Week 10 - Count Outcomes

Fred Traylor, Lab TA

4/4/2022

Data Loading and Management

Let's use the GSS again. This week, we'll be looking at three specific variables related to family life: The number of sexual partners, siblings, and children as person has. Since all three are count variables, we'll be using new methods to analyze them.

```
gss2018 <- haven::read_dta("lab-data/GSS2018.dta")

gss <- gss2018 %>%
  dplyr::select(

    # Targets: Children, Partners, and Siblings
    partners, # https://gssdataexplorer.norc.org/variables/5049/vshow
    sibs,     # https://gssdataexplorer.norc.org/variables/51/vshow
    childs,   # https://gssdataexplorer.norc.org/variables/52/vshow

    # Demographics
    marital,
    sex, age, educ,
    wtss
    ) %>%
  haven::zap_labels() %>%
  mutate(

    # New Variables
    marital = factor(marital,
                     levels = c(5, 1:4),
                     labels = c("Never Married", "Married",
                                "Widowed", "Divorced", "Separated")
                     ),

    # Variables we've used before
    age = case_when(
      age > 88 ~ NaN,
      TRUE ~ age
      ),
    sex = factor(sex,
                 levels = c(1,2),
                 labels = c("Male", "Female")
                 ),
    weight = wtss
    ) %>%
```

```
  dplyr::select(-wtss) %>%
  drop_na()
```

If you get an error...

It was probably: `Error in select(...) : unused argument (...)`. One of the packages we're using today, `MASS` has a `select()` function that will cover up `dplyr::select()`. For more info, see this page.

If this happens, restart R (CTRL + Shift + F10 on PC) and run the `dplyr` code before running `library(MASS)`.

Also, when this happens, use `package::function()` format, as I have done above (`dplyr::select()`) and below (`MASS::glm.nb`). Because of this, I still loaded `dplyr`, which is invaluable to me throughout the script, and did not load MASS, since I only need it in a few choice locations.

Descriptives

As always, let's look at our descriptive statistics.

```
datasummary_skim(gss,
                 type = "numeric",
                 fmt = 2, # Show 2 decimal places
                 histogram = F,
                 title = "Sample Descriptive Statistics: Continuous Variables",
                 notes = "Data: 2018 General Social Survey",
                 output = "flextable")
```

```
## Warning: Warning: fonts used in `flextable` are ignored because the `pdflatex`
## engine is used and not `xelatex` or `lualatex`. You can avoid this warning
## by using the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a
## compatible engine by defining `latex_engine: xelatex` in the YAML header of the
## R Markdown document.
```

Table 1: Sample Descriptive Statistics: Continuous Variables

	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max
partners	9	0	1.00	1.03	0.00	1.00	9.00
sibs	17	0	3.36	2.58	0.00	3.00	17.00
childs	9	0	1.73	1.59	0.00	2.00	8.00
age	71	0	47.32	17.43	18.00	46.00	88.00
educ	21	0	13.92	2.85	0.00	14.00	20.00
weight	14	0	1.03	0.64	0.47	0.94	5.90

Data: 2018 General Social Survey

```
datasummary_skim(gss,
                 type = "categorical",
                 title = "Sample Descriptive Statistics: Categorical Variables",
                 notes = "Data: 2018 General Social Survey",
                 output = "flextable")
```

```
## Warning: Warning: fonts used in `flextable` are ignored because the `pdflatex`
## engine is used and not `xelatex` or `lualatex`. You can avoid this warning
## by using the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a
## compatible engine by defining `latex_engine: xelatex` in the YAML header of the
## R Markdown document.
```

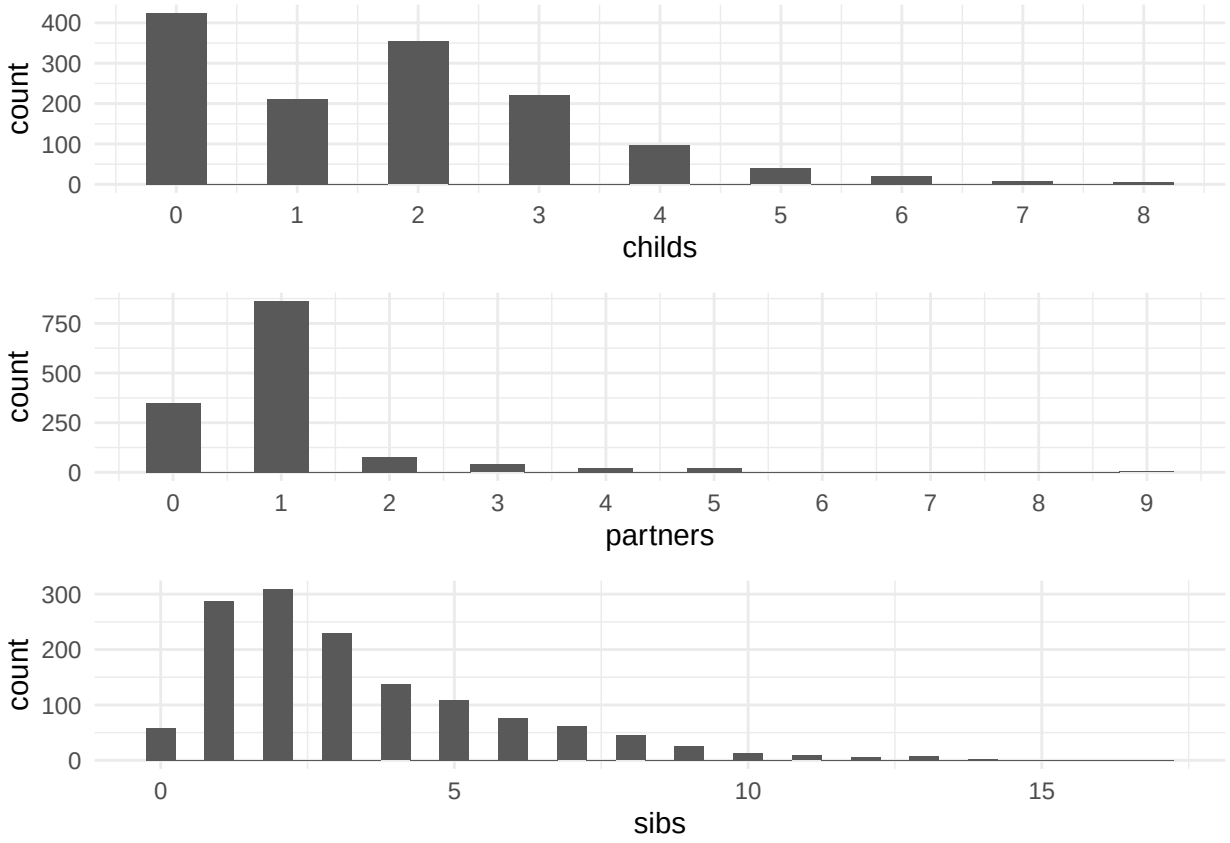
Table 2: Sample Descriptive Statistics: Categorical Variables

		N	%
marital	Never Married	418	30.3
	Married	593	43.0
	Widowed	94	6.8
	Divorced	231	16.7
	Separated	44	3.2
sex	Male	630	45.7
	Female	750	54.3

Data: 2018 General Social Survey

Let's look at our three variables.

```
cowplot::plot_grid(
  ggplot(gss, aes(childs)) + theme_minimal() +
    geom_histogram(binwidth = .5) + scale_x_continuous(breaks = seq(0,10)),
  ggplot(gss, aes(partners)) + theme_minimal() +
    geom_histogram(binwidth = .5) + scale_x_continuous(breaks = seq(0,10)),
  ggplot(gss, aes(sibs)) + theme_minimal() +
    geom_histogram(binwidth = .5),
  nrow = 3
)
```



OLS Methods

Let's start by creating three models of our target data using OLS methods. We'll come back to these later, but for now, just note that using OLS is an option, but not the most correct option.

```
partner_ols <- lm(partners ~ marital + age + educ + sex,
                  data = gss, weights = weight)
sibs_ols <- lm(sibs ~ marital + age + educ + sex,
               data = gss, weights = weight)
child_ols <- lm(childs ~ marital + age + educ + sex,
                data = gss, weights = weight)
stargazer(partner_ols, sibs_ols, child_ols,
          single.row = T,
          type = "text")
```

```
##
## ================================================================================
##                                          Dependent variable:
##                                 ---------------------------------------------
##                                    partners           sibs            childs
##                                      (1)              (2)              (3)
## --------------------------------------------------------------------------------
## maritalMarried                   -0.052 (0.069)    -0.014 (0.175)   1.167*** (0.096)
## maritalWidowed                 -0.400*** (0.150)    0.138 (0.380)   0.884*** (0.207)
## maritalDivorced                  -0.047 (0.095)     0.077 (0.241)   0.959*** (0.131)
## maritalSeparated                -0.373** (0.177)   1.092** (0.448)  1.388*** (0.244)
```

```
## age                                -0.014*** (0.002) 0.015*** (0.005)  0.028*** (0.003)
## educ                                0.019** (0.009)  -0.215*** (0.023) -0.093*** (0.012)
## sexFemale                          -0.264*** (0.052)  0.294** (0.132)    0.088 (0.072)
## Constant                           1.619*** (0.145)  5.367*** (0.367)  0.937*** (0.200)
## ------------------------------------------------------------------------------------------
## Observations                             1,380             1,380             1,380
## R2                                       0.110             0.082             0.318
## Adjusted R2                              0.105             0.077             0.314
## Residual Std. Error (df = 1372)          0.965             2.449             1.334
## F Statistic (df = 7; 1372)             24.118***         17.404***         91.227***
## ==========================================================================================
## Note:                                                         *p<0.1; **p<0.05; ***p<0.01
```

We see, generally, that widows, women, those who are separated, and those who are older tend to have fewer sexual partners, while those with more education tend to have more.

Also, those who are separated tend to have more siblings, for some reason, as do women and those who are older. Those with more education tend to have fewer.

Finally, we see that that people who are (or were) in some sort of marital relationship have more children, which is to be expected. Those who are older also have more children and those who have more education have fewer.

Poisson Regression: Number of Partners

For something like count data, we need to use a Poisson model. This models counts of discrete things. In this example, we'll model the number of sexual partners a person has.

It takes the same form as our previous GLM's, but setting the family argument to "poisson" instead of "binomial."

For help on creating the model in R, I suggest UCLA's OARC page: https://stats.oarc.ucla.edu/r/dae/poisson-regression/ Their page on interpreting the models is great as well, though the output is in STATA: https://stats.oarc.ucla.edu/stata/output/poisson-regression/

```
partner_poi <- glm(partners ~ marital + age + educ + sex,
                   data = gss, weights = weight, family = "poisson")

summary(partner_poi)
```

```
##
## Call:
## glm(formula = partners ~ marital + age + educ + sex, family = "poisson",
##     data = gss, weights = weight)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4472  -0.4893  -0.0795   0.2201   6.2427
##
## Coefficients:
##                   Estimate Std. Error z value          Pr(>|z|)
## (Intercept)       0.547332   0.148837   3.677          0.000236 ***
## maritalMarried   -0.008957   0.068161  -0.131          0.895452
## maritalWidowed   -0.939889   0.259875  -3.617          0.000298 ***
## maritalDivorced  -0.004566   0.099695  -0.046          0.963470
## maritalSeparated -0.464621   0.237254  -1.958          0.050191 .
## age              -0.015064   0.002084  -7.228 0.000000000000491 ***
```

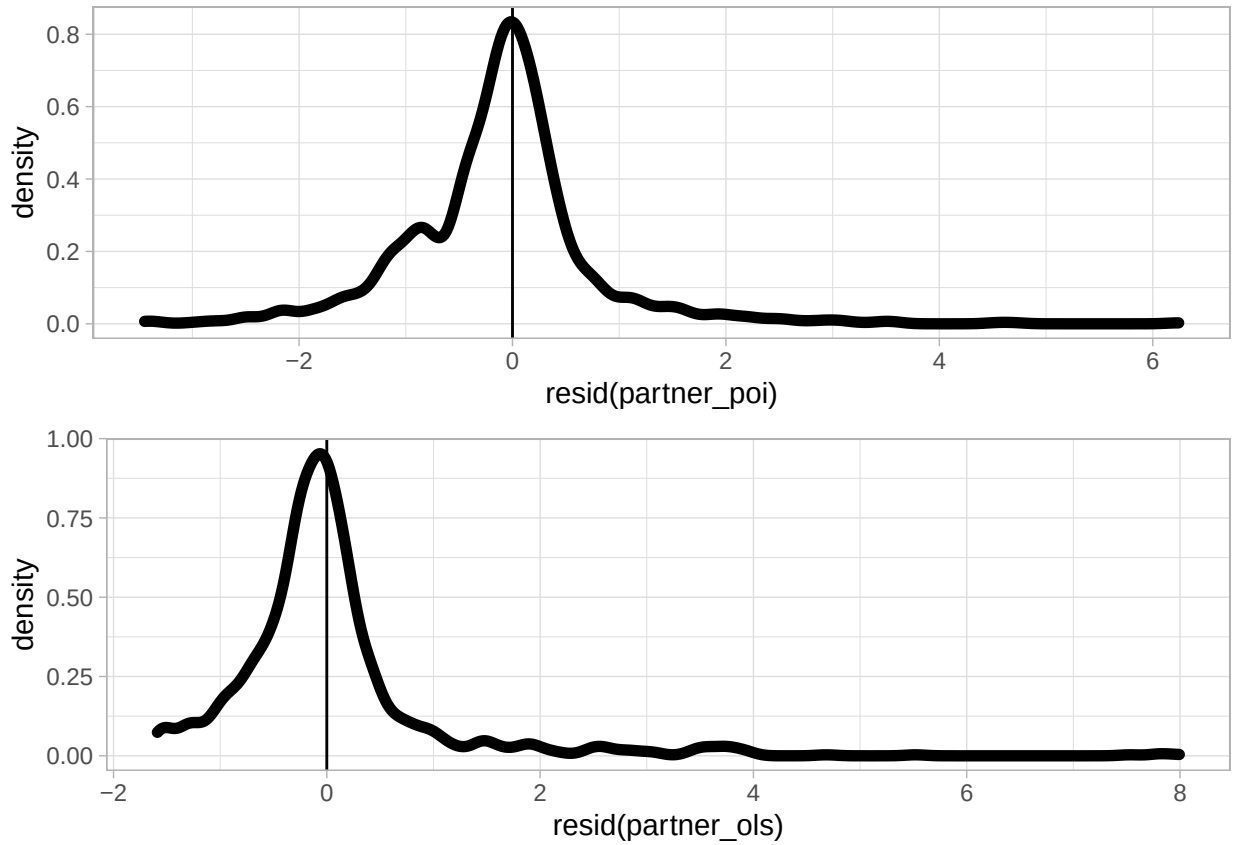
```
## educ              0.020175   0.009570   2.108          0.035014 *
## sexFemale        -0.253158   0.053026  -4.774 0.000001804251398 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1202.9  on 1379  degrees of freedom
## Residual deviance: 1032.9  on 1372  degrees of freedom
## AIC: 3416.6
##
## Number of Fisher Scoring iterations: 5
```

Model Output Interpretation

Let's break down what we see above: 1. The formula call

2. Deviance Residuals: We're looking for residuals that are not very skewed. In this case, the median is close to zero, which is good, and the 1st and 3rd quartiles are even enough. Looking at a density plot of the residuals below, we see they're slightly skewed with a tail to the right. Compared to the OLS version, though, it is much more centered.

```
cowplot::plot_grid(
  ggplot(gss, aes(x =  resid(partner_poi))) +
    geom_density(size = 2) + theme_light() + geom_vline(aes(xintercept = 0)),
  ggplot(gss, aes( x = resid(partner_ols))) +
    geom_density(size = 2) + theme_light() + geom_vline(aes(xintercept = 0)),
  nrow = 2
)
```



3. Coefficients: We interpret these (kind of) like normal. A one-year increase in age, for example, is associated with a .015 decrease in the **expected log count** of children.

4. Deviance information: We can use this to perform goodness-of-fit tests for the model. Below, I run a chi-squared test, with the null hypothesis that the deviance is acceptable for the number of data points and predictors. We fail to reject the null, so the model is adequately fit.

```
with(partner_poi, cbind(res.deviance = deviance,
                        df = df.residual,
                        p = pchisq(deviance, df.residual, lower.tail=FALSE)))
```

```
##      res.deviance   df p
## [1,]     1032.893 1372 1
```

Model Interpretation

Let's create a quick output of our coefficients.

```
stargazer(partner_poi, partner_ols, single.row = T, type = "text")
```

```
##
## ===============================================================
##                                 Dependent variable:
##                     ------------------------------------------
##                                       partners
##                           Poisson                 OLS
##                             (1)                   (2)
## ---------------------------------------------------------------
```

```
## maritalMarried       -0.009 (0.068)        -0.052 (0.069)
## maritalWidowed      -0.940*** (0.260)    -0.400*** (0.150)
## maritalDivorced      -0.005 (0.100)        -0.047 (0.095)
## maritalSeparated     -0.465* (0.237)      -0.373** (0.177)
## age                 -0.015*** (0.002)    -0.014*** (0.002)
## educ                 0.020** (0.010)      0.019** (0.009)
## sexFemale           -0.253*** (0.053)    -0.264*** (0.052)
## Constant            0.547*** (0.149)      1.619*** (0.145)
## ---------------------------------------------------------------
## Observations              1,380                 1,380
## R2                                              0.110
## Adjusted R2                                     0.105
## Log Likelihood         -1,700.293
## Akaike Inf. Crit.       3,416.586
## Residual Std. Error                     0.965 (df = 1372)
## F Statistic                           24.118*** (df = 7; 1372)
## ===============================================================
## Note:                                *p<0.1; **p<0.05; ***p<0.01
```

In interpreting Poisson models,

$$\beta = log(\mu_{x+1}) - log(\mu_x)$$

where going from x to x+1 is the change in x. For this reason, we say there is a "difference in the log of expected counts" as we move from x to x+1.

So here we can say that each additional year of age decreases the number of sexual partners a person had in the past year. The difference in the log of expected counts is expected to decrease by .015 for each year of age.

The line goes similarly for contrasts in categorical variables. Somebody who is widowed is expected to have a decreases in the log of the expected counts by .94, and women are expected to have a decrease in the log of the expected counts by .25.

Incidence Rate Ratios

For those of us who prefer odds ratios, we can create incidence rate ratios, which act similarly.

When subtracting logs, we can do some algebra to change the form:

$$\beta = log(\mu_{x+1}) - log(\mu_x) = log(\frac{\mu_{x+1}}{\mu_x})$$

.

In other words, we're examining the ratios of the rate of increasing sexual partners by moving from x to x+1. When we exponentiate this, we go from

$$\beta = log(\frac{\mu_{x+1}}{\mu_x}) - - > log(\beta) = \frac{\mu_{x+1}}{\mu_x}$$

As such, we can now talk about changes in the "rate ratio" as we move from x to x+1.

```
cbind(Estimate = coef(partner_poi),
      IRR = exp(coef(partner_poi)),
      exp(confint(partner_poi)))
```

```
## Waiting for profiling to be done...
```

```
##                      Estimate       IRR     2.5 %    97.5 %
## (Intercept)       0.547331977 1.7286348 1.2892067 2.3104404
## maritalMarried   -0.008956972 0.9910830 0.8671681 1.1328010
```

```
## maritalWidowed   -0.939889400 0.3906710 0.2264130 0.6309708
## maritalDivorced  -0.004566048 0.9954444 0.8170699 1.2079778
## maritalSeparated -0.464621360 0.6283730 0.3815330 0.9720364
## age              -0.015063992 0.9850489 0.9810081 0.9890562
## educ              0.020174668 1.0203796 1.0014862 1.0397637
## sexFemale        -0.253158207 0.7763450 0.6996174 0.8612956
```

So here we see that each year of education increases the rate ratio of sexual partners by a factor of 1.02, or a 2% increase, and each year of age decreases it by a factor of .015, or a 1.5% decrease.

When we talk about categorical contrasts, we don't have to speak of ratios, because the change from x to x+1 only occurs once. So females are expected to have a rate of sexual partners that is 23% less than males. We could also say that it changes by a factor of .77. Additionally, widows are expected to have a rate of sexual partners that is 61% lower $(1 - .39)$ than never-married individuals.

Predictions

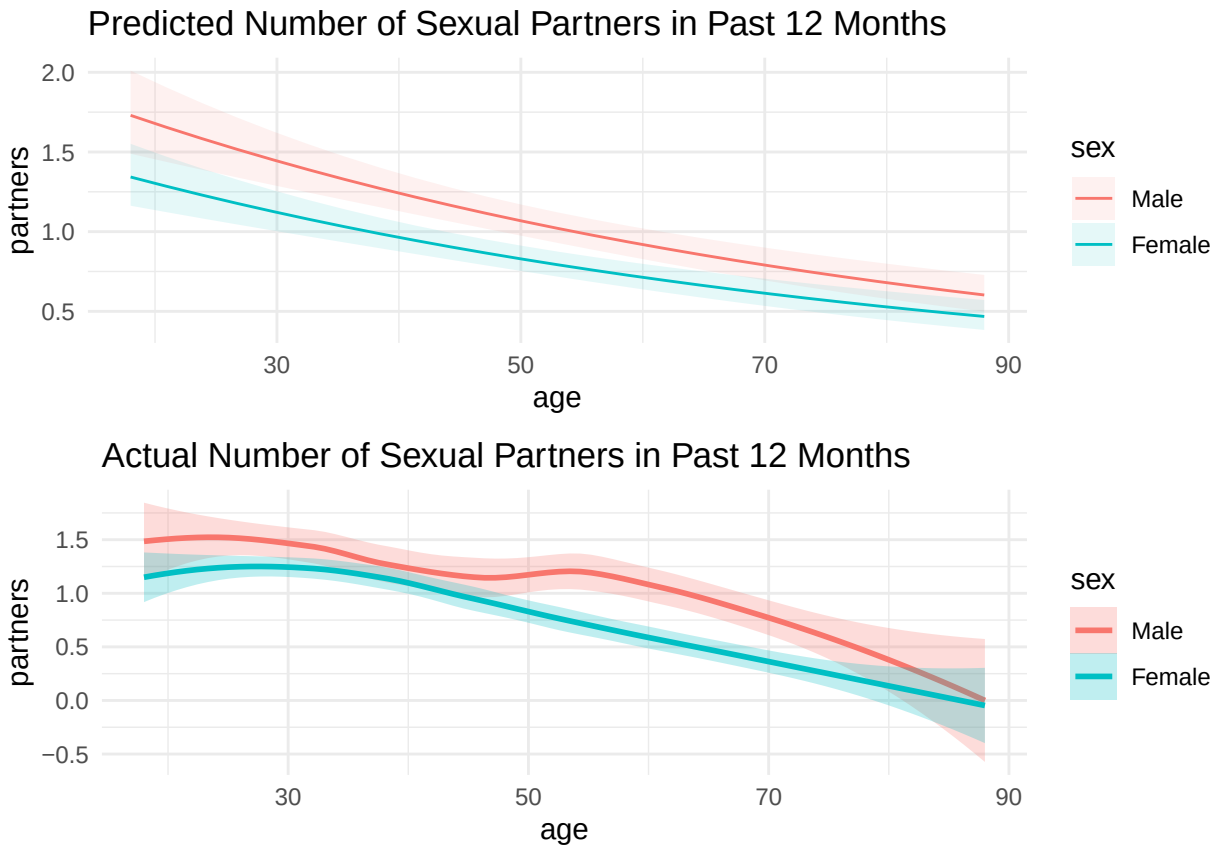
We can use our functions from the `marginaleffects` package to create and plot predicted values.

```
predictions(partner_poi, newdata = datagrid(marital = levels(gss$marital),
                                            educ = c(12, 16)))
```

```
##    rowid     type predicted  std.error  conf.low conf.high      age    sex
## 1      3 response 0.8381645 0.05804734 0.7317775 0.9600183 47.31884 Female
## 2      4 response 0.8306906 0.04461757 0.7476874 0.9229083 47.31884 Female
## 3      5 response 0.3274466 0.08076856 0.2019211 0.5310060 47.31884 Female
## 4      6 response 0.8343462 0.06858297 0.7101953 0.9802001 47.31884 Female
## 5      7 response 0.5266800 0.12110367 0.3356010 0.8265522 47.31884 Female
## 6      8 response 0.9086074 0.06292954 0.7932729 1.0407104 47.31884 Female
## 7      9 response 0.9005054 0.04438428 0.8175835 0.9918374 47.31884 Female
## 8     10 response 0.3549666 0.08804451 0.2183026 0.5771865 47.31884 Female
## 9     11 response 0.9044681 0.07525925 0.7683626 1.0646829 47.31884 Female
## 10    12 response 0.5709443 0.13193194 0.3629952 0.8980213 47.31884 Female
##       weight       marital educ
## 1  1.027619 Never Married   12
## 2  1.027619       Married   12
## 3  1.027619       Widowed   12
## 4  1.027619      Divorced   12
## 5  1.027619     Separated   12
## 6  1.027619 Never Married   16
## 7  1.027619       Married   16
## 8  1.027619       Widowed   16
## 9  1.027619      Divorced   16
## 10 1.027619     Separated   16
```

```
cowplot::plot_grid(
  plot_cap(partner_poi, condition = c("age", "sex")) +
    ggtitle("Predicted Number of Sexual Partners in Past 12 Months"),
  ggplot() + theme_minimal() + ggtitle("Actual Number of Sexual Partners in Past 12 Months") +
    geom_smooth(gss, mapping = aes(x = age, y = partners, color = sex, fill = sex), alpha = .25),
  nrow = 2
)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



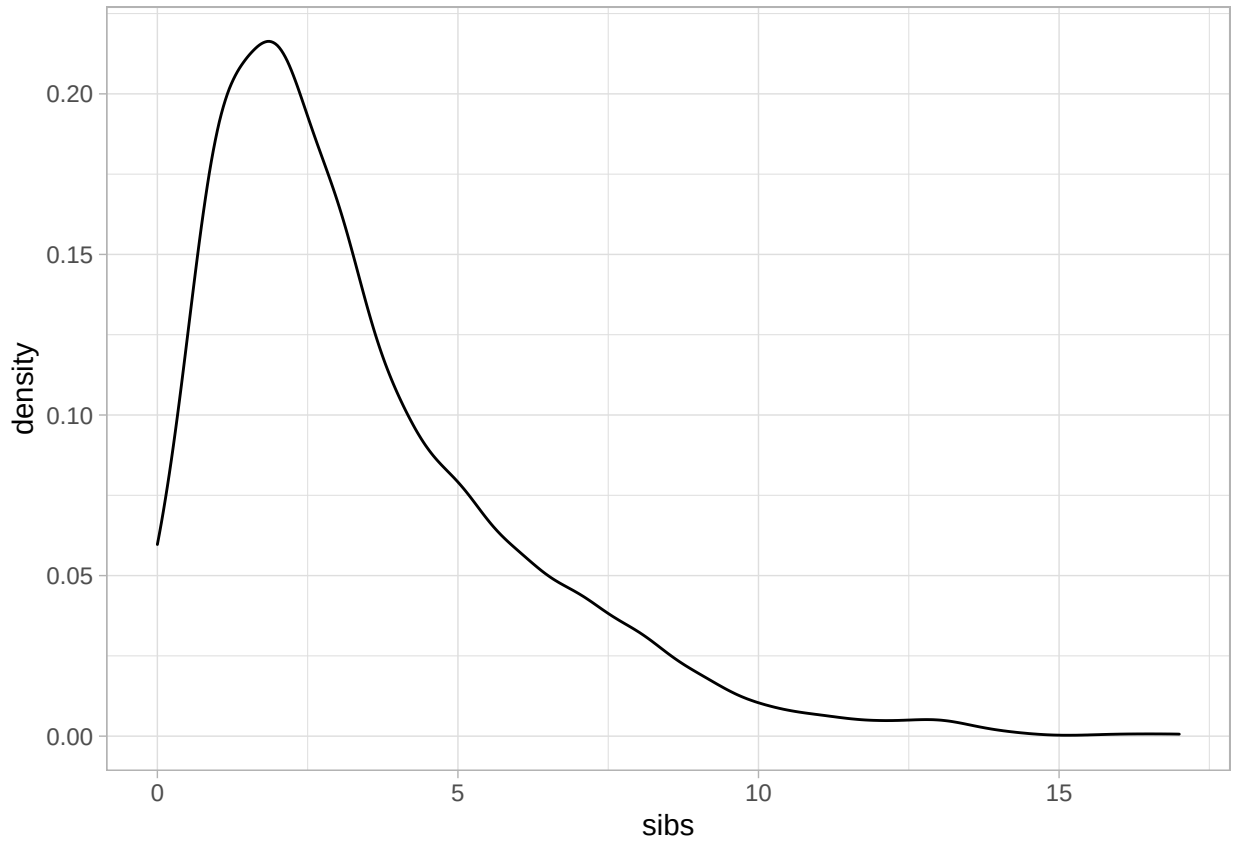
Above, we see our model did a pretty good job at predicting partners.

Negative Binomial: Number of Siblings

Overdispersion

Overdispersion occurs when our data display higher variance than expected. For example, while most people only have a few siblings, there are several who have more than ten.

```
ggplot(gss, aes(x = sibs)) + geom_density() + theme_light()
```




```
table(gss$sibs)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  16  17
##  58 288 309 230 137 109  77  61  45  26  13   9   6   8   2   1   1
```

As a rule of thumb, the residual deviance in a poisson model should be about equal to the degrees of freedom of the residuals in the model. (See top of page 4 here.

TOM: What are your thoughts on this? TD: Not something I've considered before but it sounds reasonable. I won't be discussing this in lecture.

Below, I create a Poisson model estimating the number of siblings a person has, then divide the null deviance by the DF's of the residuals.

```
sibs_poisson <- glm(sibs ~ marital + age + educ + sex,
                    data = gss, weights = weight, family = "poisson")
with(sibs_poisson, cbind("Deviance" = deviance,
                         "DF Resid" = df.residual,
                         "Dev / DF" = deviance / df.residual))
```

```
##      Deviance DF Resid Dev / DF
## [1,] 2331.426     1372  1.69929
```

For comparison, our model estimating sexual partners had a deviance:DF ratio of 0.75.

```
with(partner_poi, cbind("Deviance" = deviance,
                         "DF Resid" = df.residual,
                         "Dev / DF" = deviance / df.residual))
```

```
##      Deviance DF Resid  Dev / DF
## [1,] 1032.893     1372 0.7528374
```

Because our siblings variable is overdispersed in our currently-specified poisson model, we will use a negative binomial model.

The Negative Binomial Model

The negative binomial model includes a Theta parameter (θ) that accounts for the overdispersion among our dependent variable.

To make one, we use the package MASS. The MASS package (Modern Applied Statistics with S) was one of the original packages for R and has a ton of statistical tools. It is also the package that overwrites `dplyr::select()`, which is why we use `MASS::glm.nb()` in the code below and `dplyr::select()` in the code at the top.

To create our negative binomial model, we will use the `MASS::glm.nb()` function. Because there is only one "family" of distributions inside, we don't have to specify it, like we do with Poisson and Binomial models.

```
sibs_negbi <- MASS::glm.nb(sibs ~ marital + age + educ + sex,
                           data = gss, weights = weight)
with(sibs_negbi, cbind("Deviance" = deviance,
                       "DF Resid" = df.residual,
                       "Dev / DF" = deviance / df.residual))
```

```
##      Deviance DF Resid Dev / DF
## [1,] 1432.523     1372 1.044113
```

With this model, we see the deviance:DF ratio is back in order.

Let's take a look at our output.

```
summary(sibs_negbi)
```

```
##
## Call:
## MASS::glm.nb(formula = sibs ~ marital + age + educ + sex, data = gss,
##     weights = weight, init.theta = 5.029090549, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6320  -0.8027  -0.2423   0.4517   3.9295
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       1.749357   0.104469  16.745  < 2e-16 ***
## maritalMarried   -0.012157   0.051633  -0.235 0.813863
## maritalWidowed    0.014396   0.106895   0.135 0.892871
## maritalDivorced   0.022216   0.069567   0.319 0.749457
## maritalSeparated  0.275028   0.120059   2.291 0.021977 *
## age               0.004771   0.001392   3.427 0.000611 ***
## educ             -0.061885   0.006523  -9.488  < 2e-16 ***
## sexFemale         0.094332   0.038747   2.435 0.014910 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(5.0291) family taken to be 1)
##
```

```
##     Null deviance: 1560.5  on 1379  degrees of freedom
## Residual deviance: 1432.5  on 1372  degrees of freedom
## AIC: 6045
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  5.029
##           Std. Err.:  0.478
##
##  2 x log-likelihood:  -6026.977
```

We see similar output here as we did in our poisson model above. Note the Theta parameter at the bottom. This is the measure of dispersion in the model. Smaller thetas provide better justification for using negative binomial instead of Poisson. - Stata, SAS, SPSS, and every other software gives an alpha parameter which is equal to the inverse of Theta. - "The model becomes Poisson as the value of alpha approaches zero. ... Where alpha is closer to zero, the model statistics displayed in Poisson output are nearly the same as those of a negative binomial." Joseph Hilbe (2011): Negative Binomial Regression.

So our model Theta of 5 points to a high level of dispersion.

Interpretation

With the modeling done, we can interpret negative binomial coefficients just like Poisson ones.

Let's create a table to look at our output.

```
sibscompare <- list("Negative Binomial" = sibs_negbi,
                    "Poisson" = sibs_poisson,
                    "OLS" = sibs_ols)
modelsummary(sibscompare, output = "huxtable", stars = T,
             add_rows = data.frame("Theta", sibs_negbi$theta, "",""))
```

We interpret this just like we do for Poisson models. Note, however, that I had to add in a row to display the Theta parameter for our NB model.

Notice also that there is only a slight difference in coefficients in our NB model, but the Poisson model has smaller standard errors. Compared to the OLS model however, the standard errors are much more precise. We can see that the difference in the log of expected counts decreases with each additional year of education by about .06 and increases by about .005 with each year of age.

Incidence Rate Ratios

Just like with the Poisson model, we can also transform our estimates into incidence rate ratios. They are interpreted the same as the Poisson.

```
cbind(Estimate = coef(sibs_negbi),
      IRR = exp(coef(sibs_negbi)),
      exp(confint(sibs_negbi)))
```

```
## Waiting for profiling to be done...
```

```
##                     Estimate       IRR     2.5 %    97.5 %
## (Intercept)       1.74935678 5.7509024 4.6949653 7.0458726
## maritalMarried   -0.01215673 0.9879169 0.8938500 1.0919604
## maritalWidowed    0.01439577 1.0144999 0.8233836 1.2490733
## maritalDivorced   0.02221648 1.0224651 0.8928104 1.1706324
## maritalSeparated  0.27502842 1.3165681 1.0403651 1.6650987
```

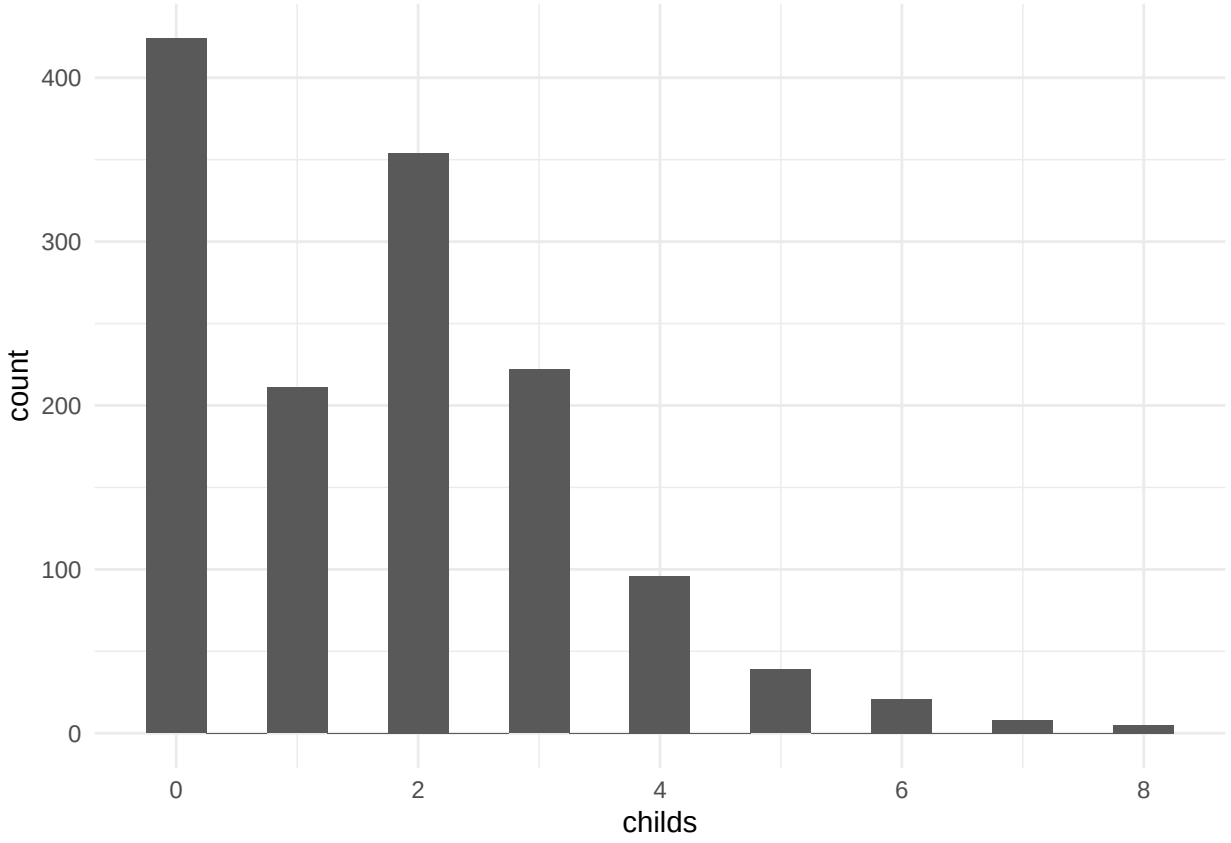
```
## age                0.00477112 1.0047825 1.0020586 1.0075133
## educ              -0.06188466 0.9399913 0.9281253 0.9519651
## sexFemale          0.09433222 1.0989248 1.0185794 1.1856813
```

So each additional year of age increases the rate ratio by a factor of 1.002 and each additional year of education decreases it by about 7%. Additionally, women tend to have about 10% more siblings than men.

Zero-Inflation: Number of Children

Sometimes, when we model count data, we have a lot of zeroes. For example, let's look back at the number of children people have:

```
ggplot(gss, aes(childs)) +
  geom_histogram(binwidth = .5) +
  theme_minimal()
```




```
table(gss$childs) %>% prop.table() %>% round(3)
```

```
##
##     0     1     2     3     4     5     6     7     8
## 0.307 0.153 0.257 0.161 0.070 0.028 0.015 0.006 0.004
```

We see that about 30% of our sample has no children. The Poisson distribution is not set up for that. After all, given some probability of having children, it's highly unlikely that there will not be any children.

What we can do instead is use a "Zero-Inflated Model." This model has two steps: 1. Model the likelihood of being zero. 2. For those who aren't zero, model the counts.

Model Creation

To create one, we use the function `pscl::zeroinfl()`. (PSCL = Political Science Computational Laboratory at Stanford U.)

In listing the regressors, we put a vertical line (found above enter on the keyboard) to separate the regressors for the count outcome (left side) from those for the zero-inflation outcome (right side). In the model below, I use marital status, age, the number of siblings, and sex for the count model, but marital, age, and education for the zero model.

If you have both sides the same, you can simply list one group of regressors, and they will be applied to both parts of the model.

After the term specification, it is optional, but recommended, to specify the distribution for the count model. The default is `dist = "poisson"`, which uses a Poisson distribution. If you think the model would be better fit by a zero-inflated negative binomial model, you can change the distribution to "negbin".

The default for the zero-inflation model is "logit," specified as `link = "logit"`. Other options (including probit) are available but less frequently used.

```
child_zip <- zeroinfl(childs ~ marital + age + sex + sibs | marital + age + educ,
                      dist = "poisson",
                      data = gss, weights = weight)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
child_zinb <- zeroinfl(childs ~ marital + age + sex + sibs | marital + age + educ,
                       dist = "negbin",
                       data = gss, weights = weight)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## Warning in sqrt(diag(vc)[np]): NaNs produced
```

We can then look at the log-likelihood for the two models and compare to see which we want. The function `lmtest::lrtest()` performs a likelihood ratio test to compare the log-likelihoods of the models and quantify the difference with a chi-square.

```
child_zip$loglik
```

```
## [1] -2046.781
```

```
child_zinb$loglik
```

```
## [1] -2046.78
```

```
lmtest::lrtest(child_zip, child_zinb)
```

```
## Likelihood ratio test
##
## Model 1: childs ~ marital + age + sex + sibs | marital + age + educ
## Model 2: childs ~ marital + age + sex + sibs | marital + age + educ
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1  15 -2046.8
## 2  16 -2046.8  1 0.0022     0.9627
```

In this case, the log-likelihood is the same, so there's no need to complicate the model by making it negative binomial.

Model Interpretation

Let's look at the zero-inflated Poisson model.

```
summary(child_zip)
```

```
##
## Call:
## zeroinfl(formula = childs ~ marital + age + sex + sibs | marital + age +
##     educ, data = gss, weights = weight, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -3.1583 -0.6181 -0.1901  0.3885  7.0462
##
## Count model coefficients (poisson with log link):
##                   Estimate Std. Error z value          Pr(>|z|)
## (Intercept)      -0.224247   0.110975  -2.021            0.0433 *
## maritalMarried    0.583250   0.079668   7.321 0.000000000000246 ***
## maritalWidowed    0.547414   0.115539   4.738 0.000002159145315 ***
## maritalDivorced   0.502010   0.091061   5.513 0.000000035302799 ***
## maritalSeparated  0.638036   0.130780   4.879 0.000001067879023 ***
## age               0.006694   0.001588   4.215 0.000024964177018 ***
## sexFemale         0.006479   0.042593   0.152            0.8791
## sibs              0.038190   0.007589   5.032 0.000000484484763 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                   Estimate Std. Error z value       Pr(>|z|)
## (Intercept)       2.39731    1.10379   2.172       0.029864 *
## maritalMarried   -1.41367    0.42318  -3.341       0.000836 ***
## maritalWidowed   -1.59534   24.15827  -0.066       0.947348
## maritalDivorced  -1.52392    1.15654  -1.318       0.187621
## maritalSeparated -7.08196   27.56115  -0.257       0.797214
## age              -0.38090    0.05935  -6.418 0.000000000138 ***
## educ              0.60292    0.10610   5.682 0.000000013279 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 32
## Log-likelihood: -2047 on 15 Df
```

We can see we have two sections. The top section is the model for counts. See that any relationship status increases the predicted number of children compared to those who were never married. Being older and having more siblings are also strongly associated with increasing the number of children. The difference in the log of the expected counts of children decreases by about .04 for each sibling a person has. Big families create big families.

Below the count model, we see the zero-inflation model. Here, we are modeling the likelihood that `childs==0`. It sounds counterintuitive, I know. Think of it as the likelihood of not being including in the first (counts) model.

As we would expect, being married decreases the likelihood of not having children, as does being older, while education increases it.

Viewing ZIM's

Viewing ZIM's is just about always going to be ugly, so be warned.

Viewing it in stargazer will give you only the Poisson portion, ignoring the zero-inflated part.

```
stargazer(child_zip, child_ols,
          single.row = T,
          type = "text")
```

```
##
## ===============================================================
##                                Dependent variable:
##                     ------------------------------------------
##                                       childs
##                        zero-inflated             OLS
##                          count data
##                             (1)                   (2)
## ---------------------------------------------------------------
## maritalMarried      0.583*** (0.080)     1.167*** (0.096)
## maritalWidowed      0.547*** (0.116)     0.884*** (0.207)
## maritalDivorced     0.502*** (0.091)     0.959*** (0.131)
## maritalSeparated    0.638*** (0.131)     1.388*** (0.244)
## age                 0.007*** (0.002)     0.028*** (0.003)
## educ                                    -0.093*** (0.012)
## sexFemale            0.006 (0.043)        0.088 (0.072)
## sibs                0.038*** (0.008)
## Constant            -0.224** (0.111)     0.937*** (0.200)
## ---------------------------------------------------------------
## Observations             1,380                 1,380
## R2                                             0.318
## Adjusted R2                                    0.314
## Log Likelihood         -2,046.781
## Residual Std. Error                     1.334 (df = 1372)
## F Statistic                           91.227*** (df = 7; 1372)
## ===============================================================
## Note:                                *p<0.1; **p<0.05; ***p<0.01
```

And viewing it in modelsummary will give you both, but will tag each of the terms separately for the zero-inflated and count portions of the model, which also means you will have separate rows if you have multiple models.

```
modelsummary(list(child_zip, child_ols), output = "huxtable", stars = T)
```

With a lot of work, we can them side by side. Here is what I'm doing: 1. Create two objects to store the estimates and standard errors for the counting model and the zero model. 2. Create an object to store whatever model diagnostic information I want. 3. For each of the objects in step 1, I create a list that has two parts: "tidy", which contains the model terms and estimates, and "glance", which has the model diagnostics. 4. Change the classes for both of these lists from step 3 into "modelssummary_list". 5. Create a list that contains the two lists from earlier, as well as the OLS model. 6. Create the modelsummary object as normal.

```
# Step 1
zimest_count <- get_estimates(child_zip) %>%
  filter(str_detect(term,"count")) %>% mutate(term = str_remove(term,"count_"))
zimest_zero <- get_estimates(child_zip) %>%
  filter(str_detect(term,"zero")) %>% mutate(term = str_remove(term,"zero_"))

# Step 2
zimest_glance <- data.frame(
  "Num.Obs." = child_zip$n,
  "Log.Lik." = logLik(child_zip),
```

```
  AIC = AIC(child_zip),
  BIC = BIC(child_zip)
  )
# Step 3
zimmod_count <- list(
  tidy = zimest_count,
  glance = zimest_glance
  )
zimmod_zero <- list(
  tidy = zimest_zero,
  glance = zimest_glance
  )

# Step 4
class(zimmod_count) <- "modelsummary_list"
class(zimmod_zero) <- "modelsummary_list"

# Step 5
freqmods <- list(
  "ZIM - Zero" = zimmod_zero,
  "ZIM - Count" = zimmod_count,
  "OLS" = child_ols
  )

# Step 6
modelsummary(freqmods, output = "huxtable", stars = T,
             title = "Model  Comparison - Number  of Children")
```

Another option is to use the `coef_map` option in `modelsummary::modelsummary()` to rename the coefficients.

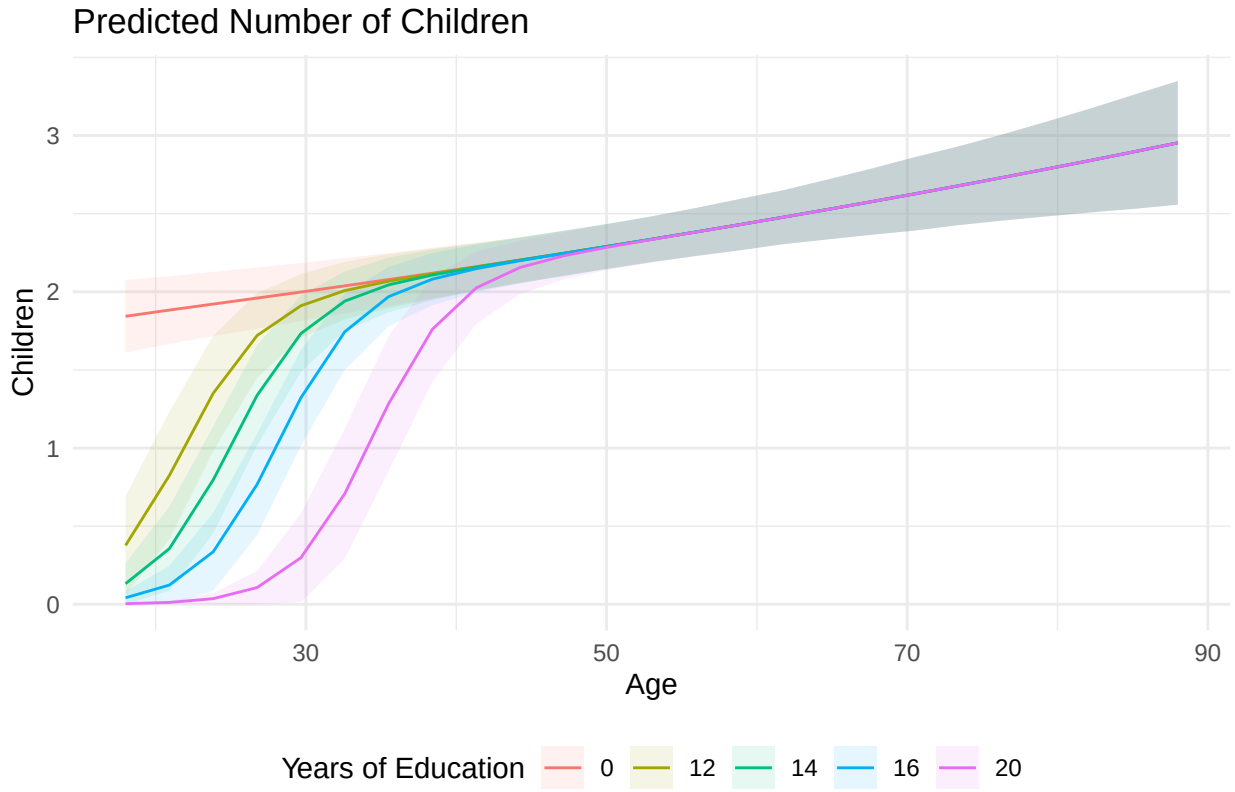
```
zipnames <- c(
  'count_(Intercept)' = "Count: Constant",
  'count_maritalMarried' = "Count: Married",
  'count_maritalWidowed' = "Count: Widowed",
  'count_maritalDivorced' = "Count: Divorced",
  'count_maritalSeparated' = "Count: Separated",
  'count_age'  = "Count: Age",
  'count_sexFemale' = "Count: Female",
  'count_sibs' = "Count: Siblings",
  'zero_(Intercept)' = "Zero: Constant",
  'zero_maritalMarried' = "Zero: Married",
  'zero_maritalWidowed' = "Zero: Widowed",
  'zero_maritalDivorced' = "Zero: Divorced",
  'zero_maritalSeparated' = "Zero: Separated",
  'zero_age' = "Zero: Age",
  'zero_sexFemale' = "Zero: Female",
  'zero_sibs' = "Zero: Siblings",
  'zero_educ' = "Zero: Education"
)
modelsummary(list("ZI Poisson" = child_zip,
                  "ZI Neg Bi" = child_zinb),
             coef_map = zipnames,
             output = "huxtable", stars = T)
```

Predictions

We can create predictions from our estimates using the `marginaleffects` package.

Below, we see that there seems to be an effect of education and age together, up to around the mid-30's, such that higher levels of education are associated with decreased odds of having children. But, as my parents would be happy to know, such effect balances out at a point and highly-educated adults still bring home grandchildren.

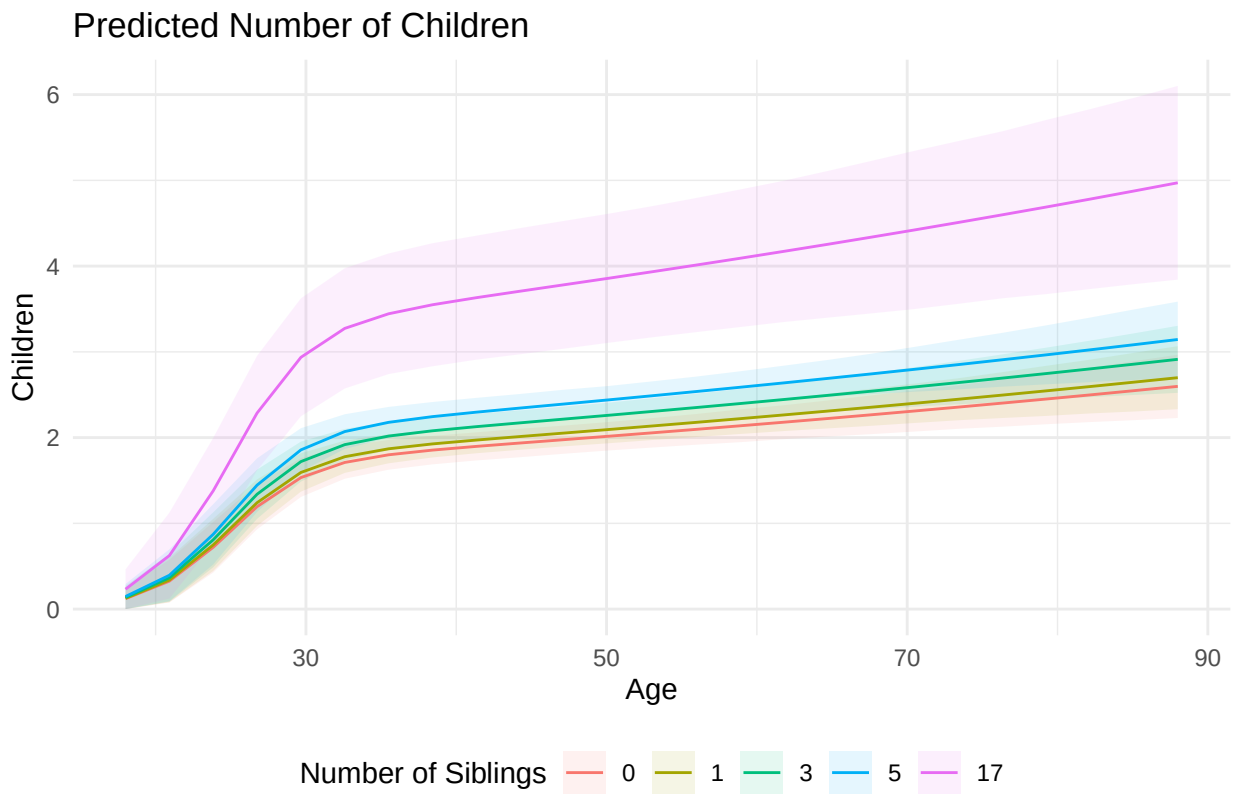
```
plot_cap(child_zip, condition = c("age", "educ"))  +
  labs(title = "Predicted Number of Children",
       caption = "Data: 2018 General Social Survey",
       color = "Years of Education",
       fill = "Years of Education",
       y = "Children", x = "Age") +
  theme(legend.position = "bottom")
```



Because I only included education in the zero-inflated portion of the model, we see a logistic curve for those with education and a more linear function for those without.

We can do this, too, with the number of siblings a person has:

```
plot_cap(child_zip, condition = c("age", "sibs"))  +
  labs(title = "Predicted Number of Children",
       caption = "Data: 2018 General Social Survey",
       color = "Number of Siblings", fill = "Number of Siblings",
       y = "Children", x = "Age") +
  theme(legend.position = "bottom")
```



In this graph, people with a lot of siblings will have more children, but everyone else is around the same number, though with a slightly increasing trend as people age.

Bayesian Models

For Bayesian modeling, as usual, the `stan_glm()` command works out similar to the usual `glm()`. For Poisson models, the `family = "poisson"` argument is also the same. Two quick differences:

1. The big difference is that negative binomial models take the form `family = "neg_binomial_2"` instead of using `glm.nb()` as we did earlier.
2. There is not (yet!) a method for zero-inflated models in `rstanarm`. As such, we'll be using just the count model (Poisson).

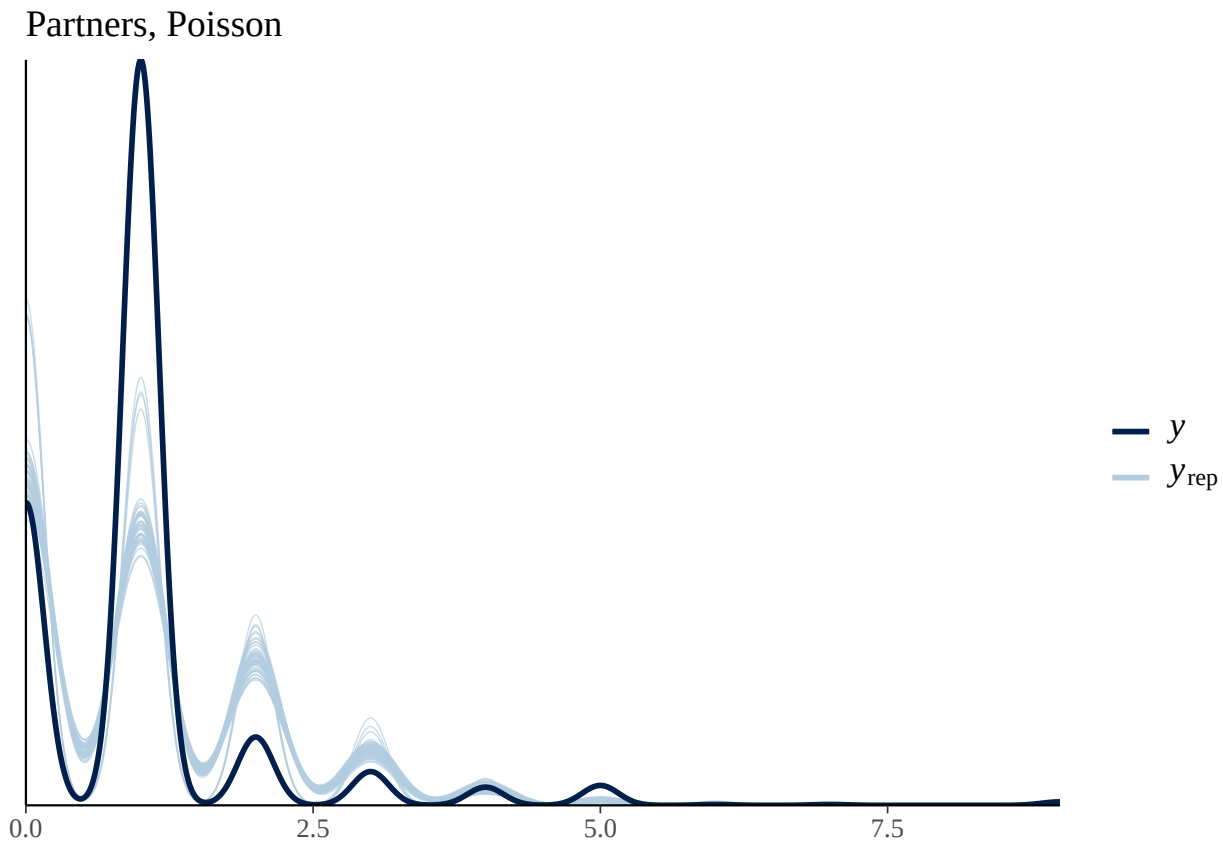
Also, if you want more info, you can follow this link to Aki Vehtari's page, where he shows how to do this modeling: https://avehtari.github.io/modelselection/roaches.html

```
partner_stan_poisson <- stan_glm(partners ~ marital + age + educ + sex,
                                 data = gss, family = "poisson",
                                 seed = seed, chains = 1, refresh = 0,
                                 iter = 2000, warmup = 1000)
sibs_stan_negbi <- stan_glm(sibs ~ marital + age + educ + sex,
                            data = gss, family = "neg_binomial_2",
                            seed = seed, chains = 1, refresh = 0,
                            iter = 2000, warmup = 1000)
child_stan_poisson <- stan_glm(childs ~ marital + age + educ + sex,
                               data = gss, family = "poisson",
                               seed = seed, chains = 1, refresh = 0,
```

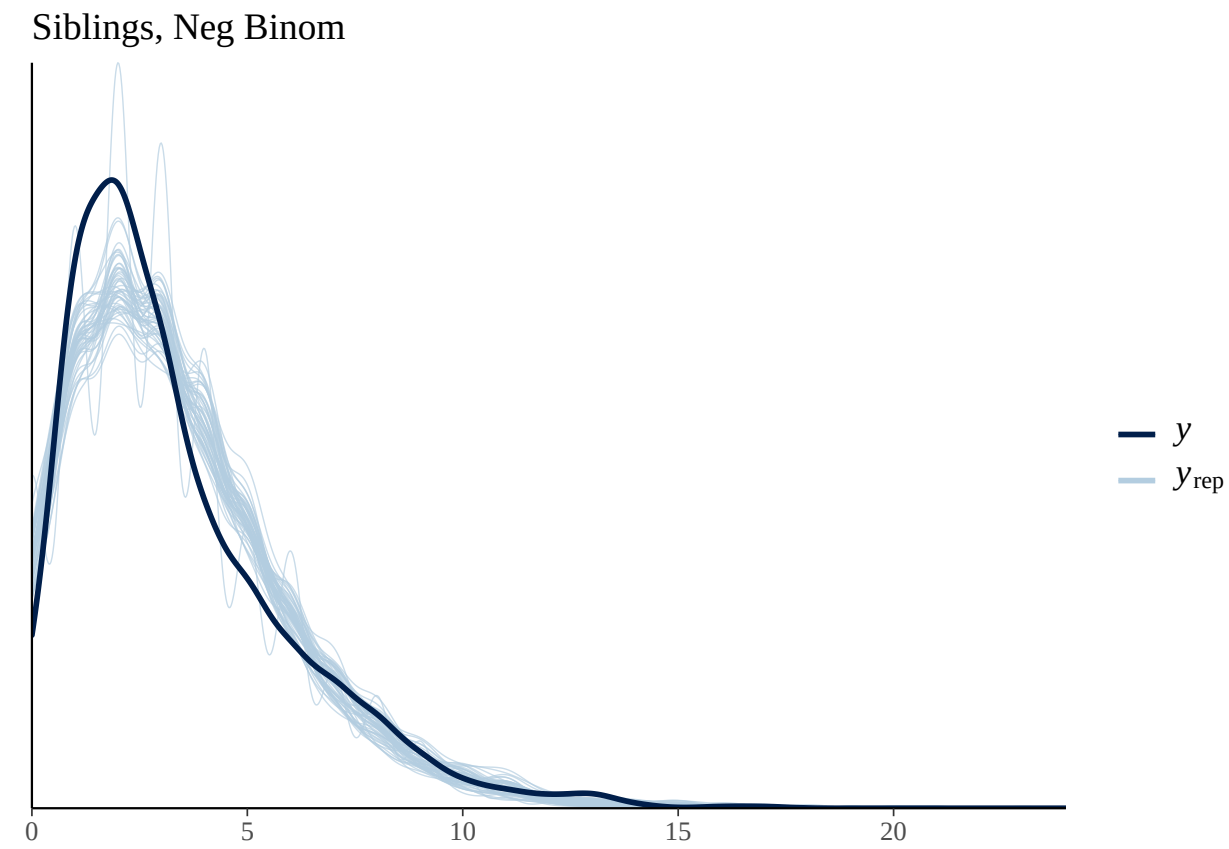
```
                                iter = 2000, warmup = 1000)

modelsummary(list(partner_stan_poisson, sibs_stan_negbi, child_stan_poisson),
             output = "huxtable")
```

Since count data is subject to often-unusual distributions, let's plot our posterior distributions and compare them to the original data.

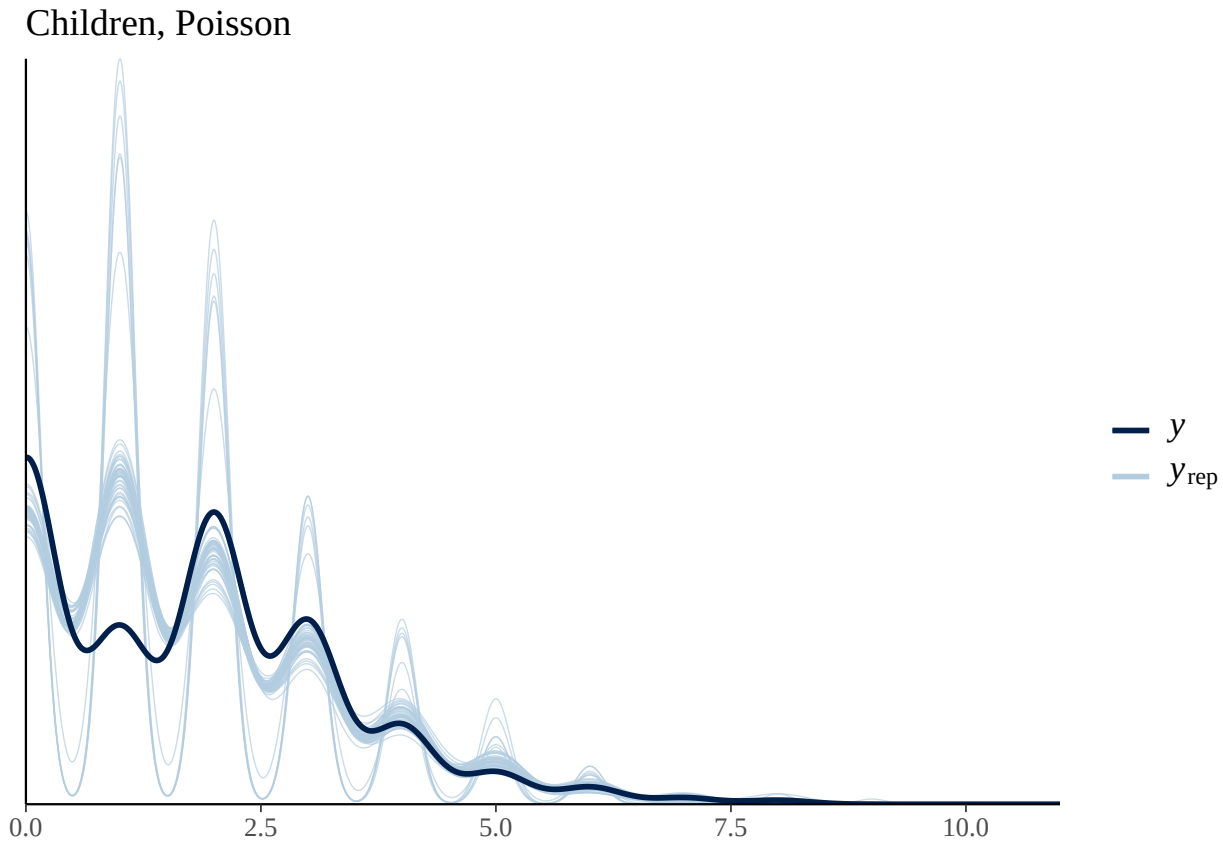
```
pp_check(partner_stan_poisson) + ggtitle("Partners, Poisson")
```




```
pp_check(sibs_stan_negbi) + ggtitle("Siblings, Neg Binom")
```




```
pp_check(child_stan_poisson) + ggtitle("Children, Poisson")
```



As we can see, our model performed exceptionally well on predicting siblings with the negative binomial distribution. For partners, it tended to predict more siblings than the actual. For children, it actually did fairly well, except for missing the mark on those with only one child. I expect that this is due to the zero-inflated nature of the data, which is not accounted for well in the Poisson model.

	Negative Binomial	Poisson	OLS
(Intercept)	1.749***	1.767***	5.367***
	(0.104)	(0.079)	(0.367)
maritalMarried	-0.012	-0.006	-0.014
	(0.052)	(0.040)	(0.175)
maritalWidowed	0.014	0.015	0.138
	(0.107)	(0.081)	(0.380)
maritalDivorced	0.022	0.025	0.077
	(0.070)	(0.054)	(0.241)
maritalSeparated	0.275*	0.276**	1.092*
	(0.120)	(0.088)	(0.448)
age	0.005***	0.004***	0.015**
	(0.001)	(0.001)	(0.005)
educ	-0.062***	-0.062***	-0.215***
	(0.007)	(0.005)	(0.023)
sexFemale	0.094*	0.092**	0.294*
	(0.039)	(0.030)	(0.132)
Num.Obs.	1380	1380	1380
R2			0.082
R2 Adj.			0.077
AIC	6045.0	6275.2	6552.0
BIC	6092.0	6317.1	6599.1
Log.Lik.	-3013.488	-3129.622	-3267.013
F	18.148	32.186	17.404
RMSE	1.02	1.30	2.45
Theta	5.029		
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001			

	Model 1	Model 2
count_(Intercept)	-0.224*	
	(0.111)	
count_maritalMarried	0.583***	
	(0.080)	
count_maritalWidowed	0.547***	
	(0.116)	
count_maritalDivorced	0.502***	
	(0.091)	
count_maritalSeparated	0.638***	
	(0.131)	
count_age	0.007***	
	(0.002)	
count_sexFemale	0.006	
	(0.043)	
count_sibs	0.038***	
	(0.008)	
zero_(Intercept)	2.397*	
	(1.104)	
zero_maritalMarried	-1.414***	
	(0.423)	
zero_maritalWidowed	-1.595	
	(24.158)	
zero_maritalDivorced	-1.524	
	(1.157)	
zero_maritalSeparated	-7.082	
	(27.561)	
zero_age	-0.381***	
	(0.059)	
zero_educ	0.603***	
	(0.106)	
(Intercept)		0.937***
		(0.200)

Table 3: Model Comparison - Number of Children

	ZIM - Zero	ZIM - Count	OLS
(Intercept)	2.397*	-0.224*	0.937***
	(1.104)	(0.111)	(0.200)
maritalMarried	-1.414***	0.583***	1.167***
	(0.423)	(0.080)	(0.096)
maritalWidowed	-1.595	0.547***	0.884***
	(24.158)	(0.116)	(0.207)
maritalDivorced	-1.524	0.502***	0.959***
	(1.157)	(0.091)	(0.131)
maritalSeparated	-7.082	0.638***	1.388***
	(27.561)	(0.131)	(0.244)
age	-0.381***	0.007***	0.028***
	(0.059)	(0.002)	(0.003)
educ	0.603***		-0.093***
	(0.106)		(0.012)
sexFemale		0.006	0.088
		(0.043)	(0.072)
sibs		0.038***	
		(0.008)	
Num.Obs.	1380.000	1380.000	1380
R2			0.318
R2 Adj.			0.314
AIC	4123.6	4123.6	4876.0
BIC	4202.0	4202.0	4923.1
Log.Lik.	-2046.781	-2046.781	-2429.005
F			91.227
RMSE			1.33
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001			

	ZI Poisson	ZI Neg Bi
Count: Constant	-0.224*	-0.224*
	(0.111)	(0.111)
Count: Married	0.583***	0.583***
	(0.080)	(0.080)
Count: Widowed	0.547***	0.547***
	(0.116)	(0.116)
Count: Divorced	0.502***	0.502***
	(0.091)	(0.091)
Count: Separated	0.638***	0.638***
	(0.131)	(0.131)
Count: Age	0.007***	0.007***
	(0.002)	(0.002)
Count: Female	0.006	0.006
	(0.043)	(0.043)
Count: Siblings	0.038***	0.038***
	(0.008)	(0.008)
Zero: Constant	2.397*	2.421*
	(1.104)	(1.103)
Zero: Married	-1.414***	-1.412***
	(0.423)	(0.423)
Zero: Widowed	-1.595	-2.239
	(24.158)	(33.146)
Zero: Divorced	-1.524	-1.512
	(1.157)	(1.147)
Zero: Separated	-7.082	-7.114
	(27.561)	(27.941)
Zero: Age	-0.381***	-0.380***
	(0.059)	(0.059)
Zero: Education	0.603***	0.600***
	(0.106)	(0.106)
Num.Obs.	1380	1380
R2	0.443	0.444

	Model 1	Model 2	Model 3
(Intercept)	0.730	1.752	-0.156
	(0.172)	(0.109)	(0.119)
maritalMarried	-0.090	-0.014	0.966
	(0.066)	(0.054)	(0.069)
maritalWidowed	-1.164	0.066	0.842
	(0.239)	(0.095)	(0.107)
maritalDivorced	-0.084	0.069	0.890
	(0.089)	(0.070)	(0.079)
maritalSeparated	-0.306	0.308	1.071
	(0.200)	(0.108)	(0.122)
age	-0.016	0.004	0.012
	(0.002)	(0.001)	(0.001)
educ	0.015	-0.057	-0.051
	(0.011)	(0.006)	(0.006)
sexFemale	-0.276	0.074	0.097
	(0.053)	(0.040)	(0.044)
Num.Obs.	1380	1380	1380
RMSE	1.00	1.00	1.00
algorithm	sampling	sampling	sampling
pss	1000.000	1000.000	1000.000