

CSE 252- Data Science

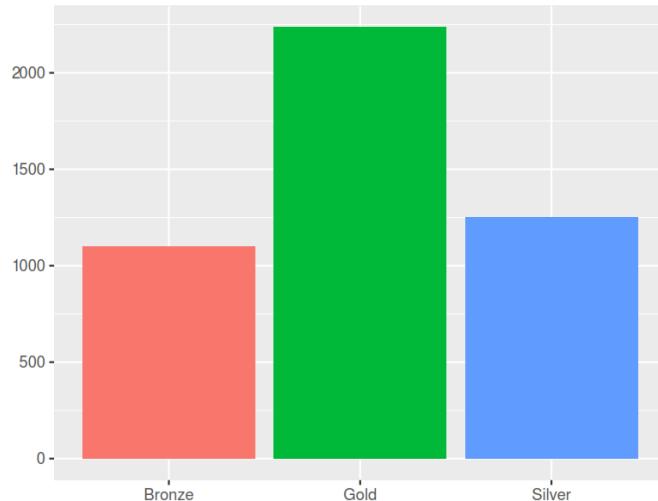
Data Vizualization

Grammar of graphics

- 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.

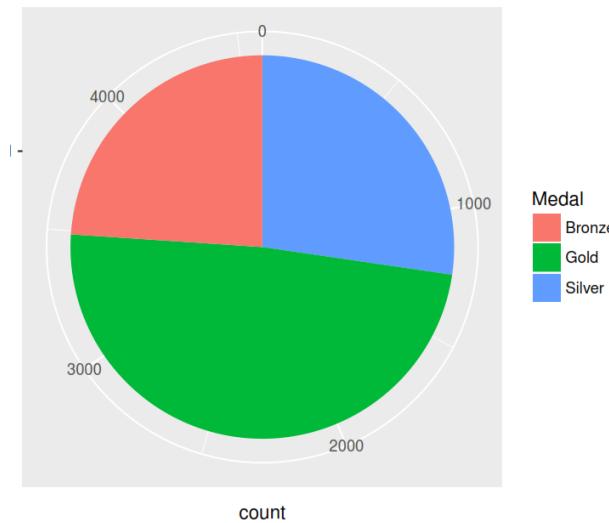
Grammar of graphics

Medals for USA at Summer Olympics games: bar chart

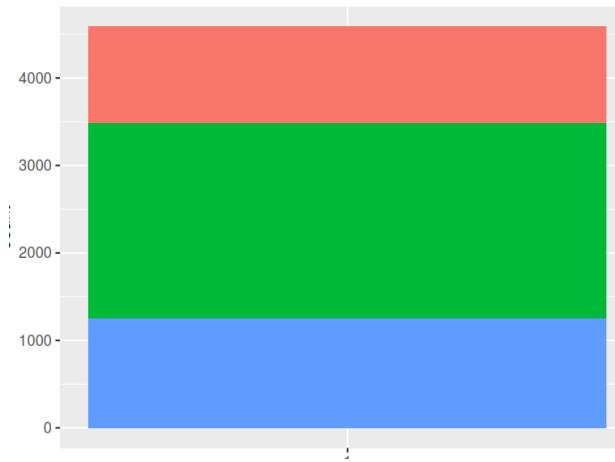


- These three graphs are representing the same information: Distribution of medals for USA team.
- They have common elements and differ only by how the coordinates are constructed

Medals for USA at Summer Olympics games: pie chart



Medals for USA at Summer Olympics games: stacked bar chart



- 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**.

Grammar of graphics: Leland Wilkinson grammar of graphics

- 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.

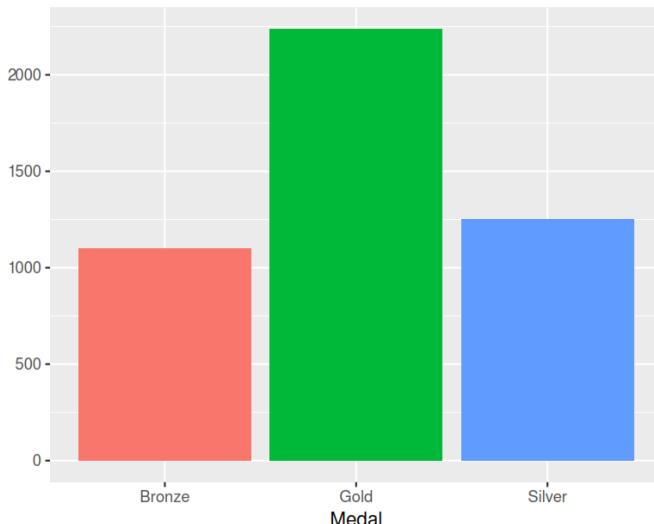
According to Wilkinson, these are components of grammar of graphics

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

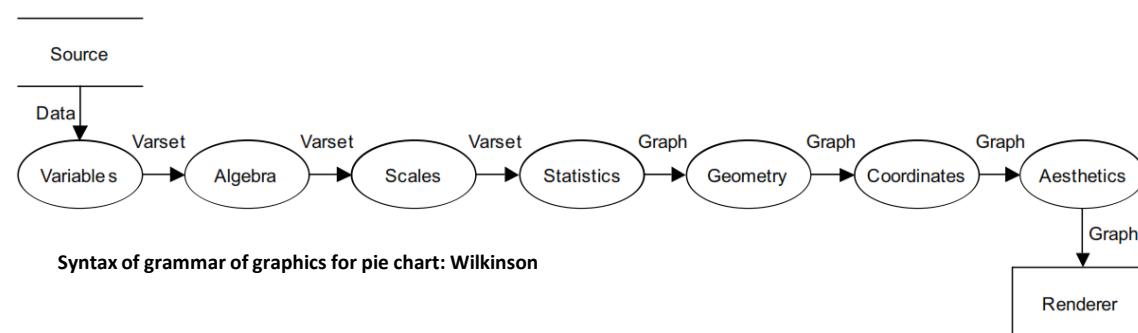
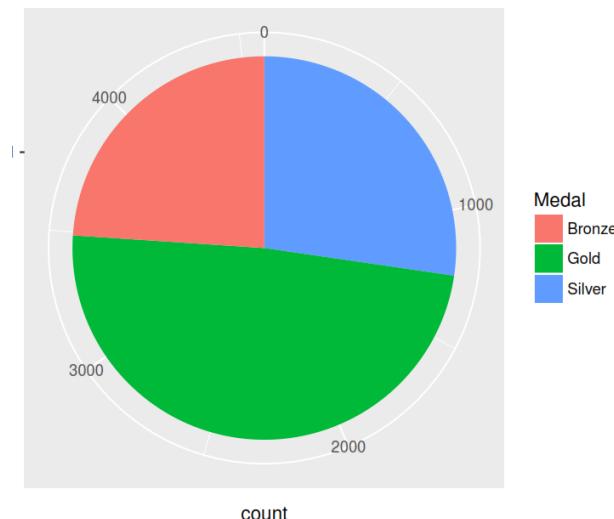
Grammar of graphics

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

Medals for USA at Summer Olympics games: bar chart



Medals for USA at Summer Olympics games: pie chart



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. pp. 4; 176

Iris dataset:

There are three classes of iris flower in the dataset

Iris setosa



Iris versicolor



Iris virginica



Iris dataset: The variables

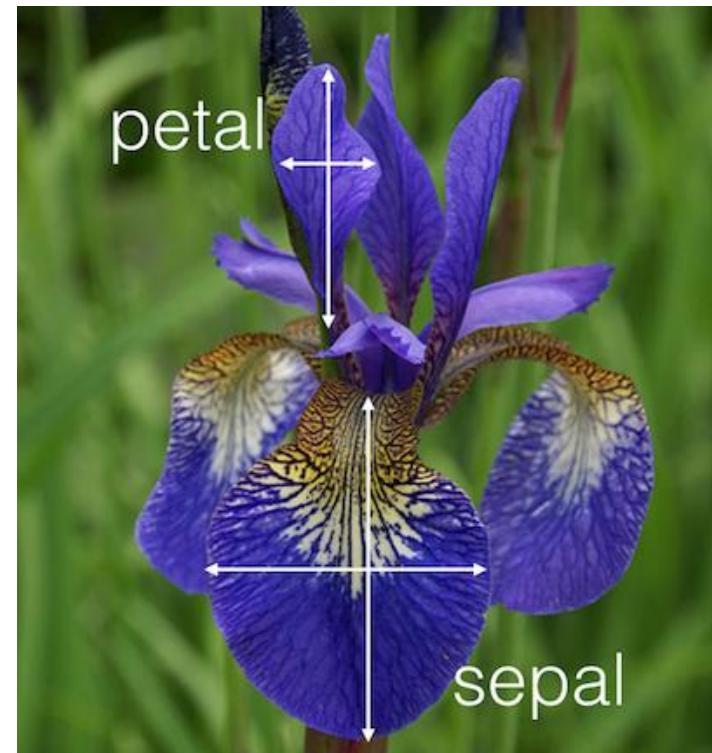
Species – nominal variables with the class of iris – *setosa*, *versicolor*, *virginica*.

Sepal. Length

Sepal.Width

Petal.Length

Petal.Width



Grammar of graphics

```
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

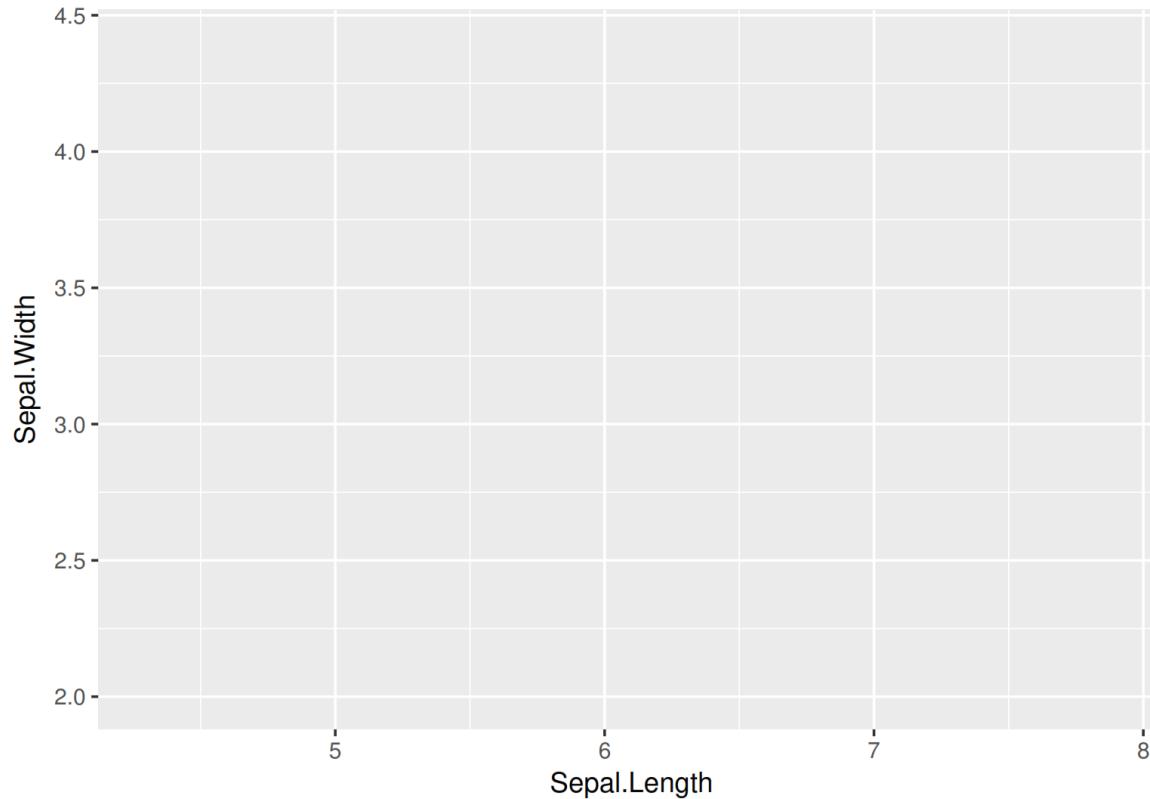
Mapping aesthetics do data

- 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

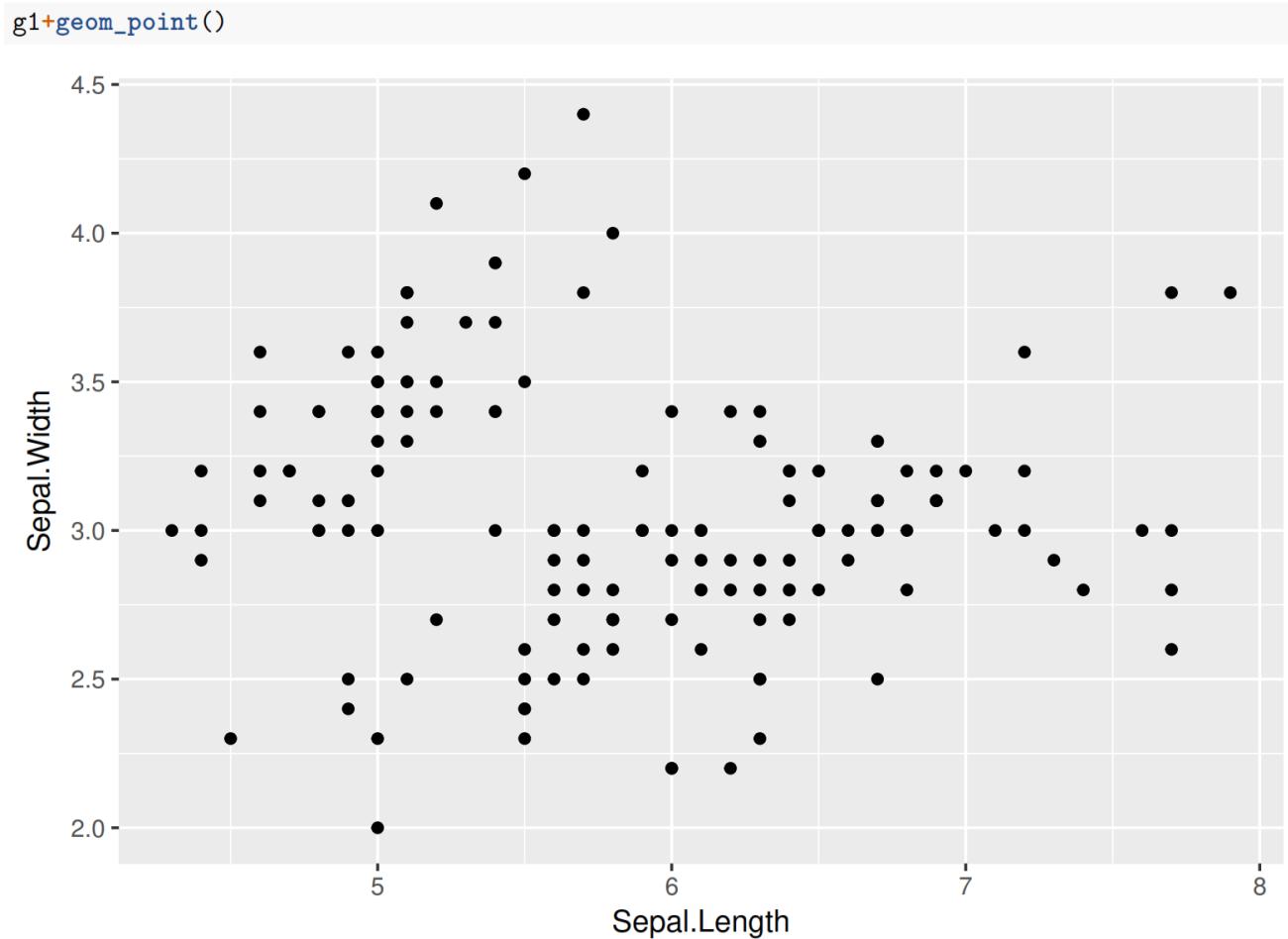
- Create ggplot object first: here you specify your dataset and need to specify aesthetics
- The arguments for ggplot can be omitted if you plan to specify them in layers later

```
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

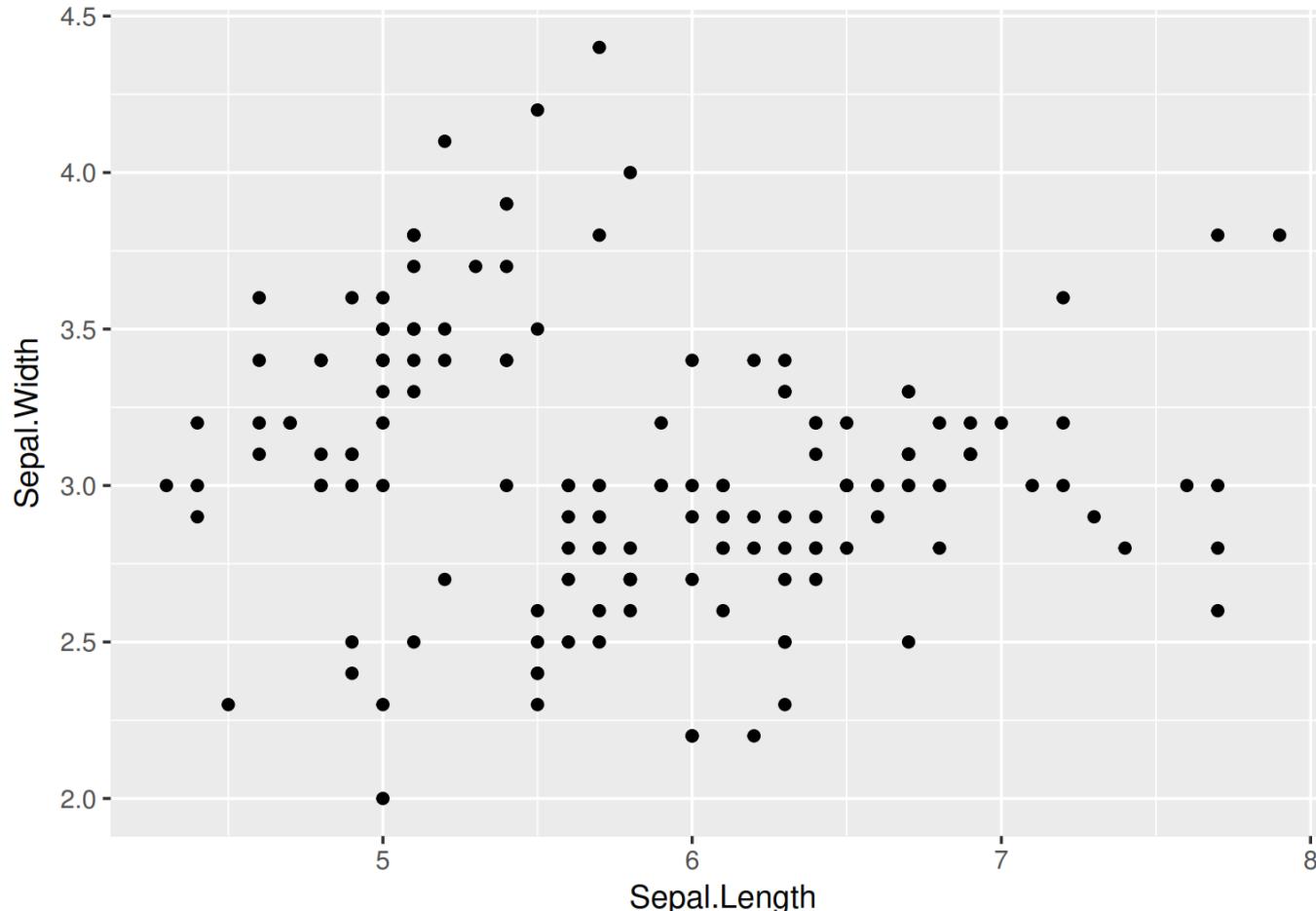


Grammar of graphics

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()
```

Data

Aesthetics

geom (geometric object)



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))
```

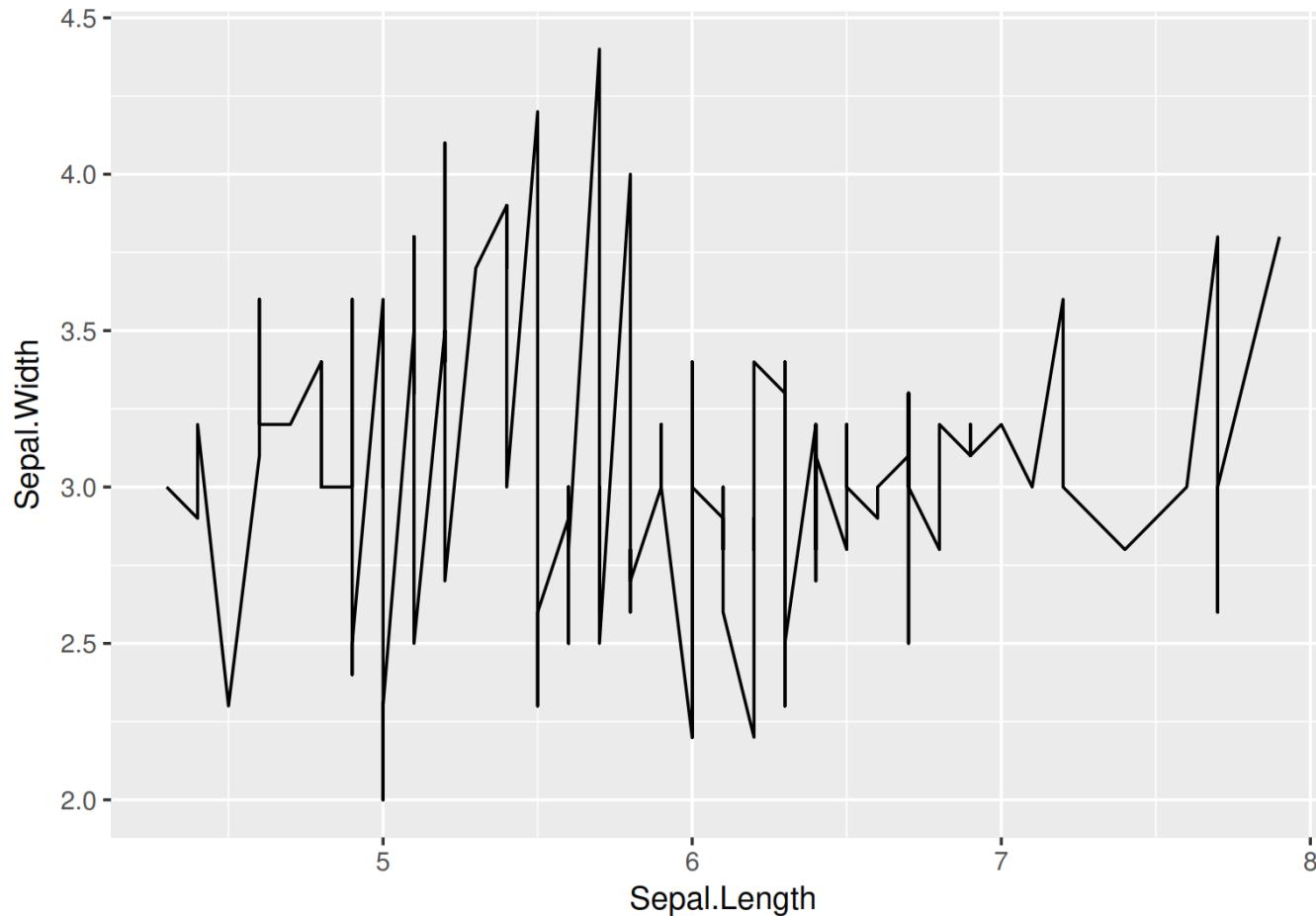
Wrong

```
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()

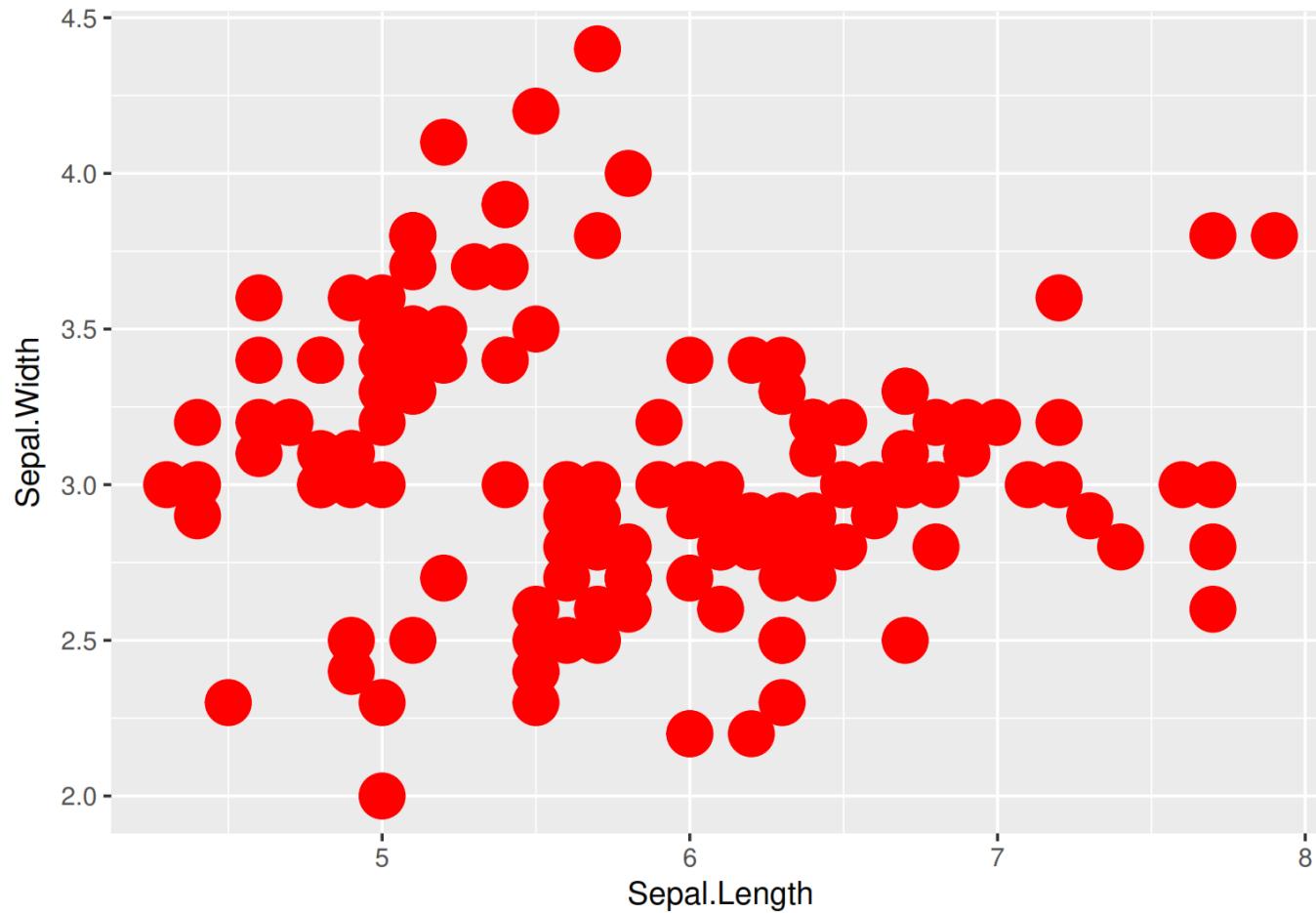
```
g1+geom_line()
```



Grammar of graphics

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

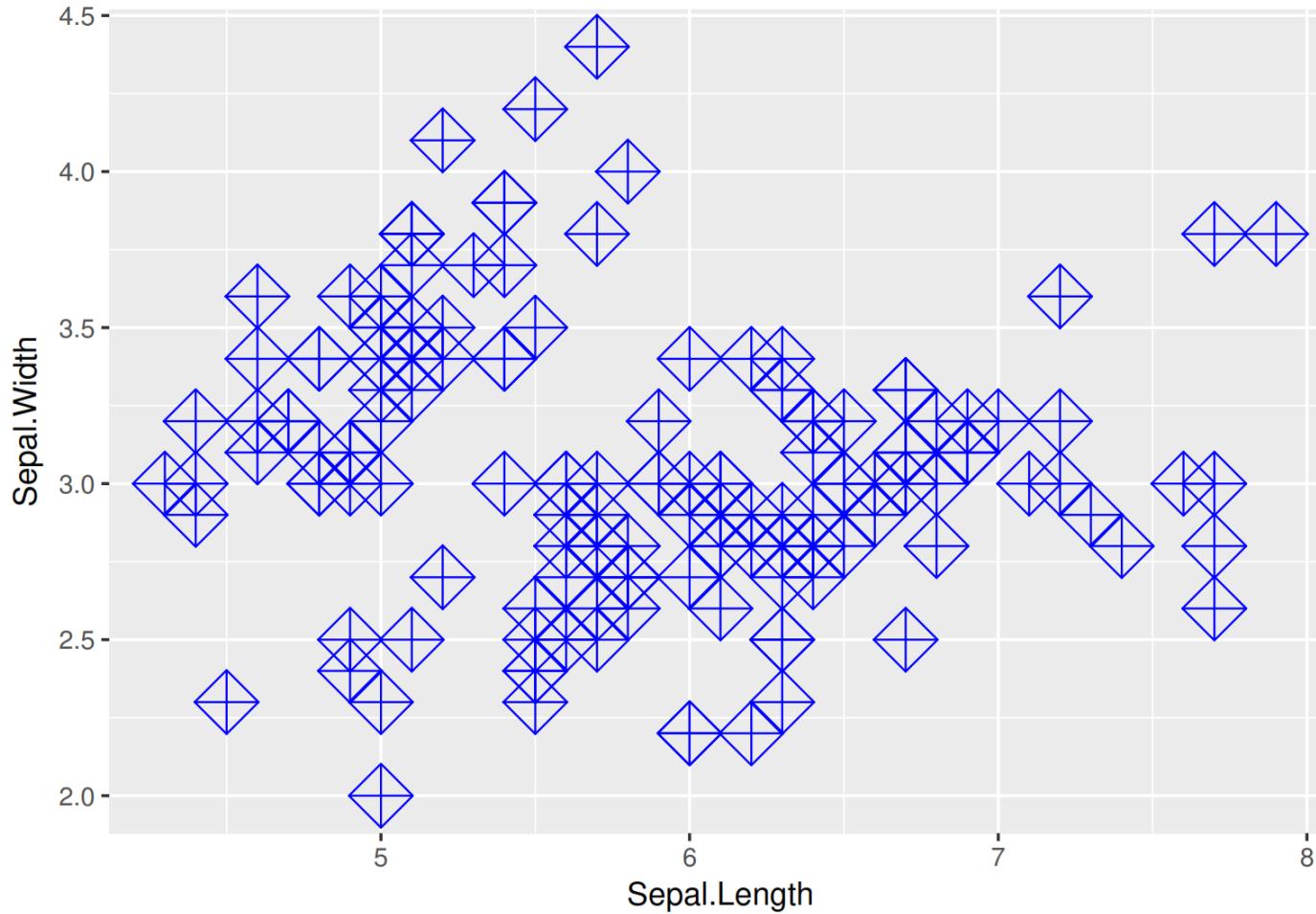
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))+
  geom_point(color="red", size=7)
```



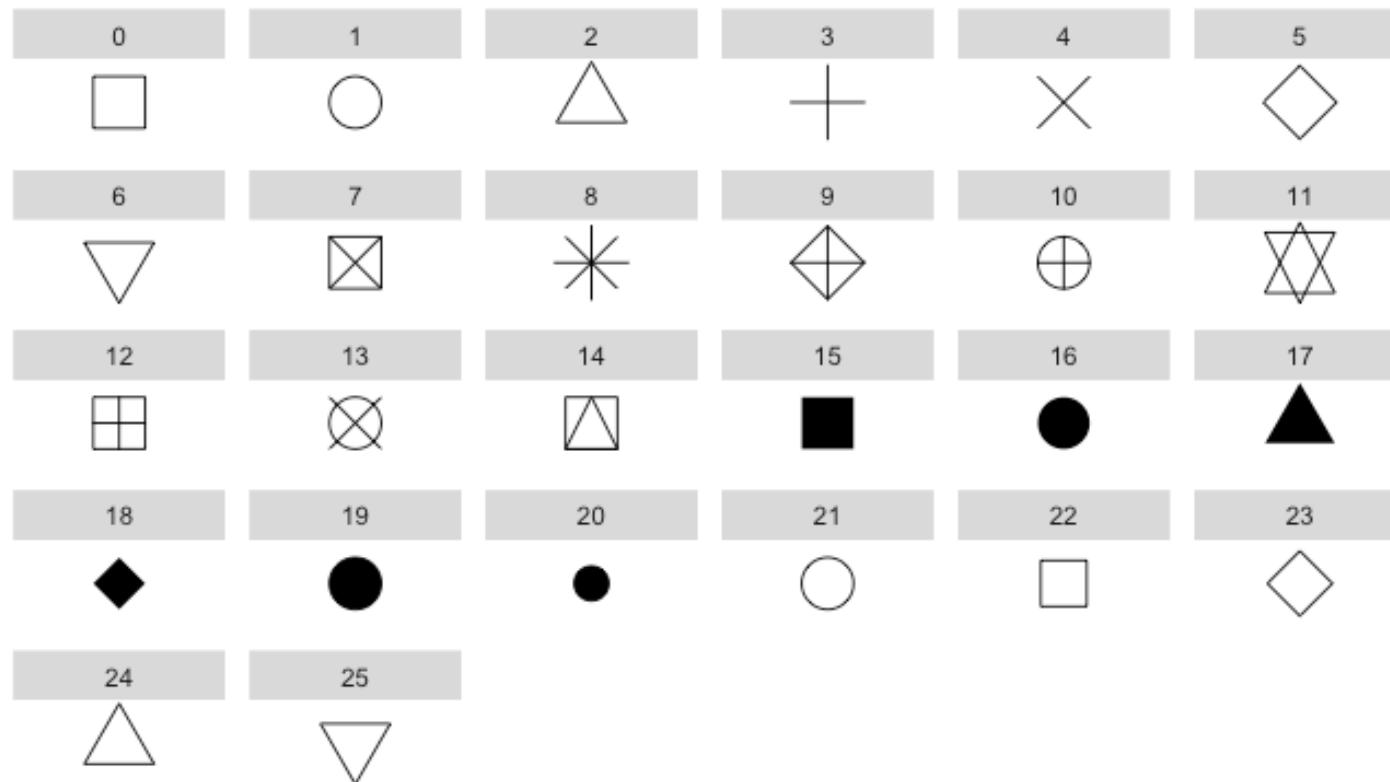
Grammar of graphics

Change also the shape

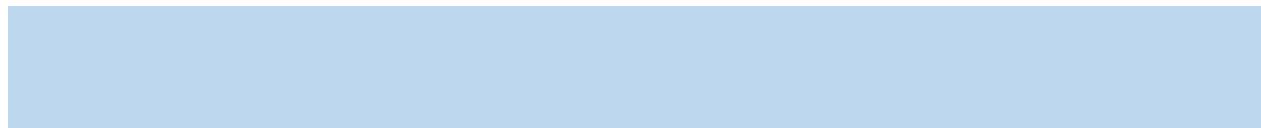
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width)) +  
  geom_point(color="blue", size=7, shape=9)
```



shapes in ggplot2



Colors in R



Colors in R

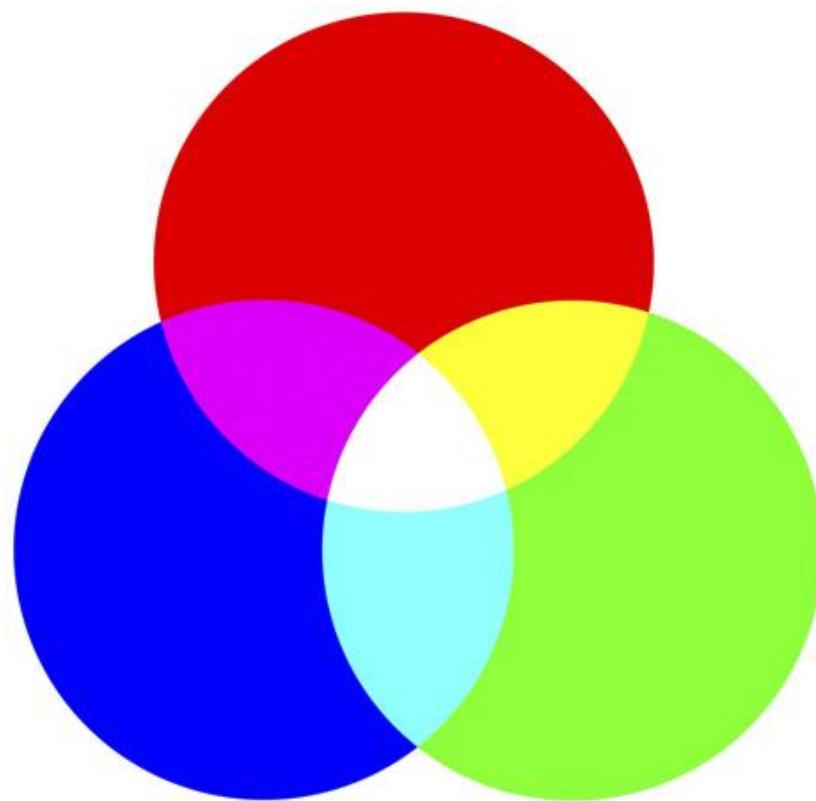
- You can specify colors in R by index, name, hexadecimal, or RGB.
For example col=1, col="white", and col="#FFFFFF" are equivalent.
- You can find names, numbers and codes of colors [here](#)

```
length(colors())  
  
## [1] 657  
  
colors()[1:10]  
  
## [1] "white"         "aliceblue"       "antiquewhite"    "antiquewhite1"  
## [5] "antiquewhite2" "antiquewhite3"  "antiquewhite4"   "aquamarine"  
## [9] "aquamarine1"   "aquamarine2"
```

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

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

- 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%)

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

Look at the first six colors

```
col2rgb(col=1:6)

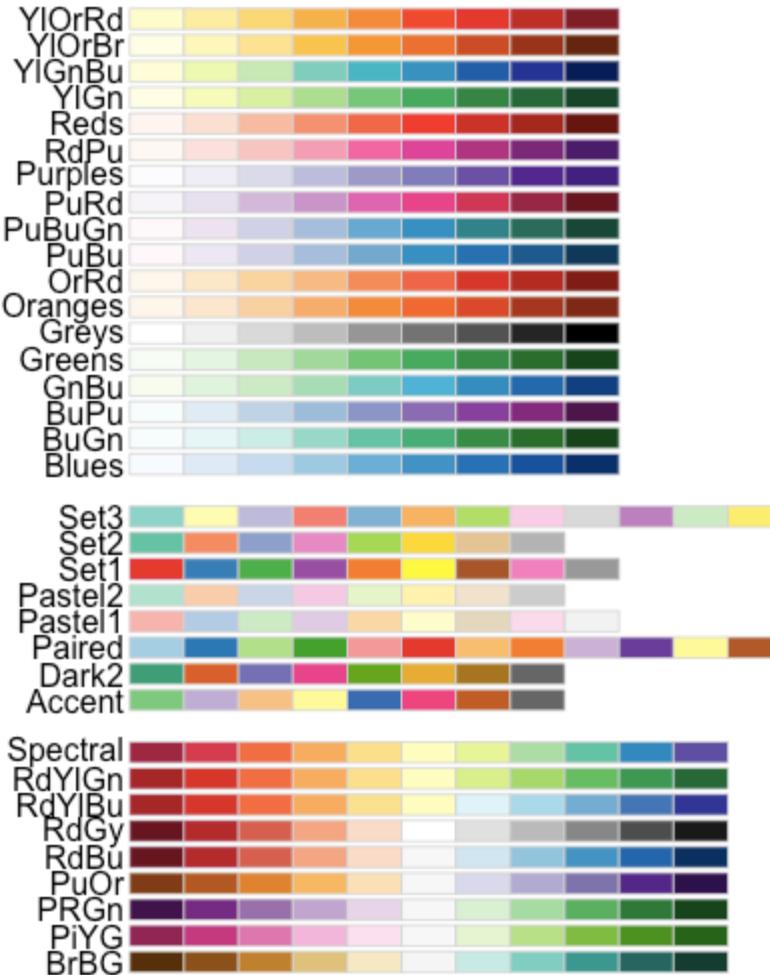
##      [,1] [,2] [,3] [,4] [,5] [,6]
## red     0   255   0   0   0   255
## green   0     0  205   0   255   0
## blue    0     0     0  255  255  255
```

Colors in R

Get RGB and Hex notation for color gold

```
col2rgb("gold")  
  
## [1] [,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 palette

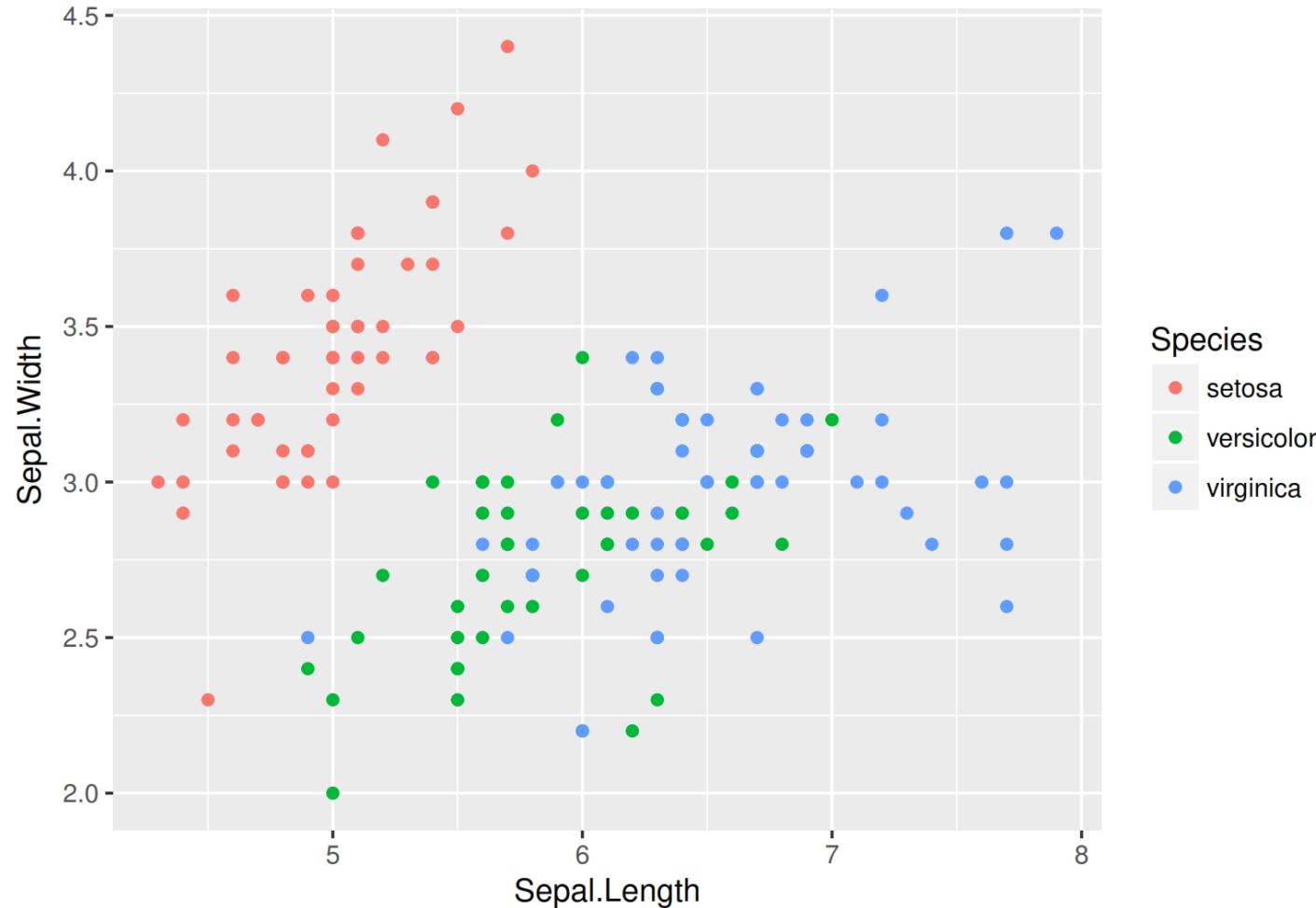
```
library(RColorBrewer)  
brewer.pal(n=5, name = "Set3")  
## [1] "#8DD3C7" "#FFFFB3" "#BEBADA" "#FB8072" "#80B1D3"
```



Grammar of graphics

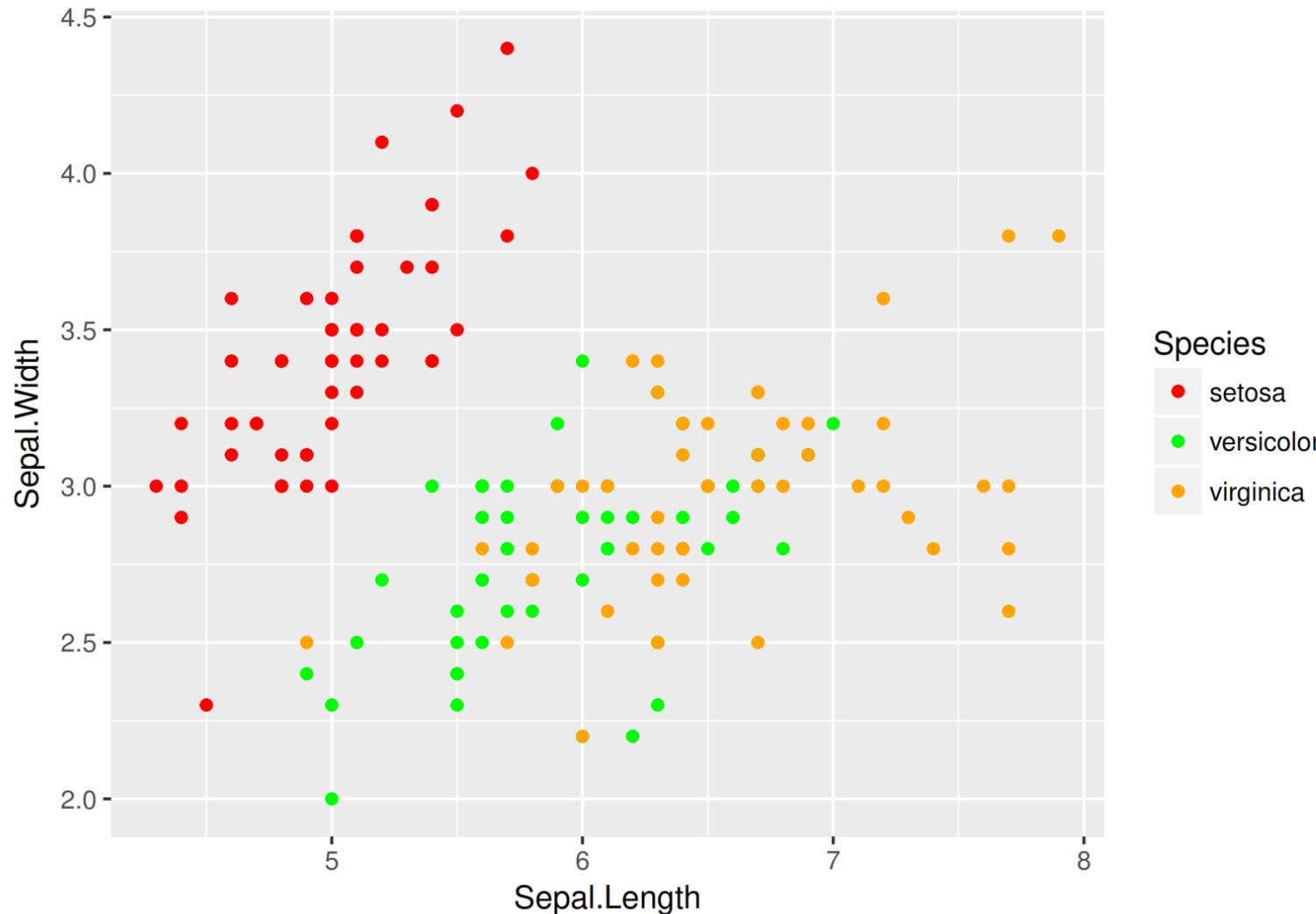
Define color aesthetics as a variable (Color is categorical variable)

```
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

```
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")
```



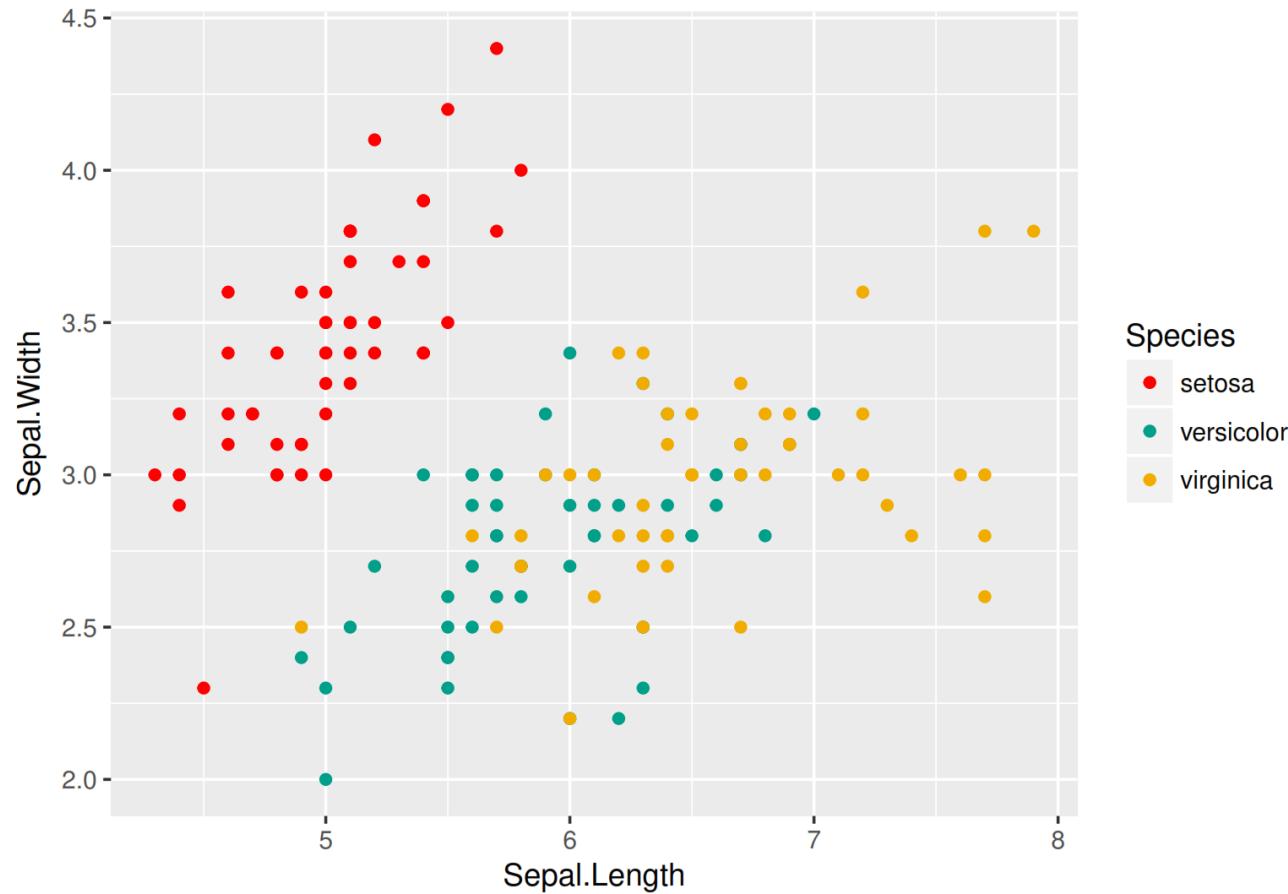
[More on Wes Anderson's colors](#)

Grammar of graphics:

Use the Hex notation and Colors from the "[The Darjeeling Limited](#)"

```
wes_palettes$Darjeeling1
```

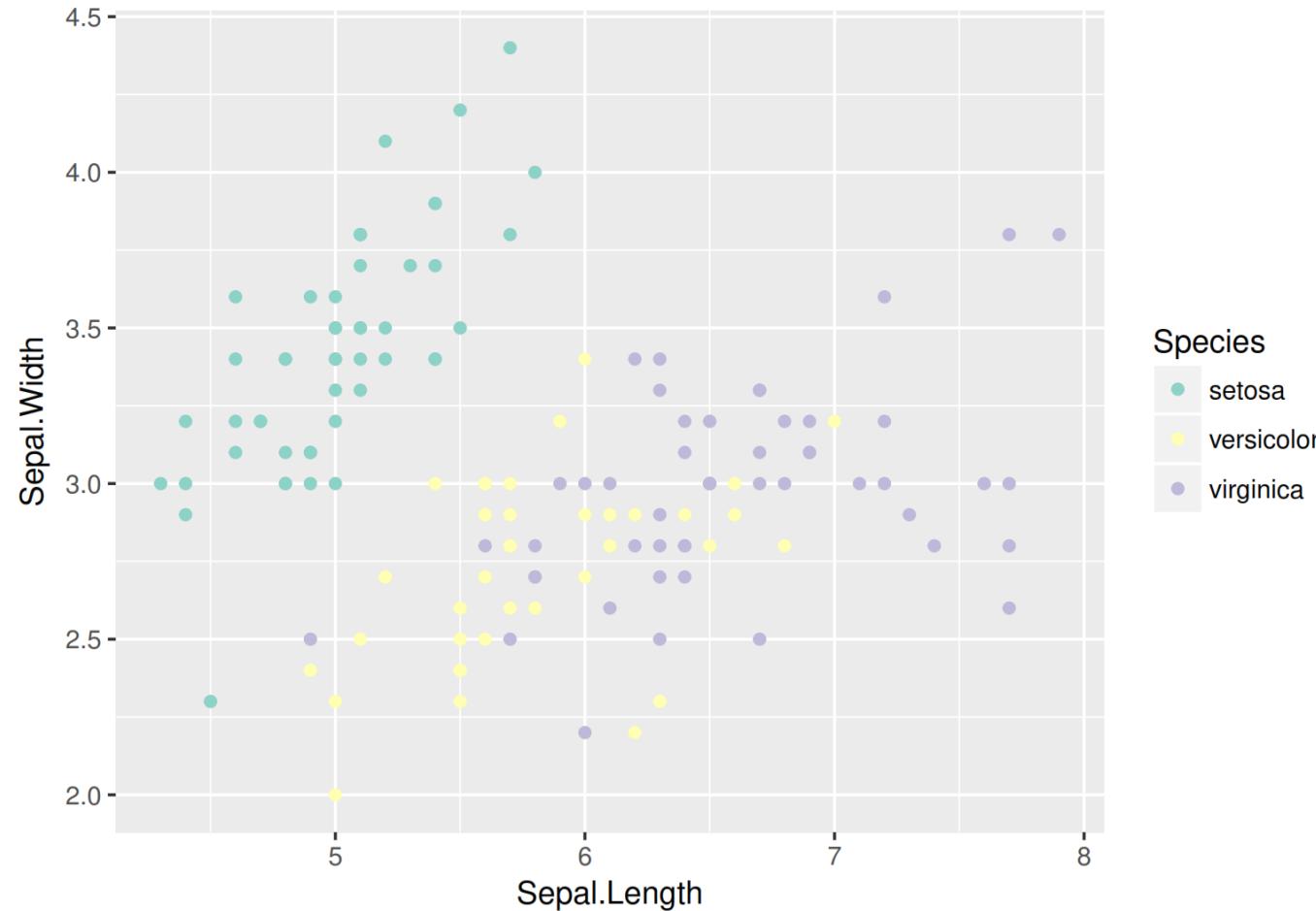
```
## [1] "#FF0000" "#00A08A" "#F2AD00" "#F98400" "#5BBCD6"  
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+ geom_point() +  
  scale_color_manual(values = c("#FF0000", "#00A08A", "#F2AD00"))
```



Grammar of graphics

You can chose colors from the RColorBrewer palettes

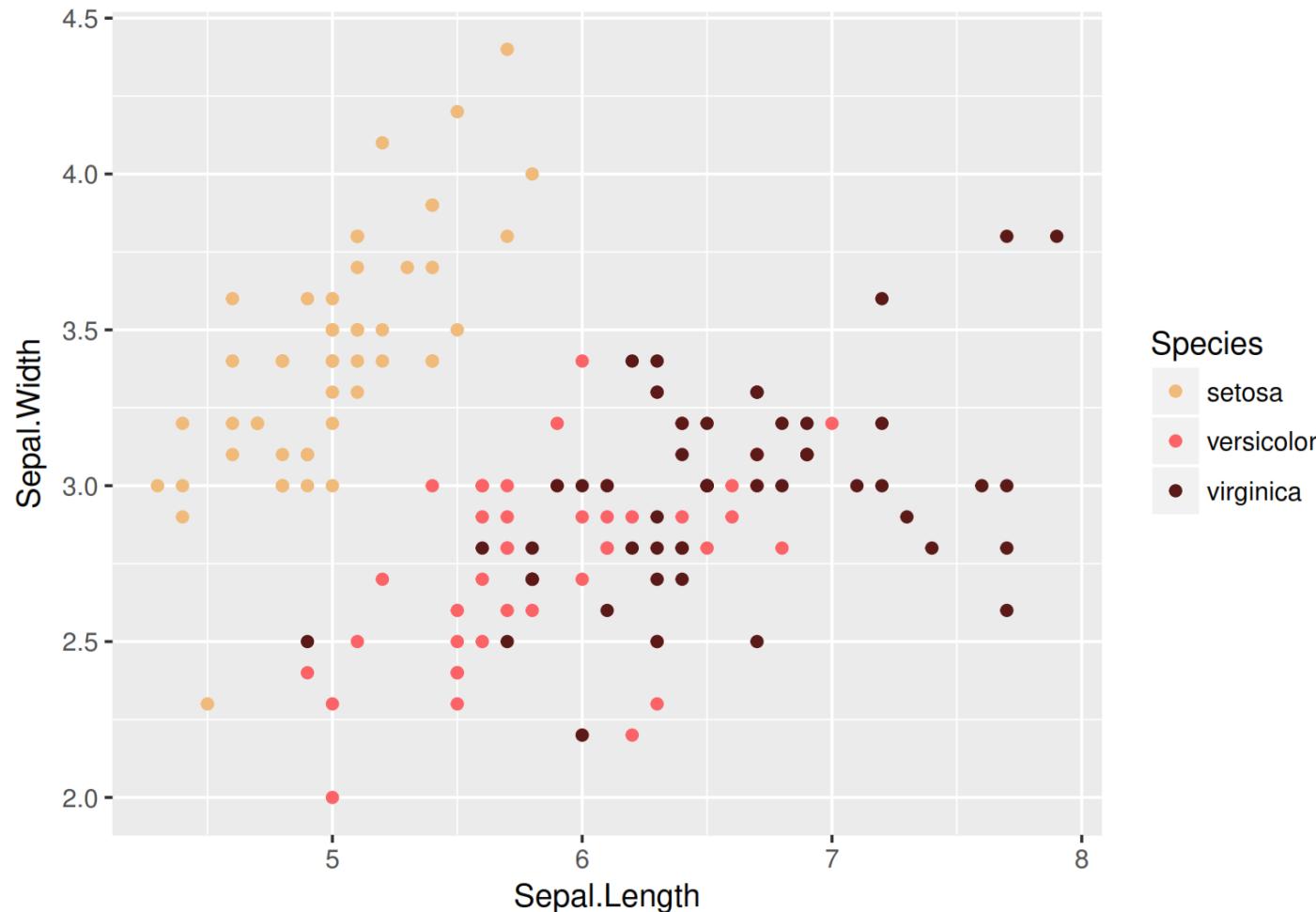
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point() +  
  scale_color_brewer(palette="Set3")
```



Grammar of graphics

wesanderson package allows to directly access the palettes as well, colors from [The Grand Budapest Hotel](#)

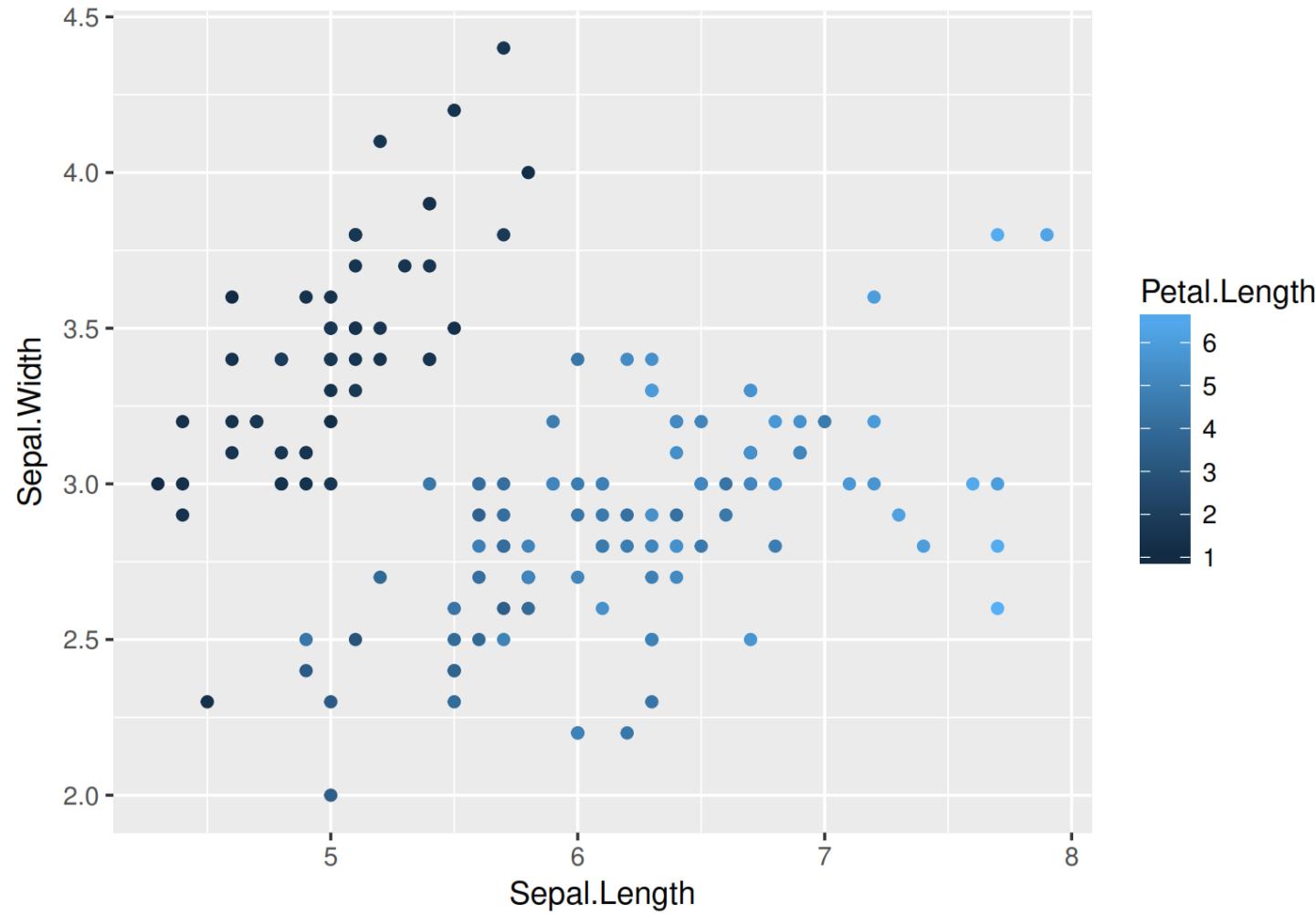
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point() +  
  scale_color_manual(values=wes_palette("GrandBudapest1"))
```



Grammar of graphics

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

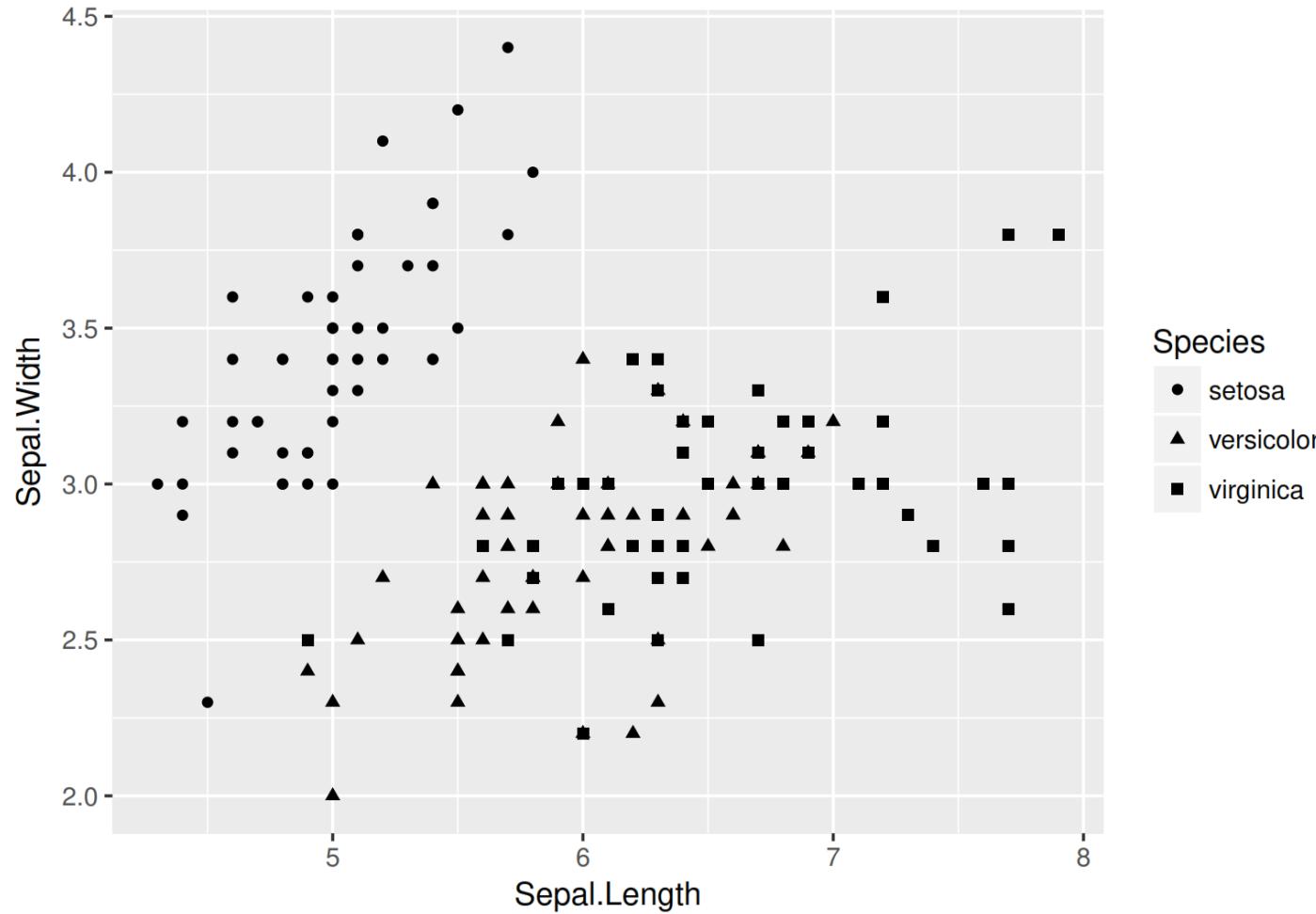
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Petal.Length))+geom_point()
```



Grammar of graphics

Species here determine the shape of the points

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



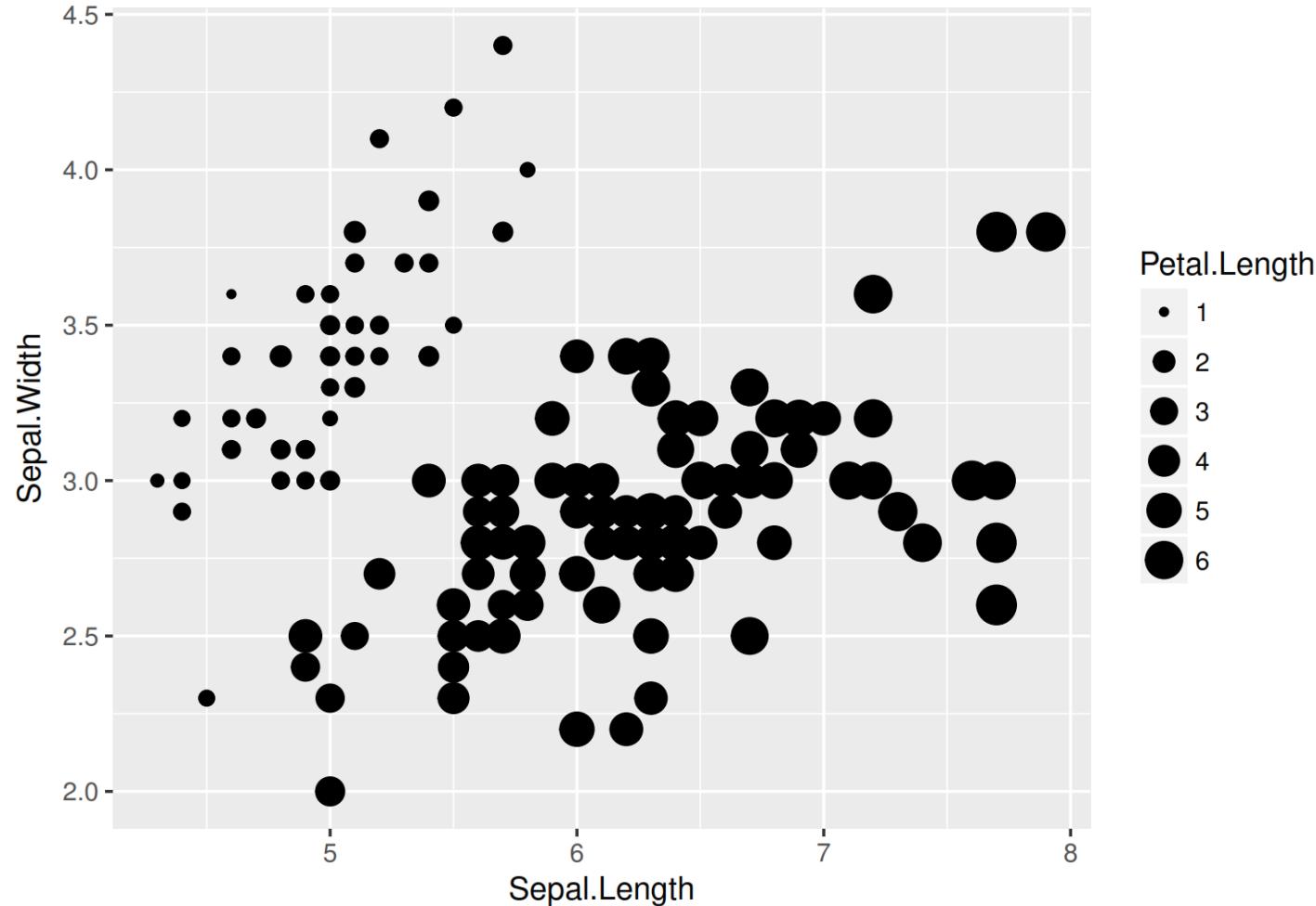
Species

- setosa
- versicolor
- virginica

Grammar of graphics

Bubble chart: continuous variable mapped as size aesthetics

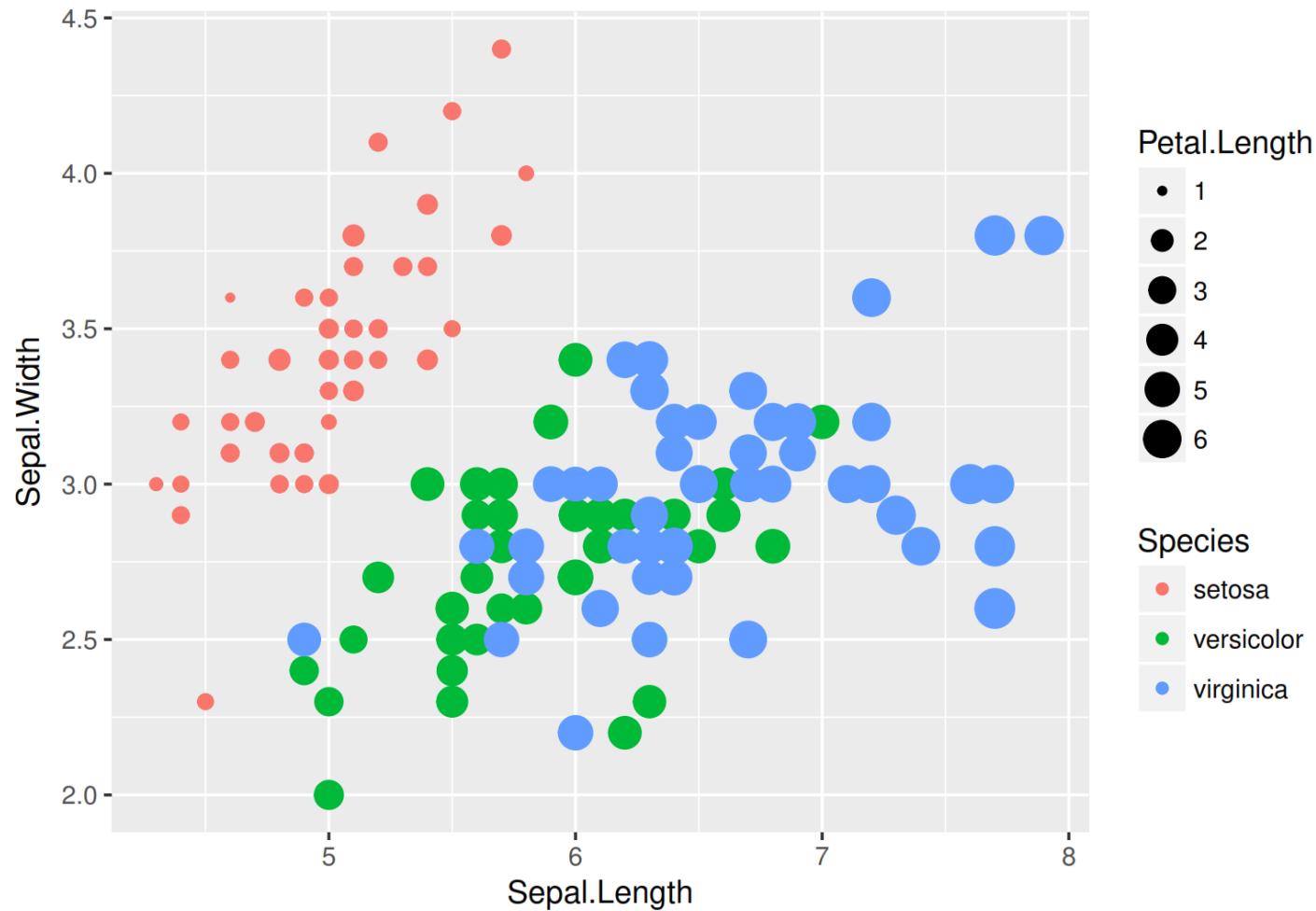
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length))+  
  geom_point()
```



Grammar of graphics

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

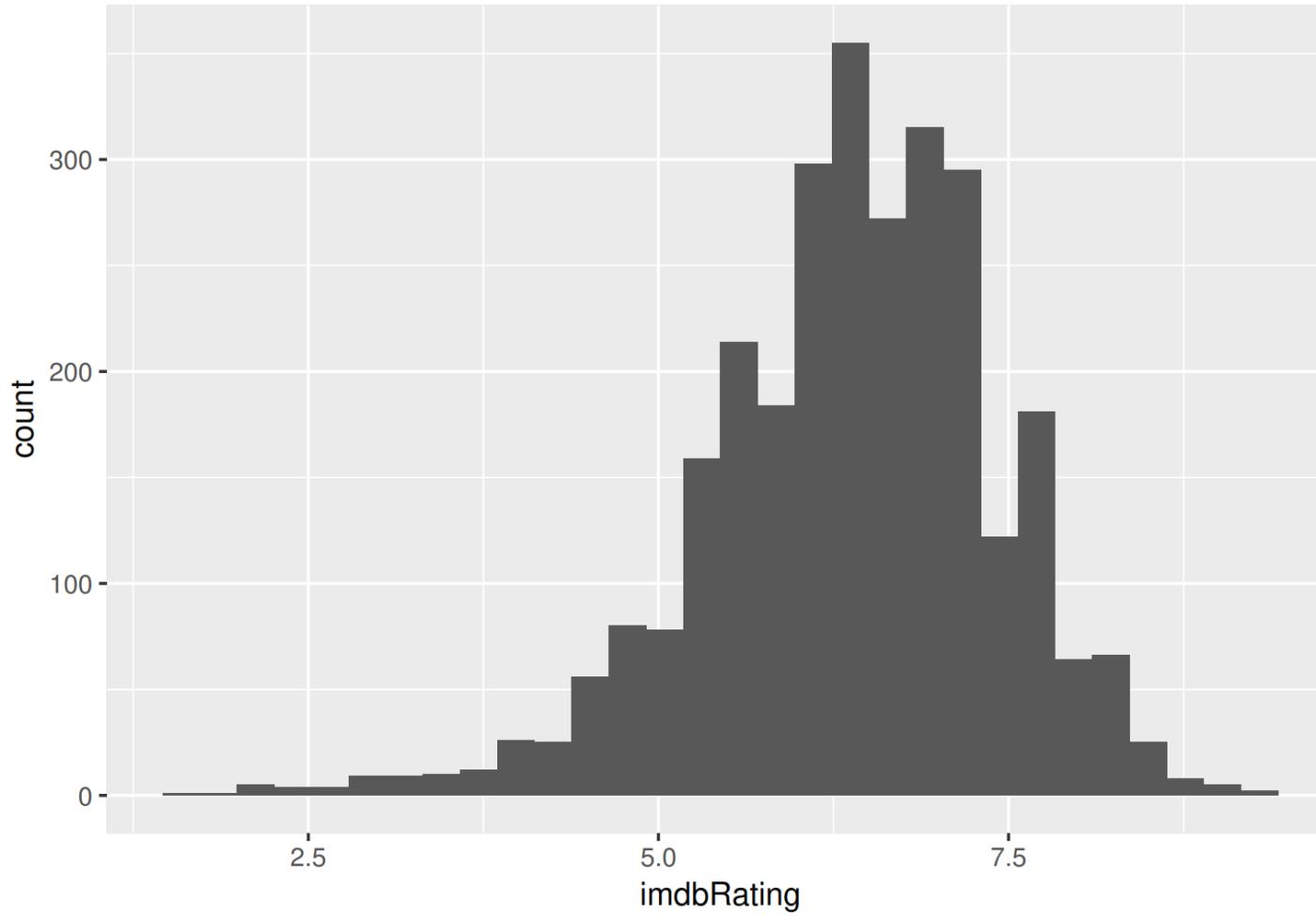
```
movies <- read.csv("movies3.csv")
colnames(movies)

## [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

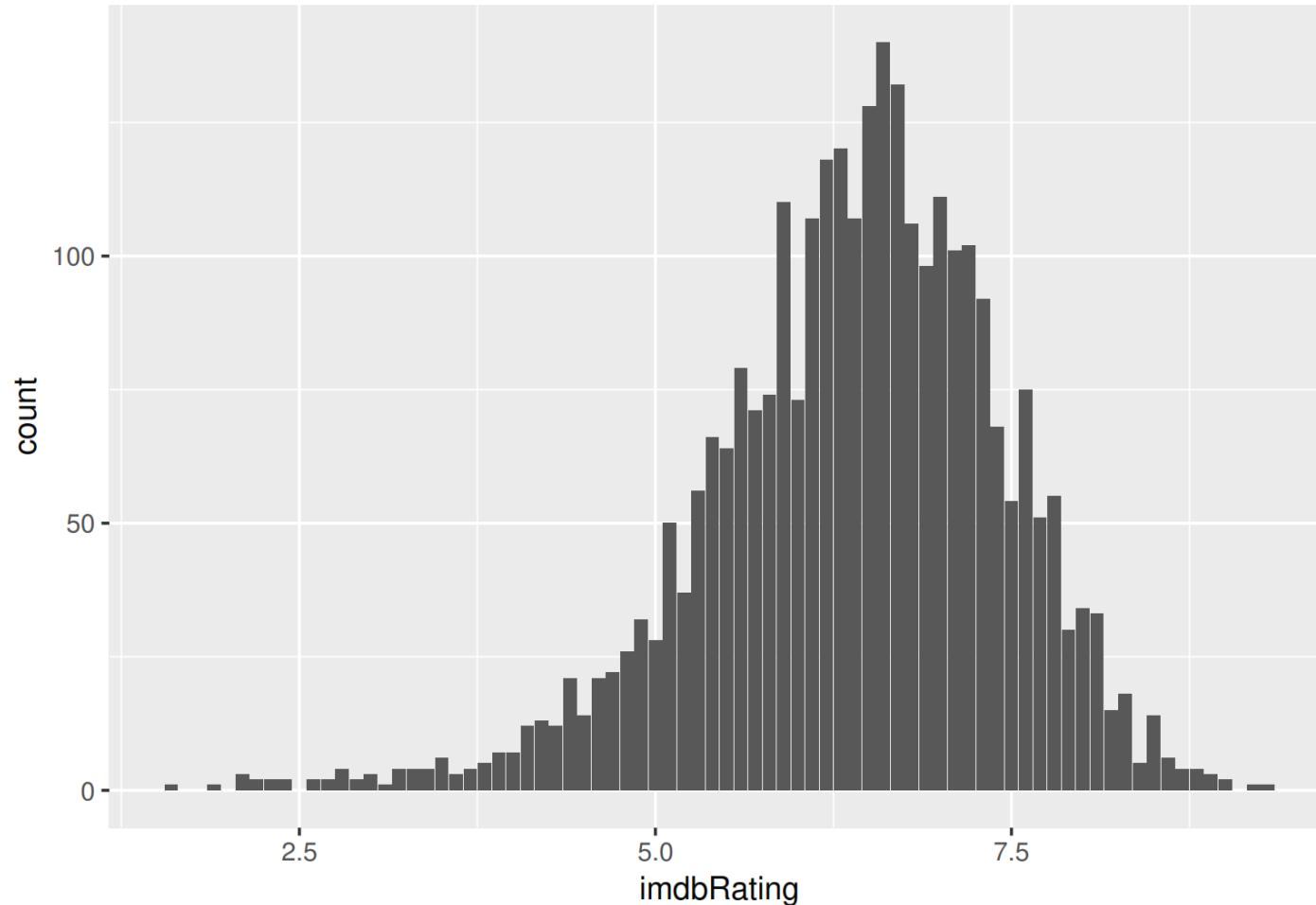
```
ggplot(movies, aes(x=imdbRating)) + geom_histogram()
```



Grammar of graphics

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

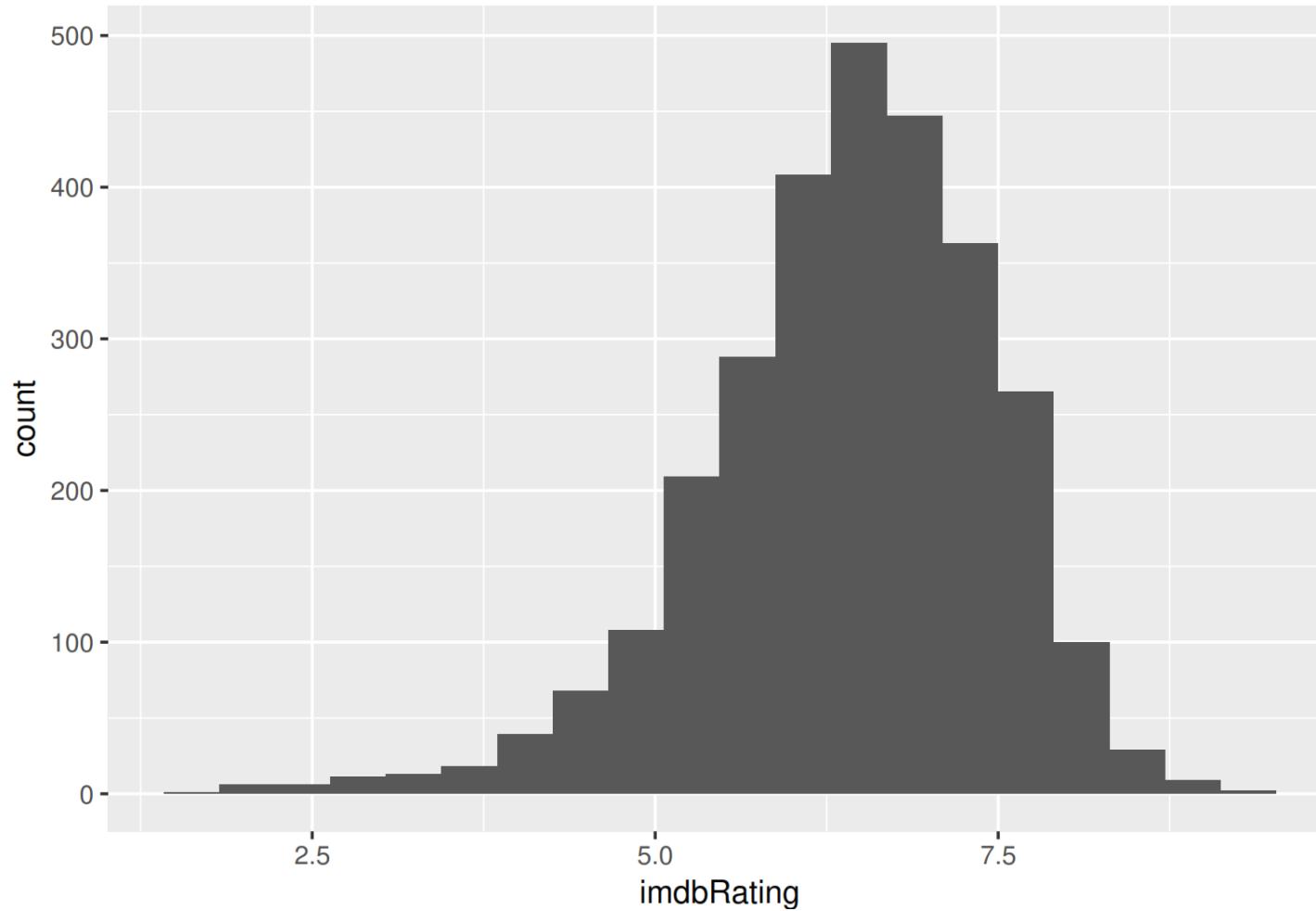
```
ggplot(movies, aes(x=imdbRating)) + geom_bar()
```



Grammar of graphics

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

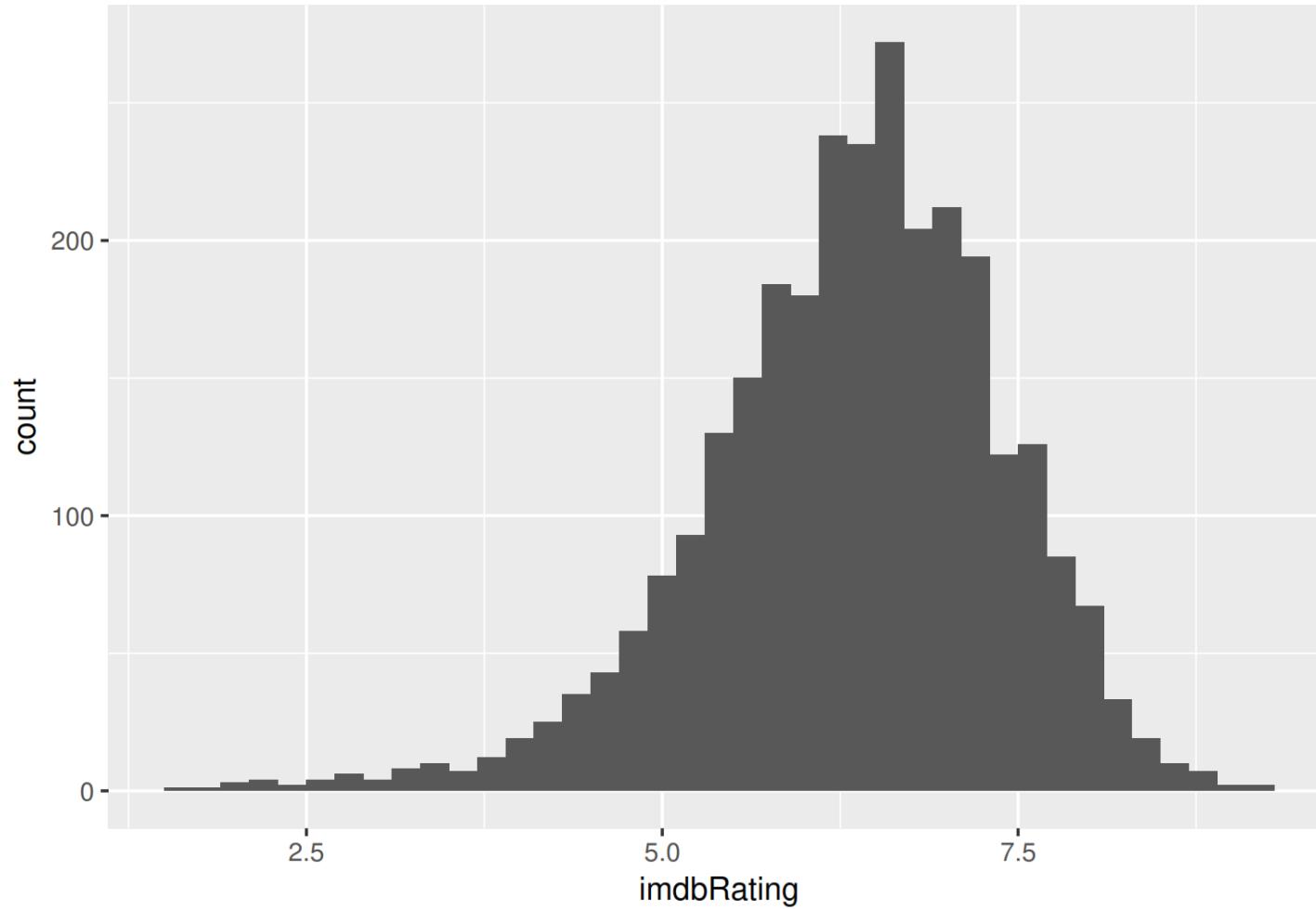
```
ggplot(movies, aes(x=imdbRating)) + geom_histogram(bins = 20)
```



Grammar of graphics

Or play with the ***binwidth***

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



Visualizing categorical data



Grammar of graphics: Visualizing categorical data

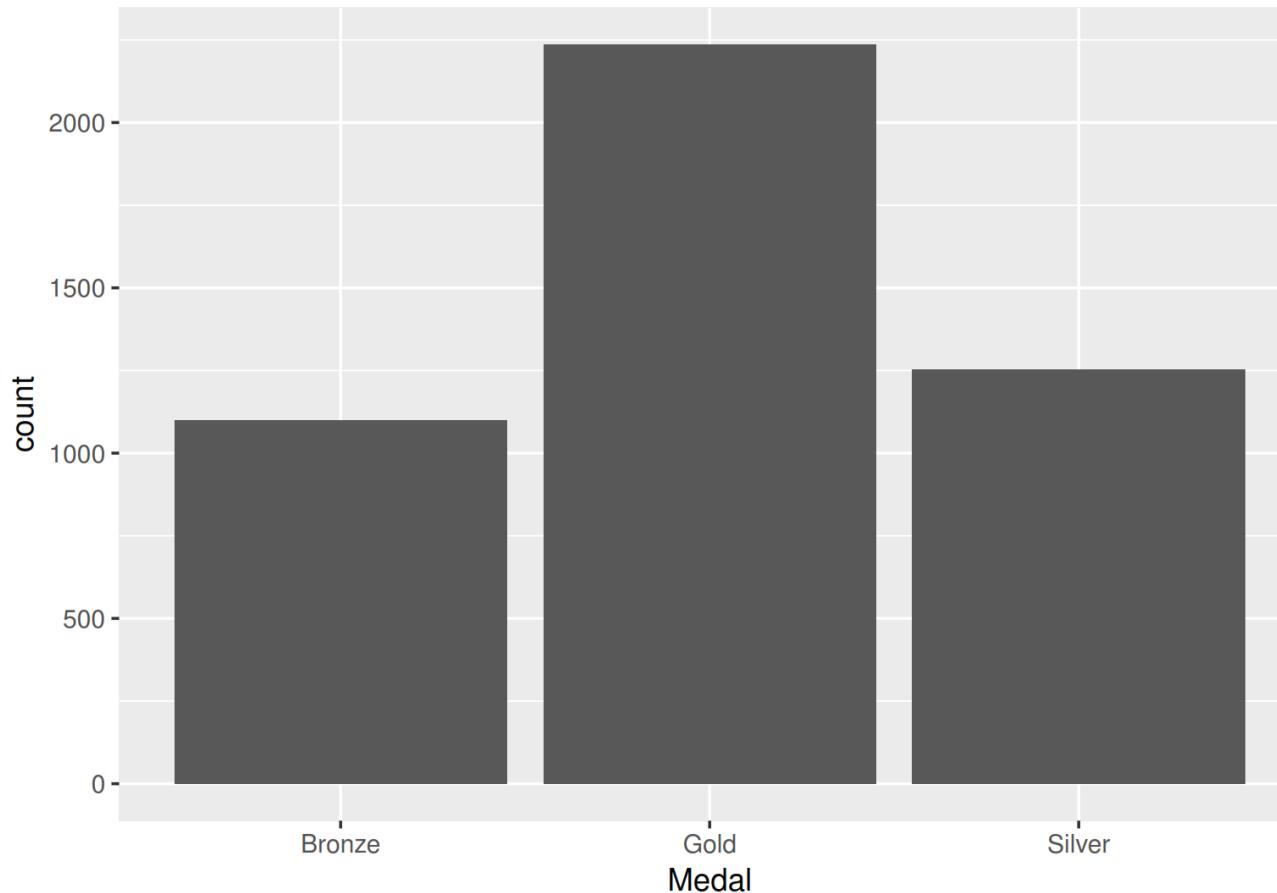
- A **bar chart** or **bar graph** is a chart or graph that presents **categorical data** with **rectangular bars** with **heights** or **lengths** proportional to the values that they represent. The bars can be plotted vertically or horizontally.
- A bar graph shows comparisons among **discrete categories**.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

Barchart 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

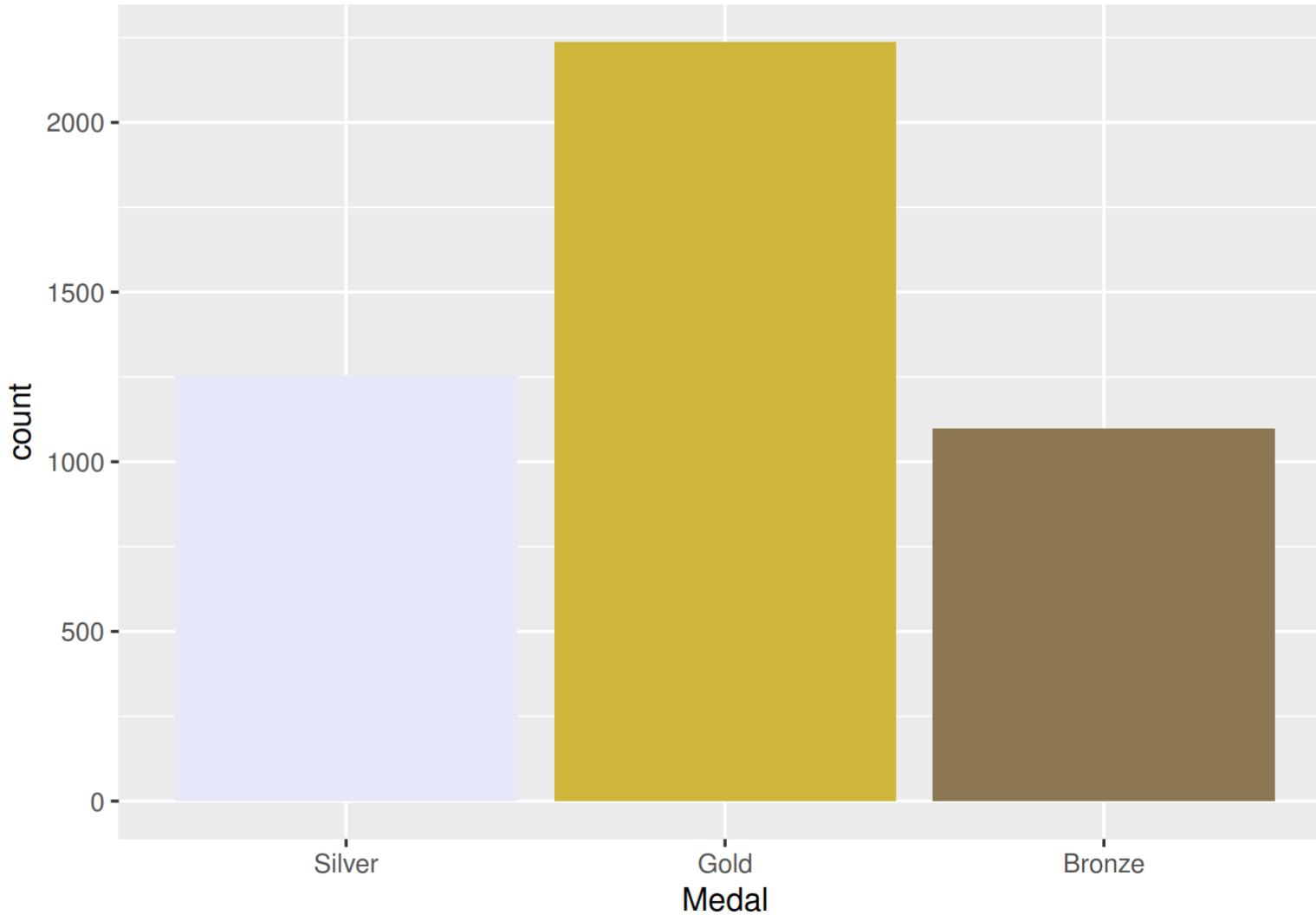
- Lets make the chart more visually appealing
- First relevel the factor levels for medals

```
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

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



Grammar of graphics: Visualizing categorical data

- Build already aggregated dataframe 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

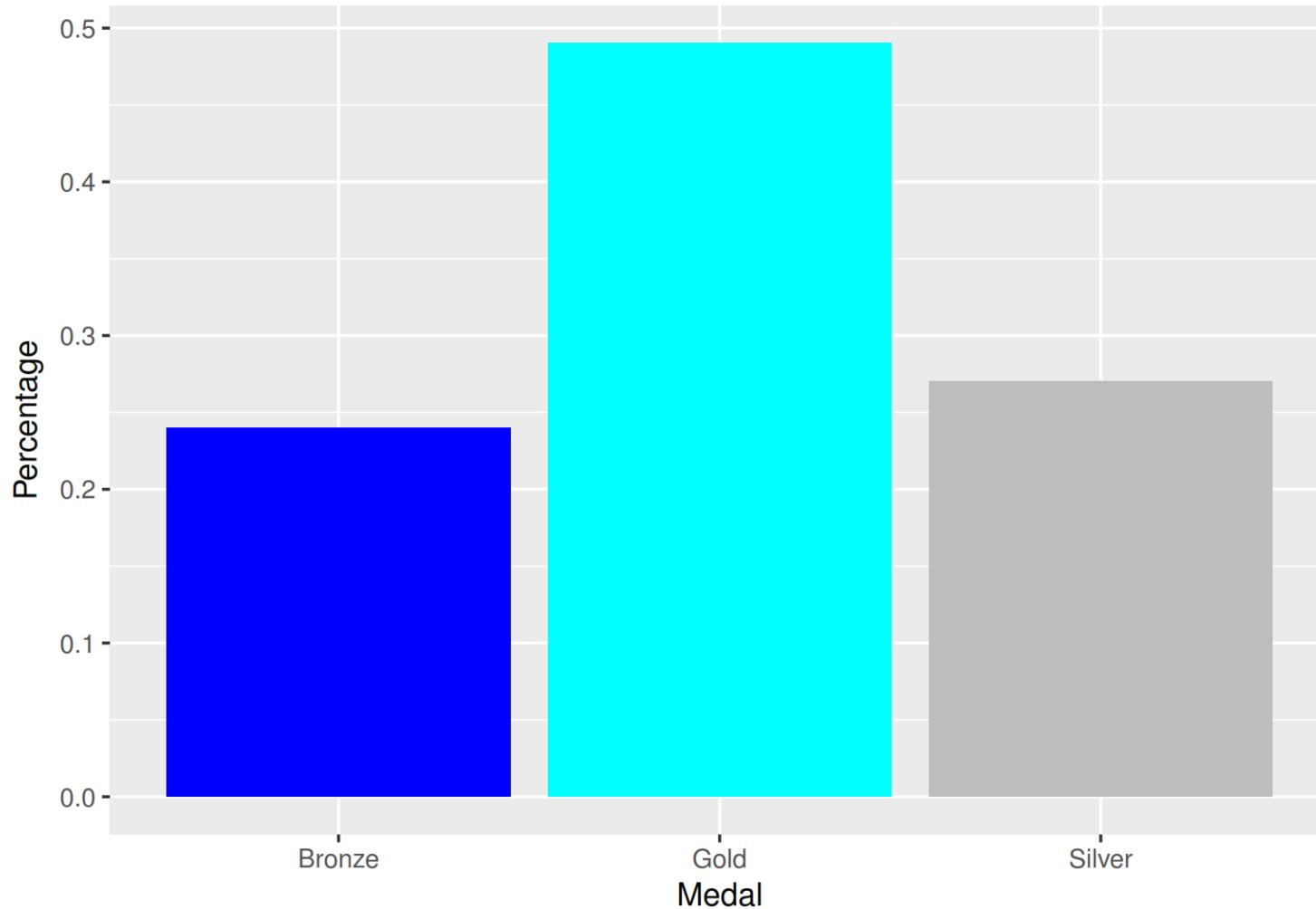
```
> ggplot(usa_medals, aes(x=Medal, y=Percentage))+geom_bar()  
Error: stat_count() must not be used with a y aesthetic.
```

- Another component of grammars 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

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

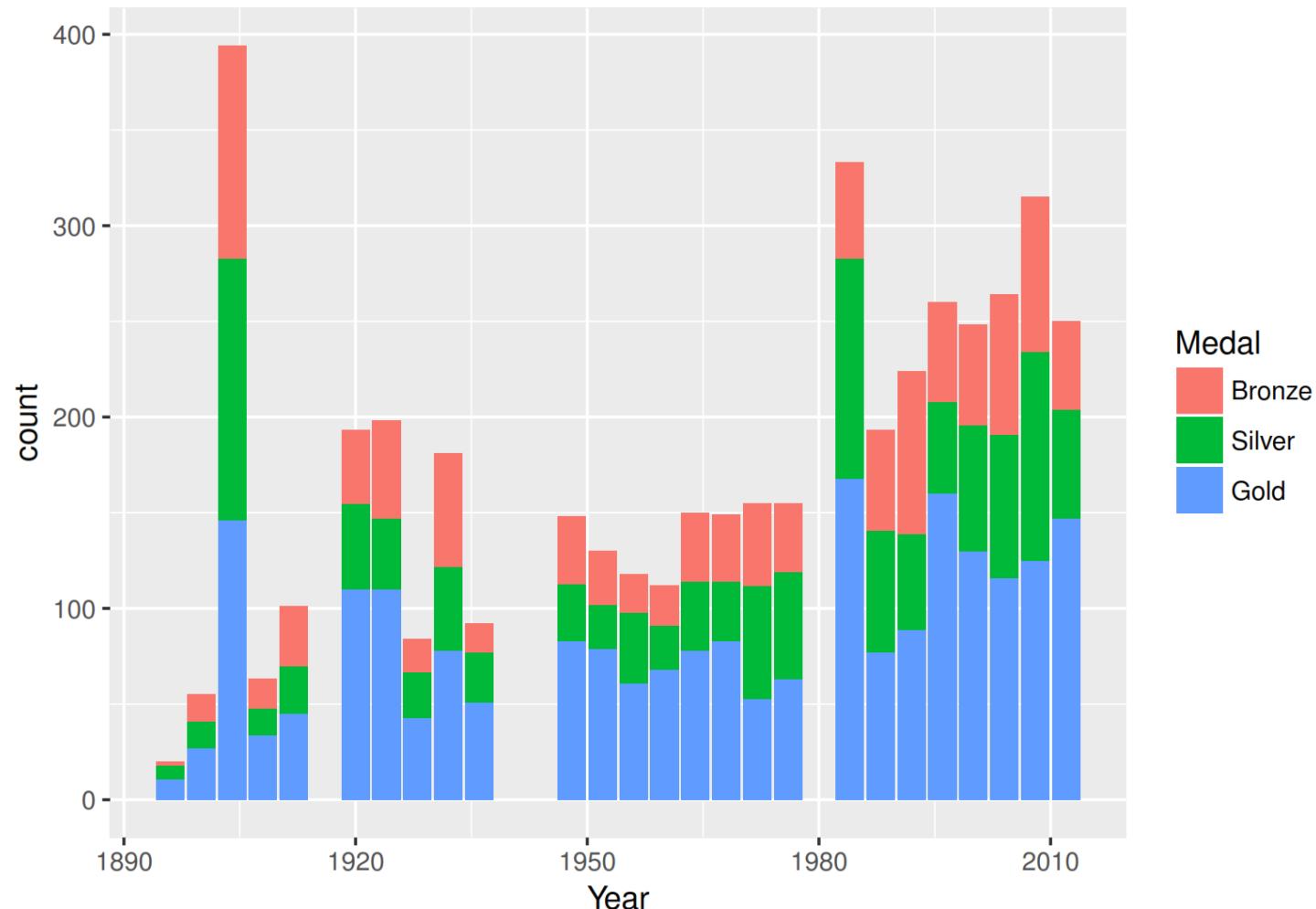
```
ggplot(usa_medals, aes(x=Medal, y=Percentage)) + geom_bar(stat="identity", fill=c(4,5,8))
```



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

change position fill to get 100% Bar chart

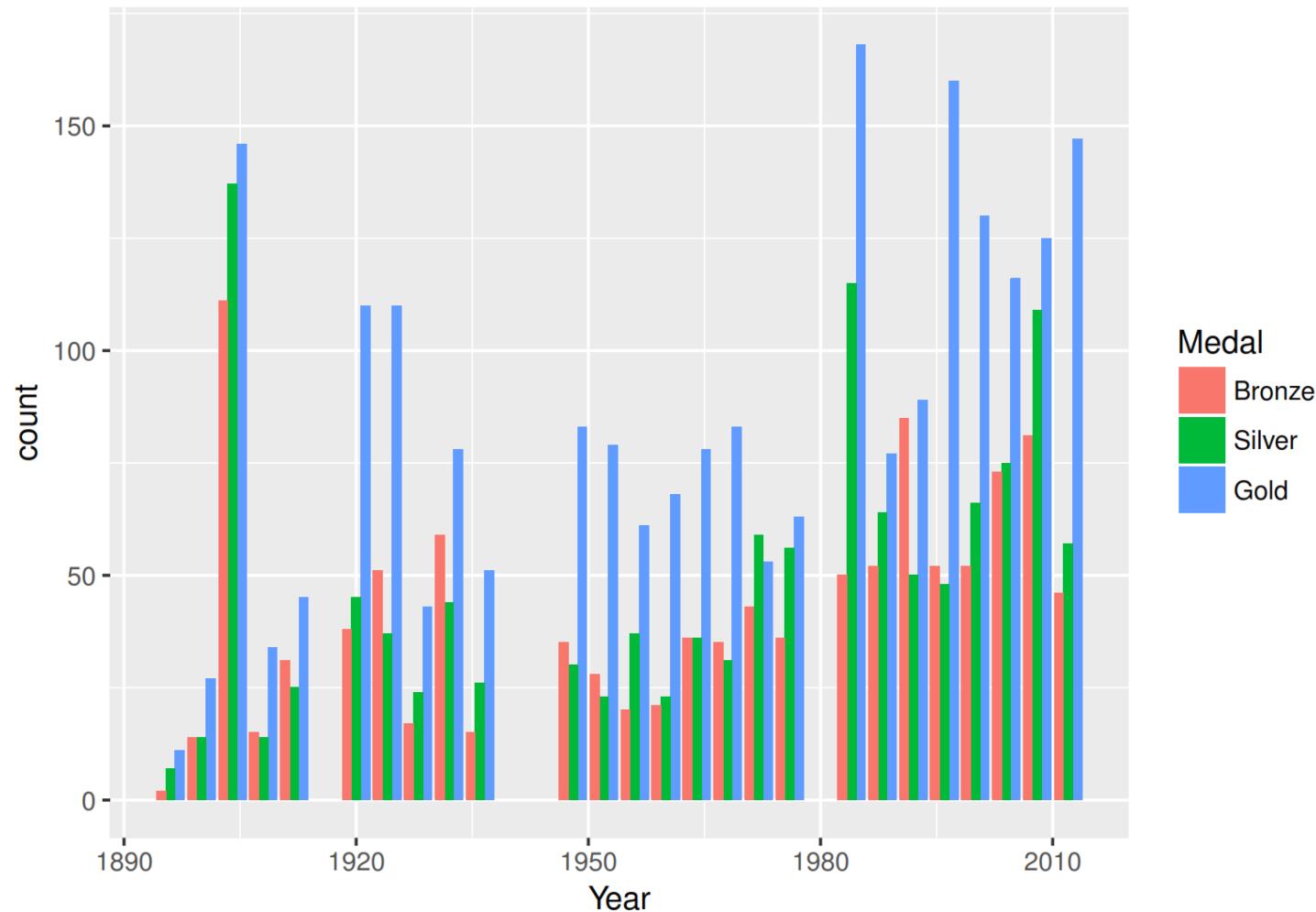
```
ggplot(summer_usa, aes(x=Year, fill=Medal)) + geom_bar(position="fill")
```



Grammar of graphics: Visualizing categorical data

Position dodge to get side-by-side bar chart

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



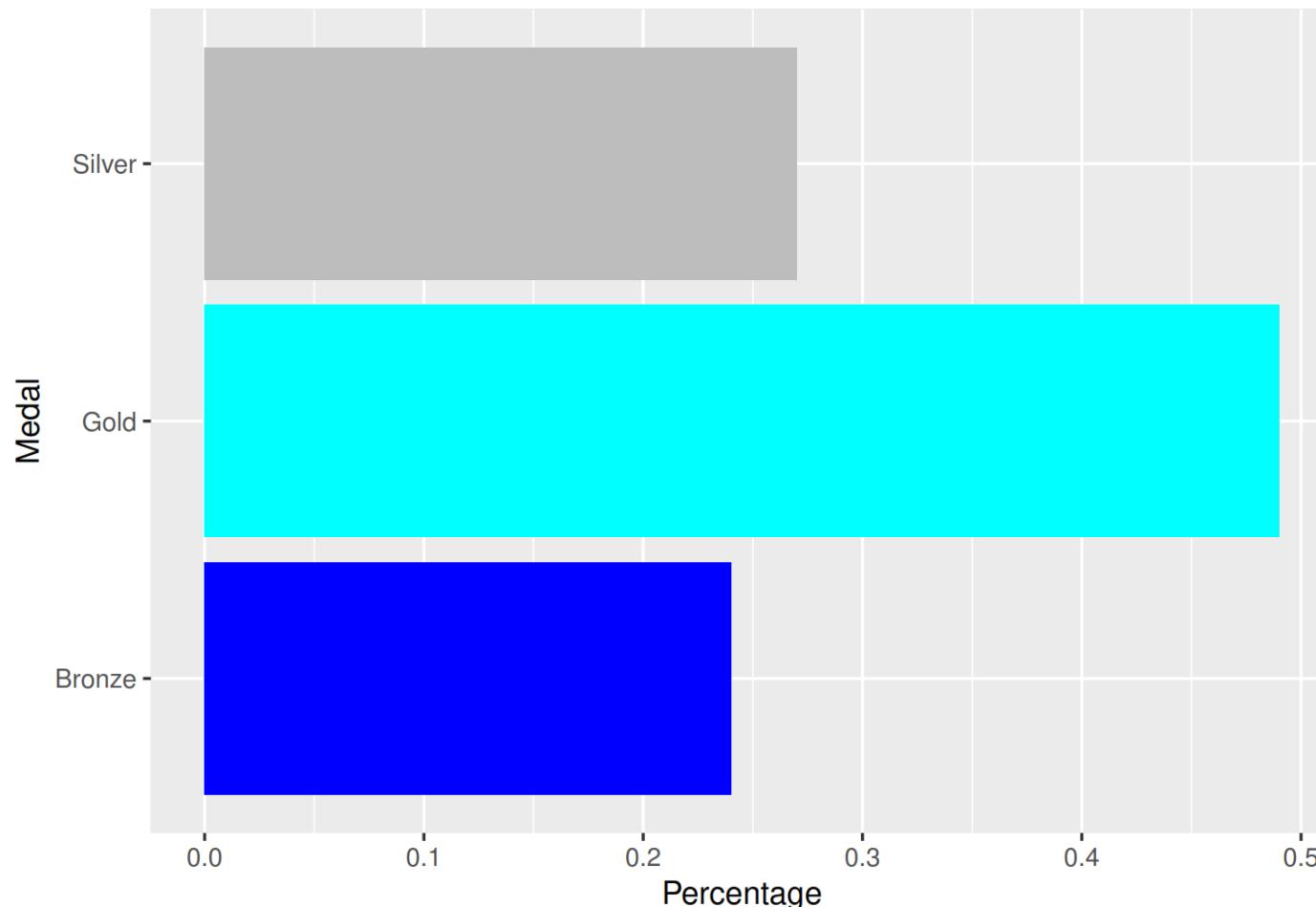
Grammar of graphics: Visualizing categorical data – coordinate system

- 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, although
- one day I hope to add 3d graphics too. 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

- **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 barplot with categories on y axis and values on x axis

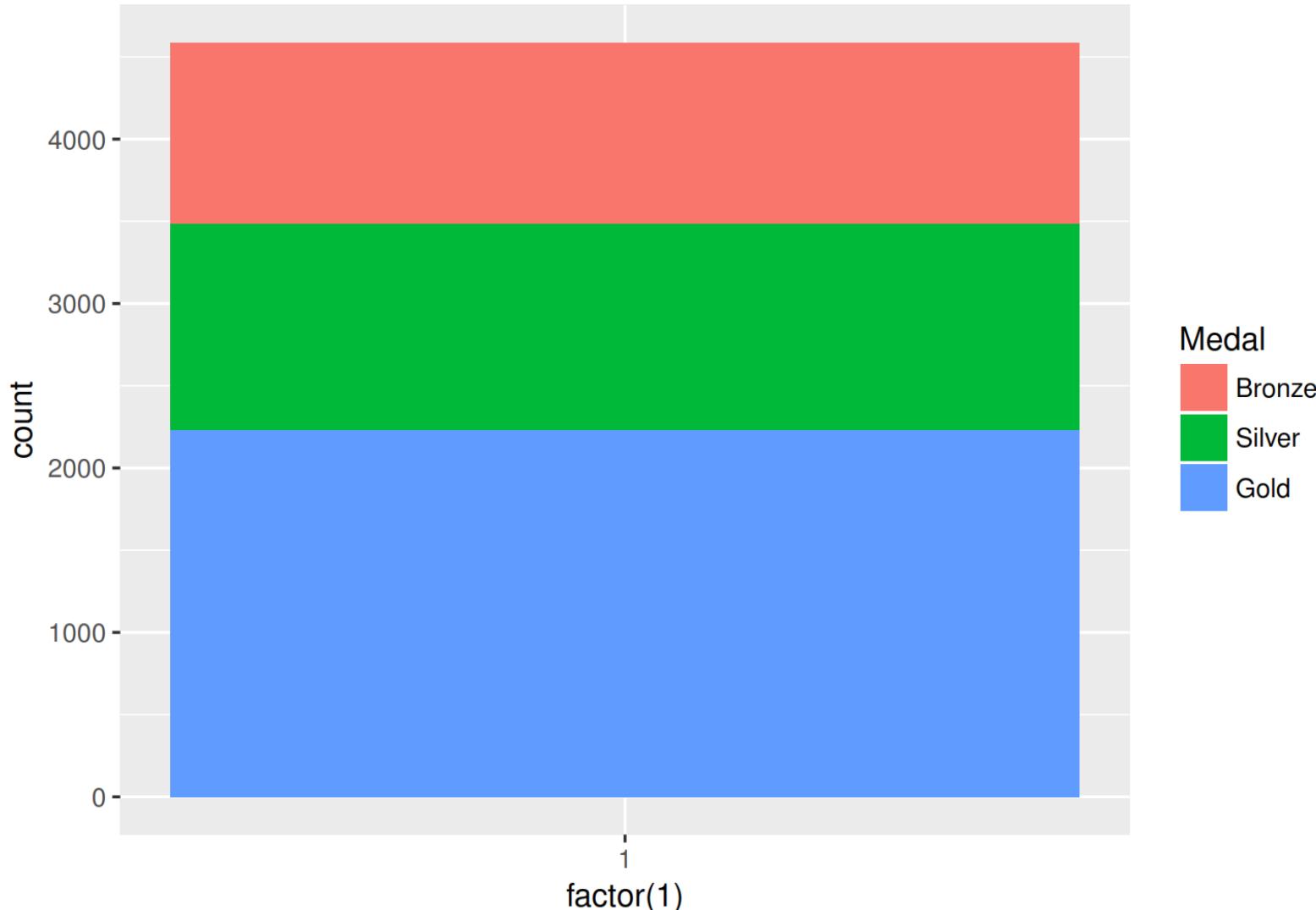
```
ggplot(usa_medals, aes(x=Medal, y=Percentage)) + geom_bar(stat="identity", fill=c(4,5,8)) +  
  coord_flip()
```



Grammar of graphics: Visualizing categorical data

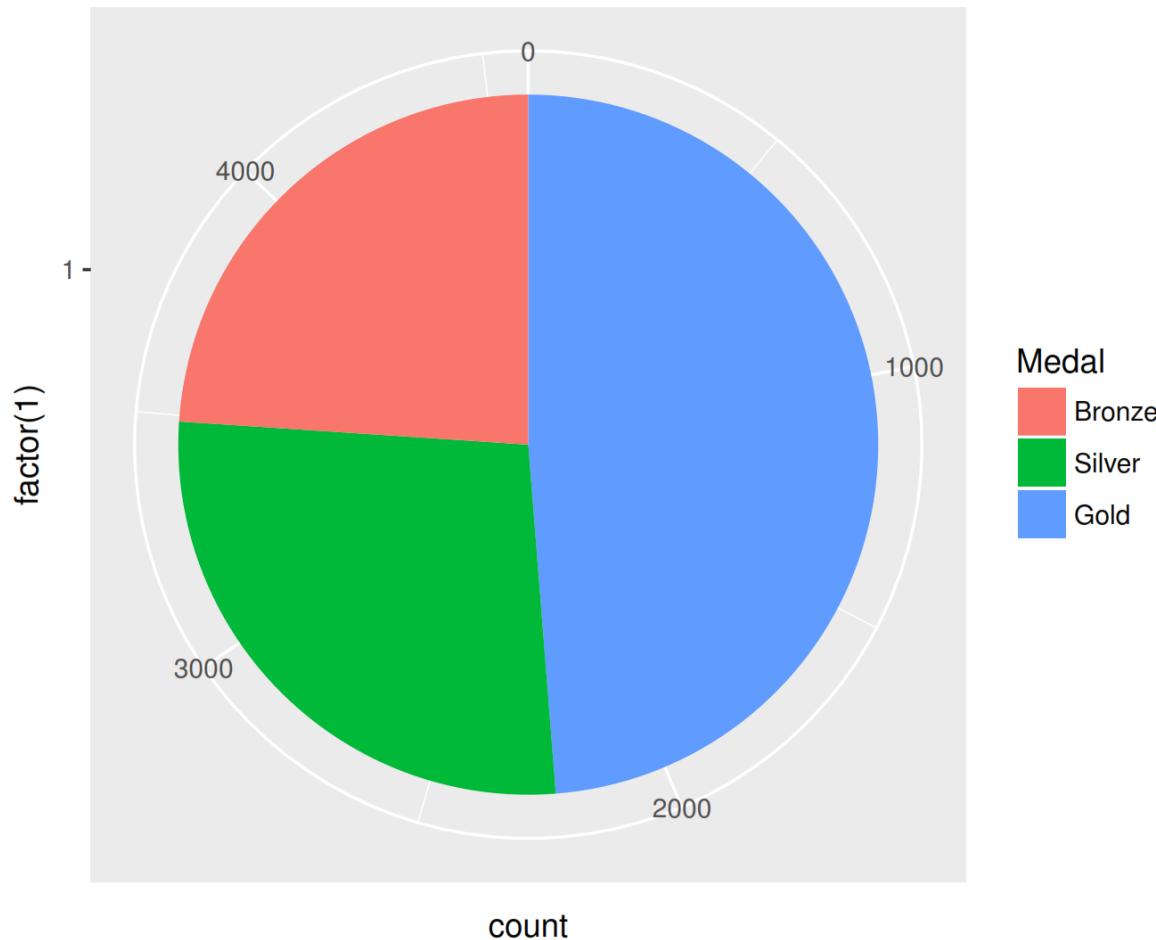
Stacked bar chart where for x variable you use 1 value (you can put anything in factor())

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



Grammar of graphics: Visualizing categorical data

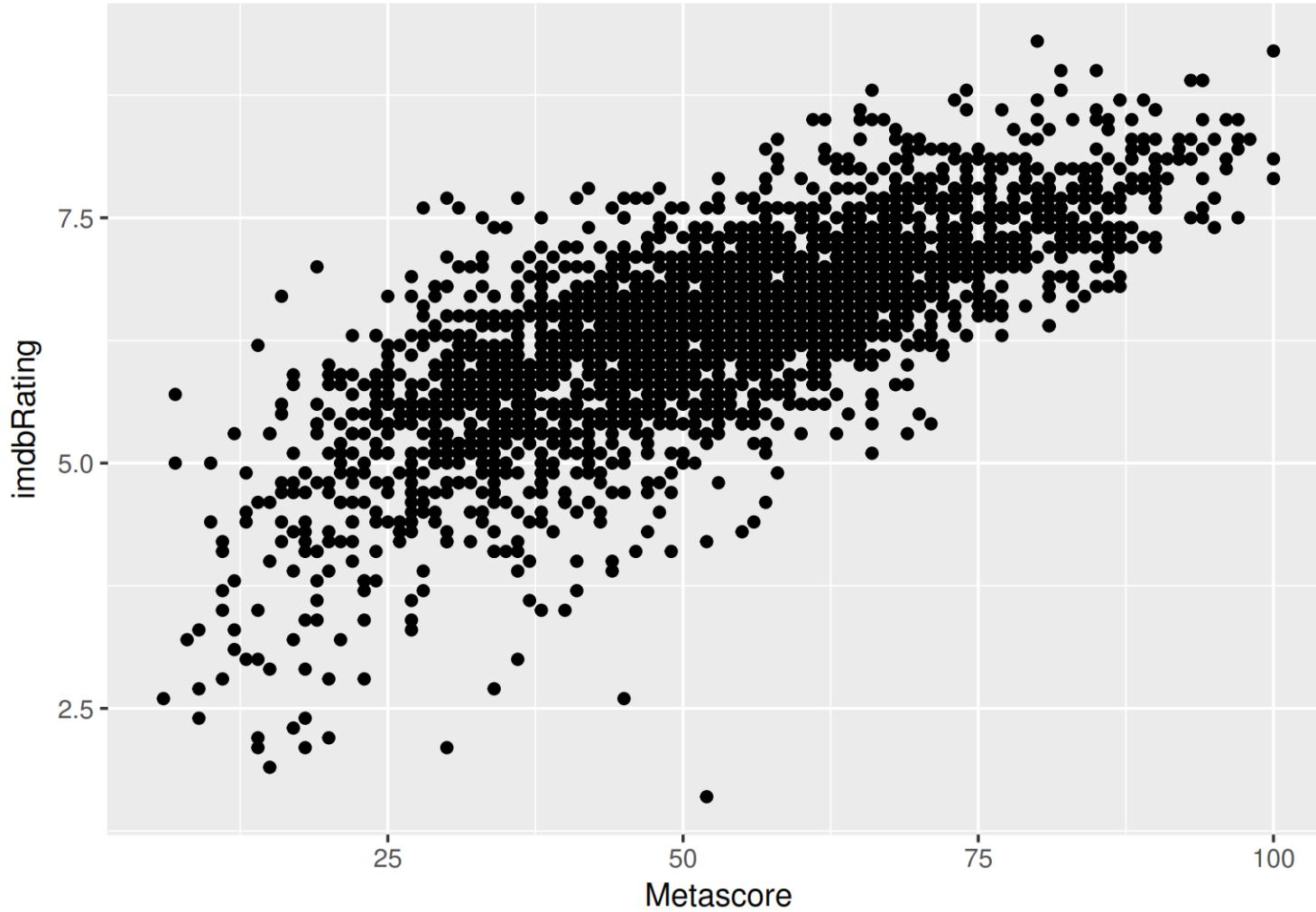
```
ggplot(summer_usa, aes(x=factor(1), fill=Medal)) + geom_bar(width=1) +  
  coord_polar(theta="y")
```



Grammar of graphics: Using coordinates on continuous data

Metascore vs imdbRating

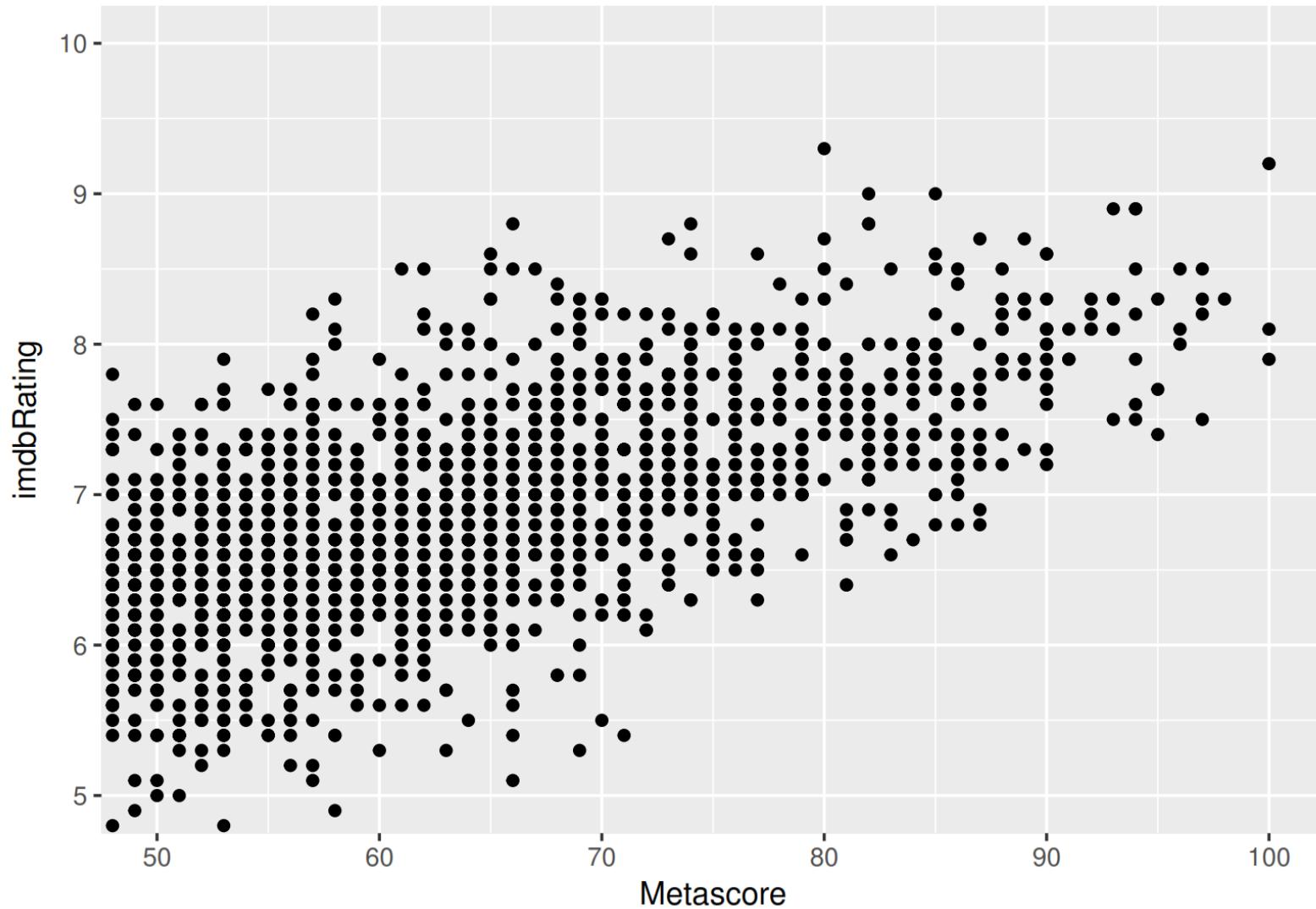
```
ggplot(movies, aes(x=Metascore, y=imdbRating)) + geom_point()
```



Grammar of graphics: Using coordinates on continuous data

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

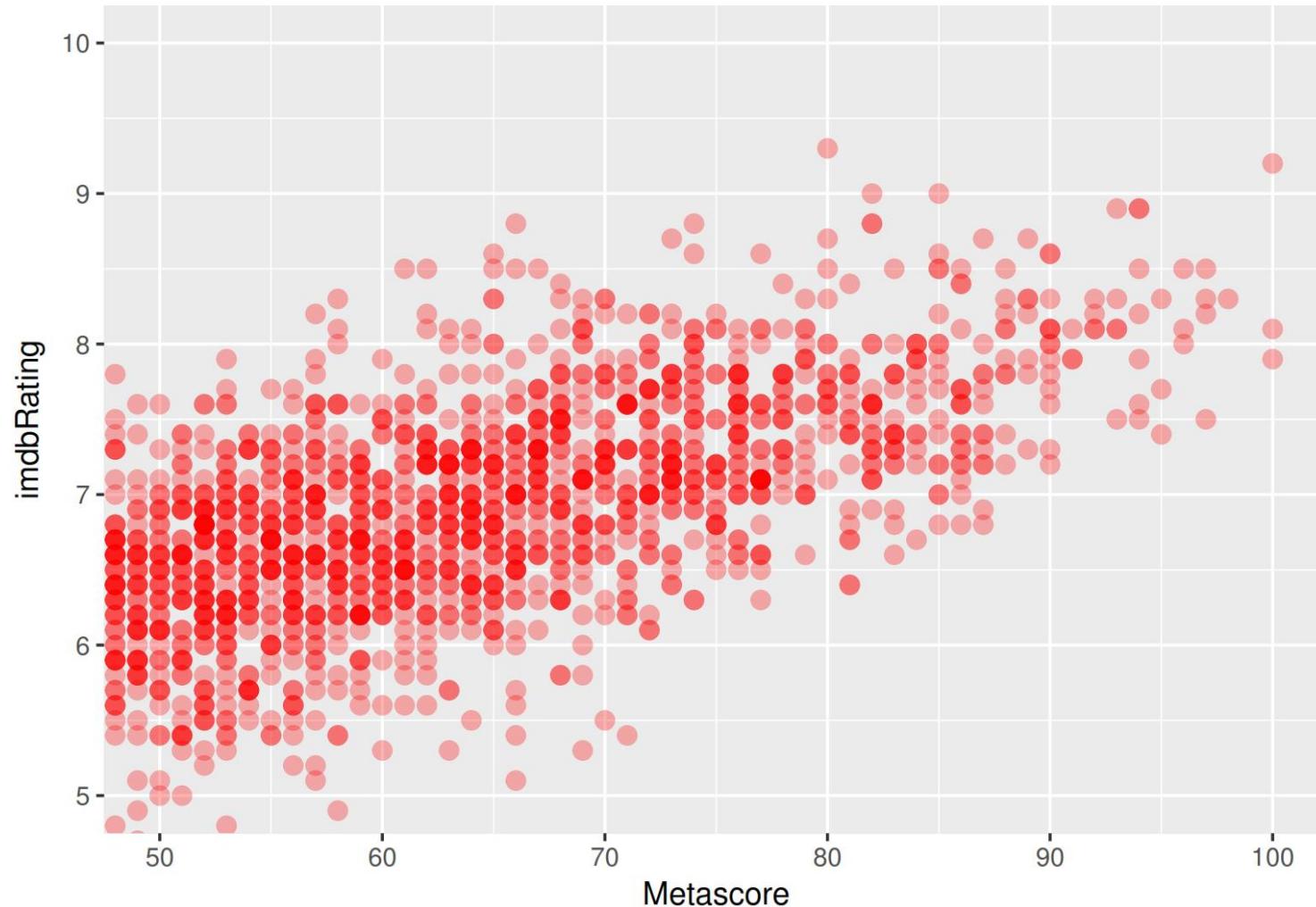
```
ggplot(movies, aes(x=Metascore, y=imdbRating)) + geom_point() +  
  coord_cartesian(ylim=c(5,10), xlim=c(50,100))
```



Grammar of graphics: Using coordinates on continuous data

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))
```



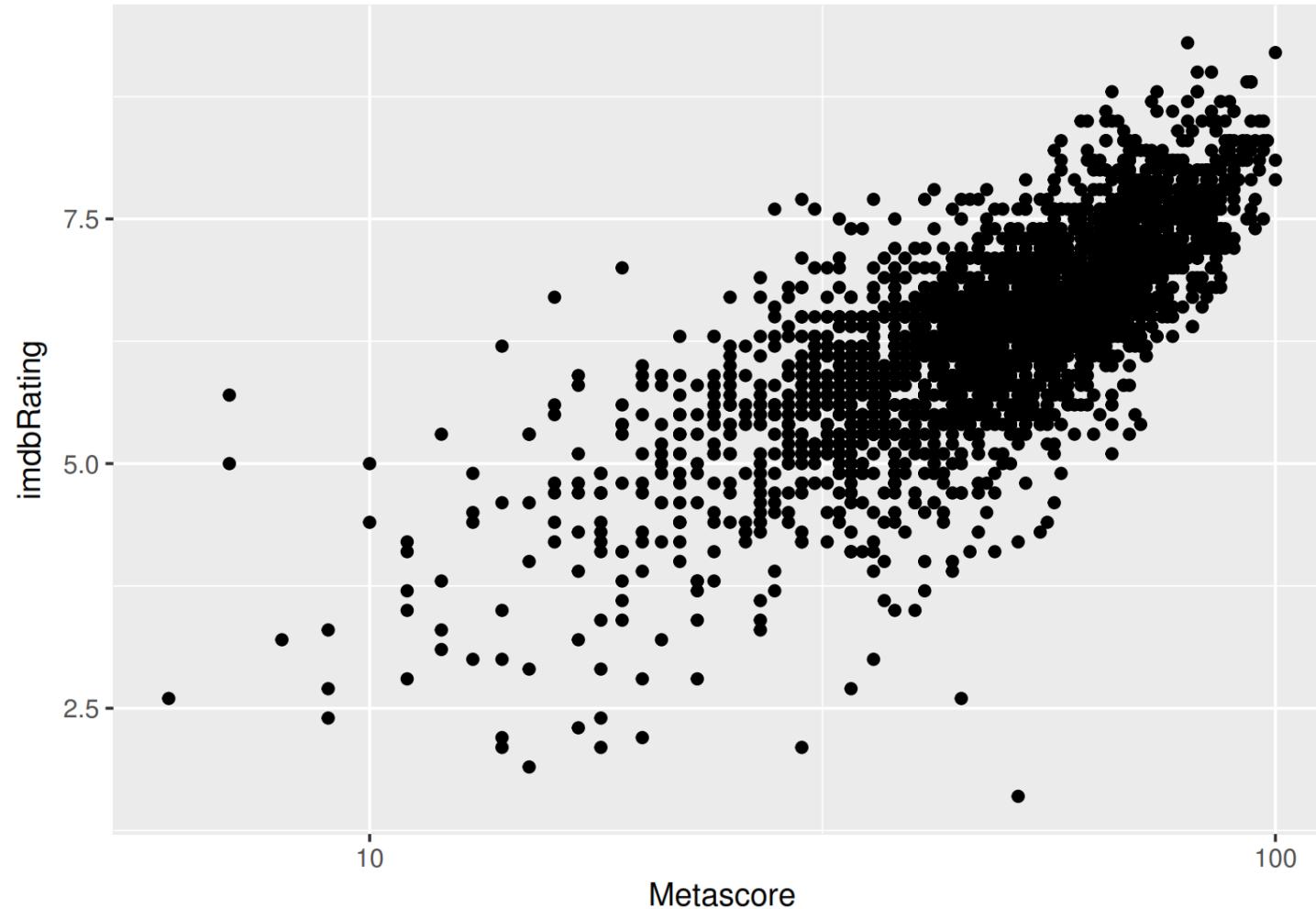
Grammar of graphics: scales

- 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

Log10 transformation of x axis

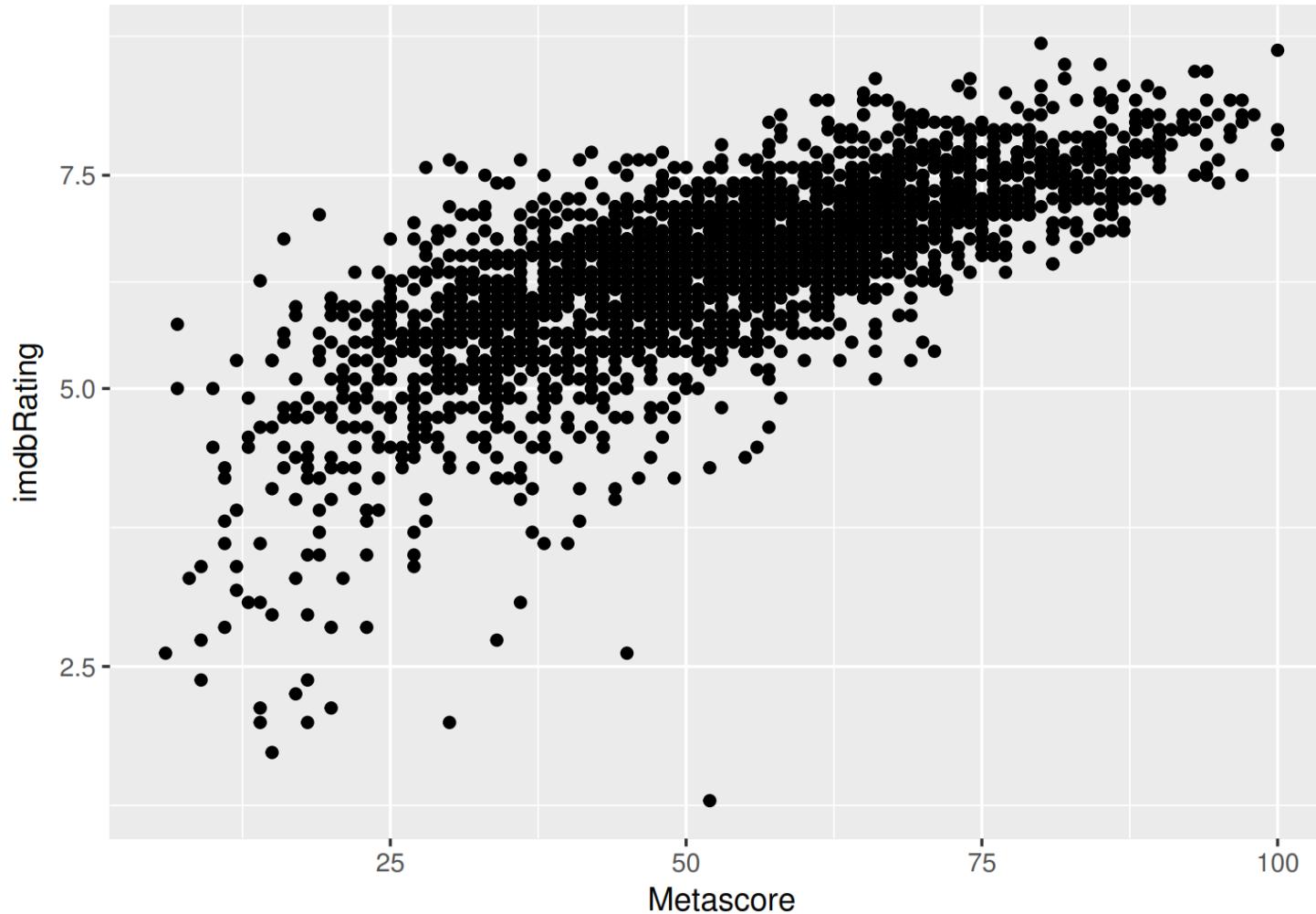
```
ggplot(movies, aes(x=Metascore, y=imdbRating)) + geom_point() +  
  scale_x_log10()
```



Grammar of graphics: scales

Square root transformation of y axis

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

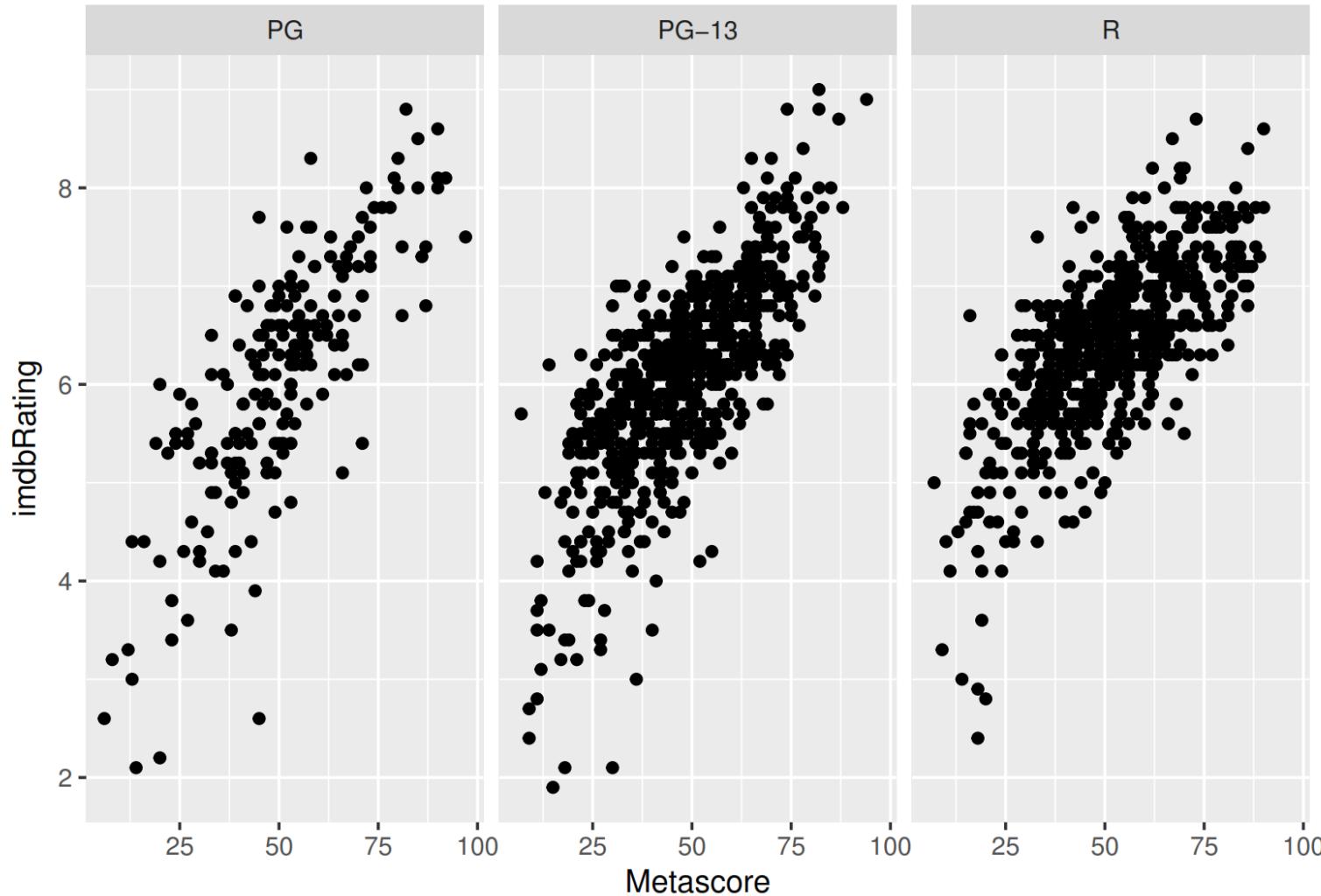


Grammar of graphics: faceting

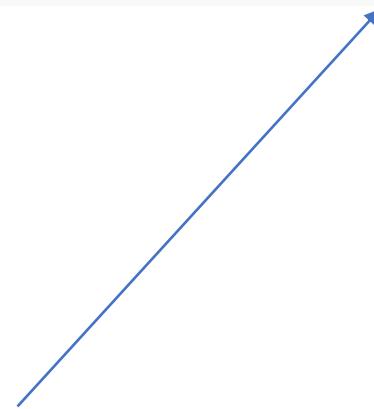
- We have already discussed using aesthetics (colour and shape) to compare subgroups, drawing all groups on the same plot.
- Faceting takes an alternative approach: It creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset in an arrangement that facilitates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

Grammar of graphics: faceting

```
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)
```



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

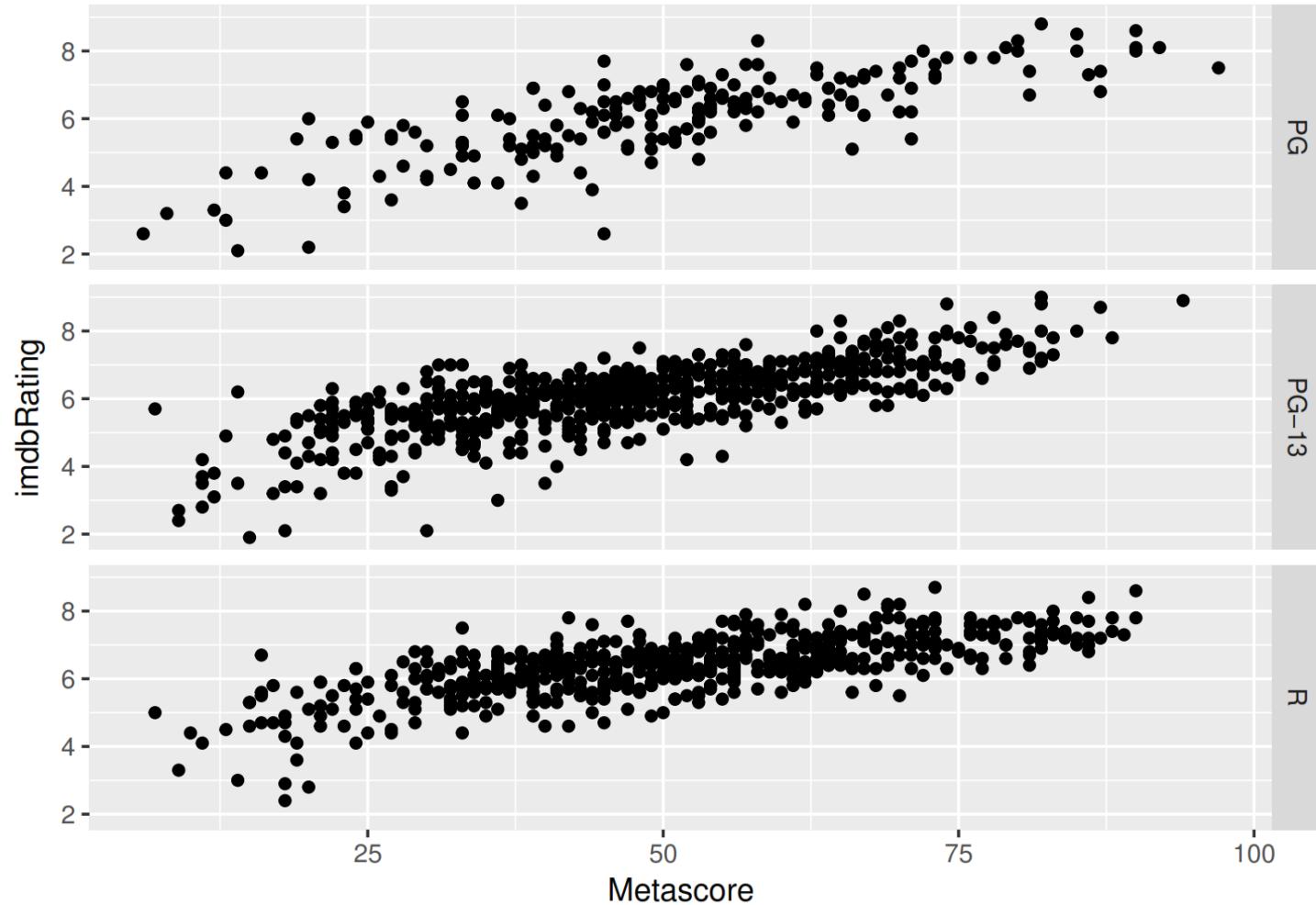


a formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column).

Grammar of graphics: faceting

faceting by rows

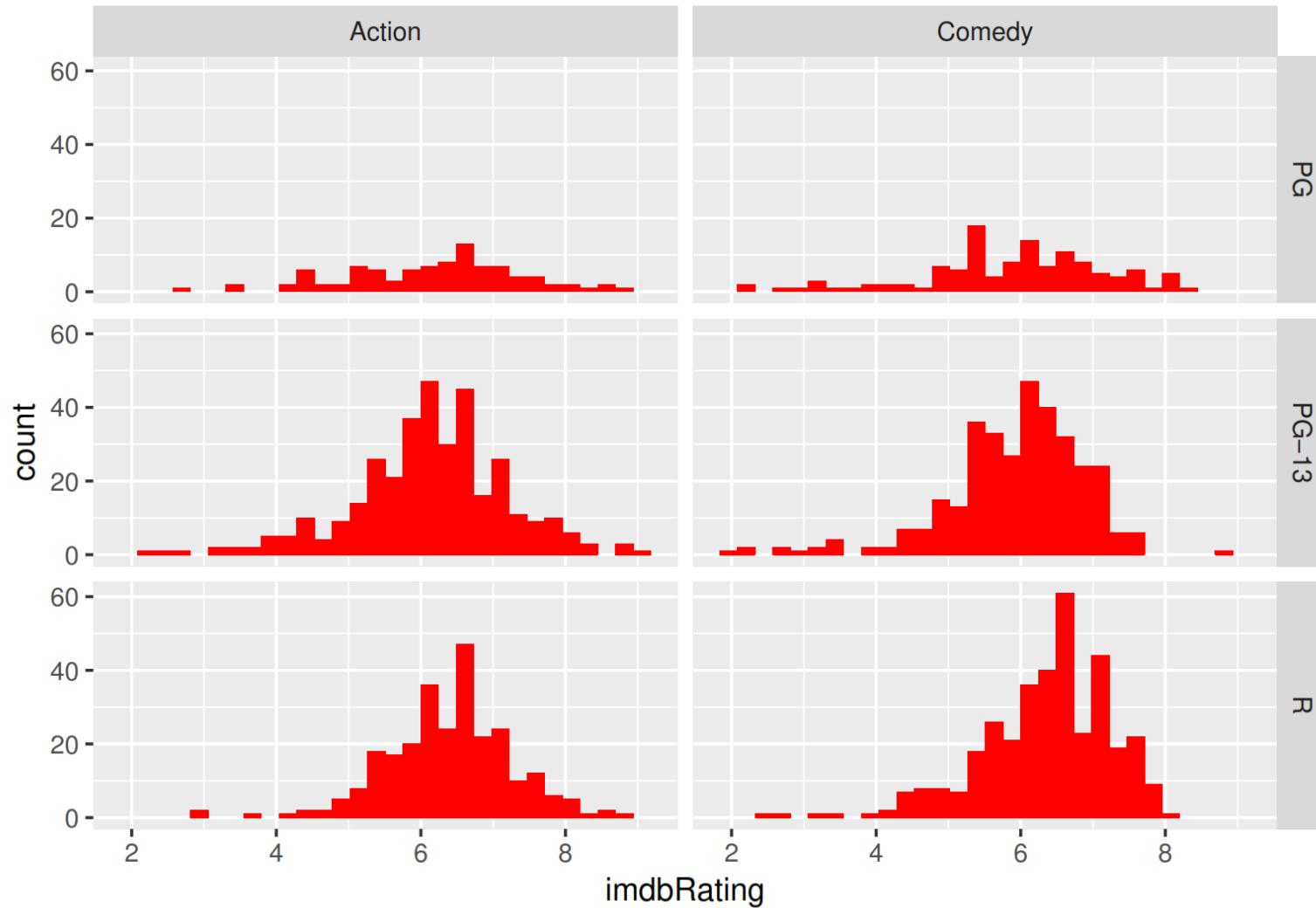
```
ggplot(movies_sub, aes(x=Metascore, y=imdbRating))+  
  geom_point() + facet_grid(Rated~.)
```



Grammar of graphics: faceting

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

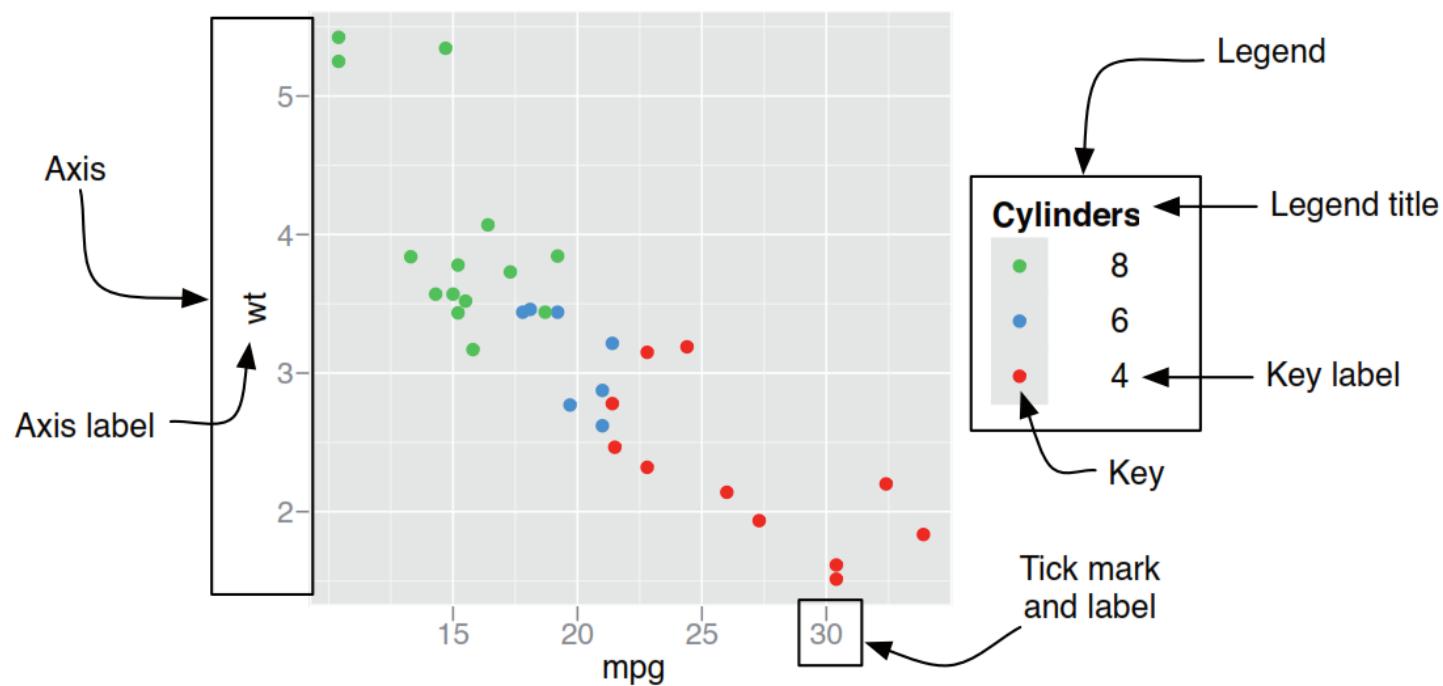
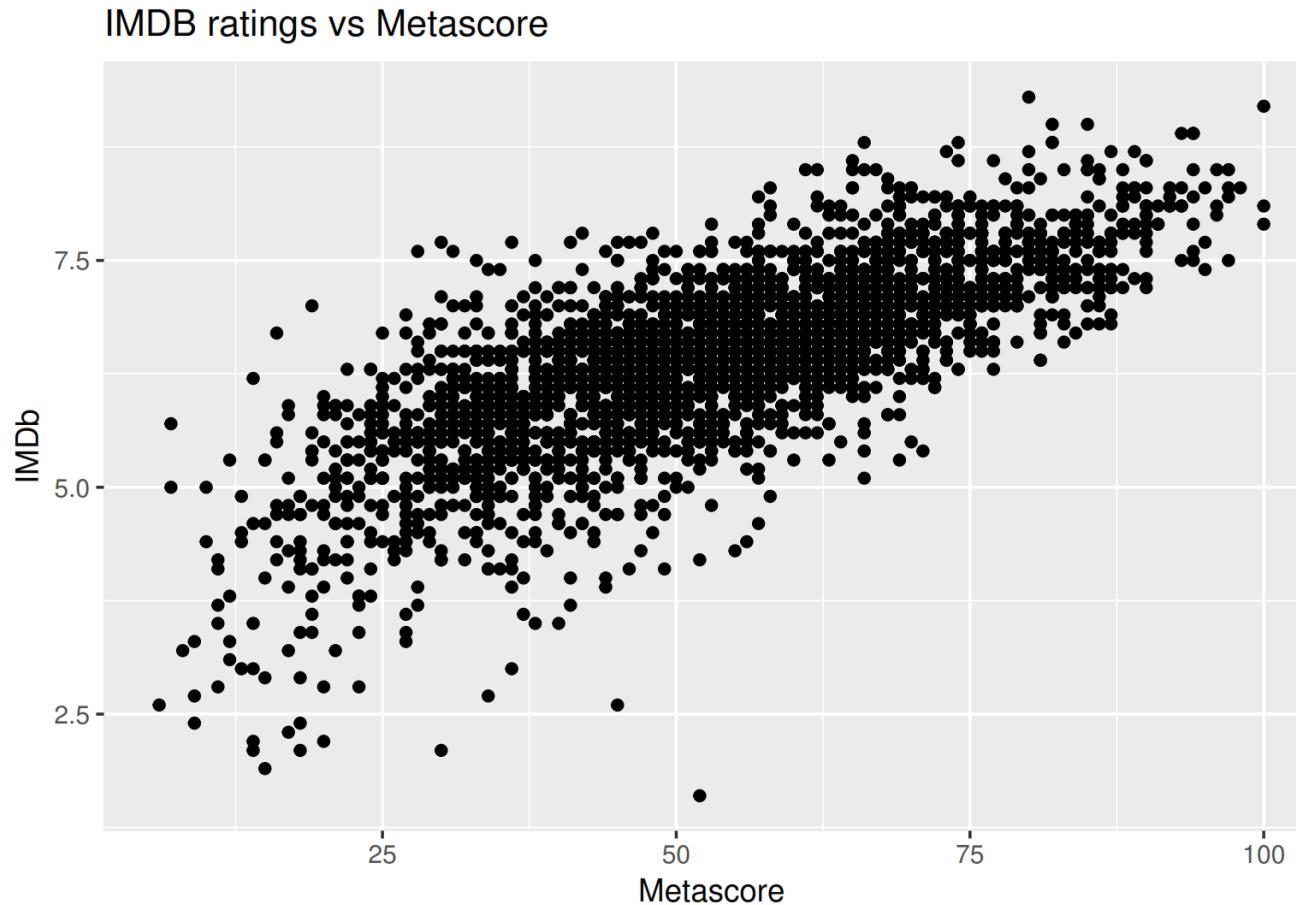


Fig. 6.12: The components of the axes and legend.

Grammar of graphics: guides and themes

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")
```

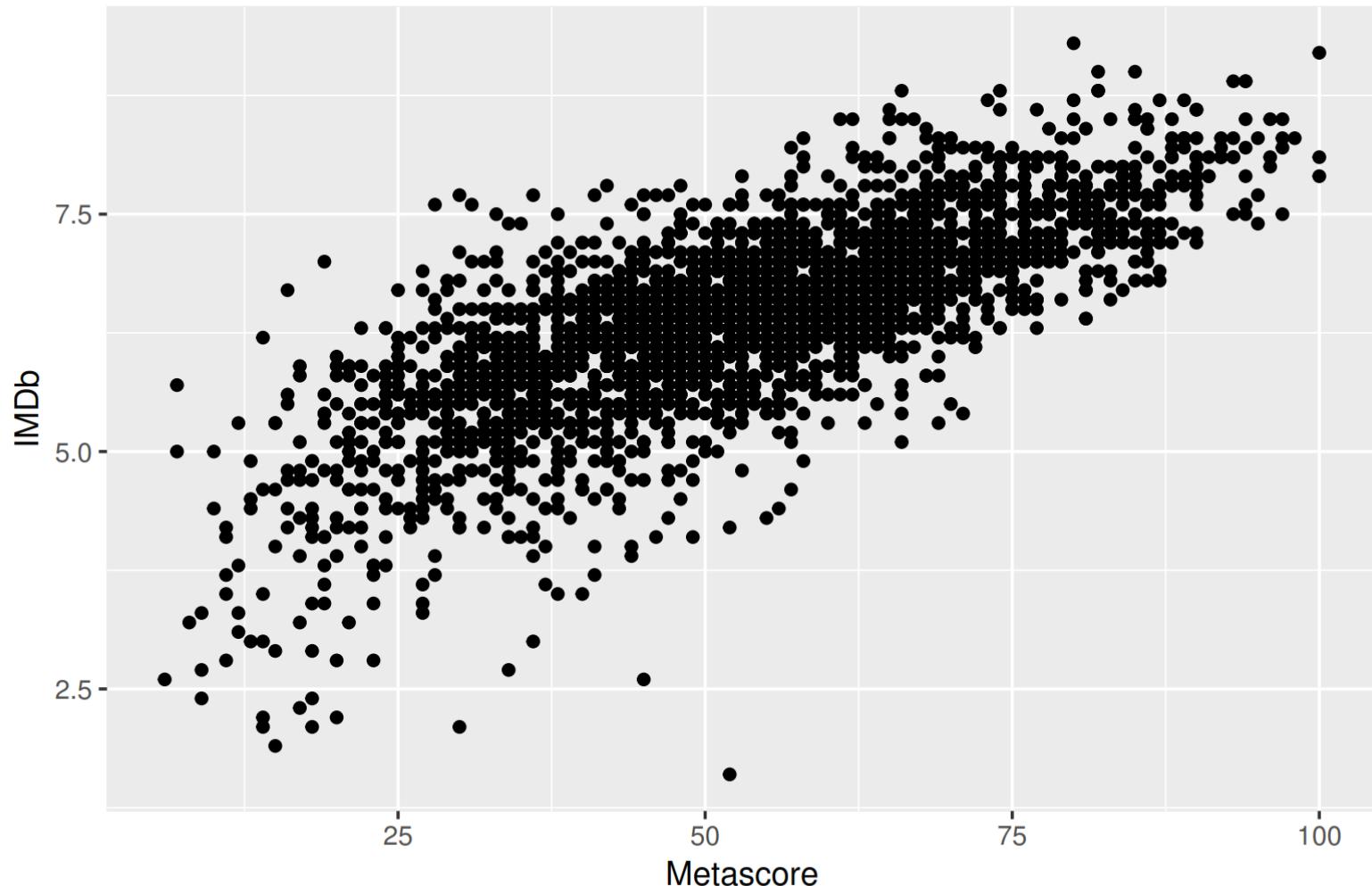


Grammar of graphics: guides and themes

Title, labels for x and y axes

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

IMDB ratings vs Metascore



Grammar of graphics: guides and themes

- 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

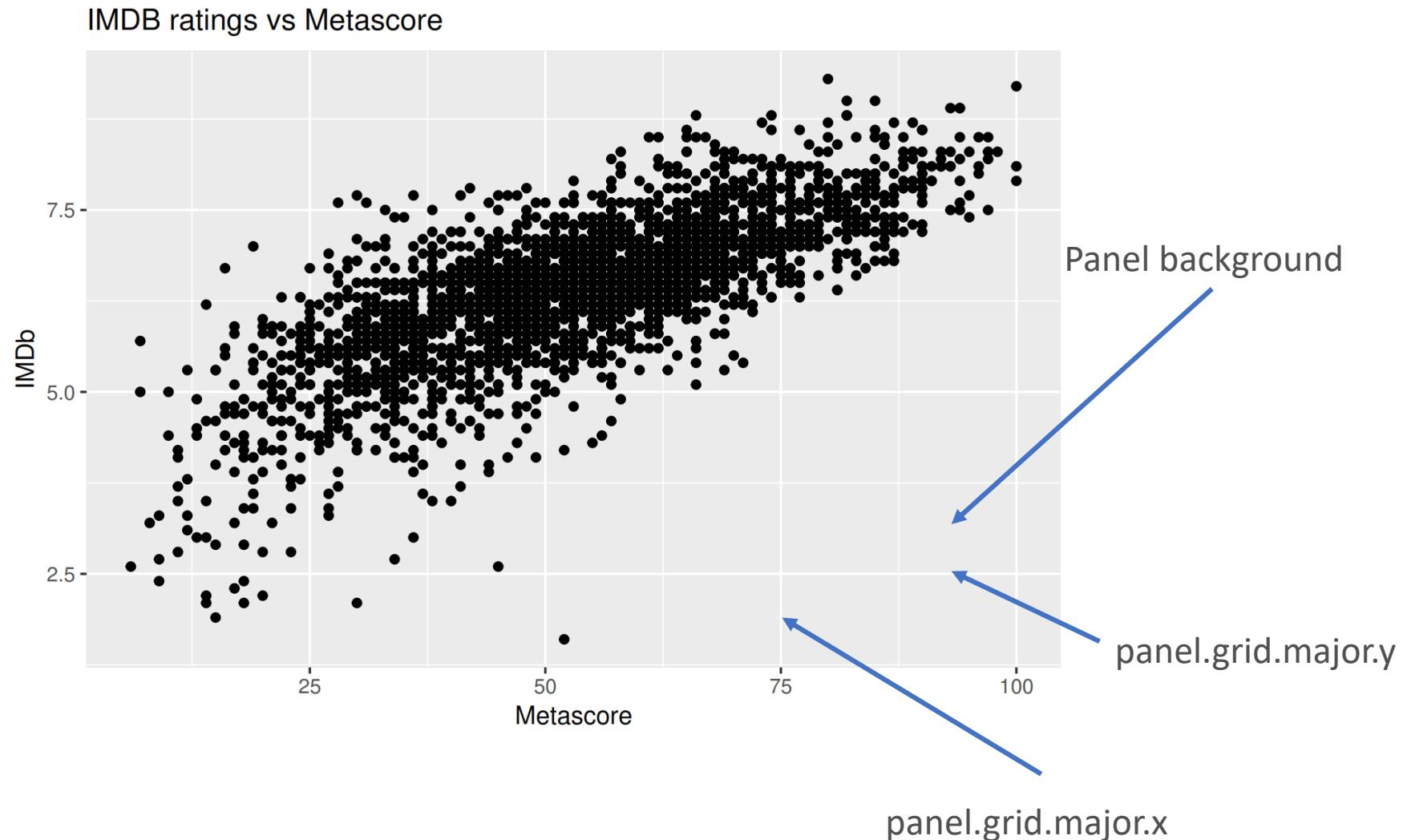
Some elements of the theme

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

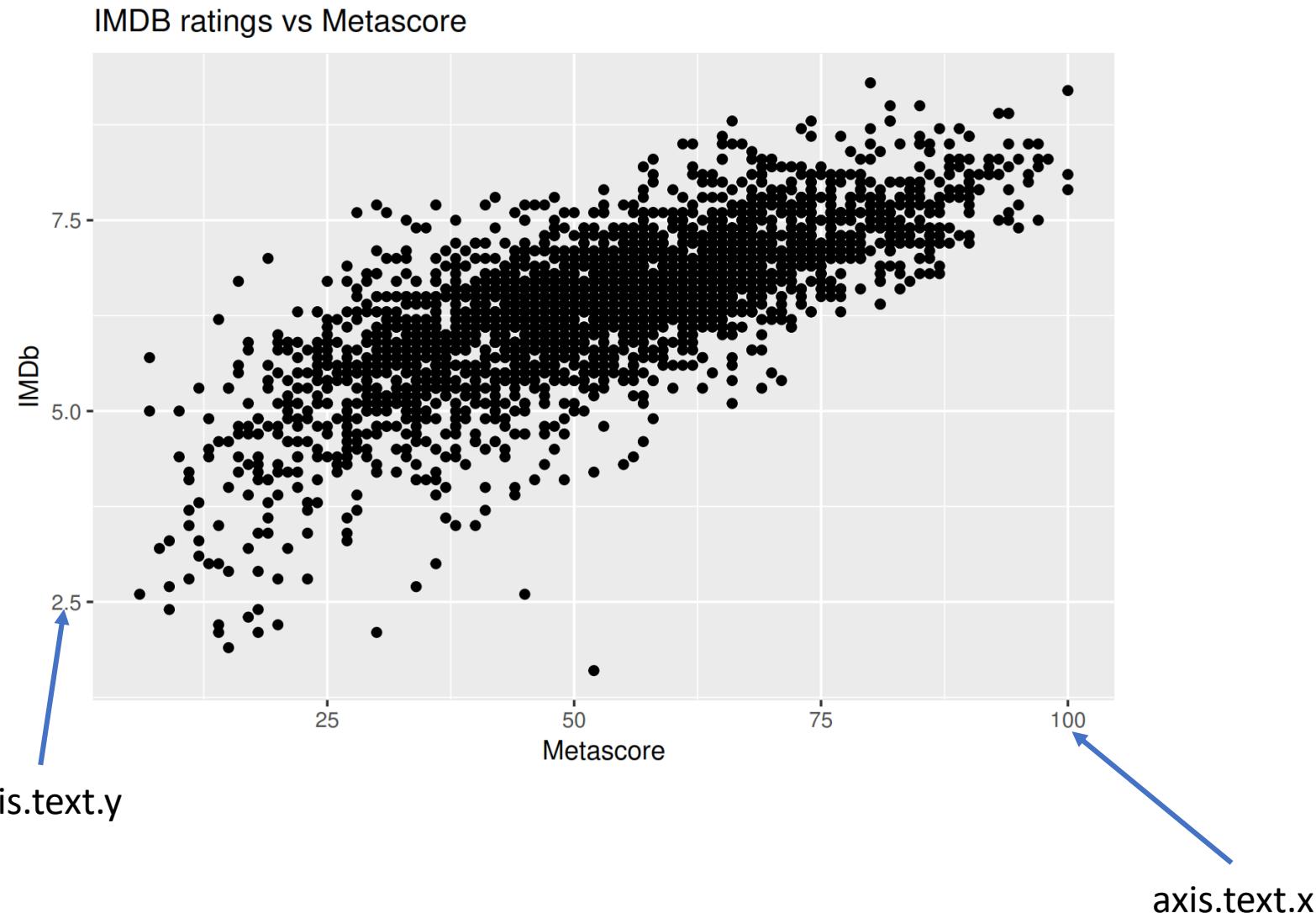
The elements

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

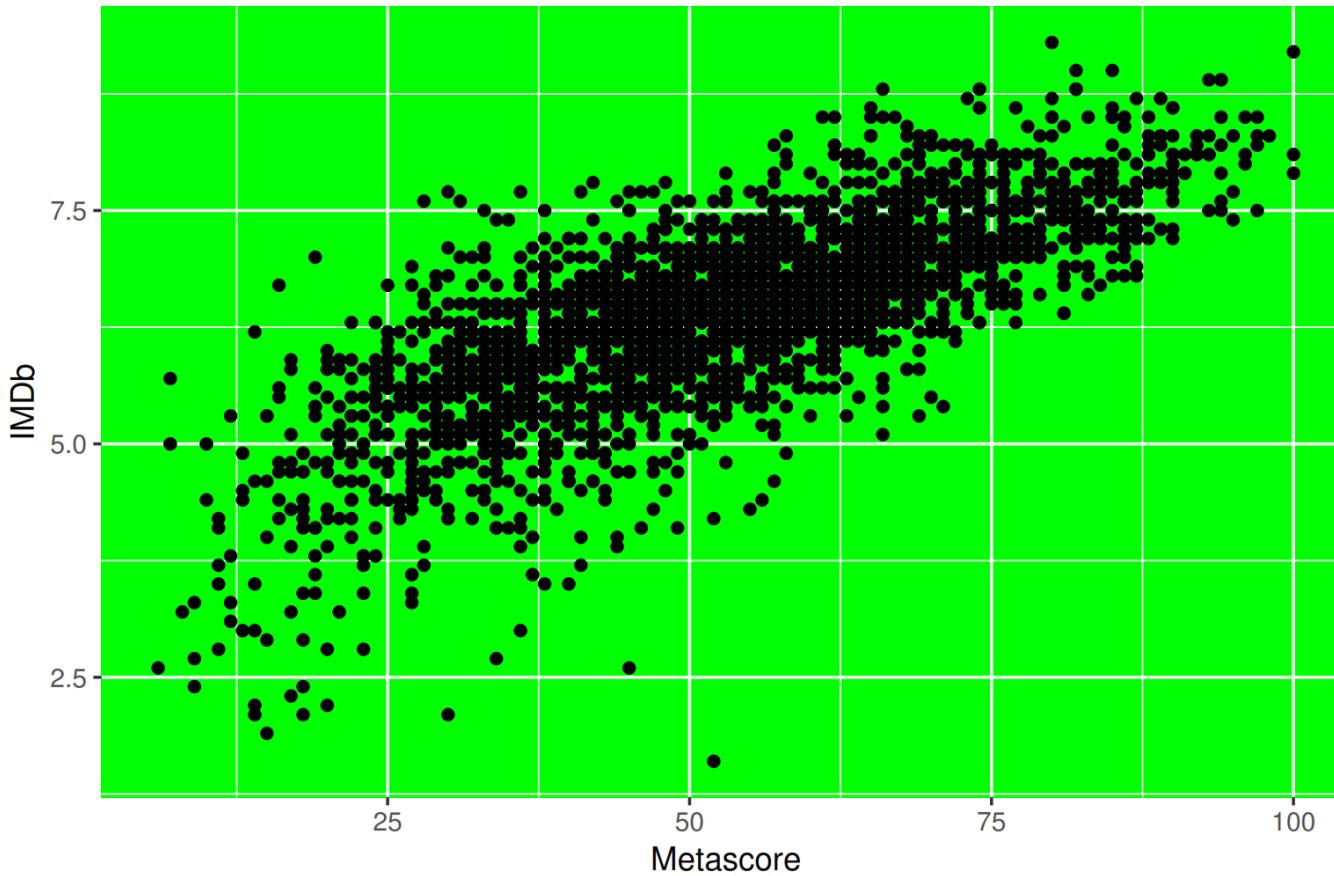


Grammar of graphics: guides and themes

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"))
```

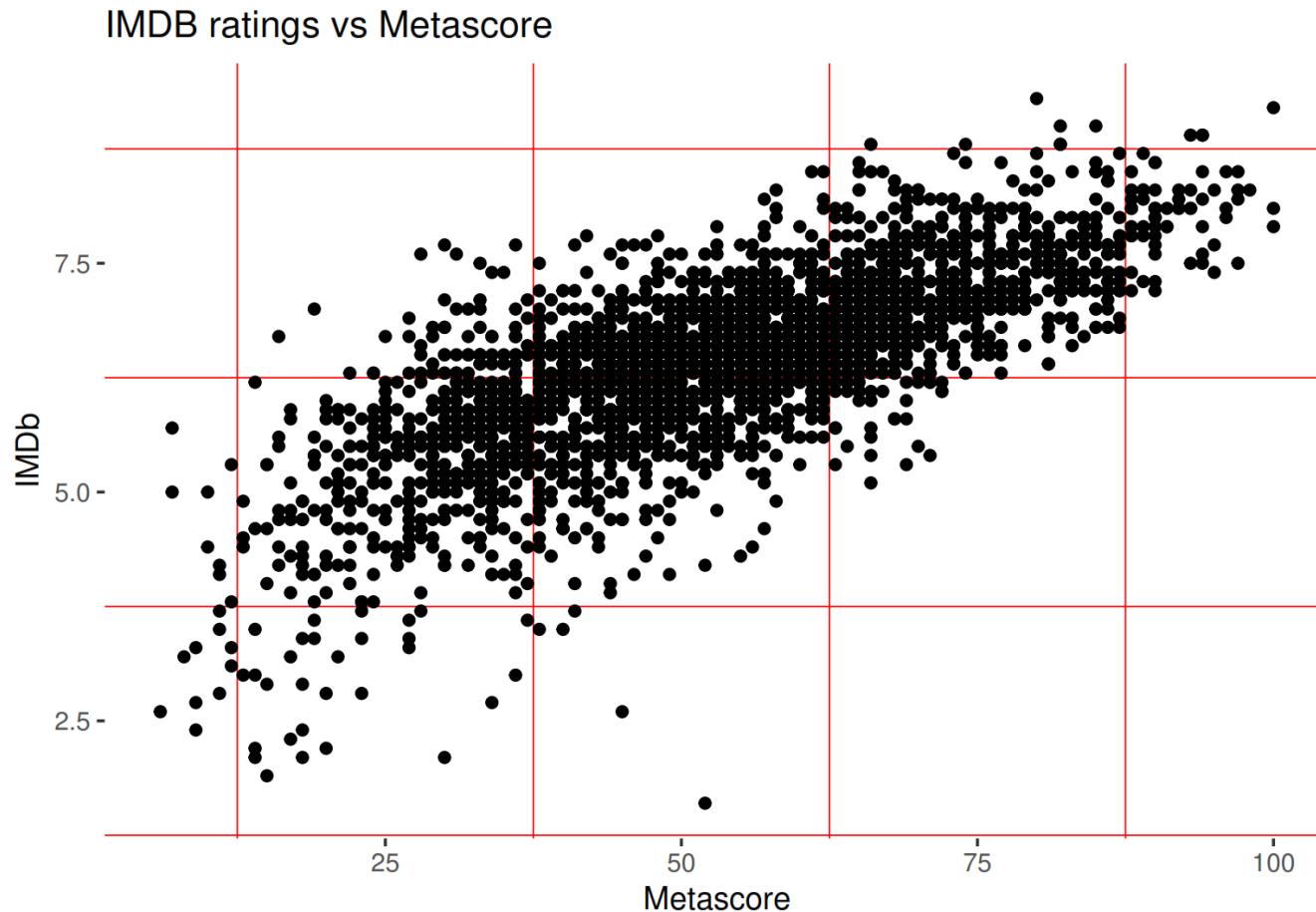
IMDB ratings vs Metascore



Grammar of graphics: guides and themes

Control the grid lines

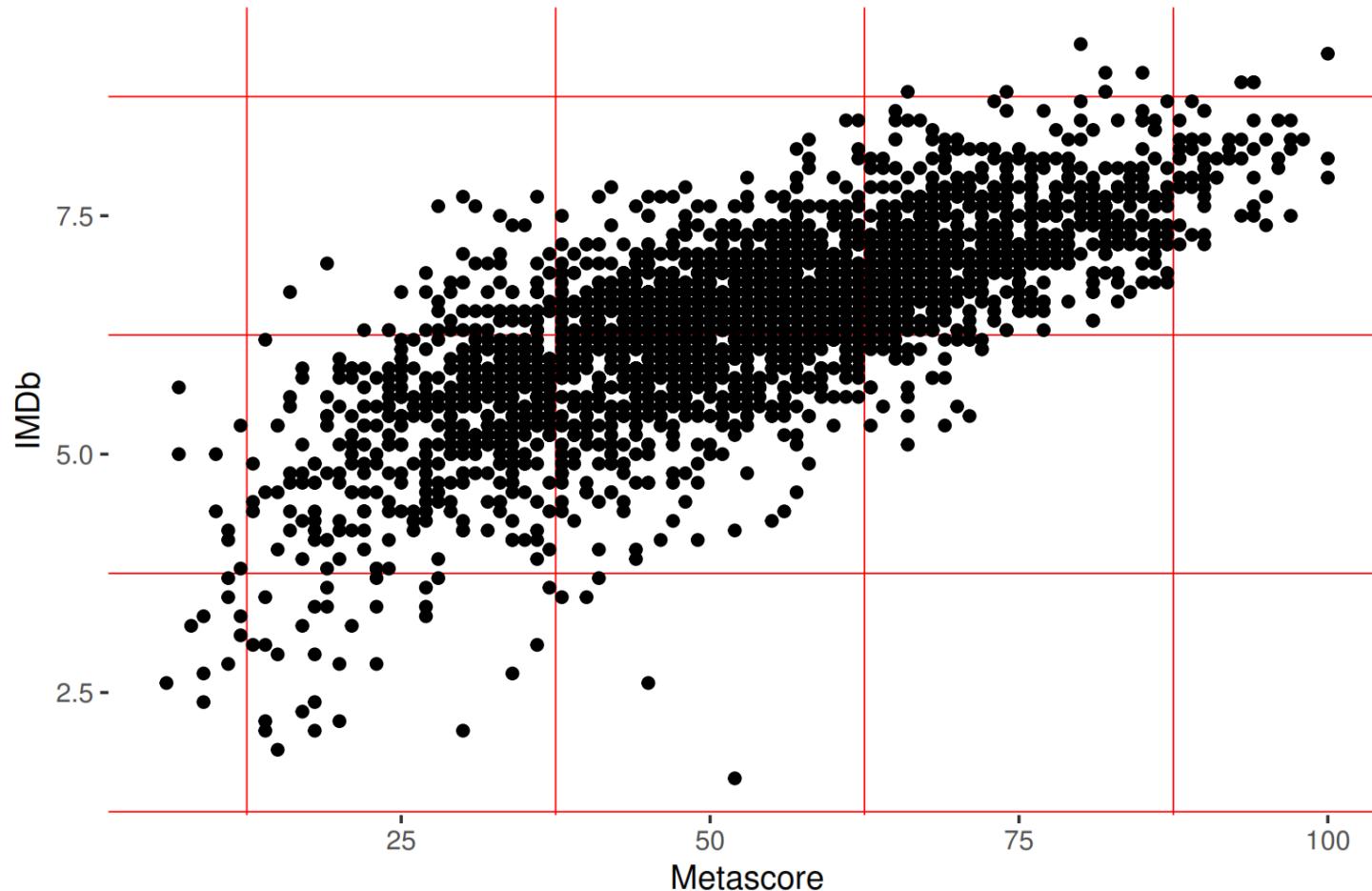
```
ggplot(movies, aes(x=Metascore, y=imdbRating))+geom_point()+
  labs(title="IMDB ratings vs Metascore", x="Metascore", y="IMDb")+
  theme(panel.background = element_blank(),
        panel.grid.major = element_line(),
        panel.grid.minor = element_line(color="red"))
```



Grammar of graphics: guides and themes: example

```
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"))
```

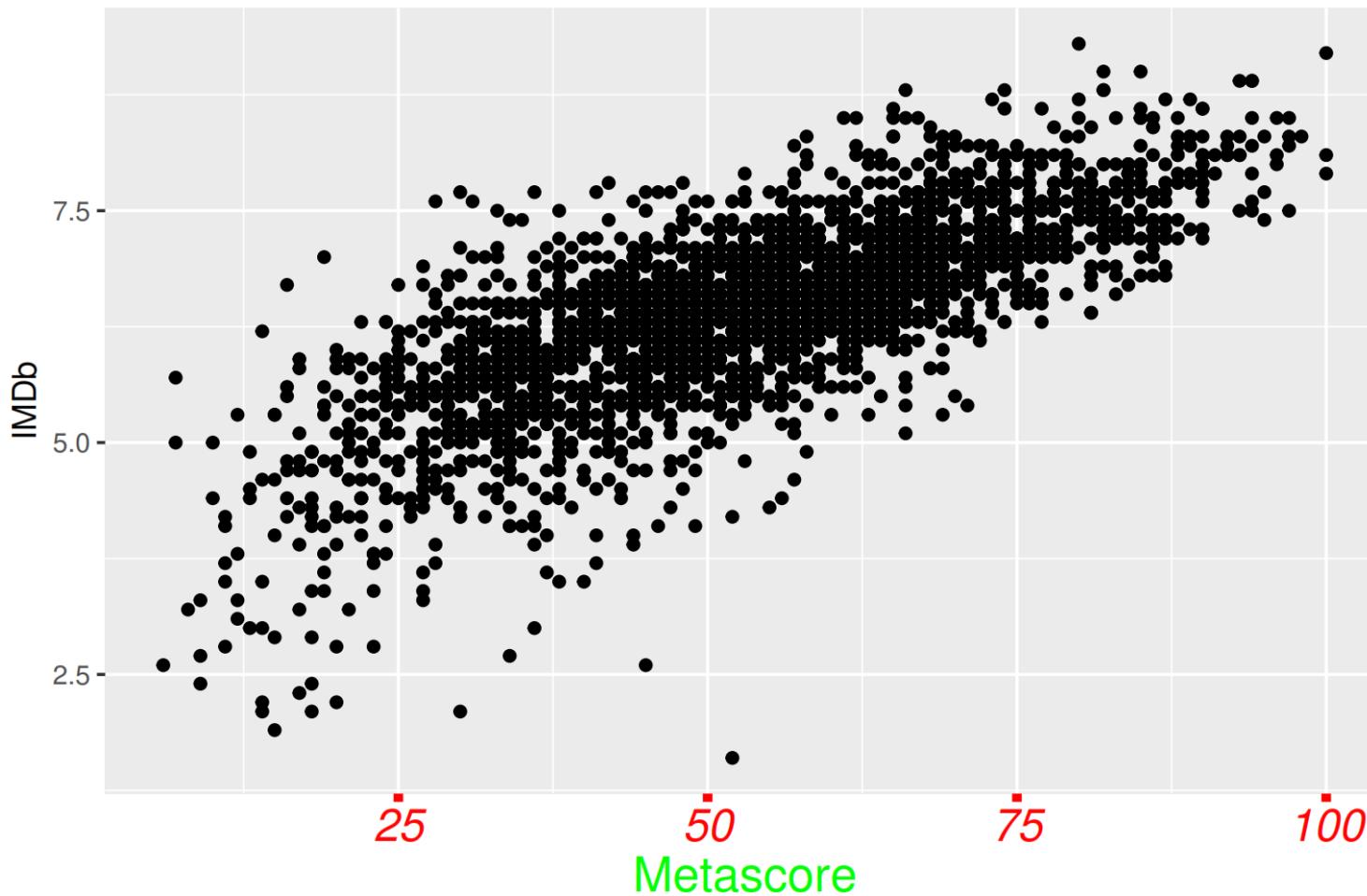
IMDB ratings vs Metascore



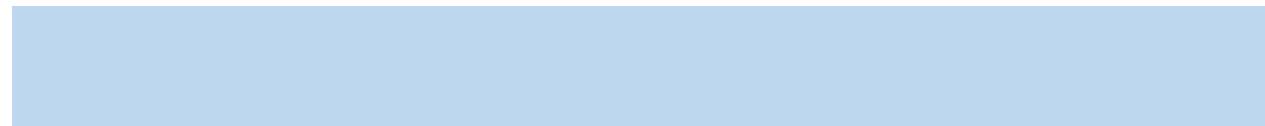
Grammar of graphics: guides and themes - example

```
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"))
```

IMDB ratings vs Metascore



Data Cleaning



Data Cleaning

- When you read the data, look at the structure, specifically the class/types of variables, to be sure that they are consistent with your data definition
- Use `stringsAsFactors=F` here to not convert string values to factors, but keep them as character

```
movies4 <- read.csv("movies4c.csv", stringsAsFactors = F)
str(movies4)

## 'data.frame': 2912 obs. of 33 variables:
## $ title           : chr "Zoom" "Zoolander 2" "Zookeeper" "Zombieland" ...
## $ genre_first     : chr "Action" "Comedy" "Comedy" "Adventure" ...
## $ year            : int 2006 2016 2011 2009 2007 1998 2012 2005 2008 1998 ...
## $ duration         : int 83 102 102 88 162 116 157 101 101 119 ...
## $ gross_adjusted   : num 14142117 29451448 87570655 86365946 39077724 ...
## $ budget_adjusted  : int 42555556 51065177 87177413 26964263 76858659 7519018 42704949 81
## $ gross            : int 11631245 28837115 80360866 75590286 33048353 1980338 95720716 28
## $ budget           : int 35000000 50000000 80000000 23600000 65000000 5000000 40000000 61
## $ cast_facebook_likes: int 5022 24107 5392 28011 36928 1209 2759 32232 638 17768 ...
## $ reviews          : int 176 376 305 998 966 232 1198 338 490 709 ...
## $ index            : num 1.22 1.02 1.09 1.14 1.18 ...
## $ Rated            : chr "PG" "PG-13" "PG" "R" ...
## $ Genre             : chr "Action, Adventure, Comedy" "Comedy" "Comedy, Family, Romance" "A
## $ Director          : chr "Peter Hewitt" "Ben Stiller" "Frank Coraci" "Ruben Fleischer" .
## $ Writer            : chr "Adam Rifkin (screenplay), David Berenbaum (screenplay), Adam R
## $ Actors             : chr "Tim Allen, Courteney Cox, Chevy Chase, Spencer Breslin" "Justin
## $ Plot               : chr "Former superhero Jack is called back to work to transform an un
## $ Language           : chr "English" "English, Italian, Spanish" "English" "English" ...
## $ Country            : chr "USA" "USA" "USA" "USA" ...
## $ Awards             : chr "4 wins & 8 nominations." "7 wins & 17 nominations." "1 win & 2
## $ Metascore          : int 26 34 30 73 78 NA 95 67 56 57 ...
## $ imdbRating         : chr "4.3" "4.7" "5.2" "7.7" ...
## $ imdbVotes          : chr "16367" "53943" "49098" "426786" ...
```

Data Cleaning

supply function is traversing over a set of data like a list or dataframe, and calling the specified function for each item.

```
sapply(movies4, class)
```

```
##          title      genre_first           year
## "character" "character"      "integer"
##      duration gross_adjusted budget_adjusted
## "integer"     "numeric"      "integer"
##       gross      budget cast_facebook_likes
## "integer"     "integer"      "integer"
##      reviews        index           Rated
## "integer"     "numeric"      "character"
##      Genre        Director          Writer
## "character" "character"      "character"
##      Actors         Plot           Language
## "character" "character"      "character"
##      Country        Awards        Metascore
## "character" "character"      "integer"
##   imdbRating      imdbVotes      Production
## "character" "character"      "character"
##       DVD        Release Release_Month
## "character" "character"      "integer"
## Release_Day    Release_year      OscarWon
## "integer"     "integer"      "integer"
##      OtherWin      OscarNom      OtherNom
## "integer"     "integer"      "integer"
```

Data Cleaning

`sapply` function is traversing over a set of data like a list or dataframe, and calling the specified function for each item.

```
sapply(movies4, class)
```

```
##          title      genre_first           year
## "character" "character"      "integer"
##      duration gross_adjusted budget_adjusted
## "integer"     "numeric"      "integer"
##       gross      budget cast_facebook_likes
## "integer"     "integer"      "integer"
##      reviews      index           Rated
## "integer"     "numeric"      "character"
##        Genre      Director          Writer
## "character" "character"      "character"
##      Actors          Plot           Language
## "character" "character"      "character"
##      Country        Awards        Metascore
## "character" "character"      "integer"
##      imdbRating    imdbVotes      Production
## "character" "character"      "character"
##          DVD        Release   Release_Month
## "character" "character"      "integer"
## Release_Day    Release_year      OscarWon
## "integer"     "integer"      "integer"
##      OtherWin      OscarNom      OtherNom
## "integer"     "integer"      "integer"
```

Data Cleaning

Look at the Unique observations for imdb rating (note, this is not going to be always the right way to check for the errors)

```
unique(movies4$imdbRating)

## [1] "4.3"      "4.7"      "5.2"      "7.7"      "12"       "7.4"      "6.1"
## [8] "6.6"      "6.5"      "5.5"      "6.8"      "#DIV/0!"   "8"       "6.3"
## [15] "5.6"      "#NAME?"    "3.6"      "5.8"      "4.6"      "4.9"      "4.4"
## [22] "5.9"      "7.8"      "7"        "6.7"      "7.3"      "11"       "8.1"
## [29] "5.3"      "6.2"      "3.2"      "7.2"      "10.5"     "4.2"      "6"
## [36] "4.8"      "7.1"      "5.7"      "4.5"      "2.1"      "6.4"      "8.5"
## [43] "6.9"      "7.6"      "5.1"      "5.4"      "11.6"     "8.2"      NA
## [50] "7.9"      "3.4"      "8.3"      "4.1"      "7.5"      "4"        "5"
## [57] "8.6"      "9.3"      "2.7"      "3.5"      "8.7"      "3.3"      "3.8"
## [64] "8.9"      "3.9"      "9"        "9.2"      "8.4"      "2.8"      "8.8"
## [71] "2.2"      "3"        "3.7"      "3.1"      "1.6"      "2.9"      "2.4"
## [78] "2.3"      "2.6"      "1.9"
```

Data Cleaning

Look at the Unique observations for IMDb rating (note, this is not going to be always the right way to check for the errors)

```
unique(movies4$imdbRating)

## [1] "4.3"      "4.7"      "5.2"      "7.7"      "12"       "7.4"      "6.1"
## [8] "6.6"      "6.5"      "5.5"      "6.8"      "#DIV/0!"   "8"       "6.3"
## [15] "5.6"      "#NAME?"    "3.6"      "5.8"      "4.6"      "4.9"      "4.4"
## [22] "5.9"      "7.8"      "7"        "6.7"      "7.3"      "11"      "8.1"
## [29] "5.3"      "6.2"      "3.2"      "7.2"      "10.5"     "4.2"      "6"
## [36] "4.8"      "7.1"      "5.7"      "4.5"      "2.1"      "6.4"      "8.5"
## [43] "6.9"      "7.6"      "5.1"      "5.4"      "11.6"     "8.2"      NA
## [50] "7.9"      "3.4"      "8.3"      "4.1"      "7.5"      "4"       "5"
## [57] "8.6"      "9.3"      "2.7"      "3.5"      "8.7"      "3.3"      "3.8"
## [64] "8.9"      "3.9"      "9"        "9.2"      "8.4"      "2.8"      "8.8"
## [71] "2.2"      "3"        "3.7"      "3.1"      "1.6"      "2.9"      "2.4"
## [78] "2.3"      "2.6"      "1.9"
```

because we have string in the variable, the whole variable is coerced to character

Solution:

- With `read.csv` R understands the following characters as being missing values: empty cell and `NA`
- However Excel might generate some other strings such as `#DIV/0` when the number is divided by 0.
- Now you would like to specify these values as missing values as well.

Data Cleaning

```
movies4 <- read.csv("movies4c.csv", stringsAsFactors = F,
                     na.strings = c("NA", "#NAME?", "#DIV/0!"))
sapply(movies4, class)
```

##	title	genre_first	year
##	"character"	"character"	"integer"
##	duration	gross_adjusted	budget_adjusted
##	"integer"	"numeric"	"integer"
##	gross	budget	cast_facebook_likes
##	"integer"	"integer"	"integer"
##	reviews	index	Rated
##	"integer"	"numeric"	"character"
##	Genre	Director	Writer
##	"character"	"character"	"character"
##	Actors	Plot	Language
##	"character"	"character"	"character"
##	Country	Awards	Metascore
##	"character"	"character"	"integer"
##	imdbRating	imdbVotes	Production
##	"numeric"	"character"	"character"
##	DVD	Release	Release_Month
##	"character"	"character"	"integer"
##	Release_Day	Release_year	OscarWon
##	"integer"	"integer"	"integer"
##	OtherWin	OscarNom	OtherNom
##	"integer"	"integer"	"integer"

Data Cleaning

- We know that `imdbRating` is a positive number within the interval [0:10]
- However when we run `summary`, we can see that the max is 12
- This is inconsistent with what we know about the data
- Most probably a human error

```
summary(movies4$imdbRating)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.    NA's
## 1.600   5.800  6.500  6.394   7.100 12.000     30
```

Data Cleaning

You can transform all values for imdbRating that are above 10 to be NA (missing values)

```
movies4$imdbRating<-ifelse(movies4$imdbRating>10,NA, movies4$imdbRating)
summary(movies4$imdbRating)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.    Max.    NA's
## 1.600   5.800  6.500  6.388  7.100  9.300    34
```

Data Cleaning

- Human made errors are not rare in datasets.
- Here for the movies genre we have **Action** and **action**, two different words, thus two different categories

```
unique(movies4$genre_first)
```

```
## [1] "Action"      "Comedy"       "Adventure"     "Crime"        "Drama"
## [6] NA            "action"       "Documentary"   "Crime"        "Biography"
## [11] "Drama"       "Horror"       "Fantasy"       "Sci-Fi"        "Animation"
## [16] "Family"       "Mystery"      "Western"       "Musical"      "Romance"
## [21] "Thriller"
```

If you look at the frequencies, you're going to get false picture

```
table(movies4$genre_first)
```

```
##
##          Crime    action    Action Adventure Animation Biography
##          13        31      690      281        35      135
##          Comedy   Crime Documentary Drama   Drama   Family
##          843      186      25      469        15       3
##          Fantasy Horror Musical Mystery Romance Sci-Fi
##          26        127      2       16        2       7
##          Thriller Western
##          1          2
```

Data Cleaning

Solution: make everything lower or upper case

```
movies4$genre_first <- toupper(movies4$genre_first)
table(movies4$genre_first)

##          CRIME      ACTION ADVENTURE ANIMATION BIOGRAPHY COMEDY
##          13        721     281       35      135      843
##          CRIME DOCUMENTARY      DRAMA      DRAMA   FAMILY FANTASY
##          186         25      469       15        3      26
##          HORROR    MUSICAL MYSTERY ROMANCE SCI-FI THRILLER
##          127         2       16        2        7        1
##          WESTERN
##          2
```

Now we have only 1 ACTION category

Data Cleaning

- Still have a problem: two categories DRAMA
- Problem: trailing whitespaces on right

```
movies4$genre_first <- toupper(movies4$genre_first)
table(movies4$genre_first)
```

```
##
##          CRIME      ACTION     ADVENTURE     ANIMATION     BIOGRAPHY     COMEDY
##          13        721        281          35        135        843
##          CRIME DOCUMENTARY      DRAMA      DRAMA       FAMILY       FANTASY
##          186         25        469          15          3        26
##          HORROR      MUSICAL     MYSTERY     ROMANCE      SCI-FI      THRILLER
##          127          2        16            2          7          1
##          WESTERN
##          2
```

DRAMA	DRAMA
469	15

Data Cleaning

use trimws() – trim white space

```
movies4$genre_first <- trimws(movies4$genre_first)
table(movies4$genre_first)
```

```
##
##          ACTION      ADVENTURE      ANIMATION      BIOGRAPHY      COMEDY      CRIME
##          721           281            35           135           843          199
## DOCUMENTARY      DRAMA      FAMILY      FANTASY      HORROR      MUSICAL
##          25           484            3            26           127             2
## MYSTERY      ROMANCE      SCI-FI      THRILLER      WESTERN
##          16            2            7             1             2
```

Fixed !

Data Cleaning

The variable is coerced to character

```
unique(movies4$imdbVotes) [1:50]
```

```
## [1] "16367"    "53943"    "49098"    "426786"   "353948"   "12912"    "235120"  
## [8] "76203"    "158735"   "167217"   "68,302"   "90107"    "16142"    "46,727"  
## [15] "125997"   "68995"    "72,358"   "39,918"   "16721"    "24298"    "168657"  
## [22] "24635"    "44225"    "17,886"   "293665"   "81296"    "59,902"    "158266"  
## [29] "572117"   "578,004"   "310209"   "405777"   "501463"   "38923"    "95386"  
## [36] "49315"    "305140"   "164,004"  "517415"   "71359"    "13751"    "19619"  
## [43] "57065"    "9131"     "46665"    "72,923"   "5617"     "6091"     "39552"  
## [50] "32705"
```



Problem with 1000 separator

Data Cleaning

If you try to make it numeric, you will get NA with problematic cells

```
x1 <- as.numeric(movies4$imdbVotes)
x1[1:50]

## [1] 16367 53943 49098 426786 353948 12912 235120 76203 158735 167217
## [11] NA 90107 16142 NA 125997 68995 NA NA 16721 24298
## [21] 168657 24635 44225 NA 293665 81296 NA 158266 572117 NA
## [31] 310209 405777 501463 38923 95386 49315 305140 NA 517415 71359
## [41] 13751 19619 57065 9131 46665 NA 5617 6091 39552 32705
```

Data Cleaning

Solution: Delete comma then transform to numeric

```
movies4$imdbVotes <- gsub("[,]", "", movies4$imdbVotes)
movies4$imdbVotes <- as.numeric(movies4$imdbVotes)
summary(movies4$imdbVotes)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.      NA's
##        7    21425   61163  121245  145429 1914618         27
```

Deletes comma, we will talk about this function next week in more details

```
movies4$imdbVotes <- gsub("[,]", "", movies4$imdbVotes)
```