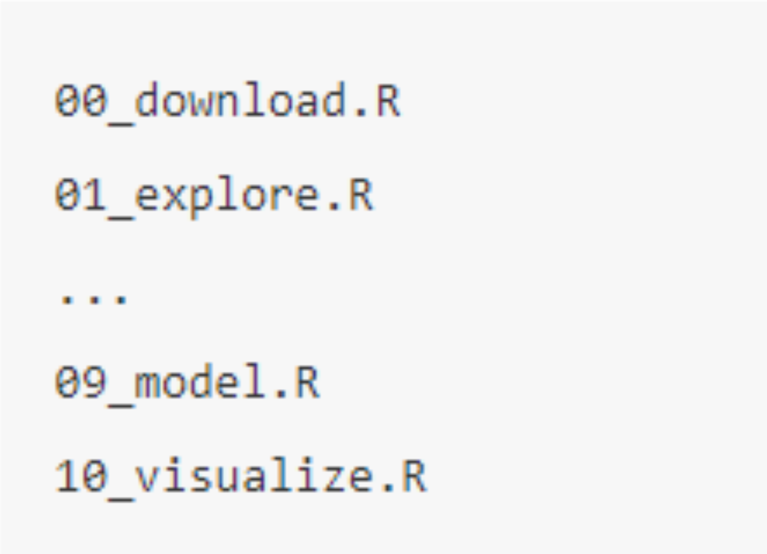
```
#import coding results
allfilenames = dir(pattern = "SynC_pilot*")
accdata = data.frame()
for (isubj in 1:length(allfilenames)) {

  #open file
  filename = allfilenames[isubj]
  temp = read.table(allfilenames[isubj], header = T, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[2]
  temp$participant = as.factor(isubj)

  #combine with other files
  accdata = rbind(accdata,temp)
}


#import onset-offset results
praatfilenames = dir(pattern = "*onsets_durationAutomatic.txt")
praatdata = data.frame()
for (isubj in 1:length(praatfilenames)) {

  #open file
  filename = praatfilenames[isubj]
  temp = read.table(praatfilenames[isubj], header = F, sep = '\t')

  # add subj numer
  subjnr = unlist(strsplit(filename, "_"))[1]
  temp$participant = subjnr

  #combine with other files
  praatdata = rbind(praatdata,temp)
}
```



```
#-----Subject Wise Data Retrieval--------------
allFileNames = dir(pattern = "SynC_pilot*")
allAccuracyData = data.frame()
for (subject in 1:length(allFileNames)) {


  subjectAccuracyFileName = allFileNames[subject]
  subjectAccuracyFile = read.table(subjectAccuracyFileName, header = T, sep = '\t')

  # Including subject number in dataframe before combining with other subjects' data
  subjectID = unlist(strsplit(subjectAccuracyFileName, "_"))[2]
  subjectAccuracyFile$participant = as.factor(subjectID)
  allAccuracyData = rbind(allAccuracyData,subjectAccuracyFile)
}


#------Onset and Offset Time Data Retrieval-----------
onsetOffsetFileNames = dir(pattern = "*onsets_durationAutomatic.txt")
allOnsetOffsetData = data.frame()
for (subject in 1:length(onsetOffsetFileNames)) {


  subjectOnsetOffsetFileName = onsetOffsetFileNames[subject]
  subjectOnsetOffsetData = read.table(onsetOffsetFileNames[subject], header = F, sep = '\t'

  # Including subject number in dataframe before combining with other subjects' data
  subjectID = unlist(strsplit(subjectOnsetOffsetFileName, "_"))[1]
  subjectOnsetOffsetData$participant = subjectID

  allOnsetOffsetData = rbind(allOnsetOffsetData,subjectOnsetOffsetData)


}
```

Modularity & User-Defined Functions

- ❑ Function creation helps with:
  - ❑ Readability
    - ❑ Break code chunks that do different things
    - ❑ Turn into function instead of sections
    - ❑ Informative names make it easy to understand

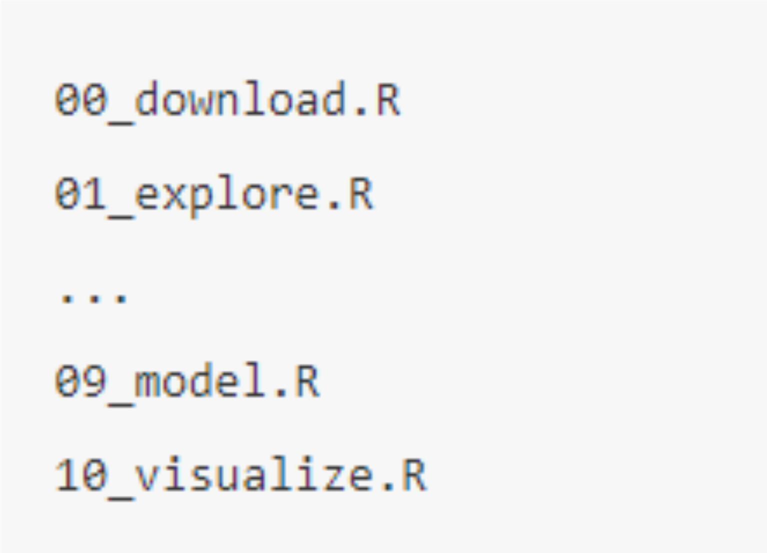


```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

Example source: https://style.tidyverse.org/files.html#names

Modularity & User-Defined Functions

- ❑ Function creation helps with:
  - ❑ Readability
    - ❑ Break code chunks that do different things
    - ❑ Turn into function instead of sections
    - ❑ Informative names make it easy to understand
  - ❑ Simplification

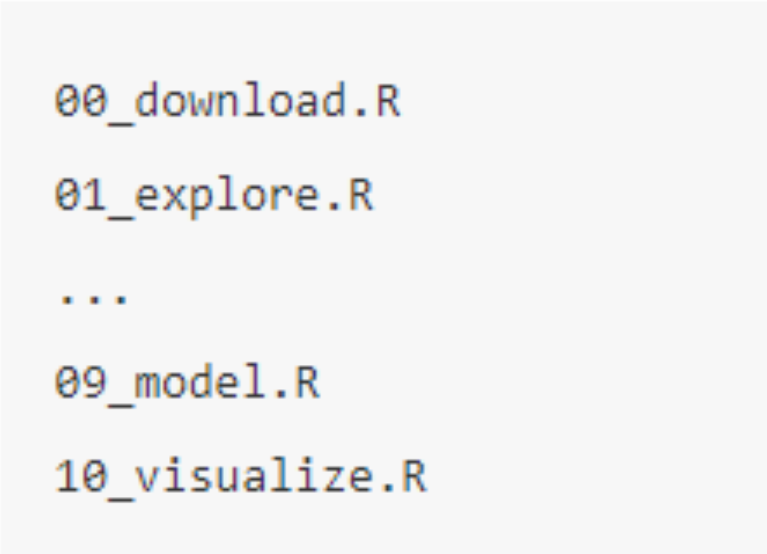


```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

Example source: https://style.tidyverse.org/files.html#names

Modularity & User-Defined Functions

- Function creation helps with:
  - Readability
    - Break code chunks that do different things
    - Turn into function instead of sections
    - Informative names make it easy to understand
  - Simplification
  - Reusability

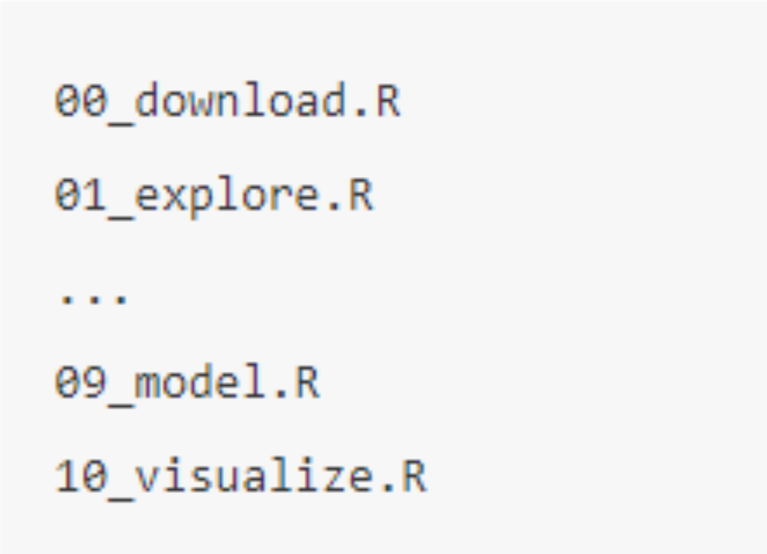


```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

Example source: https://style.tidyverse.org/files.html#names

Modularity & User-Defined Functions

- Function creation helps with:
  - Readability
    - Break code chunks that do different things
    - Turn into function instead of sections
    - Informative names make it easy to understand
  - Simplification
  - Reusability
  - Debugging

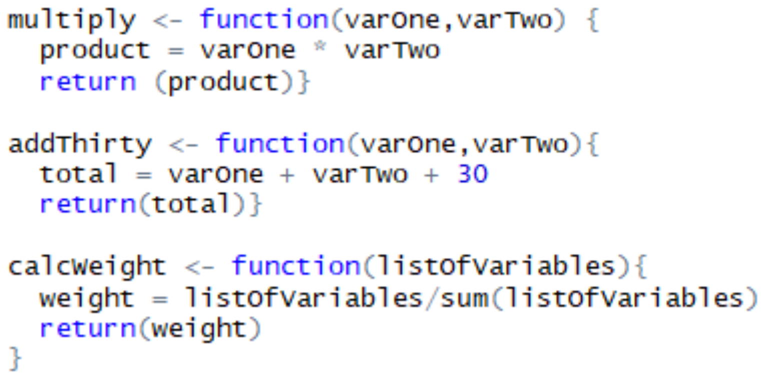


```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

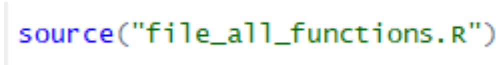
Example source: https://style.tidyverse.org/files.html#names

Modularity & User-Defined Functions

- Another way is to have a file with all user-defined functions and then source that file



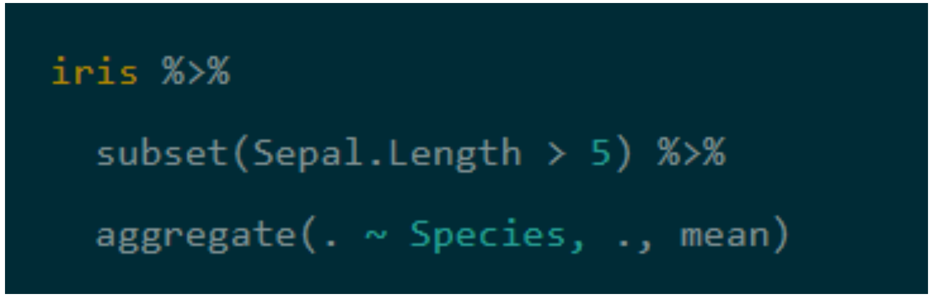
```
multiply <- function(varOne,varTwo) {
  product = varOne * varTwo
  return (product)}

addThirty <- function(varOne,varTwo){
  total = varOne + varTwo + 30
  return(total)}

calcWeight <- function(listOfVariables){
  weight = listOfVariables/sum(listOfVariables)
  return(weight)
}
```



```
source("file_all_functions.R")
```

Bonus mention: Pipe Functions & goodpractice

- magrittr package
- tidyverse or dplyr frequently used



```
iris %>%
  subset(Sepal.Length > 5) %>%
  aggregate(. ~ Species, ., mean)
```

- goodpractice

Example Source:
https://www.datacamp.com/community/tutorials/pipe-r-tutorial

Thank you!

DO PEER REVIEWS!