Lecture 9: Cumulative logistic regression

Nominal outcome

We will follow the alligator example from class.

```
Alligator <- read.csv('alligator_data.csv')[,-1]

# look at the data (original format)
Alligator
```

```
##        lake    sex  size    food count
## 1   Hancock   male small    fish     7
## 2   Hancock   male small  invert     1
## 3   Hancock   male small reptile     0
## 4   Hancock   male small    bird     0
## 5   Hancock   male small   other     5
## 6   Hancock   male large    fish     4
## 7   Hancock   male large  invert     0
## 8   Hancock   male large reptile     0
## 9   Hancock   male large    bird     1
## 10  Hancock   male large   other     2
## 11  Hancock female small    fish    16
## 12  Hancock female small  invert     3
## 13  Hancock female small reptile     2
## 14  Hancock female small    bird     2
## 15  Hancock female small   other     3
## 16  Hancock female large    fish     3
## 17  Hancock female large  invert     0
## 18  Hancock female large reptile     1
## 19  Hancock female large    bird     2
## 20  Hancock female large   other     3
## 21 Oklawaha   male small    fish     2
## 22 Oklawaha   male small  invert     2
## 23 Oklawaha   male small reptile     0
## 24 Oklawaha   male small    bird     0
## 25 Oklawaha   male small   other     1
## 26 Oklawaha   male large    fish    13
## 27 Oklawaha   male large  invert     7
## 28 Oklawaha   male large reptile     6
## 29 Oklawaha   male large    bird     0
## 30 Oklawaha   male large   other     0
## 31 Oklawaha female small    fish     3
## 32 Oklawaha female small  invert     9
## 33 Oklawaha female small reptile     1
## 34 Oklawaha female small    bird     0
## 35 Oklawaha female small   other     2
```

```
## 36 Oklawaha female large    fish     0
## 37 Oklawaha female large  invert     1
## 38 Oklawaha female large reptile     0
## 39 Oklawaha female large    bird     1
## 40 Oklawaha female large   other     0
## 41 Trafford   male small    fish     3
## 42 Trafford   male small  invert     7
## 43 Trafford   male small reptile     1
## 44 Trafford   male small    bird     0
## 45 Trafford   male small   other     1
## 46 Trafford   male large    fish     8
## 47 Trafford   male large  invert     6
## 48 Trafford   male large reptile     6
## 49 Trafford   male large    bird     3
## 50 Trafford   male large   other     5
## 51 Trafford female small    fish     2
## 52 Trafford female small  invert     4
## 53 Trafford female small reptile     1
## 54 Trafford female small    bird     1
## 55 Trafford female small   other     4
## 56 Trafford female large    fish     0
## 57 Trafford female large  invert     1
## 58 Trafford female large reptile     0
## 59 Trafford female large    bird     0
## 60 Trafford female large   other     0
## 61   George   male small    fish    13
## 62   George   male small  invert    10
## 63   George   male small reptile     0
## 64   George   male small    bird     2
## 65   George   male small   other     2
## 66   George   male large    fish     9
## 67   George   male large  invert     0
## 68   George   male large reptile     0
## 69   George   male large    bird     1
## 70   George   male large   other     2
## 71   George female small    fish     3
## 72   George female small  invert     9
## 73   George female small reptile     1
## 74   George female small    bird     0
## 75   George female small   other     1
## 76   George female large    fish     8
## 77   George female large  invert     1
## 78   George female large reptile     0
## 79   George female large    bird     0
## 80   George female large   other     1
```

```
# convert data to long format for modeling
alligator.long <- Alligator[rep(1:nrow(Alligator),Alligator$count),-5]
```

We will use the function `multinom()` from the `nnet` package.

```r
library(nnet)
# using default reference categories
mod1 <- multinom(food ~ lake + size + sex,
                 data=alligator.long,trace=FALSE)
# summary(mod1)

# change reference categories to line up with lecture slides
alligator.long$lake <- relevel(alligator.long$lake,ref='Hancock')
alligator.long$size <- relevel(alligator.long$size,ref='small')
alligator.long$sex <- relevel(alligator.long$sex,ref='male')

mod2 <- multinom(relevel(food,ref='fish') ~ lake + size + sex,
                 data=alligator.long,trace=FALSE)

# estimates and standard errors
# (matrix form)
summary(mod2)
```

```
## Call:
## multinom(formula = relevel(food, ref = "fish") ~ lake + size +
##     sex, data = alligator.long, trace = FALSE)
##
## Coefficients:
##         (Intercept) lakeGeorge lakeOklawaha lakeTrafford sizelarge
## bird        -2.4632    -0.5754      -1.1258       0.6617    0.7303
## invert      -2.0744     1.7805       2.6937       2.9363   -1.3362
## other       -0.9168    -0.7665      -0.7406       0.7912   -0.2905
## reptile     -2.9143    -1.1291       1.4009       1.9317    0.5571
##         sexfemale
## bird       0.6065
## invert     0.4629
## other      0.2526
## reptile    0.6276
##
## Std. Errors:
##         (Intercept) lakeGeorge lakeOklawaha lakeTrafford sizelarge
## bird         0.7739     0.7952       1.1924       0.8460    0.6523
## invert       0.6116     0.6232       0.6692       0.6874    0.4112
## other        0.4782     0.5685       0.7422       0.5879    0.4599
## reptile      0.8857     1.1927       0.8105       0.8253    0.6466
##         sexfemale
## bird       0.6888
## invert     0.3955
## other      0.4663
## reptile    0.6853
##
## Residual Deviance: 537.9
## AIC: 585.9
```

```r
# Wald 95% confidence intervals
# (on odds ratio scale)
waldci.alligator <- exp(confint(mod2))
```

We can use the likelihood ratio test to see if the `lake` effect is significant (helpful since it is a categorical covariate with more than two levels). We can't use the `anova()` function because the `nnet` package functions don't have a method for that, so we will have to compute the likelihoods under each model separately.

```r
# we already have the full model, now we fit the reduced model
mod2nolake <- multinom(relevel(food,ref='fish') ~ size + sex,
                 data=alligator.long,trace=FALSE)

# test statistic
2*(logLik(mod2)-logLik(mod2nolake))

## 'log Lik.' 50.32 (df=24)

# p-value
pchisq(2*(logLik(mod2)-logLik(mod2nolake)), # test statistic
       attr(logLik(mod2),'df')-attr(logLik(mod2nolake),'df'),# difference in
number of parameters between models
       lower.tail=FALSE)

## 'log Lik.' 1.228e-06 (df=24)
```

This shows that which lake the alligators were from is a significant predictor of their food choice, adjusting for `size` and `sex`.

Let's see what happens if we fit separate logistic models instead.

```r
foodcats <- rownames(summary(mod2)$coefficients)
indiv.mods <- list()
for(j in seq_along(foodcats)) {
  indiv.mods[[j]] <- glm(I(food==foodcats[j])~lake+size+sex,
                family=binomial,
                data=alligator.long,
                # want to confine this to data for either the baseline or
category of interest
                subset=(food %in% c('fish',foodcats[j])))
}
names(indiv.mods) <- foodcats
# lapply(indiv.mods,summary)
indiv.est <- lapply(indiv.mods,function(x) summary(x)$coef[,1])
indiv.se <- lapply(indiv.mods,function(x) summary(x)$coef[,2])

# estimates (log odds ratios)
do.call(rbind,indiv.est)

##         (Intercept) lakeGeorge lakeOklawaha lakeTrafford sizelarge
## bird        -2.4837    -0.6468      -1.1870       0.6327    0.8373
```

```
## invert      -2.3190     1.9869       3.0344       3.2924   -1.5072
## other       -0.8898    -0.7777      -0.8311       0.6913   -0.1996
## reptile     -3.5153    -1.2354       1.6260       2.1901    0.9578
##         sexfemale
## bird       0.5927
## invert     0.6744
## other      0.1818
## reptile    1.2344

# standard errors
do.call(rbind,indiv.se)

##         (Intercept) lakeGeorge lakeOklawaha lakeTrafford sizelarge
## bird         0.7926     0.8188       1.2070       0.8540    0.6740
## invert       0.6375     0.6392       0.7160       0.7389    0.4314
## other        0.4655     0.5713       0.7298       0.5670    0.4577
## reptile      1.0829     1.2240       0.8894       0.9053    0.7466
##         sexfemale
## bird       0.7076
## invert     0.4252
## other      0.4614
## reptile    0.8673
```

Estimates are generally pretty similar, but standard errors are generally higher for the models fit separately (loss of efficiency).

Continuation ratio models

These are appropriate for data where there is a natural ordering to the response levels (ordinal outcomes). We will use the pregnant mice example from Agresti (page 290).

```
pregmice <- data.frame(concentration=c(0,62.5,125,250,500),
                       nonlive=c(15,17,22,38,144),
                       malformation=c(1,0,7,59,132),
                       normal=c(281,225,283,202,9))
pregmice

##   concentration nonlive malformation normal
## 1           0.0      15            1    281
## 2          62.5      17            0    225
## 3         125.0      22            7    283
## 4         250.0      38           59    202
## 5         500.0     144          132      9

# convert to long format
pregmice.long <- reshape(pregmice,
                         direction='long',
                         varying=2:4, # which columns vary over "time"
                         v.names='count', # the new variable we are creating
                         times=colnames(pregmice)[-1], # the names of the
"times"
```

```
                        timevar='outcome',
                        idvar='concentration')
rownames(pregmice.long) <- NULL
pregmice.long

##    concentration      outcome count
## 1            0.0      nonlive    15
## 2           62.5      nonlive    17
## 3          125.0      nonlive    22
## 4          250.0      nonlive    38
## 5          500.0      nonlive   144
## 6            0.0 malformation     1
## 7           62.5 malformation     0
## 8          125.0 malformation     7
## 9          250.0 malformation    59
## 10         500.0 malformation   132
## 11           0.0       normal   281
## 12          62.5       normal   225
## 13         125.0       normal   283
## 14         250.0       normal   202
## 15         500.0       normal     9
```

Because of how the likelihood for this model factorizes, we can fit separate ordinary logistic models and obtain the same results as a simultaneous fit (unlike the case for the multinomial model above). This data has three response categories, nonlive, malformed, and normal, so there will be two ordinary logistic models under the contination-ratio logits formulation.

```
# data for model comparing lowest level (nonlive) to higher two levels
(malformation and normal)
mice1 <- aggregate(count ~ concentration+I(outcome %in%
c('malformation','normal')),
                   FUN=sum,data=pregmice.long)
# data for conditional model comparing malformation and normal, given live
mice2 <- aggregate(count ~ concentration+I(outcome=='malformation'),
                   FUN=sum,data=subset(pregmice.long,outcome %in%
c('malformation','normal')))
colnames(mice1) <- c('conc','live','count')
colnames(mice2) <- c('conc','malformed','count')
mice1

##     conc  live count
## 1    0.0 FALSE    15
## 2   62.5 FALSE    17
## 3  125.0 FALSE    22
## 4  250.0 FALSE    38
## 5  500.0 FALSE   144
## 6    0.0  TRUE   282
## 7   62.5  TRUE   225
## 8  125.0  TRUE   290
```

```
## 9  250.0  TRUE   261
## 10 500.0  TRUE   141
```

```
mice2
```

```
##     conc malformed count
## 1    0.0     FALSE   281
## 2   62.5     FALSE   225
## 3  125.0     FALSE   283
## 4  250.0     FALSE   202
## 5  500.0     FALSE     9
## 6    0.0      TRUE     1
## 7   62.5      TRUE     0
## 8  125.0      TRUE     7
## 9  250.0      TRUE    59
## 10 500.0      TRUE   132
```

```
# modeling probability of nonlive birth
micemod1 <- glm(!live ~ conc,family=binomial,data=mice1,weights=count)
summary(micemod1)
```

```
##
## Call:
## glm(formula = !live ~ conc, family = binomial, data = mice1,
##     weights = count)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -13.71   -6.46    2.65   10.40   14.40
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.247934   0.157660   -20.6   <2e-16 ***
## conc         0.006389   0.000435    14.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1282.9  on 9  degrees of freedom
## Residual deviance: 1029.5  on 8  degrees of freedom
## AIC: 1034
##
## Number of Fisher Scoring iterations: 6
```

```
# modeling probability of malformation, given live birth
micemod2 <- glm(malformed ~ conc,family=binomial,data=mice2,weights=count)
summary(micemod2)
```

```
##
## Call:
```

```
## glm(formula = malformed ~ conc, family = binomial, data = mice2,
##     weights = count)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -9.615  -3.559  -0.684   3.550  13.685
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.70190    0.33224   -17.2   <2e-16 ***
## conc         0.01737    0.00123    14.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1077.76  on 8  degrees of freedom
## Residual deviance:  431.24  on 7  degrees of freedom
## AIC: 435.2
##
## Number of Fisher Scoring iterations: 5
```

The interpretation is that the negative outcome is significantly more likely with increasing concentration. For example, given a live birth, the odds that there was malformation multiplies by

```
exp(coef(micemod2)[2]*100)
```

```
##  conc
## 5.683
```

for every 100-unit increase in concentration.

What about goodness of fit? The way we have treated this data in arguments to `glm()` (using the `weights=` argument) means that R is interpreting this as ungrouped data.

```
mice1.wide <- reshape(mice1,direction='wide',timevar='live',idvar='conc')
mice1.wide$n <- apply(mice1.wide[,-1],1,sum)
mice2.wide <-
reshape(mice2,direction='wide',timevar='malformed',idvar='conc')
mice2.wide$n <- apply(mice2.wide[,-1],1,sum)

mice1.wide
```

```
##    conc count.FALSE count.TRUE   n
## 1   0.0          15        282 297
## 2  62.5          17        225 242
## 3 125.0          22        290 312
## 4 250.0          38        261 299
## 5 500.0         144        141 285
```

```
mice2.wide
```

```
##    conc count.FALSE count.TRUE   n
## 1   0.0         281          1 282
## 2  62.5         225          0 225
## 3 125.0         283          7 290
## 4 250.0         202         59 261
## 5 500.0           9        132 141
```

```
# grouped log-likelihoods for models of interest
llmod1 <- logLik(micemod1)+sum(lchoose(mice1.wide$n,mice1.wide$count.FALSE))
llmod2 <- logLik(micemod2)+sum(lchoose(mice2.wide$n,mice2.wide$count.TRUE))

# now we want to fit saturated models
# modeling probability of nonlive birth
micemod1.sat <- glm(!live ~
as.factor(conc),family=binomial,data=mice1,weights=count)
# modeling probability of malformation, given live birth
micemod2.sat <- glm(malformed ~
as.factor(conc),family=binomial,data=mice2,weights=count)

# grouped log-likelihoods for saturated models
llsat1 <-
logLik(micemod1.sat)+sum(lchoose(mice1.wide$n,mice1.wide$count.FALSE))
llsat2 <-
logLik(micemod2.sat)+sum(lchoose(mice2.wide$n,mice2.wide$count.TRUE))

# we can add the deviances from these two models
# deviance statistic for model 1 (probability of nonlive birth)
dev.mice <- 2*(llsat1-llmod1) +
# deviance statistic for model 2 (probability of malformation given live
birth)
  2*(llsat2-llmod2)
# test statistic
dev.mice
```

```
## 'log Lik.' 11.84 (df=5)
```

```
# p-value
pchisq(dev.mice,
       # degrees of freedom add between the two models as well
       micemod1$df.residual-micemod1.sat$df.residual +
         micemod2$df.residual-micemod2.sat$df.residual,
       lower.tail=FALSE)
```

```
## 'log Lik.' 0.06567 (df=5)
```

So there is “borderline” evidence of lack of fit here. However, as Agresti notes, this could be due to overdispersion in the data; even after adjusting for this, the linear relationship between concentration and log odds of a bad outcome is significant.

Adjacent-categories logits

These models can be expressed as baseline-category logit models (which we used for nominal responses above). We will use the job satisfaction example from Agresti (pages 287-288).

```
jobsat <- data.frame(gender=c(rep('Female',4),rep('Male',4)),
                     income.level=rep(1:4,2),
                     # not using original data because we want to have this
enter as a linear term in the model
                     #income=rep(c('<5K','5K-15K','15K-25K','>25K'),2),
                     very.dissat=c(1,2,0,0,1,0,0,0),
                     little.sat=c(3,3,1,2,1,3,0,1),
                     mod.sat=c(11,17,8,4,2,5,7,9),
                     very.sat=c(2,3,5,2,1,1,3,6))
jobsat$gender <- relevel(jobsat$gender,ref='Male')
```

The outcome variable, job satisfaction, is to be modeled as a function of gender and income level. This is clearly an ordinal variable, so we should be able to better by taking advantage of its natural ordering.

We will fit the model using the `vglm()` function in the `VGAM` package.

```
library(VGAM)

## Loading required package: stats4

## Loading required package: splines

# first look at a model with no covariates to be sure what probabilities we
are modeling
jobsat.mod0 <- vglm(cbind(very.sat, # order of categories matters here
                          mod.sat,
                          little.sat,
                          very.dissat) ~ 1,
                    data=jobsat,
                    # this tells R to use adjacent-categories logits
                    # and to force the effects to be the same for each logit
                    acat(parallel=TRUE))

summary(jobsat.mod0,presid=FALSE)

##
## Call:
## vglm(formula = cbind(very.sat, mod.sat, little.sat, very.dissat) ~
##     1, family = acat(parallel = TRUE), data = jobsat)
##
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept):1    1.008      0.244    4.14  3.5e-05 ***
```

```
## (Intercept):2   -1.504      0.295   -5.09  3.6e-07 ***
## (Intercept):3   -1.253      0.567   -2.21    0.027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  3
##
## Names of linear predictors:
## loge(P[Y=2]/P[Y=1]), loge(P[Y=3]/P[Y=2]), loge(P[Y=4]/P[Y=3])
##
## Residual deviance: 20.67 on 21 degrees of freedom
##
## Log-likelihood: -31.42 on 21 degrees of freedom
##
## Number of iterations: 4
##
## No Hauck-Donner effect found in any of the estimates
```

Where do the intercepts come from? These are the logits of certain probabilities, with these probabilities depending on how the particular software package treats the outcome. In this case, we have entered the outcome as a matrix with highest satisfaction column first (left), lowest coming last (right).

```
# we use rev() so that the columns are in the same order we gave to the
software to fit the model
prob.jobsat <- rev(prop.table(colSums(jobsat[,-c(1,2)])))
# conditional probabilities being modeled are based on the individual outcome
group probabilities
prob.jobsat
```

```
##    very.sat     mod.sat  little.sat very.dissat
##     0.22115     0.60577     0.13462     0.03846
```

```
# intercept labeled 1 above:
qlogis(prob.jobsat[2]/(prob.jobsat[1]+prob.jobsat[2]))
```

```
## mod.sat
##   1.008
```

```
# for each intercept at once:
qlogis(1-sapply(1:3,function(j) prob.jobsat[j]/sum(prob.jobsat[j:(j+1)])))
```

```
##   very.sat    mod.sat little.sat
##      1.008     -1.504     -1.253
```

This means that each intercept is the logit of the probability of being in the less satisfied of two adjacent categories, given you are in one of the two.

```
jobsat.mod1 <- vglm(cbind(very.sat, # order of categories matters here
                          mod.sat,
                          little.sat,
```

```
                        very.dissat) ~
                    # now predictors as usual
                    as.numeric(income.level)+gender,
                  data=jobsat,
                  # this tells R to use adjacent-categories logits
                  # and to force the effects to be the same for each logit
                  acat(parallel=TRUE))

summary(jobsat.mod1,presid=FALSE)

##
## Call:
## vglm(formula = cbind(very.sat, mod.sat, little.sat, very.dissat) ~
##     as.numeric(income.level) + gender, family = acat(parallel = TRUE),
##     data = jobsat)
##
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept):1             2.0259     0.5758    3.52  0.00043 ***
## (Intercept):2            -0.6550     0.5253   -1.25  0.21240
## (Intercept):3            -0.5507     0.6795   -0.81  0.41768
## as.numeric(income.level)  -0.3888     0.1547   -2.51  0.01195 *
## genderFemale               0.0447     0.3144    0.14  0.88697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  3
##
## Names of linear predictors:
## loge(P[Y=2]/P[Y=1]), loge(P[Y=3]/P[Y=2]), loge(P[Y=4]/P[Y=3])
##
## Residual deviance: 12.55 on 19 degrees of freedom
##
## Log-likelihood: -27.36 on 19 degrees of freedom
##
## Number of iterations: 4
##
## No Hauck-Donner effect found in any of the estimates
```

The interpretation of this model is that since the coefficient estimate for income level is negative, the odds of lower job satisfaction decrease as income increases. Specifically, controlling for gender, the odds of a satisfaction level in the lower of two adjacent categories is multiplied by

```
exp(coef(jobsat.mod1)[4])

## as.numeric(income.level)
##                   0.6779
```

for each category increase in income.

In terms of lack of fit, this is grouped data, so we can use the residual deviance to calculate the p-value for the goodness of fit chi-square test.

```
pchisq(deviance(jobsat.mod1),
       attr(jobsat.mod1,'df.residual'),
       lower.tail=FALSE)
```

```
## [1] 0.8608
```

This shows that there is no evidence of lack of fit for this model including gender and income level as a linear term.

Cumulative logit model

We can look at this same example with a cumulative logistic model fit.

```
jobsat.mod2 <- vglm(cbind(very.sat, # order of categories matters here
                          mod.sat,
                          little.sat,
                          very.dissat) ~
                      # now predictors as usual
                      as.numeric(income.level)+gender,
                    data=jobsat,
                    # this tells R to use cumulative logits
                    cumulative(parallel=TRUE,reverse=TRUE))

summary(jobsat.mod2,presid=FALSE)
```

```
##
## Call:
## vglm(formula = cbind(very.sat, mod.sat, little.sat, very.dissat) ~
##     as.numeric(income.level) + gender, family = cumulative(parallel =
TRUE,
##     reverse = TRUE), data = jobsat)
##
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept):1              2.5950     0.7147    3.63  0.00028 ***
## (Intercept):2             -0.3945     0.6491   -0.61  0.54329
## (Intercept):3             -2.0868     0.7729   -2.70  0.00694 **
## as.numeric(income.level)  -0.5105     0.2022   -2.52  0.01160 *
## genderFemale               0.0176     0.4251    0.04  0.96698
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  3
##
## Names of linear predictors:
```

```
## logit(P[Y>=2]), logit(P[Y>=3]), logit(P[Y>=4])
##
## Residual deviance: 13.31 on 19 degrees of freedom
##
## Log-likelihood: -27.73 on 19 degrees of freedom
##
## Number of iterations: 5
##
## No Hauck-Donner effect found in any of the estimates
##
## Exponentiated coefficients:
## as.numeric(income.level)             genderFemale
##                   0.6002                   1.0178
```

This model fit gives similar results to the adjacent-categories logits model:

```
# odds ratio for income level for adjacent-categories logits
exp(coef(jobsat.mod1)[4])

## as.numeric(income.level)
##                   0.6779

# odds ratio for income level for cumulative logit
exp(coef(jobsat.mod2)[4])

## as.numeric(income.level)
##                   0.6002
```

We have been using the restriction `parallel=TRUE` in these models to require that the covariate effects be the same between logits. This corresponds to the proportional odds model, where the effect of covariates is constant across logits

This restriction is not necessary, however, so let's look at what happens when we relax it.

```
jobsat.mod3 <- vglm(cbind(very.sat, # order of categories matters here
                          mod.sat,
                          little.sat,
                          very.dissat) ~
                      # now predictors as usual
                      as.numeric(income.level)+gender,
                    data=jobsat,
                    # this tells R to use cumulative logits
                    cumulative(parallel=FALSE,reverse=TRUE))

summary(jobsat.mod3,presid=FALSE)

##
## Call:
## vglm(formula = cbind(very.sat, mod.sat, little.sat, very.dissat) ~
##     as.numeric(income.level) + gender, family = cumulative(parallel =
FALSE,
```

```
##     reverse = TRUE), data = jobsat)
##
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept):1                2.5108     0.8697    2.89   0.0039 **
## (Intercept):2               -0.3597     0.8295   -0.43   0.6645
## (Intercept):3               -1.2060     1.6434   -0.73   0.4631
## as.numeric(income.level):1  -0.4952     0.2481   -2.00   0.0460 *
## as.numeric(income.level):2  -0.4971     0.2751   -1.81   0.0708 .
## as.numeric(income.level):3  -1.1377     0.6981      NA       NA
## genderFemale:1               0.0931     0.5168    0.18   0.8571
## genderFemale:2              -0.0892     0.5783   -0.15   0.8774
## genderFemale:3               0.2930     1.2385    0.24   0.8130
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  3
##
## Names of linear predictors:
## logit(P[Y>=2]), logit(P[Y>=3]), logit(P[Y>=4])
##
## Residual deviance: 11.74 on 15 degrees of freedom
##
## Log-likelihood: -26.95 on 15 degrees of freedom
##
## Number of iterations: 6
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'as.numeric(income.level):3'
##
## Exponentiated coefficients:
## as.numeric(income.level):1 as.numeric(income.level):2
##                     0.6095                     0.6083
## as.numeric(income.level):3             genderFemale:1
##                     0.3206                     1.0976
##             genderFemale:2             genderFemale:3
##                     0.9146                     1.3405
```

This allows different slopes for the different logits, so has more estimated parameters (and therefore degrees of freedom) than the model assuming proportional odds. These models are nested, so we may compare them with the likelihood ratio test.

```
options(digits=7)
anova(jobsat.mod2,jobsat.mod3,type='1')

## Analysis of Deviance Table
##
## Model 1: cbind(very.sat, mod.sat, little.sat, very.dissat) ~
as.numeric(income.level) +
```

```
##     gender
## Model 2: cbind(very.sat, mod.sat, little.sat, very.dissat) ~
as.numeric(income.level) +
##     gender
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        19     13.309
## 2        15     11.742  4    1.567   0.8147
```

This test results in a non-significant p-value, so we can conclude that for this data set, the proportional odds assumption is reasonable.

Let's look at the cheese-tasting example from the notes now.

```
cheese <- data.frame(A=c(0,0,1,7,8,8,19,8,1),
                     B=c(6,9,12,11,7,6,1,0,0),
                     C=c(1,1,6,8,23,7,5,1,0),
                     D=c(0,0,0,1,3,7,14,16,11))

cheese2 <- as.data.frame(t(cheese))
colnames(cheese2) <- paste('response.cat',1:ncol(cheese2),sep='')
cheese2$cheesetype <- as.factor(c('A','B','C','D'))
cheese2
```

```
##   response.cat1 response.cat2 response.cat3 response.cat4 response.cat5
## A             0             0             1             7             8
## B             6             9            12            11             7
## C             1             1             6             8            23
## D             0             0             0             1             3
##   response.cat6 response.cat7 response.cat8 response.cat9 cheesetype
## A             8            19             8             1          A
## B             6             1             0             0          B
## C             7             5             1             0          C
## D             7            14            16            11          D
```

```
options(digits=4)
# have to adjust the data format first
long.cheese <- reshape(cheese2,direction='long',varying=1:9)
long.cheese2 <-
long.cheese[rep(1:nrow(long.cheese),long.cheese$response),c('cheesetype','tim
e')]
colnames(long.cheese2)[2] <- 'response'
cheese.mod0 <- vglm(response ~ 1,
                    data=long.cheese2,
                    cumulative(parallel=TRUE,reverse=FALSE))
# see what we are modeling in terms of probabilities
plogis(coef(cheese.mod0))
```

```
## (Intercept):1 (Intercept):2 (Intercept):3 (Intercept):4 (Intercept):5
##       0.03365       0.08173       0.17308       0.30288       0.50000
## (Intercept):6 (Intercept):7 (Intercept):8
##       0.63462       0.82212       0.94231
```

```r
cumsum(apply(cheese,1,sum))/sum(cheese)
```

```
## [1] 0.03365 0.08173 0.17308 0.30288 0.50000 0.63462 0.82212 0.94231
1.00000
```

```r
# with reverse=FALSE, we are modeling the cumulative probability of being
less than or equal to the indicated level of the response variable
# now fit the model with proportional odds assumption
cheese.mod1 <- vglm(response ~ cheesetype,
                    data=long.cheese2,
                    cumulative(parallel=TRUE,reverse=TRUE))
cheese.mod2 <- vglm(response ~ cheesetype,
                    data=long.cheese2,
                    cumulative(parallel=TRUE,reverse=FALSE))

# with reverse=TRUE (same as results in notes)
summary(cheese.mod1,presid=FALSE)
```

```
##
## Call:
## vglm(formula = response ~ cheesetype, family = cumulative(parallel = TRUE,
##     reverse = TRUE), data = long.cheese2)
##
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept):1   5.4674     0.5202   10.51  < 2e-16 ***
## (Intercept):2   4.4122     0.4247   10.39  < 2e-16 ***
## (Intercept):3   3.3126     0.3697    8.96  < 2e-16 ***
## (Intercept):4   2.2440     0.3262    6.88  6.0e-12 ***
## (Intercept):5   0.9078     0.2748    3.30  0.00095 ***
## (Intercept):6  -0.0443     0.2598   -0.17  0.86476
## (Intercept):7  -1.5459     0.3042   -5.08  3.7e-07 ***
## (Intercept):8  -3.1058     0.4044   -7.68  1.6e-14 ***
## cheesetypeB    -3.3518     0.4235   -7.91  2.5e-15 ***
## cheesetypeC    -1.7099     0.3731   -4.58  4.6e-06 ***
## cheesetypeD     1.6128     0.3778    4.27  2.0e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  8
##
## Residual deviance: 711.3 on 1653 degrees of freedom
##
## Log-likelihood: -355.7 on 1653 degrees of freedom
##
## Number of iterations: 6
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## '(Intercept):1'
```

```
##
## Exponentiated coefficients:
## cheesetypeB cheesetypeC cheesetypeD
##     0.03502     0.18089     5.01681
```

```
# with reverse=FALSE
summary(cheese.mod2,presid=FALSE)
```

```
##
## Call:
## vglm(formula = response ~ cheesetype, family = cumulative(parallel = TRUE,
##     reverse = FALSE), data = long.cheese2)
##
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  -5.4674     0.5202  -10.51  < 2e-16 ***
## (Intercept):2  -4.4122     0.4247  -10.39  < 2e-16 ***
## (Intercept):3  -3.3126     0.3697   -8.96  < 2e-16 ***
## (Intercept):4  -2.2440     0.3262   -6.88  6.0e-12 ***
## (Intercept):5  -0.9078     0.2748   -3.30  0.00095 ***
## (Intercept):6   0.0443     0.2598    0.17  0.86476
## (Intercept):7   1.5459     0.3042    5.08  3.7e-07 ***
## (Intercept):8   3.1058     0.4044    7.68  1.6e-14 ***
## cheesetypeB     3.3518     0.4235    7.91  2.5e-15 ***
## cheesetypeC     1.7099     0.3731    4.58  4.6e-06 ***
## cheesetypeD    -1.6128     0.3778   -4.27  2.0e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  8
##
## Residual deviance: 711.3 on 1653 degrees of freedom
##
## Log-likelihood: -355.7 on 1653 degrees of freedom
##
## Number of iterations: 6
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## '(Intercept):1'
##
## Exponentiated coefficients:
## cheesetypeB cheesetypeC cheesetypeD
##     28.5552      5.5283      0.1993
```

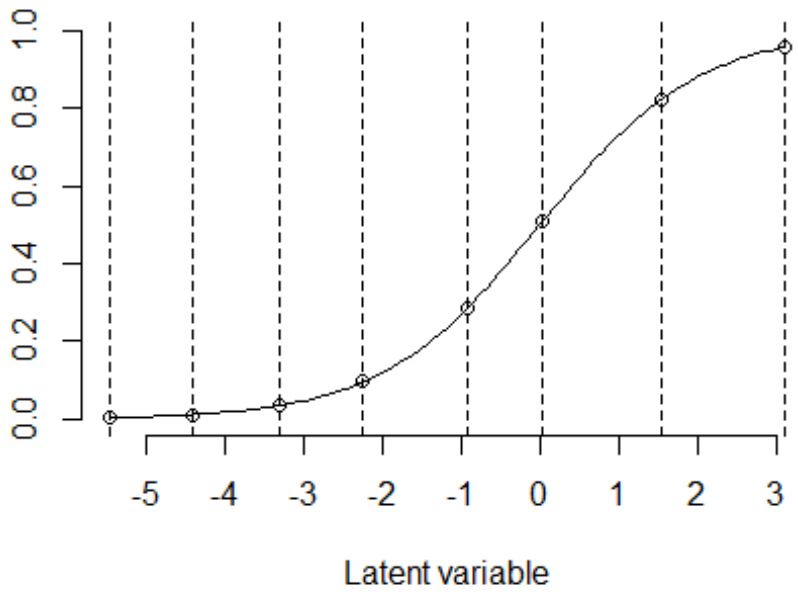
These estimates tell us that cheese B and cheese C are not thought by these study participants to taste as good as cheese A, but that cheese D is thought to be better than cheese A. All of these comparisons are highly significant.

We can look at the latent variable specification of the model graphically.

```r
plogis(coef(cheese.mod2)[1:8])

## (Intercept):1 (Intercept):2 (Intercept):3 (Intercept):4 (Intercept):5
##      0.004205      0.011983      0.035141      0.095868      0.287459
## (Intercept):6 (Intercept):7 (Intercept):8
##      0.511062      0.824323      0.957130

plot(coef(cheese.mod2)[1:8],rep(0,8),axes=FALSE,
     type='n',
     ylim=c(0,1),
     xlab='Latent variable',
     ylab='')
curve(plogis(x),add=TRUE)
axis(1,at=seq(-5,3,by=1))
axis(2)
abline(v=coef(cheese.mod2)[1:8],lty=2)
points(coef(cheese.mod2)[1:8],plogis(coef(cheese.mod2)[1:8]))
```



The dashed lines are the cutpoints for the latent variable under the ordinal regression model.