Looking through Lattice

A response to Andreas Krause

Martin Fink

Modeling & Simulation,
Novartis Pharma AG

My Background

… is Matlab…

- Maths/Physics/Physiology
- Matlab user for more than a decade…
- Novartis and R for 2½ years

- R is still a mystery to me
  - Little consistency (if not chaos)
  - Documentation is illegible
  - "…" is great but what arguments can I give?

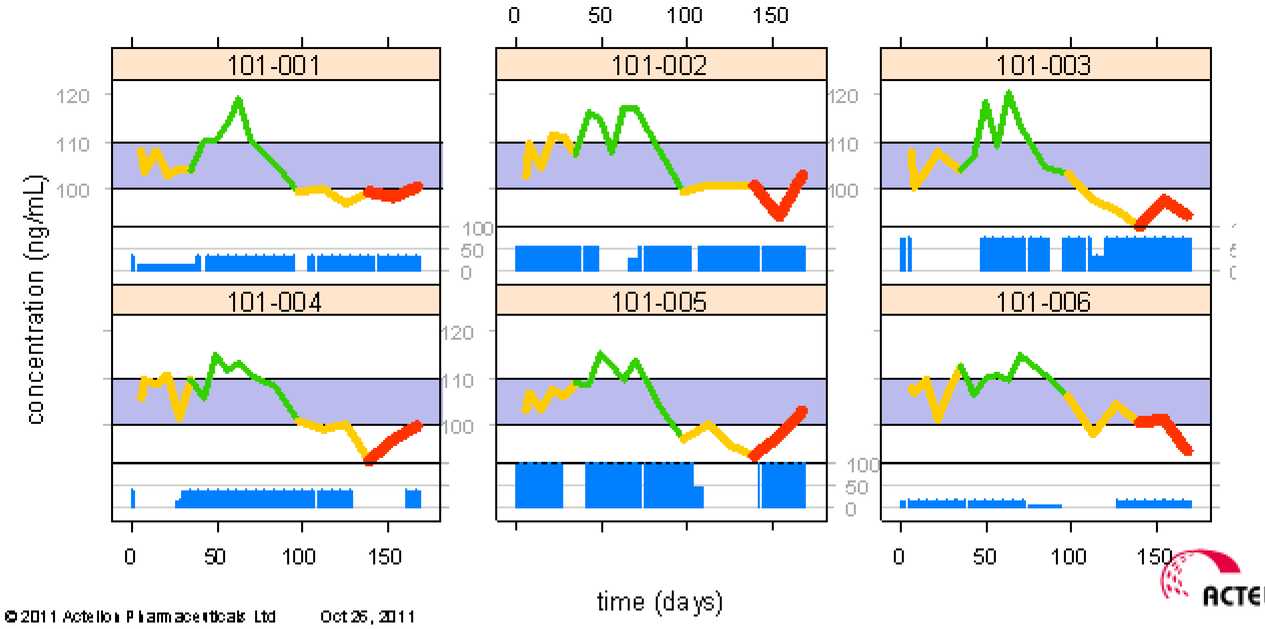
Reprise from Andreas' talk

Can one plot several variables without using [subscripts]?

PANEL FUNCTIONS

USE OF ADDITIONAL DATA

- Biomarker over time by patient
- The color indicates the disease status (red-progressive, yellow-stable, green-response to treatment)
- The additional data is the disease status, the normal range, and the dose history

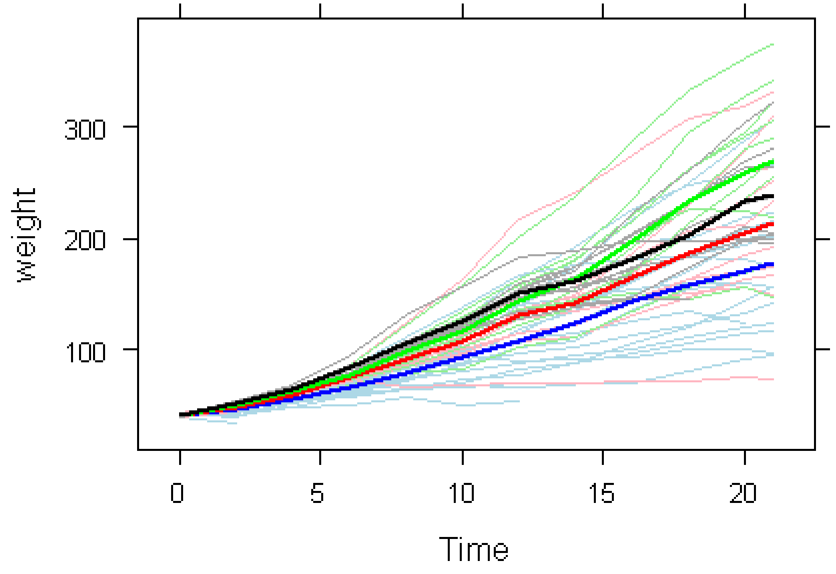


13 © 2011 Actelion Pharmaceuticals Ltd Oct 26, 2011

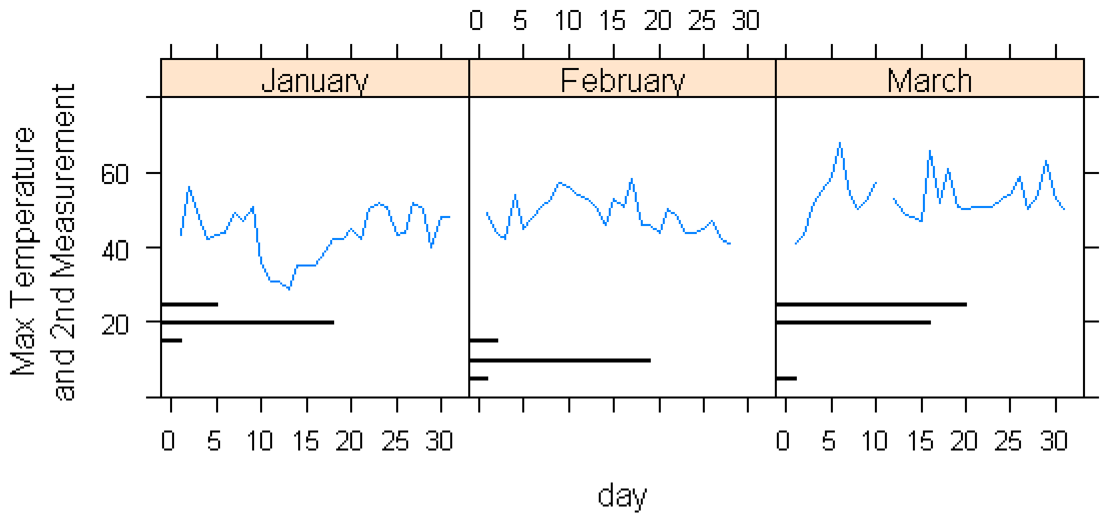
Goal: 2 examples

How can we create the following graphs?

- One measurement
  – grouped + mean



- Two measurements
  – overlaid on each panel



Overview

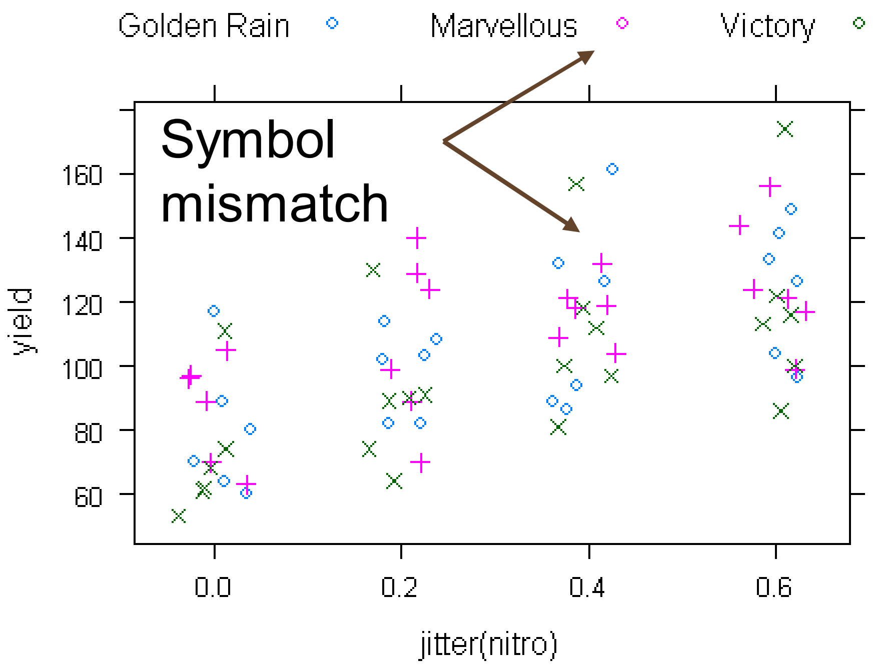
Lattice and Lattice Extra

- Some minor but important tricks
  - Customizing Output Parameters versus Themes
  - Predefine recurring plotting functions
- panel.superpose, panel.groups
- Plotting various information on one measurement
- Data structures (long & wide)
- Plotting multiple measurements on each panel

Customizing output parameters

Problems with legends…

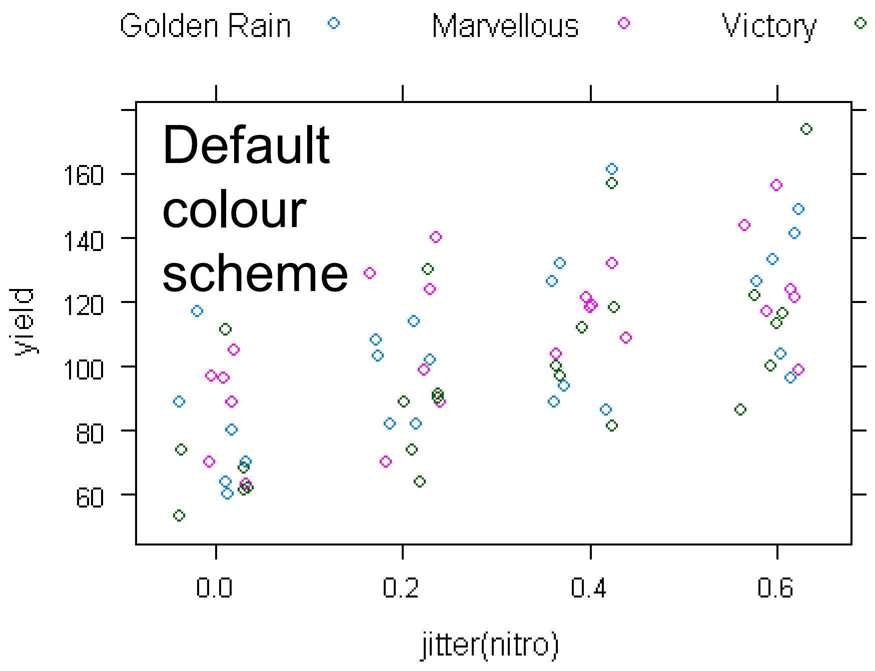
- When defining colours, line styles, etc., directly the legends are not updated accordingly
  - xyplot(yield ~ jitter(nitro), data=Oats, group=Variety, auto.key=list(columns=3), pch=c(1,3,4))

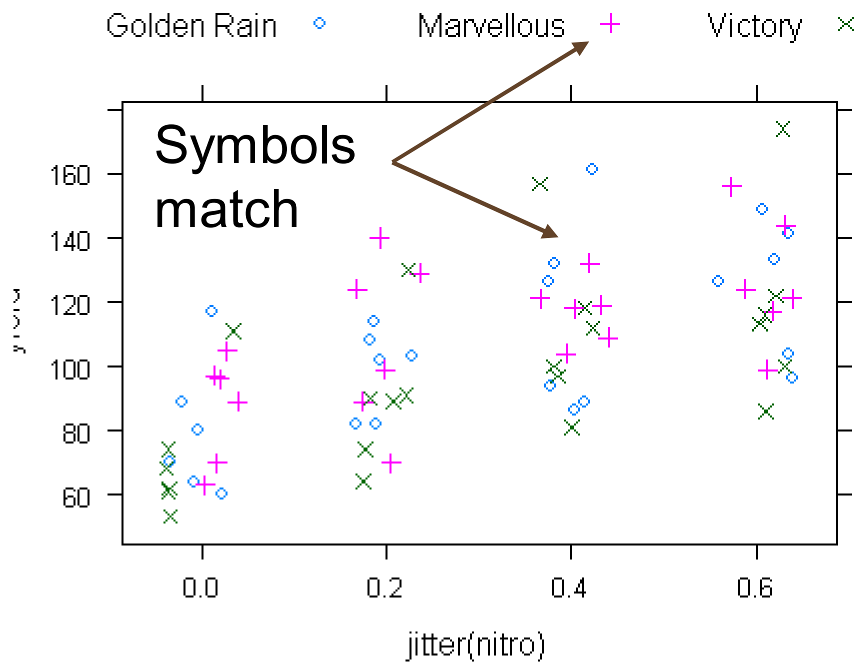


Themes

Legends are correct immediately

- Coloured (default) & B/W given by Lattice
  - myColorPars <- standard.theme('pdf')
    - see also show.settings()
  - xyplot(yield ~ jitter(nitro), data=Oats, group=Variety,
    auto.key=list(columns=3), par.settings=myColorPars)
  - myColorPars$superpose.symbol$pch <- c(1,3,4); xyplot(…)





Predefine customized plotting functions

myPlot <- function(…) {xyplot(…, myParameterSettings)}

- Default: Messy
  - xyplot(yield ~ jitter(nitro), data=Oats, group=Variety,
        auto.key=list(columns=3), par.settings=myColorPars)
  - xyplot(nitrate ~ jitter(nitro), data=Oats, group=Variety,
        auto.key=list(columns=3), par.settings=myColorPars)
  - xyplot(phosphate ~ jitter(nitro), data=Oats, group=Variety,
        auto.key=list(columns=3), par.settings=myColorPars)
- Pre-defined: Cleaner code and easier to maintain

  -
    ```
    myPlot <- function(...) {
        xyplot(..., data=Oats, group=Variety, auto.key=list(columns=3),
                    par.settings=myColorPars)
    }
    ```
  - myPlot(yield ~ jitter(nitro))
  - myPlot(nitrate ~ jitter(nitro))
  - myPlot(phosphate ~ jitter(nitro))

Introduction panel functions

Customizing what is plotted in each panel

- Default (without and with groups)
  - xyplot(y ~ x | strat, data=data, panel=panel.xyplot) # without groups
  - xyplot(y ~ x | strat, data=data, group=grp, panel=panel.superpose, panel.groups=panel.xyplot) # with groups
- panel.superpose calls panel.groups separately for each group
- One can re-define both panel and panel.groups
  - … using any of the predefined panel-functions (panel.xyplot, panel.abline, panel.histogram, panel…)
  - … or using a user-defined function that calls the panel-functions

```
panel = function(x, y, ...) {
        panel.abline(2, 1)
        panel.xyplot(x, y, ...)
        panel.average(x, y, ..., horizontal=FALSE)
}
```

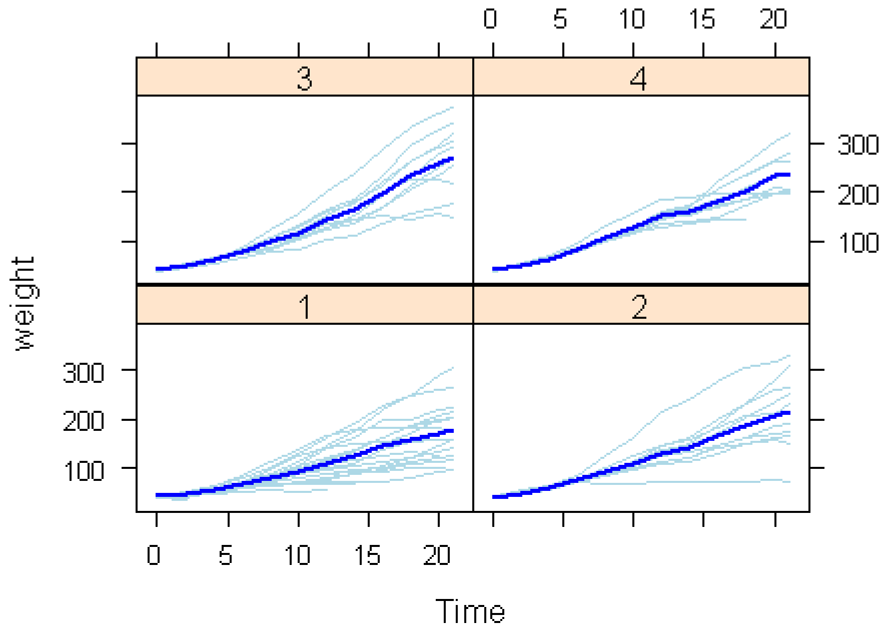
- Slides starting from here use abbreviated code
- For details please refer to the appendix

One measurement – grouped + mean

Redefining panel function for supposition based on "same" information

- Plotting individual dynamics plus mean over time
  - ***xyplot(***

```
        weight ~ Time | Diet, group=Chick, ...,
        panel=function(x, y, ...) {
                panel.xyplot(x, y, ...)
                panel.average(x, y, ...)
        }
)
```



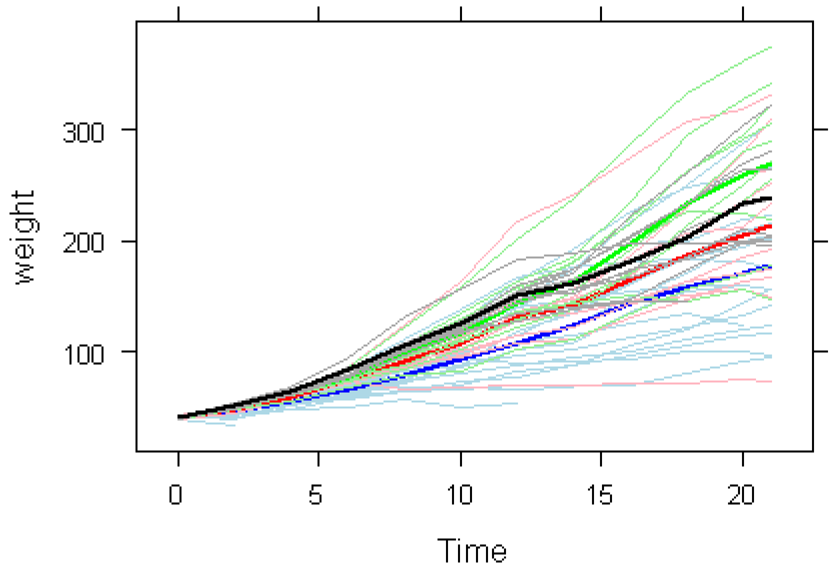
One measurement – grouped + mean

Plot all groups within one panel – with just panel.groups not ideal

- Now we group according to the diet
  ... but still want to plot lines for each individual
  - xyplot(weight ~ Time, group=Diet, ID=ChickWeight$Chick, ...,

```
          panel=panel.superpose,
          panel.groups=function(x, y, ..., ID) {
                    panel.xyplot(x, y, group=ID, ...)
                    panel.average(x, y, ...)
          }
  )
```

- Individual lines cover the means – not ideal!



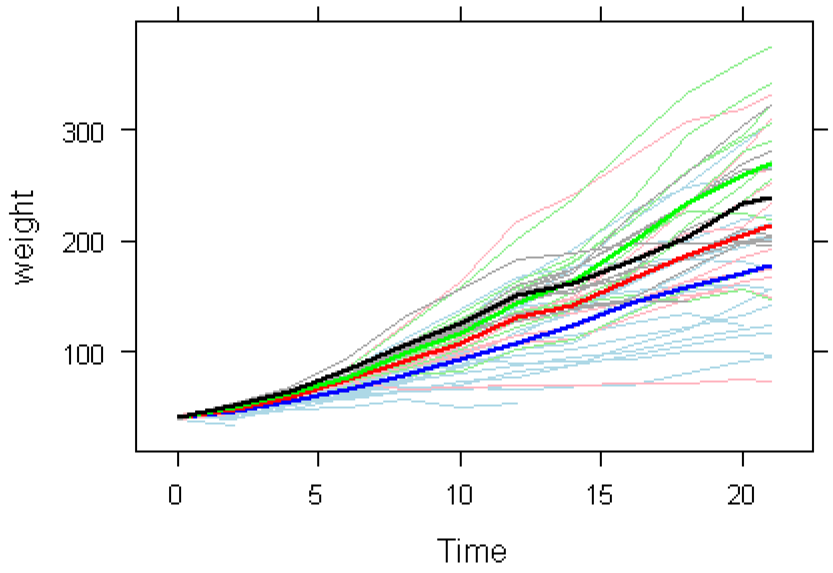
One measurement – grouped + mean

Plot all groups within one panel – first individuals and means on top

- Same goal but now we first plot all lines for individuals then all means

  -
    ```
    xyplot(weight ~ Time, group=Diet, ID=ChickWeight$Chick, ...,
            panel=function(x, y, ..., ID) {
                    panel.superpose(x, y, ..., ID=ID,
                            panel.groups=panel.xyplot(x, y, ..., group=ID))
                    panel.superpose(x, y, ...,
                            panel.groups=panel.average)
            }
    )
    ```

- Now the order of plotting is correct



Two measurements – Possible Data Structures

2 sets of data: Separate, Wide, Long Structures

- Possibilities to maintain two sets of data
  - 2 separate data.frame(s)
  - 1 wide data.frame (good for correlation plots)
  - 1 long data.frame (good if very different sampling times)

- With separate data.frame(s): LatticeExtra but not Lattice!
- Switch between wide and long format using *reshape()*
  - Once through the help and with a working example it's great!

Two measurements overlaid (wide format)

Scaling, different types, colours, …

- Using the same panel-function for all measurements:
  => Relatively easy
  - SeatacWeather$rain <- 45.2 * SeatacWeather$precip # scaling
  - xyplot(min.temp + max.temp + rain ~ day | month, ...,
    type = c("l", "l", "h"), distribute.type = TRUE)

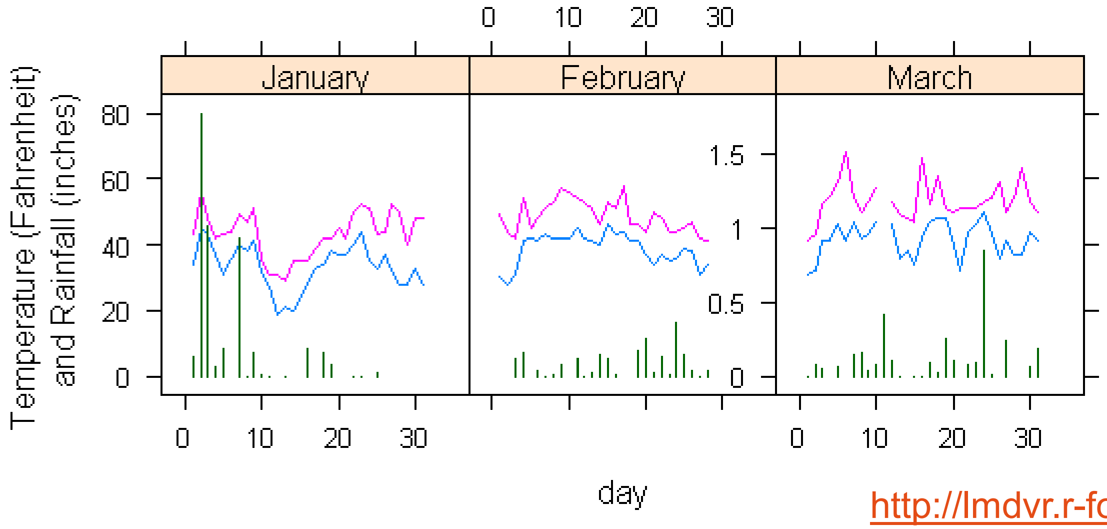
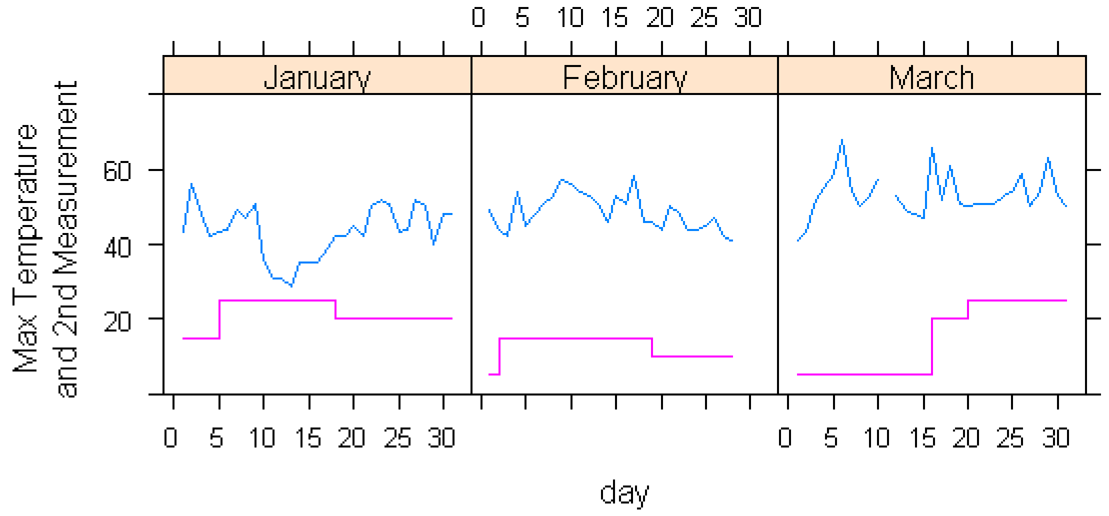


Figure 5.13
http://lmdvr.r-forge.r-project.org/figures/figures.html

Two measurements overlaid (long format)

Different amount of time points available => long format

- Long format: DV contains values, CMT denotes variables
- As easy as for the wide format – using distributed "type"
  - xyplot(DV ~ day | month, ..., group=CMT,
    type = c("l", "s"), distribute.type=TRUE)
- Advantage to wide format:
  No hard-coding of variable names & number of variables



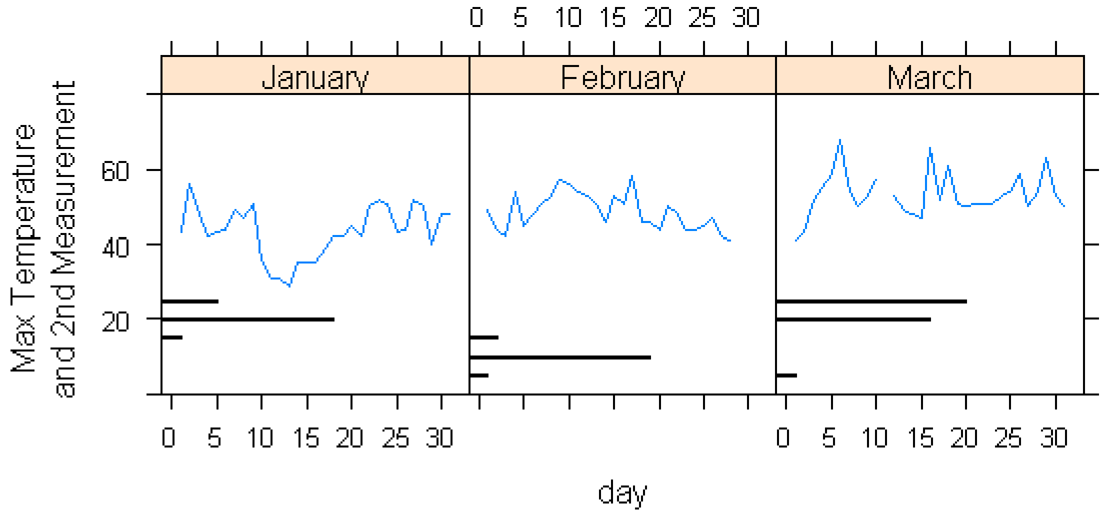
Two measurements overlaid (long format)

Different amount of time points available => long format

- Same data, but using different panel function for each variable

  -
    ```
    panel.data <- list()
    panel.data[[1]] <- panel.xyplot
    panel.data[[2]] <- panel.barchart
    ```

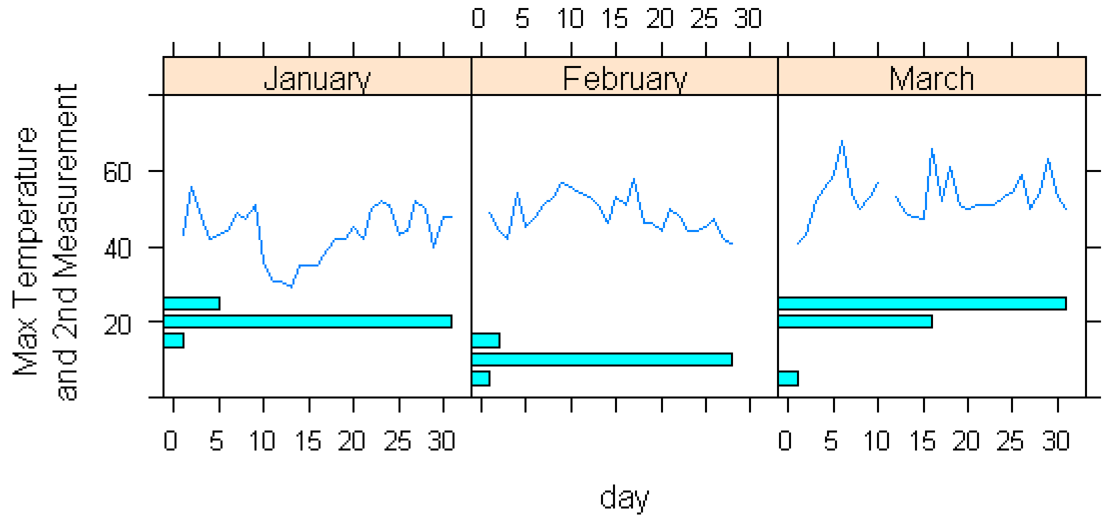
  -
    ```
    mySecPlot(DV ~ day | month,
            panel=panel.superpose,
            panel.groups=function(x,y, ..., group.number) {
                        panel.data[[group.number]](x,y,...)
            }
    )
    ```



Two measurements overlaid (any format)

With latticeExtra everything is possible ☺

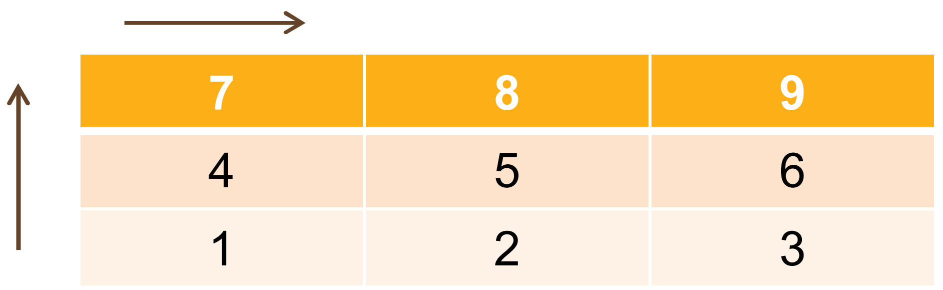
- LatticeExtra is very flexible
- Can also overlay plots from two different data sets!
  - xyplot(DV ~ day | month, data=PD, ...) +
  - as.layer(barchart(DV ~ day | month, data=Dose, ...), y.same = FALSE, axes=NULL)   # again: scale as necessary
- Overlay any plots as long as they have the same panels…



Information on groups and panels

Functions and parameters for customizing appearance…

- Arguments to panel.groups provide information on which group to plot
  - panel.groups=function(x,y, ..., group.number, group.value)
- Within panel, strip, axis,... functions additional information on the panel location useful
  - current.row(prefix)
  - current.column(prefix)
  - panel.number(prefix)
  - packet.number(prefix)
  - which.packet(prefix)



7	8	9
4	5	6
1	2	3

- panel ≈ packet
- prefix: if more than one plot per page – defaults to last plot

Conclusions

Good and better ways but none optimal

- “Help” difficult to follow
  - Not everything is documented
  - Difficult to follow the … arguments in functions
- Lattice has some inherent inconsistencies
- Predefine customized functions!
- panel.superpose and panel.groups can be handy to use

Documentation

Freely available in the internet

- Most helpful tutorial/examples I found so far
  - http://lmdvr.r-forge.r-project.org/figures/figures.html
- Some more documentation on lattice and underlying grid
  - http://www.stat.auckland.ac.nz/~paul/RGraphics/chapter4.pdf
  - http://www.stat.auckland.ac.nz/~paul/RGraphics/chapter5.pdf
- Good overview of latticeExtra
  - http://latticeextra.r-forge.r-project.org/#

Appendix

Slides including the detailed code

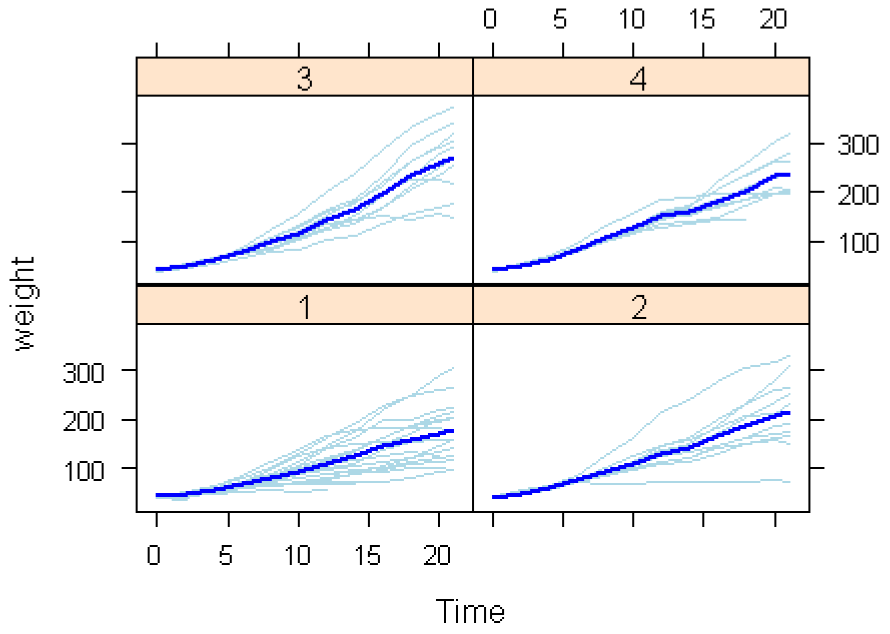
- Libraries & Data used
  - library(lattice) – one example with library(latticeExtra)
  - data(Oats, package = "MEMSS")
  - data(ChickWeight)
  - data(SeatacWeather, package = "latticeExtra")
  - PKPD = PD (SeatacWeather) + Dose (random second measurement)
    see respective slide for details

One measurement – grouped + mean

Redefining panel function for supposition based on "same" information

- Plotting individual dynamics plus mean over time
  - ***xyplot(***

```
        weight ~ Time | Diet, data=ChickWeight, group=Chick, type='l',
        panel=function(x, y, col=NULL, ...) {
                panel.xyplot(x, y, col='lightblue', ...)
                panel.average(x, y, col='blue', lwd=2, horizontal=F, ...)}
)
```

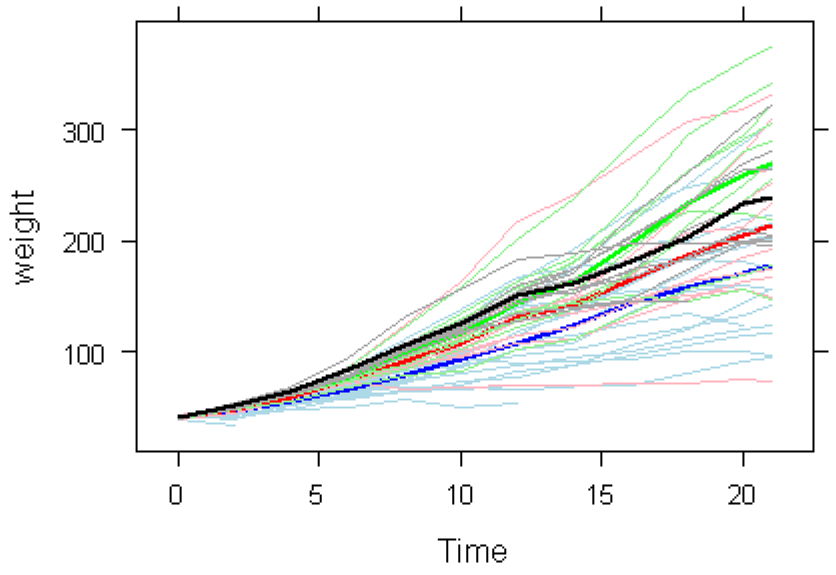


One measurement – grouped + mean

Plot all groups within one panel – with just panel.groups not ideal

- colStrLight <- c('lightblue','lightpink','lightgreen','gray65') # individuals
  colStr <- c('blue','red','green','black') # means
- myPlot <- function(...) {xyplot(..., data=ChickWeight, group=Diet, ID=ChickWeight$Chick, col=colStr, col.line=colStrLight, type='l')}
- myPlot(weight ~ Time, panel=panel.superpose, panel.groups=function(x, y, ..., col, col.line, lwd=1, ID) {

```
                    panel.xyplot(x, y, col.line=col.line, group=ID, ...)
                    panel.average(x, y, col.line=col, lwd=2, horizontal=F, ...)
      }
)
```

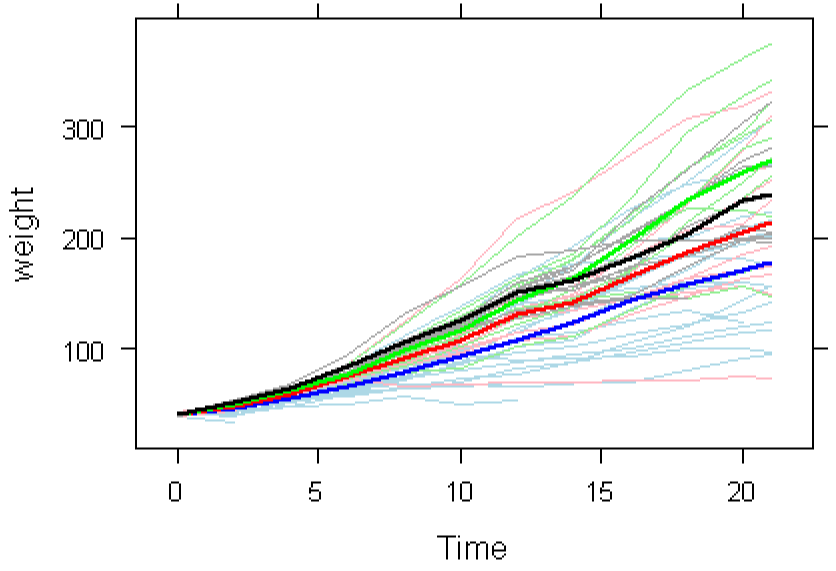


One measurement – grouped + mean

Plot all groups within one panel – first individuals and means on top

- panel.groupPID=function(x, y, ..., ID) {panel.xyplot(x, y, ..., group=ID)}
- myPlot(weight ~ Time,

```
        panel=function(x, y, ..., ID, col, col.line, panel.groups) {
                panel.superpose(x, y, ..., col=colStrLight, ID=ID,
                        panel.groups=panel.groupPID)
                panel.superpose(x, y, ..., col=colStr, lwd=2, horizontal=F,
                        panel.groups=panel.average)
        }
)
```



Two measurements overlaid (wide format)

Scaling, different types, colours, …

- Using the same panel-function for all measurements:
  => Relatively easy
  - SeatacWeather$rain <- 80 * SeatacWeather$precip /
    max(SeatacWeather$precip, na.rm = TRUE) # scaling
  - xyplot(min.temp + max.temp + rain ~ day | month,
    data = SeatacWeather, lty = 1, ylab = "Temperature and Rainfall",
    type = c("l", "l", "h"), distribute.type = TRUE)

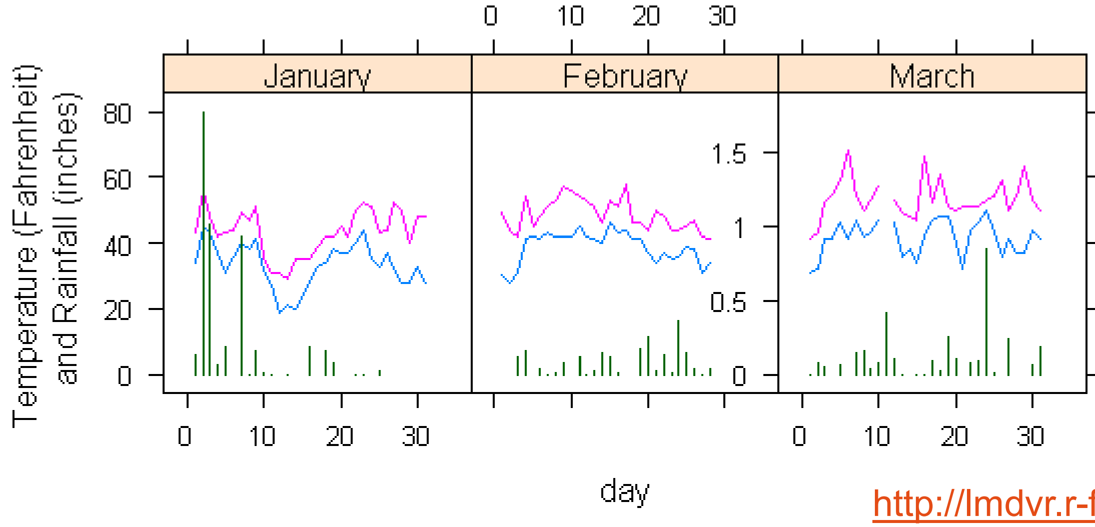


Figure 5.13
http://lmdvr.r-forge.r-project.org/figures/figures.html

Customized data for this presentation

Creation of PKPD = rbind(PD, Dose)

- Creation of random second measure
  - PD <- SeatacWeather
  - PD$DV <- PD$max.temp # 1st measurement
  - PD$CMT <- 1 # indicator for 1st measurement
  - set.seed(1234) # ensure reproducability
  - Dose <- do.call(rbind, lapply(split(PD, PD$month),

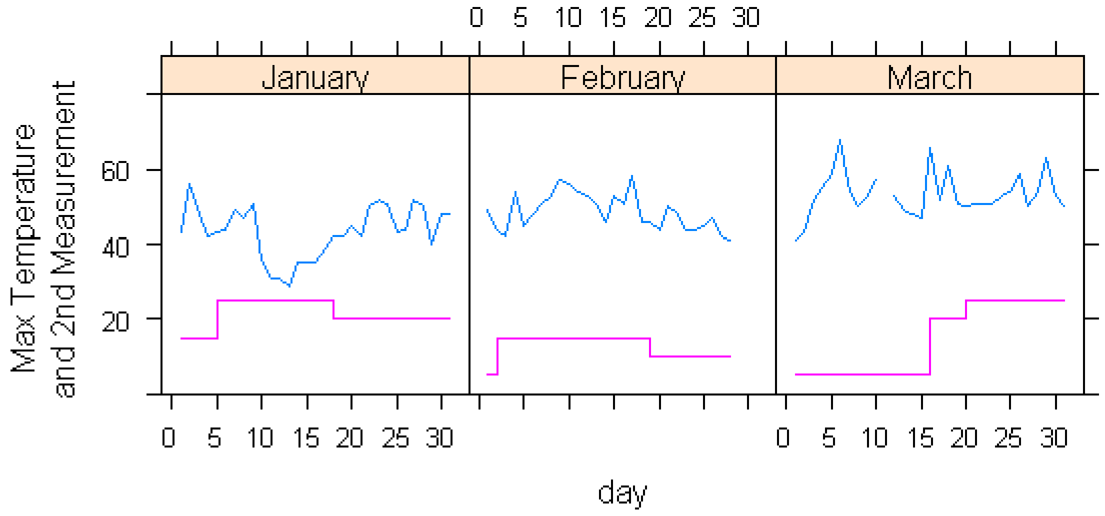
```
        function(X) {
                days <- sample.int(27,2)+1
                Y <- X[c(1,1,1,1),]
                Y$day <- c(1, sort(days), max(X$day))
                Y$DV <- sample(c(0:5)*5,3)[c(1:3,3)] # 2nd measurement
                Y$CMT <- 2        # indicator for 2nd measurement
        return(Y)
        }))
```

  - PKPD <- rbind(PD,Dose)

Two measurements overlaid (long format)

Different amount of time points available => long format

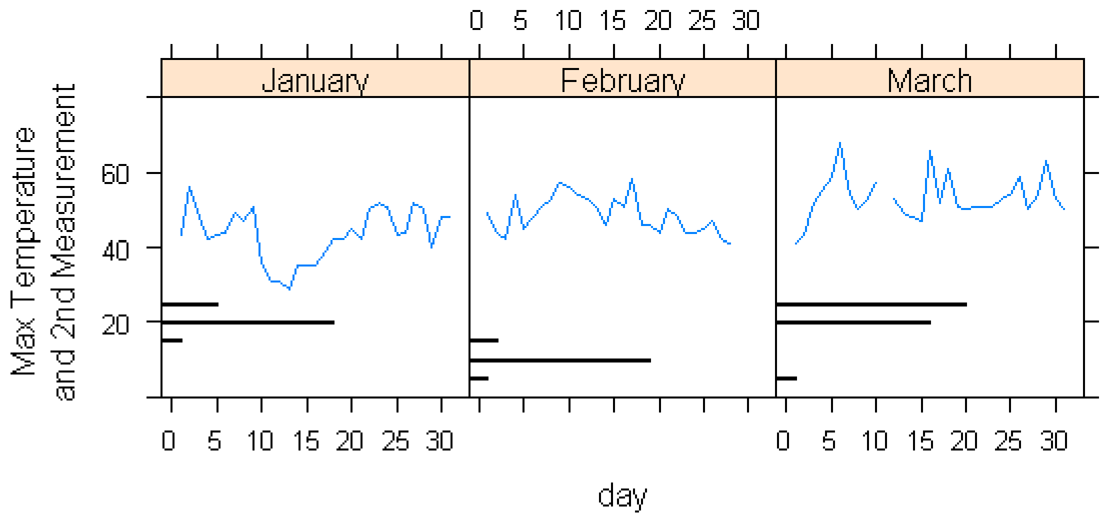
- DV contains value to draw, CMT denotes measurements
  - mySecPlot <- function(…) {
    xyplot(…, data = PKPD, group=CMT, lty = 1,
    ylab = "Max Temperature \n and 2nd Measurement",
    type = c("l", "s"), distribute.type=TRUE)
    }
  - mySecPlot(DV ~ day | month)



Two measurements overlaid (long format)

Different amount of time points available => long format

- Same data as before
  - panel.data <- list()
    panel.data[[1]] <- panel.xyplot
    panel.data[[2]] <- panel.barchart
  - mySecPlot(DV ~ day | month, panel=panel.superpose,
            panel.groups=function(x,y, ..., group.number) {
                      panel.data[[group.number]](x,y,...)
            }
)

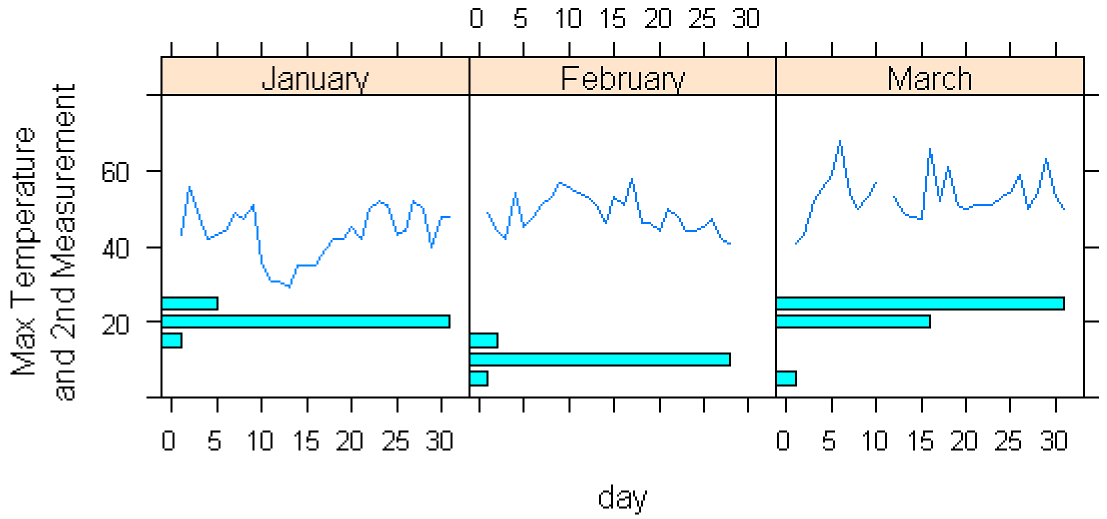


Two measurements overlaid (any format)

With latticeExtra everything is possible ☺

- require(latticeExtra)
- xyplot(DV ~ day | month, data=PD, lty = 1, type = c("l"),
  ylab = "Max Temperature \n and 2nd Measurement",
  ylim=c(0,80)) +
- as.layer(barchart(DV ~ day | month, data=Dose, ylim=c(0,80/(25/5))),
  y.same = FALSE, axes=NULL) # scale as necessary

- Overlay any plots as long as they have the same panels…


January
February
March
Max Temperature
and 2nd Measurement
day