TDU

R FOR DATA VISUALIZATIONS

Day 1

Public Health Agency of Canada
Agence de la santé publique du Canada

Canada

ACKNOWLEDGEMENTS

This course has been developed in collaboration with several friends of the Training and Development Unit. Notably, we'd like to acknowledge the contributions of Naj Saqib (PHAC) and Chris Mill (BCCDC)

https://native-land.ca

TERRITORIAL ACKNOWLEDGEMENT

Marc-André (MAB)
Land of the Coast Salish Peoples - Squamish, Tsleil-Waututh & Musqueam First Nations (Vancouver, BC)

Joanne
Land of the Beothuk and Mi'kma'ki people (St. John's, NL)

Penelope
Land of the Mississauga, Anishinabewaki, Haudenosaunee, Wendake-Nionwentsïo & Mississauga of the Credit First Nation (Toronto, ON)

Ben
Treaty One Territory, Land of the Anishinabe, Ininew, Oji-Cree, Dene, Dakota, and Métis people (Winnipeg, MB)

(P)ICEBREAKER

Source: Edward Tufte

HOUSEKEEPING

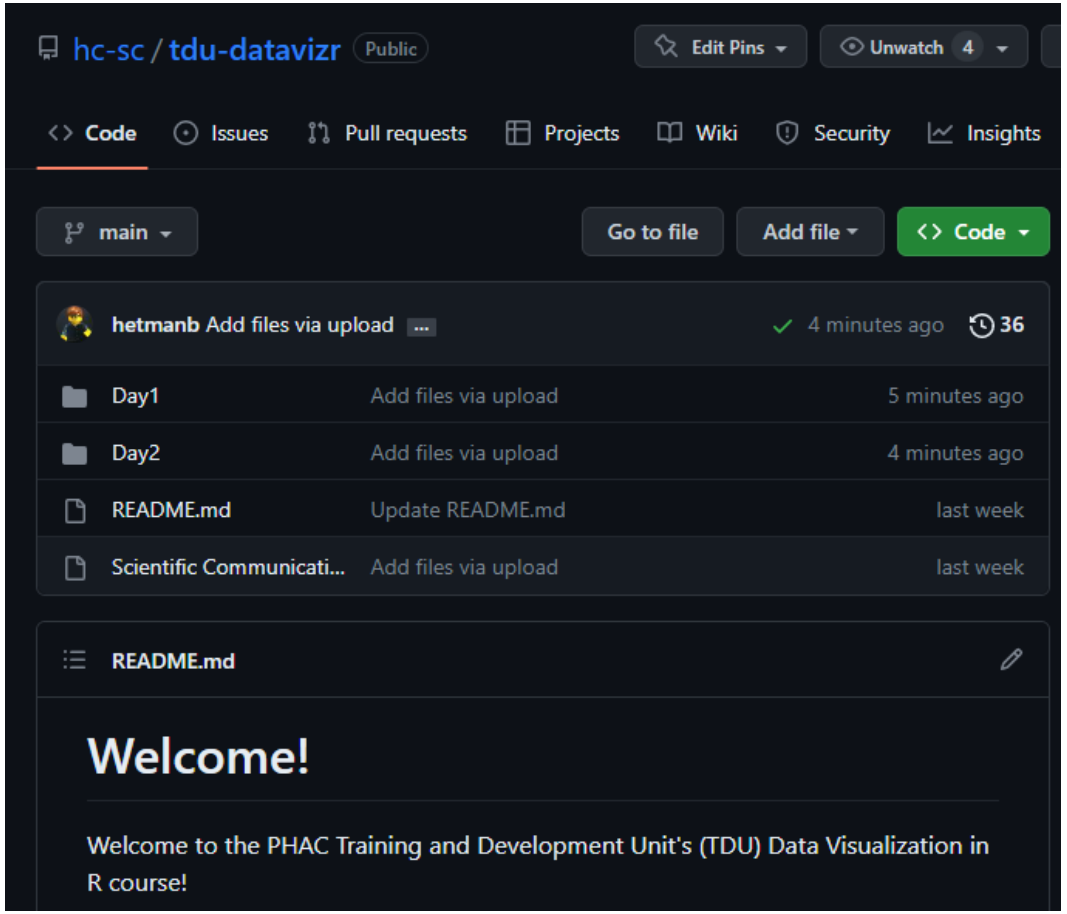
- Optimal set-up for Zoom:
  - Disconnect from VPN
  - If you are experiencing significant connectivity issues, please reach out

- Etiquette
  - Stepping away – use coffee cup, or just let us know
  - Device ringers off
  - Mute if not speaking

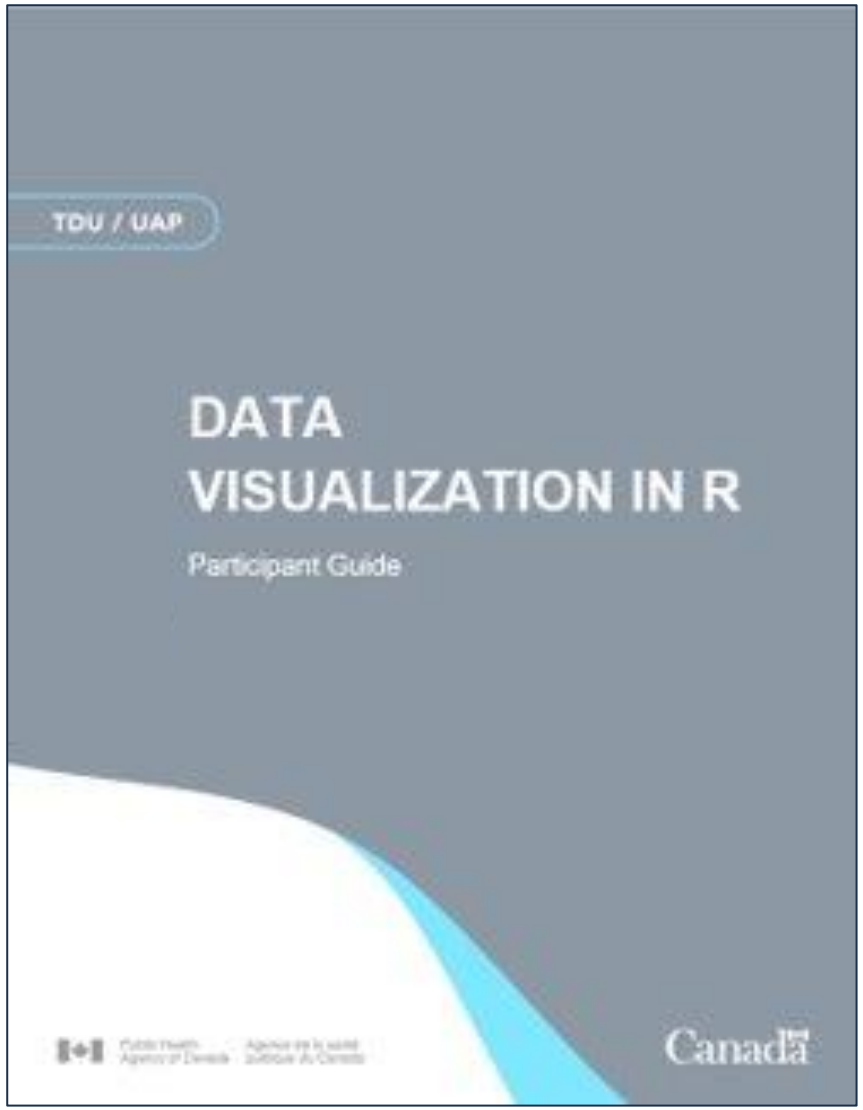
COURSE MATERIALS

- File sharing platform: Github
  - Virtual classroom materials
  - Self-Study
  - Additional resources

- hc-sc/tdu-datavizr (github.com)



COURSE MATERIALS



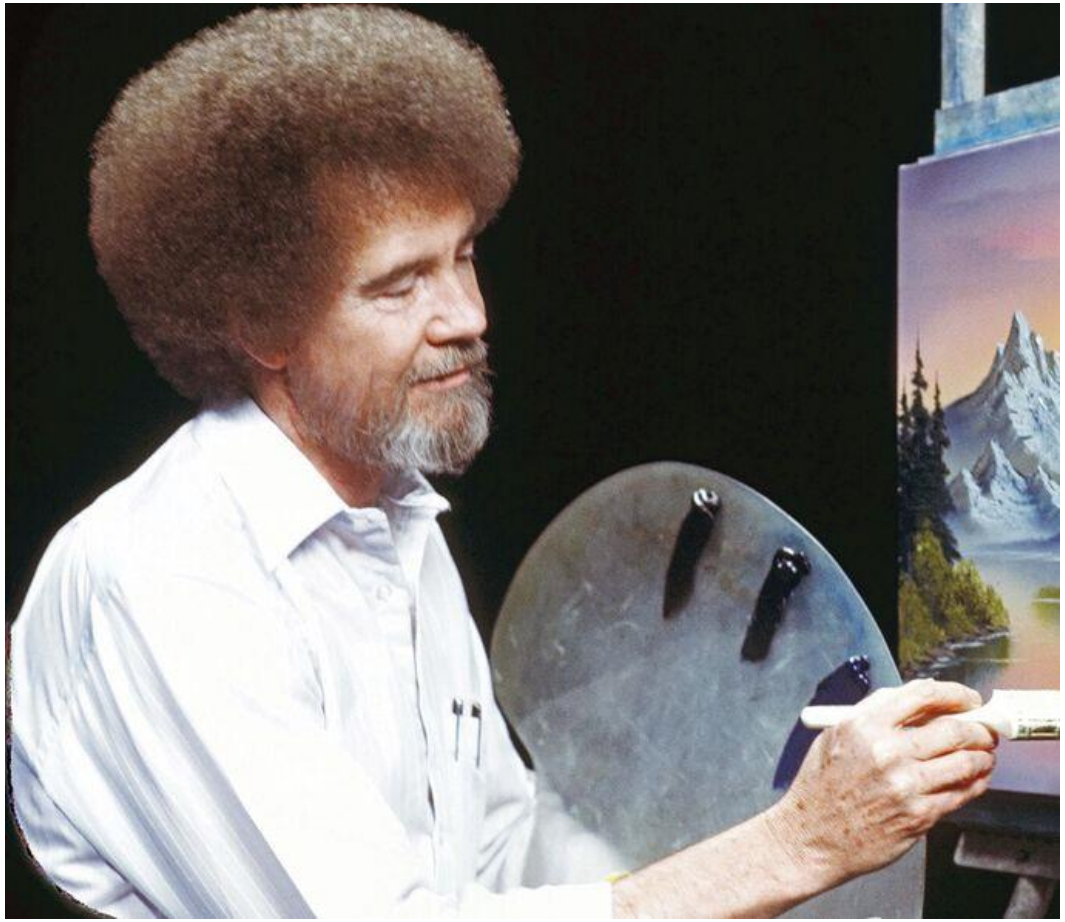
COURSE OVERVIEW

Day 1	Day 2	Day 3
Base R graphics system	Best practices in data visualization	Review activity
Ggplot graphics system	Worst practices in data visualization	Demo
Demo	Rules to ‘viz’ by	Supported practice time
Independent activity	Demo	
	Independent activity	

COURSE LEARNING OBJECTIVES

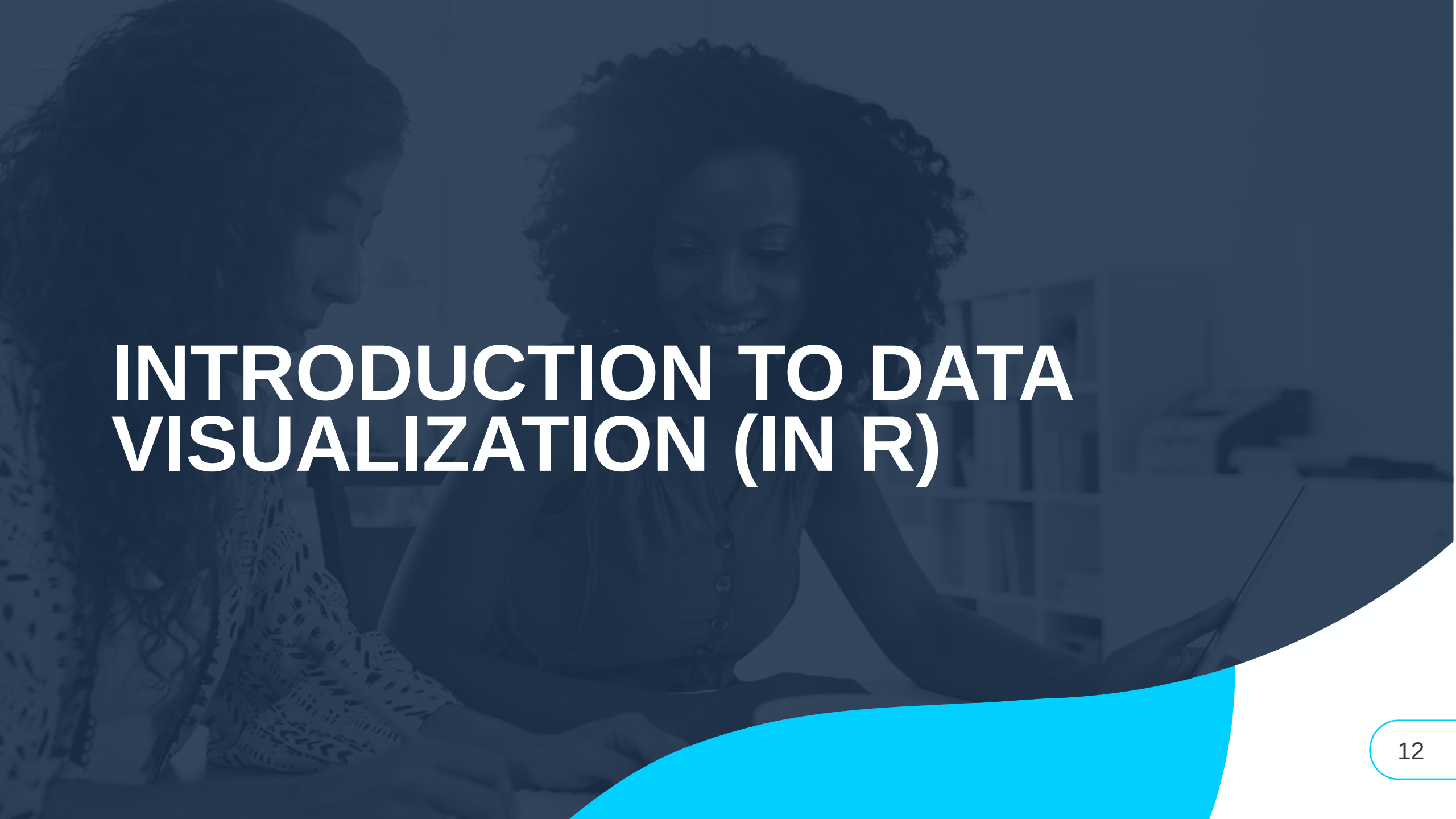
- By the end of this course, participants will be able to:
  - Discuss differences between base R and the grammar of graphics (ggplot) coding styles for data visualization.
  - Apply knowledge of R-coding to automate common graphics used in public health.
  - Connect elements of effective graphic design to R coding practices in data visualization.
  - Discuss considerations for data visualisation to avoid inflicting harm

COURSE PHILOSOPHY



1. We learn from our mistakes more than from our successes
2. Embrace the errors
3. Ask all the questions
4. Be kind to one another, but most importantly to yourself
5. Have fun!

“Ever make mistakes in life? Let’s make them birds. Yeah, they’re birds now.” – Bob Ross



INTRODUCTION TO DATA VISUALIZATION (IN R)

WHAT IS DATA VISUALIZATION (“DATA VIZ”)

- What do YOU think?
- Feel free to come off mute, or type in the chat!

WHAT IS DATA VISUALIZATION (“DATA VIZ”)

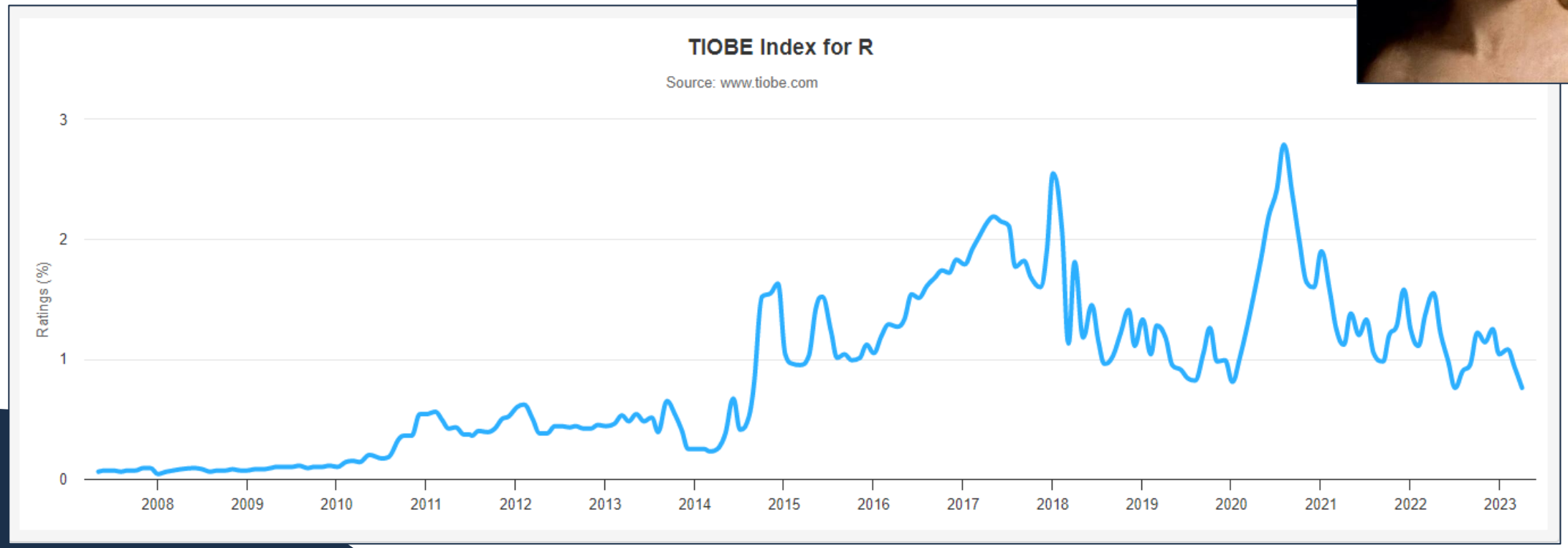
- Data visualization ("data viz") is the process of **organizing and presenting data** in a way that **conveys a message or story**

WHY USE R FOR DATA VIZ?

- R is **powerful**

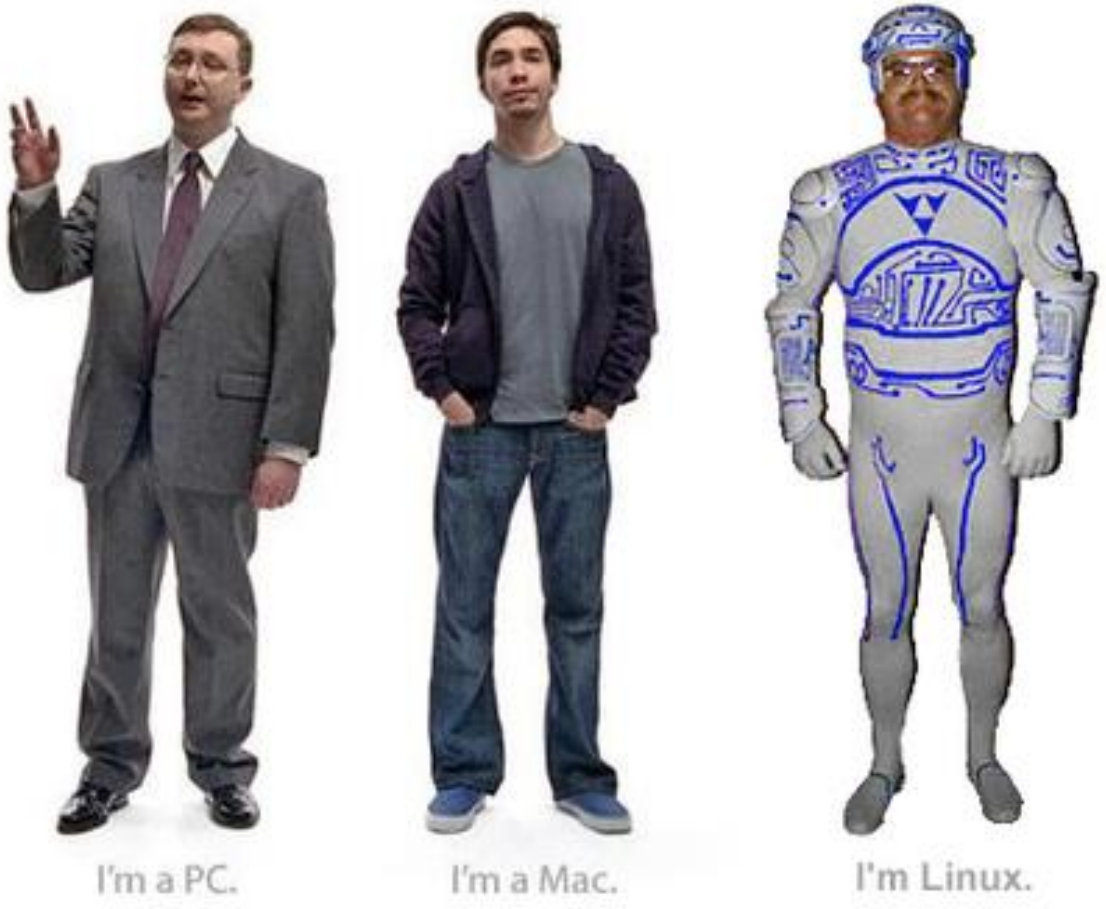
WHY USE R FOR DATA VIZ?

- R is powerful
- R is **popular**



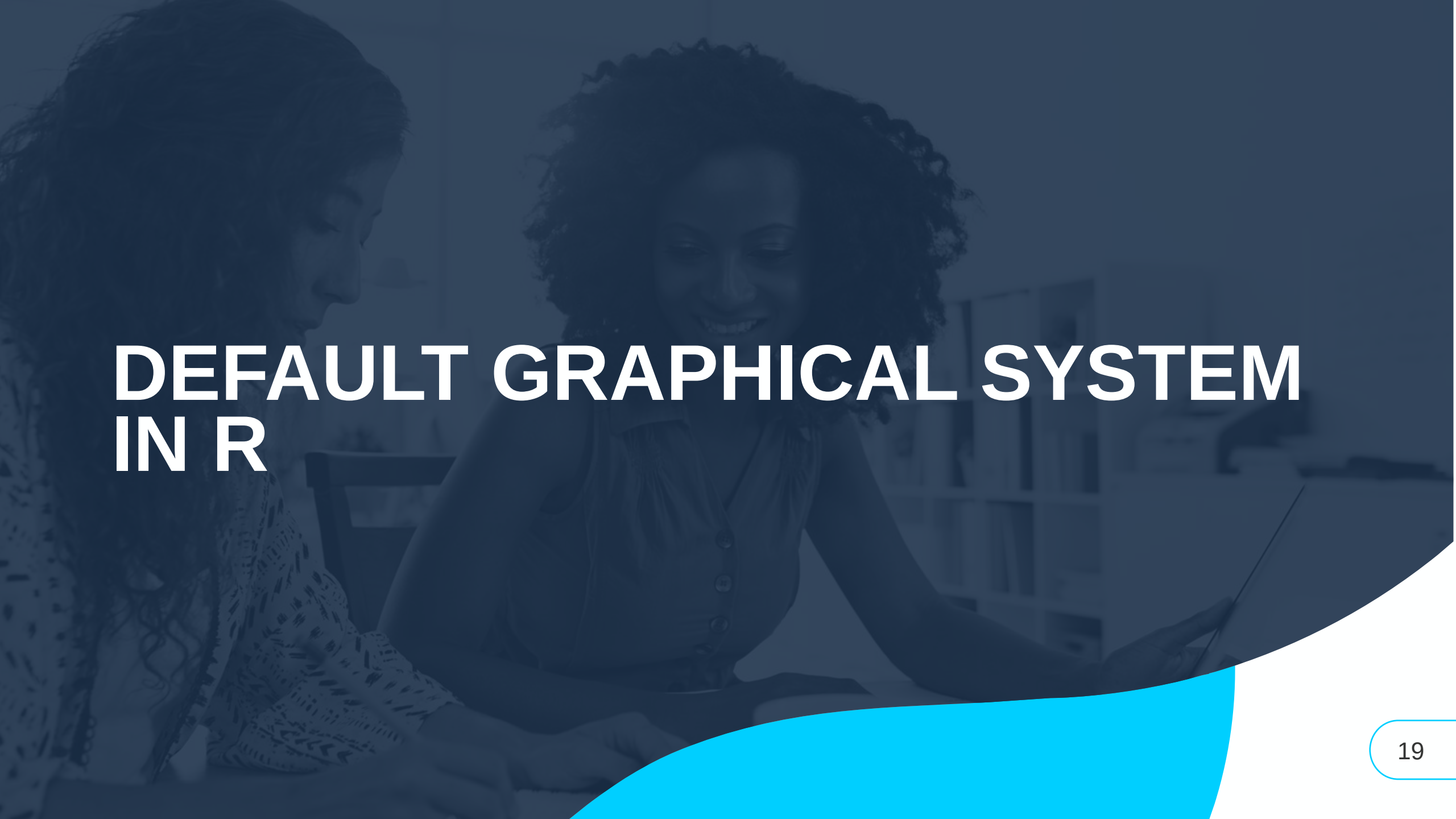
WHY USE R FOR DATA VIZ?

- R is powerful
- R is popular
- R is **widely available**



WHY USE R FOR DATA VIZ?

- R is powerful
- R is popular
- R is widely available
- R is **free!**



DEFAULT GRAPHICAL SYSTEM IN R

BASE R GRAPHICS

- Original data visualization system in R
- Plenty of graphics options available in base R
- No need to install additional packages to run
- Quick, efficient, simple commands
- Less popular than ggplot()

BASE R GRAPHICS

- Graphics in base R are created in a special graphics window, allowing for the display of your data visualization
  - Different display driver based on your operating system:
    - Windows = window()
    - Linux = x11()
    - MacOS = quartz()

Tip:

- These display drivers are sometimes the reason your outputs will look slightly different on different computers.
- Occasionally, you may encounter errors that are related to the graphics driver, not the R code. If this happens, try using *dev.off()* to reset your sessions graphics device.

GETTING STARTED WITH BASE R

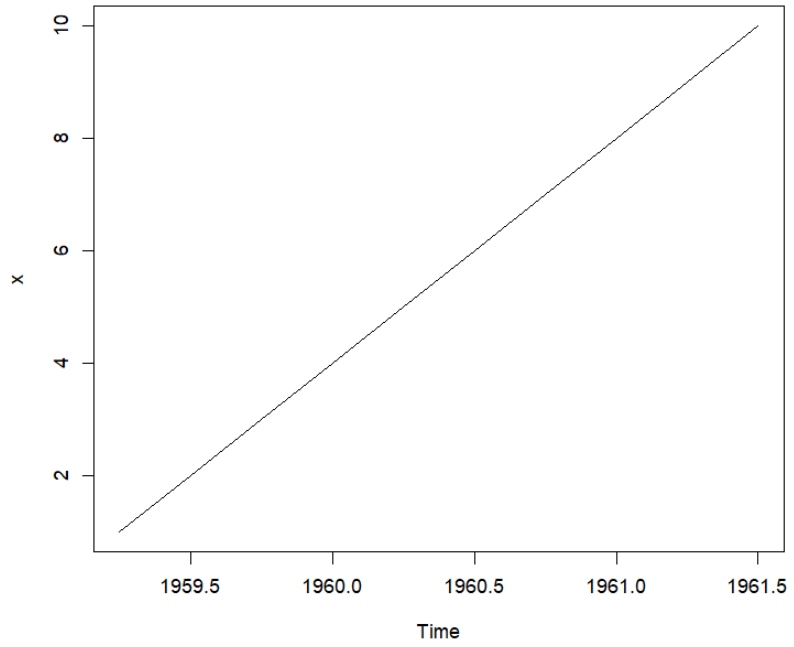
- ‘High-level’ plotting command for base R plotting: **plot()**

- **plot()** is a *general* function: therefore it **produces different results** depending on input

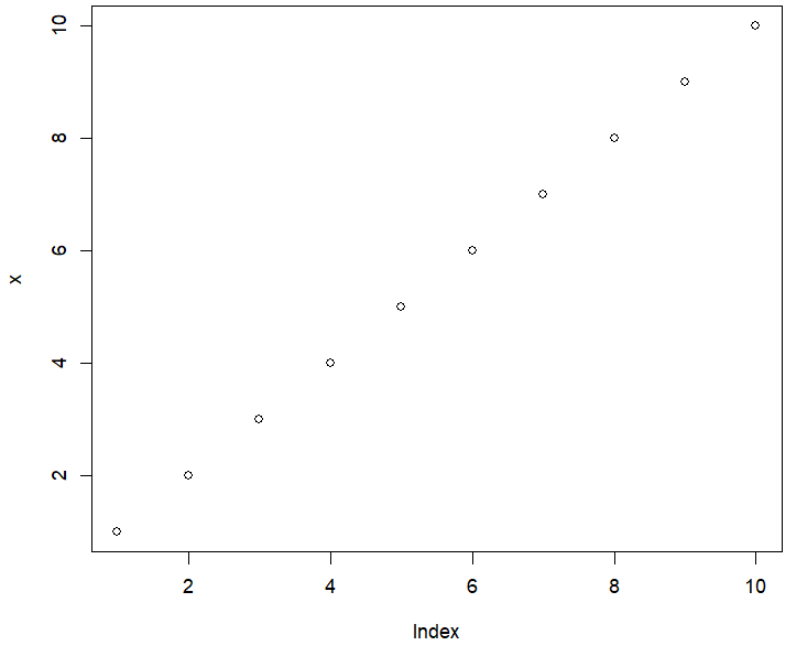
GETTING STARTED WITH BASE R

- E.g., What happens when I enter: plot(x)?
  - If x is a time series object, will produce a time-series plot
  - If x is a numeric vector, it will produce a plot of the value against it's index in the vector

plot(x) #time series



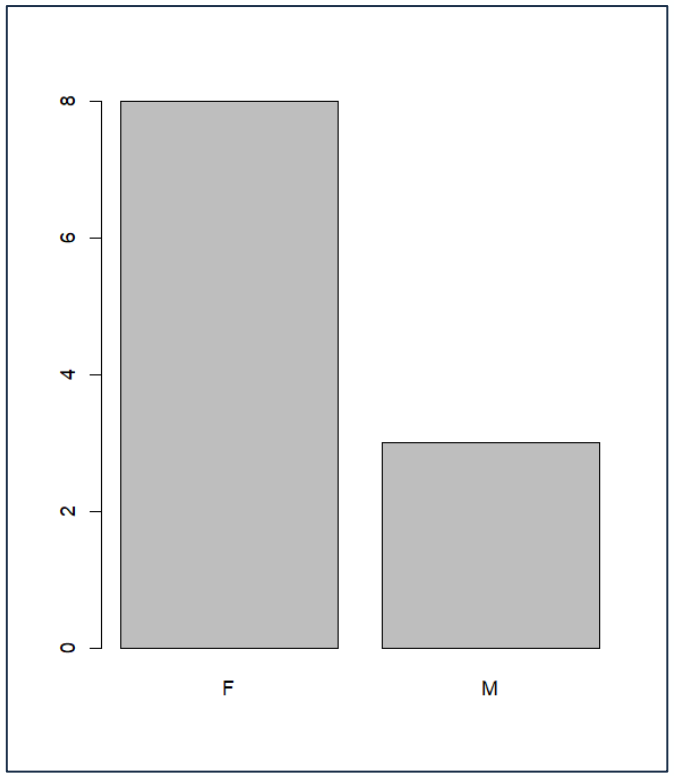
plot(x) #numeric vector



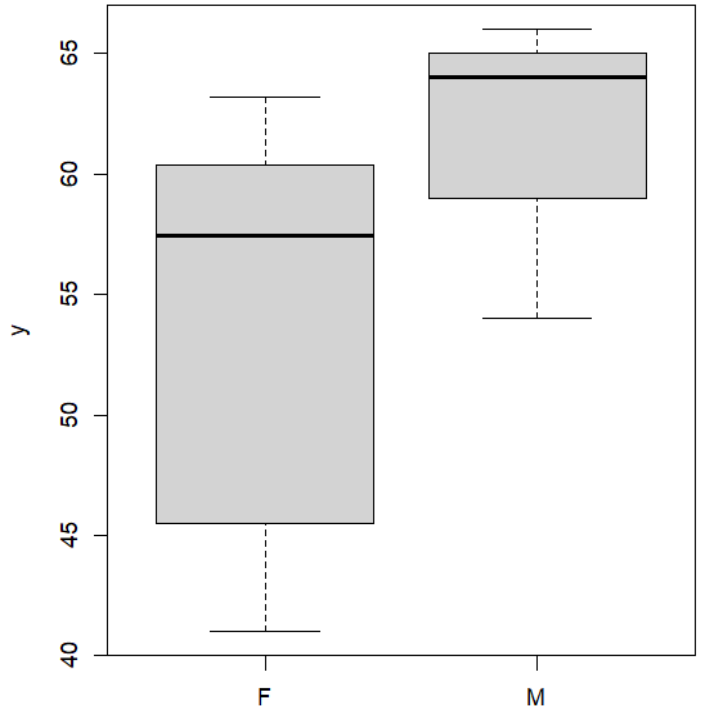
GETTING STARTED WITH BASE R

- What about other data types? E.g., factors? data frames?
  - plot(*f*) : generates a **barplot** of the factor, *f*
  - plot(*f*, y) : generates a **boxplot** of the numeric vector *y* for <u>each level</u> of the factor, *f*

plot(f) #factor



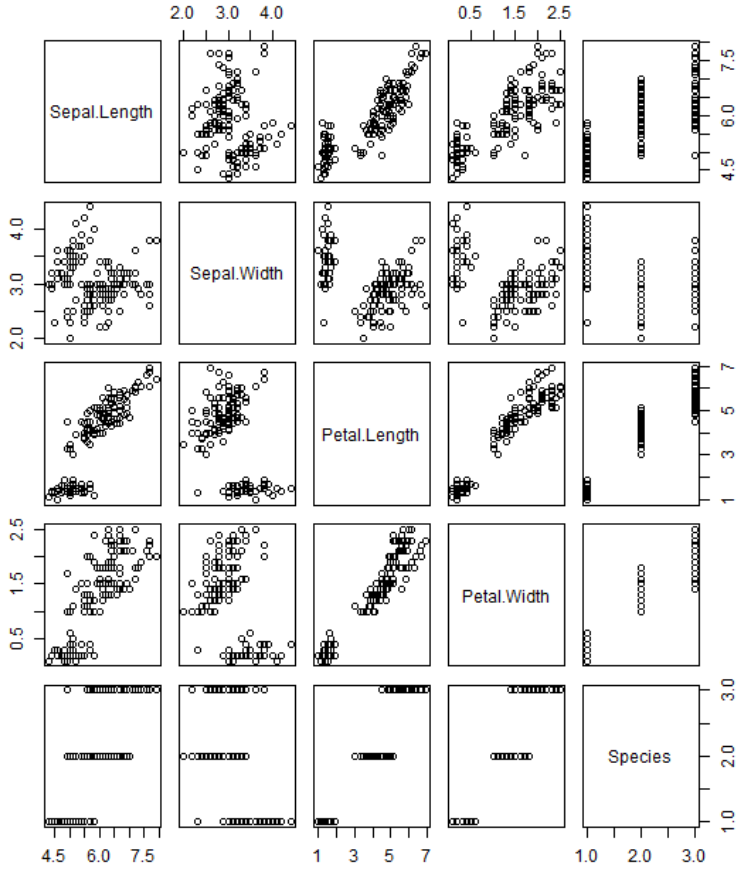
plot(f, y) #vector by group level



GETTING STARTED WITH BASE R

- Plotting with data frames
  - plot(df) generates distribution plots of all the variables in the data frame!

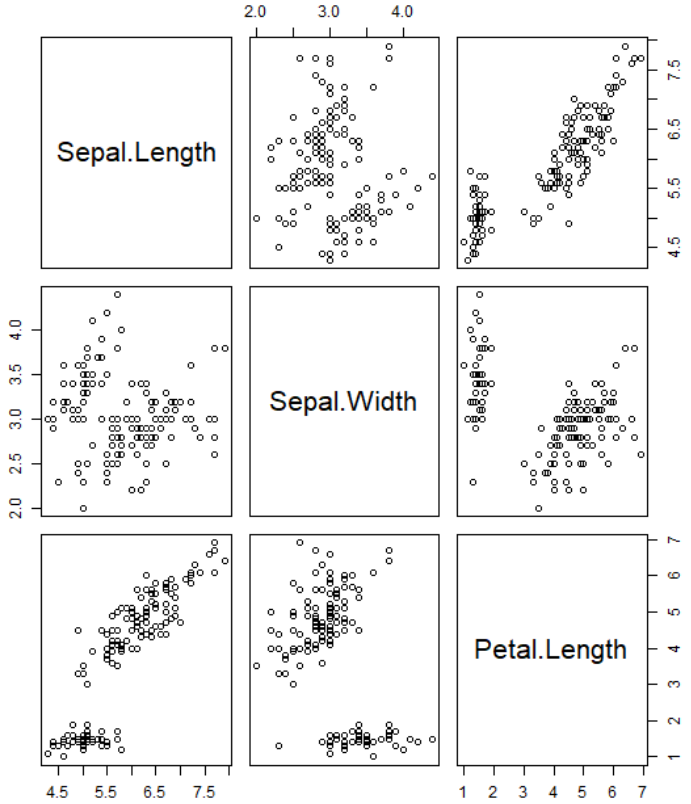
- e.g., plot(iris)



GETTING STARTED WITH BASE R

plot(~ expr): lets you choose which distributions are plotted

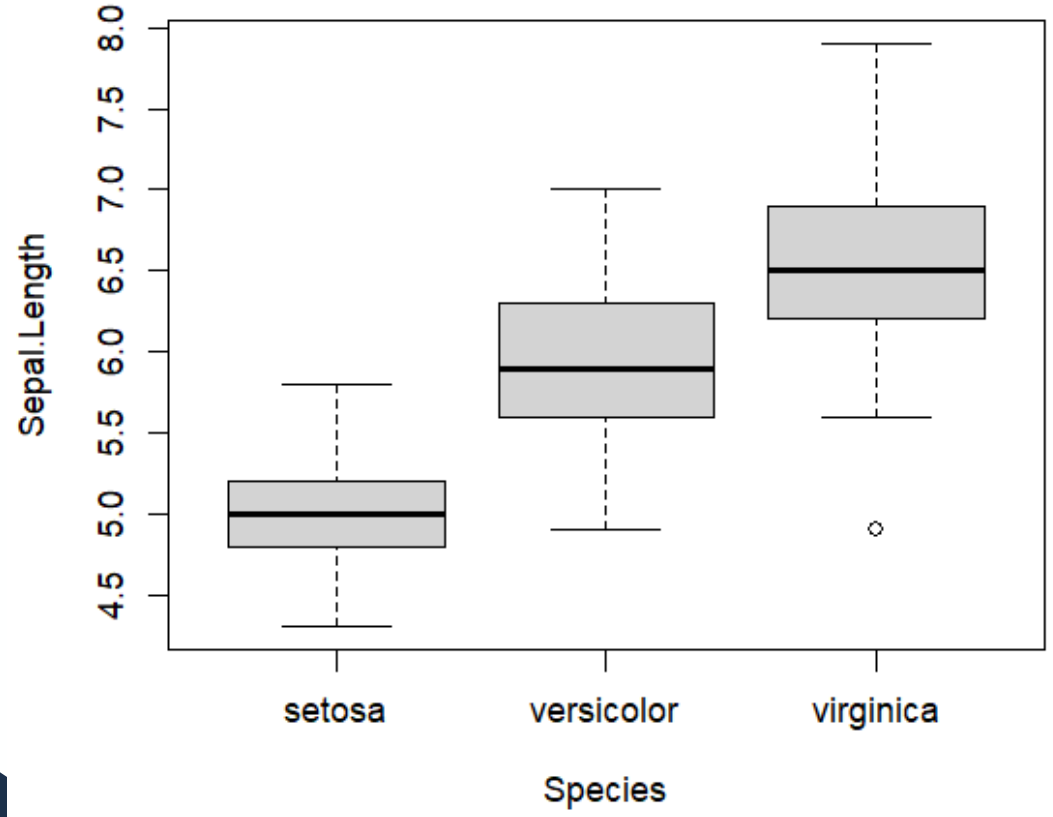
- E.g., plot(~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = iris)



GETTING STARTED WITH BASE R

You can specify which object to compare against by adding plot(y ~ expr)

- E.g., plot(Sepal.Length ~ Species, data = iris)



GETTING STARTED WITH BASE R

Optional arguments to pass to **plot()**:

type = change the plot type (e.g., “l” = line, “p” = point, “b” = both, ...)

main = specify the plot title

sub = specify the subtitle

xlab = specify the title for the x-axis

ylab = specify the title for the y-axis

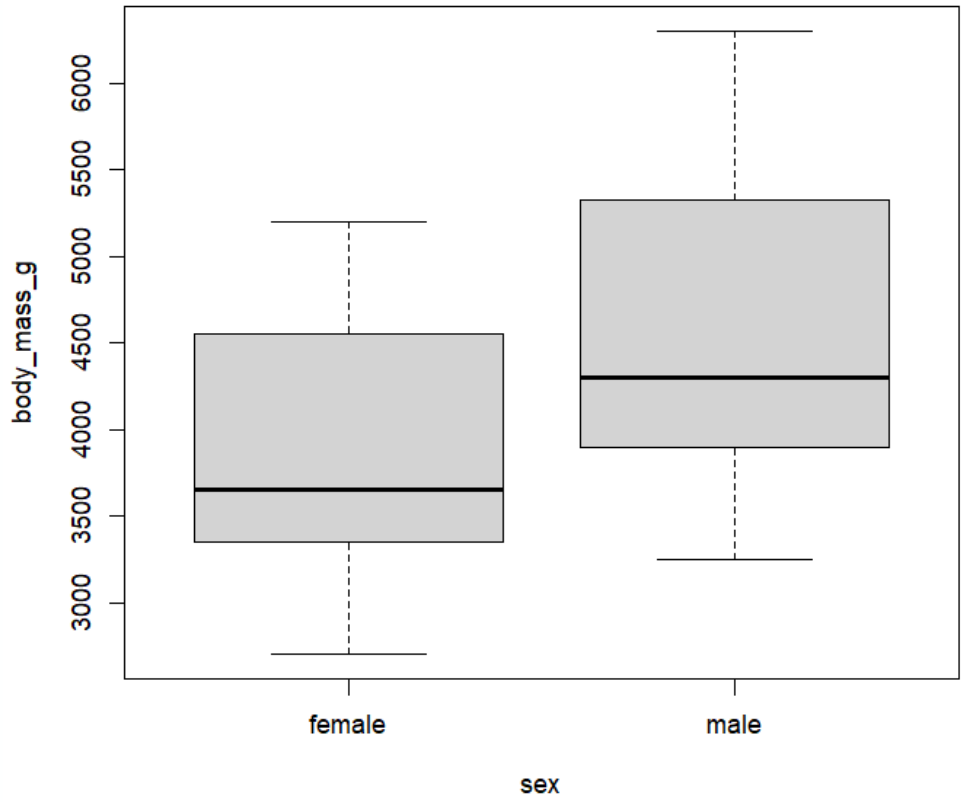
asp = specify the y / x aspect ratio of the plot

GETTING STARTED WITH BASE R

EXAMPLE

```
plot(body_mass_g ~ sex,
data = penguins)
```

OUTPUT

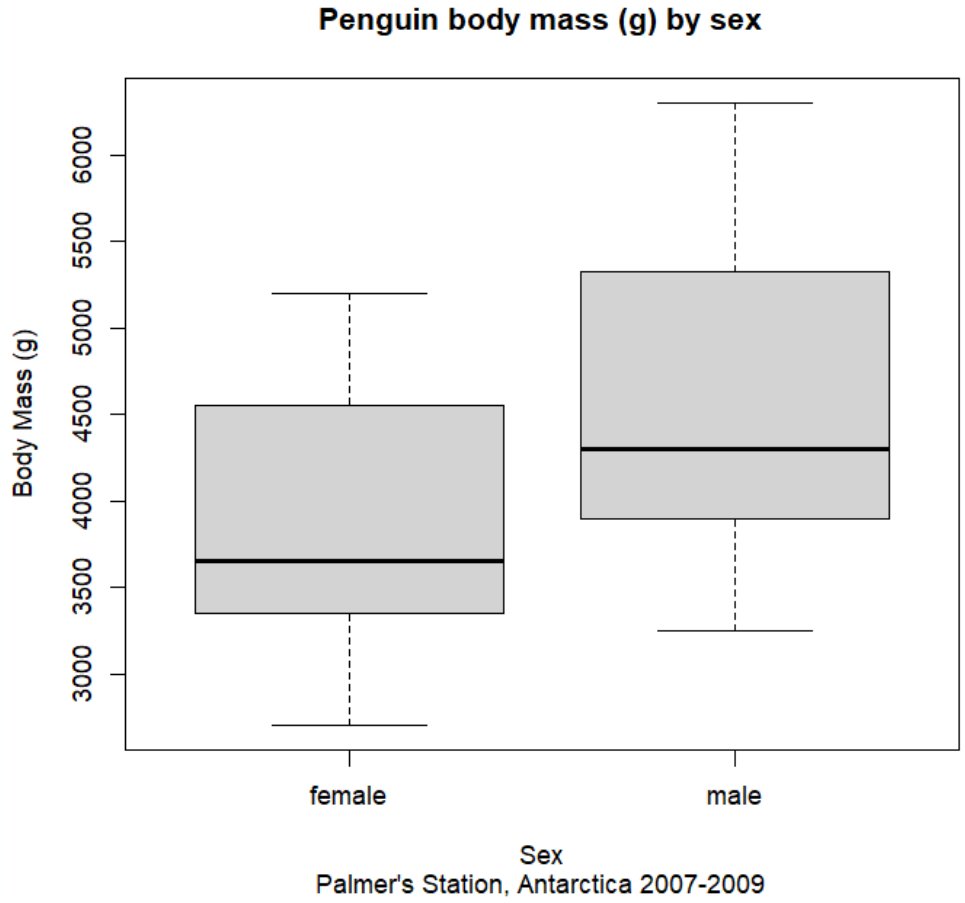


GETTING STARTED WITH BASE R

EXAMPLE

```
plot(body_mass_g ~ sex,
main = "Penguin body mass (g) by sex",
sub = "Palmer's Station, Antarctica 2007-2009",
xlab = "Sex",
ylab = "Body Mass (g)",
data = penguins)
```

OUTPUT



GETTING STARTED WITH BASE R

Even more options to pass to **plot():**

col = change the colors used in the plot
pch = change the plotting character (for *point* plots)
cex = character expansion (default scale = 1.0)
lty = change the line type (for *line* plots)
bty = the box type surrounding the plot (e.g., “o”, “n”, “1”, “7”, “c”, “u”, “]”)
las = axes label style (0:parallel to axis, 1:horizontal,
2:perpendicular, 3:vertical)

KNOWLEDGE CHECK!

- Q1) What function is our hot tip about resetting your R session's graphics device?
  - new.plot()
  - gc()
  - dev.off()
  - plot.reset()

- Q2) If *f* is a factor, plot(f) will produce a boxplot
  - TRUE
  - FALSE

BUILDING CUSTOM DATA VISUALIZATIONS IN BASE R

CUSTOM DATA VIZ IN BASE R

What happens if you want to display your data differently than the default plot()?

- Many customization options are available within the base graphics environment!
- These are called ‘low-level’ plotting commands
- Generally, they work as *layers* on top of your current graphics driver

Examples:

- Layering multiple data series on the same plot
- Adding custom labels, polygons, legends, etc., to the current plot image

LOW-LEVEL COMMANDS

- **Low-level plotting commands to customize your base R plot**

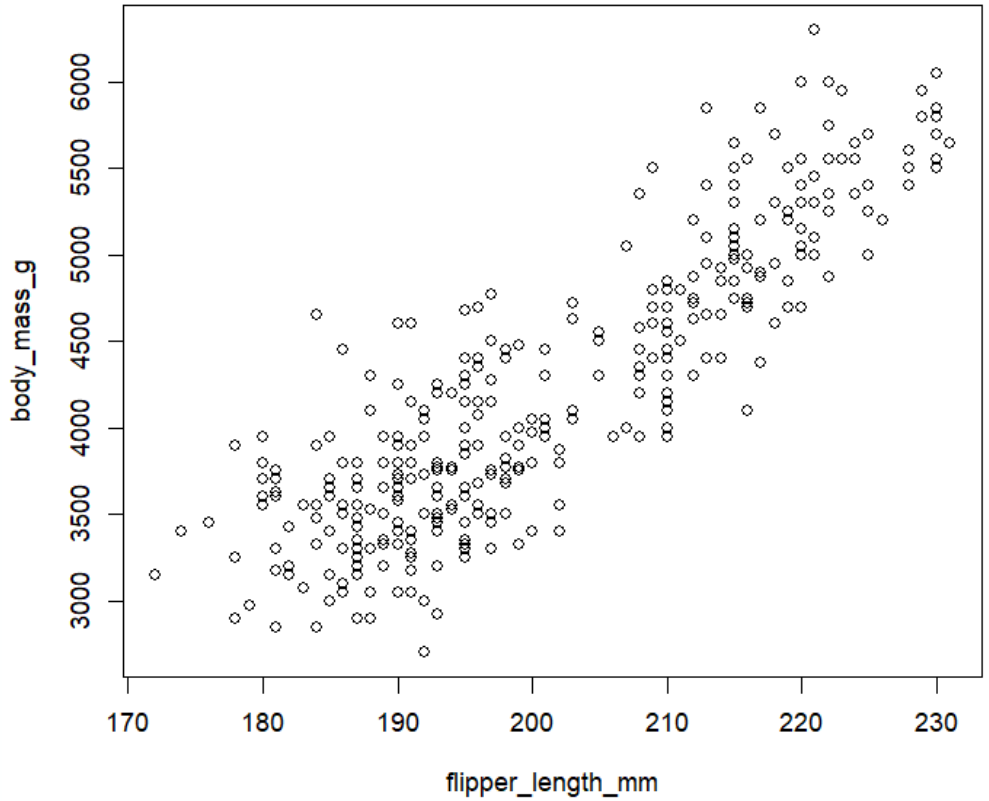
points() = adds a *points* layer to the current graphics device
lines() = adds a lines layer
abline() = adds a line between coordinates (x, y)
text() = adds text using specified coordinates
polygon() = adds a polygon at the specified coordinates (*x, y)*
legend() = adds a legend to the plot (with custom options)
title() = update the title with additional options
axis() = update the axis titles with additional options

BODY MASS VERSUS FLIPPER LENGTH

EXAMPLE

```
plot(body_mass_g ~ flipper_length_mm,
data = penguins)
```

OUTPUT

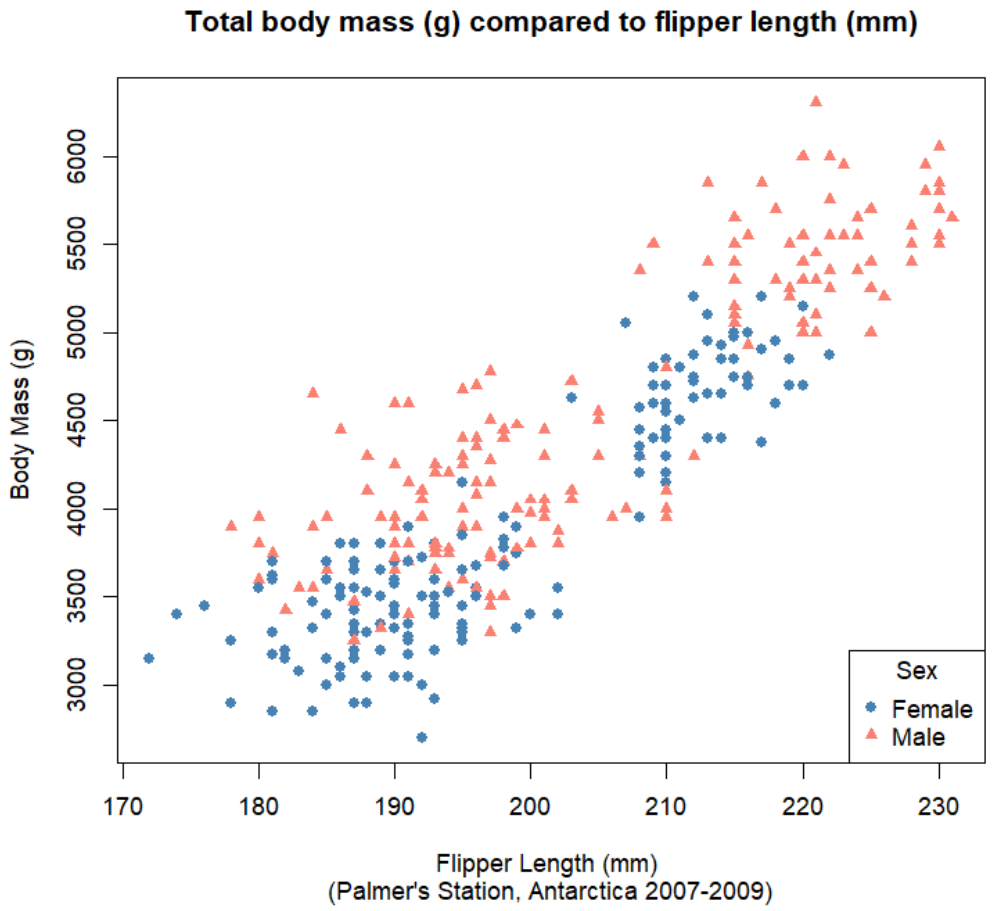


ADDING LOW-LEVEL ARGUMENTS

EXAMPLE

```
plot(body_mass_g ~ flipper_length_mm,
     main = "Total body mass (g) compared to flipper length (mm)",
     sub = "(Palmer's Station, Antarctica 2007-2009)",
     xlab = "Flipper Length (mm)", ylab = "Body Mass (g)",
     pch = c(16, 17)[as.numeric(penguins$sex)],
     col = c("steelblue", "salmon")[penguins$sex],
     data = penguins)
legend("bottomright",
       title = "Sex",
       legend = c("Female", "Male"),
       pch = c(16, 17),
       col = c("steelblue", "salmon"))
```

OUTPUT



CREATING TRENDLINES

EXAMPLE

```
female.lm <- lm(body_mass_g ~ flipper_length_mm,
data = penguins[penguins$sex == "female",])

male.lm <- lm(body_mass_g ~ flipper_length_mm,
data = penguins[penguins$sex == "male",])
```

OUTPUT

female.lm

```
Coefficients:
      (Intercept)  flipper_length_mm
          -5037.16              46.86
```

male.lm

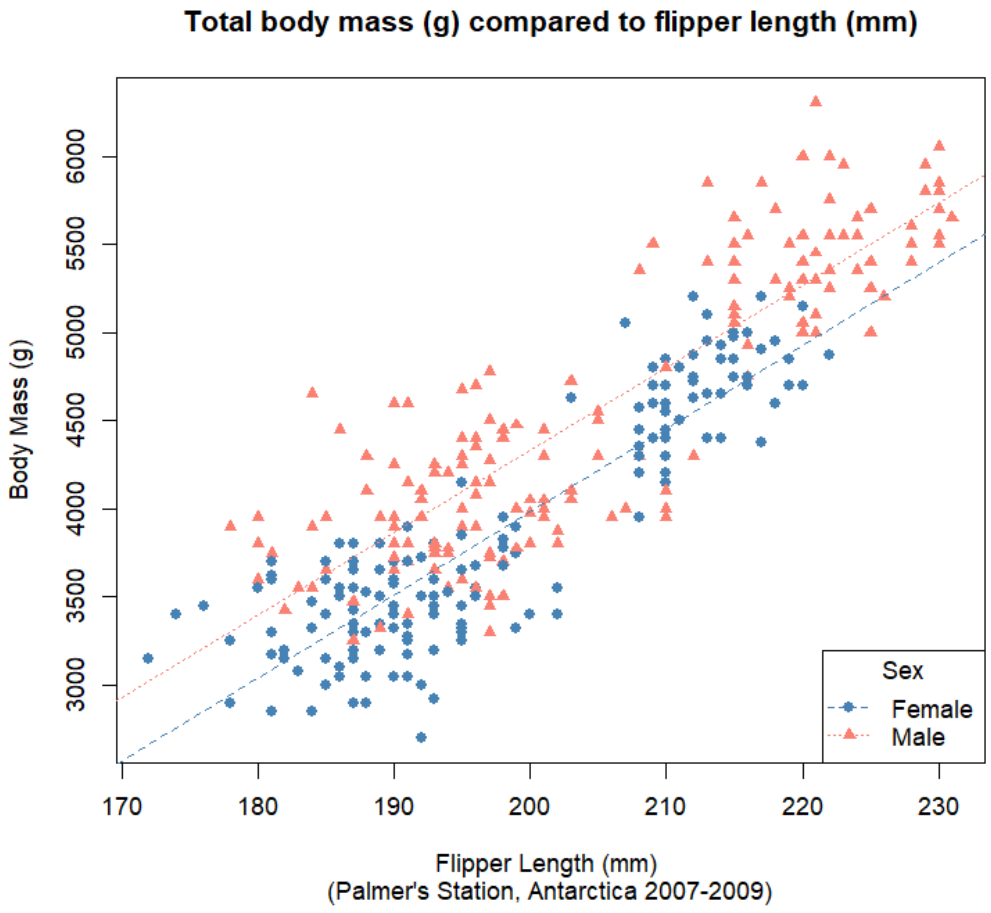
```
Coefficients:
      (Intercept)  flipper_length_mm
          -5443.96              47.15
```

ADDING TRENDLINES TO YOUR PLOT

EXAMPLE

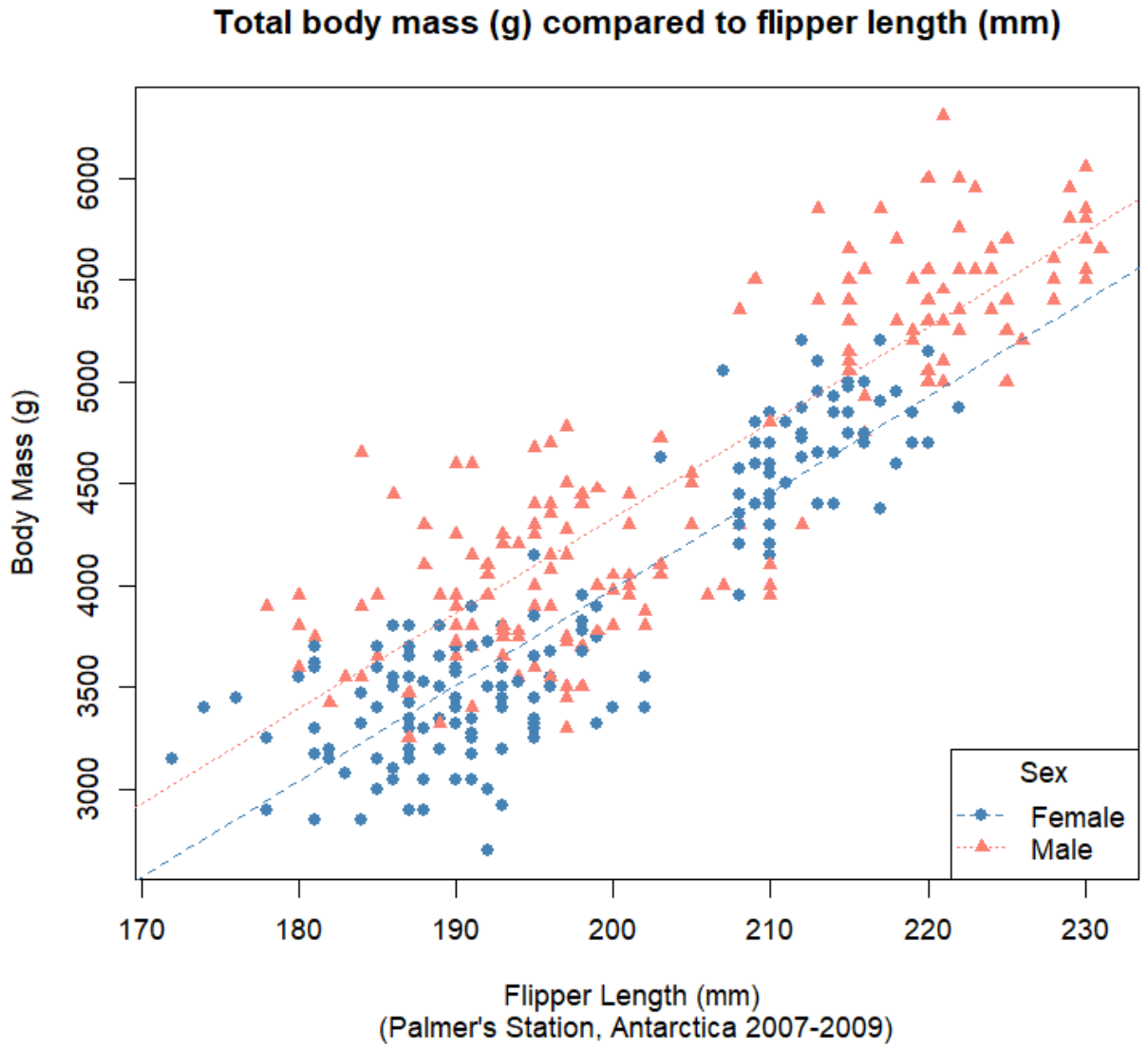
```
abline(female.lm, col = “steelblue, lty = “dashed”)
abline(male.lm, col = “salmon”, “dotted”)

legend("bottomright",
       title = "Sex",
       legend = c("Female", "Male"),
       pch = c(16, 17),
       lty = c(“dashed”, “dotted”),
       col = c("steelblue", "salmon"))
```

OUTPUT



SUMMARY: DATA VIZ WITH BASE R

- High-level plotting functions depend on data input
- Low-level plotting functions allow you to adjust parameters of the plot, and add plotting layers
- Overall, it is very possible to make publication-worthy figures using only base R graphics



FORMATTING DATA FOR VISUALIZING

MANIPULATING DATA FOR SUMMARIES

- Wide format
- A table or dataset with one row per subject
- All characteristics of that subject are stored across columns in that single row
  - E.g., an outbreak line list

WIDE FORMAT

	case_id	generation	date_infection	date_onset	date_hospitalisation	date_outcome	outcome	gender	age	age_unit	age_years	age_cat	age_cat5	hospital
1	5fe599	4	2014-05-08	2014-05-13	2014-05-15	NA	NA	m	2	years	2	0-4	0-4	Other
2	8689b7	4	NA	2014-05-13	2014-05-14	2014-05-18	Recover	f	3	years	3	0-4	0-4	Missing
3	11f8ea	2	NA	2014-05-16	2014-05-18	2014-05-30	Recover	m	56	years	56	50-69	55-59	St. Mark's Maternity Hospital (SMMH)
4	b8812a	3	2014-05-04	2014-05-18	2014-05-20	NA	NA	f	18	years	18	15-19	15-19	Port Hospital
5	893f25	3	2014-05-18	2014-05-21	2014-05-22	2014-05-29	Recover	m	3	years	3	0-4	0-4	Military Hospital
6	be99c8	3	2014-05-03	2014-05-22	2014-05-23	2014-05-24	Recover	f	16	years	16	15-19	15-19	Port Hospital
7	07e3e8	4	2014-05-22	2014-05-27	2014-05-29	2014-06-01	Recover	f	16	years	16	15-19	15-19	Missing
8	369449	4	2014-05-28	2014-06-02	2014-06-03	2014-06-07	Death	f	0	years	0	0-4	0-4	Missing
9	f393b4	4	NA	2014-06-05	2014-06-06	2014-06-18	Recover	m	61	years	61	50-69	60-64	Missing
10	1389ca	4	NA	2014-06-05	2014-06-07	2014-06-09	Death	f	27	years	27	20-29	25-29	Missing
11	2978ac	4	2014-05-30	2014-06-06	2014-06-08	2014-06-15	Death	m	12	years	12	10-14	10-14	Port Hospital
12	57a565	4	2014-05-28	2014-06-13	2014-06-15	NA	Death	m	42	years	42	30-49	40-44	Military Hospital
13	fc15ef	6	2014-06-14	2014-06-16	2014-06-17	2014-07-09	Recover	m	19	years	19	15-19	15-19	Missing
14	2eaa9a	5	2014-06-07	2014-06-17	2014-06-17	NA	Recover	f	7	years	7	5-9	5-9	Missing
15	bbfa93	6	2014-06-09	2014-06-18	2014-06-20	2014-06-30	NA	f	7	years	7	5-9	5-9	Other

WIDE FORMAT

- Function: pivot_wider()

```
df_intermediate <-
  linelist %>%
  count(age_cat, gender)

df_intermediate

table_wide <-
  df_intermediate %>%
  pivot_wider(id_cols = age_cat,
              names_from = gender,
              values_from = n)

table_wide
```

```
> df_intermediate
   age_cat gender   n
1      0-4      f 640
2      0-4      m 416
3      0-4   <NA>  39
4    10-14      f 518
5    10-14      m 383
6    10-14   <NA>  40
7    15-19      f 359
8    15-19      m 364
9    15-19   <NA>  20
10   20-29      f 468
11   20-29      m 575
12   20-29   <NA>  30
13   30-49      f 179
14   30-49      m 557
15   30-49   <NA>  18
16     5-9      f 641
17     5-9      m 412
18     5-9   <NA>  42
19   50-69      f   2
20   50-69      m  91
21   50-69   <NA>   2
22     70+      m   5
23     70+   <NA>   1
24    <NA>   <NA>  86
```

```
> table_wide
# A tibble: 9 × 4
  age_cat     f     m  `NA`
  <chr>   <int> <int> <int>
1 0-4       640   416    39
2 10-14     518   383    40
3 15-19     359   364    20
4 20-29     468   575    30
5 30-49     179   557    18
6 5-9       641   412    42
7 50-69       2    91     2
8 70+        NA     5     1
9 NA         NA    NA    86
```

WIDE FORMAT

Advantages:

- Useful for creating high quality tables for reports, publications, etc.
- Easy to read
- Easy to enter (i.e., spreadsheets)

Disadvantages:

- Often further manipulation is required for use in graphing

MANIPULATING DATA FOR VISUALIZATION

- Long format
- Information for each subject is contained in multiple rows
- Characteristics for each subject are contained in multiple columns across those rows
- Separate columns containing values and the context of those values
  - E.g., A column containing temperature values and a column with data assigning that value to daily minimum, maximum, or average

LONG FORMAT

- Function: pivot_longer()

	location_name	data_date	submitted_date	Province	District	malaria_rdt_0-4	malaria_rdt_5-14	malaria_rdt_15	malaria_tot	newid
1	Facility 5	2020-08-08	2020-08-12	North	Bolo	19	15	15	49	5
2	Facility 19	2020-08-09	2020-08-12	North	Bolo	5	3	24	32	19
3	Facility 5	2020-08-09	2020-08-12	North	Bolo	11	11	13	35	5
4	Facility 19	2020-08-10	2020-08-12	North	Bolo	4	7	41	52	19
5	Facility 5	2020-08-10	2020-08-12	North	Bolo	15	14	13	42	5
6	Facility 6	2020-08-10	2020-08-12	North	Dingo	4	0	3	7	6
7	Facility 1	2020-08-11	2020-08-12	North	Spring	11	12	23	46	1
8	Facility 10	2020-08-11	2020-08-12	North	Bolo	0	10	24	34	10
9	Facility 11	2020-08-11	2020-08-12	North	Spring	17	15	28	60	11
10	Facility 12	2020-08-11	2020-08-12	North	Dingo	36	36	62	134	12
11	Facility 13	2020-08-11	2020-08-12	North	Bolo	0	0	0	0	13
12	Facility 14	2020-08-11	2020-08-12	North	Dingo	5	1	10	16	14
13	Facility 15	2020-08-11	2020-08-12	North	Barnard	6	3	2	11	15
14	Facility 16	2020-08-11	2020-08-12	North	Barnard	9	9	16	34	16
15	Facility 17	2020-08-11	2020-08-12	North	Barnard	4	4	1	9	17
16	Facility 18	2020-08-11	2020-08-12	North	Bolo	7	0	6	13	18
17	Facility 19	2020-08-11	2020-08-12	North	Bolo	12	2	18	32	19
18	Facility 2	2020-08-11	2020-08-12	North	Bolo	11	10	5	26	2
19	Facility 20	2020-08-11	2020-08-12	North	Barnard	14	9	14	37	20

LONG FORMAT

- Function: pivot_longer()

```
df_long <- count_data %>%
  pivot_longer(
    cols = c(`malaria_rdt_0-4`, `malaria_rdt_5-14`, `malaria_rdt_15`,
             `malaria_tot`)
  )

df_long
```

```
> df_long
# A tibble: 12,152 x 8
   location_name data_date  submitted_date Province District newid name             value
   <chr>         <date>     <date>         <chr>    <chr>    <int> <chr>            <int>
 1 Facility 1    2020-08-11 2020-08-12     North    Spring       1 malaria_rdt_0-4     11
 2 Facility 1    2020-08-11 2020-08-12     North    Spring       1 malaria_rdt_5-14    12
 3 Facility 1    2020-08-11 2020-08-12     North    Spring       1 malaria_rdt_15      23
 4 Facility 1    2020-08-11 2020-08-12     North    Spring       1 malaria_tot         46
 5 Facility 2    2020-08-11 2020-08-12     North    Bolo         2 malaria_rdt_0-4     11
 6 Facility 2    2020-08-11 2020-08-12     North    Bolo         2 malaria_rdt_5-14    10
 7 Facility 2    2020-08-11 2020-08-12     North    Bolo         2 malaria_rdt_15       5
 8 Facility 2    2020-08-11 2020-08-12     North    Bolo         2 malaria_tot         26
 9 Facility 3    2020-08-11 2020-08-12     North    Dingo        3 malaria_rdt_0-4      8
10 Facility 3    2020-08-11 2020-08-12     North    Dingo        3 malaria_rdt_5-14     5
```

LONG FORMAT

Advantages:

- Easy to use in creating graphics

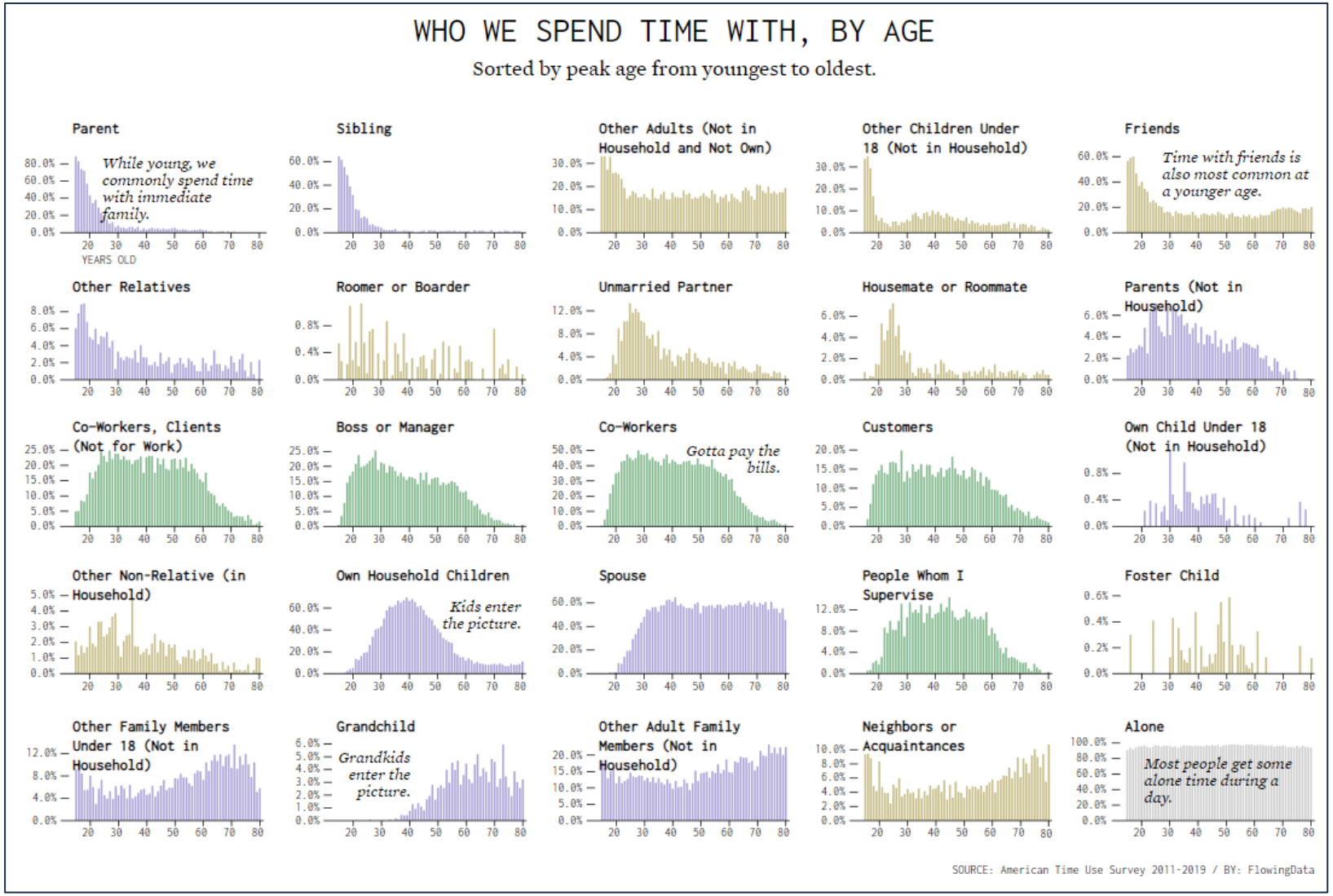
Disadvantages:

- Density makes it difficult to read
- Manipulation often required to produce high quality tables for reports, publications, etc.

SUMMARY

- Data formatting can be used to easily facilitate specific tasks
- Wide format stores data in a way that is easier to read, and format into summary
- Long format stores data more densely and is easier to use for graphing in statistical and data science-oriented software
- Knowing what you want to accomplish at the outset will help you decide what format will be most useful and what data elements should be included

TAKE A BREAK!



Please come back from break by 1:30pm ET

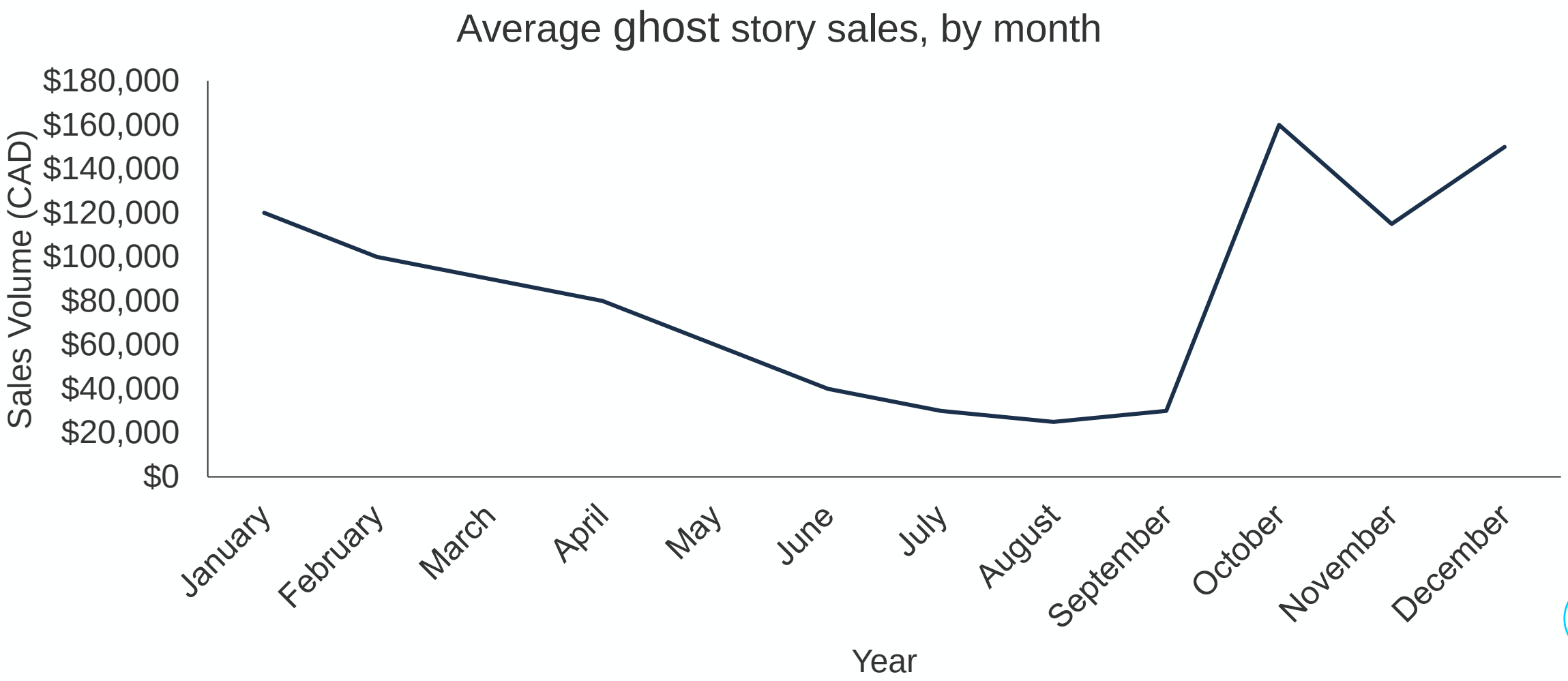
ENERGIZER: SILLY HATS AND GLITTERY PANTS

ENERGIZER: SILLY HATS AND GLITTERY PANTS

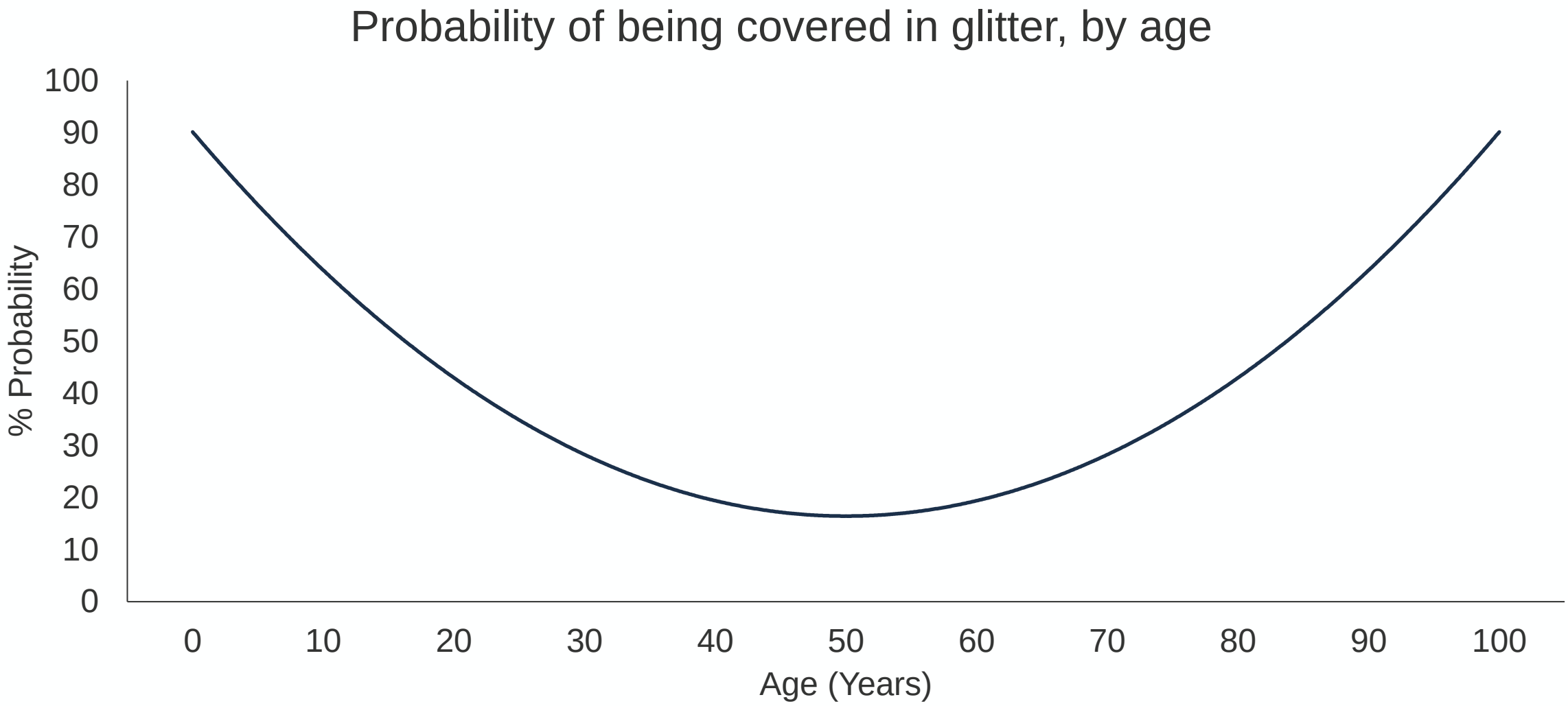
- You will be sent into small groups to complete this activity.
- Each group will be provided a link to a Google document with a figure depicting a silly, nonsensical relationship (links are in your participant guide)
- You will have 5 minutes to develop your theory behind the relationship depicted
- Each group will have 2 minutes to explain the relationship in the figure to their peers in the large group

ENERGIZER: SILLY HATS AND GLITTERY PANTS

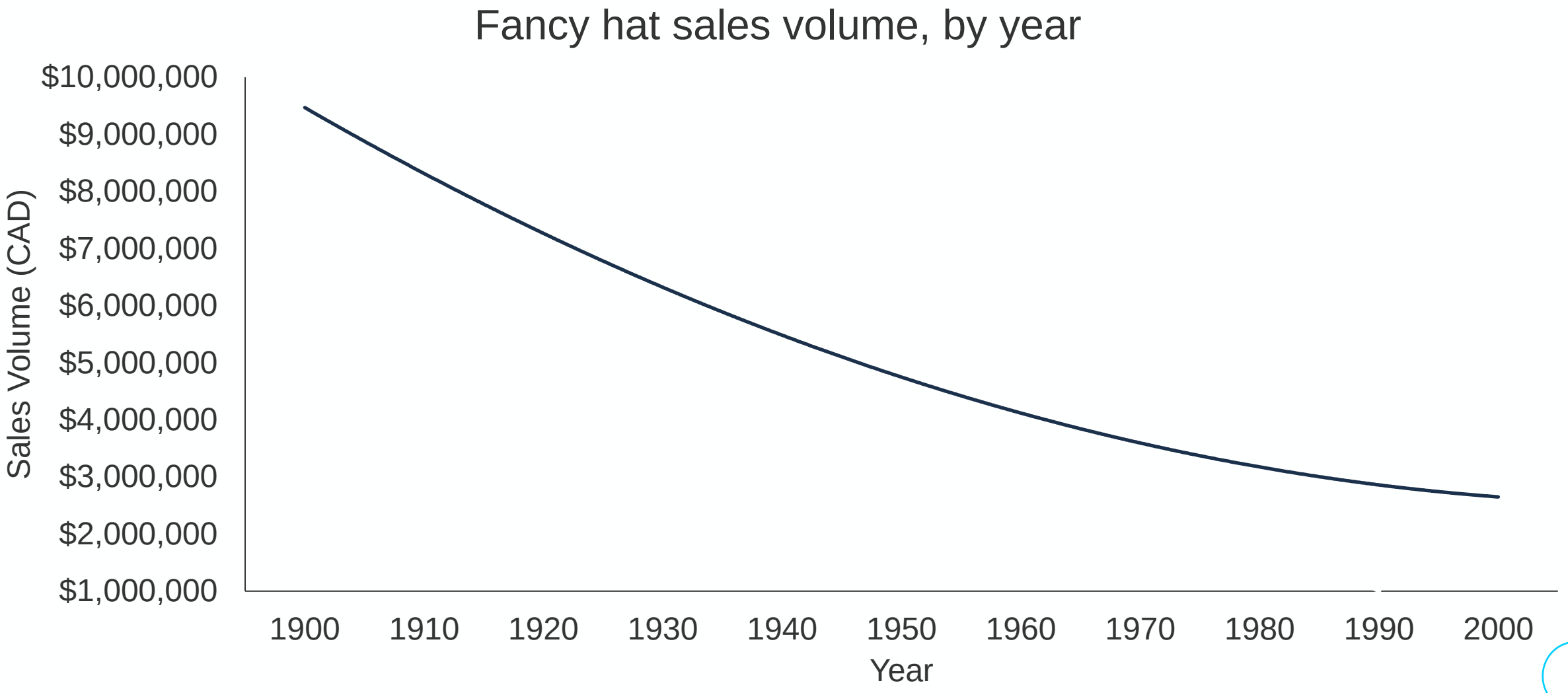
Example:



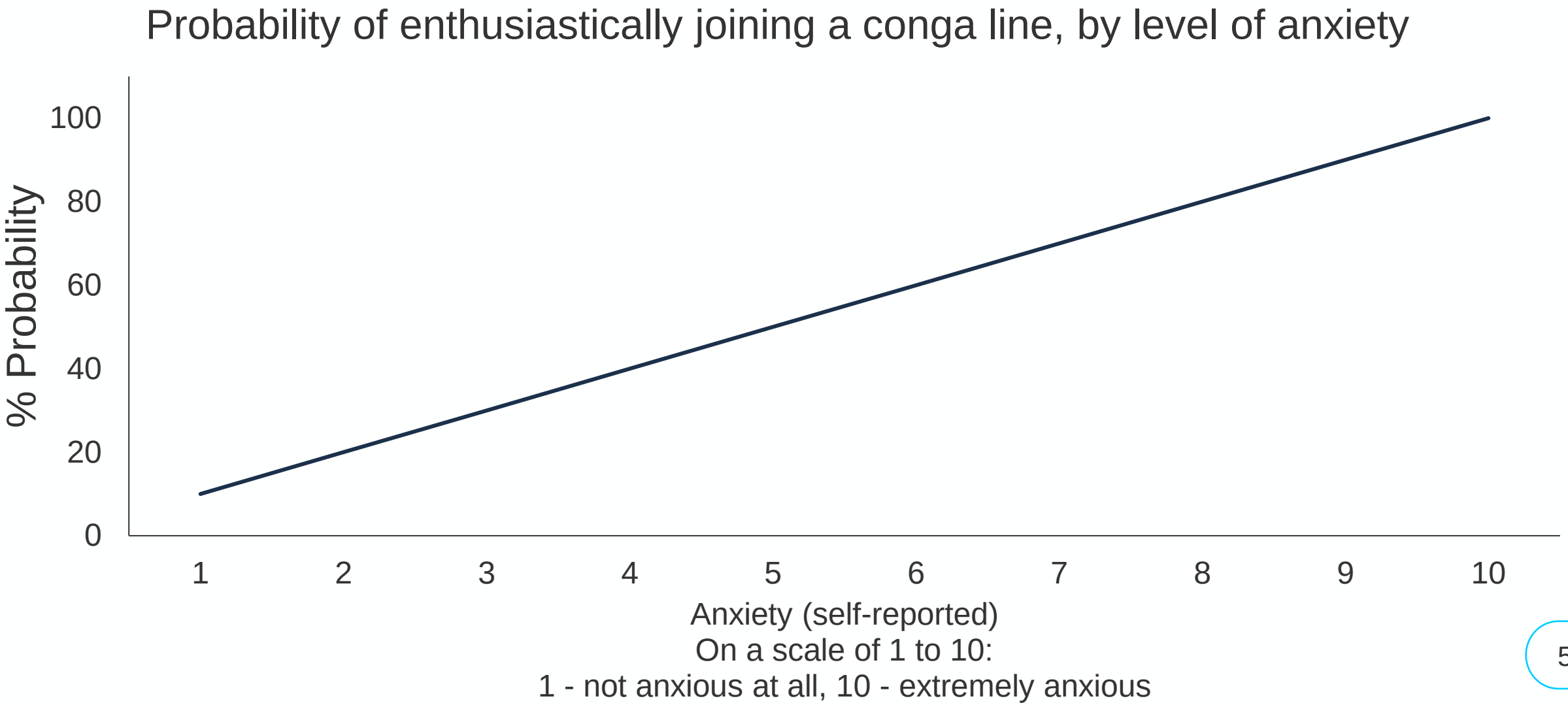
GROUP 1


Probability of being covered in glitter, by age
% Probability
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
Age (Years)

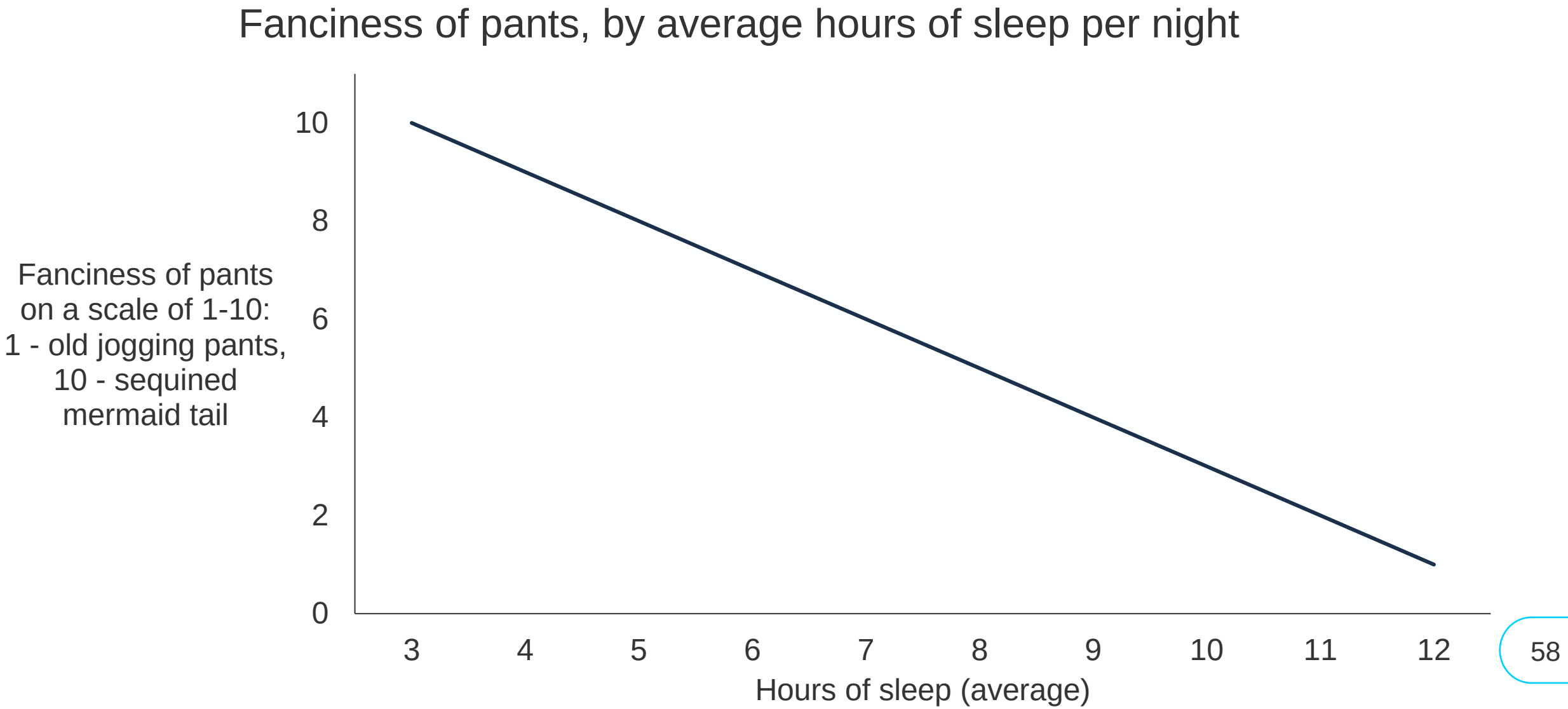
GROUP 2


Fancy hat sales volume, by year
$10,000,000
$9,000,000
$8,000,000
$7,000,000
$6,000,000
$5,000,000
$4,000,000
$3,000,000
$2,000,000
$1,000,000
Sales Volume (CAD)
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
Year

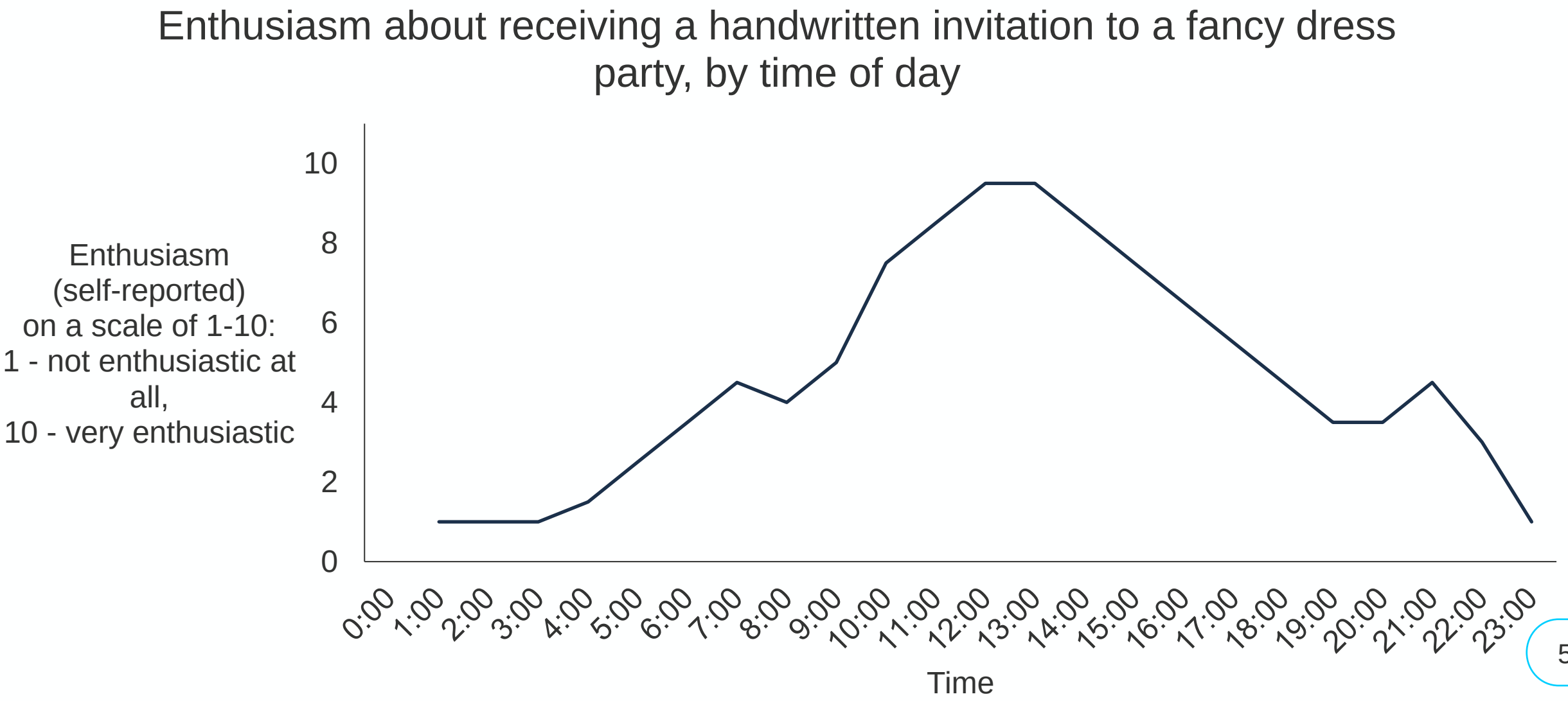
GROUP 3


Probability of enthusiastically joining a conga line, by level of anxiety
% Probability
100
80
60
40
20
0
1
2
3
4
5
6
7
8
9
10
Anxiety (self-reported)
On a scale of 1 to 10:
1 - not anxious at all, 10 - extremely anxious

GROUP 4


Fanciness of pants, by average hours of sleep per night
Fanciness of pants on a scale of 1-10: 1 - old jogging pants, 10 - sequined mermaid tail
10
8
6
4
2
0
3
4
5
6
7
8
9
10
11
12
Hours of sleep (average)

GROUP 5


Enthusiasm about receiving a handwritten invitation to a fancy dress party, by time of day
Enthusiasm (self-reported) on a scale of 1-10: 1 - not enthusiastic at all, 10 - very enthusiastic
10
8
6
4
2
0
0:00
1:00
2:00
3:00
4:00
5:00
6:00
7:00
8:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
Time

GGPLOT

INTRODUCING: GGPLOT

A Layered Grammar of Graphics

Hadley WICKHAM

A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the "scatterplot") and gain insight into the deep structure that underlies statistical graphics. This article builds on Wilkinson, Anand, and Grossman (2005), describing extensions and refinements developed while building an open source implementation of the grammar of graphics for R, `ggplot2`.

The topics in this article include an introduction to the grammar by working through the process of creating a plot, and discussing the components that we need. The grammar is then presented formally and compared to Wilkinson's grammar, highlighting the hierarchy of defaults, and the implications of embedding a graphical grammar into a programming language. The power of the grammar is illustrated with a selection of examples that explore different components and their interactions, in more detail. The article concludes by discussing some perceptual issues, and thinking about how we can build on the grammar to learn how to create graphical "poems."

Supplemental materials are available online.

Key Words: Grammar of graphics; Statistical graphics.

“GRAMMAR” OF GRAPHICS?

- A grammar can be described as “the fundamental principles or rules of an art or science”

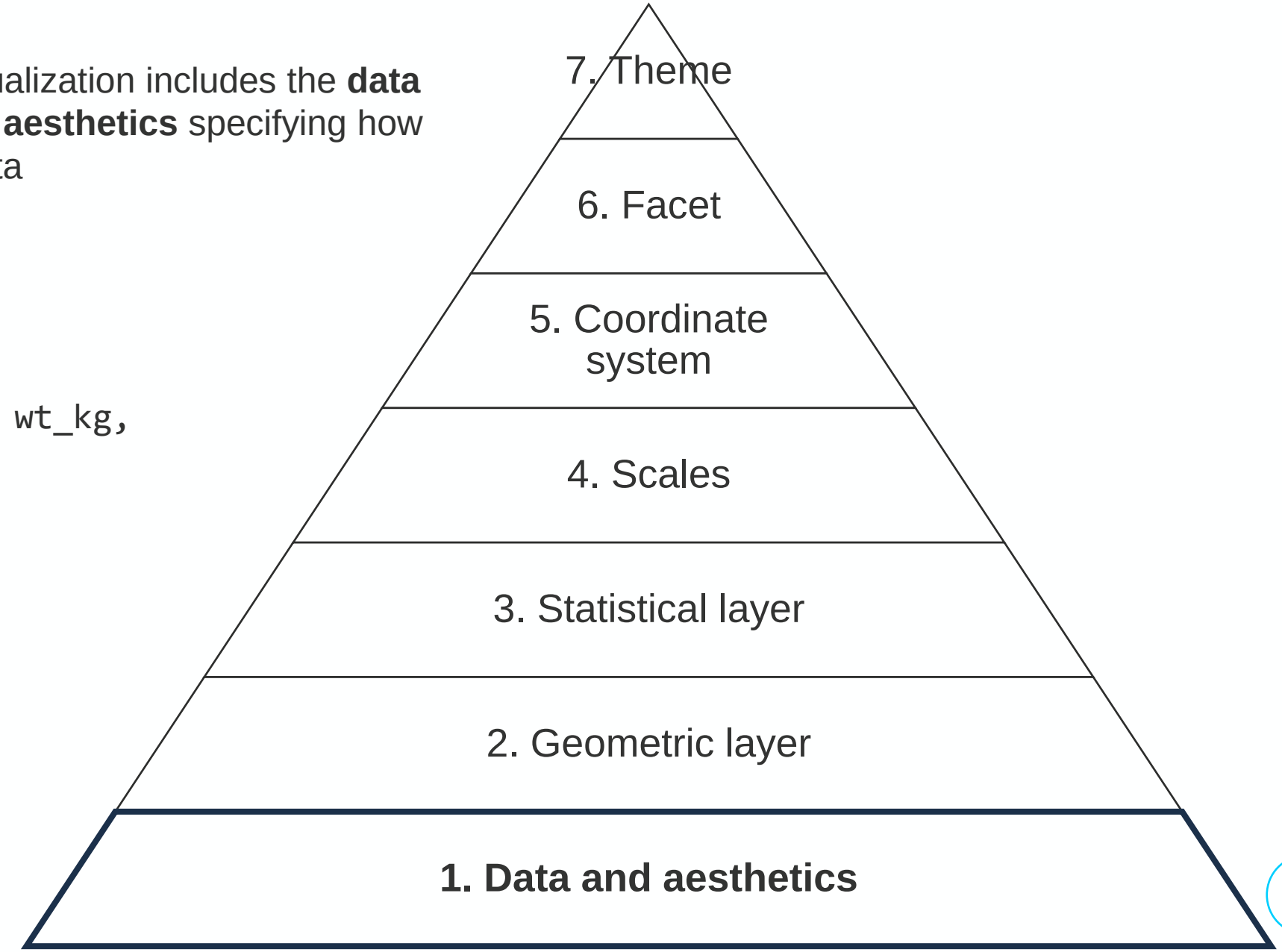
- Ggplot designed using a ‘layered’ grammar:
  - Starting with your data, it facilitates iteratively adding multiple layers onto a basic plot to achieve the custom result you want.

GRAMMAR OF GRAPHICS

1. The foundation of your visualization includes the **data** you want to visualize and the **aesthetics** specifying how you want to present those data

e.g.,

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg,
           color = age,
           size = age)
)
```

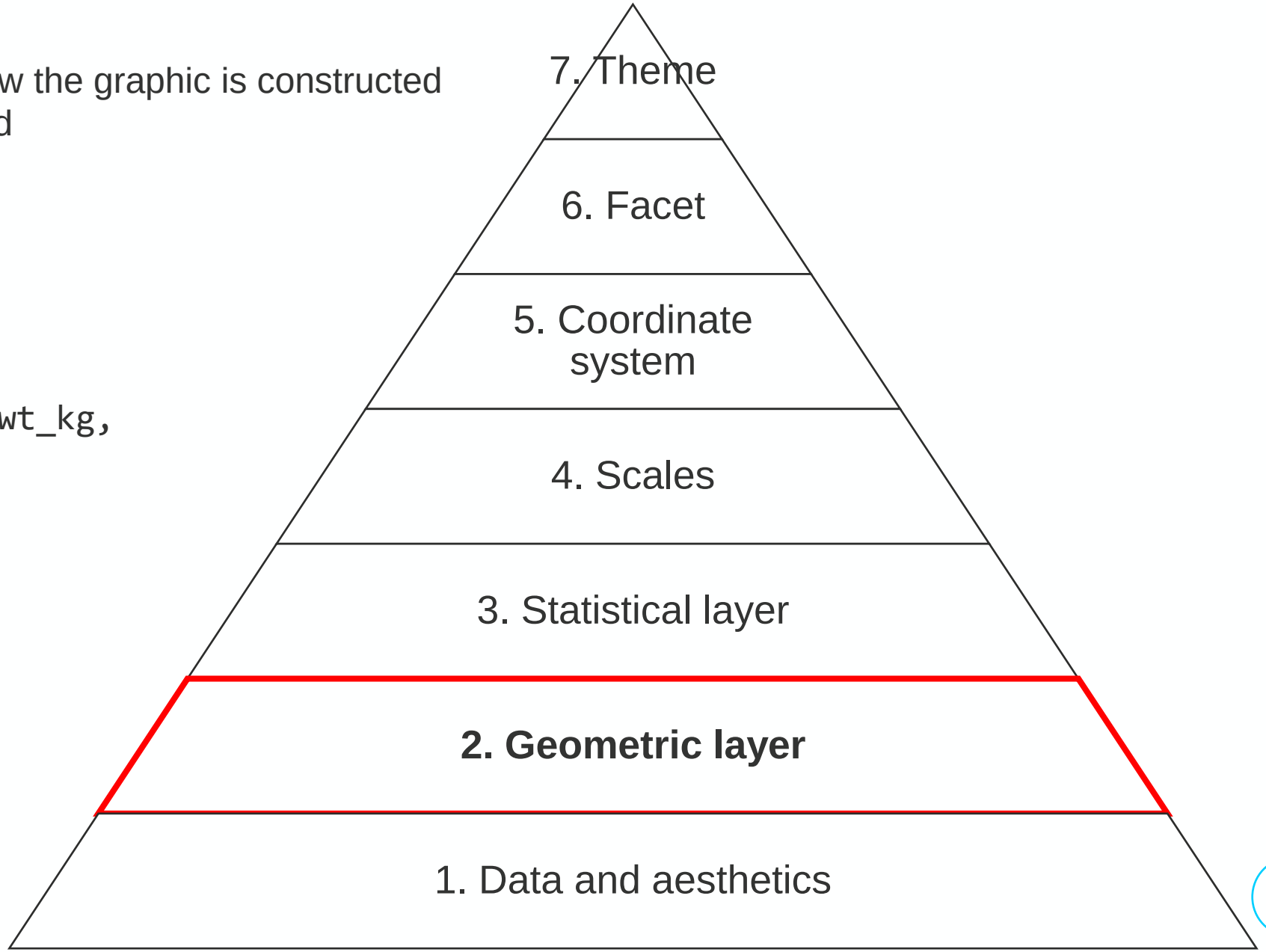


GRAMMAR OF GRAPHICS

2. Geometric layers specify how the graphic is constructed and how the data are displayed

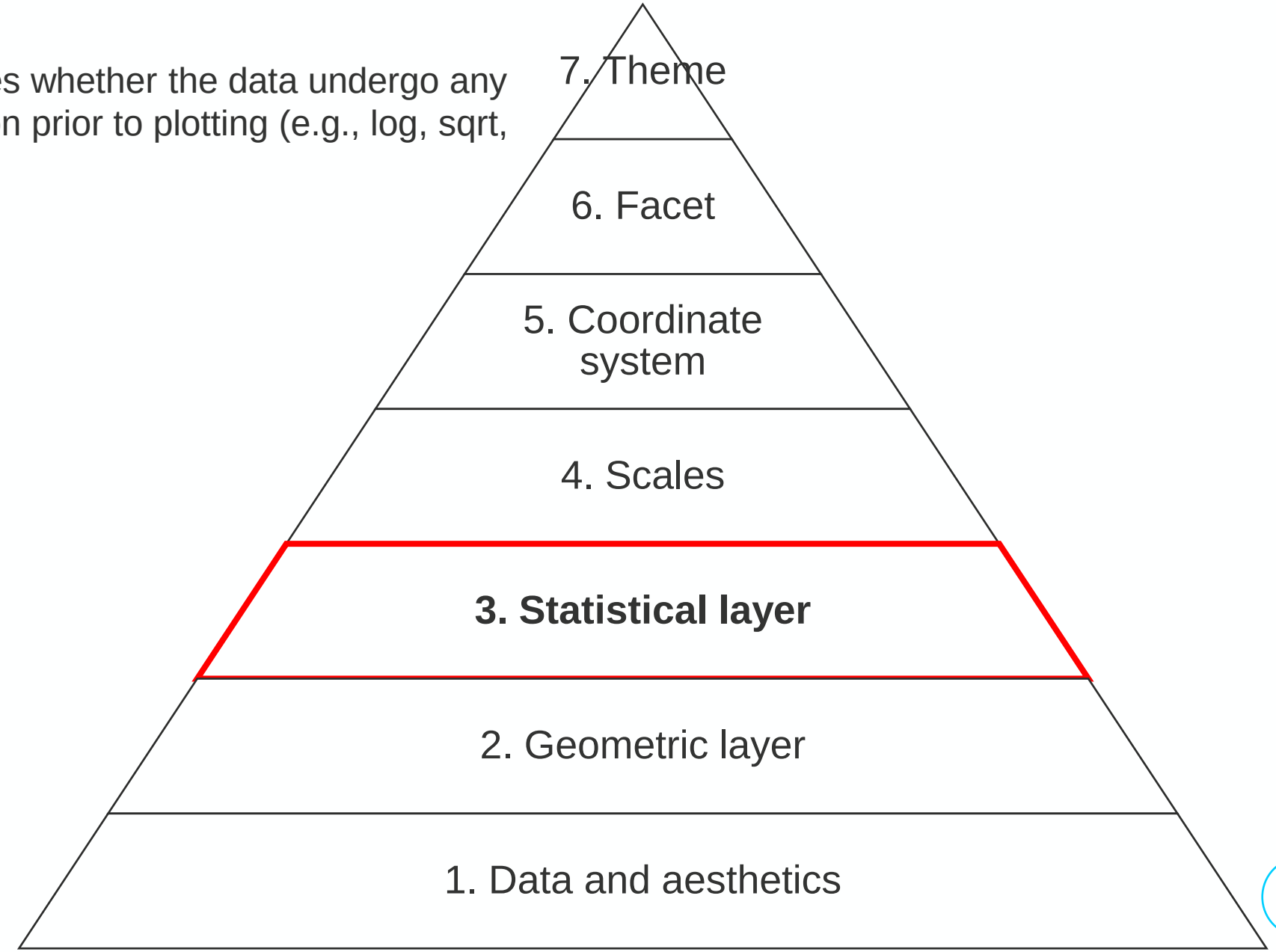
e.g.,

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg,
           color = age,
           size = age)
) +
 geom_point()
```



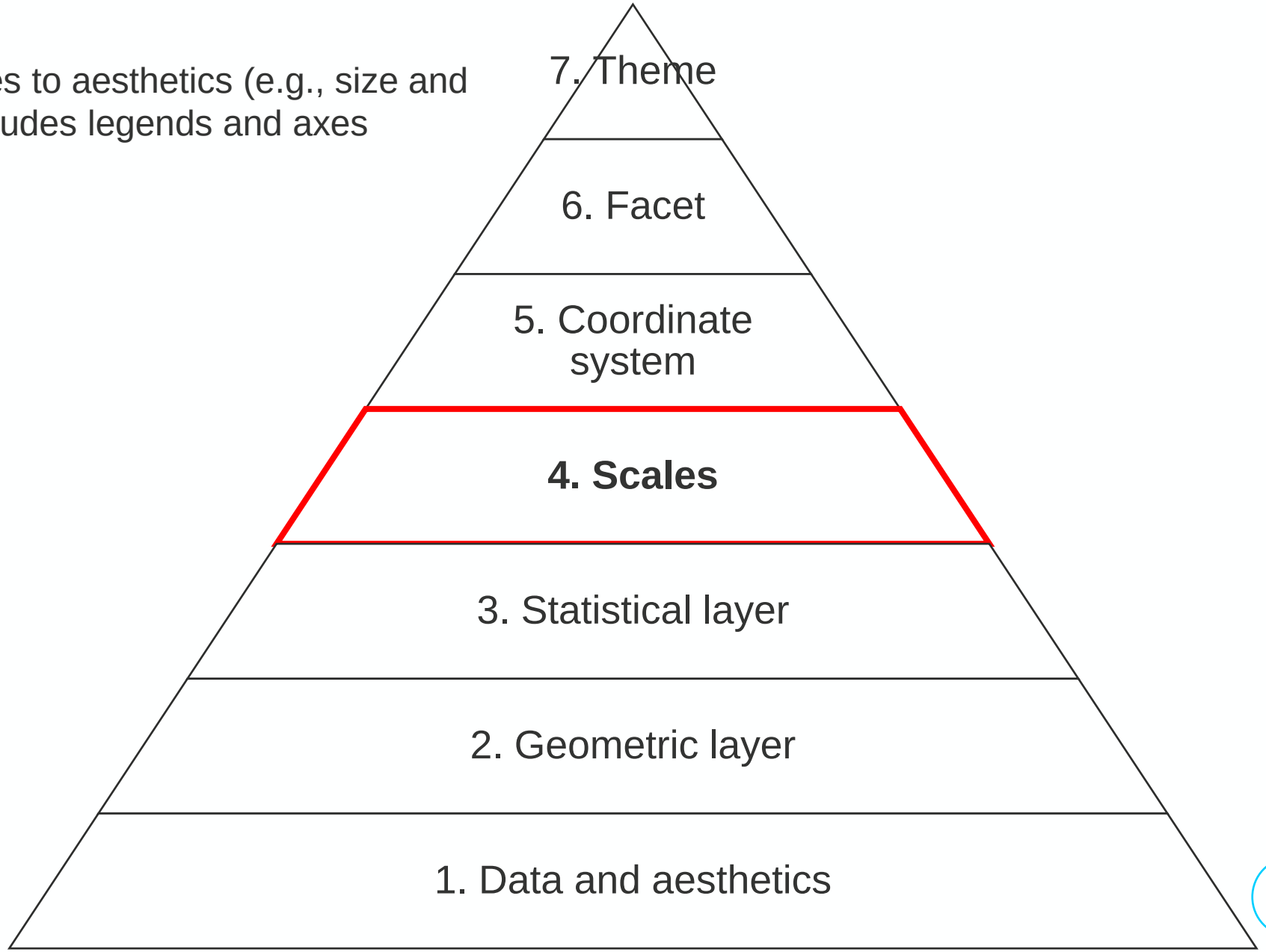
GRAMMAR OF GRAPHICS

3. The statistical layer specifies whether the data undergo any sort of statistical transformation prior to plotting (e.g., log, sqrt, etc.)



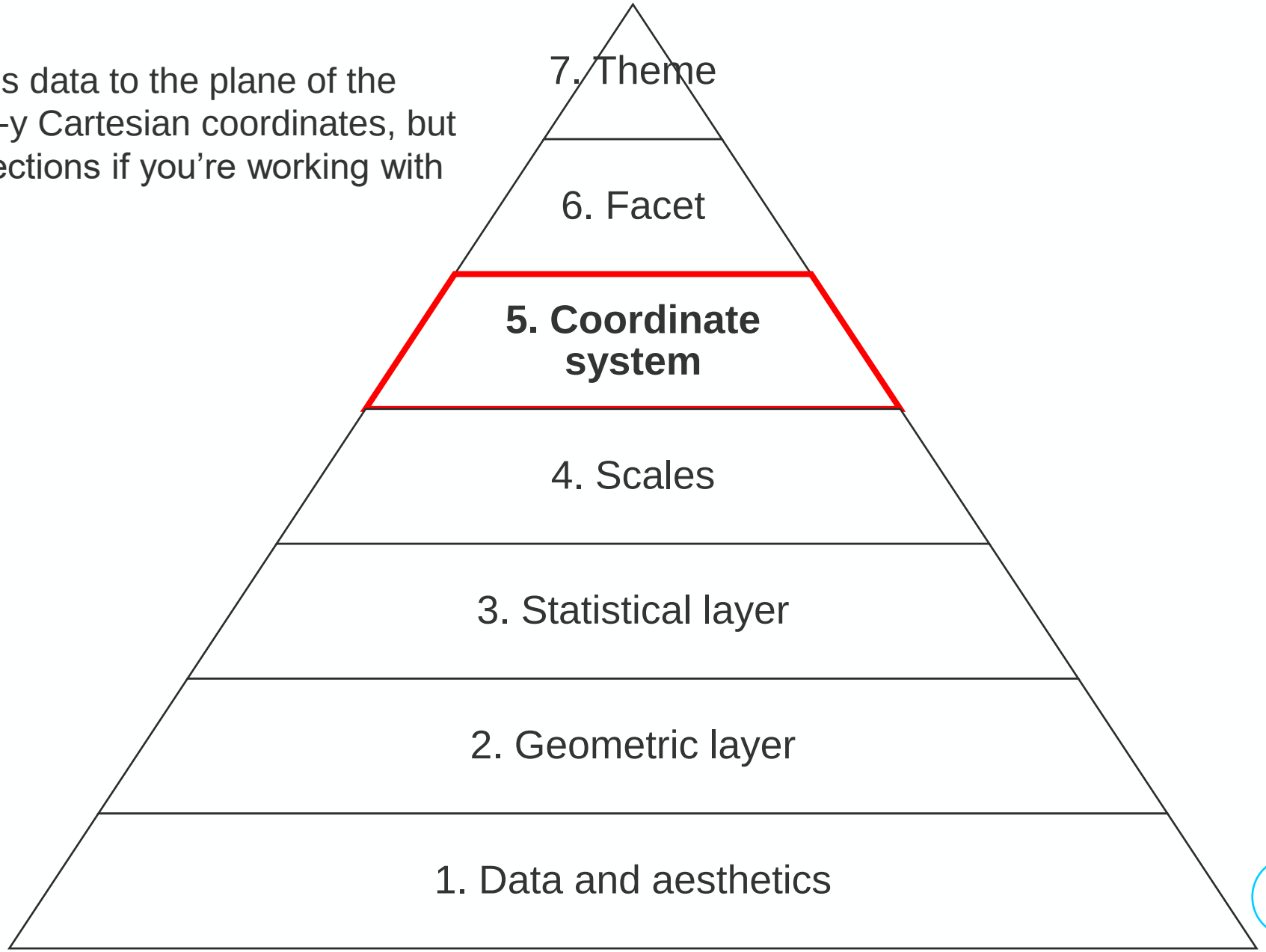
GRAMMAR OF GRAPHICS

4. Scales adjust the data values to aesthetics (e.g., size and shape of data points), also includes legends and axes



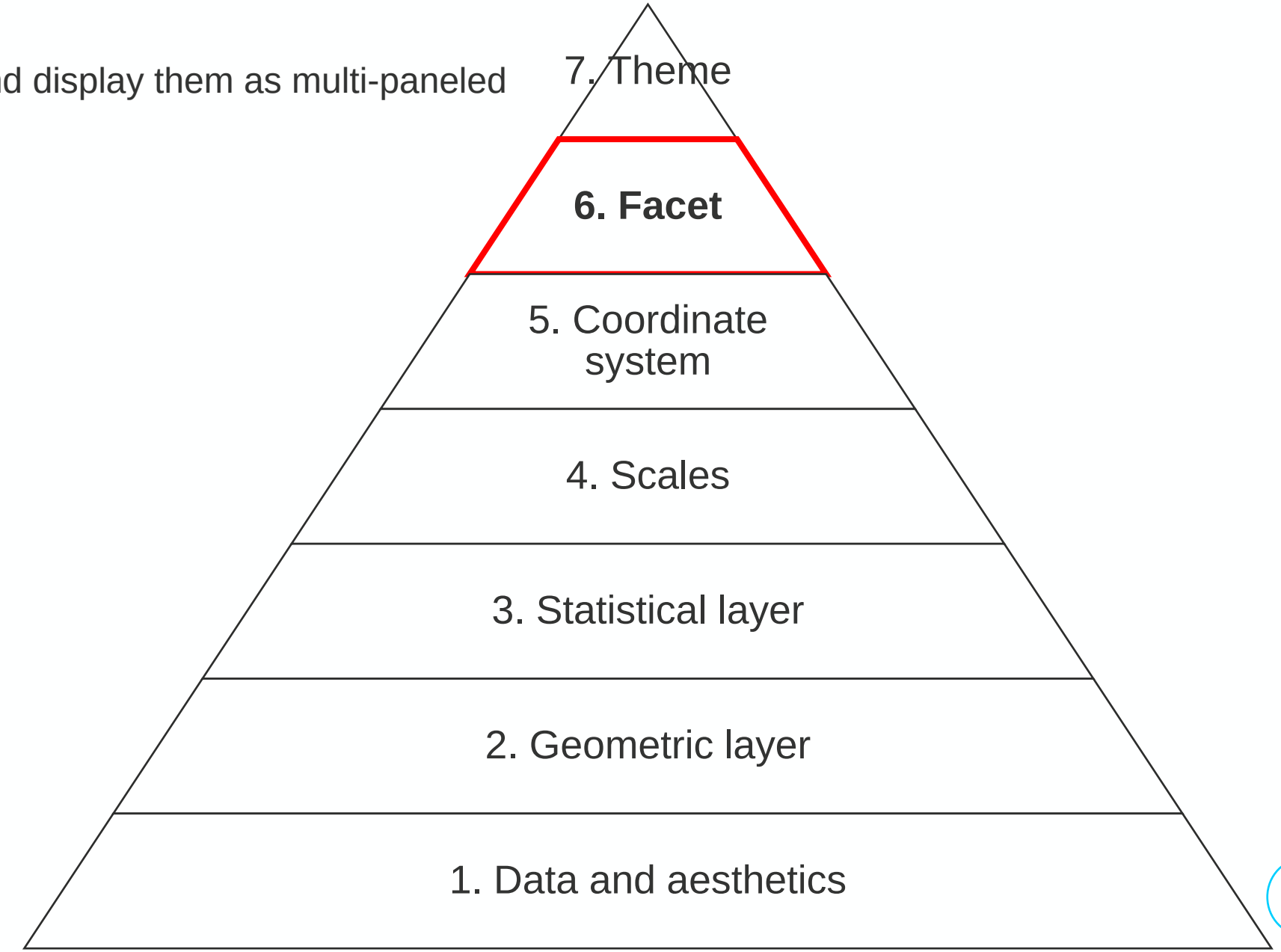
GRAMMAR OF GRAPHICS

5. Coordinate system translates data to the plane of the graphic for visualization (i.e., x-y Cartesian coordinates, but also this can include map projections if you're working with spatial data)



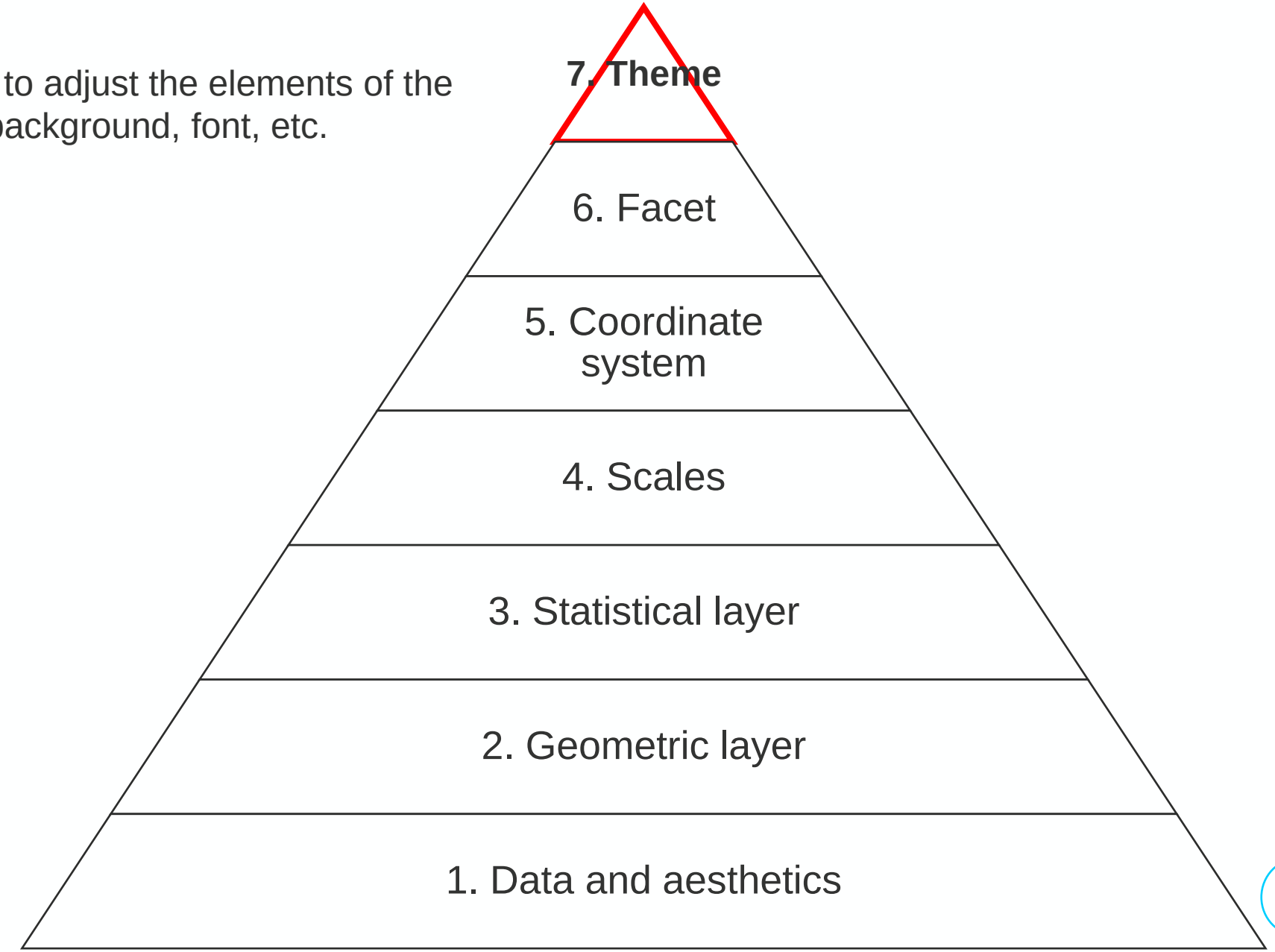
GRAMMAR OF GRAPHICS

6. Facets subset your data and display them as multi-paneled plots.



GRAMMAR OF GRAPHICS

7. Theme elements allow you to adjust the elements of the plot not related to data: e.g., background, font, etc.



GRAMMAR OF GRAPHICS

- Graphics are built iteratively using a layered approach based on this specific grammar of independent components
- The grammar and coding ultimately relies on our human thinking brain!
- Need to decide what graphic(s) you need for the question you intend to answer and to tell the story(ies) that need to be told from the data

EXAMPLE

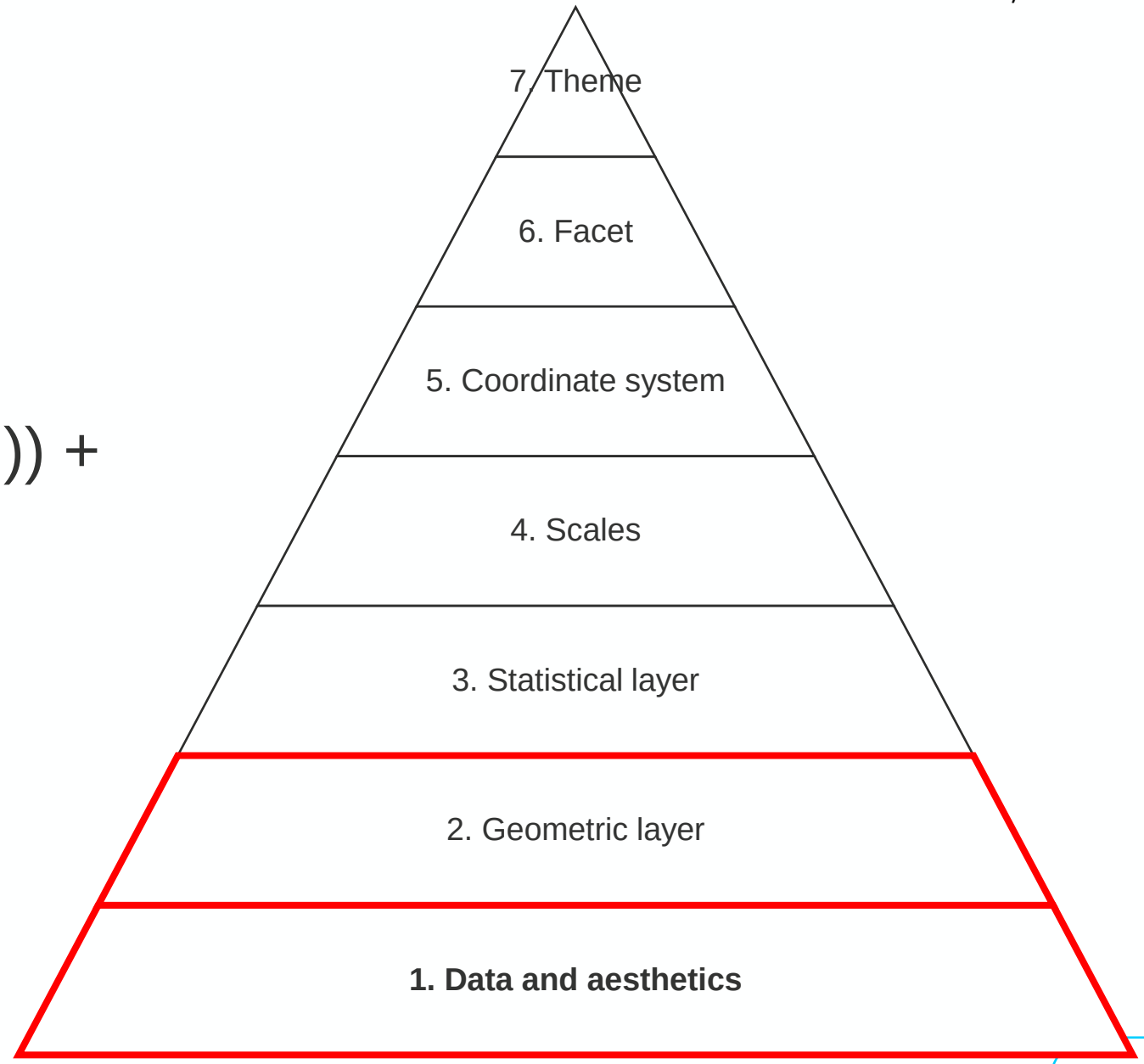
- Will walk through some selected code from Epidemiologist R Handbook created by AppliedEpi.org

- Review the layered approach from previous slides and thought processes to get from there to some fancy graphical representations of data!

  - Listen in or code along while we talk

  - Share script containing code from the book

```
ggplot(data = linelist,

       aes(x = age, y = wt_kg)) +

geom_point()
```



```
ggplot(data = linelist,

       aes(x = age, y = wt_kg)) +

geom_point()
```

Creates the plot and specifies the data

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg)) +
geom_point()
```

Creates the plot and specifies the data

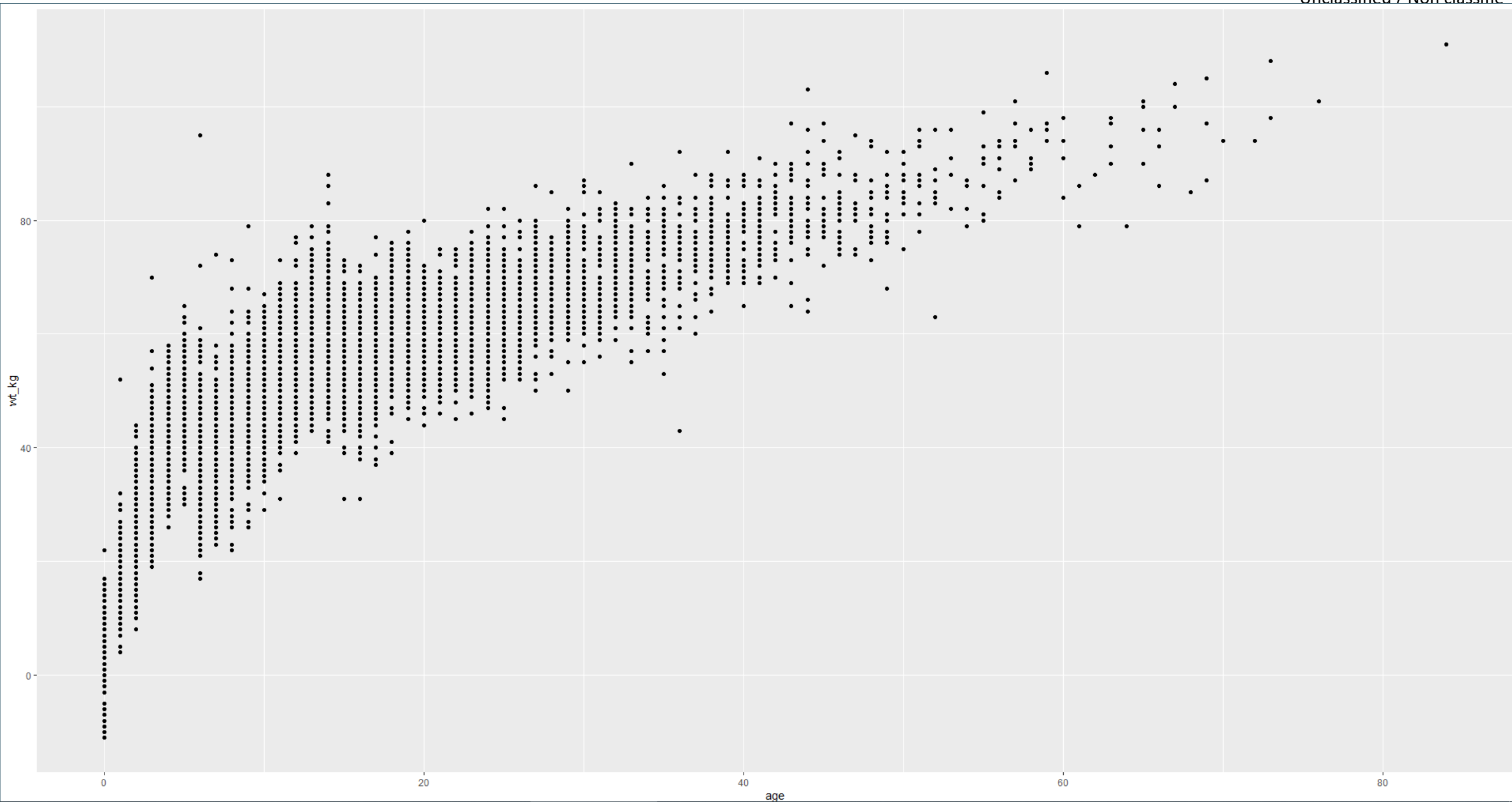
The arguments used in aes() maps the age to the x-axis and weight to the y-axis

```
ggplot(data = linelist,
    aes(x = age, y = wt_kg)) +
geom_point()
```

Creates the plot and specifies the data

The arguments used in aes() maps the age to the x-axis and weight to the y-axis

Creates the shape to draw in the plot


wt_kg
80
40
0
0
20
40
60
80
age

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg,
         color = age,
         size = age)) +

geom_point(
    shape = "diamond",
    alpha = 0.5)
```

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg,
         color = age,
         size = age)) +

geom_point(
    shape = "diamond",
    alpha = 0.5)
```

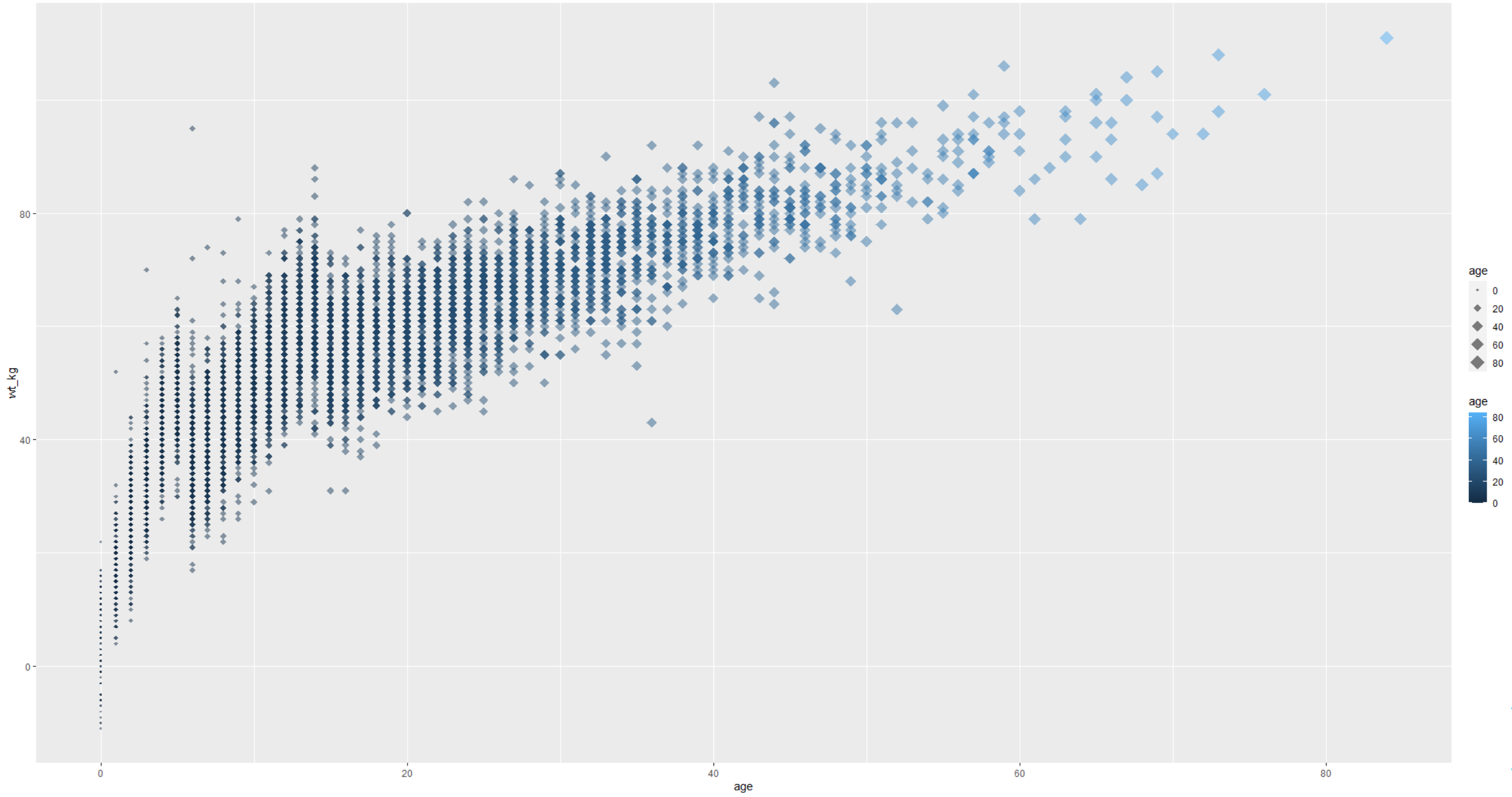
Visual display of the data is scaled by the values of age (colour and size)

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg,
         color = age,
         size = age)) +
```

Visual display of the data is scaled by the values of age (colour and size)

```
geom_point(
    shape = "diamond",
    alpha = 0.5)
```

Visual display of the point data is static in terms of shape and transparency (50%)


wt_kg
80
40
0
0
20
40
60
80
age
age
0
20
40
60
80
age
80
60
40
20
0

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg, color = age)) +

  geom_point(
    shape = "diamond",
    alpha = 0.5,
    size = 1) +

   geom_smooth(
    method = "lm",
    size = 2)
```

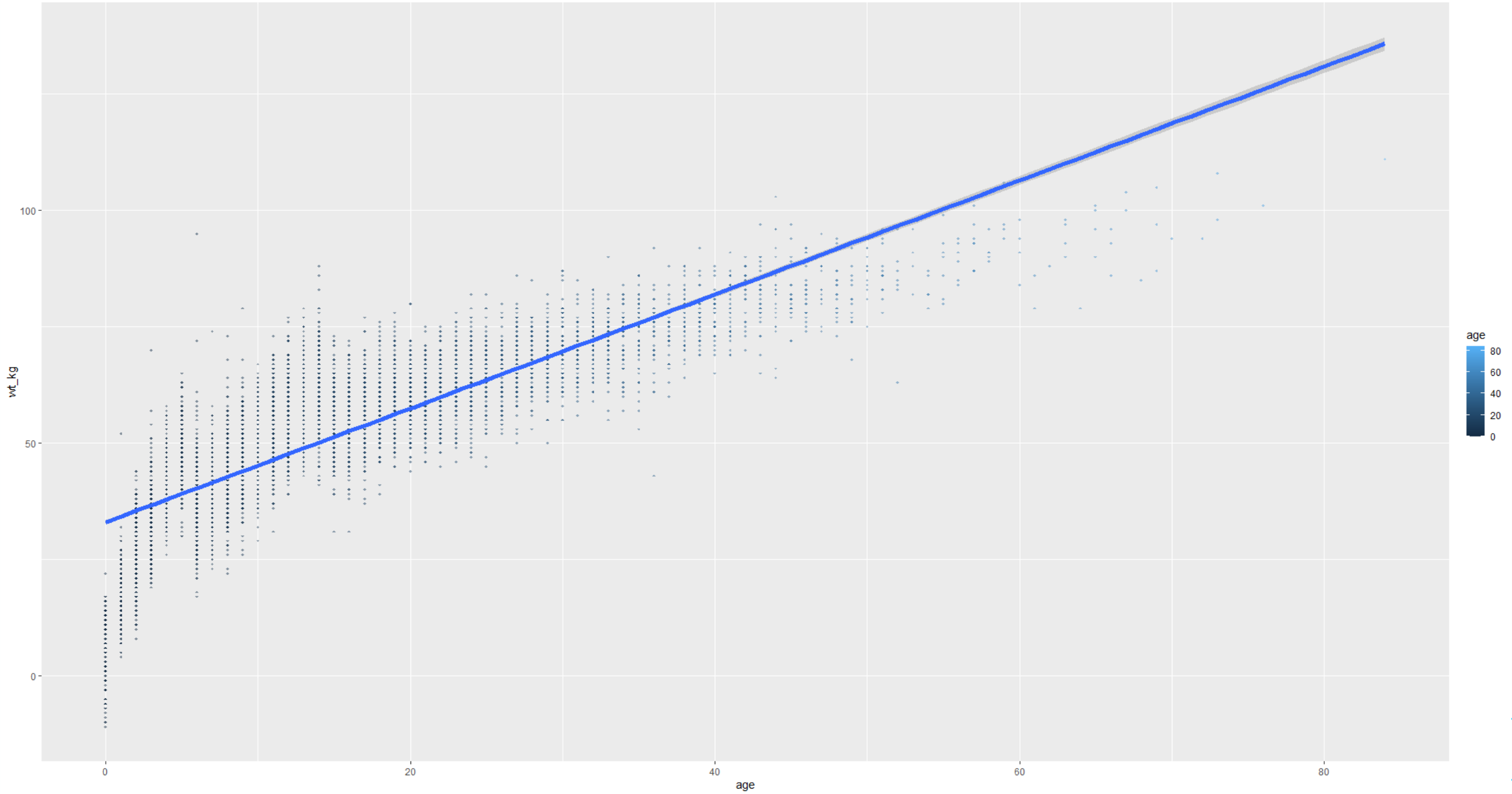
```
ggplot(data = linelist,
       aes(x = age, y = wt_kg, color = age)) +

  geom_point(
    shape = "diamond",
    alpha = 0.5,
    size = 1) +

   geom_smooth(
    method = "lm",
    size = 2)
```

Visual display of the point data is static in terms of shape, size, and transparency (50%)

```
ggplot(data = linelist,
       aes(x = age, y = wt_kg, color = age)) +

  geom_point(
    shape = "diamond",
    alpha = 0.5,
    size = 1) +

   geom_smooth(
    method = "lm",
    size = 2)
```

Visual display of the point data is static in terms of shape, size, and transparency (50%)

Adding a second item to the plot: a line of specified width representing smoothed conditional means (method: linear model)


wt_kg
100
50
0
0
20
40
60
80
age
age
80
60
40
20
0

```
ggplot(hospital_data, aes(x = date_onset)) +

geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +

theme_minimal() +

labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +

facet_wrap(~hospital)
```

```
ggplot(hospital_data, aes(x = date_onset)) +
```

← The data to be visualized

```
geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +

theme_minimal() +

labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +

facet_wrap(~hospital)
```

```
ggplot(hospital_data, aes(x = date_onset)) +

geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +

theme_minimal() +

labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +

facet_wrap(~hospital)
```

The data to be visualized

The shape to be drawn, as well as bin size, colour, and fill elements.

```
ggplot(hospital_data, aes(x = date_onset)) +
```

The data to be visualized

```
geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +
```

The shape to be drawn, as well as size and colour elements.

```
theme_minimal() +
```

Application of an existing theme to minimize background elements

```
labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +

facet_wrap(~hospital)
```

```
ggplot(hospital_data, aes(x = date_onset)) +

geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +

theme_minimal() +

labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +

facet_wrap(~hospital)
```

The data to be visualized

The shape to be drawn, as well as size and colour elements.

Application of an existing theme to minimize background elements

Adding x- and y-axis labels and a title for the figure

```
ggplot(hospital_data, aes(x = date_onset)) +
```

The data to be visualized

```
geom_histogram (binwidth = 7,
                colour = "black",
                fill = "darkred") +
```

The shape to be drawn, as well as size and colour elements.

```
theme_minimal() +
```

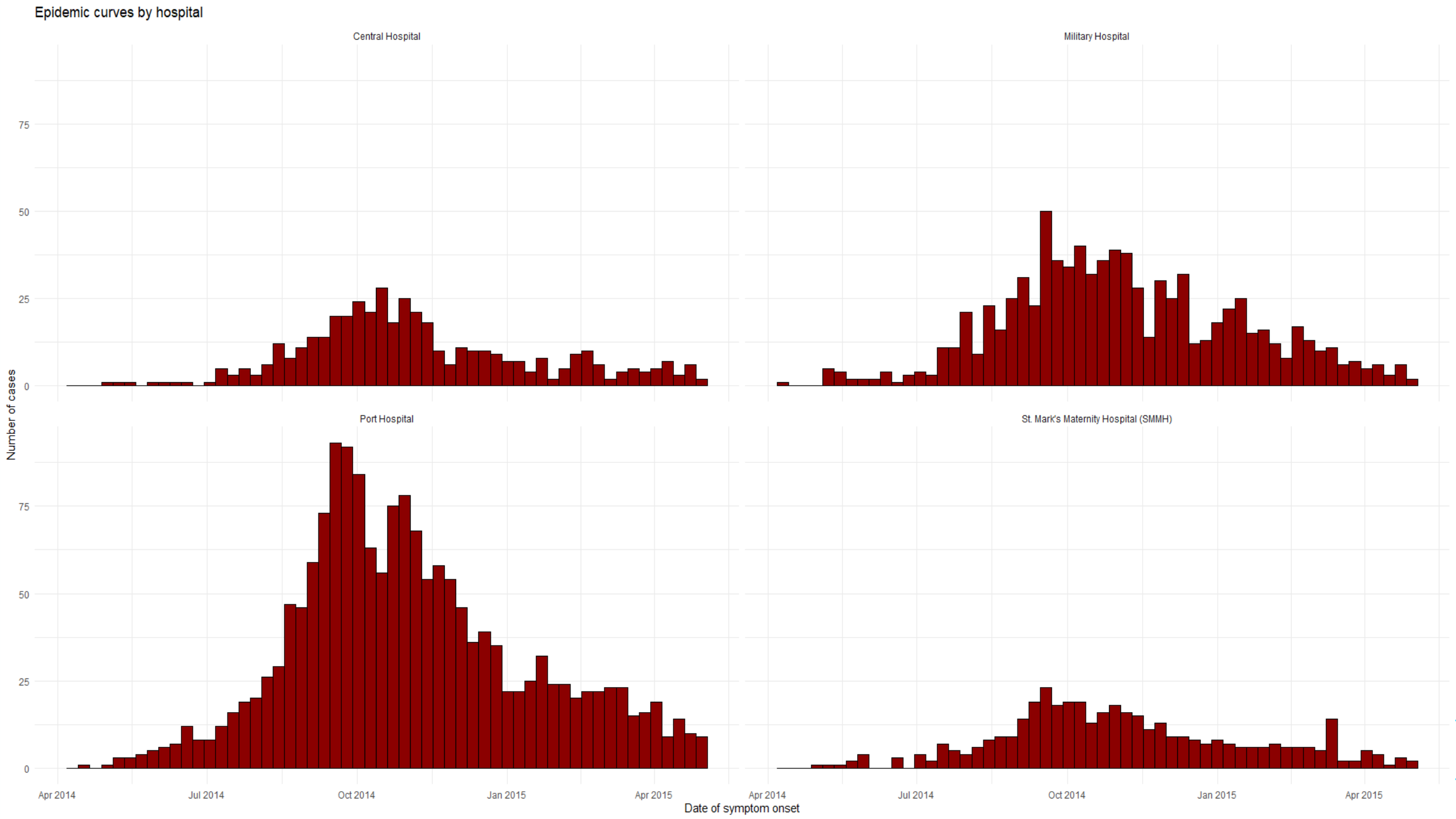
Application of an existing theme to minimize background elements

```
labs(
    x = "Date of symptom onset",
    y = "Number of cases",
    title = "Epidemic curves by hospital") +
```

Adding x- and y-axis labels and a title for the figure

```
facet_wrap(~hospital)
```

Faceting based on the **hospital** variable for a multi-panel plot


Epidemic curves by hospital
Central Hospital
Military Hospital
Port Hospital
St. Mark's Maternity Hospital (SMMH)
Number of cases
0
25
50
75
Apr 2014
Jul 2014
Oct 2014
Jan 2015
Apr 2015
Date of symptom onset

DEMO

SUMMARY

- Ggplot uses the "grammar of graphics" to
  - Organize statistical graphics into independent components
  - Make it easier to interact with specific components of a statistical graphic to create a variety of beautiful visualizations

- Graphics are built iteratively using a layered approach based on this specific grammar of independent components

- The power of ggplot to visualise data does not include the intention and thoughtfulness required to produce meaningful graphics
  - At minimum: data visualisations that make no sense or aren't useful
  - At most: data visualisations that actively cause or perpetuate harm

QUESTIONS?

WRAP-UP

SUMMARY AND WRAP-UP

- By the end of this course, participants will be able to:

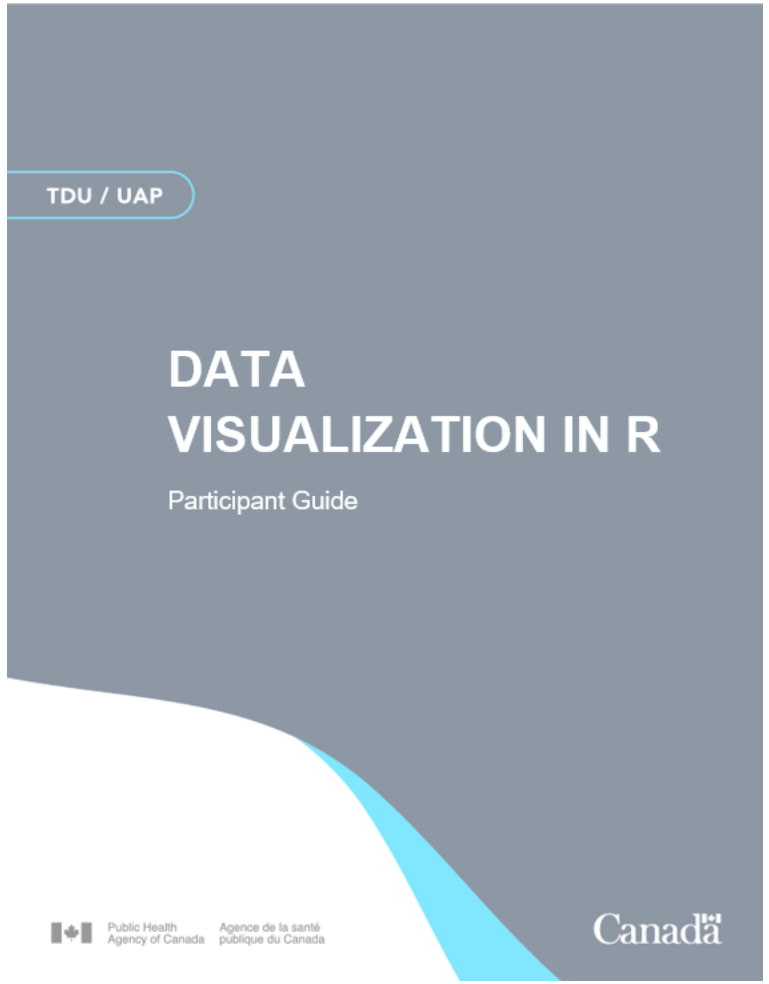
✓ Discuss differences between base R and the grammar of graphics (ggplot) coding styles for data visualization.

✓ Apply knowledge of R-coding to automate common graphics used in public health.

- Connect elements of effective graphic design to R coding practices in data visualization.

- Discuss considerations for data visualization to avoid potential harms to small or vulnerable communities.

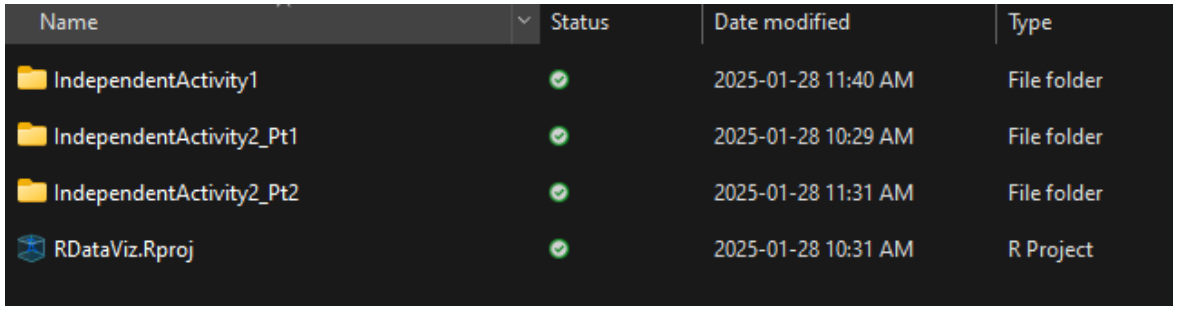
SUMMARY AND WRAP-UP

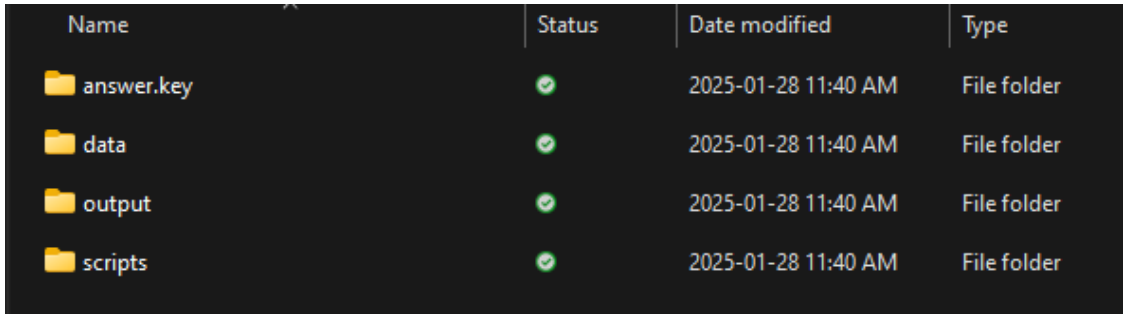


- Independent activity 1
- Create meaningful graphics using:
  - base R
  - ggplot

SETTING UP YOUR FILES FROM GITHUB

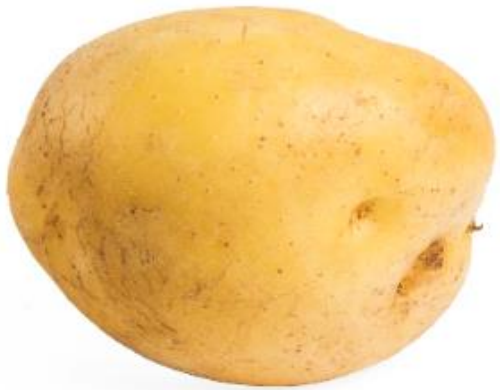
- Create a new file for R Data Viz training on your computer
- Create an Rproject file in this folder
- Move the **IndependentActivity1** to this new folder
  - Create new scripts to populate under /scripts/
  - OR: use the scripts in /answer.key/ to walk through the exercises
  - Note, you may need to update the location of data files if you're using the answer.key scripts to reflect how you set up your folder structure.





Would you use the words "challenging" and "fun" to describe today's session?

Mark where your experience falls on a scale of potato
(not challenging and not fun) to watermelon (challenging and fun)



Not challenging and not fun

Challenging and fun!

DAY 1 QUESTIONS

Please use the course Sli.do to leave us any questions that you have about the course content from Day 1.

We'll monitor your input and address any issues when we come back for Day 2

www.slido.com : #RDV2025