Best Practices in R

2019-08-29

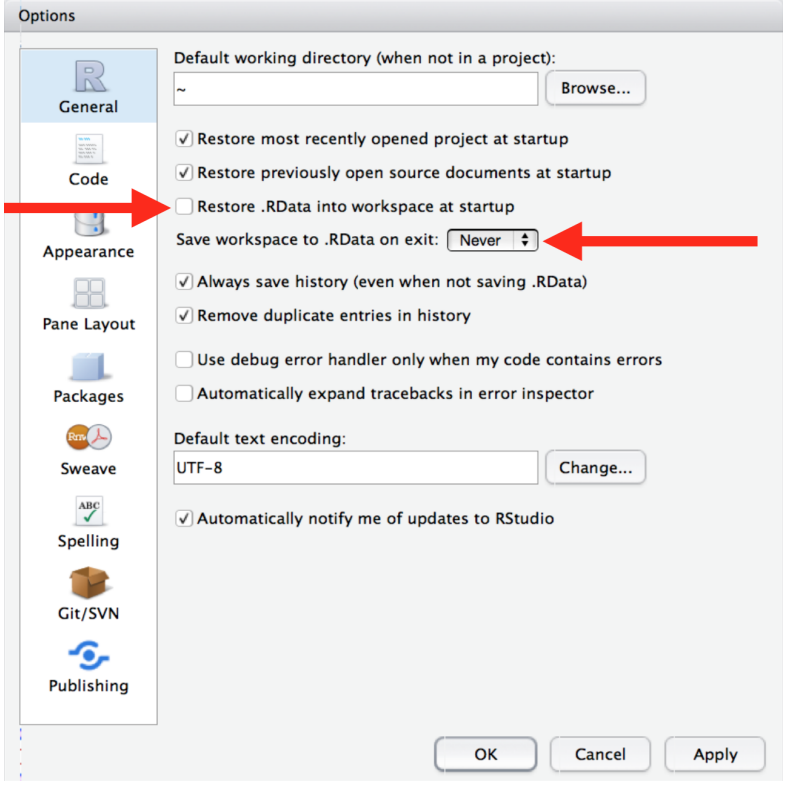
developed by Emil Hvitfeldt

Change Settings

Keyboard shortcut to open settings
⌘ + , in Mac OS,
ctrl + , in Windows

✓ - Uncheck "Restore .RData into work space at start up"

✓ - Set "Save work space to .Rdata on exit" to "Never"



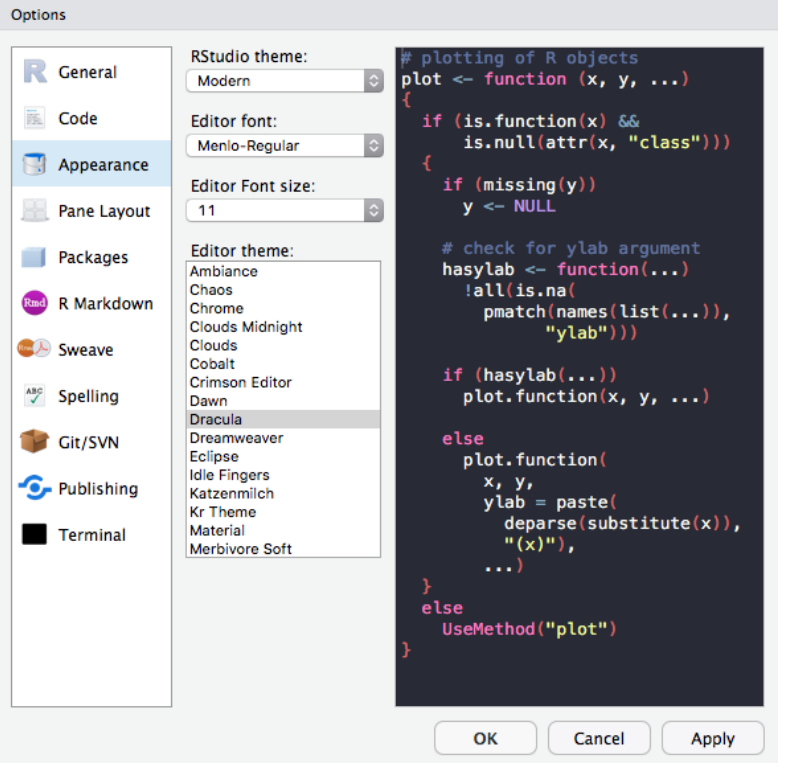
Change Appearance

RStudio themes

Fonts

Font Sizes

Editor Themes



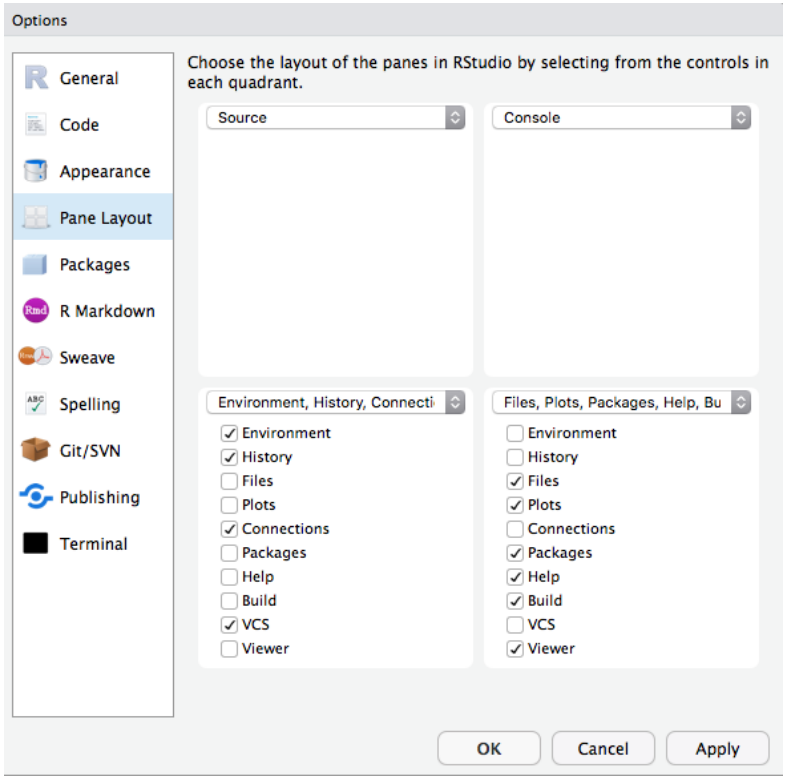
Pane layouts

Change the layout of the panes

Source on top?

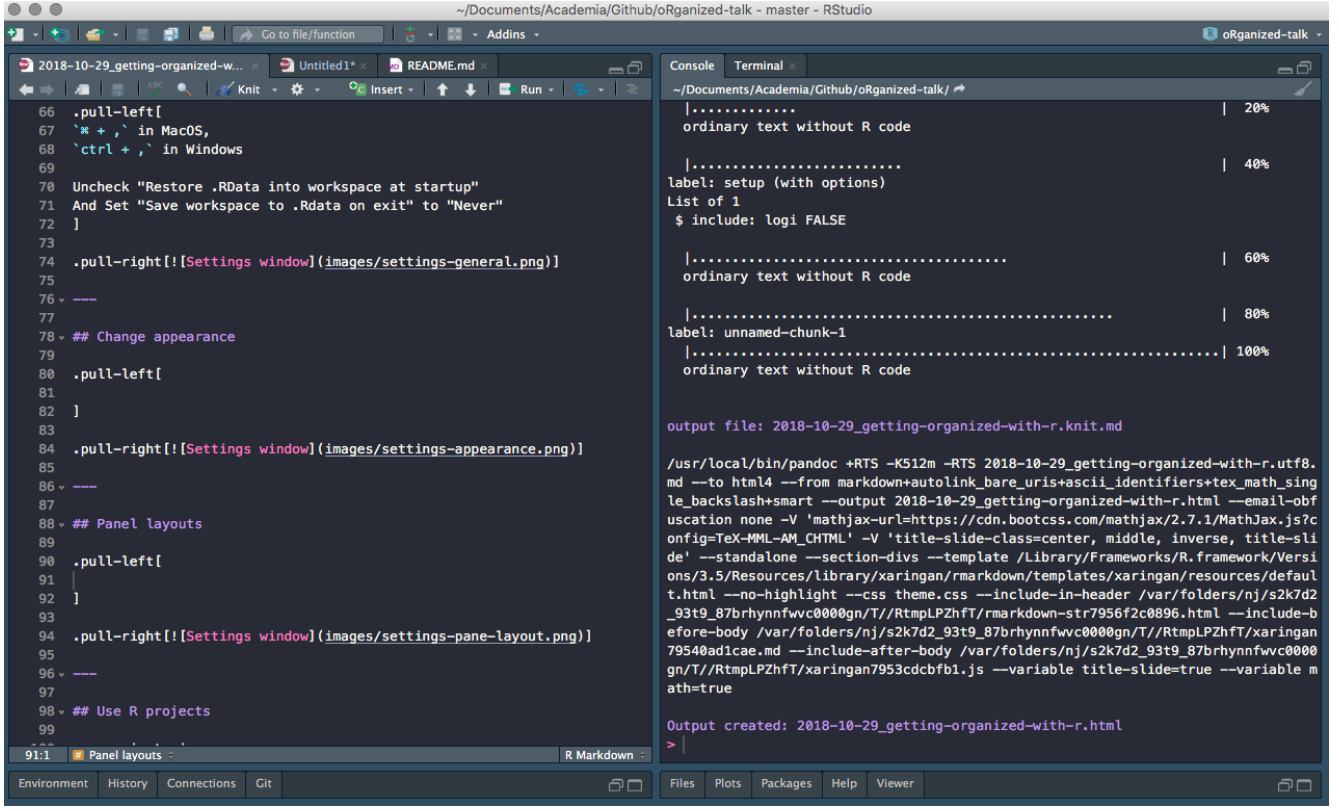
Source down to the right?

It's all up to you!



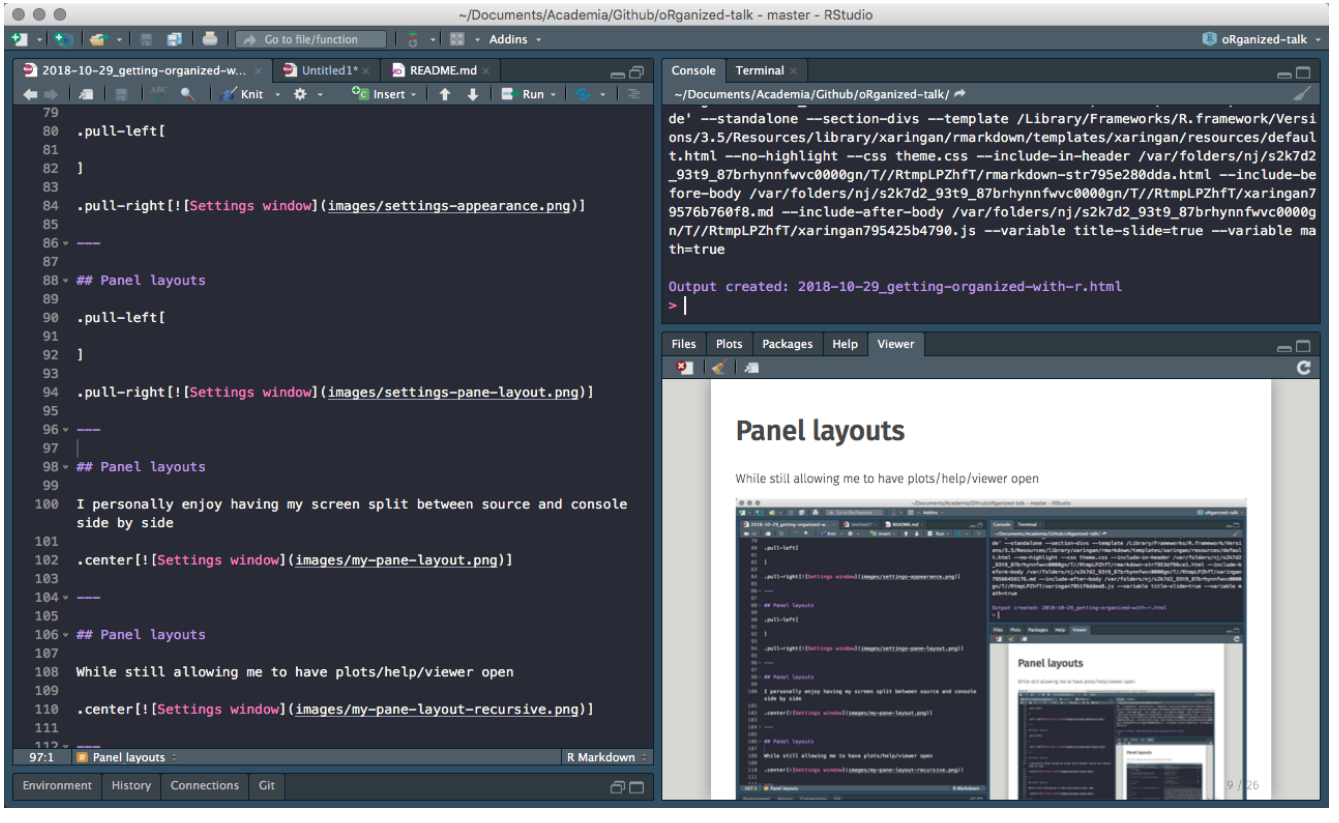
Pane layouts

Some like having both source and console open



Pane layouts

...while still allowing to have viewer open



RStudio Projects

Keep all files from one project together. Use RStudio projects.

Self contained

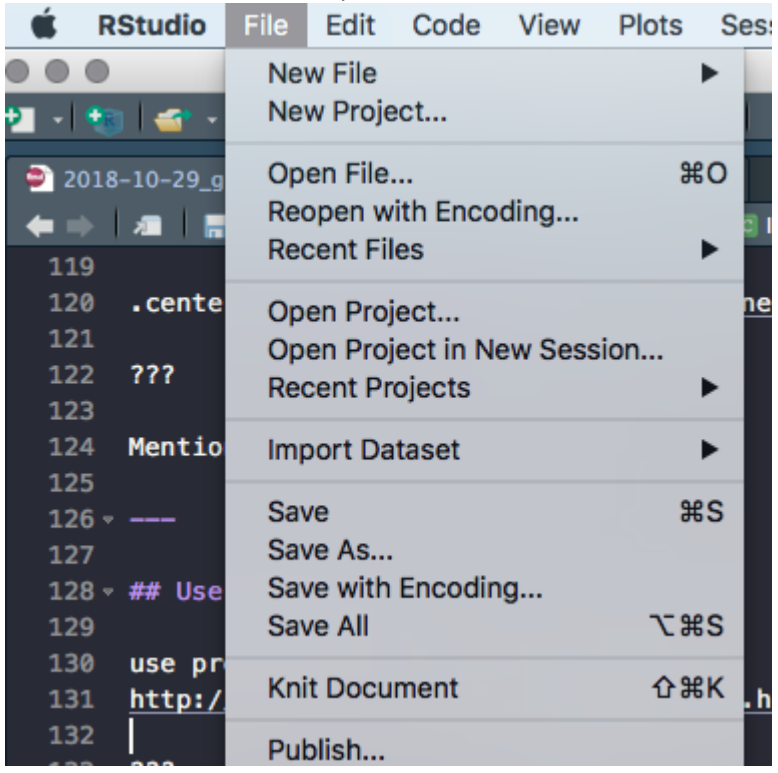
Project orientated

usethis

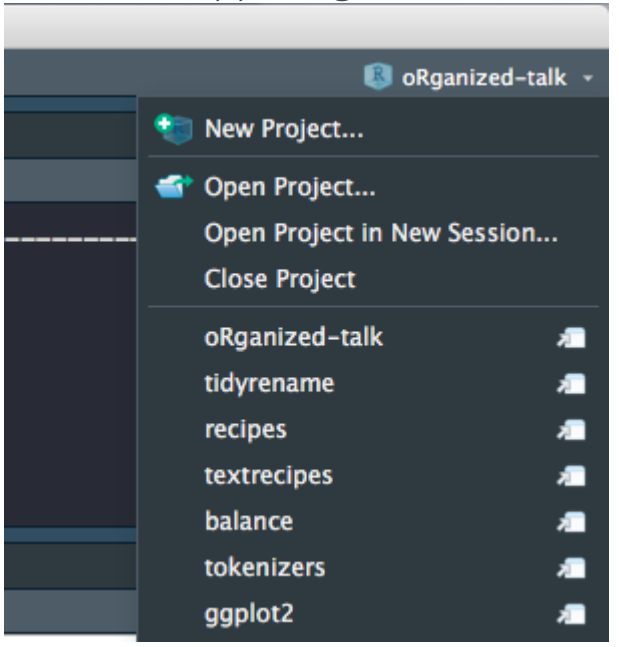
```
usethis::create_project("project_name")
```

RStudio Projects - Creation 1 / 4

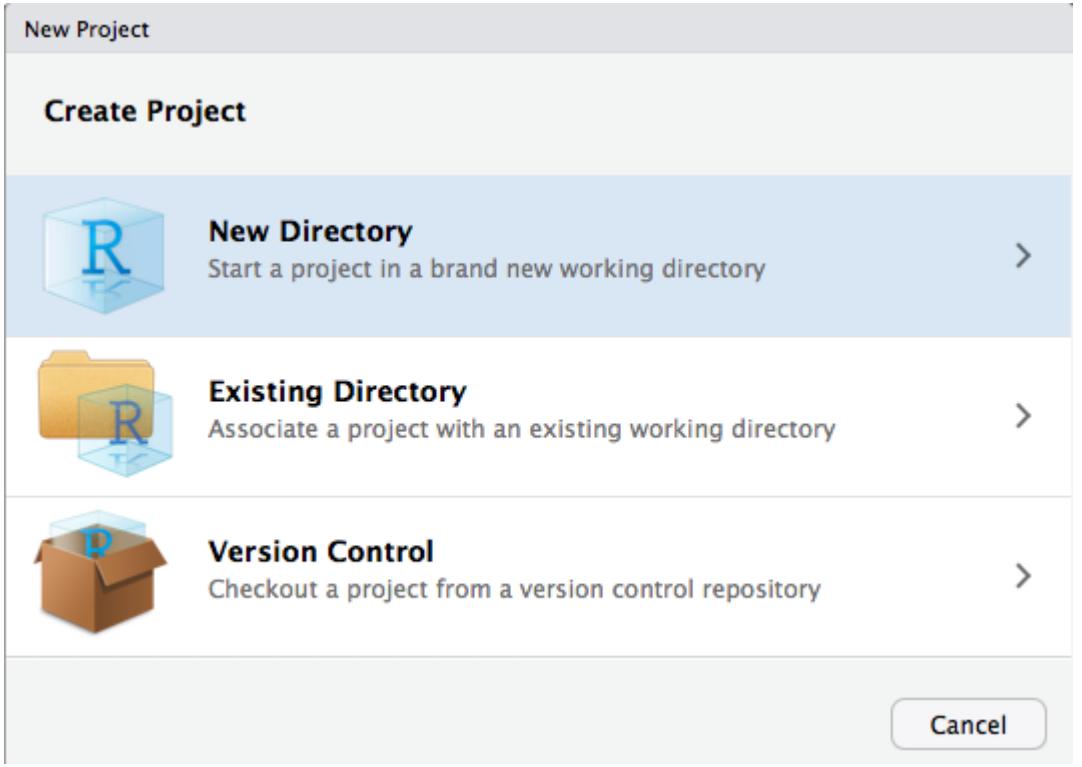
Click File > New Project



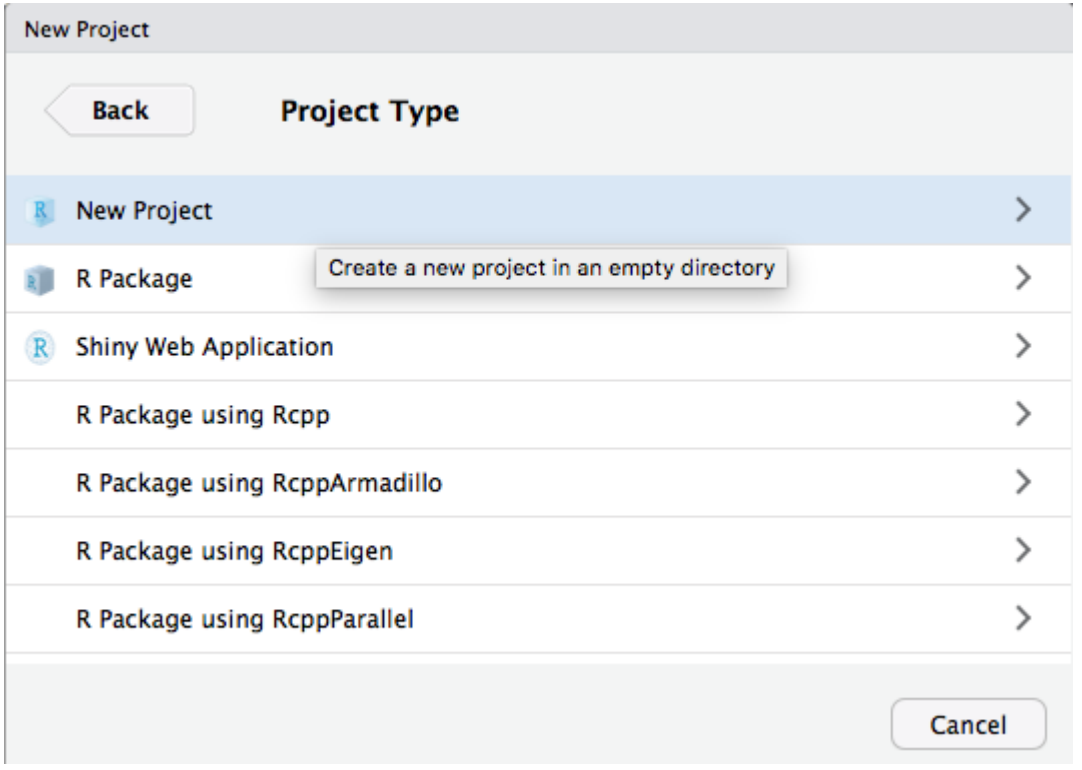
Or click on the upper right



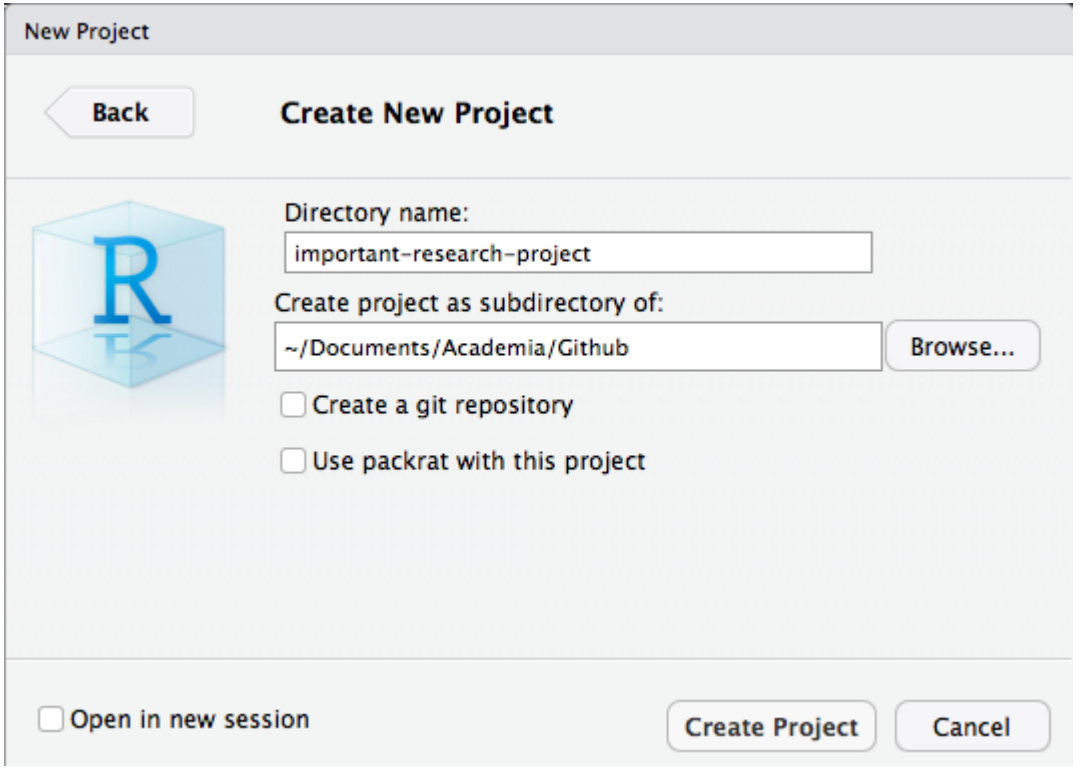
RStudio Projects - Creation 2 / 4



RStudio Projects - Creation 3 / 4



RStudio Projects - Creation 4 / 4



Folder Structure

```
name_of_project
|--raw_data
    |--WhateverData.xlsx
    |--report_2017.csv
|--output_data
    |--summary2017.csv
|--rmd
    |--01-analysis.Rmd
|--docs
    |--01-analysis.html
    |--01-analysis.pdf
|--scripts
    |--exploratory_analysis.R
    |--pdf_scraper.R
|--figures
    |--weather_2017.png
|--name_of_project.Rproj
|--run_all.R
```

1. Raw data separate from cleaned data
2. Reports and scrips are separated
3. Generated and imported figures has its own place
4. Numbered using 2 digits
5. Reusable and easily understandable

Folder Structure

```
library(fs)
folder_names <- c("raw_data", "output_data", "rmd", "docs",
                  "scripts", "figures")

dir_create(fldr_names)
```

Paths

```
library(tidyverse)

# data import
data <- read_csv("/Users/Emil/Research/Health/amazing_data.csv")
```

Paths

```
library(tidyverse)

# data import
data <- read_csv("/Users/Emil/Research/Health/amazing_data.csv")
```

```
## Error: '/Users/Emil/Research/Health/amazing_data.csv' does not exist.
```

Only use relative paths, never absolute paths

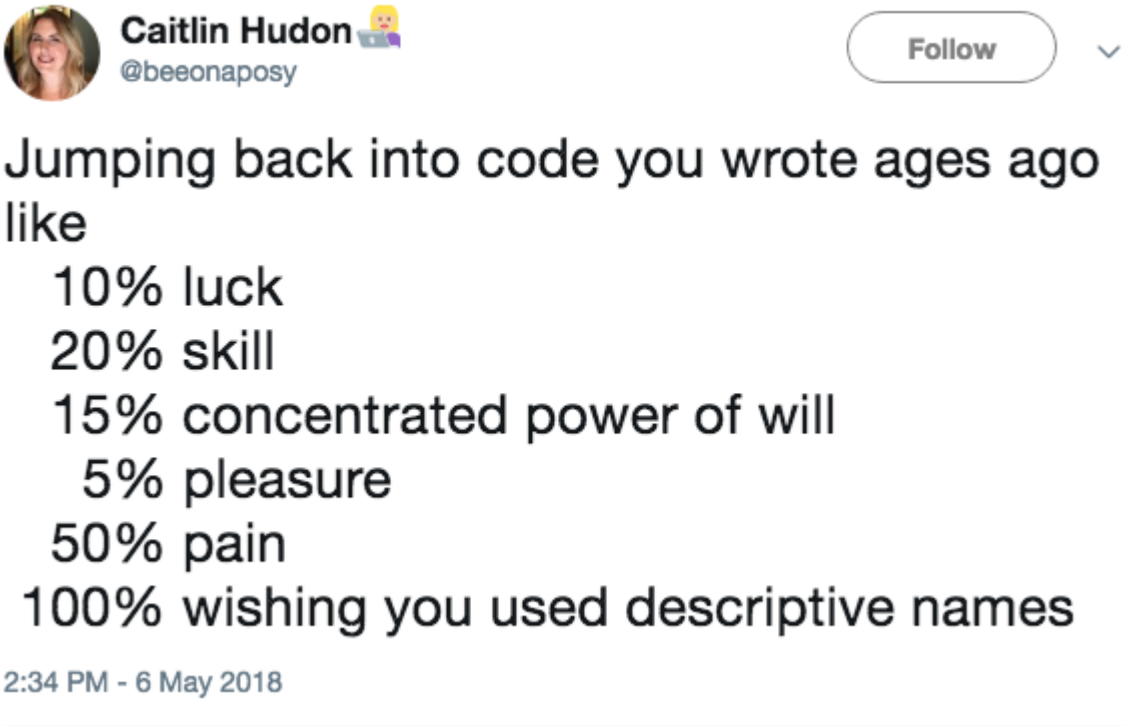
Introducing the here package.

```
library(here)
here()
```

```
## [1] "/Users/Emil/Research/Health"
```

```
library(here)
data <- read_csv(here("raw_data", "amazing_data.csv"))
```

Naming Things



Naming Things - Files

NO

```
report.pdf
reportv2.pdf
reportthisisthelastone.pages
Figure 2.png
3465-234szx.r
foo.R
```

YES

```
2018-10-01_01_report-for-cdc.pdf
01_data.rmd
01_data.pdf
02_data-filtering.rmd
02_data-filtering.pdf
```

1. Avoid spaces, punctuation, special characters and case sensitivity
2. Deliberate use of delimiters
3. Describe the contents of the file
4. Put something numeric first
5. Left pad numbers with zeroes
6. Use a standard date (YYYY-MM-DD)

Naming Things - Files

```
library(fs)
dir_ls("data/", regexp = "health-study")
```

```
## 2018-02-23_health-study_power-100_group-A1.csv
## 2018-02-23_health-study_power-100_group-B1.csv
## 2018-02-23_health-study_power-100_group-C1.csv
## 2018-02-23_health-study_power-200_group-A1.csv
## 2018-02-23_health-study_power-200_group-B1.csv
## 2018-02-23_health-study_power-200_group-C1.csv
```

```
stringr::str_split_fixed(x, "[_\\.]", 5)
```

```
##      [,1]         [,2]           [,3]        [,4]       [,5]
## [1,] "2018-02-23" "health-study" "power-100" "group-A1" "csv"
## [2,] "2018-02-23" "health-study" "power-100" "group-B1" "csv"
## [3,] "2018-02-23" "health-study" "power-100" "group-C1" "csv"
## [4,] "2018-02-23" "health-study" "power-200" "group-A1" "csv"
## [5,] "2018-02-23" "health-study" "power-200" "group-B1" "csv"
## [6,] "2018-02-23" "health-study" "power-200" "group-C1" "csv"
```

Naming Things - Files

```
library(tidyverse)
map_df(dir_ls("data/", regexp = "health-study"), read_csv)

# or

dir_ls("data/", regexp = "health-study") %>%
  map_df(read_csv)
```

Naming Things - Objects

1. Only use lowercase letters, numbers, and _
2. Use names that are not jargony, weight instead of K
3. Use informative names

Naming Things - Objects

```
# Bad
df
e
tuningVar

# Good
health_data
error
tuning_var
```

What To Avoid - attach()

Never use attach()

```
attach(mtcars)
mean(mpg)
```

```
## [1] 20.09062
```

Loads lots of names into the search path, ambiguous selections.

Try with() or withr instead

What To Avoid - attach()

Never use rm(list=ls())

Instead, restart the R session

CTRL+SHIFT+F10 **for Windows**

CMD+SHIFT+ALT+F10 **for Mac OS**

R Markdown documents versus R scripts

You can use R scripts for simple self contained tasks.

source() R scripts into your R Markdown document where you will do analyses, visualizations and reporting.

R Markdown

```
- 01-import.R
- 02-clean-names.R
- 03-tidy.R
- etc
```

Include at the start of R Markdown file

```
{r load_scripts, include = FALSE}
library(here)
source(here("scripts", "01-import.R"))
source(here("scripts", "02-clean-names.R"))
source(here("scripts", "03-tidy.R"))
```

Naming Chunks

Names can be placed after the comma

```
```{r, chunk-label, results='hide', fig.height=4}
```

or before

```
```{r chunk-label, results='hide', fig.height=4}
```

> In general it is recommended to use alphabetic characters with words separated by - and avoid other characters. - Yihui Xie

1. Makes navigating the R Markdown document easier
2. Makes your R Markdown easier to understand
3. Clarifies error reports or progress of knitting
4. Caching when moving chunks around

Setup Chunk

In a fresh R Markdown document you see this

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

The setup chunk is run before another code - use to your advantage
```

Setting figure path

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(fig.path = "figures/")
````
````

Styling Code

Use consistent style when writing code

http://style.tidyverse.org/

All about preferences but keep it consistent!!!

Use the styler package to style your code for you

Keep .Rprofile Clean

Your computer contains a file called .Rprofile.

This file runs first in every session. Think of it as configuration file.

```
options(stringsAsFactors = FALSE)
options(max.print = 100)
```

Keep .Rprofile Clean

Only put interactive code in

Yes

```
# add this with usethis::use_usethis()
library(usethis)
```

No

```
library(tidyverse)
```

Comment Your Code

Functions: Arguments and purpose

Code: What or why, NOT how

```
# Takes a data.frame (data) and replaces the columns with the names
# (names) and converts them from factor variable to character
# variables. Keeps characters variables unchanged.
factor_to_text <- function(data, names) {
  for (i in seq_along(names)) {
    if(is.factor(data[, names[i], drop = TRUE]))
      data[, names[i]] <- as.character.factor(data[, names[i],
                                                   drop = TRUE])
  }
  data
}
```