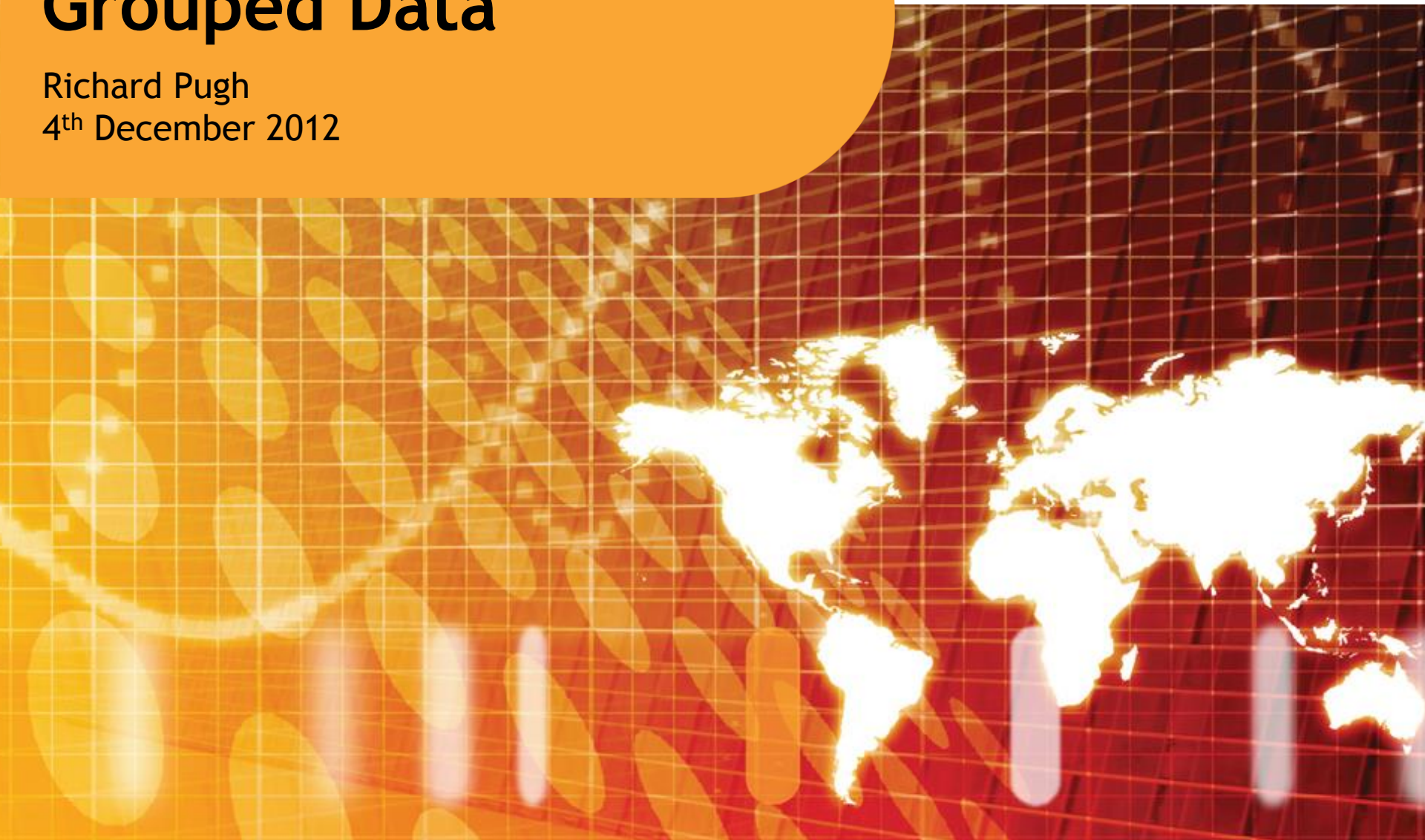
Using Lattice to Plot Grouped Data

Richard Pugh
4th December 2012

MANGOSOLUTIONS
data analysis that delivers



MANGOSOLUTIONS
data analysis that delivers

Agenda

- What are Lattice Graphics?
- Panel Functions
- Grouped data
- The mechanics of grouped plots
- Plotting more than 1 group
- Summary

MANGOSOLUTIONS
data analysis that delivers

The Data We Will Use

- Something relevant and 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

	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
7	Bakerloo	7	29.30	5.14	34.44	1906	23.2	DT	25
8	Bakerloo	8	29.30	5.73	35.03	1906	23.2	DT	25
9	Bakerloo	9	29.30	4.80	34.10	1906	23.2	DT	25
10	Bakerloo	10	29.30	5.95	35.25	1906	23.2	DT	25
11	Bakerloo	11	29.30	4.76	34.06	1906	23.2	DT	25
12	Bakerloo	12	29.30	6.00	35.30	1906	23.2	DT	25
13	Bakerloo	13	29.30	6.67	35.97	1906	23.2	DT	25
14	Bakerloo	14	29.30	5.24	34.54	1906	23.2	DT	25
15	Bakerloo	15	29.30	4.83	34.13	1906	23.2	DT	25
16	Bakerloo	16	29.30	5.50	34.80	1906	23.2	DT	25
17	Bakerloo	17	29.30	6.19	35.50	1906	23.2	DT	25
18	Bakerloo	18	29.30	5.60	34.90	1906	23.2	DT	25
19	Bakerloo	19	29.30	4.64	33.94	1906	23.2	DT	25
20	Bakerloo	20	29.30	4.74	34.04	1906	23.2	DT	25
21	Bakerloo	21	29.46	6.96	36.42	1906	23.2	DT	25
22	Bakerloo	22	29.30	5.72	35.02	1906	23.2	DT	25
23	Bakerloo	23	29.28	5.40	34.67	1906	23.2	DT	25
24	Bakerloo	24	29.28	5.11	34.38	1906	23.2	DT	25
25	Bakerloo	25	29.28	5.65	34.93	1906	23.2	DT	25

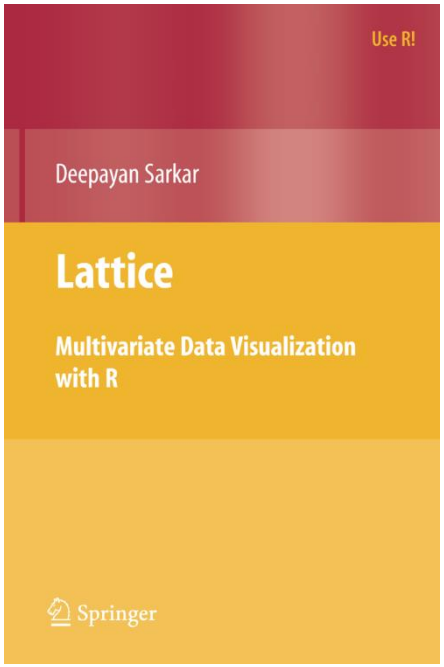
MANGOSOLUTIONS
data analysis that delivers

What are Lattice Graphics?

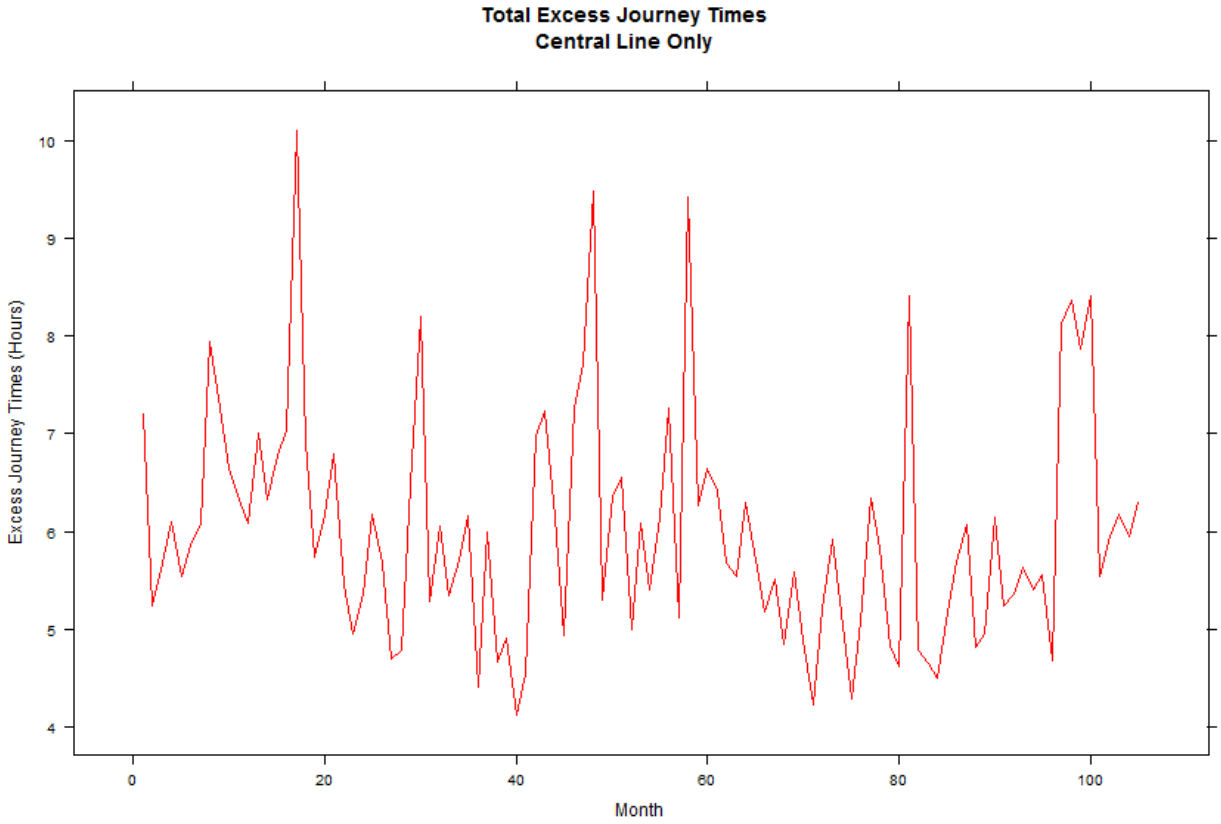
MANGOSOLUTIONS
data analysis that delivers

Overview of Lattice Graphics

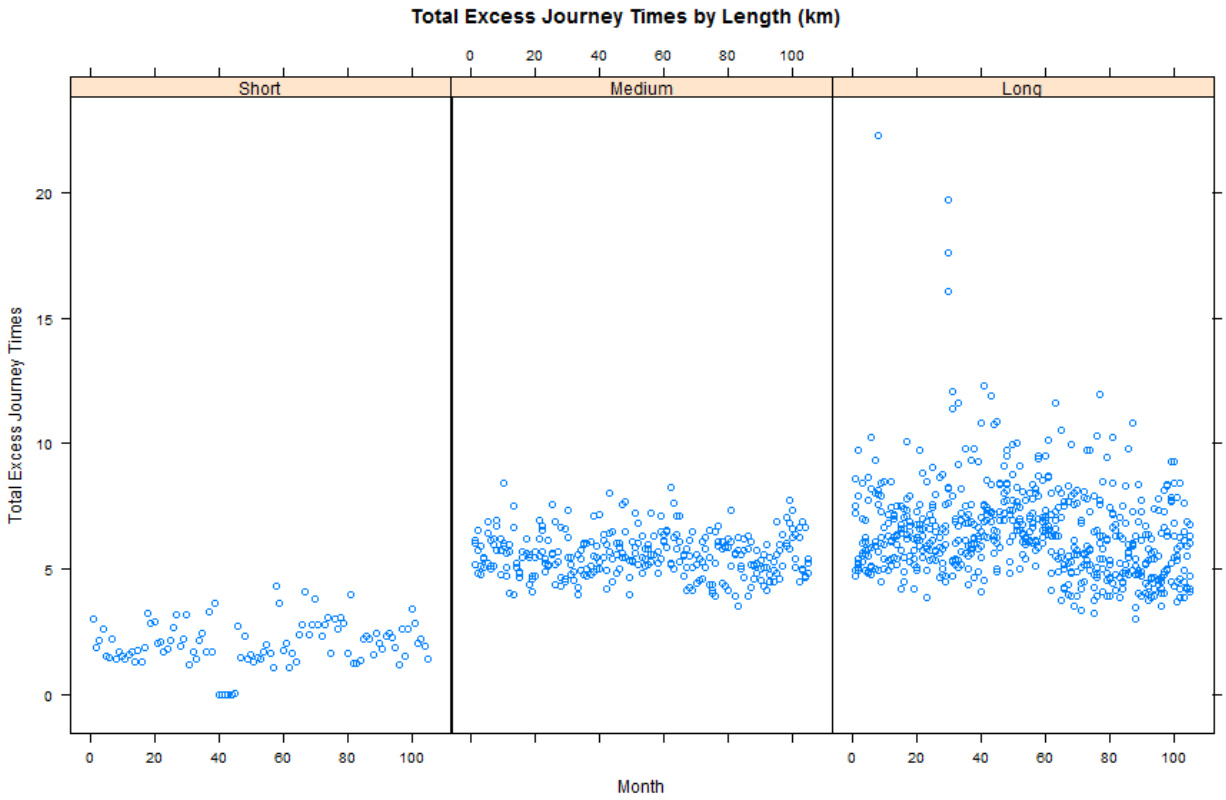
- One of the graphic systems of R (others include “Traditional” and “GGPlot”)
- An implementation of the S+ “Trellis” Graphics
- Written by Deepayan Sarkar, Fred Hutchinson Cancer Research Center



```
xyplot(Excess ~ Month, data = tubeData, subset = Line == "Central",
       col = "red", type = "l", ylab = "Excess Journey Times (Hours)",
       main = "Total Excess Journey Times\nCentral Line Only")
```



```
tubeData <- transform(tubeData,
      MLevel = factor(ifelse(Length > 50, "Long", ifelse(Length > 20, "Medium", "Short")),
                      levels = c("Short", "Medium", "Long")))
xyplot(Excess ~ Month | MLevel,
       data = tubeData, type = "p", ylab = "Total Excess Journey Times",
       main = "Total Excess Journey Times by Length (km)", layout = c(3, 1))
```



MANGOSOLUTIONS
data analysis that delivers

Panel Functions

Panel Functions

- For each lattice graph, R performs the following actions:
  - Partitions the data
  - Draws the graph “outline” (i.e. the “panels”)
  - Passes the data for each panel into the “panel” function
- We can overwrite this panel function and supply our own …

MANGOSOLUTIONS
data analysis that delivers

Panel Functions

- The default “panel” function for a lattice function is “panel.*NameOfFunction*”
- Let’s look at **`panel.xyplot`** …

```
function (x, y, type = "p", groups = NULL, pch = if (is.null(groups)) plot.symbol$pch else superpose.symbol$pch,
    col, col.line = if (is.null(groups)) plot.line$col else superpose.line$col,
    col.symbol = if (is.null(groups)) plot.symbol$col else superpose.symbol$col,
    font = if (is.null(groups)) plot.symbol$font else superpose.symbol$font,
    fontfamily = if (is.null(groups)) plot.symbol$fontfamily else superpose.symbol$fontfamily,
    fontface = if (is.null(groups)) plot.symbol$fontface else superpose.symbol$fontface,
    lty = if (is.null(groups)) plot.line$lty else superpose.line$lty,
    cex = if (is.null(groups)) plot.symbol$cex else superpose.symbol$cex,
    fill = if (is.null(groups)) plot.symbol$fill else superpose.symbol$fill,
    lwd = if (is.null(groups)) plot.line$lwd else superpose.line$lwd,
    horizontal = FALSE, ..., grid = FALSE, abline = NULL, jitter.x = FALSE,
    jitter.y = FALSE, factor = 0.5, amount = NULL, identifier = "xyplot")
```

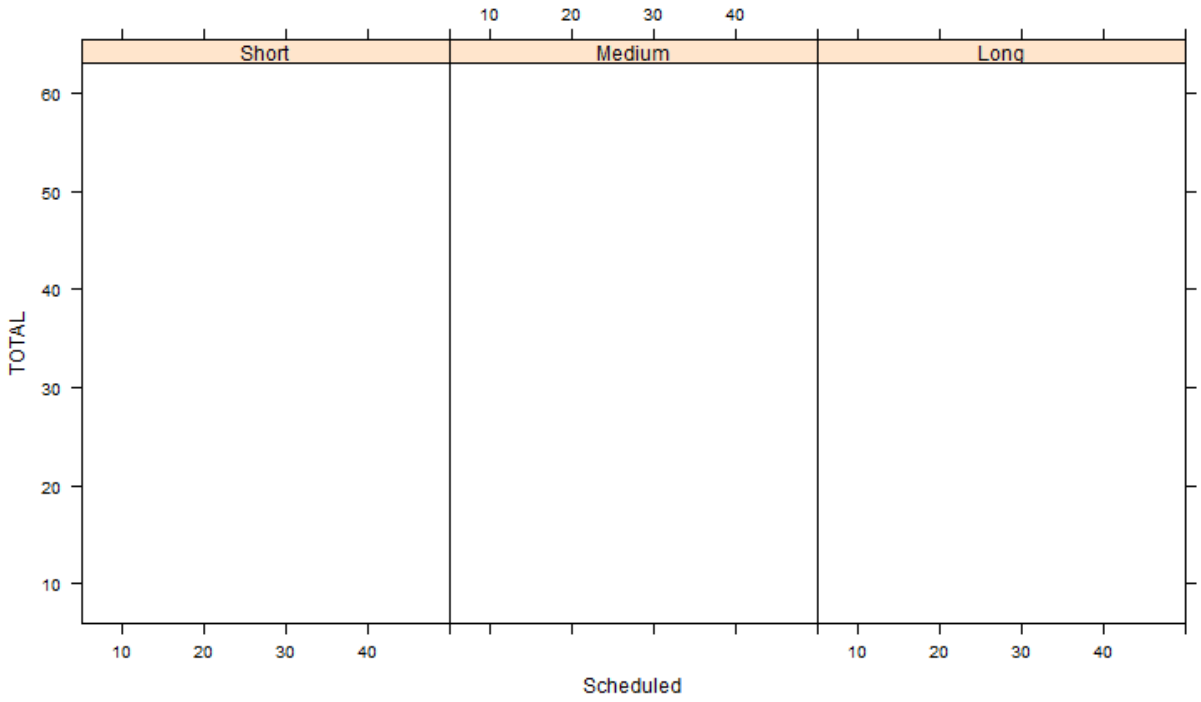
Arguments for styling

Jitter & other less used bits

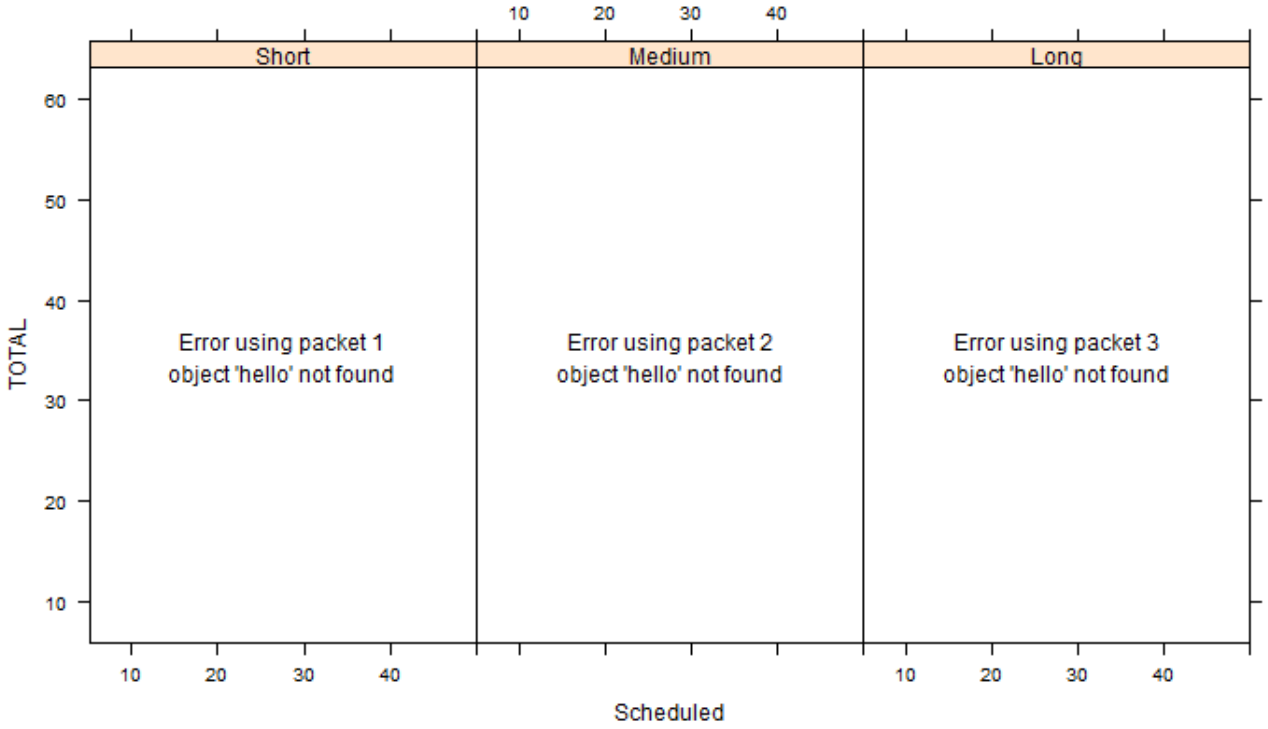
```
function (x, y, type = "p", groups = NULL,
```

Let's write our own panel function …….

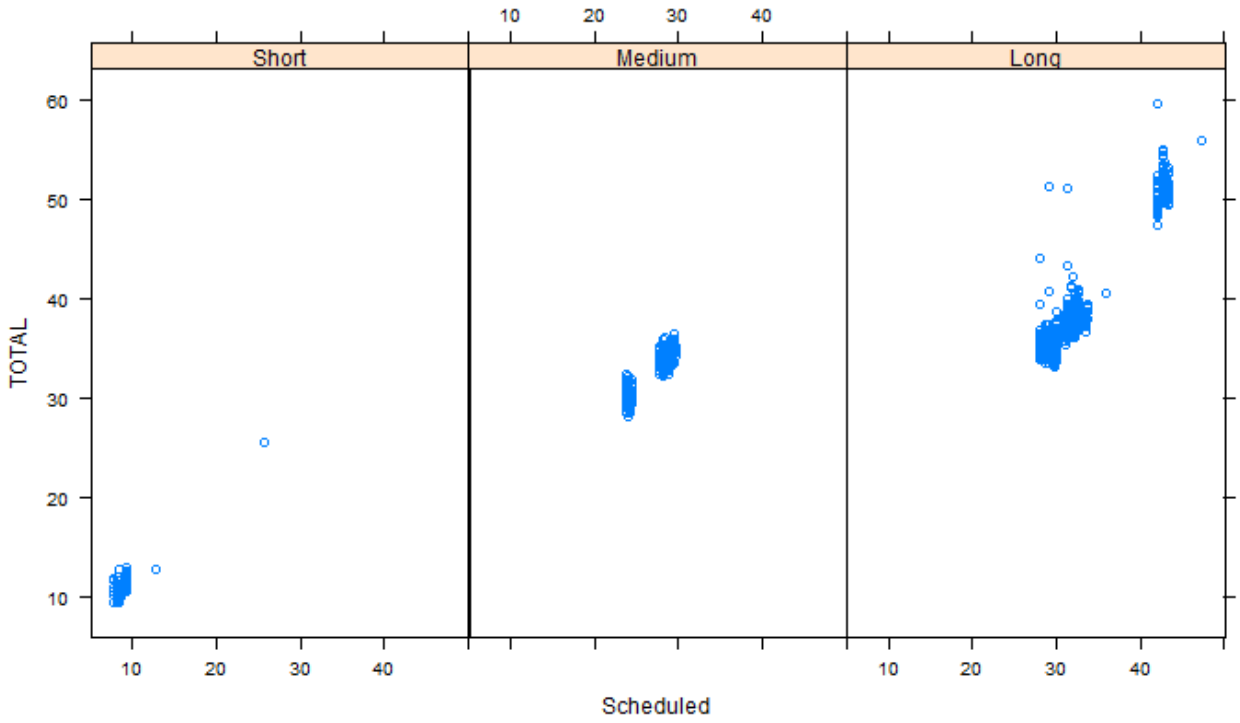
```
panel.nothing <- function(x, y, ...) {
  # Nothing in here!
}
xyplot( TOTAL ~ Scheduled | MLevel, data = tubeData, panel = panel.nothing)
```



```
panel.error <- function(x, y, ...) {
  hello
}
xyplot( TOTAL ~ Scheduled | MLevel, data = tubeData, panel = panel.error)
```



```
panel.itWorks <- function(x, y, ...) {
  panel.xyplot(x, y, ...)
}
xyplot( TOTAL ~ Scheduled | MLevel, data = tubeData, panel = panel.itWorks)
```

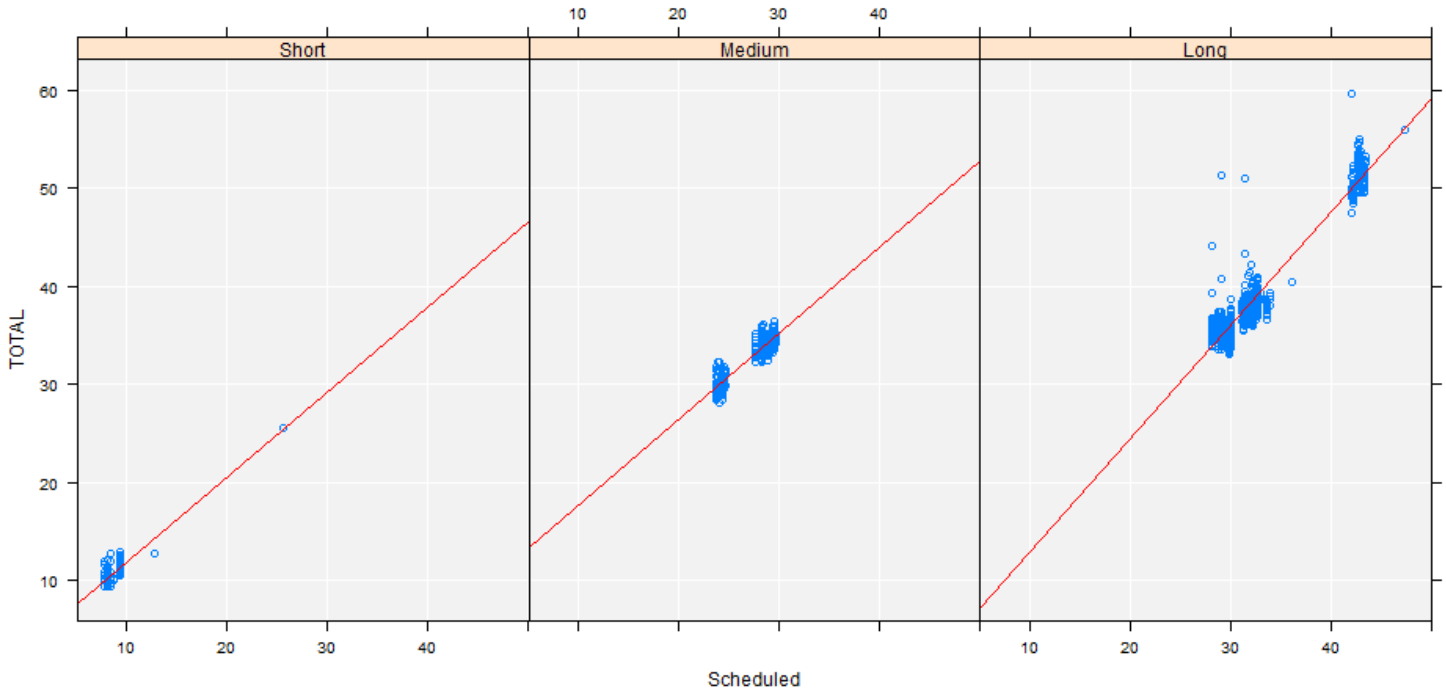


Now what?

- Now we can insert other functions into a panel function which will then impact each individual plot panel
- We can call:
  - Lattice “low level functions” (**`lpoints`**, **`ltext`**, **`llines`**, **`lpolygon`** etc)
  - Existing “panel.” functions that exist for this purpose

```
> apropos("^panel")
 [1] "panel.3dscatter"       "panel.3dwire"          "panel.abline"
 [4] "panel.arrows"          "panel.average"         "panel.axis"
 [7] "panel.barchart"        "panel.brush.splom"     "panel.bwplot"
[10] "panel.cloud"           "panel.contourplot"     "panel.curve"
[13] "panel.densityplot"     "panel.dotplot"         "panel.error"
[16] "panel.error"           "panel.fill"            "panel.grid"
[19] "panel.histogram"       "panel.identify"        "panel.identify.cloud"
[22] "panel.identify.qqmath" "panel.itWorks"         "panel.levelplot"
[25] "panel.levelplot.raster" "panel.linejoin"       "panel.lines"
[28] "panel.link.splom"      "panel.lmline"          "panel.loess"
[31] "panel.mathdensity"     "panel.nothing"         "panel.number"
[34] "panel.pairs"           "panel.parallel"        "panel.points"
[37] "panel.polygon"         "panel.qq"              "panel.qqmath"
[40] "panel.qqmathline"      "panel.rect"            "panel.refline"
[43] "panel.rug"             "panel.segments"        "panel.smooth"
[46] "panel.smoothScatter"   "panel.spline"         "panel.splom"
[49] "panel.stripplot"       "panel.superpose"       "panel.superpose.2"
[52] "panel.superpose.plain" "panel.text"            "panel.tmd.default"
[55] "panel.tmd.qqmath"      "panel.violin"          "panel.wireframe"
[58] "panel.xyplot"
```

```
panel.itWorks <- function(x, y, ...) {
  panel.fill(col = "grey95")                    # Light grey background
  panel.grid(h = -1, v = -1, col = "white")     # Some grid lines
  panel.xyplot(x, y, ...)                       # Draw the data
  panel.lmline(x, y, col = "red")               # Add a regression line
}
xyplot( TOTAL ~ Scheduled | MLevel, data = tubeData, panel = panel.itWorks)
```



MANGOSOLUTIONS
data analysis that delivers

Grouped Data

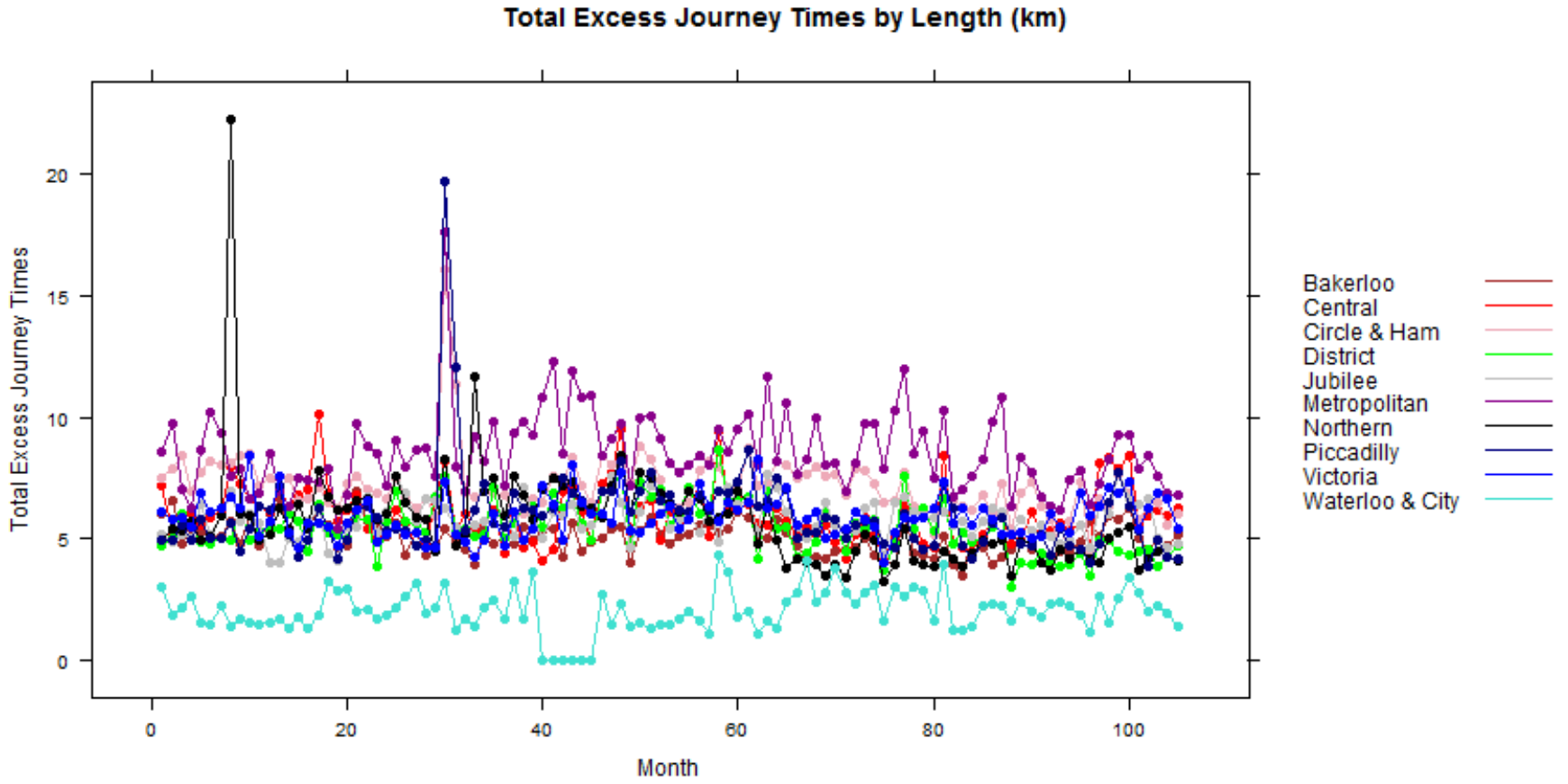
Grouped Data

- Many datasets will have inherit groupings
- Examples may include:
  - “Stock” for finance data
  - “Subject” for pharma data
- We *always* want to take this into account when working with graphics

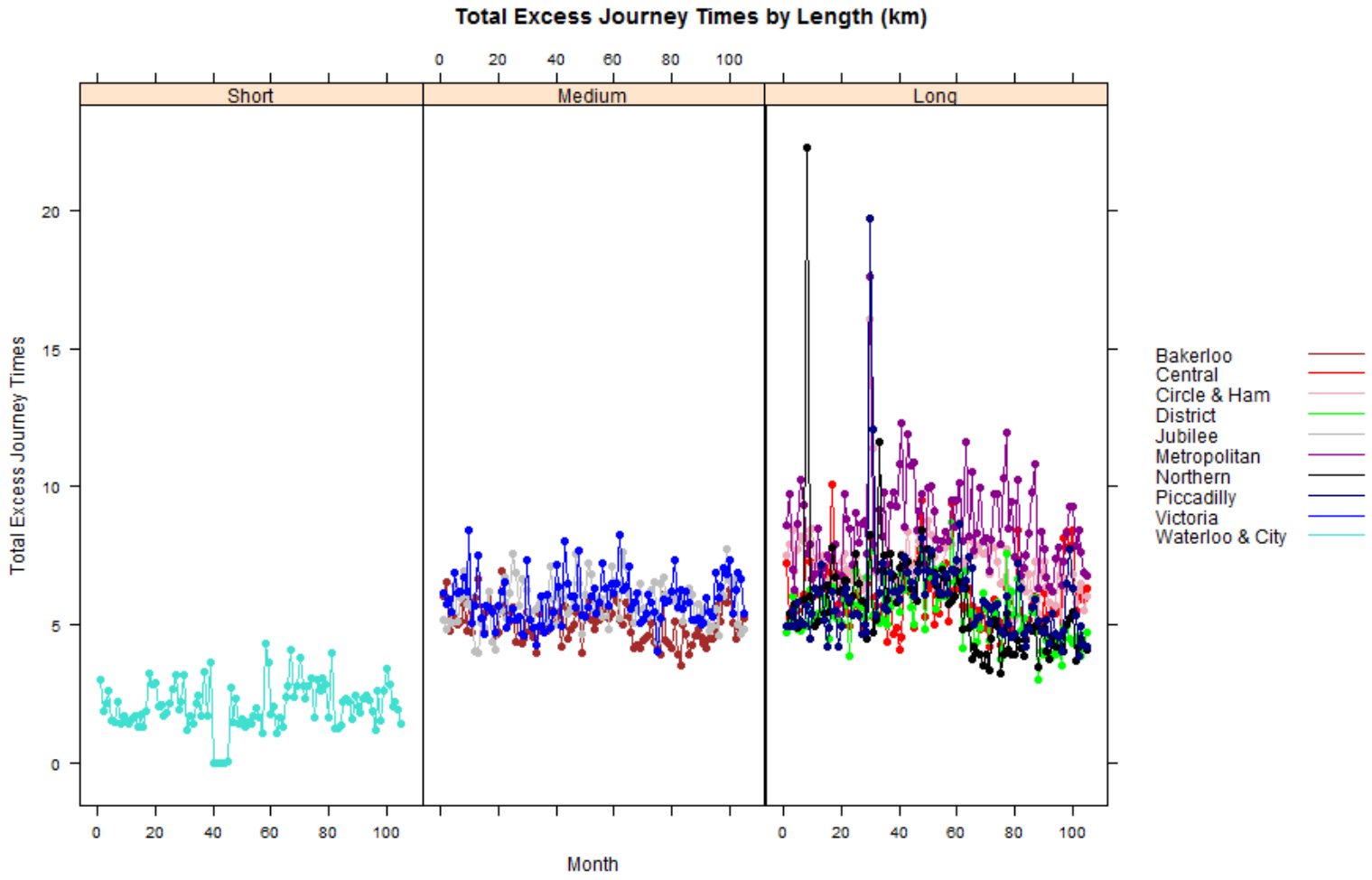
Grouped Data

- We can specify groupings within our data in order to plot these groupings separately
- By default, lattice will vary styling of the groups specified
- The key argument is the “groups” argument

```
xyplot(Excess ~ Month, groups = Line, type = "b",
       data = tubeData, ylab = "Total Excess Journey Times",
       main = "Total Excess Journey Times by Length (km)", lwd = 1.5,
       auto.key = list(space = "right", points = F, lines = T), par.settings = myStyles)
```



```
xyplot(Excess ~ Month | MLevel, groups = Line, type = "b",
       data = tubeData, ylab = "Total Excess Journey Times",
       main = "Total Excess Journey Times by Length (km)", lwd = 1.5,
       auto.key = list(space = "right", points = F, lines = T), par.settings = myStyles)
```



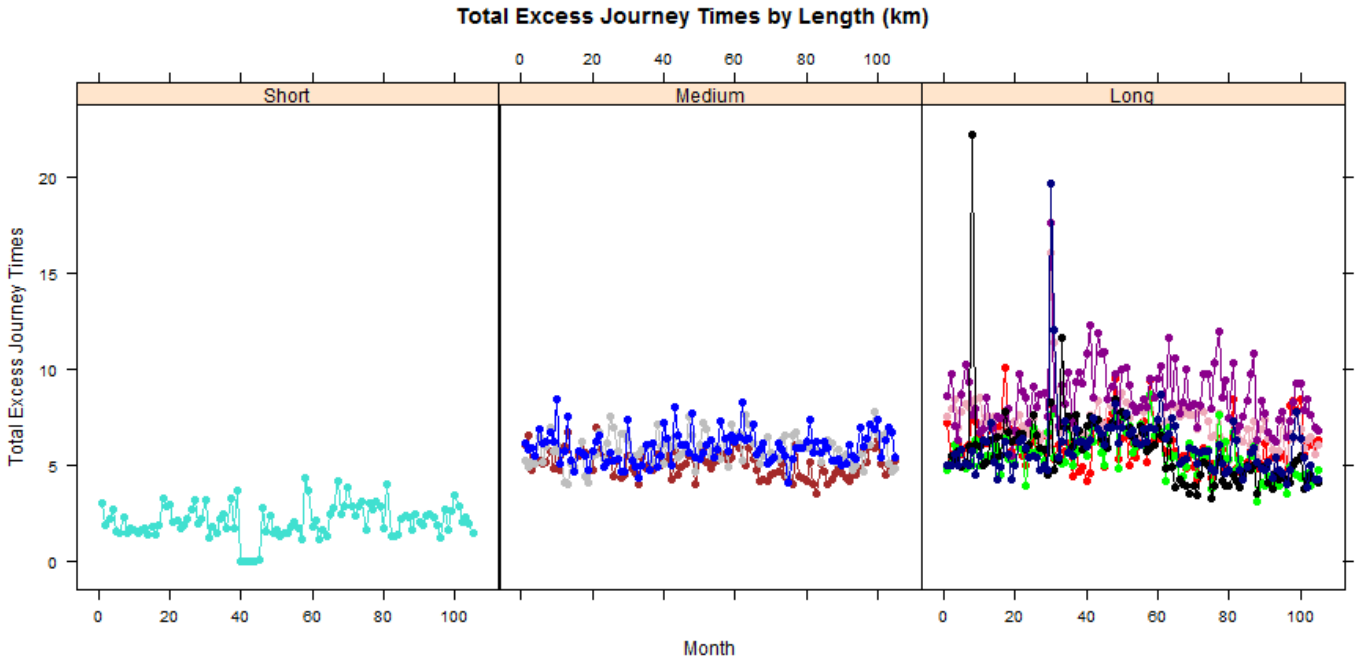
MANGOSOLUTIONS
data analysis that delivers

The Mechanics of Grouped Data

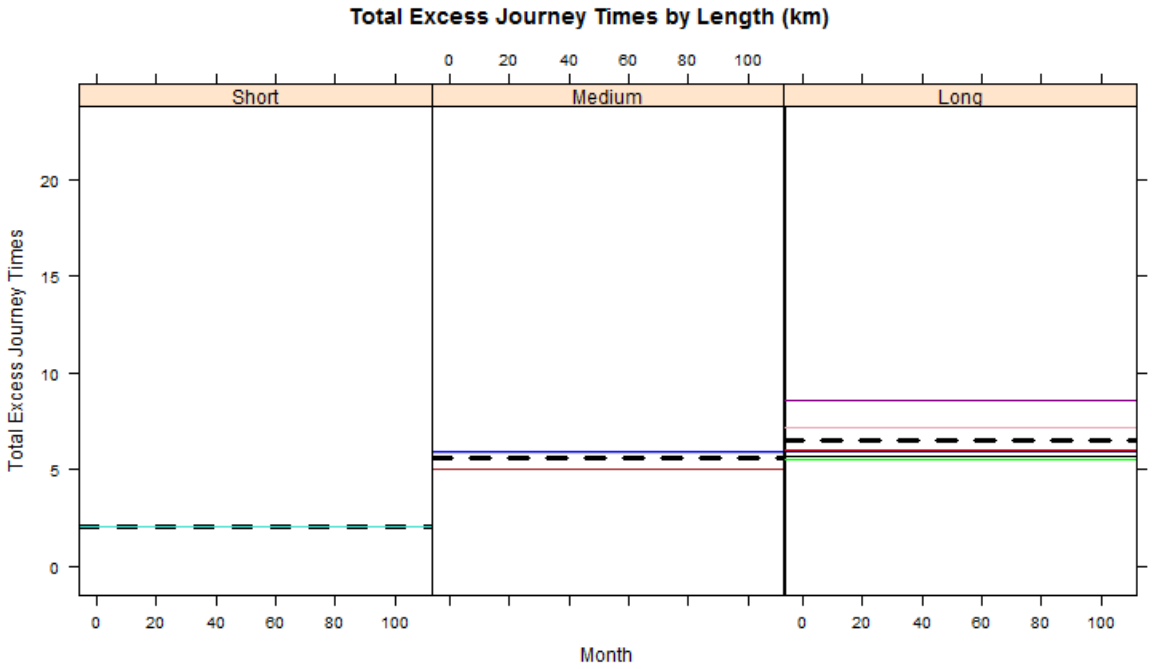
```
xyplot(Excess ~ Month | MLevel,
    groups = Line, type = "b",
    data = tubeData,
    panel = function(x, y, ...) {

      # Controls what happens in each "panel"
      panel.superpose(x, y, ...)  # Passes info to panel.groups

    },
    panel.groups = function(x, y, ...) {

      # Controls what happens for each "group"
      panel.xyplot(x, y, ...)     # Draws individual group

    }
)
```

```
xyplot(Excess ~ Month | MLevel,
   groups = Line, type = "b",
   data = tubeData,
   panel = function(x, y, ...) {

     # Controls what happens in each "panel"
     panel.superpose(x, y, ...)  # Passes info to panel.groups

   },
   panel.groups = function(x, y, ...) {

     # Controls what happens for each "group"
     panel.xyplot(x, y, ...)     # Draws individual group

   }
)
```

MANGOSOLUTIONS
data analysis that delivers



```
xyplot(Excess ~ Month | MLevel,
   groups = Line, type = "b",
   data = tubeData,
   panel = function(x, y, ...) {

     panel.abline(h = mean(y), col = "black", lty = 2, lwd = 3)
     panel.superpose(x, y, ...)  # Passes info to panel.groups

   },
   panel.groups = function(x, y, ...) {

     # Controls what happens for each "group"
     panel.abline(h = mean(y), ...)

   }
)
```

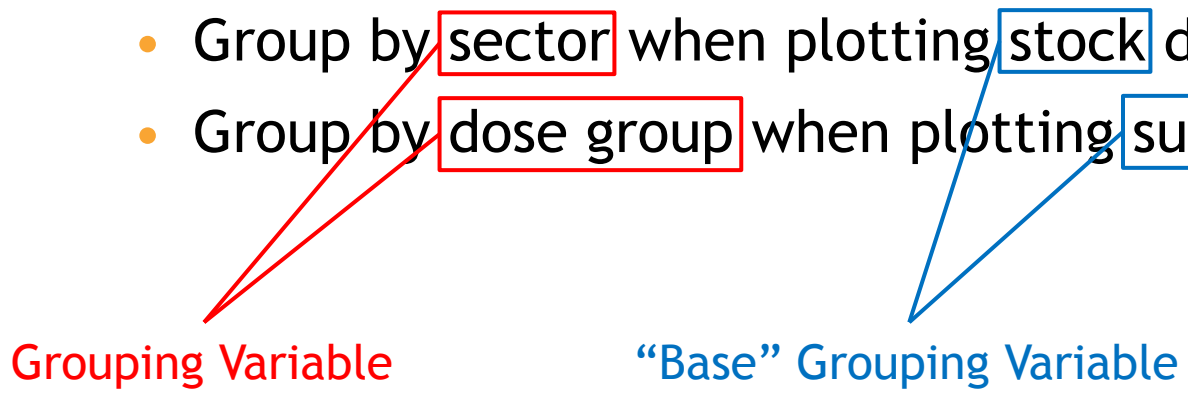


MANGOSOLUTIONS
data analysis that delivers

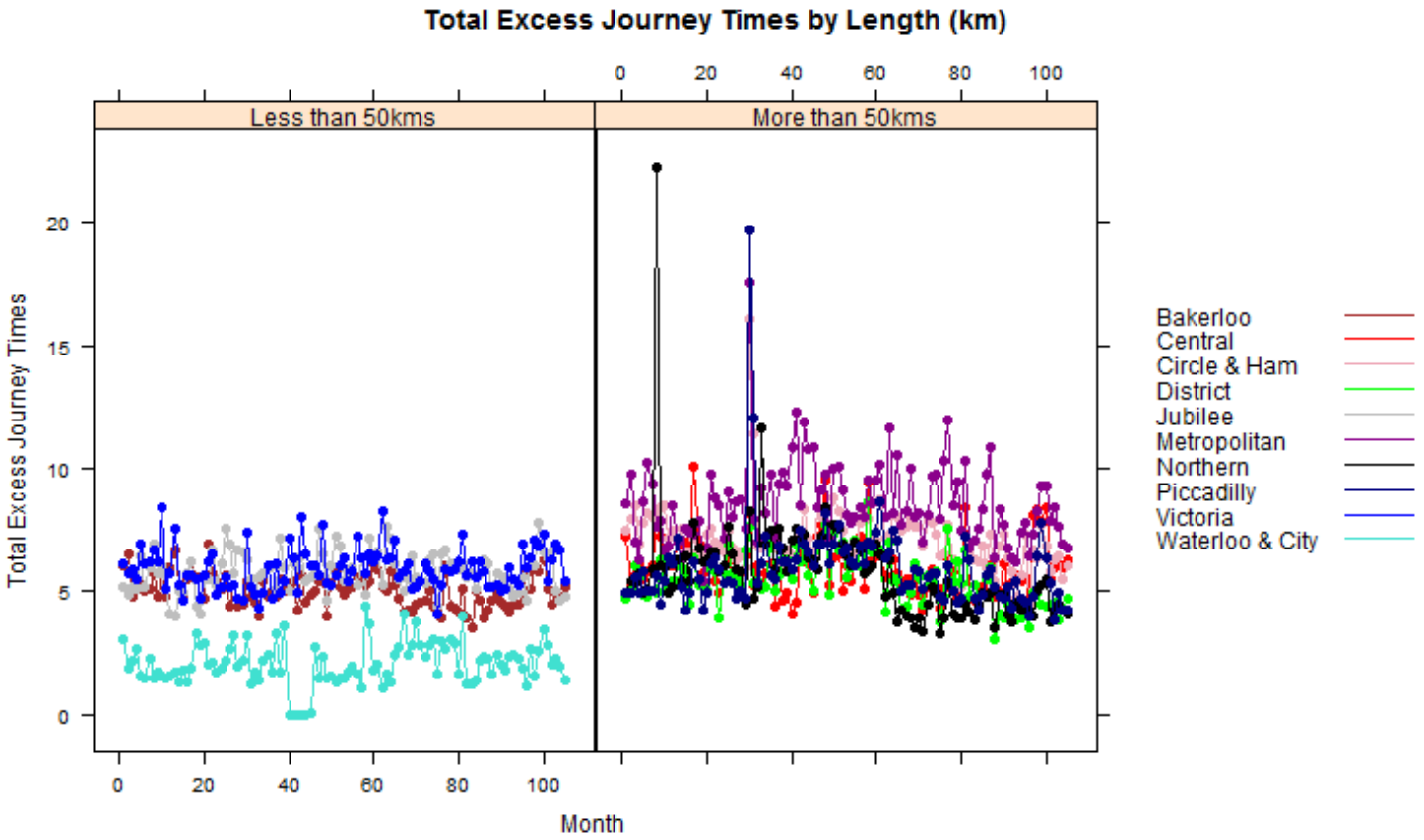
Plotting more than 1 group of data

Plotting more than 1 group of data

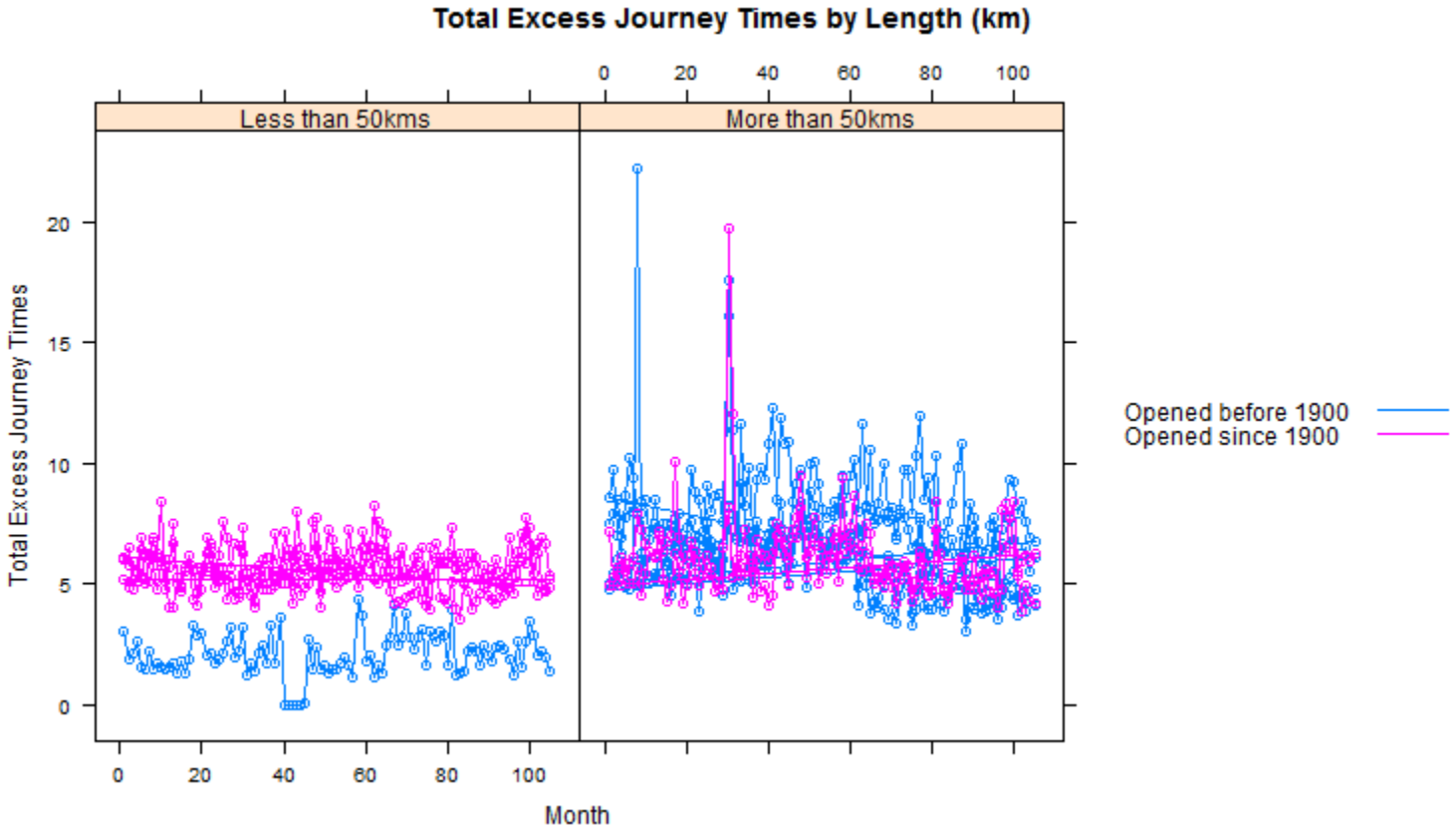
- Something I want to a do a lot
- Examples:
  - Group by sector when plotting stock data
  - Group by dose group when plotting subject data



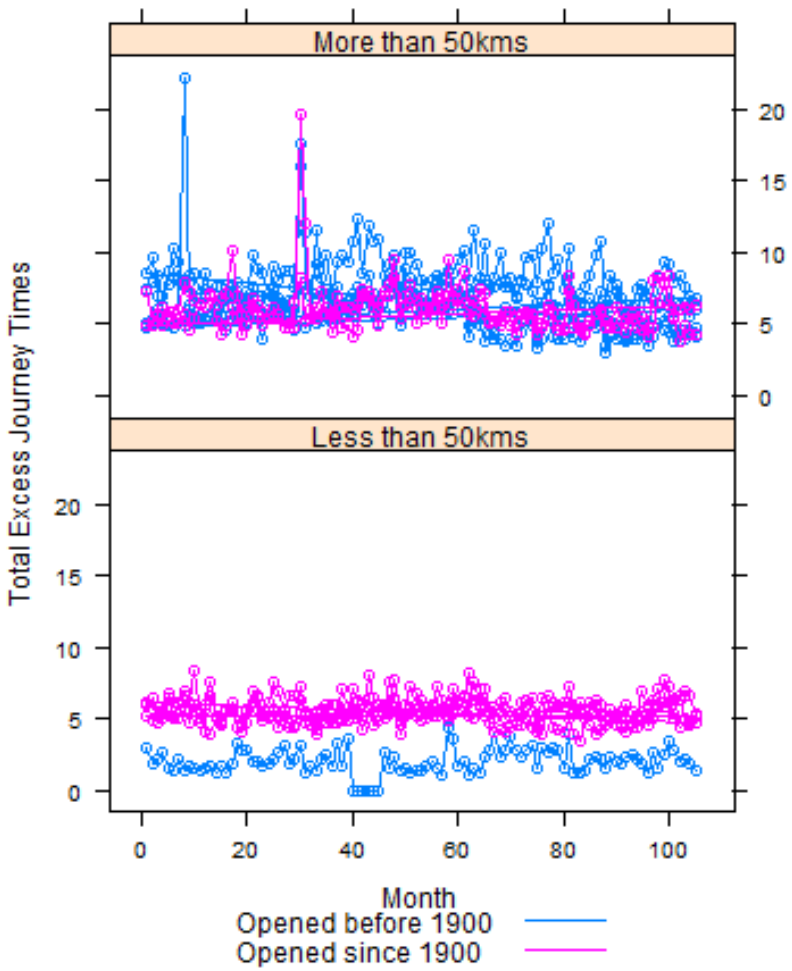
```
xyplot(Excess ~ Month | HowLong, groups = Line, type = "b",
       data = tubeData, ylab = "Total Excess Journey Times",
       main = "Total Excess Journey Times by Length (km)", lwd = 1.5,
       auto.key = list(space = "right", points = F, lines = T), par.settings = myStyles)
```



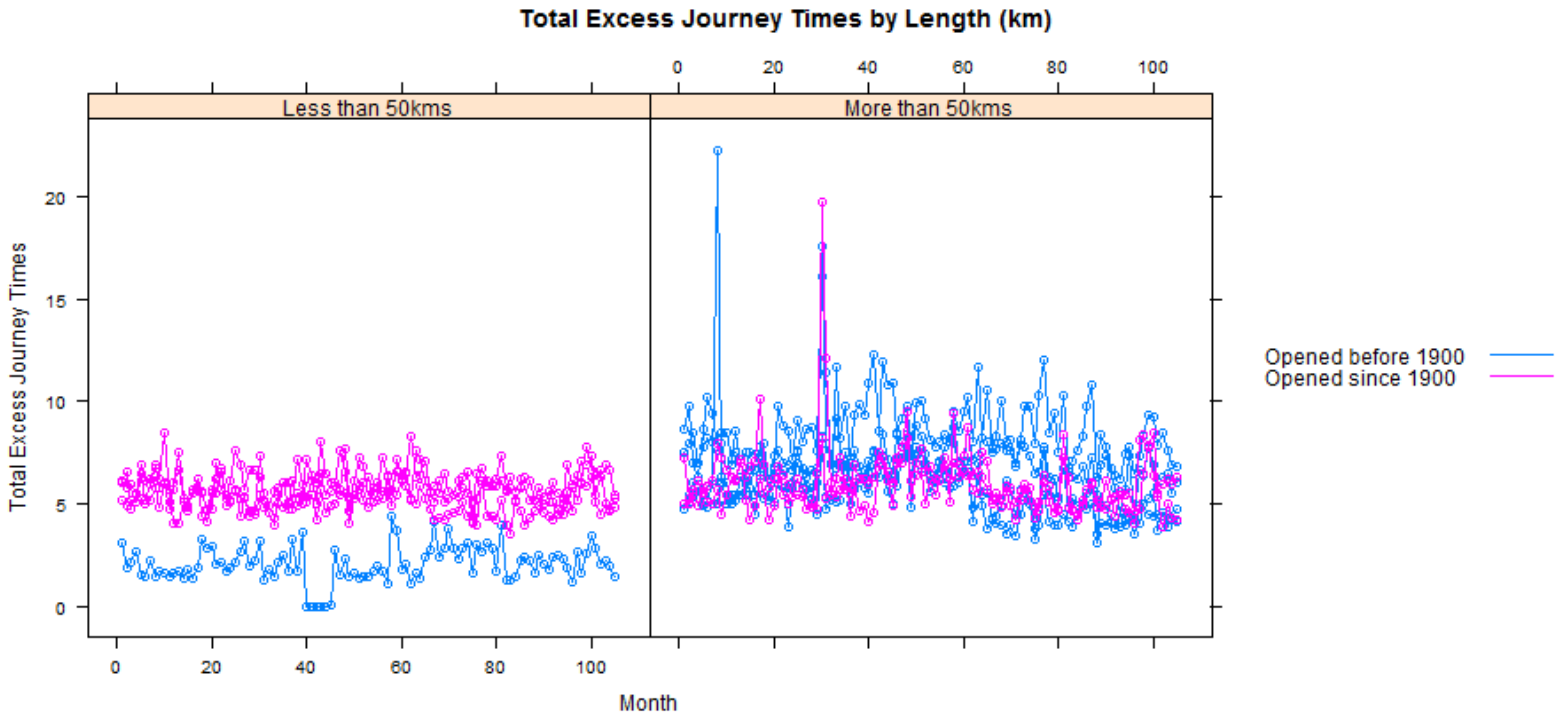
```
xyplot(Excess ~ Month | HowLong, groups = whenOpen, type = "b",
       data = tubeData, ylab = "Total Excess Journey Times",
       main = "Total Excess Journey Times by Length (km)",
       auto.key = list(space = "right", lines = T, points = F))
```



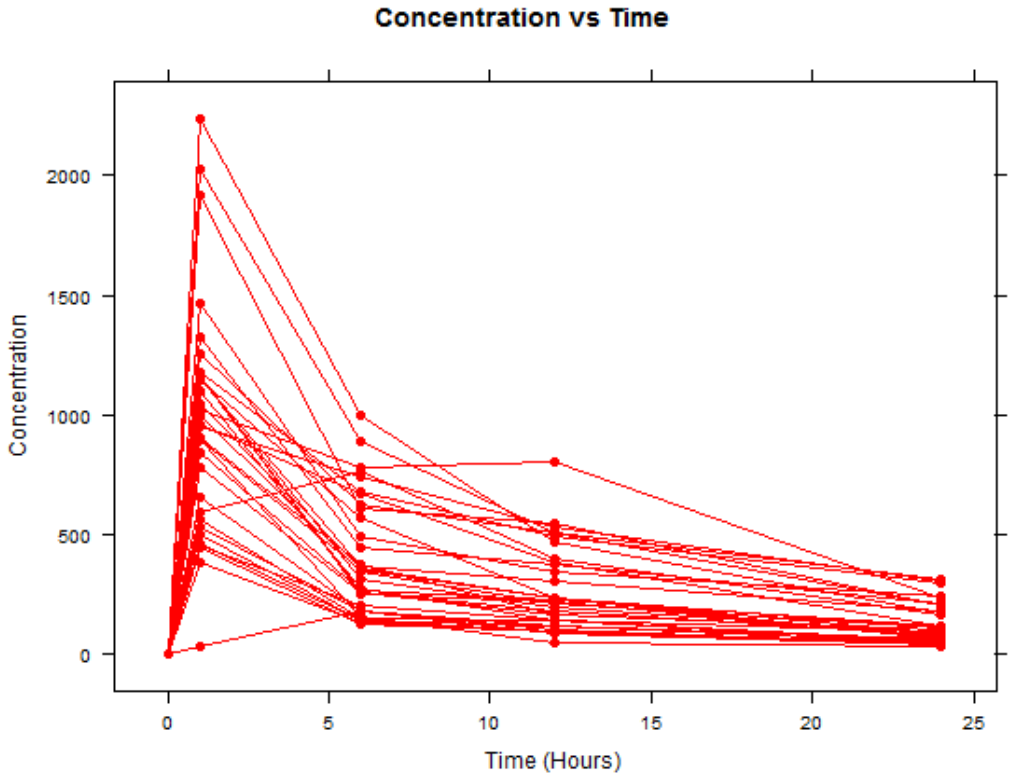
```
xyplot(Excess ~ Month | HowLong,
  groups = WhenOpen, type = "b",
  panel = function(x, y, ...) {
    panel.superpose(x, y, ...)
  },
  panel.groups = function(x, y, ...) {
    panel.xyplot(x, y, ...)
  },
  data = tubeData
  )
```



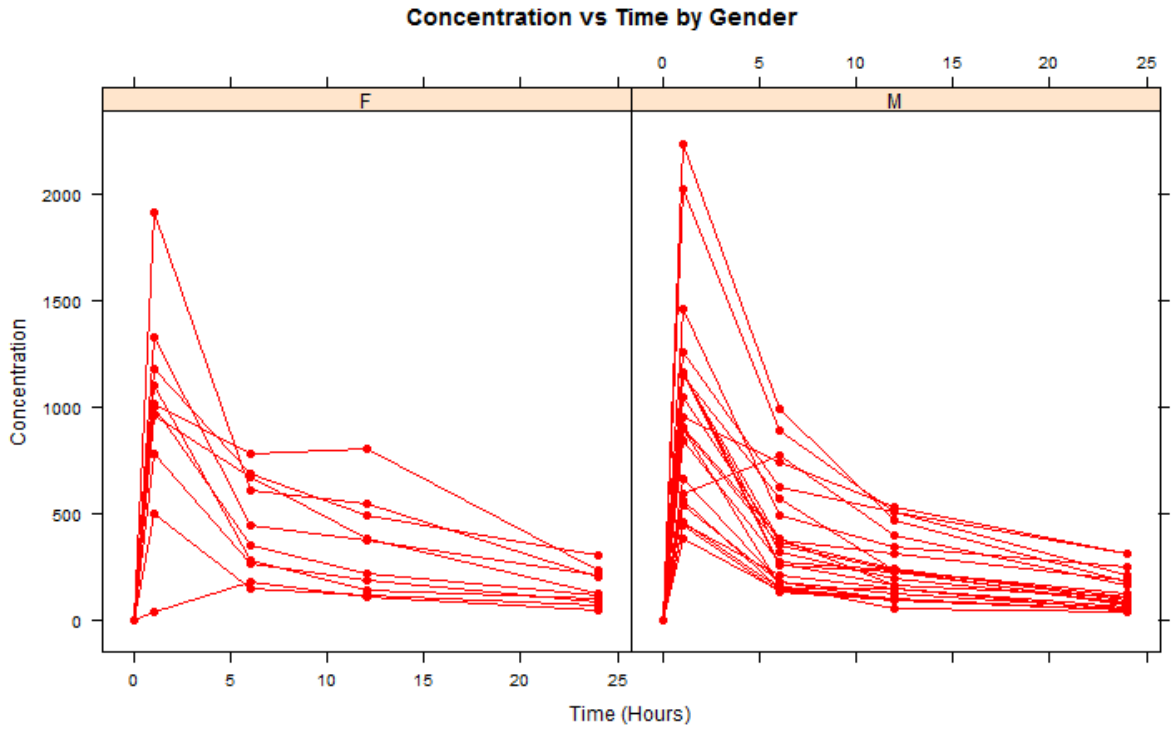
```
xyplot(Excess ~ Month | HowLong,
       groups = WhenOpen, type = "b",
       Line = tubeData$Line,
       panel = panel.superpose,
       panel.groups = function(x, y, subscripts, Line, ...) {
         panel.xyplot(x, y, groups = Line[subscripts],
                      subscripts = 1:length(x), ...)
       },
       data = tubeData
)
```



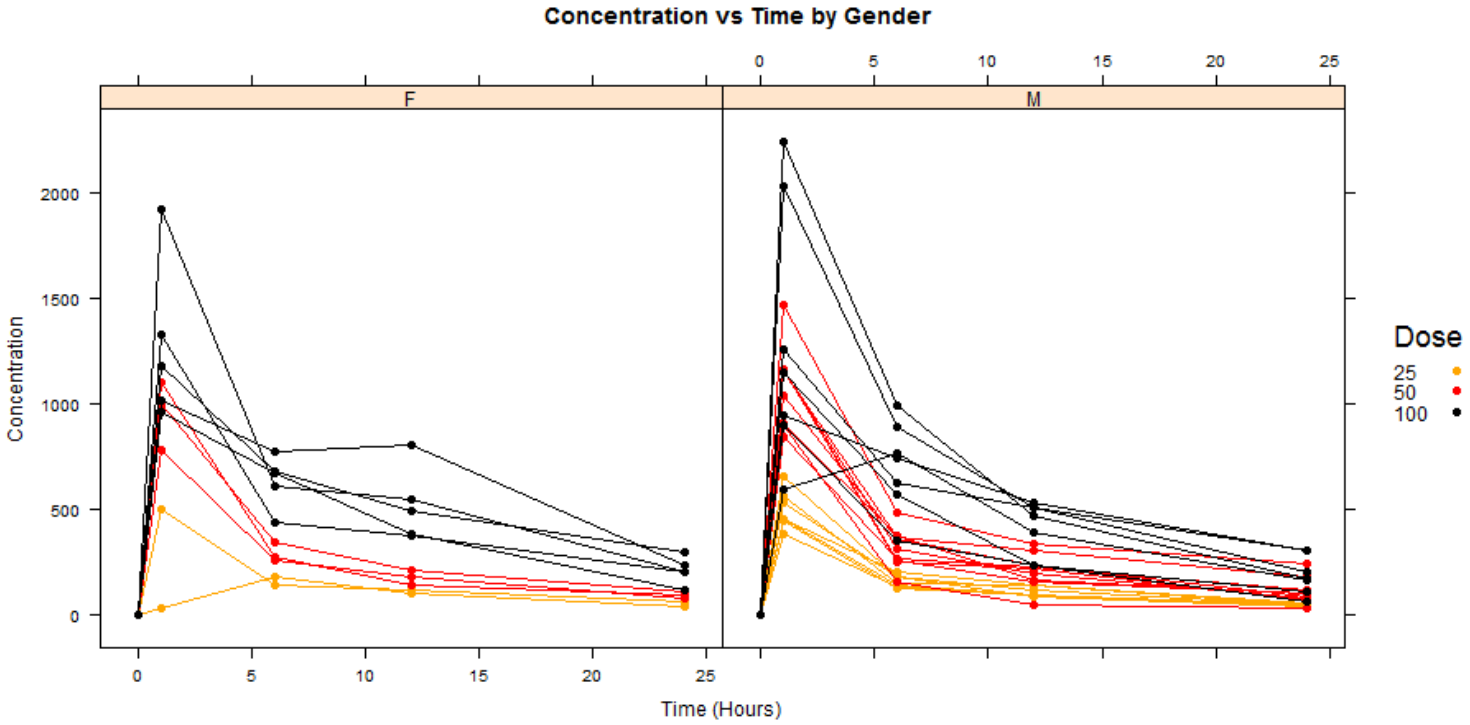
```
xyplot(Conc ~ Time,
       groups = Subject, type = "b",
       data = pkData, col = "red", pch = 16,
       main = "Concentration vs Time",
       xlab = "Time (Hours)", ylab = "Concentration"
)
```



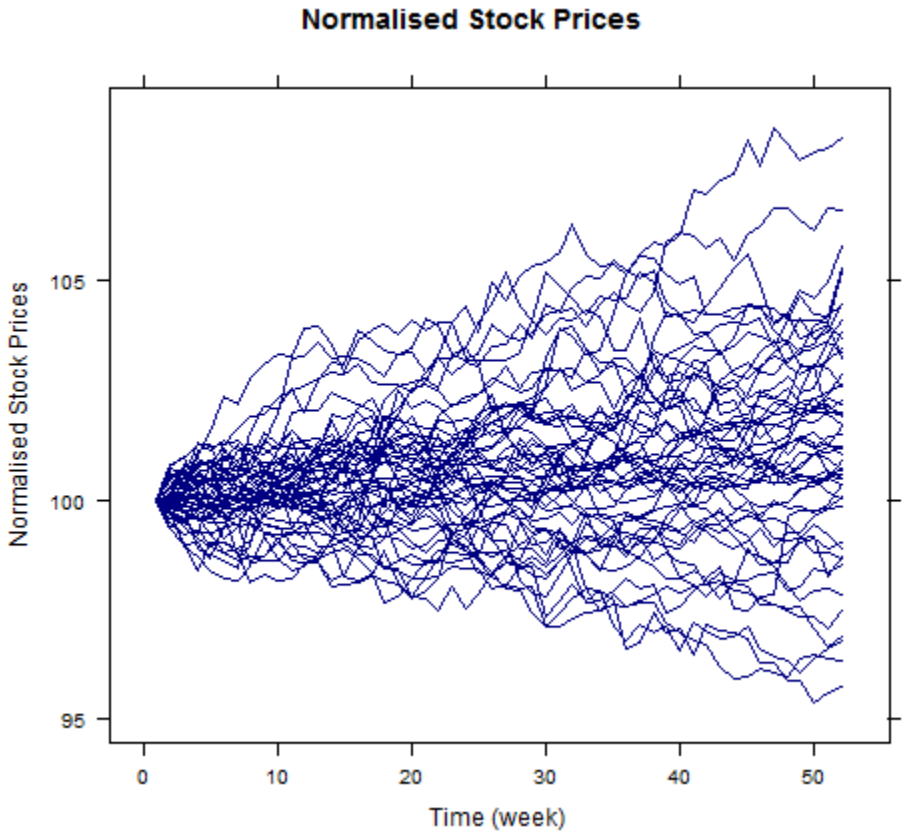
```
xyplot(Conc ~ Time | Sex,
       groups = Subject, type = "b",
       data = pkData, col = "red", pch = 16,
       main = "Concentration vs Time by Gender",
       xlab = "Time (Hours)", ylab = "Concentration"
)
```



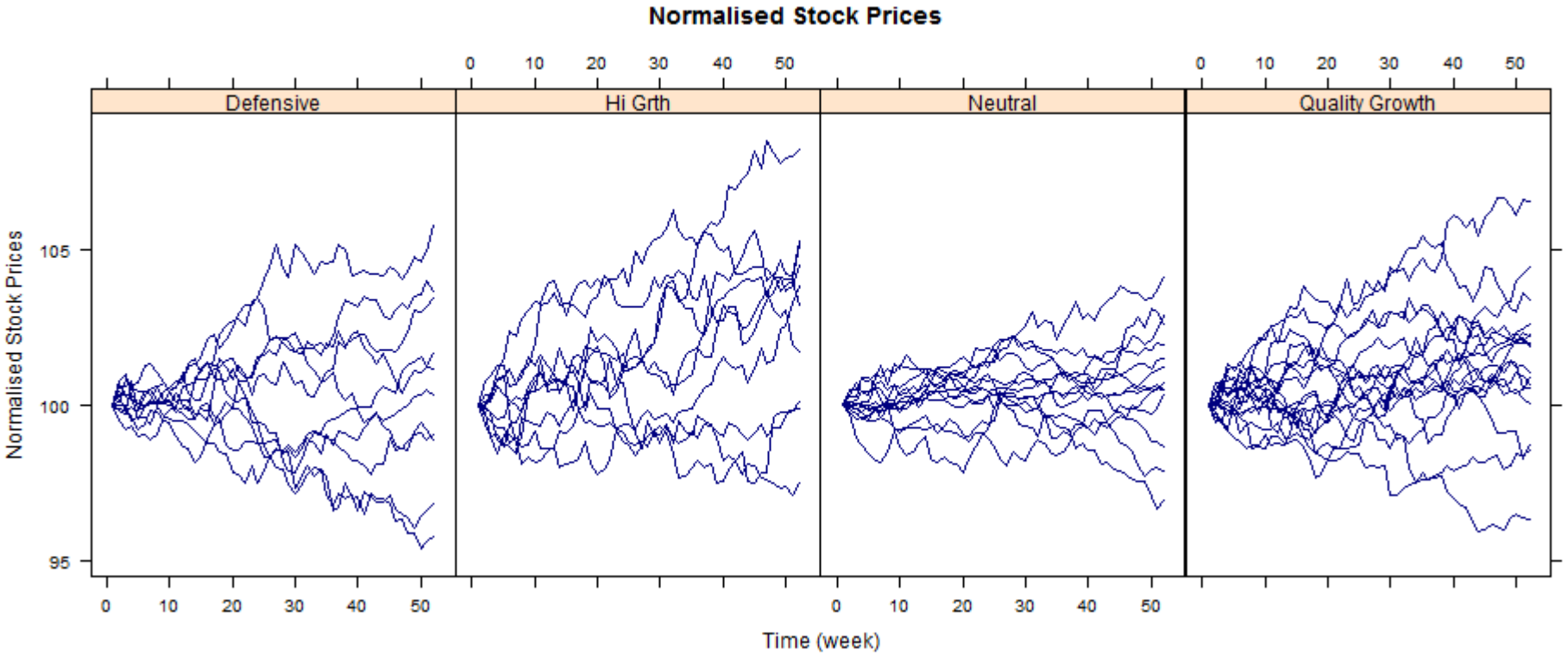
```
xyplot(Conc ~ Time | Sex,
       groups = Dose, type = "b",
       ID = pkData$Subject,
       panel = panel.superpose,
       panel.groups = function(x, y, subscripts, ID, ...) {
         panel.xyplot(x, y, groups = ID[subscripts],
                      subscripts = 1:length(x), ...)
       },
       data = fullPk
)
```



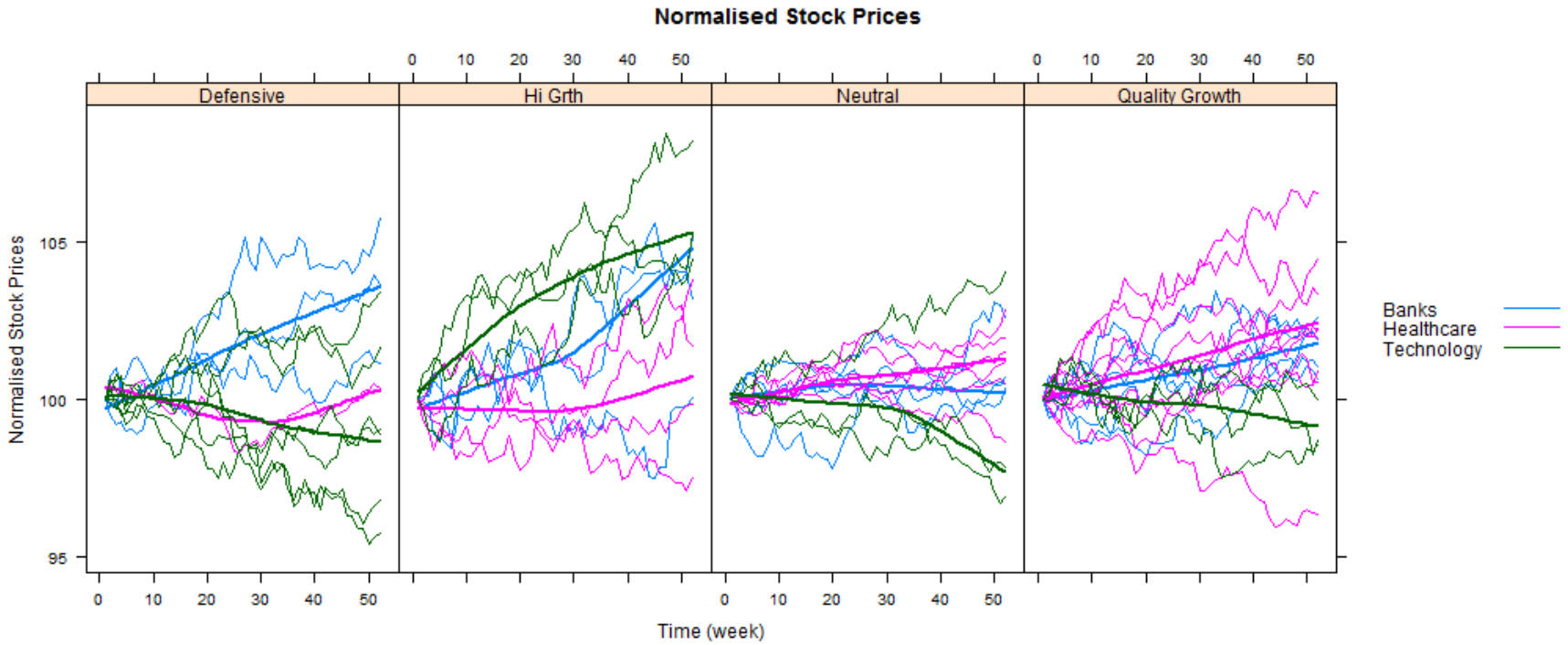
```
xyplot(Return ~ Time, data = genDf, type = "l", groups = Stock,
       col = "navy", xlab = "Time (week)", ylab = "Normalised Stock Prices",
       main = "Normalised Stock Prices")
```



```
xyplot(Return ~ Time | Style, data = genDf, type = "l", groups = Stock,
       col = "navy", xlab = "Time (week)", ylab = "Normalised Stock Prices",
       main = "Normalised Stock Prices")
```



```
xyplot(Return ~ Time | Style, data = genDf, type = "l", groups = Sector,
      Stock = genDf$Stock, panel = panel.superpose,
      xlab = "Time (week)", ylab = "Normalised Stock Prices",
      panel.groups = function(x, y, Stock, subscripts, ..., lwd = NULL) {
         panel.loess(x, y, lwd = 2, ...)
         panel.xyplot(x, y, groups = Stock [ subscripts ],
                      subscripts = 1:length(x), lwd = .8, ...)
      },
      main = "Normalised Stock Prices", layout = c(4, 1),
      auto.key = list(space = "right", lines = T, points = F))
```



MANGOSOLUTIONS
data analysis that delivers

Summary

MANGOSOLUTIONS
data analysis that delivers

Summary

- Lattice is a great framework to use when your data is highly structured
- We can use the “panel.groups” structure in order to ensure a “base” grouping variable is always taken into account
- Slides, Data and Scripts will be at LondonR.org soon

MANGOSOLUTIONS
data analysis that delivers

Questions?