VISUALISATION IN R
(FOCUS ON GGPLOTS)

Rstats Friday! | Dr. Erin I. Walsh

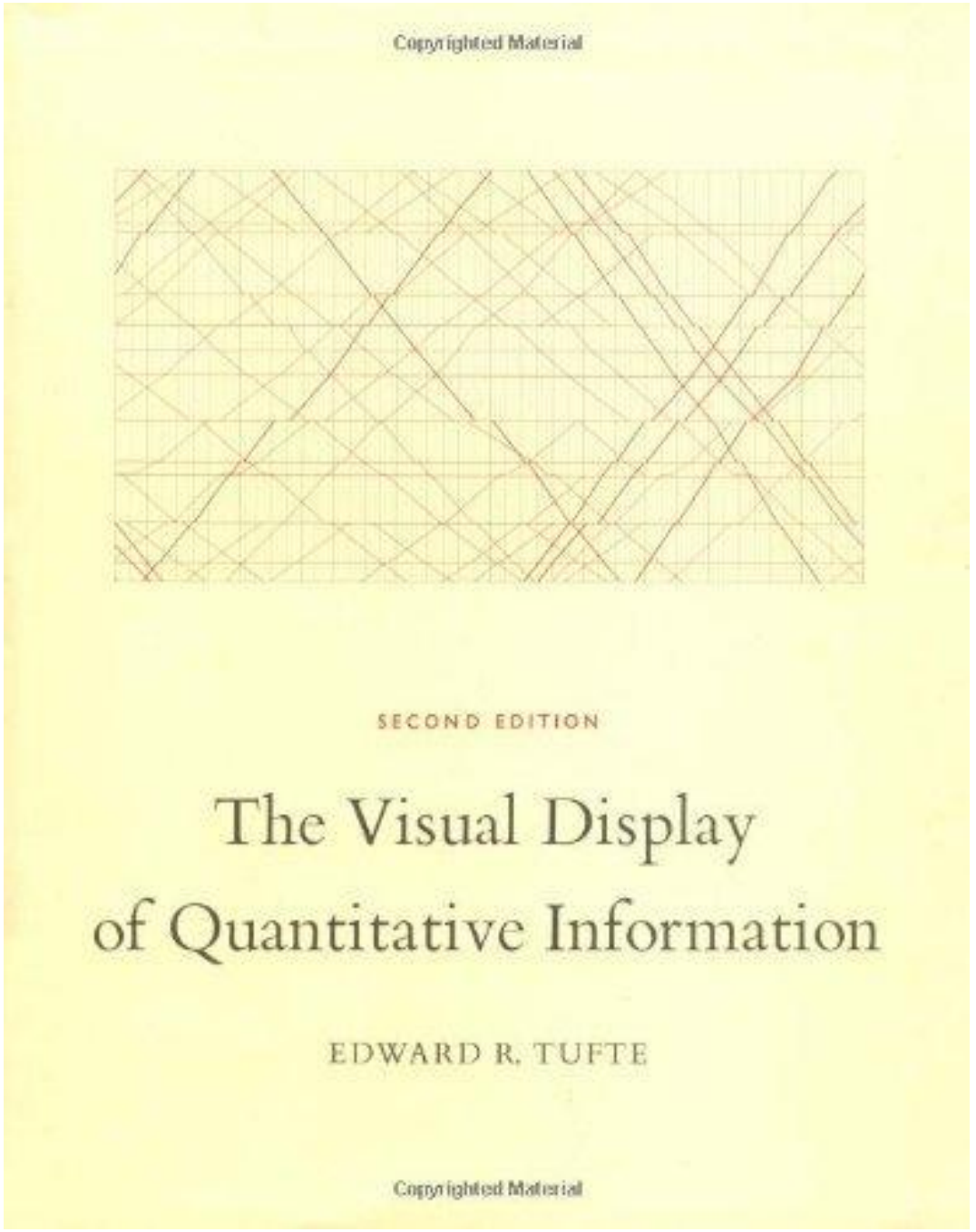
Session overview

- The thoughtful figure
  - Philosophy
    Key source: Tufte, E., & Graves-Morris, P. (2014). The visual display of quantitative information.; 1983
  - Base R graphics
- The grammar of graphics
  (aka How I learned to stop worrying and learned to love ggplots)
  - Grammar
    Key source: Wilkinson, L. (2006). *The grammar of graphics*. Springer Science & Business Media. (or the 2010 paper)
  - Ggplot2
    Key source: Elegent graphics for data analysis
- Q & A
  - Optional bonus slides on the grammar of graphics if you're gluttons for punishment

Graphics aren't just for Christmas

- Foundation of your data exploration/analysis
- Part of communicating with colleagues and yourself (record keeping) as you work on a project
- Should be drafted like any other part of the manuscript
  - Iteratively
  - Not an afterthought!

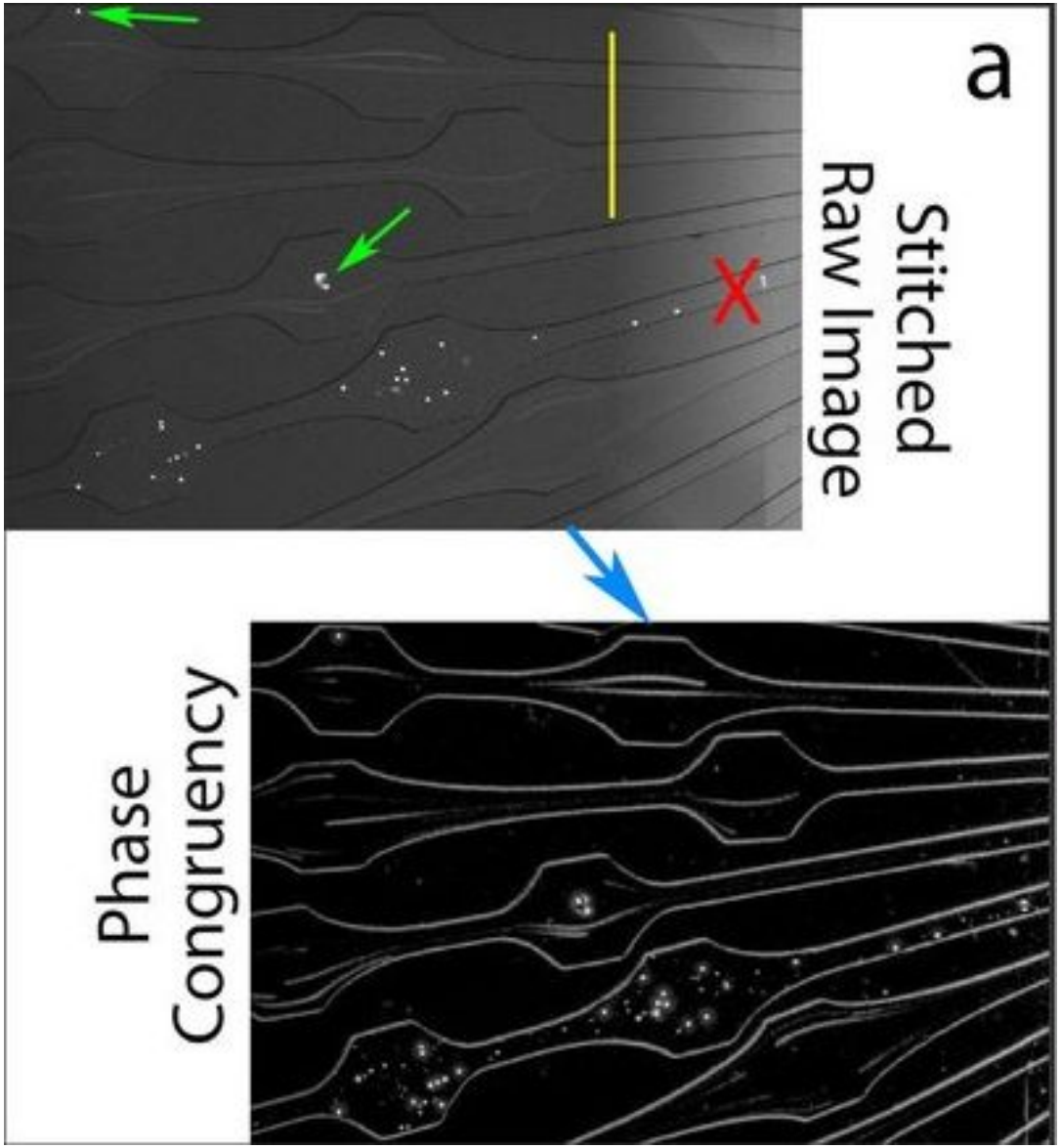
Tufte, E., & Graves-Morris, P. (2014). The visual display of quantitative information.; 1983



Principles of Graphical Excellence

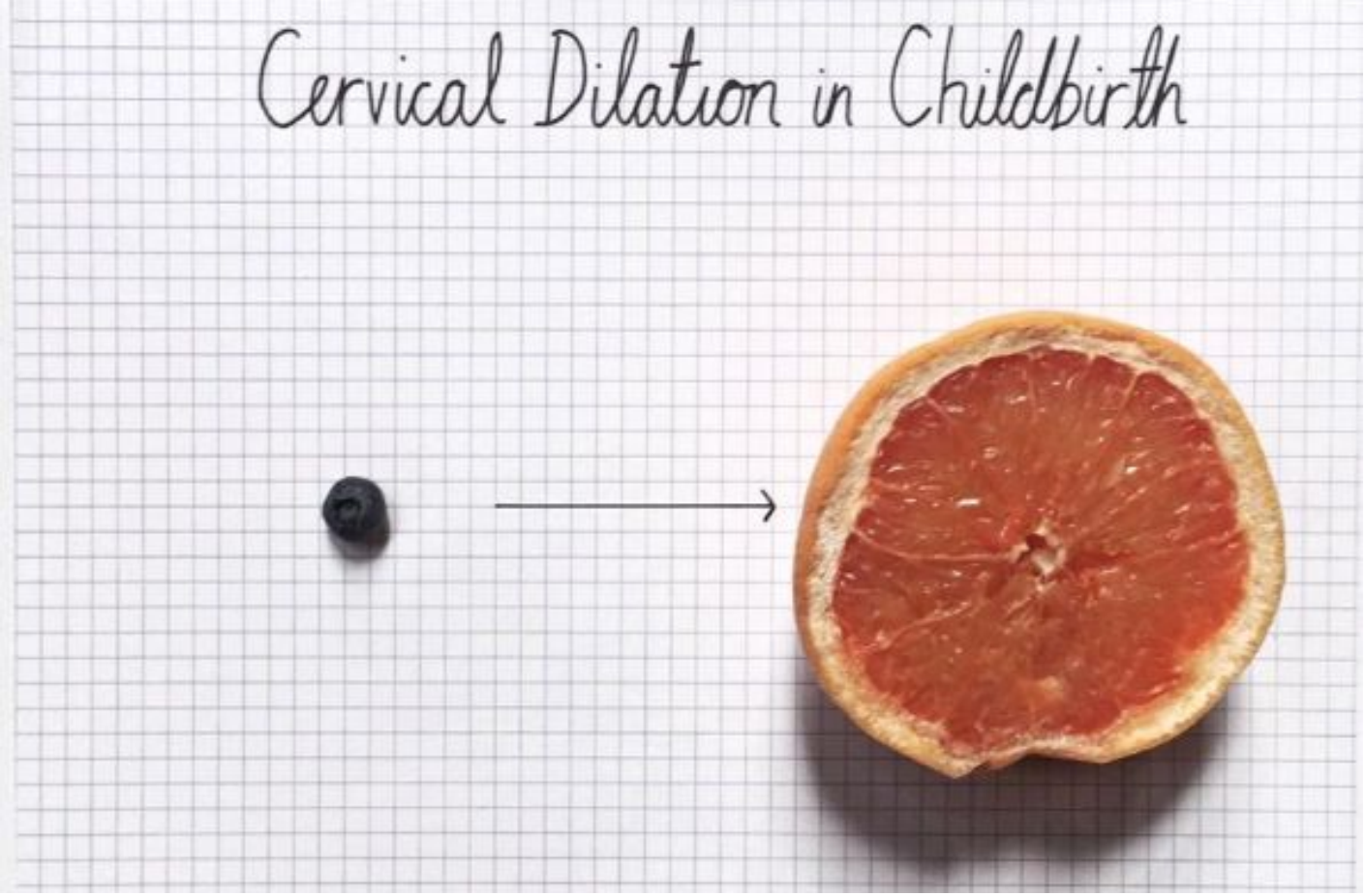
- Show the data
- Induce the viewer to think about the substance rather than about methodology, graphic design, the technology of the graphic production or something else
- Avoid distorting what the data have to say
- Present many numbers in a small space
- Make large datasets coherent
- Encourage the eye to compare different pieces of data
- Reveal the data at several levels of detail, from a broad overview to a fine structure
- Serve a reasonably clear purpose: description, exploration, tabulation or decoration
- Be closely integrated with the statistical verbal descriptions of a dataset

Show the data



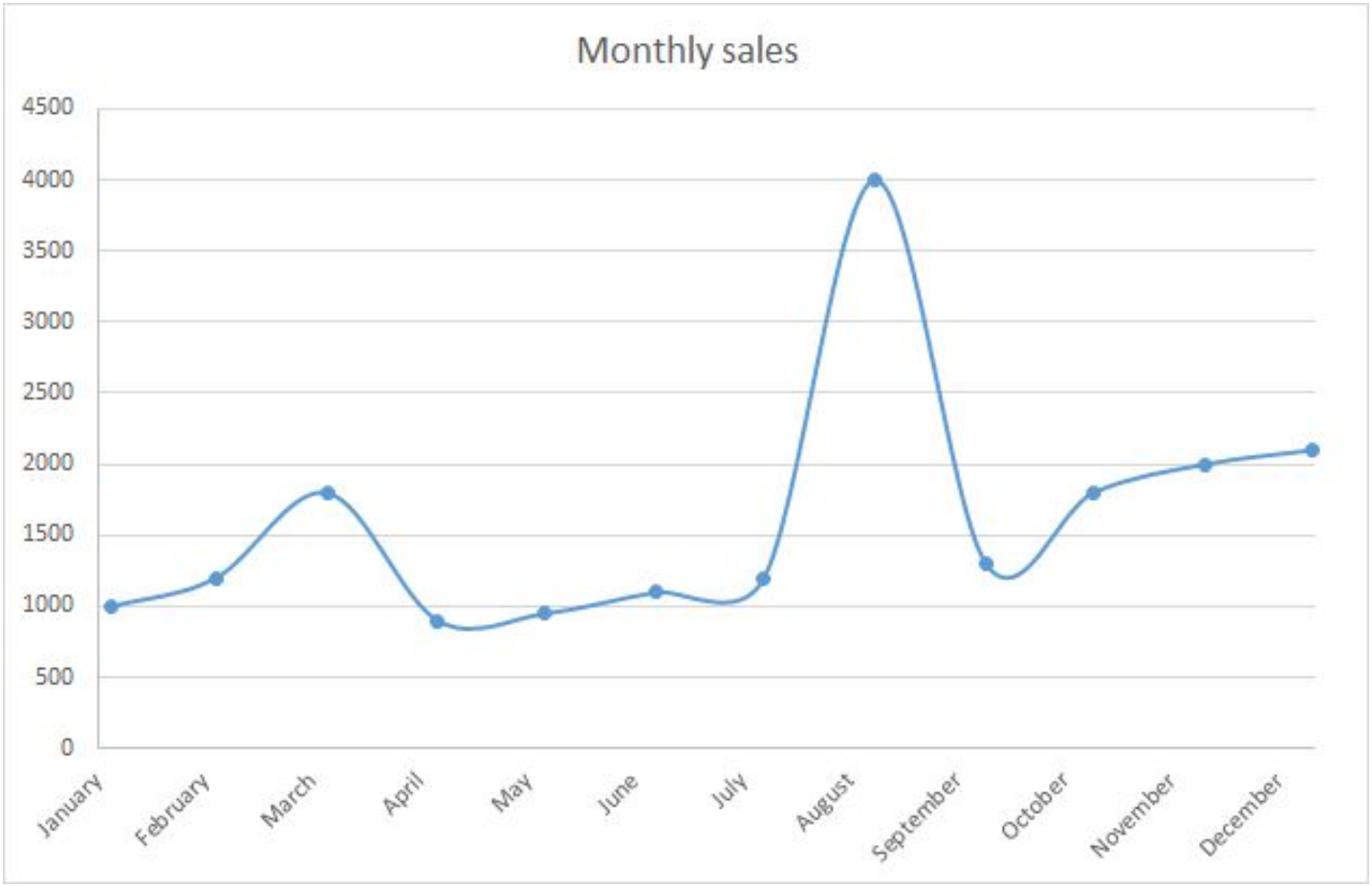
Example from: Benson, B. L., Li, L., Myers, J. T., Dorand, R. D., Gurkan, U. A., Huang, A. Y., & Ransohoff, R. M. (2018). Biomimetic post-capillary venule expansions for leukocyte adhesion studies. *Scientific Reports*, *8*(1), 9328.

Induce the viewer to think about the substance



Example from: Mona Chalabi from
http://sites.uci.edu/fmd2017fa/files/2017/05/Illustrator-2.pdf

Avoid distorting what the data have to say

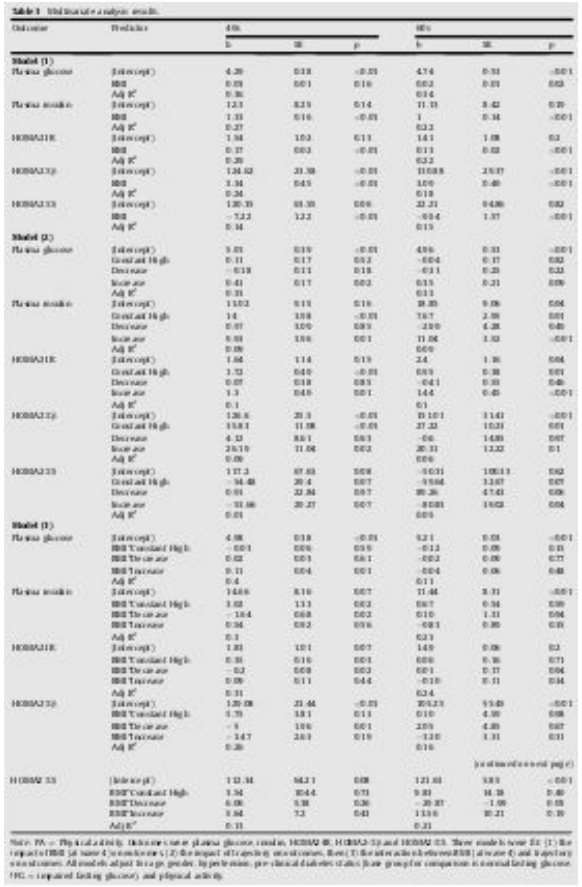


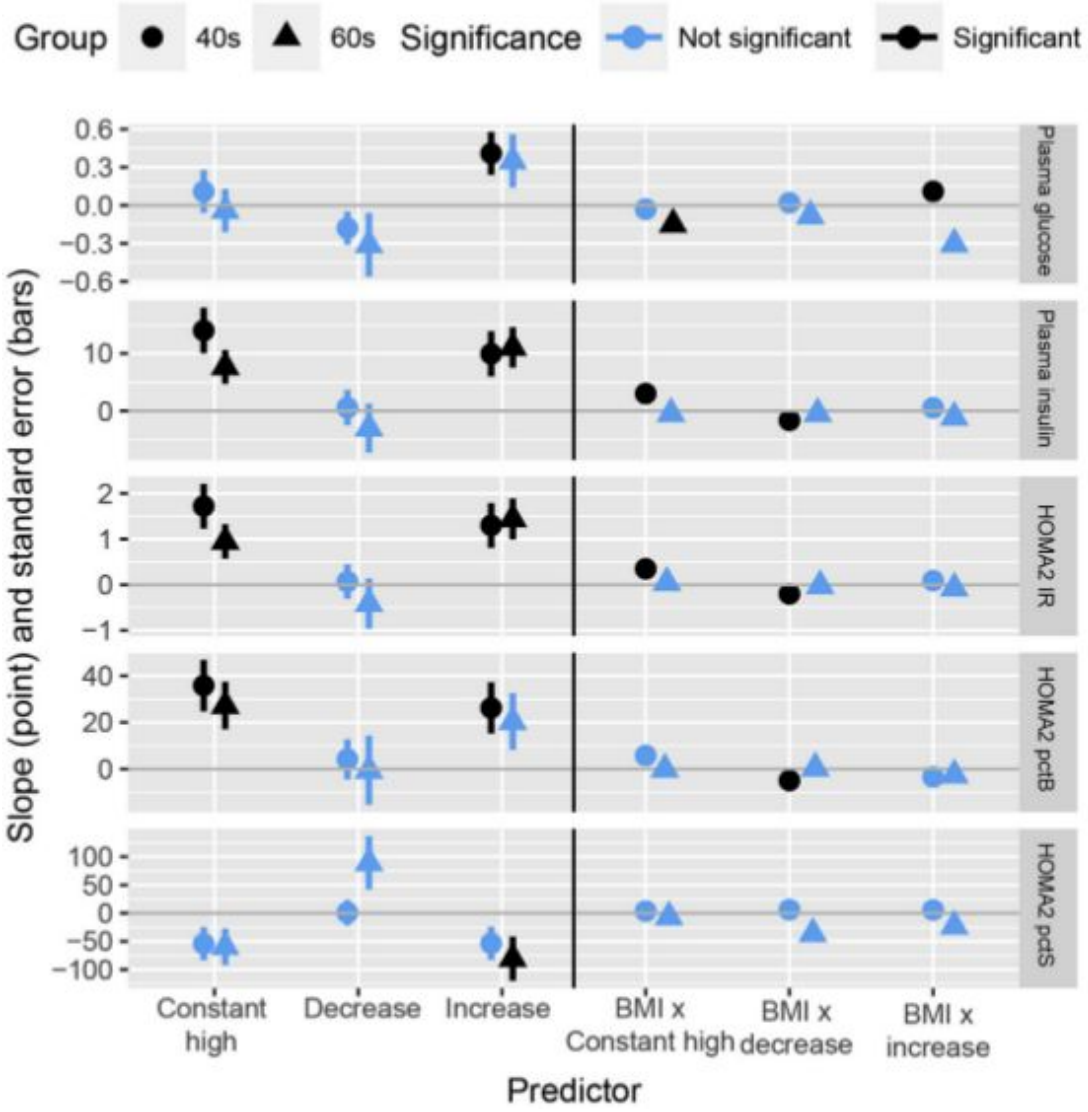
Example from: https://m.signalvnoise.com/lets-chart-stop-those-lying-line-charts/

Avoid distorting what the data have to say

Example from: https://m.signalvnoise.com/lets-chart-stop-those-lying-line-charts/

Present many numbers in a small space



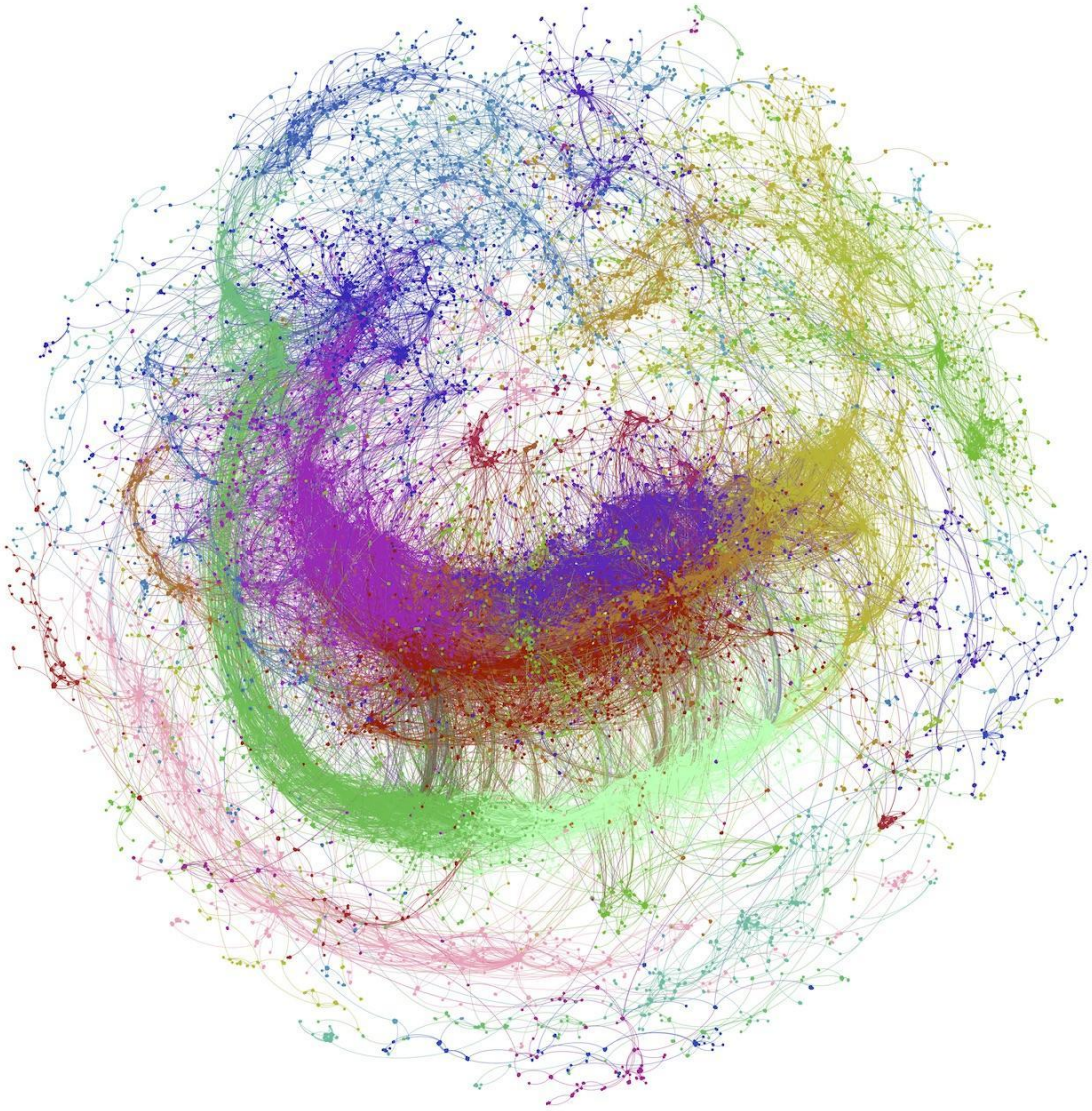


Example from: Walsh, E. I., Shaw, J., & Cherbuin, N. (2018). Trajectories of BMI change impact glucose and insulin metabolism. *Nutrition, Metabolism and Cardiovascular Diseases*, *28*(3), 243-251.

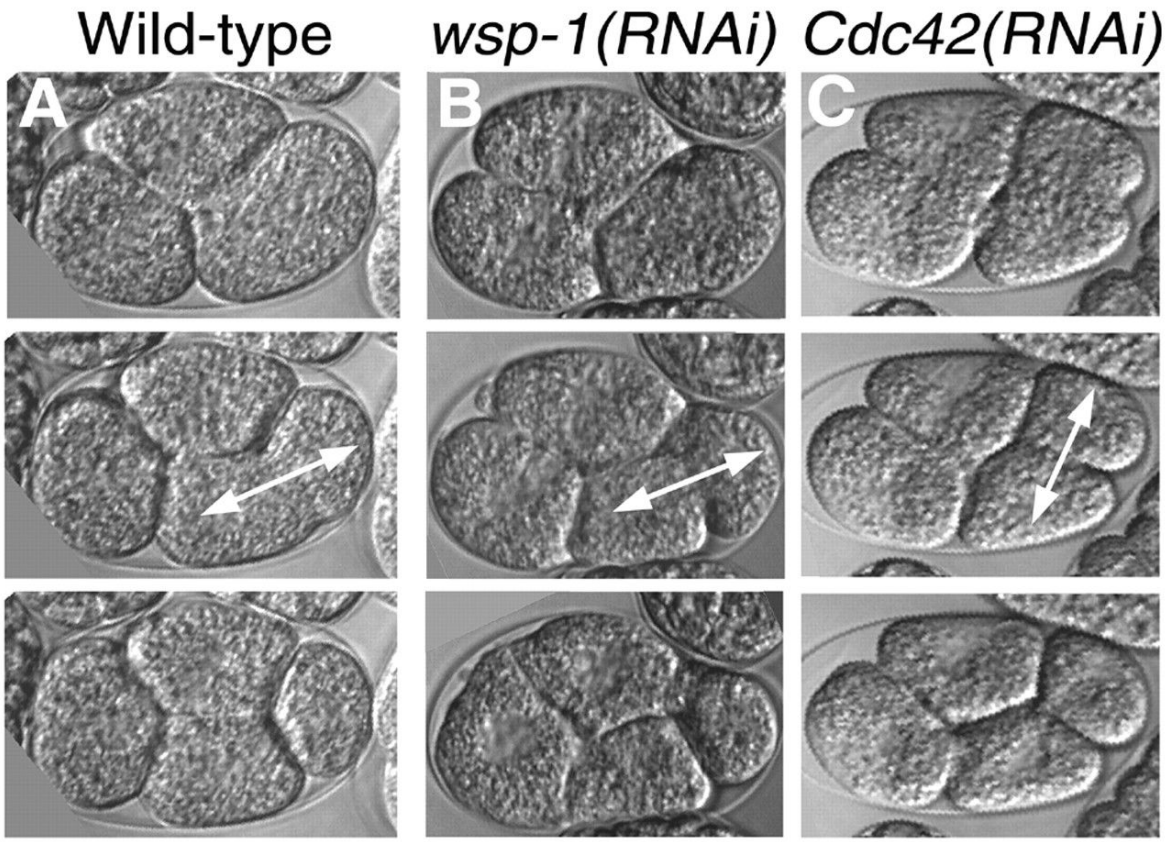
Make large datasets coherent

Example from: GBR coherent patterns from Higgs dataset https://lts2.epfl.ch/blog/gbr/2015/05/01/principal-patterns-on-graphs-discovering-coherent-structures-in-datasets/

Benzi, K., Ricaud, B., & Vandergheynst, P. (2016). Principal Patterns on Graphs: Discovering Coherent Structures in Datasets. *IEEE Trans. Signal and Information Processing over Networks*, 2(2), 160-173.

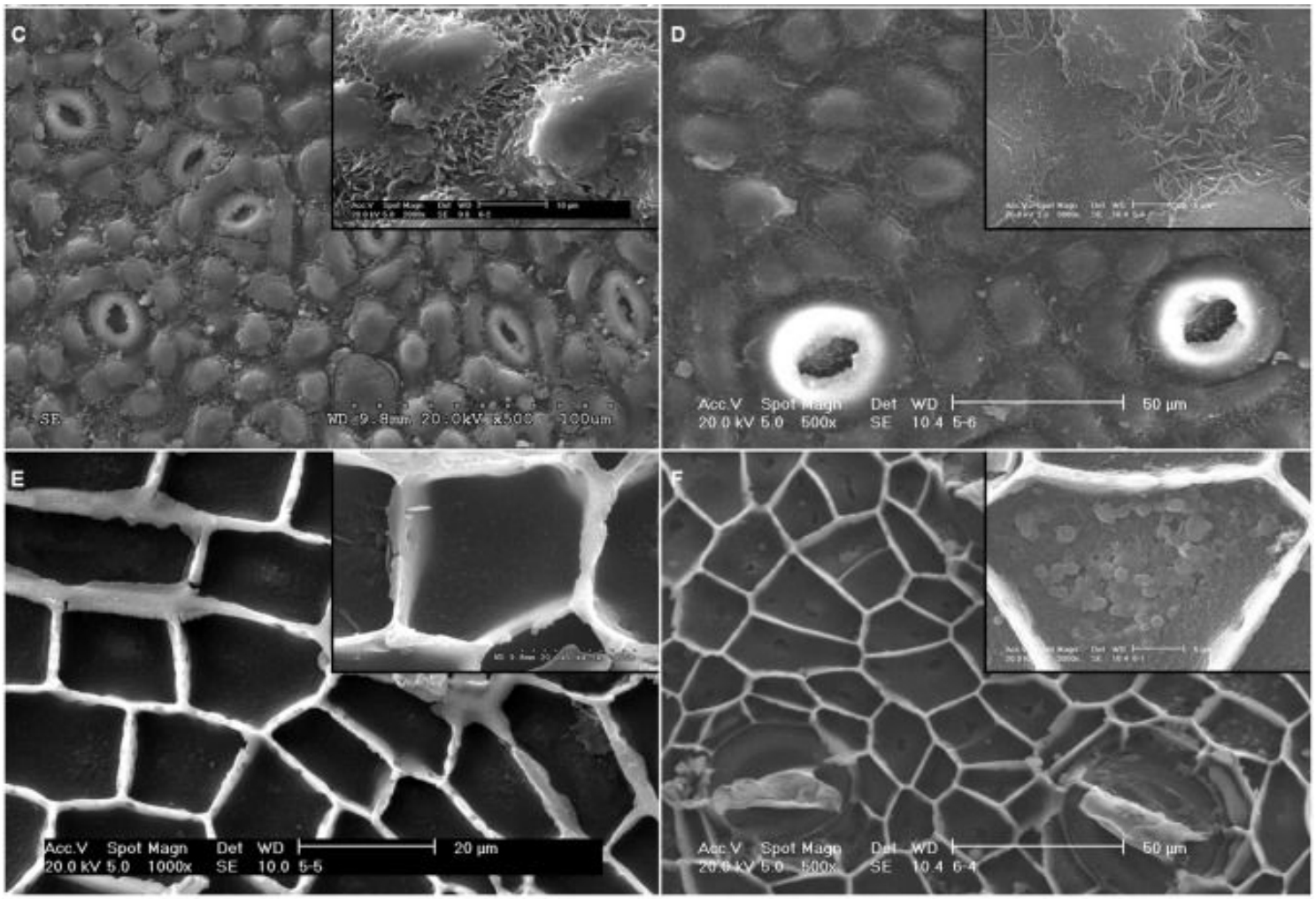


Encourage the eye to compare different pieces of data



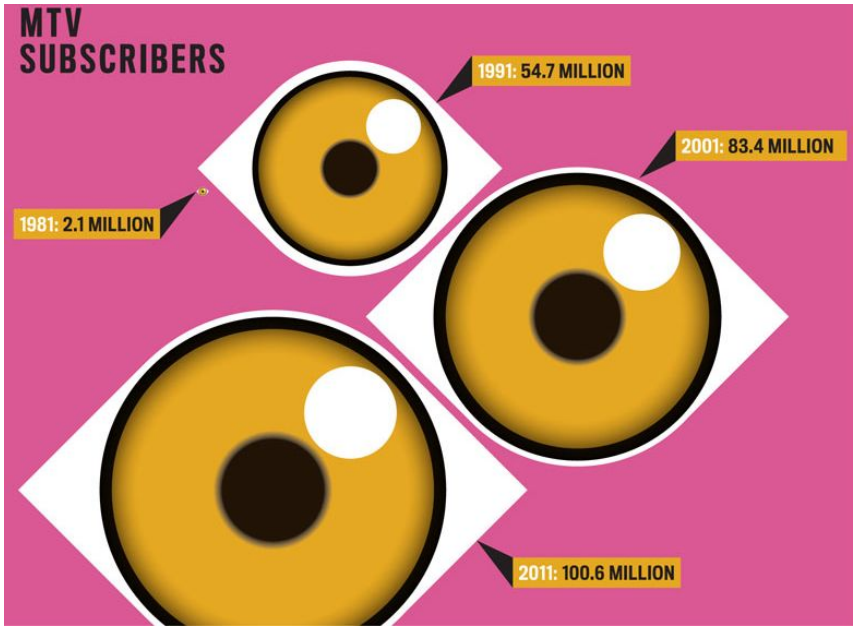
Example from: Sawa, M., Suetsugu, S., Sugimoto, A., Miki, H., Yamamoto, M., & Takenawa, T. (2003). Essential role of the C. elegans Arp2/3 complex in cell migration during ventral enclosure. *Journal of cell science*, *116*(8), 1505-1518. http://jcs.biologists.org/content/116/8/1505

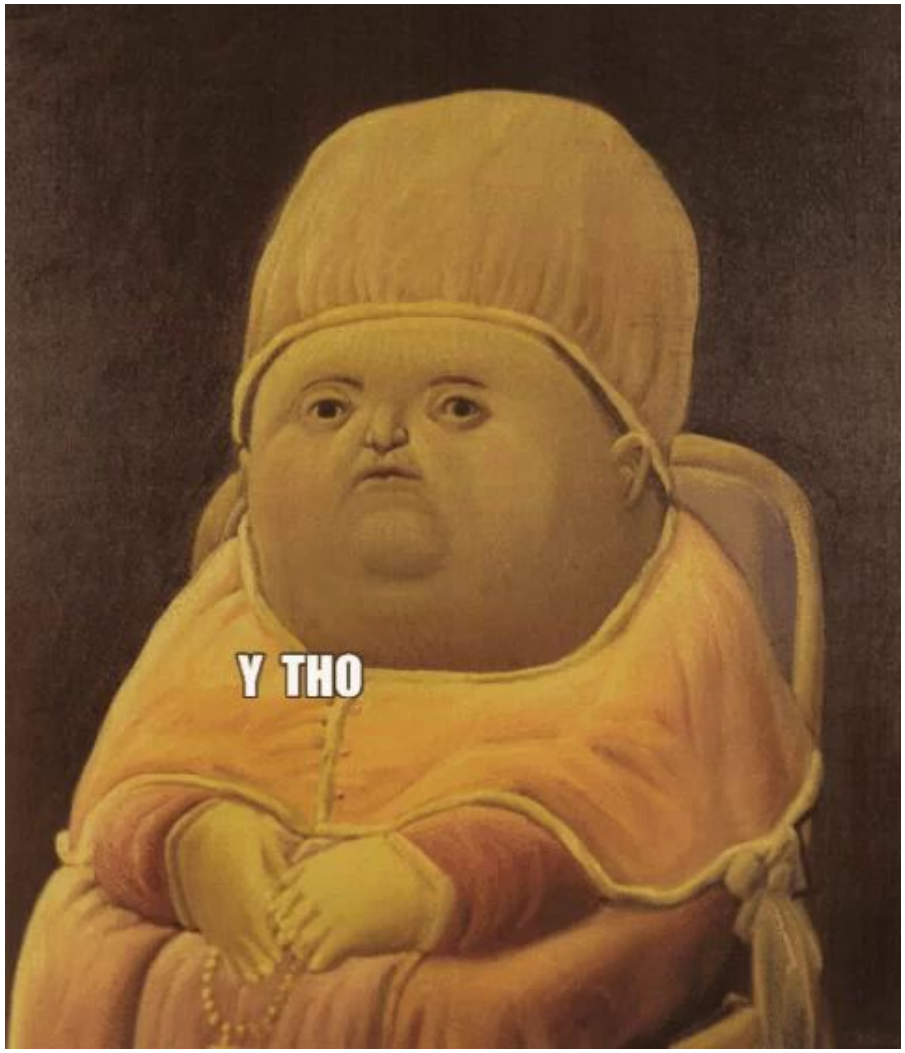
Reveal the data at several levels of detail



Example from: Guzmán, P., Fernández, V., Graça, J., Cabral, V., Kayali, N., Khayet, M., & Gil, L. (2014). Chemical and structural analysis of Eucalyptus globulus and E. camaldulensis leaf cuticles: a lipidized cell wall region. *Frontiers in plant science*, *5*, 481.

Serve a reasonably clear purpose



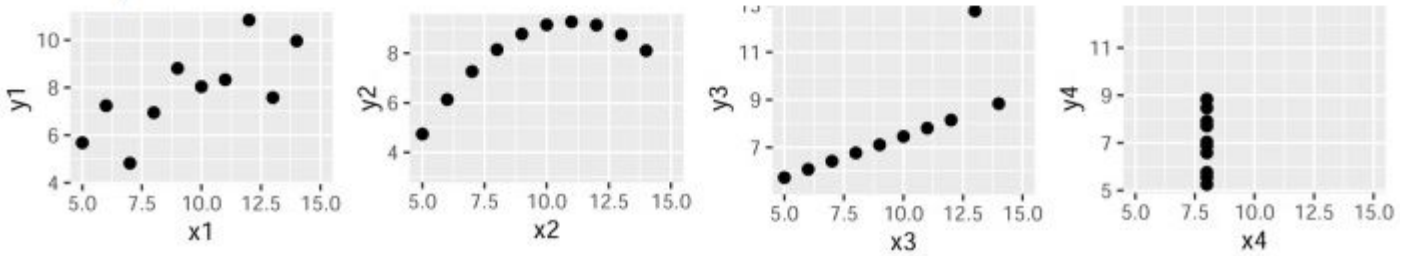


Be closely integrated with the statistical verbal descriptions of a dataset

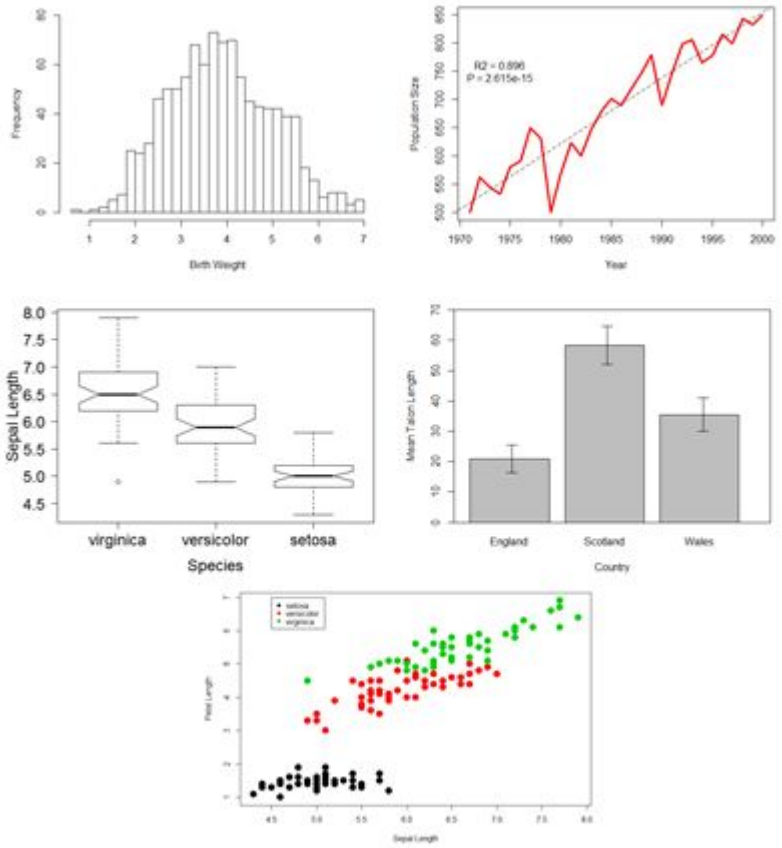
Viewed as pure numbers, it is difficult to see any difference between the sets:

x1	y1	x2	y2	x3	y3	x4	y4
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.1	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.1	4	5.39	19	12.5
12	10.84	12	9.13	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89

Striking differences become immediately obvious once they are displayed as scatterplots.



Base R Graphics



Pros:

- Quick and simple
- Most basic needs implemented
- Extremely convenient

Cons:

- Unusual plots difficult to create
- Not going to win any pageants
  - (but good if you get stuck on aesthetics when you should be focussing on the next step of analysis).

Recommended usage scenario: Part of initial data exploration and discovery.

Examples from "R Base Graphics: An Idiot's Guide" by Susan Johnson, http://rpubs.com/SusanEJohnston/7953

Exercise 01 time!

Single variable plots

- Histograms
- Dot plots
- Strip charts

Multiple variable plots

- Scatterplots
- Paired scatterplots
- Boxplots

Output of other functions or operations

- Density plots
- Linear model diagnostics
- Bar plots

Load the example data

```
data("iris")
```

```
iris$Species<-as.factor(iris$Species)
```

```
head(iris)
```

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Single variable plots

Histogram

```
hist(iris$Sepal.Length)
```

Dot chart

```
dotchart(iris$Sepal.Length)
```

Strip chart

```
stripchart(iris$Sepal.Length)
```

Multiple variable plots

Scatterplot

```
plot(iris$Petal.Length, iris$Petal.Width)
```

Adding lines to a scatterplot

```
plot.new()
lines(iris$Petal.Length, iris$Petal.Width)
plot(iris$Petal.Length, iris$Petal.Width)
lines(iris$Petal.Length, iris$Petal.Width)

plot(iris$Petal.Length, iris$Petal.Width, type="l")
```

Multiple variable plots

Pairs plot

```
pairs(~iris$Sepal.Length +
        iris$Petal.Width +
          iris$Sepal.Width)
```

Box plot

```
plot(iris$Species, iris$Sepal.Length, type="b")
plot(iris$Species, iris$Sepal.Length, type="b",
                                      notch=TRUE)
```

Output of other functions or operations

Density plot (cousin of the histogram)

```
sepal_density<-density(iris$Sepal.Length)
plot(sepal_density)
```

Linear model diagnostics

```
sepal_linear<-lm(iris$Sepal.Length ~iris$Petal.Width)
plot(sepal_linear)
```

Linear model output on points

```
plot(iris$Sepal.Length ~iris$Petal.Width)
abline(sepal_linear)
```

Output of other functions or operations

Bar plots

```
sepal_table<-table(iris$Sepal.Length)
barplot(sepal_table)
barplot(sepal_table,horiz=TRUE)
```

Stacked bar plots

```
sepal_table_stacked<-table(iris$Species,
                           iris$Sepal.Length)
barplot(sepal_table_stacked)
barplot(sepal_table_stacked,
                legend = names(table(iris$Species)))
```

Pie chart

```
petal_width_bins<-rep("Under 1", length(iris$Petal.Width))
petal_width_bins[which(iris$Petal.Width>1)]<-"Over 1"
petal_width_bins[which(iris$Petal.Width>2)]<-"Over 2"
pie(table(petal_width_bins))
```

Multiple plots

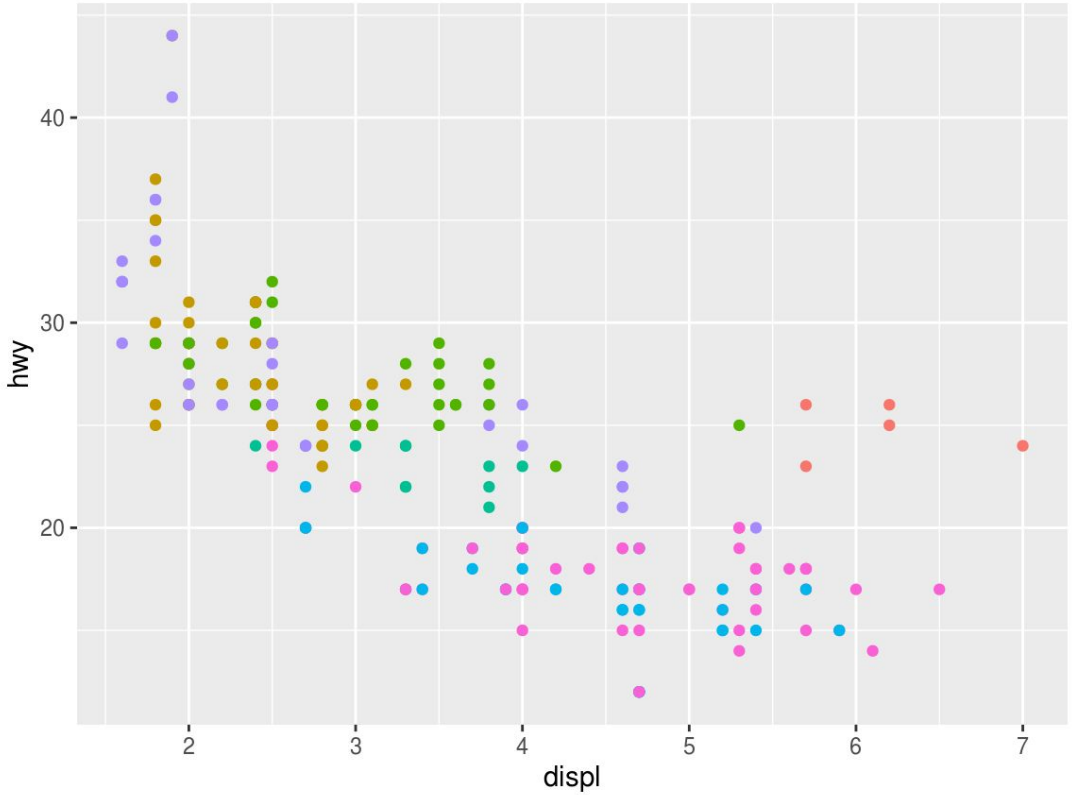
```
par(mfrow=c(1,1))
hist(iris$Sepal.Length)

par(mfrow=c(1,2))
hist(iris$Sepal.Length)
hist(iris$Sepal.Length)

par(mfrow=c(2,1))
hist(iris$Sepal.Length)
hist(iris$Sepal.Length)

par(mfrow=c(2,2))
hist(iris$Sepal.Length)
hist(iris$Sepal.Length)
hist(iris$Sepal.Length)
hist(iris$Sepal.Length)

hist(iris$Sepal.Length)
plot(sepal_density)
stripchart(iris$Sepal.Length)
barplot(sepal_table)

par(mfrow=c(1,1))
```

Annotations

```
plot(iris$Petal.Length, iris$Petal.Width)

text(1,2.5, labels="Obscured text")
text(1,2, labels="Text adjusted to 0", adj=0)
text(1,1.5, labels="Text adjusted to 0.5", adj=0.5)
text(1,1, labels="Text adjusted to 1", adj=1)

text(5,0.5, labels="Text with different X position")

text(4,2.5, labels="Obscured text")
text(4,2, labels="Text adjusted to 0", adj=0)
text(4,1.5, labels="Text adjusted to 0.5", adj=0.5)
text(4,1, labels="Text adjusted to 1", adj=1)
```


SCOTTY, I NEED
MORE POWER!

ggplot2



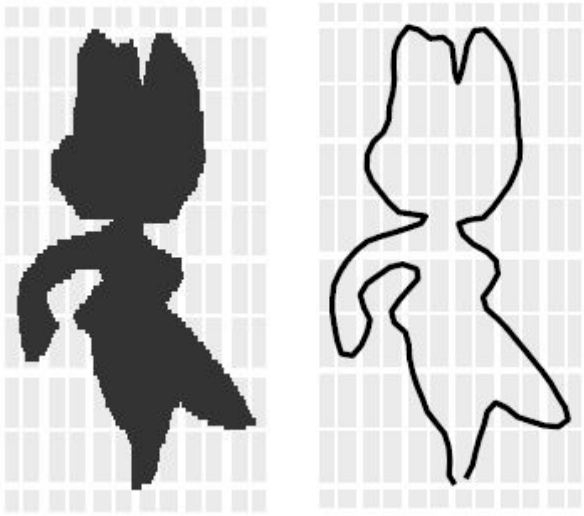
Pros:

- Beautiful (some say)
- Extensible
- Flexible

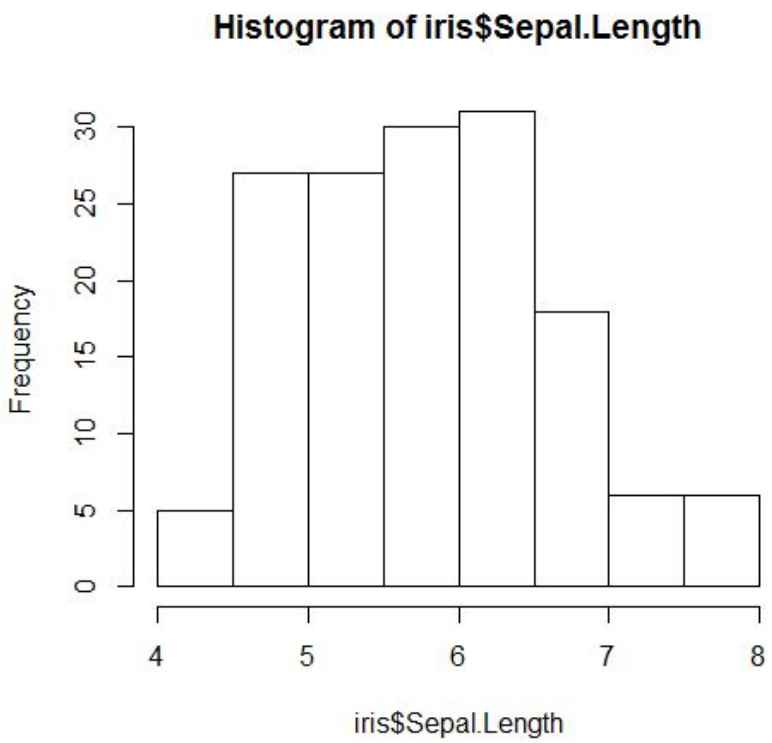
Cons:

- Less convenient
- Different logic can lead to unexpected behaviour

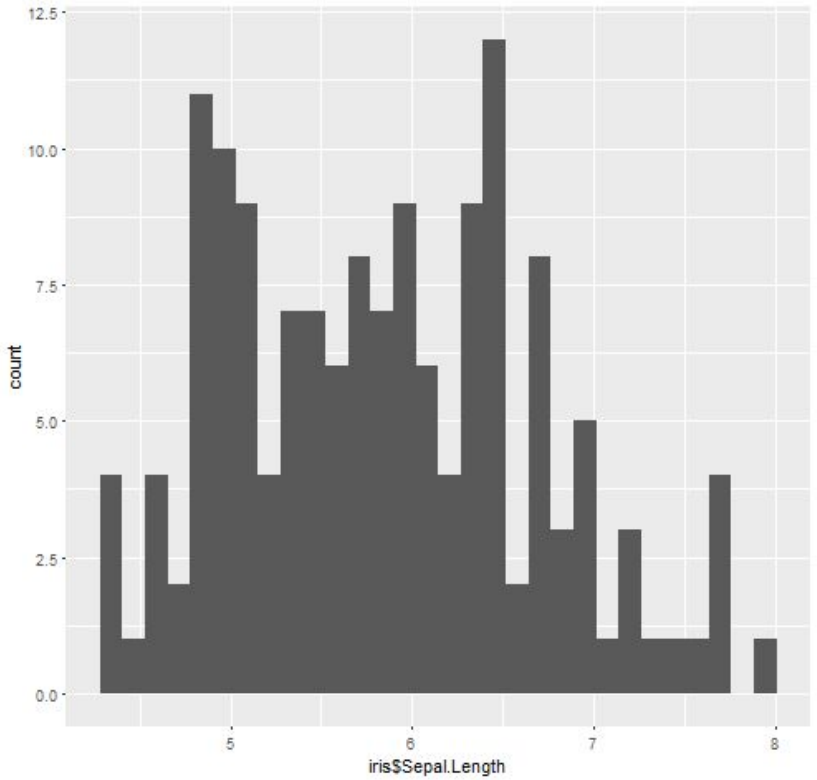
Recommended usage scenario:
Complex or final data visualisation.
Also fun.



(can be) as easy to use as base r



hist(iris$Sepal.Length)



qplot(iris$Sepal.Length)

Exercise 02

Quick recreations of base R plots in ggplots

Single variable plots

- Histograms
- Dot plots
- Strip charts

Multiple variable plots

- Scatterplots
- Paired scatterplots
- Boxplots

Output of other functions or operations

- Density plots
- Linear model diagnostics
- Bar plots
- Stacked bar plots
- Pie charts

Multiple plots

Annotations

Single variable plots

Histogram

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

Dot chart

```
ggplot(data=iris, aes(x=Sepal.Length)) + geom_dotplot()
```

Strip chart

- No base ggplot version of a strip chart

Multiple variable plots

Scatterplot

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

Adding lines to a scatterplot

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

Pairs plot

```
library(GGally)
ggpairs(iris[,c("Sepal.Length","Petal.Width","Sepal.Width")])
```

Box plot

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

Output of other functions or operations

Density plot (cousin of the histogram)

```
ggplot(data=iris, aes(x=Sepal.Length)) + geom_density()
```

Linear model diagnostics

- No base ggplot version of linear model diagnostics

Linear model output on points

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

Bar plots

```
ggplot(data=iris, aes(x=Sepal.Length)) + geom_bar()
```

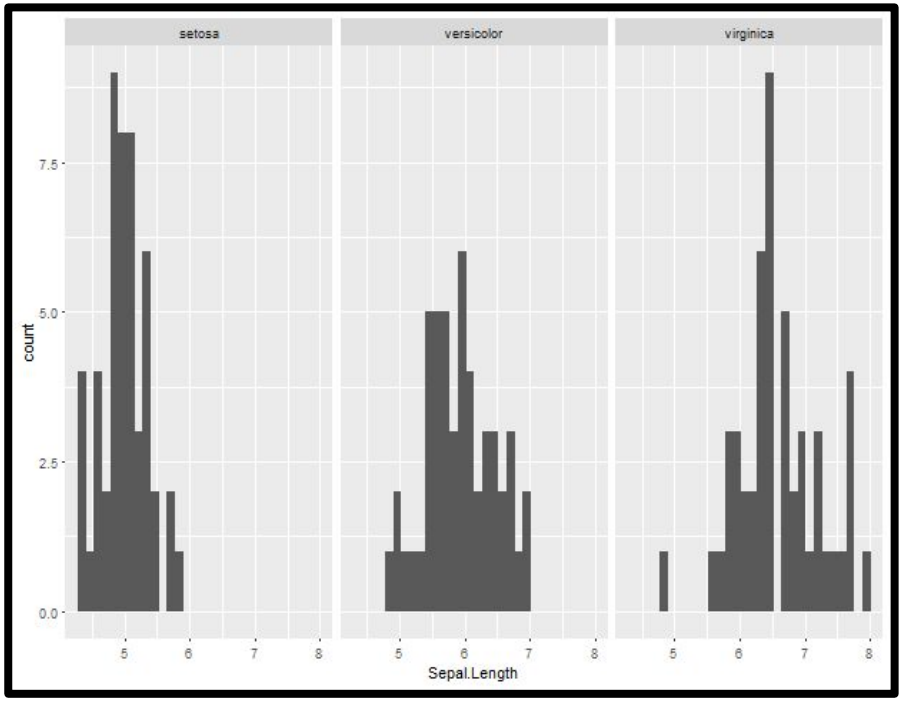
Stacked bar plots

```
ggplot(data=iris, aes(x=Sepal.Length)) +
                                  geom_bar(aes(fill=Species))
```

Pie chart

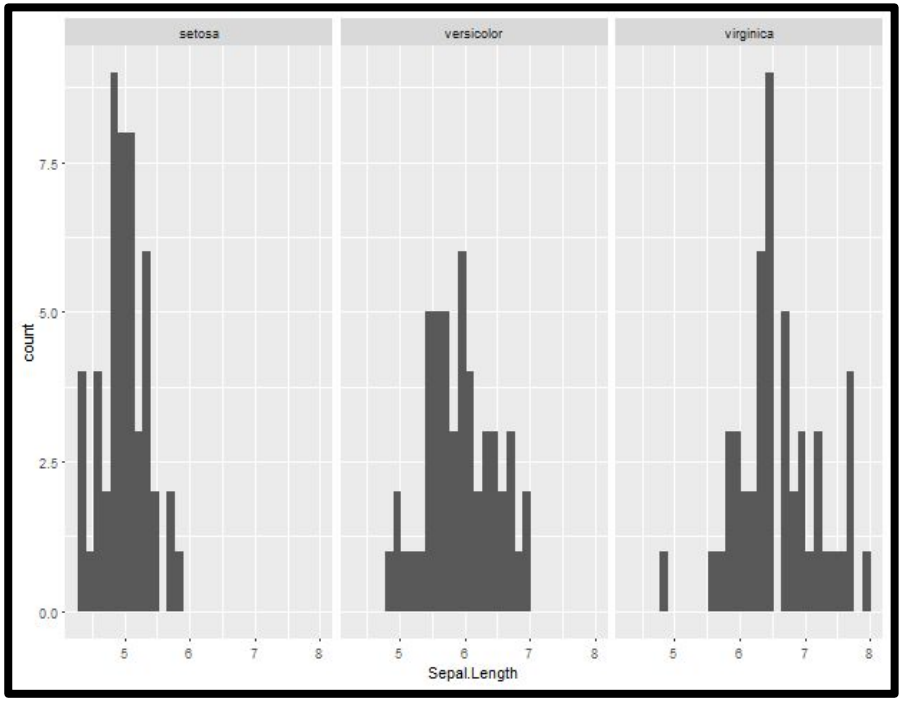
```
ggplot(iris, aes(x="", y=Petal.Width, fill=Species))+
               geom_bar(width = 1, stat = "identity") +
                              coord_polar("y", start=0)
```

Multiple plots

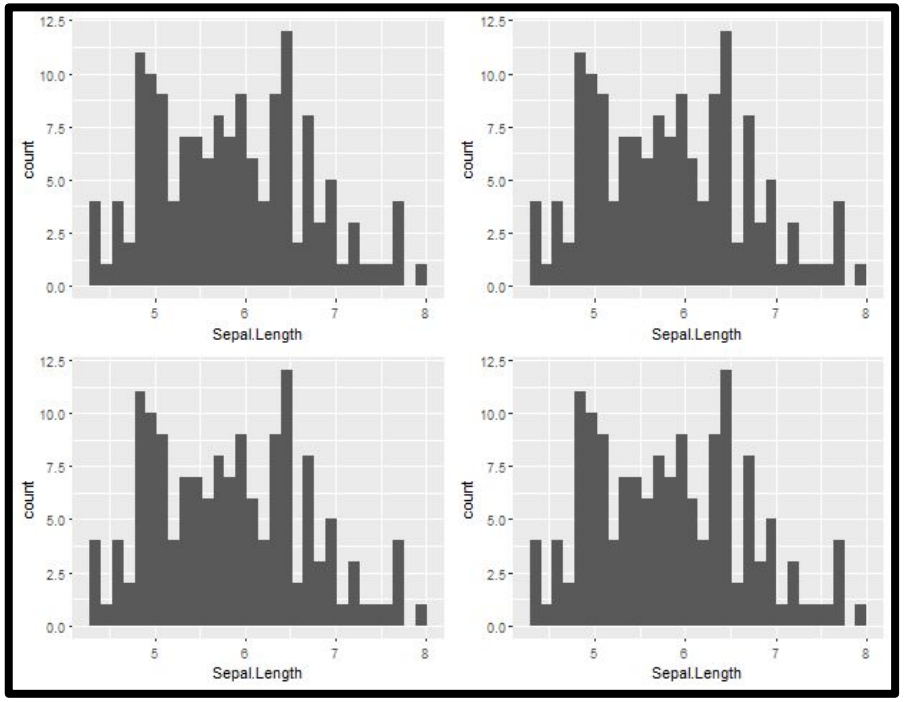



```
ggplot(data=iris, aes(x=Sepal.Length)) +
geom_histogram() + facet_wrap(~Species)
```

Multiple plots




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




```
library(gridExtra)
grid.arrange(plot_1, plot_2, plot_3, plot_4)
```

Multiple plots

Facets

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

Works across almost all plot types

```
ggplot(data=iris, aes(x=Petal.Length, y=Petal.Width)) +
                      geom_point() + facet_wrap(~Species)
```

Still works, even if it is redundant.

```
ggplot(data=iris, aes(x=Species, y=Sepal.Length)) +
                                 geom_boxplot()+ facet_wrap(~Species)
ggplot(data=iris, aes(x=Sepal.Length)) +
                     geom_bar(aes(fill=Species))+ facet_wrap(~Species)
```

Multiple plots

gridExtra

```
library(gridExtra)

   grid.arrange(ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram())
```

This gives extra flexibility, e.g. specifying rows and columns

```
   grid.arrange(nrow=1,
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram())

   grid.arrange(ncol=1,
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram())
```

Multiple plots

gridExtra

Nesting can be done conveniently via Grobs:

```
grid.arrange(ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
             arrangeGrob(nrow=1,
                 ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                 ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                 ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram()))
```

The major title is easily added

```
grid.arrange(top ="This is a title",
             ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
             arrangeGrob(nrow=1,
                         ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                         ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                         ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram())
             )
```

Both plots and grobs can be saved as **objects**

```
plot_object<-ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram()
grob_object<- arrangeGrob(nrow=1,
                          ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                          ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(),
                          ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram())

grid.arrange(plot_object,grob_object)
```

Multiple plots

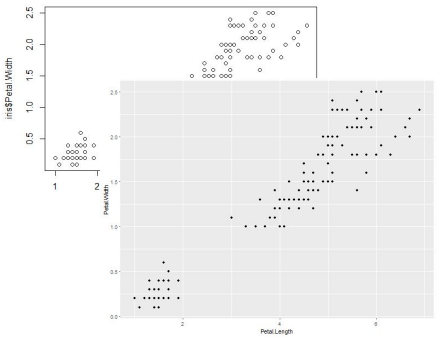
gridExtra

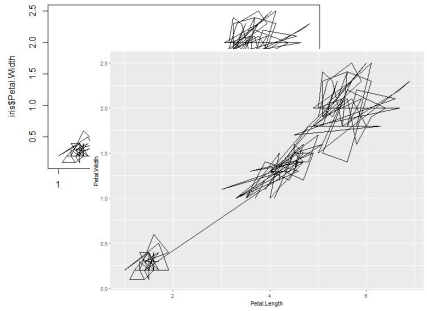
Used in combination with functions and loops,
this is EXTREMELY convenient for iterative plotting.

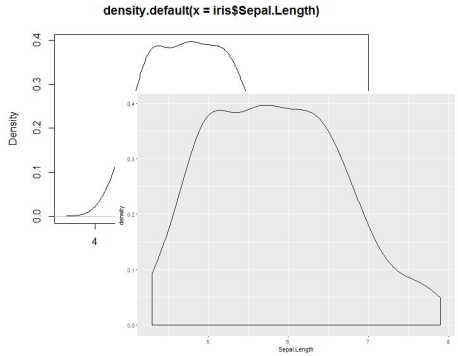
```
iterative_plot<-list()

for(i in 1:10){iterative_plot[[i]]<-ggplot(data=iris[sample(nrow(iris), 15), ],
                          aes(x=Petal.Length, y=Petal.Width)) + geom_point()}

do.call(grid.arrange, iterative_plot)


do.call(grid.arrange, iterative_plot[1:4])


do.call(grid.arrange, c(iterative_plot[1:4], top="Title"))
```

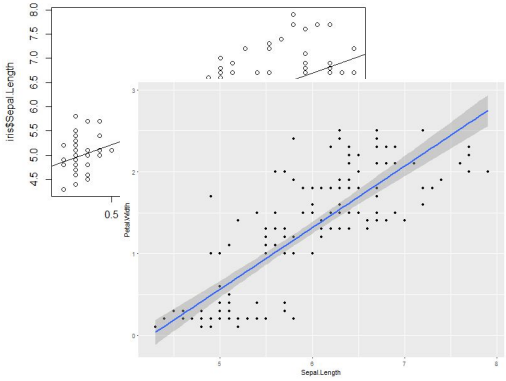
Annotations

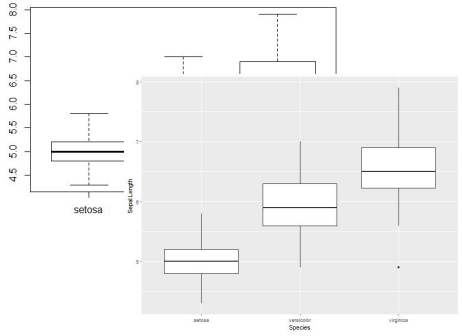
```
ggplot(data=iris, aes(x=Petal.Length, y=Petal.Width)) +
geom_point() + geom_text(aes(x=5, y=0.5,label="Text"))
```

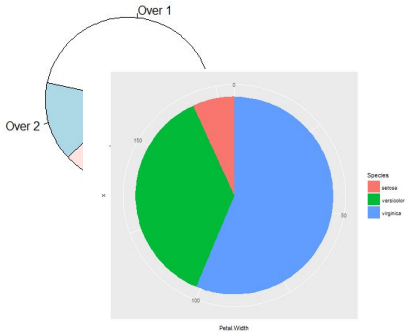

iris$Petal.Width
Petal.Length

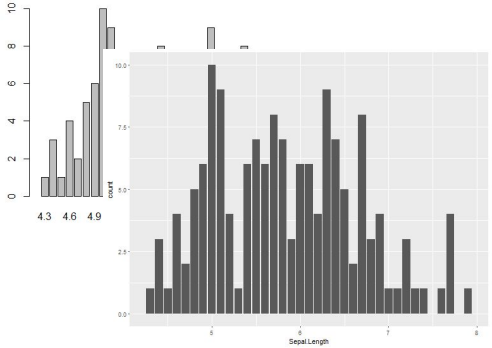

iris$Petal.Width
Petal.Length

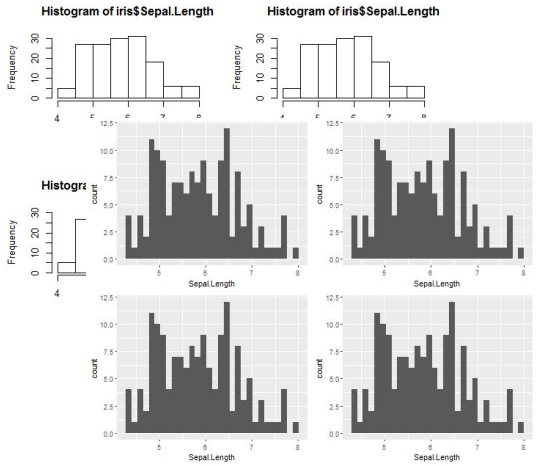

density.default(x = iris$Sepal.Length)
Density
Sepal.Length

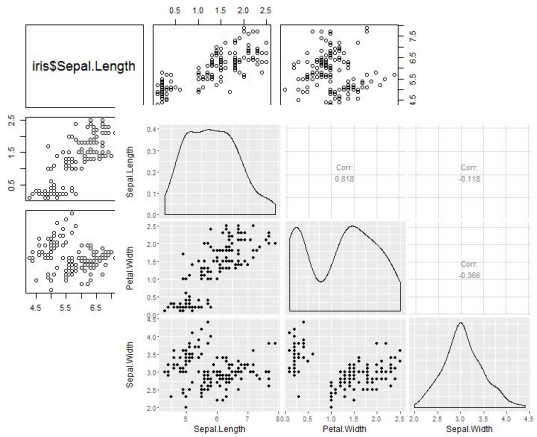

iris$Sepal.Length
Petal.Width
Sepal.Length

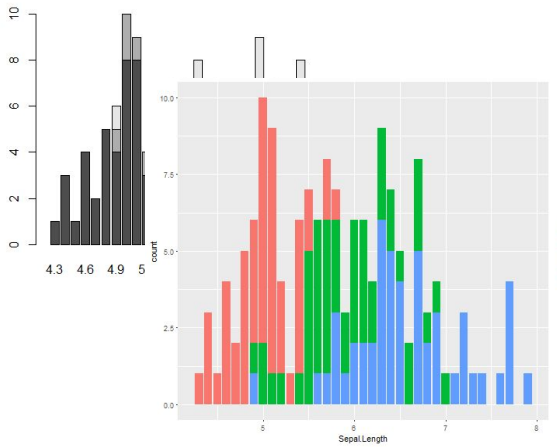

setosa
Sepal.Length
Species


Over 1
Over 2
Species
setosa
versicolor
virginica
Petal.Width


Sepal.Length


Histogram of iris$Sepal.Length
Histogram of iris$Sepal.Length
Frequency
count
Sepal.Length

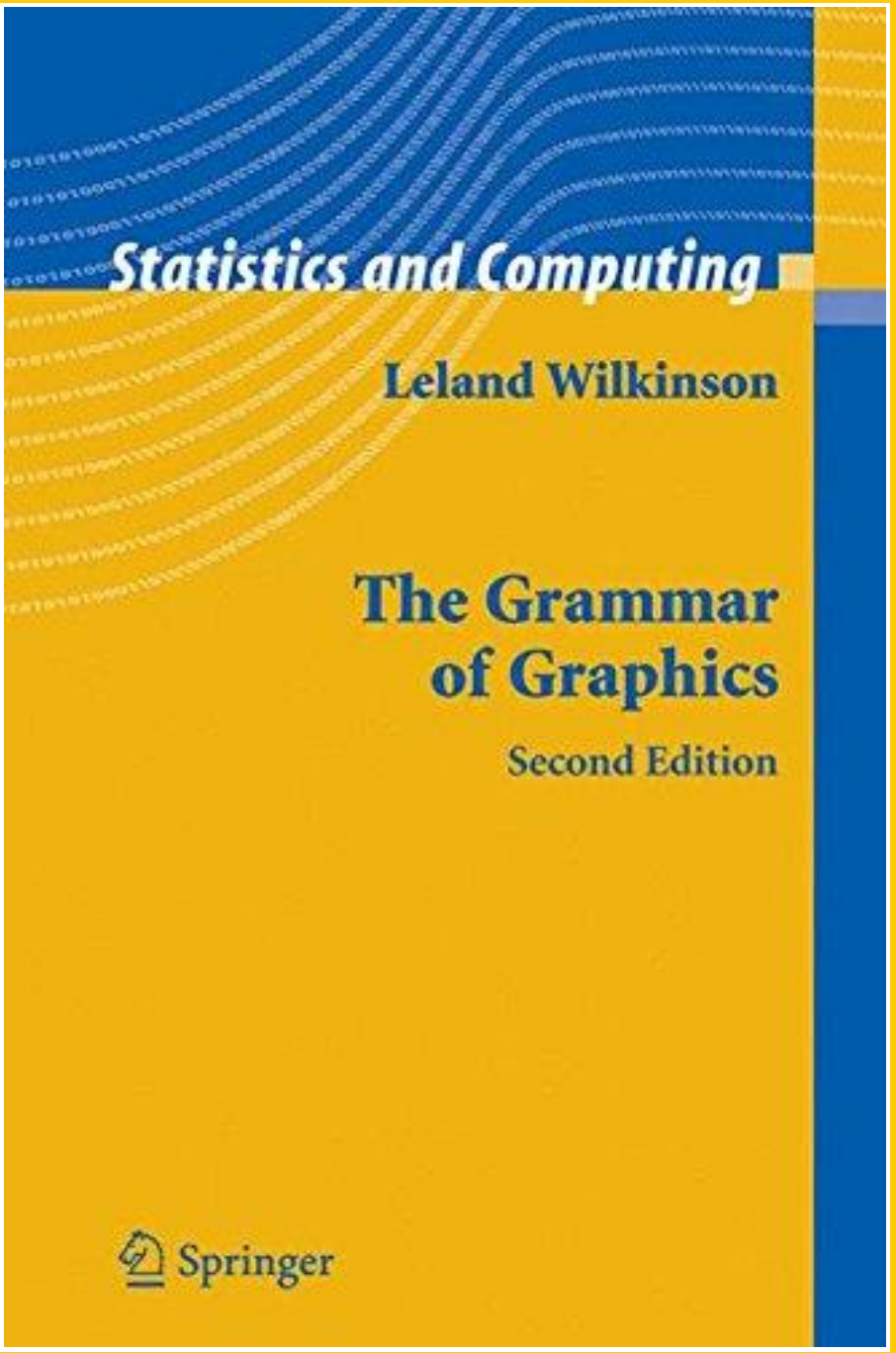

iris$Sepal.Length
Sepal.Length
Petal.Width
Sepal.Width
Corr: 0.818
Corr: -0.118
Corr: -0.366


count
Species
setosa
versicolor
virginica
Sepal.Length

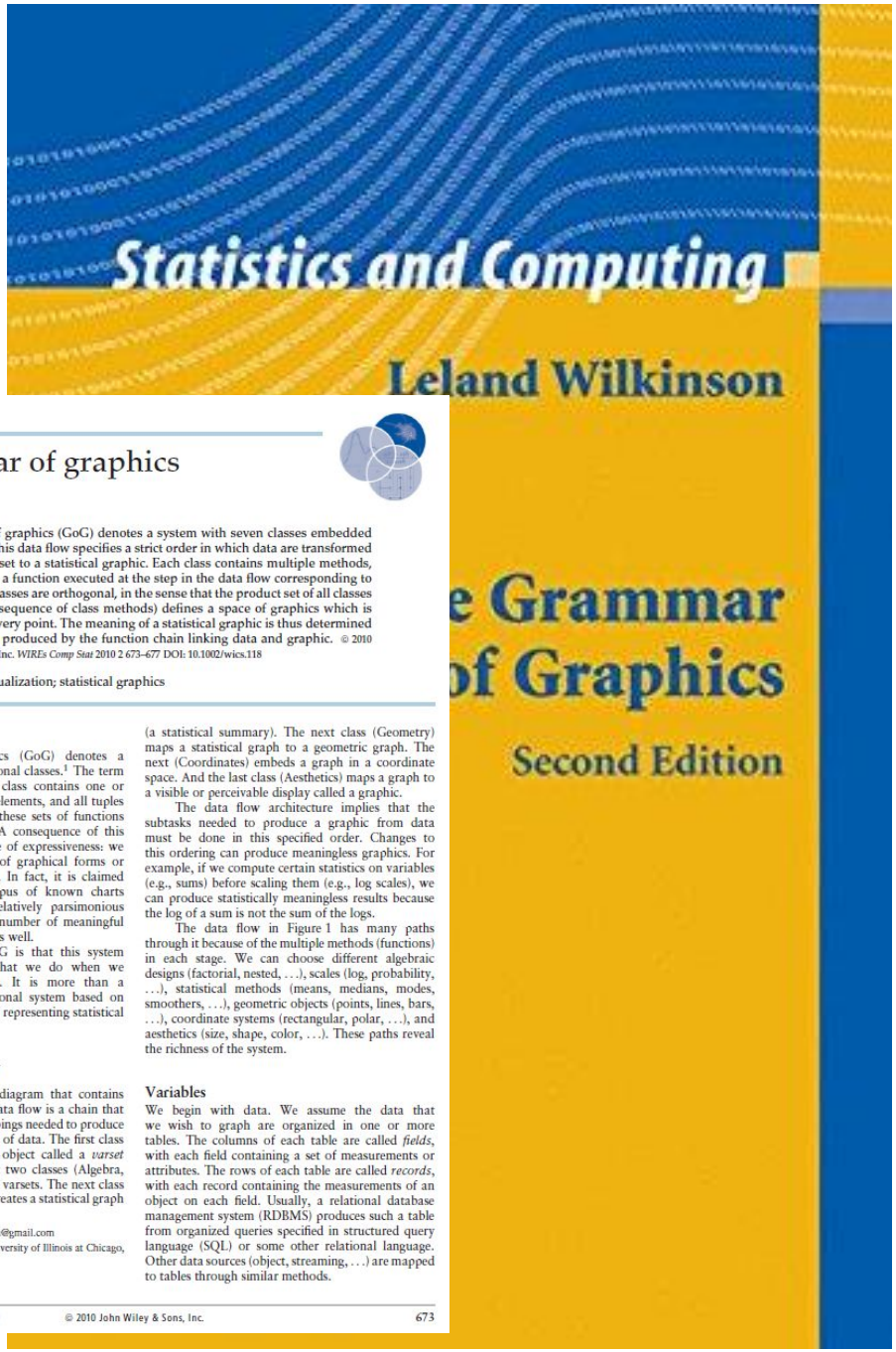

PLEASE [redacted] R
I WANT SOME MORE
quickmeme.com

Examples from https://www.r-graph-gallery.com/

Wilkinson, L. (2006). *The grammar of graphics*. Springer Science & Business Media.



Wilkinson, L. (2006). *The grammar of graphics*. Springer Science & Business Media.



Advanced Review

The grammar of graphics

Leland Wilkinson*

The grammar of graphics (GoG) denotes a system with seven classes embedded in a data flow. This data flow specifies a strict order in which data are transformed from a raw dataset to a statistical graphic. Each class contains multiple methods, each of which is a function executed at the step in the data flow corresponding to that class. The classes are orthogonal, in the sense that the product set of all classes (every possible sequence of class methods) defines a space of graphics which is meaningful at every point. The meaning of a statistical graphic is thus determined by the mapping produced by the function chain linking data and graphic. *WIREs Comp Stat* 2010 2 673–677 DOI: 10.1002/wics.118

Keywords: visualization; statistical graphics

INTRODUCTION

The grammar of graphics (GoG) denotes a system with seven orthogonal classes.[1] The term *orthogonal* means that each class contains one or more methods (functions) as elements, and all tuples in the seven-fold product of these sets of functions produce meaningful graphs. A consequence of this orthogonality is a high degree of expressiveness: we can produce a huge variety of graphical forms or chart types in such a system. In fact, it is claimed that virtually the entire corpus of known charts can be generated by this relatively parsimonious system, and perhaps a great number of meaningful but undiscovered chart types as well.

A second claim of GoG is that this system describes the *meaning* of what we do when we construct statistical graphics. It is more than a taxonomy. It is a computational system based on the underlying mathematics of representing statistical functions of data.

THE GoG DATA FLOW

Figure 1 shows a *data flow* diagram that contains the seven GoG classes. This data flow is a chain that describes the sequence of mappings needed to produce a statistical graphic from a set of data. The first class (Variables) maps data to an object called a *varset* (a set of variables). The next two classes (Algebra, Scales) are transformations on varsets. The next class (Statistics) takes a varset and creates a statistical graph (a statistical summary). The next class (Geometry) maps a statistical graph to a geometric graph. The next (Coordinates) embeds a graph in a coordinate space. And the last class (Aesthetics) maps a graph to a visible or perceivable display called a graphic.

The data flow architecture implies that the subtasks needed to produce a graphic from data must be done in this specified order. Changes to this ordering can produce meaningless graphics. For example, if we compute certain statistics on variables (e.g., sums) before scaling them (e.g., log scales), we can produce statistically meaningless results because the log of a sum is not the sum of the logs.

The data flow in Figure 1 has many paths through it because of the multiple methods (functions) in each stage. We can choose different algebraic designs (factorial, nested, ...), scales (log, probability, ...), statistical methods (means, medians, modes, smoothers, ...), geometric objects (points, lines, bars, ...), coordinate systems (rectangular, polar, ...), and aesthetics (size, shape, color, ...). These paths reveal the richness of the system.

Variables

We begin with data. We assume the data that we wish to graph are organized in one or more tables. The columns of each table are called *fields*, with each field containing a set of measurements or attributes. The rows of each table are called *records*, with each record containing the measurements of an object on each field. Usually, a relational database management system (RDBMS) produces such a table from organized queries specified in structured query language (SQL) or some other relational language. Other data sources (object, streaming, ...) are mapped to tables through similar methods.

*Correspondence to: Leland.wilkinson@gmail.com
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
DOI: 10.1002/wics.118

DOI: 10.1002/wics.118

Sequential steps in visualisation…

Data goes in

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Variables

Algebra

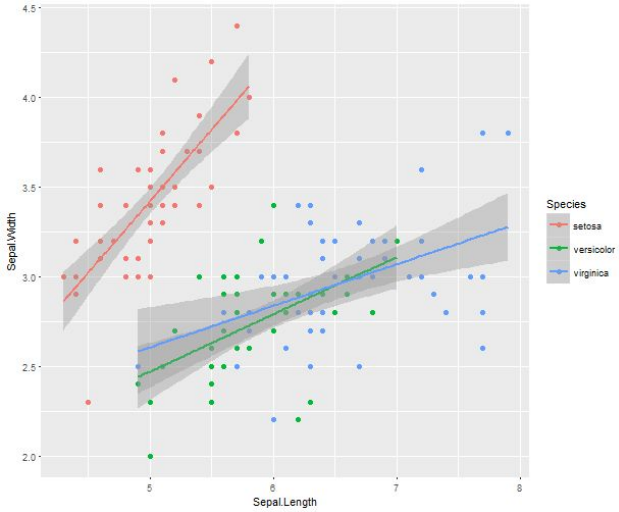
Scales

Statistics

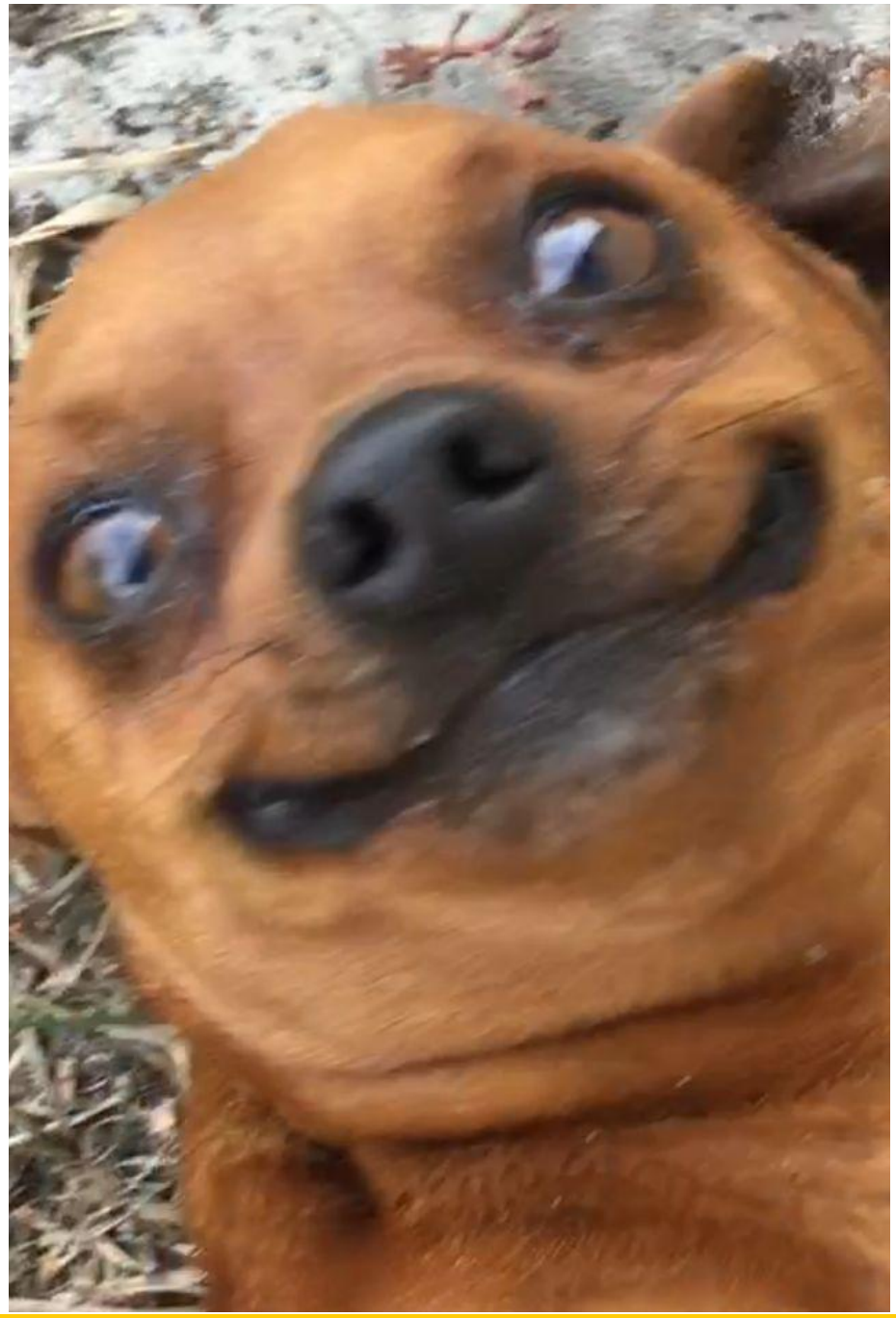
Geometry

Coordinates

Aesthetics



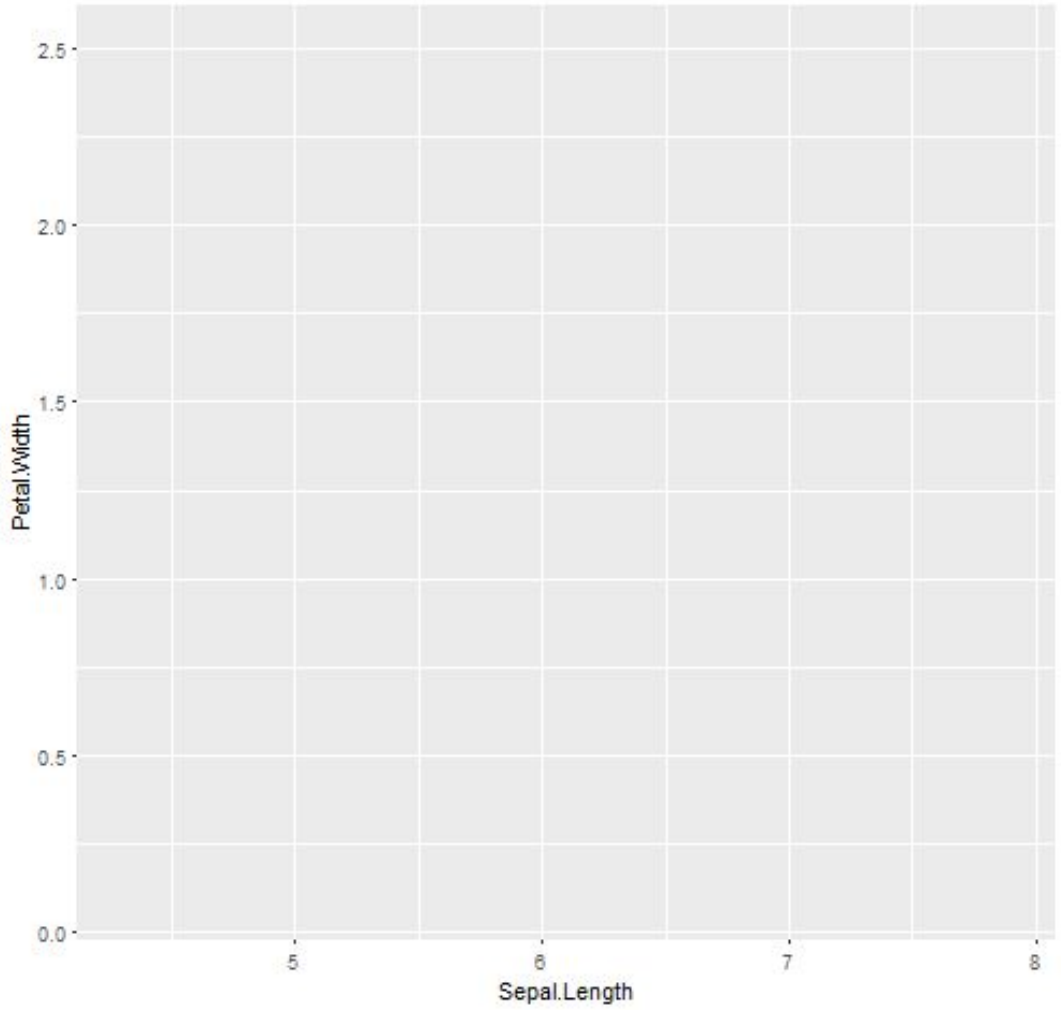
Graphics come out



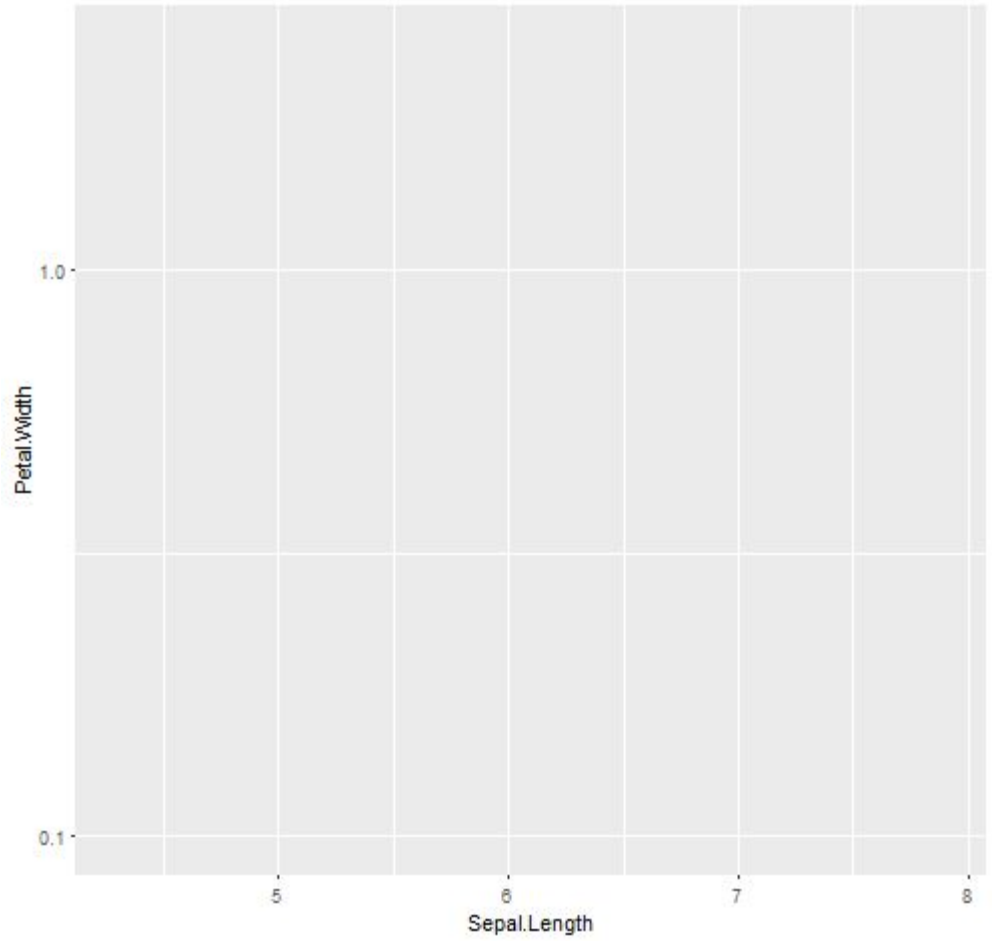
Exercise 03

Follow along, with the **iris** dataset,
looking at **Sepal.Length** and **Petal.Width**

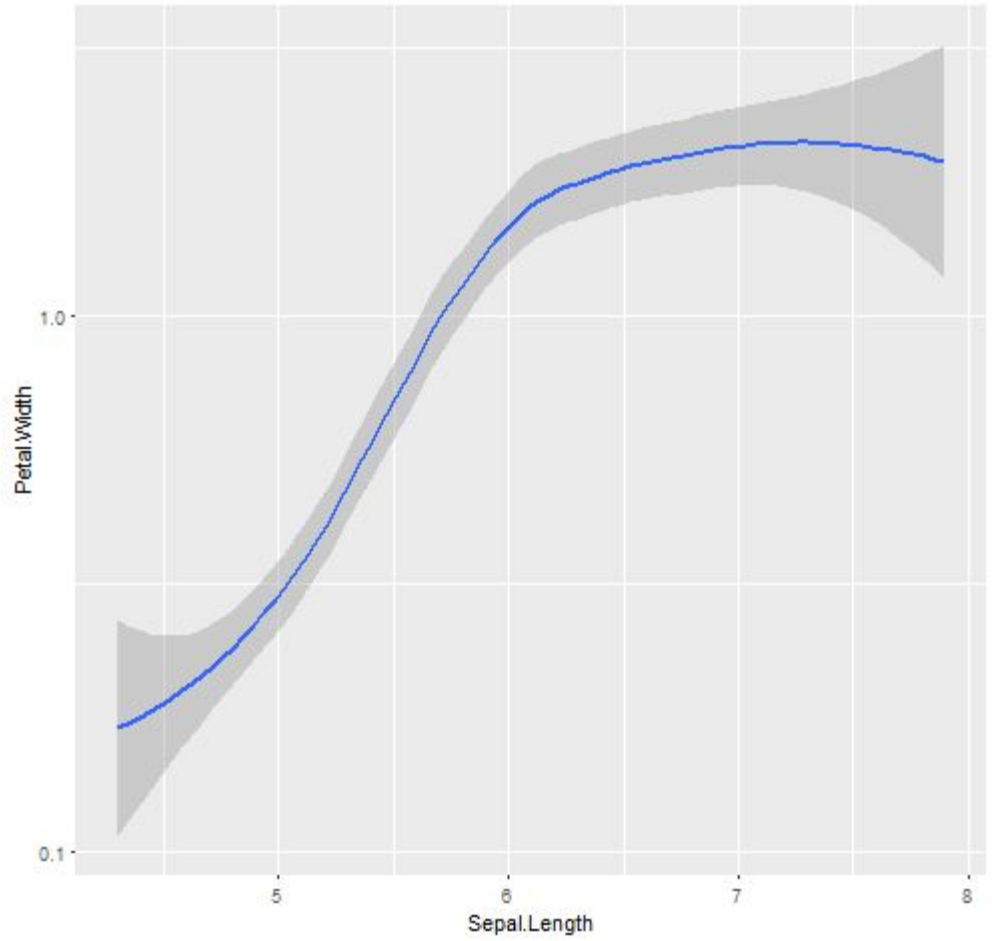

```
ggplot(data=dataset, aes(x=var, y=var2, …))
```



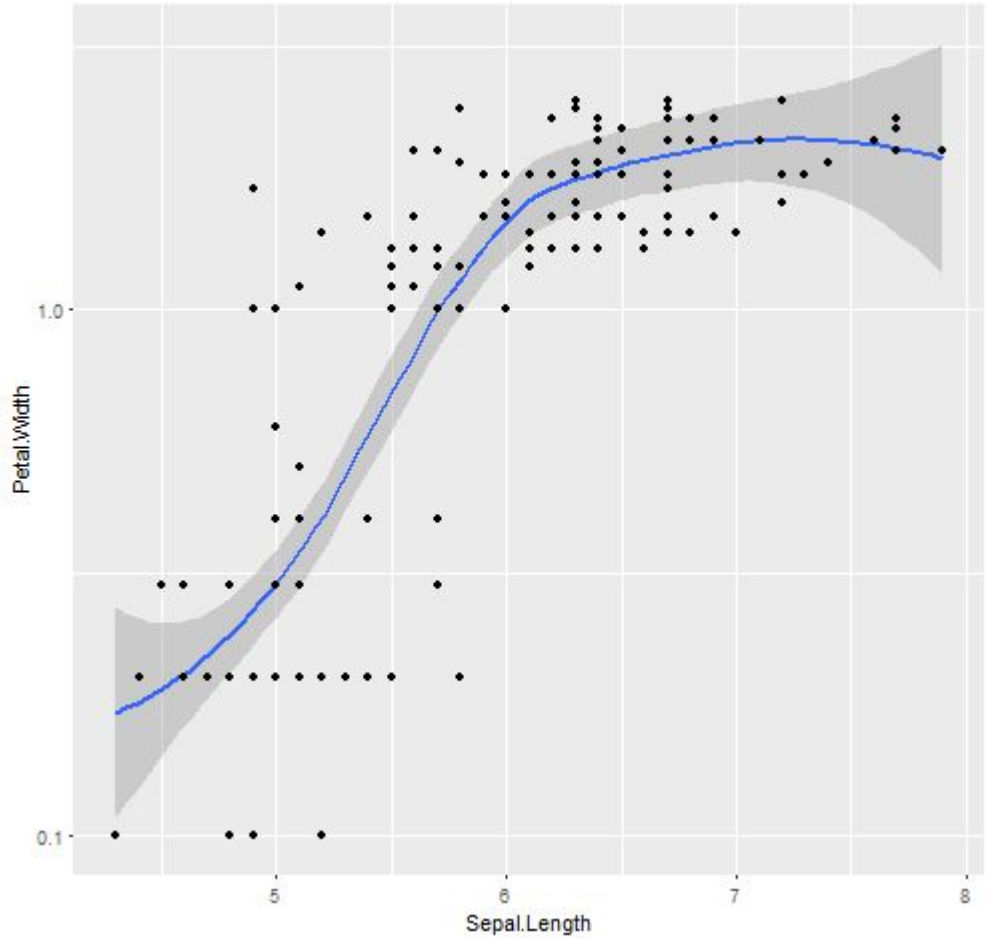

```
ggplot(data=dataset, aes(x=var, y=var2, …)) +
                                    scale_y_log10()
```



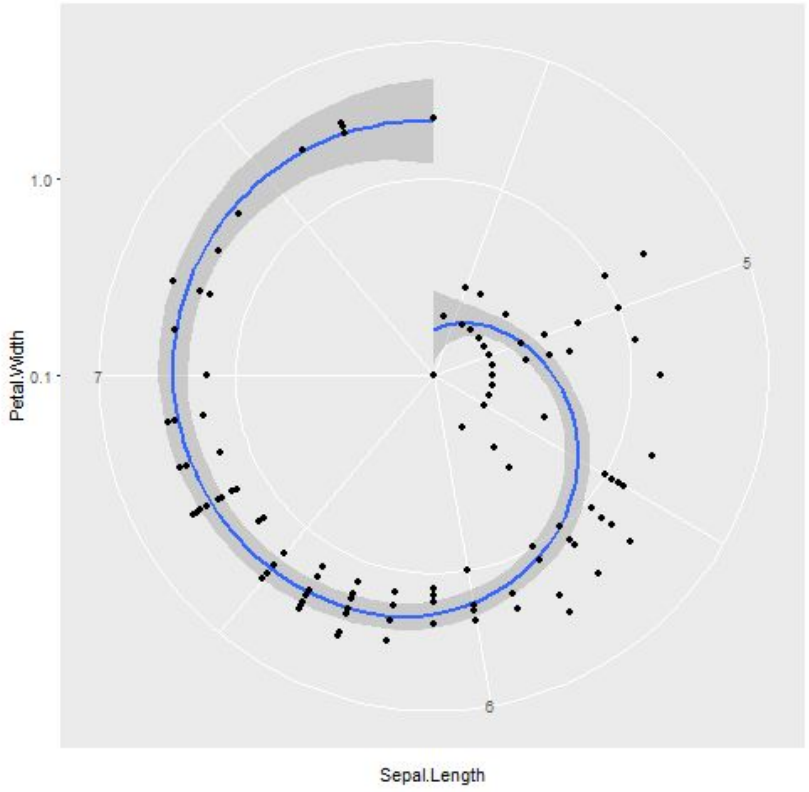

```
ggplot(data=dataset, aes(x=var, y=var2, …)) +
                                  scale_y_log10() +
                                    geom_smooth()
```




```
ggplot(data=dataset, aes(x=var, y=var2, …)) +
                                  scale_y_log10() +
                                     geom_smooth() +
                                  eom_point() +
```




```
ggplot(data=dataset, aes(x=var, y=var2, …)) +
                                  scale_y_log10() +
                                    geom_smooth() +
                                  geom_point() +
                                coord_polar()
```




```
ggplot(data=dataset, aes(x=var, y=var2, …)) +
                                scale_y_log10() +
                        geom_smooth(colour="red") +
                    geom_point(colour=yellow) +
                coord_polar()  +
                            theme_bw()
```

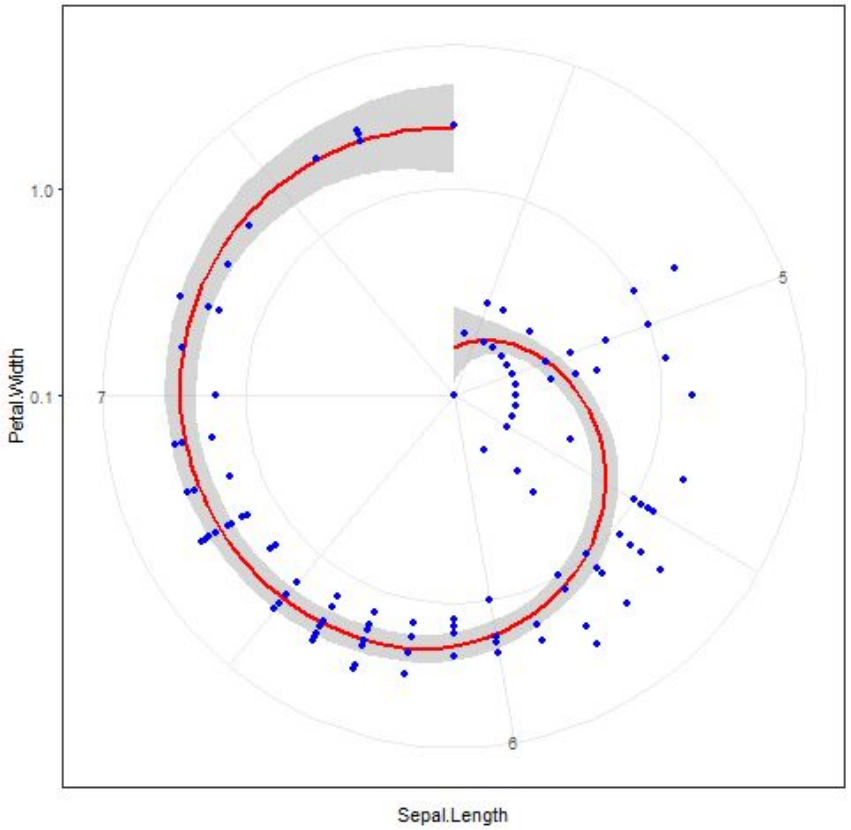
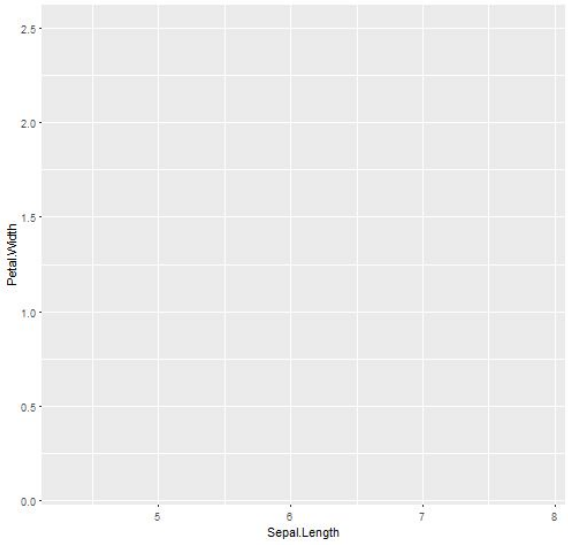
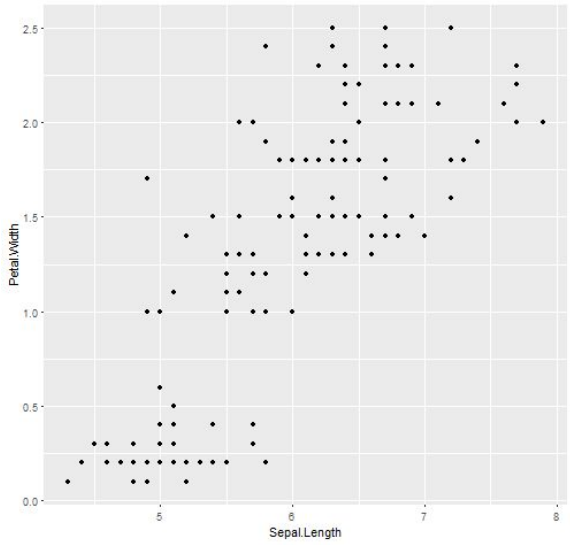


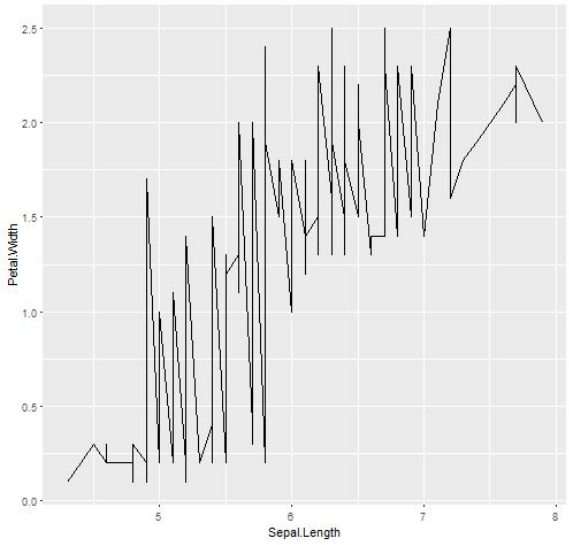
Image source: https://cityofbyroncity.files.wordpress.com/2012/03/puppy-pics32.jpg

ggplots are built up in layers

```
base_plot<-ggplot(data=dataset, aes(x=var, y=var2, …))

point_plot<-base_plot + geom_point()

line_plot<-base_plot + geom_line()
```

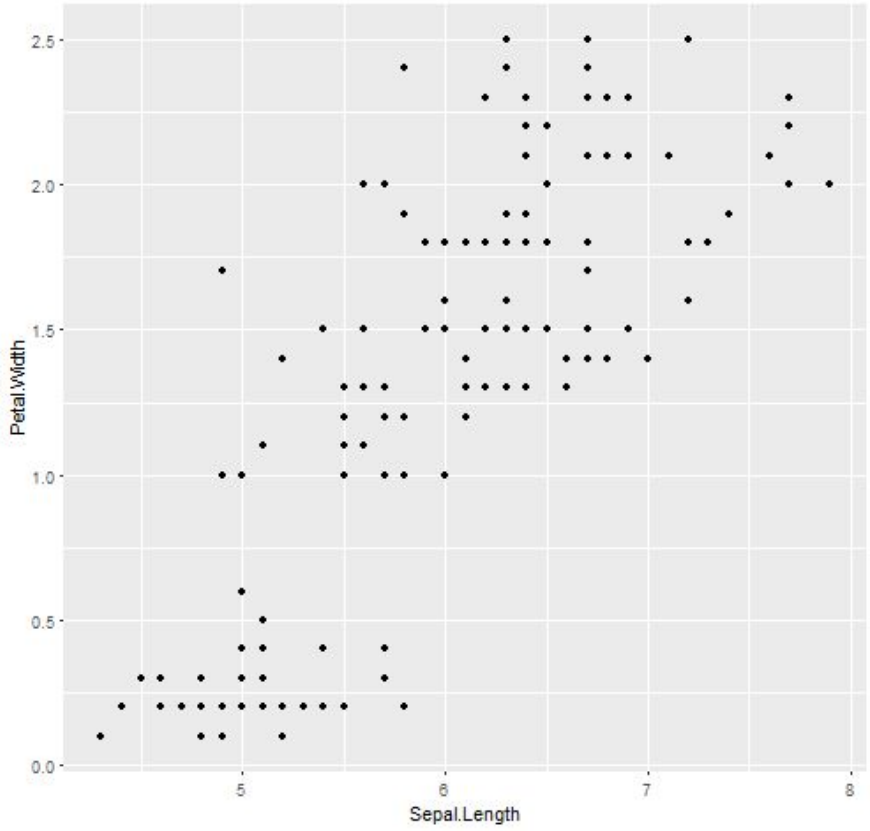


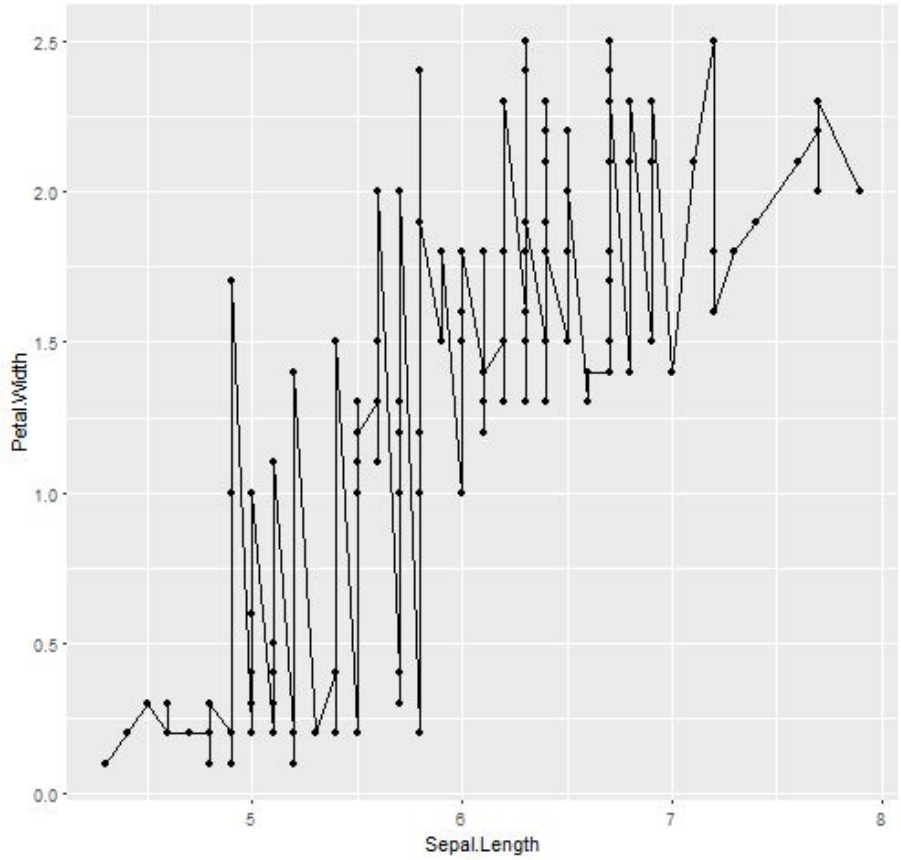




Each step of the grammar can be saved as variables and added together.

```
base_plot + geom_point()

base_plot + geom_point() + geom_line()
```

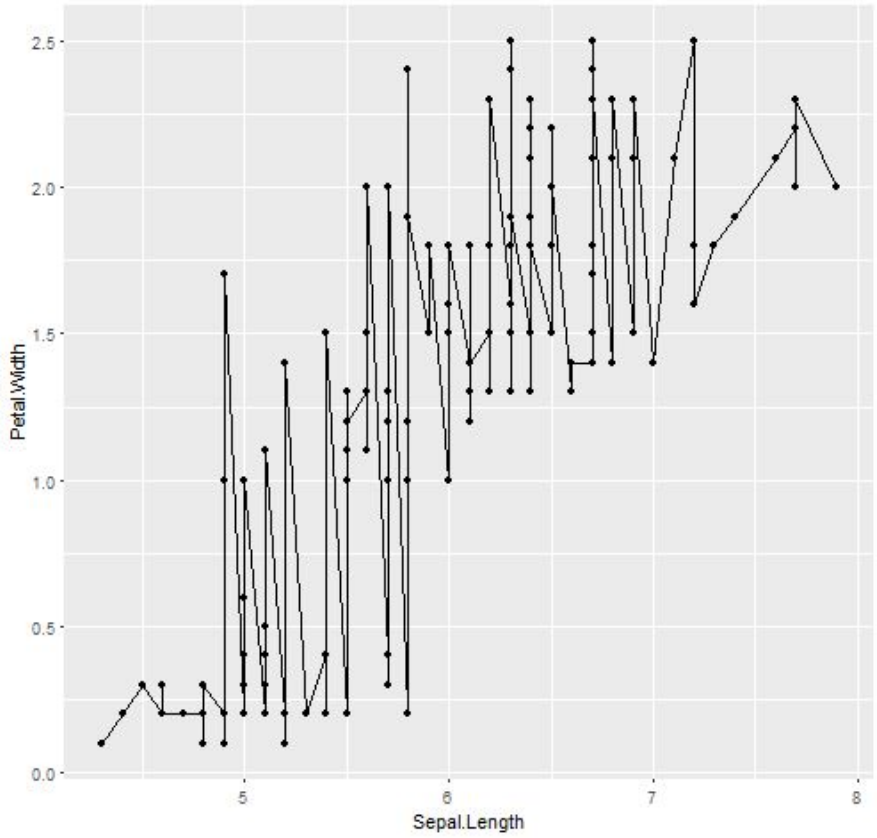




But you can't just add two finished plots.

```
point_plot + line_plot
Error: Don't know how to add o to a plot
```

```
both_plot<-base_plot + geom_point() + geom_line()
```

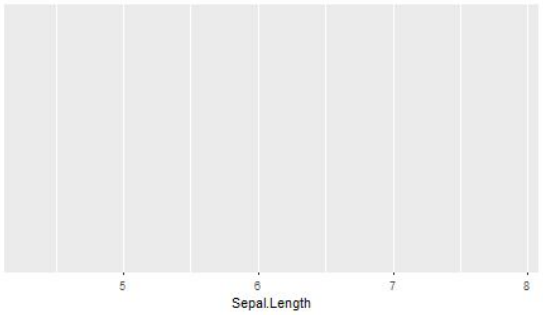


General procedure

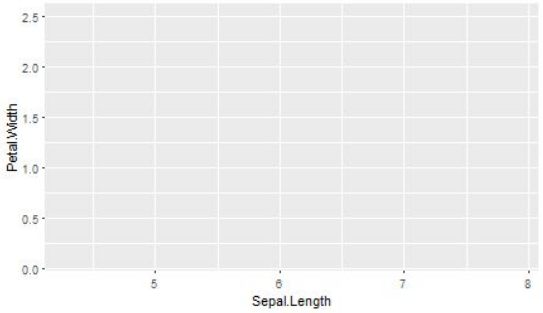
- Make your empty plot with `ggplot()` function
  - This is the point where you specify the data and variables
- Add the actual information with `geom()`
  - You can add multiple layers of data in multiple geoms
- Tweak the axes with `scales()`
- Choose the projection with `coords()`
  - Mostly you skip this step
- Play with the aesthetic within previous steps and use `themes()`

Exercise 04: the plot thickens

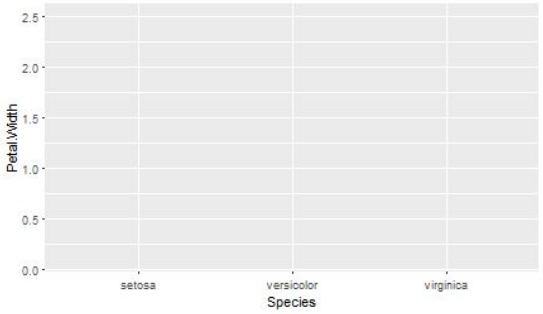
- Go through each step and explore in the Iris data…
- Then build your own plots out of your own data



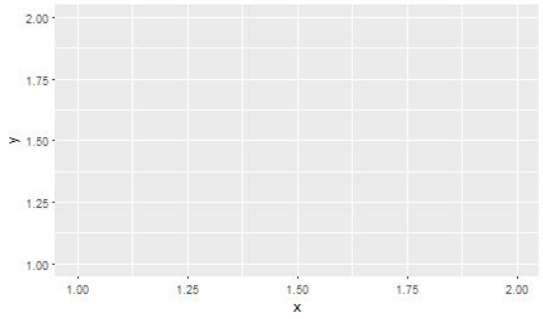

```
base_cont<-ggplot(data=iris,
aes(x=Sepal.Length))
```




```
base_cont_cont<-ggplot(data=iris,
aes(x=Sepal.Length, y=Petal.Width))
```




```
base_cont_cat<-ggplot(data=iris,
aes(x=Species, y=Petal.Width))
```




```
square<-data.frame(x=c(1,2,2,1,1),
                   y=c(2,2,1,1,2))
base_cont_shape<-ggplot(data=square,
                   aes(x=x, y=y))
```

Geoms for single continuous variables

```
geom_bar()
geom_histogram()
geom_density()
geom_freqpoly()
```

Try each one with base_cont

Further references: http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Geoms for continuous by continuous variables

```
geom_point()
geom_jitter()
geom_line()
geom_rug()
geom_step()
geom_tile()
geom_bin2d()
geom_hex()
geom_quantile()
geom_smooth()
```

Try each one with base_cont_cont

Further references: http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Geoms for continuous by categorical variables

```
geom_boxplot()
```

```
geom_boxplot() + geom_jitter()
```

Try with base_cont_cat

Further references: http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Geoms for shapes

```
geom_path()
geom_polygon()
geom_area()
```

Try each one with base_cont_shape

Further references: http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Geoms for error bars

Ggplots doesn't do this for you, so you need to prepare the data first…

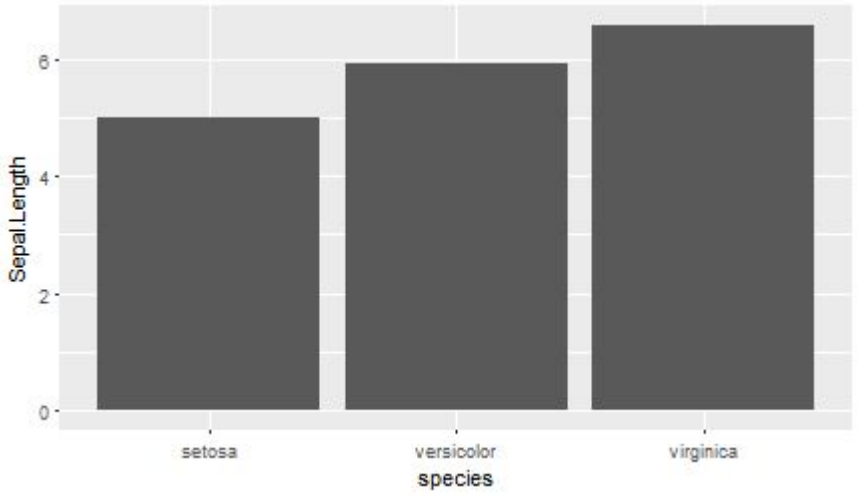
e.g. for error bars reflecting standard deviations by group:

```
summary_grabber<-function(x,group){data.frame(species=group,
                      sepal.Length=mean(iris[which(iris$Species==group),x]),
                      sd=sd(iris[which(iris$Species==group),x]))}

      se_dat<-rbind(summary_grabber("Sepal.Length","setosa"),
                    summary_grabber("Sepal.Length","versicolor"),
                    summary_grabber("Sepal.Length","virginica"))

se_dat$sd_lower<-se_dat$Sepal.Length-se_dat$sd
se_dat$sd_upper<-se_dat$Sepal.Length+se_dat$sd
```

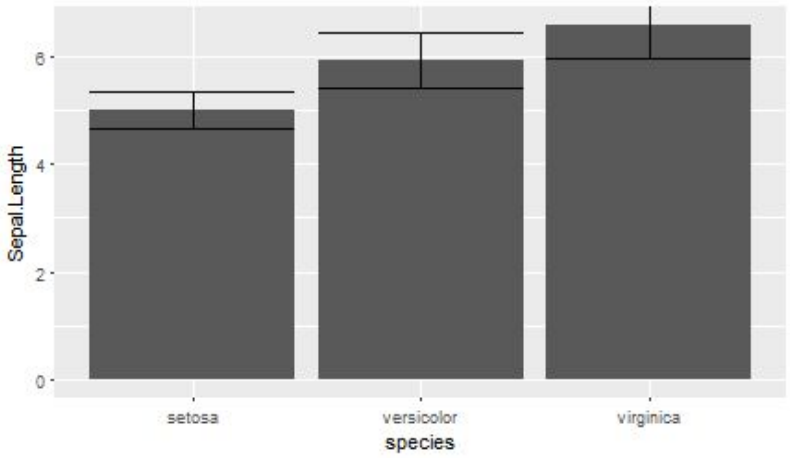
Geoms for error bars

```
> se_dat
     species Sepal.Length        sd     ymin     ymax sd_lower sd_upper
1     setosa        5.006 0.3524897 4.653510 5.358490 4.653510 5.358490
2 versicolor        5.936 0.5161711 5.419829 6.452171 5.419829 6.452171
3  virginica        6.588 0.6358796 5.952120 7.223880 5.952120 7.223880
```

```
bar_base <-ggplot(data=se_dat,
            aes(x=species,
            y=Sepal.Length)) +
      geom_bar(stat="identity")
```




```
                                  bar_base +
geom_errorbar(ymin=se_dat$sd_lower,
              ymax=se_dat$sd_upper)
```



Geoms for error bars

```
geom_errorbar()
geom_linerange()
geom_pointrange()
geom_crossbar()
```

Note: `geom_errorbarh()` works the same as `geom_errorbar()` but for the x axis

Try each one with bar_base

Scales and axes

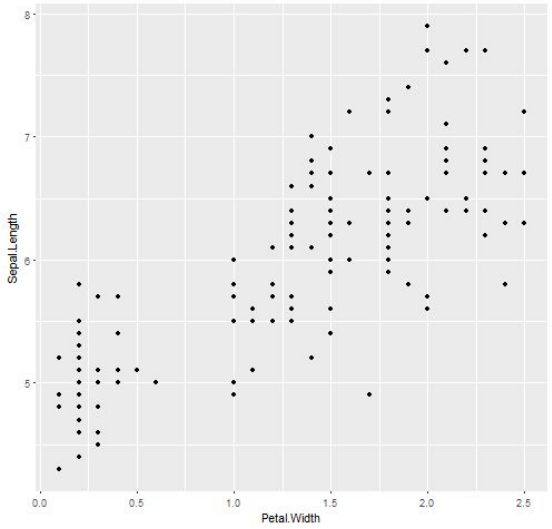
Reverse the axes: scale_x_reverse() and scale_y_reverse()

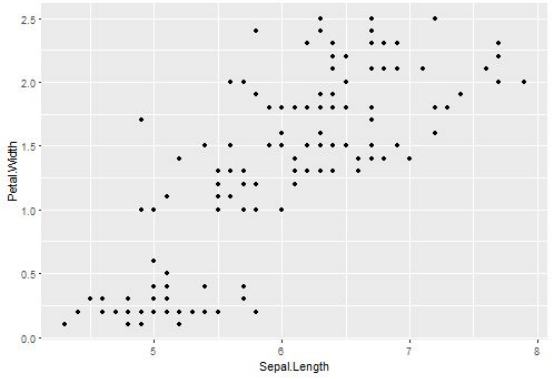
- Transform the axes:
  - At the point of plotting:
    - ggplot(data=dataset, aes(x=log10(var), y=scale(var2))) + geom_point()
  - After plotting:
    - scale_x_log10() and scale_y_log10()
    - scale_x_sqrt() and scale_y_sqrt()
    - coord_trans(x="log10", y="sqrt")

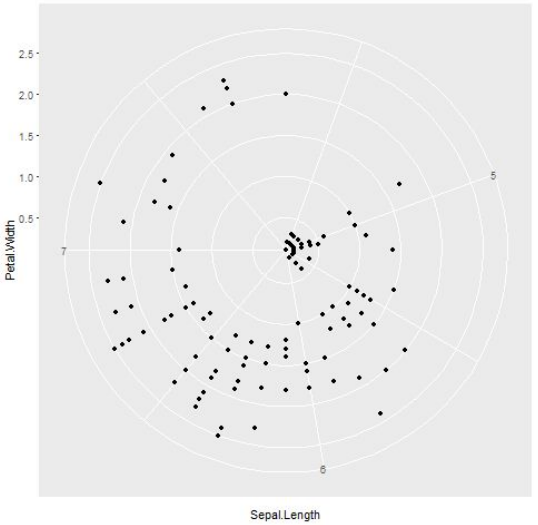
Coords

Change the projection

- **coord_flip**() – reverse x and y axes
- **coord_equal()** – fix aspect ratio, like `asp(TRUE)` in base R
- **coord_polar**() – polar coordinates
- Also **coord_map**() for map projections and **coord_cartesian**(), which is just the default.







Try each one!

Adding and changing text

For the plot area, use a geom:

```
geom_text(label="text", x=xpos,y=ypos)
```

Try each of these steps with bar_base:

```
geom_text(label="hello")
geom_text(label="hello", x=1)
geom_text(label="hello", x=1,y=1)
geom_text(label="hello", x=1,y=1, colour="white")
```

The title has its own segment;

```
ggtitle("Title text goes here")
```

Changing axis labels is similar:

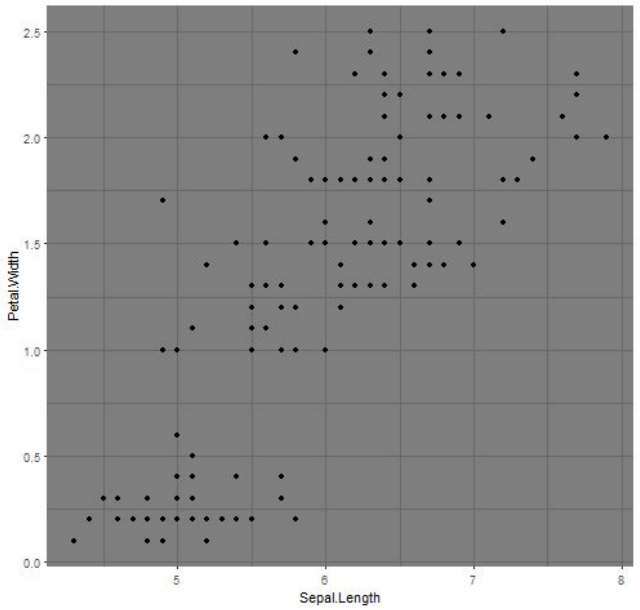
```
xlab("X axis label")
ylab("Y axis label")
```

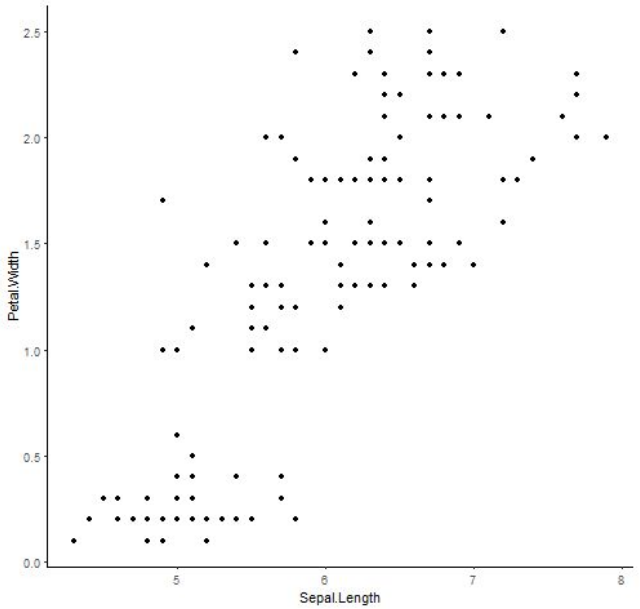
Try to give bar_base a unique title and axis labels.

Themes

Many existing themes

- theme_grey() (default)
- theme_bw()
- theme_linedraw()
- theme_light()
- theme_dark()
- theme_minimal()
- theme_classic()
- theme_void()





Try each one!

Themes

You can make your own, save it as a variable you can add on. Here's just one example to play with:

```
custom_theme <- theme(

                      )
```

Further education:
https://bookdown.org/rdpeng/RProgDA/building-a-new-theme.html

Themes

You can make your own, save it as a variable you can add on. Here's just one example to play with:

```
custom_theme <- theme(
                  plot.title = element_text(
                                                          ),
                  panel.grid.major = element_line(
                                                           ),
                  panel.background = element_rect(         )
                         )
```

Further education:
https://bookdown.org/rdpeng/RProgDA/building-a-new-theme.html

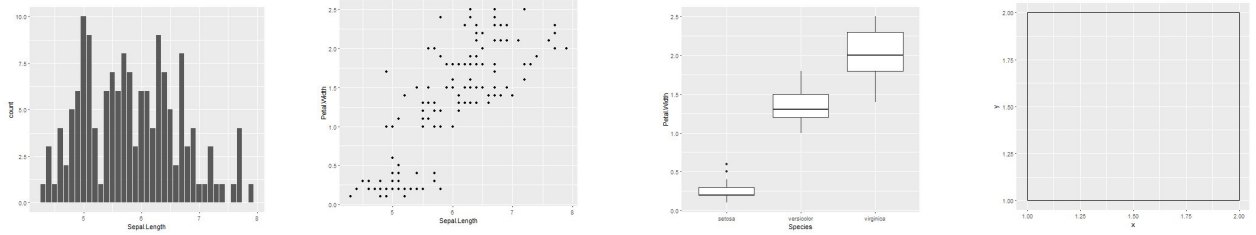
Themes

You can make your own, save it as a variable you can add on. Here's just one example to play with:

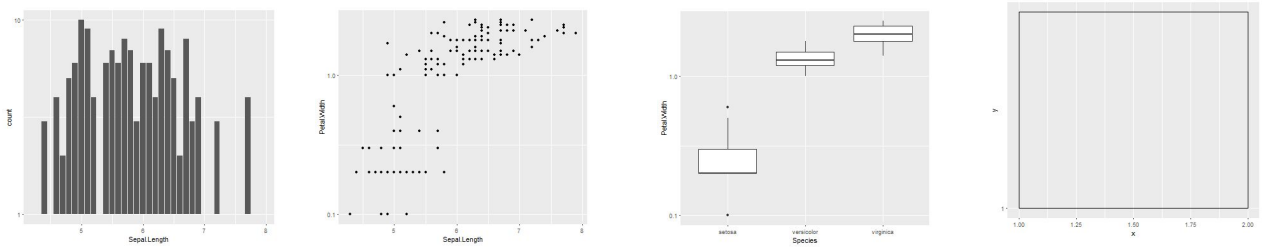
```
custom_theme <- theme(
                  plot.title = element_text(color = "blue",
                                            face = "bold", size=rel(2)),
                  panel.grid.major = element_line(colour="red",
                                                  linetype = "dotted"),
                  panel.background = element_rect(fill = "yellow")
                         )

point_plot + ggtitle("Your own theme") + custom_theme
```

Further education:
https://bookdown.org/rdpeng/RProgDA/building-a-new-theme.html

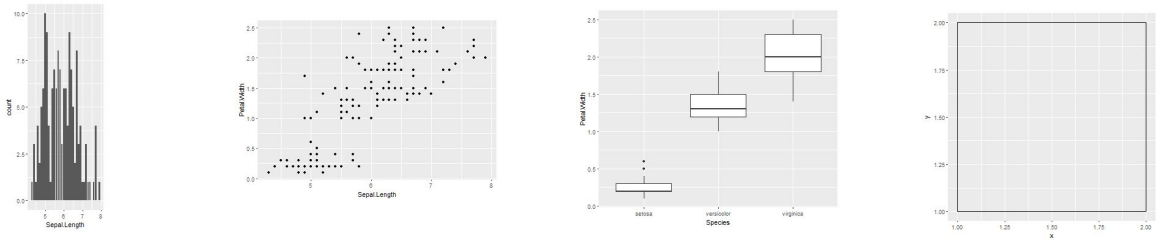
geoms



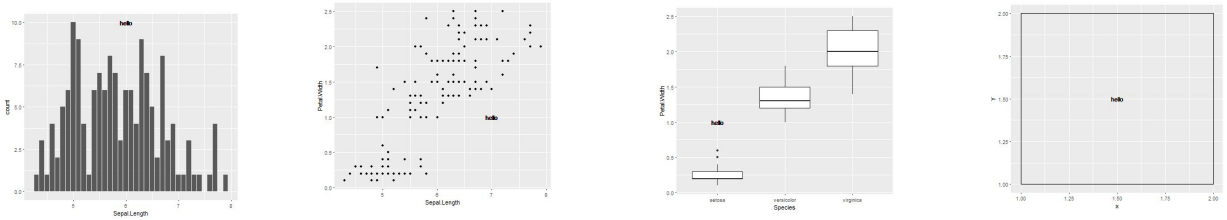
scales and axes
scale_y_log()



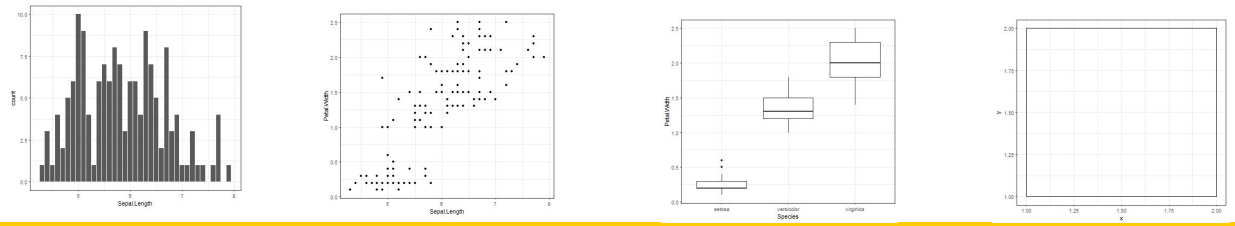
coords
coord_equal()



Adding and changing text
geom_text()



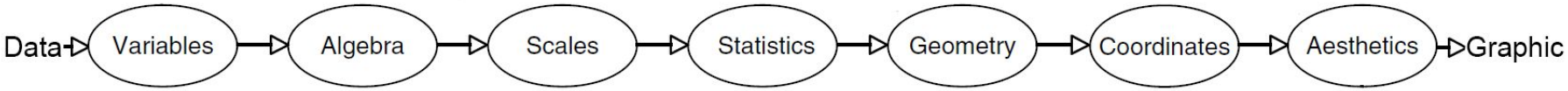
Themes
Theme_bw()



Recommended reading

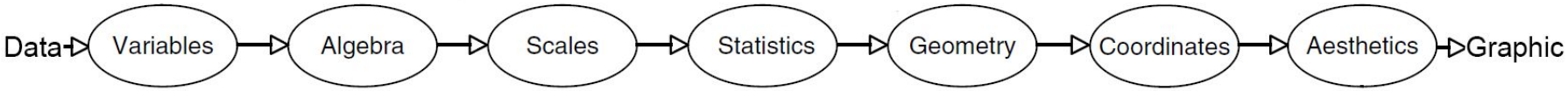
- Tufte, E., & Graves-Morris, P. (2014). The visual display of quantitative information.; 1983
- Johnston, S. (2014) R Base graphics: an Idiot's guide http://rpubs.com/SusanEJohnston/7953
- Wilkinson, L. (2007) The grammar of graphics. *Journal of statistical software* 17(3). doi: 10.1002/wics.118
- Wickham, H. (2016). ggplot2: elegant graphics for data analysis. Springer.
(ANU library has one copy)

Bonus slides: the grammar of graphics



- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)

```
> iris$Sepal.Length
  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2
 [30] 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9
 [59] 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7
 [88] 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4
[117] 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7
[146] 6.7 6.3 6.5 6.2 5.9
>
> iris$Sepal.Width
  [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4
 [30] 3.2 3.1 3.4 4.1 4.2 3.1 3.2 3.5 3.6 3.0 3.4 3.5 2.3 3.2 3.5 3.8 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4
 [59] 2.9 2.7 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7 2.2 2.5 3.2 2.8 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0 3.4 3.1
 [88] 2.3 3.0 2.5 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7 3.0 2.9 3.0 3.0 2.5 2.9 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2
[117] 3.0 3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6 3.0 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2 3.3
[146] 3.0 2.5 3.0 3.4 3.0
>
> iris$Species
  [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [11] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [21] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [41] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [51] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [61] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [71] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [81] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[101] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
[111] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
[131] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
[141] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

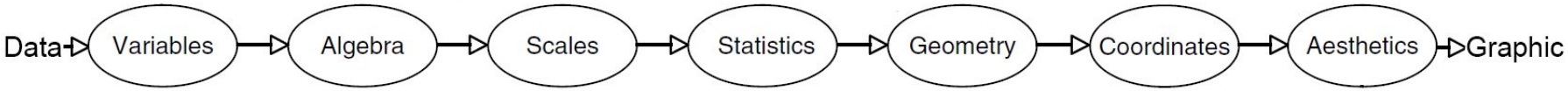


- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)

- Algebra = combining the varset together.

```
> iris[,c("Sepal.Length","Sepal.Width","Species")]
```

	Sepal.Length	Sepal.Width	Species
1	5.1	3.5	setosa
2	4.9	3.0	setosa
3	4.7	3.2	setosa
4	4.6	3.1	setosa
5	5.0	3.6	setosa
6	5.4	3.9	setosa
7	4.6	3.4	setosa
8	5.0	3.4	setosa
9	4.4	2.9	setosa
10	4.9	3.1	setosa
11	5.4	3.7	setosa
12	4.8	3.4	setosa
13	4.8	3.0	setosa
14	4.3	3.0	setosa
15	5.8	4.0	setosa
16	5.7	4.4	setosa
17	5.4	3.9	setosa
18	5.1	3.5	setosa
19	5.7	3.8	setosa
20	5.1	3.8	setosa
21	5.4	3.4	setosa
22	5.1	3.7	setosa
23	4.6	3.6	setosa
24	5.1	3.3	setosa
25	4.8	3.4	setosa
26	5.0	3.0	setosa
27	5.0	3.4	setosa
28	5.2	3.5	setosa
29	5.2	3.4	setosa
30	4.7	3.2	setosa
31	4.8	3.1	setosa
32	5.4	3.4	setosa
33	5.2	4.1	setosa
34	5.5	4.2	setosa
59	6.6	2.9	versicolor
60	5.2	2.7	versicolor
61	5.0	2.0	versicolor
62	5.9	3.0	versicolor
63	6.0	2.2	versicolor
64	6.1	2.9	versicolor
65	5.6	2.9	versicolor
66	6.7	3.1	versicolor
67	5.6	3.0	versicolor
68	5.8	2.7	versicolor
69	6.2	2.2	versicolor
70	5.6	2.5	versicolor
71	5.9	3.2	versicolor
72	6.1	2.8	versicolor
73	6.3	2.5	versicolor
74	6.1	2.8	versicolor
75	6.4	2.9	versicolor
76	6.6	3.0	versicolor
77	6.8	2.8	versicolor
78	6.7	3.0	versicolor
79	6.0	2.9	versicolor
80	5.7	2.6	versicolor
81	5.5	2.4	versicolor
82	5.5	2.4	versicolor
83	5.8	2.7	versicolor
84	6.0	2.7	versicolor
85	5.4	3.0	versicolor
86	6.0	3.4	versicolor
87	6.7	3.1	versicolor
88	6.3	2.3	versicolor
89	5.6	3.0	versicolor
90	5.5	2.5	versicolor
91	5.5	2.6	versicolor
92	6.1	3.0	versicolor
93	5.8	2.6	versicolor
94	5.0	2.3	versicolor
95	5.6	2.7	versicolor
116	6.4	3.2	virginica
117	6.5	3.0	virginica
118	7.7	3.8	virginica
119	7.7	2.6	virginica
120	6.0	2.2	virginica
121	6.9	3.2	virginica
122	5.6	2.8	virginica
123	7.7	2.8	virginica
124	6.3	2.7	virginica
125	6.7	3.3	virginica
126	7.2	3.2	virginica
127	6.2	2.8	virginica
128	6.1	3.0	virginica
129	6.4	2.8	virginica
130	7.2	3.0	virginica
131	7.4	2.8	virginica
132	7.9	3.8	virginica
133	6.4	2.8	virginica
134	6.3	2.8	virginica
135	6.1	2.6	virginica
136	7.7	3.0	virginica
137	6.3	3.4	virginica
138	6.4	3.1	virginica
139	6.0	3.0	virginica
140	6.9	3.1	virginica
141	6.7	3.1	virginica
142	6.9	3.1	virginica
143	5.8	2.7	virginica
144	6.8	3.2	virginica
145	6.7	3.3	virginica
146	6.7	3.0	virginica
147	6.3	2.5	virginica
148	6.5	3.0	virginica
149	6.2	3.4	virginica
150	5.9	3.0	virginica

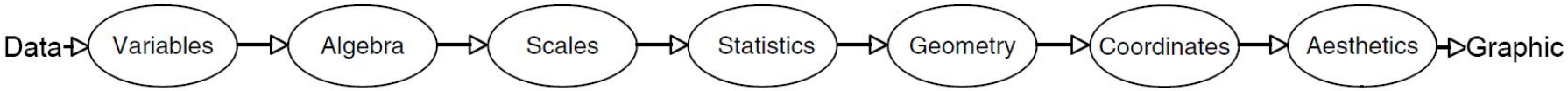


- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)
- Algebra = combining the varset together.
- Scales = identifying whether it is nominal, ordinal, interval, or ratio (because each one should be plotted differently)

```
> head(iris[,c("Sepal.Length","Sepal.Width","Species")])
  Sepal.Length Sepal.Width Species
1          5.1         3.5  setosa
2          4.9         3.0  setosa
3          4.7         3.2  setosa
4          4.6         3.1  setosa
5          5.0         3.6  setosa
6          5.4         3.9  setosa
```

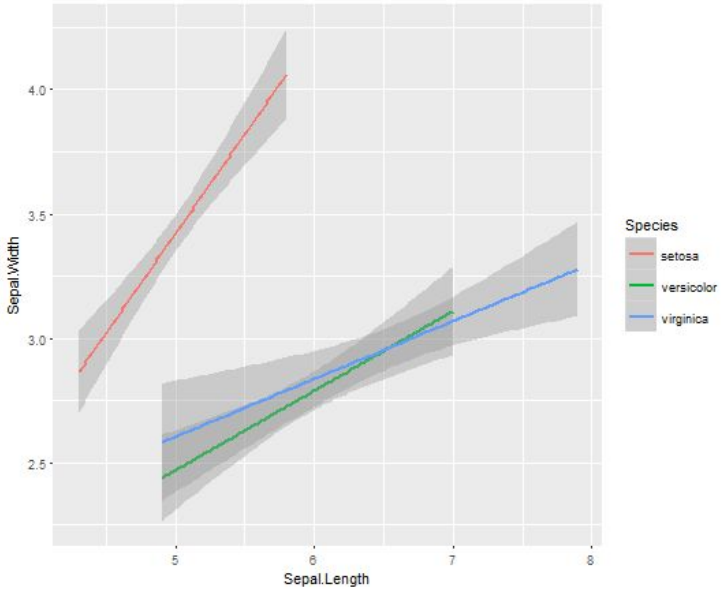
Ratio (continuous)	Ratio (continuous)	Ordinal (categorical)

Fun Tip: The order in the factor will be the order in the plot: to change the order in the plot, change the order of the factor AT THIS POINT.

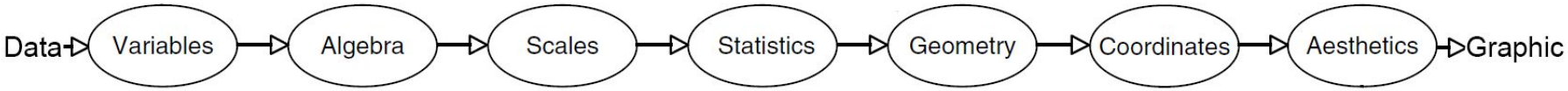


- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)
- Algebra = combining the varset together.
- Scales = identifying whether it is nominal, ordinal, interval, or ratio (because each one should be plotted differently)
- Statistics = receives a varset, computes statistical summaries, and outputs another varset (e.g. means, medians, correlations…)

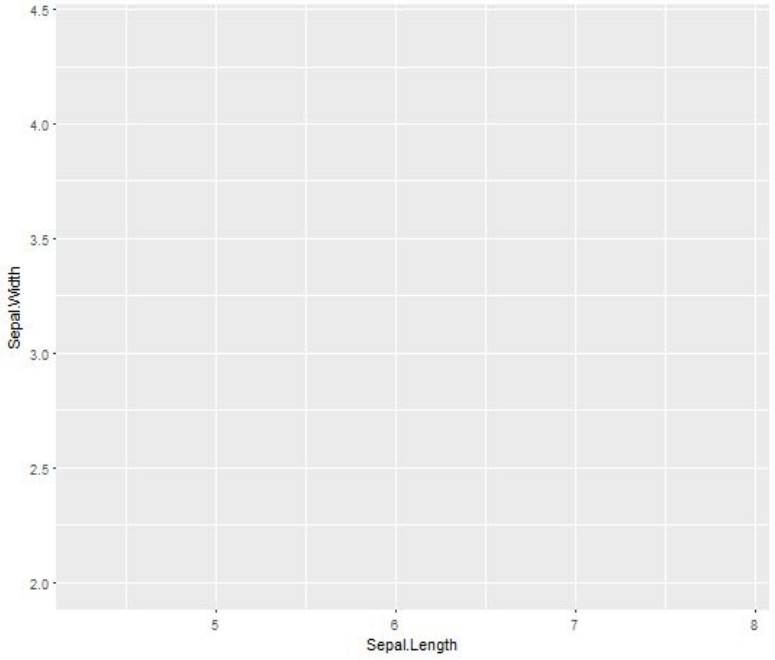
```
> summary(glm(Sepal.Length ~ Sepal.Width + Species, dat=iris))

Call:
glm(formula = Sepal.Length ~ Sepal.Width + Species, data = iris)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.30711  -0.25713  -0.05325   0.19542   1.41253

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.2514     0.3698   6.089 9.57e-09 ***
Sepal.Width         0.8036     0.1063   7.557 4.19e-12 ***
Speciesversicolor   1.4587     0.1121  13.012  < 2e-16 ***
Speciesvirginica    1.9468     0.1000  19.465  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1918059)

    Null deviance: 102.168  on 149  degrees of freedom
Residual deviance:  28.004  on 146  degrees of freedom
AIC: 183.94

Number of Fisher Scoring iterations: 2
```



Fun Tip: For most statistics, you can do this manually beforehand, or let ggplot do it for you.

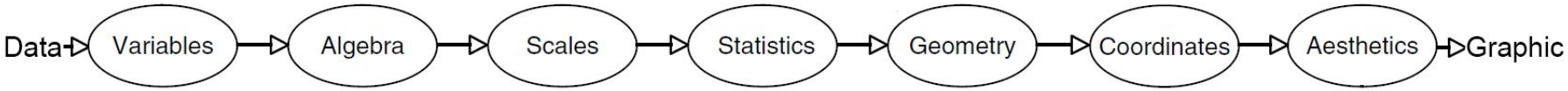


- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)
- Algebra = combining the varset together.
- Scales = identifying whether it is nominal, ordinal, interval, or ratio (because each one should be plotted differently)
- Statistics = receives a varset, computes statistical summaries, and outputs another varset (e.g. means, medians, correlations…)
- Geometry = graphing functions injectively map an *m*-dimensional bounded region to an *n*-dimensional real space

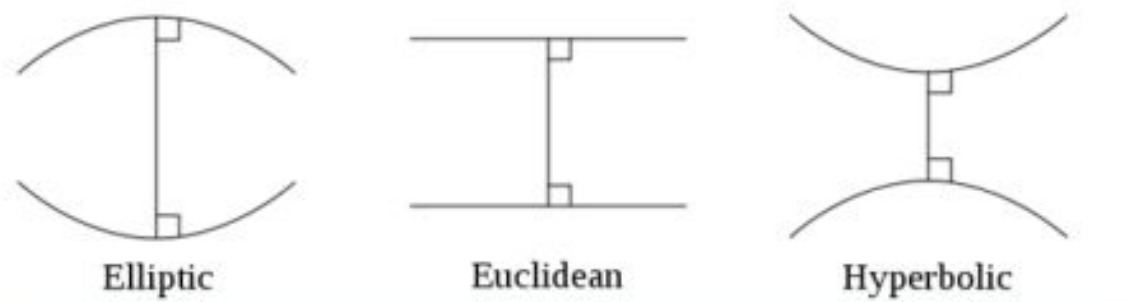


AKA the extent of the plot

Fun Tip: ggplot plots the data AFTER this. So if you manually change the axes, you change the plot fundamentally (not just plotting everything and shrinking the window down).



- Variable = each piece of data we wish to visualise
  - (Varset is a set of one or more variables)
- Algebra = combining the varset together.
- Scales = identifying whether it is nominal, ordinal, interval, or ratio (because each one should be plotted differently)
- Statistics = receives a varset, computes statistical summaries, and outputs another varset (e.g. means, medians, correlations…)
- Geometry = graphing functions injectively map an *m*-dimensional bounded region to an *n*-dimensional real space
- Coordinates = Cartesian coordinates. 'nuff said.



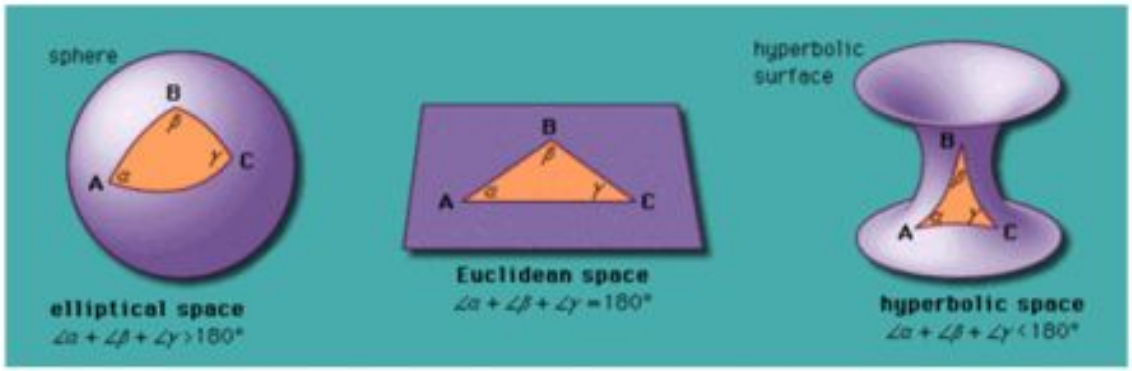
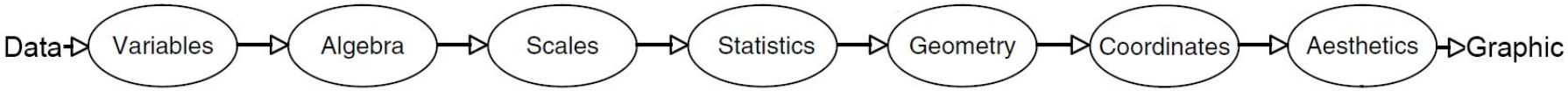
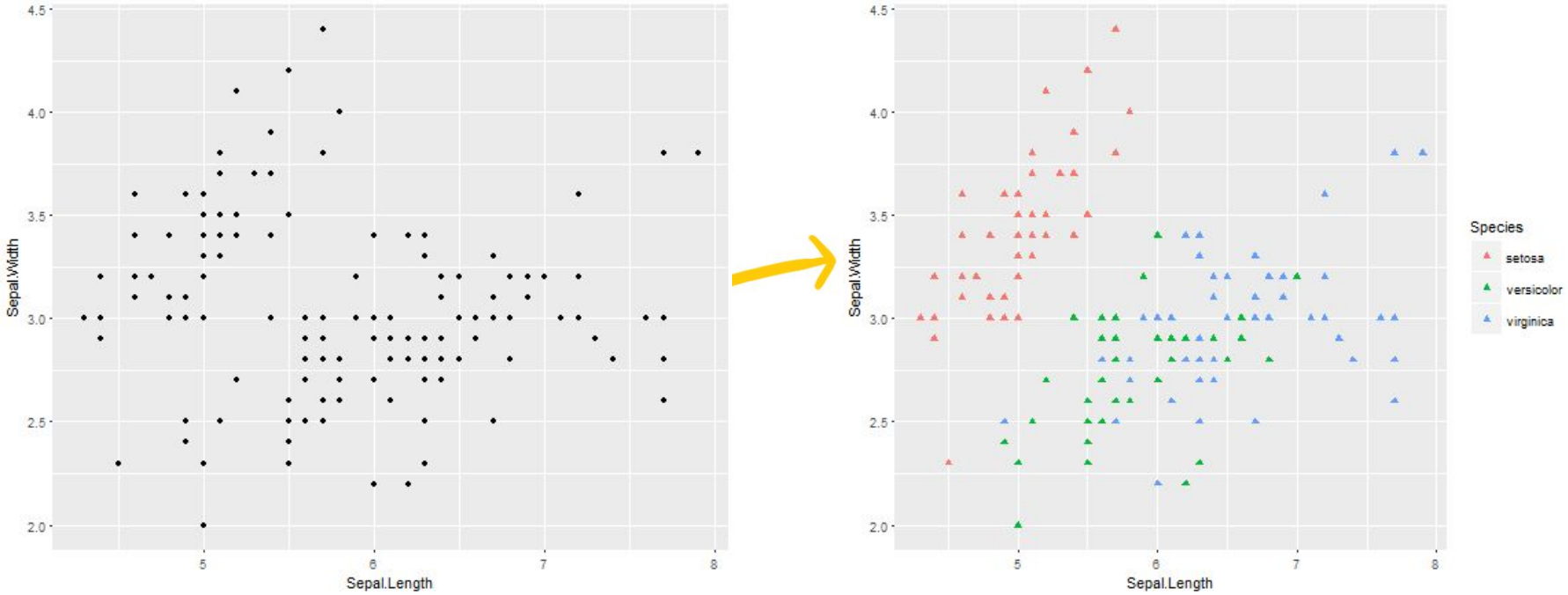
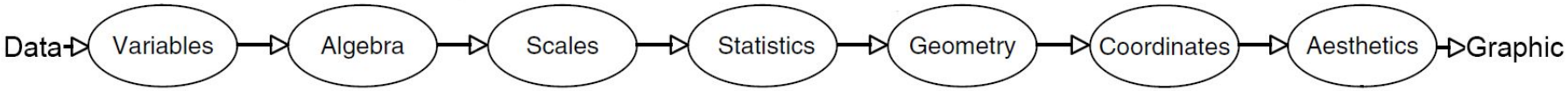


Image source: https://naiadseye.files.wordpress.com/2014/10/euclidean-udnerstand-e1414490051530.png?w=470&h=289



- Algebra = combining the varset together.
- Scales = identifying whether it is nominal, ordinal, interval, or ratio (because each one should be plotted differently)
- Statistics = receives a varset, computes statistical summaries, and outputs another varset (e.g. means, medians, correlations…)
- Geometry = graphing functions injectively map an *m*-dimensional bounded region to an *n*-dimensional real space
- Coordinates = Cartesian coordinates. 'nuff said.
- Aesthetics = colours, size, shapes,





… But every step is customisable!

```
> iris[,c("Sepal.Length","Sepal.Width","Species")]
    Sepal.Length Sepal.Width    Species
1            5.1         3.5     setosa
2            4.9
3            4.7
4            4.6
5            5.0
6            5.4
7            4.6
8            5.0
9            4.4
10           4.9
11           5.4
12           4.8
13           4.8
14           4.3
15           5.8
16           5.7
17           5.4
18           5.1
```