# Week-2: Introduction to Data & Visualization

## NM2207: Computational Media Literacy

**Narayani Vedam, Ph.D.**

**Department of Communications and New Media**

NUS National University of Singapore | Faculty of Arts & Social Sciences

# This week

# Table of contents

# I. Introduction to data

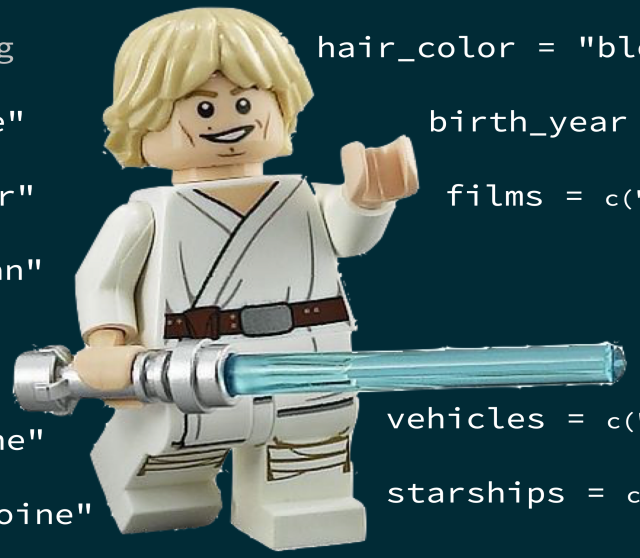
# What constitutes a data-set?

```r
# Load R packages for data science
library(tidyverse)
# Data in starwars data-set
starwars
```

```
## # A tibble: 87 × 14
##    name      height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>      <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr> <chr>
##  1 Luke Sk…    172    77 blond      fair       blue             19   male  mascu…
##  2 C-3PO       167    75 <NA>       gold       yellow          112   none  mascu…
##  3 R2-D2        96    32 <NA>       white, bl… red              33   none  mascu…
##  4 Darth V…    202   136 none       white      yellow           41.9 male  mascu…
##  5 Leia Or…    150    49 brown      light      brown            19   fema… femin…
##  6 Owen La…    178   120 brown, gr… light      blue             52   male  mascu…
##  7 Beru Wh…    165    75 brown      light      blue             47   fema… femin…
##  8 R5-D4        97    32 <NA>       white, red red              NA   none  mascu…
##  9 Biggs D…    183    84 black      light      brown            24   male  mascu…
## 10 Obi-Wan…    182    77 auburn, w… fair       blue-gray        57   male  mascu…
## # ℹ 77 more rows
## # ℹ 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

# Luke Skywalker

```
height = 172 cm
weight = 77 kg
eye_color = "blue"
skin_color = "fair"
species = "Human"
sex = "male"
gender = "masculine"
homeworld = "Tatooine"
name = "Luke Skywalker"
hair_color = "blond"
birth_year = 19 BBY
films = c("The Empire Strikes Back",
          "Revenge of the Sith",
          "Return of the Jedi",
          "A New Hope",
          "The Force Awakens")
vehicles = c("Snowspeeder",
             "Imperial Speeder Bike")
starships = c("X-wing",
              "Imperial shuttle")
```



Source: https://datasciencebox.org/

# Overview of the data-set

question mark operator: seek to elicit more info

```
# Salient features of the data-set
?starwars
```

if you use question mark before command, it always tells you greater details abt how command can be used

starwars {dplyr} R Documentation

## Starwars characters

### Description

The original data, from SWAPI, the Star Wars API, https://swapi.dev/, has been revised to reflect additional research into gender and sex determinations of characters.

### Usage

```
starwars
```

### Format

A tibble with 87 rows and 14 variables:

name
: Name of the character

height
: Height (cm)

mass
: Weight (kg)

hair_color,skin_color,eye_color
: Hair, skin, and eye colors

birth_year
: Year born (BBY = Before Battle of Yavin)

sex
: The biological sex of the character, namely male, female, hermaphroditic, or none (as in the case for Droids).

gender
: The gender role or gender identity of the character as determined by their personality or the way they were programmed (as in the case for Droids).

# Dissecting the data-set: a glimpse

```
# Catch a glimpse starwars data-set
glimpse(starwars)
```

**glimpse() command: glimpse followed by the dataset within parenthesis; glimpse gives us an alternative representation with the columns stacked one below another, followed by you type, and then the values for these various different attributes**

```
## Rows: 87
## Columns: 14
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or…
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2…
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.…
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N…
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "…
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",…
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, …
## $ sex        <chr> "male", "none", "none", "male", "female", "male", "female",…
## $ gender     <chr> "masculine", "masculine", "masculine", "masculine", "femini…
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T…
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma…
## $ films      <list> <"The Empire Strikes Back", "Revenge of the Sith", "Return…
## $ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <>, <>, <>, "Imp…
## $ starships  <list> <"X-wing", "Imperial shuttle">, <>, <>, "TIE Advanced x1",…
```

# Dissecting the data-set: columns

> Each **column** is a variable

Notice the *type* of each variable in the previous slide. They are,

1. `chr` (**character** for text-based entries)

2. `int` (**integer** for whole numbers)

3. `dbl` (**double** for decimal numbers)

4. `list` (**list** for entries of more than one type `chr`, `int`, `dbl`)

# Dissecting the data-set: column of type integer

Let us look at the entries of one of the columns of type `int`

```
# Access column "height"
starwars$height
```

```
##  [1] 172 167  96 202 150 178 165  97 183 182 188 180 228 180 173 175 170 180  66
## [20] 170 183 200 190 177 175 180 150  NA  88 160 193 191 170 196 224 206 183 137
## [39] 112 183 163 175 180 178  94 122 163 188 198 196 171 184 188 264 188 196 185
## [58] 157 183 183 170 166 165 193 191 183 168 198 229 213 167  79  96 193 191 178
## [77] 216 234 188 178 206  NA  NA  NA  NA  NA 165
```

# Dissecting the data-set: column of type double

Let us look at the entries of one of the columns of type `dbl`

```
# Access column "mass"
starwars$mass
```

```
##  [1]   77.0   75.0   32.0  136.0   49.0  120.0   75.0   32.0   84.0   77.0
## [11]   84.0     NA  112.0   80.0   74.0 1358.0   77.0  110.0   17.0   75.0
## [21]   78.2  140.0  113.0   79.0   79.0   83.0     NA     NA   20.0   68.0
## [31]   89.0   90.0     NA   66.0   82.0     NA     NA     NA   40.0     NA
## [41]     NA   80.0     NA   55.0   45.0     NA   65.0   84.0   82.0   87.0
## [51]     NA   50.0     NA     NA   80.0     NA   85.0     NA     NA   80.0
## [61]   56.2   50.0     NA   80.0     NA   79.0   55.0  102.0   88.0     NA
## [71]     NA   15.0     NA   48.0     NA   57.0  159.0  136.0   79.0   48.0
## [81]   80.0     NA     NA     NA     NA     NA   45.0
```

# Dissecting the data-set: column of type character

Let us look at entries of one of the columns of type `chr`

```
# Access column "gender"
starwars$gender
```

the entry of a column of this type will always be enclosed within double quotes; the entry can be alphabetical, a string of alphabets or even numerals

if the entries are double quotes, then they belong to the type character; if not double quotes, they are either double or of type integer

```
##  [1] "masculine" "masculine" "masculine" "masculine" "feminine"  "masculine"
##  [7] "feminine"  "masculine" "masculine" "masculine" "masculine" "masculine"
## [13] "masculine" "masculine" "masculine" "masculine" "masculine" "masculine"
## [19] "masculine" "masculine" "masculine" "masculine" "masculine" "masculine"
## [25] "masculine" "masculine" "feminine"  "masculine" "masculine" "masculine"
## [31] "masculine" "masculine" "masculine" "masculine" "masculine" "masculine"
## [37] NA          "masculine" "masculine" NA          "feminine"  "masculine"
## [43] "masculine" "feminine"  "masculine" "masculine" "masculine" "masculine"
## [49] "masculine" "masculine" "masculine" "feminine"  "masculine" "masculine"
## [55] "masculine" "masculine" "masculine" "feminine"  "masculine" "masculine"
## [61] "feminine"  "feminine"  "feminine"  "masculine" "masculine" "masculine"
## [67] "feminine"  "masculine" "masculine" "feminine"  "feminine"  "masculine"
## [73] "feminine"  "masculine" "masculine" "feminine"  "masculine" "masculine"
## [79] "masculine" NA          "masculine" "masculine" "feminine"  "masculine"
## [85] "masculine" NA          "feminine"
```

# Dissecting the data-set: column of type list

Let us look at entries of one of the columns of type `list`

```
# Access column "films"
starwars$films[1:3]
```

row 1 to 3

```
## [[1]]
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
## [5] "The Force Awakens"
##
## [[2]]
## [1] "The Empire Strikes Back" "Attack of the Clones"
## [3] "The Phantom Menace"      "Revenge of the Sith"
## [5] "Return of the Jedi"      "A New Hope"
##
## [[3]]
## [1] "The Empire Strikes Back" "Attack of the Clones"
## [3] "The Phantom Menace"      "Revenge of the Sith"
## [5] "Return of the Jedi"      "A New Hope"
## [7] "The Force Awakens"
```

# Dissecting the data-set: rows

Each row is an observation

filtering the rows first by the name and then selecting the columns of interest for you

assignment operator= is equal to

To assign this value to this variable

```
# Rows of interest
filter_rows <- c("Luke Skywalker","R2-D2","Darth Vader")
# Extract row corresponding to Luke Skywalker
starwars %>% filter(name%in%filter_rows)
```

piping operator

operator: pass the dataset to this command as the input, the output of this command is passed at the input to this

```
## # A tibble: 3 × 14
##   name      height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr> <chr>
## 1 Luke Sky…    172    77 blond      fair       blue             19   male  mascu…
## 2 R2-D2         96    32 <NA>       white, bl… red              33   none  mascu…
## 3 Darth Va…    202   136 none       white      yellow           41.9 male  mascu…
## # ℹ 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

# Dissecting the data-set: rows with selected variables

*Filter* some rows and **select** some of their attributes

```
# Rows of interest
filter_rows <- c("Luke Skywalker","R2-D2")
# Extract row corresponding to Luke Skywalker
starwars %>% filter(name%in%filter_rows) %>% select(name,height,mass,homeworld,films)
```

```
## # A tibble: 2 × 5
##   name           height  mass homeworld films
##   <chr>           <int> <dbl> <chr>     <list>
## 1 Luke Skywalker    172    77 Tatooine  <chr [5]>
## 2 R2-D2              96    32 Naboo     <chr [7]>
```

**Notice** how the entries of the column, films, do not show up!

# Dissecting the data-set: rows with selected variables (continued)

Select attributes of type `list`

```
# Rows of interest
filter_rows <- c("Luke Skywalker","R2-D2")
# Extract rows in 'rows'
starwars %>% filter(name%in%filter_rows) %>% pull(films)
```

So pull() creates a vector -- which, in this case, is numeric -- whereas select() creates a data frame.

```
## [[1]]
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
## [5] "The Force Awakens"
##
## [[2]]
## [1] "The Empire Strikes Back" "Attack of the Clones"
## [3] "The Phantom Menace"      "Revenge of the Sith"
## [5] "Return of the Jedi"      "A New Hope"
## [7] "The Force Awakens"
```

# Dissecting the data-set: basic properties

```
# Number of rows in the data-set
nrow(starwars)
```

```
## [1] 87
```

```
# Number of columns in the data-set
ncol(starwars)
```

```
## [1] 14
```

```
# Number of rows and columns in the data-set
dim(starwars)
```

dimension

```
## [1] 87 14
```

# II. Exploratory Data Analyses (EDA)

# What is EDA?

- It is a systematic and iterative approach to explore data, and includes,
  - A hypothesis
  - Visualization, transformation or modelling data for answers
  - Substantiation or refinement of the hypothesis

- We will focus on **exploring** -- wrangling, manipulating, transforming --- and **visualizing** data

# Need for EDA

1. Are the attributes of the starwars characters related?
    - For instance, mass and height of the characters

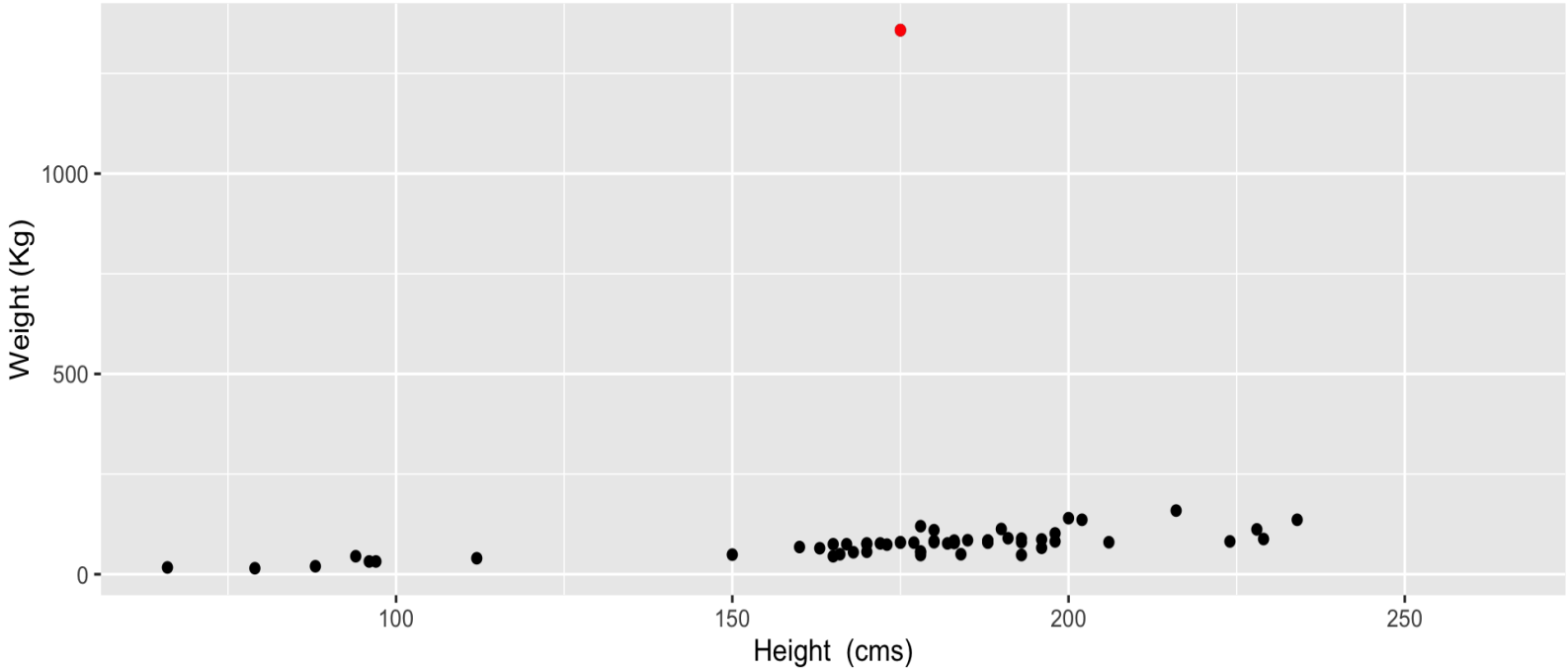
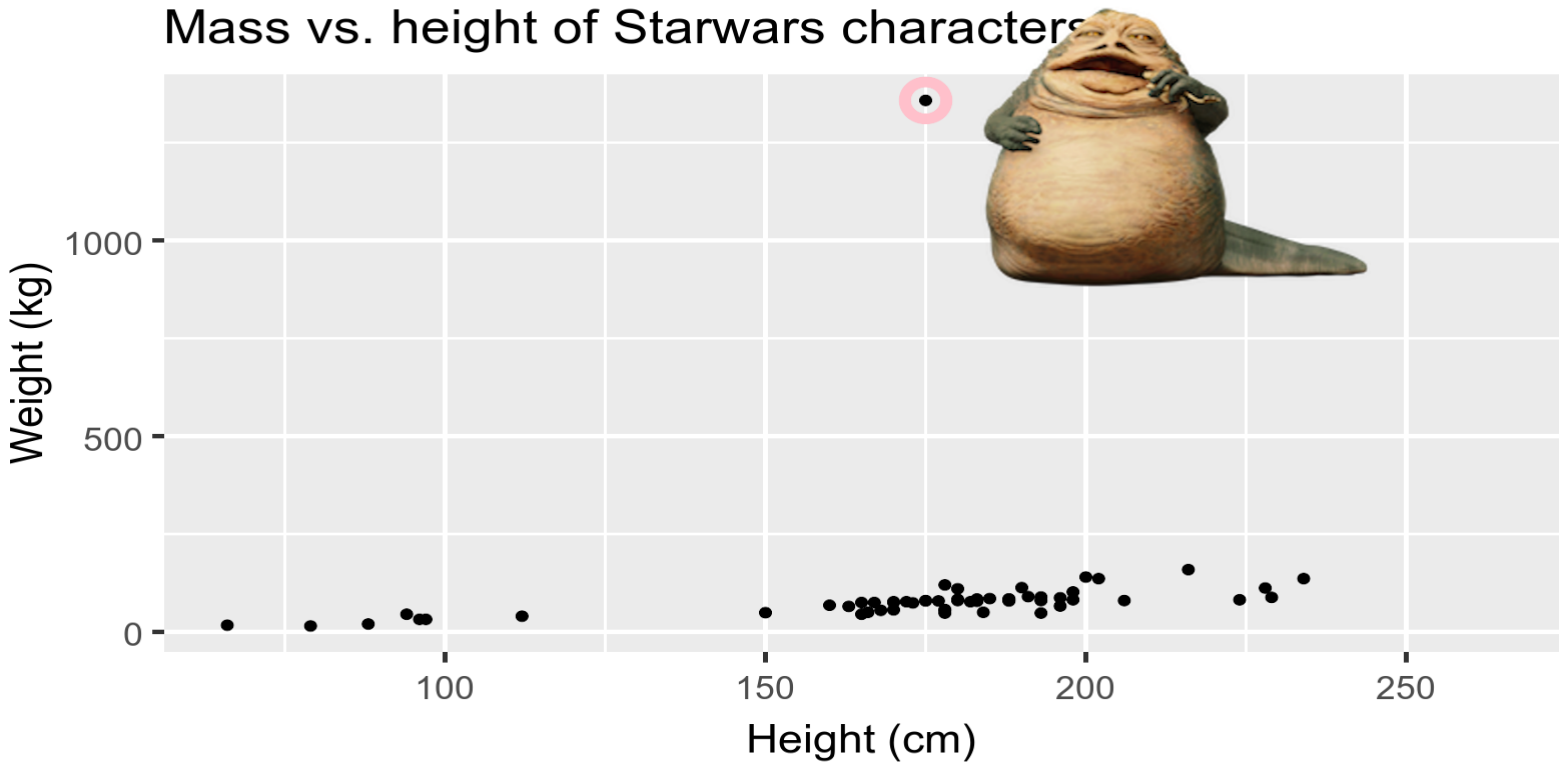



Figure: Relationship between weight and height

# Jabba!




Source: https://datasciencebox.org/

# Data visualization

> "A picture is worth a thousand words" - Frederick R. Bernard

1. Is just a visual representation of data
2. Entails creation and study of such representations
3. Among the many packages that are available for visualization, we will use `ggplot2` and `Shiny`

# Need for visualization: Anscombe's quartet

1. Consider a data-set, the **Anscombe's quartet**
2. It has four sets of data indicated by *I*, *II*, *III* and *IV*
3. Each set of data has 11 points, `(x,y)`

```
# Invoke the library
library(Tmisc)
# Filter data-set I in quartet
quartet %>% filter(set=="I")
```

```
##    set  x     y
## 1    I 10  8.04
## 2    I  8  6.95
## 3    I 13  7.58
## 4    I  9  8.81
## 5    I 11  8.33
## 6    I 14  9.96
## 7    I  6  7.24
## 8    I  4  4.26
## 9    I 12 10.84
```

```
# Invoke the library
library(Tmisc)
# Filter data-set II in quartet
quartet %>% filter(set=="II")
```

```
##    set  x    y
## 1   II 10 9.14
## 2   II  8 8.14
## 3   II 13 8.74
## 4   II  9 8.77
## 5   II 11 9.26
## 6   II 14 8.10
## 7   II  6 6.13
## 8   II  4 3.10
## 9   II 12 9.13
```

# Need for visualization: Anscombe's quartet

1. Consider a data-set, the **Anscombe's quartet**
2. It has four sets of data indicated by *I*, *II*, *III* and *IV*
3. Each set of data has 11 points, `(x,y)`

```
# Invoke the library
library(Tmisc)
# Filter data-set III in quartet
quartet %>% filter(set=="III")
```

```
##    set  x     y
## 1  III 10  7.46
## 2  III  8  6.77
## 3  III 13 12.74
## 4  III  9  7.11
## 5  III 11  7.81
## 6  III 14  8.84
## 7  III  6  6.08
## 8  III  4  5.39
## 9  III 12  8.15
```

```
# Invoke the library
library(Tmisc)
# Filter data-set IV in quartet
quartet %>% filter(set=="IV")
```

```
##    set  x     y
## 1   IV  8  6.58
## 2   IV  8  5.76
## 3   IV  8  7.71
## 4   IV  8  8.84
## 5   IV  8  8.47
## 6   IV  8  7.04
## 7   IV  8  5.25
## 8   IV 19 12.50
## 9   IV  8  5.56
```

# Anscombe's quartet: features (continued)

- And look at some statistics,

  a. Mean of `x` in each set, `mean_x`

  b. Mean of `y` in each set, `mean_y`

  c. Standard deviation of `x` in each set, `sd_x`

  d. Standard deviation of `y` in each set, `sd_y`

  e. Correlation between `x` and `y`, `r`

```
# Obtain the needed statistics
grouped_quartet %>%
  summarise(
    mean_x = mean(x),
    mean_y = mean(y),
    sd_x = sd(x),
    sd_y = sd(y),
    r = cor(x, y)
  )
```

```
## # A tibble: 4 × 6
##   set   mean_x mean_y  sd_x  sd_y     r
##   <fct>  <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 I          9   7.50  3.32  2.03 0.816
## 2 II         9   7.50  3.32  2.03 0.816
## 3 III        9   7.5   3.32  2.03 0.816
## 4 IV         9   7.50  3.32  2.03 0.817
```

# Anscombe's quartet: features (continued)

Will the **distributions** of the four sets of data points be identical too? Let us find out!

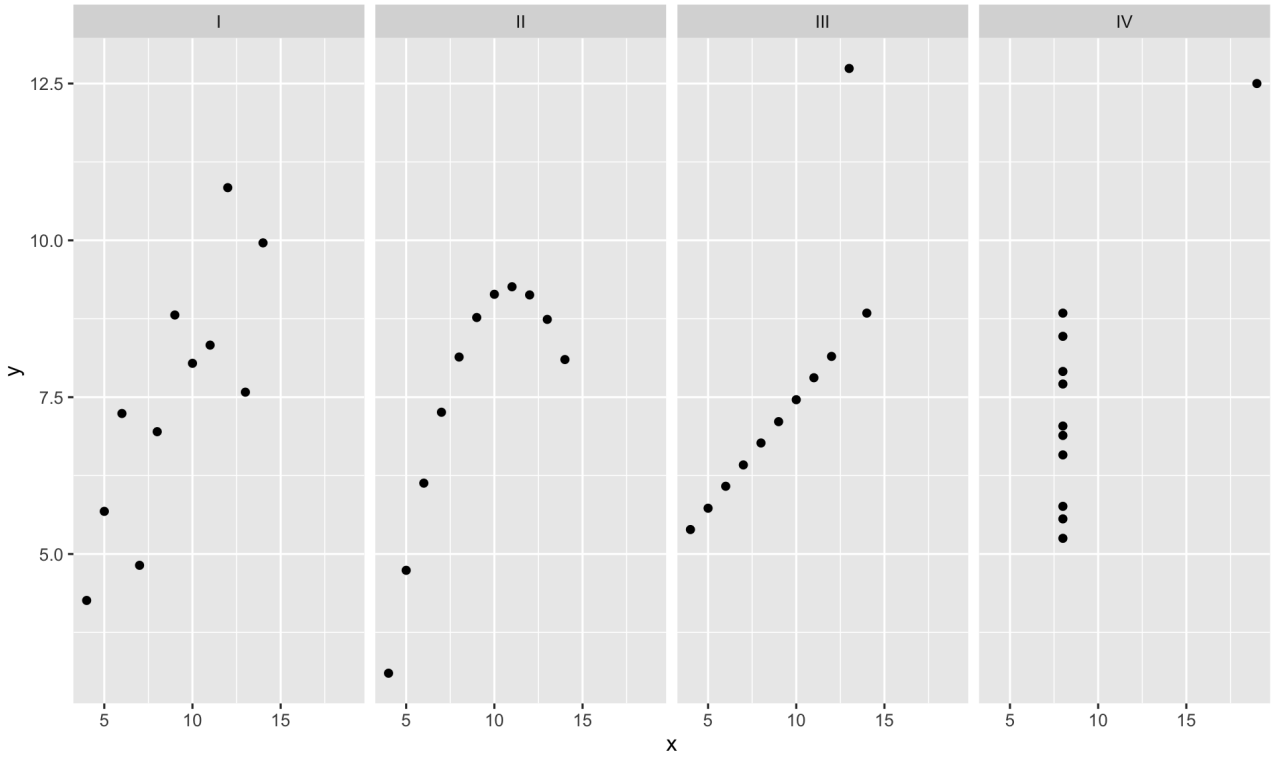



Figure: Data in each of the four data sets

# III. `ggplot2`

# Data visualization with `ggplot2`

**ggplot2 has ready-made commands that you can just plug into your code and plot data**

- It is the `tidyverse` library's visualization package
- `gg` in `ggplot` stands for **G**rammar of **G**raphics
- Enables us to describe different components of a graphic


Figure: Different layers of ggplot

# Let us construct our first plot: Mass versus Height

- The main function is `ggplot()`
- We will work with `starwars` data-set

**after passing data to the function ggplot, you need to add aesthetics layer**

```
# Plot the data
ggplot(data=starwars) # <-- Look here!
```



Figure: Constructing Mass versus Height plot

# Mass versus Height: independent variable (IV)

- The **independent variable** (along x-axis) is the variable/column -- `height`

```
# Plot height along x-axis
ggplot(data=starwars) +
  aes(x=height) # <-- Look here!
```

**this is required to superimpose one layer on top of another layer**

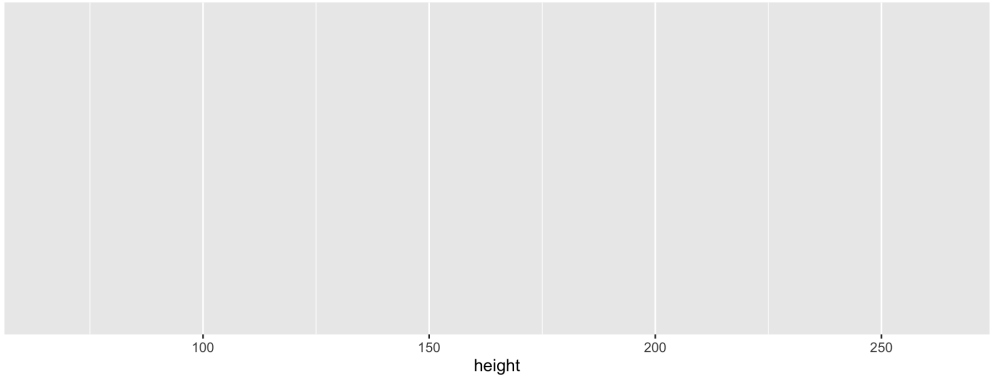



Figure: Adding independent variable

# Mass versus Height: dependent variable (DV)

- Similarly, the **dependent variable** (along y-axis) is the variable/column -- `mass`

```
# Plot mass along y-axis
ggplot(data=starwars) +
  aes(x=height,y=mass) # <-- Look here!
```

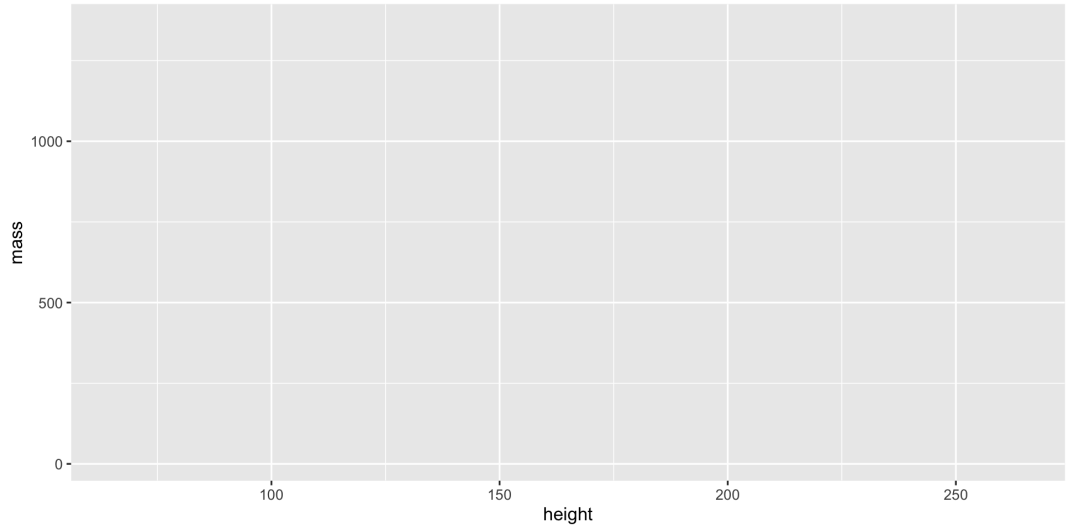



Figure: Adding dependent variable

# Mass versus Height: visualize data as dots

- Let us choose to visualize the data as **points** (dots)

```
ggplot(data=starwars) + aes(x=height,y=mass) +
  geom_point() # <-- Look here!
```

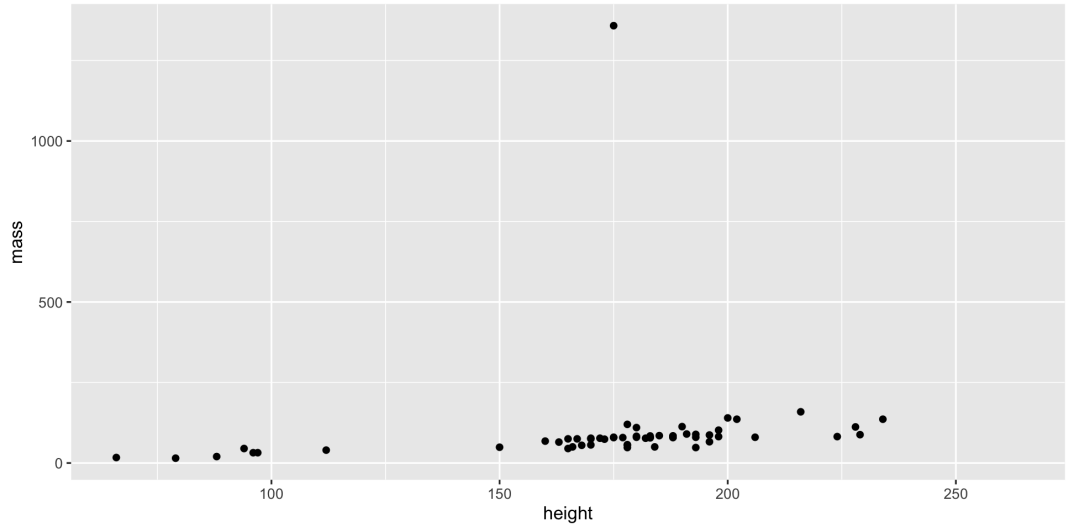



Figure: Visualizing data as points

# Mass versus Height: re-write axes labels

- Let us rewrite **labels** for the **axes** -- `x` and `y`

```
ggplot(data=starwars) + aes(x=height,y=mass) + geom_point() +
  labs(x="Height (cm)",y="Weight (Kg)") # <-- Look here!
```

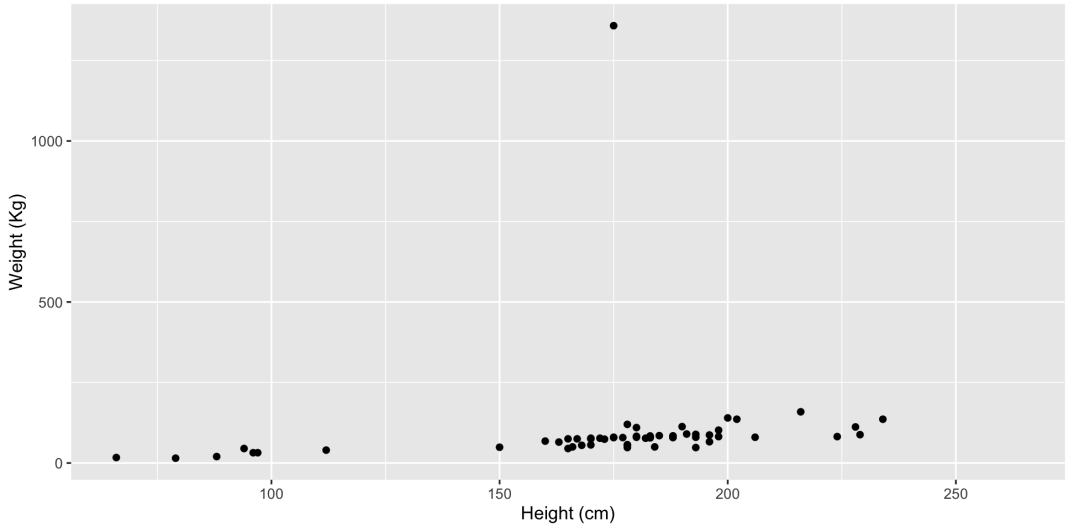



Figure: Inserting labels

# Mass versus Height: add title

- Let us insert a **title** for the plot

```
ggplot(data=starwars,mapping=aes(x=height,y=mass)) +
  geom_point() +
  labs(x="Height (cm)",y="Weight (Kg)",
       title="Mass versus Height")# <-- Look here!
```

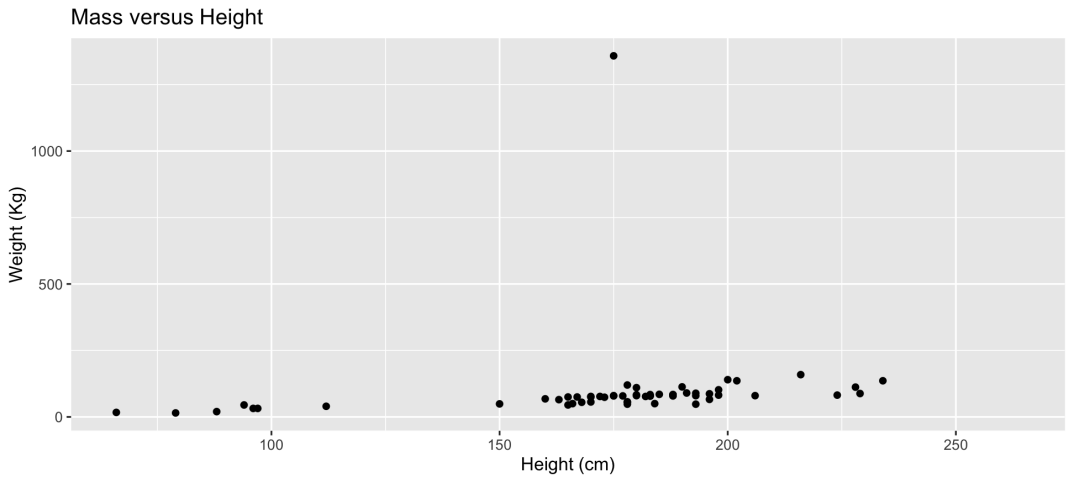
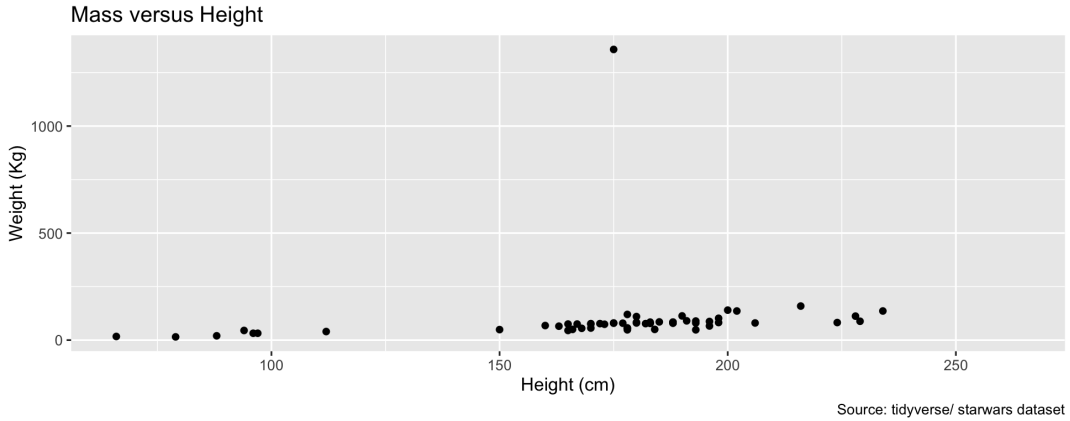



Figure: Inserting title

# Mass versus Height: add caption

- Let us insert a **caption** for the plot

```
ggplot(data=starwars,mapping=aes(x=height,y=mass)) +
  geom_point() +
  labs(x="Height (cm)",y="Weight (Kg)",
       title="Mass versus Height",
       caption="Source: tidyverse/ starwars dataset") # <-- Look here!
```




capture some additional info which may not be as important as the title itselfi

Figure: Inserting caption

# Mass versus Height: complete code

```
ggplot(starwars) +
  aes(x=height,y=mass) +
    geom_point() +
      labs(x="Height (cm)",
             y="Weight (Kg)",
               title="Mass versus Height",
                 caption="Source: tidyverse/ starwars dataset")
```

# Mass versus Height: tadaa!

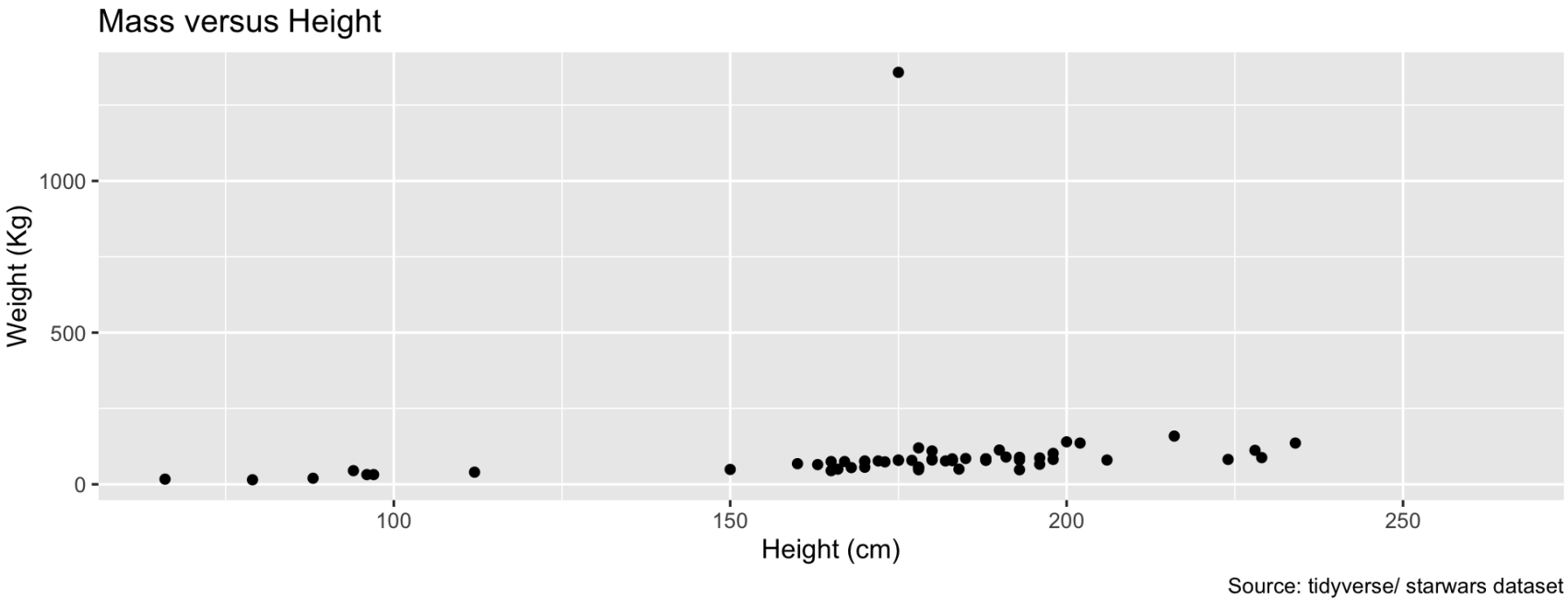



Figure: Final Mass versus Height plot

# IV. Shiny

# `Shiny`: introduction

1. `Shiny` is an R package that facilitates interactive web apps more here. Using which, apps
   a. can be hosted on a webpage
   b. embedded in a dashboard, or
   c. embedded in an R Markdown document
2. We shall explore some examples in the subsequent slides

# `Shiny`: example

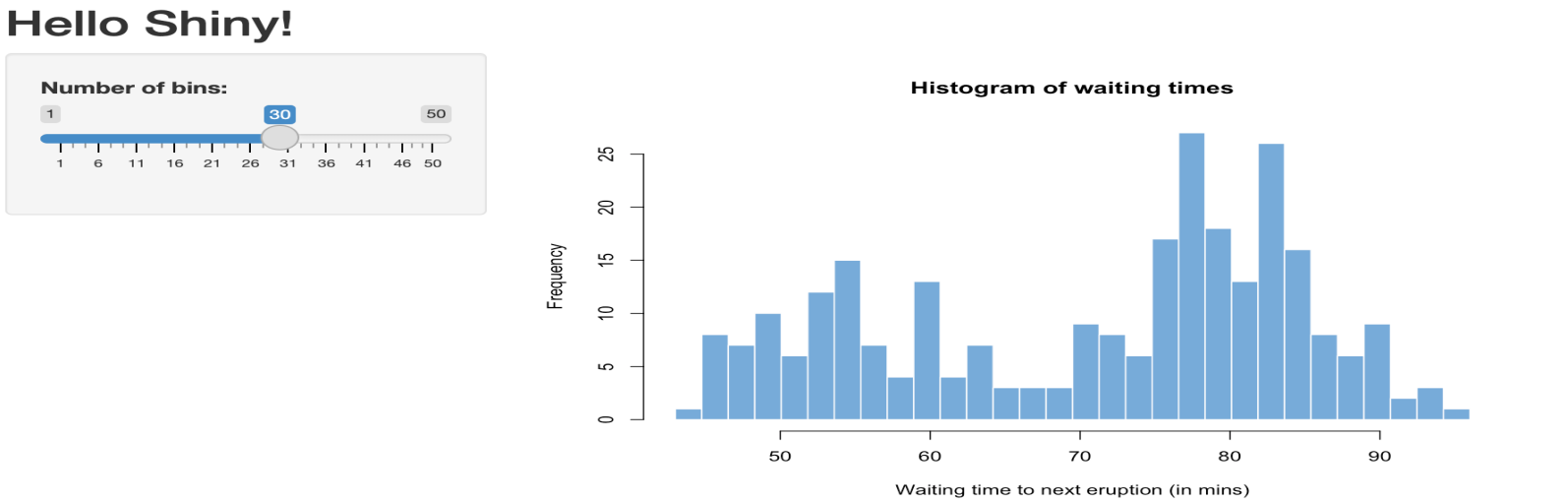
- The package has eleven built-in examples to demonstrate how `Shiny` works
- To explore the first example, run the code below

```
# Install package
install.packages("shiny")

# Invoke the package
library(shiny)

# Run an example from the library
runExample("01_hello")
```

- You could try to list all the examples using `runExample()`
- If you haven't been using `?` operator before every function you use, start doing so
- It reveals a lot of information about the function
- If you tried the code, move on to the next slide for the output

# Example

`Shiny` dashboard with example - "01_hello"




This small Shiny application demonstrates Shiny's automatic UI updates.

Move the *Number of bins* slider and notice how the `renderPlot` expression is automatically re-evaluated when its dependant, `input$bins`, changes, causing a histogram with a new number of bins to be rendered.

app.R

show with app

```
library(shiny)

# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # App title ----
  titlePanel("Hello Shiny!")
```

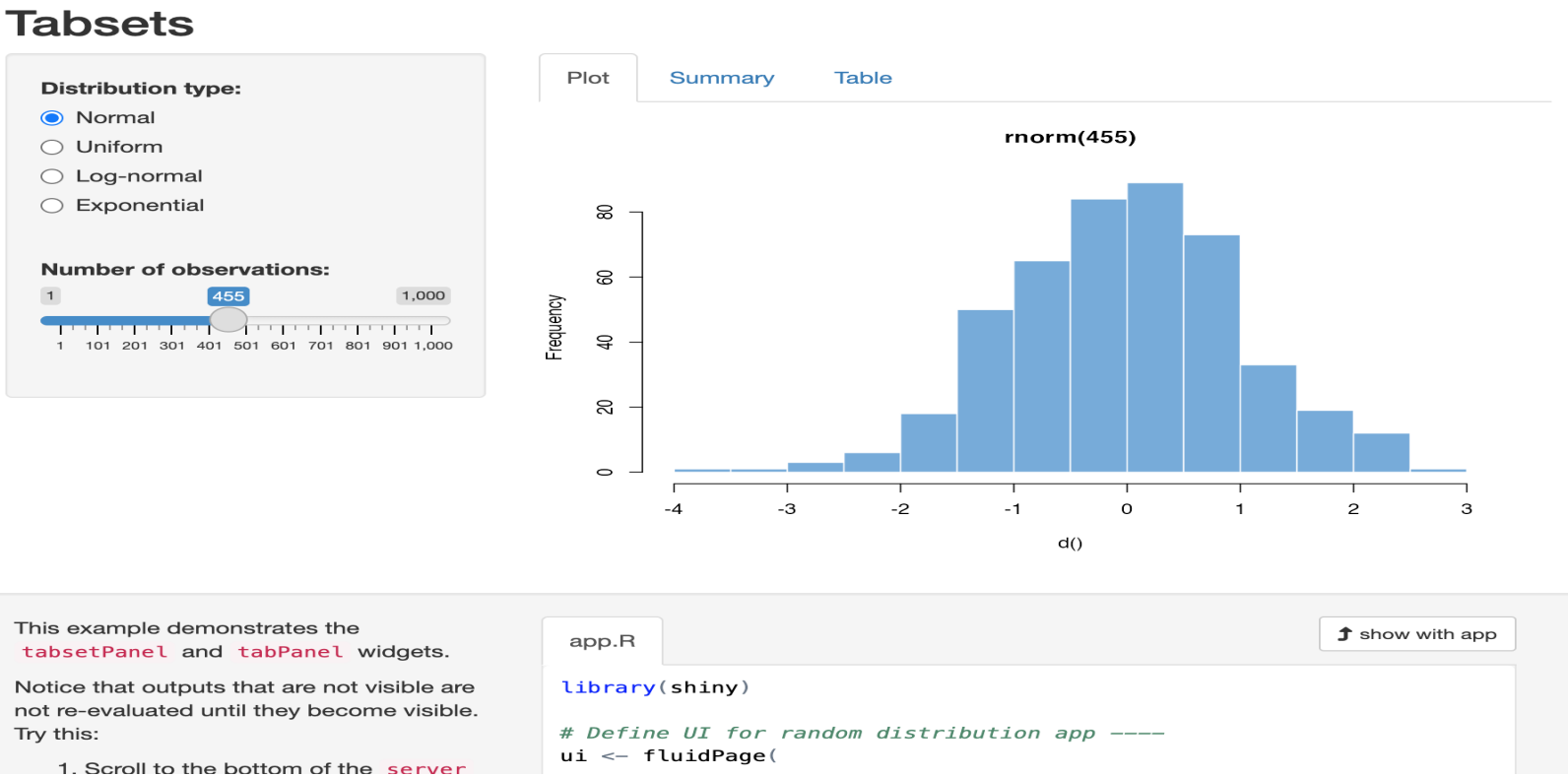
# `Shiny`: comprehensive example

```
# Install package
install.packages("shiny")

# Invoke the package
library(shiny)

# Run an example from the library
runExample("06_tabsets")
```

Head to the next slide for the output!

# Comprehensive example

`Shiny` dashboard with example - "06_tabsets"



# More examples

To try more examples, just replace the example name in the code snippet from the list

1. 01_hello
2. 02_text
3. 03_reactivity
4. 04_mpg
5. 05_sliders
6. 06_tabsets
7. 07_widgets
8. 08_html
9. 09_upload
10. 10_download
11. 11_timer

```
# Install package
install.packages("shiny")

# Invoke the package
library(shiny)

# Run an example from the library
runExample("insert_example_name_here")
```

# Commands and operators revisited

1. `?` - operator to learn more about an R command
2. `$` - operator to extract a specific part of data
3. `%>%` - pipe operator that passes the output of one function as an input to another
4. `%in%` - operator to check if an element belongs to a data-set
5. `library(package_name)` - to load an add-on package
6. `glimpe(dataset_name)` - get a glimpse of your data
7. `filter(row_attributes)` - subset rows using column values
8. `select(col_names)` - subset columns using their names or types
9. `pull(column_name)` - extracts a single column
10. `nrow/ncol/dim(dataset_name)` - number of rows/columns/dimension of a data-set

# Thanks!

Slides created via the R packages:

xaringan
gadenbuie/xaringanthemer.


Faculty of Arts
& Social Sciences