Logistic and binomial regression

Ben Bolker

September 30, 2018

Version: 2018-09-30 10:25:52

```
library(ggplot2)
theme_set(theme_bw())
library(grid)
zmargin <- theme_update(panel.spacing=unit(0,"lines"))
library(dotwhisker)
library(descr)
```

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:

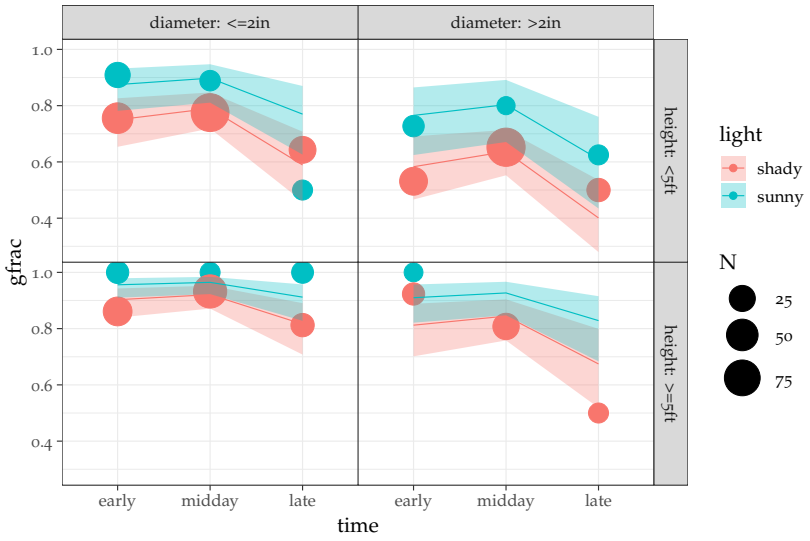
```
predFrame <- data.frame(newdata,gfrac=pred,lwr,upr)
```

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)
```

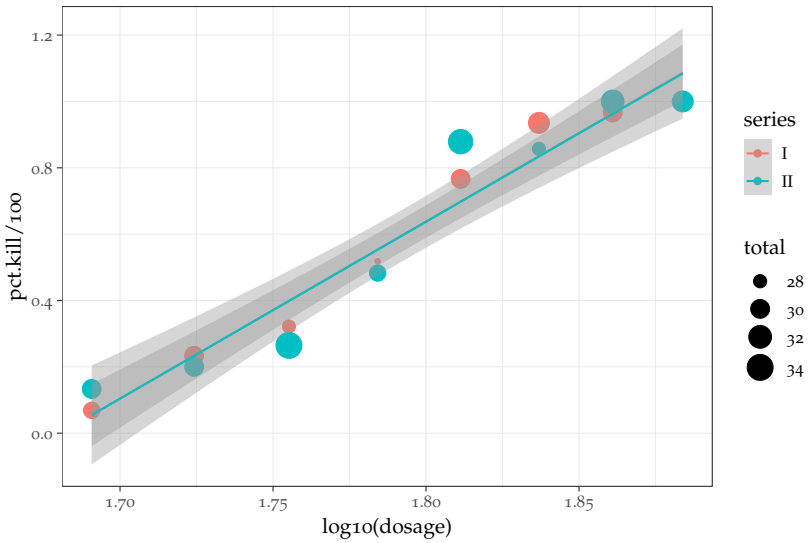


CIs on nonlinear functions of parameters Tricky. An example presented in [1], originally from [2]:

```
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),
                family=binomial(link="probit")))

## Warning:  Ignoring unknown parameters:  family
```

[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



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

```
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

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

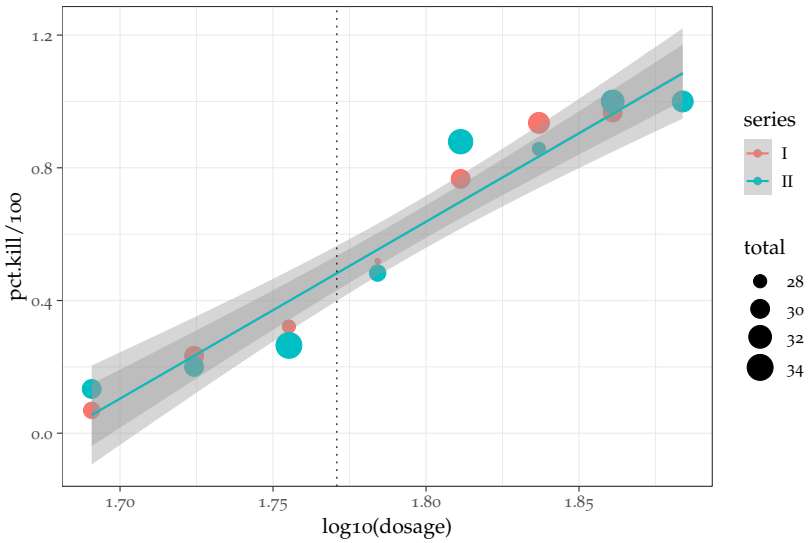
(the units work out correctly too)

So the estimate is

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

Double-check graphically:

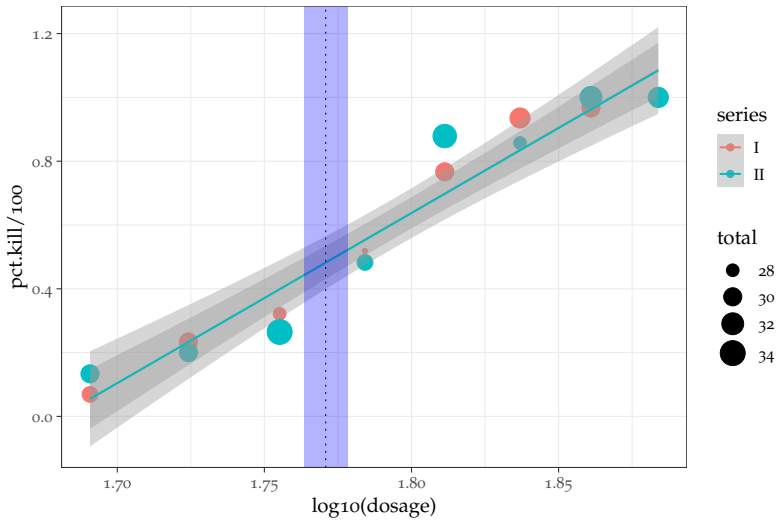
```
pplot+geom_vline(xintercept=ld50,linetype=3)
```



But what about the confidence intervals?

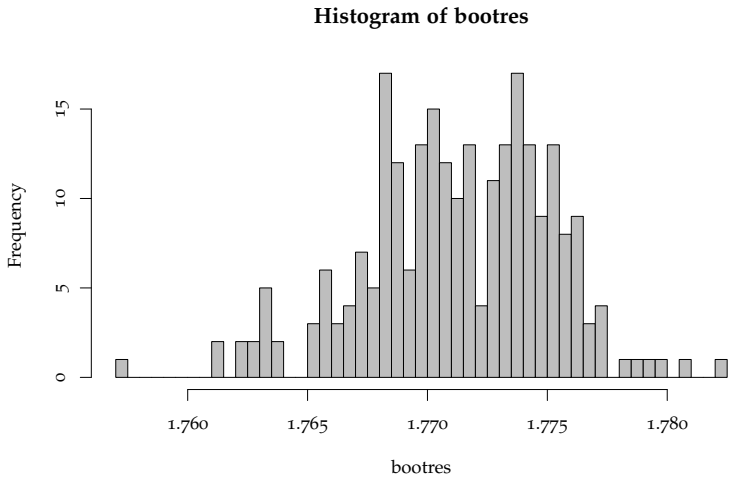
- delta method: If we want to compute the variance of $f(x,y,z)$ and $g = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$ then the variance is gVg^T (which reduces to $\mathrm{CV}^2(f(x,y)) = \mathrm{CV}^2(x) + \mathrm{CV}^2(y)$ for the case of independent 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 <- sqrt(ld50_var)
deltaCI <- ld50+c(-1,1)*1.96*ld50_se

## Warning in c(-1, 1) * 1.96 * ld50_se:  Recycling array
of length 1 in vector-array arithmetic is deprecated.
##  Use c() or as.vector() instead.

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

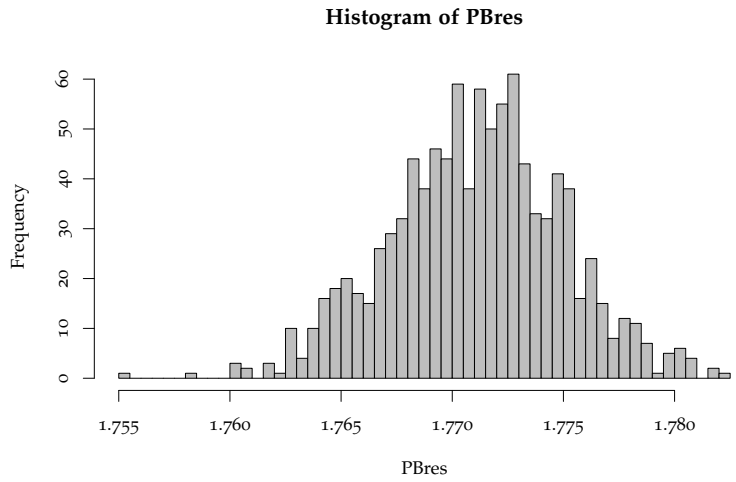
```
bootres <- numeric(250)
for (i in 1:250) {
    bootdat <- beetle[sample(nrow(beetle),replace=TRUE),]
    bootmodel <- update(g2,data=bootdat)
    bootcc <- coef(bootmodel)
    bootres[i] <- -bootcc[1]/bootcc[2]
}
hist(bootres,col="gray",breaks=50)
```




```
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:

```
rbind(deltaCI,bootCI,PBCI)

##             2.5%    97.5%
## deltaCI 1.763397 1.778304
## bootCI  1.763040 1.777379
## PBCI    1.763498 1.778639
```

Logistic example

```
## data from http://data.princeton.edu/wws509/datasets/cuse.dat
try(download.file("http://data.princeton.edu/wws509/datasets/cuse.dat",
                  dest="../data/cuse.dat"),
    silent=TRUE)

## Warning in download.file("http://data.princeton.edu/wws509/datasets/cuse.dat",
:  URL 'http://data.princeton.edu/wws509/datasets/cuse.dat':
status was 'Couldn't resolve host name'

cuse <- read.table("../data/cuse.dat",header=TRUE)
```

```
## Warning in file(file, "rt"):  cannot open file '../data/cuse.dat':
No such file or directory
## Error in file(file, "rt"):  cannot open the connection
```

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

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

## Error in transform(cuse, propUsing = using/(using + notUsing),
tot = using + :  object 'cuse' not found
```

`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

```
(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)

## Error in ggplot(cuse, aes(x = age, y = propUsing, size =
tot, colour = wantsMore)):  object 'cuse' not found
```

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)

## Error in is.data.frame(data):  object 'cuse' not found

library(aods3)
gof(fit2)

## Error in deviance(object):  object 'fit2' not found
```

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

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

## Error in drop1(fit2, test = "Chisq"):  object 'fit2' not
found
```

```
dwplot(fit2)

## Error in is.data.frame(x):  object 'fit2' not found
```

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.

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

## Error in coef(fit2):  object 'fit2' not found

(logodds <- sum(rcoefs))

## Error in eval(expr, envir, enclos):  object 'rcoefs'
not found

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

## Error in eval(expr, envir, enclos):  object 'logodds'
not found
```

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

## Error in eval(expr, envir, enclos):  object 'odds' not
found

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

## Error in plogis(logodds):  object 'logodds' not found
```

Or

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

## Error in predict(fit2, newdata):  object 'fit2' not
found

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

## Error in predict(fit2, newdata, type = "response"):
object 'fit2' not found
```

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:

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

## Error in coef(fit2):  object 'fit2' not found

(odds <- exp(logodds))

## Error in eval(expr, envir, enclos):  object 'logodds'
not found
```

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)"])

## Error in coef(fit2):  object 'fit2' not found

(lowed_prob <- plogis(lowed_logodds))
```

```
## Error in plogis(lowed_logodds):  object 'lowed_logodds'
not found

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

## Error in coef(fit2):  object 'fit2' not found

(highed_prob <- plogis(highed_logodds))

## Error in plogis(highed_logodds):  object 'highed_logodds'
not found

(probdiff <- highed_prob-lowed_prob)

## Error in eval(expr, envir, enclos):  object 'highed_prob'
not found
```

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)

## Error in is.data.frame(data):  object 'cuse' not found
```

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"]))

## Error in coef(fit2S): object 'fit2S' not found

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

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

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

## Error in coef(fit2S): object 'fit2S' not found
```

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

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

## Error in coef(fit2S): object 'fit2S' not found
```

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)

## Error in summary(fit2):  object 'fit2' not found
```

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")
```

```
## Error in drop1(fit2, test = "Chisq"):  object 'fit2' not
found
```

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)
```

```
## Error in is.data.frame(x):  object 'fit2' not found
```

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")
```

```
## Error in anova(fit2, test = "Chisq"):  object 'fit2' not
found
```

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)
```

```
## Error in update(fit2, .  ~ .  - age:education):  object
'fit2' not found
```

```
anova(fit3,fit2,test="Chisq")
```

```
## Error in anova(fit3, fit2, test = "Chisq"):  object 'fit3'
not found
```

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")
```

```
## Error in drop1(fit2, test = "Chisq"):  object 'fit2' not
found
```

```
dwplot(fit2)
```

```
## Error in is.data.frame(x):  object 'fit2' not found
```

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

```
## Error in is.data.frame(data):  object 'cuse' not found
```

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)
```

```
## Error in summary(fit1S): object 'fit1S' not found
```

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",.~.)
```

```
## Error in drop1(fit1S, test = "Chisq", .  ~ .):  object
'fit1S' not found
```

```
dwplot(fit1S)
```

```
## Error in is.data.frame(x):  object 'fit1S' not found
```

`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)

## Error in coef(object):  object 'fit1S' not found

confint(fit1S)

## Error in confint(fit1S): object 'fit1S' not found
```

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.