R You Ready? Visualizing Data in R

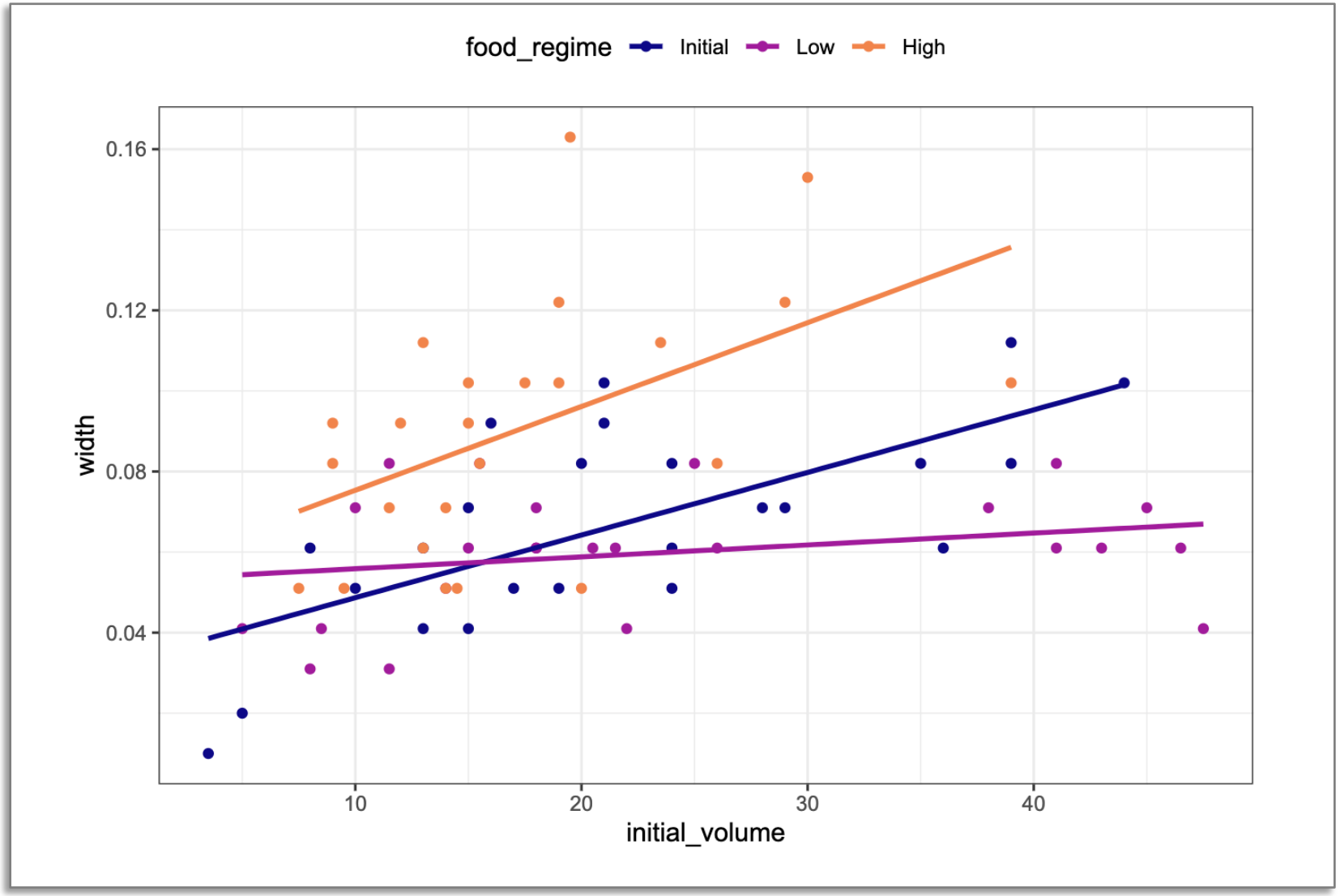
Workshop by Ketura Elie

Data Visualization

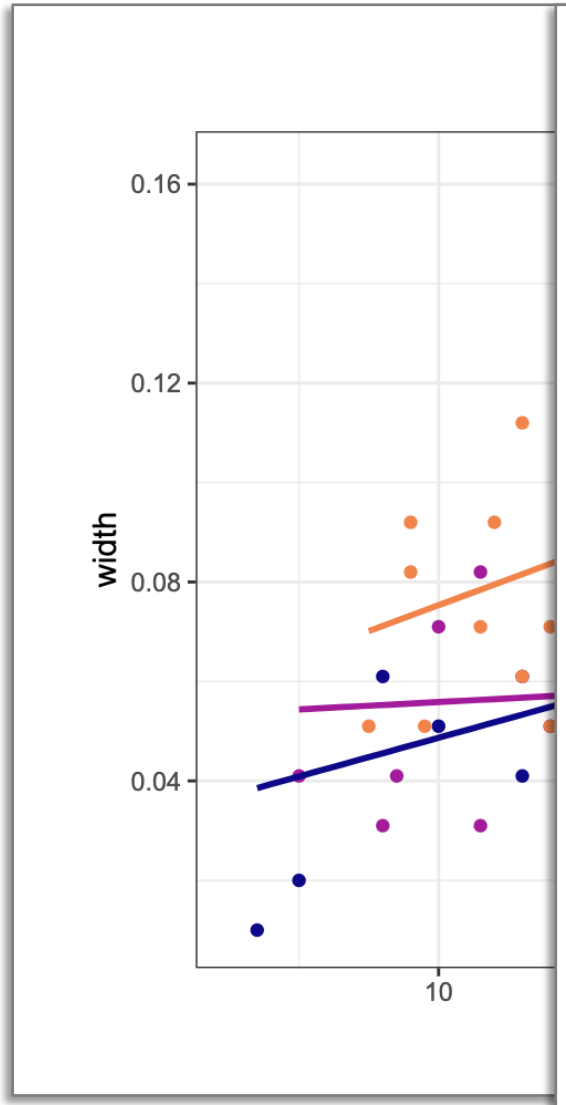
Simply put – it is art of turning numbers into useful knowledge

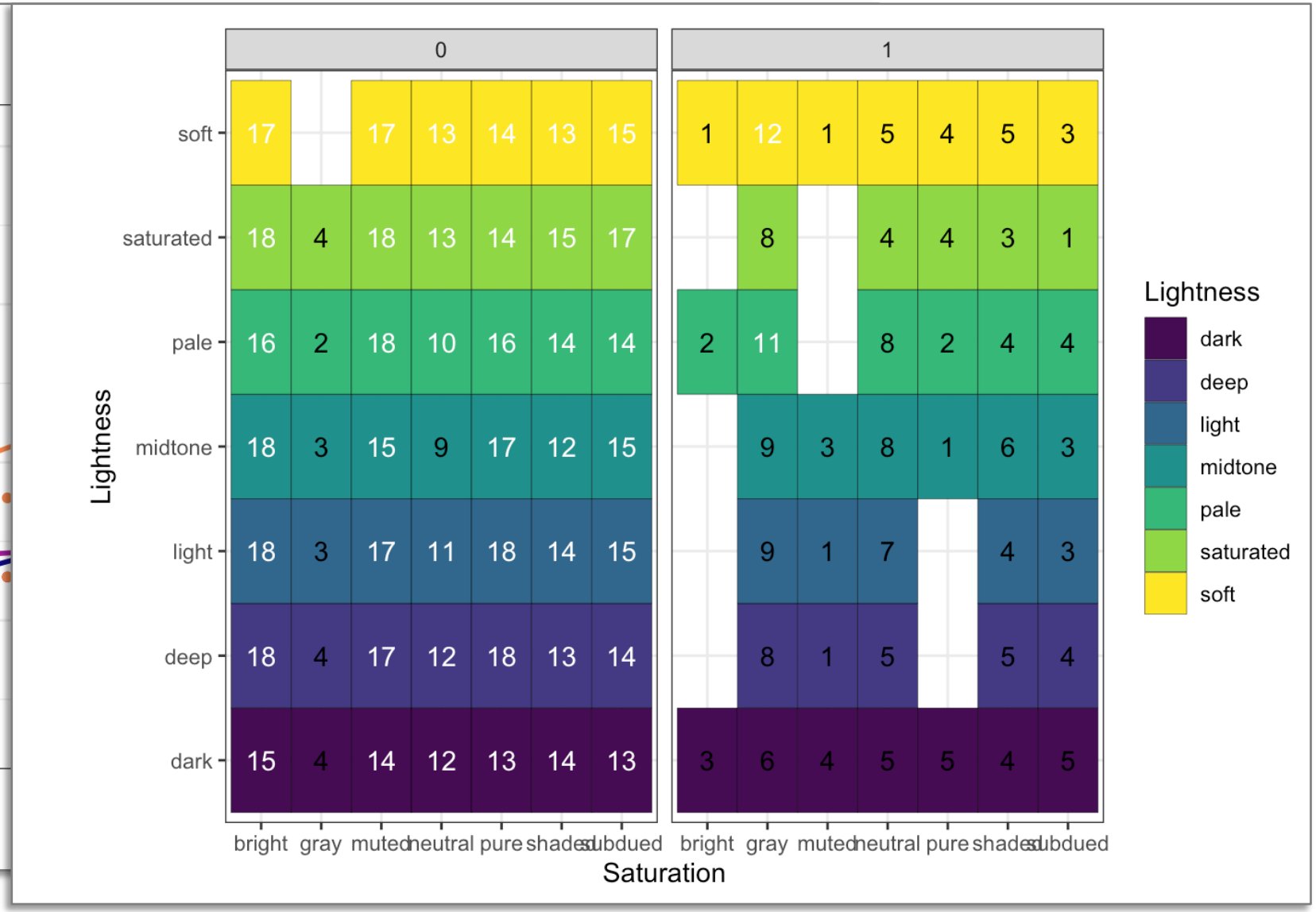
Goal - efficient communication of complex quantitative ideas

Data Visualization

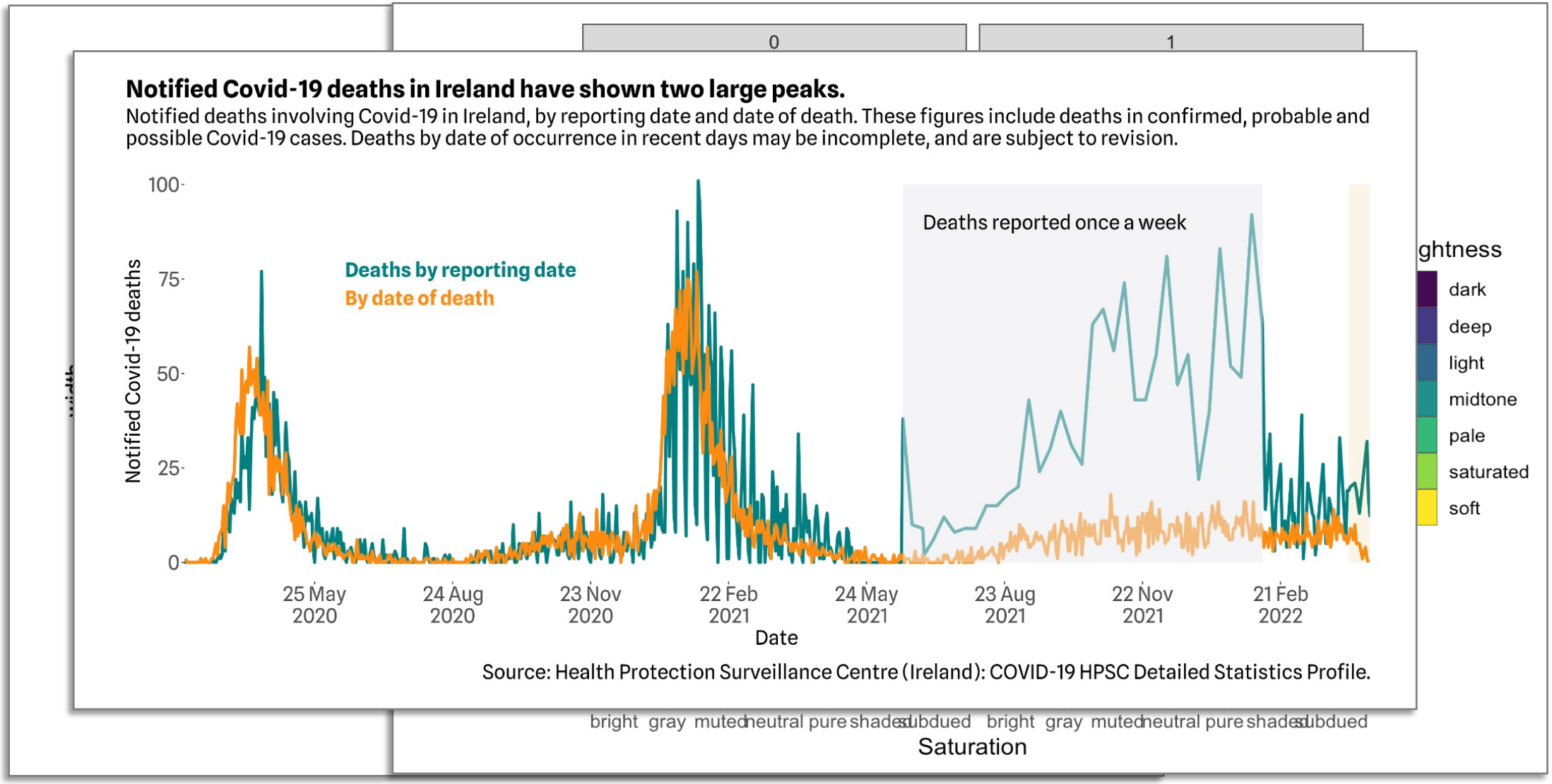


Data Visualization





Data Visualization



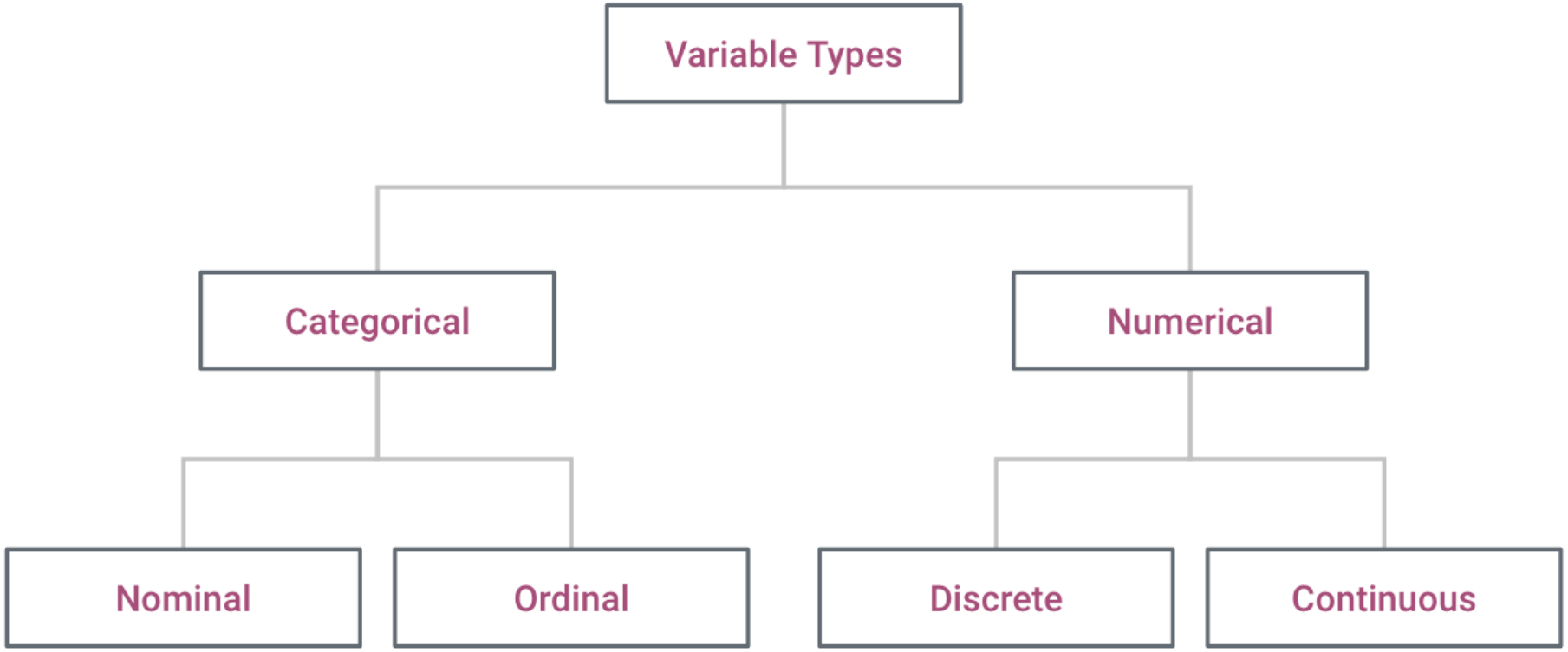
https://rpubs.com/anthonybmasters/notified-covid-deaths-ireland

But before we can do any of that, we have to understand the fundamentals of ggplot

Which is the goal of this workshop!

Data Variable Types

Variable types inform how best to visualize your data



Grammar of Graphics

GGplot – R package for producing data visualizations

Elements of ggplot

Data

Aesthetics

Geometries

Facets

Statistics

Coordinates

Themes

Grammar of Graphics

GGplot – R package for producing data visualizations

Elements of ggplot

- **Data**
- **Aesthetics**
- **Geometries**
- **Facets**
- **Statistics**
- **Coordinates**
- **Themes**

Grammar of Graphics

GGplot – R package for producing data visualizations

Elements of ggplot

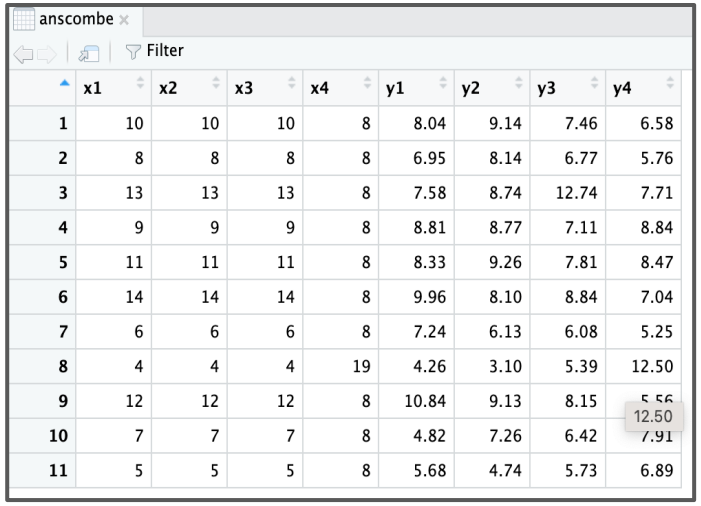
- **Data**
- **Aesthetics**
- **Geometries**
- **Facets**
- **Statistics**
- **Coordinates**
- **Themes**

Data, Aesthetics, Geometries

Basis for all plots

Data

the data / data set itself



anscombe

Filter

	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

12.50

Data, Aesthetics, Geometries

Basis for all plots

Data

the data / data set itself

anscombe

Filter

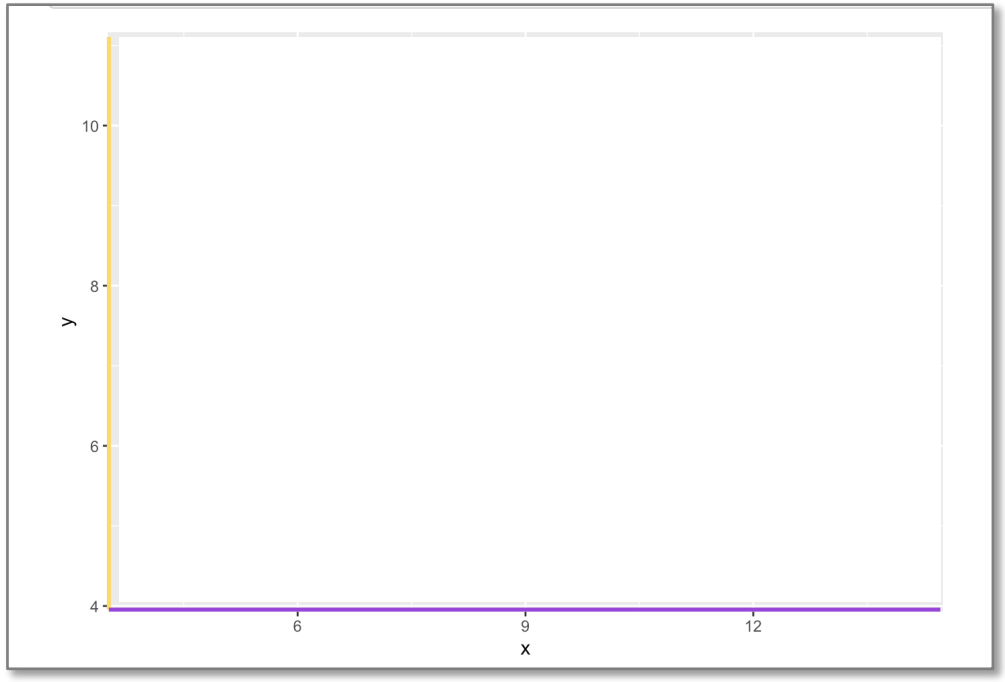
	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

12.50

Aesthetics

“variables” / data mapped onto the aesthetic elements

x1	y1
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68



Data, Aesthetics, Geometries

Basis for all plots

Data

the data / data set itself

anscombe

Filter

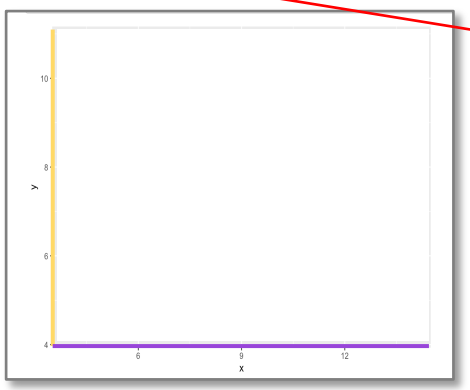
	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

12.50

Aesthetics

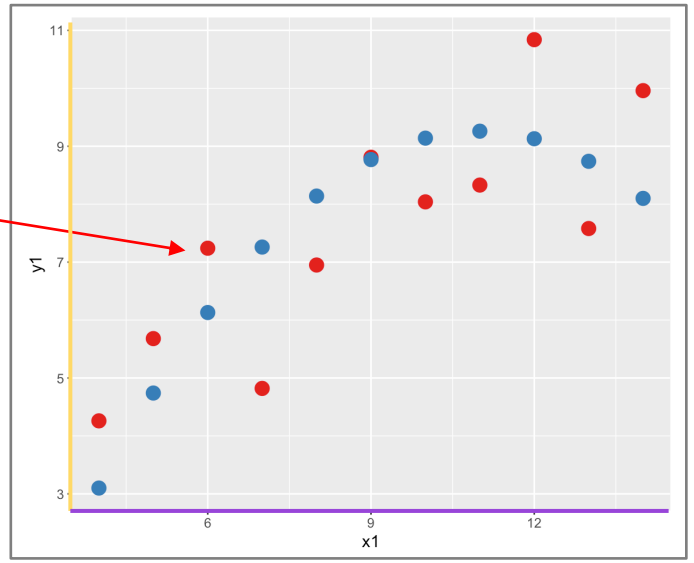
“variables” / data mapped onto the aesthetic elements

x1	y1
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68



Geometries

determines how data is displayed



Data, Aesthetics, Geometries

the (your) data / data set

```
```{r, load_packages}
library(tidyverse)
```

Anscombe

```{r,a_1}
anscombe %>% glimpse()
```
```

```
Rows: 11
Columns: 8
$ x1 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
$ x2 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
$ x3 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
$ x4 <dbl> 8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8
$ y1 <dbl> 8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68
$ y2 <dbl> 9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74
$ y3 <dbl> 7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73
$ y4 <dbl> 6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89
```

Variable Name **Data Type** **Values**

How do I go about visualizing this data ?

Data, Aesthetics, Geometries

Basis for all plots

Data

the data / data set itself

anscombe

Filter

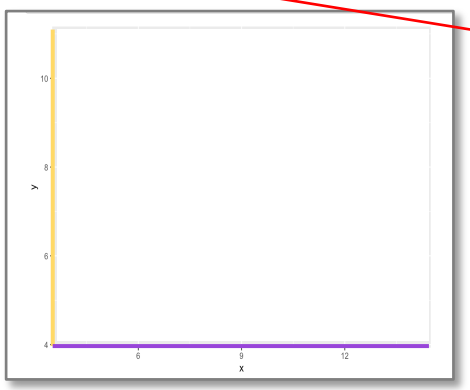
	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

12.50

Aesthetics

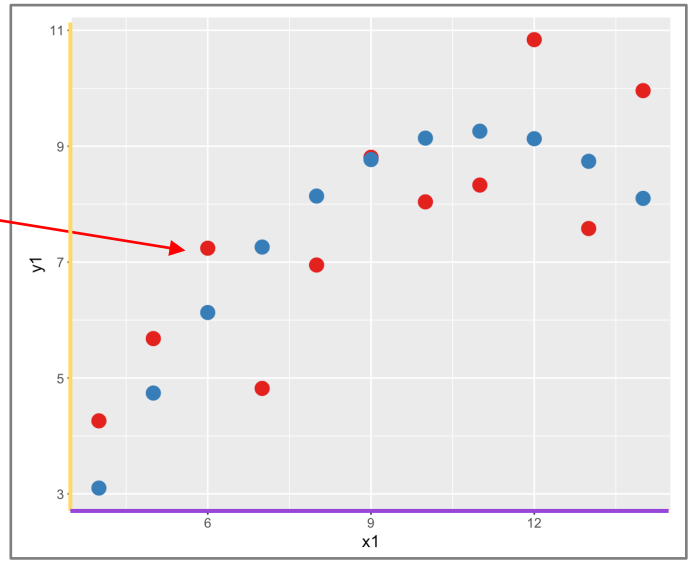
“variables” / data mapped onto the aesthetic elements

x1	y1
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68



Geometries

determines how data is displayed



Data, Aesthetics, Geo

the (your) data / data set

”Tidy” the data

transform the data from “wide” to “long”

anscombe

Filter

	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

12.50

	rowid	dataset	x	y
1	1	1	10	8.04
2	2	1	8	6.95
3	3	1	13	7.58
4	4	1	9	8.81
5	5	1	11	8.33
6	6	1	14	9.96
7	7	1	6	7.24
8	8	1	4	4.26
9	9	1	12	10.84
10	10	1	7	4.82
11	11	1	5	5.68
12	1	2	10	9.14
13	2	2	8	8.14
14	3	2	13	8.74
15	4	2	9	8.77
16	5	2	11	9.26
17	6	2	14	8.10
18	7	2	6	6.13
19	8	2	4	3.10
20	9	2	12	9.13
21	10	2	7	7.26
22	11	2	5	4.74
23	1	3	10	7.46
24	2	3	8	6.77
25	3	3	13	12.74
26	4	3	9	7.11

Data, Aesthetics, Geo

the (your) data / data set

”Tidy” the data

transform the data from “wide” to “long”

anscombe

Filter

	x1	x2	x
1	10	10	
2	8	8	
3	13	13	
4	9	9	
5	11	11	
6	14	14	
7	6	6	
8	4	4	
9	12	12	
10	7	7	
11	5	5	

```{r, tt_1}
a_tidy <- anscombe %>%
    tibble::rowid_to_column() %>%
    pivot_longer(cols = !c("rowid")) %>%
    tidyr::separate(name,
                    c("variable","dataset"),
                    sep = 1) %>%
    pivot_wider(names_from='variable',values_from = 'value') %>%
    arrange(dataset)
```

	rowid	dataset	x	y
1	1	1	10	8.04
2	2	1	8	6.95
3	3	1	13	7.58
4	4	1	9	8.81
5	5	1	11	8.33
6	6	1	14	9.96
23	1	3	10	7.46
24	2	3	8	6.77
25	3	3	13	12.74
26	4	3	9	7.11

Data, **Aesthetics,** Geometries

the variables of your data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%


```
```


1. Always want to start out with your data

Data, **Aesthetics,** Geometries

the variables of your data

”Grammar” of ggplot code

“Parent Call”

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
```
```

x = variable to display on x-axis

y = variable to display on y-axis

2. Within the “ggplot” call is where you indicate your ”**aesthetics**” or map your variables onto the aesthetics elements

Data, **Aesthetics,** Geometries

the variables of your data

"Grammar" of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +

```
```


***Aesthetics:** Anything within the **aes( )** will **always** refer to the **values** in your dataset

Data, Aesthetics, Geometries

Basis for all plots

Data

the data / data set itself

anscombe

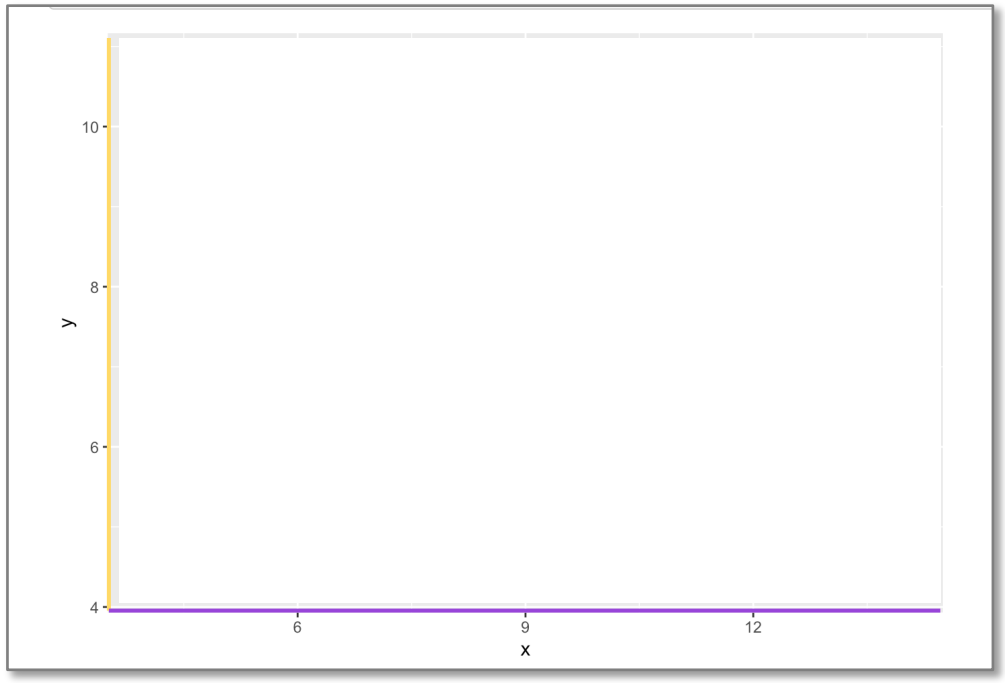
Filter

	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04
7	6	6	6	8	7.24	6.13	6.08	5.25
8	4	4	4	19	4.26	3.10	5.39	12.50
9	12	12	12	8	10.84	9.13	8.15	5.56
10	7	7	7	8	4.82	7.26	6.42	7.91
11	5	5	5	8	5.68	4.74	5.73	6.89

Aesthetics

values from your data mapped onto the aesthetic elements

x1	y1
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68



Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

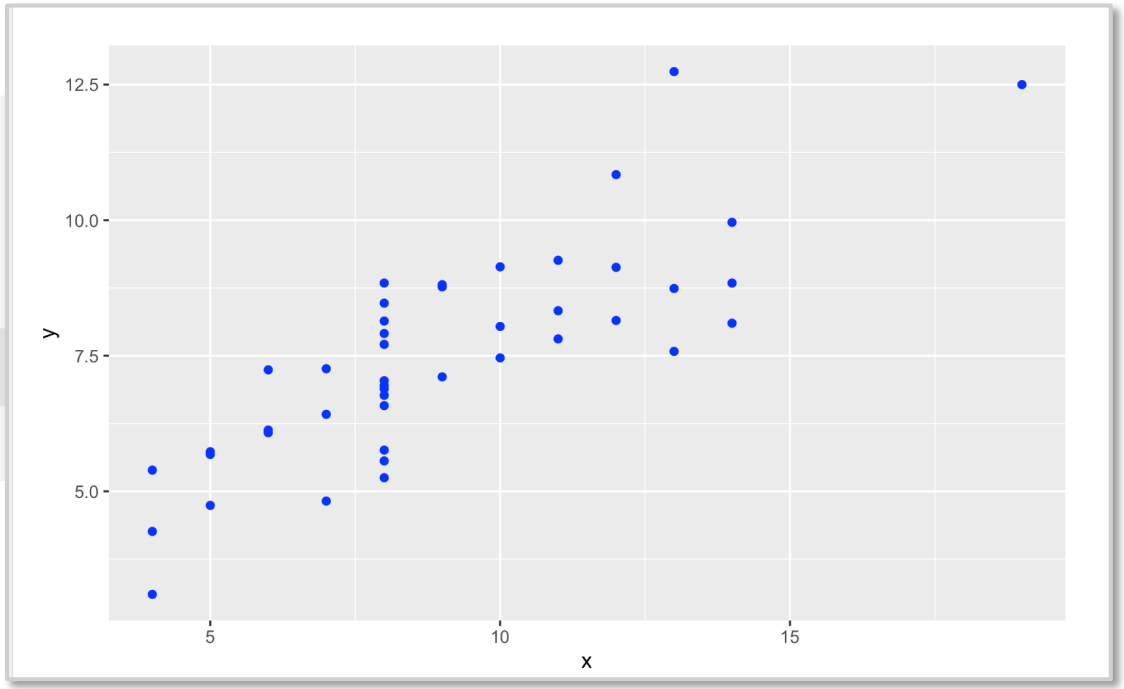
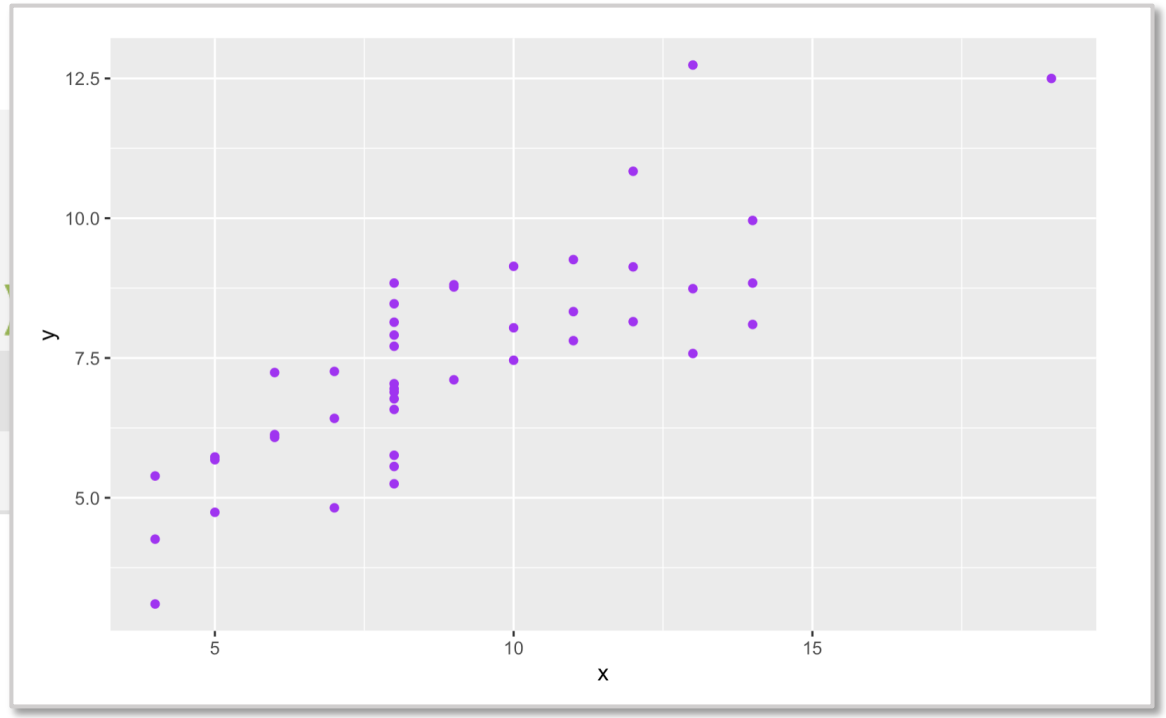
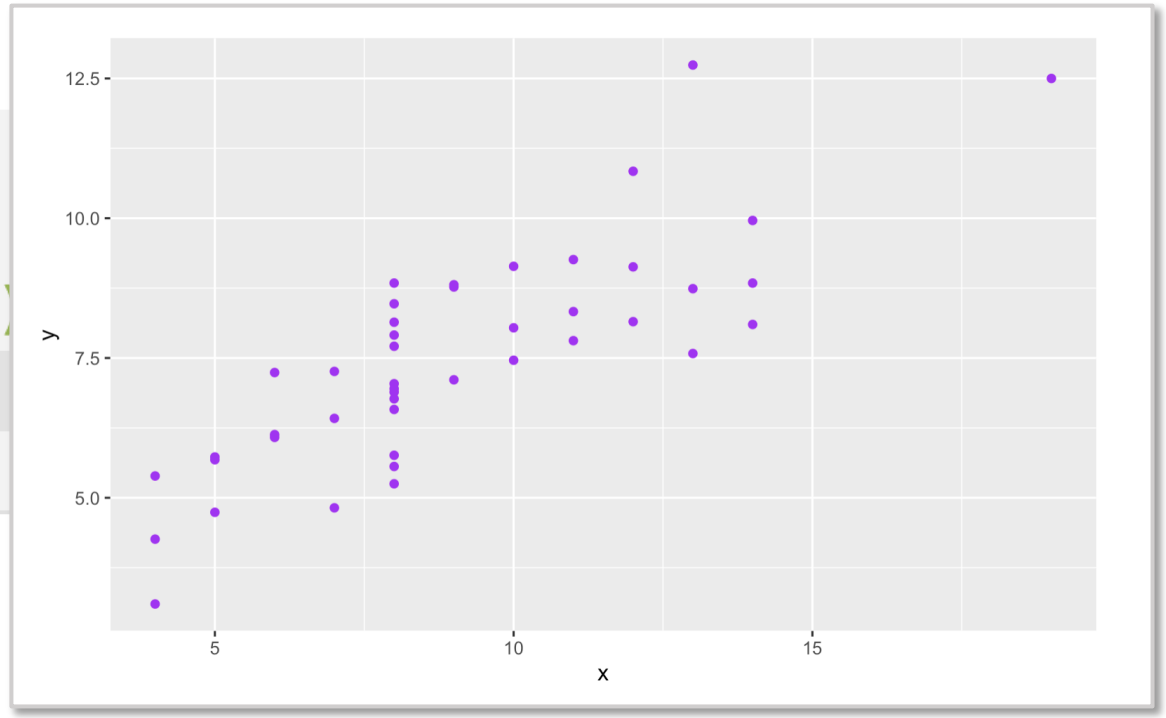
```r
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point()
```


3. Geometrics or “geom_” control how data is display
Here geom**_point** means that our data will be displayed in dots or “points”

Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point(color = "blue")
```
```

Within the **“geom_”** call, you can customize your points. Here I am “hard-coding” the point in my plot to be “blue”

Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y)
    geom_point(color = "blue")
```
```



Within the **“geom_”** call, you can customize your points. Here I am “hard-coding” the point in my plot to be “blue”

Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point(color = "purple")
```
```

Here I changed it to “purple”

Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y)
    geom_point(color = "purple")
```
```



Here I changed it to “purple”

Data, Aesthetics, **Geometries**

Display of the data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y)
    geom_point(color = "purple")
```
```



Here I changed it to “purple”

You can **also** include **aes() within the geom_()’s!!**

Data, **Aesthetics, Geometries**

displaying the variables of your data

”Grammar” of ggplot code

- When **aes()** is used within a **geom_()’,** it can control aesthetics attributes such as color, shape, size, fill, etc.
- How **aes()** and **geom_()’** work together depends on your **datatype**

color – points, lines

size - points

alpha - transparency

shape - points

fill – bars, violins, cloud

Continuous

Data, **Aesthetics, Geometries**

displaying the variables of your data

”Grammar” of ggplot code

- When **aes()** is used within a **geom_()’,** it can control aesthetics attributes such as color, shape, size, fill, etc.
- How **aes()** and **geom_()’** work together depends on your **datatype**

color – points, lines

size - points

alpha - transparency

shape - points

fill – bars, violins, cloud

Continuous

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

- When **aes()** is used within a **geom_()’,** it can control aesthetics attributes such as color, shape, size, fill, etc.
- How **aes()** and **geom_()’** work together depends on your **datatype**

color – points, lines

size - points

alpha - transparency

shape - points

fill – bars, violins, cloud

Continuous

***Discrete**

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

- When **aes()** is used within a **geom_()’,** it can control aesthetics attributes such as color, shape, size, fill, etc.
- How **aes()** and **geom_()’** work together depends on your **datatype**

color – points, lines

size - points

alpha - transparency

shape - points

fill – bars, violins, cloud

Continuous

***Discrete**

Categorical

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point(mapping = aes())
```
```

Given the data is continuous, what attribute would be best to visualize the data with?

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

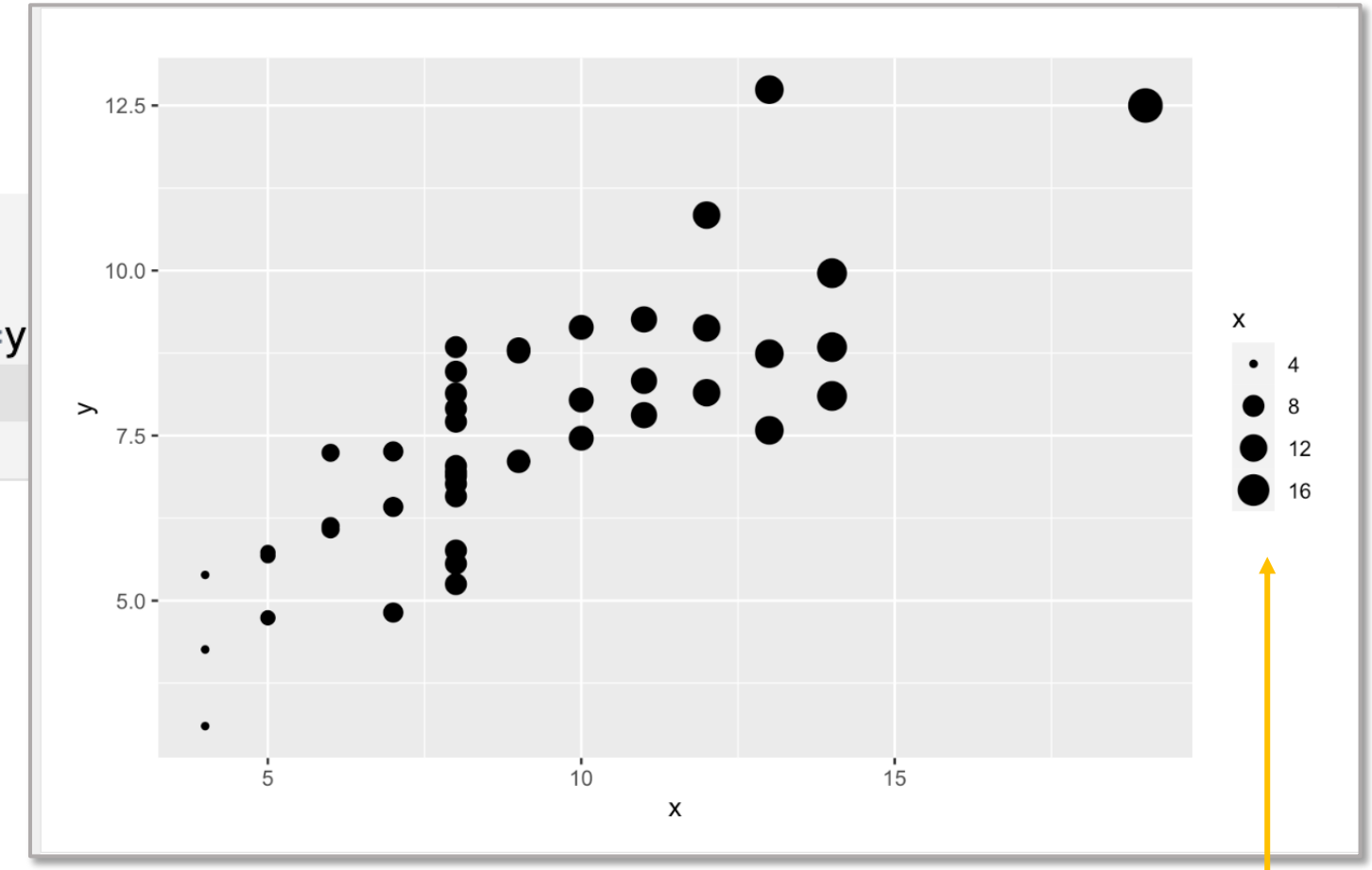
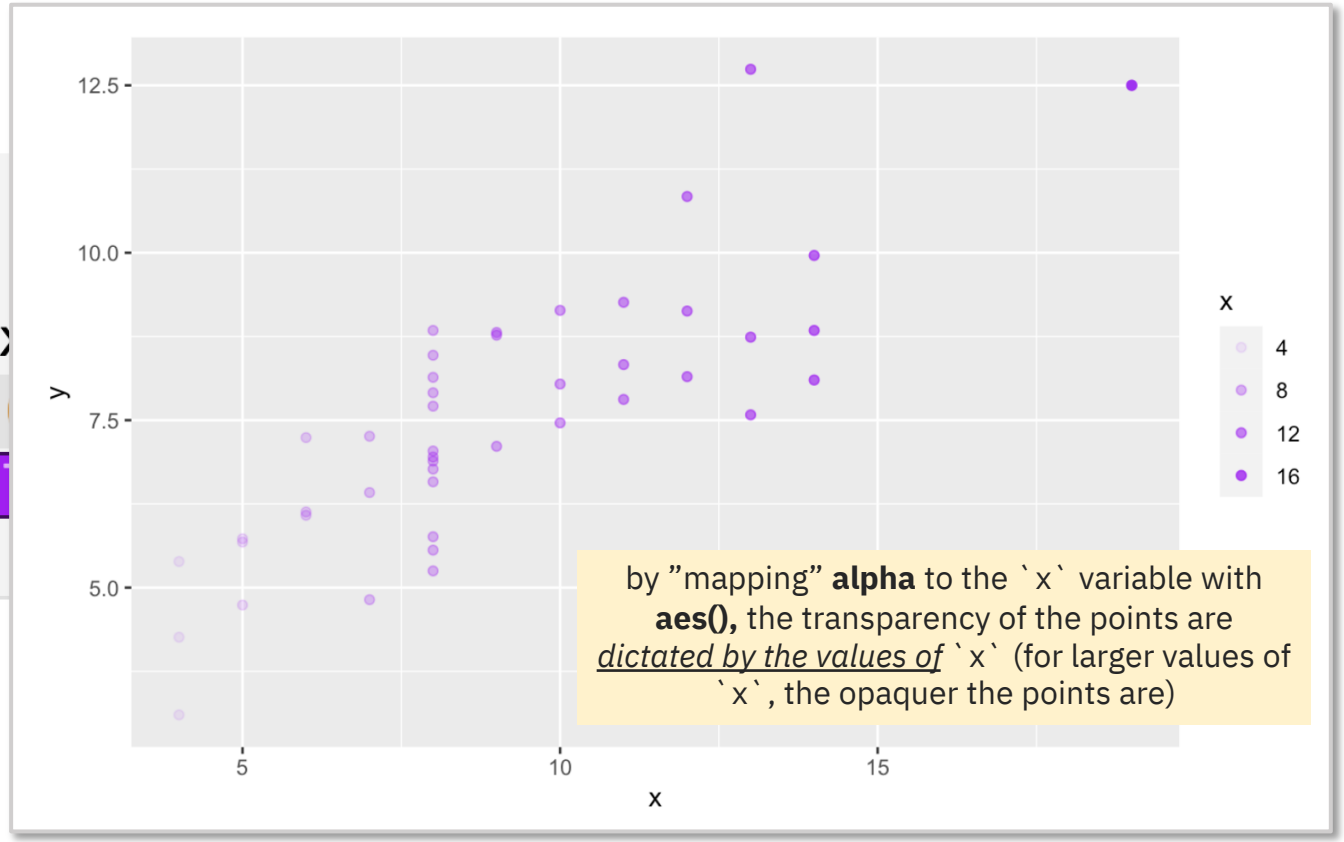
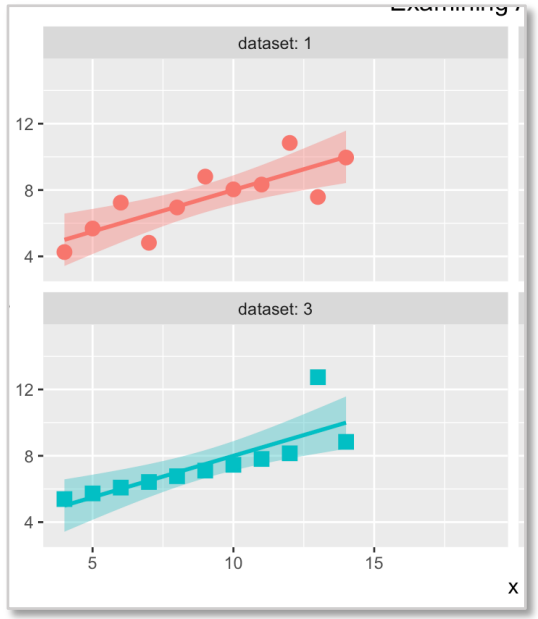
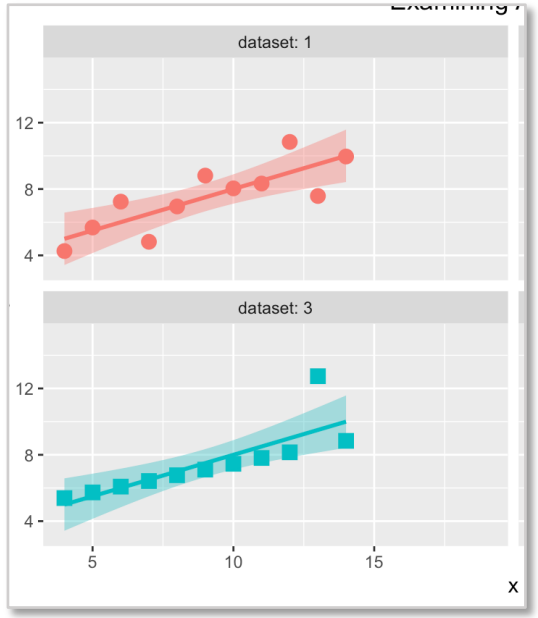
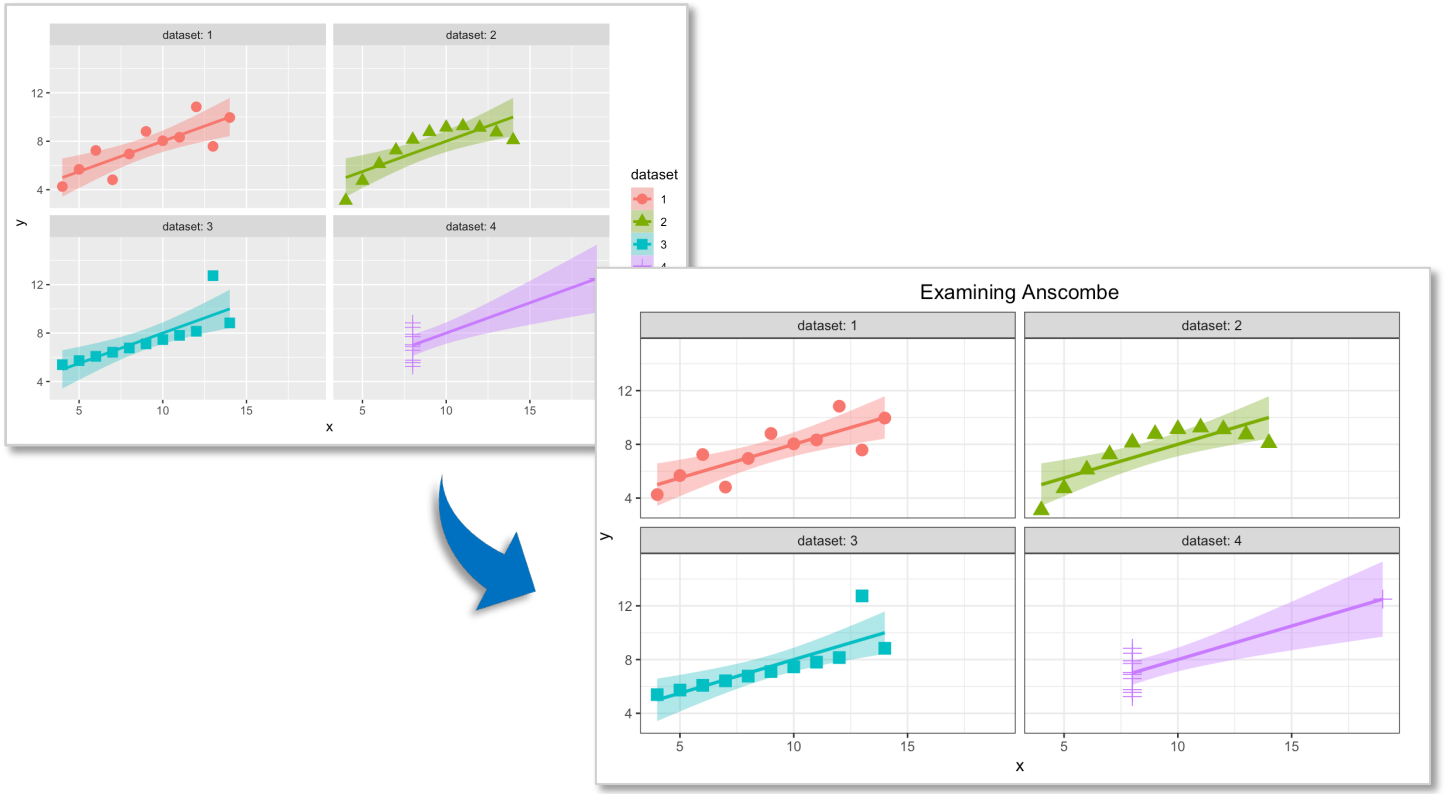
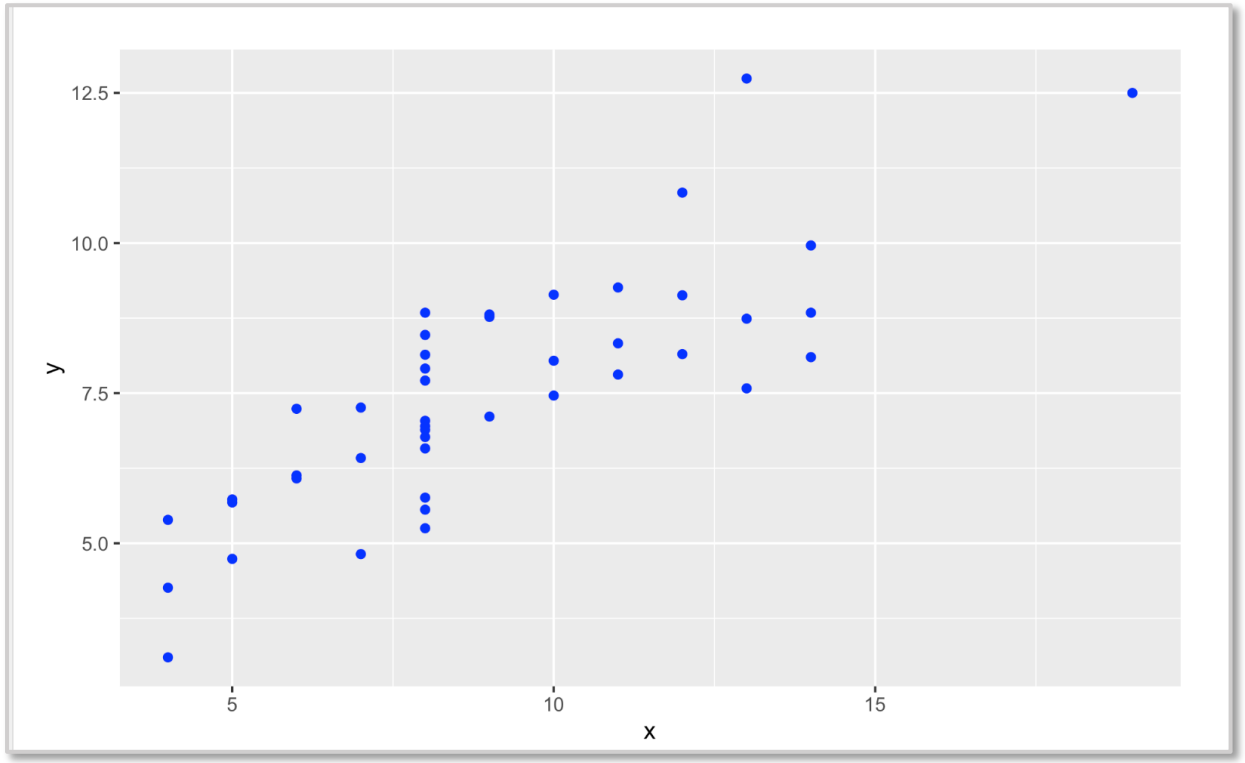
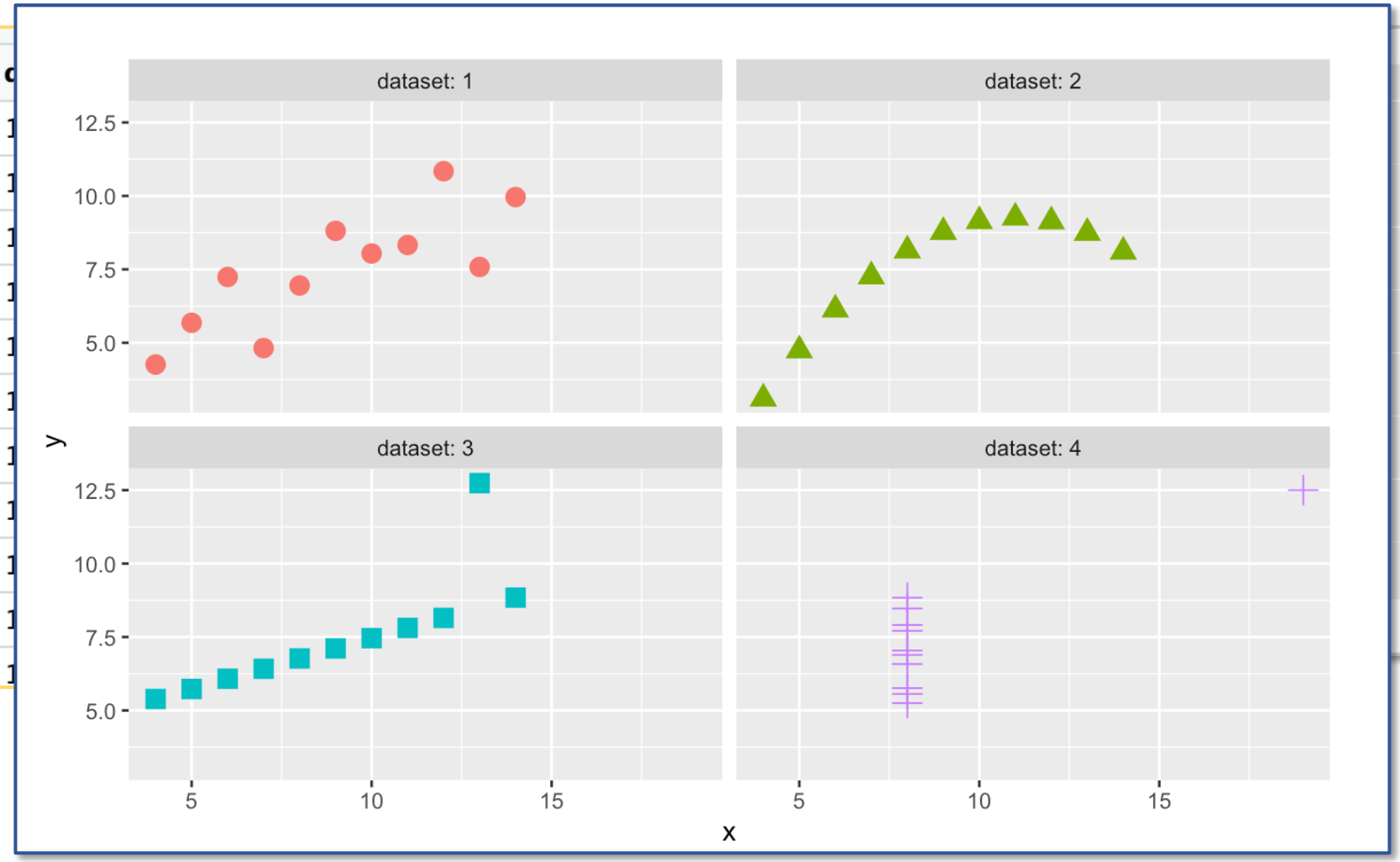
```r
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point(mapping = aes(size = x))
```


We could do size!

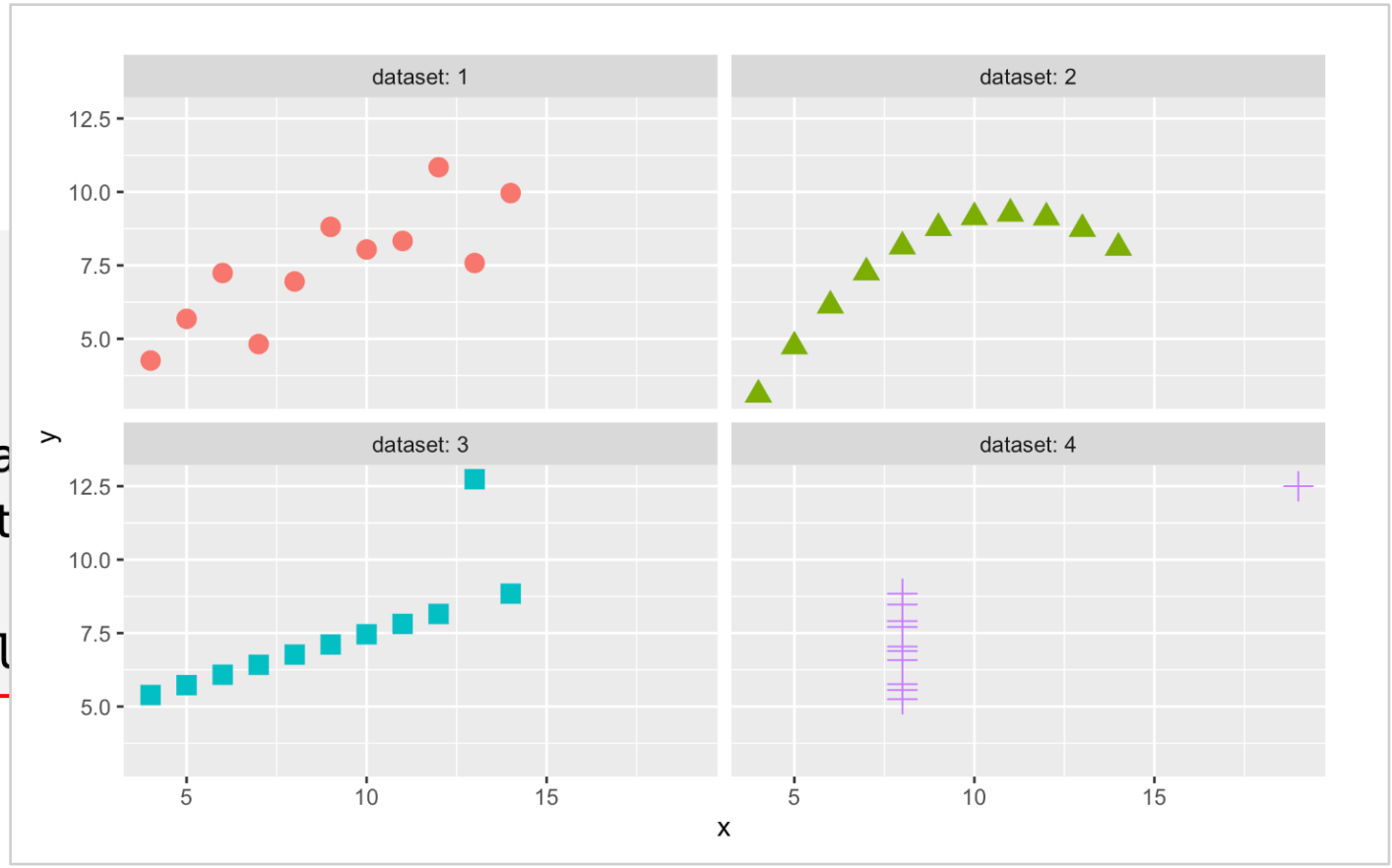
Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y
    geom_point(aes(size = x))
```
```

We could do size!



To highlight that **aes() pulls** from your data, notice the legend!

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x,y=y))) +
    geom_point(mapping = aes(alpha = x),
               color = "purple")
```
```

you can also include “hard” coded attributes along with **aes()**

*note here I changed the aes() attribute to alpha (the transparency of the points) ,where **values** of the x will control the transparency of the points!*

Data, **Aesthetics, Geometries**

the variables of your data

”Grammar” of ggplot code

```
```{r}
a_tidy %>%
    ggplot(mapping = (aes(x=x
    geom_point(mapping = aes
                  color = "purp
```
```

you can also include “hard” coded attributes along with **aes()**



Together, **aesthetics and geoms** are often used to display additional in variables in your data set

Grammar of Graphics

GGplot – R package for producing data visualizations

Elements of ggplot

- **Data**
- **Aesthetics**
- **Geometries**
- ★ **Facets**
- **Statistics**
- **Coordinates**
- ★ **Themes**

Facets, Themes

Facets

faceting splits data into subsets; displays same graph for each subset



Facets, Themes

Facets

faceting splits data into subsets; displays same graph for each subset



Themes

Control “look” of your graphic (background, legend, axis size, header)



Facets, Themes

faceting splits data into subsets

	rowid	dataset	x	y
1	1	1	10	8.04
2	2	1	8	6.95
3	3	1	13	7.58
4	4	1	9	8.81
5	5	1	11	8.33
6	6	1	14	9.96
7	7	1	6	7.24
8	8	1	4	4.26
9	9	1	12	10.84
10	10	1	7	4.82
11	11	1	5	5.68

Facets, Themes

faceting splits data into subsets

	rowid	dataset	x	y
1	1	1	10	8.04
2	2	1	8	6.95
3	3	1	13	7.58
4	4	1	9	8.81
5	5	1	11	8.33
6	6	1	14	9.96
7	7	1	6	7.24
8	8	1	4	4.26
9	9	1	12	10.84
10	10	1	7	4.82
11	11	1	5	5.68



Note that my data set has “dataset” column. How could I take this variable into account in my graphic? **Facet!**

Facets, Themes

faceting splits data into subsets

	rowid	
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1
11	11	1



Facets, Themes

faceting splits data into subsets

```
```{r, tt_4}
a_tidy %>%
    ggplot(aes(x=x,y=y)) +
    geom_point(aes(color =dataset,
                   shape = dataset),
               size = 3.5) +
    facet_wrap(~dataset, labeller = 'label_both') +
```


Within the **facet_wrap()** call, you indicate the variable you want to subset by with a "~" followed by the **variable** name
the "labeller" function labels your subset graph by both the *variable column name* and the *variable value*
```

Facets, Themes

faceting splits data into subsets

```
```{r, tt_4}
a_tidy %>%
    ggplot(aes(x=x,y=y)) +
    geom_point(aes(color =data
                   shape = dat
               size = 3.5) +
    facet_wrap(~dataset, label
```

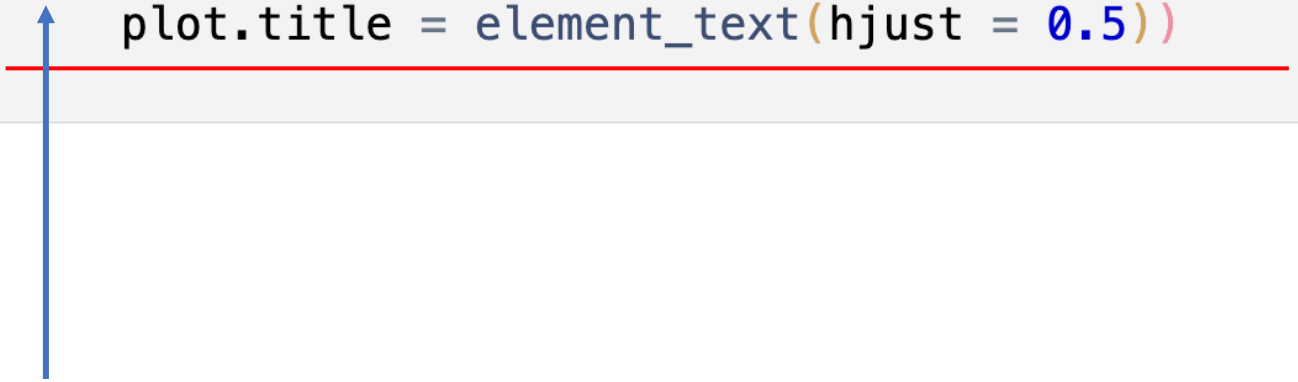


Within the **facet_wrap()** call, you indicate the variable you want to subset by with a "~" followed by the **variable** name
the "labeller" function labels your subset graph by both the *variable column name* and the *variable value*
```

Facets, Themes

the “look” of your graphic

```
    labs(title = "Examining Anscombe") +
    theme_bw() +
    theme(legend.position='none',
          plot.title = element_text(hjust = 0.5))
```



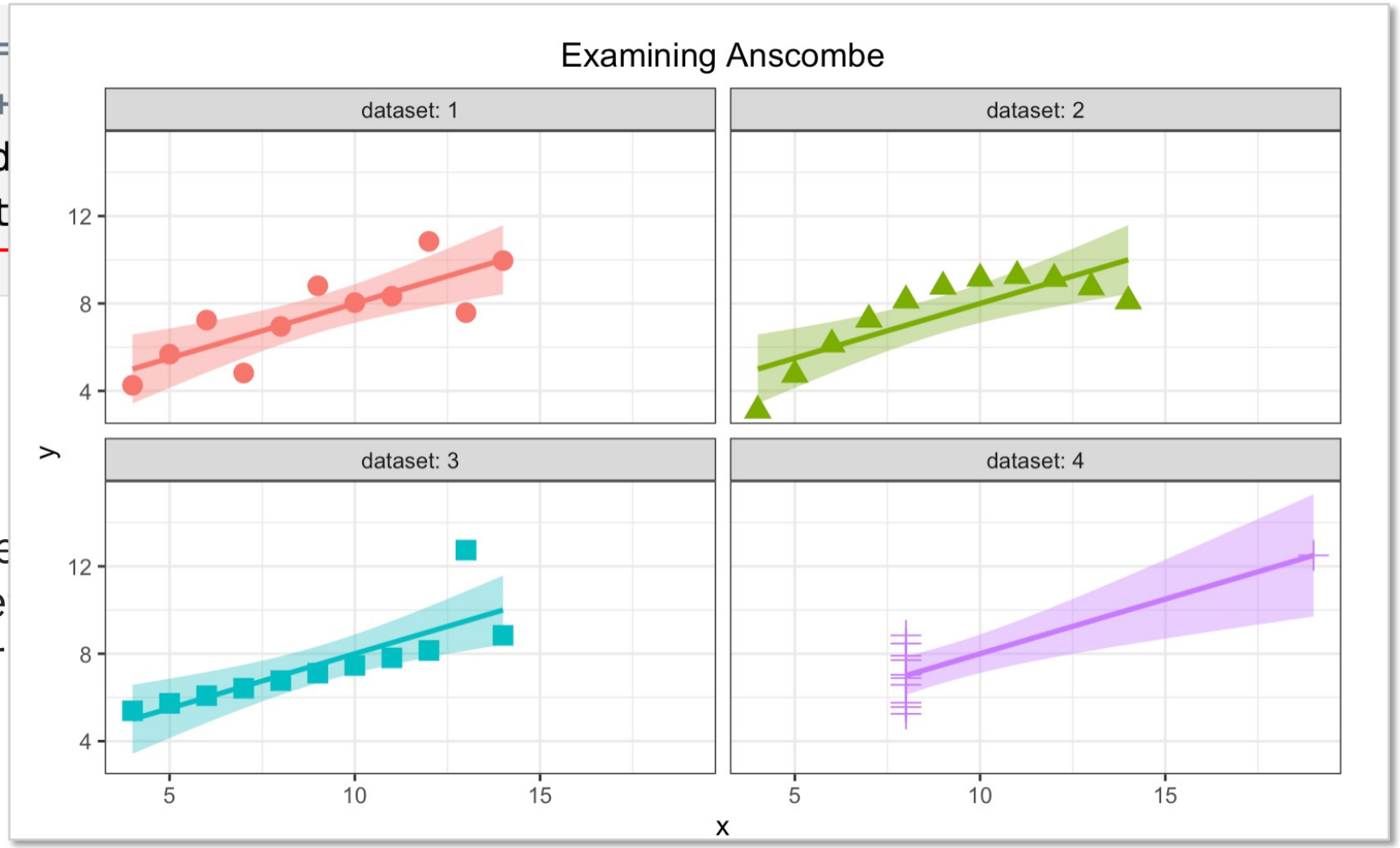
theme_ are for present customization that comes with R. I like **theme_bw()**.
To control finer details of your graphic, use **theme()** to control different aspects / elements of your graphic like the legend position or title of the plot

Facets, Themes

the “look” of your graphic

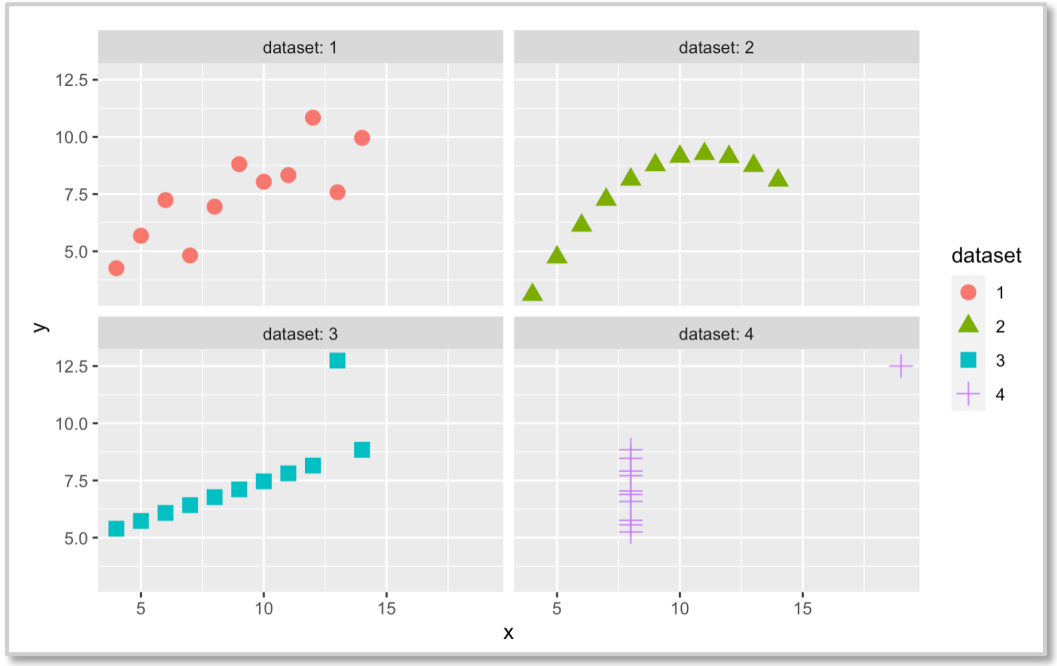
```
    labs(title =
    theme_bw() +
    theme(legend
        plot.t
```

theme_ are for pre
To control finer de
/ elements of your



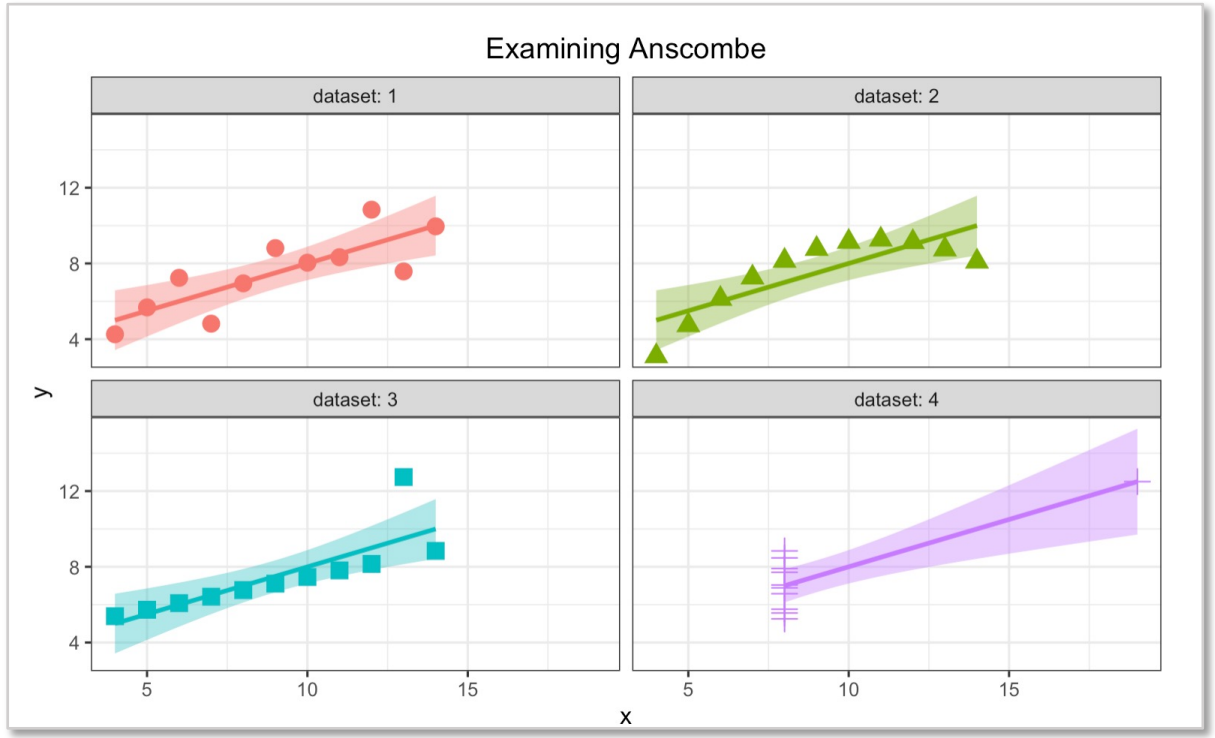
Facets, Themes

the “look” of your graphic



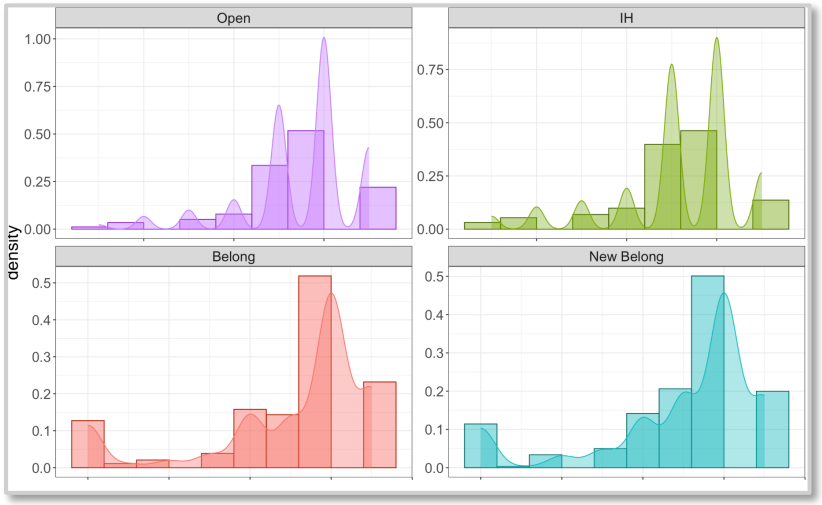
Used **theme()** to center my plot title and remove the legend (the legend was redundant information)

Use **theme_bw()** to change from gray background to white and added the black borders around my facet headers

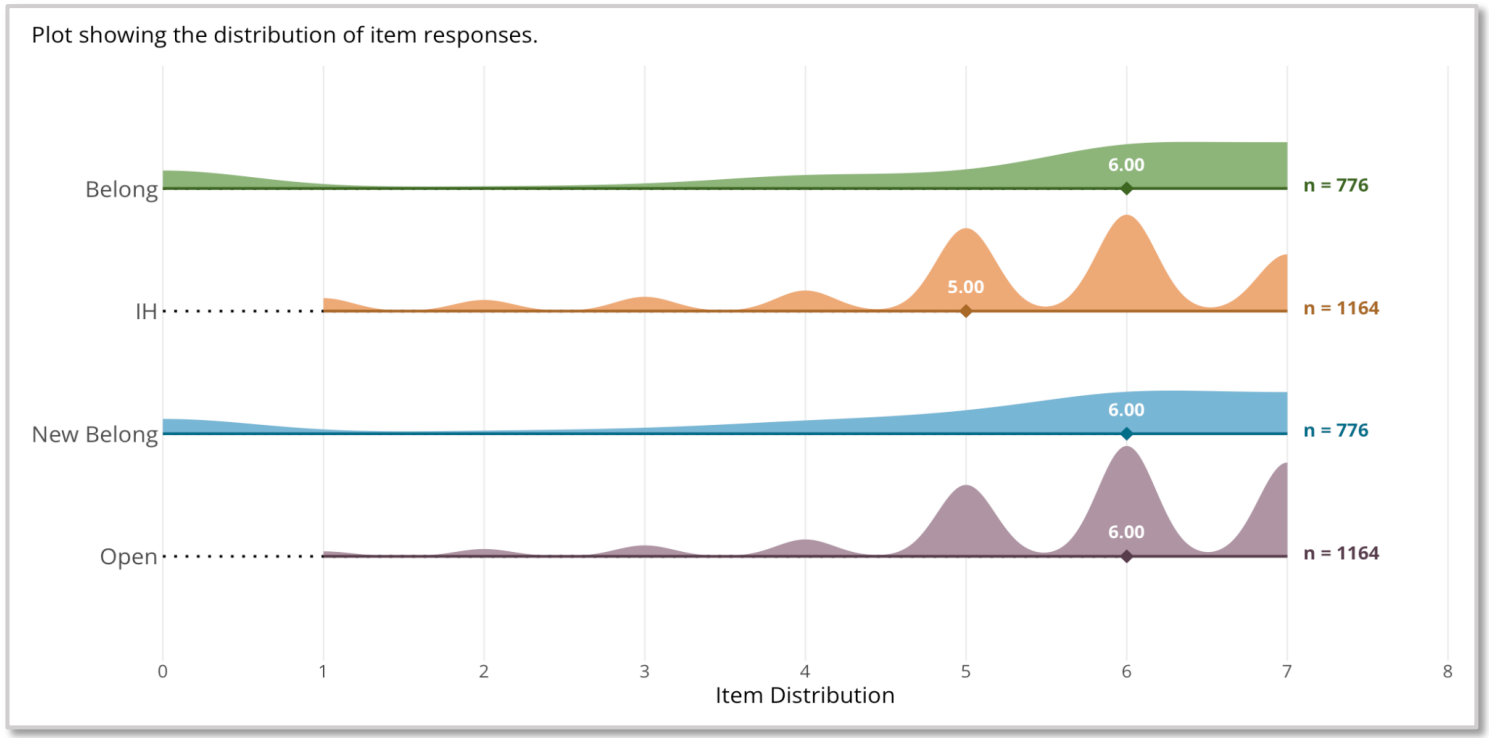


Facets, Themes

the “look” of your graphic



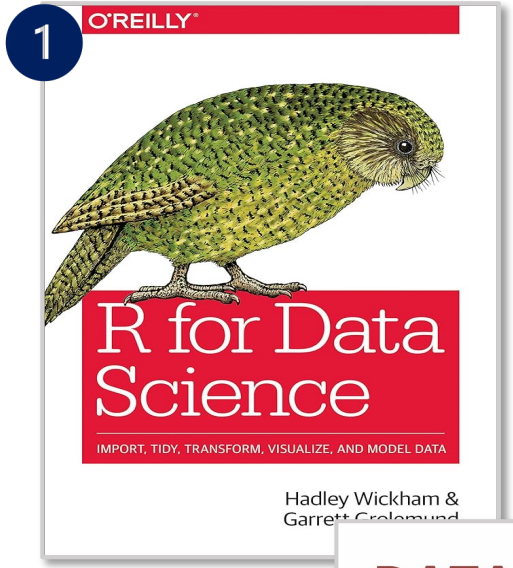
Extreme example of using **theme()** to completely change the look of your visualization!

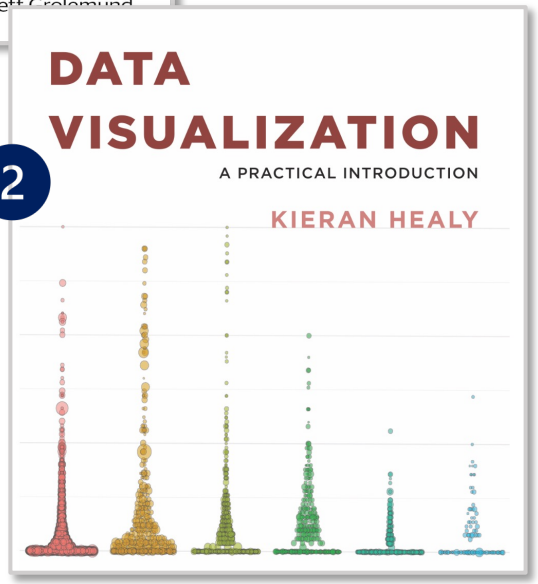


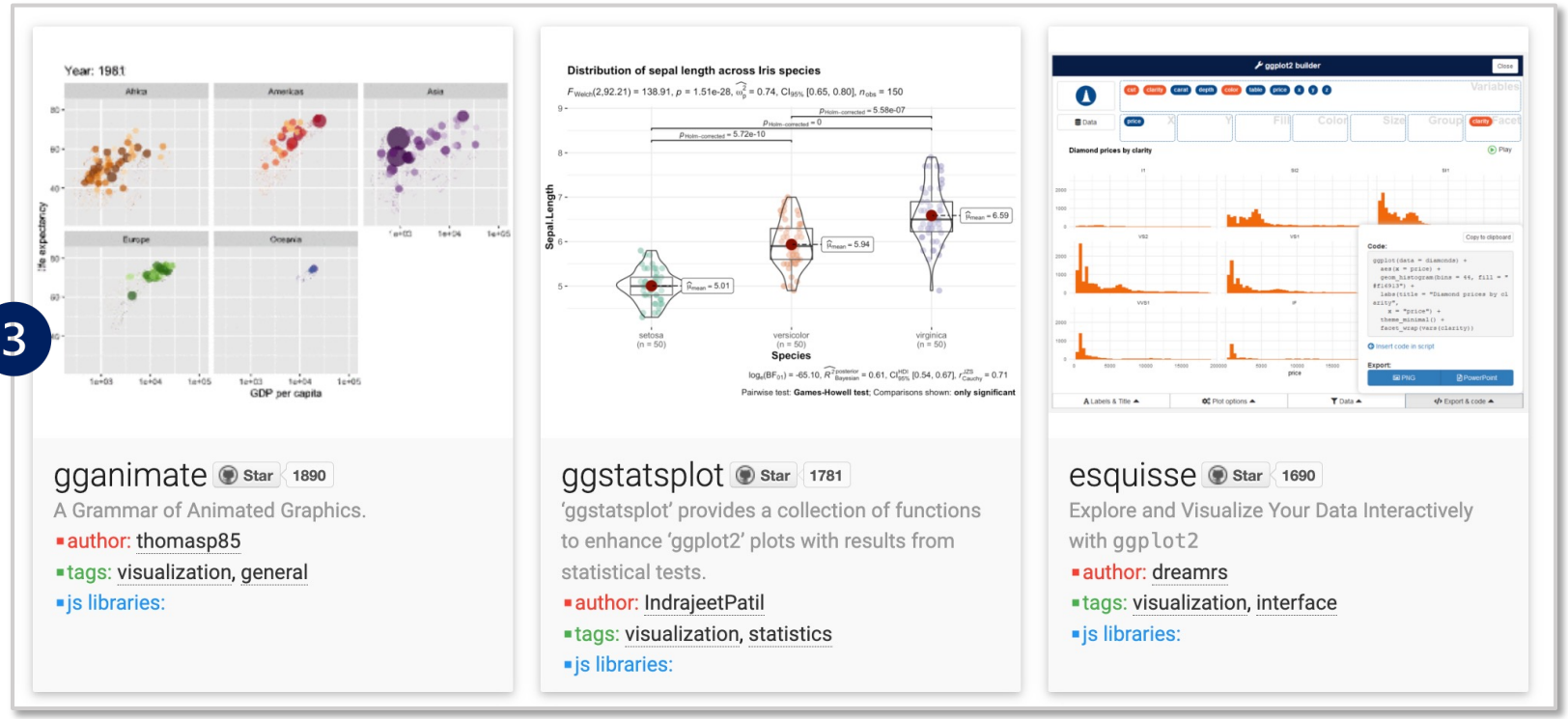
Resources

Resources that might be helpful

1. R for Data Science (Wickham & Grolemund)
2. Data Visualization: A Practical Introduction (Healy) (Package)
3. ggplot extensions gallery (extensions for cool and plots!)







note the second link is to the package, *not* the book

Thank you!
Questions ?

Next Steps

R Markdown take-home to practice the basics of ggplot

Next workshop (July 23rd):
Review
Advance ggplot tips
Overview of visualization Basics (Bar plot, histogram, boxplot)
Implementing visualization in analysis pipeline
(EDA → Analysis → Visualizing) (if time permits)