

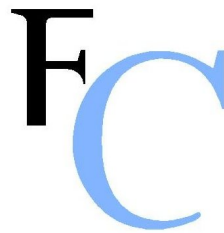
Data Visualization in R

L. Torgo

ltorgo@fc.up.pt

Departamento de Ciência de Computadores / Faculdade de Ciências
Universidade do Porto

Mar, 2021



FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Introduction

Motivation for Data Visualization

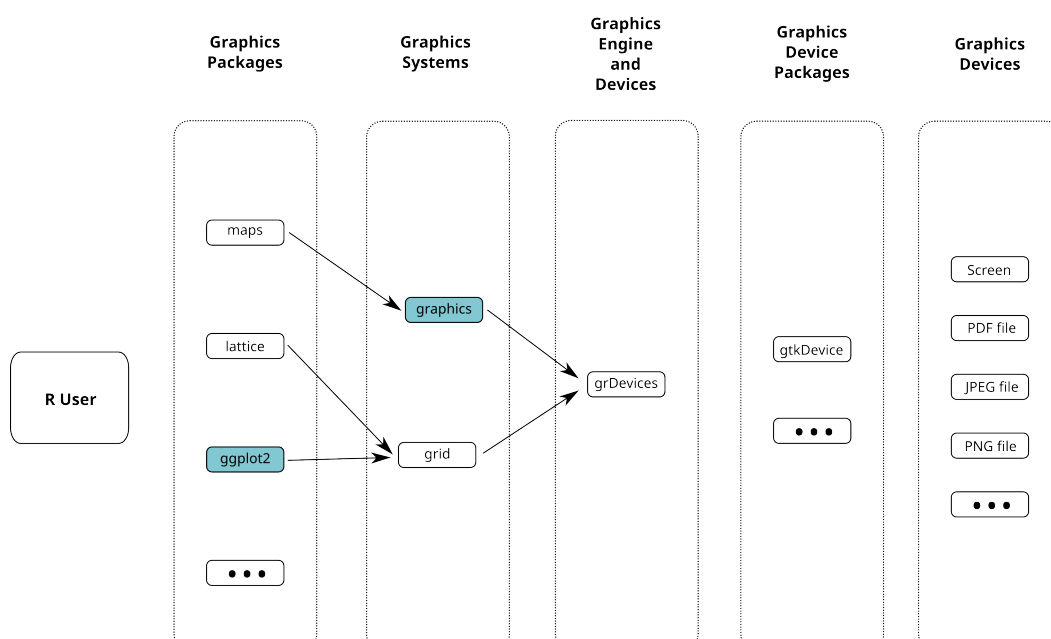
- Humans are outstanding at detecting patterns and structures with their eyes
- Data visualization methods try to explore these capabilities
- In spite of all advantages visualization methods also have several problems, particularly with very large data sets

Outline of what we will learn

- Tools for visualizing amounts
- Tools for visualizing distributions of values
- Tools for visualizing proportions
- Tools for visualizing x-y relationships
- Multivariate visualization tools

Introduction

R Graphics



Standard Graphics (the `graphics` package)

- R standard graphics, available through package **graphics**, includes several functions that provide standard statistical plots, like:
 - Scatterplots
 - Boxplots
 - Piecharts
 - Barplots
 - etc.
- These graphs can be obtained typically by a single function call
 - Example of a scatterplot

```
plot(1:10, sin(1:10))
```

- These graphs can be easily augmented by adding several elements to these graphs (lines, text, etc.)

Graphics Devices

- R graphics functions produce output that depends on the active graphics device
- The default and more frequently used device is the **screen**
- There are many more graphical devices in R, like the **pdf** device, the **jpeg** device, etc.
- The user just needs to open (and in the end close) the graphics output device she/he wants. R takes care of producing the **type of output required by the device**
- This means that to produce a certain plot on the screen or as a GIF graphics file the R code is exactly the same. You only need to open the target output device before!
- Several devices may be open at the same time, but only one is the **active** device

A few examples

A scatterplot

```
plot(seq(0, 4*pi, 0.1), sin(seq(0, 4*pi, 0.1)))
```

The same but stored on a jpeg file

```
jpeg('exp.jpg')
plot(seq(0, 4*pi, 0.1), sin(seq(0, 4*pi, 0.1)))
dev.off()
```

And now as a pdf file

```
pdf('exp.pdf', width=6, height=6)
plot(seq(0, 4*pi, 0.1), sin(seq(0, 4*pi, 0.1)))
dev.off()
```

Introduction to ggplot

Package ggplot2

- Package `ggplot2` implements the ideas created by Wilkinson (2005) on a grammar of graphics
- This grammar is the result of a theoretical study on what is a statistical graphic
- `ggplot2` builds upon this theory by implementing the concept of a layered grammar of graphics (Wickham, 2009)
- The grammar defines a statistical graphic as:
 - a mapping from data into **aesthetic attributes** (color, shape, size, etc.) of **geometric objects** (points, lines, bars, etc.)

L. Wilkinson (2005). The Grammar of Graphics. Springer.

H. Wickham (2009). A layered grammar of graphics. Journal of Computational and Graphical Statistics.

The Basics of the Grammar of Graphics

■ Key elements of a statistical graphic:

- data
- aesthetic mappings
- geometric objects
- statistical transformations
- scales
- coordinate system
- faceting

Aesthetic Mappings

■ Controls the relation between data variables and graphic properties

- map the *Temperature* variable of a data set into the *x coordinate* in a scatter plot
- map the *Species* of a plant into the *colour* of dots in a graphic
- map the *Citizenship* of a person into the *shape* of a dot
- map the *Age* of a car into the *line width* of lines in a graphic
- etc.

Geometric Objects / Graphical Elements

- Controls what is shown in the graphics, e.g.
 - show each observation by a point using the aesthetic mappings that map two variables in the data set into the x , y coordinates of the plot
 - etc.

Statistical Transformations

- Allows us to calculate and do statistical analysis over the data in the plot
 - Use the data and approximate it by a regression line on the x , y coordinates
 - Count occurrences of certain values
 - etc.

Scales

- Decide how the data values are mapped into the aesthetics properties
 - A scale defines a one to one mapping between the data values and the values of some aesthetical property

Coordinate System

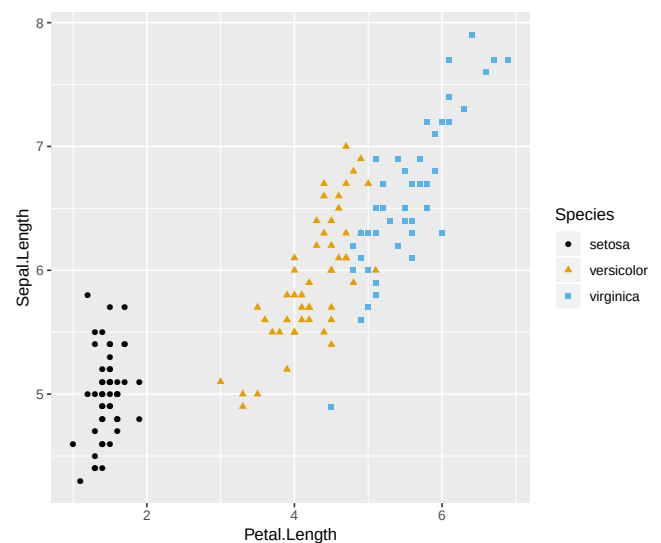
- The coordinate system used to plot the data
 - Cartesian
 - Polar
 - etc.

Faceting

- Split the data into sub-groups and draw sub-graphs for each group

A Simple Example

```
data(iris)
library(ggplot2)
library(ggthemes)
ggplot(iris,
  aes(x=Petal.Length, y=Sepal.Length,
    colour=Species, shape=Species)
) +
  geom_point() +
  scale_colour_colorblind()
```



Visualizing Amounts

Visualizing Amounts

Visualizing Amounts

- Comparing the magnitude of a set of numbers
- May include comparisons across different groups

Bar plots

- Displays the set of values as heights of different bars

Example data set: wins and losses of NBA teams of Atlantic Division

Team	Wins	Losses	Season
Toronto Raptors	58	24	2018_19
Philadelphia 76ers	51	31	2018_19
Boston Celtics	49	33	2018_19
Brooklyn Nets	42	40	2018_19

Bar plots

Two frequent settings

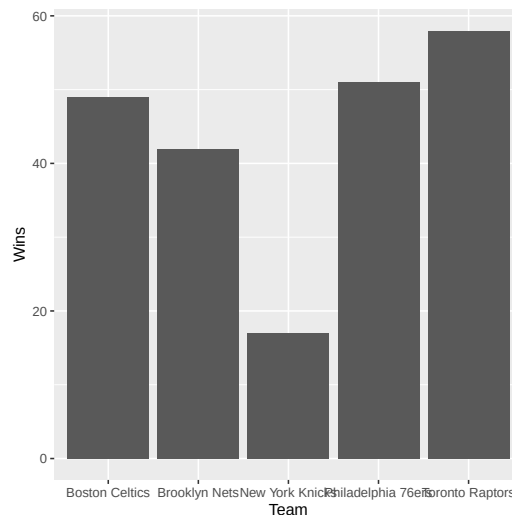
- 1 The values to show are already in the data set
- 2 The values to show need to be computed from the data

Bar plots

Values in the data set

■ Showing the number of wins on 2018/19

```
library(ggplot2)
ggplot(filter(nba, Season=="2018_19"), aes(x=Team, y=Wins)) + geom_col()
```

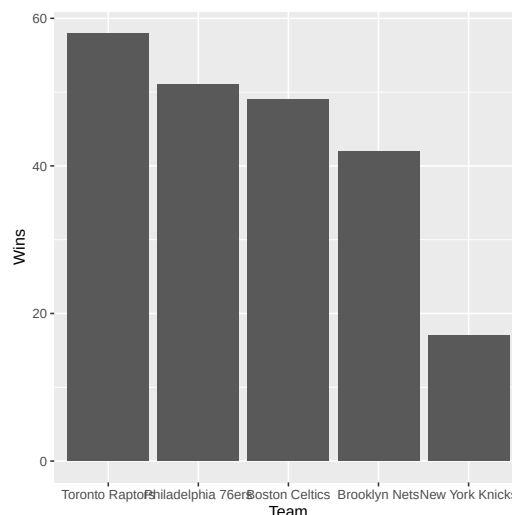


Bar plots

Ordering the bars

■ If the order of the values (team names) has no meaning then the bars should be order by decreasing value for better readability

```
ggplot(filter(nba, Season=="2018_19"), aes(x=reorder(Team, -Wins), y=Wins)) +  
  geom_col() + xlab("Team")
```

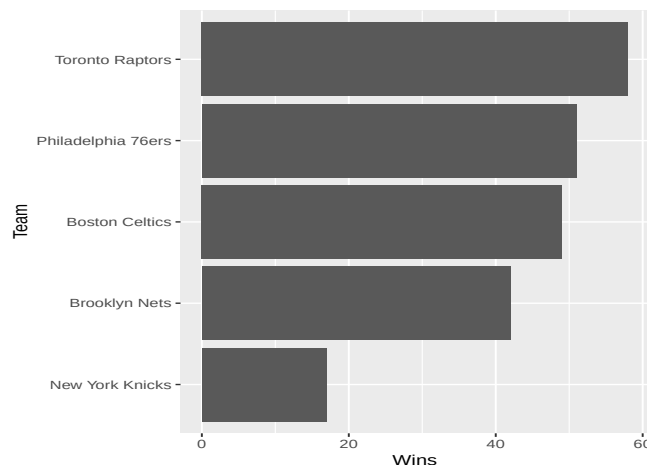


Bar plots

Horizontal bars

- When the X labels are long or are far too many, horizontal bars may be more readable

```
ggplot(filter(nba, Season=="2018_19"), aes(x=reorder(Team, Wins), y=Wins)) +  
  coord_flip() + geom_col() + xlab("Team")
```

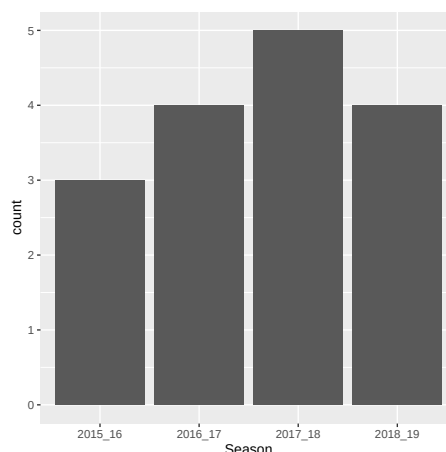


Bar plots

Values need to be calculated

- Showing how many teams had a percentage of wins larger than 30% per season

```
ggplot(filter(nba, Wins/(Wins+Losses) > .3), aes(x=Season)) + geom_bar()
```



NOTE: ordering the bars here would be wrong as the labels have an implicit ordering (time)

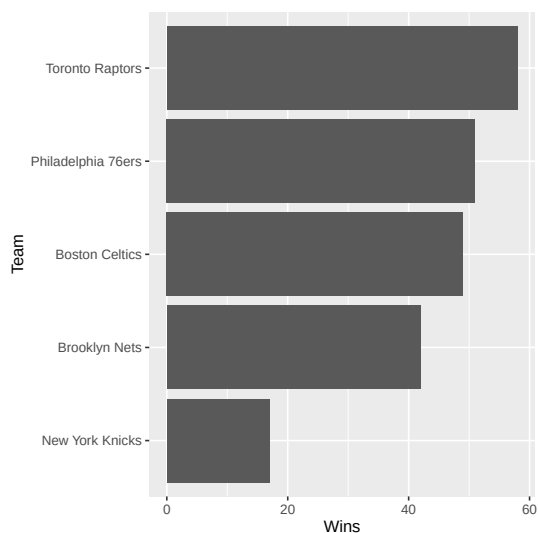
Bar plots

Problems

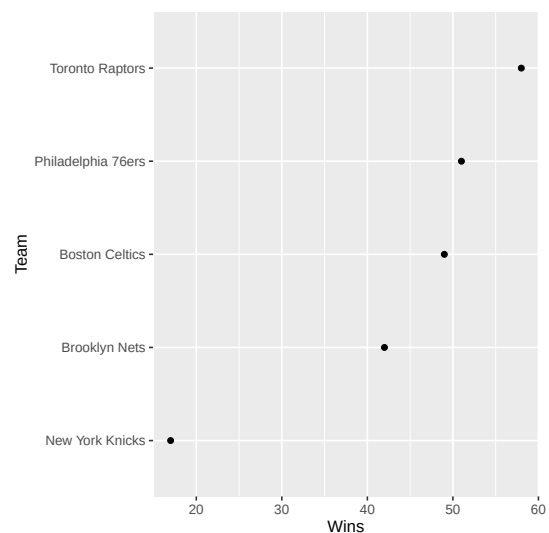
- Bars always must start from zero
- This may create a problem known as *zero baseline*
- When the bars have similar height, due to the start on zero, the differences are hard to perceive
- Barplots should not be used in these situations - dot plots are better representations

Dot plots

```
ggplot (filter (nba, Season=="2018_19"),
  aes (x=reorder (Team,Wins), y=Wins)) +
  geom_col () + coord_flip () + xlab ("Team")
```



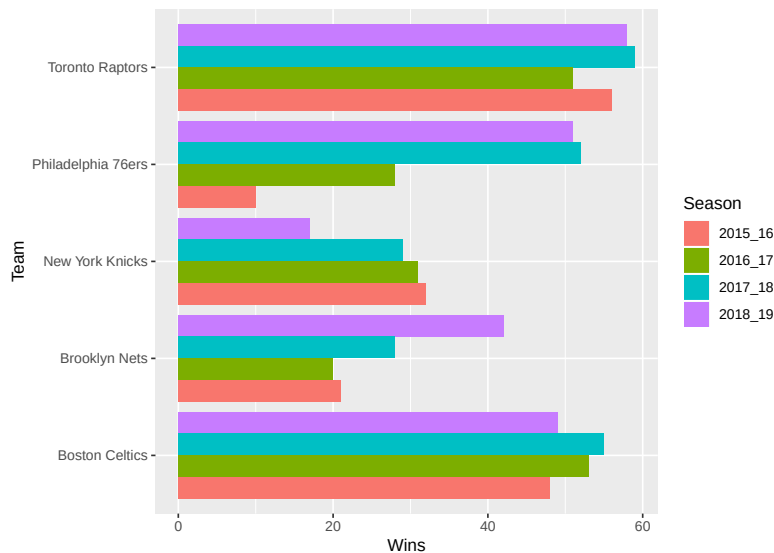
```
ggplot (filter (nba, Season=="2018_19"),
  aes (x=reorder (Team,Wins), y=Wins)) +
  geom_point () + coord_flip () + xlab ("Team")
```



Grouped bar plots

- Sometimes we want to contrast amounts for different groups
- Number of wins of the Teams along the Seasons

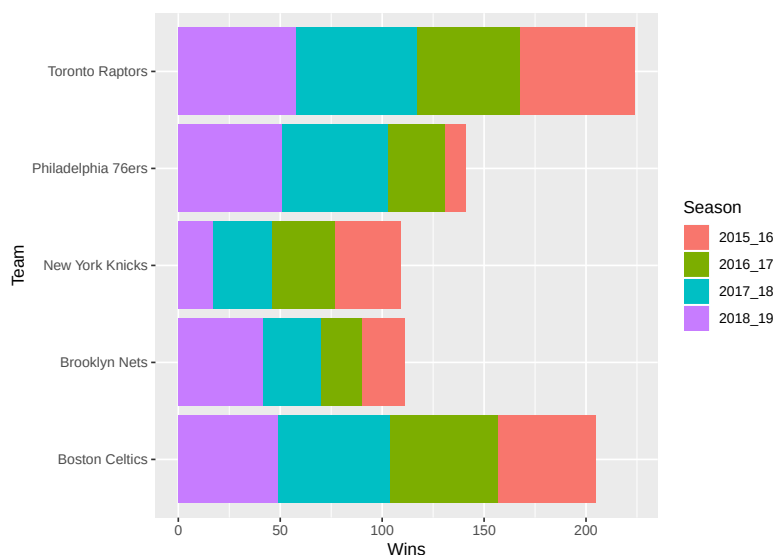
```
ggplot(nba, aes(x=Team, y=Wins, fill=Season)) + geom_col(position="dodge") + coord_flip()
```



Stacked bar plots

- Sometimes we want to contrast amounts for different groups
- Number of wins of the Teams along the Seasons

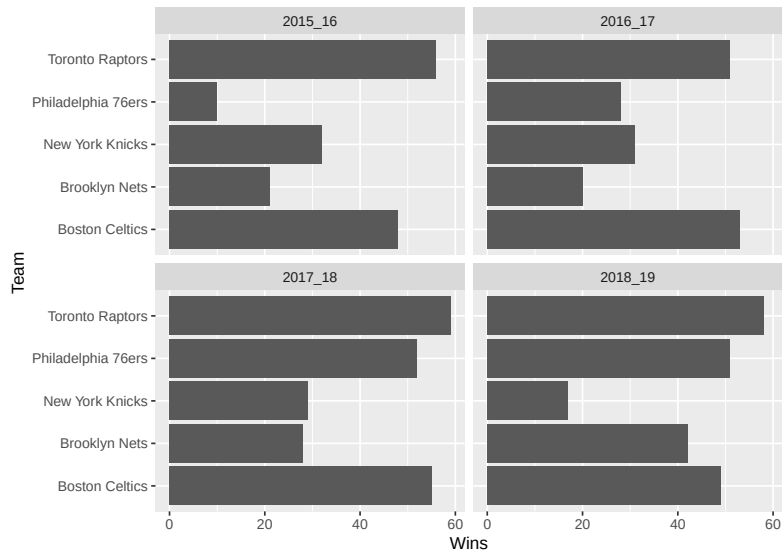
```
ggplot(nba, aes(x=Team, y=Wins, fill=Season)) + geom_col() + coord_flip()
```



Facets

■ Yet another alternative of contrasting different groups

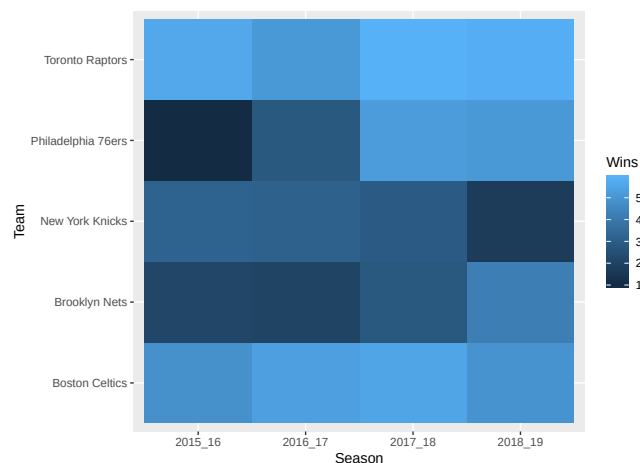
```
ggplot(nba, aes(x=Team, y=Wins)) + geom_col() + facet_wrap(~Season) + coord_flip()
```



Heatmaps

■ Heatmaps can also be used to contrast different groups

```
ggplot(nba, aes(x=Season, y=Team, fill=Wins)) + geom_tile()
```



Visualizing Distributions of Variables

Visualizing Distributions

Visualizing Distributions

- The goal of these graphs is to understand how the values of a certain variable are distributed in our data set
- We will distinguish two common setups
 - Visualize a single distribution
 - Visualize multiple distributions

Histograms

- Histograms visually look like barplots but they are rather different
- The number of bars in an histogram is determined by a **binning process** that divides the range of a numeric variable into a set of bins
- For each bin the number of values fitting inside of the bin is **counted** and that determines the height of the respective bar

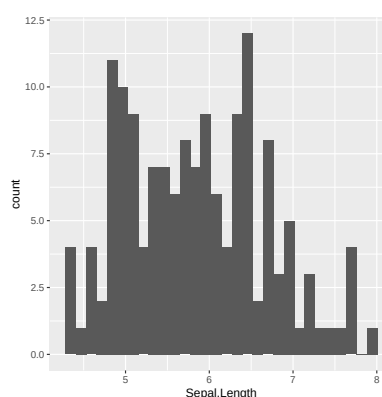
Histograms

An example with the Iris dataset

- What is the distribution of the values of `Sepal.Length` in the **Iris** data set?

```
library(ggplot2)
data(iris)
ggplot(iris, aes(x=Sepal.Length)) + geom_histogram()

## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```

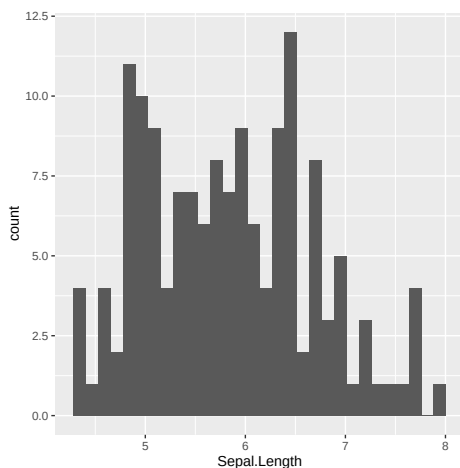


Histograms

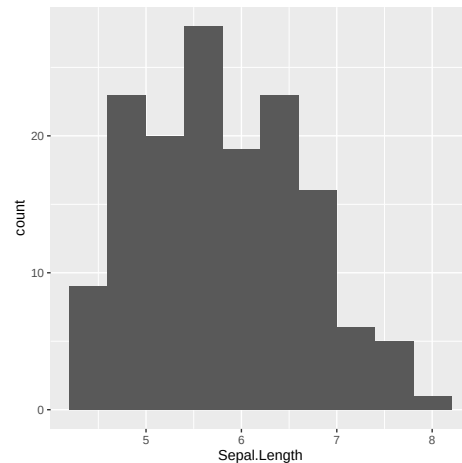
Effects of changing the number of bins

By default the range is divided into 30 bins

```
ggplot(iris, aes(x=Sepal.Length)) +  
  geom_histogram()  
  
## 'stat_bin()' using 'bins = 30'. Pick  
better value with 'binwidth'.
```



```
ggplot(iris, aes(x=Sepal.Length)) +  
  geom_histogram(bins=10)
```



Note: it is a good idea to try several values

Density plots

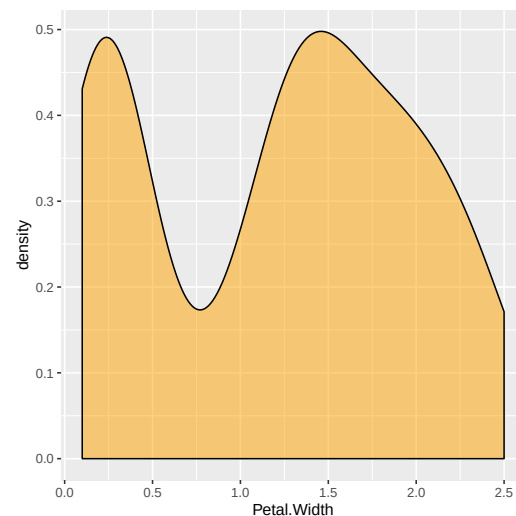
- An alternative to histograms is to use density plots
- They obtain an approximation of the probability density function of the variable
- They are usually obtained using kernel density estimators (frequently the Gaussian kernel)
- These estimators typically depend on a parameter known as **bandwidth** that controls how sensitive are the estimates at each point
 - Changing its value has a similar effect as changing the number of bins in histograms

Density plots

An example with the Iris dataset

- What is the distribution of the values of `Petal.Width` in the **Iris** data set?

```
ggplot(iris, aes(x=Petal.Width)) +  
  geom_density(fill="orange", alpha=0.5)
```



Note: by default this uses a bandwidth determined by a rule whose theoretical justification can be seen at the reference provided in the help page of function `bw.nrd0()`

Boxplots

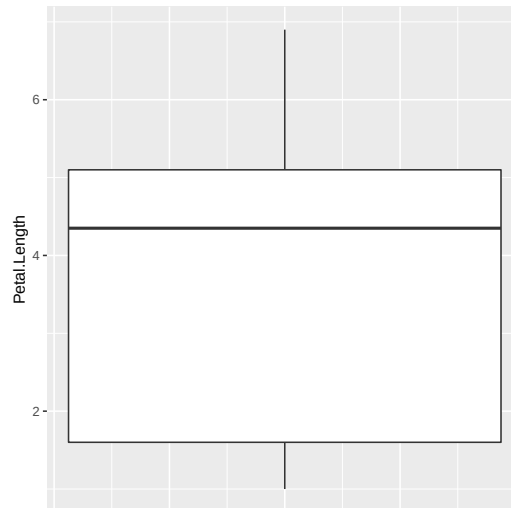
- Boxplots provide a synthetic representation of a distribution by showing a few summary statistics
- This obviously implies some loss of information on the full distribution
- Still, they provide interesting features like for instance identifying outliers
- They are better suited for comparing multiple distributions but they can also be used for a single distribution

Boxplots

An example with the Iris dataset

- What is the distribution of the values of `Petal.Length` in the **Iris** data set?

```
ggplot(iris, aes(y=Petal.Length)) + geom_boxplot() +  
  theme(axis.ticks.x = element_blank(), axis.text.x = element_blank())
```



Empirical cumulative distribution functions (ECDF's)

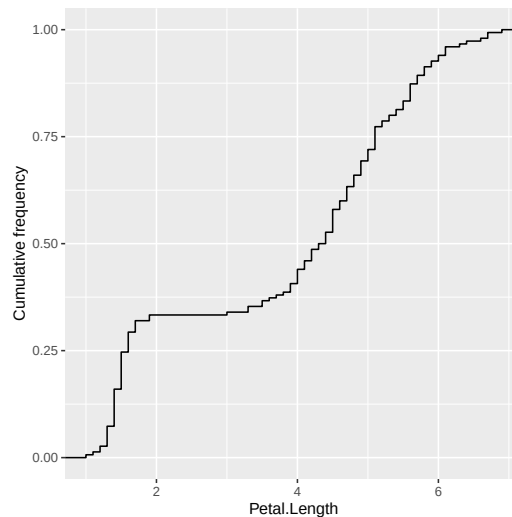
- ECDF's are interesting tools to visualize the distribution of a continuous variable
- They are obtained by ordering the sample values and then plotting them against the cumulative frequency proportion (a value between 0 and 1)

ECDFs

An example with the Iris dataset

- Plot the empirical cumulative distribution function of `Petal.Length` in the **Iris** data set?

```
ggplot(iris, aes(x=Petal.Length)) + stat_ecdf() + ylab("Cumulative frequency")
```



Quantile-quantile Plots (Q-Q plots)

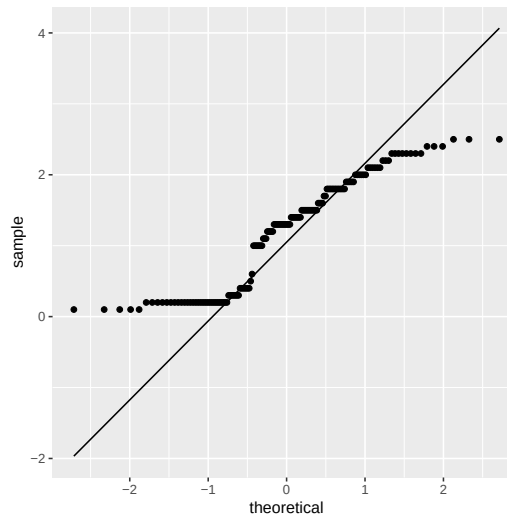
- These plots compare the sample quantiles of a continuous variable with the theoretical quantiles of a known distribution
- They are useful to check if some variable follows a certain distribution
- Frequently we compare the sample against the Normal distribution

Q-Q plots

An example with the Iris dataset

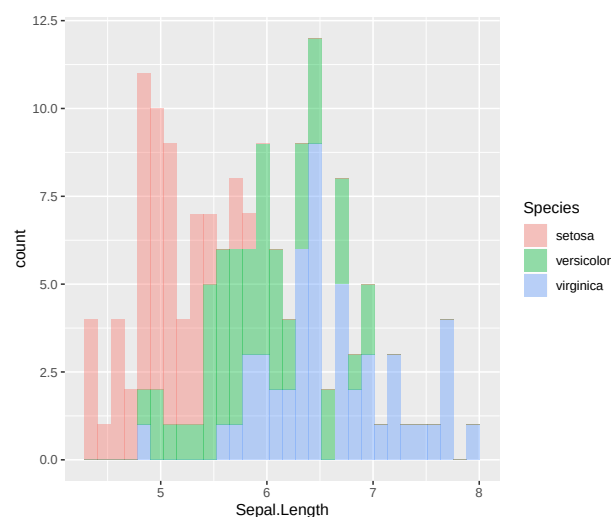
- Is it reasonable to think the variable `Petal.width` of the **Iris** data set follows a Normal distribution?

```
ggplot(iris, aes(sample = Petal.Width)) + stat_qq() + stat_qq_line()
```



Multiple Distributions using Histograms

```
ggplot(iris, aes(x=Sepal.Length, fill=Species)) +  
  geom_histogram(alpha=0.4)  
  
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```

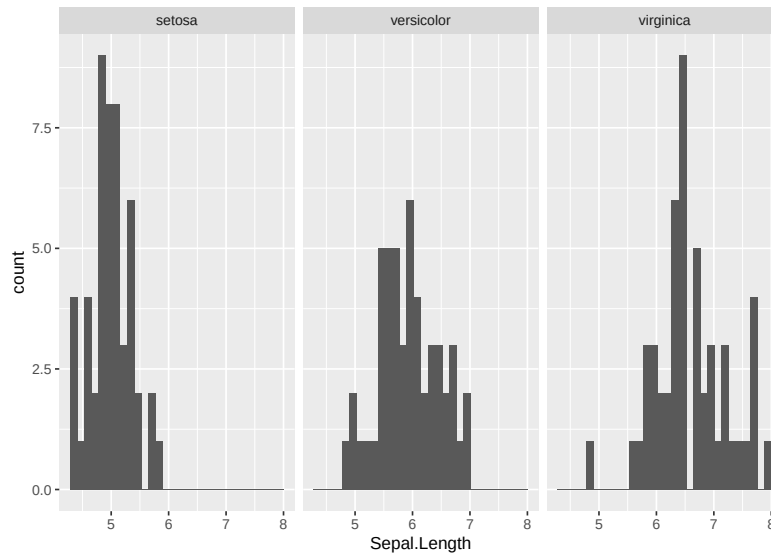


- **This is bad - avoid!**
- Overplotting causes visualization problems

Multiple Distributions using Histograms - 2

```
ggplot(iris, aes(x=Sepal.Length)) + geom_histogram() + facet_wrap(~ Species)
```

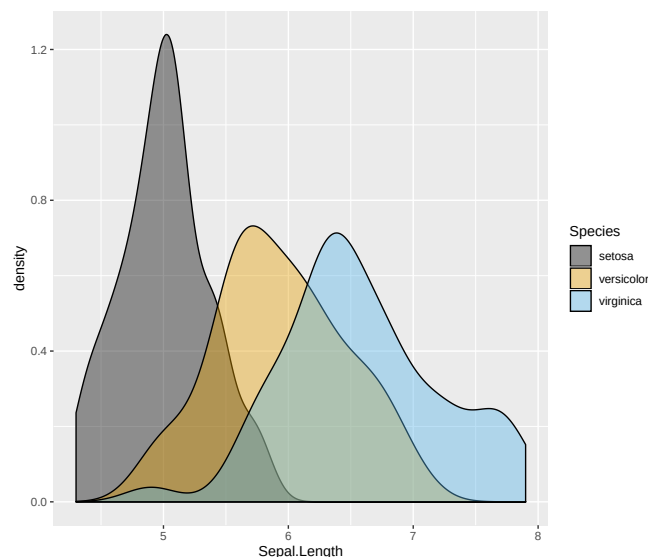
```
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



■ This is better

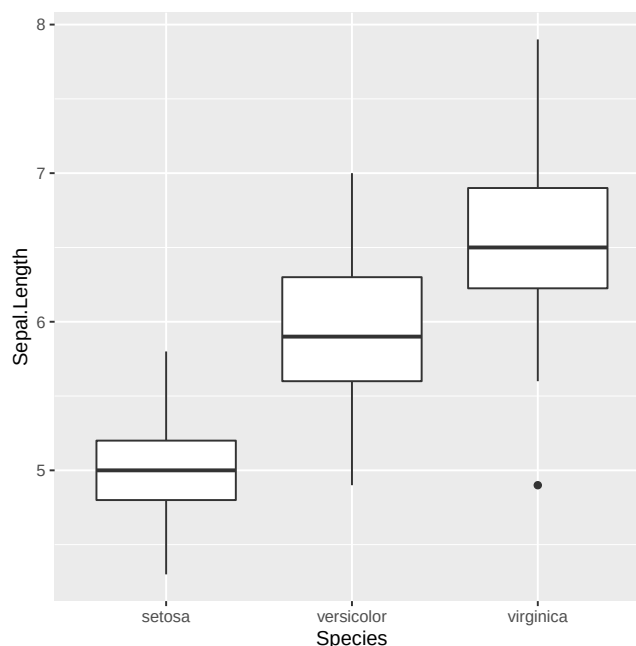
Multiple Distributions using Density Plots

```
library(ggthemes)
ggplot(iris, aes(x=Sepal.Length, fill=Species)) +
  geom_density(alpha=0.4) + scale_fill_colorblind()
```



Multiple Distributions with Box Plots

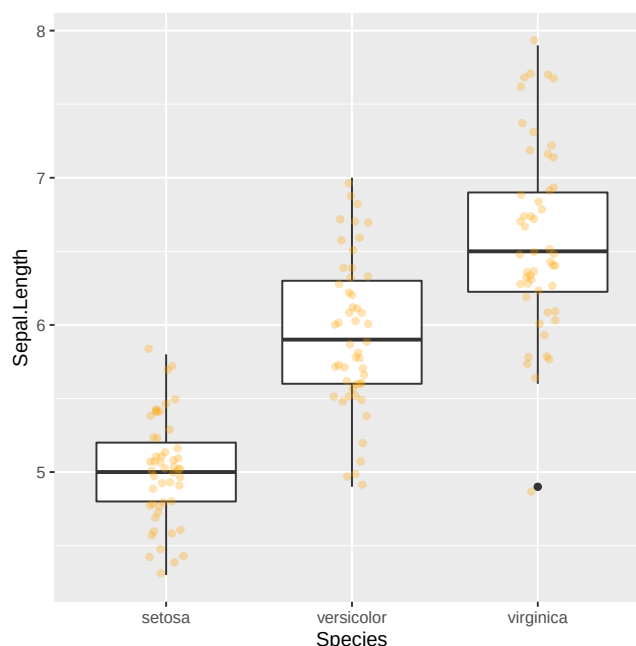
```
ggplot(iris, aes(x=Species, y=Sepal.Length)) +  
  geom_boxplot()
```



Multiple Distributions with Box Plots

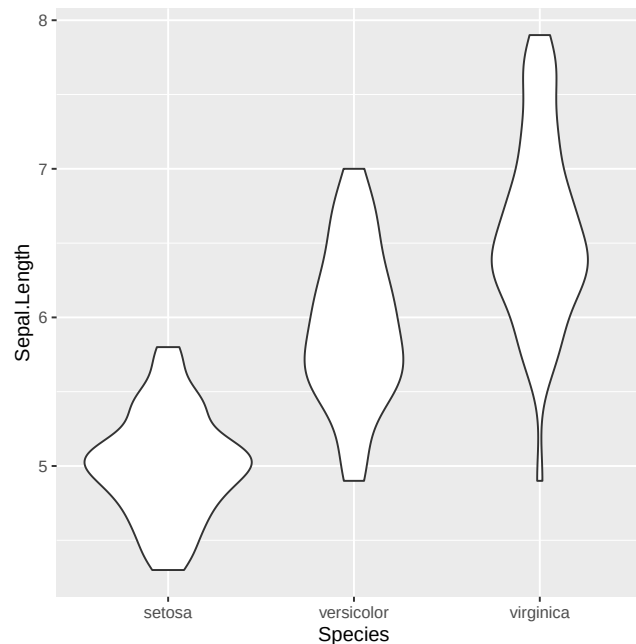
Adding information on the amount of values

```
ggplot(iris, aes(x=Species, y=Sepal.Length)) +  
  geom_boxplot() + geom_jitter(color="orange", alpha=0.3, width=0.1)
```



Multiple Distributions with Violin Plots

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

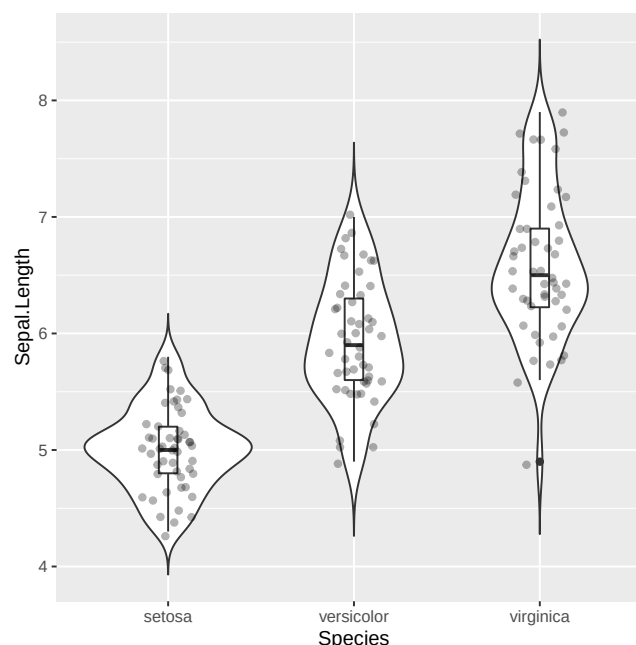


Note: similar to density plots...

Variations on Violin Plots

```
ggplot(iris, aes(x=Species, y=Sepal.Length)) + geom_violin(trim=FALSE) +  
  geom_boxplot(width=0.1) + geom_jitter(position=position_jitter(0.15), alpha=0.3)
```

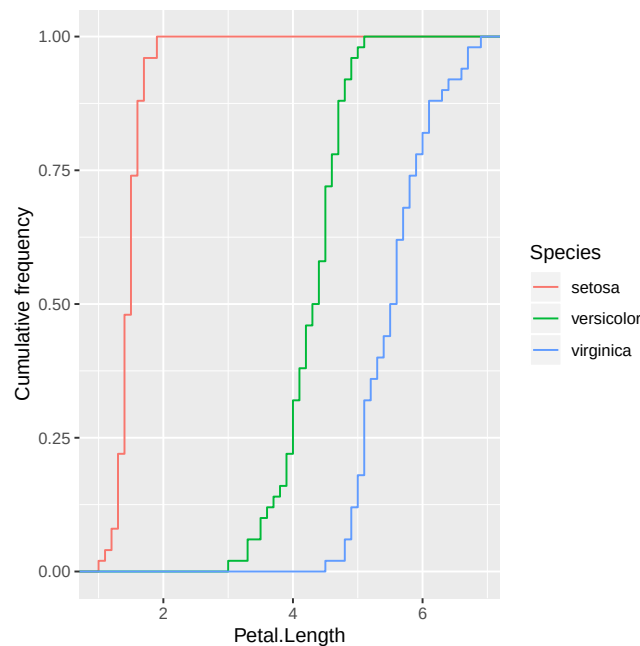
```
## Warning in sample.int(.Machine$integer.max, 1L): '.Random.seed[1]' is not a valid integer,  
so ignored
```



Note: similar to density plots...

Multiple Distributions with ECDFs

```
ggplot(iris, aes(x=Petal.Length, color=Species)) + stat_ecdf() + ylab("Cumulative frequency")
```



Hands on Data Visualization - the Algae data set

Using the Algae data set from package `DMwR` answer to the following questions:

- 1 Create a graph that you find adequate to show the distribution of the values of algae `a6`
- 2 Show the distribution of the values of `size`
- 3 Check visually if it is plausible to consider that `oP04` follows a normal distribution

Visualizing Proportions

Visualizing Proportions

Visualizing Proportions

- Goal: show how a certain information breaks down into proportions
 - e.g. how the clients of some company are distributed across different genders
- We will study three main tools for handling properties, each with pros and cons
 - Pie charts
 - Stacked barcharts
 - Bar plots

Piecharts

- These plots associate the proportion values with slices of a pie
- Reading the values thus involves comparing areas of slices
- Piecharts are frequently criticized by being harder on the human eye, on the basis that it is easier to compare height of bars than areas of slices
- Technical note: pie charts are much easier to obtain with R base graphs than with ggplot2

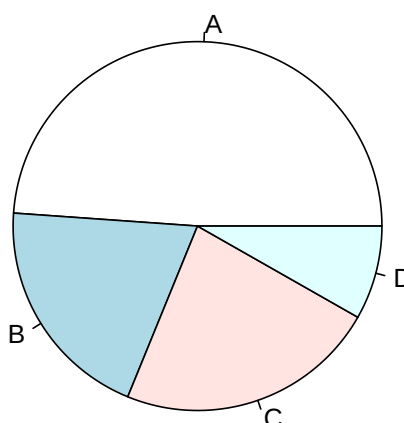
Example data set: the number of seats of four political parties in an election

Party	Seats
A	196
B	80
C	92
D	33

Pie Charts

- Comparing the parliament seats of the parties

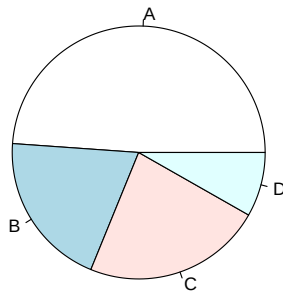
```
pie(dat$Seats, labels=dat$Party)
```



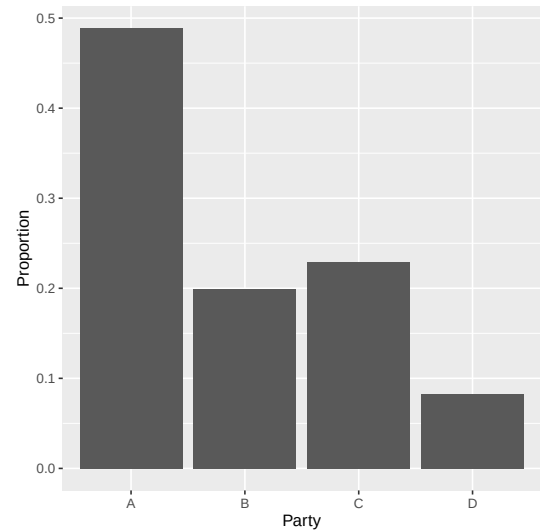
Pie Charts

Comparing with a barplot

```
pie(dat$Seats, labels=dat$Party)
```

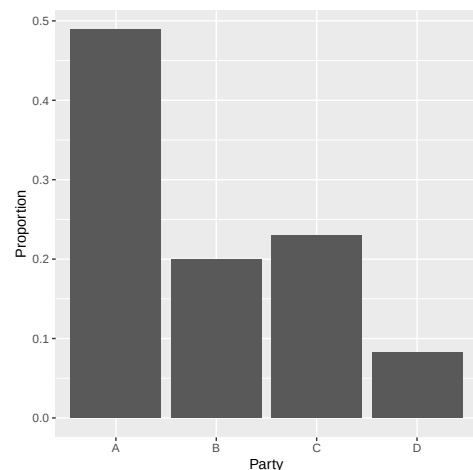
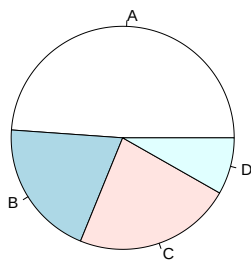


```
ggplot(mutate(dat, Prop=Seats/sum(dat$Seats)),  
  aes(x=Party, y=Prop)) + geom_col() +  
  ylab("Proportion")
```



Pie Charts

Discussion

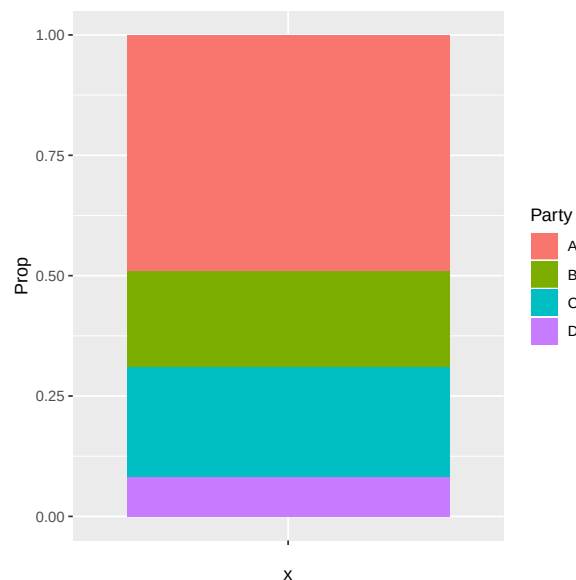


- On the pie it is not so easy to see that C has more seats than B
- On the bar plot it is not easy to see that A and D together have the majority of the seats and thus could form a government

Stacked barcharts

■ Comparing the parliament seats of the parties

```
## Note sure why this is not working....  
## ggplot(mutate(dat, Prop=Seats/sum(dat$Seats), ypos=cumsum(Prop)-0.5*Prop), aes(x="", y=Prop, fill=Party))  
## Removing the labels till I understand why it works in RStudio and not on the slides!  
ggplot(mutate(dat, Prop=Seats/sum(dat$Seats)),  
        aes(x="", y=Prop, fill=Party)) + geom_col()
```



X-Y Associations

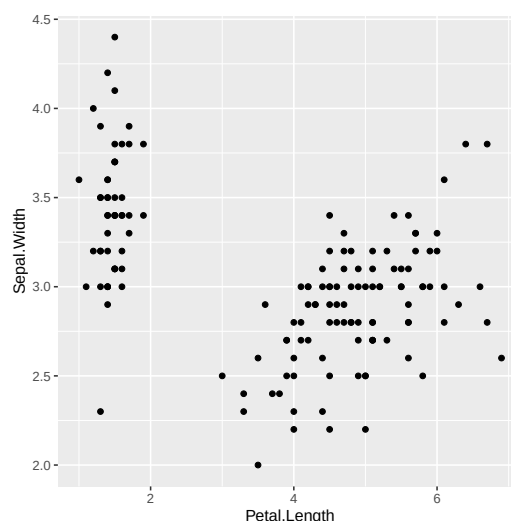
X-Y Associations

- Goal: understand how two continuous variables related with each other
 - e.g. is the growth of Petal.Length associated with a growth in Sepal.Width?

Scatter Plots

- The simplest way of checking this type of relationships is through a scatter plot
- Each of the numeric variables is associated with one of the plot axis

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

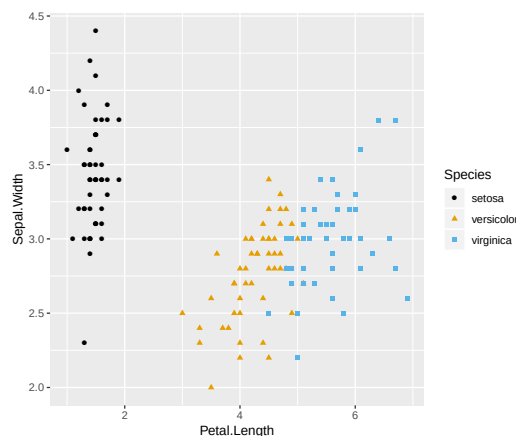


Scatter Plots

Grouped associations

- Sometimes the association may be different for different sub-groups of data

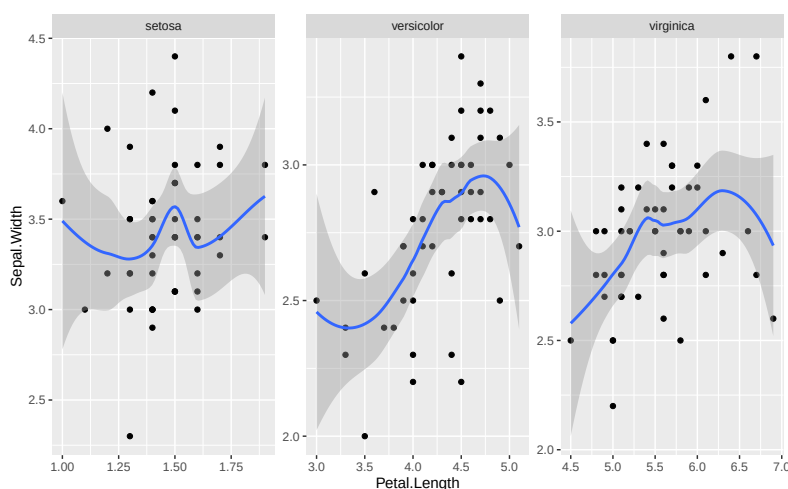
```
library(ggthemes)
ggplot(iris, aes(x=Petal.Length, y= Sepal.Width, color=Species, shape=Species)) +
  geom_point() + scale_color_colorblind()
```



Scatter Plots

Trying to “model” the associations

```
ggplot(iris, aes(x=Petal.Length, y= Sepal.Width)) +
  geom_point() + geom_smooth(method="loess", se=TRUE, level=0.95) +
  facet_wrap(~ Species, scales = "free")
```



Hands on Data Visualization - Algae data set

Using the Algae data set from package `DMwR` answer to the following questions:

- 1 Produce a graph that allows you to understand how the values of NO3 are distributed across the sizes of river
- 2 Try to understand (using a graph) if the distribution of algae a1 varies with the speed of the river

Multiple Associations

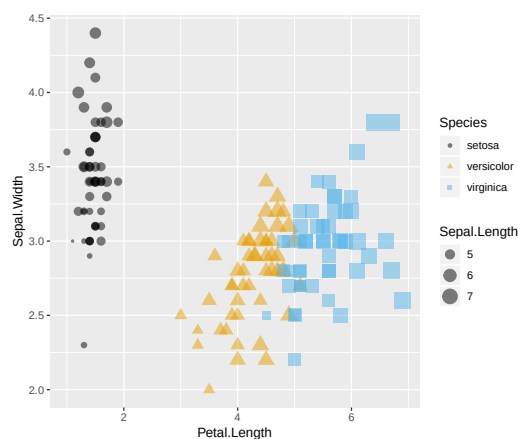
Multiple Associations

- Goal: understand how multiple variables related with each other

Scatter plots for more than two variables

- Using other aesthetic properties to code other variables
- Use with caution, it may get too confusing!

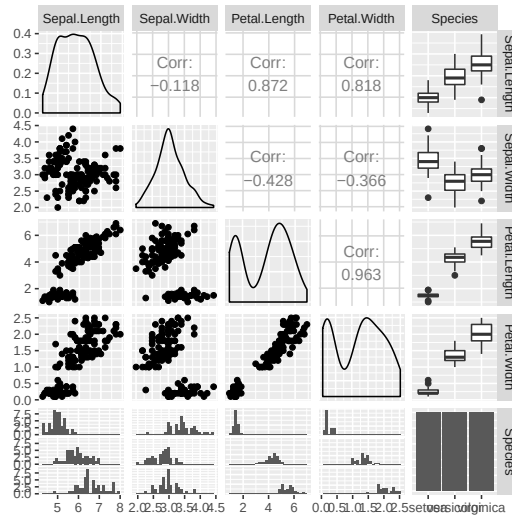
```
ggplot(iris,
  aes(x=Petal.Length, y= Sepal.Width, size=Sepal.Length, color= Species, shape=Species)) +
  geom_point(alpha=0.5) + scale_color_colorblind()
```



Scatterplot matrices through package GGally

- These graphs try to present all pairwise scatterplots in a data set.
- They are unfeasible for data sets with many variables.

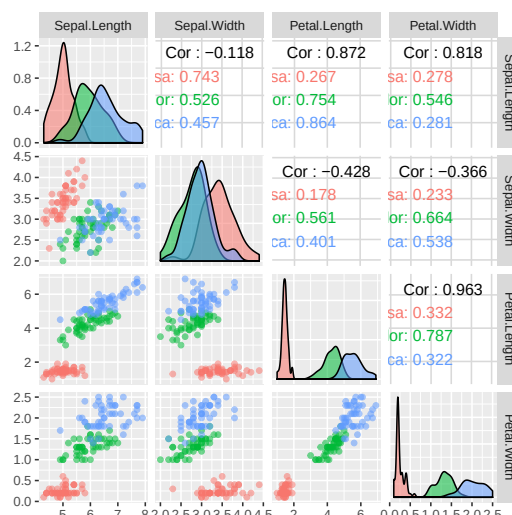
```
library(GGally)
ggpairs(iris)
```



Scatterplot matrices - a more interesting variant

- Differentiating with a nominal variable

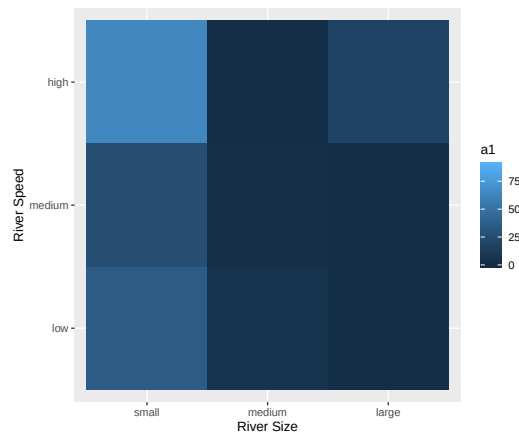
```
ggpairs(iris, columns = 1:4, ggplot2::aes(color=Species, alpha=0.5))
```



Heat Maps for more than two variables

■ Axes are two nominal variables

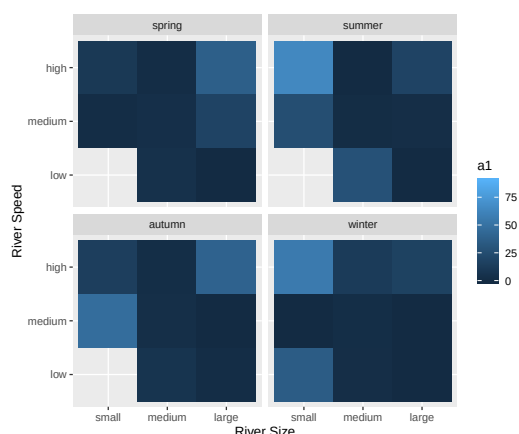
```
data(algae, package="DMwR2")
library(dplyr)
library(forcats)
mutate(algae,
  size = fct_relevel(size, c("small", "medium", "large")),
  speed = fct_relevel(speed, c("low", "medium", "high"))) %>%
ggplot(aes(x = size, y = speed, fill = a1)) +
  geom_tile() + xlab("River Size") + ylab("River Speed")
```



Heat Maps for more than two variables - 2

■ Axes are two nominal variables, and a third as a facet

```
mutate(algae,
  size = fct_relevel(size, c("small", "medium", "large")),
  speed = fct_relevel(speed, c("low", "medium", "high")),
  season = fct_relevel(season, c("spring", "summer", "autumn", "winter"))) %>%
ggplot(aes(x = size, y = speed, fill = a1)) +
  geom_tile() + facet_wrap(~ season) + xlab("River Size") + ylab("River Speed")
```



Correlograms

- For numeric variables we can calculate and represent the correlation value

```
library(GGally)
ggcorr(iris[, -5])
```



```
library(GGally)
ggcorr(iris[, -5], label = TRUE)
```



Parallel Plots

- Parallel plots are also interesting for visualizing a full data set

```
library(GGally)
ggparcoord(iris, columns = 1:4, groupColumn = "Species", alphaLines = 0.5)
```

