■Introduction to R

Project Set-Up

Data Management

Coding

Alexander Hurley - November 27, 2015

UNIVERSITY OF BIRMINGHAM

Introduction

Why & How?

Why & How?

provide knowledge and tools for developing an individual project management framework that allows reproducing your research and sharing it with others

Why & How?

Why & How?

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

UNIVERSITY OF BIRMINGHAM

Why & How?

- RStudio
- PackRat
- Git and GitHub
- Online Resources

Project Set-Up

Why invest time?

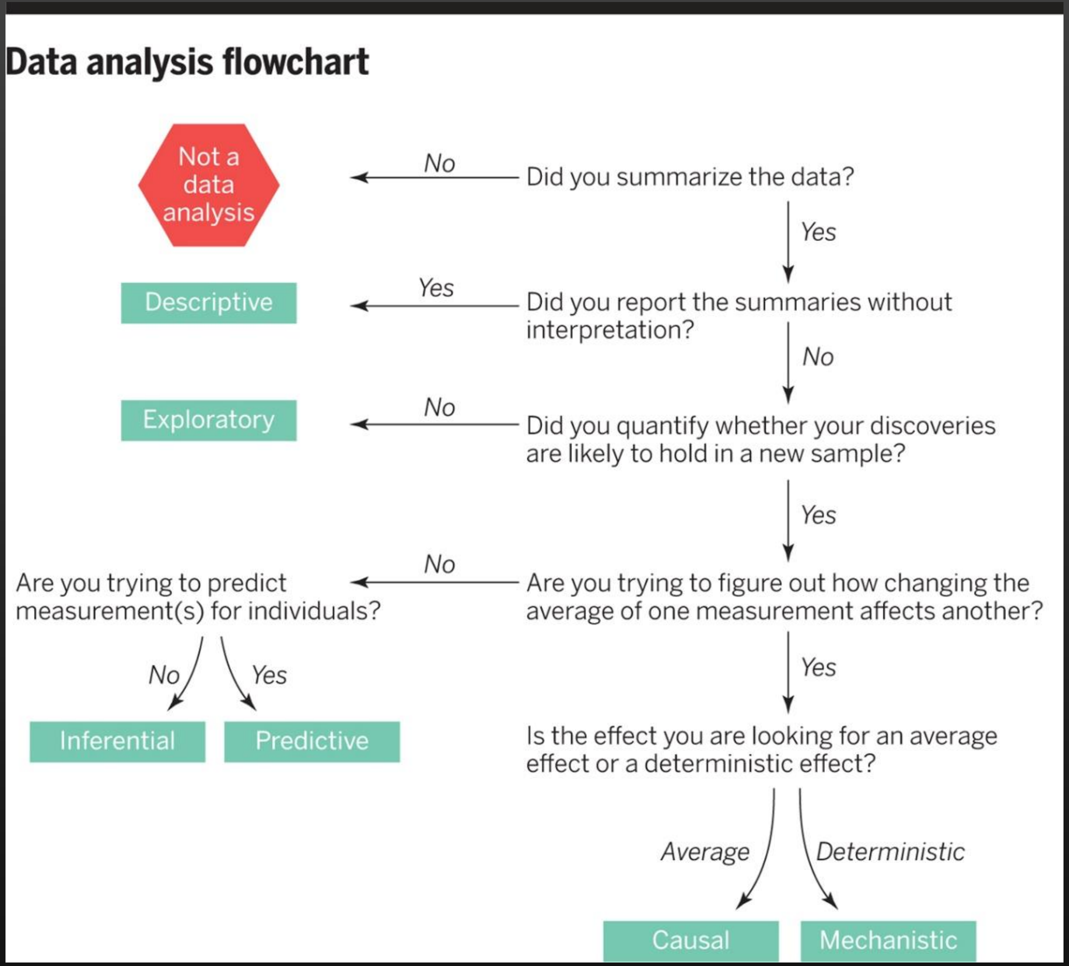
Why invest time?

- Develop and manage your workflow
- Save time and energy in the long run
- Structure allows ‘outsiders’ to understand quicker

Why invest time?

- Develop and manage your workflow
- Save time and energy in the long run
- Structure allows ‘outsiders’ to understand quicker
- Supports and reinforces design of data analyses and interpretation

Why invest time?


Data analysis flowchart
Not a data analysis
No
Did you summarize the data?
Yes
Descriptive
Yes
Did you report the summaries without interpretation?
No
Exploratory
No
Did you quantify whether your discoveries are likely to hold in a new sample?
Yes
Are you trying to predict measurement(s) for individuals?
No
Are you trying to figure out how changing the average of one measurement affects another?
No
Yes
Inferential
Predictive
Yes
Is the effect you are looking for an average effect or a deterministic effect?
Average
Deterministic
Causal
Mechanistic

Jeffery T. Leek, and Roger D. Peng. Science 2015;347:1314-1315

Components

Components

- Raw data
- Tools for data processing (i.e. cleaning)
- Tools for analyses

Components

- Raw data
- Tools for data processing (i.e. cleaning)
- Tools for analyses

Storage (tools, scripts, outputs)

Components

- Raw data
- Tools for data processing (i.e. cleaning)
- Tools for analyses

✓ Storage (tools, scripts, outputs)

- Sharing / publishing results

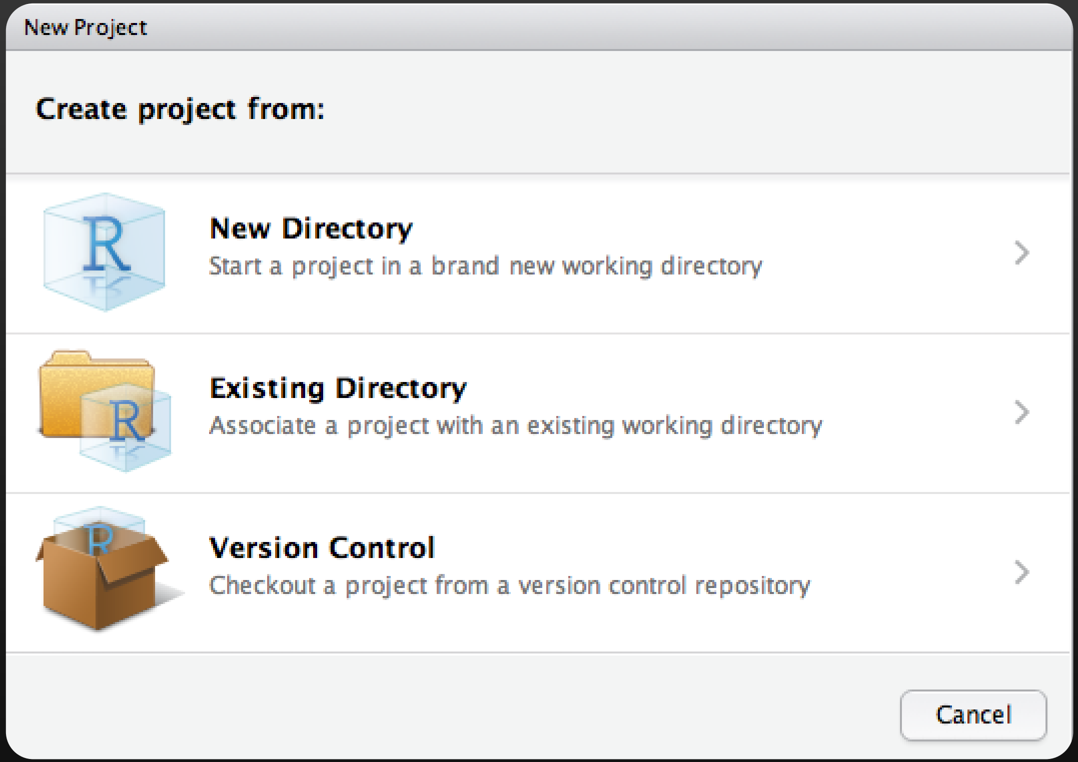
RStudio helps..

Use built-in project functionality:

RStudio helps..

Use built-in project functionality:

`File >`



New Project

Create project from:

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

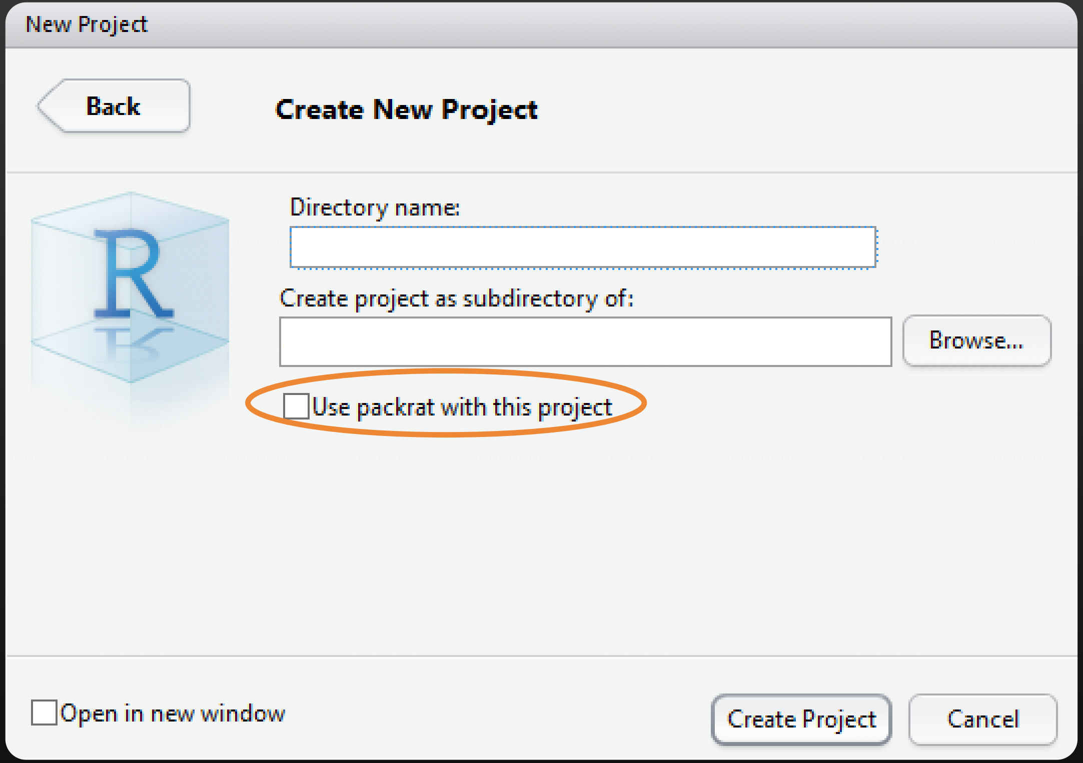
Version Control
Checkout a project from a version control repository

UNIVERSITY OF BIRMINGHAM

RStudio helps..

Use built-in project functionality:

`File >`



Directory name:

Create project as subdirectory of:

☐ Use packrat with this project

RStudio helps..

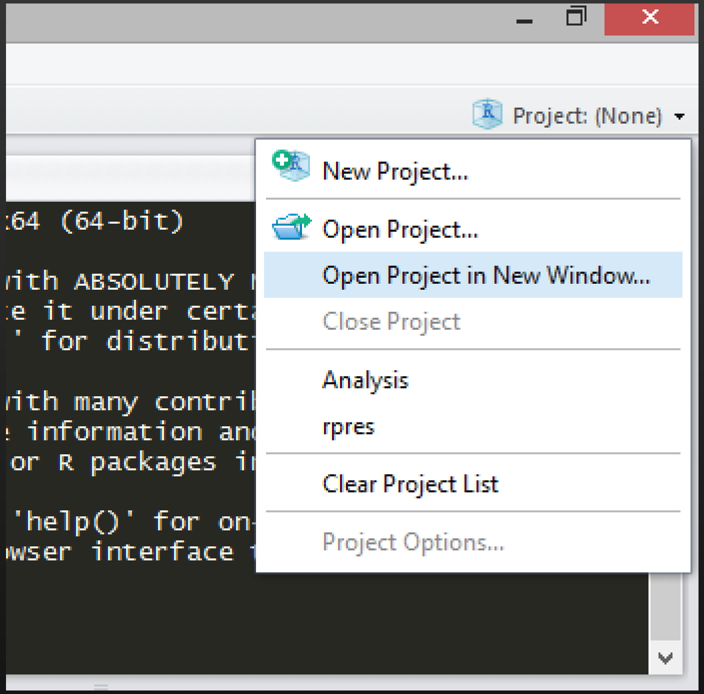
Use built-in project functionality:

- *.Rprofile* automates tasks
- *.Rdata* keeps a snapshot of your working directory
- Sets working directory automatically
- Reloads open scripts, history, project-specific settings
- …

See: https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

RStudio helps..

Use built-in project functionality:



```
:64 (64-bit)

with ABSOLUTELY N
e it under certa
' for distribut

with many contri
 information and
or R packages in

'help()' for on-
wser interface t
```

Keep track of multiple projects at the same time!

RStudio helps..

.RProfile:

```
source(‘~/your_functions.R') # Smart working directory (relative paths)
                             # Auto-load your own functions / exec. scripts

# First function called when you start R:
.First <- function(){
      cat("\nHello UseR!\n“)
}

# Last function called when you quit R:
.Last <- function(){
      cat("\nBye UseR!\n“)
}
```

UNIVERSITY OF BIRMINGHAM

RStudio helps..

.RProfile:

```
source(‘~/your_functions.R') # Smart working directory (relative paths)
                              # Auto-load your own functions / exec. scripts

# First function called when you start R:
.First <- function(){
      cat("\nHello UseR!\n“)
}

# Last function called when you quit R:
.Last <- function(){
      cat("\nBye UseR!\n“)
}
```

Transfer between systems!

UNIVERSITY OF BIRMINGHAM

PackRat helps..

New versions of R & packages..

PackRat helps..

New versions of R & packages..

- Sharing your analysis with someone = nightmare?

PackRat helps..

New versions of R & packages..

- Sharing your analysis with someone = nightmare?
- Re-running analysis after x years = nightmare?

PackRat helps..

New versions of R & packages..

- Sharing your analysis with someone = nightmare?
- Re-running analysis after x years = nightmare?
- Updating packages half-way through analysis = disaster!?

PackRat helps..

New versions of R & packages..

- Sharing your analysis with someone = nightmare?
- Re-running analysis after x years = nightmare?
- Updating packages half-way through analysis = disaster!?

PackRat acts as a safe, storing packages in the versions you used

PackRat helps..

New versions of R & packages..

- Sharing your analysis with someone = nightmare?
- Re-running analysis after x years = nightmare?
- Updating packages half-way through analysis = disaster!?

PackRat acts as a safe, storing packages in the versions you used

› isolated, portable & reproducible.

See: https://rstudio.github.io/packrat/

PackRat helps..

Getting and using PackRat:

(Requires Rtools; see: https://support.rstudio.com/hc/en-us/articles/200486498-Package-Development-Prerequisites)

```
install.packages(‘packrat')

## Tell packrat where to work
packrat::init (‘~/projects/experiment-1') # creates the ‘safe’ (local library)

## Start the job
install.packages(‘reshape2')
library(‘reshape2') # load package

## Tell packrat to save the packages your working with
packrat::snapshot()

## Check what packrat’s doing
packrat::status() # shows unused and missing packages from the ‘safe’
```

UNIVERSITY OF BIRMINGHAM

Git and GitHub help..

Git and GitHub help..

- Saving your work (scripts, analysis) is important.
- But: Overwriting a file can be catastrophic.

Git and GitHub help..

- Saving your work (scripts, analysis) is important.
- But: Overwriting a file can be catastrophic.

> Git is a *version control system* keeping track of files in your project

UNIVERSITY OF BIRMINGHAM

Git and GitHub help..

- Saving your work (scripts, analysis) is important.
- But: Overwriting a file can be catastrophic.

> Git is a *version control system* keeping track of files in your project

> Git can *mirror* your project across computers and operating systems

See: https://help.github.com/articles/good-resources-for-learning-git-and-github/

See: https://try.github.io/levels/1/challenges/1

UNIVERSITY OF BIRMINGHAM

Git and GitHub help..

Git vocabulary:

Directory	=	Local file storage
Repository	=	Central file storage tracked by Git
Save	=	Local operation
Commit	=	Tell Git to take a snapshot
Push	=	Move snapshot to repository
Clone	=	Mirror repository to your local directory

UNIVERSITY OF BIRMINGHAM

Git and GitHub help..

Git vocabulary:

Directory	=	Local file storage
Repository	=	Central file storage tracked by Git
Save	=	Local operation
Commit	=	Tell Git to take a snapshot
Push	=	Move snapshot to repository
Clone	=	Mirror repository to your local directory

> GitHub is a user-friendly way of implementing Git

See: https://github.com/

Git and GitHub help..

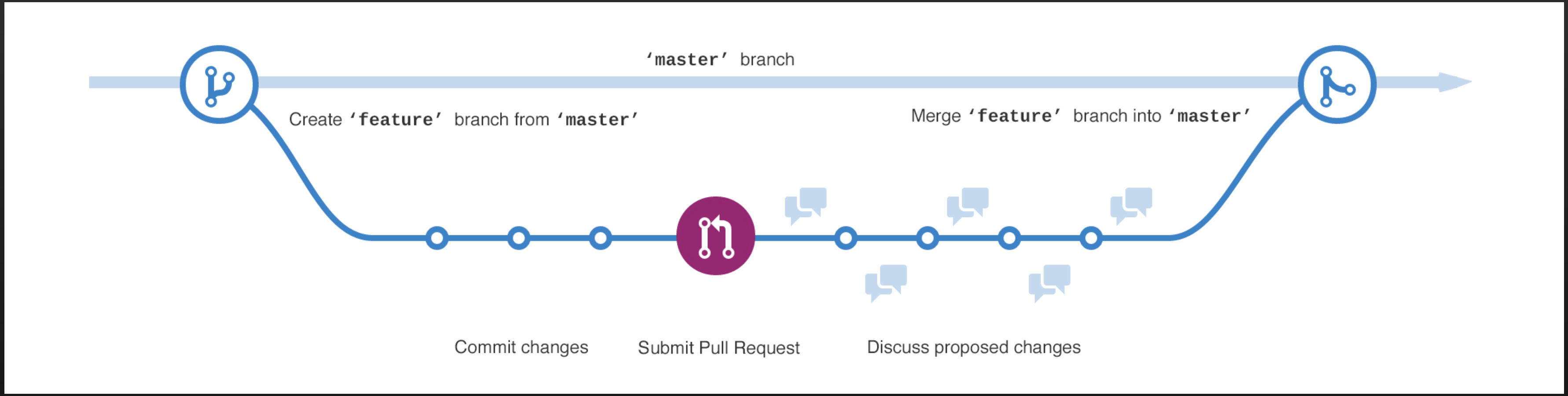
GitHub provides:

- Easy-to-use GUI
- track changes, discuss issues and collaborate

Git and GitHub help..

GitHub provides:

- Easy-to-use GUI
- track changes, discuss issues and collaborate

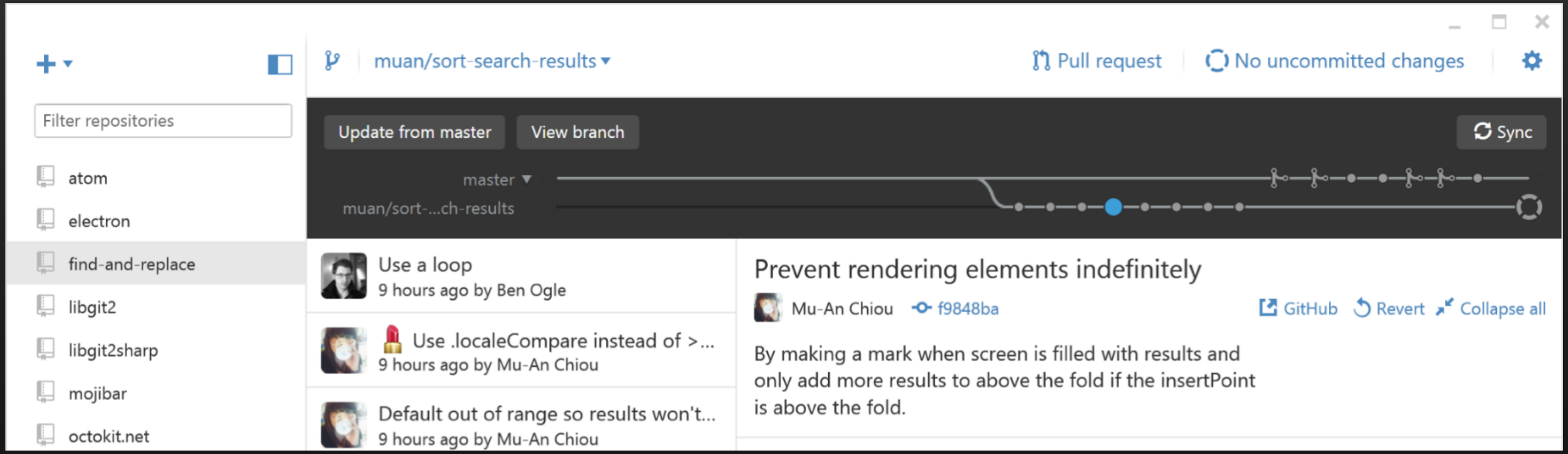

'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

See: https://guides.github.com/activities/hello-world/
See: https://github.com/the-Hull/Diss (personal repo)

Git and GitHub help..

GitHub provides:

- Easy-to-use GUI
- track changes, discuss issues and collaborate



Use a loop
9 hours ago by Ben Ogle

Use .localeCompare instead of >...
9 hours ago by Mu-An Chiou

Default out of range so results won't...
9 hours ago by Mu-An Chiou

Prevent rendering elements indefinitely

Mu-An Chiou f9848ba

By making a mark when screen is filled with results and only add more results to above the fold if the insertPoint is above the fold.

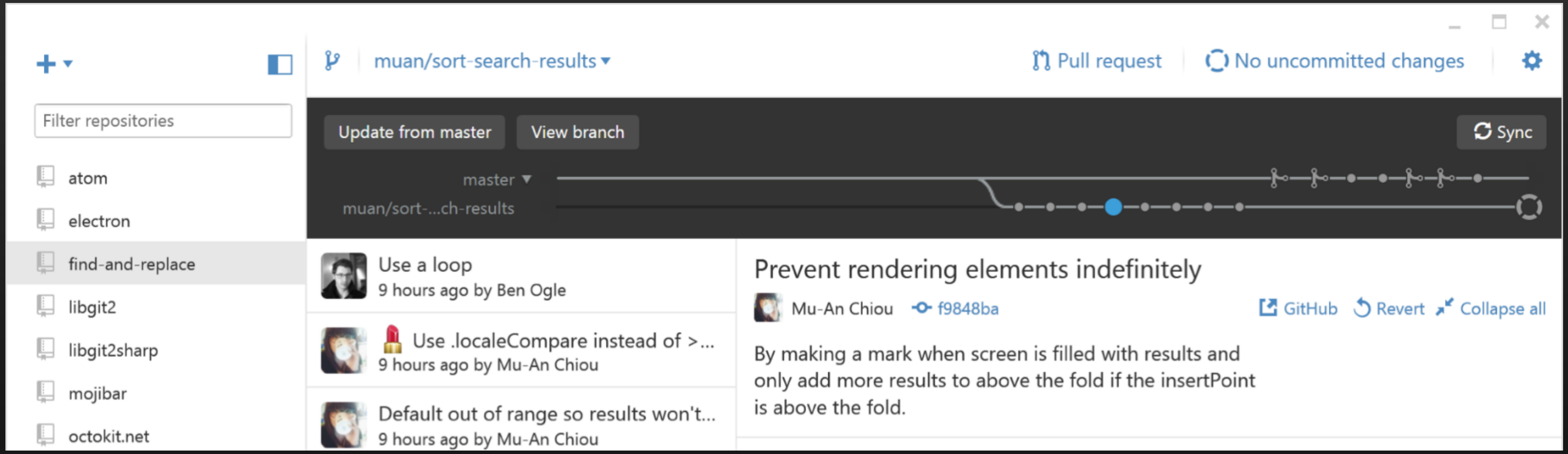
See: https://desktop.github.com/

UNIVERSITY OF BIRMINGHAM

Git and GitHub help..

GitHub provides:

- Easy-to-use GUI
- track changes, discuss issues and collaborate



Use a loop
9 hours ago by Ben Ogle

Use .localeCompare instead of >...
9 hours ago by Mu-An Chiou

Default out of range so results won't...
9 hours ago by Mu-An Chiou

Prevent rendering elements indefinitely
Mu-An Chiou f9848ba

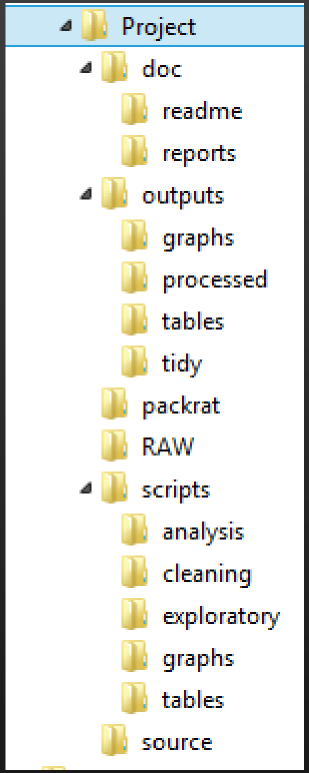
By making a mark when screen is filled with results and only add more results to above the fold if the insertPoint is above the fold.

See: https://desktop.github.com/

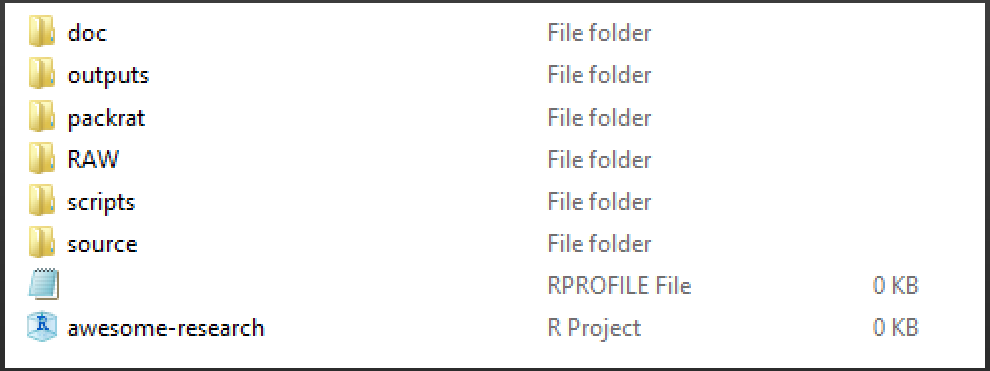
Hierarchy & Order help..

Personal preference of a project:

- RAW
- Scripts
  - cleaning
  - exploratory
  - analysis
  - graphs
  - tables
- Source (helper functions)
- Outputs
  - tidy data
  - processed
  - graphs
  - tables
- Documentation (readme files, reports)
- PackRat folders



```
Project
  doc
    readme
    reports
  outputs
    graphs
    processed
    tables
    tidy
  packrat
  RAW
  scripts
    analysis
    cleaning
    exploratory
    graphs
    tables
  source
```



doc	File folder	
outputs	File folder	
packrat	File folder	
RAW	File folder	
scripts	File folder	
source	File folder	
	RPROFILE File	0 KB
awesome-research	R Project	0 KB

Data Management

Data.. set?

Components:

Data.. set?

Components:

- Raw Data
- Scripts (processing)
- Tidy Data

Data.. set?

Components:

- Raw Data
- Scripts (processing)
- Tidy Data
  - Code book

Data.. set?

Code Book:

Data.. set?

Code Book:

- Information on origin and type of raw data (incl. missing values, units)
- Detail choices and steps made for processing data (e.g. renaming, averaging)
- Outline study design (e.g. for identifying confounding factors)

Data.. set?

Code Book:

- Information on origin and type of raw data (incl. missing values, units)
- Detail choices and steps made for processing data (e.g. renaming, averaging)
- Outline study design (e.g. for identifying confounding factors)

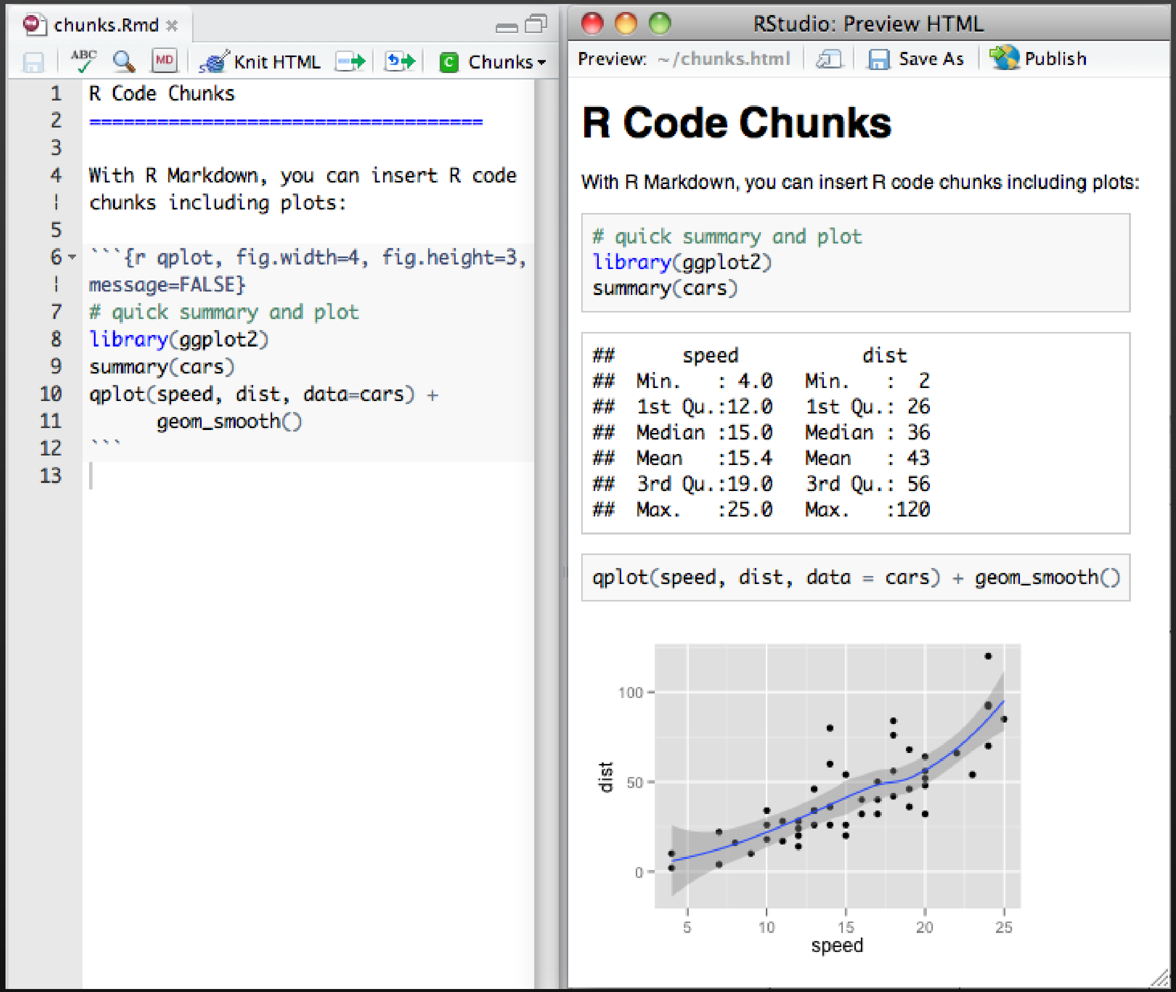
- Formats: **.docx*, **.txt*, **.Rmd* (RMarkdown)

See: http://rmarkdown.rstudio.com/

Data.. set?

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Code Book:



~~~
R Code Chunks
=====================================

With R Markdown, you can insert R code
chunks including plots:

```{r qplot, fig.width=4, fig.height=3,
message=FALSE}
# quick summary and plot
library(ggplot2)
summary(cars)
qplot(speed, dist, data=cars) +
      geom_smooth()
```

~~~

R Code Chunks

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

```
qplot(speed, dist, data = cars) + geom_smooth()
```

See: http://rmarkdown.rstudio.com/

UNIVERSITY OF BIRMINGHAM

Data.. storage?

Think ahead:

Data.. storage?

Think ahead:

- Use clear, systematic names with common identifiers:

Data.. storage?

Think ahead:

- Use clear, systematic names with common identifiers:

exp1-plot1.csv
exp1-plot2.csv
exp2-plot1.csv
exp2-plot2.csv

Data.. storage?

Think ahead:

- Use clear, systematic names with common identifiers:

exp1-plot1.csv
exp1-plot2.csv
exp2-plot1.csv
exp2-plot2.csv

```
## get a list of all files of exp1
exp1_list <- list.files(path = '~/raw',
                        pattern = "exp1")

## load all data from exp1 into the working environment
## and do the same operation…

## use file names to label objects
exp_labels <-gsub(pattern = '-plot.*', replacement = '', x = exp1_list)
```

UNIVERSITY OF BIRMINGHAM

Data.. storage?

Think ahead:

- Use clear, systematic names with common identifiers:

exp1-plot1.csv
exp1-plot2.csv
exp2-plot1.csv
exp2-plot2.csv

```
## get a list of all files of exp1
exp1_list <- list.files(path = '~/raw',
                        pattern = "exp1")

## load all data from exp1 into the working environment
## and do the same operation…

## use file names to label objects
exp_labels <-gsub(pattern = '-plot.*', replacement = '', x = exp1_list)
```

UNIVERSITY OF BIRMINGHAM

Data.. storage?

Keeping your work organized in R:

environments are ‘folders’ in R’s working environment

Data.. storage?

Keeping your work organized in R:

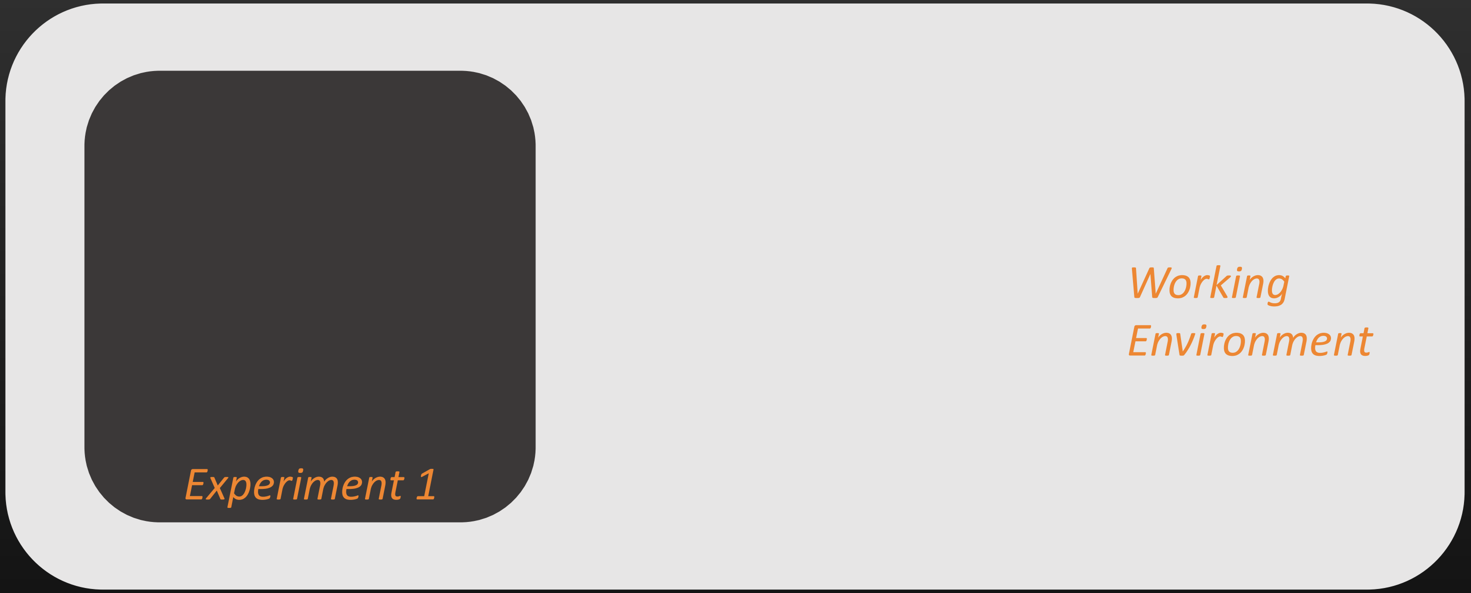
environments are ‘folders’ in R’s working environment

Data.. storage?

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Keeping your work organized in R:

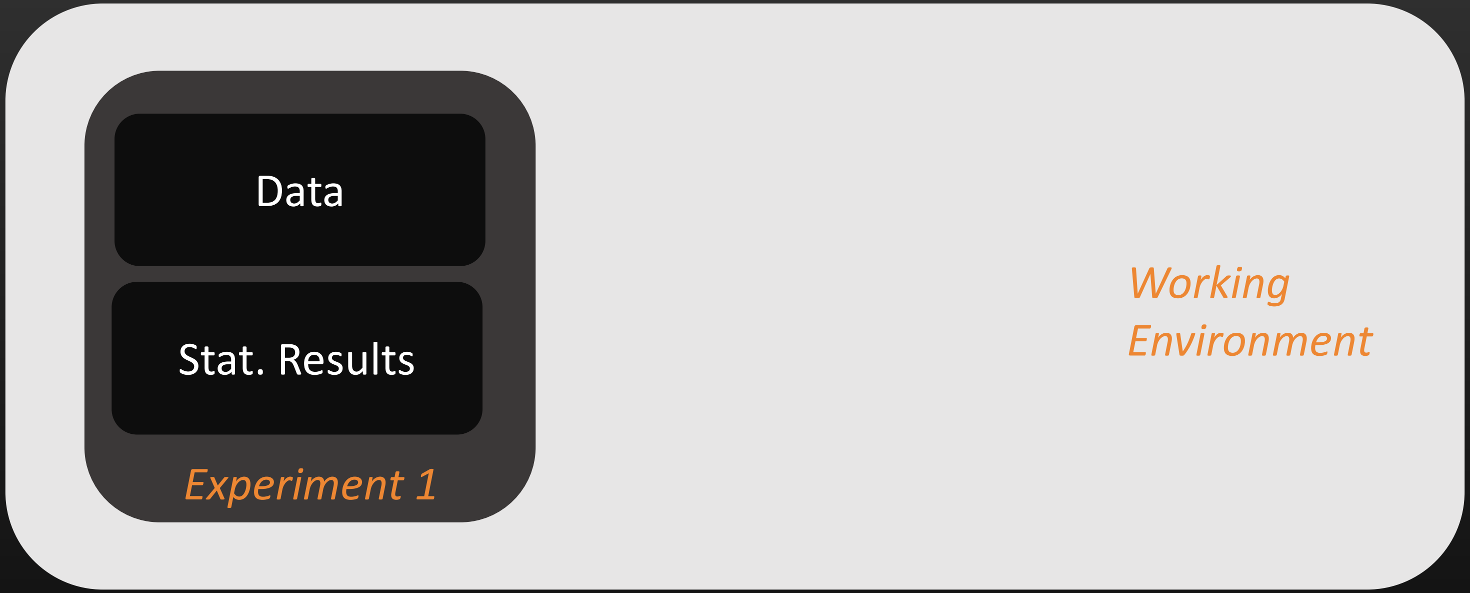
environments are ‘folders’ in R’s working environment



Data.. storage?

Keeping your work organized in R:

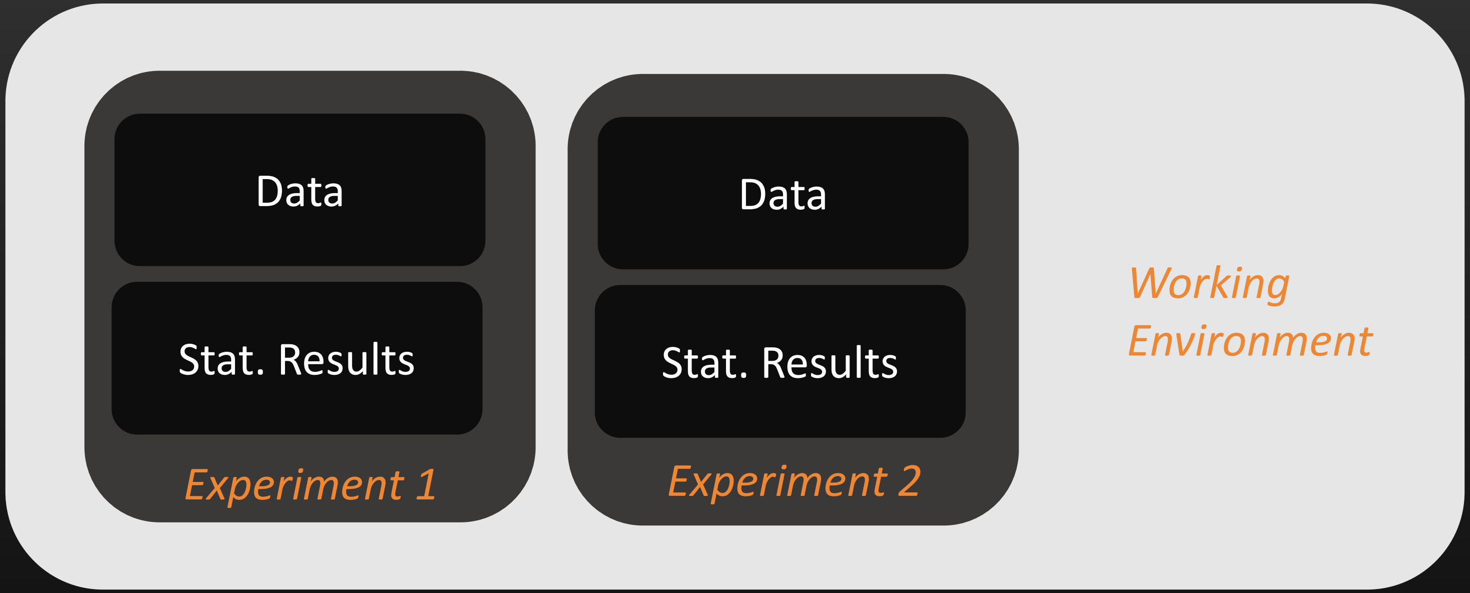
environments are ‘folders’ in R’s working environment



Data.. storage?

Keeping your work organized in R:

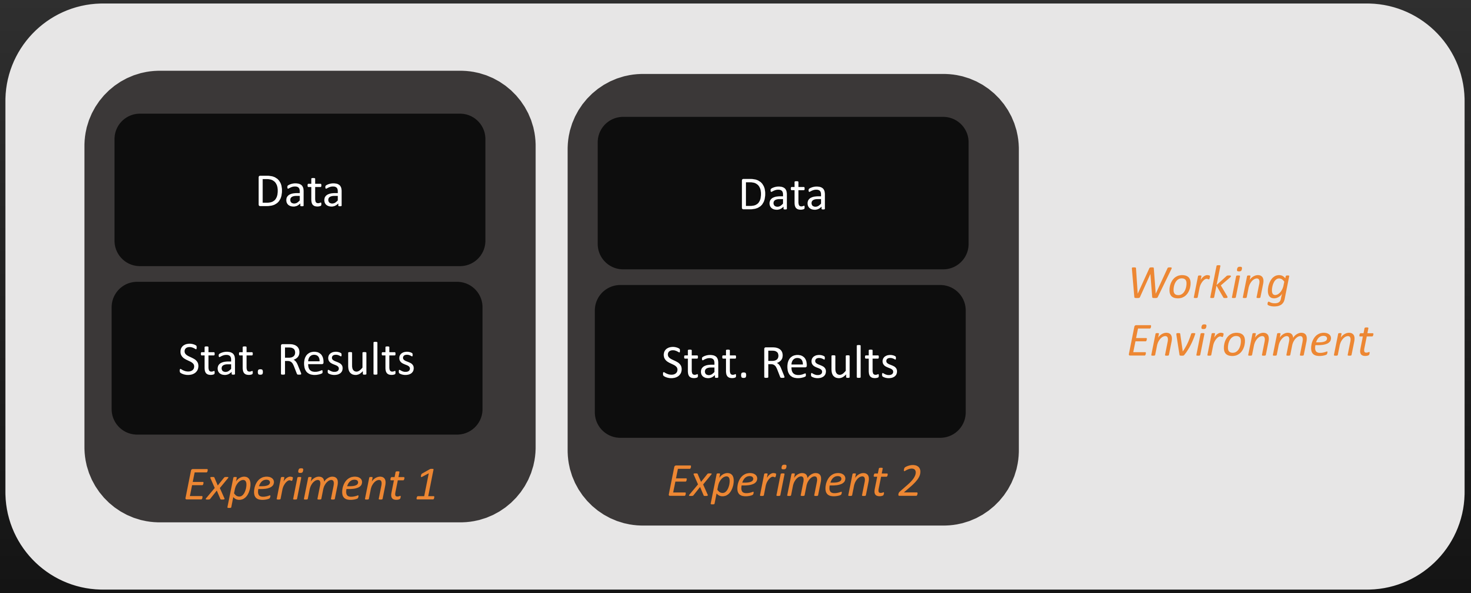
environments are ‘folders’ in R’s working environment



Data.. storage?

Keeping your work organized in R:

environments are 'folders' in R's working environment

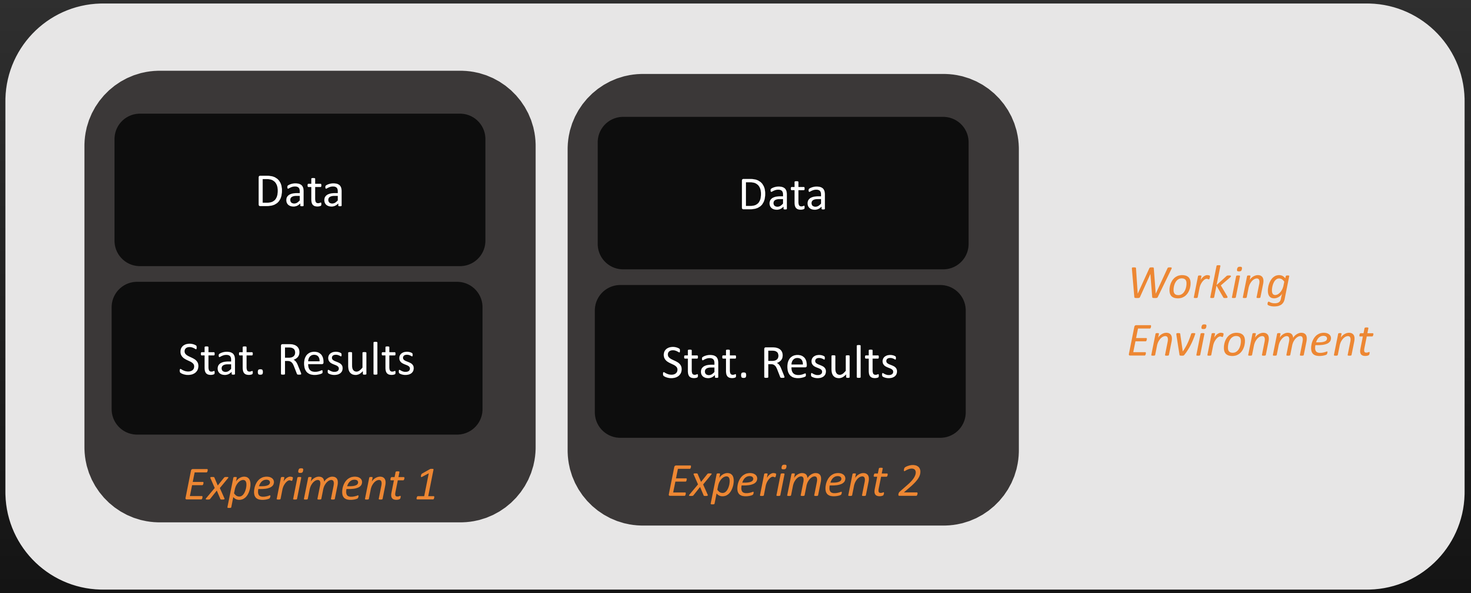


Important functions:
new.env()
assign()
get()
save() / *load*()

Data.. storage?

Keeping your work organized in R:

environments are ‘folders’ in R’s working environment



Important functions:

new.env()
assign()
get()
save() / *load*()

Benefits:

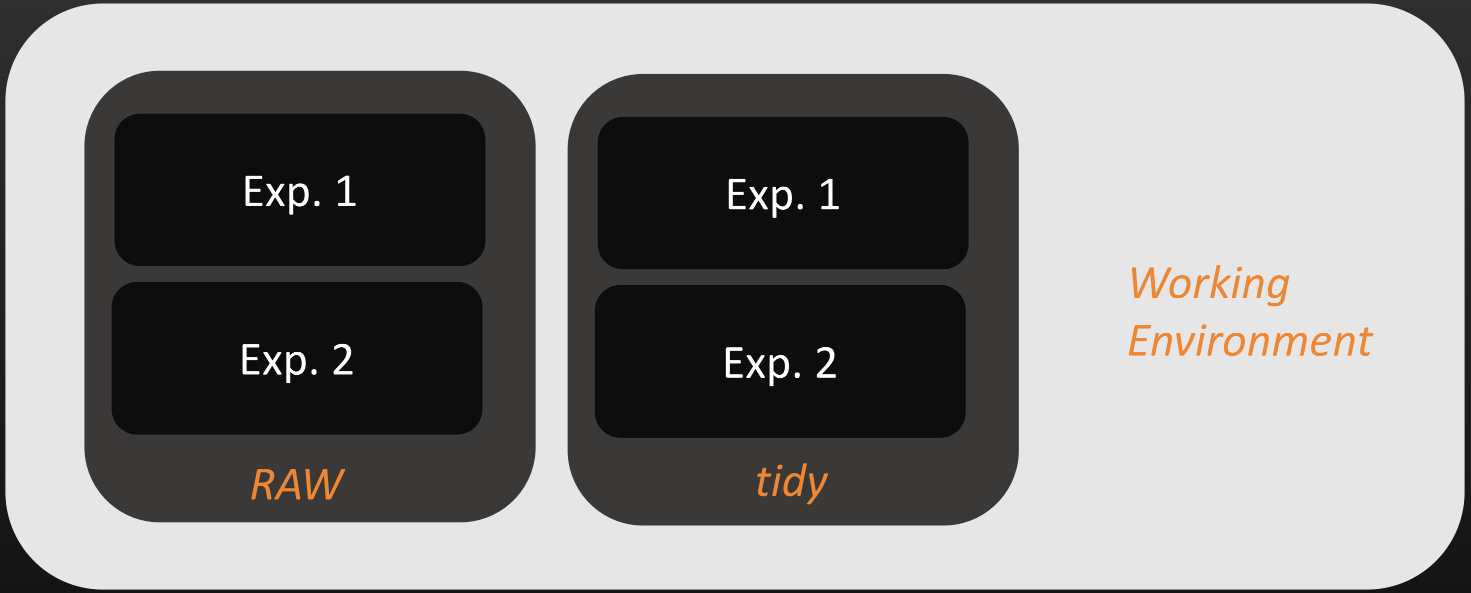
- Clean working env.
- Less chance of confusing objects
- Repeating processes (e.g. loops)
- Saving & re-using outputs

See: http://adv-r.had.co.nz/Environments.html

Data.. storage?

Keeping your work organized in R:

environments are 'folders' in R's working environment



Important functions:

new.env()
assign()
get()
save() / *load*()

Benefits:

- Clean working env.
- Less chance of confusing objects
- Repeating processes (e.g. loops)
- Saving & re-using outputs

See: http://adv-r.had.co.nz/Environments.html

Data.. help?

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Useful packages and links for data processing:

- dplyr
- reshape2

See: http://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf (!)

UNIVERSITY OF BIRMINGHAM

Coding

Expressing yourself

- Writing code is like using a ‘natural’ language

Expressing yourself

- Writing code is like using a ‘natural’ language
  - syntax, semantics (and beauty?)
  - learning is time consuming
  - rewarding

Guidelines

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c("exp1", "exp2") # consider using function ls() to obtain names in working env.

# Necessary functions ----------------------------------------------------------------------------------
averageExp <- function(experiments){
                    ## function takes character vector of experiment names
                    ## names and averages outcomes. Results are stored in new vector


                    data.temporary <- mget(experiments,
                                           envir = .GlobalEnv) # get values from environment

                    result <- sapply(data.temporary, mean)
}
# Calculation ------------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data ------------------------------------------------------------------------------------------
save(file = "~/output/processed/exp.means.rda", x = exp.means)
```

UNIVERSITY OF BIRMINGHAM

Guidelines

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c("exp1", "exp2") # consider using function ls() to obtain names in working env.

# Necessary functions -------------------------------------------------------------------------------
averageExp <- function(experiments){
                              ## function takes character vector of experiment names
                              ## names and averages outcomes. Results are stored in new vector


                              data.temporary <- mget(experiments,
                                                     envir = .GlobalEnv) # get values from environment

                              result <- sapply(data.temporary, mean)
}
# Calculation ------------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data ------------------------------------------------------------------------------------------
save(file = "~/output/processed/exp.means.rda", x = exp.means)
```

Clear description at top

UNIVERSITY OF BIRMINGHAM

Guidelines

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Example:

```r
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c(“exp1”, “exp2” # consider using function ls() to obtain names in working env.

# Necessary functions ------------------------------------------------------------------------------
averageExp <- function(experiments){
                    ## function takes character vector of experiment names
                    ## names and averages outcomes. Results are stored in new vector


                    data.temporary <- mget(experiments,
                                           envir = .GlobalEnv) # get values from environment

                    result <- sapply(data.temporary, mean)
}
# Calculation ------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data -----------------------------------------------------------------------------------
save(file = “~/output/processed/exp.means.rda”, x = exp.means)
```

Clear description at top

Useful comments when needed

Guidelines

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c(“exp1”, “exp2”) # consider using function ls() to obtain names in working env.

# Necessary functions -----------------------------------------------------------------------------
averageExp <- function(experiments){
                           ## function takes character vector of experiment names
                           ## names and averages outcomes. Results are stored in new vector


                           data.temporary <- mget(experiments,
                                                  envir = .GlobalEnv) # get values from environment

                           result <- sapply(data.temporary, mean)

# Calculation -------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data -------------------------------------------------------------------------------------
save(file = “~/output/processed/exp.means.rda”, x = exp.means)
```

Clear description at top

Useful comments when needed

Separated into chunks

Guidelines

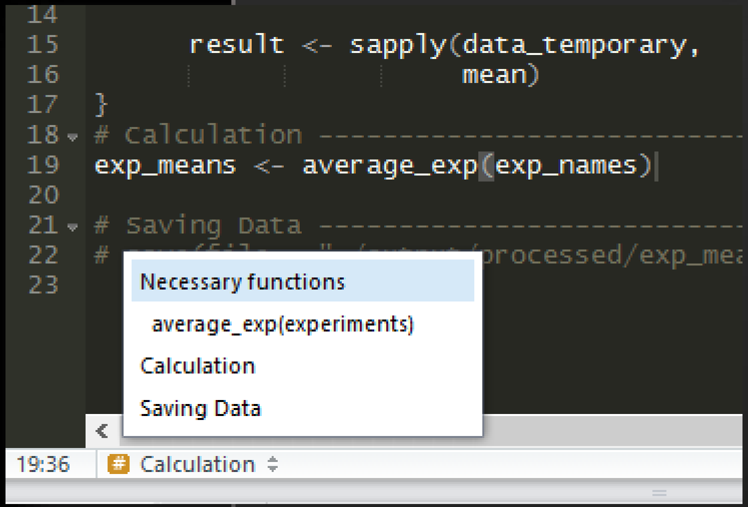
Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c("exp1", "exp2") # consider using function ls() to obtain names in working env.

# Necessary functions ---------------------------------------------------------------------------
averageExp <- function(experiments){
                         ## function takes character vector of experiment names
                         ## names and averages outcomes. Results are stored in new vector


                         data.temporary <- mget(experiments,
                                                  envir = .GlobalEnv) # get values from environment

                         result <- sapply(data.temporary, mean)

# Calculation ------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data ------------------------------------------------------------------------------------
save(file = "~/output/processed/exp.means.rda", x = exp.means)
```

Clear description at top

Useful comments when needed

Separated into chunks



```
        result <- sapply(data_temporary,
                         mean)
}
# Calculation ---------------------------
exp_means <- average_exp(exp_names)

# Saving Data ---------------------------
#                    processed/exp_mea
```

UNIVERSITY OF BIRMINGHAM

Guidelines

● Introduction ● Project Set-Up ● Data Management ● Coding ● Conclusion

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c("exp1", "exp2") # consider using function ls() to obtain names in working env.

# Necessary functions ----------------------------------------------------------------------------------
averageExp <- function(experiments){
                    ## function takes character vector of experiment names
                    ## names and averages outcomes. Results are stored in new vector


                    data.temporary <- mget(experiments,
                                           envir = .GlobalEnv) # get values from environment

                    result <- sapply(data.temporary, mean)
}
# Calculation ---------------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data --------------------------------------------------------------------------------------------
save(file = "~/output/processed/exp.means.rda", x = exp.means)
```

Clear description at top

Useful comments when needed

Separated into chunks

Defined function

Guidelines

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c(“exp1”, “exp2”) # consider using function ls() to obtain names in working env.

# Necessary functions ---------------------------------------------------------------------------------
averageExp <- function(experiments){
                                ## function takes character vector of experiment names
                                ## names and averages outcomes. Results are stored in new vector


                                data.temporary <- mget(experiments,
                                                        envir = .GlobalEnv) # get values from environment

                                result <- sapply(data.temporary, mean)
}
# Calculation -------------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data -------------------------------------------------------------------------------------------
save(file = “~/output/processed/exp.means.rda”, x = exp.means)
```

Clear description at top

Useful comments when needed

Separated into chunks

Defined function

Useful object names

UNIVERSITY OF BIRMINGHAM

Guidelines

Example:

```
## This script is part of awesome-research phase 1. It takes raw data gathered from exp 1
## and exp 2 (g/cm), calculates their means, and saves the result for later use in output/processed.

exp1 <- c(1, 2, 3) #loads raw data
exp2 <- c(4, 5, 6)
exp.names <- c(“exp1”, “exp2”) # consider using function ls() to obtain names in working env.

# Necessary functions ---------------------------------------------------------------------------------
averageExp <- function(experiments){
                    ## function takes character vector of experiment names
                    ## names and averages outcomes. Results are stored in new vector


                    data.temporary <- mget(experiments,
                                           envir = .GlobalEnv) # get values from environment

                    result <- sapply(data.temporary, mean)
}
# Calculation -----------------------------------------------------------------------------------------
exp.means <- averageExp(exp.names)

# Saving Data -----------------------------------------------------------------------------------------
save(file = “~/output/processed/exp.means.rda”, x = exp.means)
```

Clear description at top

Useful comments when needed

Separated into chunks

Defined function

Useful object names

Line wrapping

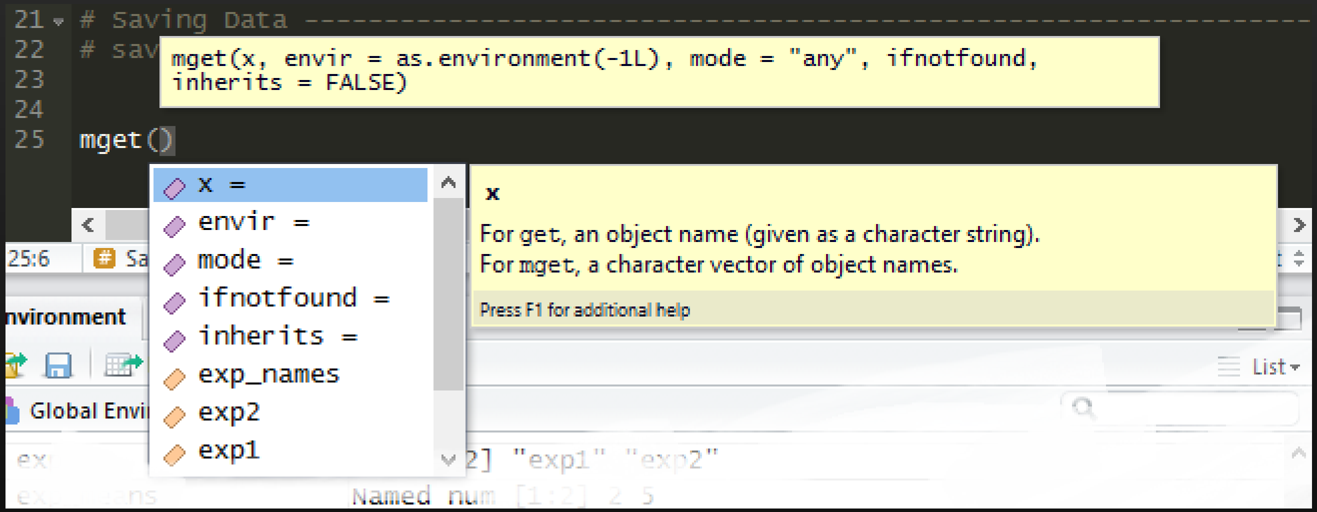
Guidelines

- Other Guidelines for R:
  - naming (funCtions, var.iables, CONSTANTS)
  - indention (2 to 4 spaces)
  - spacing for operators (4 + 4; x[1,])
  - assign with <- not =

See: https://google.github.io/styleguide/Rguide.xml

RStudio helps again..

- Great built-in features:
  - code completion on tab



```
# Saving Data ---------------------------------------------------------
# sav
mget(x, envir = as.environment(-1L), mode = "any", ifnotfound,
inherits = FALSE)

mget()
```

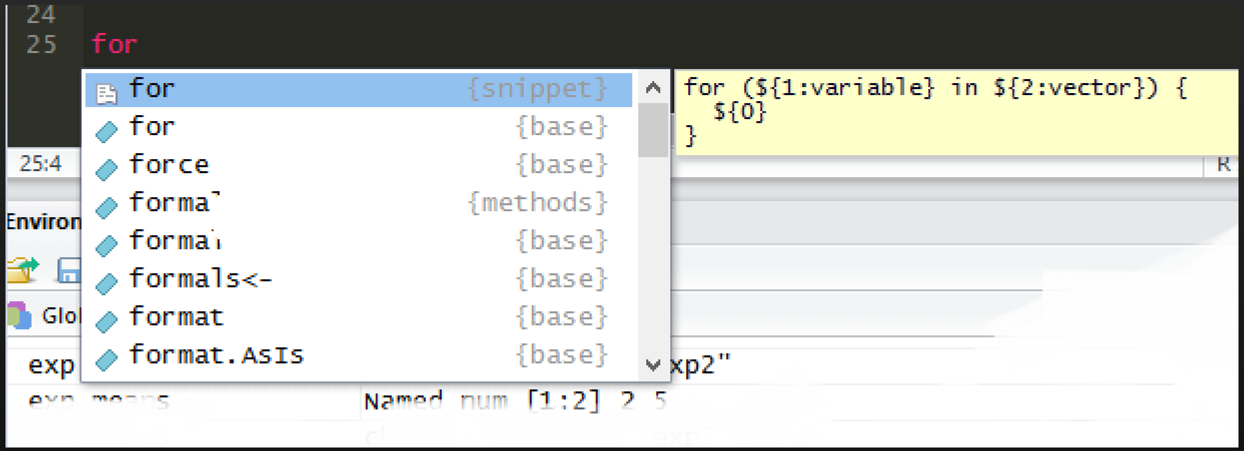
x

For get, an object name (given as a character string).
For mget, a character vector of object names.

Press F1 for additional help

RStudio helps again..

- Great built-in features:
  - code completion on tab
  - code snippets for loops / if-else / functions



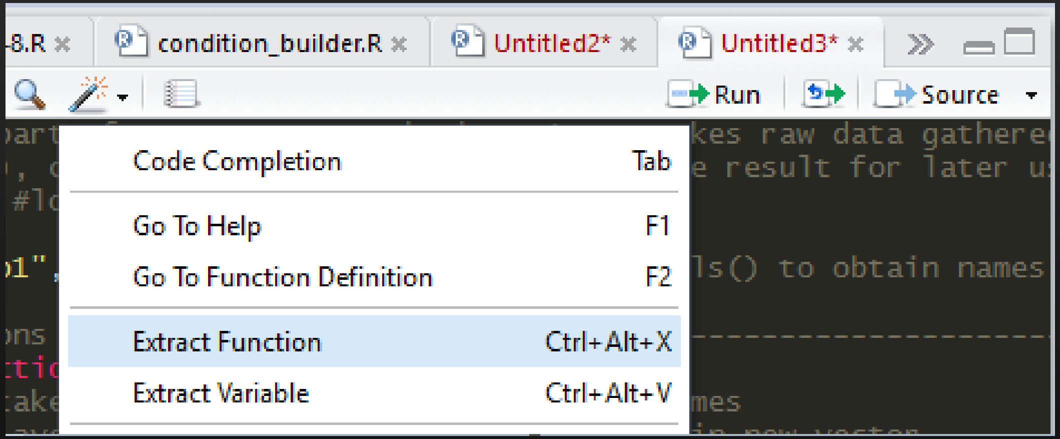
```
for
```

```
for          {snippet}
for          {base}
force        {base}
formals      {methods}
formals      {base}
formals<-    {base}
format       {base}
format.AsIs  {base}
```

```
for (${1:variable} in ${2:vector}) {
  ${0}
}
```

RStudio helps again..

- Great built-in features:
  - code completion on tab
  - code snippets for loops / if-else / functions
  - function wizard



UNIVERSITY OF BIRMINGHAM

Conclusion

Reproducible?

- Using the provided tools and guidelines you should be able to:
  - understand some of the terminology used in books / online
  - find other helpful resources
  - develop your own workflow
  - move your analyses between computers
  - share your work with others and collaborate with them
  - **reproduce your research**

Reproducible?

- Using the provided tools and guidelines you should be able to:
  - understand some of the terminology used in books / online
  - find other helpful resources
  - develop your own workflow
  - move your analyses between computers
  - share your work with others and collaborate with them
  - **reproduce your research**

… Enjoy.

UNIVERSITY OF BIRMINGHAM

Thanks for enduRing!

UNIVERSITY OF BIRMINGHAM