Head to Head: Lattice vs ggplot2

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Head to Head: ggplot2 vs Lattice

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Aim

- To present R graphics users with enough information to make an informed choice as to which graphics package best meets their needs

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Agenda

- Why are we here?
- Introduction to Lattice
- Introduction to ggplot2
- The Challenge!
- Why and Why Not Lattice
- Why and Why Not ggplot2
- Conclusions

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Why are we here?

- Mango have traditionally used lattice for our software products, training, etc
- ggplot2 is increasingly popular in the community

- Rich likes Lattice
- Andy likes ggplot2

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Approach

- Demonstrate the common package features
  - Panelling
  - Grouping
  - Legends
  - Styling
  - Advanced control
- Create the same graphic in the two technologies and compare the code
- Discuss

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

The Data

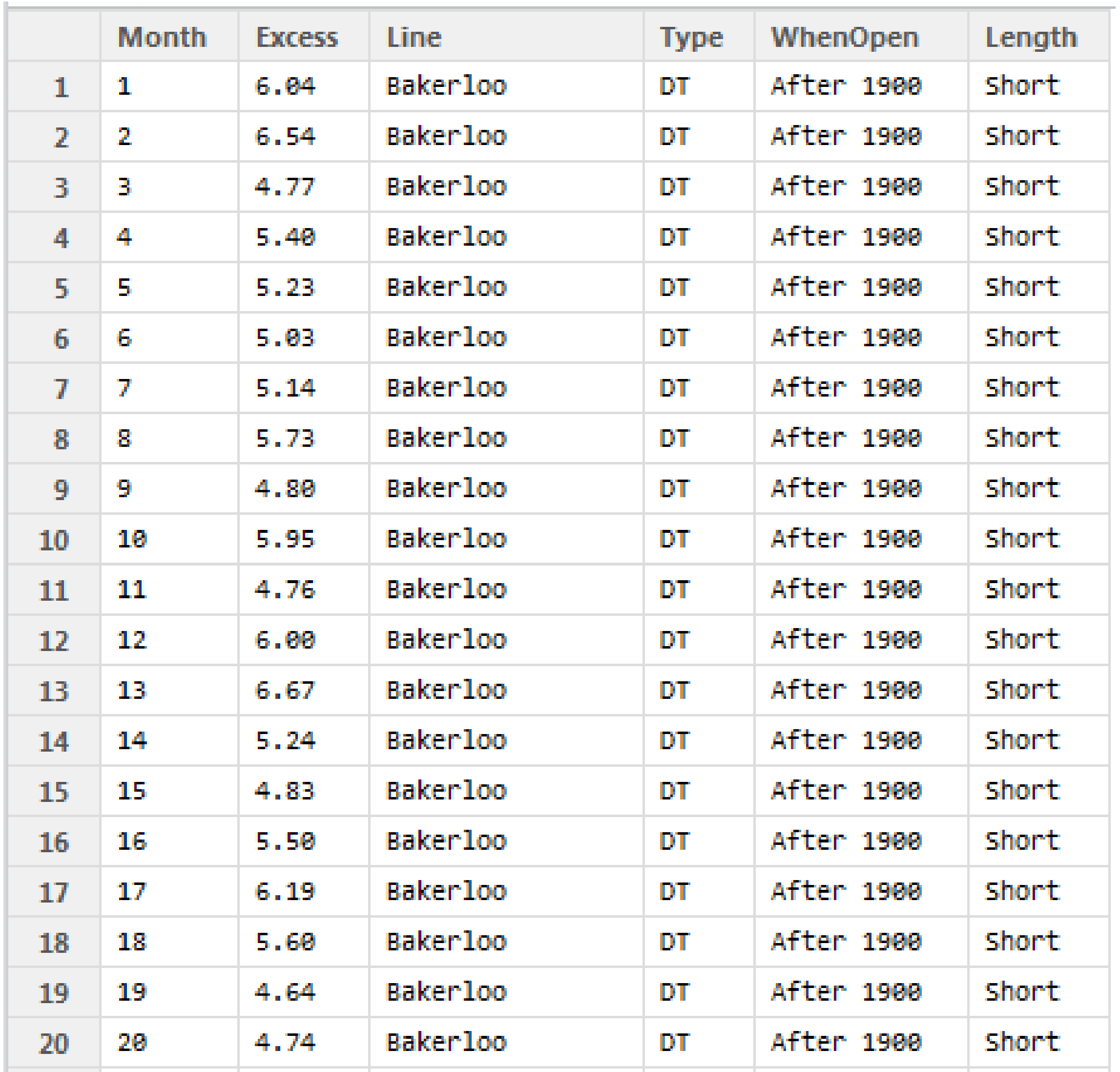
- Something sector independent
- London Tube Performance Data from the TFL website
- Excess Travel Hours by Line

  http://data.london.gov.uk/datastore/package/tube-network-performance-data

  http://en.wikipedia.org/wiki/London_Underground

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

The Data



	Month	Excess	Line	Type	WhenOpen	Length
1	1	6.04	Bakerloo	DT	After 1900	Short
2	2	6.54	Bakerloo	DT	After 1900	Short
3	3	4.77	Bakerloo	DT	After 1900	Short
4	4	5.40	Bakerloo	DT	After 1900	Short
5	5	5.23	Bakerloo	DT	After 1900	Short
6	6	5.03	Bakerloo	DT	After 1900	Short
7	7	5.14	Bakerloo	DT	After 1900	Short
8	8	5.73	Bakerloo	DT	After 1900	Short
9	9	4.80	Bakerloo	DT	After 1900	Short
10	10	5.95	Bakerloo	DT	After 1900	Short
11	11	4.76	Bakerloo	DT	After 1900	Short
12	12	6.00	Bakerloo	DT	After 1900	Short
13	13	6.67	Bakerloo	DT	After 1900	Short
14	14	5.24	Bakerloo	DT	After 1900	Short
15	15	4.83	Bakerloo	DT	After 1900	Short
16	16	5.50	Bakerloo	DT	After 1900	Short
17	17	6.19	Bakerloo	DT	After 1900	Short
18	18	5.60	Bakerloo	DT	After 1900	Short
19	19	4.64	Bakerloo	DT	After 1900	Short
20	20	4.74	Bakerloo	DT	After 1900	Short

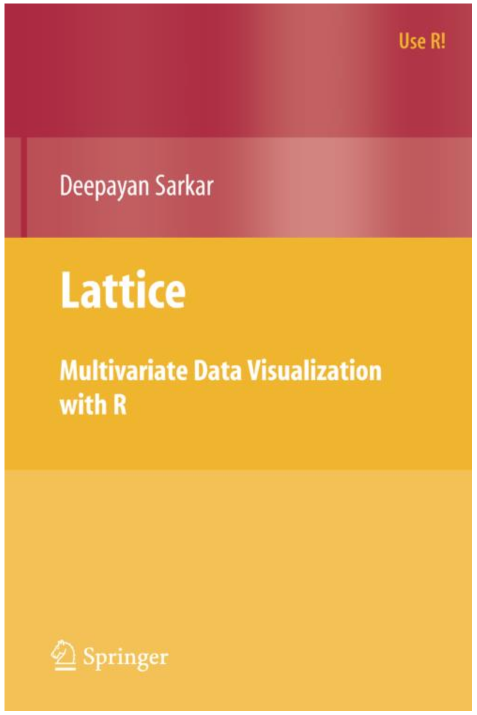
Lattice

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Overview of Lattice Graphics

- One of the graphic systems of R
- An implementation of the S+ “Trellis” Graphics
- Written by Deepayan Sarkar, Fred Hutchinson Cancer Research Center

```
> require(lattice)
Loading required package: lattice
```



List of Lattice Graphic Functions

Function	Description	Graph Type
`xyplot`	Scatter plot	Bivariate
`histogram`	Univariate histogram	Univariate
`densityplot`	Univariate density line plot	Univariate
`barchart`	Bar chart	Univariate
`bwplot`	Box and whisker plot	Bivariate
`qq`	Normal QQ plot	Univariate
`dotplot`	Label dot plot	Bivariate
`cloud`	3D scatter plot	3D
`wireframe`	3D surface plot	3D
`splom`	Scatter matrix plot	Data Frame
`parallel`	Multivariate parallel plot	Data Frame

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Key Function Arguments

Argument	Description
`x`	Plot definition, typically as a formula
`data`	The data frame used for the graphic
`subset`	Any subsets to be applied to the data
`panel`	Function used to draw data in each "panel"
`groups`	Grouping variable for the plot

Type of graph	Formula	Y axis	X axis	Z axis
Univariate	~ Y	Y	-	-
Bivariate	Y ~ X	Y	X	-
3D	Z ~ X*Y	Y	X	Z
Data Frame	~ Data	Data	-	-

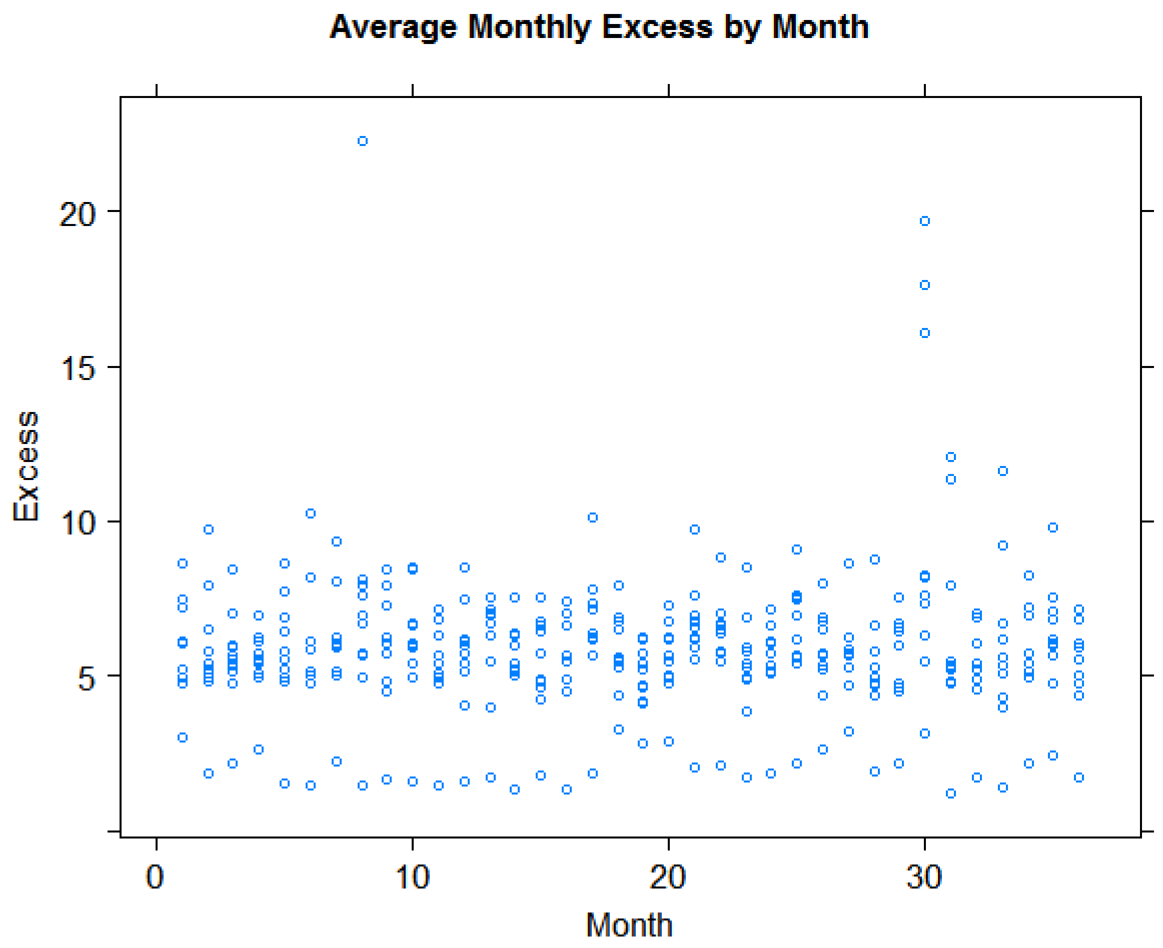
Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Building A Graphic

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

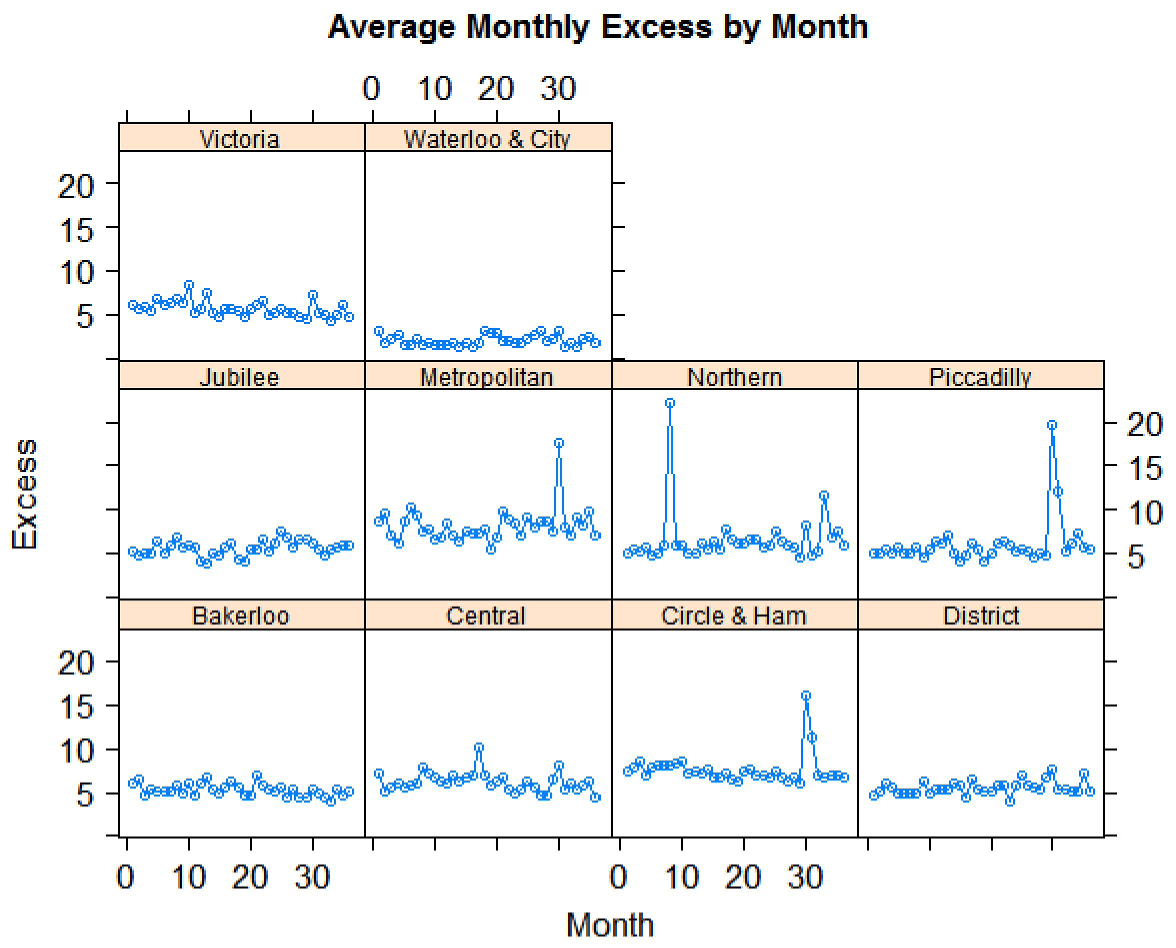
A Simple Scatter Plot

```
xyplot(Excess ~ Month, data = tubeData,
       main = "Average Monthly Excess")
```



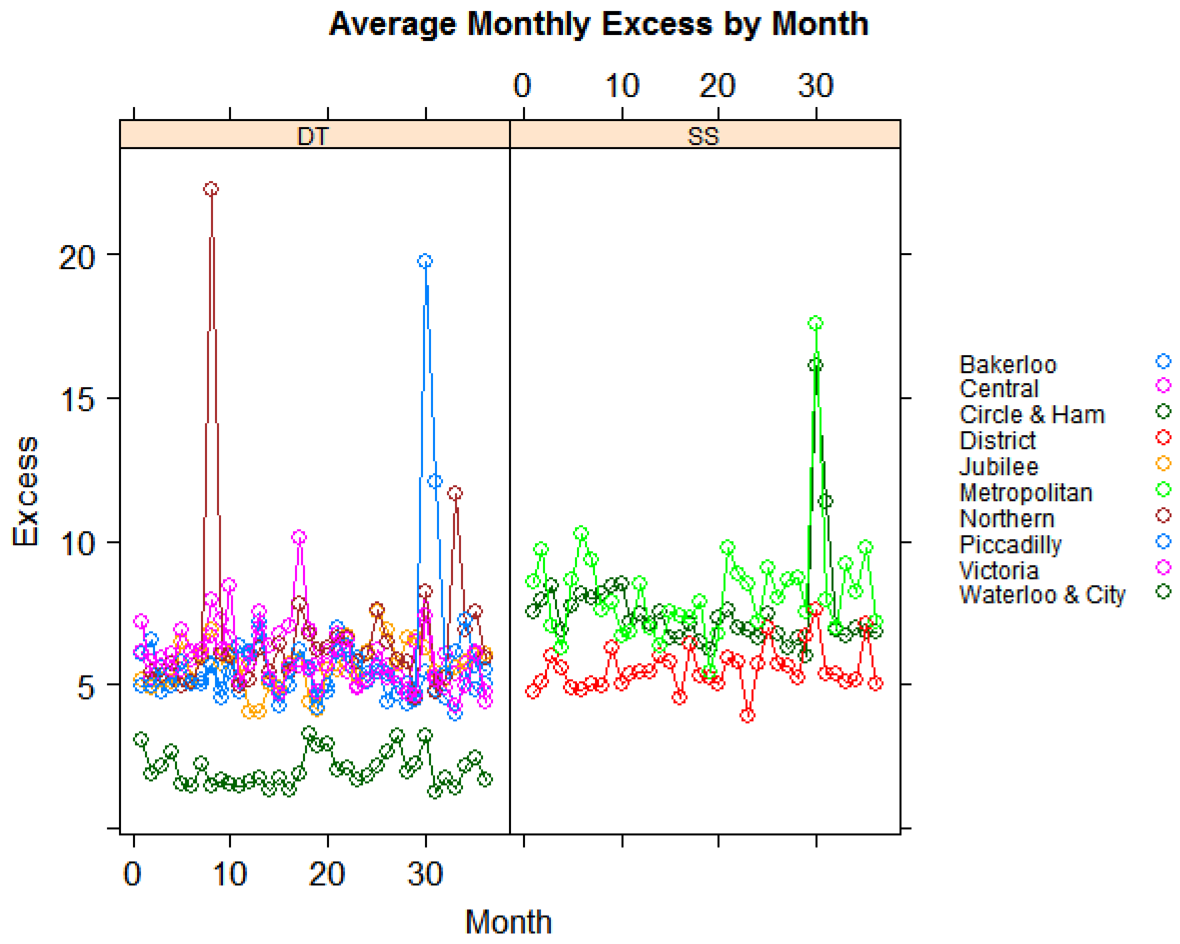
Panelling

```
xyplot(Excess ~ Month | Line, data = tubeData, type = "o",
       main = "Average Monthly Excess by Line")
```



Grouping

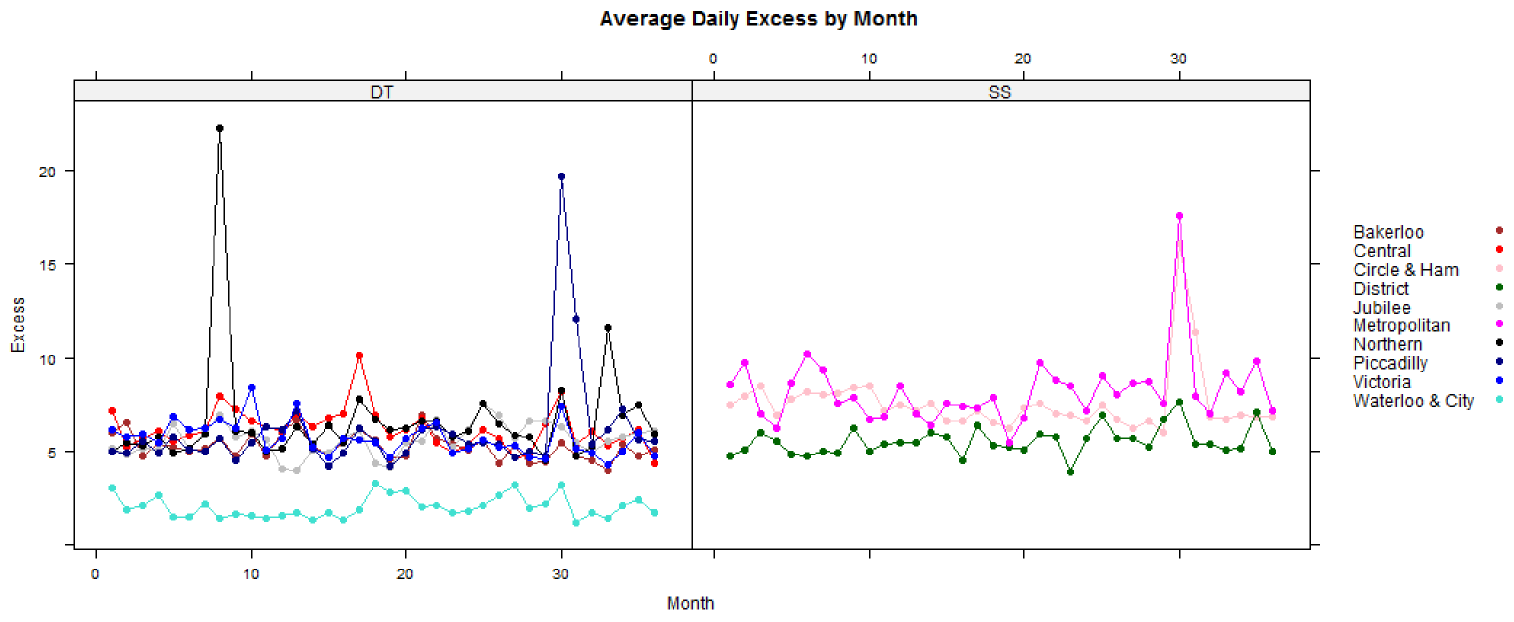
```
xyplot(Excess ~ Month | Type, data = tubeData, type = "o",
       groups = Line, main = "Average Monthly Excess by Line",
       auto.key = list(space = "right"))
```



Styling

```
lineCols <- c("brown", "red", "pink", "darkgreen", "grey",
              "magenta", "black", "navy", "blue", "turquoise")
myStyles <- standard.theme("pdf")  # Get a basic theme
myStyles$superpose.symbol$col <- lineCols
myStyles$superpose.line$col <-  lineCols
myStyles$superpose.symbol$pch <- 16
myStyles$strip.background$col <- c("grey95", "pink", "grey65")

xyplot(Excess ~ Month | Type, data = tubeData, type = "o",
       groups = Line, main = "Average Monthly Excess by Line\nSplit by Type",
       auto.key = list(space = "right"), par.settings = myStyles)
```



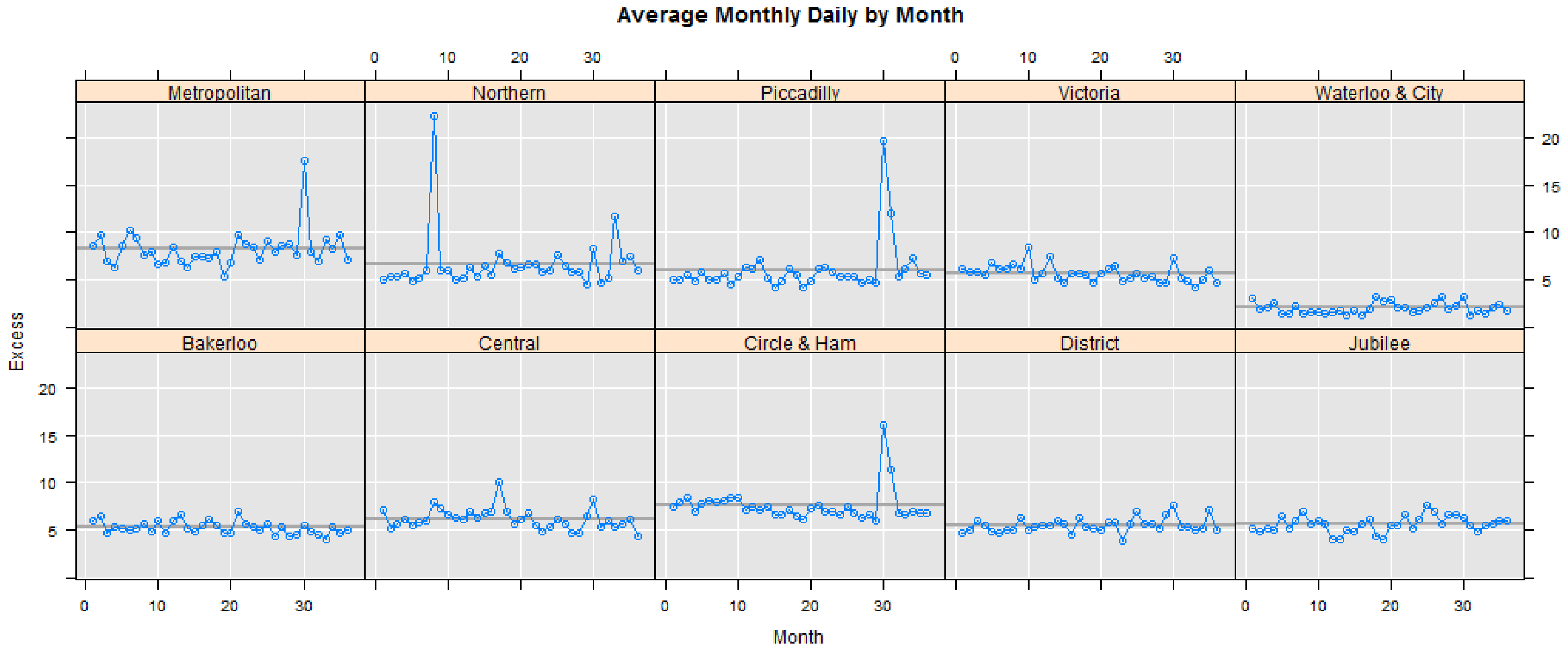
Manipulating Plot Structure

- You can control the exact plot created at 2 levels:
  - Panel: Plot for each plot “panel”
  - Panel.groups: Plot for each “group” of data
- Each input takes a function
- Panel.groups is called from “within” your panel function

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

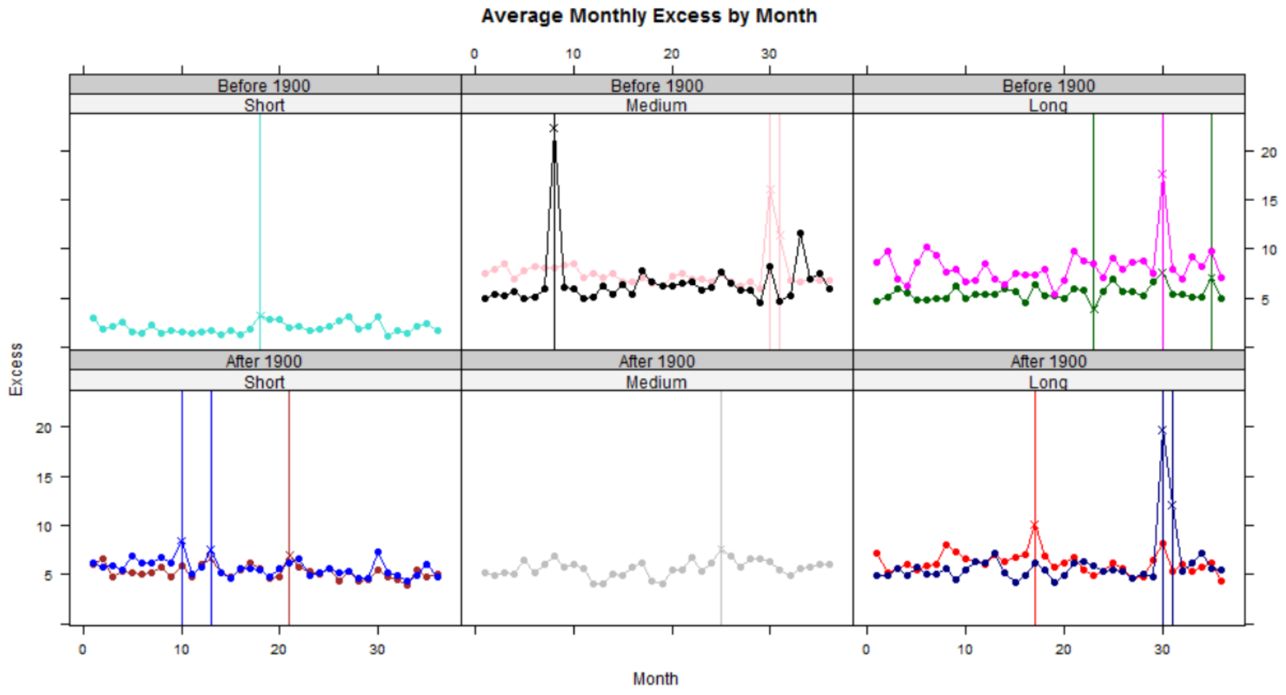
Panel Functions

```
xyplot(Excess ~ Month | Line,
       data = tubeData, type = "o",
       main = "Average Monthly Excess by Line\nSplit by Type",
       panel = function(x, y, ...) {
           panel.fill(col = "grey90")
           panel.grid(h = -1, v = -1, col = "white")
           panel.abline(h = mean(y), col = "darkgrey", lwd = 2)
           panel.xyplot(x, y, ...)
        })
```



The “panel.groups” Function

```
xyplot(Excess ~ Month | Length * WhenOpen, data = tubeData, groups = Line,
   main = "Average Monthly Excess by Month", panel = panel.superpose,
   panel.groups = function(x, y, col.symbol, ...) {
      theMean <- mean(y); theSd <- sd(y)
      theLower <- theMean - 2 * theSd; theUpper <- theMean + 2 * theSd
      isOut <- y > theUpper | y < theLower
      panel.xyplot(x, y, col = col.symbol, type = "o", pch = ifelse(isOut, 4, 16))
      if (any(isOut)) panel.abline(v = x[isOut], col = col.symbol)
   }, par.settings = myStyles)
```



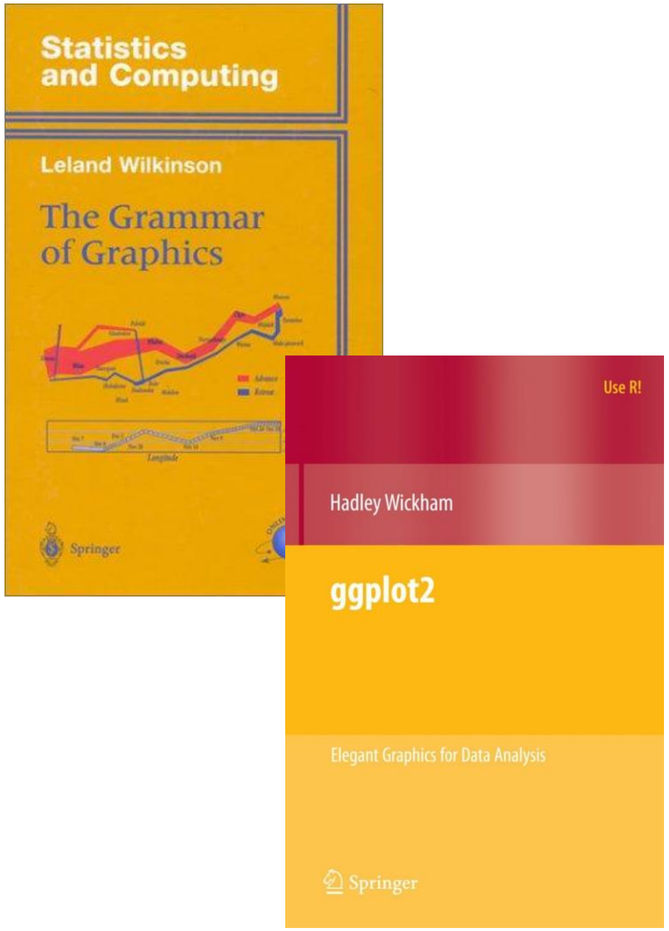
ggplot2

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

GGplot2 Graphics

- Graphical package created by Hadley Wickham
- Implements the ideas found in the book The Grammar of Graphics

```
> require(ggplot2)
Loading required package: ggplot2
```



ggplot2 Graphics

- Like lattice:
  - Plots are stored in objects
  - Graphs may be controlled with a formula syntax
  - It is easy to create “panelled” graphics
- Plots built by “layering” features
- Heavy use of “aesthetics” and “facets” (as per Wilkinson’s book)

Using ggplot2

- Two primary ways of creating a plot:
  - Create a “quick plot” using **`qplot`**
  - Create plot at a more granular level using **`ggplot`**
- We can use a mixture of the above approaches

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Using ggplot2

- We then modify this plot by adding “layers”:
  - New data
  - Scales mapping aesthetics to data
  - A geometric object
  - A statistical transformation
  - Position adjustments within the plot area
  - Faceting (panelling)
  - The coordinate systems itself

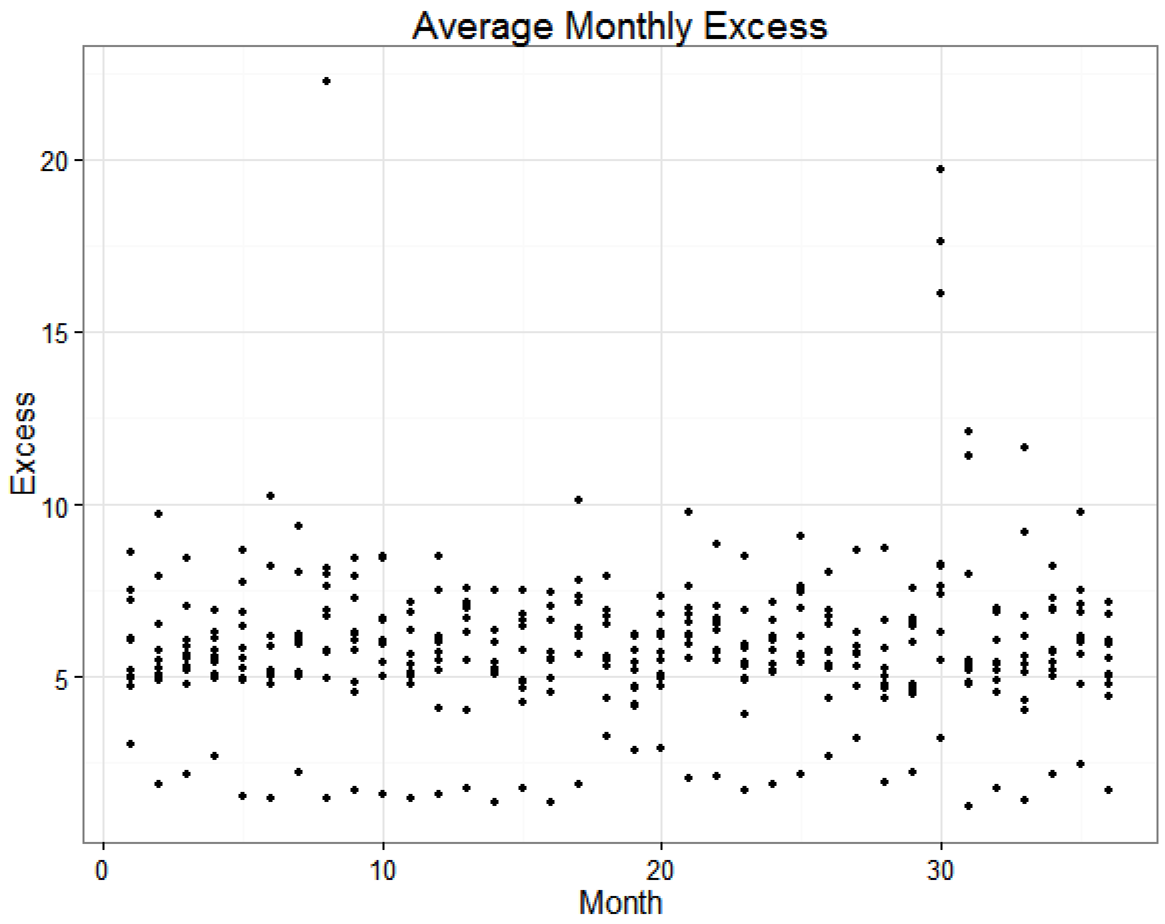
Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Building A Graphic

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

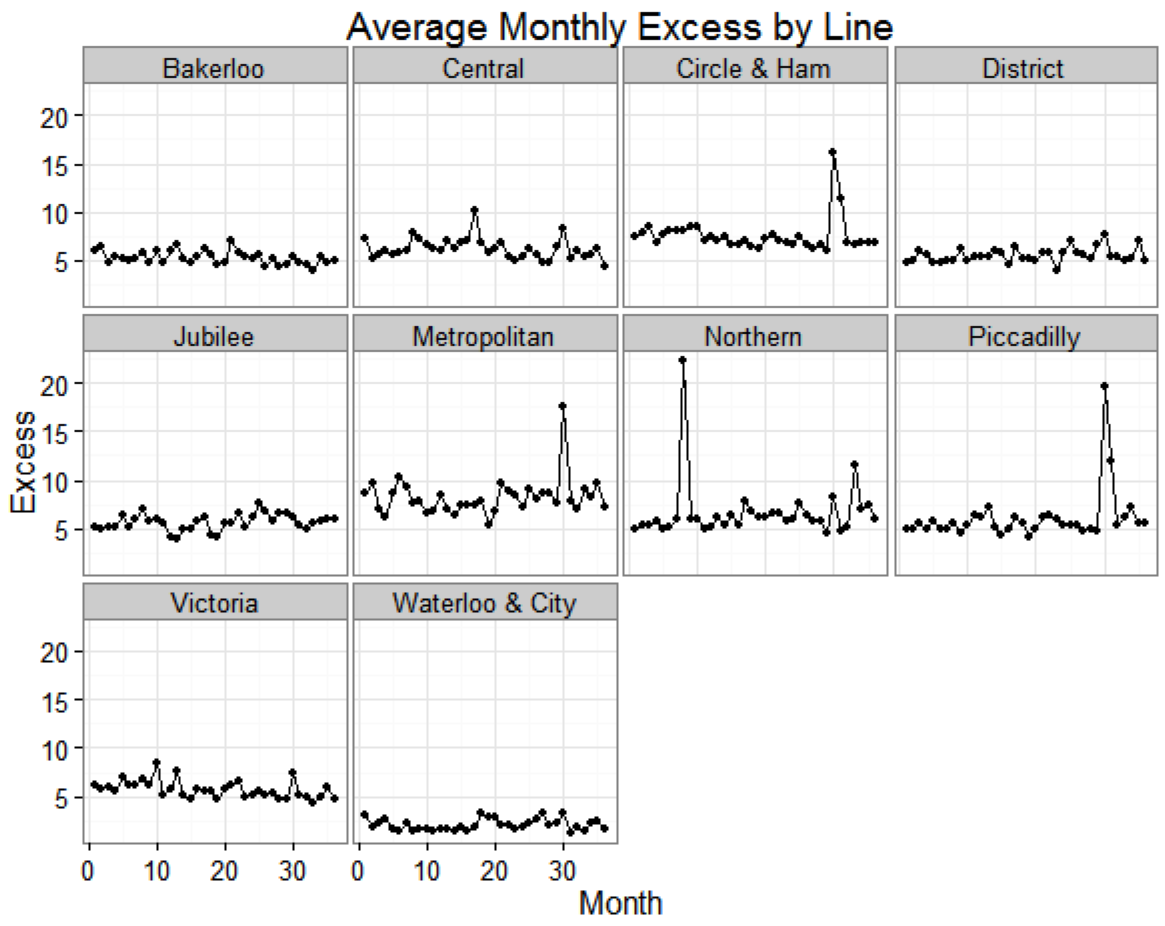
A Simple Scatter Plot

```
qplot(Month, Excess, data = tubeData,
      main = "Average Monthly Excess by Month")
```



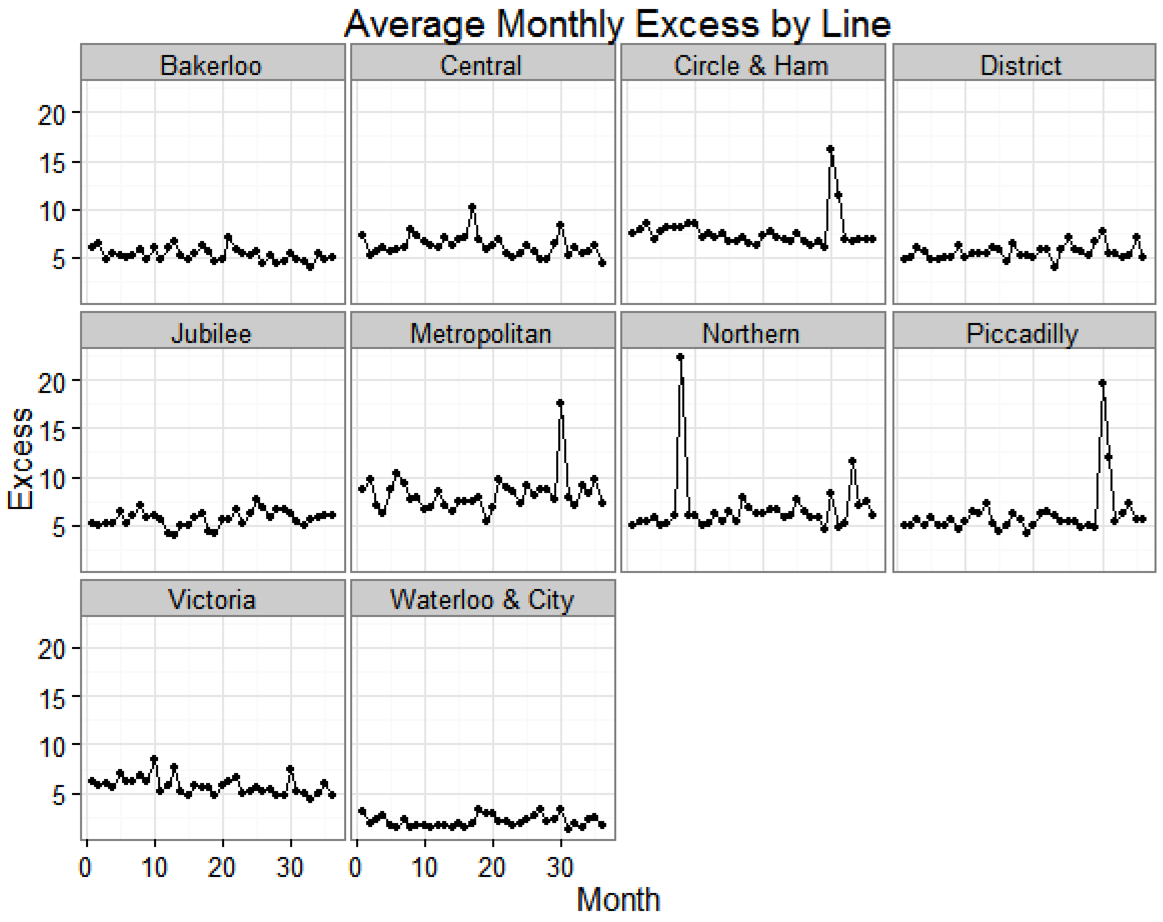
Panelling

```
qplot(Month, Excess, data = tubeData,
      facets = ~ Line, geom = c("point","line"),
      main = "Average Monthly Excess by Line")
```



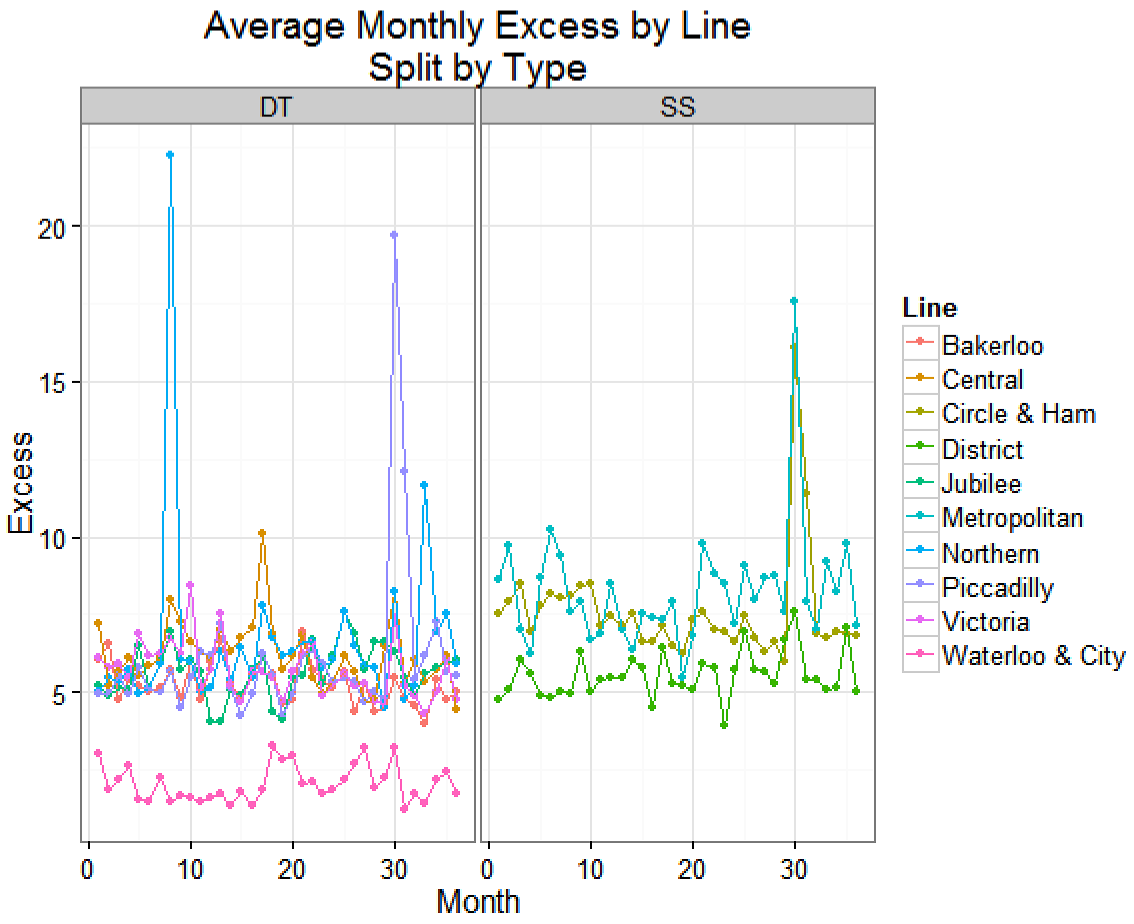
Panelling (Alternative)

```
gg <- qplot(Month, Excess, data = tubeData,
            geom = c("point", "line"),
            main = "Average Monthly Excess by Month")
gg + facet_wrap(~ Line)
```



Grouping

```
qplot(Month, Excess, facets = ~ Type, data = tubeData,
      geom = c("point","line"), colour = Line,
      main = "Average Monthly Excess by Line\nSplit by Type")
```



Styling

- Styling appears in many places in ggplot2
- The graphics shown so far have already been "styled" to some degree
- In-built themes control general page styling:

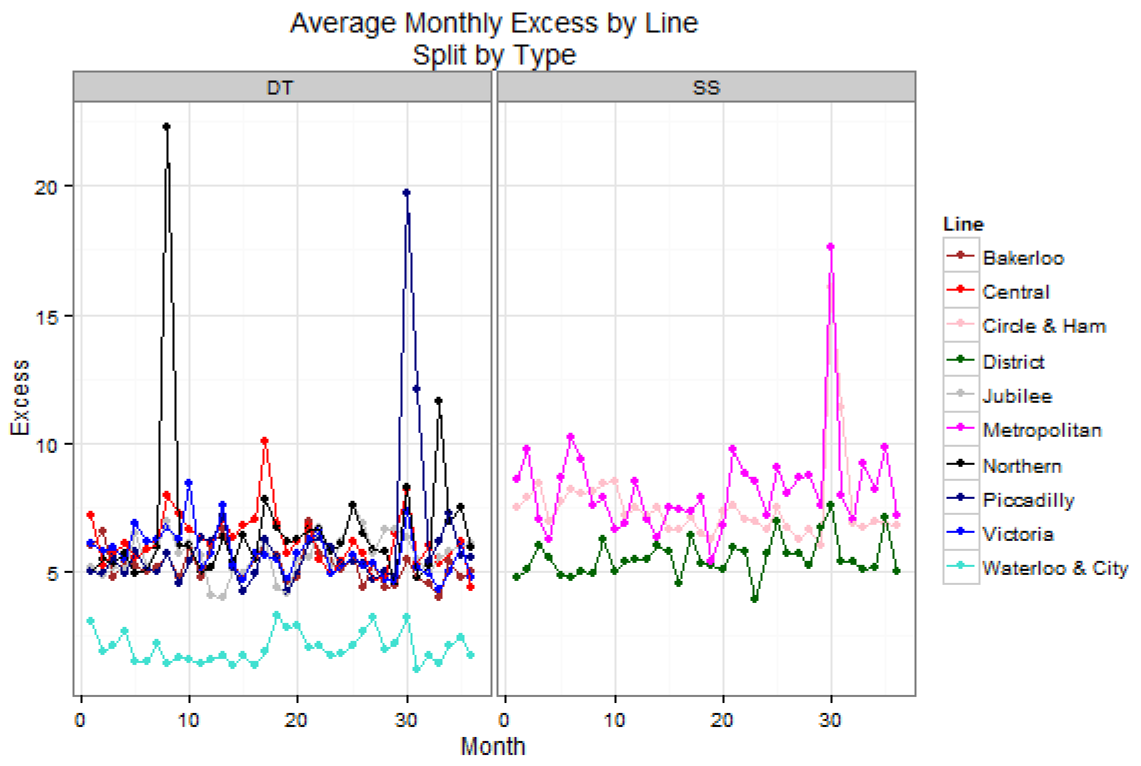
```
theme_set(theme_bw(base_size = 16))
theme_update(
  strip.background = element_rect(fill = c("grey95", "grey80", "grey65"))
)
```

- Plot styling is controlled by scale layers…

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Styling

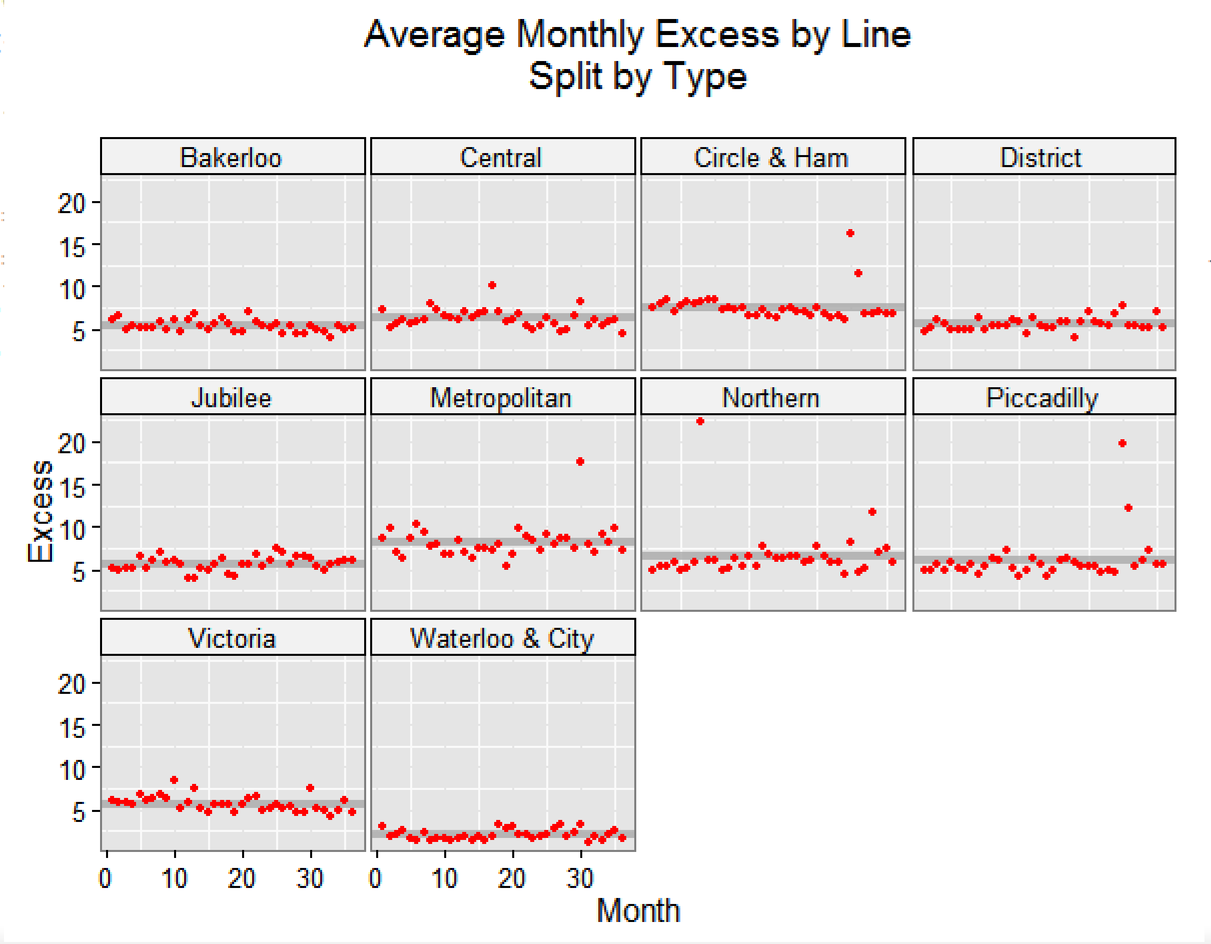
```
lineCols <- c("brown", "red", "pink", "darkgreen", "grey",
              "magenta", "black", "navy", "blue", "turquoise")
myPlot <- qplot(Month, Excess, facets = ~ Type, data = tubeData,
                geom = c("point", "line"), colour = Line,
                main = "Average Monthly Excess by Line\nSplit by Type")
myPlot + scale_colour_manual(values = lineCols)
```



Customisation

```
# Ablines ar
ablineData <

# 'Simple' p
qplot(Month,
       data =
       main =
  theme(pane
  geom_hline
```

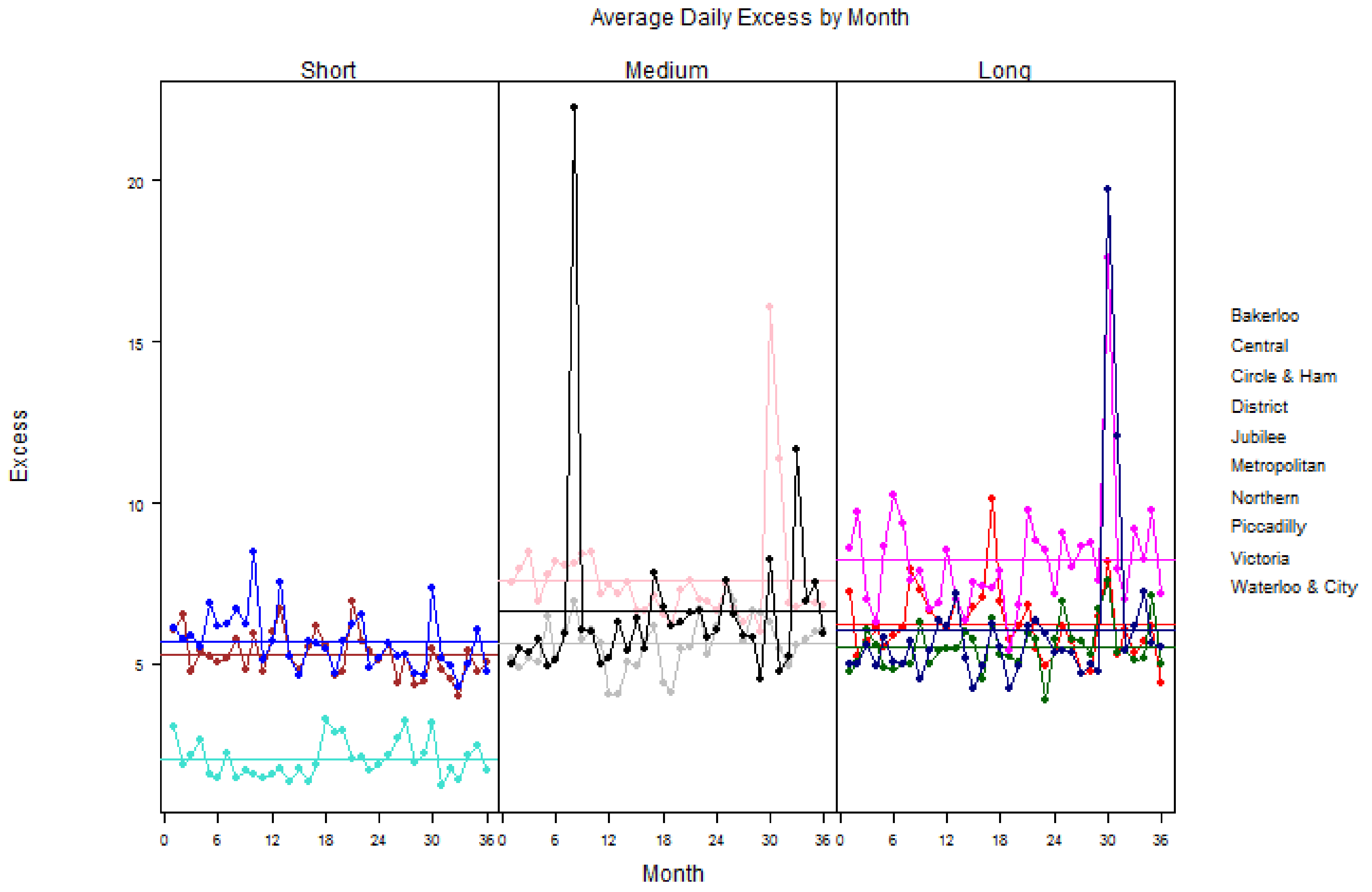


Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

The Challenge

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

The Challenge



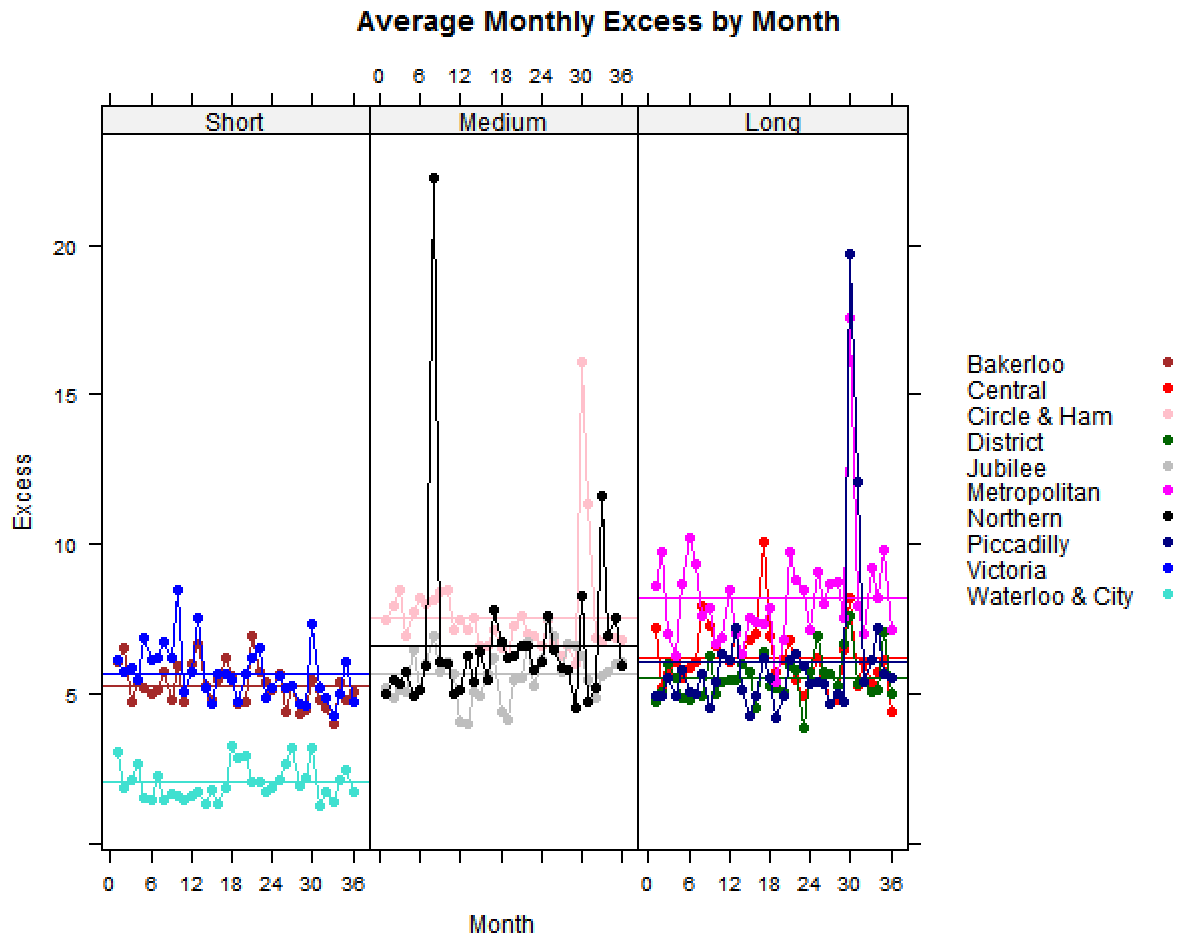
The Challenge: Lattice

```
xyplot(Excess ~ Month | Length,
       data = tubeData, type = "o", groups = Line,
       main = "Average Monthly Excess by Line\nSplit by Line Length\n",
       panel = panel.superpose,
       panel.groups = function(x, y, ...) {
         panel.abline(h = mean(y), ...)
         panel.xyplot(x, y, ...)
       },
       scales = list(x = list(at = 6*0:6)),
       par.settings = myStyles,
       auto.key = list(space = "right"),
       layout = c(3, 1))
```

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

The Challenge: Lattice

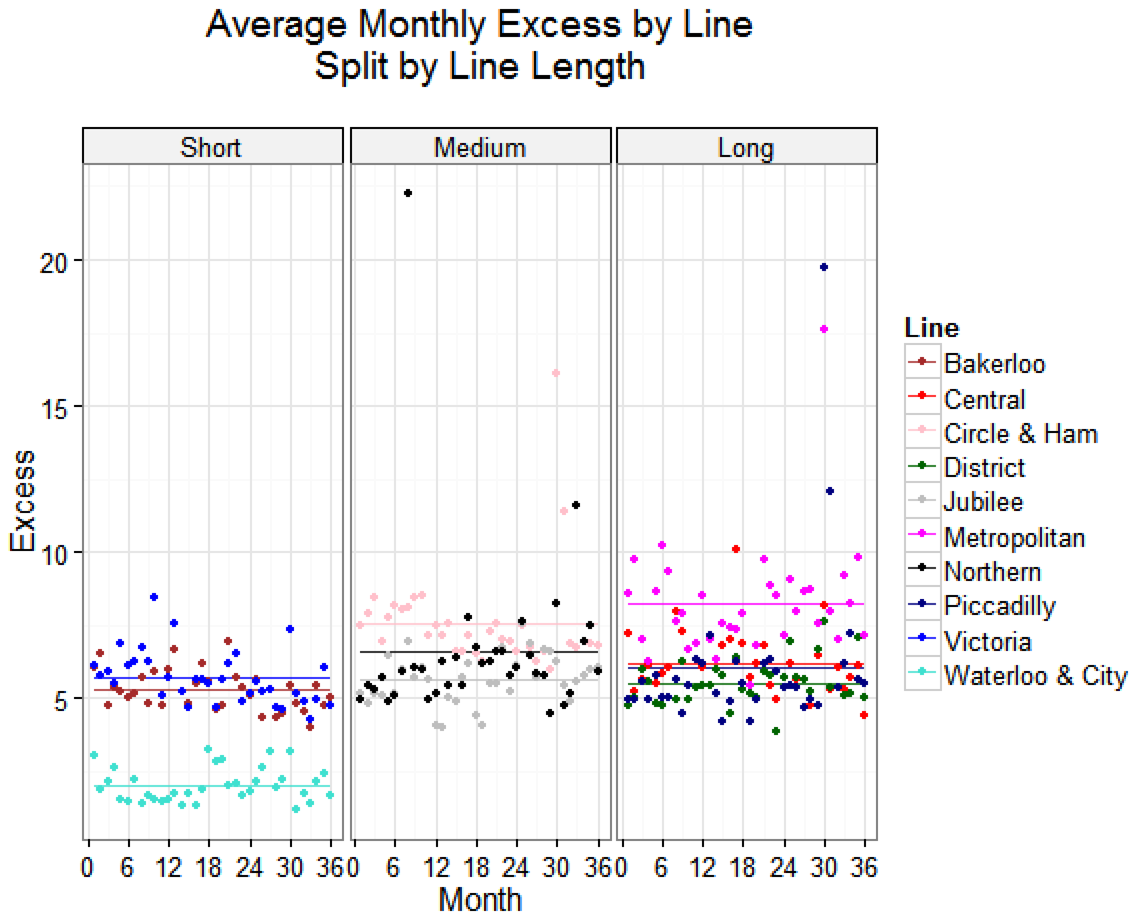
```
xyplot(Excess ~ Month | Length,
       data = tubeData, type = "o", groups = Line,
       main = "Average Monthly Excess by Line\nSplit by Line Length\n",
       panel = panel.superpose,
       panel.groups = function(x, y, ...) {
         panel.abline(h = me
         panel.xyplot(x, y,
       },
       scales = list(x = lis
       par.settings = myStyl
       auto.key = list(space
       layout = c(3, 1))
```



The Challenge: ggplot2

```
# Another ggplot2 data trick for ablines
tubeData$ablineValues <- ave(tubeData$Excess, tubeData$Line)

ggplot(data = tubeData, aes
  facet_grid(. ~ Length) +
  geom_line(aes(y = ablineV
  geom_point(aes(col = Line
  ggtitle("Average Monthly
  scale_colour_manual(value
  scale_x_continuous(breaks
```



Comparison

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Why Lattice

- Intuitive structure for controlled data at a group / subgroup level
- Achieve simple panelled graphics very quickly
- Well documented
- Extensions available (latticeExtra, nlme)
- A lot faster than ggplot2! ☺

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

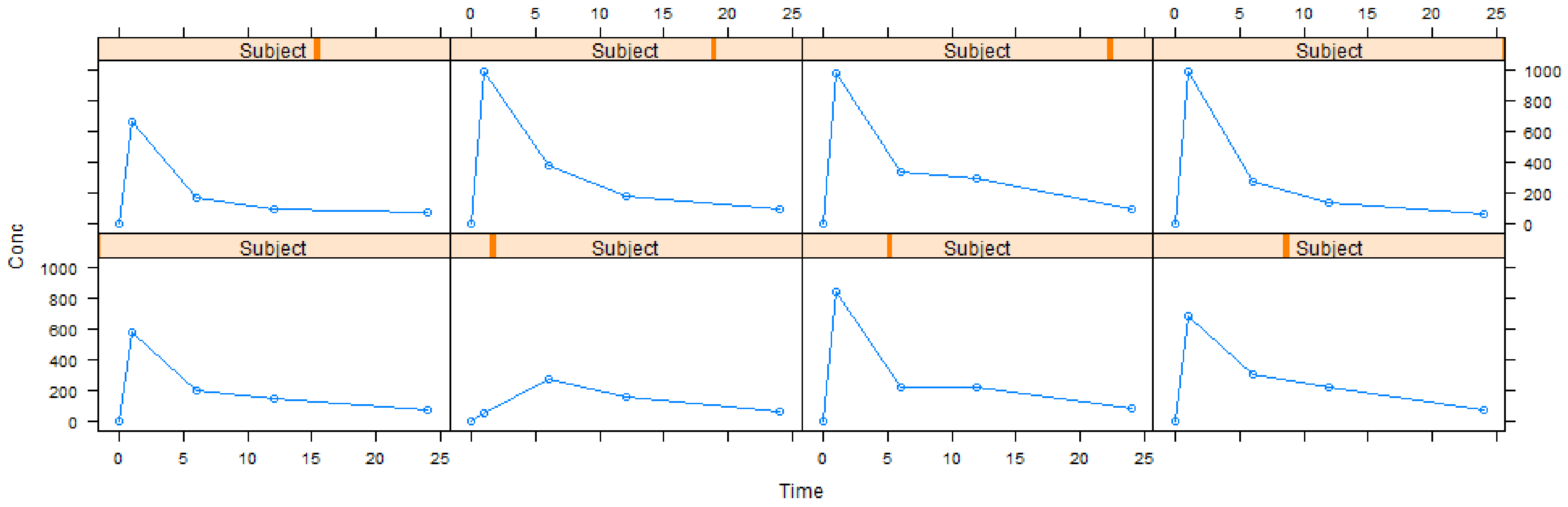
Why Not Lattice?

- Default options can be frustrating
- Default styling doesn't look great
- Making good use of the panel / panel.groups structure needs lots of "function" knowledge
- Some "tricks" needed to do more than 2 levels of nested grouping

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

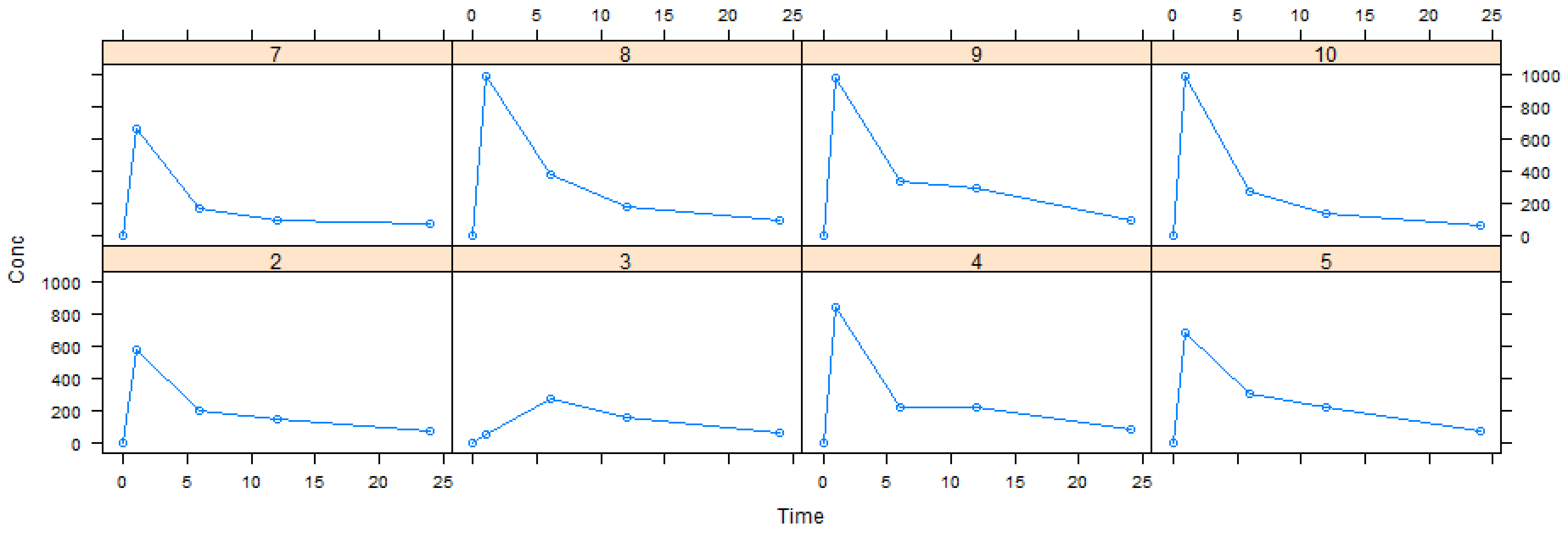
Frustration #1: Panel Headers

```
> head(pkData)
  Subject Dose Time      Conc
1       2   25    0   0.00000
2       2   25    1 574.28537
3       2   25    6 201.29697
4       2   25   12 146.88094
5       2   25   24  70.23041
6       3   25    0   0.00000
> xyplot(Conc ~ Time | Subject, data = pkData, subset = Subject <= 10, type = "o")
```



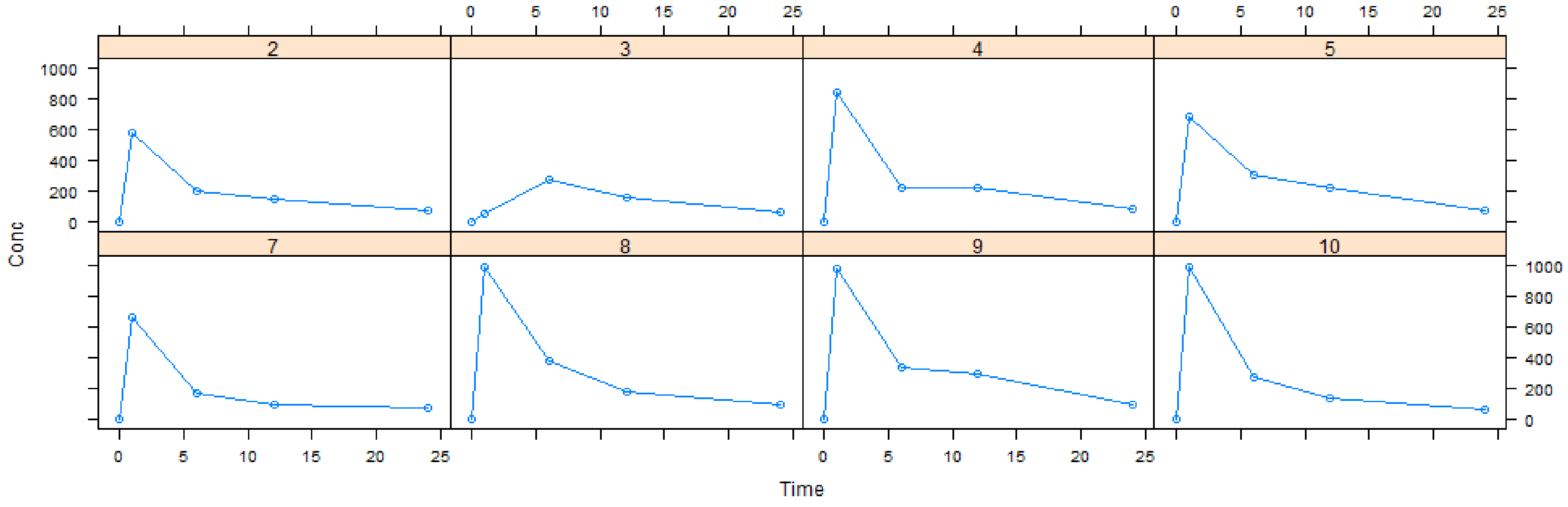
Frustration #2: Panel Order

```
> head(pkData)
  Subject Dose Time      Conc
1       2   25    0   0.00000
2       2   25    1 574.28537
3       2   25    6 201.29697
4       2   25   12 146.88094
5       2   25   24  70.23041
6       3   25    0   0.00000
> xyplot(Conc ~ Time | factor(Subject), data = pkData, subset = Subject <= 10, type = "o")
```



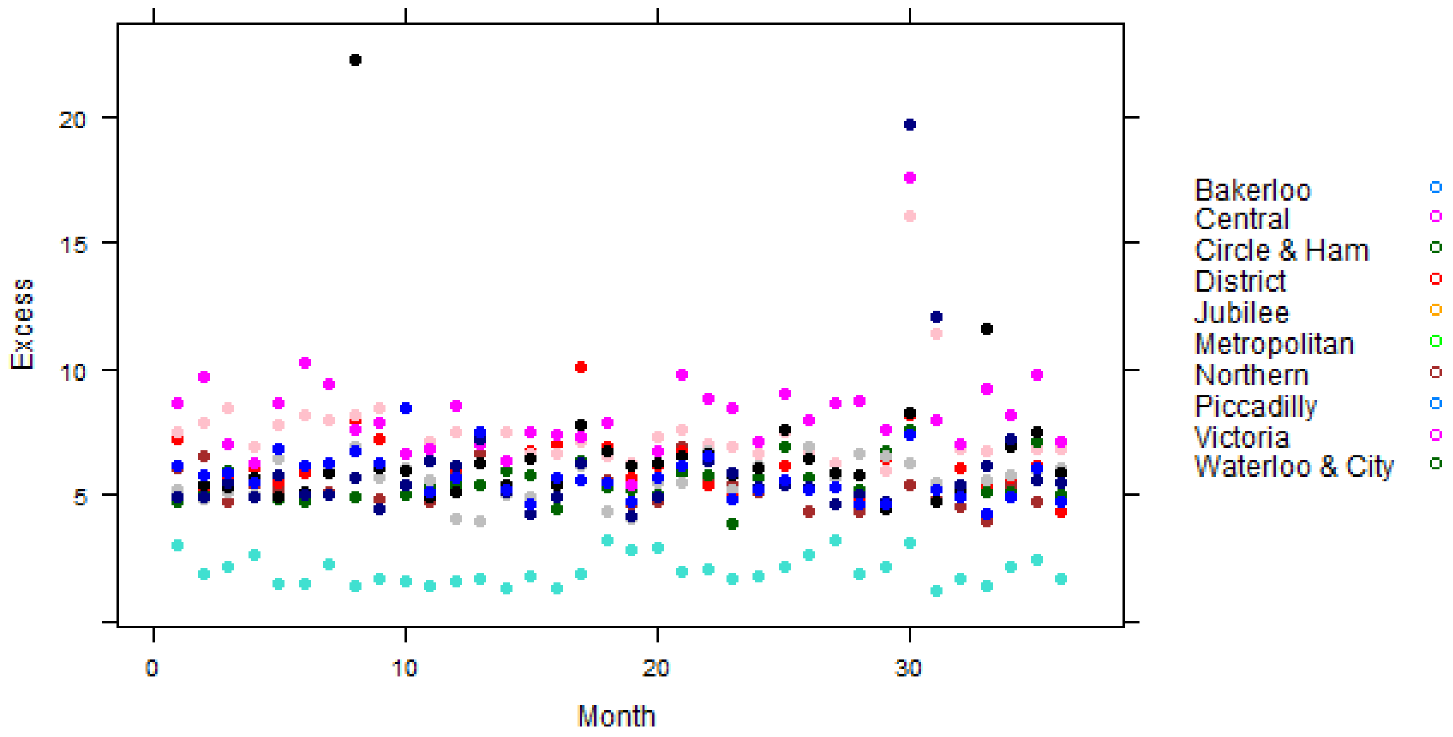
Frustration #2: Panel Order

```
> xyplot(Conc ~ Time | factor(Subject), data = pkData,
+          subset = Subject <= 10, type = "o", as.table = TRUE)
```



Frustration #3: Using styles

```
> lineCols
 [1] "brown"     "red"       "pink"      "darkgreen" "grey"      "magenta"
 [7] "black"     "navy"      "blue"      "turquoise"
> xyplot(Excess ~ Month, data = tubeData, groups = Line, pch = 16,
+        col = lineCols, auto.key = list(space = "right"))
```



Why ggplot2?

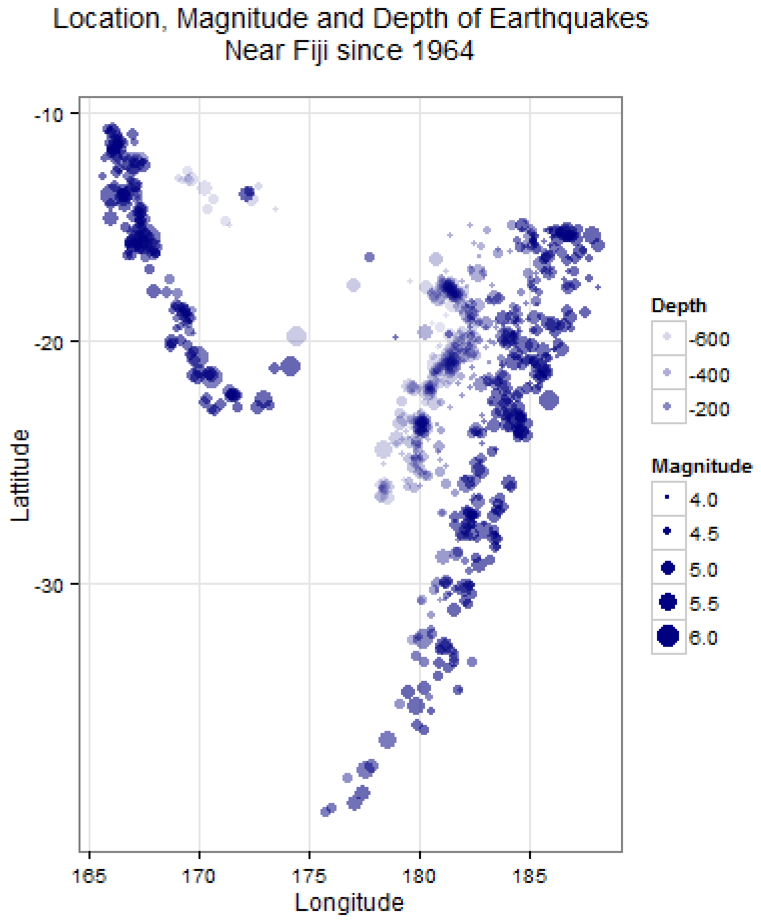
All the panelling advantages of lattice plus …

- It's pretty
- It's quick (to type)
- Styling is handled for you

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Why ggplot2?

```
qplot(long, lat, data = quakes, alpha = -depth, size = mag, col = I("navy"),
      main= "Location, Magnitude and Depth of Earthquakes\nNear Fiji since 1964\n",
      xlab = "Longitude", ylab = "Lattitude") +
  scale_alpha_continuous("Depth", range = c(0.1, 0.6)) +
  scale_size_continuous("Magnitude") +
  coord_map()
```



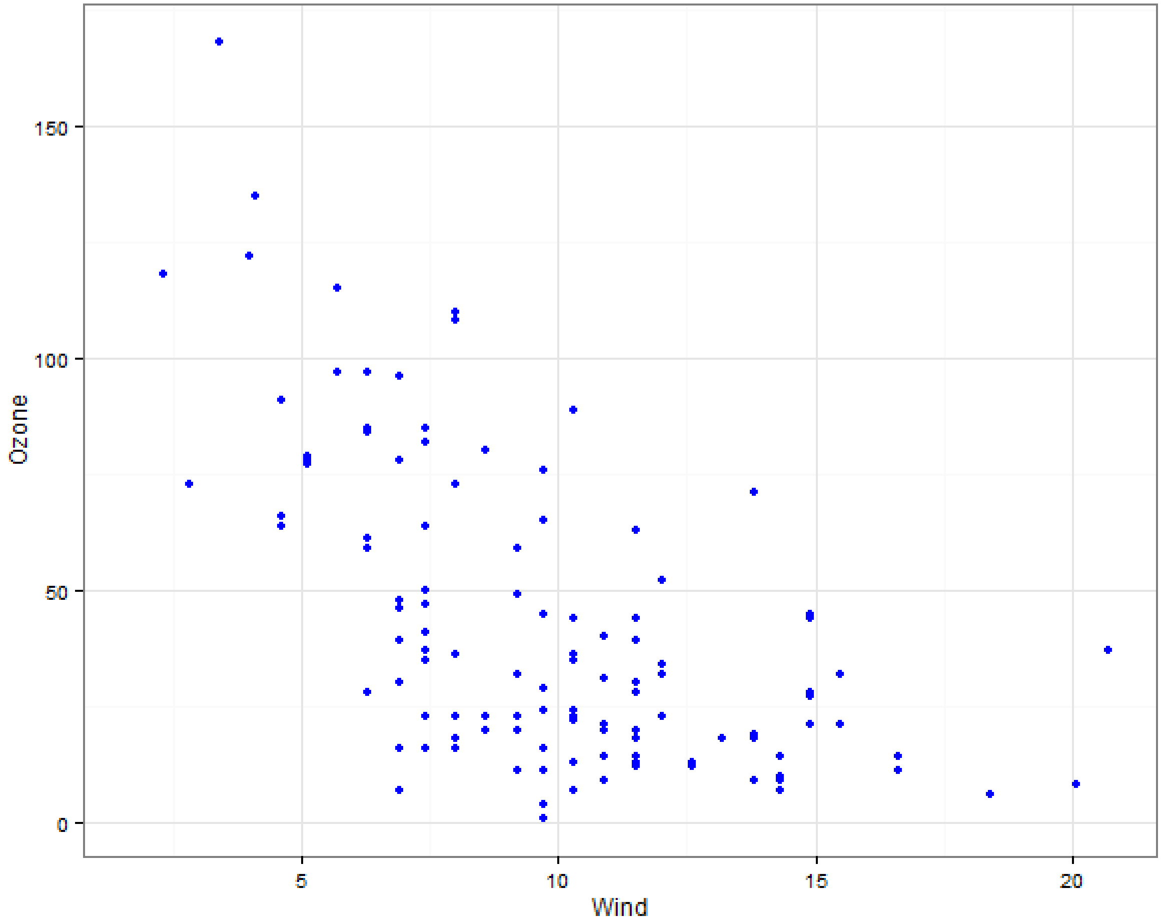
Why Not ggplot2?

- Steep learning curve

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Steep Learning Curve

```
qplot(Wind, Ozone, data = airquality, col = "blue")
qplot(Wind, Ozone, data = airquality, col = I("blue"))
```



Why Not ggplot2?

- Steep learning curve
- Help files are difficult to navigate
- Graphics are slower to render
- Limitations of framework
  - Can feel “hacky” for non-standard graphics
  - No 3D graphics
  - Complex examples may require “grid” knowledge

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)

Conclusions

- Both save huge amounts of time vs “graphics”
- ggplot2 styling is nice and easier to control
- Lattice is more flexible and is quicker to render

- Audience Vote!

Rich Pugh (rpugh@mango-solutions.com)
Andy Nicholls (anicholls@mango-solutions.com)