Getting Started

R-Ladies El Paso

Elise Bell

Welcome!

Today we will

- learn about R and RStudio [1]
- load some data
- examine and manipulate the data

[1]: h/t to Mine Cetinkaya-Rundel for slideshow inspiration (http://rpubs.com/minebocek/rladies-dplyr-tidyr)

R and RStudio

- R is the programming language
- RStudio is an environment that makes using R easier
- Free and open-source!

Getting started

- Install R: https://cran.r-project.org/
- Install RStudio: https://www.rstudio.com/products/RStudio/#Desktop

Anatomy of RStudio

- Left: Console
    - Text on top at launch: version of R that you're running
    - Below that is the prompt
- Upper right: Workspace and command history
- Lower right: Plots, access to files, help, packages, data viewer

What version am I using?

- The version of R is in the text that pops up in the Console when you start RStudio
- To find out the version of RStudio go to Help > About RStudio
- It's good practice to keep both R and RStudio up to date

R packages

- Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and (often) sample data. (From: http://r-pkgs.had.co.nz)

- Install these packages by running the following in the Console:

```
install.packages("readr")
install.packages("tidyr")
install.packages("dplyr")
install.packages("ggplot2")
```

- Then, load the packages by running the following:

```
library(readr)
library(tidyr)
library(dplyr)
#library(ggplot2)
```

--- .class #id

R Script

- Type your code in the R Script
- Use cursor + Run or highlight + Run
- Shortcut for Run button: Command + Enter
- Use # for comments

R Markdown

- R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R.
- R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).
- Code goes in code 'chunks'
- Comments can go in the chunks with # or just as plain text outside the chunks

Source: http://rmarkdown.rstudio.com/

Where are you?

```
# Find working directory
getwd()
```

```
## [1] "/Users/elisebell1/Documents/GitHub/rladies-ep-intro"
```

```
# Change working directory
setwd("~/Documents/GitHub/rladies-ep-intro/")
# Check
#getwd()
# What is in your working directory?
list.files()
```

```
## [1] "assets"                "index.html"
## [3] "index.md"              "index.Rmd"
## [5] "libraries"             "marvel-wikia-data.csv"
## [7] "marvel-wikia-data1.csv" "movies.csv"
## [9] "StarWars.csv"
```

9/39

Loading data

Comic book characters (FiveThirtyEight https://fivethirtyeight.com/features/women-in-comic-books/)

```
data <- read.csv("marvel-wikia-data1.csv")
# You can download the data from https://github.com/fivethirtyeight/
# data/blob/master/comic-characters/marvel-wikia-data.csv
names(data)
```

```
##  [1] "name"            "urlslug"          "ID"
##  [4] "ALIGN"           "EYE"              "HAIR"
##  [7] "SEX"             "GSM"              "ALIVE"
## [10] "APPEARANCES"     "FIRST.APPEARANCE" "Year"
```

Other ways to load data

- `read.csv()` converts character vectors into factors, which you may or may not want
- You can also use File --> Import Dataset

Viewing your data

Click the name of the data frame in the Environment (top right)

Use the `str()` function to compactly display the internal structure of an R object

```
str(data)
```

```
## 'data.frame':    16376 obs. of  12 variables:
##  $ name           : Factor w/ 16376 levels "'Spinner (Earth-616)",..: 13958 2331 16000 677
##  $ urlslug        : Factor w/ 16376 levels "\\/%22Spider-Girl%22_(Mutant\\/Spider_Clone)_(
##  $ ID             : Factor w/ 5 levels "","Known to Authorities Identity",..: 5 4 4 4 3 4
##  $ ALIGN          : Factor w/ 4 levels "","Bad Characters",..: 3 3 4 3 3 3 3 3 4 3 ...
##  $ EYE            : Factor w/ 25 levels "","Amber Eyes",..: 11 5 5 5 5 5 6 6 6 5 ...
##  $ HAIR           : Factor w/ 26 levels "","Auburn Hair",..: 8 25 4 4 5 15 8 8 8 5 ...
##  $ SEX            : Factor w/ 5 levels "","Agender Characters",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ GSM            : Factor w/ 7 levels "","Bisexual Characters",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ALIVE          : Factor w/ 3 levels "","Deceased Characters",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ APPEARANCES    : int  4043 3360 3061 2961 2258 2255 2072 2017 1955 1934 ...
##  $ FIRST.APPEARANCE: Factor w/ 833 levels "","1-Apr","1-Aug",..: 226 554 752 572 675 683 68
##  $ Year           : int  1962 1941 1974 1963 1950 1961 1961 1962 1963 1961 ...
```

Viewing your data

Use the `glimpse()` function to see all variables and the data in them

```
glimpse(data)
```

```
## Observations: 16,376
## Variables: 12
## $ name             <fct> Spider-Man (Peter Parker), Captain America (Ste…
## $ urlslug          <fct> \/Spider-Man_(Peter_Parker), \/Captain_America_…
## $ ID               <fct> Secret Identity, Public Identity, Public Identi…
## $ ALIGN            <fct> Good Characters, Good Characters, Neutral Chara…
## $ EYE              <fct> Hazel Eyes, Blue Eyes, Blue Eyes, Blue Eyes, Bl…
## $ HAIR             <fct> Brown Hair, White Hair, Black Hair, Black Hair,…
## $ SEX              <fct> Male Characters, Male Characters, Male Characte…
## $ GSM              <fct> , , , , , , , , , , , , , , , , , , , , , , , ,
## $ ALIVE            <fct> Living Characters, Living Characters, Living Ch…
## $ APPEARANCES      <int> 4043, 3360, 3061, 2961, 2258, 2255, 2072, 2017,…
## $ FIRST.APPEARANCE <fct> Aug-62, Mar-41, Oct-74, Mar-63, Nov-50, Nov-61,…
## $ Year             <int> 1962, 1941, 1974, 1963, 1950, 1961, 1961, 1962,…
```

Tidy data

- In tidy data

  1. Each variable forms a column
  2. Each observation forms a row
  3. Each type of observational unit forms a table

- Messy data is any other other arrangement of the data

  - `tidyr` package is helpful for converting messy data to tidy data

- We'll start with some tidy data

Data manipulation

- Using base R functions
- Using the `tidyr` and `dplyr` packages < our focus today

Verbs of `dplyr`

The `dplyr` package is based on the concepts of functions as verbs that manipulate data frames:

- `filter()`: pick rows matching criteria
- `select()`: pick columns by name
- `rename()`: rename specific columns
- `arrange()`: reorder rows
- `mutate()`: add new variables
- `transmute()`: create new data frame with variables
- `sample_n()` / `sample_frac()`: randomly sample rows
- `summarize()`: reduce variables to values

`dplyr` rules

- First argument is a data frame
- Subsequent arguments say what to do with data frame
- Always returns a data frame
- Avoid modify in place

Filter rows with `filter()`

- Select a subset of rows in a data frame.
- Easily filter for many conditions at once.

Filter for 'good' characters

```
data %>%
  filter(ALIGN == "Good Characters") %>%
  head() # just show the first 6 rows
```

```
##                                 name
## 1           Spider-Man (Peter Parker)
## 2     Captain America (Steven Rogers)
## 3 Iron Man (Anthony \\"Tony\\" Stark)
## 4                 Thor (Thor Odinson)
## 5          Benjamin Grimm (Earth-616)
## 6           Reed Richards (Earth-616)
##                                 urlslug                ID           ALIGN
## 1           \\/Spider-Man_(Peter_Parker)  Secret Identity Good Characters
## 2     \\/Captain_America_(Steven_Rogers)  Public Identity Good Characters
```

Filter rows with `filter()`

- Select a subset of rows in a data frame.
- Easily filter for many conditions at once.

Filter for 'good' characters introduced after 2000

```
data %>%
  filter(ALIGN == "Good Characters", Year > 2000) %>%
  head() # just show the first 6 rows
```

```
##                          name                          urlslug
## 1       Maria Hill (Earth-616)       \\/Maria_Hill_(Earth-616)
## 2     Laura Kinney (Earth-616)     \\/Laura_Kinney_(Earth-616)
## 3   Santo Vaccarro (Earth-616)   \\/Santo_Vaccarro_(Earth-616)
## 4      Megan Gwynn (Earth-616)      \\/Megan_Gwynn_(Earth-616)
## 5     Hope Summers (Earth-616)     \\/Hope_Summers_(Earth-616)
## 6 Victor Borkowski (Earth-616) \\/Victor_Borkowski_(Earth-616)
##                  ID           ALIGN            EYE       HAIR
## 1 No Dual Identity Good Characters     Brown Eyes Black Hair
## 2  Secret Identity Good Characters     Green Eyes Black Hair
```

18/39

Commonly used logical operators in R

OPERATOR	DEFINITION
`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to
`==`	exactly equal to
`!=`	not equal to
`x | y`	`x` OR `y`
`x & y`	`x` AND `y`

Commonly used logical operators in R

OPERATOR	DEFINITION
`is.na(x)`	test if `x` is `NA`
`!is.na(x)`	test if `x` is not `NA`
`x %in% y`	test if `x` is in `y`
`!(x %in% y)`	test if `x` is not in `y`
`!x`	not `x`

Real data is messy

Careful data scientists clean up their data first!

- You may need to do some text parsing to clarify your data
    - `Good Characters` can be `good`
    - `Bad Characters` can be `bad`
    - `Neutral Characters` can be `neutral`
- New R package: `stringr`

Install and load: `stringr`

```
#Install
install.packages("stringr")
# load package
library(stringr)
```

- Package reference: Most R packages come with a vignette that describes in detail what each function does and how to use them, they're incredibly useful resources (in addition to other worked out examples on the web) https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html

`rename()` specific columns

Correct typos and rename to make variable names shorter and/or more informative

```
# Original names
names(data)
```

```
##  [1] "name"            "urlslug"          "ID"
##  [4] "ALIGN"           "EYE"              "HAIR"
##  [7] "SEX"             "GSM"              "ALIVE"
## [10] "APPEARANCES"     "FIRST.APPEARANCE" "Year"
```

```
# Rename `ALIGN` to `align`
data <- data %>%
  rename(align = ALIGN)

# Outside of the tidyverse, you can also do this:
oldnames <- names(data)
names(data) <- tolower(oldnames)
```

Replace variables with `str_replace()` and add new variables with `mutate()`

```
data <- data %>%
  mutate(align = str_replace(align, "Good Characters", "good")) %>%
  mutate(align = str_replace(align, "Bad Characters", "bad")) %>%
  mutate(align = str_replace(align, "Neutral Characters", "neutral"))
```

- Note that we're overwriting existing data and columns, so be careful!
    - But remember, it's easy to revert if you make a mistake since we didn't touch the raw data, we can always reload it and start over

Check before you move on

```
data %>%
  group_by(align) %>%
  summarize(count = n())
```

```
## # A tibble: 4 x 2
##   align   count
##   <chr>   <int>
## 1 ""       2812
## 2 bad      6720
## 3 good     4636
## 4 neutral  2208
```

Check before you move on

```
# You can also count using the `count()` function
data %>%
  group_by(align) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   align [4]
##   align       n
##   <chr>   <int>
## 1 ""       2812
## 2 bad      6720
## 3 good     4636
## 4 neutral  2208
```

Summary statistics

We can use the same format for calculating other summary statistics

```
data %>%
  group_by(sex) %>%
  summarize(mean_appearances = mean(appearances, na.rm = TRUE),
            median_appearances = median(appearances, na.rm = TRUE))
```

```
## # A tibble: 5 x 3
##   sex                    mean_appearances median_appearances
##   <fct>                             <dbl>              <dbl>
## 1 ""                                 4.43                2
## 2 Agender Characters                19.7                 9.5
## 3 Female Characters                 20.3                 4
## 4 Genderfluid Characters           282.                282.
## 5 Male Characters                   16.8                 3
```

`slice()` for certain row numbers

```
# First five
data %>%
  slice(1:5)
```

```
##                                  name
## 1             Spider-Man (Peter Parker)
## 2       Captain America (Steven Rogers)
## 3 Wolverine (James \\"Logan\\" Howlett)
## 4   Iron Man (Anthony \\"Tony\\" Stark)
## 5                  Thor (Thor Odinson)
##                                  urlslug                id   align
## 1             \\/Spider-Man_(Peter_Parker)  Secret Identity    good
## 2       \\/Captain_America_(Steven_Rogers)  Public Identity    good
## 3 \\/Wolverine_(James_%22Logan%22_Howlett)  Public Identity neutral
## 4   \\/Iron_Man_(Anthony_%22Tony%22_Stark)  Public Identity    good
## 5                  \\/Thor_(Thor_Odinson) No Dual Identity    good
##          eye       hair             sex gsm             alive appearances
## 1 Hazel Eyes Brown Hair Male Characters     Living Characters        4043
## 2  Blue Eyes White Hair Male Characters     Living Characters        3360
```

select()

```
# Include only specific variables
data %>%
  select(align, sex) %>%
  table()
```

```
##          sex
## align          Agender Characters Female Characters Genderfluid Characters
##              232                   2                684                     0
##   bad        386                  20                976                     0
##   good       122                  10               1537                     1
##   neutral    114                  13                640                     1
##          sex
## align     Male Characters
##                      1894
##   bad                5338
##   good               2966
##   neutral            1440
```

29/39

select()

```
# Exclude variable (column)
# data %>%
#   select(-eye)

# To keep only certain levels of a factor (and get rid of empty levels)
data %>%
  select(sex, align) %>%
  table()
```

```
##                          align
## sex                          bad good neutral
##                          232  386  122     114
##   Agender Characters       2   20   10      13
##   Female Characters      684  976 1537     640
##   Genderfluid Characters   0    0    1       1
##   Male Characters       1894 5338 2966    1440
```

select()

```
data %>%
  select(sex, align) %>%
  filter(sex != "", align != "") %>%
  droplevels() %>%
  table()
```

```
##                          align
## sex                        bad good neutral
##   Agender Characters        20   10      13
##   Female Characters        976 1537     640
##   Genderfluid Characters     0    1       1
##   Male Characters         5338 2966    1440
```

`summarize()` in a new data frame with new calculated variables

```
data.summ <- data %>%
  filter(sex != "", align != "") %>%
  group_by(sex, year, align) %>%
  summarize(n = n()) %>%
  mutate(perc.sex = n / sum(n))
```

`summarize()` by decade

```
# Write a function to round years to their decade
get_decade = function(year){ return(floor(year / 10) * 10) }

# Re-do data.summ
data.summ <- data %>%
  filter(sex != "", align != "") %>%
  mutate(decade = get_decade(year)) %>%
  group_by(sex, decade, align) %>%
  summarize(n = n())
```

Select rows with `sample_n()` or `sample_frac()`

```
# `sample_n()`: randomly sample 5 observations
data_n5 <- data %>%
  sample_n(5, replace = FALSE)
dim(data_n5)
```

```
## [1]  5 12
```

```
# `sample_frac()`: randomly sample 20% of observations
data_perc20 <-data %>%
  sample_frac(0.2, replace = FALSE)
dim(data_perc20)
```

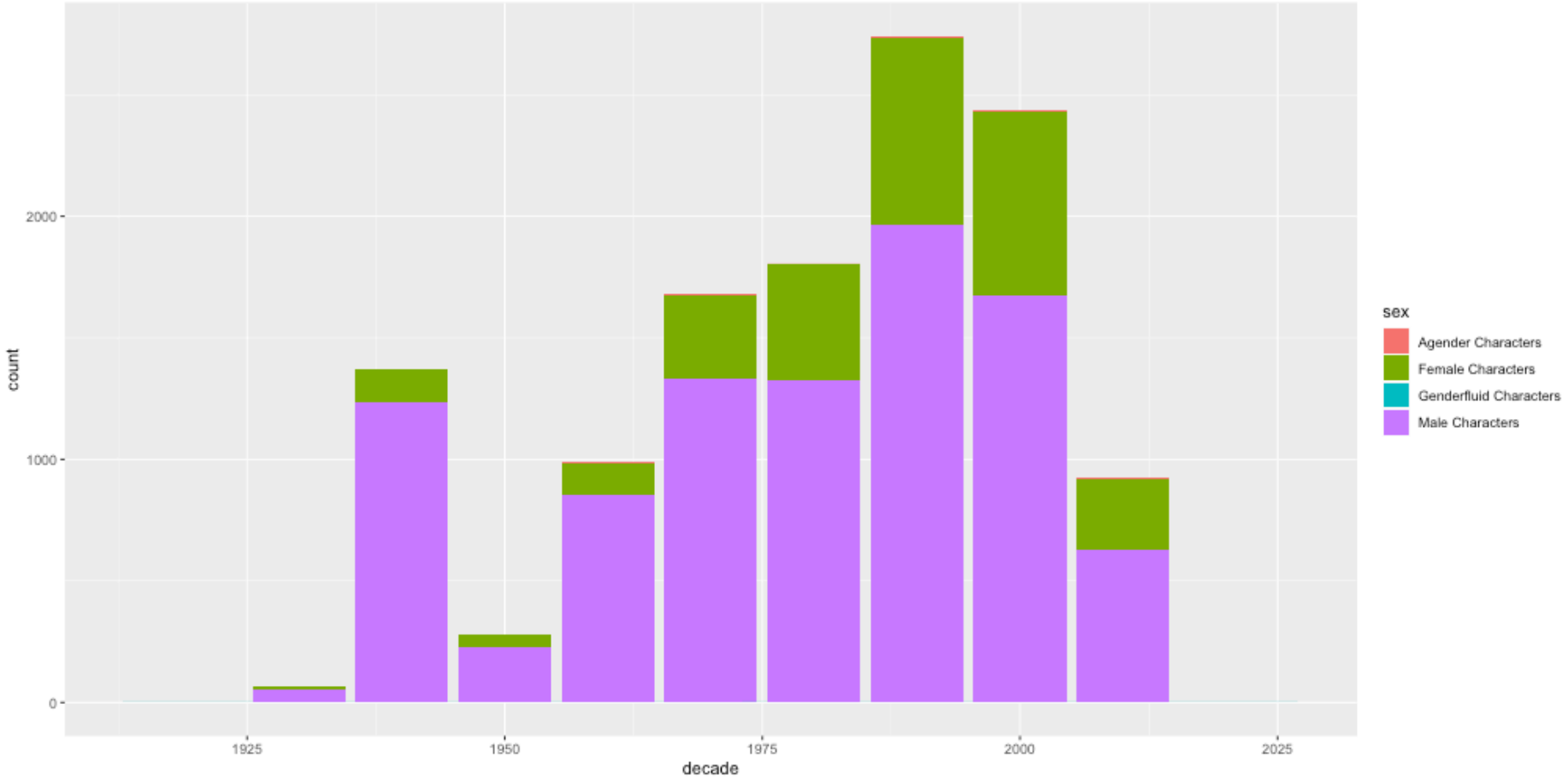
```
## [1] 3275   12
```

Simple plotting

```
# load ggplot2
library(ggplot2)

decade.plot <- data %>%
  mutate(decade=get_decade(year)) %>%
  filter(sex != "", align != "") %>%
  ggplot(aes(x=decade, fill=sex)) + geom_bar()
```

```
decade.plot
```



36/39

More `dplyr` resources

- Visit https://cran.r-project.org/web/packages/dplyr/vignettes/introduction.html for the package vignette.
- Refer to the `dplyr` cheatsheet.

Basic R syntax

For when not working with `dplyr`

- Refer to a variable in a dataset as `data$name`
- Access any element in a dataframe using square brackets

```
data[1,5] # row 1, column 5
```

```
## [1] Hazel Eyes
## 25 Levels:  Amber Eyes Black Eyeballs Black Eyes Blue Eyes ... Yellow Eyes
```

- For all observations in row 1: `data[1, ]`
- For all observations in column 5: `data[, 5]`

Other things to learn: how to make a dataframe, how to add rows/columns

Want more R?

- Resources for learning R:
  - Coursera
  - DataCamp
  - Many many online demos, resources, examples, as well as books
- Debugging R errors:
  - Read the error!
  - StackOverflow
- Keeping up with what's new in R land:
  - R-bloggers
  - Twitter: #rstats