Practical R: Introductions

Abhijit Dasgupta

Fall, 2019

What does R look like?

- A scripting language
  - Provide instructions to the computer
  - in a structured manner
  - to do statistical analysis

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

What does R look like?

- Start with a data set

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

What does R look like?

- Start with a data set
- Create new variables from old variables

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

What does R look like?

- Start with a data set
- Create new variables from old variables
- Deal with missing values

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

What does R look like?

- Start with a data set
- Create new variables from old variables
- Deal with missing values
- Find the frequencies

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

What does R look like?

- Start with a data set
- Create new variables from old variables
- Deal with missing values
- Find the frequencies
- Comment on what you're doing

```
# Determining frequencies of breast cancer subtypes

type_frequencies <-
  breast_cancer %>%
    mutate(luminalA = ifelse(ER == '+' & PR == '+' & HER2 == '-', 1, 0),
         luminalB = ifelse(ER == '+' & PR == '-' & HER2 == '+', 1, 0),
         her2 = ifelse(ER == '-' & PR == '-' & HER2 == '+', 1, 0),
         basal = ifelse(ER == '-' & PR == '-' & HER2 == '-', 1, 0)) %>%
    mutate(type = case_when(
      luminalA == 1 ~ "Luminal A",
      luminalB == 1 ~ "Luminal B",
      her2 == 1 ~ "HER2",
      basal == 1 ~ "Basal",
      TRUE ~ NA)) %>%
    count(type)
```

This is an example of a pipeline in R. We'll develop different aspects of this progressively throughout the semester

Why use a scripting language for analysis?

Pros:

1. Have to think
2. Reproducible (custom) workflows
3. Much less error-prone
4. Much lower costs to repeat analyses, or as you learn more
5. Easily leverage work of smarter developers
6. Easier to work with larger datasets (more than size of screen)

Cons:

1. Have to type
2. Have to know the language
3. Higher initial startup cost
4. Have to think

Why use a scripting language for analysis?

Pros:

1. Have to think
2. Reproducible (custom) workflows
3. Much less error-prone
4. Much lower costs to repeat analyses, or as you learn more
5. Easily leverage work of smarter developers
6. Easier to work with larger datasets (more than size of screen)

- You're giving instructions to a fast but stupid machine
- This machine will do **exactly** what you tell it
- The machine is capable of amazing things
- Can't just *menu-mine* and try things that **seem** to be what you want

With great power comes great responsibility

But also great benefits

Why use a scripting language for analysis?

Pros:

1. Have to think
2. Reproducible (custom) workflows
3. Much less error-prone
4. Much lower costs to repeat analyses, or as you learn more
5. Easily leverage work of smarter developers
6. Easier to work with larger datasets (more than size of screen)

- If your code is not right, it won't run
  - Can be frustrating
  - But if it runs you're much more confident
- If you screw up in Excel
  - almost impossible to recover
- You have much more control over what you're doing

Why use a scripting language for analysis?

Pros:

1. Have to think
2. Reproducible (custom) workflows
3. Much less error-prone
4. Much lower costs to repeat analyses, or as you learn more
5. Easily leverage work of smarter developers
6. Easier to work with larger datasets (more than size of screen)

- Can use *modules* or *packages* developed by others
  - `tidyverse`, `Seurat`, `ggplot2`
- Can "steal" code from others (provided license allows)

Why use a scripting language for analysis?

Pros:

1. Have to think
2. Reproducible (custom) workflows
3. Much less error-prone
4. Much lower costs to repeat analyses, or as you learn more
5. Easily leverage work of smarter developers
6. Easier to work with larger datasets (more than size of screen)

Good luck working with

- GWAS data
- fMRI data
- Stocks and bonds data
- Sports data
- Many more ...

in a unified environment

Learn once, use everywhere

Why use R?

- Specializes in statistics and data visualization
- Flexible
  - ~~*If you can do it*~~ **How you can do it**
- Large ecosystem
  - Over 16,000 packages, 1500+ dedicated to bioinformatics
  - Can read from most sources of data
  - Generic and specialized analyses
- Fantastic community
  - Twitter, StackOverflow, blogosphere, conferences, online books

Why use R?

R is a very high-quality product that is accepted and used widely in government agencies, corporations and universities worldwide

- Standard data analytic software in bioinformatics, behavioral health and many aspects of quantitative finance
- Increasingly used in pharma, economics, political science and engineering

R is open-source, in that anyone can see the actual code and validate the computations directly

Why use R?

- R has a well-deserved reputation for being a great data visualization tool, with users being able to create complex, customizable graphs with relative ease
- As a scripting language, it allows the same workflows to be coded and re-used.
- You can set up workflows to validate data, in terms of data quality and missingness, which avoids visual inspection which can be time-consuming and mistake-prone.
- R can handle large datasets, and can work with multiple datasets at the same time

Why use R?

Specialized packages available for many domains

- Bioinformatics
- Econometrics
- Maps and spatial analytics
- Text mining and Natural Language Processing

The CRAN Task Views provide curated lists of packages based on different domains

The Bioconductor Project is THE source for bioinformatics packages in R. It is the gold standard for many bioinformatic workflows

Things  is not (in this class)

- The 18th letter of the English alphabet
- A magic incantation that will produce an analysis
- Just something the cool kids are doing
- Point-and-click, automatic, WYSIWYG (What you see is what you get)
  - So it's not Excel, SPSS, Prism, GraphPad
  - It's much more!!!

A note on Excel 

Excel is omnipresent, so it becomes the default data medium

It is great for many things, including *quick-and-dirty* analyses

It can be error-prone

It needs to be backed up

It has size limitations

Takes a lot of effort to do complex analyses

- No way to reproduce analyses without macros
- No way to document what you are doing
- Excel has some nasty default behavior
    - Guess what the `MAR1` gene gets recorded as?
- Very hard to recover from errors
    - Shift of one error (off by one row or column)
    - Google *Duke Potti* or *Reinhart Rogoff Herndon*

About this class

Learning Objectives

- Run R and RStudio, making use of inherent R features
- Find and make use of the extensive packages (R add-ons) available for analyzing biological and other forms of data
- Load, manipulate, and combine data to make it amenable to further analyses
- Visualize data with extensive graphics capabilities of R (including ggplot)
- Use R to run statistical models and hypothesis tests and report results conforming to standards expected in scientific journals
- Write reports using the powerful `rmarkdown` package and its derivatives

Plan

Date Topic

September 11, 2019 Introduction to R, RStudio and RMarkdown
September 18, 2019 Data Structures in R (classes 5:30-7, 7-8:30)
September 25, 2019 R packages, data import/export
October 02, 2019 Towards analytic data: Data Munging
October 09, 2019 Data exploration through visualization
October 16, 2019 Statistical analyses: Table 1, estimation and confidence intervals, and more ggplot
October 23, 2019 Statistical analyses: Classical hypothesis testing and computational inference
October 30, 2019 Statistical learning: Regression models
November 06, 2019 Statistical learning: Cluster analysis and pattern recognition (classes 5:30-7, 7-8:30)
November 13, 2019 More data munging with `purrr`: grouping, mapping and functional programming
November 20, 2019 Basic bioinformatics: Bioconductor and friends
November 27, 2019 No class (Thanksgiving)
December 04, 2019 More complex visualizations and RMarkdown
December 11, 2019 Project presentations

Grading rubric

1. Homeworks, available Friday after class, due by 11:59PM the following Tuesday. (50%)
    - No late homeworks, since solutions will be available Wednesday mornings
    - I'll score the top 10 homeworks for grade
2. Final project: A RMarkdown report/presentation demonstrating an end-to-end data analysis in R using your own data, from data ingestion to munging to analyses and graphics, with a brief introduction and conclusion (20%)
3. Class participation (20%)
4. Completion and submission of class exercises (10%, marked for completion)
    - These will need to be in basic RMarkdown, showing the problem and the solution. You can add a section for questions here that I can address in the following class or online. These will have to be submitted before you leave the classroom

Submitting assignments

Homework

- Each homework will come with a submission link.
    - This link will automatically expire at deadline
- Name your RMarkdown file _HW<#>.Rmd
    - For example *FelixLeiter_HW3.Rmd*
- The RMarkdown file should be self contained, i.e. I should be able to run it

Must contain "BIOF339" and your name in the subject line

Needs to be a attached RMarkdown file that I can run

Communication

- Primarily via Slack.
  - Please join the BIOF339 Slack channel using this link.
  - You will see two channels named #*wed5-7_2019* and #*wed7-9_2019*. Please join the channel corresponding to your section.
- Slack for broadcasting messages, answering questions and the like.
  - If you have a question, you can directly message me on Slack. Expect an answer within 24 hours.
- Office hours by appointment

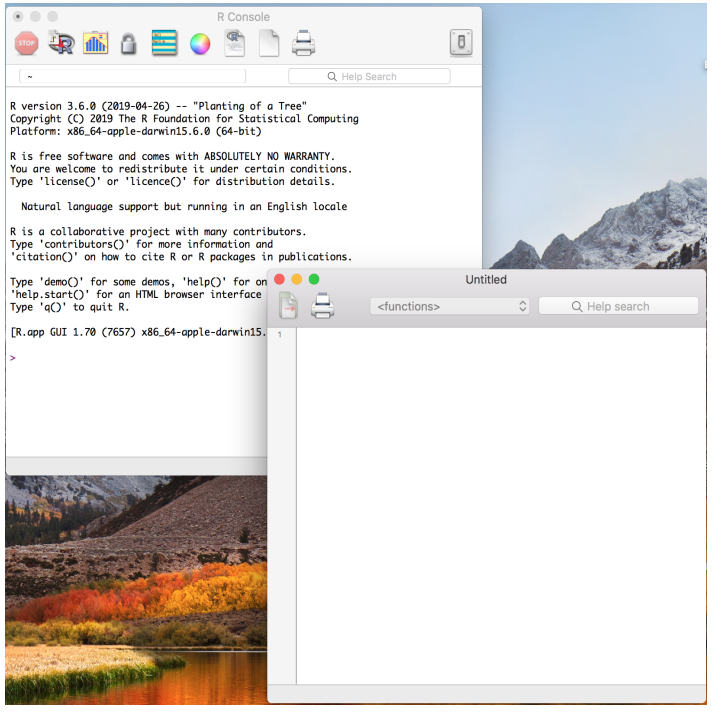
Class project

- Create a RMarkdown document or presentation
- Use your own data, or data available on the web (legally)
- Show me that you can
    - import data into R
    - manipulate (munge) the data
    - perform some analysis on the data
    - create a visualization
    - create a report in RMarkdown
- 5 minute *lightning talks* on final class day


R
RStudio
www.rstudio.com

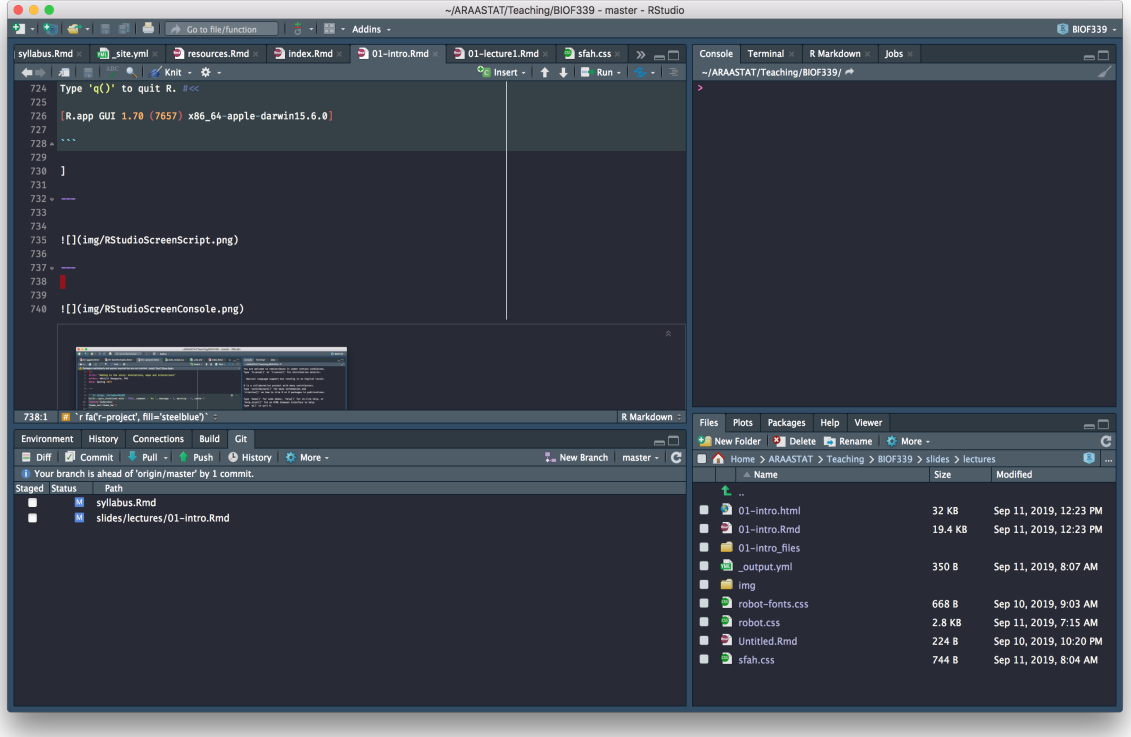
R is the scripting language we will use for statistical analysis

It comes with a very basic GUI and is not very user-friendly as is



RStudio is a much nicer Integrated Development Environment (IDE) that runs R

- You can customize the look and feel of RStudio
- We'll go through the features of RStudio in a minute



- The version of R you're running

When you start R, this is what shows up

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.

  Natural language support but running in an English locale

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.

[R.app GUI 1.70 (7657) x86_64-apple-darwin15.6.0]
```

- The version of R you're running
- The operating system you're running

When you start R, this is what shows up

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.

  Natural language support but running in an English locale

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.

[R.app GUI 1.70 (7657) x86_64-apple-darwin15.6.0]
```

- The version of R you're running
- The operating system you're running
- It's free, with no guarantees, but is known to be solid and well-maintained

When you start R, this is what shows up

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.

  Natural language support but running in an English locale

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.

[R.app GUI 1.70 (7657) x86_64-apple-darwin15.6.0]
```

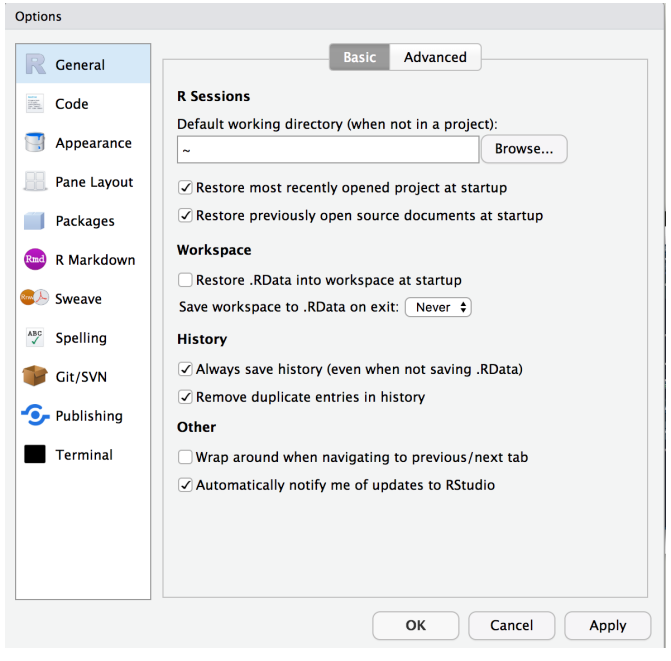
- The version of R you're running
- The operating system you're running
- It's free, with no guarantees, but is known to be solid and well-maintained
- You can find help online

When you start R, this is what shows up

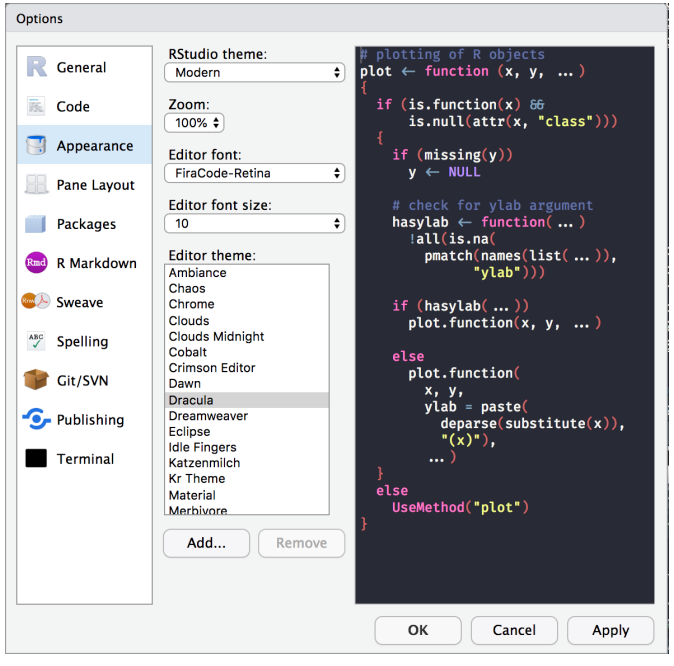
```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

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.

  Natural language support but running in an English locale

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.

[R.app GUI 1.70 (7657) x86_64-apple-darwin15.6.0]
```

The exact look won't match yours. See demo

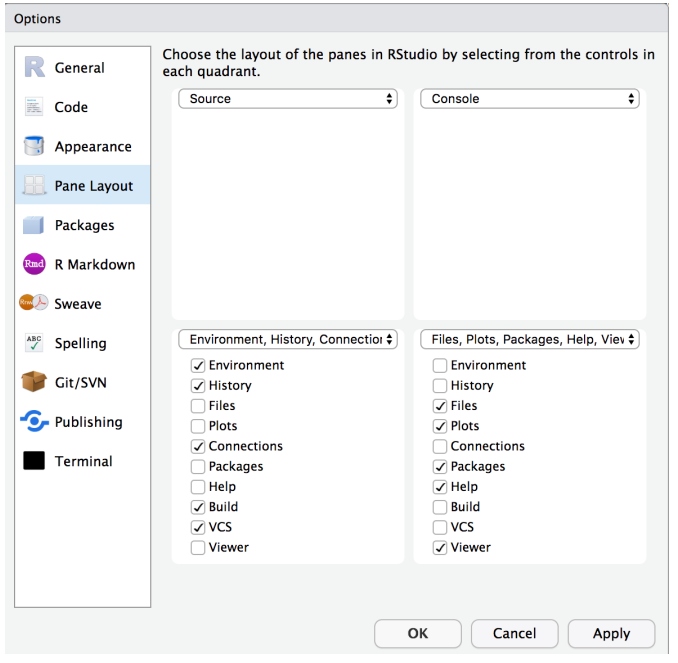
Options



General options

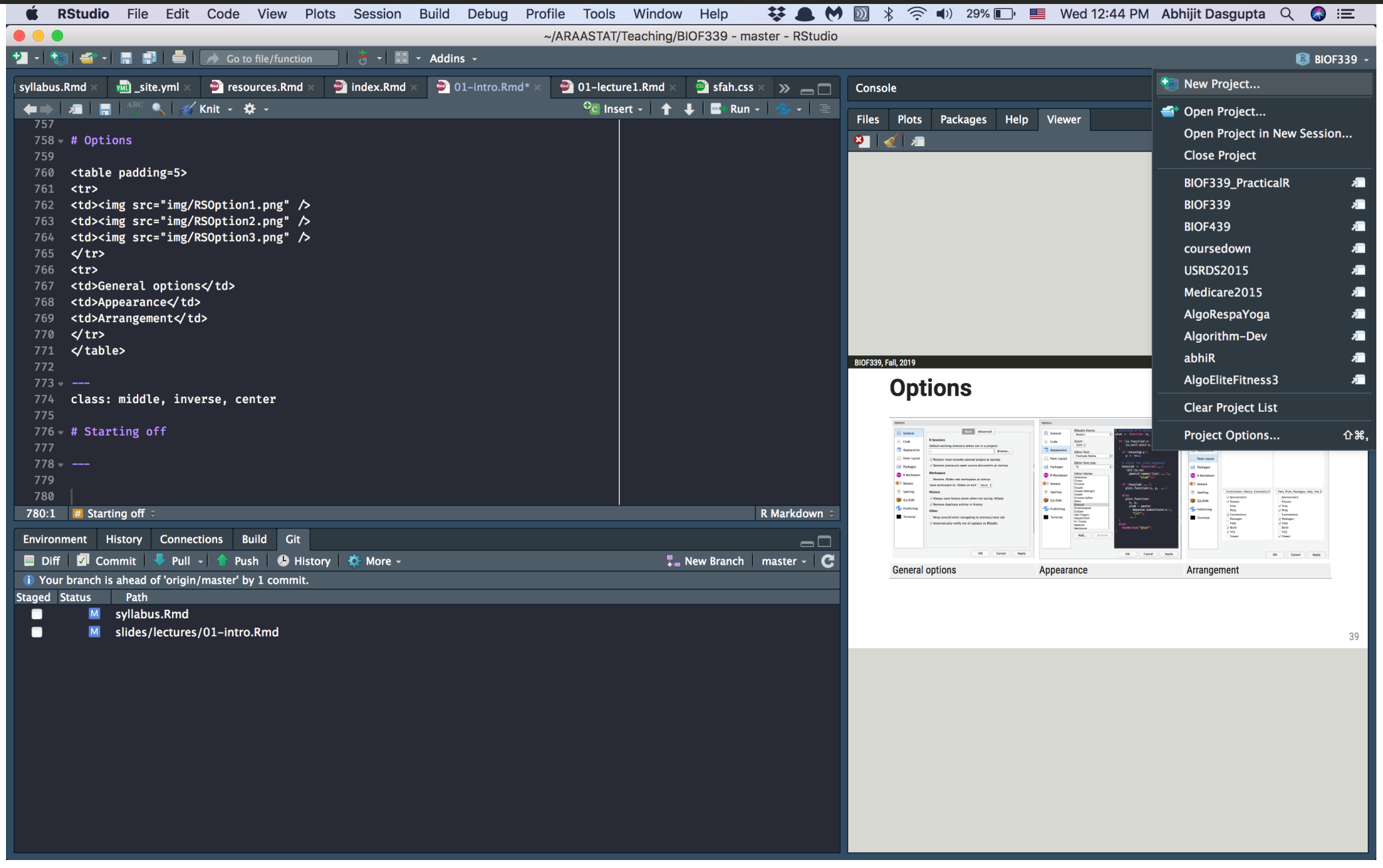


Appearance



Arrangement

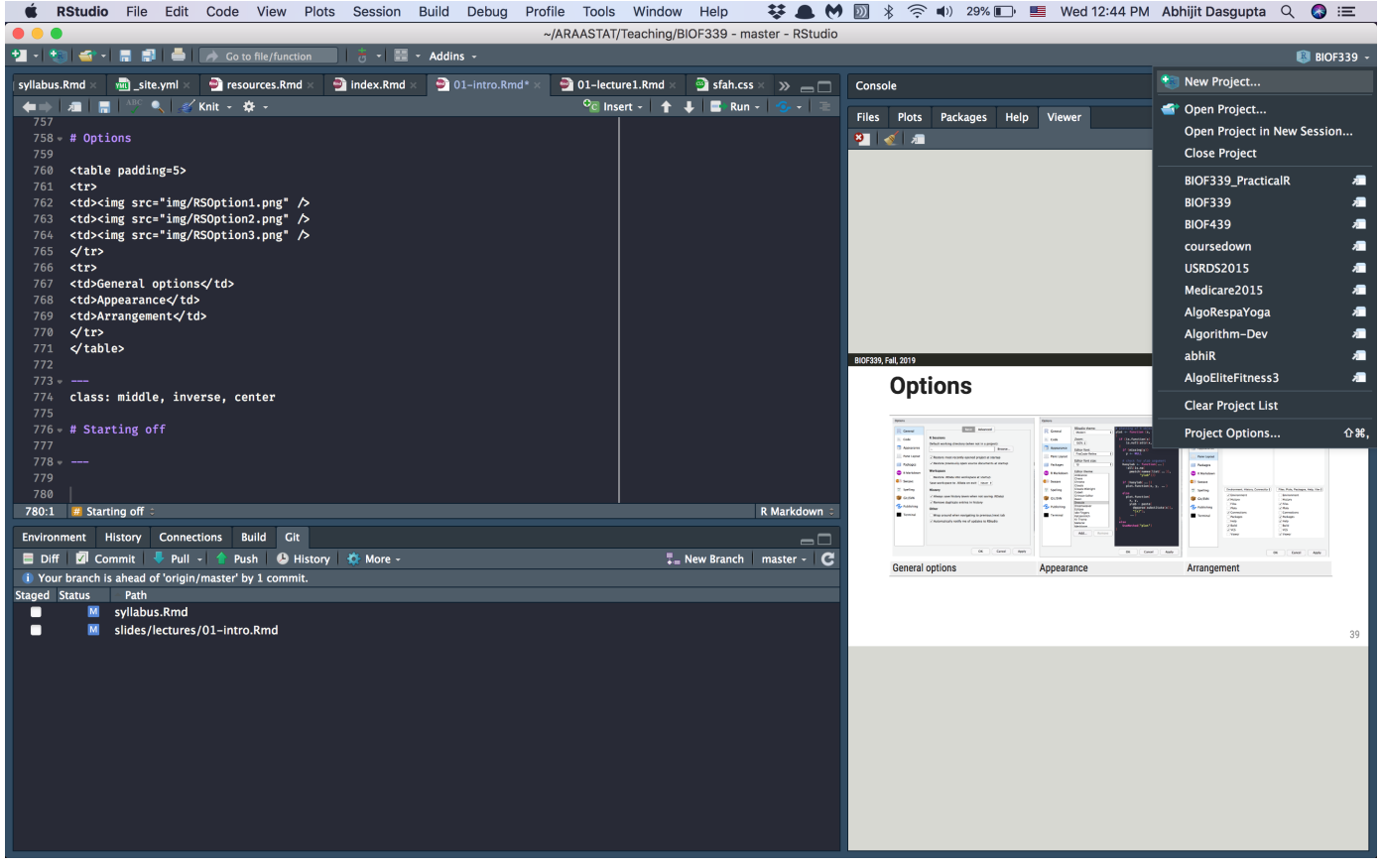
Starting off


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Wed 12:44 PM Abhijit Dasgupta
~/ARAASTAT/Teaching/BIOF339 - master - RStudio
Go to file/function
Addins
BIOF339
syllabus.Rmd
_site.yml
resources.Rmd
index.Rmd
01-intro.Rmd*
01-lecture1.Rmd
sfah.css
Knit
Insert
Run
757
758 # Options
759
760 <table padding=5>
761 <tr>
762 <td>
763 <td>
764 <td>
765 </tr>
766 <tr>
767 <td>General options</td>
768 <td>Appearance</td>
769 <td>Arrangement</td>
770 </tr>
771 </table>
772
773 ---
774 class: middle, inverse, center
775
776 # Starting off
777
778 ---
779
780
780:1
Starting off
R Markdown
Environment
History
Connections
Build
Git
Diff
Commit
Pull
Push
History
More
New Branch
master
Your branch is ahead of 'origin/master' by 1 commit.
Staged
Status
Path
syllabus.Rmd
slides/lectures/01-intro.Rmd
Console
Files
Plots
Packages
Help
Viewer
New Project...
Open Project...
Open Project in New Session...
Close Project
BIOF339_PracticalR
BIOF339
BIOF439
coursedown
USRDS2015
Medicare2015
AlgoRespaYoga
Algorithm-Dev
abhiR
AlgoEliteFitness3
Clear Project List
Project Options...
BIOF339, Fall, 2019
Options
General options
Appearance
Arrangement
39

Start a new project

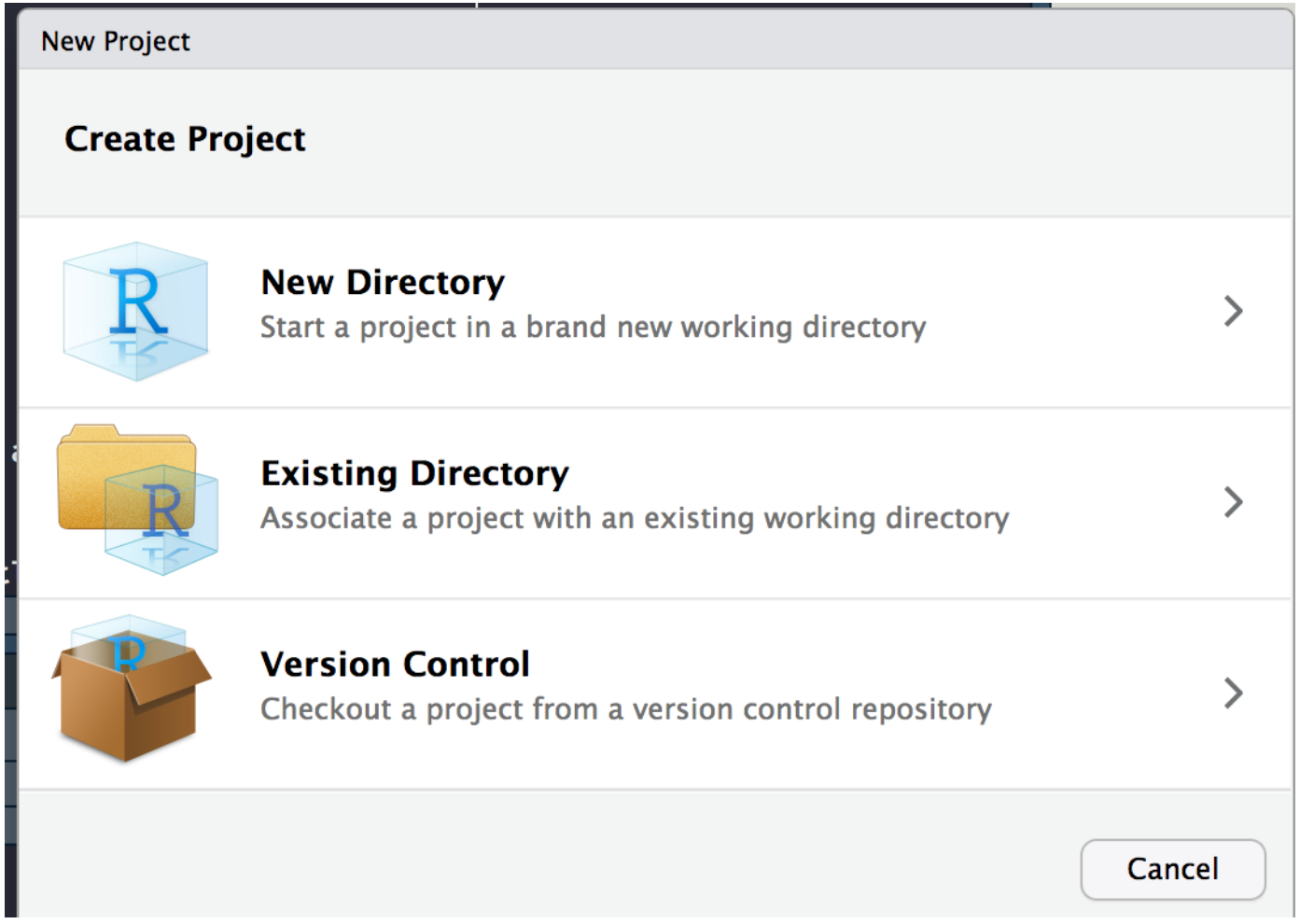
A **RStudio project** allows you to encapsulate each analysis project

- Keeps files, data, and R separate between projects
- Allows you to work on different projects concurrently



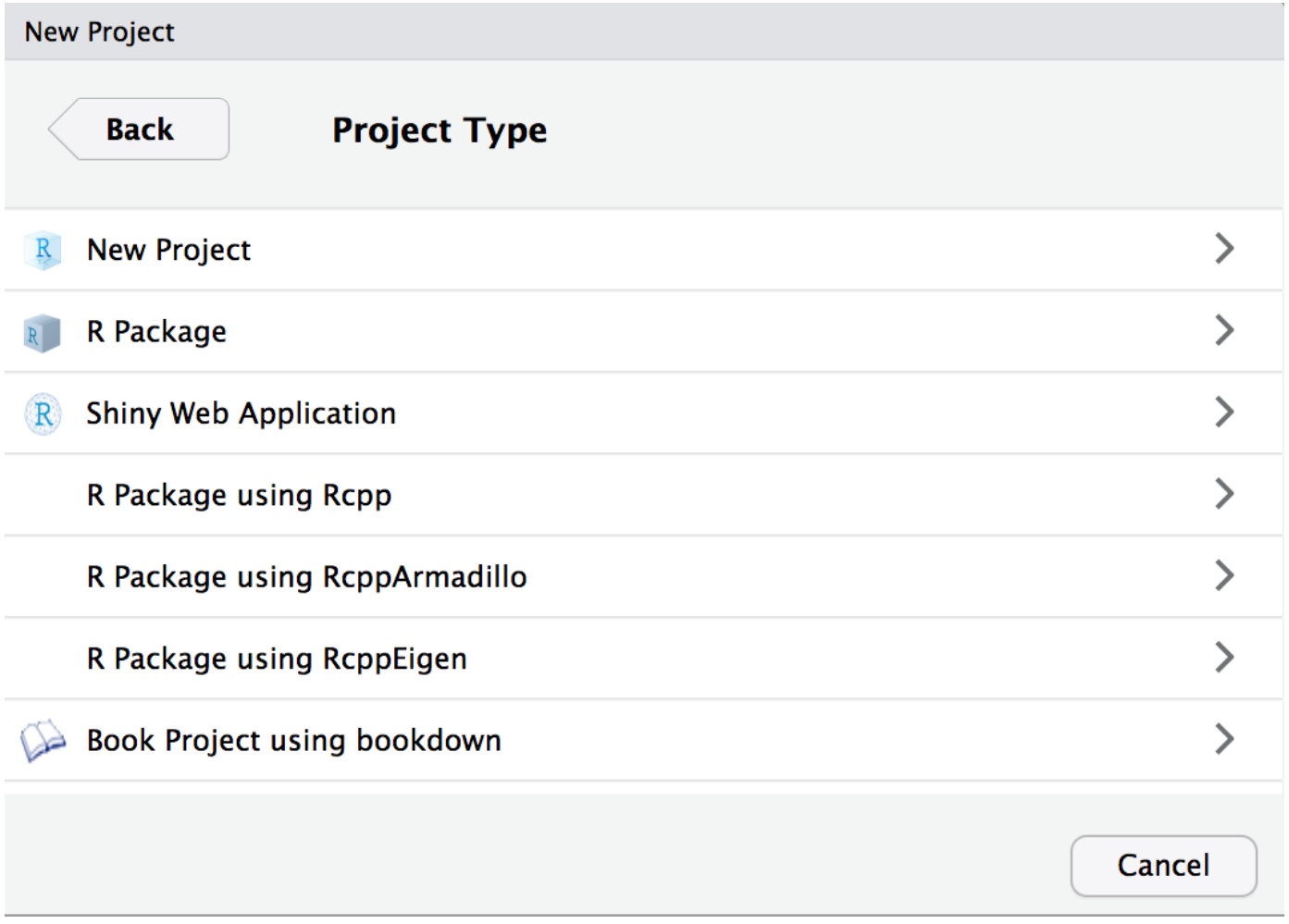
Starting a new project

- Choose a new directory



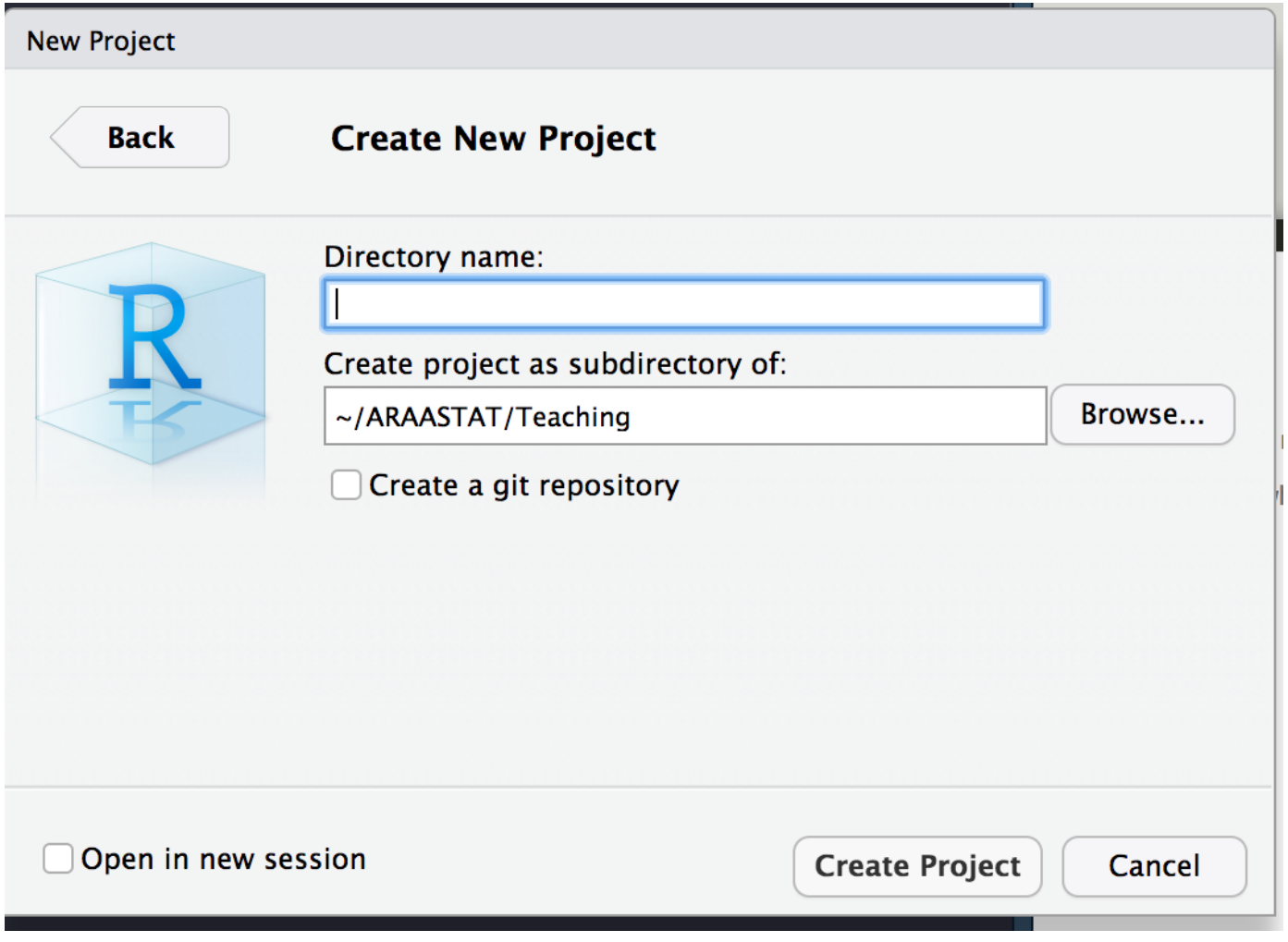
Start a new project

- Choose a new directory
- Choose what kind of project



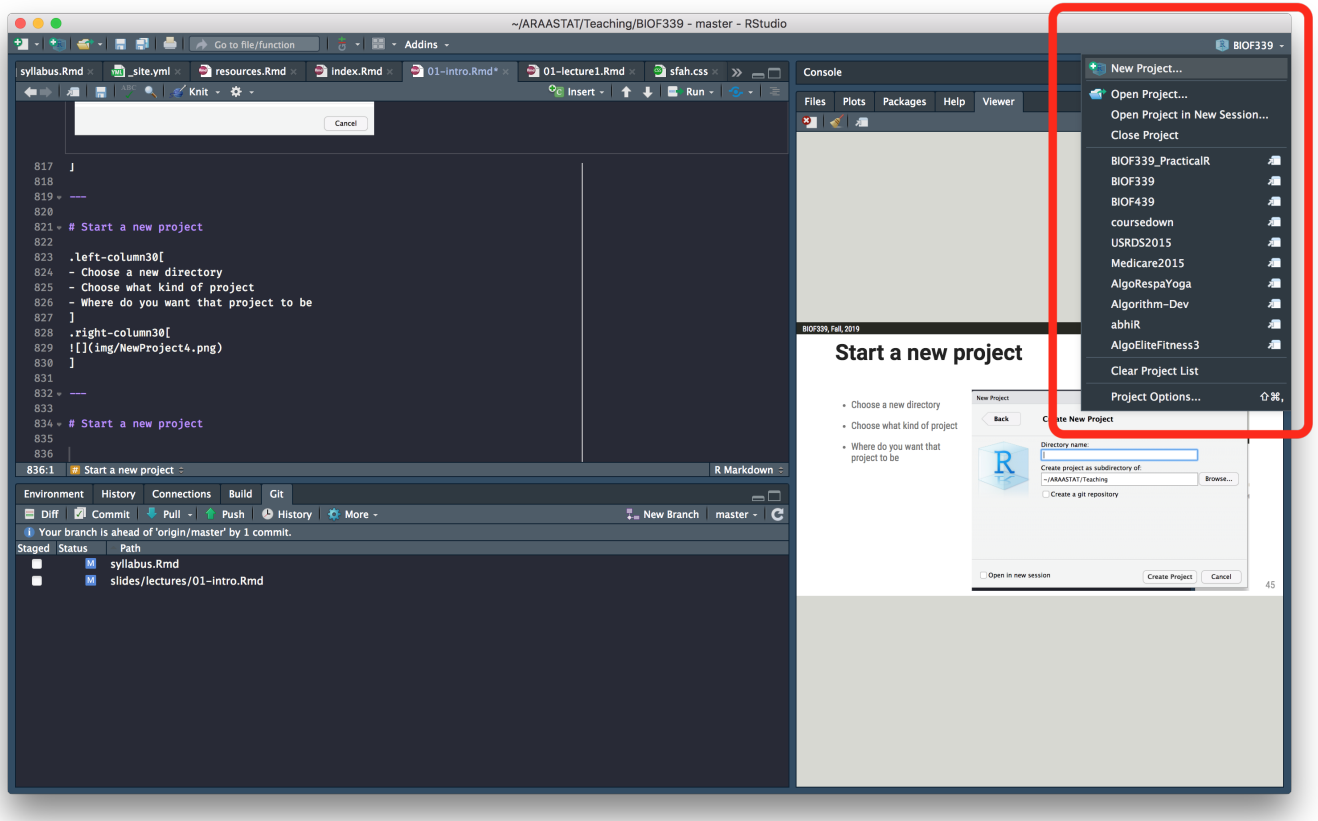
Start a new project

- Choose a new directory
- Choose what kind of project
- Where do you want that project to be



Start a new project

- Choose a new directory
- Choose what kind of project
- Where do you want that project to be
- Switch betwen projects easily

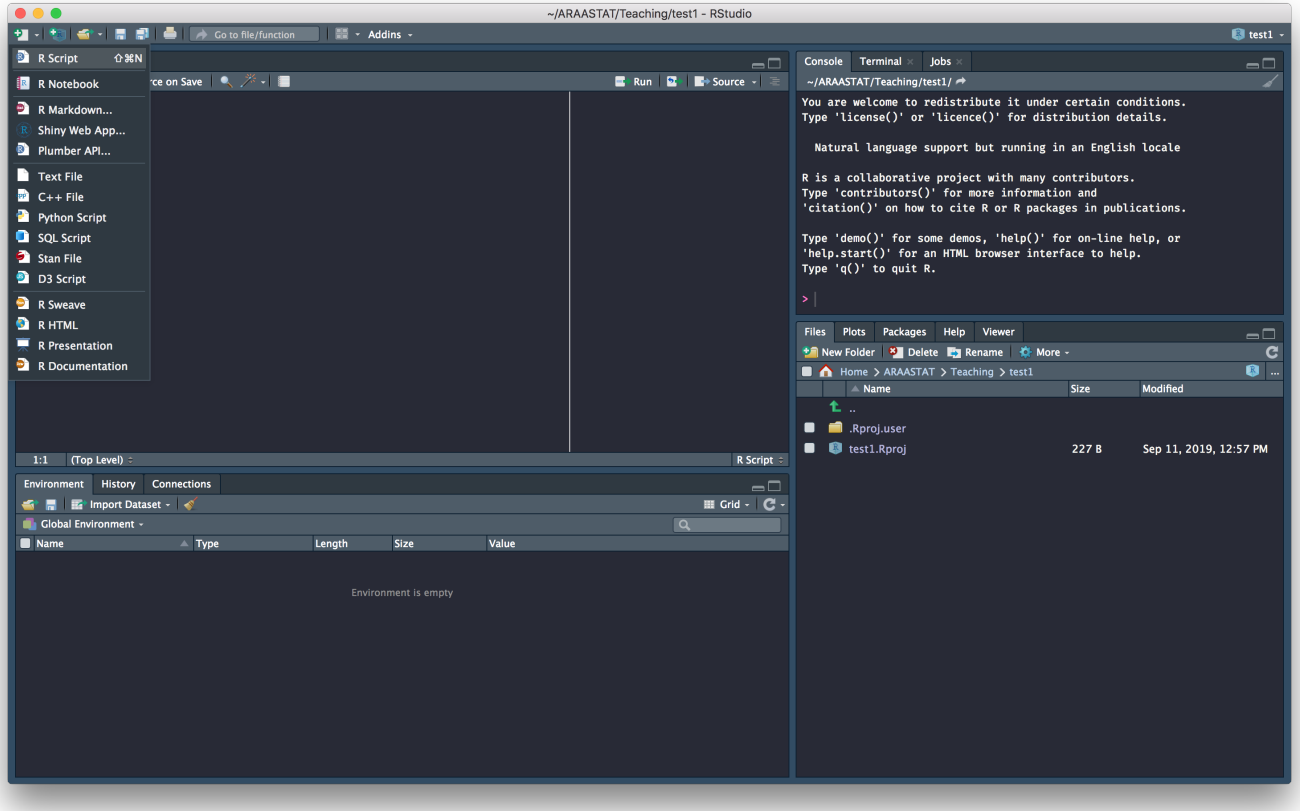


Working in a project

Open a new script

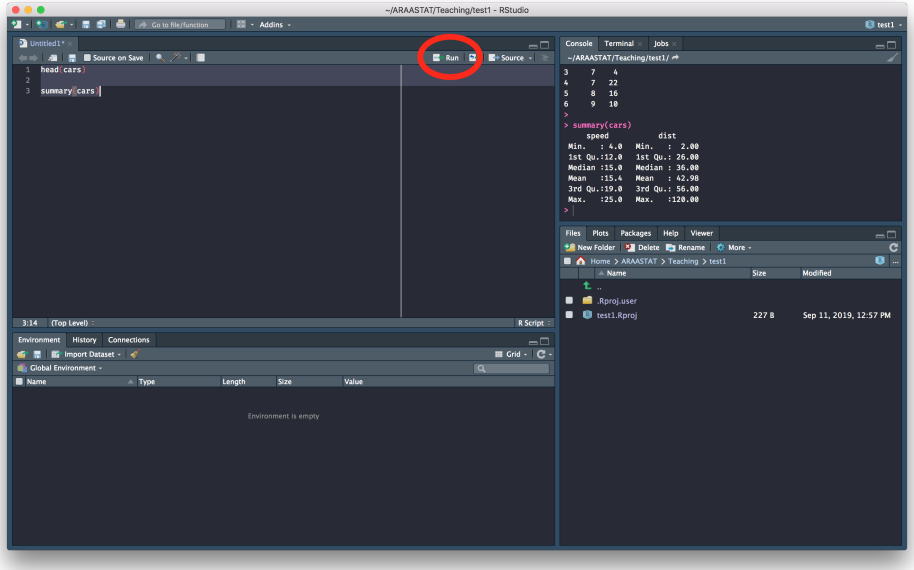
You will start by opening a new script

- This is a blank canvas
- You will write R code (instructions) in this script
- This is just a text editor, so you can copy, paste, edit just as in Notepad
- Good practice is to write code in the script window, then **send** the code to R



Run some code

Write some code, then send it to the console, where R is running



- You can use the button, but
- I prefer you learn the keyboard shortcut
- Ctrl/Cmd + Enter sends either
  - current line under cursor
  - highlighted code

Let's play for a while

A calculator

```
2 + (3 * 5) / 1.75
```

```
#>  [1] 10.57143
```

A random number generator

```
set.seed(2085)
runif(10, 0, 10)
```

```
#>   [1] 9.6383786 2.7689373 2.4291854 1.7018136 4.8812344 0.5893566 1.9033637
#>   [8] 9.7952480 2.1012500 5.6648043
```

Exploring a data set

The `airquality` dataset, built in to R, provides daily air quality measurements in New York, May to September 1973.

```
airquality
```

This automatically prints out the full data set, but that's not really useful

```
str(airquality)
```

```
#>  'data.frame':    153 obs. of  6 variables:
#>   $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
#>   $ Solar.R: int  190 118 149 313 NA NA 299 99 19
#>   $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13
#>   $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ..
#>   $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
#>   $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
head(airquality)
```

```
#>    Ozone Solar.R Wind Temp Month Day
#>  1    41     190  7.4   67     5   1
#>  2    36     118  8.0   72     5   2
#>  3    12     149 12.6   74     5   3
#>  4    18     313 11.5   62     5   4
#>  5    NA      NA 14.3   56     5   5
#>  6    28      NA 14.9   66     5   6
```

Note the missing values, denoted NA

Exploring a data set

```
summary(airquality)
```

```
#>       Ozone           Solar.R           Wind
#>   Min.   :  1.00   Min.   :  7.0   Min.   : 1.700
#>   1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400
#>   Median : 31.50   Median :205.0   Median : 9.700
#>   Mean   : 42.13   Mean   :185.9   Mean   : 9.958
#>   3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500
#>   Max.   :168.00   Max.   :334.0   Max.   :20.700
#>   NA's   :37       NA's   :7
#>       Month            Day
#>   Min.   :5.000   Min.   : 1.0
#>   1st Qu.:6.000   1st Qu.: 8.0
#>   Median :7.000   Median :16.0
#>   Mean   :6.993   Mean   :15.8
#>   3rd Qu.:8.000   3rd Qu.:23.0
#>   Max.   :9.000   Max.   :31.0
#>
```

```
library(tableone) # Use a package
kableone(CreateTableOne(data=airquality),
         format='html')
```

	Overall
n	153
Ozone (mean (SD))	42.13 (32.99)
Solar.R (mean (SD))	185.93 (90.06)
Wind (mean (SD))	9.96 (3.52)
Temp (mean (SD))	77.88 (9.47)
Month (mean (SD))	6.99 (1.42)
Day (mean (SD))	15.80 (8.86)

We'll talk more about this in Week 3 & 6

> Sorry!!! You probably don't have the `tableone` package 😿

Sidebar: Installing packages

```
install.packages('tableone')
```

You should do this directly in the console rather than in the script. Why? Because you only have to do this once (per computer). You don't need to re-install packages every time.

> Some packages I might use might not install this way, since they are developmental and not released. We'll talk about how to install those later. For you Windows folk, it will take a little bit of work

Exploring one variable

```
mean(airquality$Temp)
```

```
#>  [1] 77.88235
```

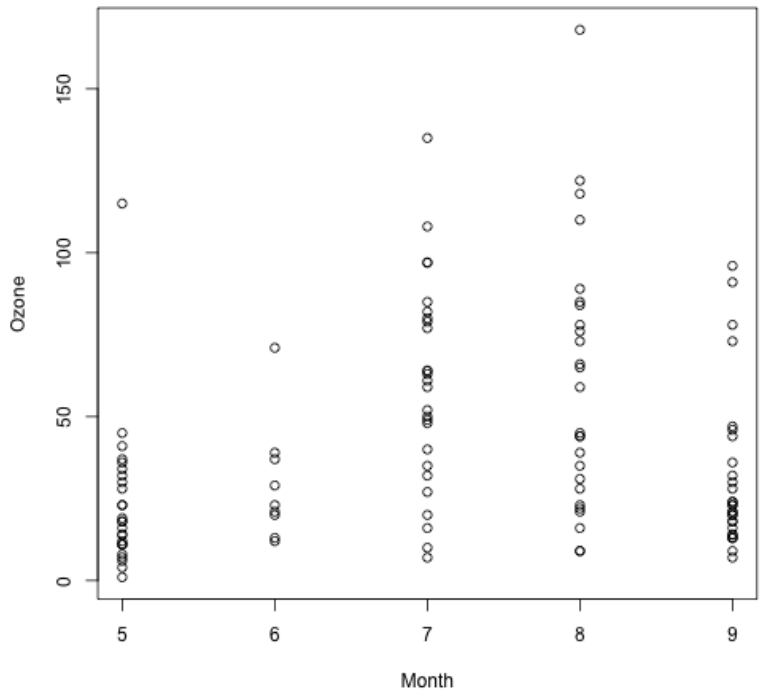
```
median(airquality$Temp)
```

```
#>  [1] 79
```

```
hist(airquality$Temp)
```

Exploring two variables

```
plot(Ozone ~ Month, data = airquality)
```




```
library(ggplot2)
ggplot(airquality) +
  geom_point(aes(x = Month, y = Ozone)) +
  geom_smooth(aes(x = Month, y = Ozone), color = 'gre
  theme_bw()
```

Is there a difference in ozone between months

```
summary(aov(Ozone ~ Month, data=airquality))
```

```
#>              Df Sum Sq Mean Sq F value Pr(>F)
#>  Month        1   3387    3387   3.171 0.0776 .
#>  Residuals  114 121756    1068
#>  ---
#>  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
#>  37 observations deleted due to missingness
```

Whoops!! Only 1 df but 5 months!!

This is doing ANCOVA/Regression

```
summary(aov(Ozone~factor(Month), data=airquality))
```

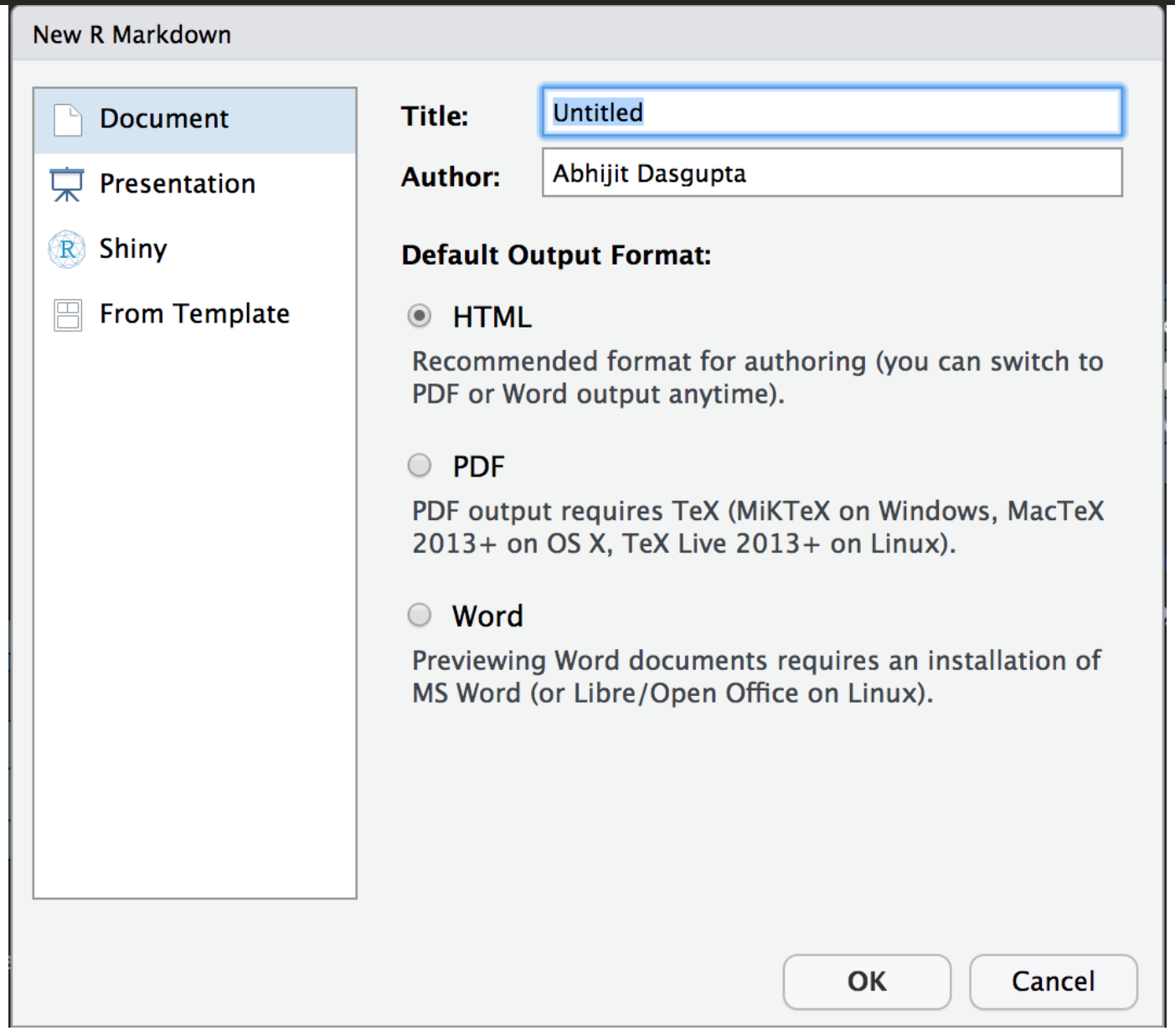
```
#>                Df Sum Sq Mean Sq F value   Pr(>F)
#>  factor(Month)  4  29438    7359   8.536 4.83e-06
#>  Residuals    111  95705     862
#>  ---
#>  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
#>  37 observations deleted due to missingness
```

`factor` can convert a numerical or character variable into a categorical variable

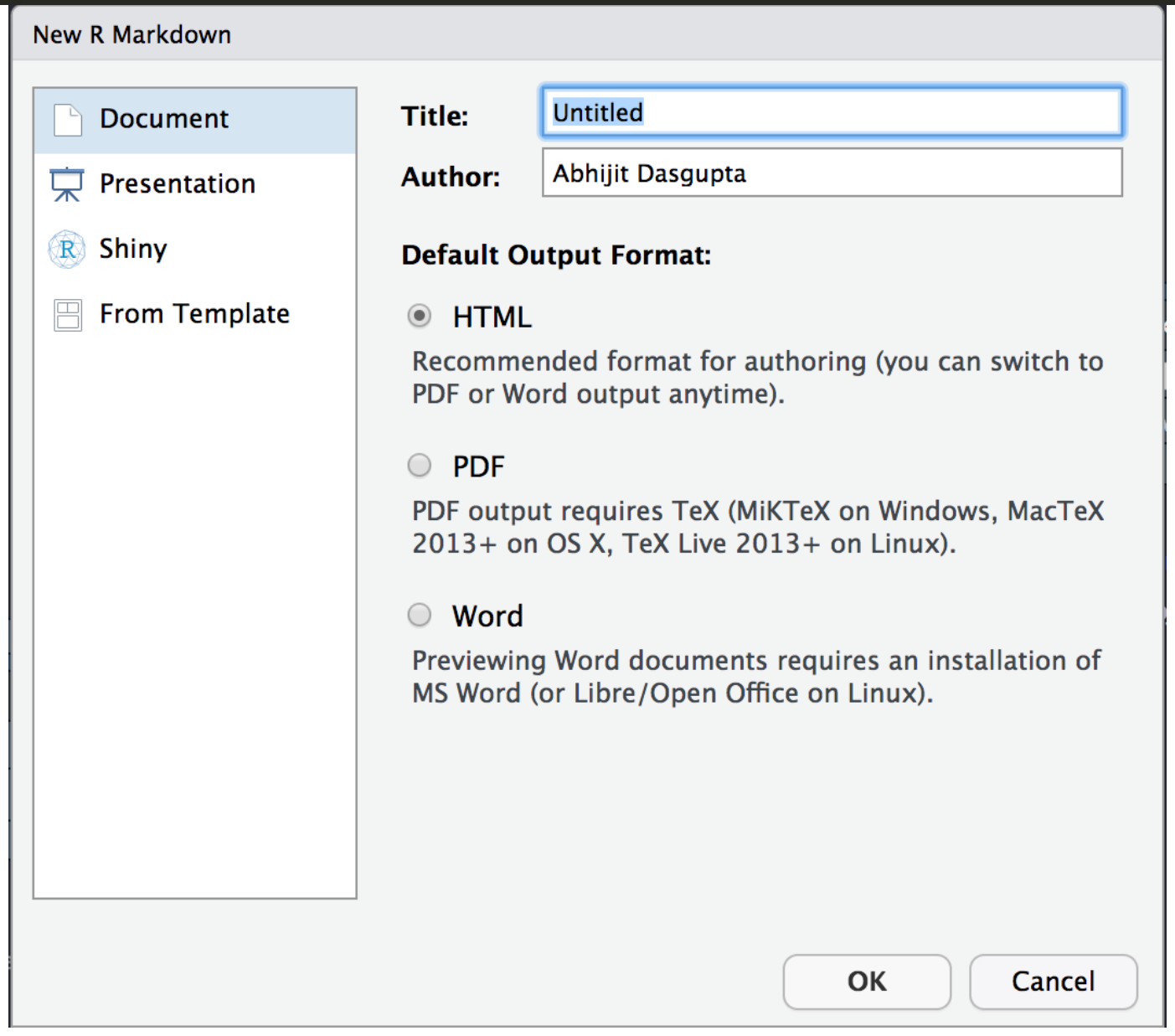
> I'm doing a lot of things on the fly here. We'll work through these at a saner pace in the coming weeks. Don't panic!! Breathe!!!

RMarkdown

- There are some choices you might need to make
  - Document is like a paper
  - Presentation is like PowerPoint
  - Shiny is an interactive web app developed in R. May talk about it towards the end
  - Various packages also provide templates for RMarkdown documents



- Elements on the right are output formats
    - Documents produce Word, PDF or HTML
    - Presentations produce PowerPoint, PDF, or HTML



```
---
title: "Untitled"
author: "Abhijit Dasgupta"
date: "9/11/2019"
output: html_document
---
```

This determines the title and author, and the output type. Typically don't modify the entry in `output` for now

````
```{r cars}
summary(cars)
```
````

This is a code chunk. RMarkdown evaluates this chunk of code and replaces the code with the results. This code chunk happens to be named "cars". The naming is optional but useful.

RMarkdown

````
```{r cars}
summary(cars)
```
````

Results

```
summary(cars)
```

```
#>       speed           dist
#>   Min.   : 4.0   Min.   :  2.00
#>   1st Qu.:12.0   1st Qu.: 26.00
#>   Median :15.0   Median : 36.00
#>   Mean   :15.4   Mean   : 42.98
#>   3rd Qu.:19.0   3rd Qu.: 56.00
#>   Max.   :25.0   Max.   :120.00
```

RMarkdown

````
```{r}
library(tableone) # Use a package
kableone(CreateTableOne(data=airquality),
         format='html')
```
````

Results

```r
library(tableone) # Use a package
kableone(CreateTableOne(data=airquality),
         format='html')
```

	Overall
n	153
Ozone (mean (SD))	42.13 (32.99)
Solar.R (mean (SD))	185.93 (90.06)
Wind (mean (SD))	9.96 (3.52)
Temp (mean (SD))	77.88 (9.47)
Month (mean (SD))	6.99 (1.42)
Day (mean (SD))	15.80 (8.86)

The code chunk on the left gets **replaced** by the table on the right in your document

Inline code evaluation

RMarkdown

The airquality data set has
`` `r nrow(airquality)` `` observations

The average ozone level is `` `r mean(airquality$Ozone)` `` parts per billion

Results

The airquality data set has 153 observations

The average ozone level is
42.1293103 parts per billion

Practice time