Intro to R for Data Science

Beginner's workshop

AbdulMajedRaja RS

About Me

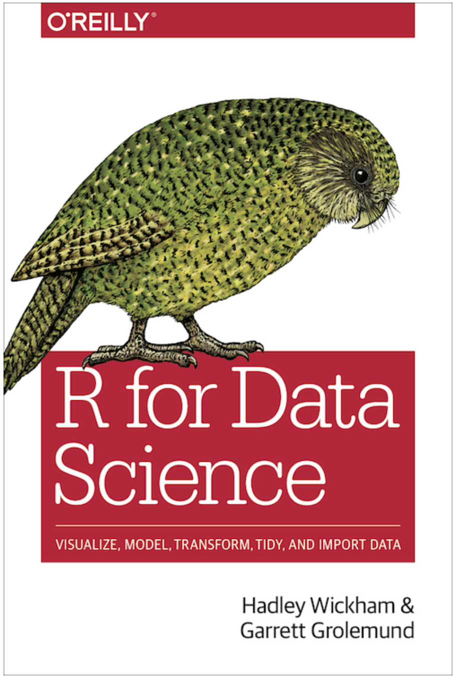
- Studied at **Government College of Technology, Coimbatore**
- Bengaluru R user group **Organizer**
- R Packages Developer (`coinmarketcapr`, `itunesr`)

Disclaimer:

- This workshop is **NOT** going to make you a Data Scientist **in a day**.
- The objective is to help you get a flavor of R and how it is used in Data Science
- Thus, get you ready to embark on your own journey to become a Data Scientist who uses R

Content:

This presentation's content is heavily borrowed from the book **R for Data Science** by **Garrett Grolemund** and **Hadley Wickham**



About R

- R is a language and environment for statistical computing and graphics. (Ref: `r-project.org`)
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand
- R is Free
- R can be extended (easily) via .
- R is an interpreted language

R Interpreter / Console / GUI

Demo

About RStudio

- RStudio is a **free and open-source IDE** for R, released by the company **RStudio, Inc.**

- RStudio and its team regularly contribute to R community by releasing new packages, such as:
    - `tidyverse`
    - `shiny`
    - `knitr`

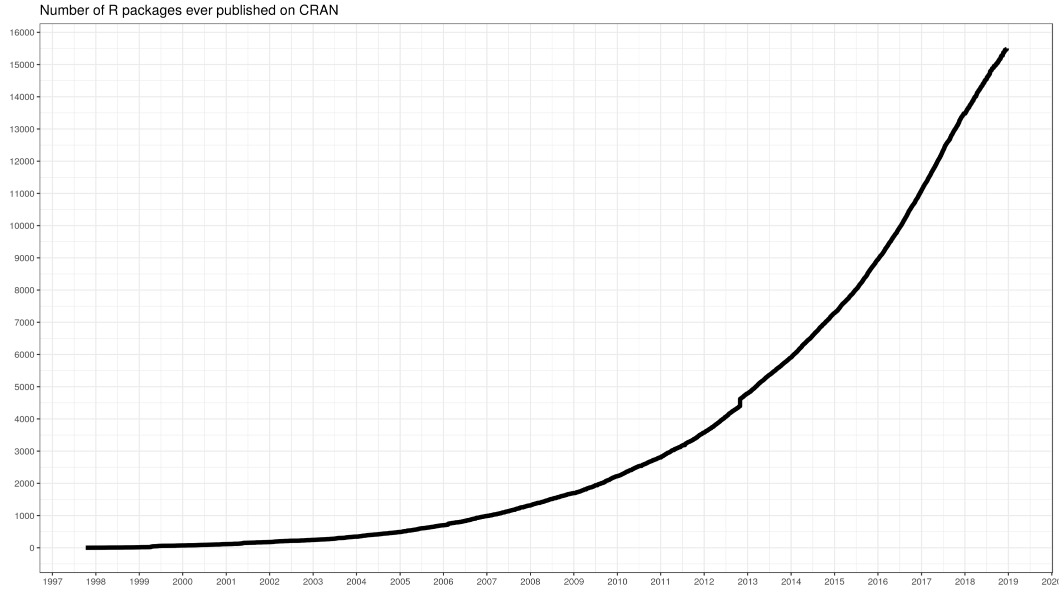
RStudio

Demo

R Ecosystem

Like `Python`, `R`'s strength lies in its Ecosystem. **Why R?** - R Packages

Growth



Source: @daroczig

Basics of R Programming

Hello, World!

The traditional first step - **Hello, World!**:

Hello, World!

The traditional first step - **Hello, World!**:

```
print("Hello, World!")
```

```
## [1] "Hello, World!"
```

> That's one small step for a man, one giant leap for mankind
>
> Neil Armstrong

Arithmetic Operations

```
2 + 3
```

```
## [1] 5
```

```
50000 * 42222
```

```
## [1] 2111100000
```

```
2304 / 233
```

```
## [1] 9.888412
```

```
(33 + 44 ) * 232 / 12
```

```
## [1] 1488.667
```

Assignment Operators

$<-$ Arrow (Less-than < and Minus -)

$=$ (Equal Sign)

```
(x <- 2 + 3)
```

```
## [1] 5
```

```
(y = x ** 4)
```

```
## [1] 625
```

```
5 * 9 -> a
a + 3
```

```
## [1] 48
```

Objects

- The entities R operates on are technically known as `objects`.

Example: Vector of numeric

```
vector_of_numeric <- c(2,4,5)

typeof(vector_of_numeric)
```

```
## [1] "double"
```

Vectors

- Atomic Vectors - Homogeneous Data Type
    - logical
    - integer
    - double
    - character
    -
    -
- Lists - (Recursive Vectors) Heterogeneous Data Type
- `NULL` is used to represent absence of a vector

Vectors + Attributes (Additional Meta Data) = Augmented vectors

- Factors are built on top of integer vectors.
- Dates and date-times are built on top of numeric vectors.
- Data frames and tibbles are built on top of lists.

Numeric Vector

```
nummy <- c(2,3,4)

nummy_int <- c(1L,2L,3L)
```

```
typeof(nummy)
```

```
## [1] "double"
```

```
typeof(nummy_int)
```

```
## [1] "integer"
```

```
is.numeric(nummy)
```

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

```
is.numeric(nummy_int)
```

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

```
is.double(nummy)
```

Character Vector

```
types <- c("int","double","character")

types
```

```
## [1] "int"       "double"    "character"
```

```
typeof(types)
```

```
## [1] "character"
```

```
length(types)
```

```
## [1] 3
```

```
is.numeric(types)
```

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

```
is.character(types)
```

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

Logical Vector

```r
logicals <- c(TRUE,F,TRUE,T, FALSE)

logicals
```

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

Coersion

Typecasting - Explicit

```
money_in_chars <- c("20","35","33")

typeof(money_in_chars)
```

```
## [1] "character"
```

```
money_money <- as.numeric(money_in_chars)

money_money
```

```
## [1] 20 35 33
```

```
typeof(money_money)
```

```
## [1] "double"
```

Typecasting - Implicit

```
money_money <- as.numeric(money_in_chars)

money_money
```

```
## [1] 20 35 33
```

```
typeof(money_money)
```

```
## [1] "double"
```

```
new_money <- c(money_money,"33")

new_money
```

```
## [1] "20" "35" "33" "33"
```

```
typeof(new_money)
```

```
## [1] "character"
```

Vector - Accessing

```r
month.abb #in-built character vector with Month Abbreviations
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

```r
month.abb[2]
```

```
## [1] "Feb"
```

```r
month.abb[4:7]
```

```
## [1] "Apr" "May" "Jun" "Jul"
```

```r
month.abb[c(2,5,7,10)]
```

```
## [1] "Feb" "May" "Jul" "Oct"
```

Vector Manipulation

Appending

```
days <- c("Mon","Tue","Wed")
days
```

```
## [1] "Mon" "Tue" "Wed"
```

```
week_end <- c("Sat","Sun")
more_days <- c(days,"Thu","Fri",week_end)
more_days
```

```
## [1] "Mon" "Tue" "Wed" "Thu" "Fri" "Sat" "Sun"
```

Vector - Arithmetic

```
set.seed(122)

so_many_numbers <- runif(10, min = 10, max = 100)

so_many_numbers
```

```
##  [1] 91.45185 91.61657 27.14995 13.68211 62.11661 66.31451 71.14383
##  [8] 10.25104 13.21914 63.69918
```

```
so_many_numbers * 200
```

```
##  [1] 18290.370 18323.314  5429.989  2736.422 12423.322 13262.902 14228.767
##  [8]  2050.209  2643.828 12739.836
```

Factors

- In R, factors are used to work with categorical variables, variables that have a fixed and known set of possible values.
- Useful with Characters where non-Alphabetical Ordering is required

```
days <- c("Thu","Wed","Sun")

sort(days)
```

```
## [1] "Sun" "Thu" "Wed"
```

```
week_levels <- c("Mon","Tue","Wed","Thu","Fri","Sat","Sun")


(days_f <- factor(days, levels = week_levels))
```

```
## [1] Thu Wed Sun
## Levels: Mon Tue Wed Thu Fri Sat Sun
```

```
sort(days_f)
```

List

Lists are a step up in complexity from atomic vectors: each element can be any type, not just vectors.

```
(a_list <- list("abcd",123,1:12,month.abb))
```

```
## [[1]]
## [1] "abcd"
## 
## [[2]]
## [1] 123
## 
## [[3]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
## 
## [[4]]
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

List Accessing

```
a_list[[1]]
```

```
## [1] "abcd"
```

```
a_list[[4]][4]
```

```
## [1] "Apr"
```

Matrix

```r
new_m <- matrix(data = 1:12, nrow = 3)

new_m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```r
new_m * 20
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   20   80  140  200
## [2,]   40  100  160  220
## [3,]   60  120  180  240
```

```r
dim(new_m)
```

```
## [1] 3 4
```

Dataframe

Tabular Structure

- dimension
- row.names
- col.names

```
colleges <- c("CIT","GCT","PSG")

year <- c(2019,2018,2017)

db <- data.frame(college_names = colleges, year_since = year)

db
```

```
##   college_names year_since
## 1           CIT       2019
## 2           GCT       2018
## 3           PSG       2017
```

Dataframe Manipulation

```
db$college_names
```

```
## [1] CIT GCT PSG
## Levels: CIT GCT PSG
```

```
db[2,2] <- 1990

db
```

```
##   college_names year_since
## 1           CIT       2019
## 2           GCT       1990
## 3           PSG       2017
```

```
db[,"year_since"]
```

```
## [1] 2019 1990 2017
```

Loops & Iterators

For Loop

```
    (month_name    month.abb[1:4]) {
  print(paste("This month", month_name, "beautiful!!!"))
}
```

```
## [1] "This month Jan beautiful!!!"
## [1] "This month Feb beautiful!!!"
## [1] "This month Mar beautiful!!!"
## [1] "This month Apr beautiful!!!"
```

As you move forward, Check the family of `apply` functions - `sapply()`, `tapply()`, `lapply()`, `apply()`.

For advanced functional programming, refer `purrr` package

Logical Operations

%in% operator

```
iris$Species %  % "virginica"
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [111]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [122]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [144]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Logical Operators

```
1:10 > 5
```

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

```
1:10 == 4
```

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

```
!1:10 == 4
```

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

Conditions

```
   (iris$Sepal.Length[2]>5) {
  print("it is gt 5")
}      print("it is not")
```

```
## [1] "it is not"
```

```
   (iris$Sepal.Length>10) {print("hello")}
```

```
## Warning in if (iris$Sepal.Length > 10) {: the condition has length > 1 and
## only the first element will be used
```

```
ifelse(iris$Sepal.Length>6, "more_than_10","les_than_10")
```

```
##   [1] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##   [5] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##   [9] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##  [13] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##  [17] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##  [21] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
##  [25] "les_than_10"  "les_than_10"  "les_than_10"  "les_than_10"
```

Functions

Types

- Base-R functions (`mean()`, `plot()`, `lm()`)
- Package functions (`dplyr::mutate()`, `stringr::str_detect()`)
- User-defined functions

```
workshop_hate_message <-          (name = "No one", n = 3) {
  text_to_print <- paste(name, "hate(s)", "this workshop")
      (i    1:n) {
    print(text_to_print)
  }

}

workshop_hate_message("All of us",4)
```

```
## [1] "All of us hate(s) this workshop"
## [1] "All of us hate(s) this workshop"
## [1] "All of us hate(s) this workshop"
## [1] "All of us hate(s) this workshop"
```

Packages

Package Installation & Loading

From CRAN (usually Stable Version)

```
install.packages("itunesr")
```

From Github (usually Development Version)

```
#install.packages("devtools")
devtools::install_github("amrrs/itunesr")
```

Loading

```
       (itunesr)
```

Help

using `help()`

```
help("runif")
```

using `?`

```
?sample
```

Help - Example

```r
example("for")
```

```
##
## for> for(i in 1:5) print(1:i)
## [1] 1
## [1] 1 2
## [1] 1 2 3
## [1] 1 2 3 4
## [1] 1 2 3 4 5
##
## for> for(n in c(2,5,10,20,50)) {
## for+    x <- stats::rnorm(n)
## for+    cat(n, ": ", sum(x^2), "\n", sep = "")
## for+ }
## 2: 2.188171
## 5: 1.936692
## 10: 15.34038
## 20: 25.59841
## 50: 49.75875
##
## for> f <- factor(sample(letters[1:5], 10, replace = TRUE))
##
```

Packages Vignette

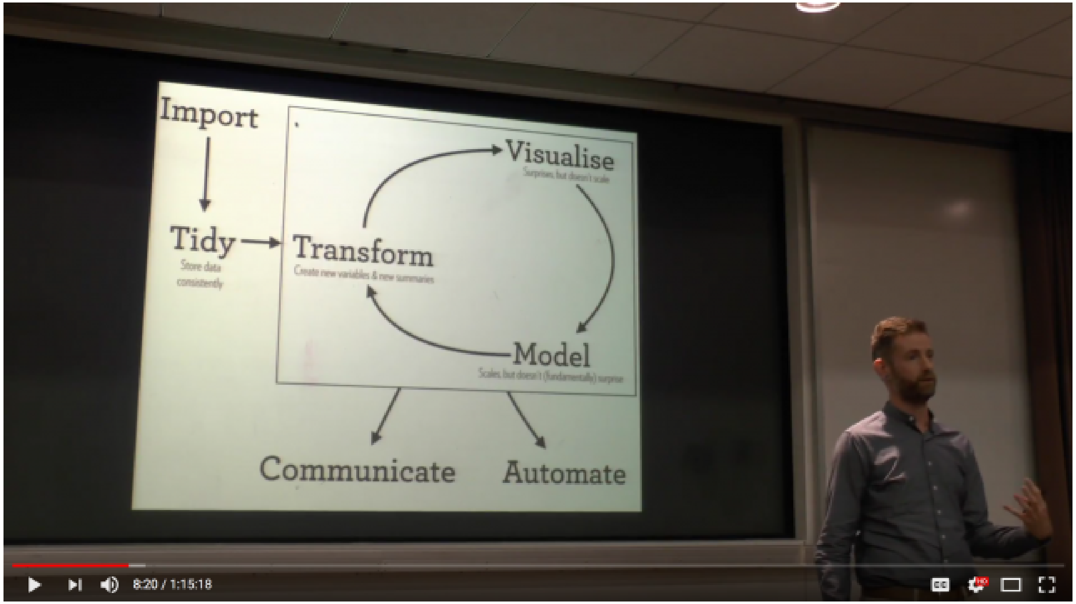
```
vignette("dplyr")

browseVignettes("dplyr")
```

Data wrangling and Visualization using Tidyverse

Data Science Framework

There are now like, you know, a billion venn diagrams showing you what data science is. But to me I think the definition is pretty simple. Whenever you're struggling with data, trying to understand what's going on with data, whenever you're trying to turn that **raw data into insight and understanding and discoveries**. I think that's **Data Science.**" - Hadley Wickham


Import
Tidy
Store data consistently
Transform
Create new variables & new summaries
Visualise
Surprises, but doesn't scale
Model
Scales, but doesn't (fundamentally) surprise
Communicate
Automate
8:20 / 1:15:18

Source: Hadley Wickham

Tidyverse

- An opinionated collection of R packages designed for data science.
- All packages share an underlying design

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

tidyverse packages

```
tidyverse::tidyverse_packages()
```

```
##  [1] "broom"         "cli"           "crayon"        "dplyr"         "dbplyr"
##  [6] "forcats"       "ggplot2"       "haven"         "hms"           "httr"
## [11] "jsonlite"      "lubridate"     "magrittr"      "modelr"        "purrr"
## [16] "readr"         "readxl\n(>=" "reprex"        "rlang"         "rstudioapi"
## [21] "rvest"         "stringr"       "tibble"        "tidyr"         "xml2"
## [26] "tidyverse"
```

Loading the Library

```
        (tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────────────────── tic

## ✔ ggplot2 3.1.0     ✔ purrr   0.2.5
## ✔ tibble  2.0.0     ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2     ✔ stringr 1.3.1
## ✔ readr   1.3.1     ✔ forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.2

## ── Conflicts ──────────────────────────────────────────────────── tidyverse
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Input Data

Reading the dataset

```
#kaggle <- read_csv("data/kaggle_survey_2018.csv")

kaggle <- read_csv("data/kaggle_survey_2018.csv", skip = 1)
```

Basic Stats

Dimension (Rows Column)

```
dim(kaggle)
```

```
## [1] 23859   395
```

```
glimpse(kaggle)
```

```
## Observations: 23,859
## Variables: 395
## $ `Duration (in seconds)`
## $ `What is your gender? - Selected Choice`
## $ `What is your gender? - Prefer to self-describe - Text`
## $ `What is your age (# years)?`
## $ `In which country do you currently reside?`
## $ `What is the highest level of formal education that you have attained or
## $ `Which best describes your undergraduate major? - Selected Choice`
## $ `Select the title most similar to your current role (or most recent titl
## $ `Select the title most similar to your current role (or most recent titl
## $ `In what industry is your current employer/contract (or your most recent
```

Dataset Overview

Demo on RStudio

Data Questions (Business Problem)

- What's the percentage of Male and Female respondents?
- What are the top 5 countries?

dyplr verbs

- `mutate()` - adds new variables that are functions of existing variables
- `select()` - picks variables based on their names.
- `filter()` - picks cases based on their values.
- `summarise()` - reduces multiple values down to a single summary.
- `arrange()` - changes the ordering of the rows.

Introducing %>% Pipe Operator

- The pipe, `%>%`, comes from the magrittr package by Stefan Milton Bache
- **Output of LHS** is given as the **input (first argument) of RHS**

Example

```
kaggle %>% dim()
```

```
## [1] 23859   395
```

Although doesn't make much sense to use `%>%` in this context, Hope it explains the function.

Percentage of Male and Female

- Column name - `What is your gender? - Selected Choice`

Pseudo-code

- `group_by` the `kaggle` dataframe on column `What is your gender? - Selected Choice`
- `count` the values
- calculate `percentage` value from the `counts`

% of Male and Female - Group By & Count - Method 1

```
kaggle %>%
  group_by(`What is your gender? - Selected Choice`) %>%
  summarise(n = n())
```

```
## # A tibble: 4 x 2
##   `What is your gender? - Selected Choice`     n
##   <chr>                                    <int>
## 1 Female                                    4010
## 2 Male                                     19430
## 3 Prefer not to say                          340
## 4 Prefer to self-describe                     79
```

% of Male and Female - Group By & Count - Method 2

```
kaggle %>%
  group_by(`What is your gender? - Selected Choice`) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   What is your gender? - Selected Choice [4]
##   `What is your gender? - Selected Choice`     n
##   <chr>                                    <int>
## 1 Female                                    4010
## 2 Male                                     19430
## 3 Prefer not to say                          340
## 4 Prefer to self-describe                     79
```

% of Male and Female - Group By & Count - Sorted

```
kaggle %>%
  group_by(`What is your gender? - Selected Choice`) %>%
  count() %>%
  arrange(desc(n))
```

```
## # A tibble: 4 x 2
## # Groups:   What is your gender? - Selected Choice [4]
##   `What is your gender? - Selected Choice`     n
##   <chr>                                    <int>
## 1 Male                                     19430
## 2 Female                                    4010
## 3 Prefer not to say                          340
## 4 Prefer to self-describe                     79
```

% of Male and Female - Percentage

```r
kaggle %>%
  group_by(`What is your gender? - Selected Choice`) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = round(n / sum(n),2))
```

```
## # A tibble: 4 x 3
##   `What is your gender? - Selected Choice`     n  perc
##   <chr>                                    <int> <dbl>
## 1 Female                                    4010  0.17
## 2 Male                                     19430  0.81
## 3 Prefer not to say                          340  0.01
## 4 Prefer to self-describe                     79  0
```

% of Male and Female - Nice_Looking_Table

```
kaggle %>%
  group_by(`What is your gender? - Selected Choice`) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = round(n / sum(n),2)) %>%
  knitr::kable(format = "html")
```

What is your gender? - Selected Choice	n	perc
Female	4010	0.17
Male	19430	0.81
Prefer not to say	340	0.01
Prefer to self-describe	79	0.00

But, Wait!!!

Go Back and See

If you have only `Male` and `Female`?

Time for some cleaning

In the form of `filter()`ing

% of Male and Female - Filtered_Nice

```
kaggle %>%
  filter(`What is your gender? - Selected Choice` %  % c("Male","Fema
  group_by(`What is your gender? - Selected Choice`) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = round(n / sum(n),2)) %>%
  knitr::kable(format = "html")
```

What is your gender? - Selected Choice	n	perc
Female	4010	0.17
Male	19430	0.83

An Awkward column name, isn't it??!

% of Male and Female - All_Nice_Table

```
       (scales) #for Percentage Formatting
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
kaggle %>%
  filter(`What is your gender? - Selected Choice` %  % c("Male","Fema
  group_by(`What is your gender? - Selected Choice`) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = round(n / sum(n),2)) %>%
  mutate(perc = scales::percent(perc)) %>%
  rename(Gender = `What is your gender? - Selected Choice`,
```

Top 5 Countries

- Column name - `In which country do you currently reside?`

Pseudo-code

- `count` number of respondents from each country
- `arrange` countries in descending order based on their count value
- `top 5` in the list is the output

Top 5 Countries - Code

```
kaggle %>%
  count(`In which country do you currently reside?`) %>%
  arrange(desc(n)) %>%
  top_n(5) %>%
  knitr::kable(format = "html")
```

```
## Selecting by n
```

In which country do you currently reside?	n
United States of America	4716
India	4417
China	1644
Other	1036
Russia	879

Is `Other` a country name???

Top 5 Countries

```
kaggle %>%
  filter(!`In which country do you currently reside?` %  % "Other") %
  count(`In which country do you currently reside?`) %>%
  rename(Country = `In which country do you currently reside?`) %>%
  arrange(desc(n)) %>%
  top_n(5) %>%
  knitr::kable(format = "html")
```

```
## Selecting by n
```

Country	n
United States of America	4716
India	4417
China	1644
Russia	879
Brazil	736

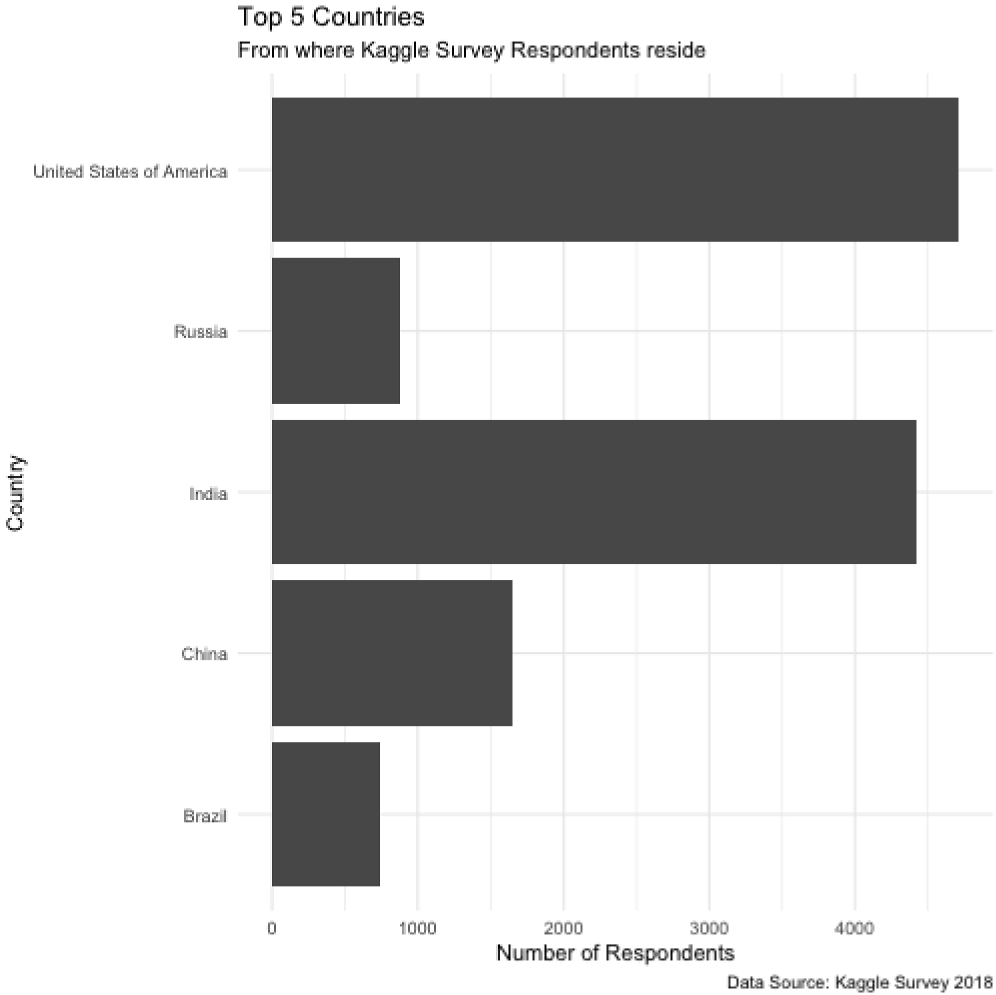
Table is nice, but a visually appealing plot is Nicer

😉

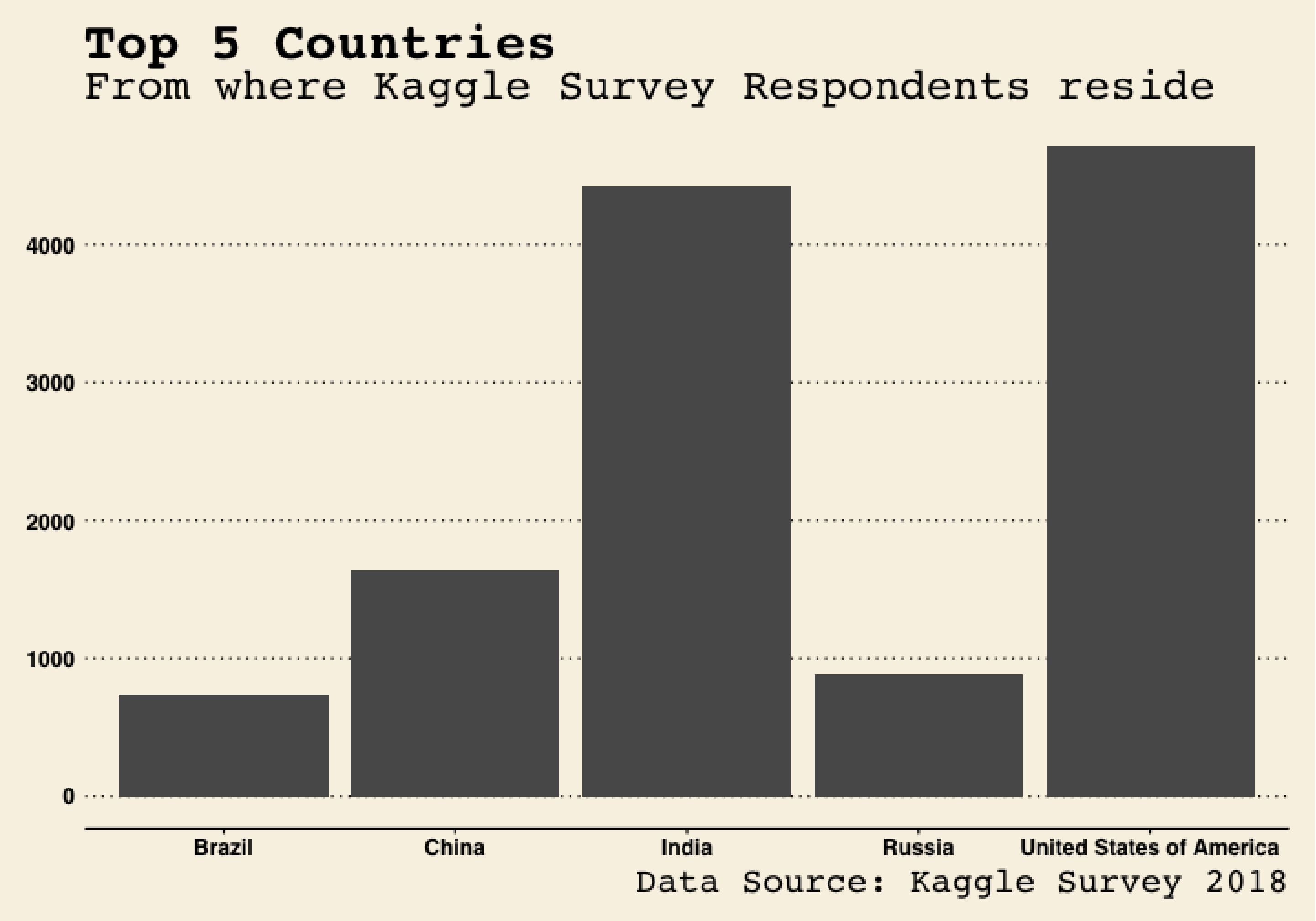
Top 5 Countries - Plot #1

```
kaggle %>%
  filter(!`In which country do you currently reside?` %  % "Other") %
  count(`In which country do you currently reside?`) %>%
  rename(Country = `In which country do you currently reside?`) %>%
  arrange(desc(n)) %>%
  top_n(5) %>%
  ggplot() + geom_bar(aes(Country,n), stat = "identity") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Top 5 Countries",
       subtitle = "From where Kaggle Survey Respondentns reside",
       x = "Country",
       y = "Number of Respondents",
       caption = "Data Source: Kaggle Survey 2018")
```

Top 5 Countries - Plot #2



Top 5 Countries - Plot #3 Themed



Documentation and Reporting using R Markdown

Demo

Project Demo

Object Detection in 3 Lines of R Code

using Tiny YOLO

-Project Demo-

References

- R for Data Science
- R-Bloggers

Thanks!

Slides created via the R package **xaringan**.

The chakra comes from remark.js, **knitr**, and R Markdown.