Linear models: review

I will not be covering this material in class, but I will expect you to be familiar with it.

\[ \mu = X\beta; \text{ either } Y_i \sim \text{Normal}(\mu_i, \sigma^2) \text{ or } Y = \mu + e \]

Solution: \[ \beta = (XX^T)^{-1}X^Ty. \] (Think about scalar/linear regression equivalents. \( XX^T \) is like a variance-covariance matrix (but uncentered); e.g. for the linear regression case it is

\[
\begin{pmatrix}
N & \sum x_i \\
\sum x_i & \sum x_i^2
\end{pmatrix}
\]

\( X^T y \) is the (unscaled) cross-covariance of \( y \) with the columns of \( X \).

e.g. in linear regression, the regression coefficient comes out as \( \sigma_{XY}/\sigma_X^2 \): cross-variation in \( X \) and \( Y \) scaled by variation in \( X \) (note that the units come out OK, \([Y]/[X])\).

Can get there by finding negative log-likelihood/sum of squares \((y - X\beta)^T(y - X\beta)\), taking the derivative, solving . . . or (as in the book) taking the more general solution (allowing for an exponential-family distribution and a link function, finding the score equations \( \frac{\partial}{\partial \beta} \) and solving them) and taking the special case.

This kind of computational linear algebra is a fairly well solved problem (QR decomposition) . . . works on reduced-rank problems. Interesting problem: make it work for sparse matrices/huge data sets.

Variance-covariance matrix of parameters is \( \sigma^2(XX^T)^{-1} \), estimated as \( 1/(N - p) \cdot (y - Xb)^T(y - Xb) \) (i.e., RSS/(N-p)).

Because \( X^TXb = X^Ty \), RSS can also be expressed as \( y^Ty - b^TX^Ty \).

Hypothesis testing: \( \frac{S_0 - S_1}{p-q} / \frac{S_1}{N-p} \) has distribution \( F(p-q,N-p) \).

(Assume that model 1 fits the data well.)

For a single input (as distinguished from a predictor variable), we get the same inference from a \( t \) test of the estimated parameter with \( N - p \) df.

Orthogonal contrasts. \( XX^T \) becomes block diagonal — and we can test sub-hypotheses independently. (Multiple contrasts issue: (a) post hoc, (b) multiple comparisons.)
Hat matrix: $H = X(X^T X)^{-1} X^T = Xb$ — projects $y$ onto “expected $y$”.

Exploratory graphics

- reshape2::melt is good for using ggplot2 to look at univariate (marginal) plots.
- scatterplot matrices: pairs, GGally::ggpairs etc.
- Reminder: don’t try to test model assumptions based on the marginal appearance of the data.

Univariate plots

```r
library(reshape2)
cdat$id <- 1:nrow(cdat) ## add ID variable
## melt, retaining ID and carbohydrate as 'identifier' columns
mcdat <- melt(cdat,id.var=c("id","carbohydrate"))
library(ggplot2)
## histogram
ggplot(mcdat,aes(x=value)) + geom_histogram(aes(y=..density..),binwidth=4)+ facet_wrap(~variable,scale="free_x")
```

```r
## scatterplot with smooth lines
ggplot(mcdat,aes(x=value,y=carbohydrate))+ facet_grid(~variable,scale="free")+
```

---

Univariate plots:

<table>
<thead>
<tr>
<th>age</th>
<th>weight</th>
<th>protein</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>60</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>140</td>
</tr>
</tbody>
</table>

Density:

- age
- weight
- protein
geom_point() +
geom_smooth(method="loess") +
geom_smooth(method="lm", colour="red", fill="red", alpha=0.2)

Scatterplot matrices

library(GGally)
ggpairs(cdat, lower=list(continuous="smooth"))
or try out `pairs(cdat,gap=FALSE); lattice::splom(cdat,type=c("p","smooth"));
car::scatterplotMatrix(cdat); PerformanceAnalytics::chart.Correlation(cdat)
...

Special considerations for non-normal data

Lots of overlapping points: jittering or counting number of overlapping points.

Binary data

- rugs (`rug()`, `geom_rug()`)
- smooth lines (`geom_smooth()`)
- proportions of binned data

Use the Contraception data set from the `mlmRev` package.
Jittering: ugly but not completely useless. Smooth line is OK …

```
library(mlmRev)
ggplot(Contraception,aes(x=age,y=use))+
   geom_point(position=position_jitter(width=0.1,height=0.1))+
   geom_smooth(aes(y=as.numeric(use)))
```
Rug plot: plot “Y” values on the top, “N” values on the bottom. Not useful in this case because the ages are rounded!

```r
ggplot(Contraception, aes(x=age, y=use)) +
  geom_rug(data=subset(Contraception, use="Y"), sides="t") +
  geom_rug(data=subset(Contraception, use="N"), sides="b")
```

Summarize data, using the plyr package:

```r
library(plyr)
sdat <- ddply(Contraception,
  "age", ## split by age
  summarise, ## magic function
  n=length(use), ## compute number per bin
  prop=mean(use="Y")) ## proportion using contraception
```
Now we have to use `as.numeric(use)-1` to get a (0,1) variable that will match up with the computed proportions:

```r
ggplot(Contraception, aes(x=age, y=use)) +
  geom_smooth(aes(y=as.numeric(use)-1), method="loess") +
  geom_point(data=sdat, aes(size=n, y=prop))
```

If we had a smaller data set, or truly continuous responses, we might have to create our own classification variable by splitting the continuous variable with `cut`:

```r
Contraception <- transform(Contraception,
  agecat=cut(age, breaks=seq(-15, 20, by=2.5)))
sdat3 <- ddply(Contraception,
  "agecat", summarise,
  n=length(use),
  prop=mean(use=="Y"),
  meanage=mean(age))

ggplot(Contraception, aes(x=age, y=use)) +
  geom_smooth(aes(y=as.numeric(use)-1), method="loess") +
  geom_point(data=sdat3, aes(x=meanage, size=n, y=prop))
```
If we want to cross-classify by more variables, we have to re-do the summarization:

```r
sdat2 <- `ddply`(Contraception, c("age","urban"), summarise, ## magic function
                  n=length(use), ## compute number per bin
                  prop=mean(use=="Y") ## proportion using contraception
                 )

ggplot(Contraception, aes(x=age,y=use,colour=urban)) +
      geom_smooth(aes(y=as.numeric(use)-1),method="loess") +
      geom_point(data=sdat2, aes(size=n,y=prop))
```

Binomial data
• plot proportions  
• size of points proportional to $N$  
• possibly add binomial confidence intervals

Data from ¹:

```r
turtle_dig <- read.csv("dufemalepers.csv")
turtle_dig <- mutate(turtle_dig,  
    n=du+notdu,  
    prop_dug=du/n)
ggplot(turtle_dig, aes(density,prop_dug)) + geom_point(aes(size=n)) +  
labs(x="Density", y="Proportion dug up")
```

(\texttt{mutate} is a version of \texttt{transform} from the \texttt{plyr} package that performs the operations in order so that later operations can depend on the results of earlier ones: in this case I want to be able to use \texttt{n} when defining \texttt{prop_dug} ...)

Now add compute standard errors and add error bars:

```r
turtle_dig <- mutate(turtle_dig,  
    se=sqrt(prop_dug*(1-prop_dug)/n),  
    lwr=prop_dug-1.96*se,  
    upr=prop_dug+1.96*se)
ggplot(turtle_dig, aes(density,prop_dug)) + geom_point(aes(size=n)) +  
geom_linerange(aes(ymin=lwr, ymax=upr)) +  
labs(x="Density", y="Proportion dug up")
```

Small-count data

- `stat_sum()`

- Horizontal dotplots/beeswarm plots: `geom_dotplot(binaxis = "y", stackdir = "center")`

Some data from Adriaan De Jong:

```r
load("DeJong_alldat.RData")
g_bee <- ggplot(alldat, aes(x=treatment,y=number/area, fill=treatment,color=treatment)) +
  geom_dotplot(stackdir="center", binaxis="y", alpha=0.8) +
  facet_wrap(~sp)
suppressMessages(print(g_bee)) ## complains about default bin width
```
Beanplots [1] are another alternative for this sort of data, although they aren’t (yet) coded for use with ggplot2 …

References
