

R NOOBZ: CLASS 1

INTRO TO R

YouthFirst Lab

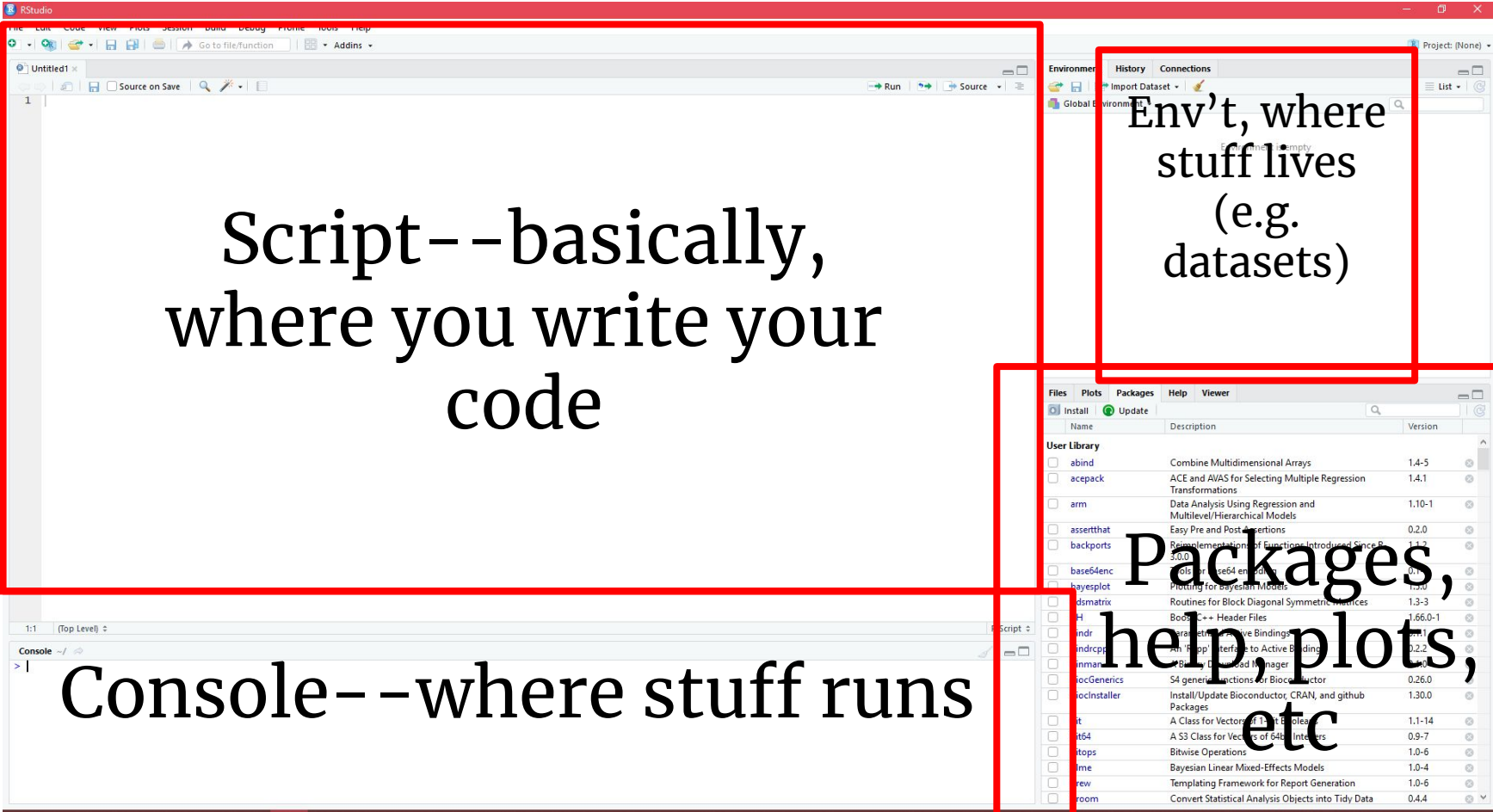
CLASS GOALS

1. Install R on all our computers
2. Open R and not have a panic attack
3. Do some small/basic stuff in R
4. Load a data set into R

Ambitious Goal:

1. Look at data types in R

WHAT IS R?



The image shows a screenshot of the RStudio application window. The main editor pane on the left contains the text 'Script -- basically, where you write your code'. The environment pane on the top right contains the text 'Env't, where stuff lives (e.g. datasets)'. The packages pane on the bottom right contains the text 'Packages, help, plots, etc'. The console pane at the bottom left contains the text 'Console -- where stuff runs'. Red rectangular boxes highlight each of these four areas.

Script -- basically,
where you write your
code

Env't, where
stuff lives
(e.g.
datasets)

Console -- where stuff runs

Packages,
help, plots,
etc

R: WHY IS IT COOL?

- It is free to everyone, and anyone can create programs for it, so cool stuff exists like:
 - *Recoder*
 - *A package with a sole purpose of recoding variables*
 - *GGplot*
 - *Literally, the most amazing graphs*
 - *Psych*
 - *Stats stuff for psychological, psychometric, and personality research*
 - *PRoc*
 - *Literally, a whole package just for ROC curves (Jason!)*

R: WHY IS IT COOL?

- You can do a lot of stats, like:

ANALYTICS

- Basic Mathematics
- Basic Statistics
- Probability Distributions
- Big Data Analytics *
- Machine Learning
- Optimization and Mathematical Programming
- Signal Processing
- Simulation and Random Number Generation
- Statistical Modeling
- Statistical Tests

GRAPHICS AND VISUALIZATION

- Static Graphics
- Dynamic Graphics
- Devices and Formats

R APPLICATIONS and EXTENSIONS***

- Applications
- Data Mining and Machine Learning
- Statistical Methodology
- Other Distributions Available in Third-Party Packages ***

• PROGRAMMING LANGUAGE FEATURES

- Input / Output
- Object-oriented programming
- Distributed Computing
- Included R Packages

R: WHY IS IT COOL?

• Data cleaning is super awesome on R

Data Wrangling with dplyr and tidyR Cheat Sheet



Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen.

```
Source: Local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1 5.1 3.5 1.4
2 4.9 3.0 1.4
3 4.7 3.2 1.3
4 4.6 3.1 1.3
5 5.0 3.6 1.4
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

dplyr::glimpse(iris)
Information dense summary of tbl data.
utils::View(iris)
View data set in spreadsheet-like display (note capital V).

```
iris %>%
  summarise(
    avg_sep = mean(Sepal.Length),
    avg_pet = mean(Petal.Length)
  )
```

dplyr::%>%
Passes object on left hand side as first argument (or argument of function on right hand side).

x %>% f(y) is the same as **f(x, y)**
y %>% f(x, .., z) is the same as **f(x, y, z)**

Piping with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Length)) %>%
  arrange(avg)
```

RStudio® is a trademark of RStudio, Inc. • <https://www.rstudio.com> • 844-448-1212 • <https://www.rstudio.com>

Tidy Data - A foundation for wrangling in R

In a tidy data set:
Each variable is saved in its own column
Each observation is saved in its own row

Tidy data complements R's vectorized operations. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Reshaping Data - Change the layout of a data set

tidy::gather(cases, "year", "n", 2:4)
Gather columns into rows.

tidy::spread(pollution, size, amount)
Spread rows into columns.

dplyr::data_frame(a=1:3, b=4:6)
Combine vectors into data frame (optimized).

dplyr::arrange(mtcars, mpg)
Order rows by values of a column (low to high).

dplyr::arrange(mtcars, desc(mpg))
Order rows by values of a column (high to low).

dplyr::rename(mtcars, y = year)
Rename the columns of a data frame.

tidy::separate(storms, date, c("y", "m", "d"))
Separate one column into several.

tidy::unite(data, col, ..., sep)
Unite several columns into one.

Subset Observations (Rows)

dplyr::filter(iris, Sepal.Length > 7)
Extract rows that meet logical criteria.

dplyr::distinct(iris)
Remove duplicate rows.

dplyr::sample_frac(iris, 0.5, replace = TRUE)
Randomly select fraction of rows.

dplyr::sample_n(iris, 10, replace = TRUE)
Randomly select n rows.

dplyr::slice(iris, 10:15)
Select rows by position.

dplyr::top_n(storms, 2, date)
Select and order top n entries (by group if grouped data).

Logic in R	Comparison	base Logic
<	Less than	<
>	Greater than	>
=	Equal to	==
<=	Less than or equal to	<=
>=	Greater than or equal to	>=

Subset Variables (Columns)

dplyr::select(iris, Sepal.Width, Petal.Length, Species)
Select columns by name or helper function.

Helper functions for select -> select

- select(iris, contains("n"))**
Select columns whose name contains a character string.
- select(iris, ends_with("year"))**
Select columns whose name ends with a character string.
- select(iris, everything())**
Select every column.
- select(iris, matches("n"))**
Select columns whose names matches a regular expression.
- select(iris, num_range("x", 1:3))**
Select columns named x1, x2, x3, x4, x5.
- select(iris, one_of(c("Species", "Genus")))**
Select columns whose names are in a group of names.
- select(iris, starts_with("year"))**
Select columns whose name starts with a character string.
- select(iris, Sepal.Length:Petal.Width)**
Select all columns between Sepal.Length and Petal.Width (inclusive).
- select(iris, -Species)**
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for dplyr

Learn more with [browniegreen\(package = "dplyr"\)](https://www.rstudio.com) • dplyr • dplyr 0.4.0-tdy 0.2.0 • updated 1/15

Summarise Data

dplyr::summarise(iris, avg = mean(Sepal.Length))
Summarise data into single row of values.

dplyr::summarise_each(iris, funs(mean))
Apply summary function to each column.

dplyr::count(iris, Species, wt = Sepal.Length)
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::first First value of a vector.	min Minimum value in a vector.
dplyr::last Last value of a vector.	max Maximum value in a vector.
dplyr::nth Nth value of a vector.	mean Mean value of a vector.
dplyr::n # of values in a vector.	median Median value of a vector.
dplyr::n_distinct # of distinct values in a vector.	var Variance of a vector.
IQR IQR of a vector.	sd Standard deviation of a vector.

Group Data

dplyr::group_by(iris, Species)
Group data into rows with the same value of Species.

dplyr::ungroup(iris)
Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(...)
Compute separate summary row for each group.

iris %>% group_by(Species) %>% summarise(...)
Compute separate summary row for each group.

RStudio® is a trademark of RStudio, Inc. • <https://www.rstudio.com> • 844-448-1212 • <https://www.rstudio.com>

Make New Variables

dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)
Compute and append one or more new columns.

dplyr::mutate_each(iris, funs(min_rank))
Apply window function to each column.

dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)
Compute one or more new columns. Drop original columns.

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::lead Copy with values shifted by 1.	dplyr::cumall Cumulative all
dplyr::lag Copy with values lagged by 1.	dplyr::cumany Cumulative any
dplyr::dense_rank Ranks with no gaps.	dplyr::cummean Cumulative mean
dplyr::min_rank Ranks. Ties get min rank.	cumsum Cumulative sum
dplyr::percent_rank Ranks rescaled to [0, 1].	cummax Cumulative max
dplyr::row_number Ranks. Ties get to first value.	cumin Cumulative min
dplyr::ntile Bin vector into n buckets.	cumprod Cumulative prod
dplyr::between Are values between a and b?	pmx Element-wise max
dplyr::cume_dist Cumulative distribution.	pmin Element-wise min

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.

devtools::install_github("rstudio/EDAWR") for dplyr

Learn more with [browniegreen\(package = "dplyr"\)](https://www.rstudio.com) • dplyr • dplyr 0.4.0-tdy 0.2.0 • updated 1/15

Combine Data Sets

dplyr::left_join(a, b, by = "x1")
Join matching rows from b to a.

dplyr::right_join(a, b, by = "x1")
Join matching rows from a to b.

dplyr::inner_join(a, b, by = "x1")
Join data. Retain only rows in both sets.

dplyr::full_join(a, b, by = "x1")
Join data. Retain all values, all rows.

dplyr::semi_join(a, b, by = "x1")
All rows in a that have a match in b.

dplyr::anti_join(a, b, by = "x1")
All rows in a that do not have a match in b.

dplyr::intersect(x, y)
Rows that appear in both x and y.

dplyr::union(x, y)
Rows that appear in either or both x and y.

dplyr::setdiff(x, y)
Rows that appear in y but not x.

dplyr::bind_rows(y, z)
Append z to y as new rows.

dplyr::bind_cols(y, z)
Append z to y as new columns. Caution: matches rows by position.

dplyr::bind_rows(y, z)
Append z to y as new rows.

dplyr::bind_cols(y, z)
Append z to y as new columns. Caution: matches rows by position.

devtools::install_github("rstudio/EDAWR") for dplyr

R: WHY IS IT COOL?

- So many other cool ways to share your analyses, like:
 - *MAPs website!*
 - *RPubs*
 - *Good option for public stuff*
 - *Github*
 - *Good option to keep private (e.g. de-identified data)*
 - *Super R noobz nerds can get their own github account---private ones are free for people who can prove they're students!!*

ALRIGHT FINE, YOU
CONVINCED ME...

:D LET'S INSTALL R!

- Step 1: go [here](#) to download R
- Step 2: choose the r download type based on your computer

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

:D LET'S INSTALL R!

- Step 3: click “install R for the first time”

R FOR WINDOWS

Subdirectories:

[base](#)

Binaries for base distribution. This is what you want to **install R for the first time**.

[contrib](#)

Binaries of contributed CRAN packages (for R \geq 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).

[Rtools](#)

Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

:D LET'S INSTALL R!

- Step 4: Click “Download R 3.5.1 for Windows”
- Step 5: Follow all the installation instructions

R-3.5.1 for Windows (32/64 bit)

Download R 3.5.1 for Windows (62 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is
[<CRAN MIRROR>/bin/windows/base/release.htm](#).

Last change: 2018-07-02

:D LET'S INSTALL R!

- Step 6: Go [here](#) to install R studio, choosing the leftmost option and clicking the “Download” button, choosing the right option for your computer OS when it directs you to the page shown on the pic to the right. Follow download instructions!

Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace. [Learn More about RStudio features.](#)



	RStudio Desktop Open Source License	RStudio Desktop Commercial License	RStudio Server Open Source License	RStudio Server Pro Commercial License	RStudio Server Pro + RStudio Connect Commercial License
	FREE	\$995 per year	FREE	\$9,995 per year	\$29,995 per year
	DOWNLOAD Learn More	BUY Learn More	DOWNLOAD Learn More	DOWNLOAD Learn More	TALK Learn More
Integrated Tools for R	●	●	●	●	●
Priority Support		●		●	●
Access via Web Browser			●	●	●
Enterprise Security				●	●
Project Sharing				●	●

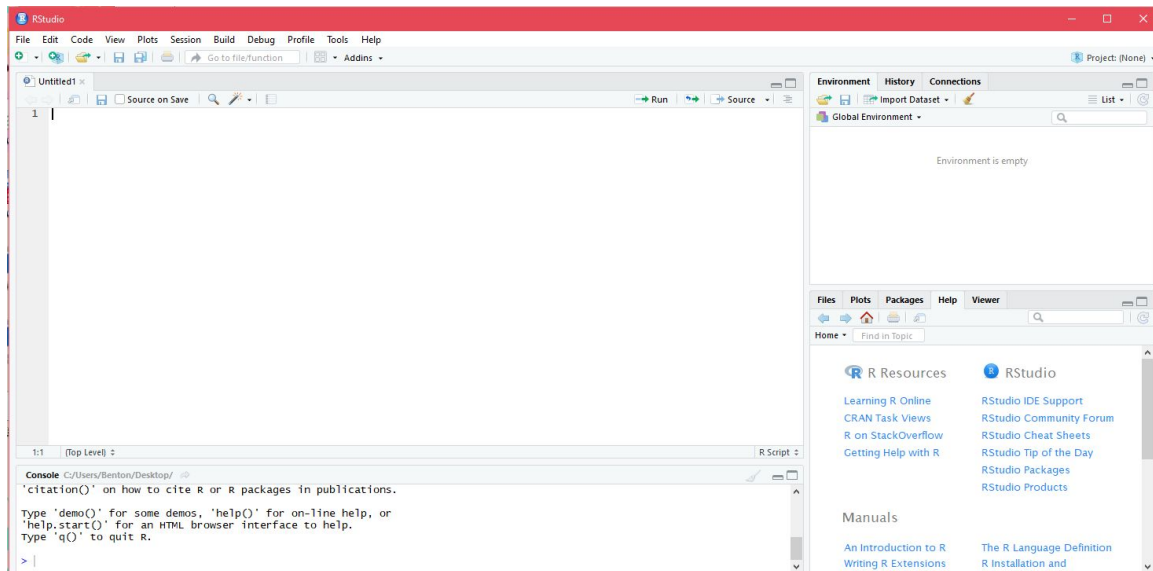
Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.453 - Windows Vista/7/8/10	85.8 MB	2018-05-16	bf287e385aef53829204023087e98735
RStudio 1.1.453 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-05-16	00a0088424ed06ac4347a966f602b9c
RStudio 1.1.453 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-05-16	6cfd86770c7b6dbc13e66f4f59c299ce
RStudio 1.1.453 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-05-16	63e36e8138e369d19f9aaf4b0e995bbc
RStudio 1.1.453 - Ubuntu 16.04+/Debian 9+ (64-bit)	64.4 MB	2018-05-16	85b3e76c9fad4613bc9cf0de1f34b183
RStudio 1.1.453 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-05-16	37cade7e162eab62483e6556e39dedee
RStudio 1.1.453 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-05-16	44cddd285bc31c41e4aec1d74b8eebb

SUH, BRO

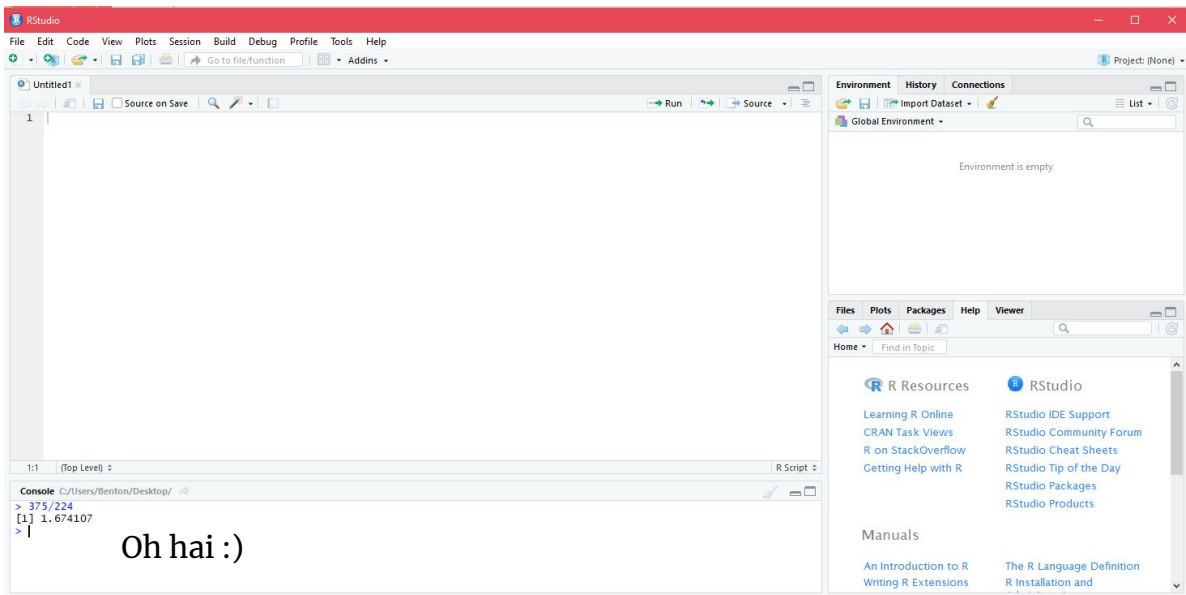
OPEN R

- You've just downloaded R! Congrats! Open the R Studio link--do you see what I showed you earlier?



IF SO, LET'S DO SOMETHING COOL

The console of R is a calculator! Type in an equation into it, and see what happens



WUT IS AN OBJECT?

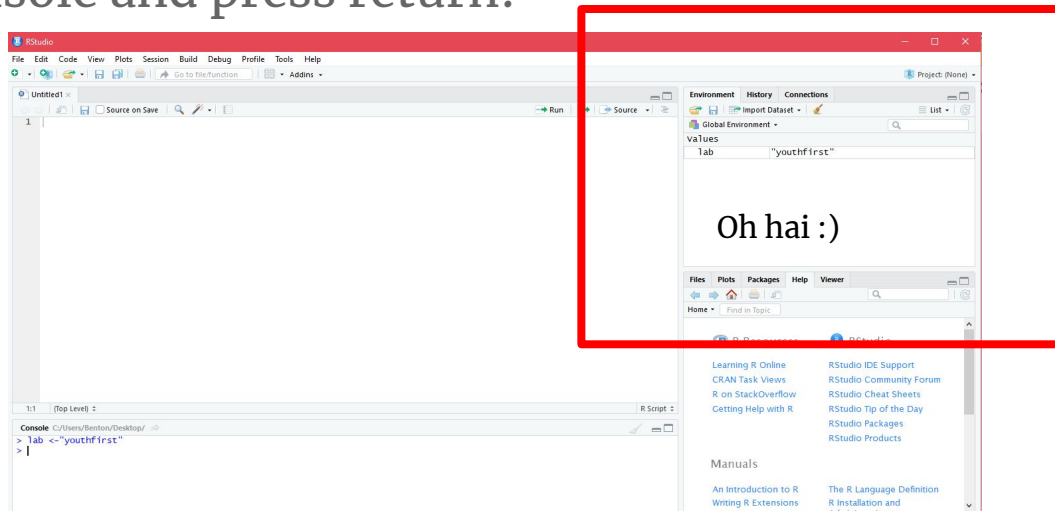
- One of the most basic things you use in R are objects. Objects are defined as:
 - *An object is a data structure having some attributes and methods which act on its attributes.*
 - *...wut?*
 - *Basically, an object is:*
 - *a thing in R*
 - *that is some kind of way*
 - *e.g. a value! A number! A word! A dataset! Whatever! Who am I to set parameters on how you define yourself, object?*
 - *that we can do things to*

MAKE AN OBJECT!

- Let's make our first object! Type in the phrase:

`lab <- "YouthFirst"`

into your console and press return!



MAKE AN OBJECT!

- You just successfully made an object! Congrats! You're basically Peter at this point! 🤪
- Some important R notes:
 - “<-” is like “=”, but we don't use = in R
 - ...IDK? It just is.
 - R is case sensitive. That means, unlike spss, if you name your object “lab” it will not think that “Lab” is in any way the same thing as “lab”...just go with it.
 - Hint...if you can't remember what you named your variable, try looking at the dropdown list that pops up above what you're typing---its like phone a friend for R
- You can figure out what type of object “lab” is by typing in”

class(lab)

into your console

MAKE AN OBJECT!

- Turns out it is a character. Huh, cool.
- R has a few types of objects:
 - *Character*
 - *Integer*
 - *Numeric*
 - *Logical*
 - *Matrix*
 - *Data Frame*

```
1:1 (Top Level) R Script
Console C:/Users/Benton/Desktop/
> lab <- "youthfirst"
> class(lab)
[1] "character"
> |
```

Oh hai :)

BUCKLE YOUR
SEATBELT

LET'S SEE IF WE CAN LOAD IN A DATASET...

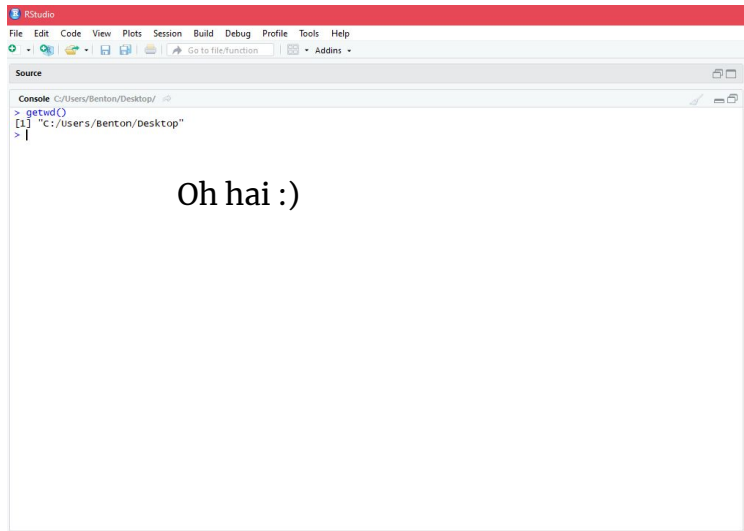
I emailed you some files. Let's see if we can load them into r, and play around with the data by seeing the class of each variable!

Go into your email, download the 2 files I sent you, and put them on your desktop.

LET'S SEE IF WE CAN LOAD IN A DATASET...

Working directory is basically where R is getting its schtuff from.

You can see what your current working directory is by typing into the console the command: “**getwd()**”

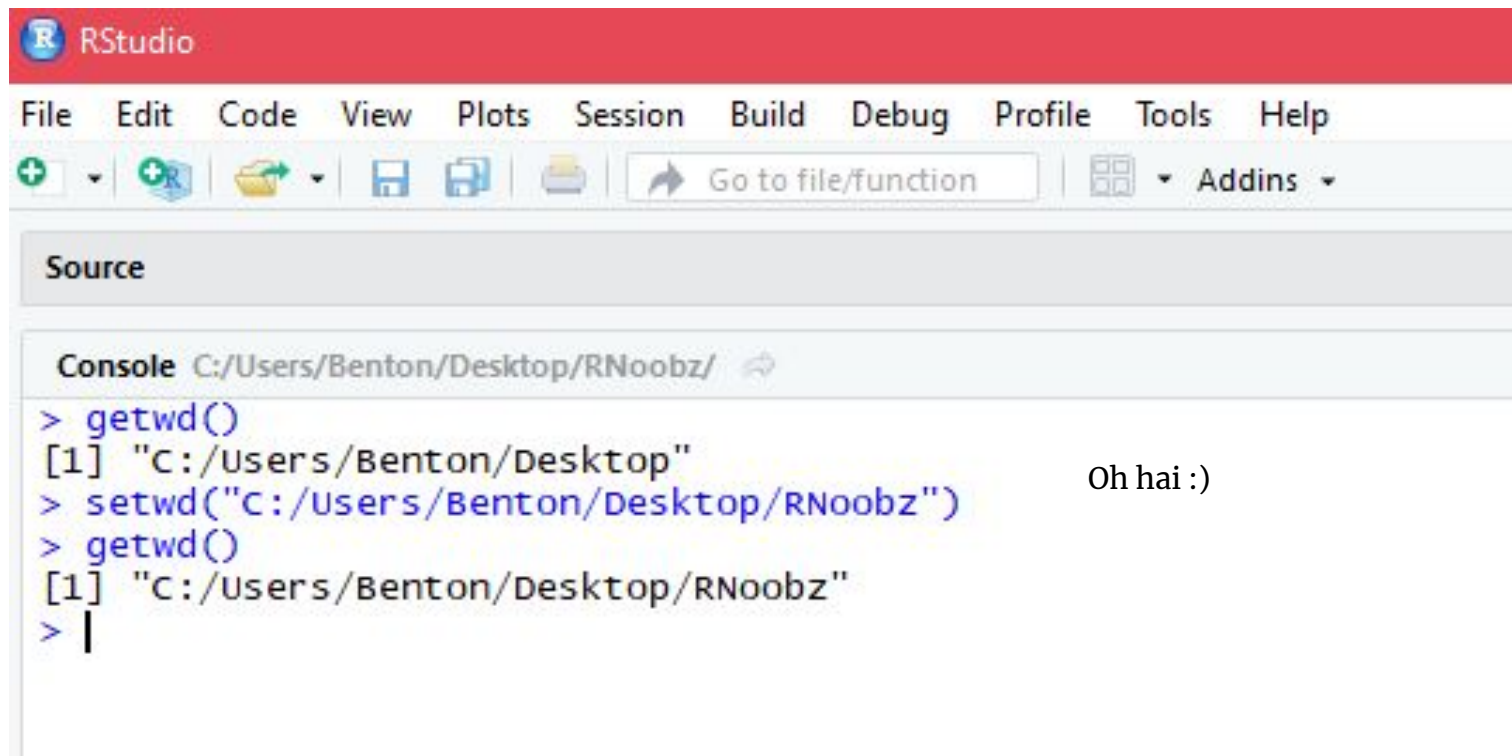


LET'S SEE IF WE CAN LOAD IN A DATASET...

You can change your working directory by typing in the command “**setwd()**” and putting a path to the place you’d like to set your directory in between the parentheses. Be sure you put quotes around it.

For example, say I wanted to make my working directory a folder on my desktop, I’d do the following:

LET'S SEE IF WE CAN LOAD IN A DATASET...



The screenshot shows the RStudio application window. The title bar is red with the RStudio logo and name. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for creating a new file, opening a file, saving, and a search bar labeled 'Go to file/function'. Below the toolbar is a 'Source' pane. At the bottom is a 'Console' pane with the title 'C:/Users/Benton/Desktop/RNoobz/'. The console shows the following commands and output:

```
> getwd()
[1] "C:/Users/Benton/Desktop"
> setwd("C:/Users/Benton/Desktop/RNoobz")
> getwd()
[1] "C:/Users/Benton/Desktop/RNoobz"
> |
```

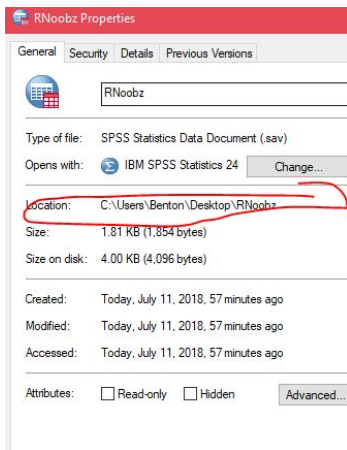
Oh hai :)

LET'S SEE IF WE CAN LOAD IN A DATASET...

- Small note here:

- *Paths are different in Mac vs. Windows. I still haven't memorized the difference (the slashes are different--it's weird).*

If you really want to know how to do the right paths to your director, files, etc, you can always find an example from getting info from a file in the working directory you're using/wanting to set...



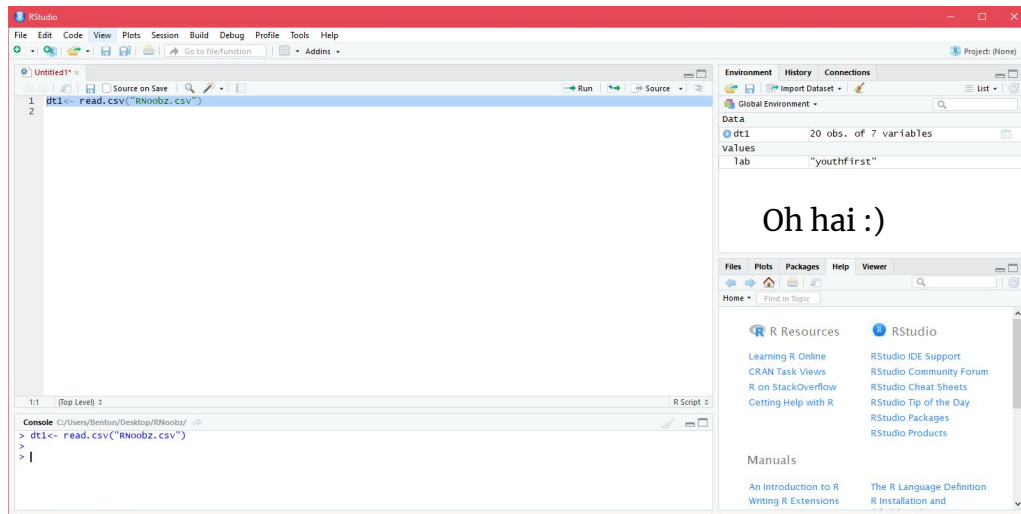
LET'S SEE IF WE CAN LOAD IN A DATASET...

OK so, let's load in the CSV file, and name it dt1

Type in the command:

```
dt1<-  
read.csv("RNoobz.csv")
```

See dt1 in your data section of the global environment?



LET'S SEE IF WE CAN LOAD IN A DATASET...

Let's see what kind of class of object dt1 is...

class(dt1)

OK, neat! It is a data frame!
Cool beans!

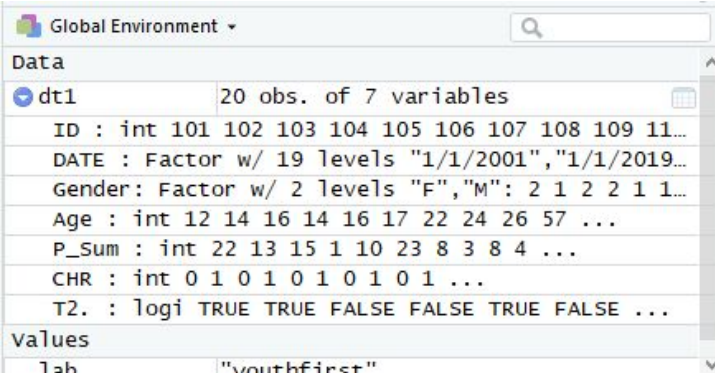
```
Console C:/Users/Benton/Desktop/RNoobz/ ↗  
> dt1<- read.csv("RNoobz.csv")  
>  
> class(dt1)  
[1] "data.frame"  
> |
```

Oh hai :)

LET'S SEE IF WE CAN LOAD IN A DATASET...

If you click the little blue button on the left, that can show you what kind of variables you've got going on, and their respective classes...

Let's take a look at these bad boys!!



The screenshot shows the R Global Environment window. Under the 'Data' tab, a dataset named 'dt1' is listed with 20 observations and 7 variables. A preview of the first 11 observations is shown below the dataset name. The variables and their classes are: ID (integer), DATE (Factor with 19 levels), Gender (Factor with 2 levels), Age (integer), P_Sum (integer), CHR (integer), and T2 (logical).

Variable	Class	Values (first 11 obs)
ID	int	101 102 103 104 105 106 107 108 109 11...
DATE	Factor w/ 19 levels	"1/1/2001", "1/1/2019..."
Gender	Factor w/ 2 levels	"F", "M": 2 1 2 2 1 1...
Age	int	12 14 16 14 16 17 22 24 26 57 ...
P_Sum	int	22 13 15 1 10 23 8 3 8 4 ...
CHR	int	0 1 0 1 0 1 0 1 0 1 ...
T2	logi	TRUE TRUE FALSE FALSE TRUE FALSE ...

Values: 1st "youthfirst"

Oh hai :)

LET'S SEE IF WE CAN LOAD IN A DATASET...

If you type in the name of your dataframe, then \$, then the variable name, that is a way we can let R know we are talking about a specific variable...like, for gender...

```
Console C:/Users/Benton/Desktop/RNoobz/
> dt1$Gender
[1] M F M M F F F M F M M F M F M F M F F M
Levels: F M
> |
```

```
> class(dt1$Gender)
[1] "factor"
> |
```

```
> summary(dt1$Gender)
  F  M
10 10
> |
```

LET'S SEE IF WE CAN LOAD IN A DATASET...

OK cool, let's see if we can do this for the variables in our dataset...



LET'S SEE IF WE CAN LOAD IN A DATASET...

OK we are bored...fine.

Let's install a package!!

Type in:

install.packages("foreign")

Then:

library(foreign)

Into your console

```
> install.packages("foreign")
Installing package into 'C:/Users/Benton/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
warning in install.packages :
  package 'foreign' is not available (for R version 3.5.1)
> library(foreign)
> |
```


LET'S SEE IF WE CAN LOAD IN A DATASET...

This package lets us import weirdo types of files into R, like SPSS files.

It is super clutch.

Try typing in the code pictured here...

Do you see a new object called dt2 in your global environment?

```
> dt2 <- read.spss("RNoobz.sav")  
re-encoding from UTF-8  
> |
```

SO...DID IT WORK?

This package is nice, because it can import all kinds of neat-o labels and values from SPSS data files!

```
Console C:/Users/Benton/Desktop/RNoobz/
> dt2 <- read.spss("RNoobz.sav")
re-encoding from UTF-8
> dt2$gender
[1] Male   Female Male   Male   Female Female Female Male   Female Male   Male   Female Male   Female Male   Female
[17] Male   Female Female Male
Levels: Female Male
> |
```

ARE WE FEELING
STRESSED OR
AMBITIOUS?

R SNOOPING

In Class homework assignment!!

See if you can figure out the class of every variable in the dataset I gave you, and let me know how variables may be different in the CSV file vs. the SPSS file!



CONGRATS, R NOOBZ!
YOU ARE LITERALLY A
DATA WIZARD NOW!

NEXT CLASS:
DATA CLEANING IN
R!

HOMework, IF YOU FEEL LIKE IT:

LOAD IN THE ALL AVAILABLE DATA DATASET INTO R,
AND LOOK AT THE VARIABLES!

QUESTIONS?
COMMENTS?
THANKS!!!
GO TEAM!