Logistic and binomial regression

Ben Bolker

October 1, 2018

modeling

data analysis road map

---

1. figure out the (subject-area) question
2. design experiment/data collection (power analysis; simulation)

---

3. *collect data*

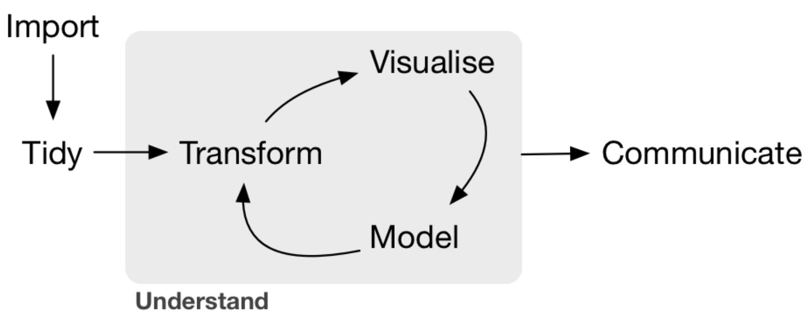
---

4. understand the data
5. specify the model; *write it down!*

---

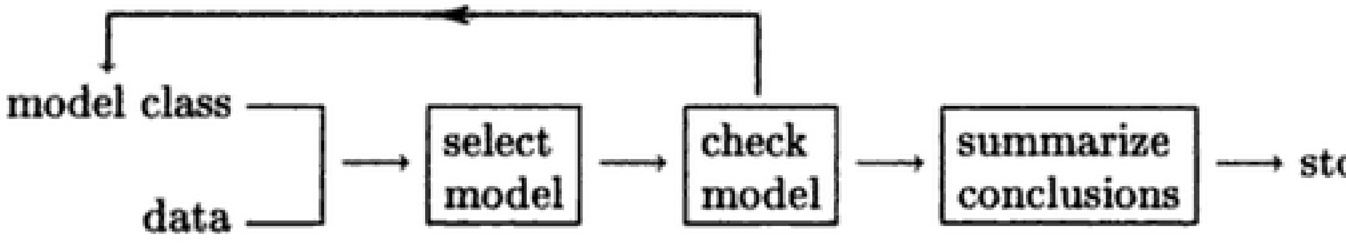
6. inspect data (Q/A) (return to 5?)
7. fit model
8. graphical diagnostics (return to 5?)
9. interpret parameters; inference; plot results

Alternatively



https://jules32.github.io/2016-07-12-Oxford/dplyr_tidyr/

also:



(McCullagh and Nelder, 1989)

These are great but *doesn't address the snooping problem*

basics

In principle can use any smooth function from $(0,1) \to \mathbb{R}$ as the link function

- *logistic* regression: binary data with a logit link (inverse-link=logistic)
- *binomial* (or *aggregated binomial* regression: binomial data (maybe logit link, maybe other)
- *probit regression*: probit link

Binary data and aggregated ($N > 1$ data) are handled slightly differently.

```
## Error in ggplot(linkdata, aes(x, prob, colour = invlink)):
could not find function "ggplot"
```

```
library(ggplot2)
theme_set(theme_bw())
library(grid)
zmargin <- theme_update(panel.spacing=unit(0,"lines"))
library(dotwhisker)
library(descr) ## for R^2 measures
library(aods3)  ## for overdispersion
```

Contraception data example

```
library(MEMSS)
data("Contraception",package="mlmRev")
head(Contraception)

##   woman district use livch      age urban
## 1     1        1   N    3+  18.4400     Y
## 2     2        1   N     0  -5.5599     Y
## 3     3        1   N     2   1.4400     Y
## 4     4        1   N    3+   8.4400     Y
## 5     5        1   N     0 -13.5590     Y
## 6     6        1   N     0 -11.5600     Y
```

```
cc <- transform(Contraception,
                use_n=as.numeric(use)-1,
                age_cat=cut(age,breaks=age_breaks))

## Error in cut.default(age, breaks = age_breaks):  object
'age_breaks' not found
```

```
gg0 <- ggplot(cc,aes(age,use_n,colour=urban))+
    stat_sum(alpha=0.5)+facet_wrap(~livch,labeller=label_both)
gg0+geom_smooth()
gg0+geom_smooth(method="gam",
                formula=y~s(x,k=20),
                method.args=list(family=binomial))
```

Alternative smoothing: binning

```
age_breaks <- seq(-15,20,by=5)
age_mids <- (age_breaks[-1]+age_breaks[-length(age_breaks)])/2
sumfun <- function(use) {
    prop <- mean(use)
    n <- length(use)
    se <- sqrt(prop*(1-prop)/n)  ## approx binomial CIs
    c(prop=prop,n=n,se=se)
}
cc_agg <- aggregate(use_n~age_cat+urban+livch,
          data=cc,
          FUN=sumfun)
## ugh!
cc_agg[,c("prop","n","se")] <- cc_agg$use_n
cc_agg$age_mid <- age_mids[as.numeric(cc_agg$age_cat)]
gg0+geom_pointrange(data=cc_agg,
                    aes(x=age_mid,
                        y=prop,
                        ymin=prop-2*se,
                        ymax=prop+2*se,
                        size=n),
                    alpha=0.5)+
    scale_size(range=c(1,3))
```

Fit the model: quadratic with all interactions

Computing model predictions

Set up an example to use:

```
lizards <- read.csv("../data/lizards.csv")
lizards <- transform(lizards,
                     time=factor(time,levels=c("early","midday","late")))
g1 <- glm(gfrac~height+diameter+light+time,
          lizards,family=binomial,weight=N)
```

Predictions Predictions are fairly easy: set up the new model matrix and multiply by coefficients, then compute the inverse link. This is what `predict` does (use `type="response"` to get the back-transformed predictions).

```
newdata <- with(lizards,
                expand.grid(height=levels(height),
                            diameter=levels(diameter),
                            light=levels(light),
                            time=levels(time)))
## [-2] deletes the response variable; you could also use
## formula(delete.response(terms(g1)))
newX <- model.matrix(formula(g1)[-2],newdata)
pred0 <- newX %*% coef(g1) ## log-odds
pred <-  plogis(pred0)     ## probability
head(c(pred))

## [1] 0.7497750 0.9026836 0.5829216 0.8122611 0.8748663 0.9558361

## or:
head(predict(g1,newdata,type="response"))

##         1         2         3         4         5         6
## 0.7497750 0.9026836 0.5829216 0.8122611 0.8748663 0.9558361
```

If you use `predict`, keep in mind that `predict` produces predictions *on the scale of the linear predictor* (`type="link"`) by default rather than on the scale of the original data (`type="response"`).

Confidence intervals Confidence intervals: get new model matrix and compute XVX^T to get variances on the link-function scale. Then compute Normal CIs on the link scale, *then* back-transform. Or use `se=TRUE` in `predict`.

```
pvar <- newX %*% vcov(g1) %*% t(newX)
pse <- sqrt(diag(pvar))
```

Or equivalently for any model type where `predict` has an `se.fit` argument:

```
pse <- predict(g1,newdata=newdata,se.fit=TRUE)$se.fit
lwr0 <- pred0-1.96*pse  ## or qnorm(0.025)
upr0 <- pred0+1.96*pse  ## or qnorm(0.975)
lwr <- plogis(lwr0)
upr <- plogis(upr0)
```

Put the predictions and confidence intervals back into a data frame with the predictor variables:

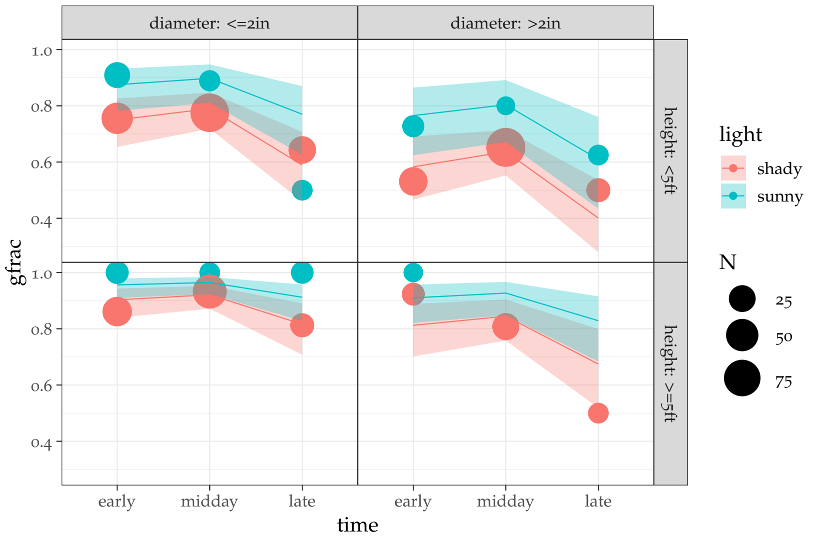
```
predFrame <- data.frame(newdata,gfrac=pred,lwr,upr)
```

Note:

- back-transforming the standard errors via a logistic usually doesn't make sense: if you want to back-transform them (approximately), you have to multiply them by $(d\mu/d\eta)$, i.e. use `dlogis`.
- if you use `response=TRUE` and `se.fit=TRUE`, R computes the standard errors, scales them as above, and uses them to compute (approximate) *symmetric* confidence intervals. Unless your sample is very large and/or your predicted probabilities are near 0.5 (so the CIs don't get near 0 or 1), it's probably best to use the approach above

Getting CIs into `ggplot` Compute a new data frame, then use `geom_ribbon` (need to set `alpha` by hand, and use `colour=NA` to suppress lines at the edges of the ribbon), unlike when using `geom_smooth`).

```
gplot0 <- ggplot(lizards,aes(time,gfrac,colour=light))+
    facet_grid(height~diameter,labeller=label_both)+geom_point(aes(size=N))+
    scale_size_continuous(range=c(3,9))
gplot0 + geom_line(data=predFrame,aes(group=light))+
    geom_ribbon(data=predFrame,
                aes(ymin=lwr,ymax=upr,group=light,fill=light),
                colour=NA,
                alpha=0.3)
```

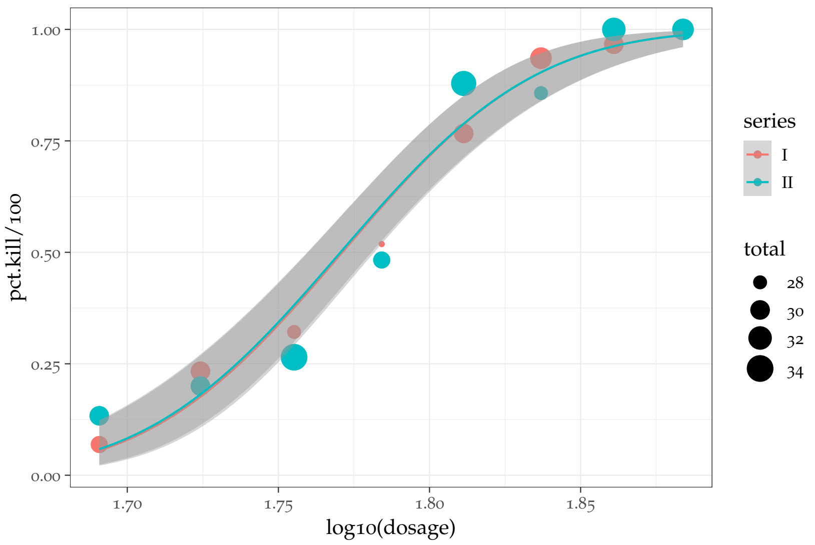


CIs on nonlinear functions of parameters Tricky. An example presented in [1], originally from [2]:

[1] Dobson, A. J. and A. Barnett (2008, May). *An Introduction to Generalized Linear Models, Third Edition* (3 ed.). Chapman and Hall/CRC

[2] Bliss, C. I. (1935). The calculation of the dosage-mortality curve. *Annals of Applied Biology* 22(1), 134âĂŞ167

```
beetle <- read.csv("../data/beetle2.csv",comment="#")
## adjust percentages so number killed=integer
beetle <- transform(beetle,
    pct.kill=100/total*round(pct.kill*total/100))
(pplot <- ggplot(beetle,aes(log10(dosage),pct.kill/100,colour=series))+
    geom_point(aes(size=total))+
    geom_smooth(method=glm,aes(weight=total),
                method.args=list(family=binomial(link="probit"))))
```



It looks like we can ignore the difference between the series . . .

```r
g2 <- glm(pct.kill/100~log10(dosage),
          data=beetle,
          family=binomial(link="probit"),
          weight=total)
```

Suppose we're interested in the LD50, i.e. the dose required to kill 50% of the population. Since

$$\begin{aligned} p = 0.5 &= \text{logistic}(\beta_0 + \beta_1 x_{0.5}) \\ 0 &= \beta_0 + \beta_1 x_{0.5} \\ x_{0.5} &= -\frac{\beta_0}{\beta_1} \end{aligned}$$

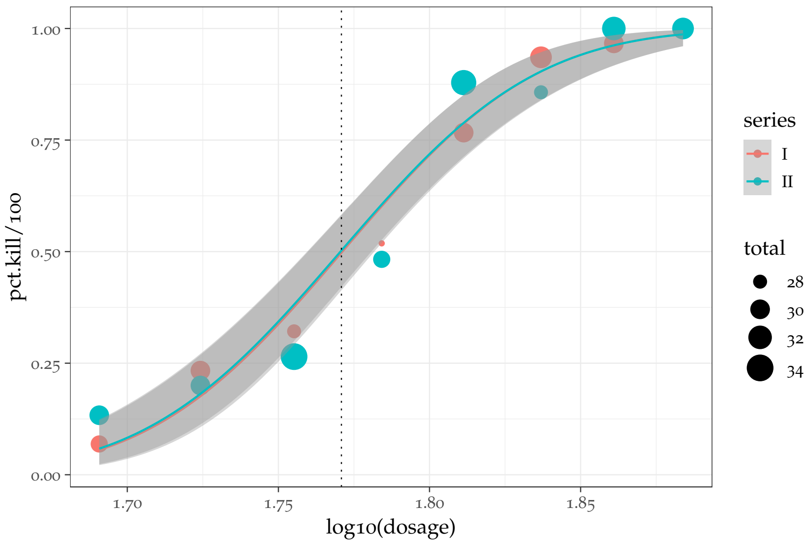
(the units work out correctly too)

So the estimate is

```r
cc <- coef(g2)
ld50 <- -cc[1]/cc[2]
```

Double-check graphically:

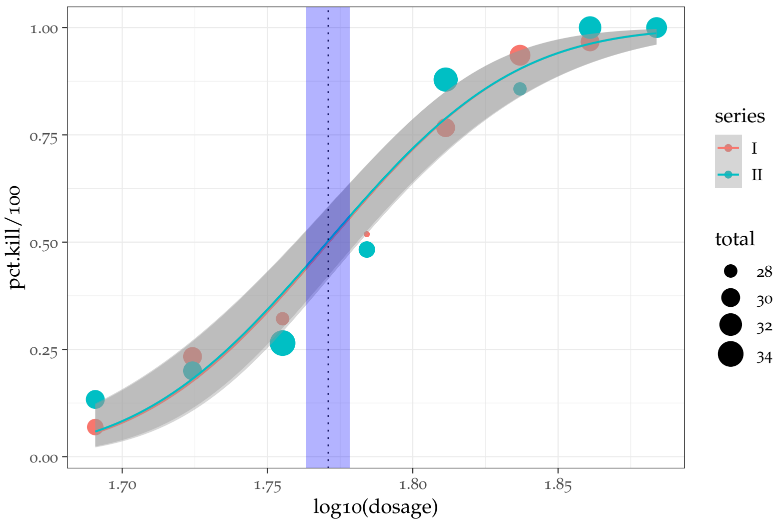
```r
pplot+geom_vline(xintercept=ld50,linetype=3)
```



But what about the confidence intervals?

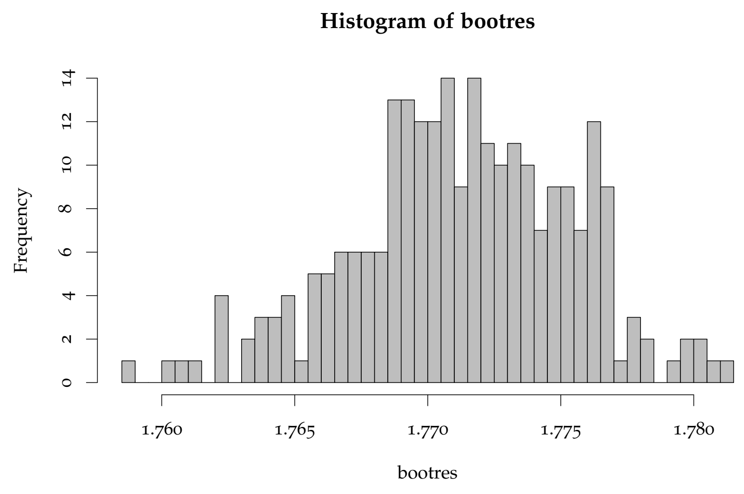
- **delta method**: If we want to compute the variance of $f(x,y,z)$ and $g = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$ then the variance is $g V g^T$ (which reduces to $\text{CV}^2(f(x,y)) = \text{CV}^2(x) + \text{CV}^2(y)$ for the case of independent values when $f(x,y) = x/y$ or xy):

```
grad <- c(-1/cc[2],cc[1]/cc[2]^2)
ld50_var <- t(grad) %*% vcov(g2) %*% grad
ld50_se <- c(sqrt(ld50_var)) ## c() converts from matrix to vector (= scalar)
deltaCI <- ld50+c(-1,1)*1.96*ld50_se
pplot+geom_vline(xintercept=ld50,linetype=3)+
    annotate("rect",
           xmin=deltaCI[1],
           xmax=deltaCI[2],
           ymin=-Inf,
           ymax=Inf,alpha=0.3,fill="blue",
             colour=NA)
```



- bootstrapping

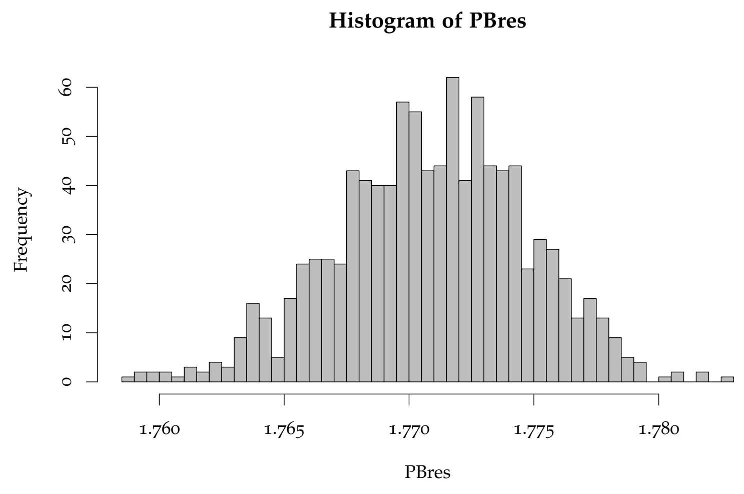
```
bootres <- numeric(250)
for (i in 1:250) {
    bootdat <- beetle[sample(nrow(beetle),replace=TRUE),]
    bootmodel <- update(g2,data=bootdat)
    bootcc <- coef(bootmodel)
    bootres[i] <- -bootcc[1]/bootcc[2]
}
hist(bootres,col="gray",breaks=50)
```



```
bootCI <- quantile(bootres,c(0.025,0.975))
```

- pseudo-Bayes: MVN sample from parameters

```
library(MASS)
PBsamp <- mvrnorm(1000,mu=coef(g2),Sigma=vcov(g2))
PBres <- -PBsamp[,1]/PBsamp[,2]
hist(PBres,col="gray",breaks=50)
```



```
PBCI <- quantile(PBres,c(0.025,0.975))
```

In this case the results are all extremely similar:

```r
rbind(deltaCI,bootCI,PBCI)

##              2.5%    97.5%
## deltaCI 1.763397 1.778304
## bootCI  1.762387 1.779041
## PBCI    1.763289 1.777993
```

Logistic example

```r
## data from http://data.princeton.edu/wws509/datasets/cuse.dat
if (FALSE) {
    try(download.file("http://data.princeton.edu/wws509/datasets/cuse.dat",
                      dest="../data/cuse.dat"),
        silent=TRUE)
}
cuse <- read.table("../data/cuse.dat",header=TRUE)
```

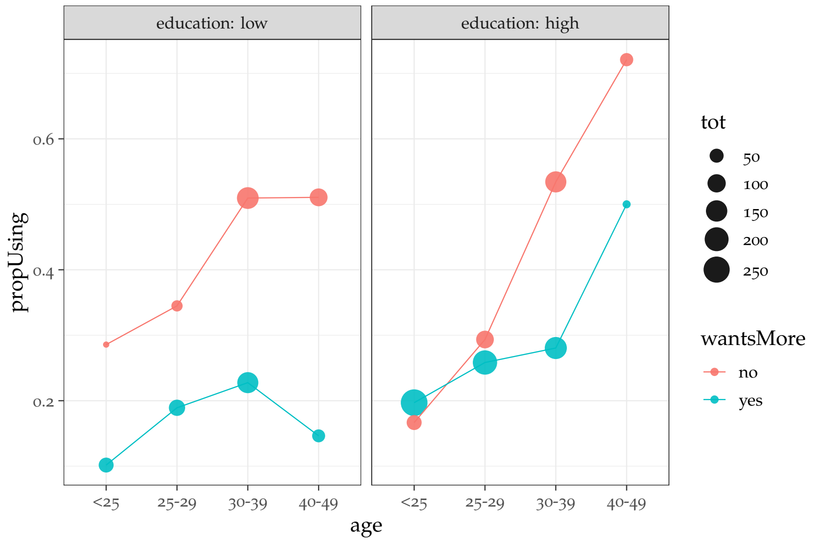
Add convenience variables (proportion and total in each group): change the `education` factor so that "low" rather than "high" is the baseline group:

```r
cuse <- transform(cuse,
                  propUsing=using/(using+notUsing),
                  tot=using+notUsing,
                  education=relevel(education,"low"))
```

`ggplot` tricks:

- use `label_both` in the `facet_grid` specification to get the subplots labelled by their factor name as well as the level name
- use `aes(x=as.numeric(age))` to convince ggplot to connect the factor levels on the *x* axis with lines; use `size=0.5` to make the lines a little skinnier

```r
(gg1 <- ggplot(cuse,aes(x=age,y=propUsing,size=tot,colour=wantsMore))+
  facet_grid(.~education,labeller=label_both)+
  geom_point(alpha=0.9)+
  geom_line(aes(x=as.numeric(age)),size=0.5)+zmargin)
```



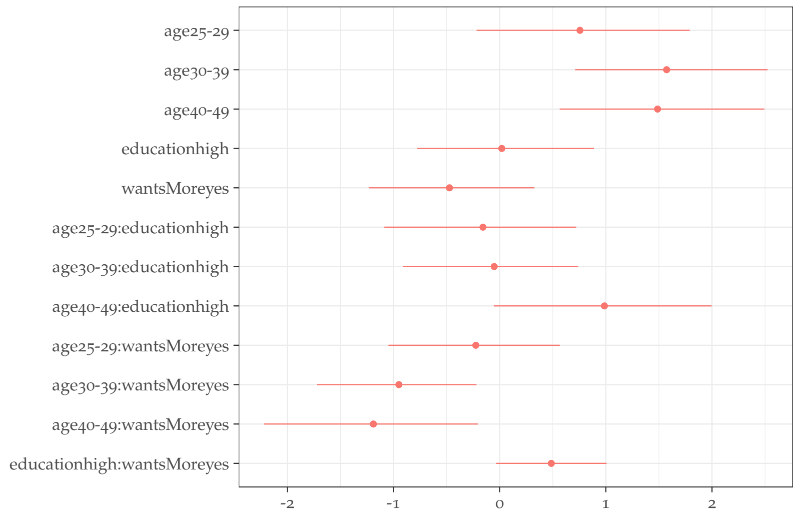
We could fit the three-way interaction, but it would be a bit silly because there would be as many parameters as observations (this is called a *saturated model*. It would probably be more sensible to worry only about two-way interactions:

```
fit2 <- glm(cbind(using,notUsing)~(age+education+wantsMore)^2,
            family=binomial,
            data=cuse)
library(aods3)
gof(fit2)

## D  = 2.4415, df = 3, P(>D) = 0.4859584
## X2 = 2.5153, df = 3, P(>X2) = 0.4725266
```

There do indeed seem to be important two-way interactions:

```
drop1(fit2,test="Chisq")

## Single term deletions
##
## Model:
## cbind(using, notUsing) ~ (age + education + wantsMore)^2
##                     Df Deviance     AIC     LRT Pr(>Chi)
## <none>                   2.4415  99.949
## age:education        3  10.8240 102.332  8.3826  0.03873 *
## age:wantsMore        3  13.7639 105.272 11.3224  0.01010 *
## education:wantsMore  1   5.7983 101.306  3.3568  0.06693 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
dwplot(fit2)
```



Interpreting parameters

Based on this model, I'm going to demonstrate how to answer a few specific questions:

1. what is the expected odds, and probability (according to the two-way interaction model), that a woman in the `age=25-29/education=high/wantsMore=no` category is using contraception?
2. what is the difference, in terms of log-odds, odds, and probabilities of contraceptive use, between women with low and high educations who are < 25 and don't want more children?
3. how would we find the *average* difference between low- and high-education women in the same terms?

1. To get the log-odds we need to add (intercept+[age effect]+[education effect]+ [age:education]). We *don't* need to add [wantsMore] or the interactions involved with [wantsMore] because we are in the reference level (no) for that factor.

```r
(rcoefs <- coef(fit2)[c("(Intercept)","age25-29","educationhigh",
                         "age25-29:educationhigh")])

##            (Intercept)               age25-29          educationhigh
##            -1.47099549             0.75577018             0.01935151
## age25-29:educationhigh
##            -0.15789791
```

```r
(logodds <- sum(rcoefs))

## [1] -0.8537717

(odds <- exp(logodds))  ## odds

## [1] 0.4258059

(prob <- odds/(1+odds)) ## probability

## [1] 0.2986423

## or
(prob <- plogis(logodds))

## [1] 0.2986423
```

Or

```r
newdata <- data.frame(age="25-29",education="high",wantsMore="no")
predict(fit2,newdata)

##          1
## -0.8537717

predict(fit2,newdata,type="response")

##         1
## 0.2986423
```

2. since "don't want more children" and "age< 25" are baseline levels and we are using treatment contrasts, we just need to look at the effects of education:

```r
(logodds <- coef(fit2)["educationhigh"])

## educationhigh
##    0.01935151

(odds <- exp(logodds))

## educationhigh
##       1.01954
```

The tricky part here is that *we can't calculate the difference in probabilities from the difference coefficients alone*; we need to go back and compute the individual probabilities.

```
(lowed_logodds <- coef(fit2)["(Intercept)"])

## (Intercept)
##   -1.470995

(lowed_prob <- plogis(lowed_logodds))

## (Intercept)
##   0.1867914

(highed_logodds <- sum(coef(fit2)[c("(Intercept)","educationhigh")]))

## [1] -1.451644

(highed_prob <- plogis(highed_logodds))

## [1] 0.1897487

(probdiff <- highed_prob-lowed_prob)

## (Intercept)
## 0.002957333
```

3. Switch to sum contrasts:

```
options(contrasts=c("contr.sum","contr.poly"))
## re-fit the same model:
fit2S <- glm(cbind(using,notUsing)~(age+education+wantsMore)^2,
            family=binomial,
            data=cuse)
```

We *double* the difference between the grand mean and low-education women to get the overall difference between low- and high-education women:

```
(logodds_eddiff <- abs(2*coef(fit2S)["education1"]))

## education1
##  0.4566298

## can calculate odds as above ...
```

To get the average log-odds for low-education women:

```
(logodds_loed <- sum(coef(fit2S)[c("(Intercept)","education1")]))

## [1] -1.049534
```

To get the average log-odds for high-education women we have to *subtract* the coefficient:

```
(logodds_hied <- coef(fit2S)["(Intercept)"]-coef(fit2S)["education1"])

## (Intercept)
##  -0.5929042
```

From here we can calculate the odds, probabilities, difference in probabilities as above . . .

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

Wald tests

Wald tests are based on the local curvature of the likelihood surface, and are the quickest but least reliable tests of significance. They are *marginal* (“type III”) tests, so they test all effects in the presence of all other effects (including interactions).

```
summary(fit2)

##
## Call:
## glm(formula = cbind(using, notUsing) ~ (age + education + wantsMore)^2,
##     family = binomial, data = cuse)
##
## Deviance Residuals:
##        1         2         3         4         5         6         7
## -0.56081   0.89772   0.21578  -0.46350  -0.13981   0.18708   0.07522
##        8         9        10        11        12        13        14
## -0.10841   0.50044  -0.39622  -0.42659   0.41098  -0.26671   0.12795
##       15        16
##  0.31058  -0.21319
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -1.47100    0.45337  -3.245 0.001176 **
## age25-29                     0.75577    0.50936   1.484 0.137869
## age30-39                     1.57249    0.45913   3.425 0.000615 ***
## age40-49                     1.48716    0.48904   3.041 0.002358 **
## educationhigh                0.01935    0.42172   0.046 0.963400
## wantsMoreyes                -0.47338    0.39674  -1.193 0.232802
## age25-29:educationhigh      -0.15790    0.45838  -0.344 0.730496
## age30-39:educationhigh      -0.05178    0.41840  -0.124 0.901501
## age40-49:educationhigh       0.98645    0.52114   1.893 0.058378 .
## age25-29:wantsMoreyes       -0.22536    0.41024  -0.549 0.582773
## age30-39:wantsMoreyes       -0.95014    0.38199  -2.487 0.012870 *
## age40-49:wantsMoreyes       -1.19012    0.51278  -2.321 0.020291 *
## educationhigh:wantsMoreyes   0.48617    0.26522   1.833 0.066785 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 165.7724  on 15  degrees of freedom
## Residual deviance:   2.4415  on  3  degrees of freedom
## AIC: 99.949
##
## Number of Fisher Scoring iterations: 4
```

This says that the log-odds of contraceptive use in the intercept (age$<$ 25, low) group is significantly different from zero (which means significantly different from probability=0.5); as usual this is not particularly interesting.

The significant values for the main effect here are for the parameters in the presence of the other effects, which means they are tests of differences with respect to age in the baseline (low-education) group. The interpretation of these main effects depends on the contrasts, however.

The other disadvantage of `summary`, besides its using Wald tests, is that it gives separate tests of each parameter (contrast), rather than a test of the overall effect of the factor (this only matters if the factor has more than two levels). While we can construct Wald F statistics that test the combined effect of several parameters, we might as well use a more accurate likelihood ratio test, based on comparing the goodness of fit (deviance) of two nested models.

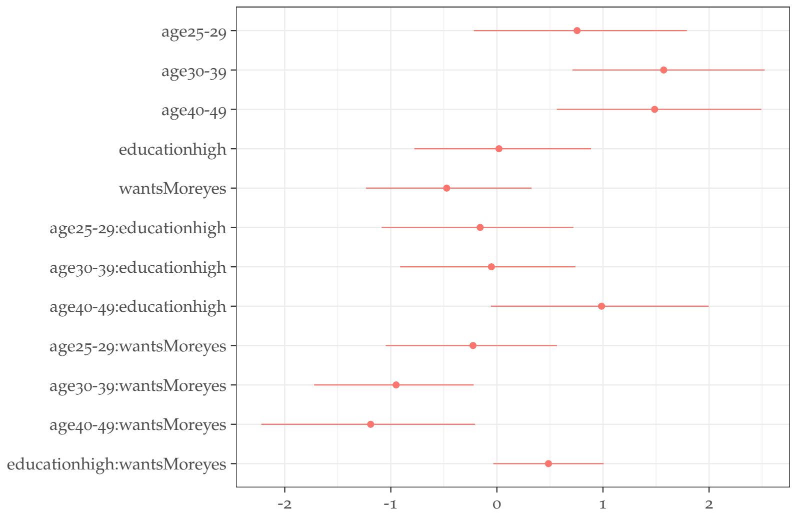
We have two choices: `drop1` and `anova`.

`drop1` tries dropping each term out of the model, but it respects marginality, so it does not try to drop main effects if there is an interaction term in the model. If we use `test="Chisq"` it gives us a likelihood ratio test (otherwise it only provides the difference and deviance and an AIC value, but not a significance test):

```
drop1(fit2,test="Chisq")

## Single term deletions
##
## Model:
## cbind(using, notUsing) ~ (age + education + wantsMore)^2
##                     Df Deviance     AIC     LRT Pr(>Chi)
## <none>                   2.4415  99.949
## age:education        3  10.8240 102.332  8.3826  0.03873 *
## age:wantsMore        3  13.7639 105.272 11.3224  0.01010 *
## education:wantsMore  1   5.7983 101.306  3.3568  0.06693 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction term is weakly significant here. I am a little bit nervous about dropping it, because (among other things) the magnitudes of the interaction terms are not that much smaller than those of the main effects:

```
dwplot(fit2)
```



If we use anova it gives us a *sequential* analysis of deviance (analogous to an analysis of variance): we again need to specify `test="Chisq"`:

```
anova(fit2,test="Chisq")

## Analysis of Deviance Table
##
## Model: binomial, link: logit
```

```
##
## Response: cbind(using, notUsing)
##
## Terms added sequentially (first to last)
##
##
##                     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                  15    165.772
## age                  3   79.192        12     86.581 < 2.2e-16 ***
## education            1    6.162        11     80.418 0.0130496 *
## wantsMore            1   50.501        10     29.917 1.191e-12 ***
## age:education        3    6.766         7     23.151 0.0797373 .
## age:wantsMore        3   17.353         4      5.798 0.0005979 ***
## education:wantsMore  1    3.357         3      2.441 0.0669289 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The significance of age quoted here is for age alone; for education, conditional on age being present in the model; for the interaction, conditional on both (it is identical to the result we got above for `drop1`).

We would get the same result (for the third time) if we explicitly dropped the interaction and did an `anova` test between the two models:

```
fit3 <- update(fit2,.~.-age:education)
anova(fit3,fit2,test="Chisq")

## Analysis of Deviance Table
##
## Model 1: cbind(using, notUsing) ~ age + education + wantsMore + age:wantsMore +
##     education:wantsMore
## Model 2: cbind(using, notUsing) ~ (age + education + wantsMore)^2
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1         6    10.8240
## 2         3     2.4415  3   8.3826  0.03873 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There are three ways to proceed with this analysis:

- if we are not really interested in the interaction at all we could split the data into two sets and analyze the low-education and high-education data separately;
- we could drop the interaction (i.e. assume that age and education

do *not* interact);

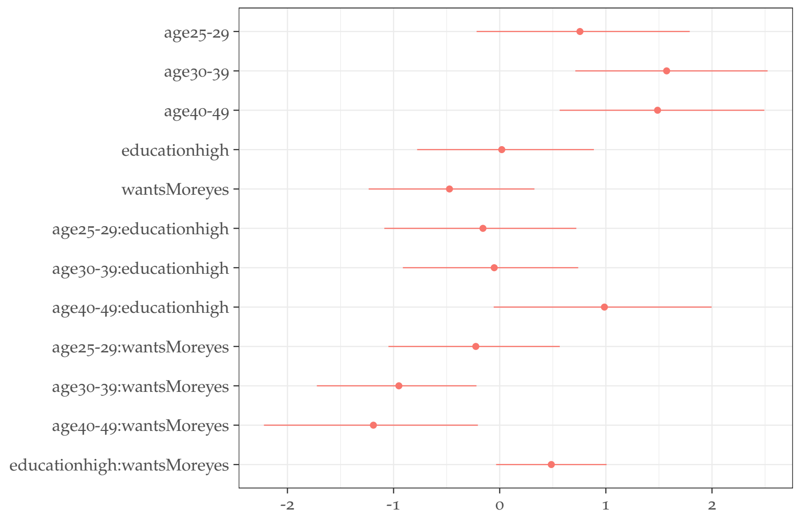
- we could set sum-to-zero contrasts and do “type III” analyses, estimating the average effect of age across education levels and vice versa.

Method 2: drop the interaction. We’ve already fitted the model (`fit2`), now we just have to look at it:

Once we have gotten rid of the interaction, the effect of education appears significant:

```
drop1(fit2,test="Chisq")

## Single term deletions
##
## Model:
## cbind(using, notUsing) ~ (age + education + wantsMore)^2
##                     Df Deviance     AIC     LRT Pr(>Chi)
## <none>                   2.4415  99.949
## age:education        3  10.8240 102.332  8.3826  0.03873 *
## age:wantsMore        3  13.7639 105.272 11.3224  0.01010 *
## education:wantsMore  1   5.7983 101.306  3.3568  0.06693 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dwplot(fit2)
```



```
options(contrasts=c("contr.sum","contr.poly"))
fit1S <- glm(cbind(using,notUsing)~age*education,family=binomial,
            data=cuse)
```

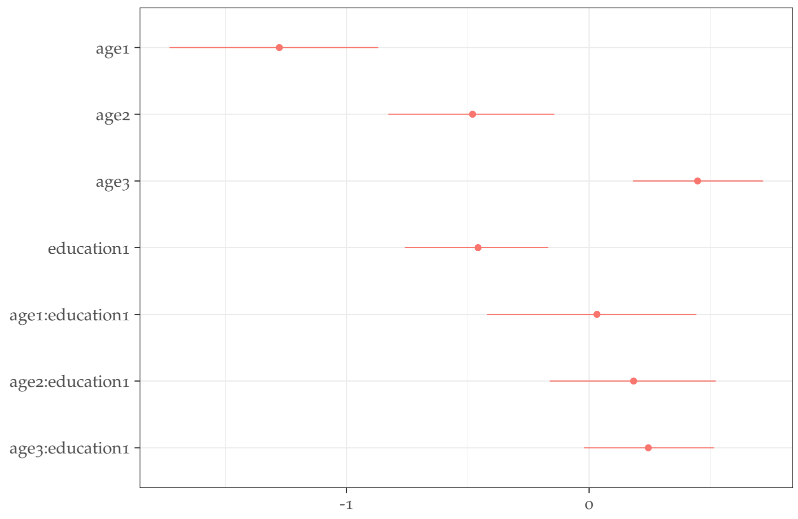
The main effects parameters now represent averages: for example, `age1` represents the difference between age (< 25) and age $(25 - -29)$ across both education levels . . .

```
summary(fit1S)

##
## Call:
## glm(formula = cbind(using, notUsing) ~ age * education, family = binomial,
##     data = cuse)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7849  -1.0158  -0.0602   1.3826   3.4456
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.76622    0.07301 -10.495  < 2e-16 ***
## age1            -0.87466    0.14936  -5.856 4.74e-09 ***
## age2            -0.32906    0.11930  -2.758  0.00581 **
## age3             0.30631    0.09369   3.269  0.00108 **
## education1      -0.22172    0.07301  -3.037  0.00239 **
## age1:education1  0.02205    0.14936   0.148  0.88264
## age2:education1  0.12561    0.11930   1.053  0.29238
## age3:education1  0.16727    0.09369   1.785  0.07420 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 165.772  on 15  degrees of freedom
## Residual deviance:  73.033  on  8  degrees of freedom
## AIC: 160.54
##
## Number of Fisher Scoring iterations: 4
```

We can use the `.~.` trick to get `drop1` to test all the terms in the model (not just the marginal ones):

```
drop1(fit1S,test="Chisq",.~.)

## Single term deletions
##
## Model:
## cbind(using, notUsing) ~ age * education
##                Df Deviance    AIC    LRT  Pr(>Chi)
## <none>               73.033 160.54
## age             3  152.440 233.95 79.407 < 2.2e-16 ***
## education       1   82.804 168.31  9.771  0.001773 **
## age:education  3   80.418 161.93  7.385  0.060575 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dwplot(fit1S)
```



confint gives us confidence intervals (95% by default) on the individual parameters. `confint.default` constructs Wald intervals; `confint` constructs likelihood profile intervals, which are more accurate — but you can see in this case that they're hardly different:

```
confint.default(fit1S)

##                        2.5 %      97.5 %
## (Intercept)     -0.90931508 -0.62313297
## age1            -1.16738905 -0.58192265
## age2            -0.56288205 -0.09524232
## age3             0.12268409  0.48993966
## education1      -0.36480902 -0.07862691
## age1:education1 -0.27068499  0.31478141
## age2:education1 -0.10820972  0.35943001
## age3:education1 -0.01635686  0.35089870

confint(fit1S)

## Waiting for profiling to be done...

##                       2.5 %      97.5 %
## (Intercept)     -0.91214655 -0.62513376
## age1            -1.18500335 -0.59507053
## age2            -0.56679298 -0.09800231
## age3             0.12356065  0.49130391
## education1      -0.36825983 -0.08135303
## age1:education1 -0.28756291  0.30273336
## age2:education1 -0.11127414  0.35760550
## age3:education1 -0.01476205  0.35293583
```

They're likely to be most different for small data sets and data sets with small numbers of samples per observation: although this data set is only 16 rows, it represents a sample of 1607 total individuals (`sum(cuse$tot)`).

pseudo-R^2 measures

The UCLA statistics site has a very nice description of pseudo-R^2 measures.

- fraction of variance explained
- model improvement
- fraction of deviance explained: (dev(null)-dev(model))/dev(null) ("McFadden"):

```
with(g1,1-deviance/null.deviance)

## [1] 0.7973723
```

- correlation ("Efron"):

```
cor(lizards$gfrac,predict(g1,type="response"))^2

## [1] 0.6350147
```

- Cox and Snell: average deviance explained

$$1 - (L(\text{null})/L(\text{full}))^{2/n}$$

  (i.e. look at proportion on the likelihood scale, not the log-likelihood scale)

- Nagelkerke: Cox and Snell, adjusted to max=1

```
descr::LogRegR2(g1)

## Chi2                 55.89726
## Df                   5
## Sig.                 8.532219e-11
## Cox and Snell Index  0.911991
## Nagelkerke Index     0.9574288
## McFadden's R2        0.7973723
```

References

Bliss, C. I. (1935). The calculation of the dosage-mortality curve. *Annals of Applied Biology* 22(1), 134–167.

Dobson, A. J. and A. Barnett (2008, May). *An Introduction to Generalized Linear Models, Third Edition* (3 ed.). Chapman and Hall/CRC.

McCullagh, P. and J. A. Nelder (1989). *Generalized Linear Models*. London: Chapman and Hall.