

Research Issues and Practices in Psychology

Week 1: Gentle R Introduction

Jason Geller, Ph.D.

2022-07-27

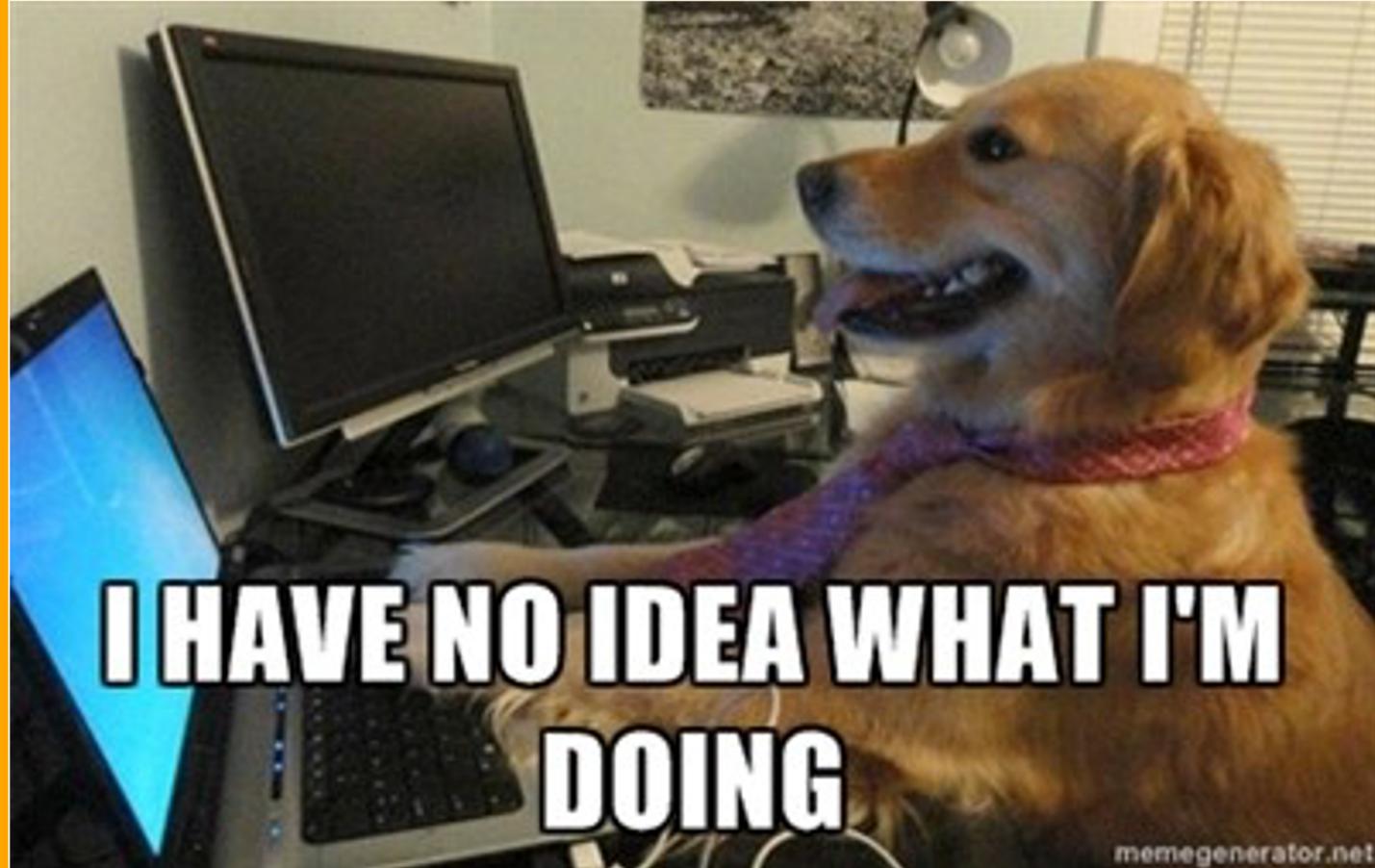
Objective



- Get you started with R
- Load your first dataset in R
- Explain some basic terminology and concepts
- Explain how to structure any data analysis project
- Learn how to run commands and save scripts



I know it is scary



Outline

- Why R
- IDE
- R commands & functions
 - Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts

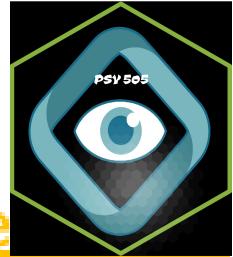


Why R?

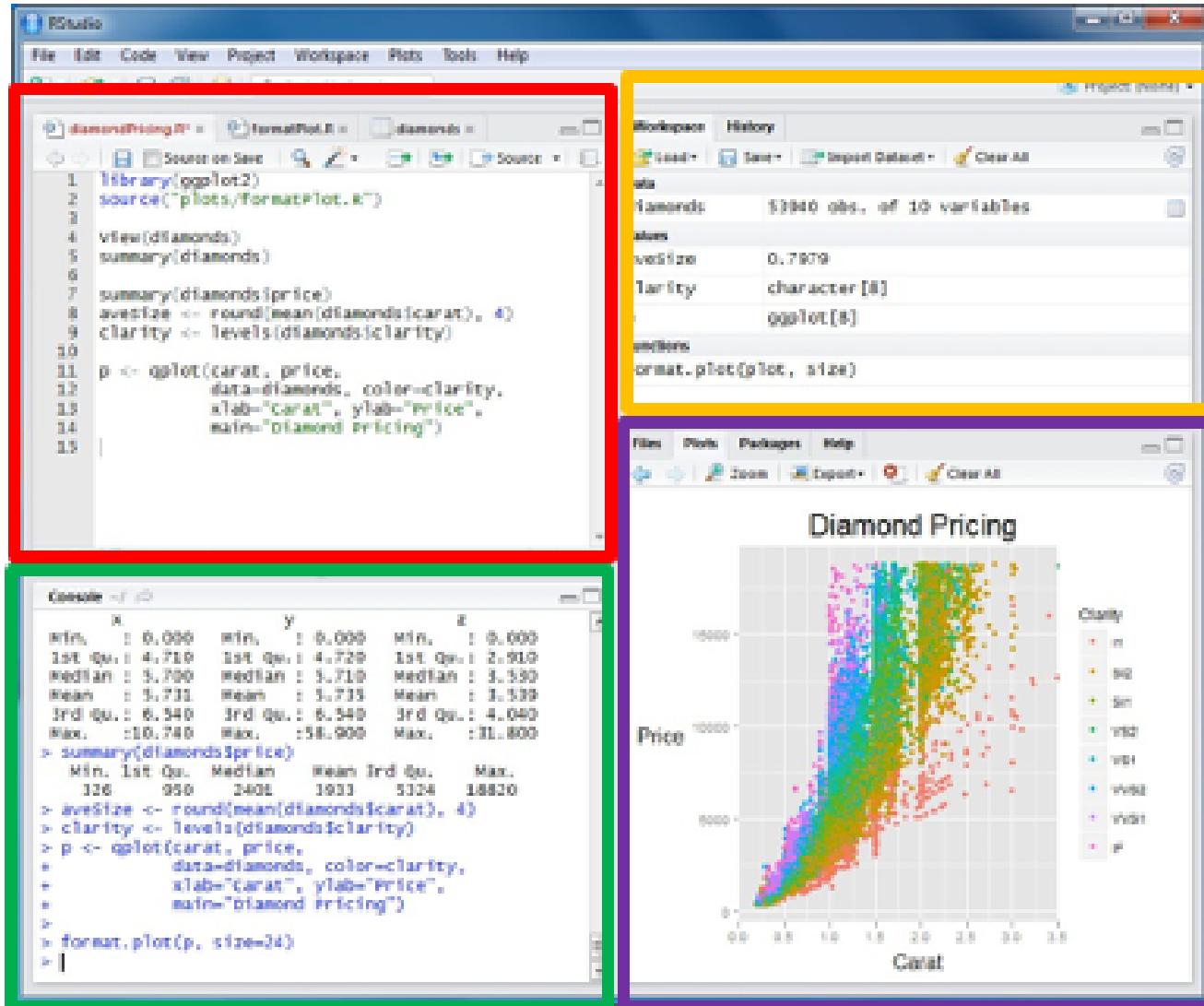
- Free and open-source
- Flexibility
- Programming language (not point-and-click)
- Excellent graphics (via `ggplot2`)
- Easy to generate with reproducible reports
- Easy to integrate with other tool
- Inclusive Community



Rstudio Integrated Development Environment (IDE)



Source: edit file that you can run again later.



Console: type/paste commands to get output from R

Workspace: see list of variables and previous commands

Files/Plots/Package s/Help

Outline

- Why R
- IDE
- R commands & functions
 - Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts



Assignment of Variables



- A variable is a symbol that stands for another value (just like “X” in algebra)

```
x <- 4  
x
```

```
## [1] 4
```

- The arrow `<-` is called an **ASSIGNMENT OPERATOR**, and tells R to save an object called `x` that has the value of 6. This is similar to saving a value in a graphing calculator.
- Can use `=` if you want

```
x + 7
```

```
## [1] 11
```

Vectors



- Allows multiple types of classes to be concatenated together
 - Numeric

```
x <- c(2, 6, 16)  
x
```

```
## [1] 2 6 16  
- `x` here is called an object
```

- Logical (TRUE/FALSE)

```
x[1]==2
```

```
## [1] TRUE
```

Vectors



- Character

```
x <- c("cat", "bat")  
gender <- c("male", "female")
```

- Factors
- Turn character strings into specific categories

```
gender <- as.factor(gender)  
gender
```

```
## [1] male   female  
## Levels: female male
```

Indexing



- Vectors can be indexed

```
x[1] # retreve first
```

```
## [1] "cat"
```

```
x[2] # retreve second
```

```
## [1] "bat"
```

```
x[-2] # everythin but that numbe
```

```
## [1] "cat"
```

- Change values in vector

```
x[1] <- 7
```

```
x
```

Vectors

- Logical vectors



```
x==7
```

```
## [1] TRUE FALSE
```

R as a Calculator



- Typing in a simple calculation show us the result

```
608 + 28
```

```
## [1] 636
```

```
11527 - 283
```

```
## [1] 11244
```

```
# division  
400/65
```

```
## [1] 6.153846
```

```
#multiplication  
2*4
```

```
## [1] 8
```

Functions



- Take an object, do something to it, and return the result
- More complex calculations can be done with functions:
 - What is square root of 64?

```
# sqrt function
# in parenthesis: what we want to perform function on
sqrt(64)
```

```
## [1] 8
```

```
sr=function(a, b){  
  a=x  
  b=x  
  c=a + b  
  return(c)  
}  
sr(2,3)
```

Arguments



- Some functions have settings (“arguments”) that we can adjust:
- `round(3.14)`
 - Rounds off to the nearest integer (zero decimal places)
- `round(3.14, digits=1)`
 - One decimal place

Getting Help

1. Help files



Anatomy of an R help file

Two ways to access:

1. Peruse in Help pane
2. ?<name> in console

The name of the function, and the library it is in.

Documentation Author(s): R Core Team

What it does.

Description General function for the (trimmed) arithmetic mean.

Usage `mean(x, ..., trim = 0, na.rm = FALSE, ...)`

Arguments

- `x`: An R object. Currently there are methods for numeric/logical vectors and [date-time](#) and [time-interval](#) objects. Complex vectors are allowed for `trim > 0`, only.
- `trim`: the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.
- `na.rm`: a logical value indicating whether `NA` values should be stripped before the computation proceeds.
- `...`: further arguments passed to or from other methods.

The ellipsis allows other arguments to be passed to and from the function.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `na.rm` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also [mean.default](#), [mean.matrix](#), [mean.POSIXct](#), [mean.POSIXt](#) for row and column means.

Other related functions

Examples

```
x <- rbinom(100, 100)
m1 <- mean(x)
t1(m1, mean(x, trim = 0.1))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

(Package last version 3.4.3 [Index](#))

visit the package's index page to look for Demos and Vignettes detailing how it works.



Exercises



1. Open a blank new script

1.1 File -> New File > R Script

1.2 Ctrl + Shift + N

1.3 Click on new script icon

1. In one of these scripts type "Hello World"

2. To paste strings together you can use the `paste()` function (e.g., `paste("Hello", "World")`).

3.1 Use the `paste` function to string together a sentence of your choice. Assign it to a variable or object.

1. Modify the function above and instead of returning the sum return the .

Tidyverse and Pipes



- The **tidyverse** is an ecosystem of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.



tidyverse

string manipulation



data manipulation



data
tidying



reading
data



data
visualisation



data
wrangling



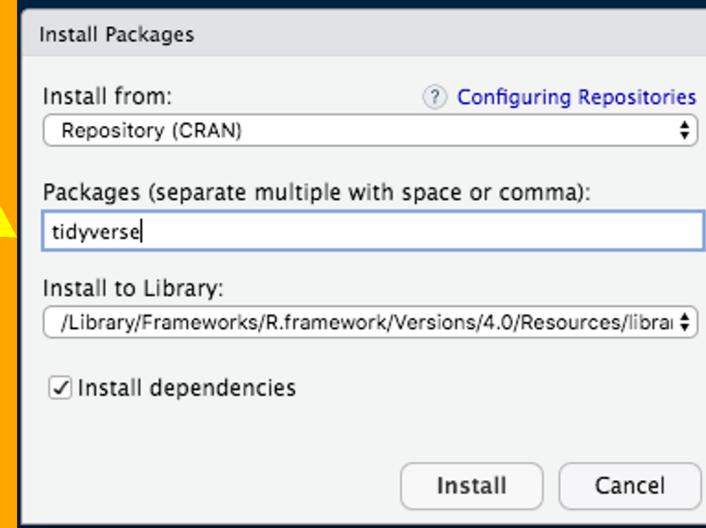
mod
data frames



functional
programming



Installing Tidyverse



```
install.packages(tidyverse)
```

- Load package

```
library(tidyverse)
```

Pipes



- `tidyverse` provides another interface to functions—the pipe operator
 - Makes code easier to read and follow:
- This:

```
a %>%  
  round()
```

- Can be converted into:
 - Start with a and then round
- Become popular is the `|>` pipe

Outline

- Why R
- IDE
- R commands & functions
 - Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts



Multiple Functions



- Pipe operator makes it easy to do multiple functions in a row

```
-16 %>%  
sqrt() %>%  
abs()
```

- What is this doing?

Outline

- Why R
- IDE
- R commands & functions
 - Tidyverse & the Pipe Operator
- Multiple Functions
- [Reading in data](#)
- Saving R scripts



Reading in Data



- Download the file



function	reads
<code>read_csv()</code>	Comma separatedvalues
<code>read_csv2()</code>	Semi-colon separatedvalues
<code>read_delim()</code>	General delimited files
<code>read_fwf()</code>	Fixed widthfiles
<code>read_table()</code>	Space separated
<code>read_tsv()</code>	Tab delimited values

- General form: `dataframe.name <- read.csv('filename')`

Data Frames



A data frame is like an Excel spreadsheet. It is two-dimensional with rows and columns.

- Instead of creating a number of vectors we store all the vectors into a single DF
- Can store numeric data (phone number, postal code, coordinates, etc.), float data (internet IP address, etc.), logical data (wants to receive ads: FALSE/TRUE, etc.), etc.

```
car_model <- c("Ford Fusion", "Hyundai Accent", "Toyota Corolla")
car_price <- c(25000, 16000, 18000)
car_mileage <- c(27, 36, 32)

cars_df <- data.frame(model=car_model, price=car_price, mileage=car_mileage)

flextable::flextable(cars_df) %>% flextable::autofit()
```



Tibbles

- More modern take on Data frames
 - Never changes input's type
 - Never adjusts the names of variables
 - It evaluates arguments lazily and sequentially
- Differences
 - Printing

```
as.tibble()
```

Here package



- `Here` helps set relative as opposed to absolute paths
 - Why would this be a problem?

```
#setwd("your path here")
```

```
#install here
library(here)

# here
here::here()
```

```
## [1] "/Users/jgeller1/Desktop/Psy505_issues_application"
```

```
# can use with read.csv
```

Loading the Data



| Disclaimer: Create new project folder first, open up R second

The faculty dataset contains aggregated data per faculty:

- faculty: Business, Economics, Political Science, Sociology
- students: number of students
- profs: number of profs
- salary: amount of salary
- costs: amount of costs dataset entails demographic and school-related information on imaginary students, such as

Load the data



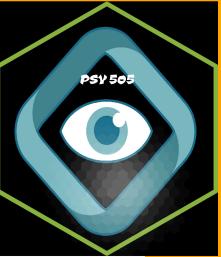
CSV

```
fac=read.csv(here::here("static", "slides", "01-R", "datasets", "faculty.csv"))

fac
```

```
##          faculty students profs salary costs
## 1      Business      339     76  57273 33346
## 2 Economics       225     79  83292 33527
## 3 Political Science  264     63  66425 24965
## 4 Sociology        162     77  54246 29640
```

Looking at Data



```
fac%>%  
  summary()
```

```
##   faculty      students      profs       salary  
## Length:4      Min.   :162.0   Min.   :63.00   Min.   :54246  
## Class :character 1st Qu.:209.2   1st Qu.:72.75   1st Qu.:56516  
## Mode  :character Median :244.5   Median :76.50   Median :61849  
##                   Mean   :247.5   Mean   :73.75   Mean   :65309  
##                   3rd Qu.:282.8   3rd Qu.:77.50   3rd Qu.:70642  
##                   Max.   :339.0   Max.   :79.00   Max.   :83292  
##  
##   costs  
## Min.   :24965  
## 1st Qu.:28471  
## Median :31493  
## Mean   :30370  
## 3rd Qu.:33391  
## Max.   :33527
```

Looking at Data



```
library(skimr)  
fac%>%  
  skim()
```

Table: Data summary

Name	Piped data
Number of rows	4
Number of columns	5
-	
Column type frequency:	
character	1
numeric	4
--	

Looking at the Data



- Select specific columns
 - use \$ operator to grab one column

```
fac$column_name %>%
  summary()
```

```
##   Length Class  Mode
##       0    NULL  NULL
```

```
fac %>%
  select(column_name) %>%
  summary()
```

Looking at the Data



- Whole dataset
- First 6 observations
- Last 6 observations

```
fac # whole dataset
```

```
##          faculty students profs salary costs
## 1      Business      339     76  57273 33346
## 2    Economics      225     79  83292 33527
## 3 Political Science  264     63  66425 24965
## 4      Sociology     162     77  54246 29640
```

```
head(fac)
```

```
##          faculty students profs salary costs
## 1      Business      339     76  57273 33346
## 2    Economics      225     79  83292 33527
## 3 Political Science  264     63  66425 24965
## 4      Sociology     162     77  54246 29640
```

```
tail(fac)
```

Looking at Data



```
# look at specific variables  
table(fac$students)
```

```
##  
## 162 225 264 339  
## 1 1 1 1
```

```
# let's try another package  
library("janitor")  
tabyl(fac$students)
```

```
## fac$students n percent  
## 162 1 0.25  
## 225 1 0.25  
## 264 1 0.25  
## 339 1 0.25
```

Reading in Other File Types



- Excel

```
library(readxl)  
  
fac<- read_excel('/Users/jg/Desktop/experiment.xlsx', sheet=2)  
# excel files can have multiple sheets
```

- SPSS

```
library(haven)  
  
fac<- read_spss('/Users/jg/Desktop/experiment.spss')
```



Outline

- Why R
- IDE
- R commands & functions
 - Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts



Saving Files



```
write.csv(fac, file="df.csv")  
write.table(fac, file="df.txt")
```

Getting help

1. R help files
2. Cheat sheets (<https://rstudio.cloud/learn/cheat-sheets>)
3. Google!



Exercise



1. Create a variable called `y` with the value of 7
2. Save the results of $6 + 3$ as a variable called `a`.
3. Create a new project folder for this course entitled "psy_505" (run the `.Rproj` file)
 - 3.1 Place the `exercise.csv` file in the folder
4. Using `here` assign the file to a name of your choice
5. Explore the data set by running the commands `head(data)`, `str(data)`, `glimpse(data)` and `summary(data)` in your R script. You will use these a lot in the future, so have a closer look at the different outputs in the console (lower left). Remember to save your script!