DATAMOTUS

DATA MINING AND STATISTICAL ANALYSIS SOLUTIONS

Data Visualization

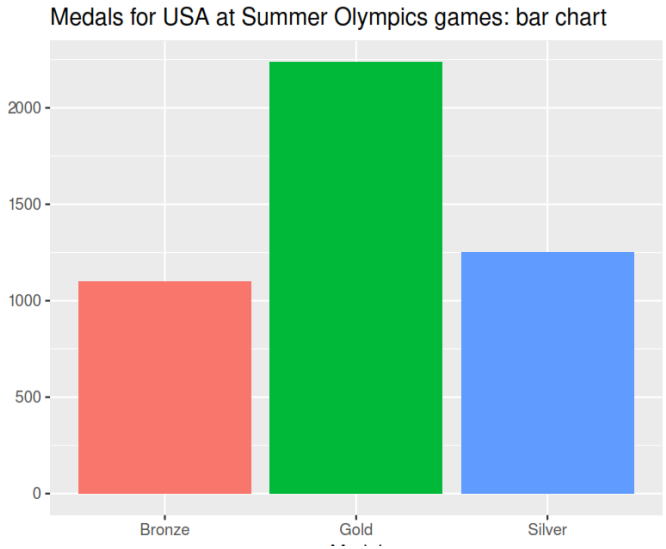
Saturday, September 28, 2019

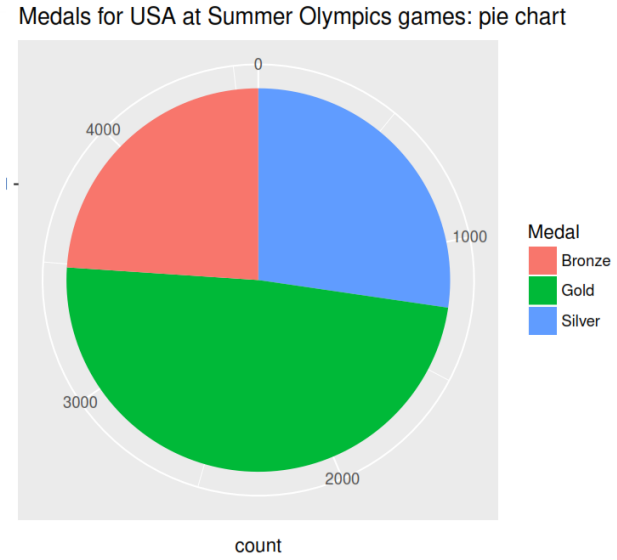
Grammar of graphics

DATAMOTUS

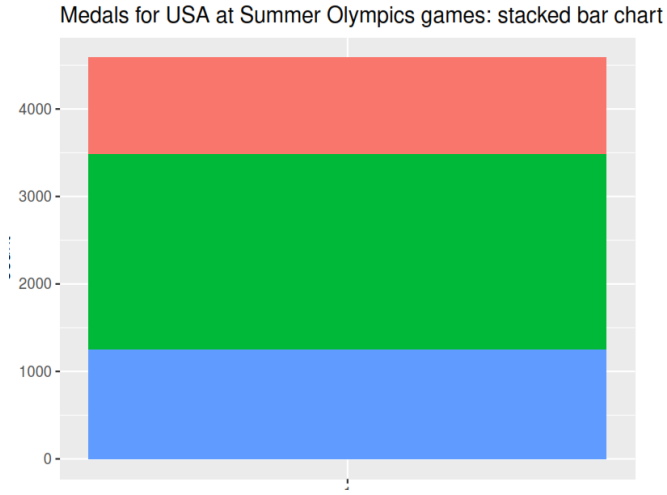
- Different types of graphs may, at first glance, appear completely distinct.
- However, graphs share many common elements, such as coordinate systems and using geometric shapes to represent data.
- By making different visual choices (Cartesian or polar coordinates, points or lines or bars to represent data), you can use graphs to highlight different aspects of the same data.

Grammar of graphics





- This three graphs are representing the same information: Distribution of medals for USA team.
- They have common elements and differ only by how the coordinates are constructed



Grammar of graphics

DATAMOTUS

- Most graphs share many aspects of their structure
- We can think of graphs as visual representations of (possibly transformed) data, along with labels (like axes and legends) that make our message clear.
- Much like the grammar of a language allows you to combine words into meaningful sentences, a grammar of graphics provides a structure to combine graphical elements into figures that display data in a meaningful way.
- The grammar of graphics was originally introduced by Leland Wilkinson in the late 1990s, and was popularized by Hadley Wickham with ggplot, an R graphical library based on the grammar of graphics.
- The grammar of graphics is implemented in many other languages as well, such as **Python**, **D3**, **Julia**.

- Leland Wilkinson defined Object-oriented graphics system and object-oriented design

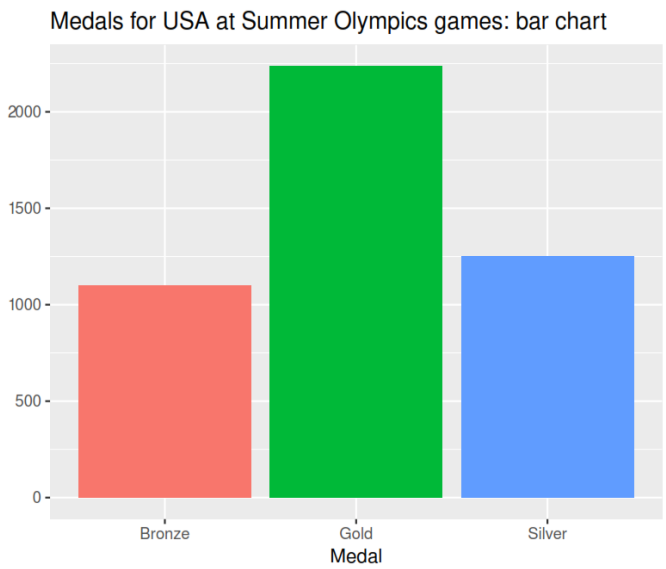
- From OOD perspective, graphics are collection of objects
- If the message between these objects follow a simple grammar, then graphs behave consistently and flexibly.

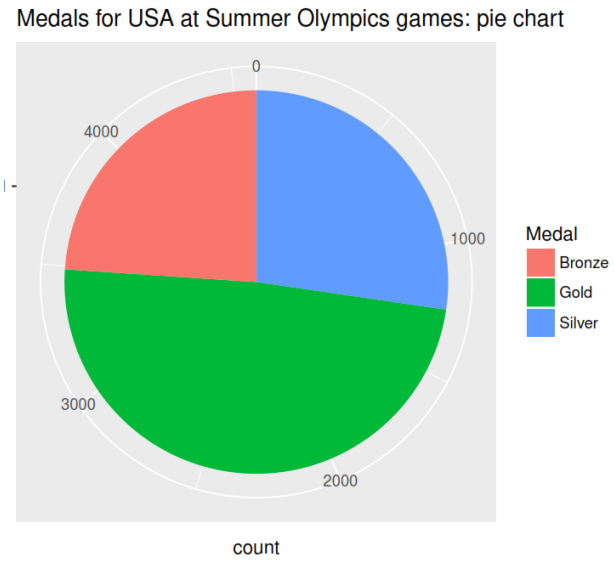
According to Wilkinson, these are components of grammar of graphics

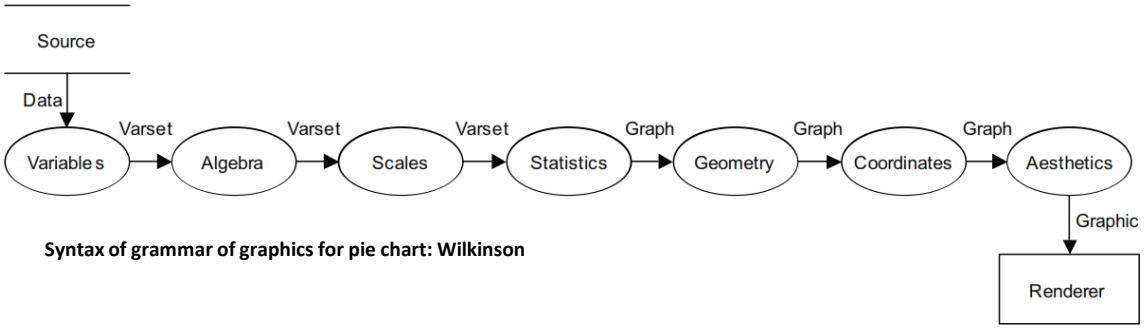
- **data**
- **transformation**: variable transformation (identity, bin, smooth, quantile, etc.)
- **scale**: scale transformation (axis limits, log scale, color, mapping, etc.)
- **coordinates**: Cartesian, polar, map projection, etc.
- **element**: graphic element (point, line, bar, etc.) with attributes (color, symbol, length, etc.)
- **guide**: axes, legends, titles, etc.

Grammar of graphics

If you take a bar chart and change its coordinates to polar coordinate system, you will get the pie chart







Syntax of grammar of graphics for pie chart: Wilkinson

Grammar of graphics

DATAMOTUS

Hadley Wickham's layered grammar of graphics

The basic idea: independently specify plot building blocks (layers) and combine them to create just about any kind of graphical display you want. Building blocks of a graph include:

- data
- aesthetic mapping
- geometric object
- statistical transformations
- scales
- coordinate system
- position adjustments
- faceting

In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars)….aesthetics…are the properties that can be perceived on the graphic. Each aesthetic can be mapped to a variable, or set to a constant value.

Iris dataset:

DATAMOTUS

There are three classes of iris flower in the dataset

Iris setosa

Iris versicolor

Iris virginica

Iris dataset: The variables

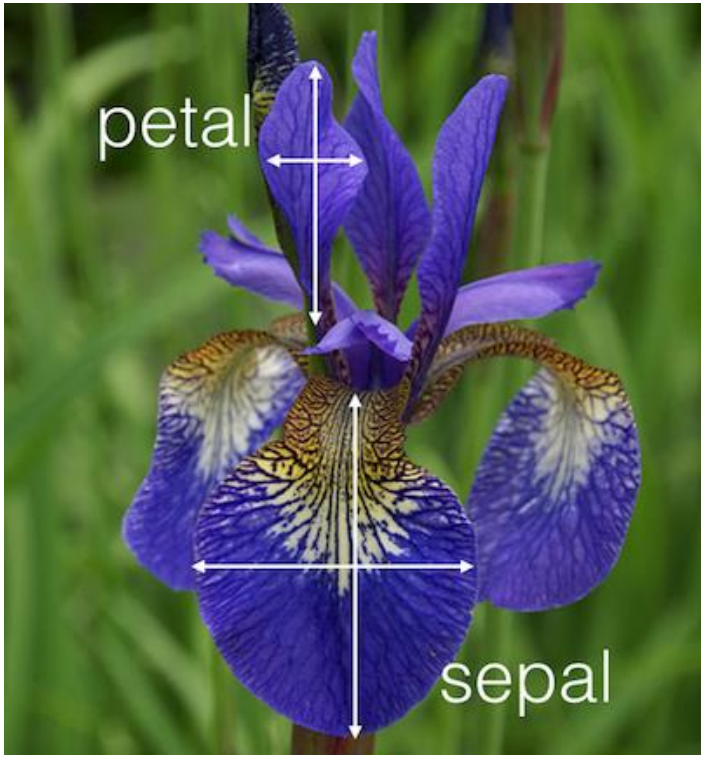
Species – nominal variables with the class of iris – *setosa, versicolor, virginica.*

Sepal. Length

Sepal.Width

Petal.Length

Petal.Width



Grammar of graphics

DATAMOTUS

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

We want to create a scatterplot showing relationship between continuous variables

DATAMOTUS

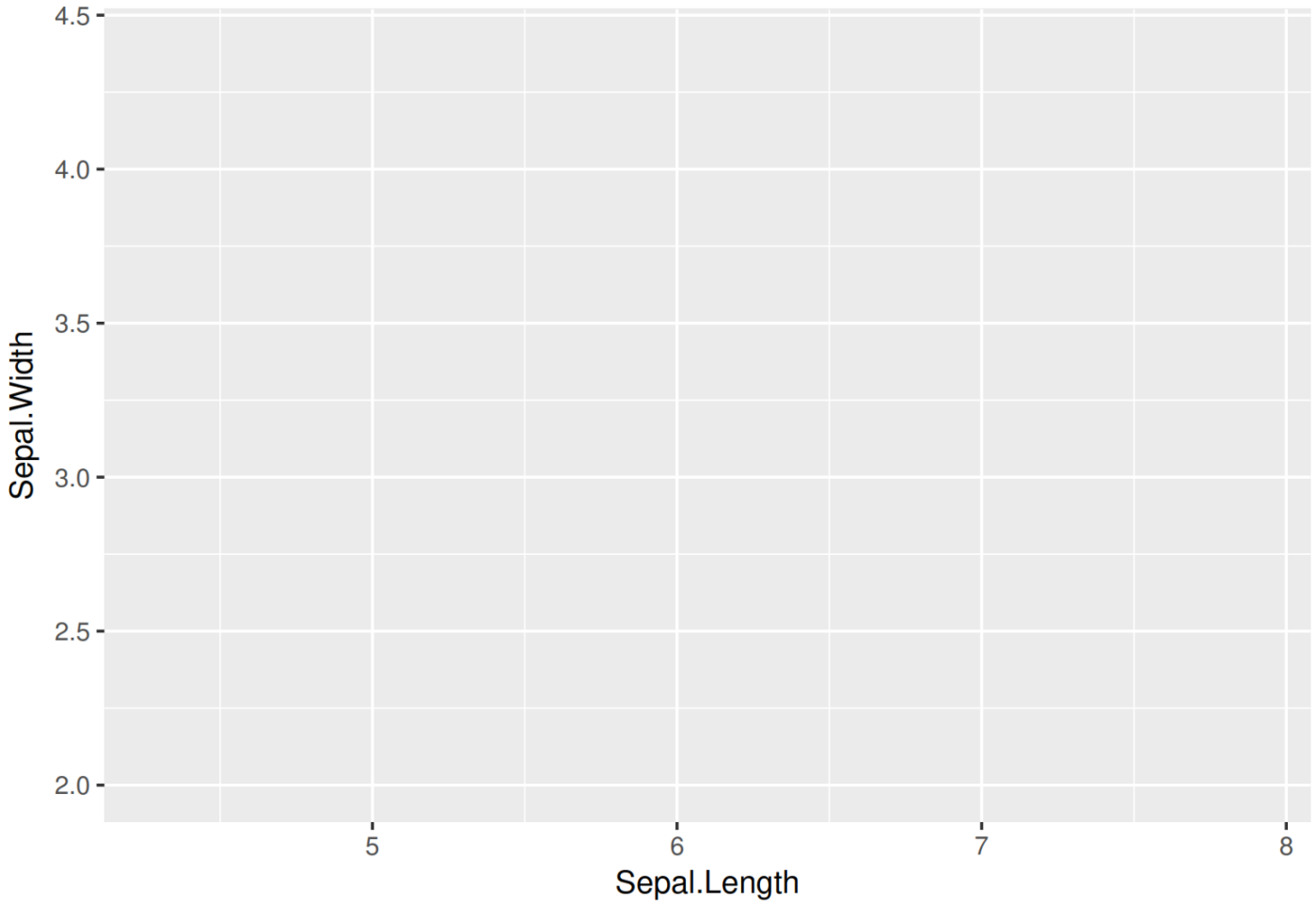
Mapping aesthetics do data

- What is a scatterplot?
- A scatterplot represents each observation as a point (•), positioned according to the value of two variables.
- As well as a horizontal and vertical position, each point also has a size, a colour and a shape. These attributes are called aesthetics, and are the properties that can be ***perceived on the graphic***.
- Each aesthetic can be mapped to a variable, or set to a constant value.

Grammar of graphics

- Create ggplot object first: here you specify your dataset and need to specify aesthetics
- The arguments for ggplot can be omitted if you plan to specify them in layers later

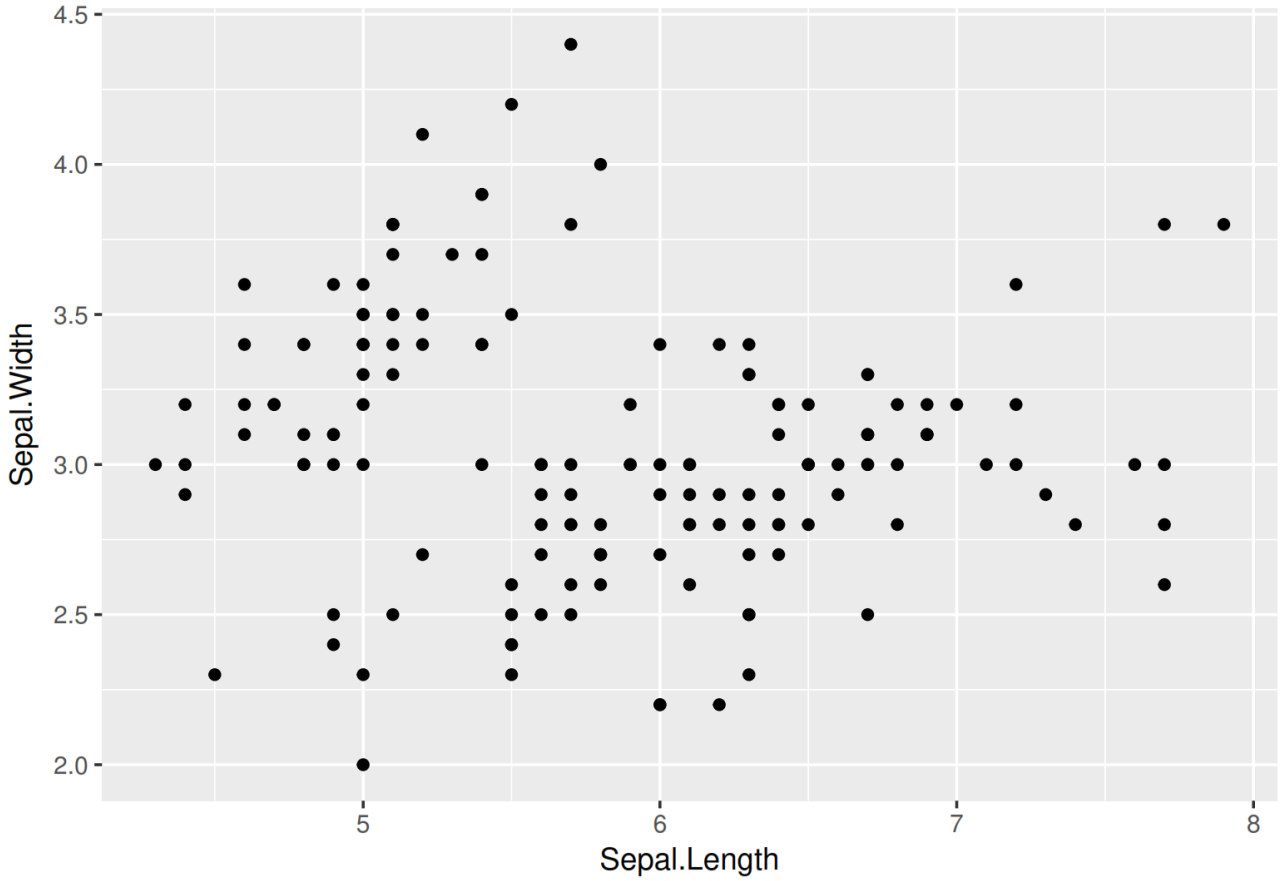
```
library(ggplot2)
g1 <- ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))
g1
```



Grammar of graphics

- Now add the first layer
- Here we add geom_point that will tell R that the shape of data representation is a point, so we will get a scatterplot

```
g1+geom_point()
```



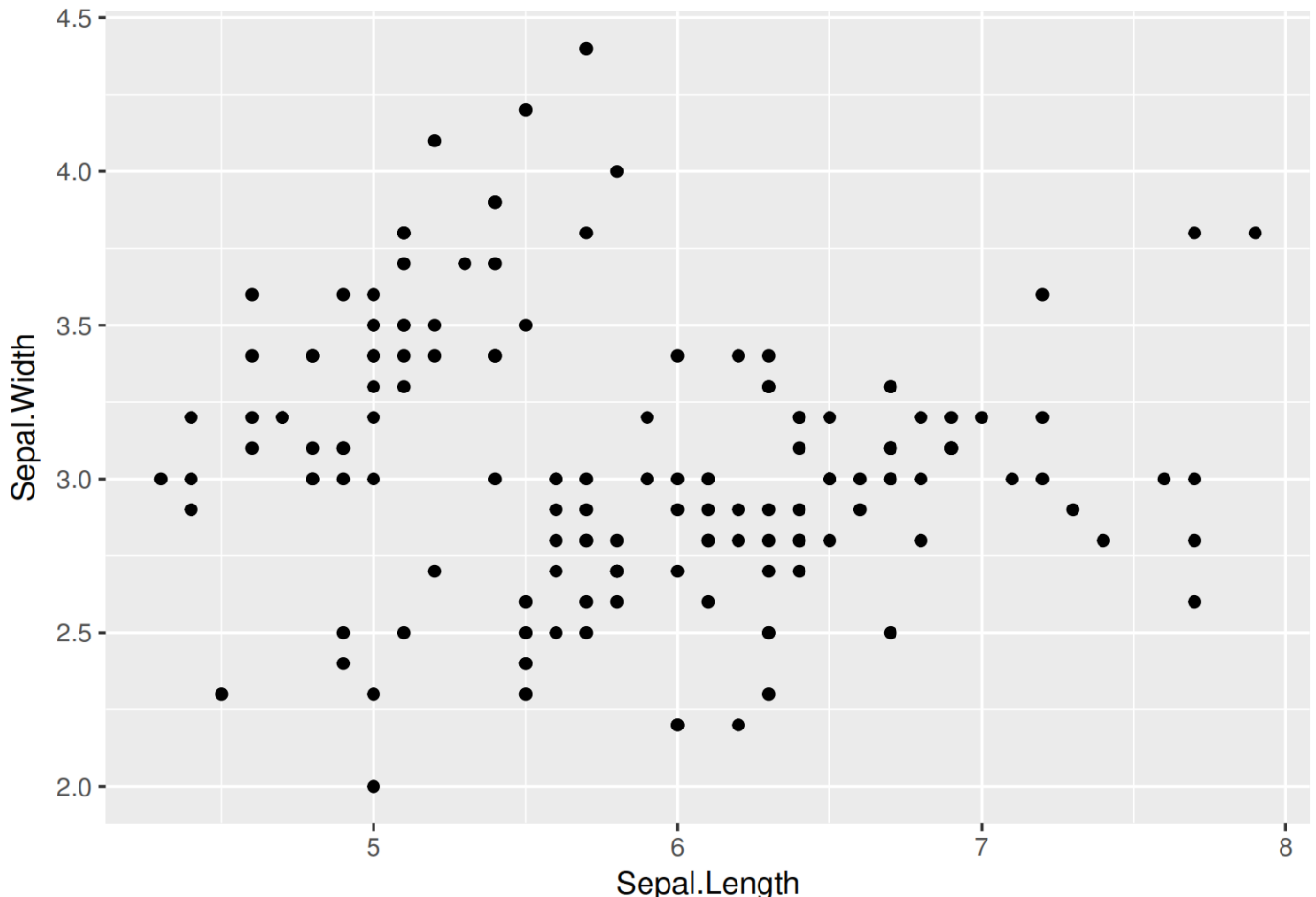
Grammar of graphics

DATAMOTUS

The following two lines are going to create the same graphic

```
ggplot()+geom_point(data=iris, aes(x=Sepal.Length, y=Sepal.Width))
```

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+geom_point()
```



Grammar of graphics

from point of view of grammar of graphics

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+geom_point()
```

Grammar of graphics

DATAMOTUS

If you want to add layer starting from a new line, then leave + at the previous line

Right

```
ggplot()+
geom_point(data=iris, aes(x=Sepal.Length,y=Sepal.Width))
```

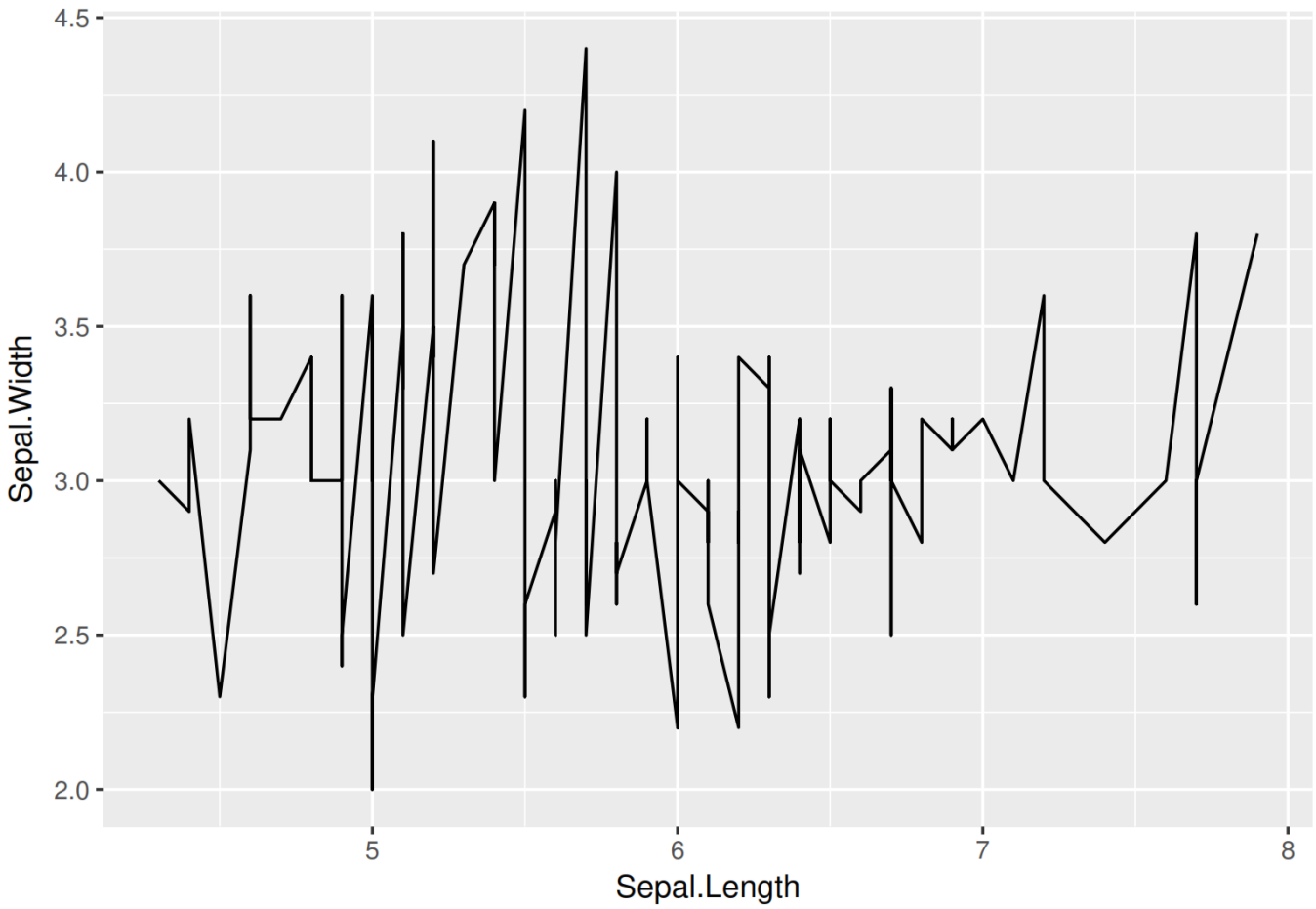
Wrong

```
ggplot()
+ geom_point(data=iris, aes(x=Sepal.Length,y=Sepal.Width))
```

Grammar of graphics

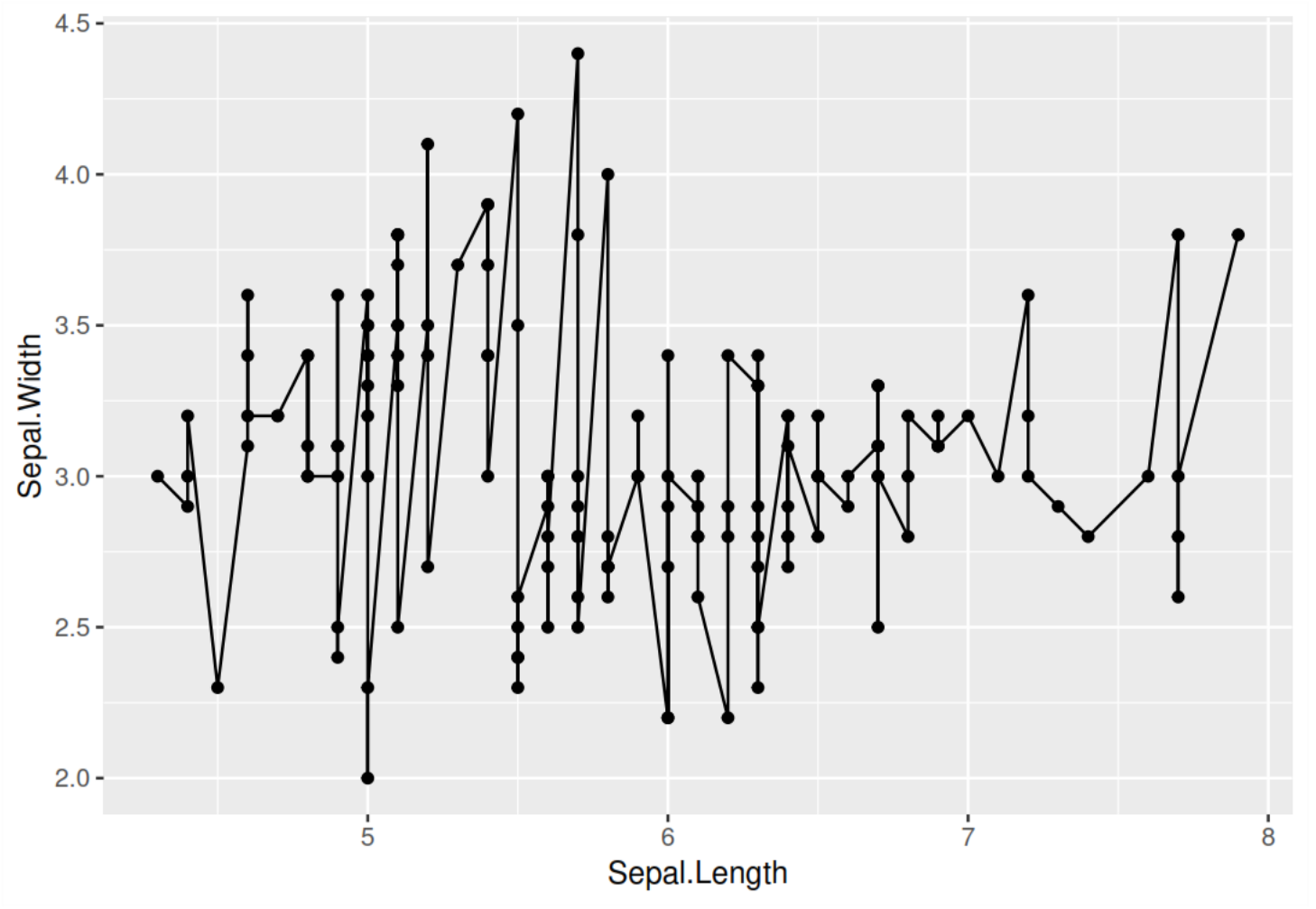
- All geometric objects in ggplot have the following naming: geom_xxx
- Now try geom_line()

```
g1+geom_line()
```



Line and scatterplots together

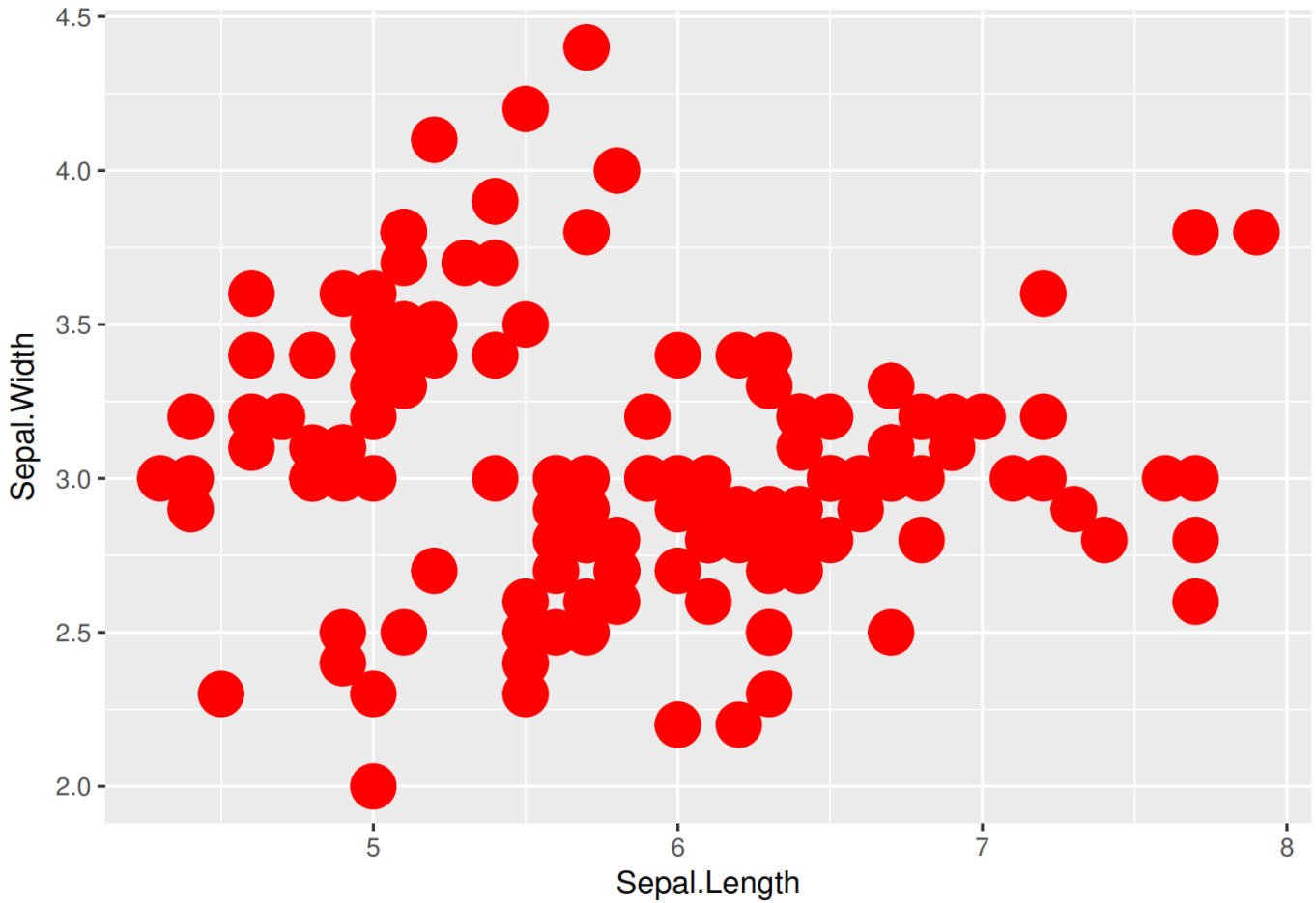
```
g1+geom_line()+geom_point()
```



Grammar of graphics

- Other aesthetics like color, size, shape are given as constants here
- You can change them

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+
  geom_point(color="red", size=7)
```



Grammar of graphics

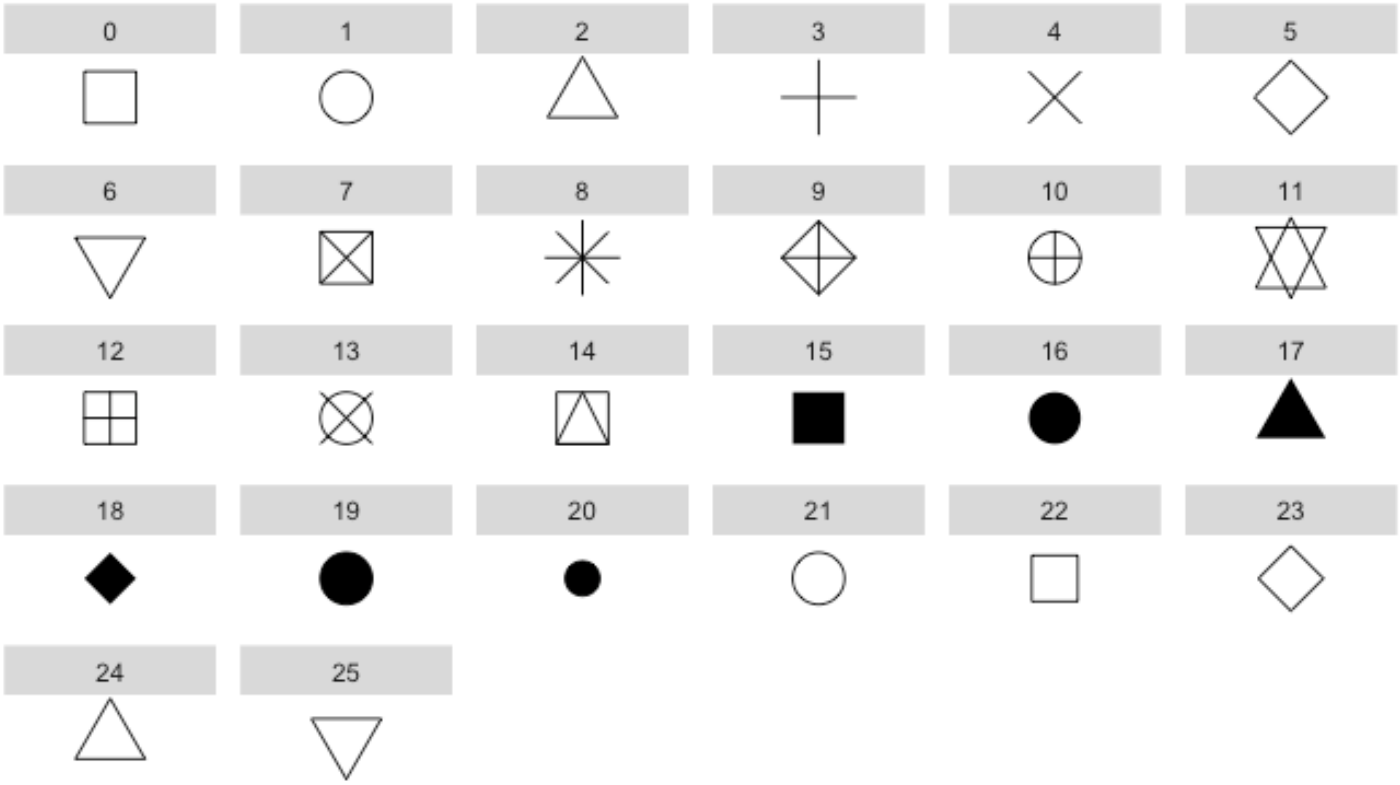
DATAMOTUS

Change also the shape

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+
  geom_point(color="blue", size=7, shape=9)
```

DATAMOTUS

shapes in ggplot2



The grammar

DATAMOTUS

Effective Visualization with Grammar of Graphics

- Grammar is defined as a set of structural rules which helps define and establish the components of a language
- The whole system and structure of a language usually consists of syntax and semantics.
- A grammar of graphics is a framework that enables us to concisely describe the components of any graphic
- Instead of random trials and errors, follow a layered approach by using defined components to build a visualization

Main components of the grammar fo graphics

- data
- aesthetic mapping
- geometric object
- scales
- statistical transformations
- coordinate system
- faceting

aesthetics

- X coordinate
- Y coordinate
- Color
- Shape
- Size
- Group
- etc

geom_

Around 30 different geometric objects to chose from
among them
geom_bar – creates a bar chart
geom_boxplot – creates a boxplot
geom_line – line chart
geom_histogram - ?

Question !
what will geom_vline(),
geom_hline() and geom_abline()
do ?

Scales

- Scales are helping us to alter, change the mapping of variables to aesthetics
- to find the layer for the specific aesthetics just type
- scale_ and fill the rest

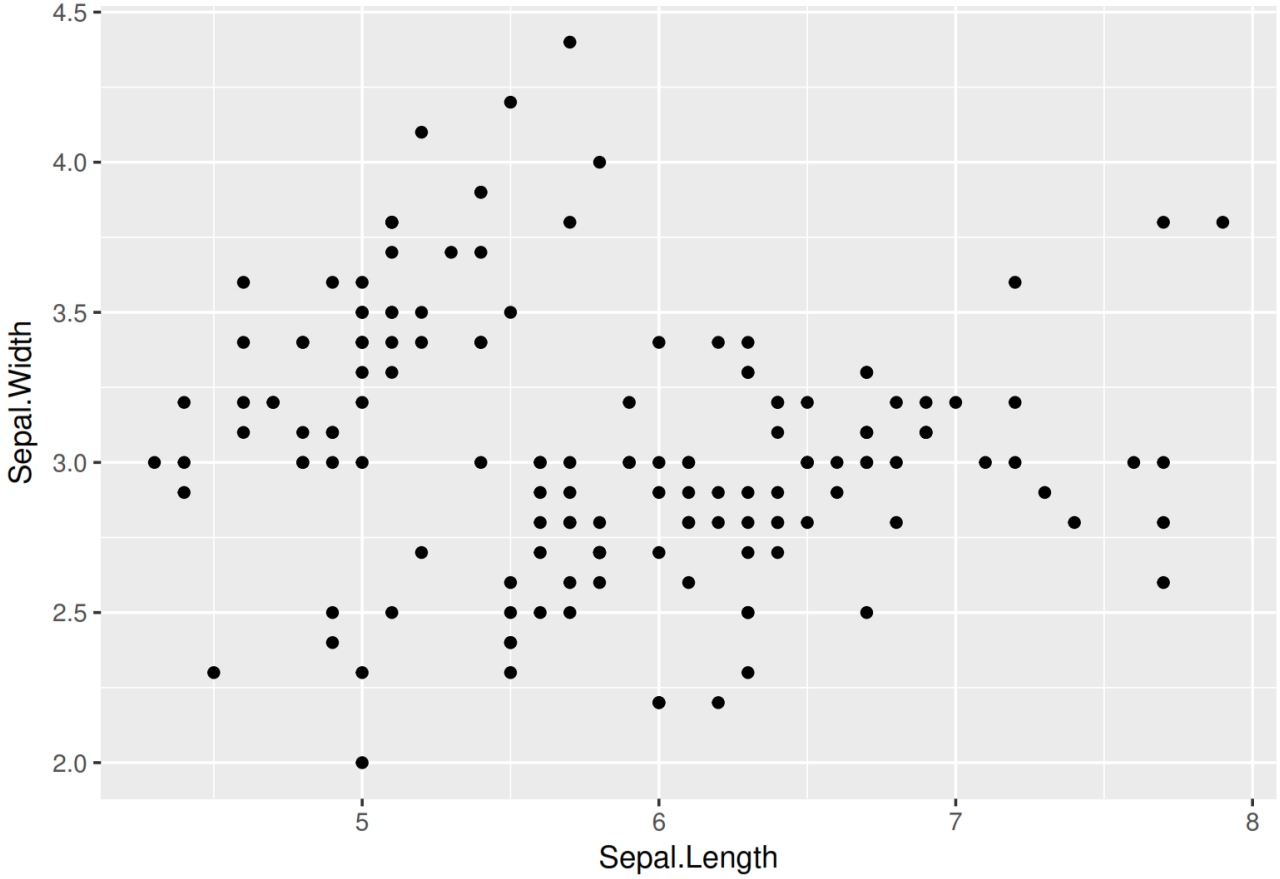
Example

scale_x_ will give you all possible scaling for x aesthetics
scale_y_ for the y aesthetics

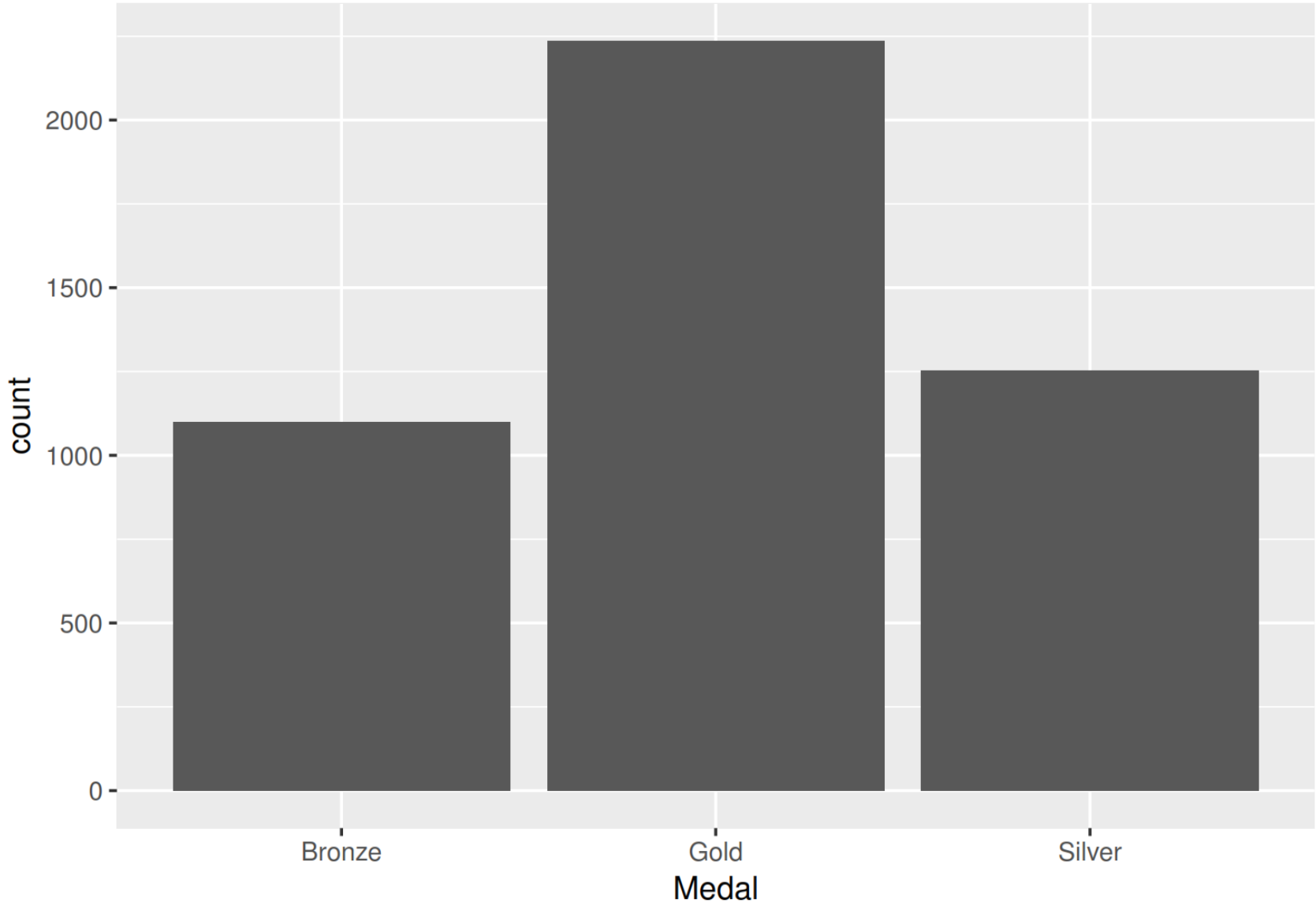
scale_color_manual()
scale_color_brewer(), etc

A statistical transformation, or stat, transforms the data, typically by summarizing them in some manner.

What kind of transformation is used here ?



What kind of statistical transformation is done here ?



geom_point()

```
geom_point(mapping = NULL, data = NULL, stat = "identity",
position = "identity", ..., na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE)
```

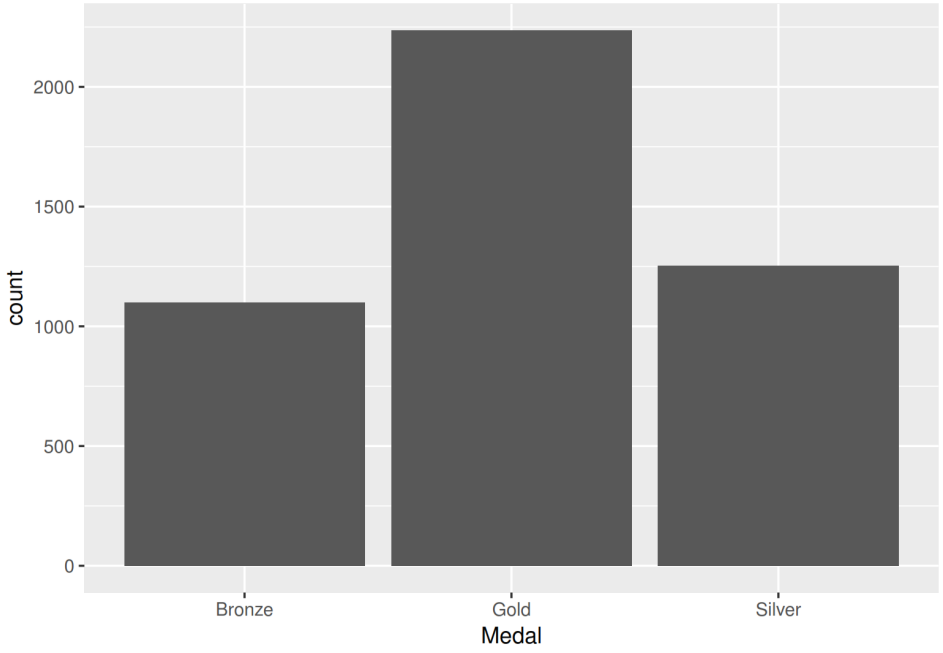
https://ggplot2.tidyverse.org/reference/geom_point.html

identity transformation

$$f(x) = x$$

geom_bar(mapping = NULL, data = NULL, **stat = "count"**,
position = "stack", ..., width = NULL, binwidth = NULL,
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

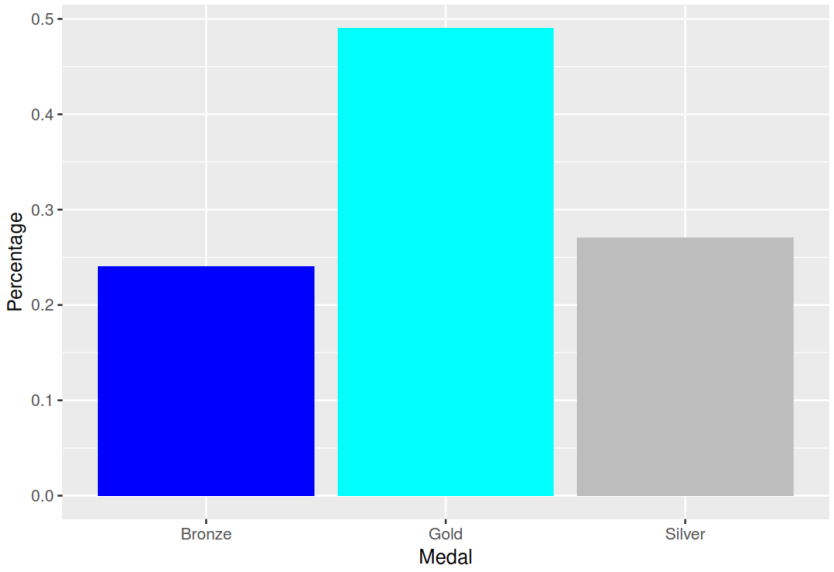
https://ggplot2.tidyverse.org/reference/geom_bar.html

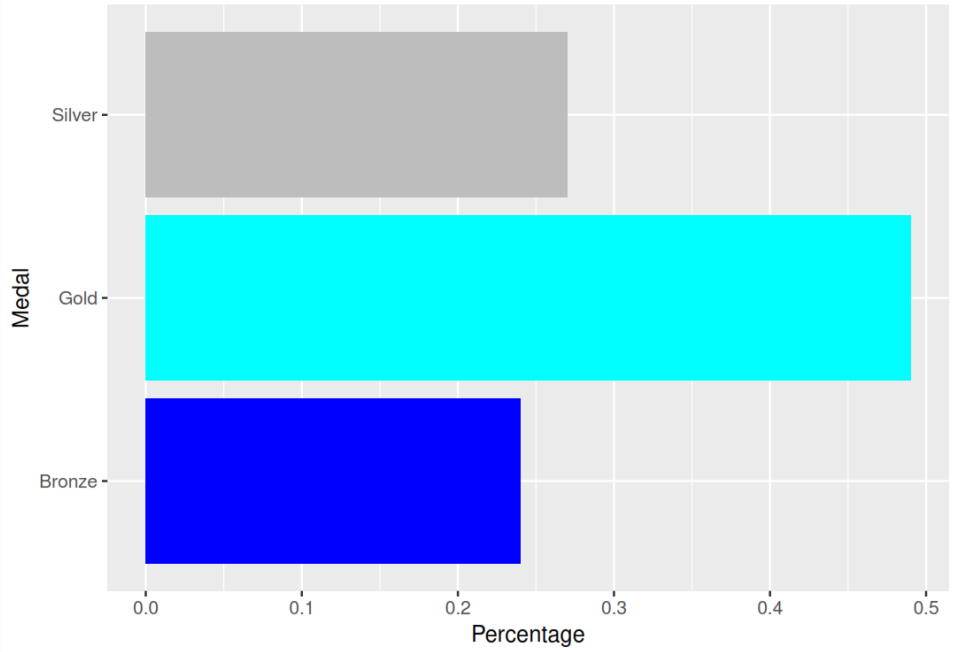


coordinate system

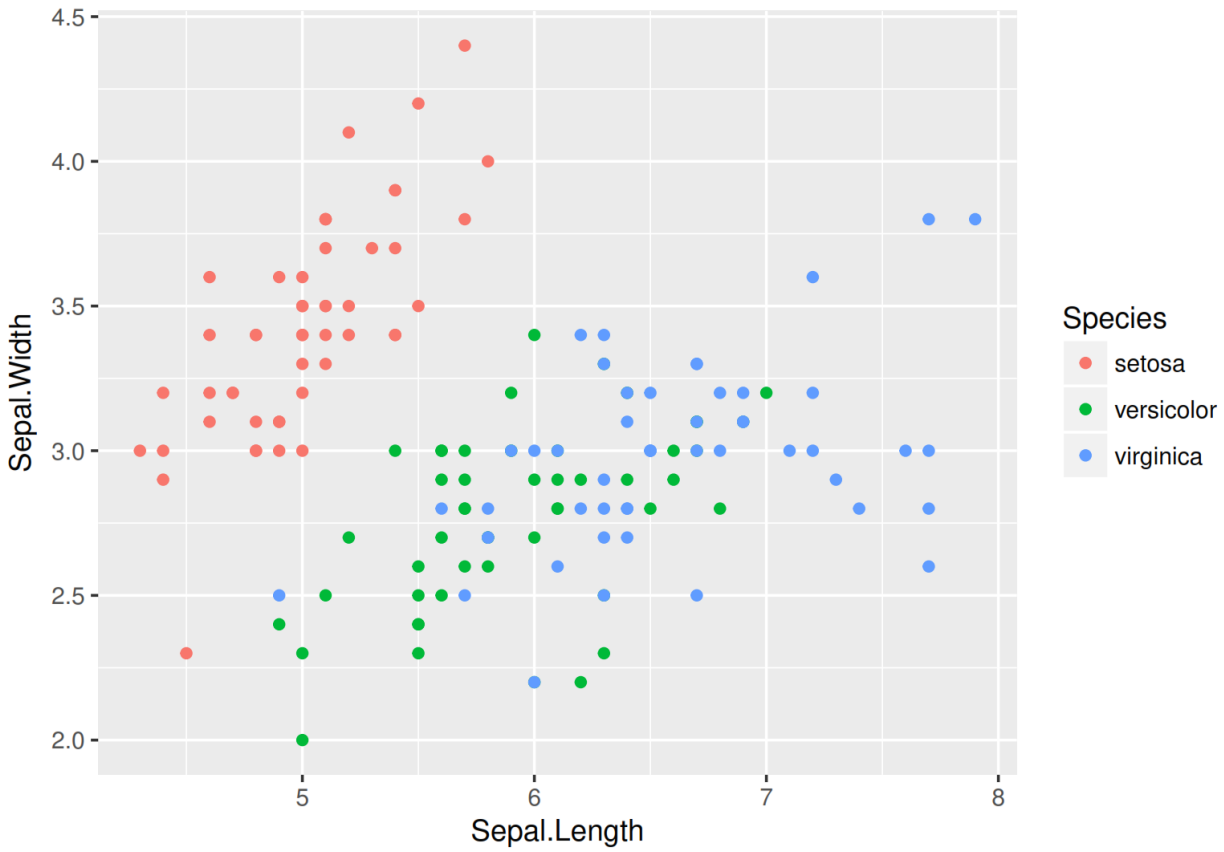
coord_cartesian()
coord _polar()
coord_flip()

Coord_flip()


0.5
0.4
0.3
0.2
0.1
0.0
Percentage
Bronze
Gold
Silver
Medal

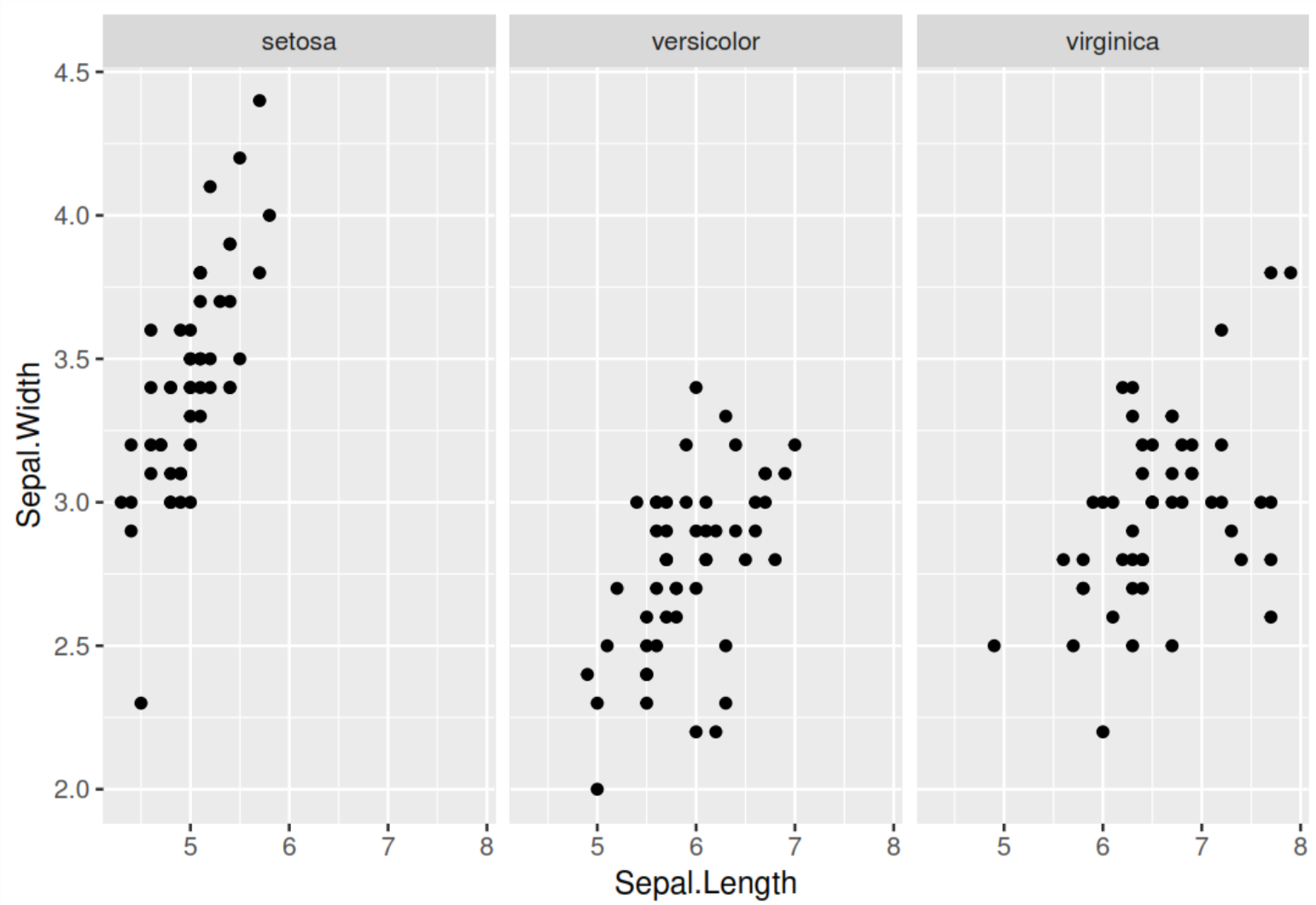

Silver
Gold
Bronze
Medal
0.0
0.1
0.2
0.3
0.4
0.5
Percentage

- We can use shape, size, color to compare subgroups on one graph.
- Faceting takes an alternative approach: It creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset in an arrangement that facilitates comparison.



DATAMOTUS

With faceting



Colors in R

Colors in R

DATAMOTUS

- You can specify colors in R by index, name, hexadecimal, or RGB.

  For example col=1, col="white", and col="#FFFFFF" are equivalent.

- You can find names, numbers and codes of colors here

```
length(colors())
```

```
## [1] 657
```

```
colors()[1:10]
```

```
##  [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
##  [5] "antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
##  [9] "aquamarine1"    "aquamarine2"
```

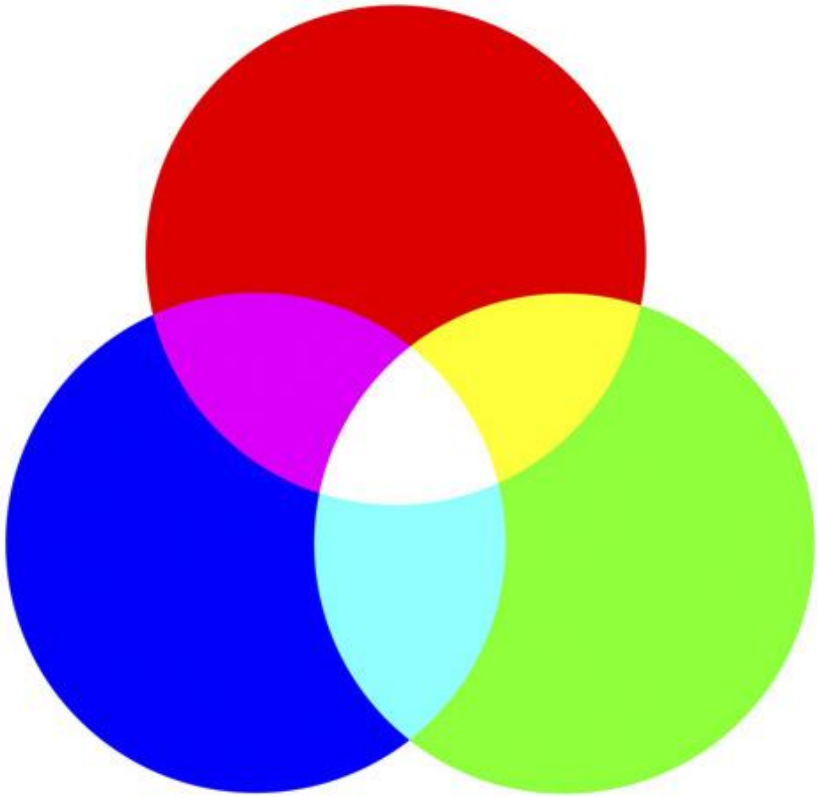
DATAMOTUS

RGB color model

- Computers create the colors we see on a monitor by combining 3 primary colors of light:
  - red
  - green
  - blue
- This combination is known as RGB color model
- Each color light is also referred to as a channel

Colors in R

A computer screen displays a color by combining **red** light, **green** light and **blue** light, the so-called RGB model



Colors in R

Any color you see on a monitor can be described by a series of 3 numbers (in the following order):

- a red value
- a green value
- a blue value
- e.g. red=30, green=200, blue=180

Colors in R

- The amount of light in each color channel is typically described on a scale from 0 (none) to 255 (full-blast)
- Alternatively, scales can be provided as percent values from 0 (none) to 1 (100%)

Colors in R

Some reference colors:

RGB Values	Color
(255, 0, 0)	red
(0, 255, 0)	green
(0, 0, 255)	blue
(0, 0, 0)	black
(255, 255, 255)	white

The closer the three values get to 255 (100%), the closer the resulting color gets to white

Colors in R

DATAMOTUS

Look at the first six colors

```
col2rgb(col=1:6)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
## red      0  255    0    0    0  255
## green    0    0  205    0  255    0
## blue     0    0    0  255  255  255
```

Colors in R

DATAMOTUS

Get RGB and Hex notation for color gold

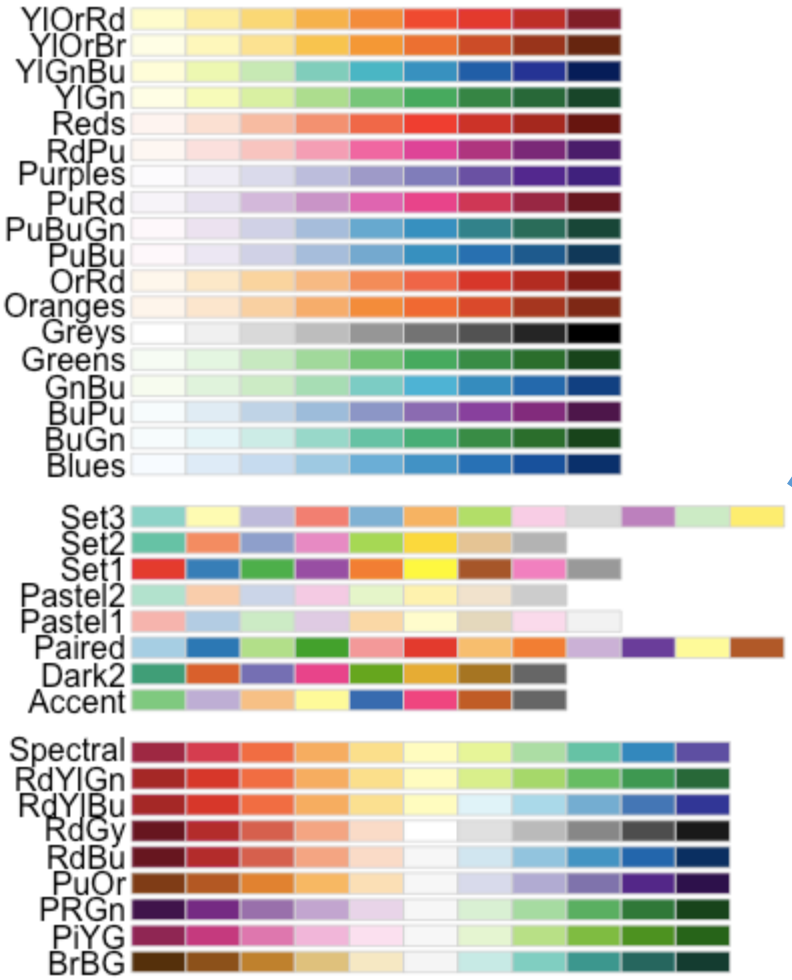
```
col2rgb("gold")
```

```
##       [,1]
## red    255
## green  215
## blue     0
```

```
# And the hexadecimal notation

rgb(255,215,0, maxColorValue = 255)
```

```
## [1] "#FFD700"
```

Colors in R



RColorBrewer allows you to chose colors from a pallete

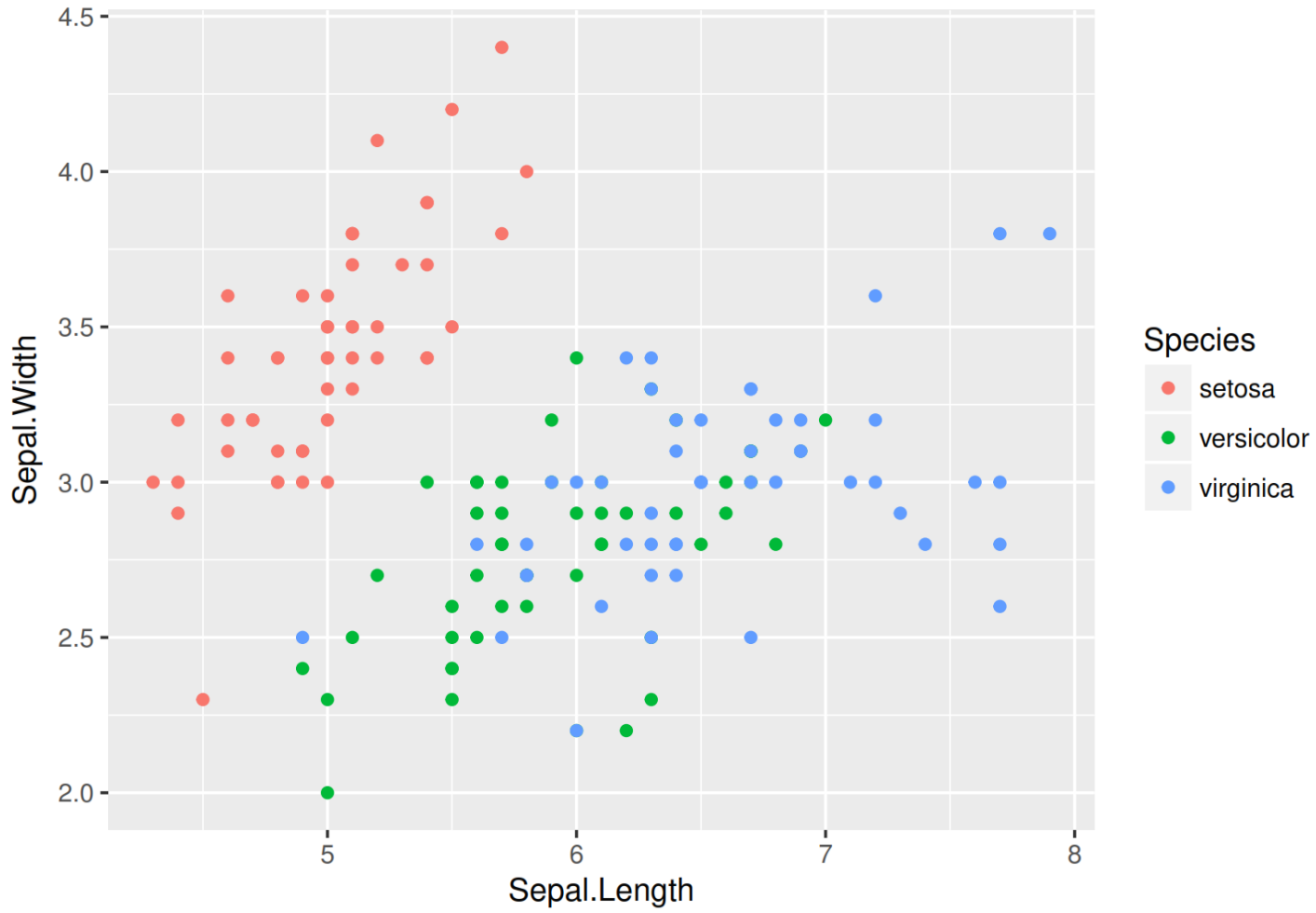
```
library(RColorBrewer)
brewer.pal(n=5, name = "Set3")

## [1] "#8DD3C7" "#FFFFB3" "#BEBADA" "#FB8072" "#80B1D3"
```

Grammar of graphics

DATAMOTUS

Define color aesthetics as a variable (Color should be categorical variable)

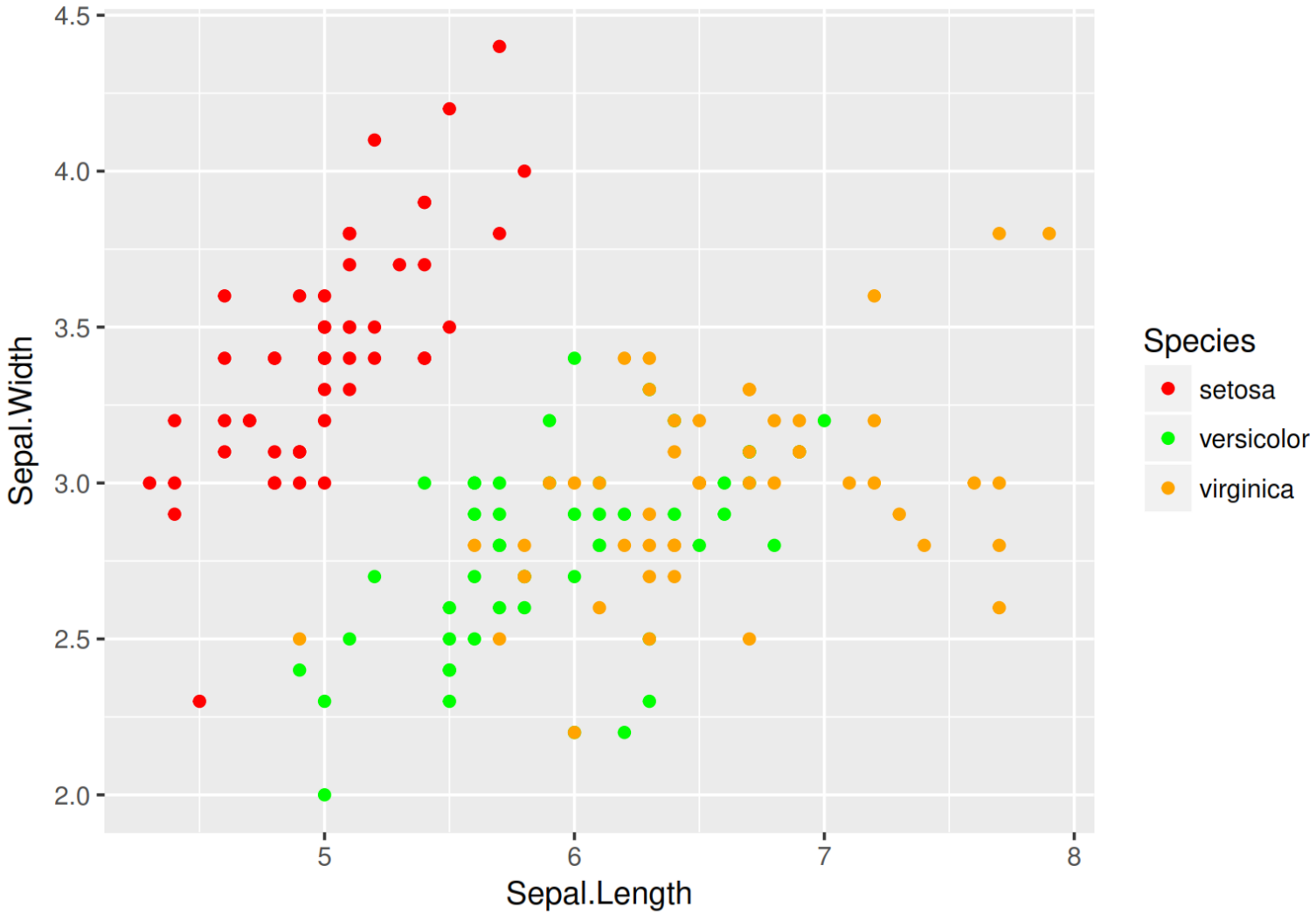
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+
  geom_point()
```



Grammar of graphics: Change the colors manually, specifying colors by names

DATAMOTUS

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+ geom_point()+
  scale_color_manual(values = c("red", "green", "orange"))
```



Colors in R

The Isle of dogs

```
install.packages("wesanderson")
```

```
library(wesanderson)
wes_palette(name="IsleofDogs2")
```

More on Wes Anderson's colors

Grammar of graphics:

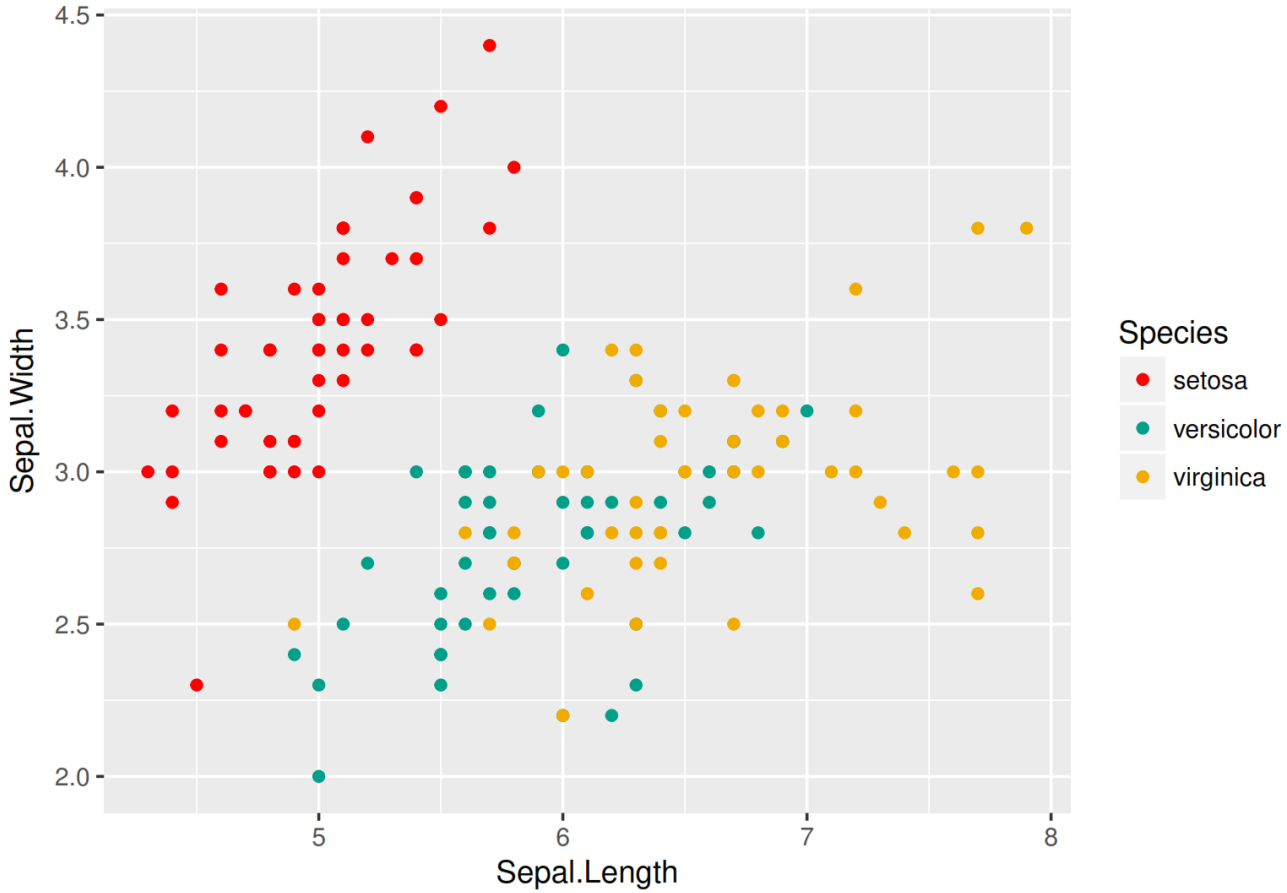
DATAMOTUS

Use the Hex notation and Colors from the [“The Darjeeling Limited”]

```
wes_palettes$Darjeeling1
```

```
## [1] "#FF0000" "#00A08A" "#F2AD00" "#F98400" "#5BBCD6"
```

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+ geom_point()+
  scale_color_manual(values = c("#FF0000", "#00A08A", "#F2AD00"))
```

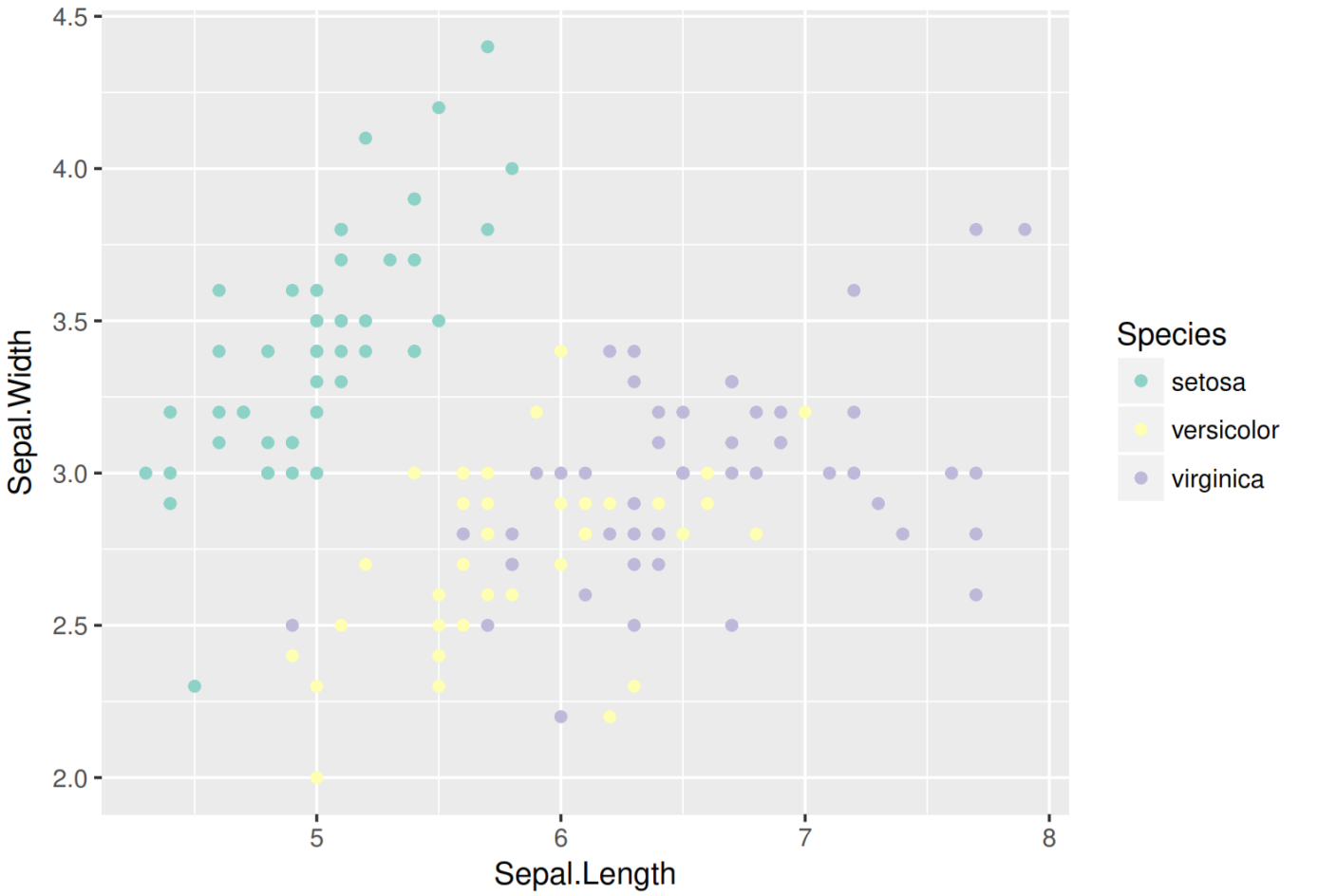


Grammar of graphics

DATAMOTUS

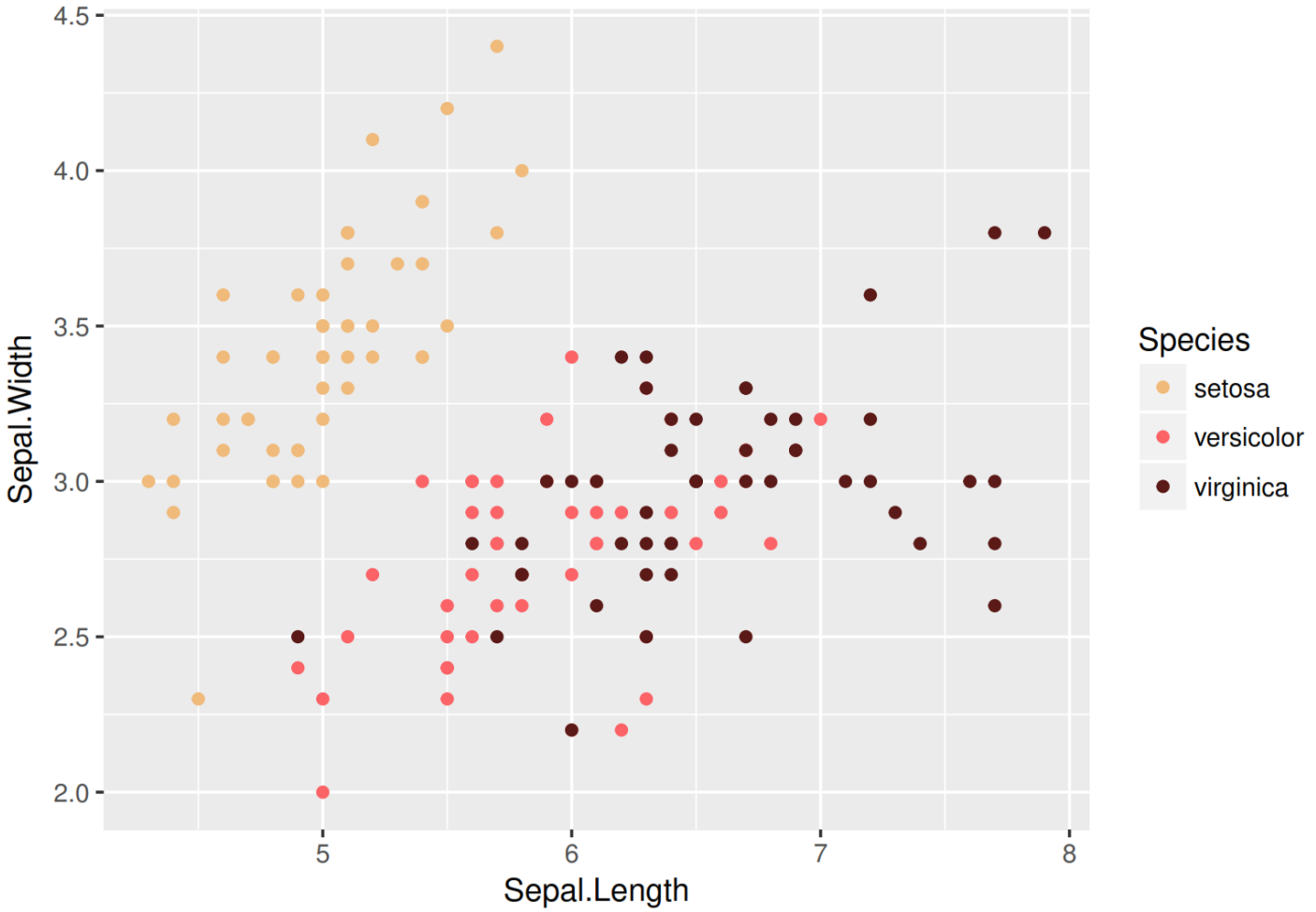
You can chose colors from the RColorBrewer palettes

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+ geom_point()+
  scale_color_brewer(palette="Set3")
```



wesanderson package allows to directly access the palettes as well, colors from The Grand Budapest Hotel

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+ geom_point()+
  scale_color_manual(values=wes_palette("GrandBudapest1"))
```



Grammar of graphics

DATAMOTUS

You can use continuous variable for color aesthetics (darker means smaller is Petal.Length)

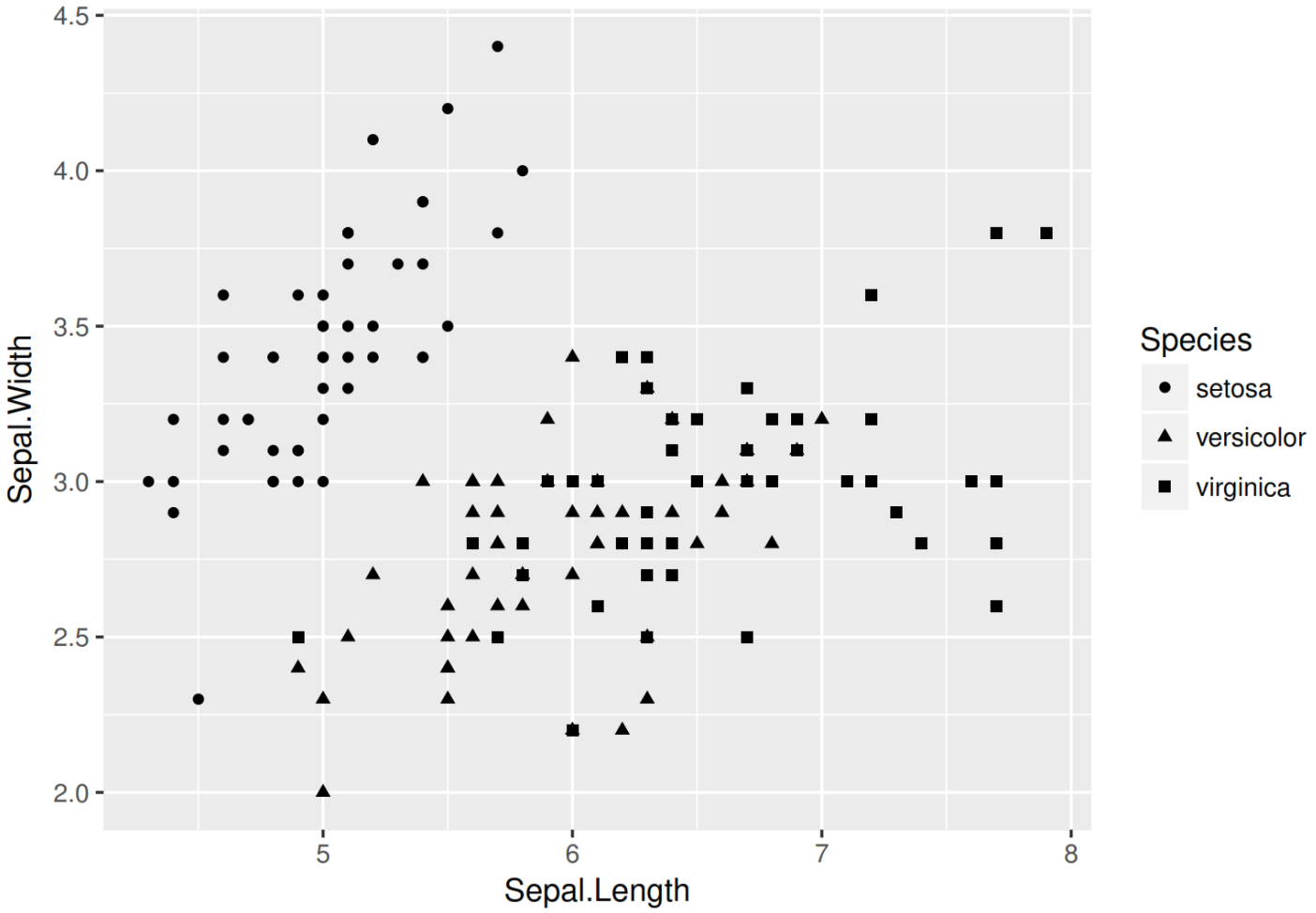
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Petal.Length))+geom_point()
```

Grammar of graphics

DATAMOTUS

Species here determine the shape of the points

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, shape=Species))+geom_point()
```

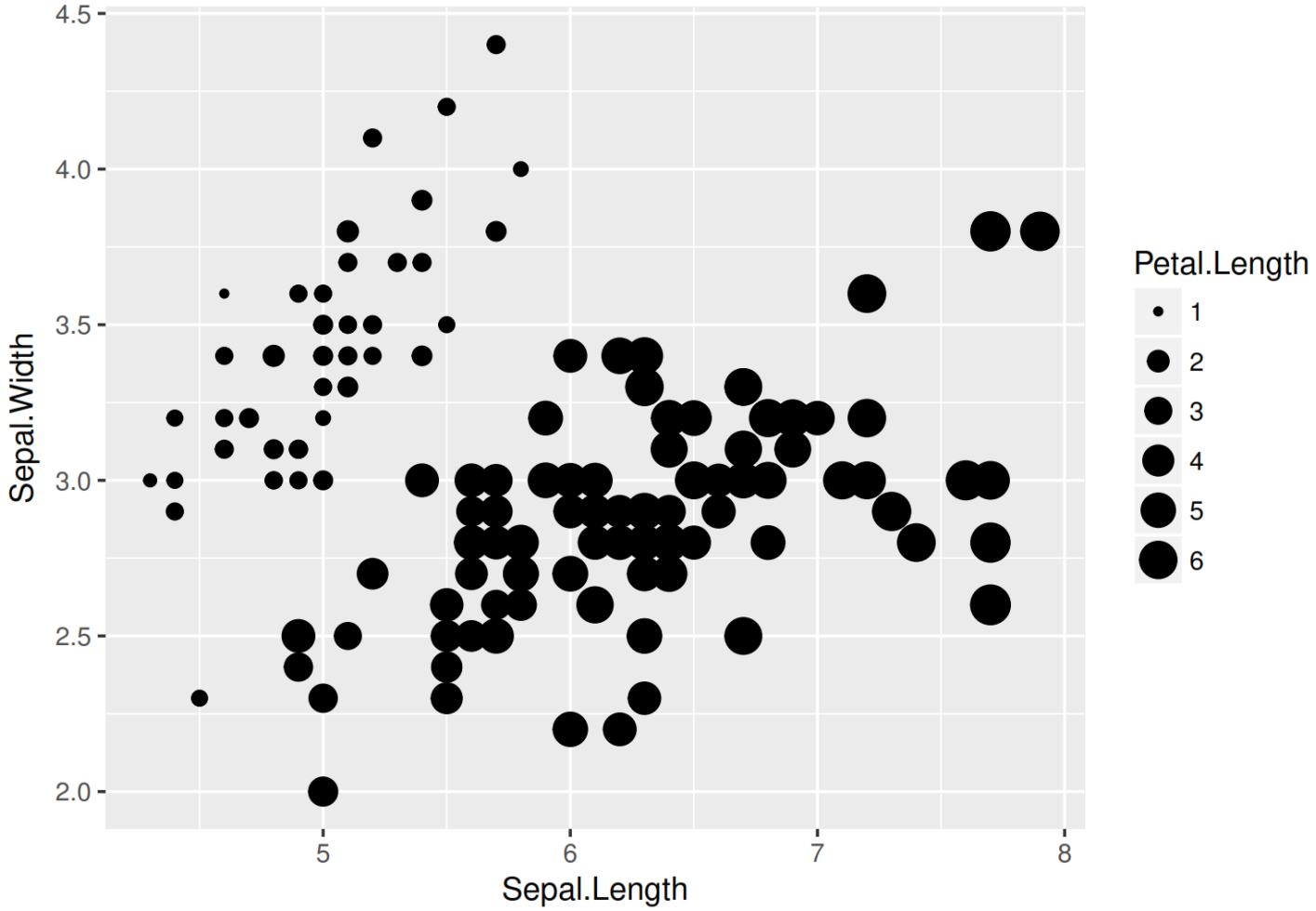


Grammar of graphics

DATAMOTUS

Bubble chart: continuous variable mapped as size aesthetics

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length))+
  geom_point()
```

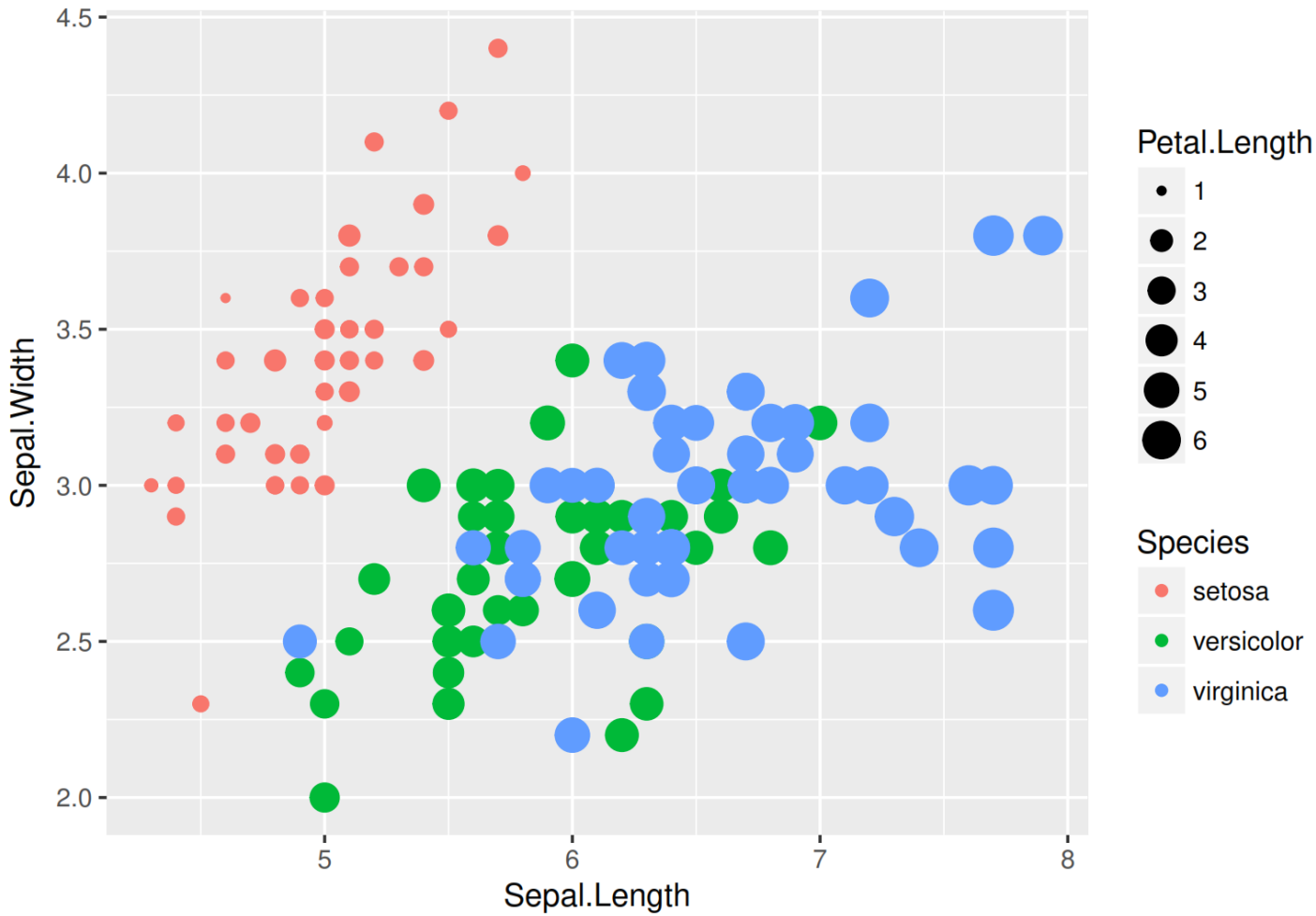


Grammar of graphics

DATAMOTUS

Using size and color aesthetics together

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length, color=Species))+
  geom_point()
```



Histogram

- A **histogram** is an accurate representation of the distribution of numerical data.
- To construct a histogram, the first step is to "bin" the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval.
- The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but are not required to be) of equal size.

Movies dataset

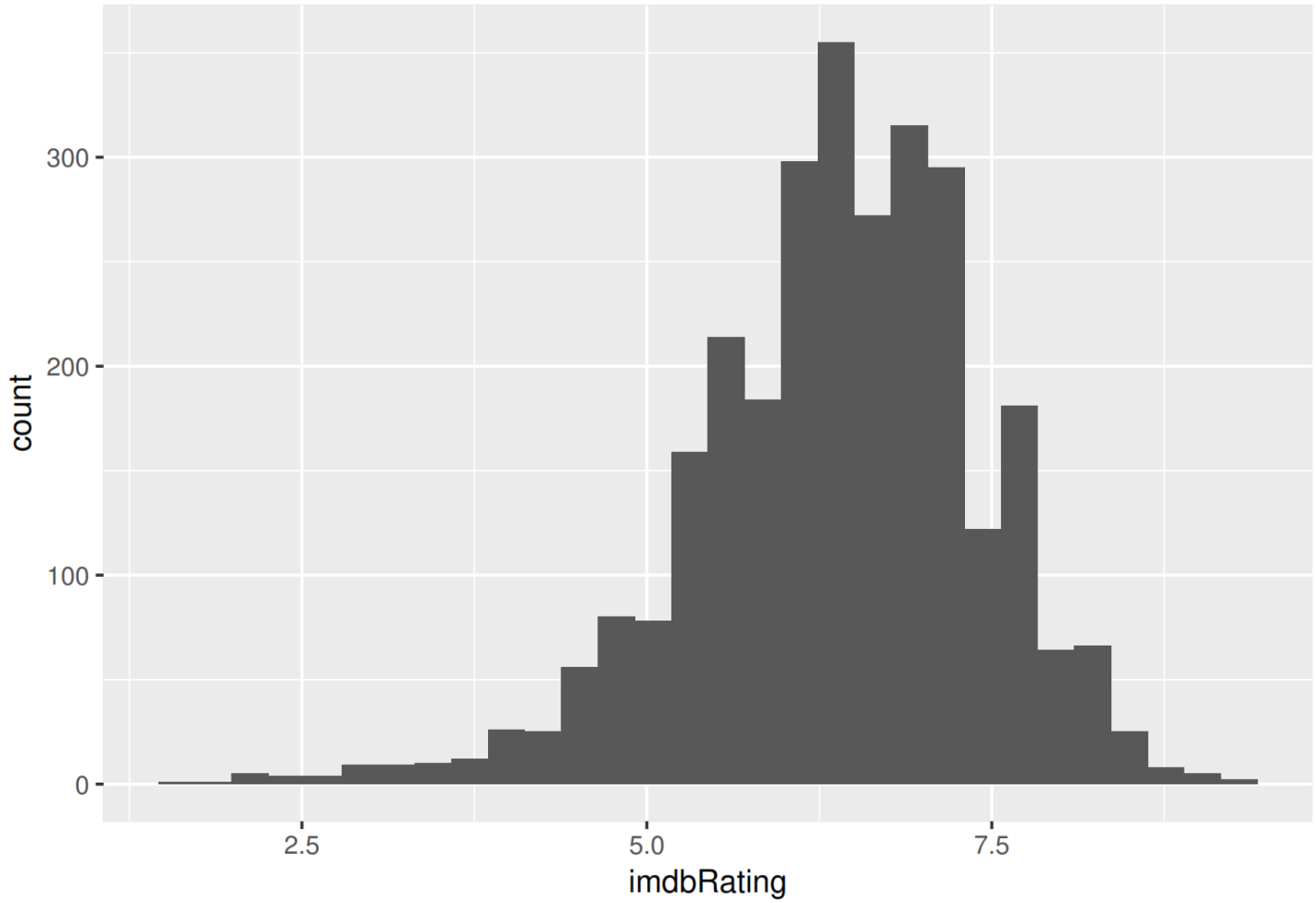
```
movies <- read.csv("movies3.csv")
colnames(movies)
```

```
##  [1] "title"              "genre_first"        "year"
##  [4] "duration"           "gross_adjusted"     "budget_adjusted"
##  [7] "gross"              "budget"             "cast_facebook_likes"
## [10] "reviews"            "index"              "Rated"
## [13] "Genre"              "Director"           "Writer"
## [16] "Actors"             "Plot"               "Language"
## [19] "Country"            "Awards"             "Metascore"
## [22] "imdbRating"         "imdbVotes"          "Production"
## [25] "DVD"                "Release"            "Release_Month"
## [28] "Release_Day"        "Release_year"       "OscarWon"
## [31] "OtherWin"           "OscarNom"           "OtherNom"
```

Grammar of graphics: other geometrical objects for continuous variable

Create basic histogram

```
ggplot(movies, aes(x=imdbRating))+geom_histogram()
```

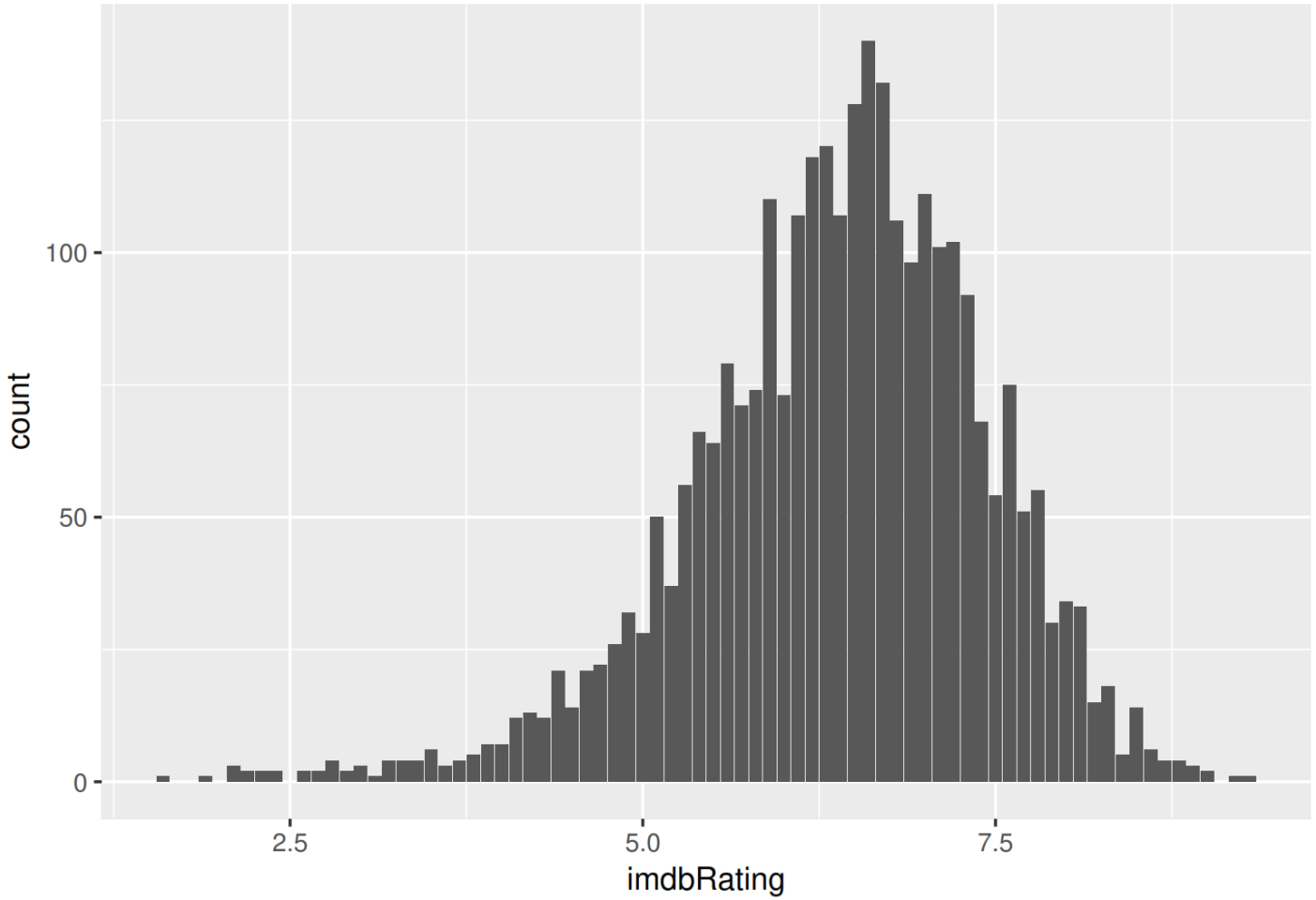


Grammar of graphics

DATAMOTUS

- You can get more or less the same with geom_bar
- With geom_histogram you can control for number of bins or binwidth

```
ggplot(movies, aes(x=imdbRating))+geom_bar()
```

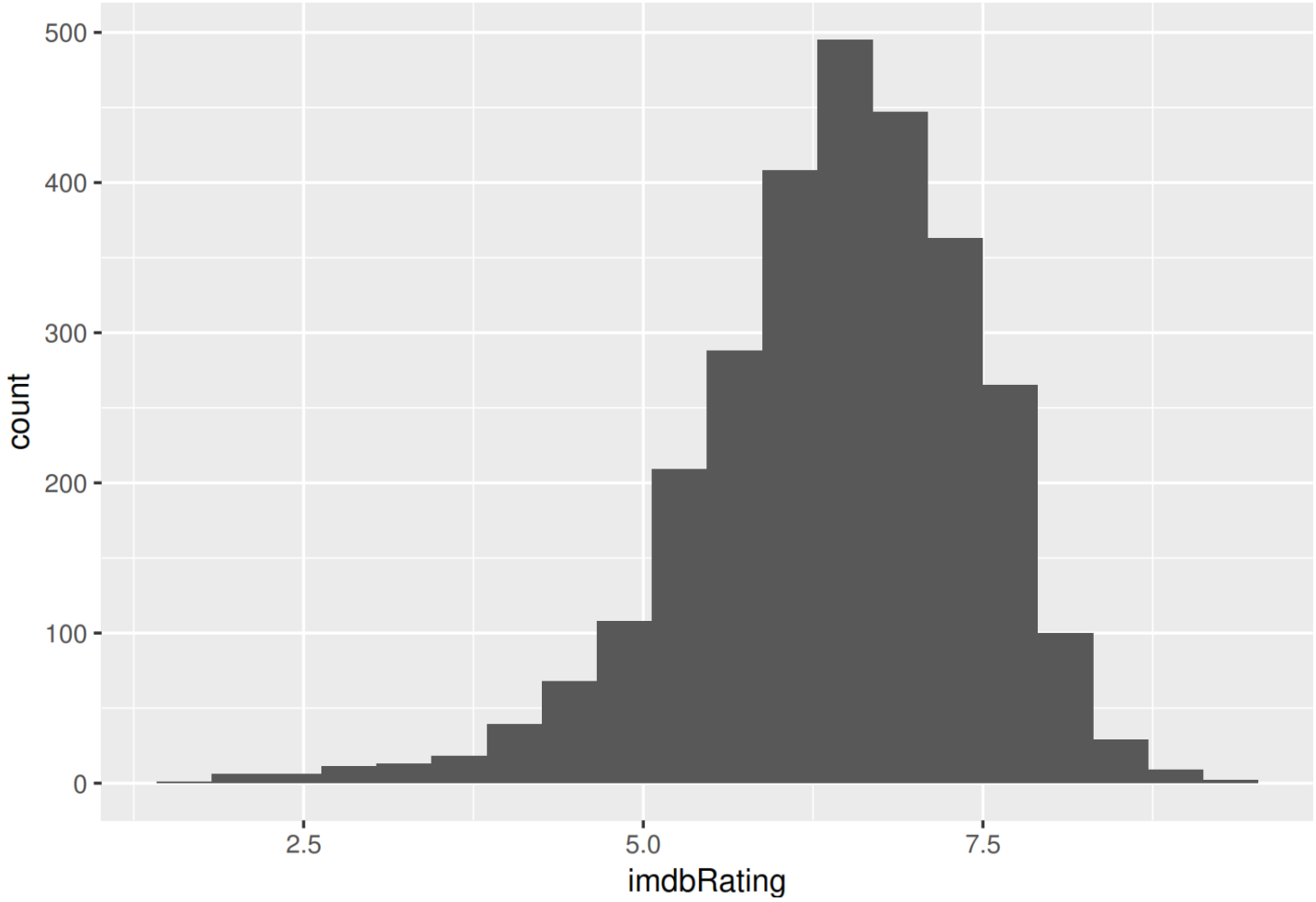


Grammar of graphics

DATAMOTUS

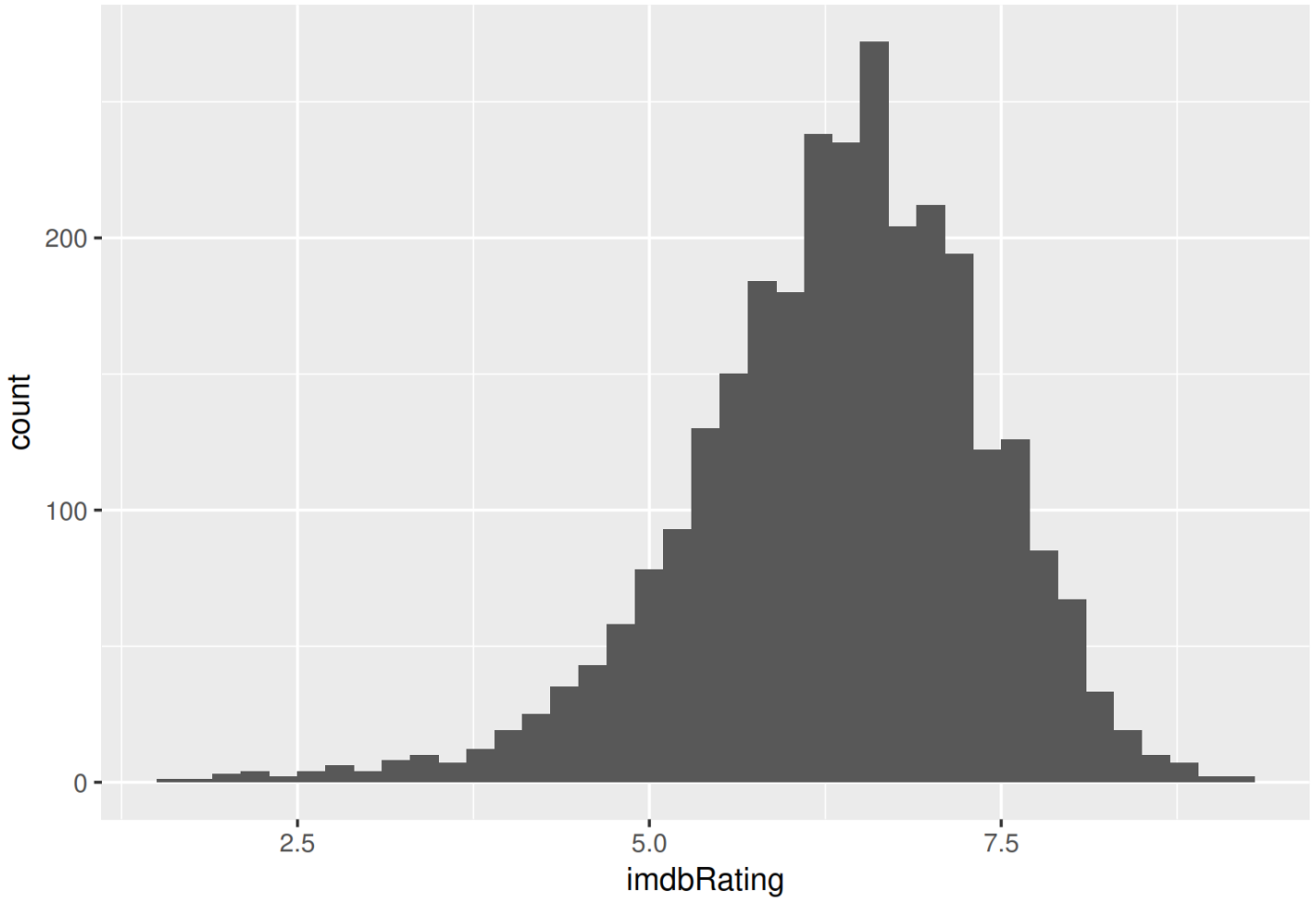
Play with the number of bins to see how the chart is changing

```
ggplot(movies, aes(x=imdbRating))+geom_histogram(bins = 20)
```



Or play with the ***binwidth***

```
ggplot(movies, aes(x=imdbRating))+geom_histogram(binwidth = 0.2)
```



Visualizing categorical data

• Grammar of graphics: Visualizing categorical data

DATAMOTUS

- A **bar chart** or **bar graph** is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

- A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value. Some bar graphs present bars clustered in groups of more than one, showing the values of more than one measured variable.

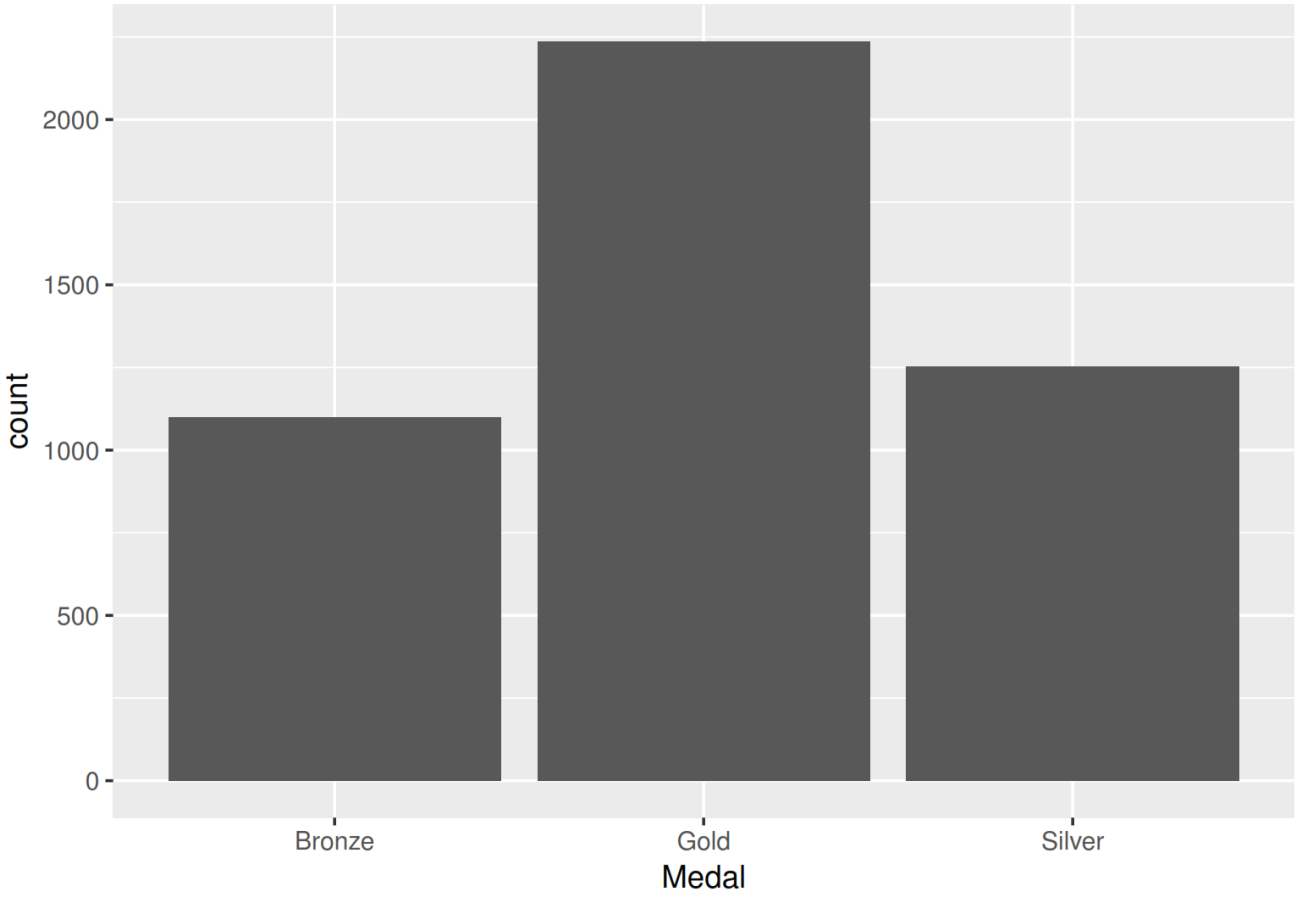
• Grammar of graphics: Visualizing categorical data

DATAMOTUS

Bar chart for USA medals from Summer Olympic games

```
summer <- read.csv("summer.csv")
summer_usa <- summer[summer$Country=="USA",]
```

```
ggplot(summer_usa, aes(x=Medal))+geom_bar()
```



• Grammar of graphics: Visualizing categorical data

DATAMOTUS

- Lets make the chart more visually appealing
- First relevel the factor levels for medals

```
levels(summer_usa$Medal)
```

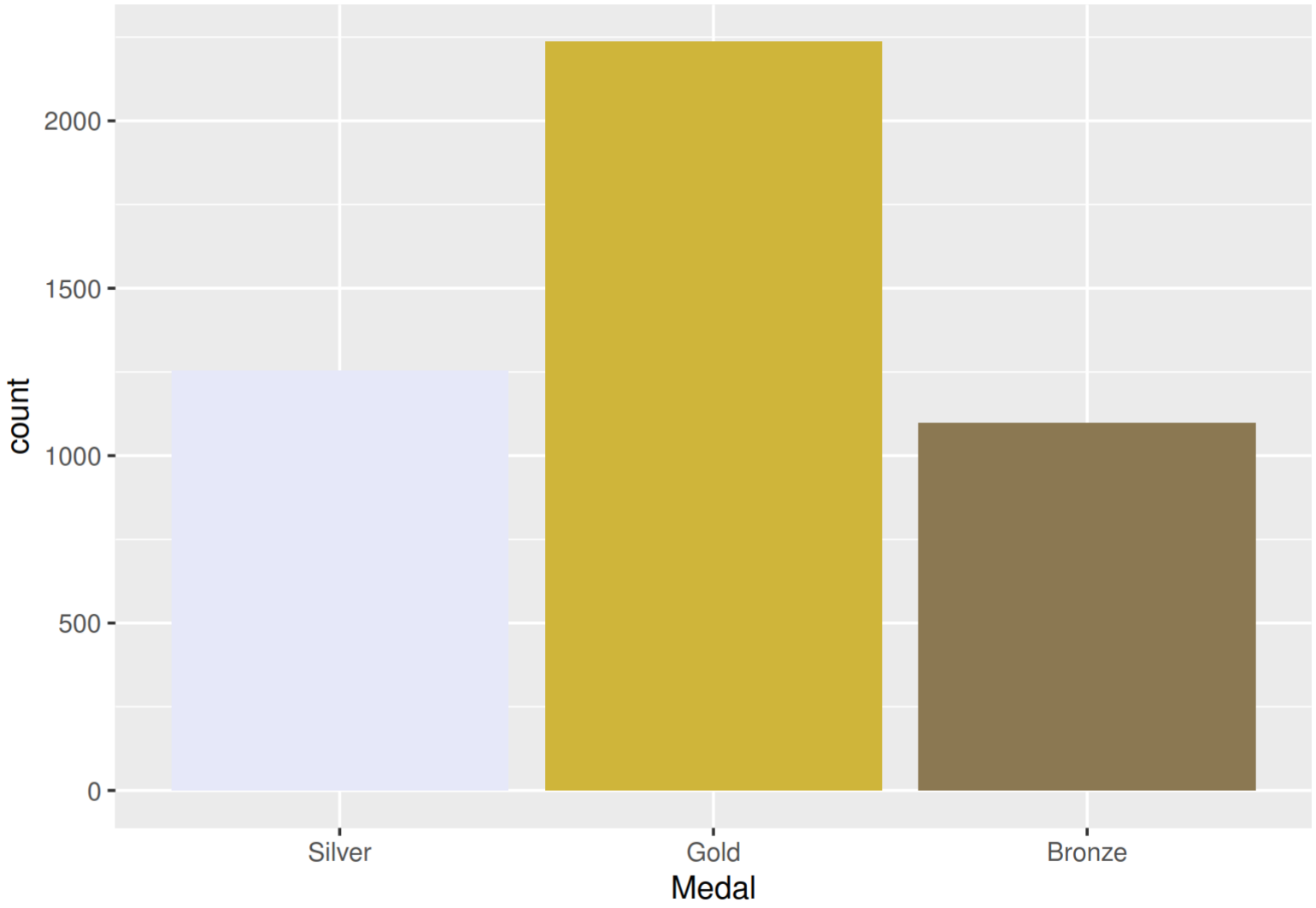
```
## [1] "Bronze" "Gold"   "Silver"
```

```
summer_usa$Medal <- factor(summer_usa$Medal, levels=c("Silver", "Gold","Bronze"))
levels(summer_usa$Medal)
```

- Grammar of graphics: Visualizing categorical data

DATAMOTUS

```
ggplot(summer_usa, aes(x=Medal))+geom_bar(fill = c("#E6E8FA", "#CFB53B", "#8C7853"))
```



• Grammar of graphics: Visualizing categorical data

DATAMOTUS

- Build already aggregated data frame with relative frequencies (%es) for each type of medal
- Use ggplot to make the barplot

```
usa_medals <- data.frame(Medal=c("Silver", "Gold", "Bronze"),
                         Percentage = c(0.27,0.49, 0.24))
```

You are going to get the following error

```
> ggplot(usa_medals, aes(x=Medal, y=Percentage))+geom_bar()
Error: stat_count() must not be used with a y aesthetic.
```

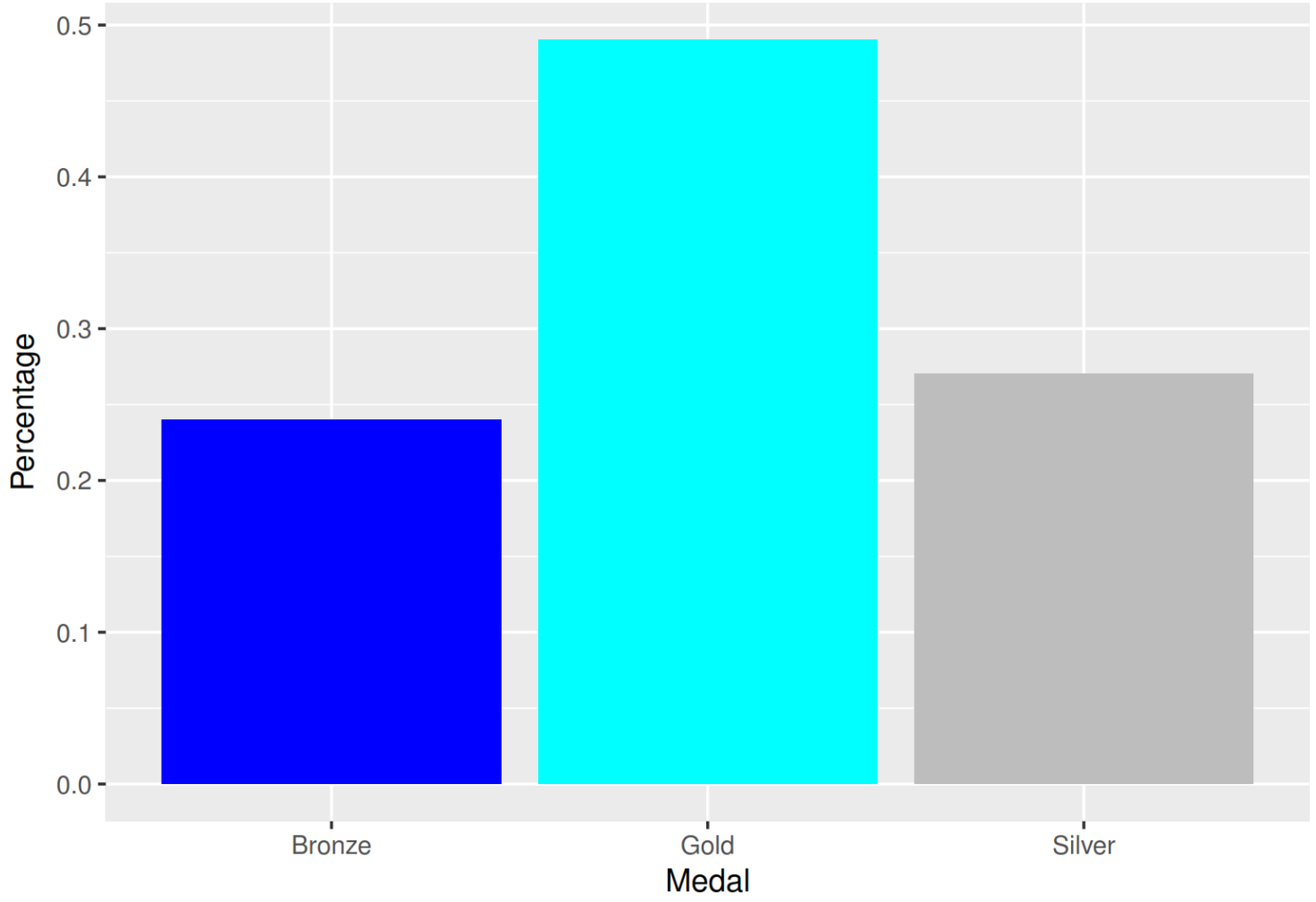
- Another component of grammas of graphics is the ***statistical transformation***
- For geom_bar by default it is stat_count() as we are constructing a frequency plot

• Grammar of graphics: Visualizing categorical data

DATAMOTUS

- To solve the issue, use stat="identity"
- identity function: $f(x) = x$

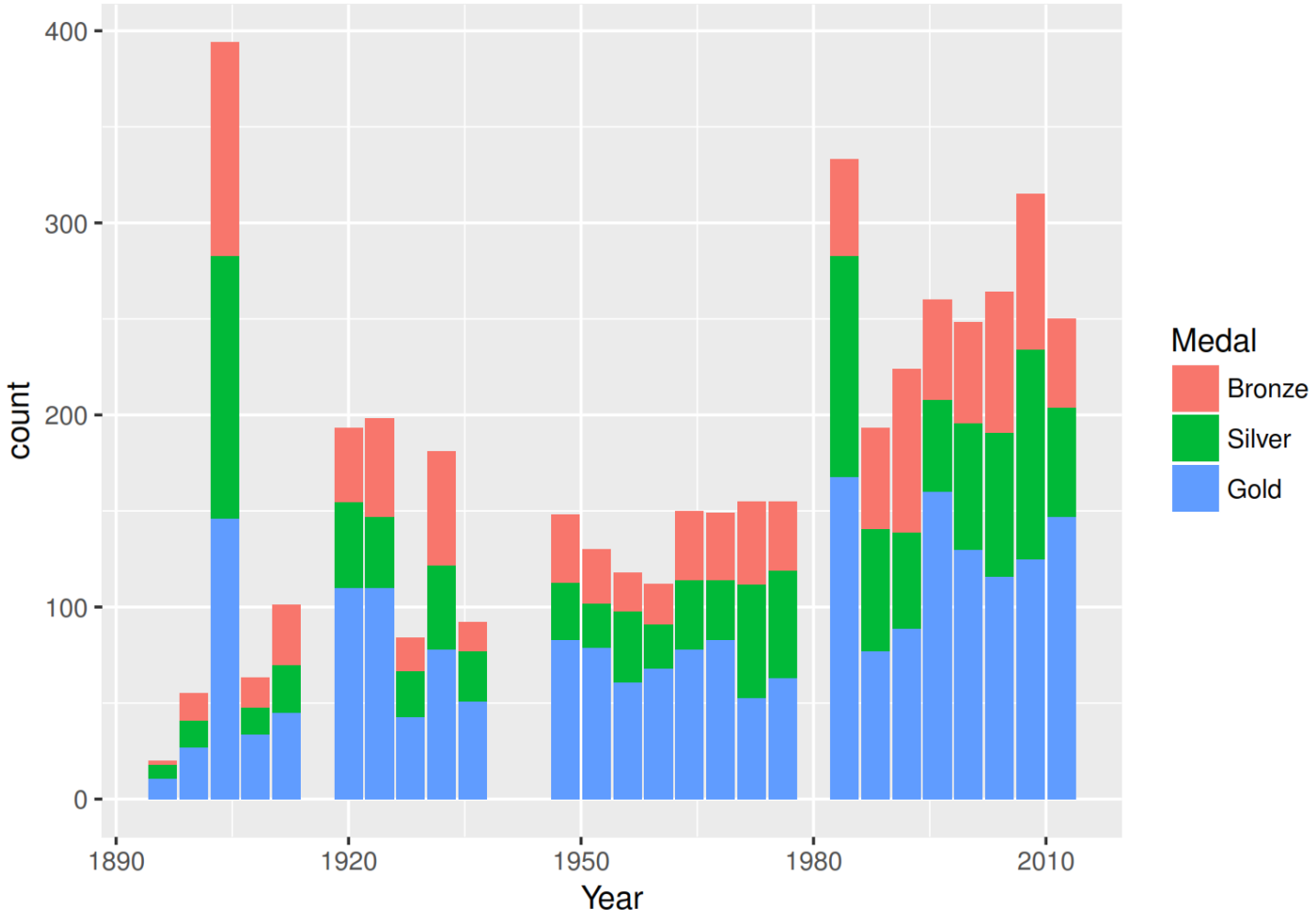
```
ggplot(usa_medals, aes(x=Medal, y=Percentage))+geom_bar(stat="identity", fill=c(4,5,8))
```



• Grammar of graphics: Visualizing categorical data

What if we want to see how the distribution of medals for USA has changed over time? You need the following aesthetics: Year on x axis, fill by medal

```
summer_usa$Medal <- factor(summer_usa$Medal, levels=c("Bronze", "Silver", "Gold"))
ggplot(summer_usa, aes(x=Year, fill=Medal))+geom_bar()
```

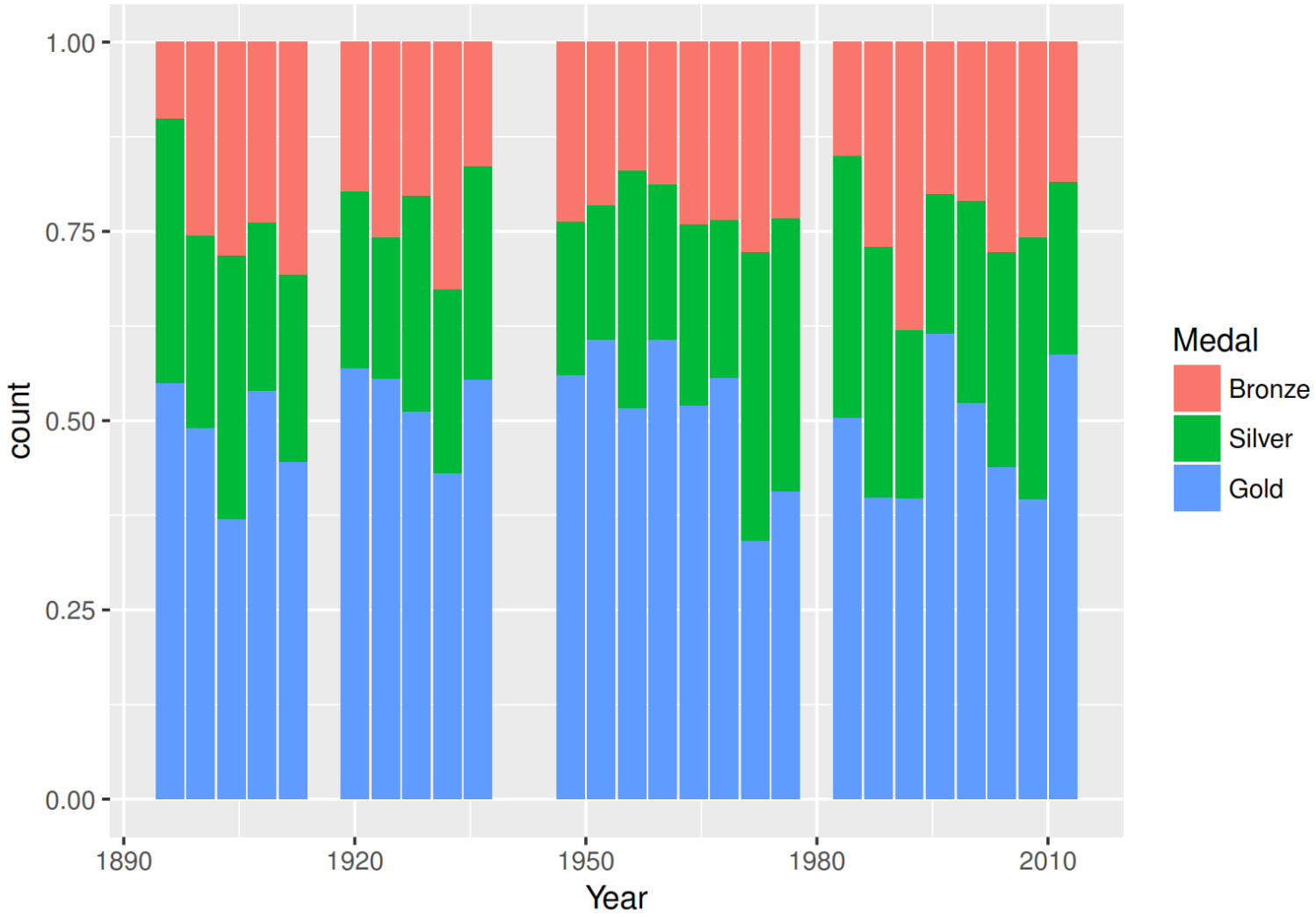


- Grammar of graphics: Visualizing categorical data

DATAMOTUS

Change position fill to get 100% bar chart

```
ggplot(summer_usa, aes(x=Year, fill=Medal))+geom_bar(position="fill")
```

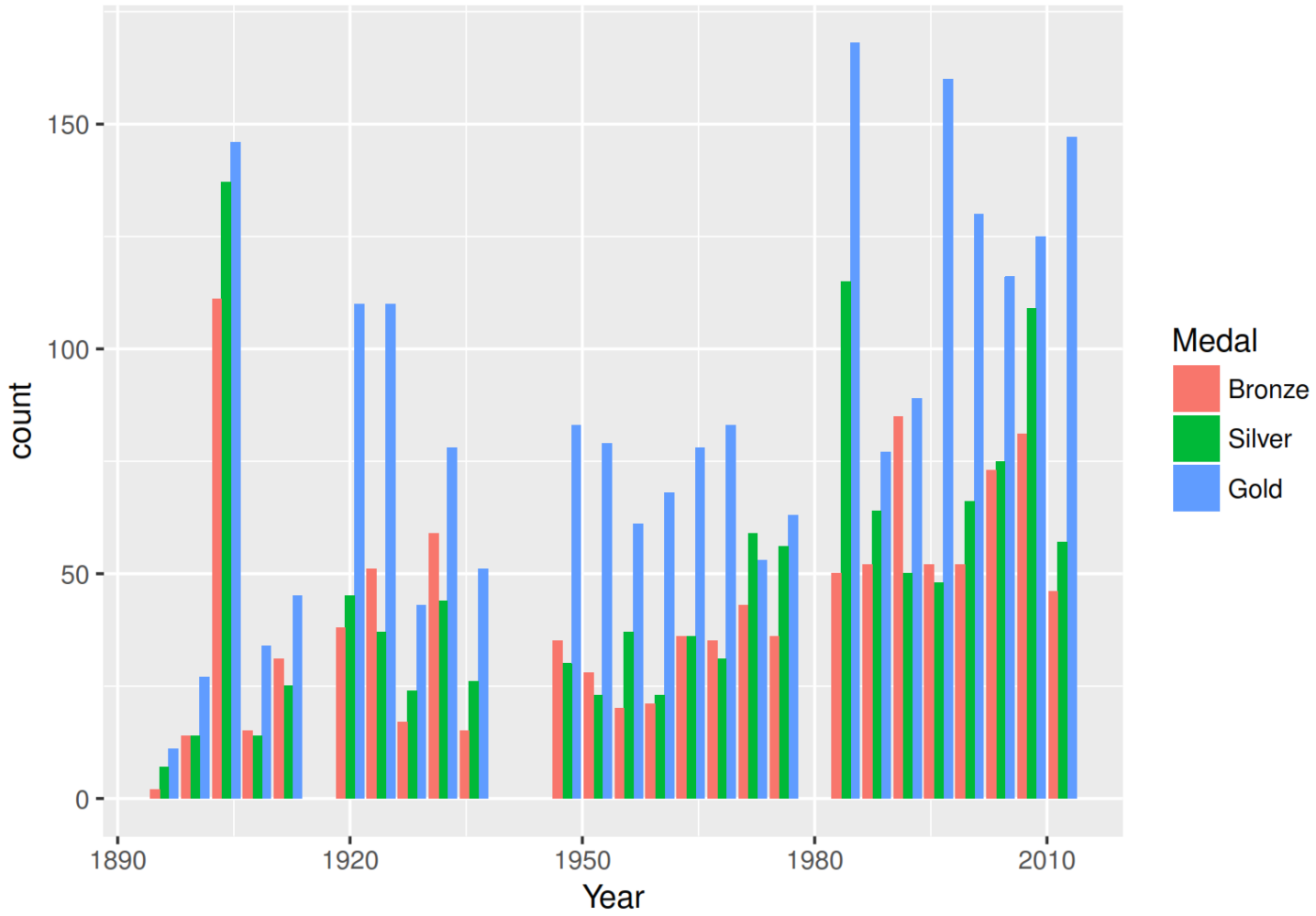


- Grammar of graphics: Visualizing categorical data

DATAMOTUS

Position dodge to get side-by-side bar chart

```
ggplot(summer_usa, aes(x=Year, fill=Medal))+geom_bar(position="dodge")
```



• Grammar of graphics: Visualizing categorical data – coordinate system

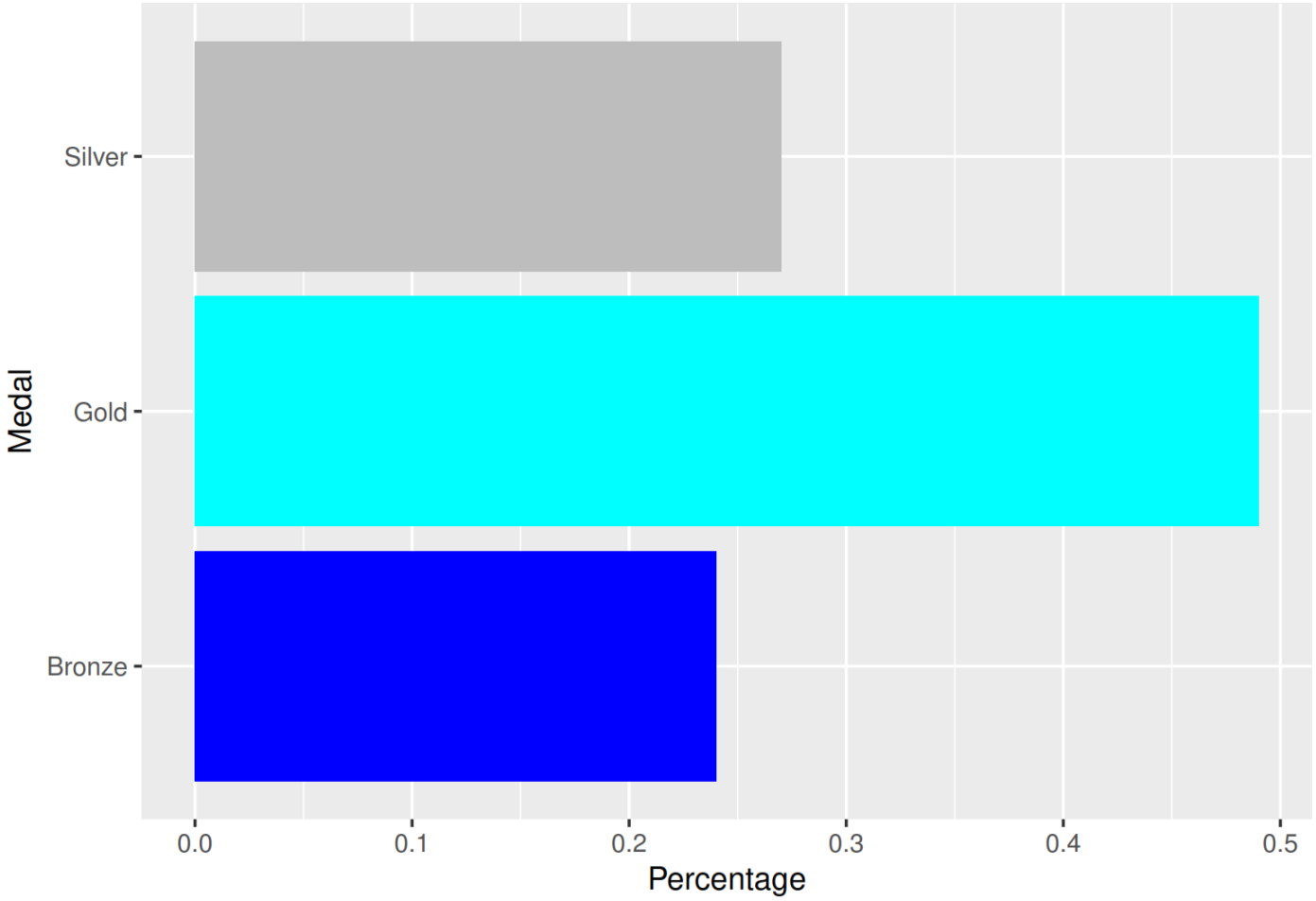
- Coordinate systems tie together the two position scales to produce a 2d location. Currently, ggplot2 comes with different coordinate systems.
- All these coordinate systems are two dimensional.
- As with the other components in ggplot2, you generate the R name by joining **coord_** and the name of the coordinate system.
- Most plots use the default Cartesian coordinate system, coord_cartesian(), where the 2d position of an element is given by the combination of the x and y positions.

• Grammar of graphics: Visualizing categorical data

DATAMOTUS

- **coord_flip** will flip the x and y coordinates
- You can add it as a layer to a barplot created before to get a vertical bar plot with categories on y axis and values on x axis

```
ggplot(usa_medals, aes(x=Medal, y=Percentage))+geom_bar(stat="identity", fill=c(4,5,8))+
  coord_flip()
```

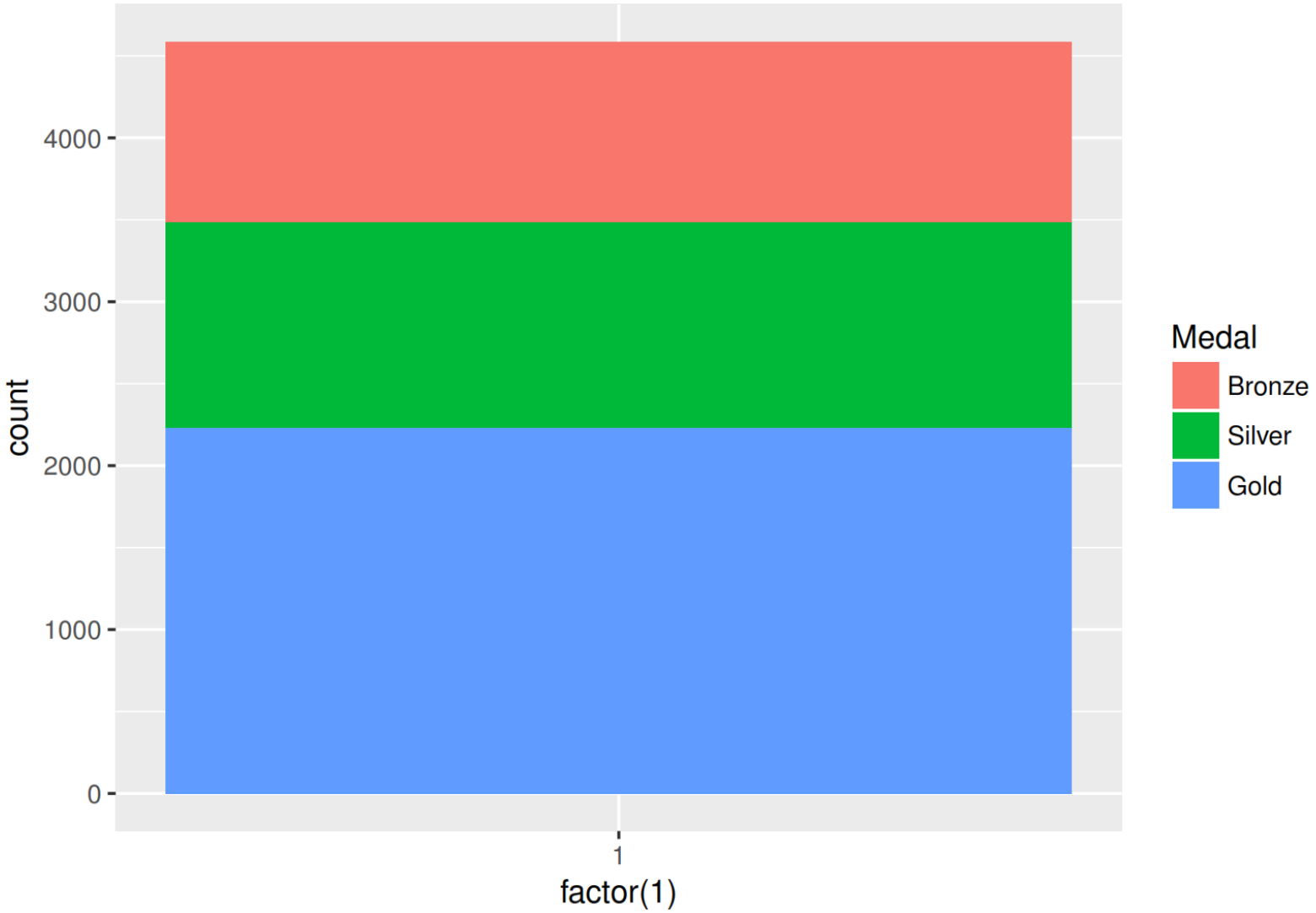


- Grammar of graphics: Visualizing categorical data

DATAMOTUS

Stacked bar chart where for x variable you use 1 value (you can put anything in factor())

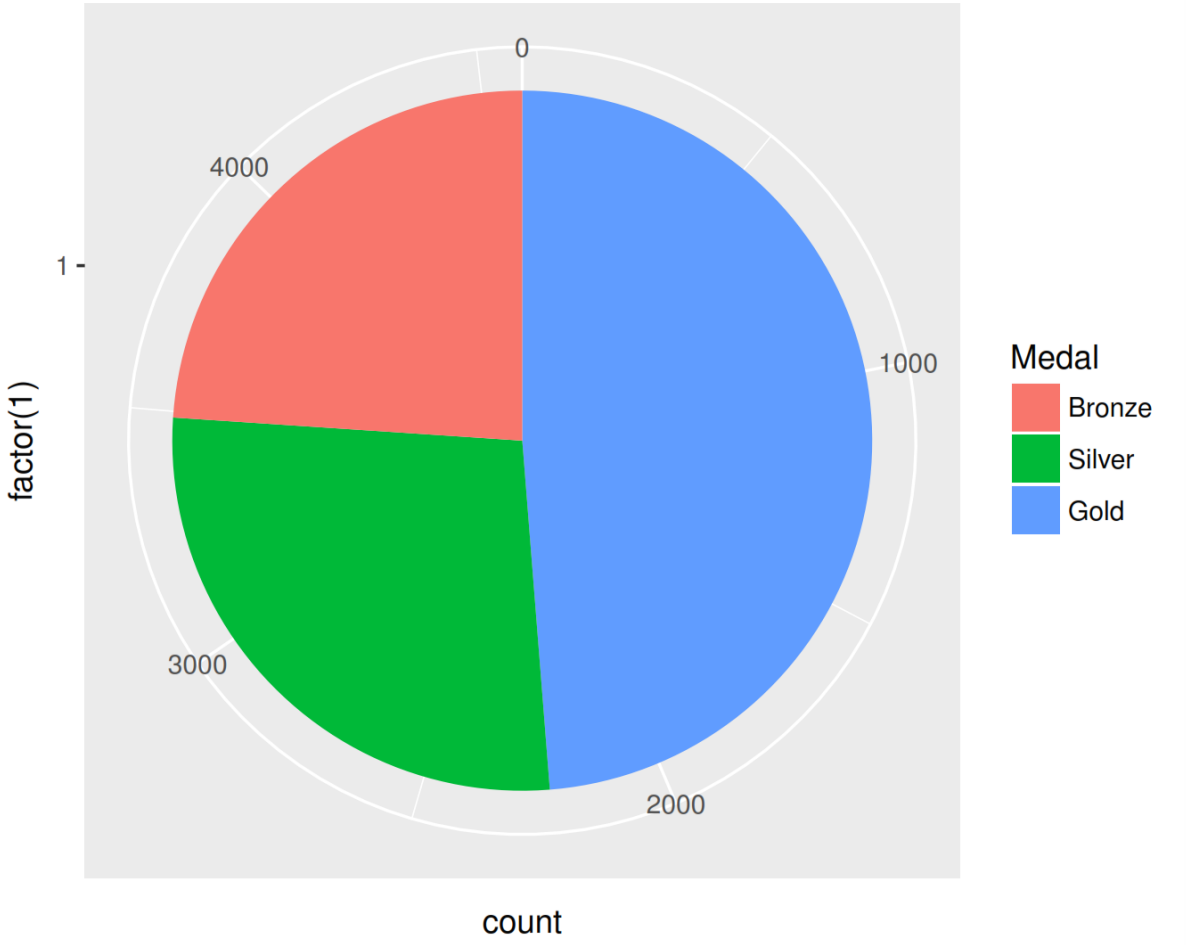
```
ggplot(summer_usa, aes(x=factor(1),fill=Medal))+geom_bar()
```



- Grammar of graphics: Visualizing categorical data

DATAMOTUS

```
ggplot(summer_usa, aes(x=factor(1), fill=Medal))+geom_bar(width=1)+
  coord_polar(theta="y")
```

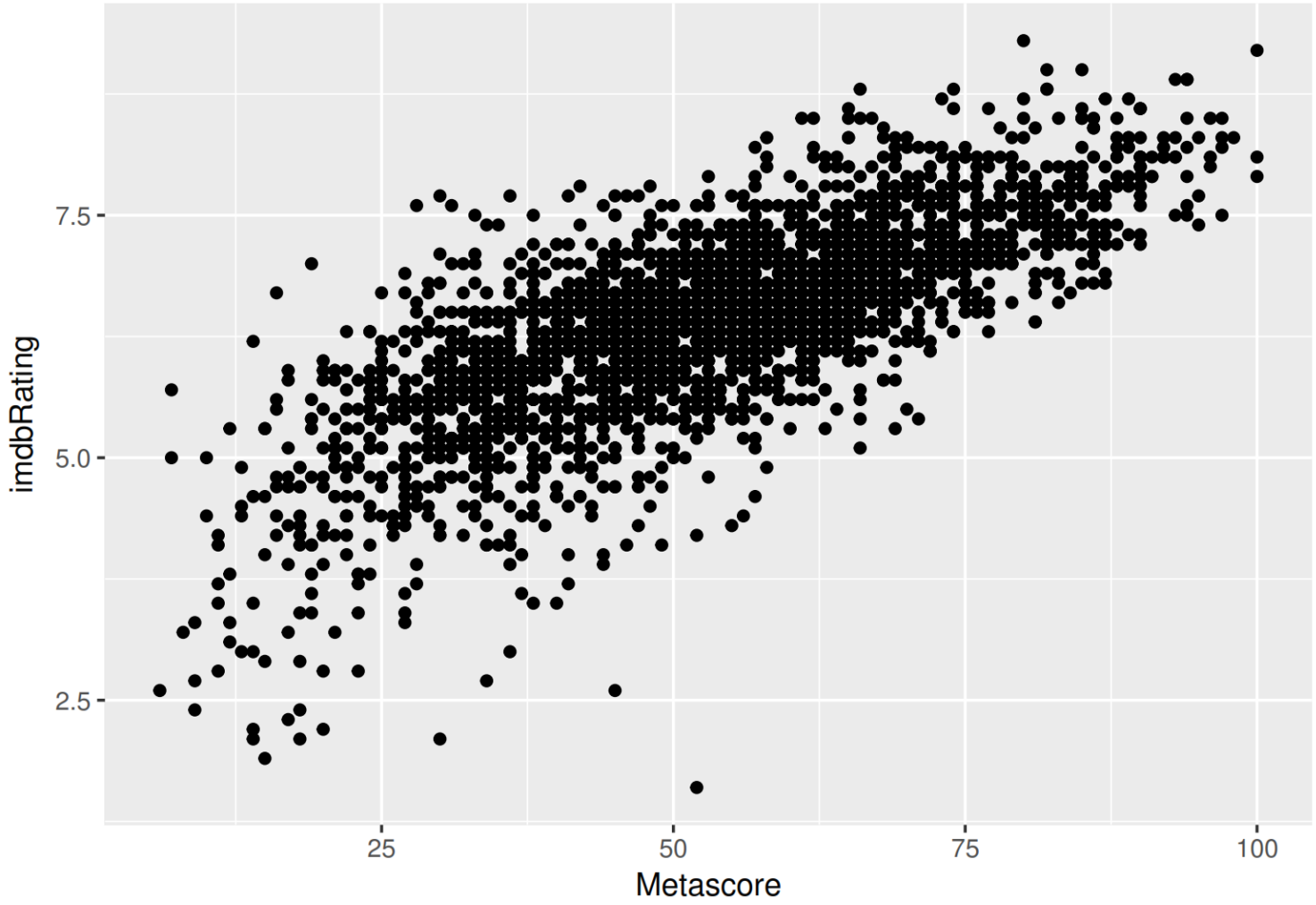


- Grammar of graphics: Using coordinates on continuous data

DATAMOTUS

Metascore vs imdbRating

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()
```

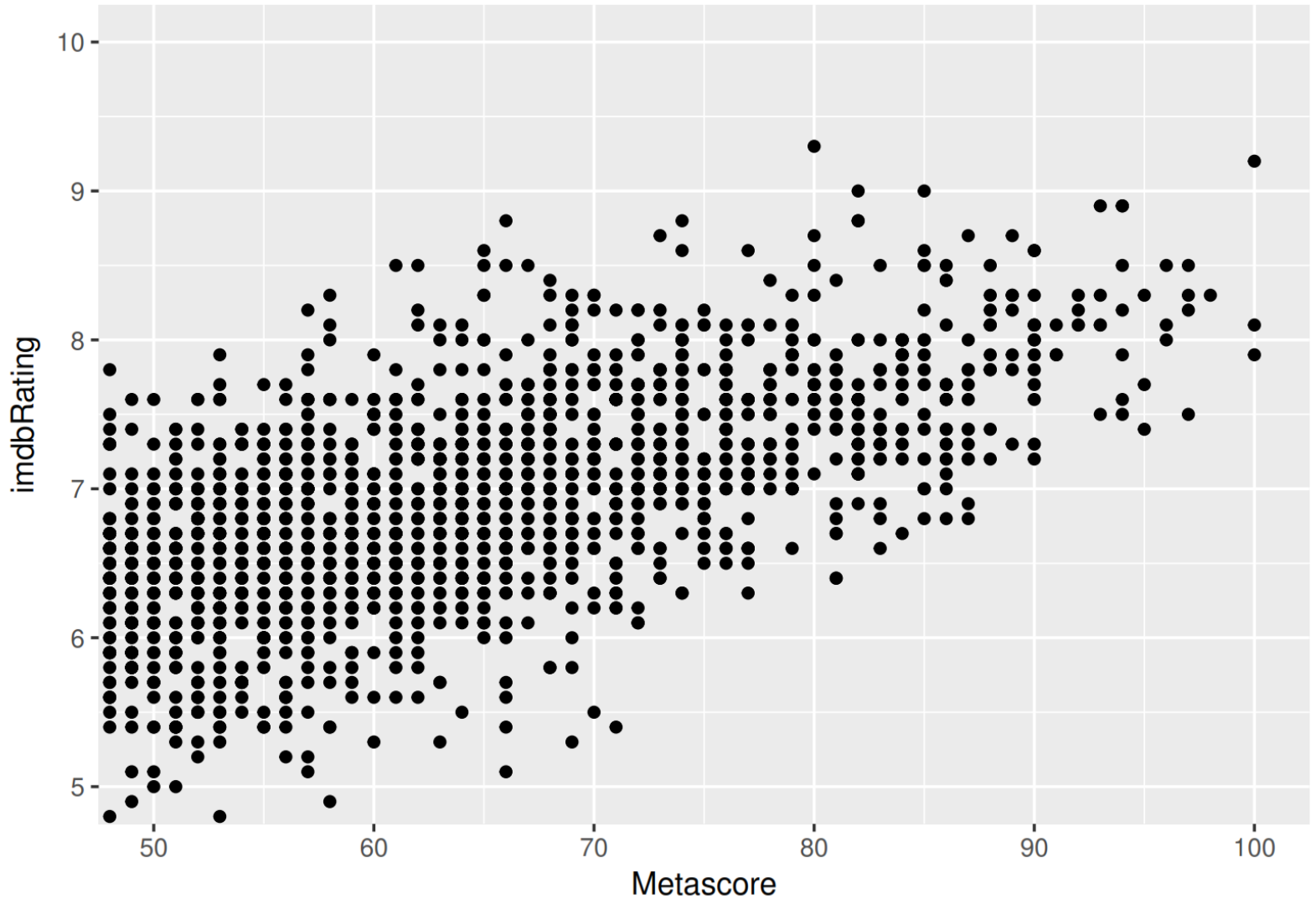


- Grammar of graphics: Using coordinates on continuous data

DATAMOTUS

We want to limit x and y axes, thus zoom in the graph

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  coord_cartesian(ylim=c(5,10), xlim=c(50,100))
```

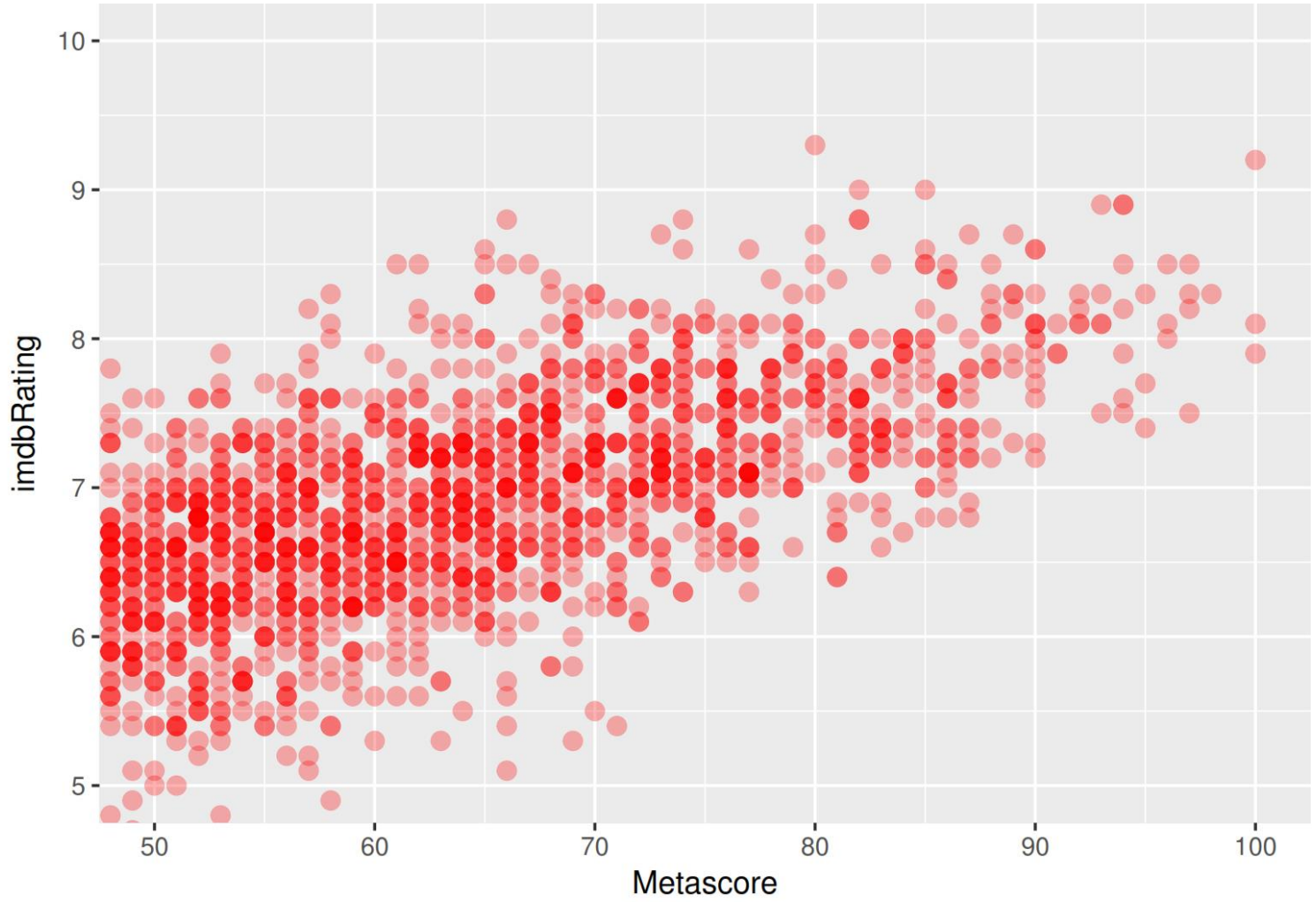


- Grammar of graphics: Using coordinates on continuous data

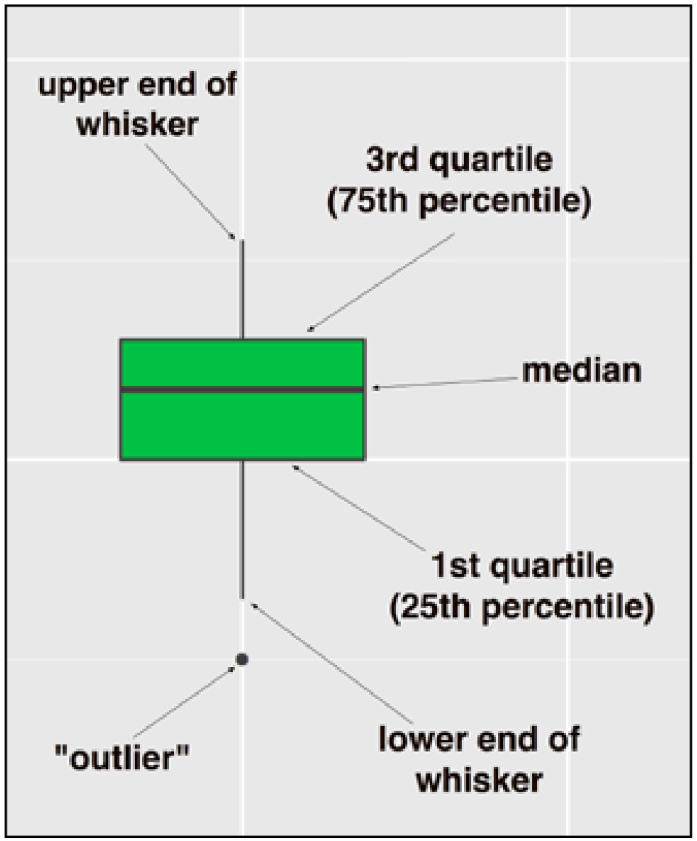
DATAMOTUS

If the sample size is big, then the points are overlapped which distorts the real picture.
Use alpha for point transparency

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point(alpha=0.3, col="red", size=2.7)+
  coord_cartesian(ylim=c(5,10), xlim=c(50,100))
```



- Grammar of graphics: plotting categorical and continuous data together



The length of the whisker = 1.58*IQR

IQR-Inter Quartile range = 3rd Quartile-1st Quartile

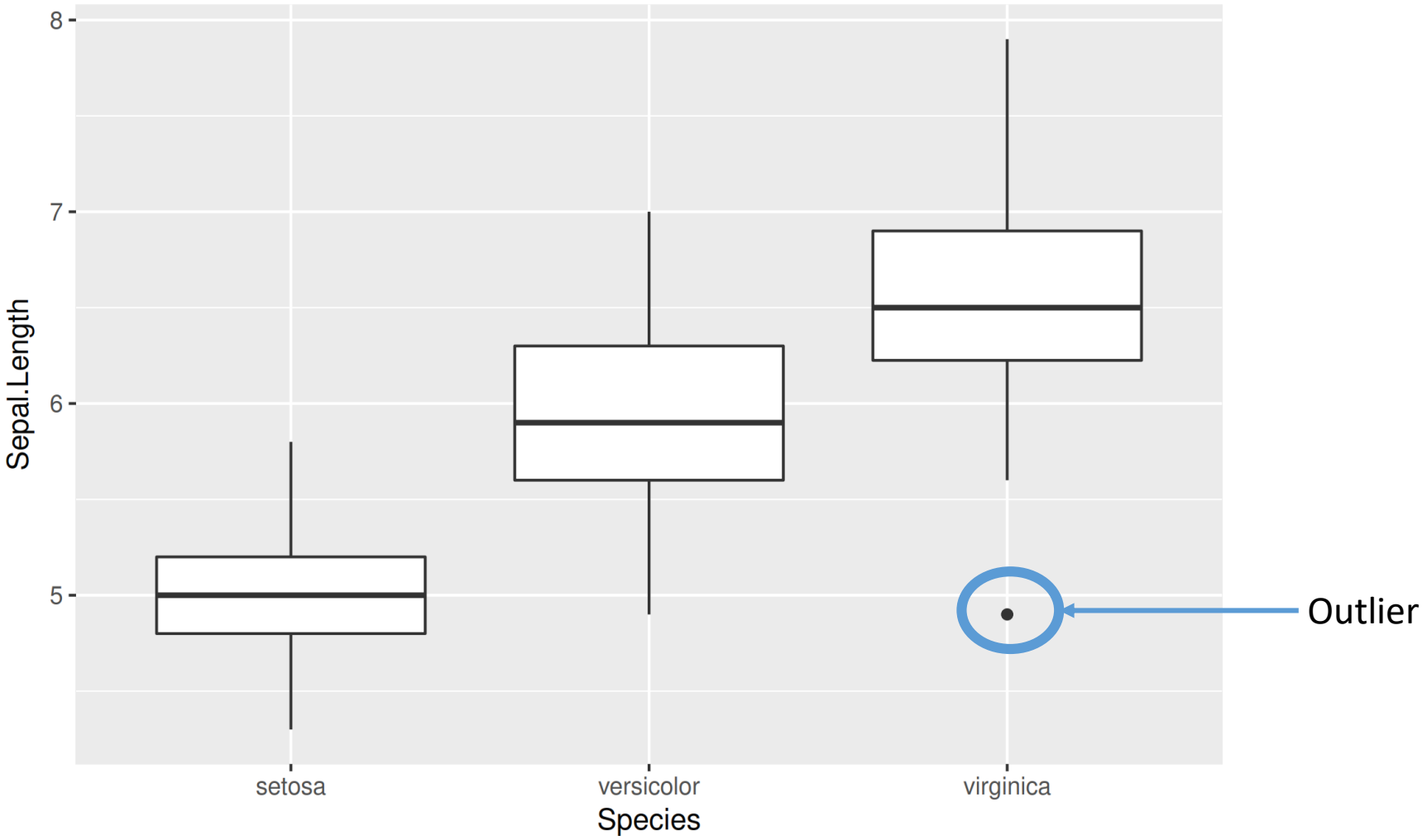
- Grammar of graphics: plotting categorical and continuous data together

DATAMOTUS

Symbol	Names	Definition
Q_1	**first quartile** **lower quartile** **25th percentile**	splits off the lowest 25% of data from the highest 75%
Q_2	**second quartile** **median** **50th percentile**	cuts data set in half
Q_3	**third quartile** **upper quartile** **75th percentile**	splits off the highest 25% of data from the lowest 75%

- Grammar of graphics: plotting categorical and continuous data together

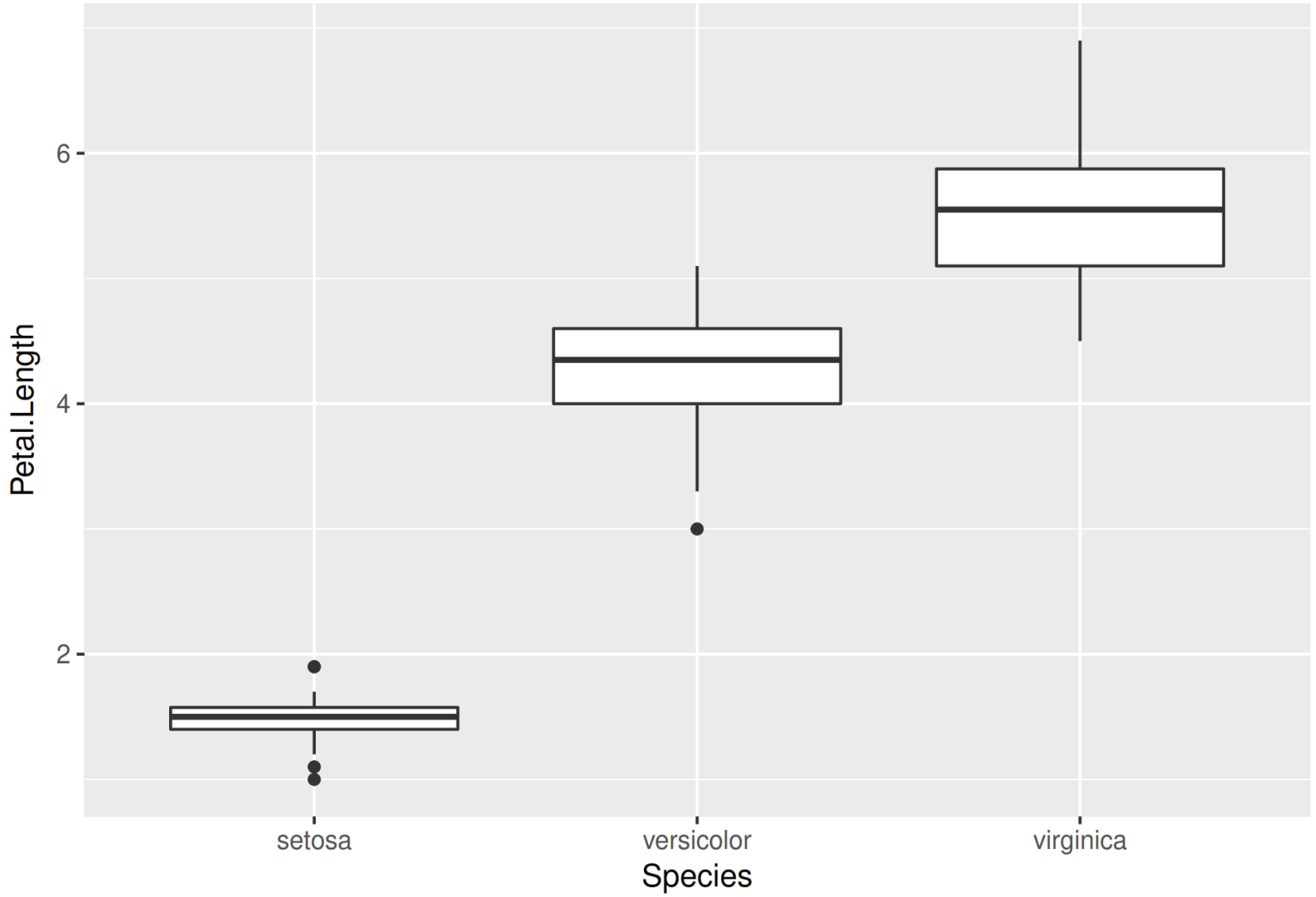

```
ggplot(iris, aes(x=Species, y=Sepal.Length))+geom_boxplot()
```



- Grammar of graphics: plotting categorical and continuous data together

Boxplot for Petal.Length

```
ggplot(iris, aes(x=Species, y=Petal.Length))+geom_boxplot()
```



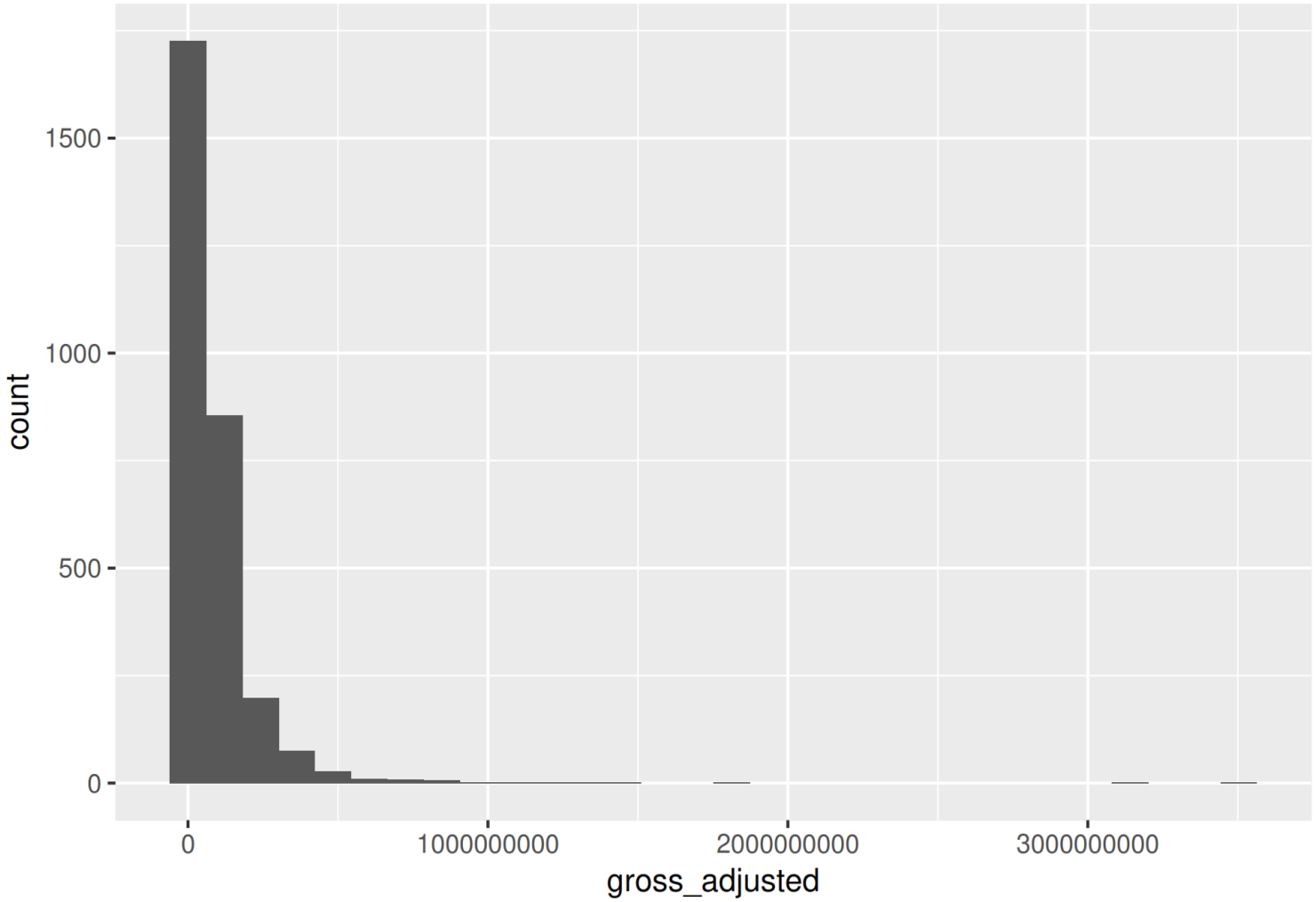
• Grammar of graphics: plotting categorical and continuous data together

- Boxplot helps to understand if the distributions by categories are different from each other.

- Also it helps to understand the shape of the distribution: whether it is symmetric or skewed

- Grammar of graphics: plotting categorical and continuous data together


```
options(scipen=999)
ggplot(movies, aes(x=gross_adjusted))+geom_histogram()
```



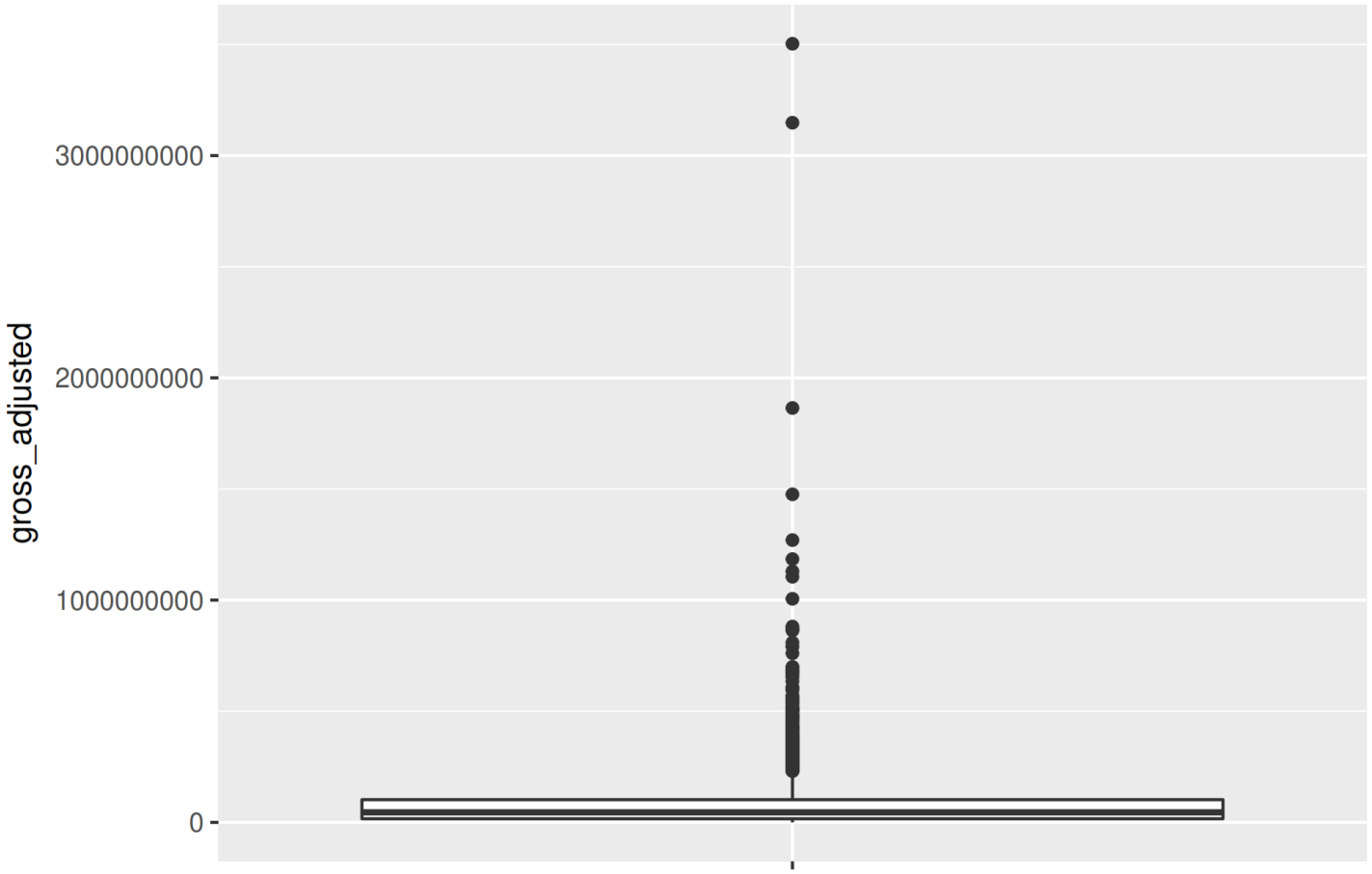
Highly skewed distribution

- Grammar of graphics: plotting categorical and continuous data together

A few outliers, few movies made a huge box office

Note: for X aesthetics we just use an empty string, as the boxplot is created for the whole column

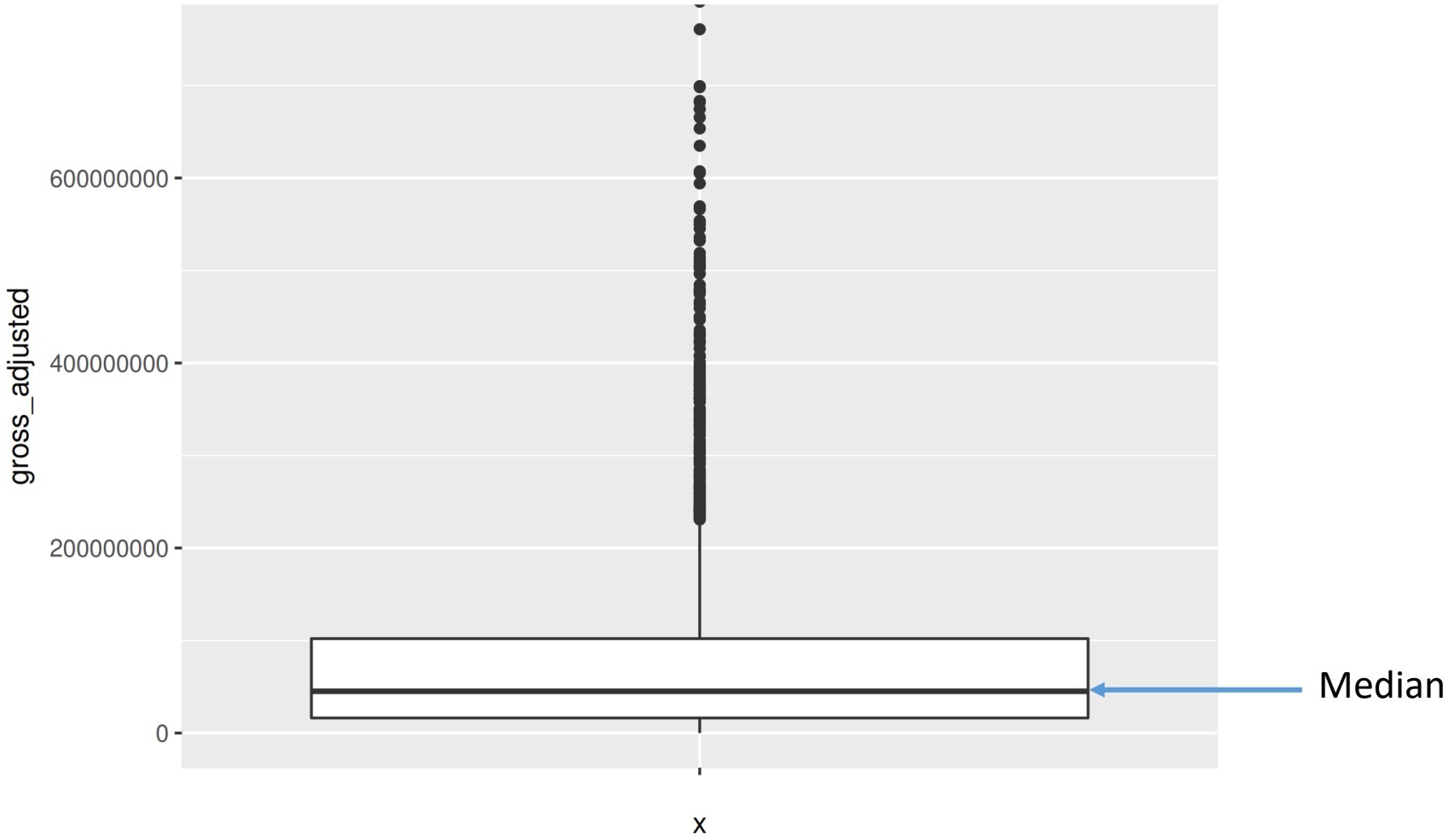
```
ggplot(movies, aes(y=gross_adjusted, x=""))+geom_boxplot()
```



- Grammar of graphics: plotting categorical and continuous data together

Zoom in: With symmetric distribution the median would be in the middle of the box

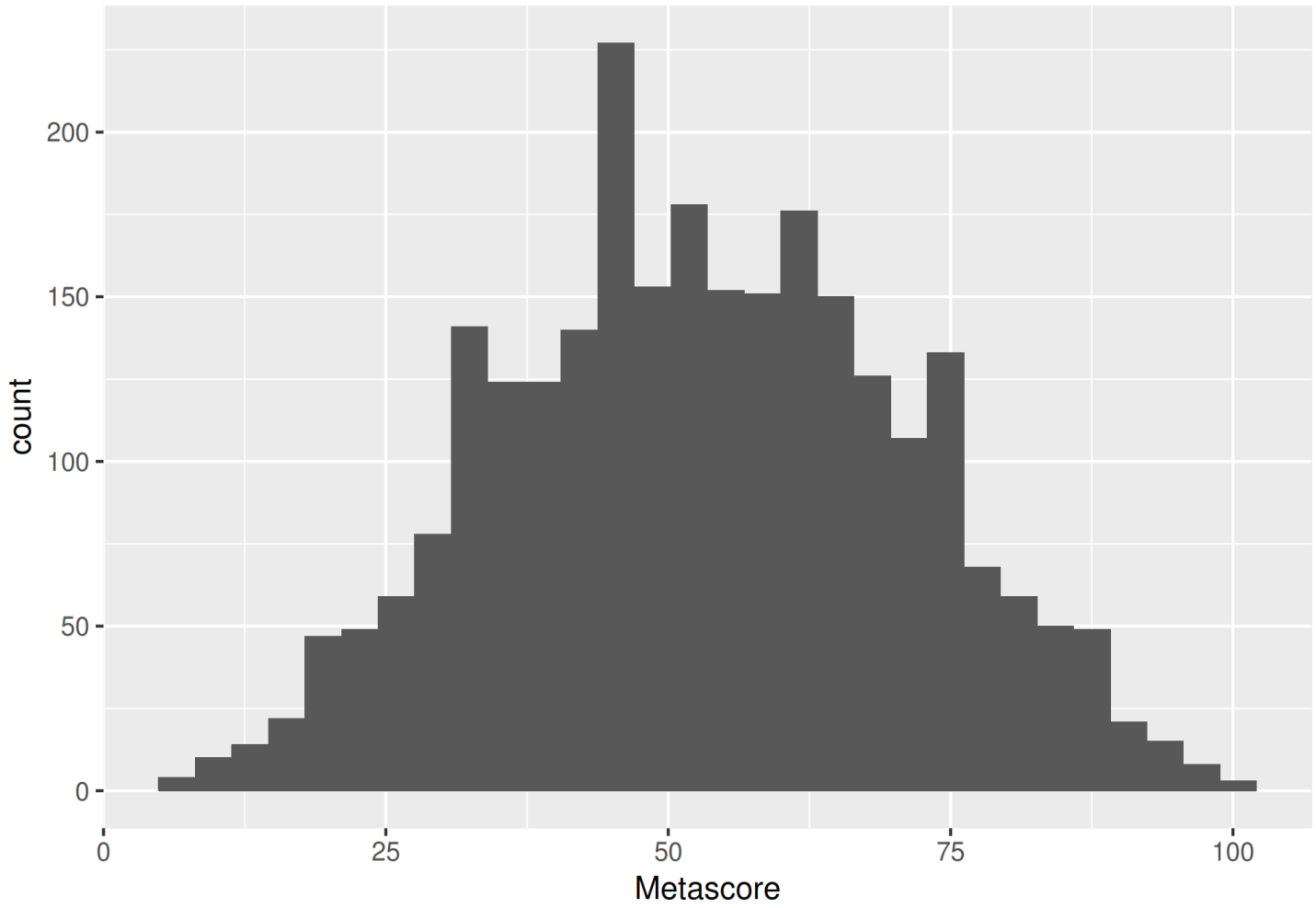
```
options(scipen=999)
ggplot(movies, aes(y=gross_adjusted, x=""))+geom_boxplot()+
  coord_cartesian(ylim=c(0,750000000))
```



- Grammar of graphics: plotting categorical and continuous data together

Look at the histogram of Metascore, almost symmetric

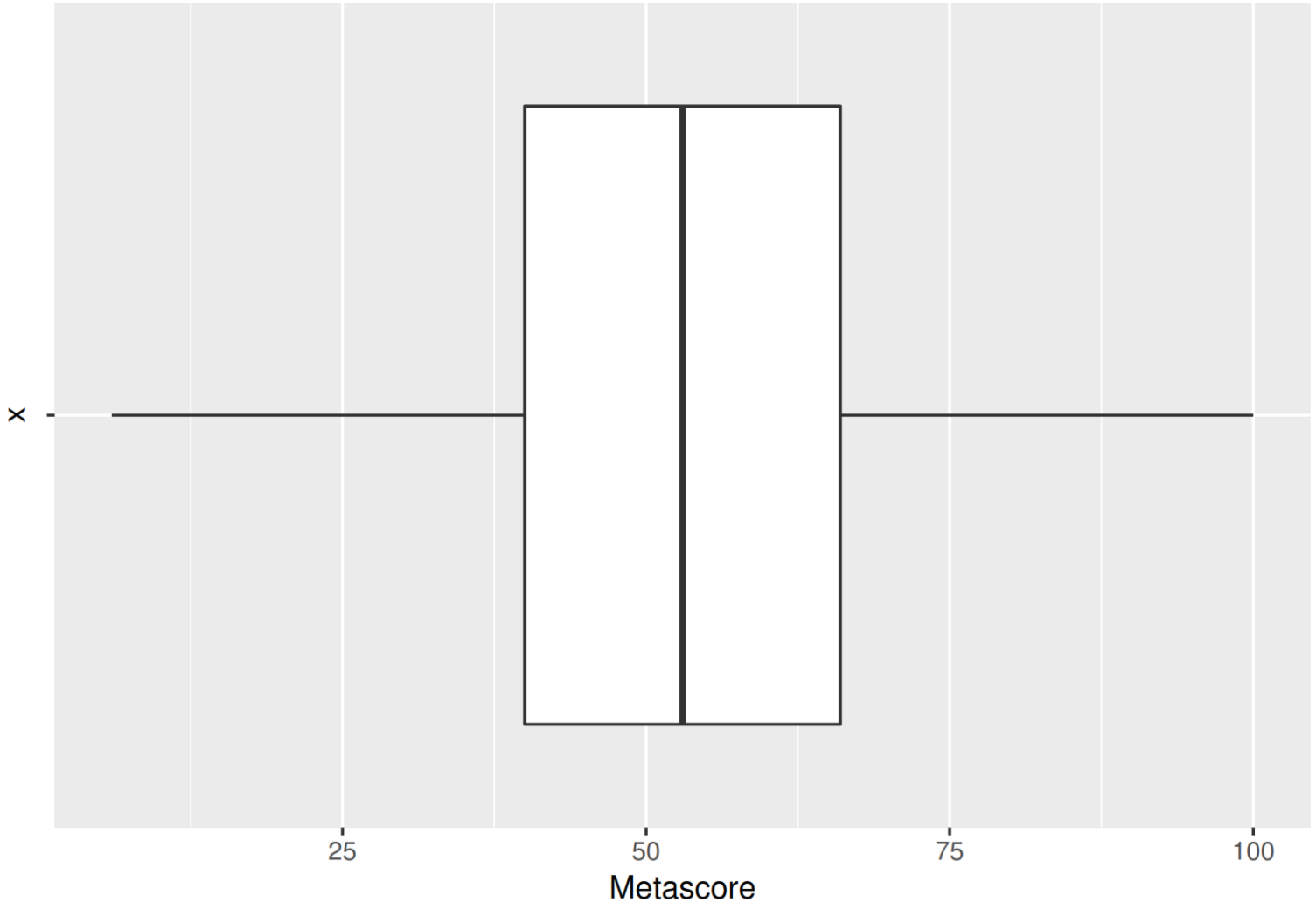
```
ggplot(movies, aes(x=Metascore))+geom_histogram()
```



- Grammar of graphics: plotting categorical and continuous data together

The boxplot with coordinates flipped

```
ggplot(movies, aes(y=Metascore, x=""))+geom_boxplot()+coord_flip()
```



• Grammar of graphics: scales

DATAMOTUS

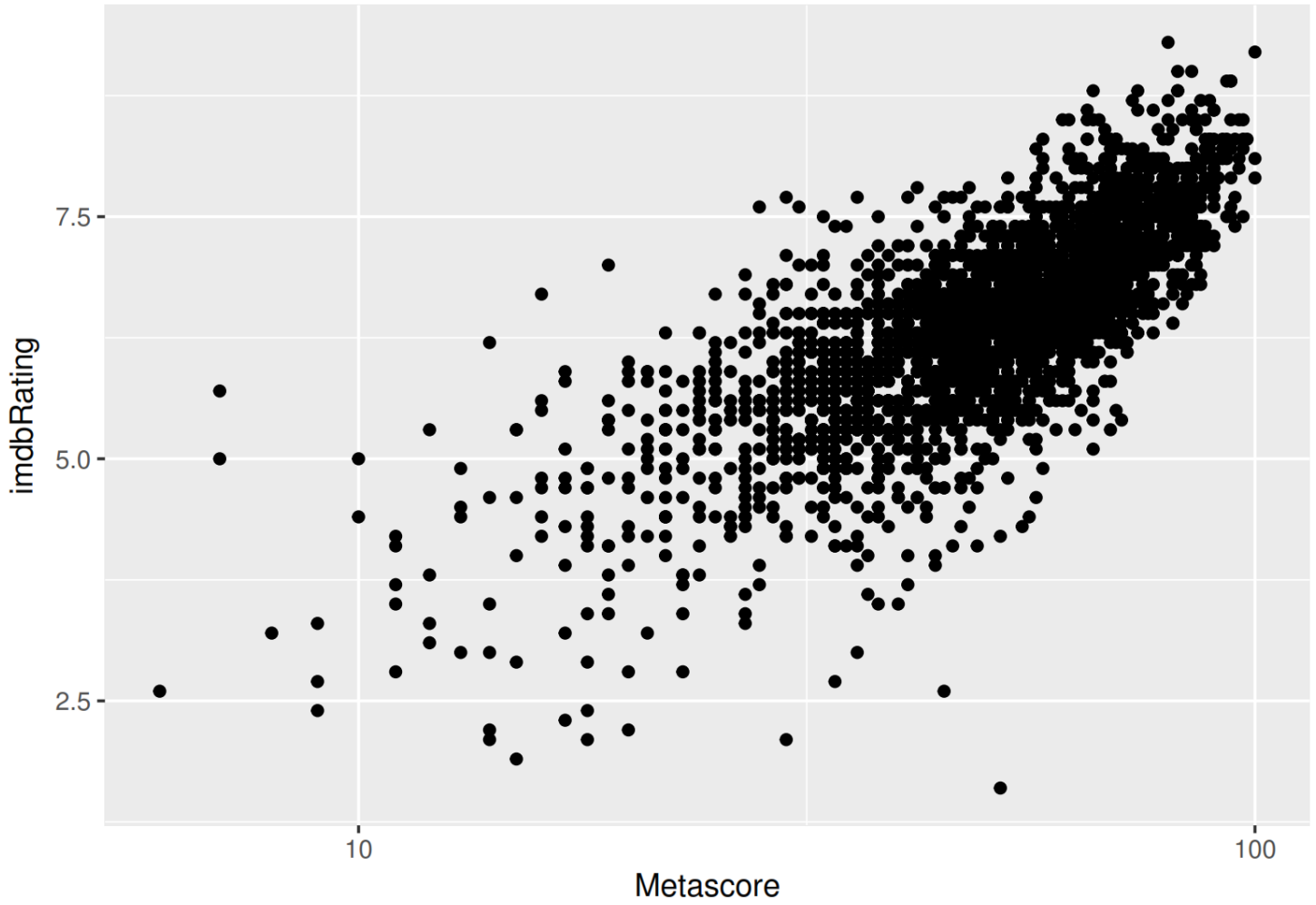
- Scales control the mapping from data to aesthetics.
- They take your data and turn it into something that you can perceive visually: e.g., size, colour, position or shape.
- Scales are also used to do mathematical transformation of the data while mapping to aesthetics.
- The common format for scale functions are scale_*_, where * is either ***x*** or ***y***

- Grammar of graphics: scales

DATAMOTUS

Log10 transformation of x axis

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  scale_x_log10()
```

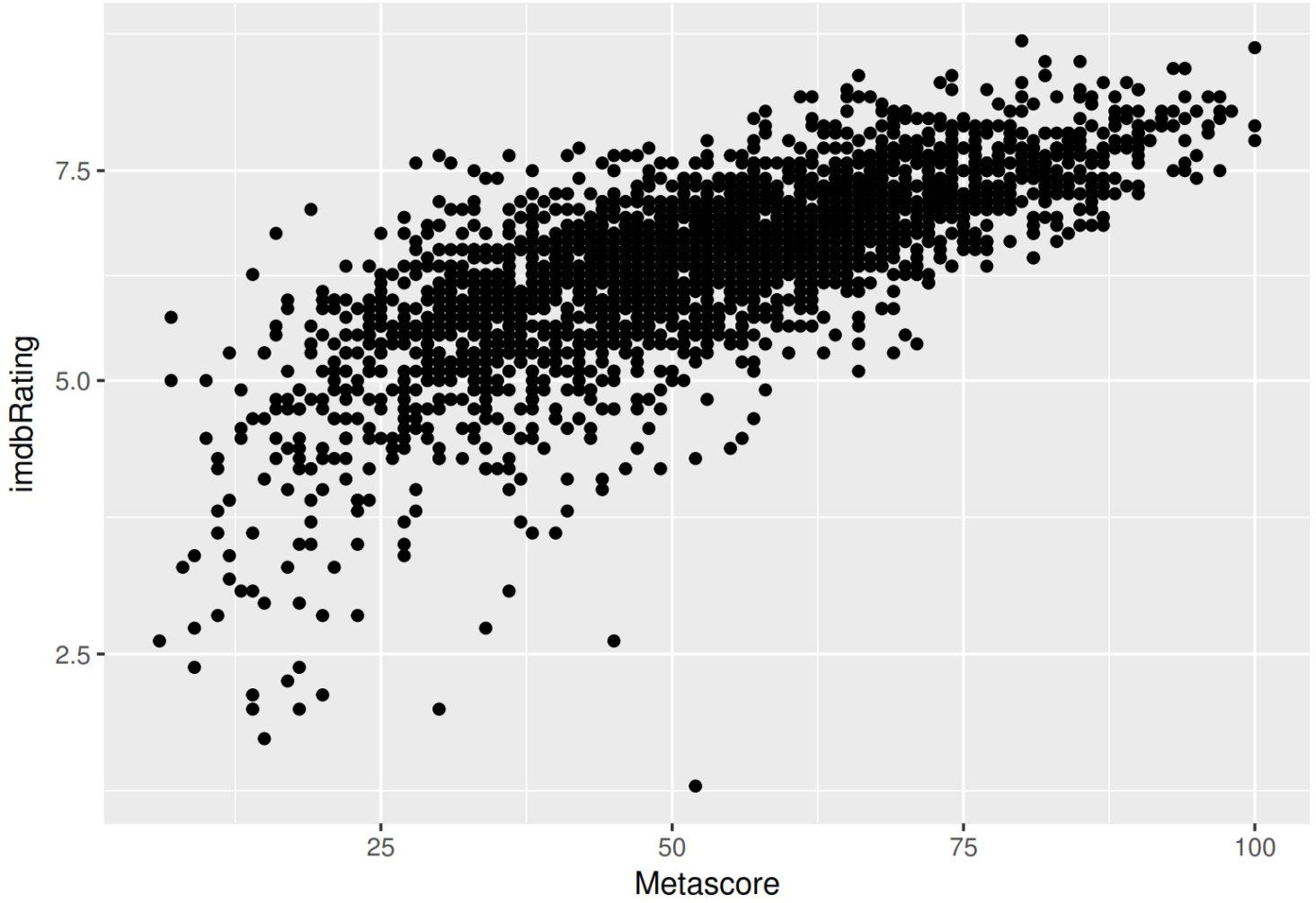


- Grammar of graphics: scales

DATAMOTUS

Square root transformation of y axis

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  scale_y_sqrt()
```



• Grammar of graphics: faceting

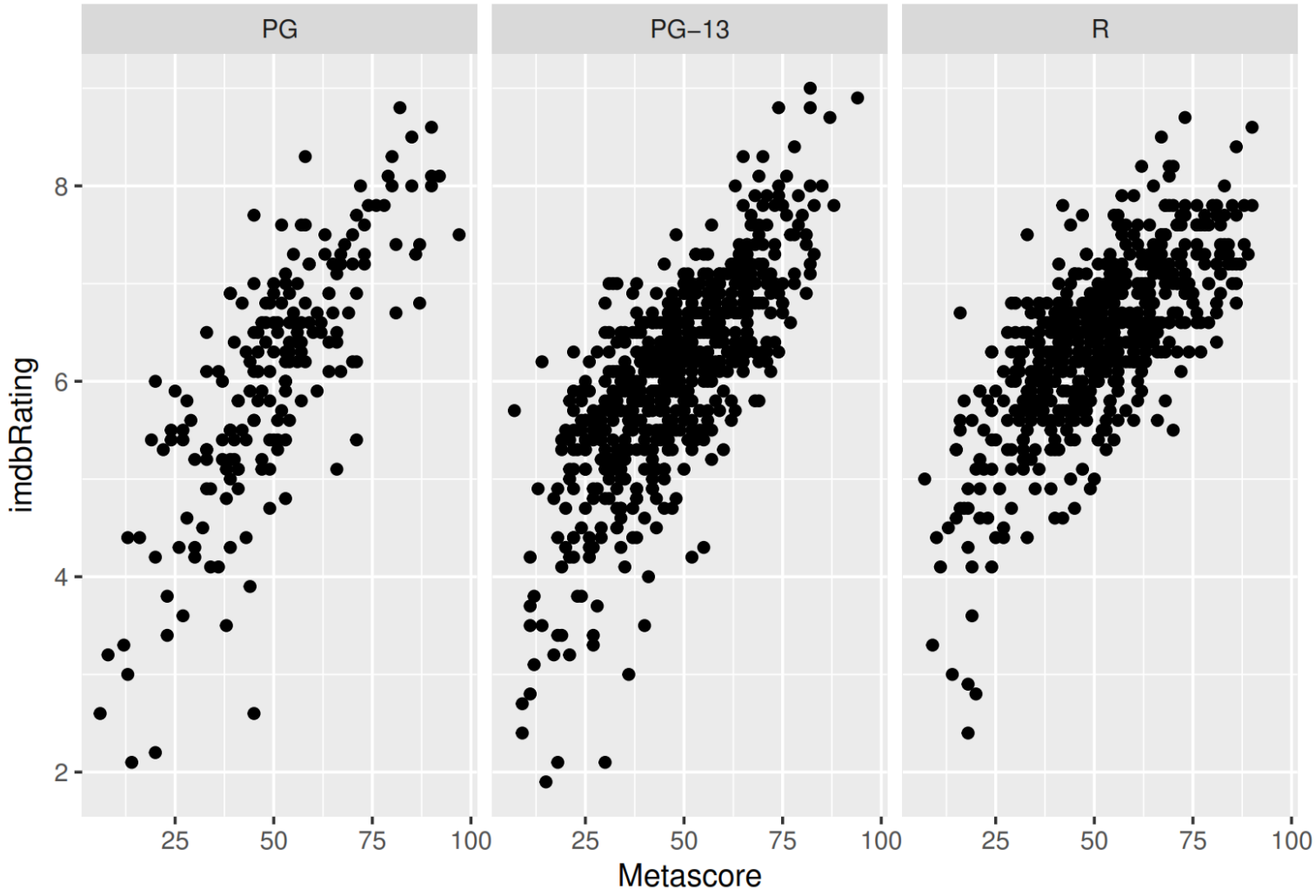
DATAMOTUS

- We have already discussed using aesthetics (colour and shape) to compare subgroups, drawing all groups on the same plot.
- Faceting takes an alternative approach: It creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset in an arrangement that facilitates comparison.

- Grammar of graphics: faceting

DATAMOTUS

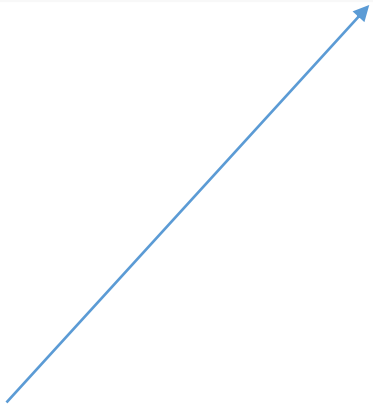
```
movies_sub <- movies[movies$genre_first %in% c("Action","Comedy") &
                     movies$Rated %in% c("PG","PG-13","R"),]
ggplot(movies_sub, aes(x=Metascore, y=imdbRating))+
  geom_point()+facet_grid(.~Rated)
```



- Grammar of graphics: faceting

DATAMOTUS

```
ggplot(movies_sub, aes(x=Metascore, y=imdbRating))+
  geom_point()+facet_grid(.~Rated)
```



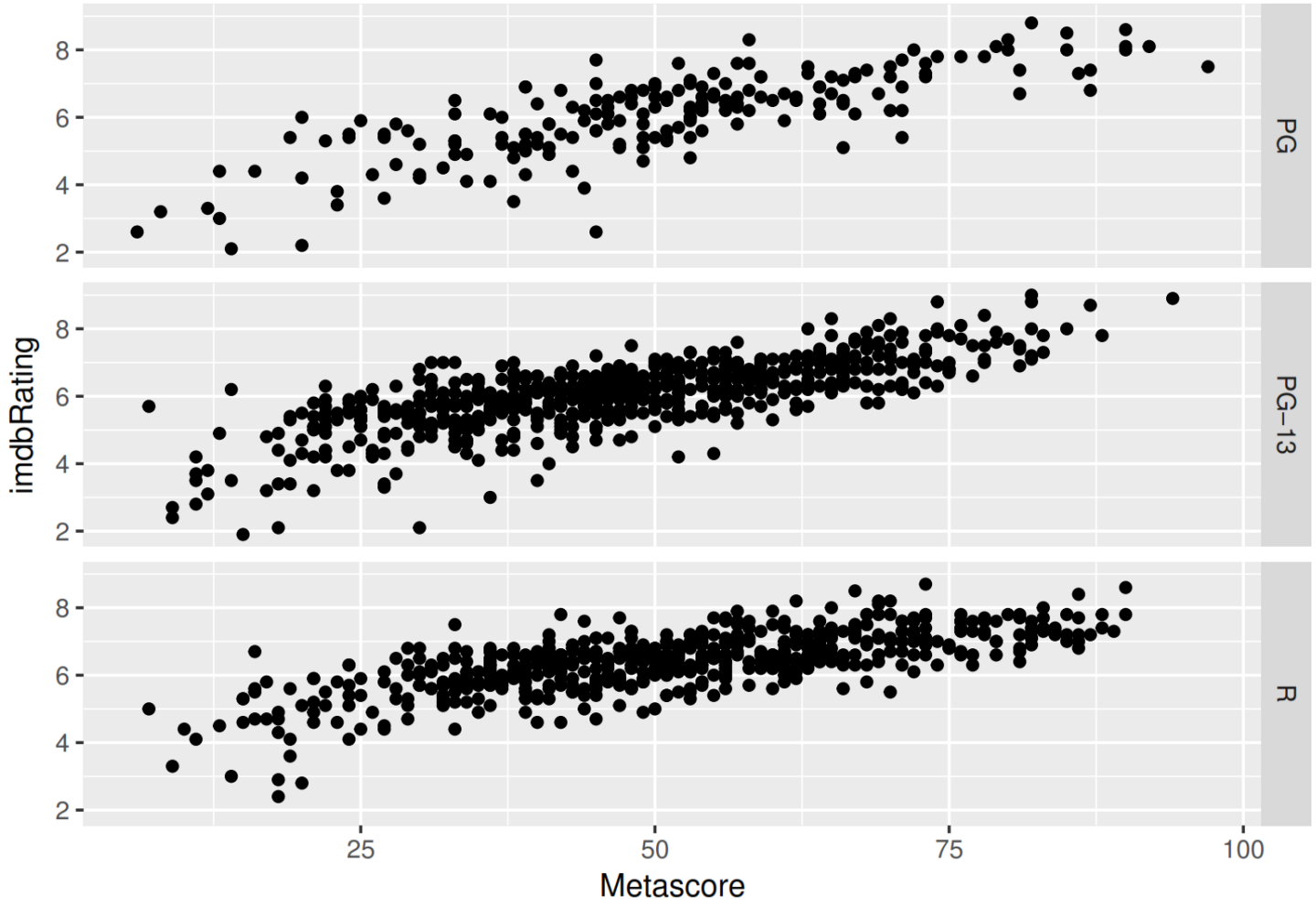
a formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column).

• Grammar of graphics: faceting

DATAMOTUS

faceting by rows

```
ggplot(movies_sub, aes(x=Metascore, y=imdbRating))+
  geom_point()+facet_grid(Rated~.)
```

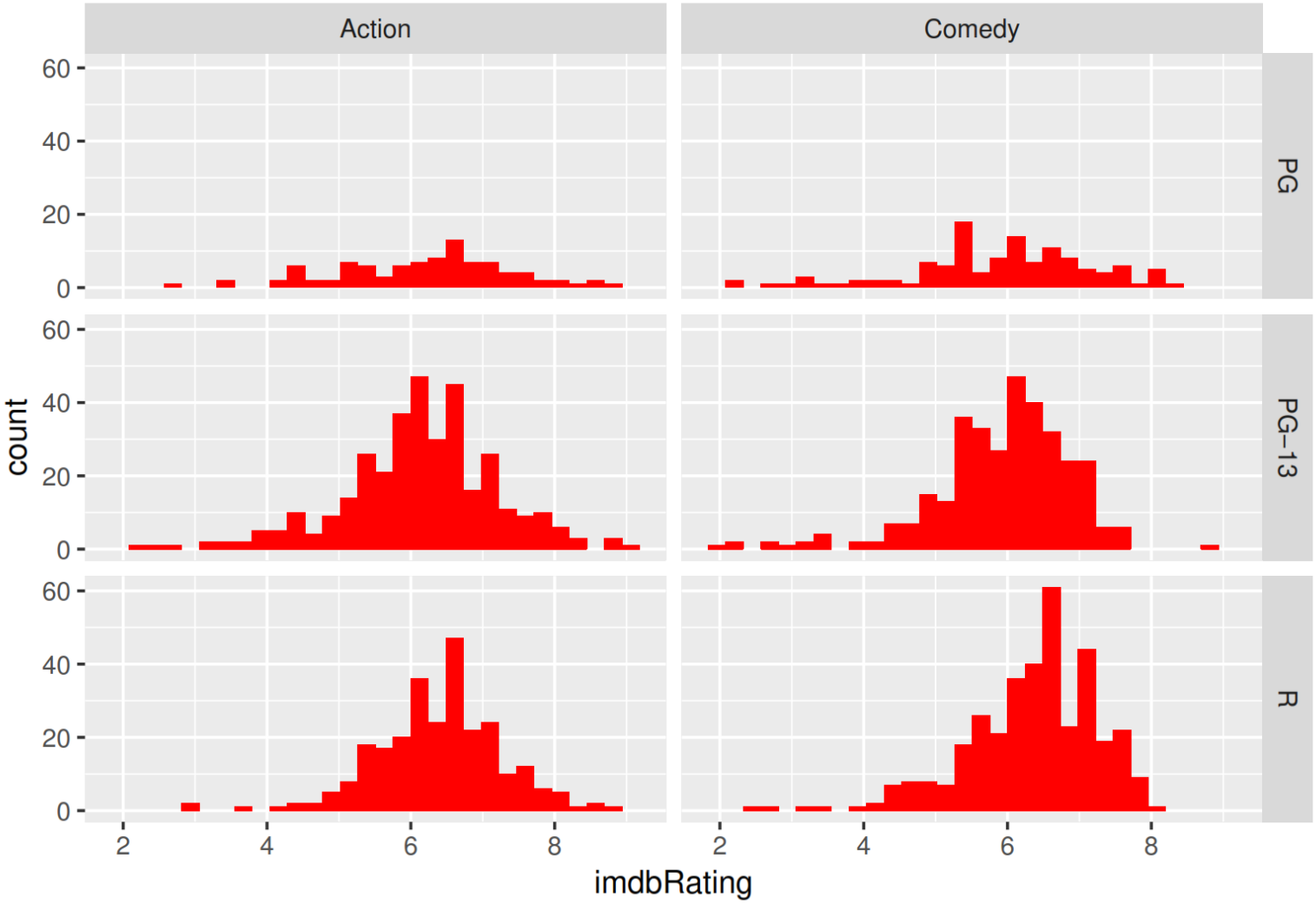


• Grammar of graphics: faceting

DATAMOTUS

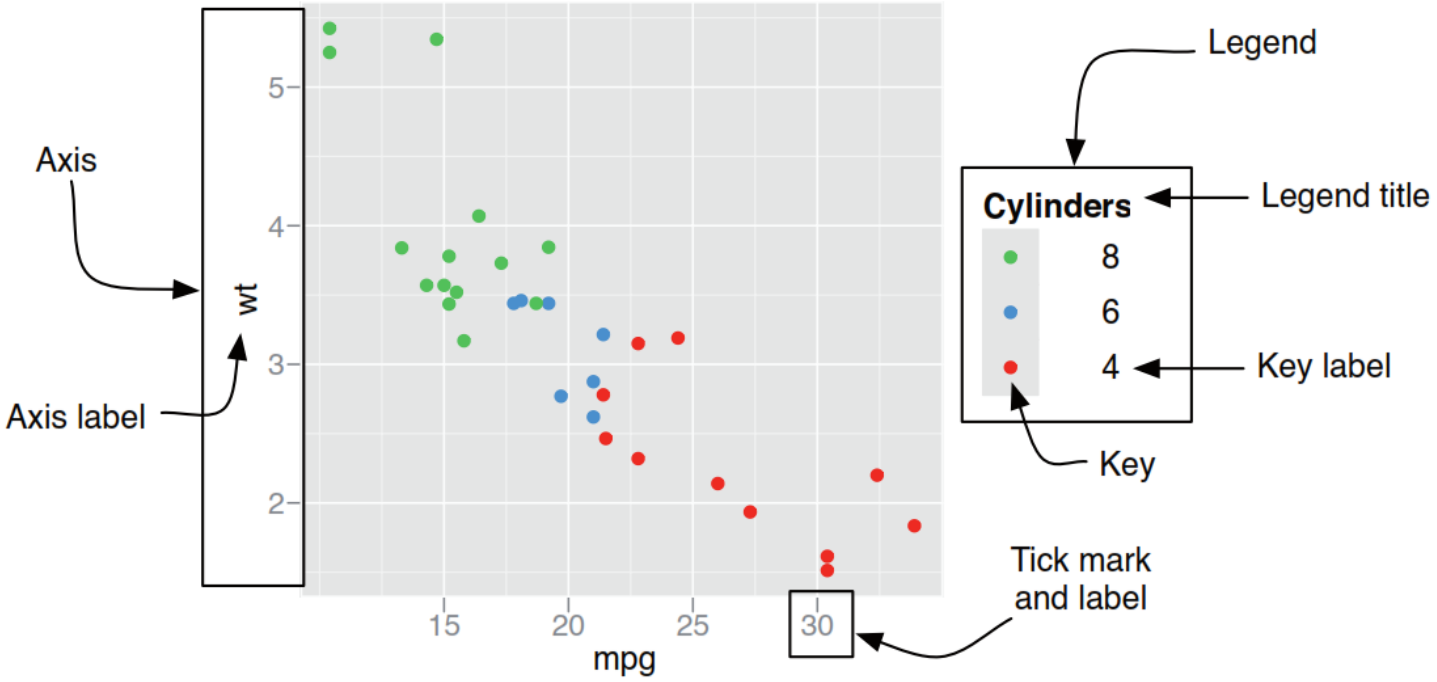
faceting by rows and columns: histogram

```
ggplot(movies_sub, aes(x=imdbRating))+ geom_histogram(fill="red")+
  facet_grid(Rated~genre_first)
```



• Grammar of graphics: guides

Collectively, axes and legends are called guides, and they are the inverse of the scale: they allow you to read observations from the plot and map them back to their original values

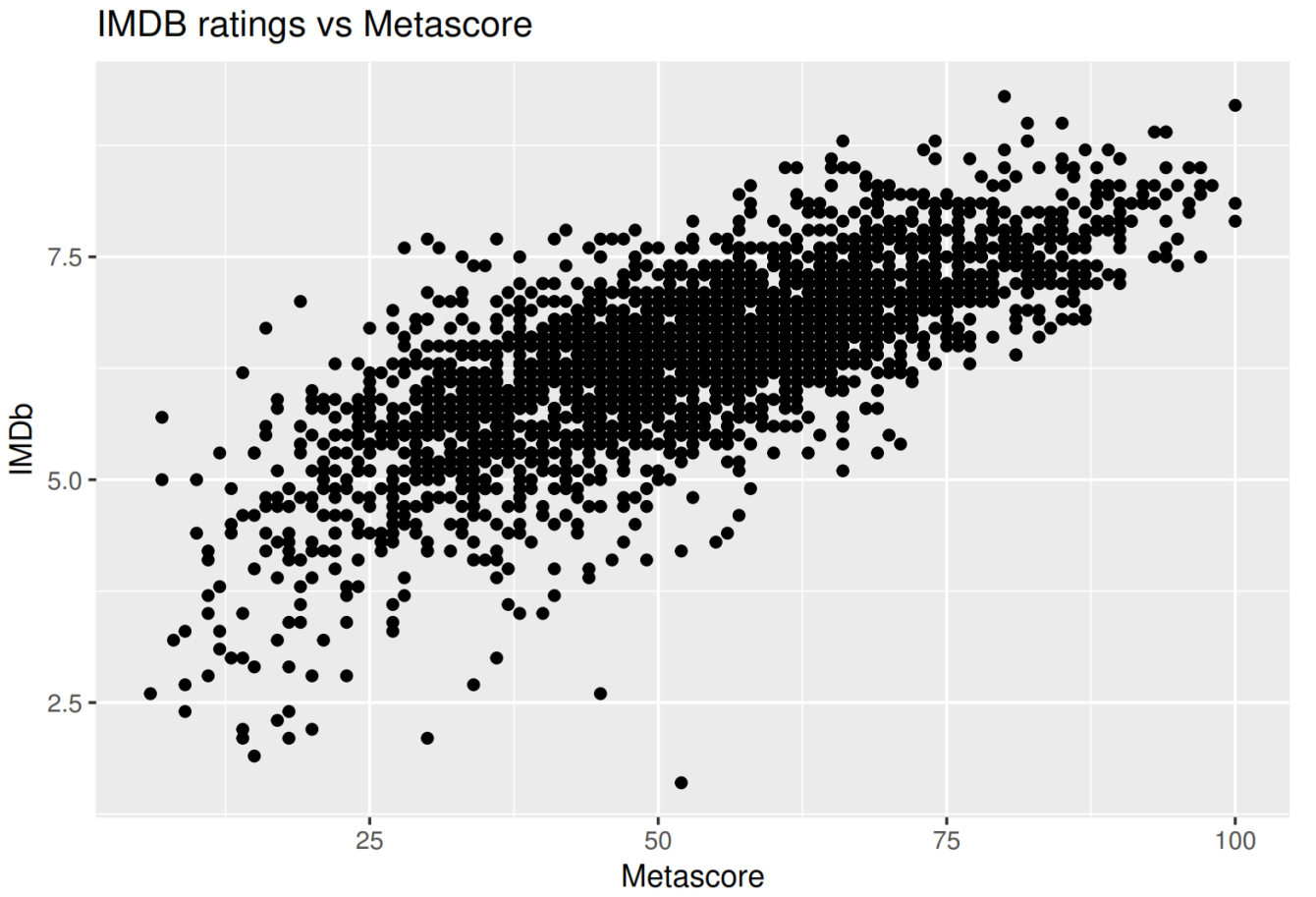


• Grammar of graphics: guides and themes

DATAMOTUS

Title, labels for x and y axes

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  ggtitle("IMDB ratings vs Metascore")+xlab("Metascore") + ylab("IMDb")
```

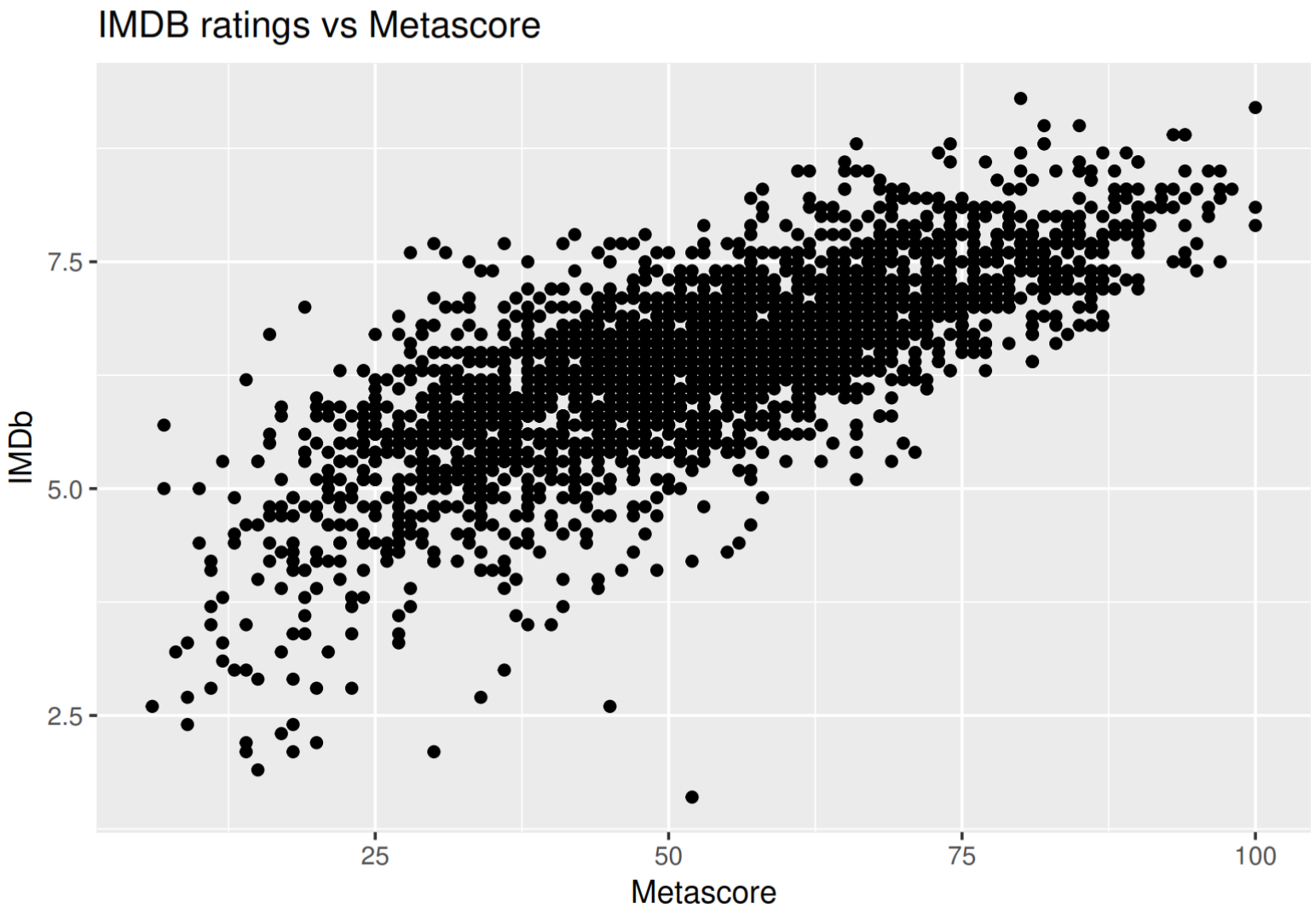


- Grammar of graphics: guides and themes

DATAMOTUS

Title. labels for x and v axes

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")
```



• Grammar of graphics: guides and themes

DATAMOTUS

- The appearance of non-data elements of the plot is controlled by the theme system.
- The theme system does not affect how the data is rendered by geoms, or how it is transformed by scales.
- Themes don’t change the perceptual properties of the plot, but they do help you make the plot aesthetically pleasing or match existing style guides.
- Themes give you control over things like the fonts in all parts of the plot: the title, axis labels, axis tick labels, strips, legend labels and legend key labels; and the colour of ticks, grid lines and backgrounds (panel, plot, strip and legend)

• Grammar of graphics: guides and themes

DATAMOTUS

Some elements of the theme

Theme element	Type	Description
`axis.line`	segment	line along axis
`axis.text.x`	text	x axis label
`axis.text.y`	text	y axis label
`axis.ticks`	segment	axis tick marks
`axis.title.x`	text	horizontal tick labels
`axis.title.y`	text	vertical tick labels
`legend.background`	rect	background of legend
`legend.key`	rect	background underneath legend keys
`legend.text`	text	legend labels
`legend.title`	text	legend name
`panel.background`	rect	background of panel
`panel.border`	rect	border around panel
`panel.grid.major`	line	major grid lines
`panel.grid.minor`	line	minor grid lines
`plot.background`	rect	background of the entire plot
`plot.title`	text	plot title
`strip.background`	rect	background of facet labels
`strip.text.x`	text	text for horizontal strips
`strip.text.y`	text	text for vertical strips

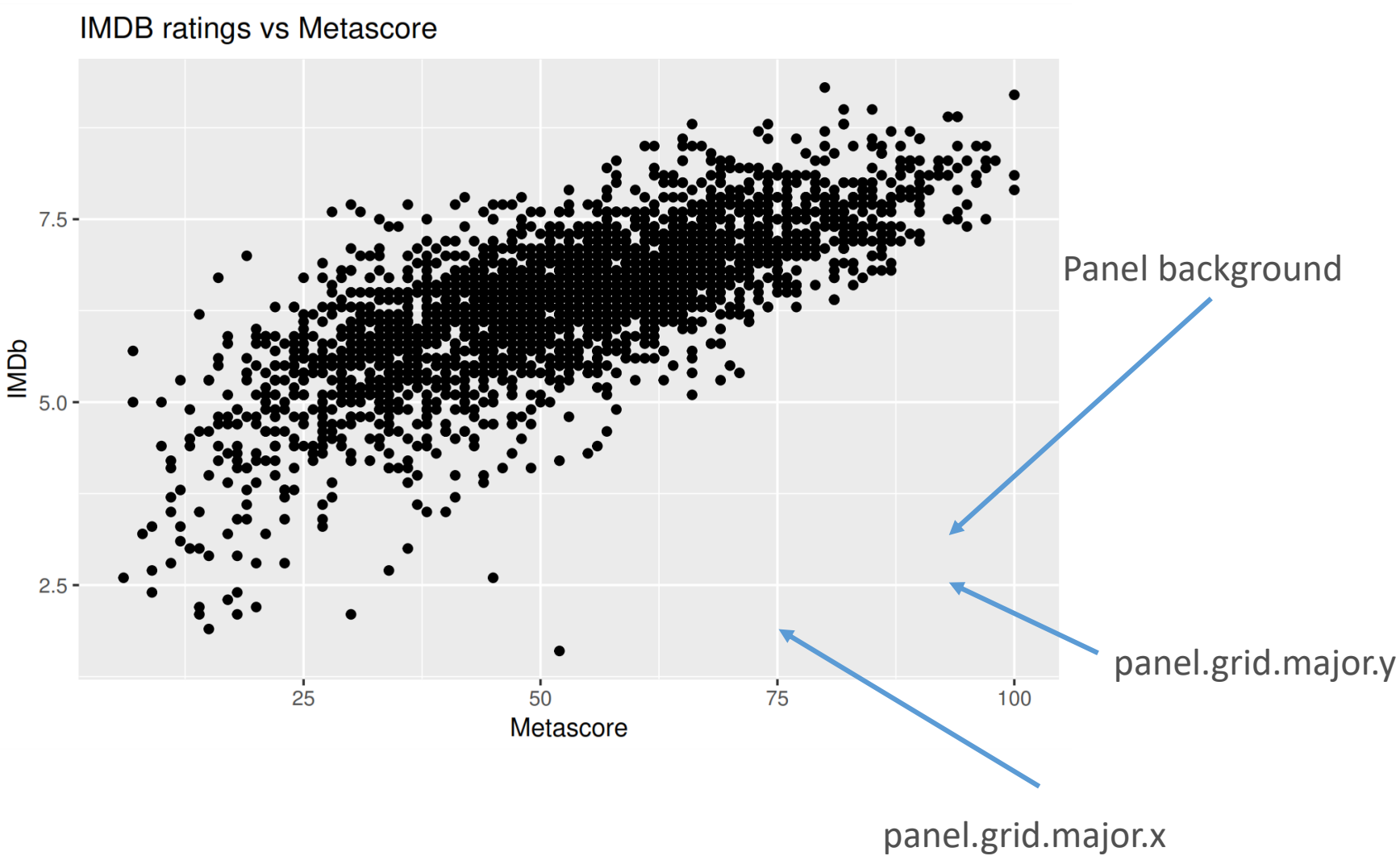
• Grammar of graphics: guides and themes

DATAMOTUS

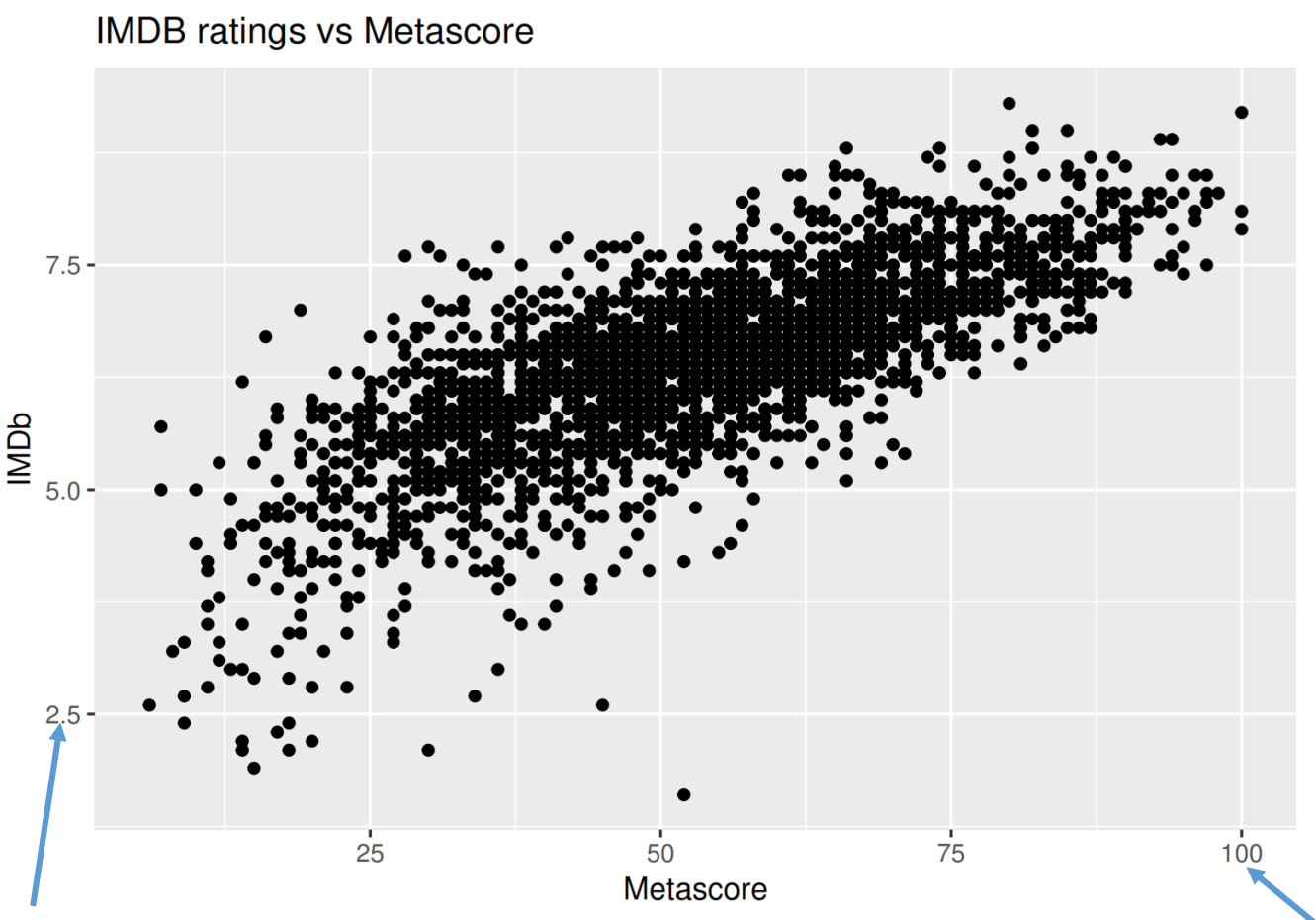
The elements

line	all line elements (element_line)
rect	all rectangular elements (element_rect)
text	all text elements (element_text)

- Grammar of graphics: guides and themes



- Grammar of graphics: guides and themes



axis.text.y

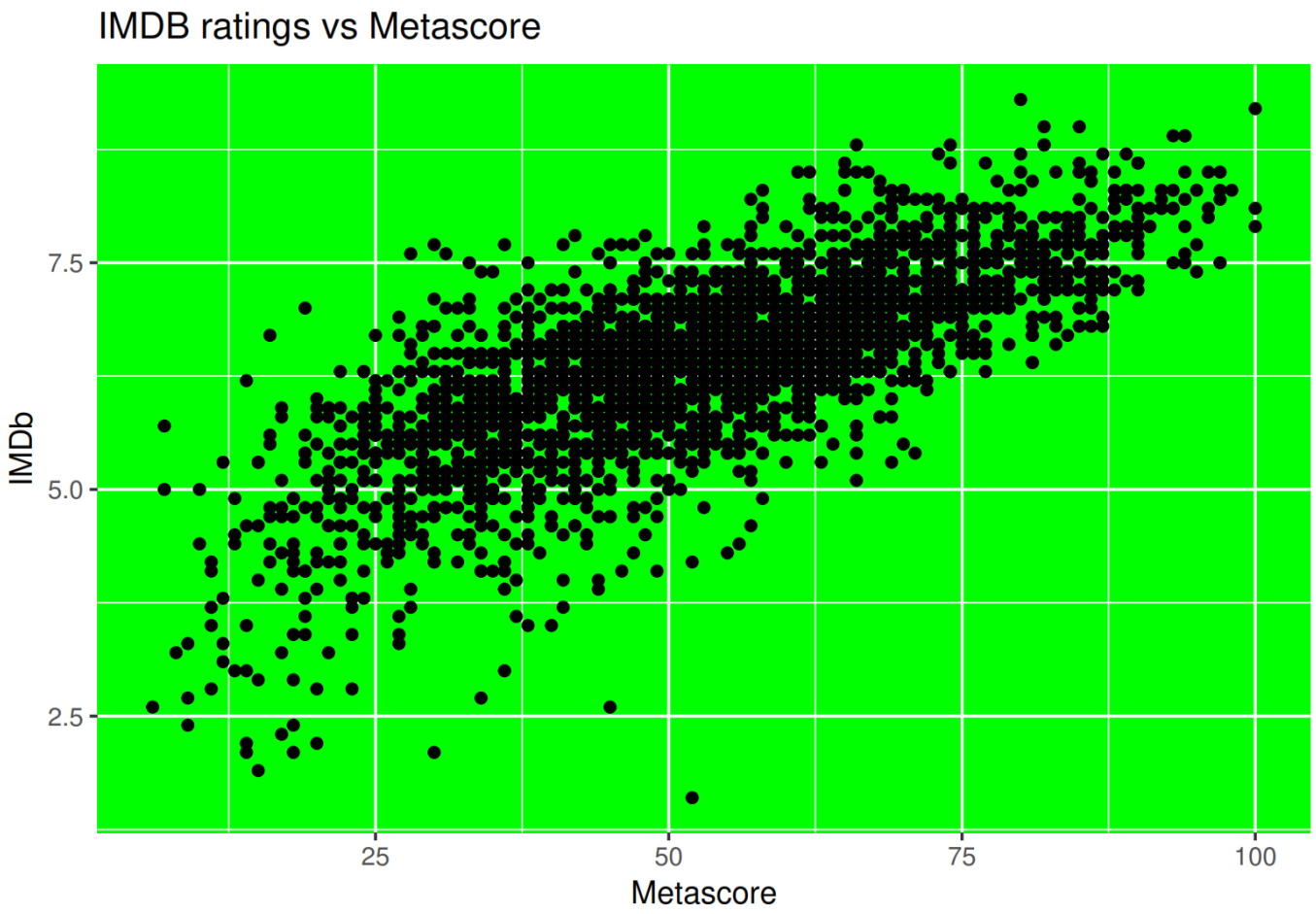
axis.text.x

• Grammar of graphics: guides and themes

DATAMOTUS

Change background color to green

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")+
  theme(panel.background =  element_rect(fill = "green"))
```

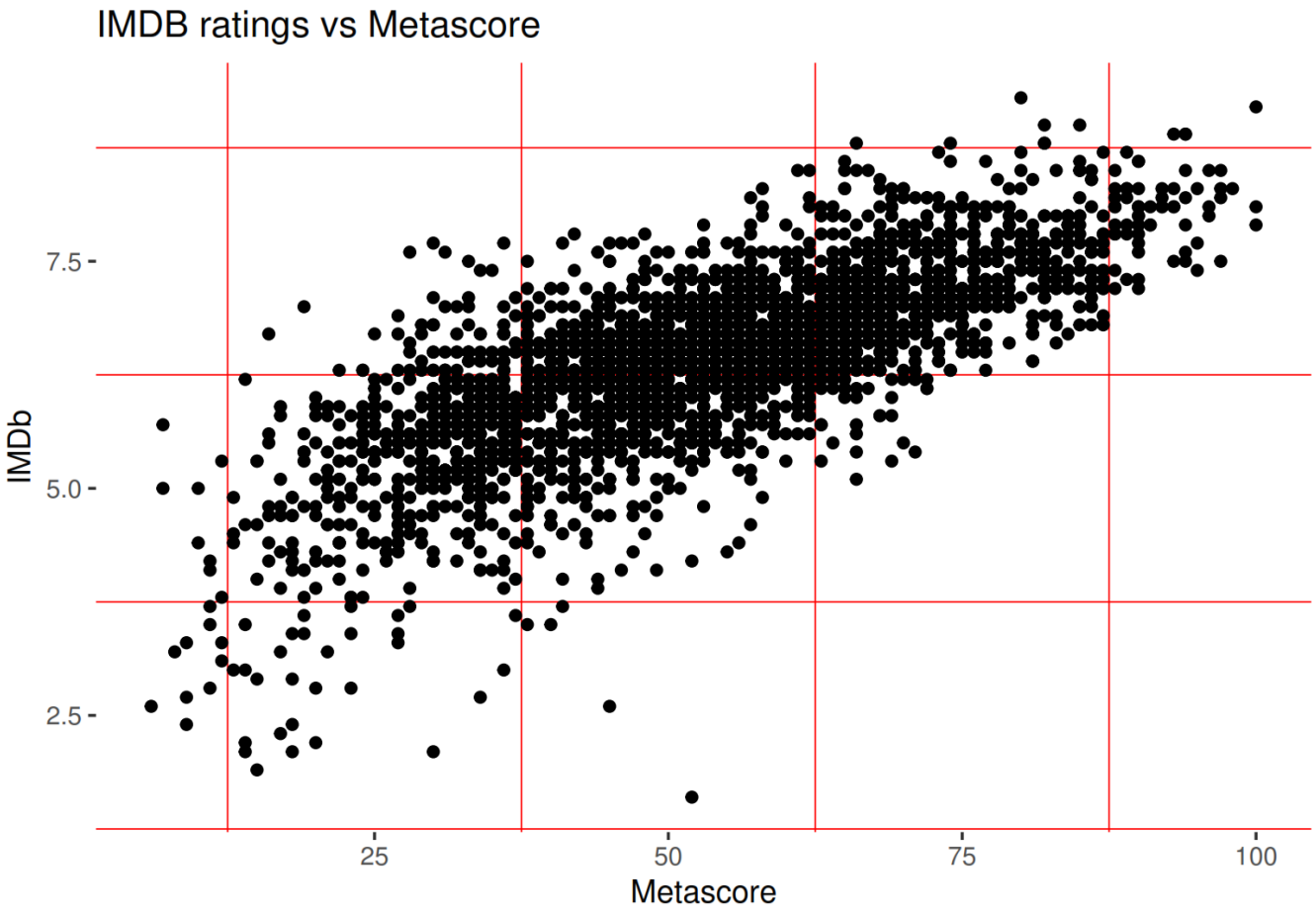


• Grammar of graphics: guides and themes

DATAMOTUS

Control the grid lines

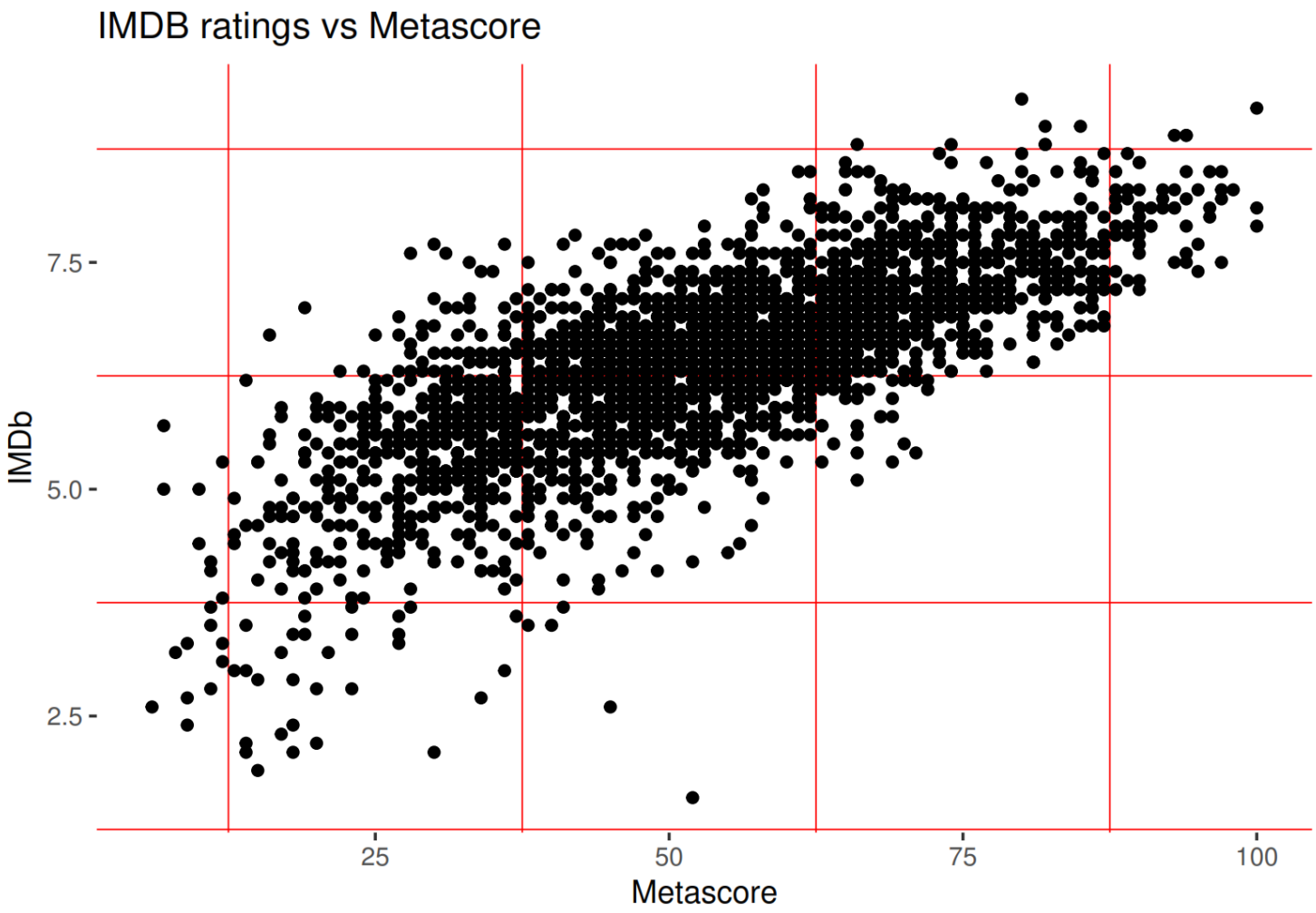
```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")+
  theme(panel.background =  element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_line(color="red"))
```



- Grammar of graphics: guides and themes: example

DATAMOTUS

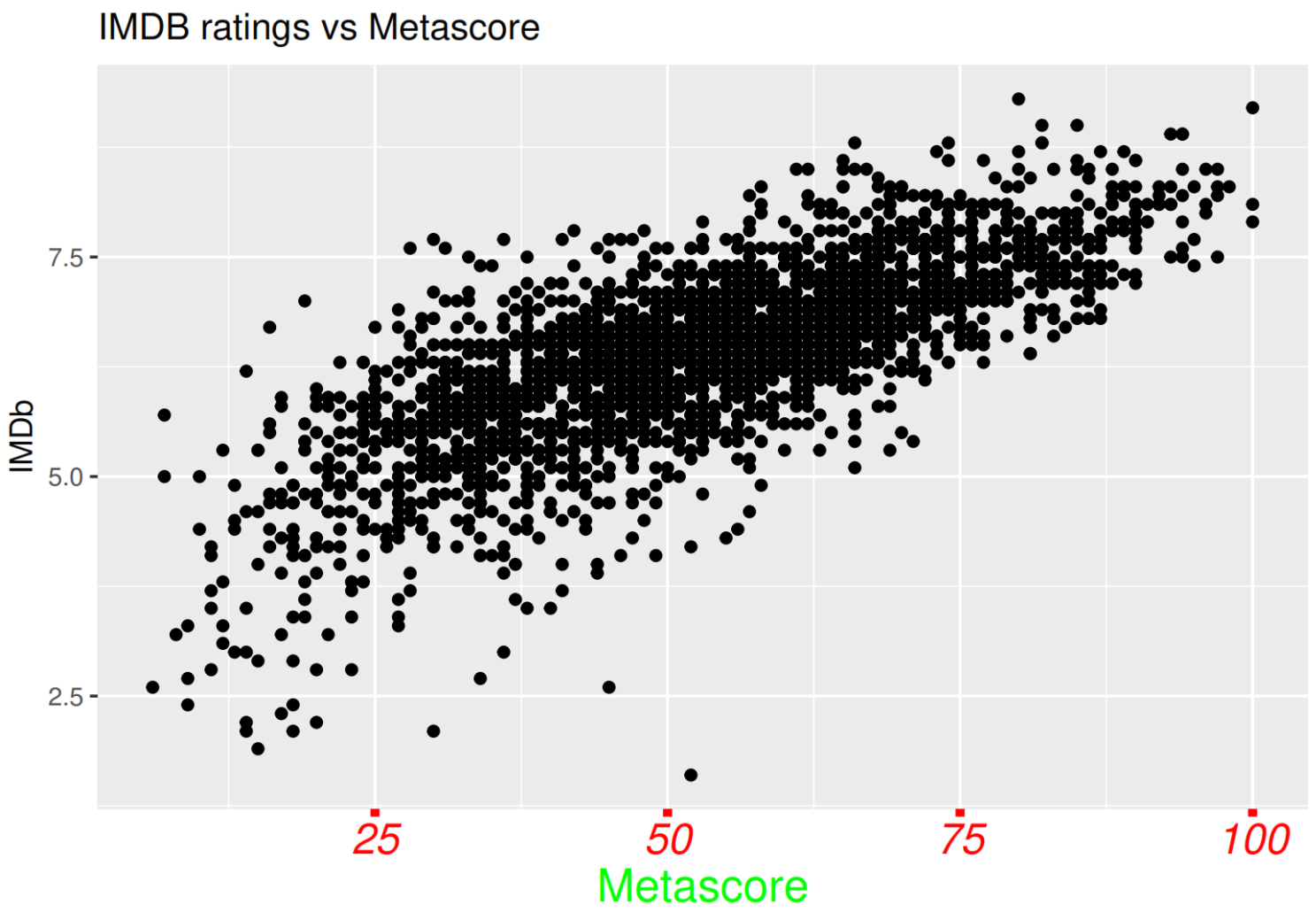
```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")+
  theme(panel.background =  element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_line(color="red"))
```



- Grammar of graphics: guides and themes - example

DATAMOTUS

```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")+
  theme(axis.text.x = element_text(size=15, color="red", face="italic"),
        axis.title.x = element_text(color="green", size=16),
        axis.ticks.x = element_line(size=1.5, color="red"))
```

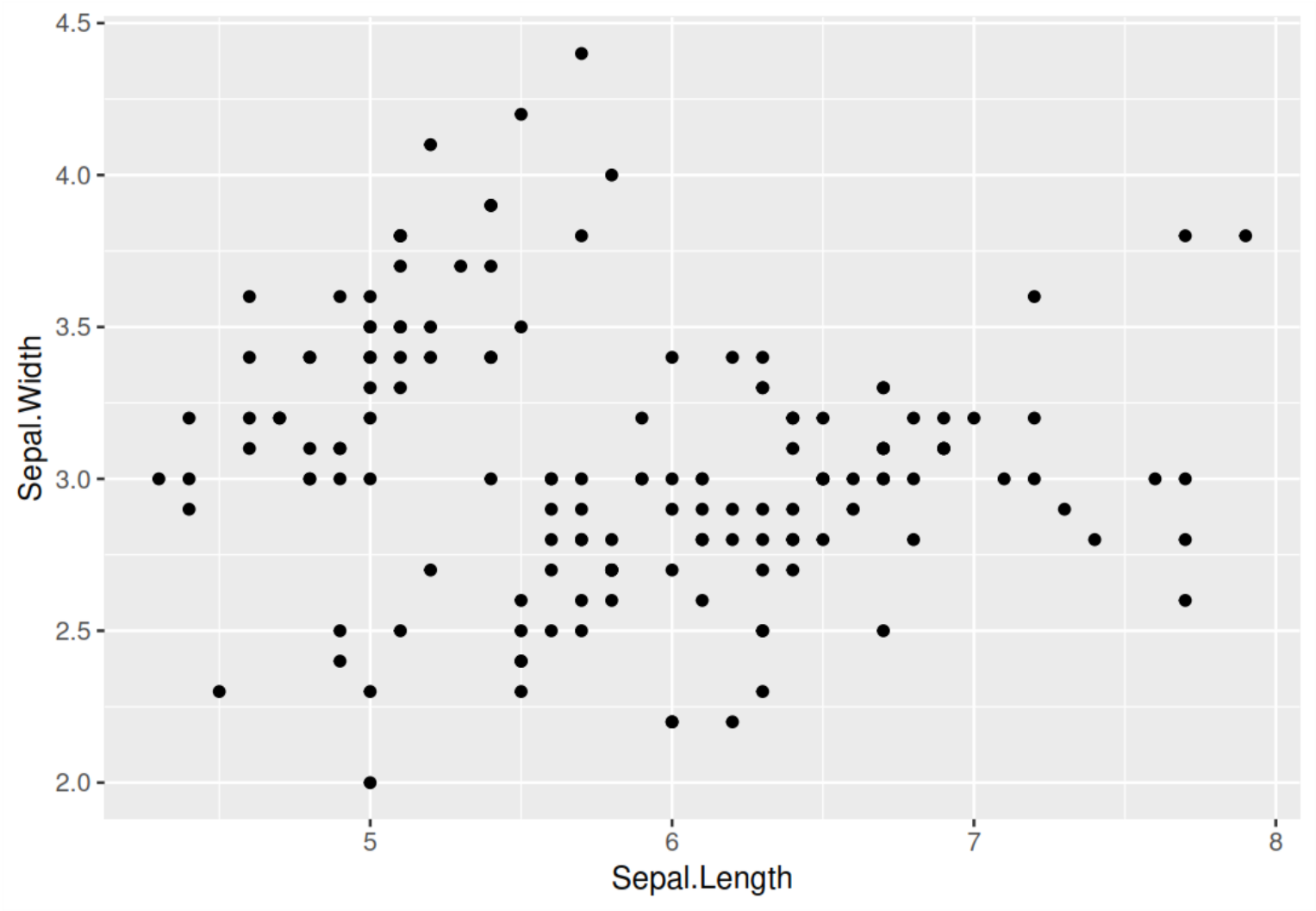


Vizualization: recap

Data viz: iris

What kind of relationship do you see ?

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+geom_point()
```



Data viz: iris

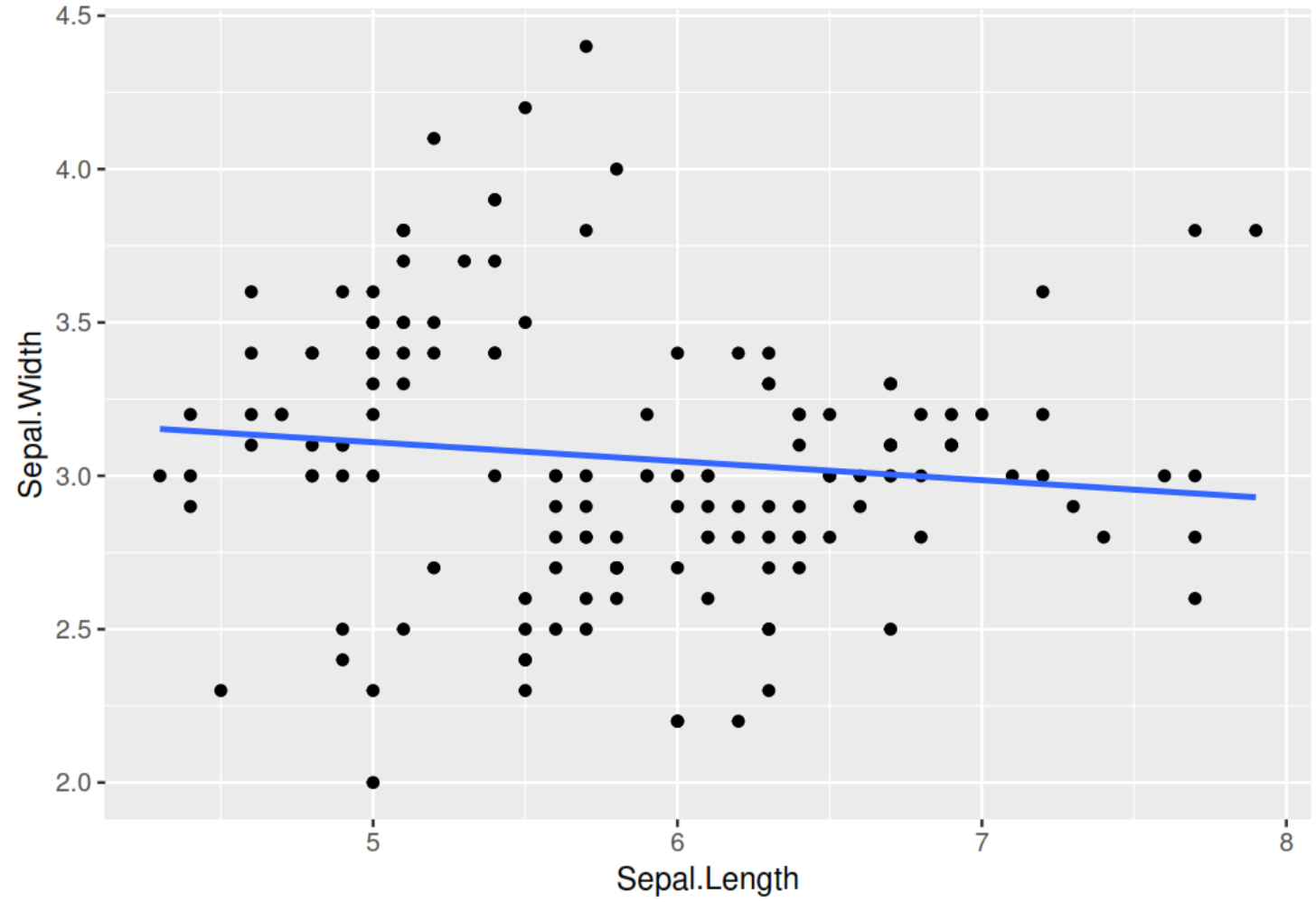
look at the correlation

```
cor(iris$Sepal.Length, iris$Sepal.Width)
```

```
## [1] -0.1175698
```

Data viz: iris

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+geom_point()+ geom_smooth(method='lm', se=F)
```



Looking by specis

Setosa

```
cor(iris$Sepal.Length[iris$Species=='setosa'],
    iris$Sepal.Width[iris$Species=='setosa'])
```

```
## [1] 0.7425467
```

Versicolor

```
cor(iris$Sepal.Length[iris$Species=='versicolor'],
    iris$Sepal.Width[iris$Species=='versicolor'])
```

```
## [1] 0.5259107
```

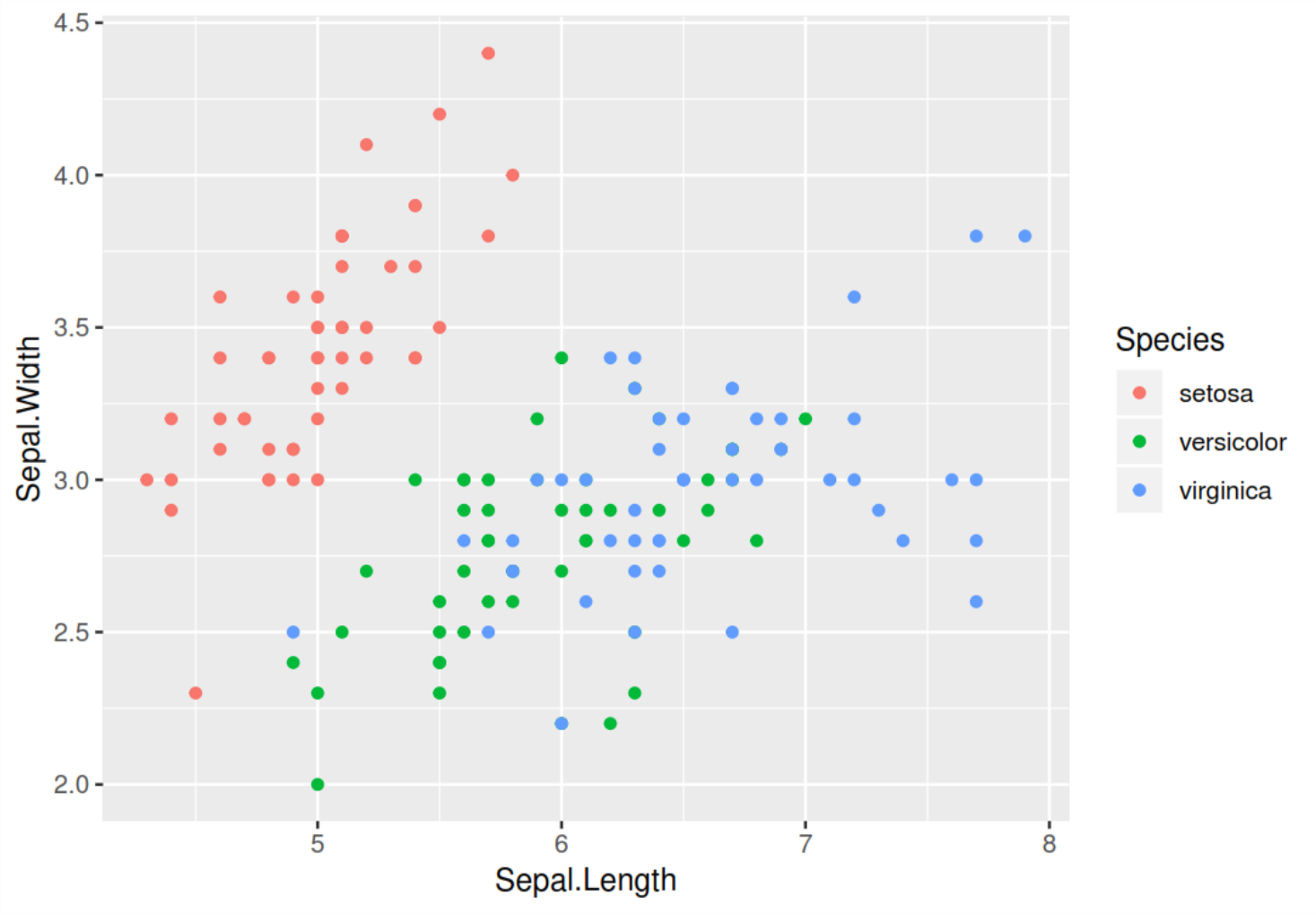
Virginica

```
cor(iris$Sepal.Length[iris$Species=='virginica'],
    iris$Sepal.Width[iris$Species=='virginica'])
```

```
## [1] 0.4572278
```

Data viz: iris

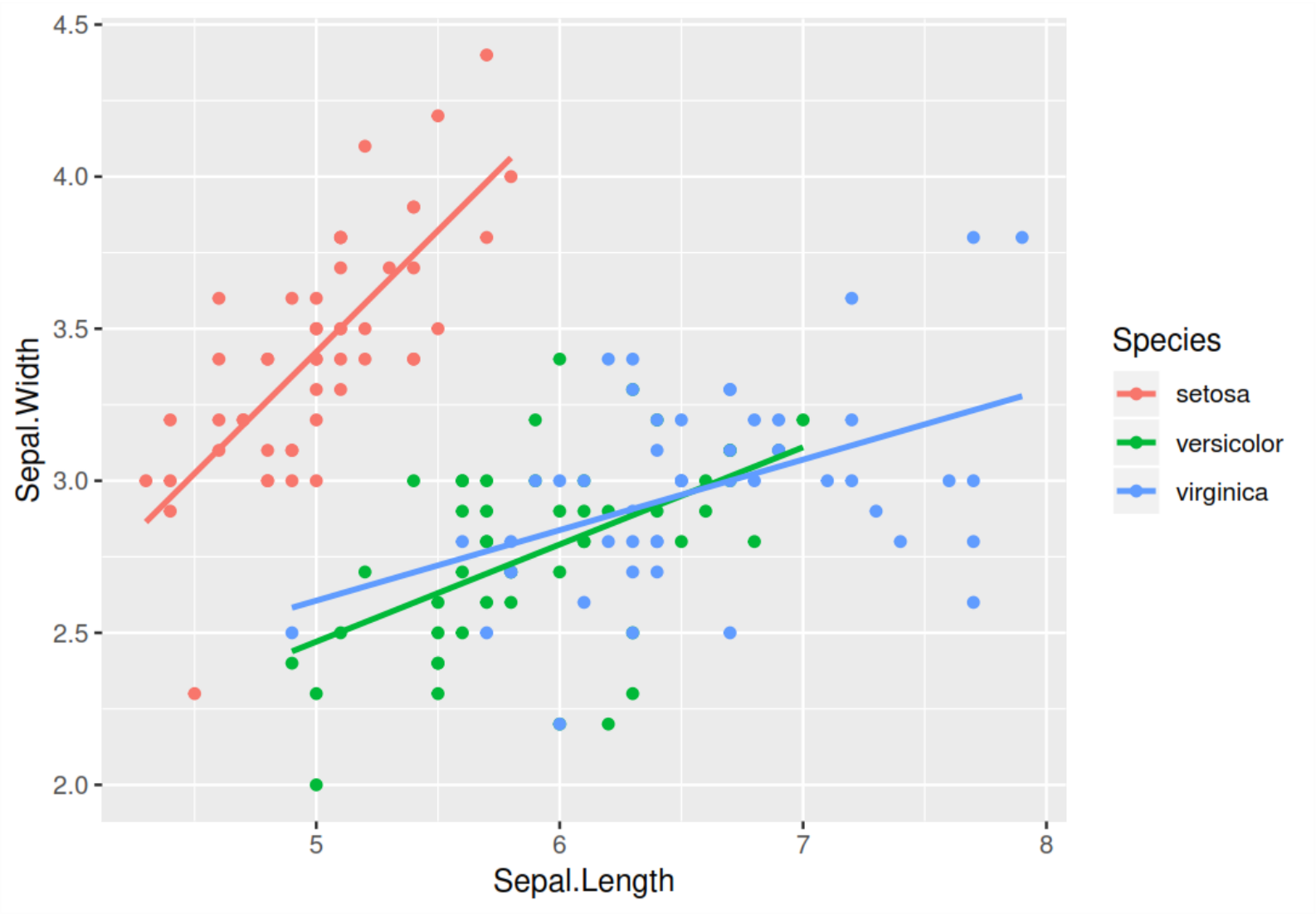
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+geom_point()
```



Data viz: iris

DATAMOTUS

```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color = Species))+geom_point()+
  geom_smooth(method = 'lm', se=F)
```



Data Manipulation, transformation and aggregation: basic R

• Apply family functions

DATAMOTUS

- The apply family consists of vectorized functions which minimize your need to explicitly create loops.

- These functions will apply a specified function to a data object and their primary difference is in the object class in which the function is applied to (list vs. matrix, etc.) and the object class that will be returned from the function.

  - **apply()** for matrices and data frames
  - **lapply()** for lists…output as list
  - **sapply()** for lists…output simplified
  - **tapply()** for vectors

• Apply family functions

DATAMOTUS

Recall the loop we have created to calculate the mean for every column in the data frame mtcars

```
x <- c()
for (i in 1:ncol(mtcars)){
  x1 <- mean(mtcars[,i])
  x <- c(x,x1)
}
x
```

```
##   [1]  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250
##   [7]  17.848750   0.437500   0.406250   3.687500   2.812500
```

• Apply family functions

The same result can be obtained by function apply()

apply(X, MARGIN, FUN)

- First you specify the data (X)
- MARGIN is a vector giving the subscripts which the function will be applied over. E.g., for a data frames/matrix 1 indicates rows, 2 indicates columns.
- FUN is the function to be applied, can be user defined function as well

• Apply family functions

DATAMOTUS

Calculate the mean for each column

```
data(mtcars)
apply(mtcars, 2, mean)
```

```
##        mpg        cyl       disp         hp       drat         wt
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250
##       qsec         vs         am       gear       carb
##  17.848750   0.437500   0.406250   3.687500   2.812500
```

• Apply family functions

DATAMOTUS

- Calculate the means for each row:
- Rows have names in the data frame, that's why the resulting vector is named

```
apply(mtcars, 1, mean)
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710
##             29.90727             29.98136             23.59818
##      Hornet 4 Drive   Hornet Sportabout              Valiant
##             38.73955             53.66455             35.04909
##          Duster 360           Merc 240D            Merc 230
##             59.72000             24.63455             27.23364
##            Merc 280           Merc 280C           Merc 450SE
##             31.86000             31.78727             46.43091
##          Merc 450SL          Merc 450SLC  Cadillac Fleetwood
##             46.50000             46.35000             66.23273
## Lincoln Continental   Chrysler Imperial            Fiat 128
##             66.05855             65.97227             19.44091
##         Honda Civic      Toyota Corolla       Toyota Corona
##             17.74227             18.81409             24.88864
##    Dodge Challenger         AMC Javelin          Camaro Z28
##             47.24091             46.00773             58.75273
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2
##             57.37955             18.92864             24.77909
##        Lotus Europa      Ford Pantera L        Ferrari Dino
##             24.88027             60.97182             34.50818
##       Maserati Bora          Volvo 142E
##             63.15545             26.26273
```

• Apply family functions

DATAMOTUS

- The generic function **quantile()** produces sample quantiles corresponding to the given probabilities.

quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
names = TRUE, type = 7, ...)

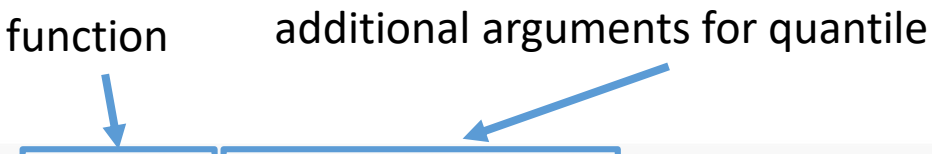
- If you want to get 0.25,0.5,0.75 quantiles, which are respectively 1st quartile, medina and 3rd quartile, you can use the following function

```
quantile(x, probs=c(0.25,0.5,0.7))
```

• Apply family functions

DATAMOTUS

With apply function, the additional arguments for the FUN are defined within the apply function




```
a <- apply(mtcars, 2, quantile, probs=c(.25,.5, .75))
class(a)
```

```
## [1] "matrix"
```

```
a
```

```
##        mpg cyl    disp    hp  drat      wt    qsec vs am gear carb
## 25% 15.425   4 120.825  96.5 3.080 2.58125 16.8925  0  0    3    2
## 50% 19.200   6 196.300 123.0 3.695 3.32500 17.7100  0  0    4    2
## 75% 22.800   8 326.000 180.0 3.920 3.61000 18.9000  1  1    4    4
```

• Apply family functions

DATAMOTUS

recall the for loop you used to get the normalized data frame

```
normalize_df <- function(df){
  # Define function normalize
  normalize <- function(x) {
    return ((x - min(x)) / (max(x) - min(x)))
  }

  df1 <-c()
  for (i in 1:ncol(df)){
    x <- normalize(df[,i])
    df1 <- cbind(df1,x)
  }
  df1 <- as.data.frame(df1)
  colnames(df1) <- colnames(df)
  # returns df
  return(df1)
}
```

- Apply family functions

DATAMOTUS

First define the min-max normalization function

```
normalize <- function(x) {
    return ((x - min(x)) / (max(x) - min(x)))
  }
```

Then use your function with apply

• Apply family functions

DATAMOTUS

The result is a matrix, transform it to data frame for convenience

```
mtcars_norm <- apply(mtcars, 2, normalize)
class(mtcars_norm)
```

```
## [1] "matrix"
```

```
mtcars_norm <- as.data.frame(mtcars_norm)
head(mtcars_norm)
```

```
##                           mpg cyl      disp        hp      drat        wt
## Mazda RX4           0.4510638 0.5 0.2217511 0.2049470 0.5253456 0.2830478
## Mazda RX4 Wag       0.4510638 0.5 0.2217511 0.2049470 0.5253456 0.3482485
## Datsun 710          0.5276596 0.0 0.0920429 0.1448763 0.5023041 0.2063411
## Hornet 4 Drive      0.4680851 0.5 0.4662010 0.2049470 0.1474654 0.4351828
## Hornet Sportabout   0.3531915 1.0 0.7206286 0.4346290 0.1797235 0.4927129
## Valiant             0.3276596 0.5 0.3838863 0.1872792 0.0000000 0.4978266
##                           qsec vs am gear      carb
## Mazda RX4           0.2333333  0  1  0.5 0.4285714
## Mazda RX4 Wag       0.3000000  0  1  0.5 0.4285714
## Datsun 710          0.4892857  1  1  0.5 0.0000000
## Hornet 4 Drive      0.5880952  1  0  0.0 0.0000000
## Hornet Sportabout   0.3000000  0  0  0.0 0.1428571
## Valiant             0.6809524  1  0  0.0 0.0000000
```

• Apply family functions

DATAMOTUS

You can define the function inside the apply() as well

user defined function

```
mtcars_norm1 <- apply(mtcars, 2, function(x) (x - min(x)) / (max(x) - min(x)))
mtcars_norm1 <- as.data.frame(mtcars_norm1)
head(mtcars_norm1, n=4)
```

```
##                      mpg cyl      disp        hp      drat        wt
## Mazda RX4       0.4510638 0.5 0.2217511 0.2049470 0.5253456 0.2830478
## Mazda RX4 Wag   0.4510638 0.5 0.2217511 0.2049470 0.5253456 0.3482485
## Datsun 710      0.5276596 0.0 0.0920429 0.1448763 0.5023041 0.2063411
## Hornet 4 Drive  0.4680851 0.5 0.4662010 0.2049470 0.1474654 0.4351828
##                     qsec vs am gear      carb
## Mazda RX4       0.2333333  0  1  0.5 0.4285714
## Mazda RX4 Wag   0.3000000  0  1  0.5 0.4285714
## Datsun 710      0.4892857  1  1  0.5 0.0000000
## Hornet 4 Drive  0.5880952  1  0  0.0 0.0000000
```

• Apply family functions

DATAMOTUS

Check if two objcts are identical

```
identical(mtcars_norm, mtcars_norm1)
```

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

• Apply family functions

DATAMOTUS

lapply is similar to apply, but it takes a list as an input, and returns a list as the output.

```
lapply(mtcars,mean)
```

```
## $mpg
## [1] 20.09062
##
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
##
## $hp
## [1] 146.6875
##
## $drat
## [1] 3.596563
```

It works with data frame as well, because in essence data frame is just a list of vectors of the same length

• Apply family functions

DATAMOTUS

Create a list with two matrices and a vector

```
l1 <- list(
  x = matrix(rnorm(50), ncol=5),
  y = matrix(rnorm(30), nrow=3),
  z = 1:25
)
l1
```

```
## $x
##              [,1]         [,2]       [,3]       [,4]        [,5]
##  [1,] -2.4959903 -1.053614412  2.3359347  0.2971772 -0.89133853
##  [2,] -1.8038803  1.503497446  0.1422945  0.1054970  0.88835587
##  [3,]  0.4092807 -0.648204108 -0.1686080 -1.3014793 -0.20254328
##  [4,] -1.0517403 -0.329860947 -0.4558838  0.3701725 -0.57157325
##  [5,] -0.6806074  0.344743058 -0.2083472 -0.9485254 -0.41614854
##  [6,] -0.5504247  0.210262366  1.1004514  0.6150502 -0.94783216
##  [7,]  2.2543309 -0.228192505 -1.2621387  1.4170380  0.06234551
##  [8,] -0.1184786 -1.033477962 -0.2222443 -0.1628075  0.46433572
##  [9,] -1.3946713 -0.041674634 -1.1061795  2.2223583 -0.06628888
## [10,]  2.7375796 -0.009833064 -0.4478952  0.7949535  0.48662753
##
## $y
##              [,1]       [,2]      [,3]       [,4]       [,5]       [,6]
## [1,] -1.19078876 -0.7887327 0.3302723  1.9651163 -1.6577105 -1.4747456
```

rnorm() is going to create random numbers from normal distribution

- Apply family functions

DATAMOTUS

```
lapply(l1, mean)
```

```
## $x
## [1] -0.04116396
##
## $y
## [1] -0.1137894
##
## $z
## [1] 13
```

• Apply family functions

DATAMOTUS

sapply()

The sapply() function behaves similarly to lapply(); the only real difference is in the return value. sapply() will try to simplify the result of lapply() if possible. Essentially, sapply() calls lapply() on its input and then applies the following algorithm:

- If the result is a list where every element is length 1, then a vector is returned
- If the result is a list where every element is a vector of the same length (> 1), a matrix is returned.
- If neither of the above simplifications can be performed then a list is returned

- Apply family functions

DATAMOTUS

```
sapply(l1, mean)
```

```
##           x           y           z
## -0.04116396 -0.11378936 13.00000000
```

```
lapply(l1, mean)
```

```
## $x
## [1] -0.04116396
##
## $y
## [1] -0.1137894
##
## $z
## [1] 13
```

• Apply family functions

- Say you want to create a new data frame with only numeric variables from the data frame movies
- sapply will return a vector with value TRUE if the columns is numeric and FALSE otherwise

```
movies <- read.csv("movies3.csv", stringsAsFactors = F)
sapply(movies, is.numeric)
```

```
##                title         genre_first                year
##                FALSE               FALSE                TRUE
##             duration      gross_adjusted     budget_adjusted
##                 TRUE                TRUE                TRUE
##                gross              budget cast_facebook_likes
##                 TRUE                TRUE                TRUE
##              reviews               index               Rated
##                 TRUE                TRUE               FALSE
```

• Apply family functions

DATAMOTUS

Now you can use it to subset the data frame

```
movies_num <- movies[,sapply(movies, is.numeric)]
summary(movies_num)
```

```
##       year         duration     gross_adjusted      budget_adjusted
##  Min.   :1920   Min.   : 37.0   Min.   :9.730e+02   Min.   :      290
##  1st Qu.:1999   1st Qu.: 95.0   1st Qu.:1.626e+07   1st Qu.: 15303856
##  Median :2004   Median :105.0   Median :4.515e+07   Median : 35386325
##  Mean   :2003   Mean   :109.6   Mean   :8.453e+07   Mean   : 51922575
##  3rd Qu.:2010   3rd Qu.:119.0   3rd Qu.:1.020e+08   3rd Qu.: 72596990
##  Max.   :2016   Max.   :330.0   Max.   :3.503e+09   Max.   :354732272
##
##      gross                budget          cast_facebook_likes
##  Min.   :      703   Min.   :      218   Min.   :     0
##  1st Qu.: 12021631   1st Qu.: 11000000   1st Qu.:  2216
##  Median : 34514650   Median : 25000000   Median :  4529
##  Mean   : 57613345   Mean   : 40183590   Mean   : 12272
##  3rd Qu.: 75073078   3rd Qu.: 55000000   3rd Qu.: 16759
##  Max.   :760505847   Max.   :300000000   Max.   :656730
##
```

Data manipulation and transformation, dplyr

• Grammar of data manipulation

DATAMOTUS

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

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

These all combine naturally with group_by() which allows you to perform any operation “by group”.

DATAMOTUS

- Not necessary, but dplyr works the best with pipe like operator from **magrittr** package

- %>% operator takes the object from its left hand side and use it as an argument in the function on the right hand side

• Grammar of data manipulation

DATAMOTUS

- The ‘pipe’ operation is a handy tool to make your code more legible: %>%. Key points:

- It takes the output of your previous operation and uses it as an input to your next operation.

- You can determine where the previous argument goes with the period symbol, . , which acts as a placeholder.

- Understand how to use it by replacing the pipe operation with ‘then’ (in your mind, not in the code). For example, **filter(data, ...) %>% select(...)** filters first then selects columns from the output of filter.

- Grammar of data manipulation

DATAMOTUS

filtering USA games only

```
summer <- read.csv("summer.csv", stringsAsFactors = F)
library(dplyr)
summer_usa <- summer %>%
  filter(Country=="USA")

table(summer_usa$Country, summer_usa$Medal)
```

```
##
##       Bronze Gold Silver
##   USA   1098 2235   1252
```

• Grammar of data manipulation

DATAMOTUS

Filter and group_by

```
summer %>%
  filter(Country %in% c("USA", "FRA", "GBR")) %>%
  group_by(Country) %>%
  summarise(Count=n())
```

```
## # A tibble: 3 x 2
##   Country Count
##   <chr>   <int>
## 1 FRA      1396
## 2 GBR      1720
## 3 USA      4585
```

The result is a tibble

Count=n() creates a new variable named **Count** with frequencies, n() calculates frequencies

- Grammar of data manipulation

DATAMOTUS

Number of medals by country (is this long or wide format ?)

```
summer %>%
  filter(Country %in% c("USA", "FRA", "GBR")) %>%
  group_by(Country, Medal) %>%
  summarise(Count=n())
```

```
## # A tibble: 9 x 3
## # Groups:   Country [?]
##   Country Medal  Count
##   <chr>   <chr>  <int>
## 1 FRA     Bronze   497
## 2 FRA     Gold     408
## 3 FRA     Silver   491
## 4 GBR     Bronze   553
## 5 GBR     Gold     546
## 6 GBR     Silver   621
## 7 USA     Bronze  1098
## 8 USA     Gold    2235
## 9 USA     Silver  1252
```

- Grammar of data manipulation

DATAMOTUS

Using ggplot with dplyr

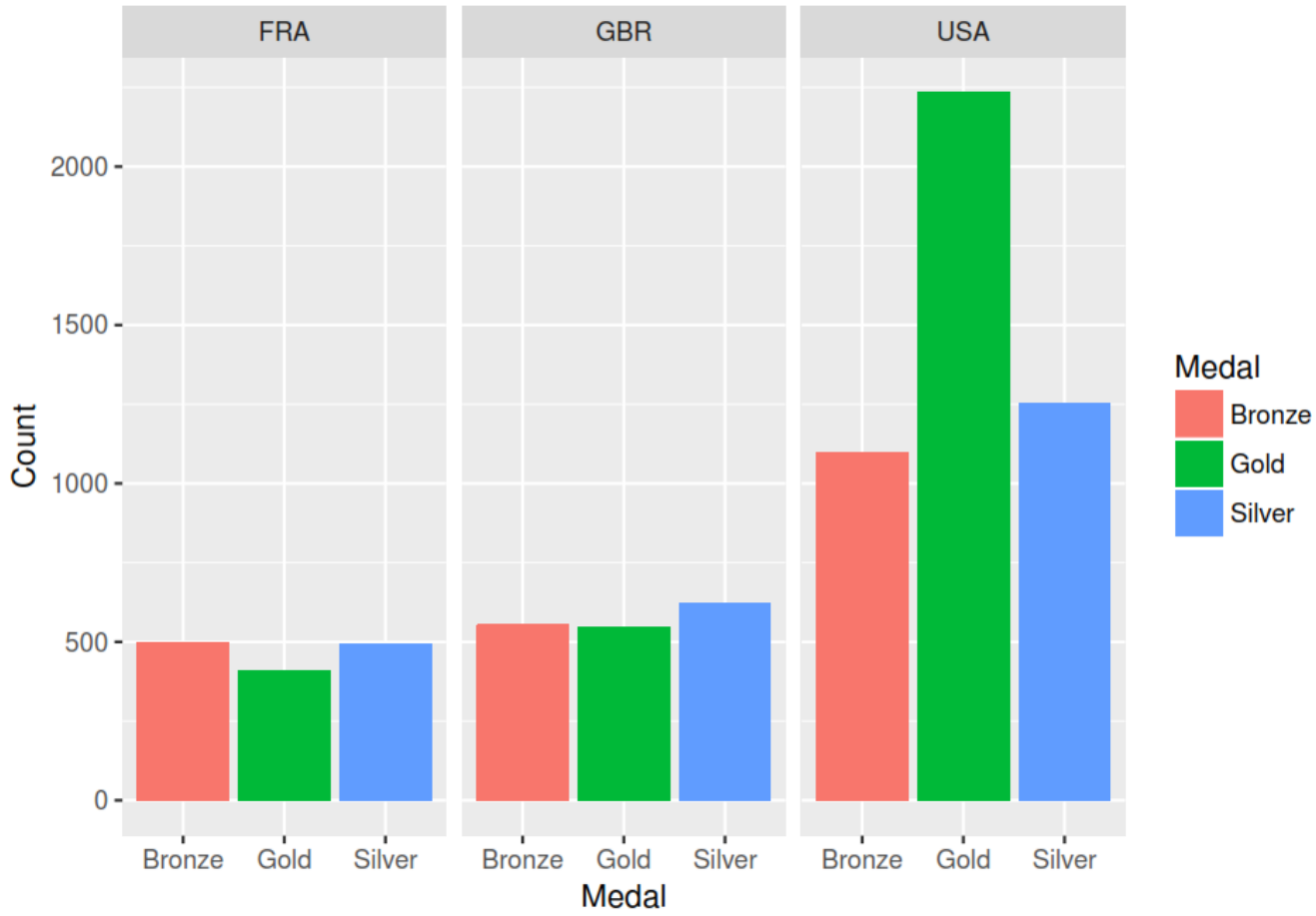
```
summer %>%
  filter(Country %in% c("USA", "FRA", "GBR")) %>%
  group_by(Country, Medal) %>%
  summarise(Count=n()) %>%
  ggplot(aes(x=Medal, y=Count, fill=Medal))+geom_bar(stat="identity")+
  facet_grid(.~Country)
```

- Grammar of data manipulation

DATAMOTUS

Using ggplot with dplyr

```
summer %>%
  filter(Country %in% c("USA", "FRA", "GBR")) %>%
  group_by(Country, Medal) %>%
  summarise(Count=n()) %>%
  ggplot(aes(x=Medal, y=Count, fill=Medal))+geom_bar(stat="identity")+
  facet_grid(.~Country)
```



“‘

• Grammar of data manipulation

DATAMOTUS

Lets summaries gross box office by genre
Then arrange the dataset by descending order on average gross box office

```
movies <- read.csv("movies3.csv")
```

```
sum_movie <- movies %>%
  group_by(genre_first) %>%
  summarise(count=n(), mean = mean(gross_adjusted),
            standard_dev = sd (gross_adjusted)) %>%
  arrange(desc(mean))
head(sum_movie)
```

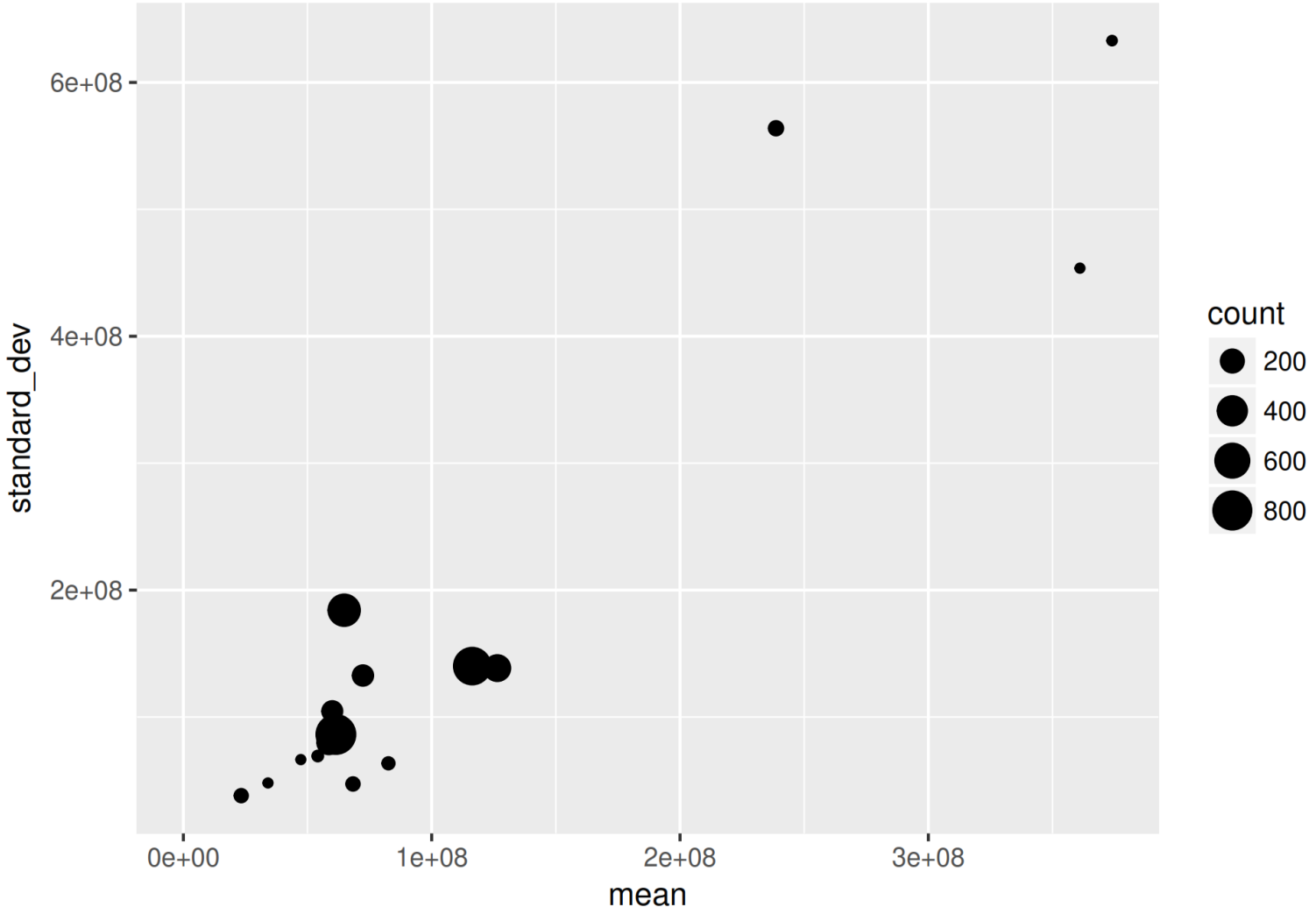
```
## # A tibble: 6 x 4
##   genre_first count       mean standard_dev
##   <fct>       <int>      <dbl>        <dbl>
## 1 Family          3 373974561.   632947270.
## 2 Musical         2 361037936.   453660835.
## 3 Animation      35 238648706.   563888709.
## 4 Adventure     281 126427147.   138554095.
## 5 Action        721 116323930.   140130142.
## 6 Mystery        16  82577251.    63530236.
```

- Grammar of data manipulation

DATAMOTUS

Bubble chart with size as the number of movies in the given category

```
ggplot(sum_movie, aes(x=mean, y=standard_dev, size=count))+geom_point()
```

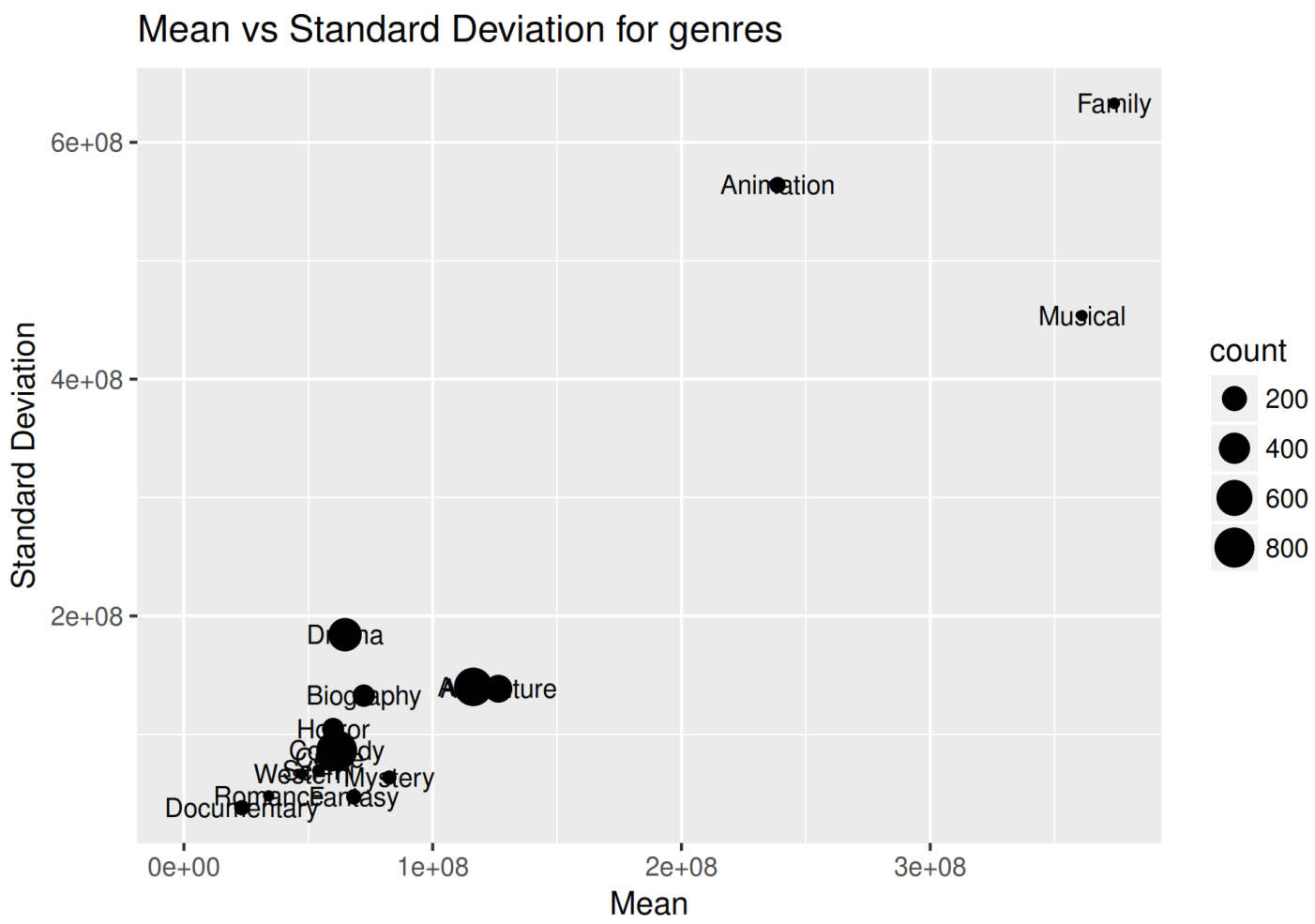


- Grammar of data manipulation

DATAMOTUS

Add labels to the plot using geom_text()

```
ggplot(sum_movie, aes(x=mean, y=standard_dev, size=count))+geom_point()+
  geom_text(aes(label=genre_first), size=3)+
  labs(x="Mean", y="Standard Deviation", title="Mean vs Standard Deviation for genres")
```



Our Contacts

Datamotus LLC,
Republic Armenia, Yerevan 0070
Marshal Baghramyan 40/323w
Web: www.datamotus.com