S4C03, HW2: model formulas, contrasts, `ggplot2`, and basic GLMs

Ben Bolker

September 23, 2013

Version: 2013-09-23 16:55:21

0 formulas and contrasts

You should read the section in Dobson & Barnett on contrasts first/in addition to this section.

0.1 contrasts for a single categorical variable

We can look at a given contrast matrix with the relevant `contr.*` function, e.g. for a treatment contrast with 4 levels:

```
(c1 <- contr.treatment(4))

##   2 3 4
## 1 0 0 0
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1
```

Note that this returns the contrast matrix without the intercept.

Test whether these contrasts are orthogonal by multiplying the matrix by its transpose and checking that it is diagonal ($\boldsymbol{C}^T\boldsymbol{C}$)

```
t(c1) %*% c1

##   2 3 4
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1

## or: crossprod(c1)
```

If we want to include the intercept we can `cbind` a row of ones (R automatically replicates the 1 into an appropriate vector):

```
c2 <- cbind(1,c1)
```

`crossprod(c2)` shows that the intercept term is *not* independent of the treatment contrasts.

We can most easily interpret the contrasts by inverting $\boldsymbol{C}$ (with the intercept term included): if $\boldsymbol{\mu}$ is the vector of means in a one-way layout and $\boldsymbol{\beta}$ is the vector of parameters, then the contrast matrix is defined as $\boldsymbol{\mu} = \boldsymbol{C}\boldsymbol{\beta}$ but we can more easily interpret $\boldsymbol{\beta} = \boldsymbol{C}^{-1}\boldsymbol{\mu}$.

For example (`solve(C)` is R-ish for matrix inversion, $\boldsymbol{C}^{-1}$):

```
solve(c2)

##    1 2 3 4
##    1 0 0 0
## 2 -1 1 0 0
## 3 -1 0 1 0
## 4 -1 0 0 1
```

shows that the first parameter represents the mean of the reference category ($\beta_1 = \mu_1$); the second parameter β_2 (represented by the second row of $\mathbf{C}^{-1}$) represents $\mu_2 - \mu_1$; $\beta_3 = \mu_3 - \mu_1$; and so forth.

Take another example, forward differences (coded by `contr.sdif` in the `MASS` package).

By itself, the contrast matrix is a bit hard to interpret:

```
library(MASS)
(c2 <- cbind(1,contr.sdif(4)))
```

```
##        2-1  3-2   4-3
## 1 1 -0.75 -0.5 -0.25
## 2 1  0.25 -0.5 -0.25
## 3 1  0.25  0.5 -0.25
## 4 1  0.25  0.5  0.75
```

(try `fractions(c2)` for a prettier version).

Inverting the matrix:

```
solve(c2)

##          1     2     3    4
##       0.25  0.25  0.25 0.25
## 2-1 -1.00  1.00  0.00 0.00
## 3-2  0.00 -1.00  1.00 0.00
## 4-3  0.00  0.00 -1.00 1.00
```

shows that β_1 represents the mean of all groups, while β_j for $j > 1$ represents $\mu_{j+1} - \mu_j$. Are the contrasts independent?

You can also go in the other direction, deriving the contrast matrix by inverting the $\boldsymbol{C}^{-1}$ matrix that describes the comparisons you want to make. I was recently working on a problem with factor levels "0", "C", "S", and "CS". I wanted to use 0 as the baseline, making the intercept equal to the mean of group 0; compare 0 with the average of the other three treatments; compare C with S; and compare the average of C and S with CS. I set up this $\mathbf{C}^{-1}$:

```
(c3 <- matrix(c(1,0,0,0,
               -1,1/3,1/3,1/3,
               0,-1,1,0,
               0,-1/2,-1/2,1),
             nrow=4,byrow=TRUE))

##      [,1]    [,2]    [,3]   [,4]
## [1,]    1  0.0000  0.0000 0.0000
## [2,]   -1  0.3333  0.3333 0.3333
## [3,]    0 -1.0000  1.0000 0.0000
## [4,]    0 -0.5000 -0.5000 1.0000
```

(It is usually clearer to specify matrices row-by-row, but then you have to tell R you've done it) Inverting gives the appropriate contrast matrix:

```
fractions(solve(c3))

##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    1    1 -1/2 -1/3
## [3,]    1    1  1/2 -1/3
## [4,]    1    1    0  2/3
```

To use these contrasts we would set the contrasts of the relevant factor, not including the intercept column:

```
contrasts(f) <- solve(c3)[,-1]
```

Exercise 1 demonstrate that for Helmert contrasts (`?contr.helmert`), the intercept and all of the contrasts are mutually independent. (You can just show an example, but you can also think about how you prove it in general.)

Exercise 2 Set up contrasts for the case where we have 5 levels (`C`, `N1`, `N2`, `P1`, `P2`) and we want to compare

- `C` to the average of everything else:
- `P1` to `P2`;
- `N1` to `N2`;
- the average of the `P` levels to the average of the `N` levels

Set up the contrast matrix. Are the contrasts (not including the intercept) independent? Are they independent of the intercept?

0.2 parameterizing interactions

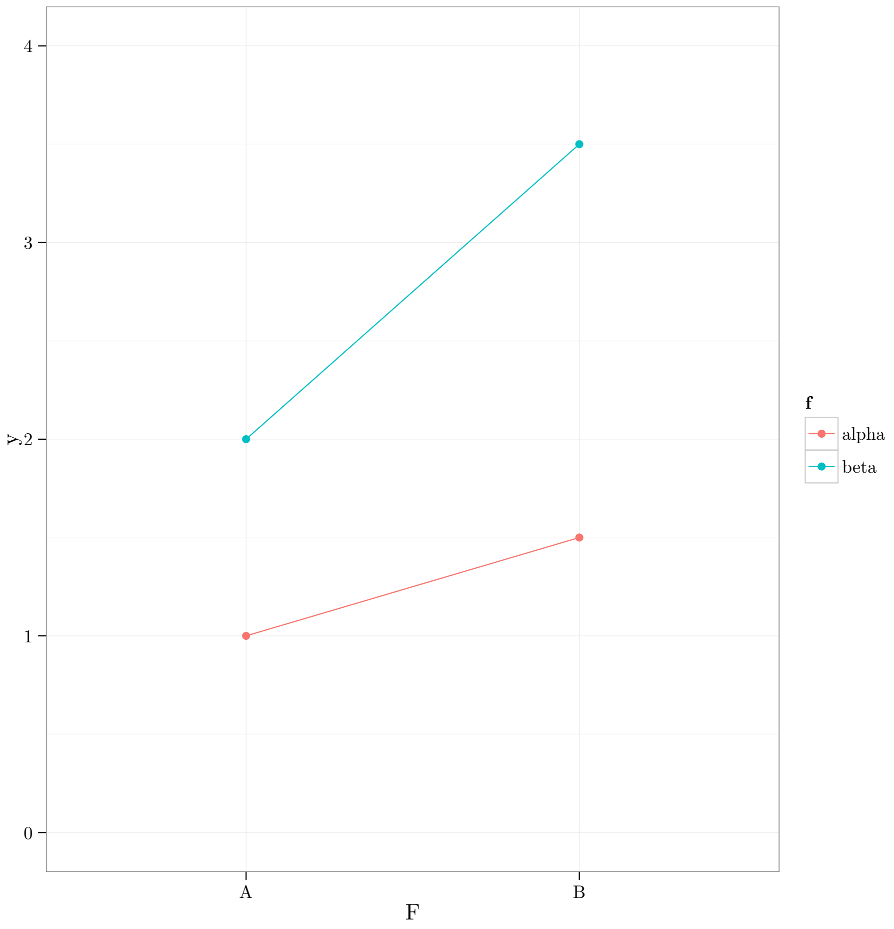
Use `expand.grid` (which creates a data frame with all possible combinations of the specified factors) to make up a data set with two treatments and exactly one observation per treatment:

```
(d <- expand.grid(F=c("A","B"),f=c("alpha","beta")))

##   F     f
## 1 A alpha
## 2 B alpha
## 3 A  beta
## 4 B  beta
```

Specify values of each treatment combination:

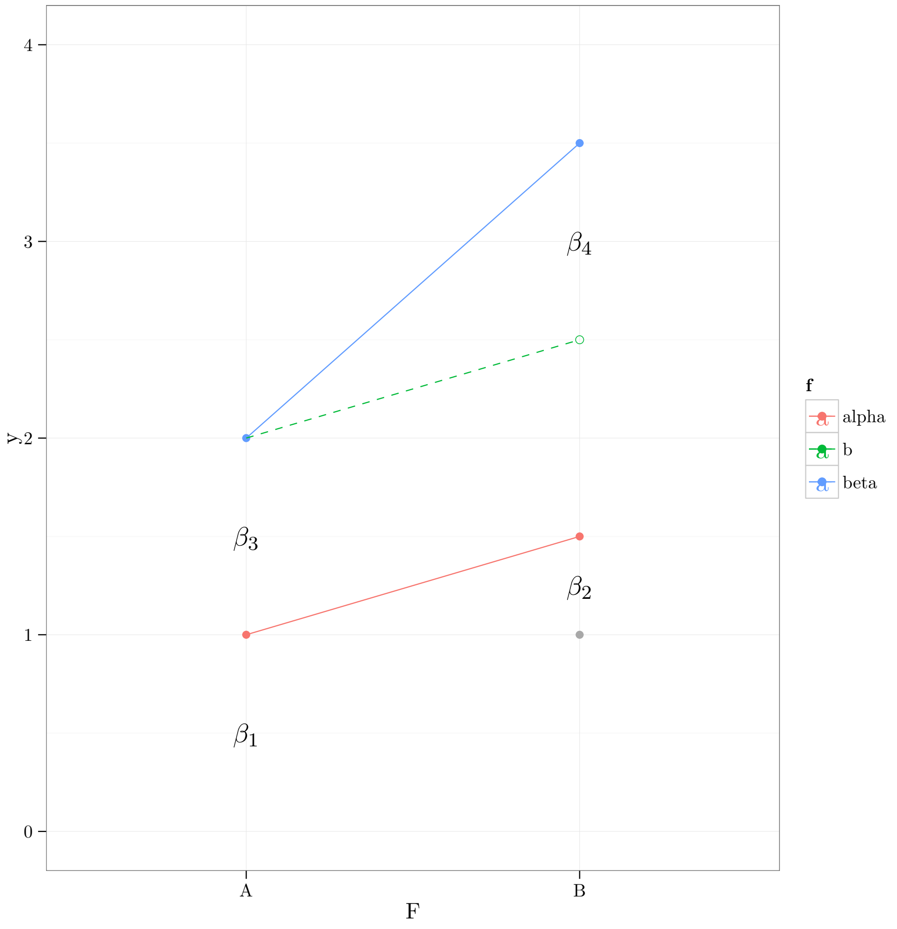
```
d$y <- c(1,1.5,2,3.5)
```

```
library(ggplot2)
theme_set(theme_bw())
(g1 <- ggplot(d,aes(x=F,y=y,colour=f))+geom_point()+
      geom_line(aes(x=as.numeric(F)))+ylim(0,4))
```



(We have to convert `F` back to numeric in the `geom_line` specification in — `ggplot` refuses to draw lines between factor levels.)

Add a hypothetical line for the *additive* effect of moving from `F=A` to `F=B` (i.e., if the two factor effects were independent of each other), and labels corresponding to the parameters in the default treatment contrasts parameterization (gory details suppressed: see the Rnw file).



Fit the full model:

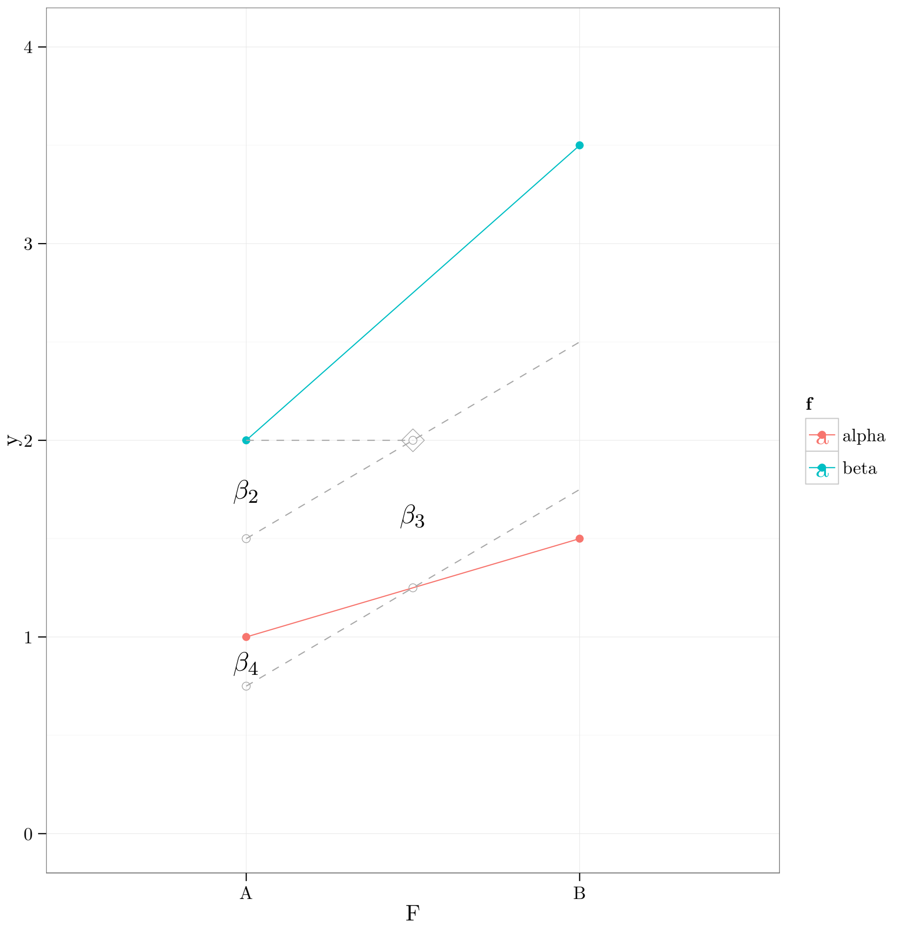
```
lm(y~F*f,data=d)

##
## Call:
## lm(formula = y ~ F * f, data = d)
##
## Coefficients:
## (Intercept)           FB        fbeta     FB:fbeta
##         1.0          0.5          1.0          1.0
```

Convince yourself that the coefficients listed match up with the labels in the picture.

We can change the default contrasts for (unordered) factors to sum contrasts as follows (the second element in the vector, `"contr.poly"`, tells R what to do with ordered factors [which we haven't dealt with yet]):

```
options(contrasts=c("contr.sum","contr.poly"))
lm(y~F*f,data=d)

##
## Call:
## lm(formula = y ~ F * f, data = d)
##
## Coefficients:
## (Intercept)           F1            f1         F1:f1
##        2.00        -0.50         -0.75          0.25
```

What does this mean? Let's look at another picture:



- β_1 (not labelled) gives the value of the overall mean (diamond) (if the data set were unbalanced it would give the average of the *group means*, not of the raw data).
- β_2 gives the difference between the overall mean and the mean of group `A` in factor `F` (we might like to have a parameter for the difference between the overall mean and the mean of group `B`, too, but this would overparameterize the model)
- β_3 gives the difference between the overall mean and the mean of group `alpha` in factor `f` (red line)
- β_4 gives the difference between the value of group (`A`, `alpha`) and that predicted from $\beta_1 + \beta_2 + \beta_3$. In this case the interaction is positive,

because the slope of the red line is less than the mean slope (dotted gray lines) and so using the mean effect of `F` (β_2) overpredicts the decrease in the mean.

If we `relevel` the data set then parameters β_2 and β_3 change sign, but the intercept and the interaction term stay the same.

```
d2 <- transform(d,
                F=relevel(F,"B"),
                f=relevel(f,"beta"))
lm(y~F*f,data=d2)

##
## Call:
## lm(formula = y ~ F * f, data = d2)
##
## Coefficients:
## (Intercept)           F1           f1        F1:f1
##        2.00         0.50         0.75         0.25
```

convince yourself that you know what's going on here.

Now reset the contrasts to their original values, to avoid confusion:

```
options(contrasts=c("contr.treatment","contr.poly"))
```

Exercise 3 Try setting the contrasts for the problem above to `contr.SAS` (which uses treatment contrasts where the *last* level of each factor is treated as the baseline). Explain where the coefficients come from. Suppose we fitted a model where the parameters were (4,2,-2,-1). Calculate (and explain) the expected value for the treatment combination $\{F = A, f = a\}$.

1 ggplot2

The ggplot2 web site `http://had.co.nz/ggplot2` is a useful reference.

1.1 Bangladesh contraception data

Load the `ggplot2` and `mlmRev` packages (you need the latter to get the `Contraception` data set we'll be working with:

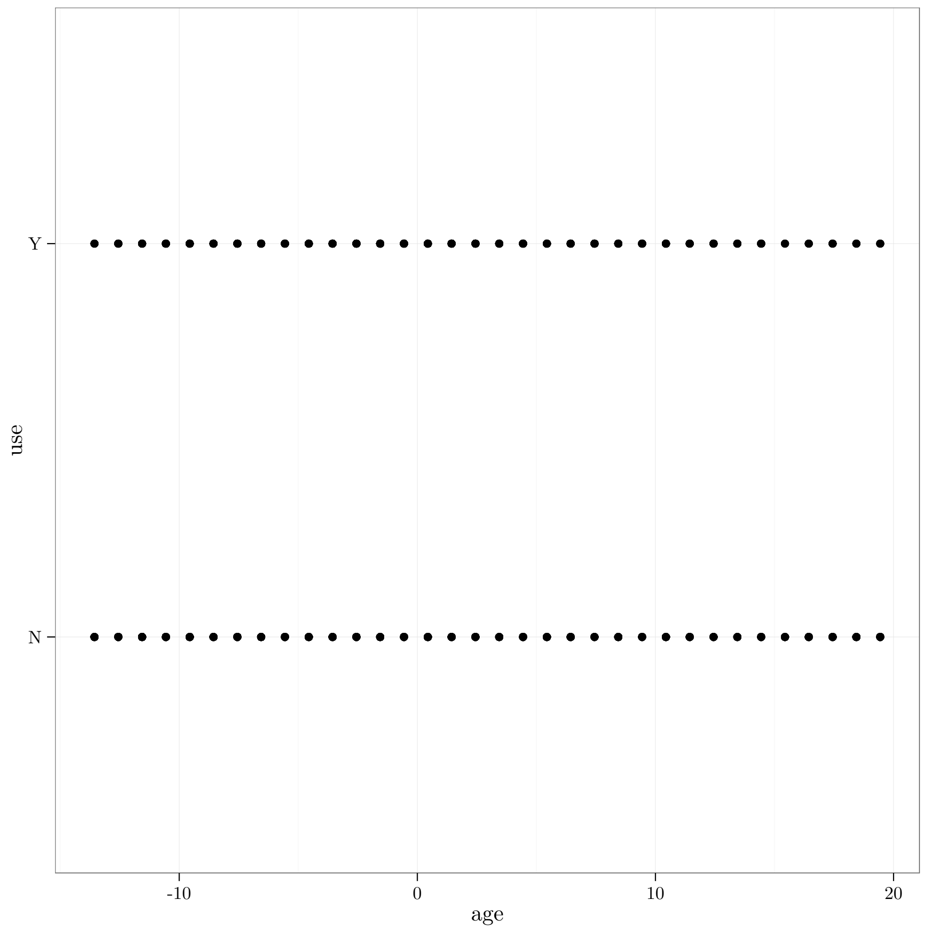
```r
library(mlmRev)
library(ggplot2)
library(grid) ## for unit()
```

Start by setting the "theme" to black & white theme (white background, gray grid lines rather than gray background, white grid lines): squash panels together when drawing multiple facets

```r
theme_set(theme_bw())
theme_update(panel.margin=unit(0,"lines"))
```

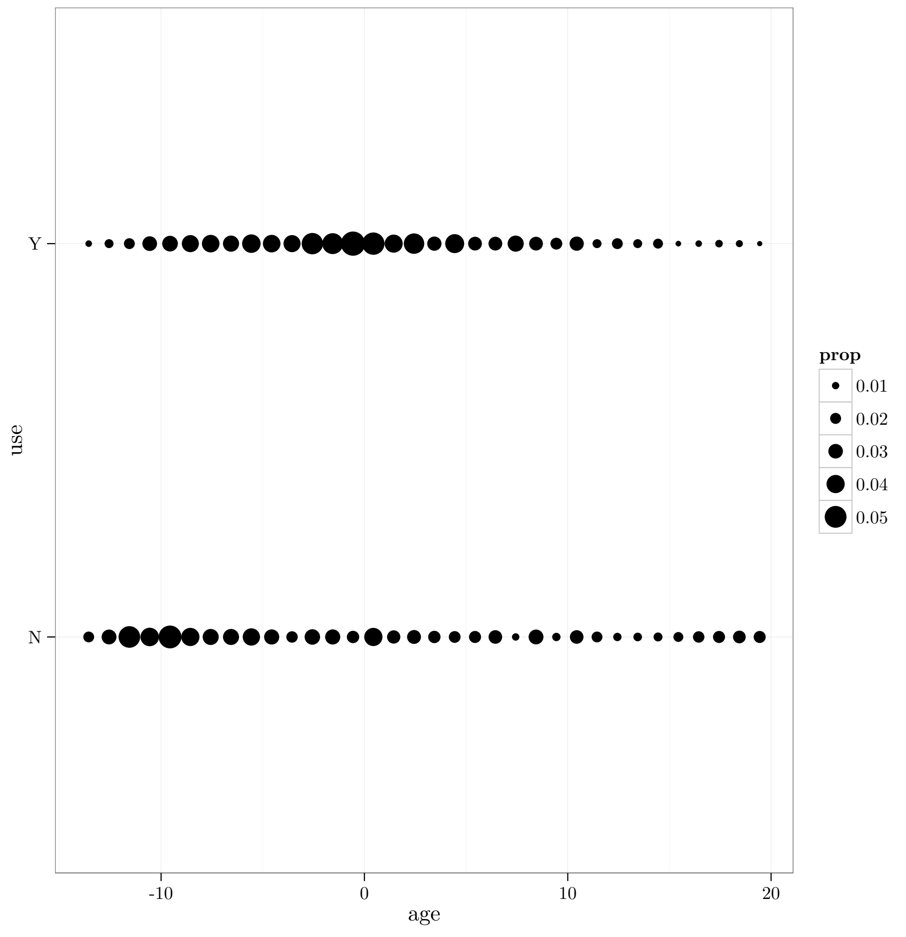
This is the basic plot, but it turns out to be useless because there are points at 0 and 1 for *every* (discrete) age in the data set:

```r
ggplot(Contraception,aes(x=age,y=use))+geom_point()
```



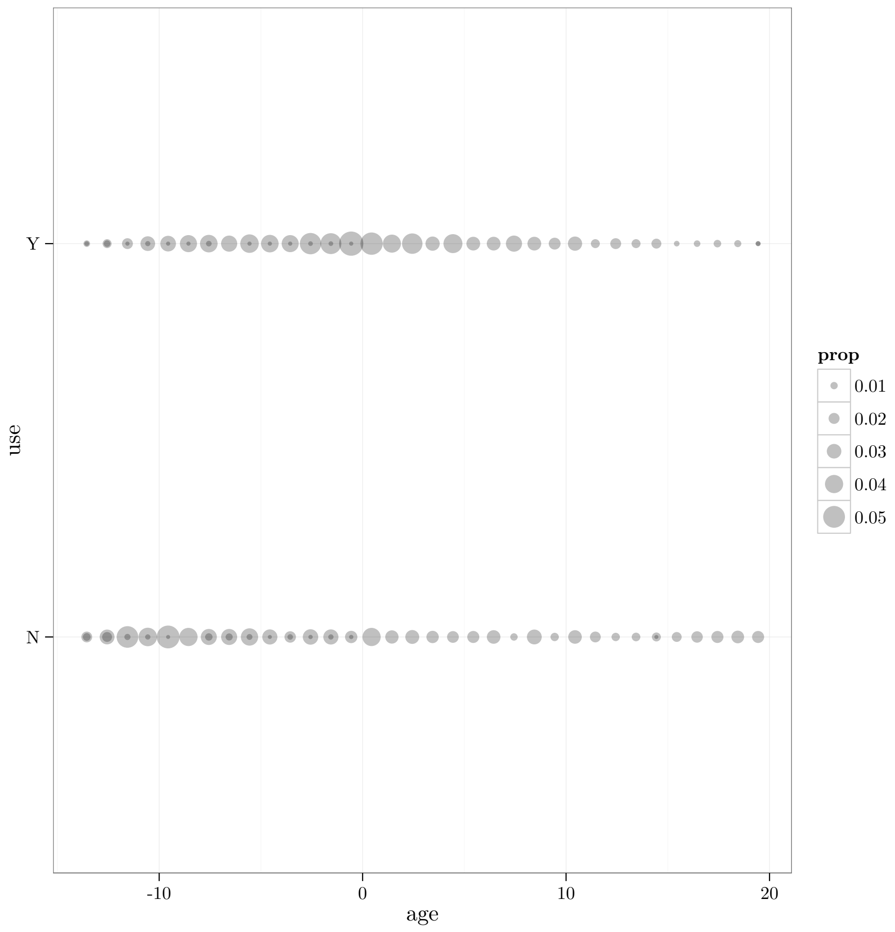
Use `stat_sum`, instead of `geom_point`, to see how many overlapping points there are (`stat_sum` automatically makes a call to `geom_point` after summarizing):

```
ggplot(Contraception,aes(x=age,y=use))+stat_sum()
```



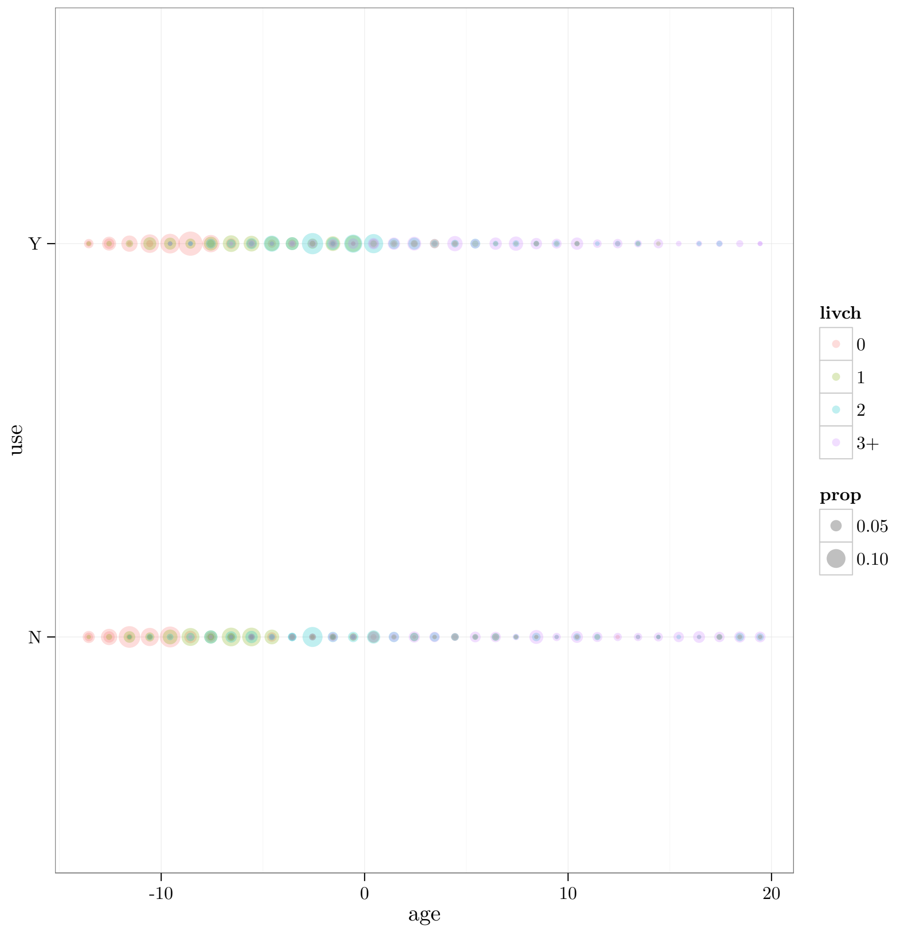
See better by making the points semi-transparent. (Graphical commands *outside* of an `aes()` statement are taken as concrete values, rather than mappings.)

```
ggplot(Contraception,aes(x=age,y=use))+stat_sum(alpha=0.25)
```



Also group by number of live children (use colour): save the result as `g1` (you need to surround the command with parentheses, or ask R to print the variable after you define it):

```
g1 <- ggplot(Contraception,aes(x=age,y=use,colour=livch))+
  stat_sum(alpha=0.25)
g1
```



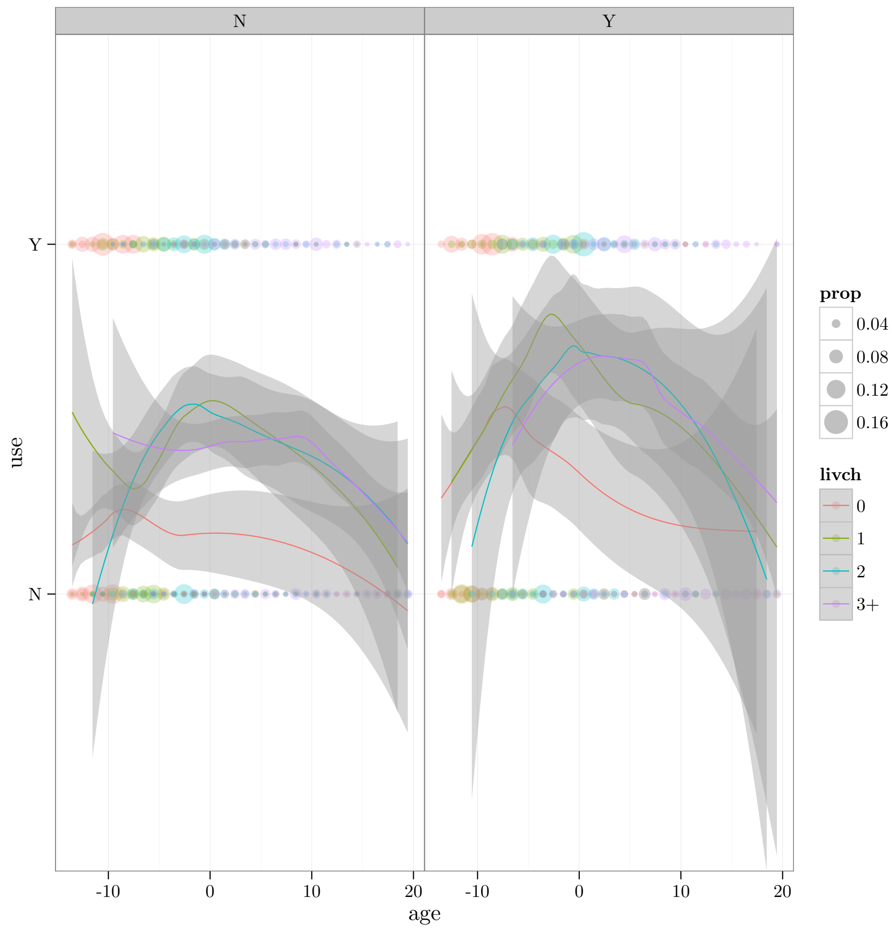
We immediately get some information about the age-dependence of the number of live children, but that's not what we're after here.

Add a smooth line (need to explicitly convert factor to numeric)

```
g2 <- g1 +  geom_smooth(aes(y=as.numeric(use)))
```

Split by urban/rural

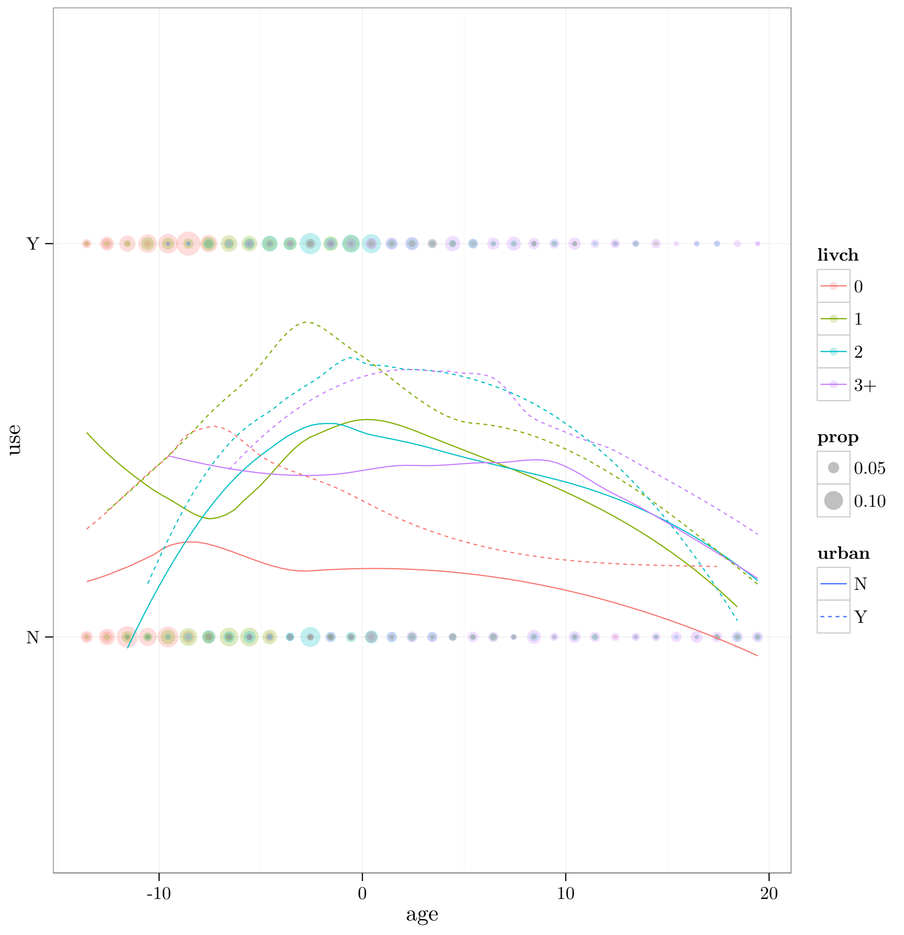
```
g2+facet_grid(.~urban)
```



(you can ignore messages about `geom smooth: method="auto" and size of largest group is` or use `method="loess"` inside the `geom_smooth()` call to suppress them . . .)

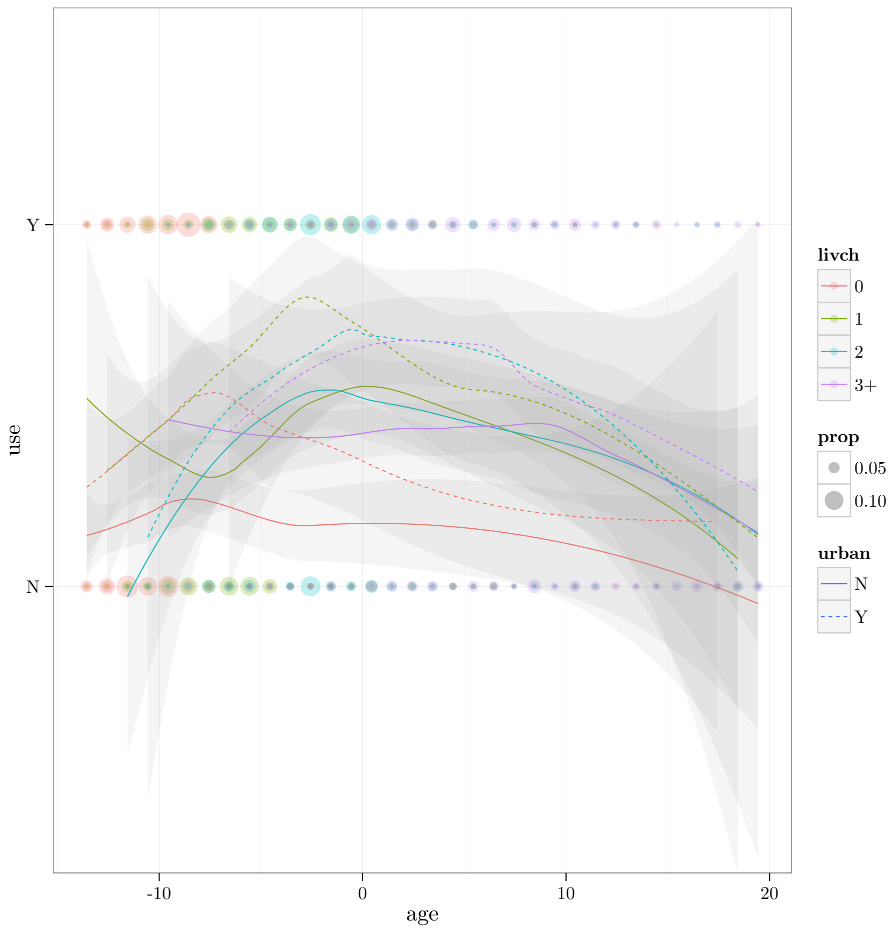
Instead of faceting, draw different line types for urban/rural; use `se=FALSE` to suppress confidence intervals.

```
g1 + geom_smooth(aes(y=as.numeric(use),
                               lty=urban),se=FALSE)
```



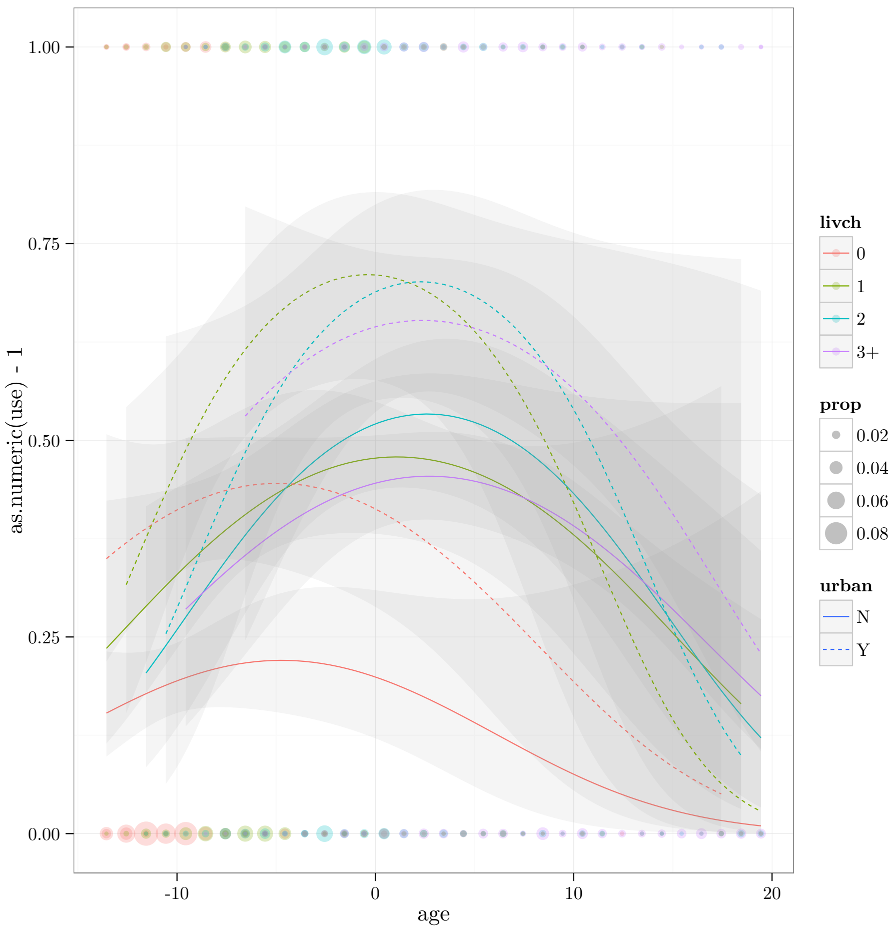
Put in confidence intervals, but draw them much lighter by adjusting `alpha`:

```
g1 + geom_smooth(aes(y=as.numeric(use),
                     lty=urban),alpha=0.1)
```



If I want to fit GLMs I have to use `as.numeric(use)-1` as a response variable (in order to make the response (0,1):

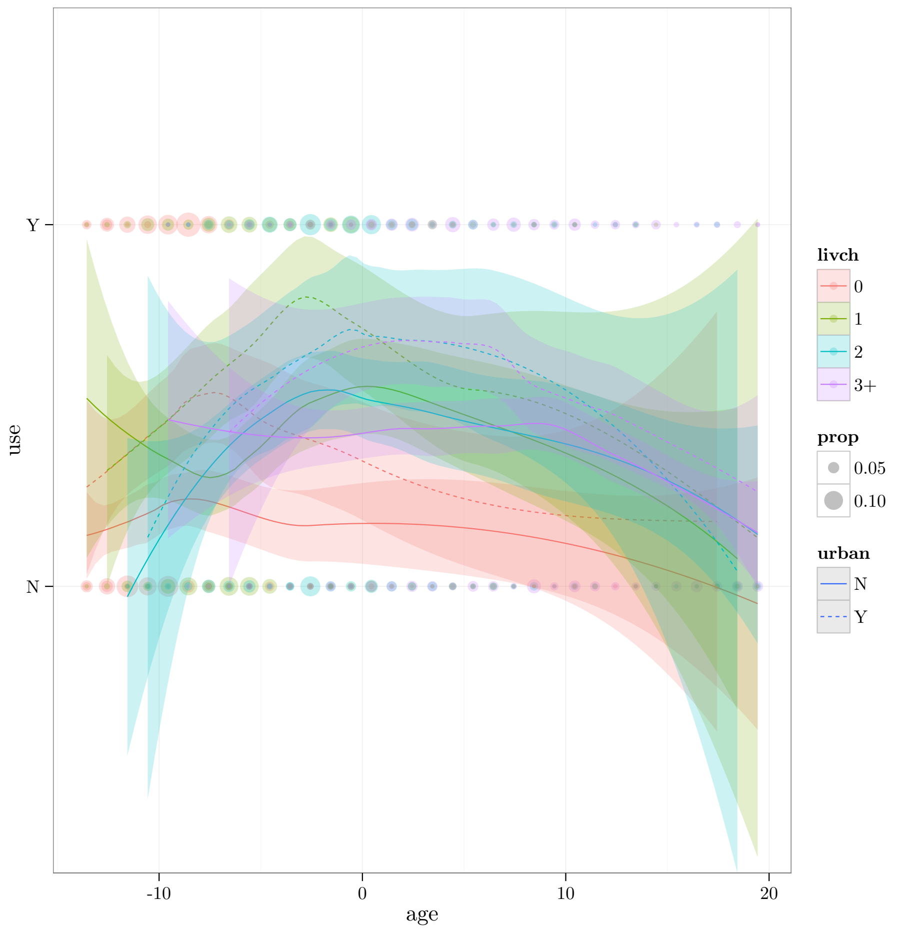
```
g3 <- ggplot(Contraception,aes(x=age,
                               y=as.numeric(use)-1,
                               colour=livch))+
  stat_sum(alpha=0.25)
g3 + geom_smooth(aes(lty=urban),method="glm",
                    family="binomial",
                    formula=y~poly(x,2),alpha=0.1)
```



Formulae that I specify in `geom_smooth` use `y` and `x` no matter what my original variables were called . . .

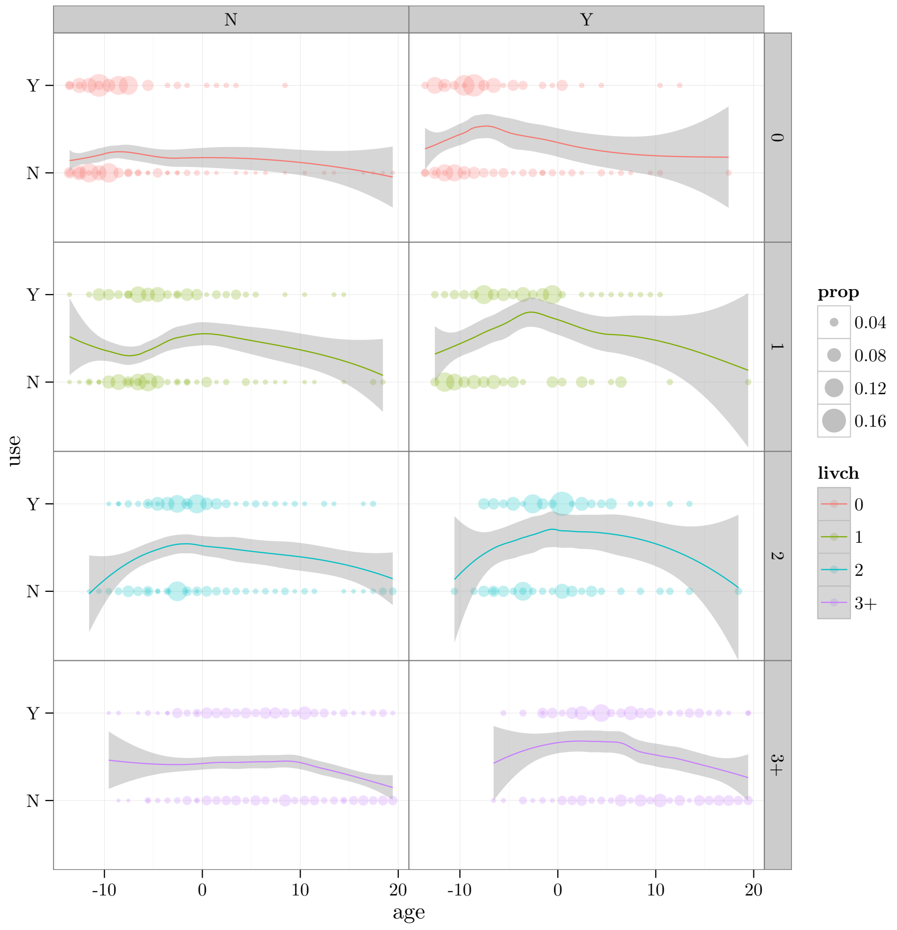
Add colour to confidence intervals:

```
g1 + geom_smooth(aes(y=as.numeric(use),
                     fill=livch,
                     lty=urban),
                 ## method="lm",
                 alpha=0.2)
```



Draw a two-way grid of live children × urban/rural:

```
g1 + facet_grid(livch~urban)+geom_smooth(aes(y=as.numeric(use)))+
  theme(panel.margin=unit(0,"lines"))
```



Exercise 4 experiment with the data, but explore the determinants of number of live children. You can either ignore the `use` (contraceptive use) variable entirely, or use it as a predictor variable (don't forget that if you put `livch` on the y axis you will need to use `as.numeric(livch)` to get numeric summaries such as smooth lines. (If you want to put `use` on the x axis and you need `stat_sum` to work properly, you need to specify `group=1` in the mapping . . .) Create three interesting plots. Explain what they show.

Alternate assignment: use `ggplot2` to create three interesting plots using a data set of interest to you.

1.2 Carbohydrate data

(this section has no exercises associated, but you might want to go through it anyway . . .)

```
cdat <- read.csv("Carbohydrate_diet.csv")
```

In order to plot all the graphs at once, we want to `melt` the data into a long format where each data point is represented once for each predictor variable (age, weight, protein). We need to add an ID variable so that individuals with the same `carbohydrate` value don't get combined . . .

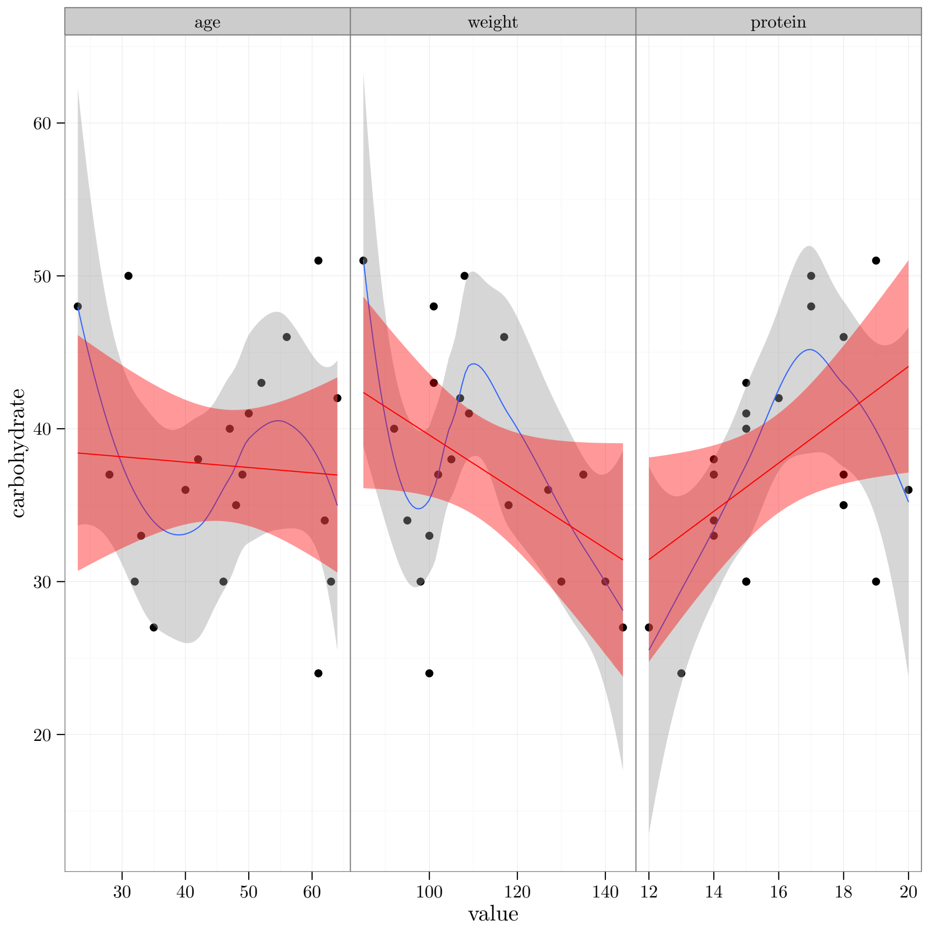
```
library(reshape2)
cdat$id <- 1:nrow(cdat)  ## add an id variable
mcdat <- melt(cdat,id.var=c("id","carbohydrate"))
head(mcdat)

##   id carbohydrate variable value
## 1  1           33      age    33
## 2  2           40      age    47
## 3  3           37      age    49
## 4  4           27      age    35
## 5  5           30      age    46
## 6  6           43      age    52
```

Plot points with a separate facet for each variable (use `scale="free"` to allow the x axis scales to vary):

```
g1 <- ggplot(mcdat,aes(x=value,y=carbohydrate))+geom_point()+
    facet_wrap(~variable,scale="free_x",nrow=1)
```

Overlay a smooth fit and a (marginal) linear model:

```
g1 + geom_smooth()+
  geom_smooth(method="lm",colour="red",fill="red")
```



2 Generalized linear models

2.1 Lizards

Install the `brglm` package (using `install.packages` or the menu) and retrieve the codelizards data set.

```
library(brglm)
data(lizards)
```

Look at `?lizards` to find out about the data set.
Exercise

- create new variables *within the data frame* (i.e. using `lizards$var <- ...` or `lizards <- transform(lizards,...)`) for the total number (*Grahami* + *Opalinus*) and the fraction of *Grahami* observed in each condition.
- create a plot with `ggplot`, putting the fraction of *Grahami* lizards on the y axis and representing all four predictor variables (light, diameter, height, time) with some combination of (1) position along the x axis; (2,3) location (rows and columns) within a facet grid; and (4) colour. Use the size of the points to represent the total number of lizards observed at each time.
- Fit a GLM including all the main effects and two-way interactions.

3 Tribolium

Get `tribolium.txt` and read it in.

```
trdat <- read.table("tribolium.txt",header=TRUE)
```

Data include adult species (all "A" in this case, so you can disregard this variable); egg species ("a", "b", "c", so species "a" represents cannibalism), day of the trial (the trial was run over three days), and number of eggs (out of 50 of each type) eaten. We are interested in whether adult *Tribolium* can recognize eggs of their own species and avoid them/preferentially predate the eggs of other species.

Exercise

- plot the data with `ggplot`, with number of eggs eaten on the y axis (we don't really need proportions here since every sample has the same total size), representing egg species by colour, and day on the x axis. You might want to use `stat_sum(aes(size=factor(..n..)))` so you can see overlapping points.
- Fit a model including egg species, day, and their interaction.
- Fit an additive model (i.e. one without the interaction).
- Refit the additive model changing the contrasts: since we really want to compare self ("a") against non-self ("b" and "c"), set up new contrasts that estimate (1) the mean of a, (2) average of b and c against a, (3) b against c (the third isn't that interesting, but we need a complete set of contrasts).