MANGOSOLUTIONS
data analysis that delivers

Introduction to ggplot2 Graphics

Leaping over the ggplot2 learning curve

Welcome to ggplot2 Workshop!

aka "Leaping over the ggplot2 learning curve"

Workshop Format

- Duration of 3 hours
- 10-15 minute break in the middle
- This is an interactive session - please ask questions at any point
- Exercises to be perfromed as a group / in groups
- Answers and scripts provided afterwards

MANGOSOLUTIONS
data analysis that delivers

Introduction to `ggplot2` Graphics

Overview

Preparing data for ggplot2

`ggplot2` - Introduction, Background and Philosophy

A Quick Introduction to the "Quick Plot"

Titles and Axes Labels

Plot Types - aka The "Geoms"

Layering

Aesthetics and Groups

Legend Control

Grouping and Panelling

Advanced Control

Data for Today

- Today we will use `tubeData`
- Something sector independent
- London Tube Performance Data from the TFL website
- Excess Travel Hours by Line
- The original data are available from http://data.london.gov.uk/datafiles/transport/assembly-tube-performance.xls

MANGOSOLUTIONS
data analysis that delivers

Data for Today

```
> tubeData <- read.csv("tubeData.csv")
> head(tubeData)
```

```
      Line Month Scheduled Excess TOTAL Opened Length Type Stations
1 Bakerloo     1     29.42   6.04 35.46   1906   23.2   DT       25
2 Bakerloo     2     29.42   6.54 35.96   1906   23.2   DT       25
3 Bakerloo     3     29.30   4.77 34.08   1906   23.2   DT       25
4 Bakerloo     4     29.30   5.40 34.70   1906   23.2   DT       25
5 Bakerloo     5     29.30   5.23 34.53   1906   23.2   DT       25
6 Bakerloo     6     29.30   5.03 34.33   1906   23.2   DT       25
```

Preparing Data for `ggplot2`

Recap: Factors

- A factor is an effecient way of storing categorical data
- Data appear to be character but are stored as numeric values
- By default R converts character data to type 'factor' in data.frames
- Factors can be confusing at first but are very important when working with ggplot2

Factor Example

```
> head(warpbreaks)
```

```
  breaks wool tension
1     26    A       L
2     30    A       L
3     54    A       L
4     25    A       L
5     70    A       L
6     52    A       L
```

```
> warpbreaks$wool
```

```
 [1] A A A A A A A A A A A A A A A A A A A A A A A A A A B B B B B B B B B
[36] B B B B B B B B B B B B B B B B B B B B
Levels: A B
```

```
> mode(warpbreaks$wool)  # Seemingly odd answer!
```

```
[1] "numeric"
```

```
> class(warpbreaks$wool)  # The wool column is a factor...
```

```
[1] "factor"
```

MANGOSOLUTIONS
data analysis that delivers

Creating Factors

- It is very easy to create a factor
- We simply use the `factor` function

```
> head(mtcars, 3)
```

```
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
> cylFactor <- factor(mtcars$cyl)
> cylFactor
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
Levels: 4 6 8
```

MANGOSOLUTIONS
data analysis that delivers

Factor Warning!!!

- What we see is not always what we get!
- The underlying numeric values of a factor may not match the values we "see"...

```
> cylFactor
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
Levels: 4 6 8
```

```
> as.numeric(cylFactor)
```

```
 [1] 2 2 1 2 3 2 3 1 1 2 2 3 3 3 3 3 3 1 1 1 1 3 3 3 3 1 1 1 3 2 3 1
```

- This leads to lines like:

```
> as.numeric(as.character(cylFactor))
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

MANGOSOLUTIONS
data analysis that delivers

So Why Bother with Factors?

- They are a useful way of representing categorical data
- Efficient way of storing data
- We have to use them to get the plots we want in ggplot2!

Data Manipulation using `reshape2`

Why use `reshape2`?

The `reshape2` package provides approaches to data aggregation and restructuring by "melting" a data frame (`melt`) and then "casting" (`dcast`) it into the required format.

- `ggplot2` requires data in the right format
- `reshape2` has two important functions for restructure data
- These functions are:
  - *`melt`*
  - *`dcast`*

MANGOSOLUTIONS
data analysis that delivers

"Melting" your data

- To melt the data frame (list or array), we must identify the ID and analysis variables
- By default, factor variables are treated as ID variables and the remaining variables as analysis (measure) variables
- We can specify our own using the arguments `id.vars` and `measure.vars`
- Specifying one assumes that you want all the others to be of the other type

Example of `melt`

```
> require(reshape2, quietly = TRUE)
> head(french_fries)
```

```
   time treatment subject rep potato buttery grassy rancid painty
61    1         1       3   1    2.9     0.0    0.0    0.0    5.5
25    1         1       3   2   14.0     0.0    0.0    1.1    0.0
62    1         1      10   1   11.0     6.4    0.0    0.0    0.0
26    1         1      10   2    9.9     5.9    2.9    2.2    0.0
63    1         1      15   1    1.2     0.1    0.0    1.1    5.1
27    1         1      15   2    8.8     3.0    3.6    1.5    2.3
```

```
> fryMelt <- melt(french_fries, id.vars = c("time", "treatment",
+     "subject", "rep"))
> head(fryMelt)
```

```
  time treatment subject rep variable value
1    1         1       3   1   potato   2.9
2    1         1       3   2   potato  14.0
3    1         1      10   1   potato  11.0
4    1         1      10   2   potato   9.9
5    1         1      15   1   potato   1.2
6    1         1      15   2   potato   8.8
```

MANGOSOLUTIONS
data analysis that delivers

Casting data into a new structure

- More often than not some further work is required in order to 'cast' the data into a new structure
- The `dcast` function accepts a formula which describes the shape of the output format

```
row_var_1 + row_var2 ~ col_var_1 + col_var2
```

- A "..." notation can be used in a formula to mean 'all other variables not included in the formula'
- A "." can be used to represent 'no variable' in the casting formula

MANGOSOLUTIONS
data analysis that delivers

Example of `cast`

```
> fryReCast <- dcast(fryMelt, ... ~ rep)
> head(fryReCast)
```

```
  time treatment subject variable    1    2
1    1         1       3   potato  2.9 14.0
2    1         1       3  buttery  0.0  0.0
3    1         1       3   grassy  0.0  0.0
4    1         1       3   rancid  0.0  1.1
5    1         1       3   painty  5.5  0.0
6    1         1      10   potato 11.0  9.9
```

MANGOSOLUTIONS
data analysis that delivers

Aggregation using `reshape2`

The `fun.aggregate` argument provides the ability to aggregate the data using summary functions such as mean

```
> meanFryMelt <- dcast(fryMelt, time + treatment + subject +
+     variable ~ ., fun.aggregate = mean)  #mean across replicates
> head(meanFryMelt)
```

```
  time treatment subject variable    NA
1    1         1       3   potato  8.45
2    1         1       3  buttery  0.00
3    1         1       3   grassy  0.00
4    1         1       3   rancid  0.55
5    1         1       3   painty  2.75
6    1         1      10   potato 10.45
```

The `recast` function (formerly `cast`)

- Whilst it can be very helpful to see the intermediate melted data, this step is not actually required
- The recast function offers the same functionality as melt and cast in a single function call

```
> fryReCast <- recast(french_fries, id.var = c("time",
+     "treatment", "subject", "rep"), formula = ... ~
+     rep)
> head(fryReCast)
```

```
  time treatment subject variable    1    2
1    1         1       3   potato  2.9 14.0
2    1         1       3  buttery  0.0  0.0
3    1         1       3   grassy  0.0  0.0
4    1         1       3   rancid  0.0  1.1
5    1         1       3   painty  5.5  0.0
6    1         1      10   potato 11.0  9.9
```

Note: The argument names for `recast` are `id.var` and `measure.var` as opposed to `id.vars` and `measure.vars` in the `melt` and `dcast` functions

MANGOSOLUTIONS
data analysis that delivers

Data aggregation

- Today we will use the `aggregate` function
- Hadley Wickham has also written `plyr` for data aggregation
- `reshape2` can also be used

To aggregate a data.frame using `aggregate` we:

- Name the data.frame using a 'data' argument
- Use a formula to specify the relationship between variables
- Use the FUN argument to specify how will aggregate
- Option arguments to the `FUN` come at the end

aggregate Example

```
> aggregate(Ozone ~ Month, data = airquality, FUN = mean, na.rm = TRUE)
```

```
  Month Ozone
1     5 23.62
2     6 29.44
3     7 59.12
4     8 59.96
5     9 31.45
```

MANGOSOLUTIONS
data analysis that delivers

Exercises

1. Create a subset of the tubeData retaining only data for the first 12 months
2. Using reshape2, recast this data into a new structure where each column contains data for a different month
3. Calculate the average Excess by line

MANGOSOLUTIONS
data analysis that delivers

One final piece of useful code...

- When working with `tubeData` it will help to use familiar colours!

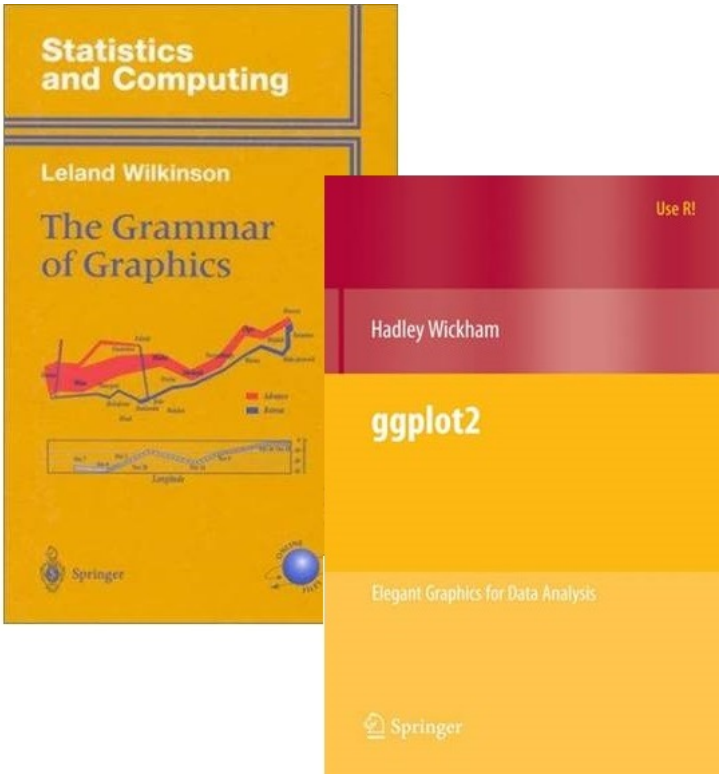
```
> lineCols <- c("brown", "red", "pink", "darkgreen", "grey",
+               "magenta", "black", "navy", "blue", "turquoise")
```

MANGOSOLUTIONS
data analysis that delivers

ggplot2

Introduction

- ggplot2 is a graphical package created by Hadley Wickham which implements the ideas found in the book "The Grammar of Graphics"



MANGOSOLUTIONS
data analysis that delivers

Introduction

- Plots are stored in objects
- Easy to split data across panels
- Plots are built up by layering
- The package is not part of the standard R installation, it must be installed from CRAN
  - `install.packages("ggplot2")`
  - `library(ggplot2)`
- Note: ggplot2 has many dependencies which must also be installed

MANGOSOLUTIONS
data analysis that delivers

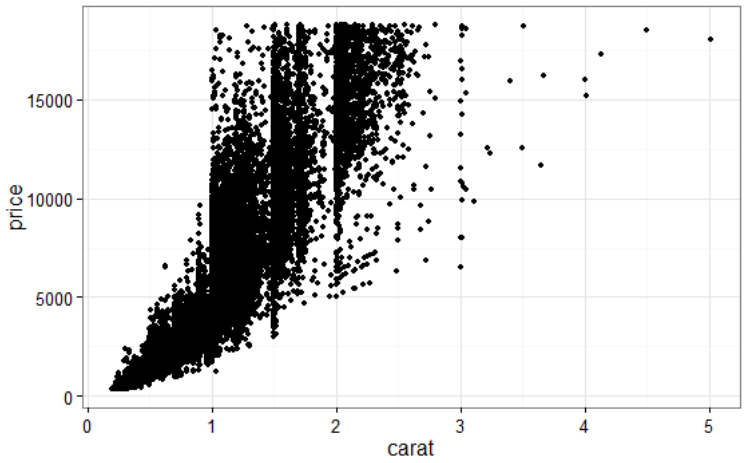
The Layering Concept

- Create a graphical ("ggplot") object using one of two primary functions: `qplot` or `ggplot`
- Modify this plot by adding "layers"
- Layers may contain:
  - *New data*
  - *Scales mapping aesthetics to data*
  - *A geometric object*
  - *A statistical transformation*
  - *Position adjustments within the plot area*
  - *Faceting (panelling)*
  - *The coordinate system itself!*
- More on this later...

The `qplot` function

- Similar to the vanilla plot function for traditional graphics
- It accepts x and y values and an optional data argument

```
> myPlot <- qplot(carat, price, data = diamonds)
> myPlot   # Note the way in which we create an object and print it
> # Alternatively
> # qplot(carat, price, data = diamonds)
```



MANGOSOLUTIONS
data analysis that delivers

The `ggplot` function

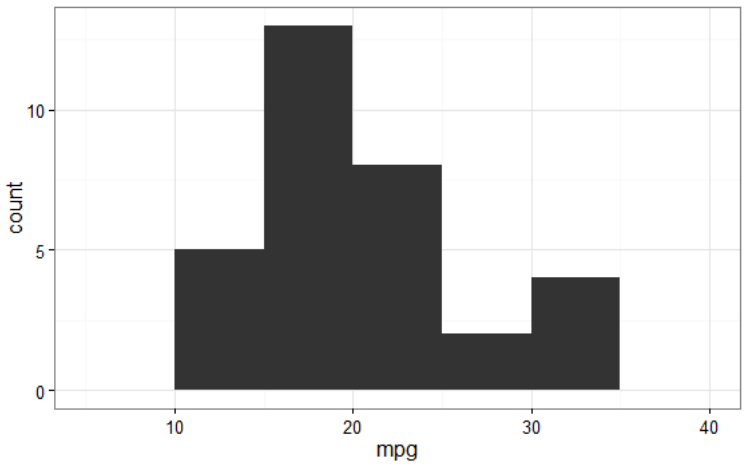
- It is almost always possible to use the `qplot` function
- However `qplot` is designed for quick plotting and therefore makes several assumptions
  - *Cartesian coordinates*
  - *Scatter plot by default*
  - *etc.*
- The `ggplot` function makes no assumptions about the format of the eventual graph
- Layers, such as geometric layers, must be specified in order to draw a graph
- However it is a lot more flexible than `qplot`
- More later...

MANGOSOLUTIONS
data analysis that delivers

A Quick Introduction to the "Quick Plot"

- The qplot function makes many assumptions about way in which you wish to plot your data
- So if only one variable is provided, the qplot function will draw a histogram
- If two variables are provided, it will draw a scatter plot

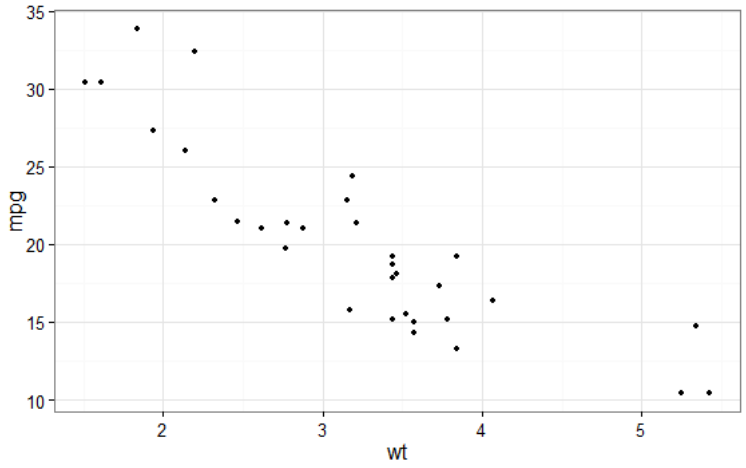
```
> # The following will draw a basic histogram with bars every 5 mph
> qplot(x = mpg, data = mtcars, binwidth = 5)
```



MANGOSOLUTIONS
data analysis that delivers

A Quick Introduction to the "Quick Plot"

```
> # The following will draw a basic histogram with bars every 5 mph
> qplot(x = wt, y = mpg, data = mtcars)
```



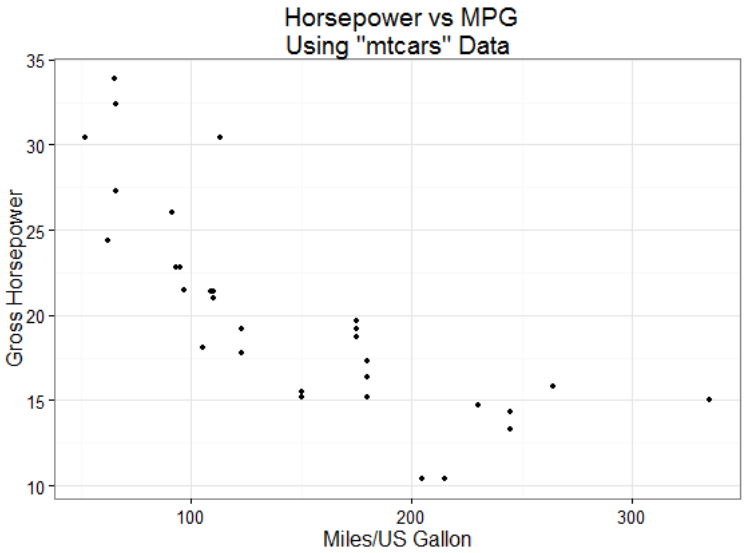
MANGOSOLUTIONS
data analysis that delivers

Titles and Axis Labels

- With `qplot` labelling is similar to the base graphics package
- We can add a main title to our plot using the `main` argument
- Arguments `xlab` and `ylab` control the axis labels for the x and y axes respectively

Titles and Axis Labels

```
> qplot(hp, mpg, data = mtcars,
+       main = ' Horsepower vs MPG\nUsing \"mtcars\" Data',
+       xlab = ' Miles/US Gallon ', ylab = ' Gross Horsepower ')
```



Note: A main title, x and y-axis labels can also be added as layers using the `ggtitle`, `xlab` and `ylab` functions respectively

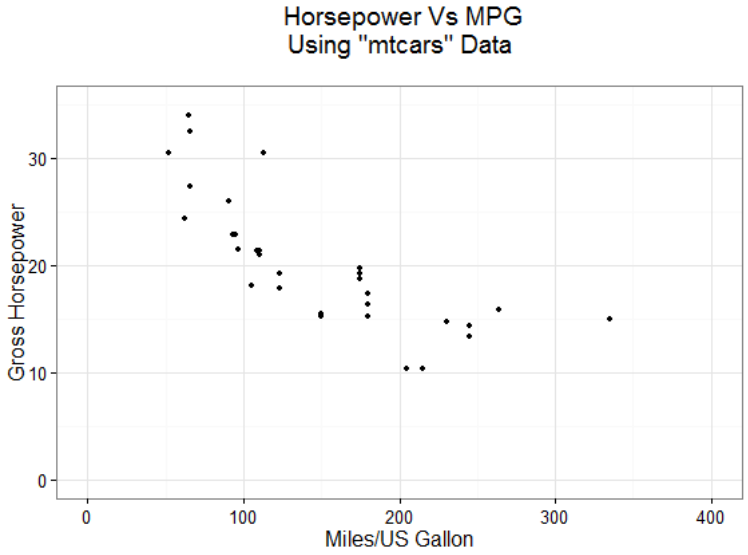
MANGOSOLUTIONS
data analysis that delivers

Axis Limits

- The `xlim` and `ylim` arguments to the plot functions allow users to control the X and Y axis limits
- These arguments must be provided with a vector of length 2

Axis Limits

```
> xRange <- c( 0, 400)  # A 2-element numeric vector
> yRange <- c( 0, 35)
> qplot (hp, mpg, data = mtcars,
+        main = ' Horsepower Vs MPG\nUsing \"mtcars\" Data\n',
+        xlab = ' Miles/US Gallon ', ylab = ' Gross Horsepower ',
+        xlim = xRange, ylim = yRange)
```



Note: The layering functions `xlim` and `ylim` can also be used to control axis limits

MANGOSOLUTIONS
data analysis that delivers

Exercises

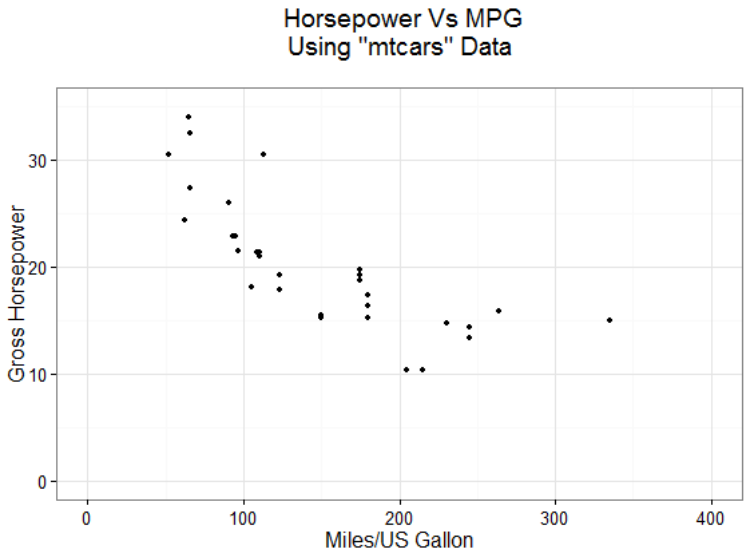
1. Create a plot of `Excess` against `Month` from `tubeData`. Ensure that the plot has appropriate titles and axis labels
2. Ensure that the `Excess` axis ranges from 0 to 25
3. Create a histogram of `TOTAL` from `tubeData`
4. Write the above plots to a single pdf file

Layering

- In the previous examples options were added as function arguments, eg `xlim=`
- Using the ggplot2 philosophy we can also add these options as "layers"
- Layers are 'added' using the + symbol

Layering

```
> qplot (hp, mpg, data = mtcars) +
+   ggtitle(' Horsepower Vs MPG\nUsing \"mtcars\" Data\n') +
+   xlab('Miles/US Gallon') +
+   ylab('Gross Horsepower') +
+   xlim(xRange) +
+   ylim(yRange)
```



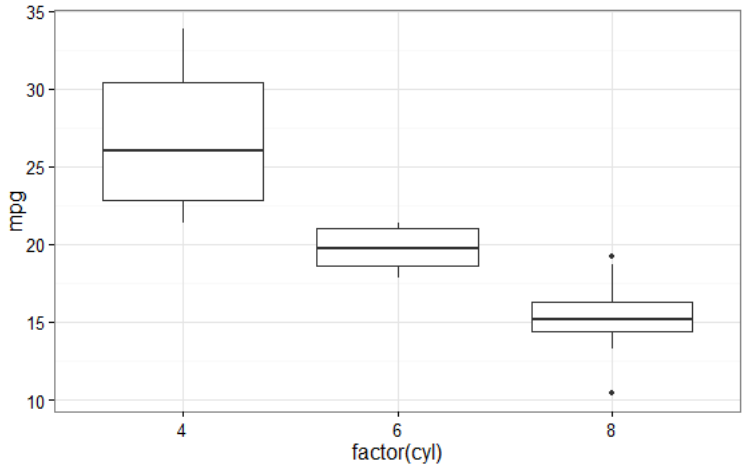
- Later we will see how layers add the flexibility to ggplot2

MANGOSOLUTIONS
data analysis that delivers

Plot Types - aka The "Geoms"

- The concept of "geoms" negates the need for multiple graphic functions
  - *Eg "histogram" or "bwplot" that are used in lattice to draw histograms and boxplots respectively*
- Within `qplot` we simply add an argument `geom` which defines how the data are to be displayed

```
> qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot")
```



Note: the use of factor in the previous example. It is important to consider the type (class) of data when using ggplot2.

MANGOSOLUTIONS
data analysis that delivers

Adding New 'geom' Layers

- When we specify `geom =` within the qplot function, we are in fact calling out to one of many `geom` functions that tell R how to display the graphic.
- All geometric functions in ggplot2 have a `geom_` prefix
- The code below finds all of the geom functions within the ggplot2 package

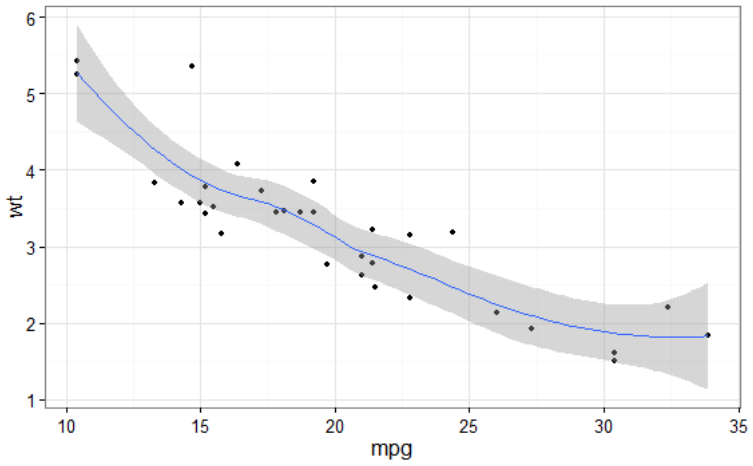
```
> grep("^geom", objects("package:ggplot2"), value = T)
```

```
 [1] "geom_abline"     "geom_area"       "geom_bar"
 [4] "geom_bin2d"      "geom_blank"      "geom_boxplot"
 [7] "geom_contour"    "geom_crossbar"   "geom_density"
[10] "geom_density2d"  "geom_dotplot"    "geom_errorbar"
[13] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"
[16] "geom_histogram"  "geom_hline"      "geom_jitter"
[19] "geom_line"       "geom_linerange"  "geom_map"
[22] "geom_path"       "geom_point"      "geom_pointrange"
[25] "geom_polygon"    "geom_quantile"   "geom_raster"
[28] "geom_rect"       "geom_ribbon"     "geom_rug"
[31] "geom_segment"    "geom_smooth"     "geom_step"
[34] "geom_text"       "geom_tile"       "geom_violin"
[37] "geom_vline"
```

Adding New "geom" Layers

- We can call any of these geom functions directly in order to add layers to our plot
- We do this using the + argument.

```
> qplot(mpg, wt, data = mtcars) + geom_smooth(method = "loess") # See ?geom_smooth for detai
```



MANGOSOLUTIONS
data analysis that delivers

Exercises

1. Create a boxplot of `TOTAL` by (against) `Type` from `tubeData`
2. Generate some random numbers using `rnorm` and draw a simple `density` plot to display the data
3. Create a plot of `Excess` against `Month` from `pkData`
4. Add a smoothing line to the plot

Extension Question:

5. Change the smoothing function in the previous exercise to a linear ("lm") smoother (not statistically sensible) and remove the error bars.

Aesthetics and Groups

- Perhaps the primary advantage of using ggplot2 over the standard R graphics, or lattice, is the ease of which aesthetics can be used in order to highlight features of our data
- However, for the ggplot2 newcomer, it is also one of the trickiest areas to get right!

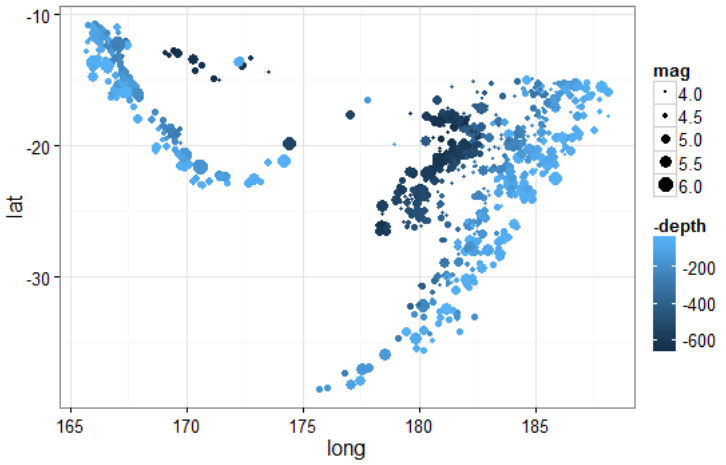
Note: The term "aesthetics" has a specific meaning in ggplot2 (later). Here we refer to colour, shape, size, etc.

- The ggplot2 package allows for both the traditional aesthetics names (e.g. col, pch, cex) and a more user-friendly naming (e.g. colour, shape, size)

Aesthetics

- Unlike the standard R graphics and lattice, we do not specify aesthetic attributes via vectors
- Instead we refer to a *variable* in the data and let ggplot do the hard work

```
> qplot(long, lat, data = quakes, size = mag, col = -depth)
```

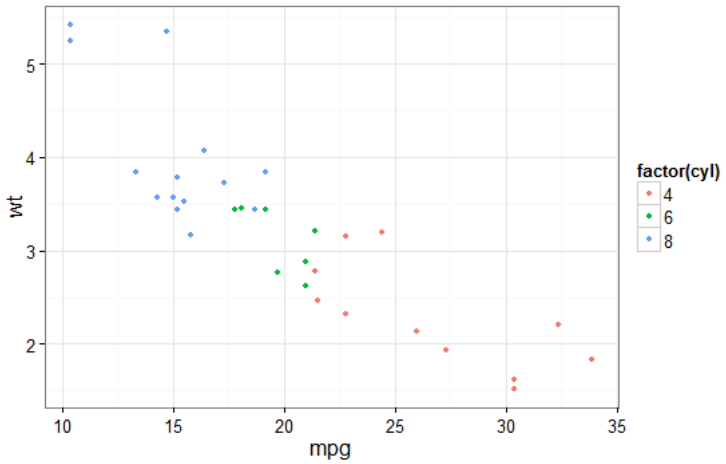


MANGOSOLUTIONS
data analysis that delivers

Factors

- The class of a variable is very important when using ggplot2
- If we have numeric data representing different categories, it is important that we convert these variables to factors
- If we do not, then a natural ordering is implied by the default aesthetics

```
> qplot(mpg, wt, data = mtcars, colour = factor(cyl))
```



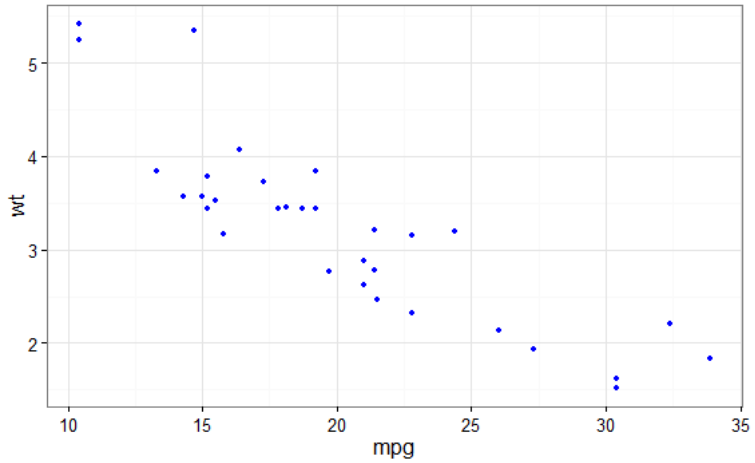
Warning: Attempting to vary the plotting shape by a numeric variable will cause an error!

MANGOSOLUTIONS
data analysis that delivers

Traditional Specification of Aesthetics

- ggplot2 has been designed to allow variables to be used with aesthetics
- We therefore have to use a special function `I()` in order to allow for a more traditional specification of aesthetics

```
> qplot(mpg, wt, data = mtcars, colour = I("blue"))
```



Warning: Attempting to vary the aesthetics manually, without using the I() function can yield some very unexpected results! Try the following line of code in your own time:

```
qplot(mpg, wt, data = mtcars, colour = "blue")
```

MANGOSOLUTIONS
data analysis that delivers

Exercises

1. Create a scatter plot of `Excess` against `Month` using `tubeData`. Use a different colour to distinguish between males and females and a different plotting symbol dependent upon whether the subject smokes or not.
2. Increase the size of the points on the previous plot to twice the default size.

Advanced Control of Aesthetics

- We can add "scaling" layers to a plot in order to control how aesthetics such as colour are used in the plot
- The scale functions begin with the prefix `scale_`

```
> grep("^scale", objects("package:ggplot2"), value = TRUE)
```

```
 [1] "scale_alpha"            "scale_alpha_continuous"
 [3] "scale_alpha_discrete"   "scale_alpha_identity"
 [5] "scale_alpha_manual"     "scale_area"
 [7] "scale_color_brewer"     "scale_color_continuous"
 [9] "scale_color_discrete"   "scale_color_gradient"
```

```
...
```

Note: ggplot2 allows for both UK and US spellings of 'colo[u]r'

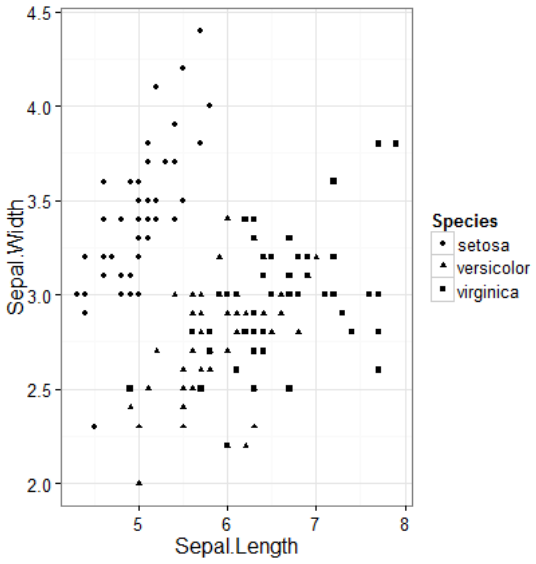
Advanced Control of Aesthetics

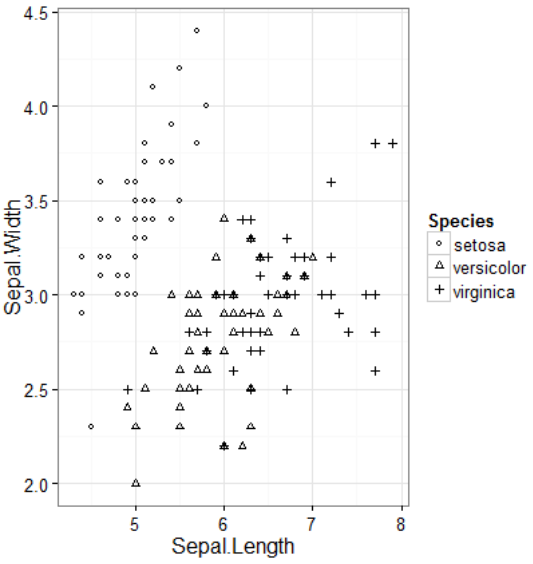
- The `scale_` function chosen *must* match the data type
- For example, if we vary colour by a continuous variable then we will need a colour scale function that works with continuous data, for example `scale_colour_continuous`
- In addition to the more obvious 'discrete' and 'continuous' scales, there are a number of other useful aesthetic scales available in ggplot2
- For example, `scale_colour_gradientn` creates a continuous colour through 'n' colours
- The `manual` suffix allows us to be specific about the aesthetics we use for discrete variables

MANGOSOLUTIONS
data analysis that delivers

Advanced Control of Aesthetics

```
> irisPlot <- qplot(Sepal.Length, Sepal.Width, data = iris, shape = Species)
> irisPlot
> irisPlot + scale_shape_manual(values = 1:3)
```





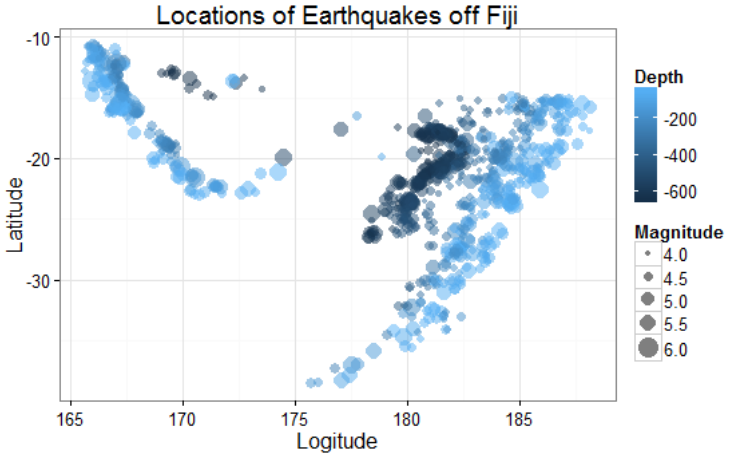
MANGOSOLUTIONS
data analysis that delivers

Legend Control

- Another advantage of ggplot2 is the auto-generation of the legend
- However this is another area that can be confusing to the ggplot2 newcomer
- The legend is generated when an aesthetic is used
- The same `scale_` functions that allow us to control the aesthetics, also allow us control of the legend
- The first argument to the aesthetic scale functions controls the title that appears in the legend

Legend Control

```
> fiji <- qplot(long, lat, data = quakes, size = mag, col = -depth,
+     alpha = I(0.5), xlab = "Logitude", ylab = "Latitude",
+     main = "Locations of Earthquakes off Fiji")
>
> fiji + scale_colour_continuous("Depth") +
+     scale_size_continuous("Magnitude", range = c(2,8))
```

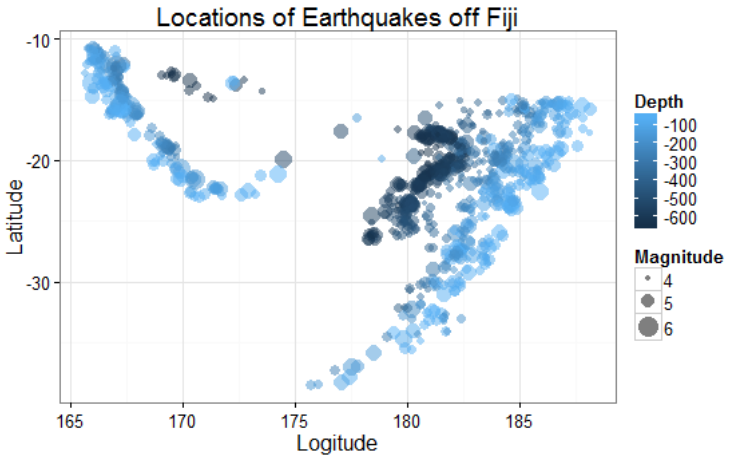


MANGOSOLUTIONS
data analysis that delivers

Legend Control

- We can also control how continuous variables are displayed within the legend

```
> fiji +
+     scale_colour_continuous("Depth", breaks = seq(0, -600, -100)) +
+     scale_size_continuous("Magnitude", range = c(2,8), breaks = 4:6)
```



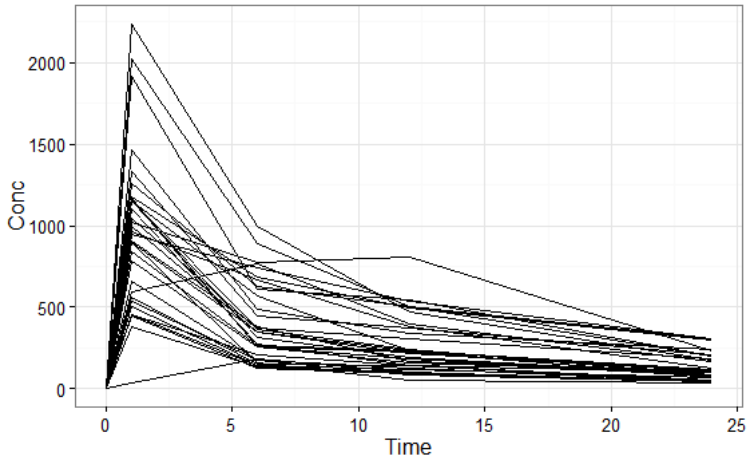
Note: We can manually control the legend text labels using the `labels` argument. If all we wish to control is the labelling of the aesthetics within the legend, we can use the `labs` function.

MANGOSOLUTIONS
data analysis that delivers

Grouping

- For certain types of plot, for example 'line' plots, we can also specify an inherent grouping
- Grouping can be used when we wish to separate out the levels of a factor but have no interest in comparing between these levels (e.g. using colour)

```
> # The following draws a separate line for each Subject
> qplot(Time, Conc, data = pkData, geom = "line", group = Subject)
```



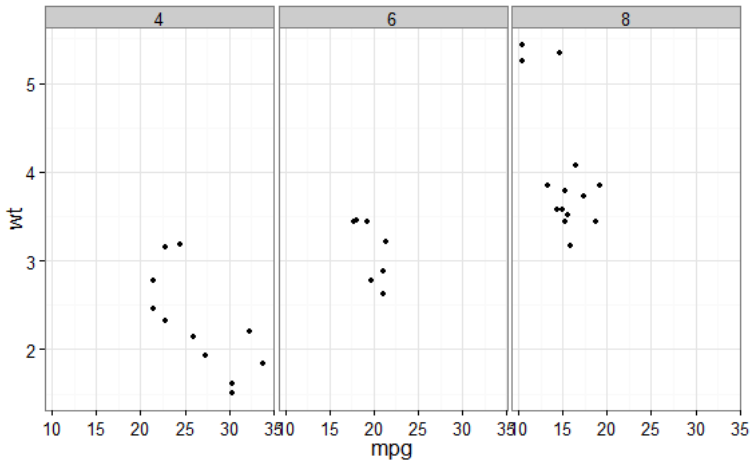
MANGOSOLUTIONS
data analysis that delivers

Panelling, aka "Faceting"

- ggplot2 provides a simple mechanism for panelling by a variable(s)
- In ggplot2, panelling is known as "faceting"
- We can either add a `facets` argument to `qplot` or add a separate layer using either `facet_grid` or `facet_wrap`
- The `facet_grid` function stacks the panels vertically or horizontally by the levels of a factor
- `facet_wrap` fills the available area much like lattice
- Faceting requires a formula of the form [rows] ~ [cols], where "rows" and "cols" can be replaced by either a variable name
- A "." can be used if panelling of one of rows or columns is not required

Panelling, aka "Faceting"

```
> qplot(mpg, wt, data = mtcars, facets = . ~ cyl)
> # Equivalently:
> # qplot(mpg, wt, data = mtcars) + facet_grid(. ~ cyl)
```



Note: If using the `facets` argument to `qplot`, we can omit the "." to invoke `facet_wrap`

MANGOSOLUTIONS
data analysis that delivers

Exercises

1. Draw a "line" plot of `Excess` against `Month` from `tubeData` and colour by `Type`. Ensure that the plot elements are labelled appropriately.
2. Change the previous plot so that each level of `Type` is instead represented by a separate panel
3. Draw a "line" plot of `Excess` against `Month` from `tubeData`. Ensure each `Line` is drawn in a different panel and vary the line type by `Type`
4. Create a plot of `Month` against `Excess` from `tubeData`. Panel (facet) the plot to show levels of `Line` in the rows and `Type` in the columns.

MANGOSOLUTIONS
data analysis that delivers

Advanced Control - the `ggplot` function

- In the examples thus far we have first created a basic ggplot2 object using qplot and added to this using a variety of layers
- The ggplot2 function makes no assumptions about the plot type or even the coordinate system
- It simply creates an empty ggplot2 graphical option onto which we can add layers of information
- This can be very useful when highly customised graphics are required

Warning Calling `ggplot()` without any additional layers will produce an error!

MANGOSOLUTIONS
data analysis that delivers

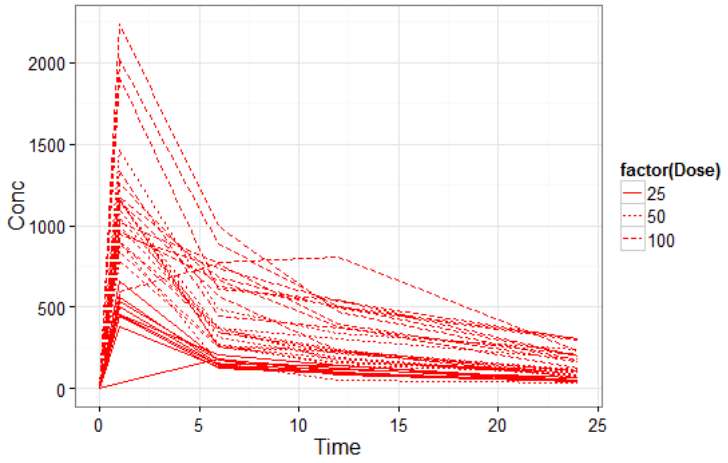
Advanced Control - the `aes` function

- The aes function is required to describe "aesthetics"
- In ggplot2 terminology, this refers not only to colour, shape or size, but to the x and y variables and a number of other graphical attributes
- For the ggplot2 newcomer, the aes function is perhaps one of the most confusing aspects of the package!
- However the function is actually very straightforward to use

The aes function

- If we wish to refer to *any variable* in the dataset via a function argument, whether this is to assign it to the x axis, or colour by the variable, we *must* 'wrap' the argument in a call to `aes`
- The dataset itself does not need to be contained within the call to `aes`

```
> # Note how any argument which references a variable is within the aes() call
> ggplot() +
+   geom_line(data = pkData,
+             aes(x = Time, y = Conc, group = Subject, linetype = factor(Dose)), # Args are
+             colour = "red")                                                    # Arg is no
```



MANGOSOLUTIONS
data analysis that delivers

Using Multiple Datasets

- The advantage of using `ggplot` is that each layer may refer to a different dataset

```
> pkSummary<- data.frame(TimeBoundary = c(0, 1, 6, 12, 24, 24, 12, 6, 1, 0),
>                        ConcBoundary = c(0.00, 35.87, 129.31, 51.43, 35.58,
>                                         310.79, 806.82, 995.85, 2240.00, 0.00))
>
> # Create an empty ggplot object
> pkGG <- ggplot()
>
> # Add the summary data to our object
> pkGG <- pkGG +
>   geom_polygon(data = pkSummary,
>                aes(x = TimeBoundary, y = ConcBoundary),
>                alpha = .2, fill = "red")
>
> # Add lines for each subject and panel by Dose
> pkGG <- pkGG +
>   geom_line(data = pkData,
>             aes(x = Time, y = Conc, group = Subject)
>    )
```

MANGOSOLUTIONS
data analysis that delivers

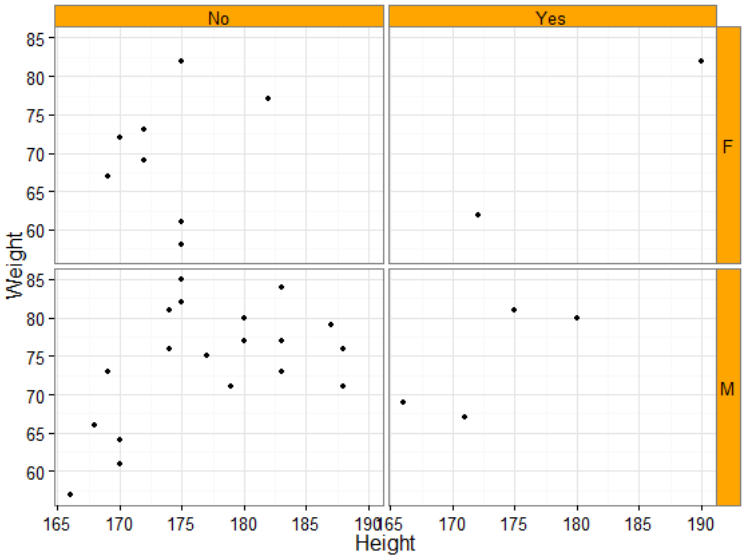
Themes

- The default ggplot2 theme is a grey background
- It is easy to modify themes using `theme_` layers
- For this presentation, `theme_bw(base_size = 16)` was used
- Themes can be set using the `theme_set` function
- Themes can be queried using the `theme_get` function
- For an individual plot, individual elements may be modified using a `theme` function layer
- Theme elements can be controlled using `element_` functions

MANGOSOLUTIONS
data analysis that delivers

Themes

```
> demoPlot <- qplot(Height, Weight, data = demoData)
> # Create a panelled plot:
> demoPlot <- demoPlot + facet_grid(Sex ~ Smokes)
> # Now rotate the text in the row strips
> demoPlot + theme(strip.text.y = element_text(angle = 0),
+                  strip.background = element_rect(fill = "orange", colour = "grey50"))
```



MANGOSOLUTIONS
data analysis that delivers

More on Themes

- The help file for `theme` is now fairly comprehensive
- However it is still useful to know which `element_` function to use
- `theme_get()$XXX` therefore comes in useful

```
> theme_get()$panel.background
```

```
List of 4
 $ fill    : chr "white"
 $ colour  : logi NA
 $ size    : NULL
 $ linetype: NULL
 - attr(*, "class")= chr [1:2] "element_rect" "element"
```

MANGOSOLUTIONS
data analysis that delivers

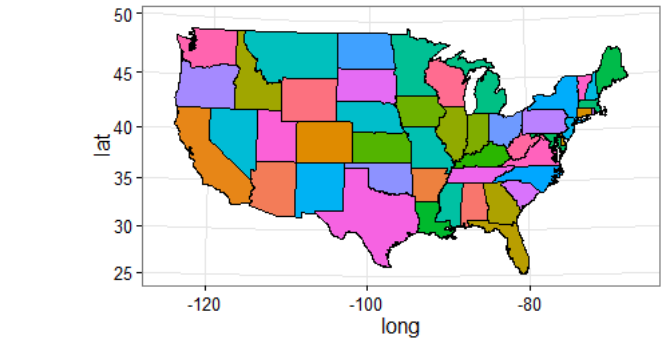
Further Advanced Control

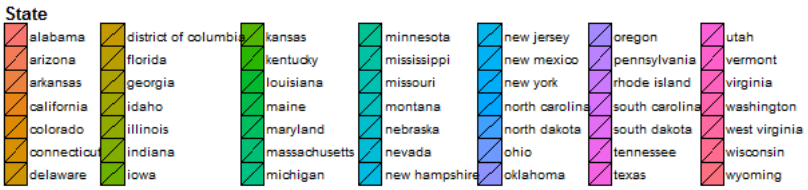
- Axes can be controlled via `scale_` function layers
- ggplot2 contains many functions for changing the coordinate system
  - *polar coordinates (`coord_polar`), map projections (`coord_map`), ...*
- A new function `guides` was recently introduced to allow users to modify the appearance of the legend

MANGOSOLUTIONS
data analysis that delivers

Further Advanced Control

```
> states <- map_data("state")
> usPlot <- qplot(long, lat, data = states, geom = "polygon", group = group,
+   fill = region, col = I("black"))
> # Split the legend into 3 columns and move to the bottom of the plot
> usPlot + theme(legend.position = "bottom") +
+    guides(fill = guide_legend(title = "State",
+                               nrow =7, title.position = "top",
+                               label.theme = element_text(size = 10, angle = 0))) +
+   coord_map("lagrange") # Change the map projection
```





MANGOSOLUTIONS
data analysis that delivers

Exercises

1. Draw a "line" plot of `Excess` against `Month` from `tubeData` and group and colour by `Line`. Ensure that the plot elements are labelled appropriately.
2. Update the colours to be the `Line` colours (e.g. Bakerloo = Brown, Central = Red, etc.)

More on ggplot2

- Hadley Wickham maintains his own website http://ggplot2.org/
- Mango run a 1-day ggplot2 training course
- Hadley Wickham is coming to EARL in September
- Speak to me afterwards

MANGOSOLUTIONS
data analysis that delivers

Thank You

- Questions?