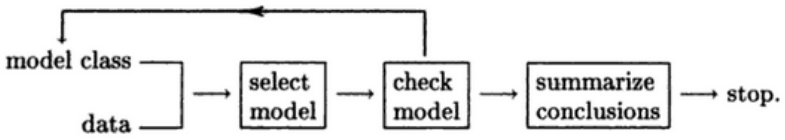
Logistic regression

Ben Bolker

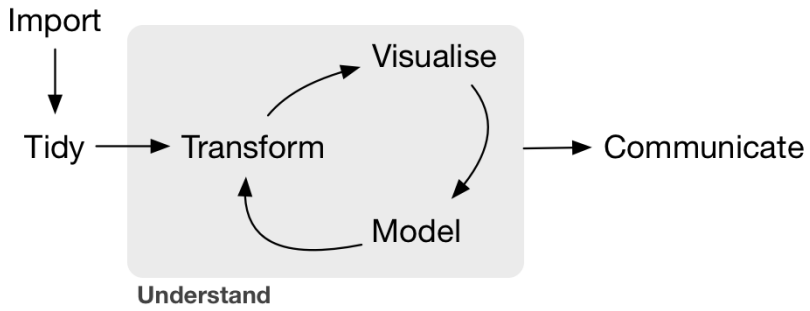
October 3, 2018

modeling

data analysis road map



(McCullagh and Nelder, 1989)



from Hadley Wickham
(https://jules32.github.io/2016-07-12-Oxford/dplyr_tidyr/)

These are good, but they don't address the **data snooping** problem.

---

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 & quantitative diagnostics (return to 5? )
9. interpret parameters; inference; plot results

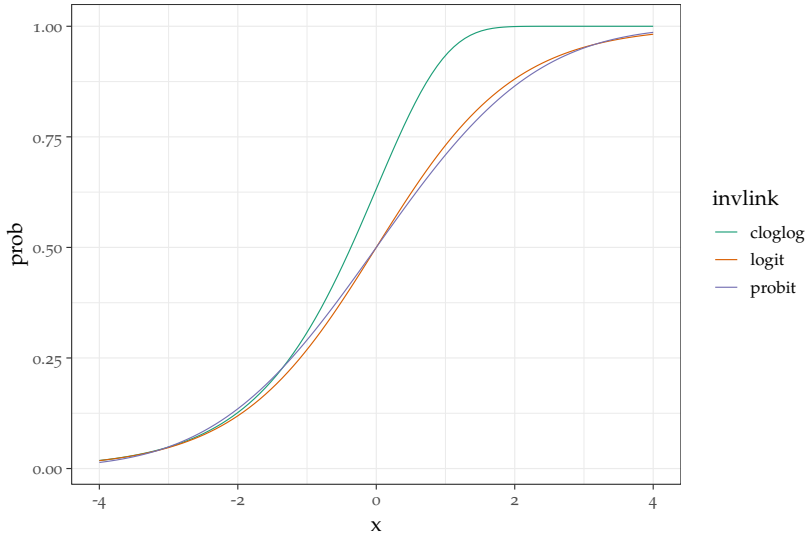
---

basics

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.




```
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
library(arm)    ## binnedplot
library(dplyr)  ## tidyverse!
library(DescTools)
library(broom)  ## for augment()
```

Contraception data example

```
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
```

See here for more documentation.
Given these variables, what model do we think we want to use?
Visualize! Try some ggplots (univariate graphs are OK but multivariate graphs are almost always more informative . . .)

```
gg0 <- ggplot(Contraception,aes(age,use,colour=urban))+
    stat_sum(alpha=0.5)+facet_wrap(~livch,labeller=label_both)
gg0 + geom_smooth(aes(group=1))
```

Hard to summarize 0/1 values!
Alternative approach: binning (also see Faraway). (Transform!)

```
## transform via tidyverse ...
cc <- (Contraception
    %>% mutate(
            ## numeric (0/1) version of 'uses contraception'
            use_n=as.numeric(use)-1)
)
cc_agg0 <- (cc
    %>% group_by(livch,urban,age)
    %>% summarise(prop=mean(use_n),
            n=length(use),
            se=sqrt(prop*(1-prop)/n))
)
```

Plot:

```
ggplot(cc_agg0,aes(age,prop,colour=urban))+
    geom_pointrange(aes(ymin=prop-2*se,
                        ymax=prop+2*se))+
    facet_wrap(~livch,labeller=label_both)
```

Bin more coarsely:

```
## specify categories; compute midpoints as well
age_breaks <- seq(-15,20,by=5)
age_mids <- (age_breaks[-1]+age_breaks[-length(age_breaks)])/2
cc_agg <- (cc
    ## discrete age categories
    %>% mutate(age_cat=cut(age,breaks=age_breaks))
    %>% group_by(age_cat,urban,livch)
    %>% summarise(
            prop=mean(use_n),
            n=length(use),
            se=sqrt(prop*(1-prop)/n)
        )
    ## numeric values of age categories
    %>% mutate(age_mid=age_mids[as.numeric(age_cat)])
)
```

Plot:

```
## use numeric response rather than Y/N response
gg0B <- ggplot(cc,aes(age,use_n,colour=urban))+
    stat_sum(alpha=0.5)+facet_wrap(~livch,labeller=label_both)+
    labs(y="prob of contraceptive use")
gg_bin <- gg0B+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(0.5,2))
```

How should we adjust our model specification based on this information?

Suppose we use a model with a quadratic function of age plus all three-way interactions:

```
model1 <- glm(use_n ~ urban*(age+I(age^2))*livch,
              data=cc,
              family=binomial,
              x=TRUE  ## include model matrix in output
              )
```

Explore diagnostics (`plot()`; `DHARMa::simulateResiduals()`; `arm::binnedplot`; `mgcv::qq.gam`).

Q-Q plot is useless for logistic regression; we know that the responses are conditionally Bernoulli-distributed! **Quantile residuals** [1] overcome many of the problems of GLM diagnostics, at the price of lots more computation.

[1] Ben, M. G. and V. J. Yohai (2004, March). Quantile-Quantile Plot for Deviance Residuals in the Generalized Linear Model. *Journal of Computational and Graphical Statistics 13*(1), 36–47; and Hartig, F. (2018). *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models*. R package version 0.2.0

```
## default plots: ugh!
plot(model1)
## binned plot
arm::binnedplot(fitted(model1),residuals(model1))
## smoothing via ggplot
ggplot(broom::augment(model1),aes(.fitted,.resid)) +
    geom_point() + geom_smooth()
## Q-Q of quantile residuals
mgcv::qq.gam(model1,pch=1)
## ... simulated quantil residuals ...
mgcv::qq.gam(model1,pch=1,rep=1000)
## alternative simulated residuals
plot(DHARMa::simulateResiduals(model1),pch=".")
```

If you really need a global goodness-of-fit test: **Hosmer-Lemeshow test** (very common) dominated by Cessie-van Houwelingen test [2].

[2] le Cessie, S. and J. C. van Houwelingen (1991, December). A goodness-of-fit test for binary regression models, based on smoothing methods. *Biometrics 47*(4), 1267–1282; and Hosmer, D. W., T. Hosmer, S. L. Cessie, and S. Lemeshow (1997, May). A Comparison of Goodness-of-Fit Tests for the Logistic Regression Model. *Statistics in Medicine 16*(9), 965–980

```
DescTools::HosmerLemeshowTest(fit=fitted(model1),
                              obs=model1$y,
                              X=model1$x)
```

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(model1,1-deviance/null.deviance)

## [1] 0.0733366
```

- correlation ("Efron"):

```
cor(cc$use_n,predict(model1,type="response"))^2

## [1] 0.09300837
```

- 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(model1)

## Chi2                 190.0085
## Df                   23
## Sig.                 0
## Cox and Snell Index  0.09357444
## Nagelkerke Index     0.1267833
## McFadden's R2        0.0733366
```

Plot predictions

```
gg_bin + geom_smooth(method="glm",
                     method.args=list(family=binomial),
                     formula=y~x+I(x^2)
                     )
```

Or by hand: `predict` function.
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(model,newdata=newdata,se.fit=TRUE)$se.fit
lwr <- plogis(pred0-2*pse)  ## or qnorm(0.025)
upr <- plogis(pred0+2*pse)  ## or qnorm(0.975)
```

Note:

- using the inverse-link function to back-transform the *standard errors* never (??) makes sense: if you want to back-transform them (approximately), you have to multiply them by $(d\mu/d\eta)$, i.e. use `dlogis` or the `mu.eta` component of `model$family`
- 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 approach 0 or 1), it's probably best to use the approach above

```
## prediction frame: all combinations of variables
pframe <- with(Contraception,
               expand.grid(age=unique(age),
                           livch=levels(livch),
                           urban=levels(urban)))
predfun <- function(model) {
    pp <- predict(model, newdata=pframe, type="link", se.fit=TRUE)
    linkinv <- family(model)$linkinv
    pframe$use_n <- linkinv(pp$fit)
    pframe$lwr <- linkinv(pp$fit-2*pp$se.fit)
    pframe$upr <- linkinv(pp$fit+2*pp$se.fit)
    return(pframe)
}
pp1 <- predfun(model1)
```

Posterior predictive simulations

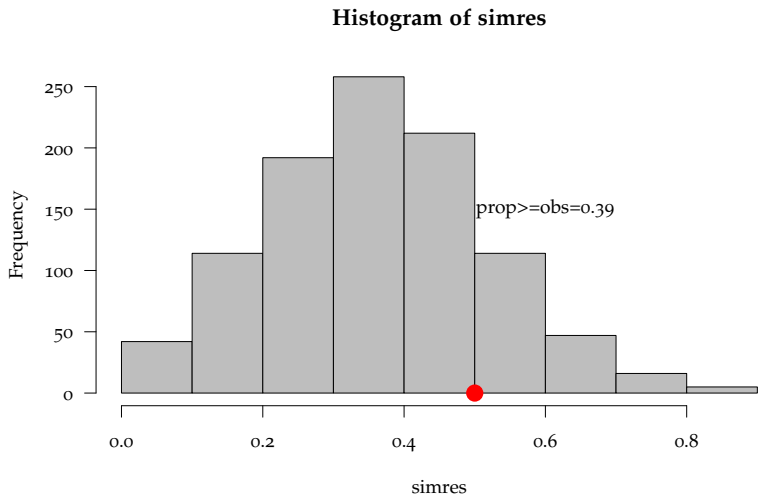
Pick a summary statistic that matters (e.g. the proportion of urban women with no living children whose age is within 1 year of the mean who are using contraception) and simulate predictions from the model: see how they match the observed value. Can we reject the null hypothesis that the model is OK?

```
ppfun <- function(dd) {
    w <- which(dd$urban=="Y" & dd$livch=="0" & abs(dd$age)<1)
    return(mean(dd$use_n[w]))
}
ppfun(cc)  ## observed value from data

## [1] 0.5

ss <- simulate(model1,1000)
```

```
simres <- rep(NA,1000)
newcc <- cc
for (i in 1:1000) {
    newcc$use_n <- ss[,i]
    simres[i] <- ppfun(newcc)
}
```

Plot results:

```
par(las=1)
hist(simres,col="gray")
points(ppfun(cc),0,col="red",cex=2,pch=16)
p_upr <- mean(simres>=ppfun(cc))
p_lwr <- mean(simres<=ppfun(cc))
text(0.6,150,paste0("prop>=obs=",round(p_upr,2)))
```




```
## 2-tailed p-value
2*min(p_upr,p_lwr)

## [1] 0.788
```

Simplify model

With caution!

```
drop1(model1,test="Chisq")

## Single term deletions
##
```

```
## Model:
## use_n ~ urban * (age + I(age^2)) * livch
##                      Df Deviance    AIC     LRT Pr(>Chi)
## <none>                    2400.9 2448.9
## urban:age:livch       3   2401.2 2443.2 0.26485   0.9665
## urban:I(age^2):livch  3   2401.8 2443.8 0.89356   0.8270

## three-way interactions NS?
model2 <- update(model1, . ~ (urban+(age+I(age^2)+livch))^2)
drop1(model2,test="Chisq")

## Single term deletions
##
## Model:
## use_n ~ urban + age + I(age^2) + livch + urban:age + urban:I(age^2) +
##     urban:livch + age:I(age^2) + age:livch + I(age^2):livch
##                Df Deviance    AIC    LRT Pr(>Chi)
## <none>              2402.1 2440.1
## urban:age       1   2402.1 2438.1 0.0068  0.93452
## urban:I(age^2)  1   2403.0 2439.0 0.9059  0.34120
## urban:livch     3   2404.4 2436.4 2.2447  0.52320
## age:I(age^2)    1   2402.1 2438.1 0.0005  0.98302
## age:livch       3   2409.8 2441.8 7.6792  0.05313 .
## I(age^2):livch  3   2403.4 2435.4 1.3214  0.72405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## two-way interactions NS?
model3 <- update(model1, . ~ (urban+(age+I(age^2)+livch)))
## or LRT
anova(model1,model2,model3,test="Chisq")

## Analysis of Deviance Table
##
## Model 1: use_n ~ urban * (age + I(age^2)) * livch
## Model 2: use_n ~ urban + age + I(age^2) + livch + urban:age + urban:I(age^2) +
##     urban:livch + age:I(age^2) + age:livch + I(age^2):livch
## Model 3: use_n ~ urban + age + I(age^2) + livch
##   Resid. Df Resid. Dev  Df Deviance Pr(>Chi)
## 1      1910     2400.9
## 2      1915     2402.1  -5  -1.2276   0.9422
## 3      1927     2417.7 -12 -15.5305   0.2137
```

Inference on the selected model

```
car::Anova(model3)

## Analysis of Deviance Table (Type II tests)
##
## Response: use_n
##          LR Chisq Df Pr(>Chisq)
## urban      52.849  1  3.602e-13 ***
## age         0.265  1      0.607
## I(age^2)   39.070  1  4.088e-10 ***
## livch      33.333  3  2.739e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

drop1(model3,test="Chisq")

## Single term deletions
##
## Model:
## use_n ~ urban + age + I(age^2) + livch
##          Df Deviance    AIC    LRT  Pr(>Chi)
## <none>        2417.7 2431.7
## urban     1   2470.5 2482.5 52.849 3.602e-13 ***
## age       1   2417.9 2429.9  0.265     0.607
## I(age^2)  1   2456.7 2468.7 39.070 4.088e-10 ***
## livch     3   2451.0 2459.0 33.333 2.739e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coefficient or "dot-whisker" plots of the reduced model, with and without standardization by 2σ of the predictors:

```
dw1 <- dwplot(model3)+geom_vline(xintercept=0,lty=2)
dw2 <- dwplot(model3,by_2sd=FALSE)+geom_vline(xintercept=0,lty=2)
```

Can compare the effect of dropping interactions (carefully!)

```
mod_list <- list(full=model1,twoway=model2,reduced=model3)
dw_comb <- dwplot(mod_list)+ geom_vline(xintercept=0,lty=2)
```

```
pp_list <- lapply(mod_list,predfun)
pp_frame <- dplyr::bind_rows(pp_list,.id="method")
gg_compare_pred <- gg0B + geom_line(data=pp_frame,
```

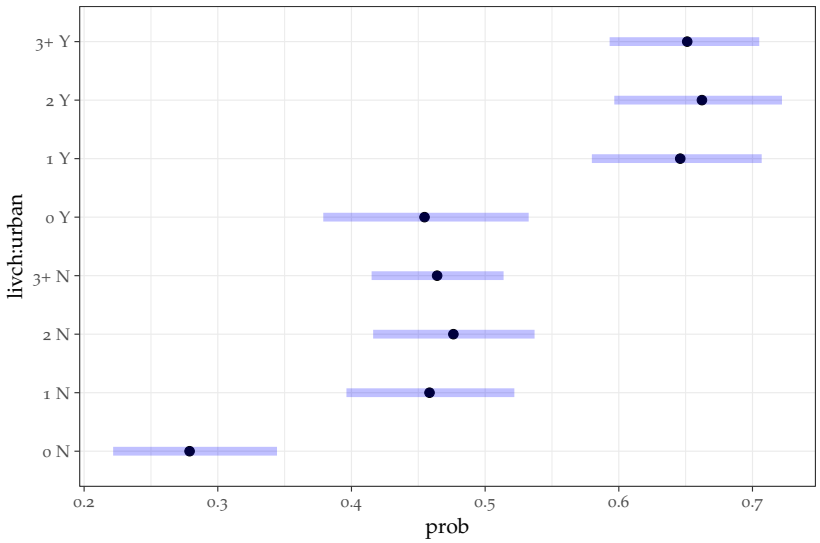
```
                                    aes(linetype=method))
pp3 <- pp_list[[3]]
gg_model3 <- gg0B + geom_line(data=pp3)+
    geom_ribbon(data=pp3,aes(ymin=lwr,ymax=upr,fill=urban),colour=NA,alpha=0.2)
```

```
summary(model3)
```

```
##
## Call:
## glm(formula = use_n ~ urban + age + I(age^2) + livch, family = binomial,
##     data = cc, x = TRUE)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4738  -1.0369  -0.6683   1.2401   1.9765
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.9499521  0.1560118  -6.089 1.14e-09 ***
## urbanY       0.7680975  0.1061916   7.233 4.72e-13 ***
## age          0.0045837  0.0089084   0.515    0.607
## I(age^2)    -0.0042865  0.0007002  -6.122 9.23e-10 ***
## livch1       0.7831128  0.1569096   4.991 6.01e-07 ***
## livch2       0.8549040  0.1783573   4.793 1.64e-06 ***
## livch3+      0.8060251  0.1784817   4.516 6.30e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2590.9  on 1933  degrees of freedom
## Residual deviance: 2417.7  on 1927  degrees of freedom
## AIC: 2431.7
##
## Number of Fisher Scoring iterations: 4
```

The `emmeans` package has a whole bunch of convenience functions for computing and plotting "expected marginal means", which are the generalization of "least-squares means", i.e. effects averaged across categories in various sensible ways:

```
plot(emmeans::emmeans(model3,~livch*urban,type="response"))
```



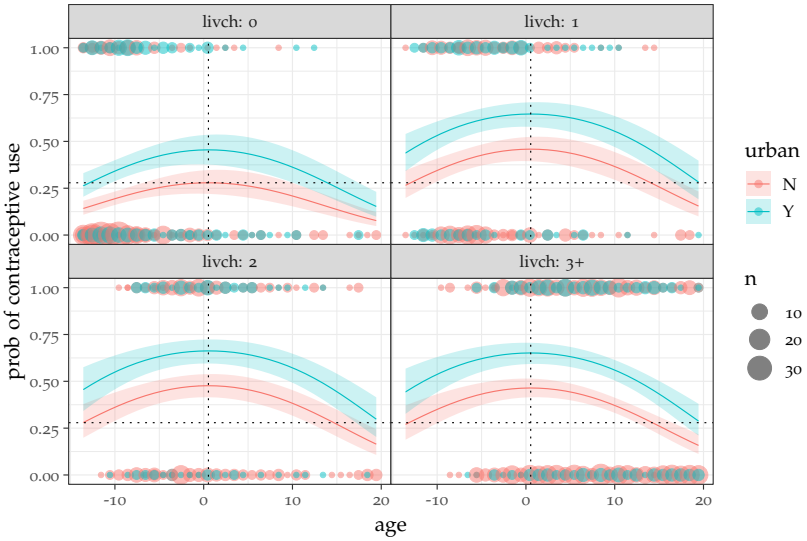
Confidence intervals on nonlinear functions of predictions

Suppose we're interested in some value that can be computed as a *nonlinear* function of the parameters. For example, suppose want to estimate the age at which contraceptive use peaks, and the level of contraception use at that point. If a quadratic is parameterized as $\beta_0 + \beta_1 x + \beta_2 x^2$, then the critical values occurs where $\beta_1 + 2\beta_2 \hat{x} = 0 \rightarrow \hat{x} = -\beta_1/(2\beta_2)$, and the value is $\beta_0 - \beta_1^2/(2\beta_2) + \beta_2(\beta_1^2/(4\beta_2^2)) = \beta_0 - \beta_1^2/(4\beta_2)$. (Since the link function if monotonic, we don't have to worry about that nonlinearity for these purposes.) Since we have only an additive model,

```
cc3 <- as.list(coef(model3))
(use_peak <- with(cc3,
                  c(-age/(2*`I(age^2)`),
                    plogis(`(Intercept)` - age^2/(4*`I(age^2)`)))))

## [1] 0.5346756 0.2791410
```

So the peak is half a year above the mean age, at about 28% use (note that peak height varies among categories; this is the prediction for the baseline category {urban, livch=0}). These numbers seem reasonable based on what we've seen so far, but checking graphically:

```
gg_model3+
    geom_vline(xintercept=use_peak[1],linetype=3)+
    geom_hline(yintercept=use_peak[2],linetype=3)
```



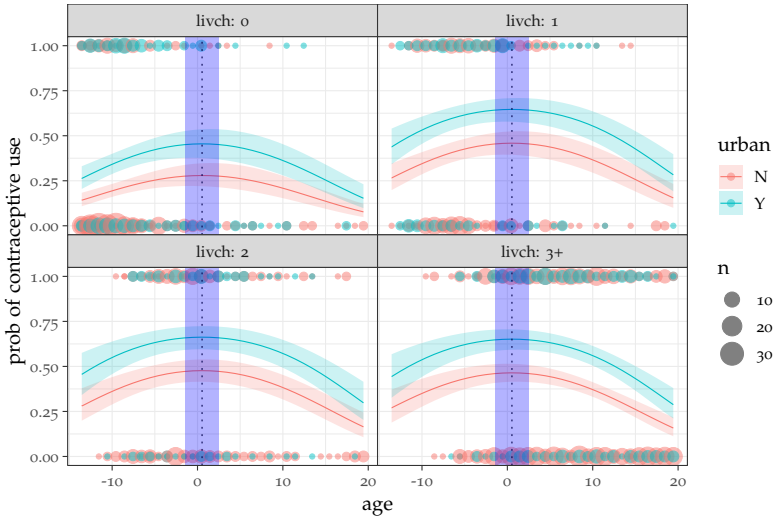
Getting the confidence intervals on these values is harder ...

- **delta method**: If we want to compute the variance on the peak location? 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 <- rep(0,length(coef(model3)))
names(grad) <- names(coef(model3))
## deriv of b1/(2*b2) = {1/(2*b2), -b1/b2^2}
grad[c("age","I(age^2)")] <-
    with(cc3,c(1/(2*`I(age^2)`),-age/`I(age^2)`^2))
peak_var <- t(grad) %*% vcov(model3) %*% grad
peak_se <- c(sqrt(peak_var)) ## c() converts from matrix to vector (= scalar)
deltaCI <- use_peak[1]+c(-1,1)*2*peak_se
```

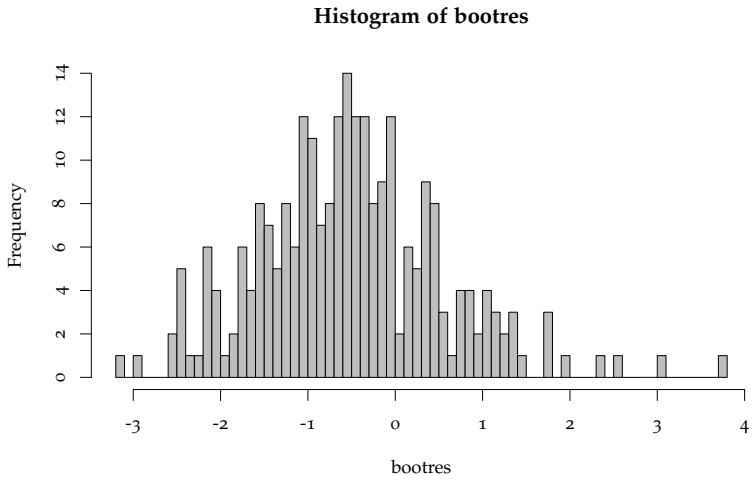
Plot:

```
gg_model3+geom_vline(xintercept=use_peak[1],linetype=3)+
    annotate("rect",
           xmin=deltaCI[1],
           xmax=deltaCI[2],
           ymin=-Inf,
           ymax=Inf,alpha=0.3,fill="blue",
             colour=NA)
```



- bootstrapping

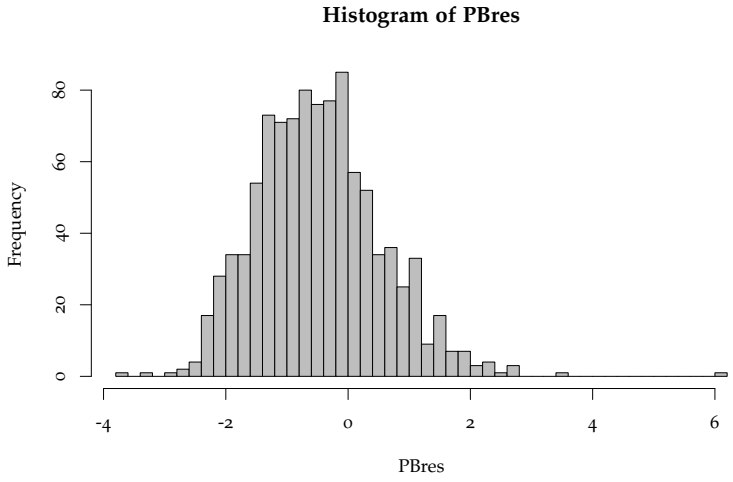
```
bootres <- numeric(250)
for (i in 1:250) {
    bootdat <- cc[sample(nrow(cc),replace=TRUE),]
    bootmodel <- update(model3,data=bootdat)
    bootcc <- coef(bootmodel)
    bootres[i] <- with(as.list(bootcc),c(age/(2*`I(age^2)`)))
}
hist(bootres,col="gray",breaks=50)
```




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

- pseudo-Bayes: MVN sample from parameters

```
library(MASS)
PBsamp <- as.data.frame(mvrnorm(1000,mu=coef(model3),Sigma=vcov(model3)))
PBres <- with(PBsamp,c(age/(2*`I(age^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,PBCI,bootCI)

##              2.5%    97.5%
## deltaCI -1.398998 2.468349
## PBCI    -2.200607 1.648390
## bootCI  -2.414925 1.727165
```

References

Ben, M. G. and V. J. Yohai (2004, March). Quantile-Quantile Plot for Deviance Residuals in the Generalized Linear Model. *Journal of Computational and Graphical Statistics 13*(1), 36–47.

Hartig, F. (2018). *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models*. R package version 0.2.0.

Hosmer, D. W., T. Hosmer, S. L. Cessie, and S. Lemeshow (1997, May). A Comparison of Goodness-of-Fit Tests for the Logistic Regression Model. *Statistics in Medicine 16*(9), 965–980.

le Cessie, S. and J. C. van Houwelingen (1991, December). A goodness-of-fit test for binary regression models, based on smoothing methods. *Biometrics* *47*(4), 1267–1282.

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