Why do exploratory data analysis?

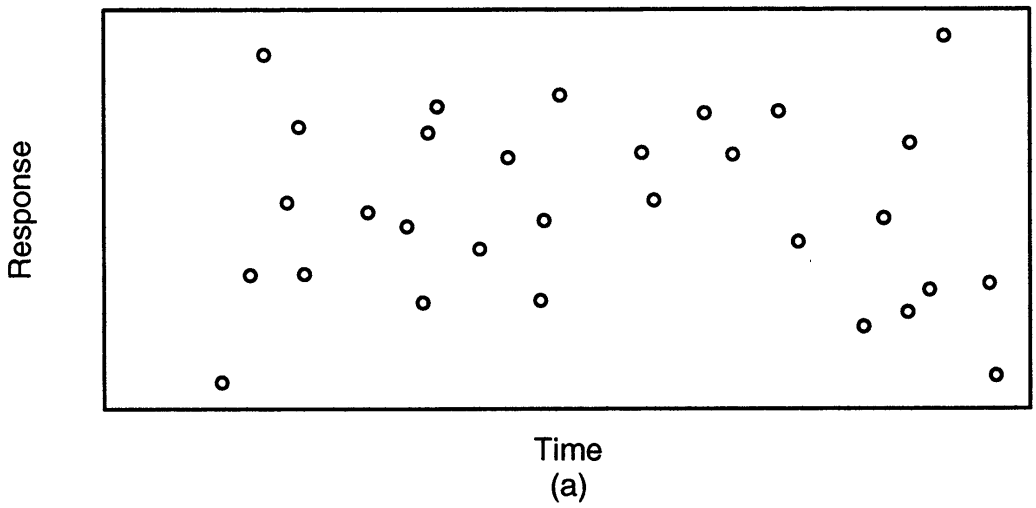
- to understand our data better
- build intuition for building inferential models
- look for outliers and data input errors
- emphasis on qualitative information about the data, not quantitative (does the mean increase, not the exact amount of the increase)

What do we look for when exploring longitudinal data?

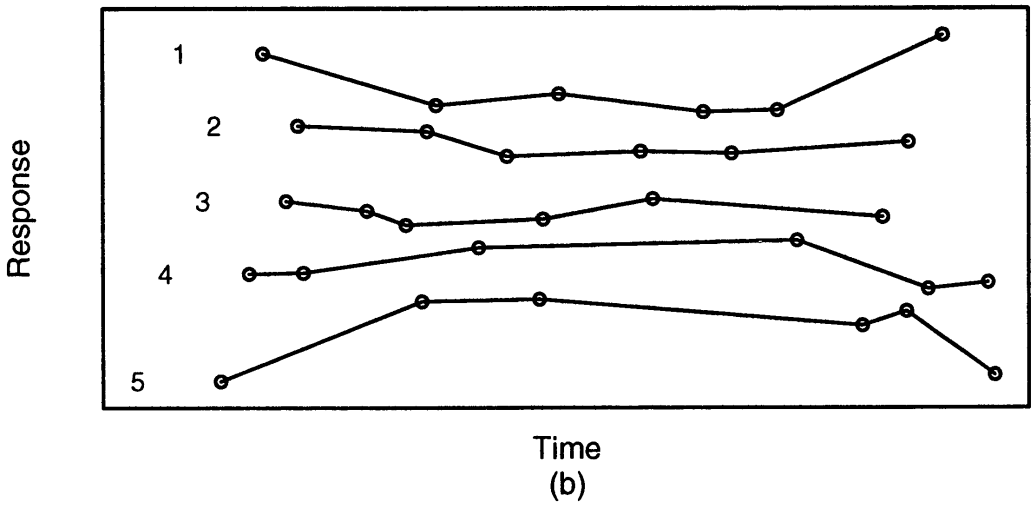
- mean, variance, correlation (between values at different times)
- and how these vary by time, and by other covariates

The primary tool for exploring longitudinal data is the profile plot. It is a plot showing how the values for each individual change over time; the individuals can all be shown together, or they can be divided up into multiple plots. However, it's important to always draw lines connecting the points for each individual.

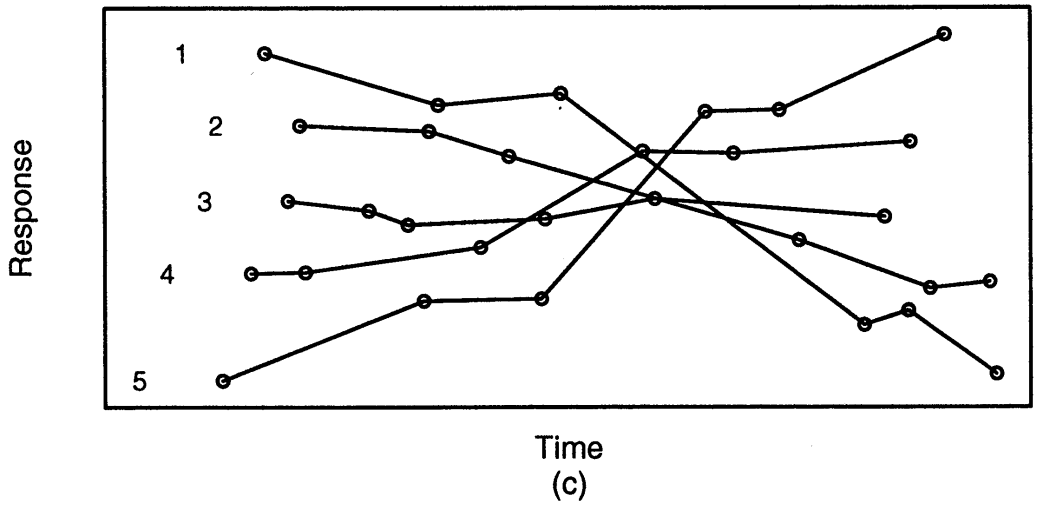
Consider the following plot, where they are not connected. How does it look like the mean and the variance are changing over time?



Now consider these plots where the lines are connected by subject, in two different possible ways. In each, how are the mean and the variance changing over time?



(b)



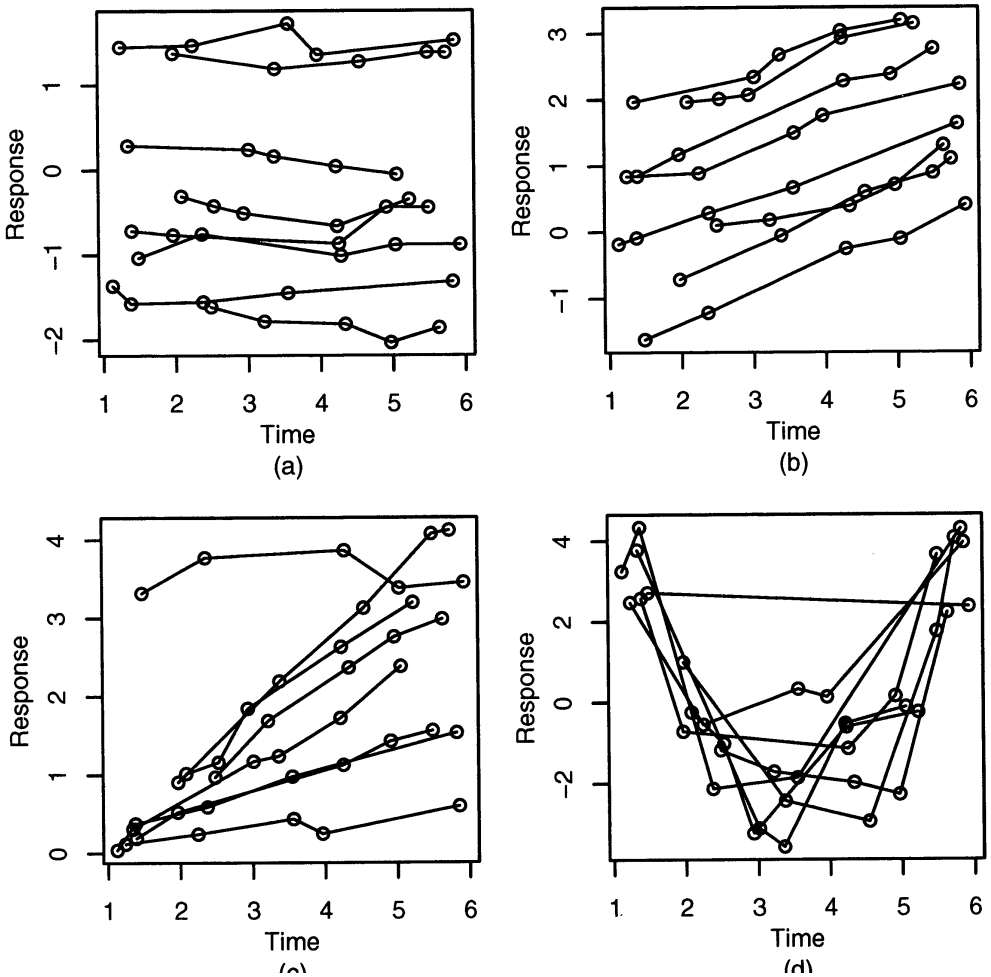
(c)

Look at the following four plots and make qualitative statements about:

Trend: Is it constant over time? Linear over time? If so, is it increasing or decreasing? Is it quadratic over time, or some other shape?

Differences between individuals: Does the intercept look the same for all (fixed) or different (random)? What about the slope (or other shape)?

Outliers: Are there any individual points different than others? Are there any individual profiles that are different than the others?



Making profile plots in R

To connect points from different subjects, we specify a `group` variable in `xyplot`; we also use the `l` type, which stands for lines. We can use the points type (`p`) as well. We also need to put the points for each subject in order by time, as the points will be connected in the order they appear in the data set. Finally, we need to remove any lines with missing values as missing values will cause breaks in the lines.

Without removing the missings, we don't get all the lines. (Not shown)

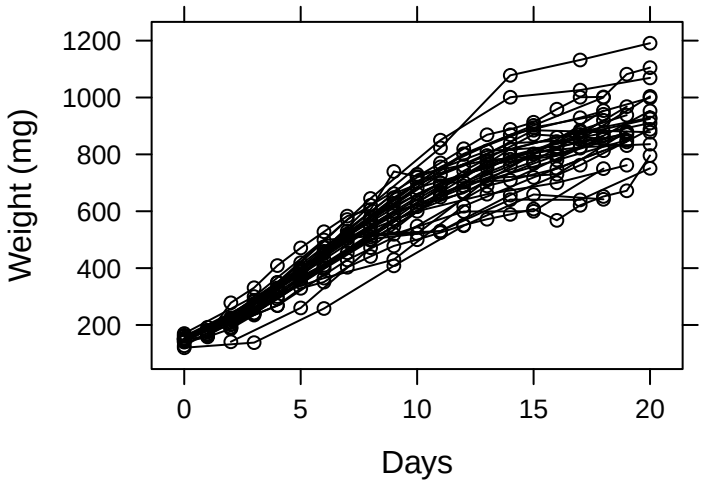
```
> library(lattice)
> library(latticeExtra)
> d <- read.delim("http://rem.ph.ucla.edu/rob/mld/data/tabdelimiteddata/bigmice.txt")
> xyplot(weight ~ day, group = id, type = c("p", "l"), data = d,
+     xlab = "Days", ylab = "Weight (mg)")
```

Here I remove the missings and order by id and day. They're already ordered in this case, but it's shown anyway. (Again, not shown.)

```
> d <- subset(d, !is.na(weight))
> d <- d[order(d$id, d$day), ]
> p1 <- xyplot(weight ~ day, group = id, type = c("p", "l"), data = d,
+     xlab = "Days", ylab = "Weight (mg)")
```

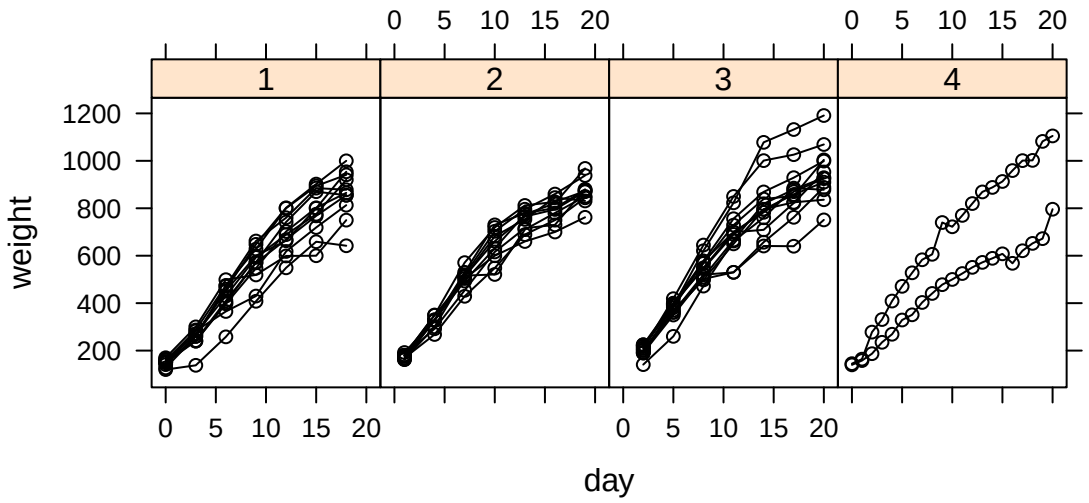
Now I make all the lines black and plot.

```
> mtheme <- standard.theme("pdf")
> mtheme$superpose.symbol$col <- "black"
> mtheme$superpose.line$col <- "black"
> p1b <- update(p1, par.settings = mtheme)
> plot(p1b)
```



That's a lot of lines on one plot. It's often very helpful to divide them up into separate plots. The `layout` option tells it to make 4 columns and 1 row of plots.

```
> p2 <- xyplot(weight ~ day | factor(group), group = id, type = c("p",
+     "l"), data = d, par.settings = mtheme, layout = c(4, 1))
> plot(p2)
```



Sample means and standard deviations

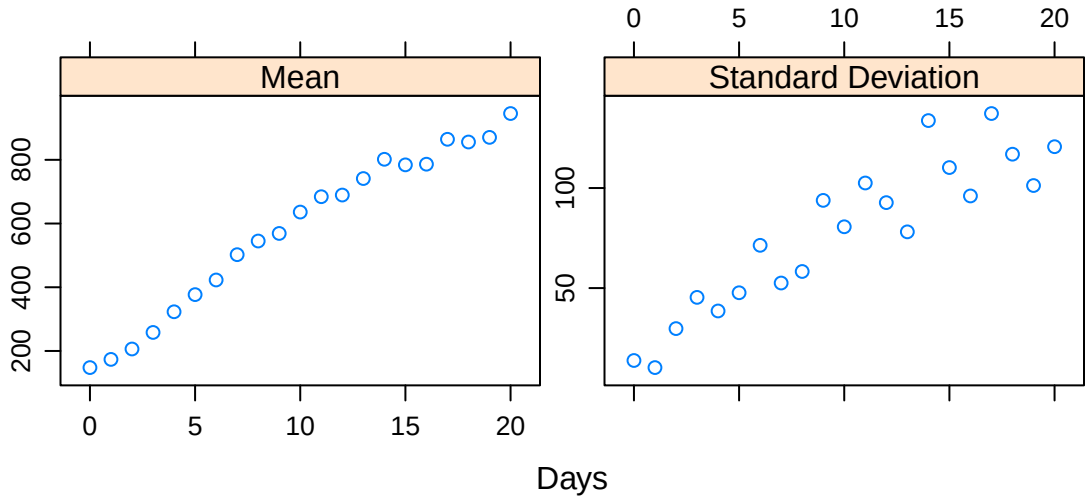
Plotting the sample means and sds by day can also be helpful; however, it can be misleading if there are missing values, as in the plots we talked about on the first couple pages. Here I use the `cast` function to get summary information for each day.

```
> library(reshape)
> d.summary <- cast(day ~ ., function(x) c(mean = mean(x), sd = sd(x),
+     n = length(x)), value = "weight", data = d)
> head(d.summary)
```

```
  day     mean       sd  n
1   0 147.9231 13.83511 13
2   1 173.6667 10.28090 12
3   2 206.2857 29.77203 14
4   3 258.2308 45.38016 13
5   4 323.1667 38.58128 12
6   5 376.9286 47.67432 14
```

To combine the plots together, I use `c`; this requires the `latticeExtra` package.

```
> s1 <- xyplot(mean ~ day, data = d.summary, xlab = "Days", ylab = NULL)
> s2 <- xyplot(sd ~ day, data = d.summary, xlab = "Days", ylab = NULL)
> s12 <- c(Mean = s1, `Standard Deviation` = s2)
> plot(s12)
```



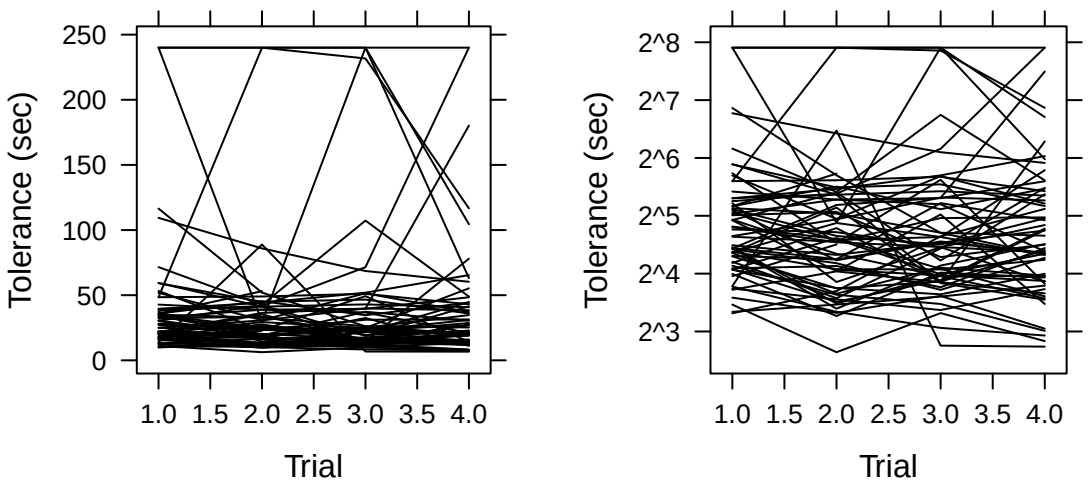
Pediatric Pain Data

The initial plot of the pediatric pain data (at left) shows that it's skewed; suggesting a log transformation; this can be done in two ways, either by making a new variable (not shown), or by plotting on the log scale.

```
> pp <- read.delim("http://rem.ph.ucla.edu/rob/mld/data/tabdelimiteddata/pain.txt")
> pp <- subset(pp, !is.na(paintol))
> p1 <- xyplot(paintol ~ trial, group = id, type = "l", data = pp,
+     par.settings = mtheme, xlab = "Trial", ylab = "Tolerance (sec)")
> pp$l2paintol <- log2(pp$paintol)
> plogv1 <- xyplot(l2paintol ~ trial, group = id, type = "l", data = pp,
+     par.settings = mtheme, xlab = "Trial", ylab = "Tolerance (log_2 sec))")
> plogv2 <- xyplot(paintol ~ trial, group = id, type = "l", data = pp,
+     par.settings = mtheme, scales = list(y = list(log = 2)),
+     xlab = "Trial", ylab = "Tolerance (sec)")
```

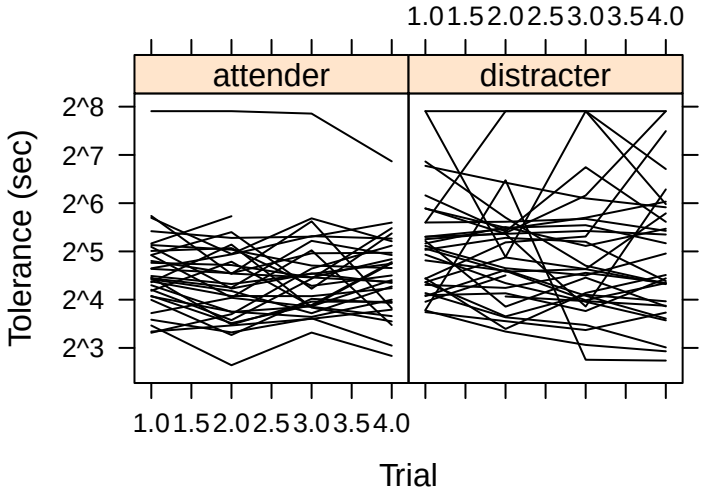
Here I combine two plots by using the `split` option, the first two numbers are the col/row this plot should be in and the last two numbers are the total number of cols/rows to use.

```
> plot(p1, split = c(1, 1, 2, 1), more = TRUE)
> plot(plogv2, split = c(2, 1, 2, 1))
```



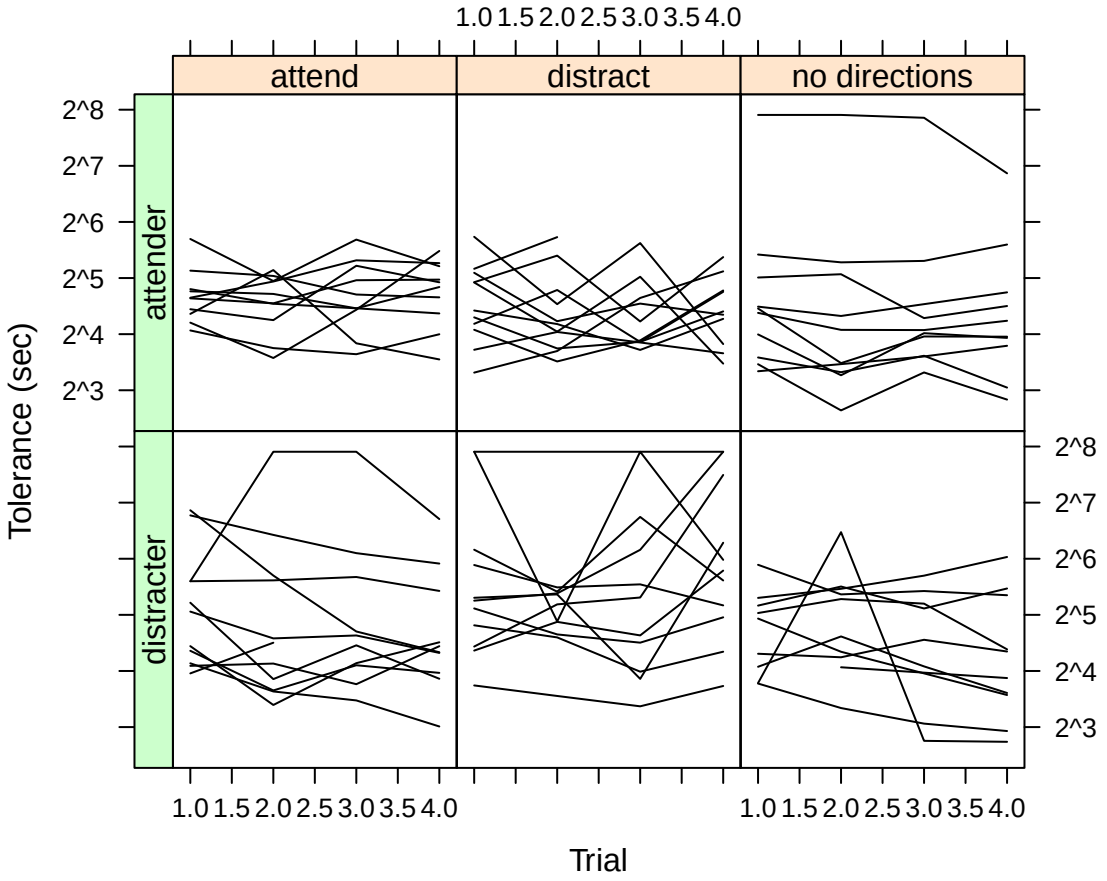
Again, that's a lot of plots; let's divide by attender/distracter.

```
> p2 <- xyplot(paintol ~ trial | cs, group = id, type = "l", data = pp,
+     par.settings = mtheme, scales = list(y = list(log = 2)),
+     xlab = "Trial", ylab = "Tolerance (sec)")
> plot(p2)
```



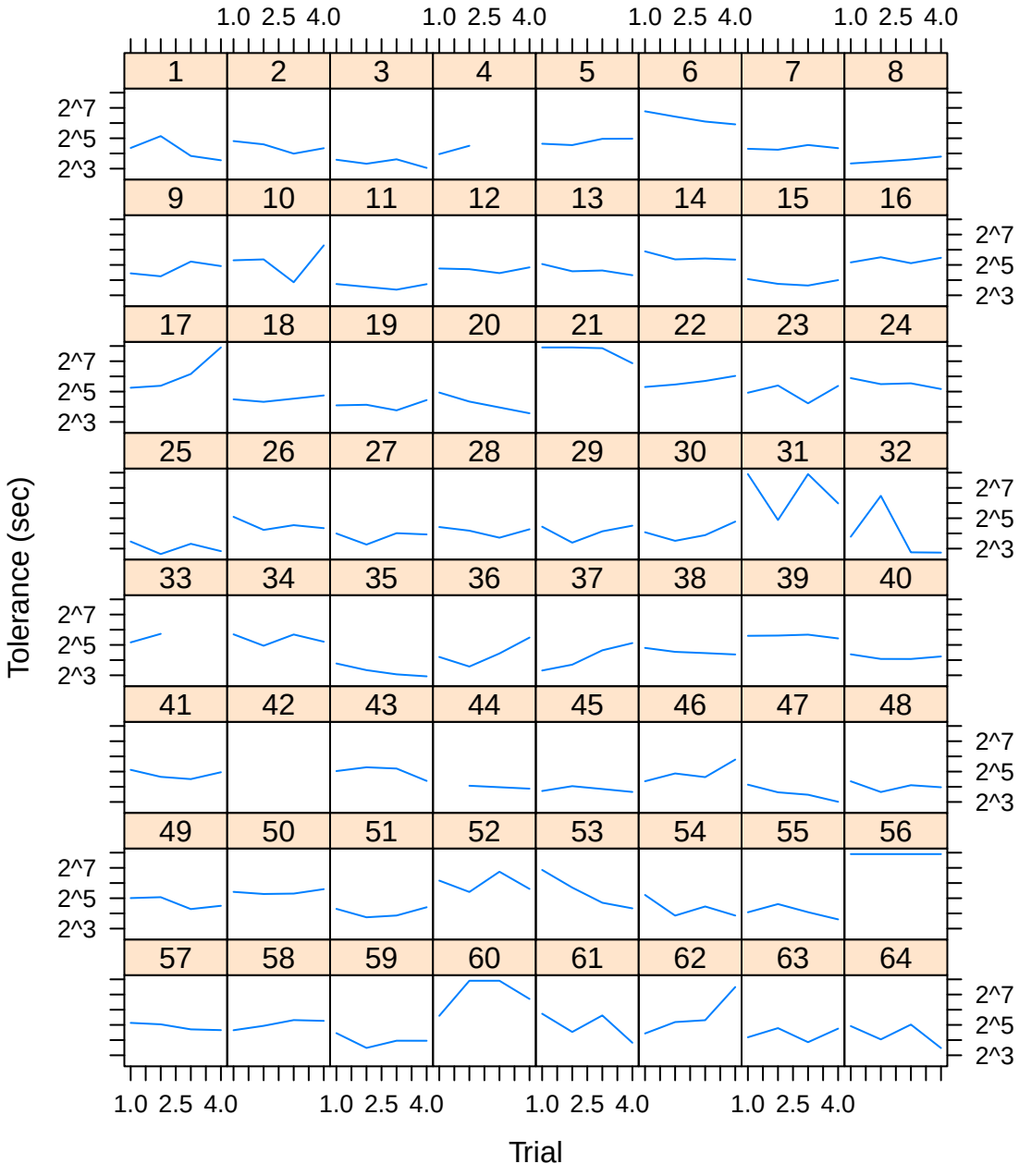
We can divide both by attender/distracter and by treatment; `useOuterStrips` is optional but makes it look nicer.

```
> p3 <- xyplot(paintol ~ trial | treatment * cs, group = id, type = "l",
+     data = pp, par.settings = mtheme, as.table = TRUE, scales = list(y = list(log =
+     xlab = "Trial", ylab = "Tolerance (sec)")
> plot(useOuterStrips(p3))
```



Or, we might want a different plot for each subject.

```
> pp$id <- factor(pp$id)
> p4 <- xyplot(paintol ~ trial | id, type = "l", data = pp, as.table = TRUE,
+     scales = list(y = list(log = 2)), xlab = "Trial", ylab = "Tolerance (sec)")
> plot(p4)
```



We can control the order of the plots by changing the order of the factor we divide the plots up by; here I use `reorder` to order the id's by the maximum value of the pain tolerance (not shown).

```
> pp$idx <- reorder(pp$id, pp$l2paintol, FUN = max)
> xyplot(paintol ~ trial | idx, type = "l", data = pp, as.table = TRUE,
+     scales = list(y = list(log = 2)), xlab = "Trial", ylab = "Tolerance (sec)")
```

This is a little fancier; I make a new id variable that also has a code for attender/distracter. I then get the max for each id using `cast`, and reorder the id factor using the `factor` command with the levels sorted first by attender/distracter and then by max (not shown).

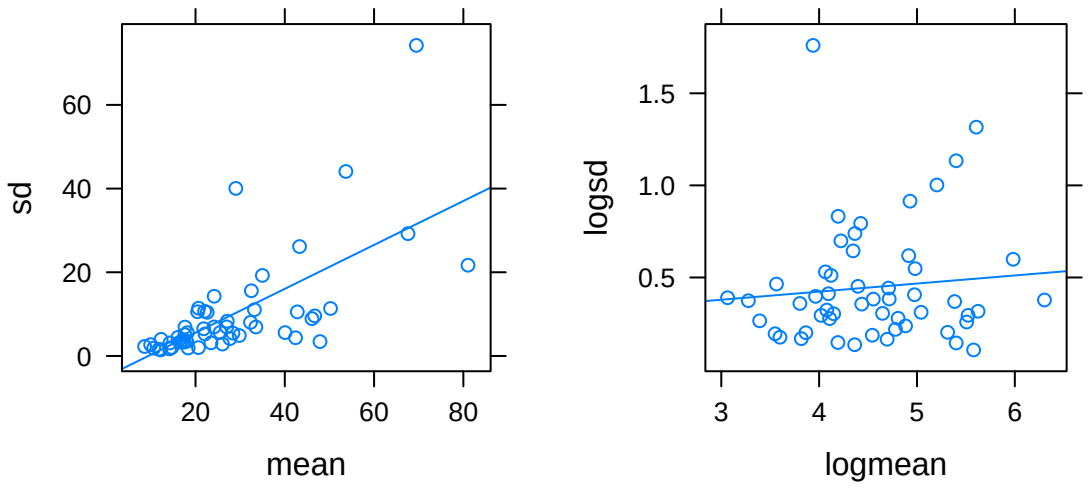
```
> pp$idcs <- factor(paste(pp$id, substr(pp$cs, 1, 1)))
> px <- cast(idcs + cs ~ ., data = pp, value = "l2paintol", fun.aggregate = max)
> pp$idcs <- factor(pp$idcs, levels = px$idcs[order(px$cs, px$`(all)`)])
> xyplot(paintol ~ trial | idcs, type = "l", data = pp, as.table = TRUE,
+     scales = list(y = list(log = 2)), xlab = "Trial", ylab = "Tolerance (sec)")
```

Means and Standard deviations, by subject:

```
> pps <- cast(id ~ result_variable, data = pp, value = "paintol",
+     fill = NA, fun.aggregate = function(x) {
+         c(mean = mean(x), logmean = mean(log2(x)), sd = sd(x),
+             logsd = sd(log2(x)), n = length(x), n240 = sum(x >=
+                 240))
+     })
> head(pps)
```

```
  id    mean  logmean        sd     logsd n n240
1  1 20.4650 4.222419 10.569798 0.6990595 4    0
2  2 22.1275 4.435723  5.260858 0.3554732 4    0
3  3 10.6300 3.392372  1.867958 0.2646742 4    0
4  4 19.1050 4.230167  5.055813 0.3863368 2    0
5  5 27.7125 4.780160  4.148593 0.2182594 4    0
6  6 81.0400 6.302907 21.692587 0.3778762 4    0
```

```
> pps <- subset(pps, n >= 4 & n240 == 0)
> p1 <- xyplot(sd ~ mean, pps, type = c("p", "r"))
> p2 <- xyplot(logsd ~ logmean, pps, type = c("p", "r"))
> plot(p1, split = c(1, 1, 2, 1), more = TRUE)
> plot(p2, split = c(2, 1, 2, 1))
```



Ozone Data

The R code on the website has code to construct the remainder of the profile plots in Chapter 2 of our text.