432 Class 14 Slides

thomaselove.github.io/432

2022-03-01

Today’s Topic

Regression Models for Count Outcomes

- Modeling approaches illustrated in these slides
    - Poisson Regression
    - Negative Binomial Regression
    - Two types of Zero-inflated model
        - ZIP (Zero-inflated Poisson)
        - ZINB (Zero-inflated Neg. Binomial)
- The slides also discuss (but we’ll largely skip)
    - Two types of Hurdle model
        - using a Poisson approach
        - using a Negative Binomial approach

Chapter 19 of the Course Notes describes this material.

Installing the `countreg` package

The `countreg` package is available on R-Forge.

To build rootograms to visualize the results of regression models on count outcomes, I have decided for the moment to continue to use the countreg package, which is currently available on R-Forge only.

To install `countreg`, type

```
install.packages("countreg",
                 repos="http://R-Forge.R-project.org")
```

into the R Console within R Studio. I will not be loading `countreg` in this work, though.

Setup

Again, we assume you have already installed the countreg package from R-Forge.

```
library(magrittr); library(here); library(janitor)
library(knitr); library(conflicted)
library(MASS)
library(pscl)
library(VGAM)
library(broom); library(rsample); library(yardstick)
library(tidyverse)

conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")

theme_set(theme_bw())
```

An Overview

Generalized Linear Models for Count Outcomes

We want to build a generalized linear model to predict count data using one or more predictors.

In count data, the observations are non-negative integers (0, 1, 2, 3, . . .)

- the number of COVID-19 hospitalizations in Ohio yesterday
- the number of mutations within a particular search grid
- the number of days in the past 30 where your mental health was poor

The Poisson and the Negative Binomial probability distributions will be useful.

The Poisson Probability Distribution

The Poisson probability model describes the probability of a given number of events occurring in a fixed interval of time or space.

- If events occur with a constant mean rate, and independently of the time since the last event, the Poisson model is appropriate.
- The probability mass function for a discrete random variable with Poisson distribution follows.

$$Pr(Y = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- k is the number of times an event occurs in an interval, and k can take the values 0, 1, 2, 3, . . .
- The parameter λ (lambda) is equal to the expected value (mean) of Y and is also equal to the variance of Y.

The Negative Binomial Probability Distribution

The Negative Binomial distribution models the number of failures in a sequence of independent and identically distributed Bernoulli trials before a specified number of successes occurs.

- The probability mass function for a discrete random variable with a negative binomial distribution follows.

$$Pr(Y = k) = \binom{k + r - 1}{k} p^r (1 - p)^k$$

- k is the number of failures (units of time) before the rth event occurs, and k can take the values 0, 1, 2, 3, ...
- The mean of the random variable Y which follows a negative binomial distribution is $rp/(1-p)$ and the variance is $rp/(1-p)^2$.

Poisson Regression and the possibility of overdispersion

- Poisson regression assumes that the outcome Y follows a Poisson distribution, and that the logarithm of the expected value of Y (its mean) can be modeled by a linear combination of a set of predictors.
  - A Poisson regression makes the strong assumption that the variance of Y is equal to its mean.
  - A Poisson model might fit poorly due to **overdispersion**, where the variance of Y is larger than we'd expect based on the mean of Y.
  - Quasipoisson models are available which estimate an overdispersion parameter, but we'll skip those.

We will show the use of `glm` to fit Poisson models, by using `family = "Poisson"`.

Negative Binomial Regression to generalize the Poisson

- Negative binomial regression is a generalization of Poisson regression which loosens the assumption that the variance of Y is equal to its mean, and thus produces models which fit a broader class of data.

We will demonstrate the use of `glm.nb` from the `MASS` package to fit negative binomial regression models.

Zero-inflated approaches

- Both the Poisson and Negative Binomial regression approaches may under-estimate the number of zeros compared to the data.
- To better match up the counts of zero, zero-inflated models fit:
  - a logistic regression to predict the extra zeros, along with
  - a Poisson or Negative Binomial model to predict the counts, including some zeros.

We will demonstrate the use of `zeroinfl` from the `pscl` package to fit zero-inflated Poisson (or ZIP) and zero-inflated negative binomial (or ZINB) regressions.

Hurdle models (in the slides, but not today's focus)

A hurdle model predicts the count outcome by making an assumption that there are two processes at work:

- a process that determines whether the count is zero or not zero (usually using logistic regression), and
- a process that determines the count when we know the subject has a positive count (usually using a truncated Poisson or Negative Binomial model where no zeros are predicted)

These slides use the `hurdle` function from the `pscl` package to fit these models.

Comparing Models

1. A key tool will be a graphical representation of the fit of the models to the count outcome, called a **rootogram**. We'll use the rootograms produced by the `countreg` package to help us.
2. We'll also demonstrate a Vuong hypothesis testing approach (from the `lmtest` package) to help us make decisions between various types of Poisson models or various types of Negative Binomial models on the basis of improvement in fit of things like bias-corrected AIC or BIC.
3. We'll also demonstrate the calculation of pseudo-R square statistics for comparing models, which can be compared in a validation sample as well as in the original modeling sample.

The `medicare` data

The medicare example

The data we will use come from the NMES1988 data set in R's AER package, although I have built a cleaner version for you in the medicare.csv file on our web site. These are essentially the same data as are used in my main resource from the University of Virginia for hurdle models.

These data are a cross-section originating from the US National Medical Expenditure Survey (NMES) conducted in 1987 and 1988. The NMES is based upon a representative, national probability sample of the civilian non-institutionalized population and individuals admitted to long-term care facilities during 1987. The data are a subsample of individuals ages 66 and over all of whom are covered by Medicare (a public insurance program providing substantial protection against health-care costs), and some of whom also have private supplemental insurance.

```r
medicare <- read_csv(here("data/medicare.csv")) %>%
    type.convert(as.is = FALSE)
```

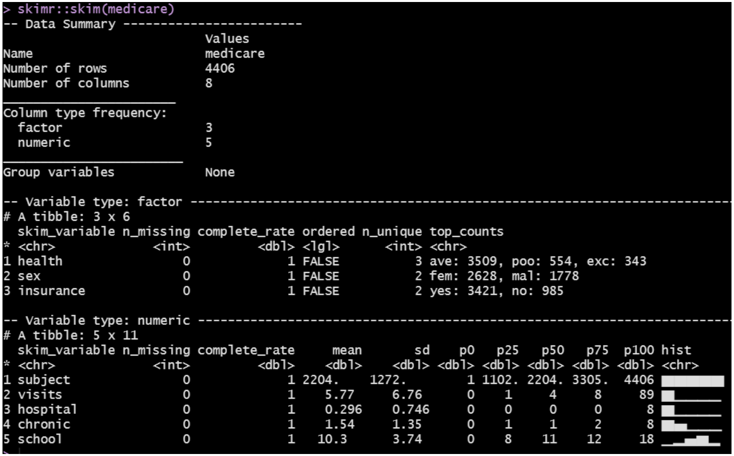
The `medicare` code book

Variable	Description
`subject`	subject number (code)
`visits`	outcome of interest: number of physician office visits
`hospital`	number of hospital stays
`health`	self-perceived health status (poor, average, excellent)
`chronic`	number of chronic conditions
`sex`	male or female
`school`	number of years of education
`insurance`	is the subject (also) covered by private insurance? (yes or no)

Today's Goal

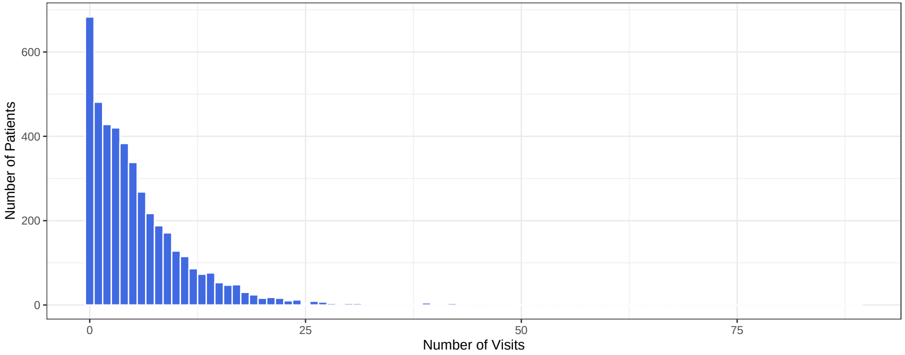
Predict `visits` using main effects of the 6 predictors (excluding `subject`)

Skimming the medicare tibble



```
> skimr::skim(medicare)
-- Data Summary ------------------------
                           Values
Name                       medicare
Number of rows             4406
Number of columns          8
_______________________
Column type frequency:
  factor                   3
  numeric                  5
________________________
Group variables            None

-- Variable type: factor ------------------------------------------------------------
# A tibble: 3 x 6
  skim_variable n_missing complete_rate ordered n_unique top_counts
* <chr>             <int>         <dbl> <lgl>      <int> <chr>
1 health                0             1 FALSE          3 ave: 3509, poo: 554, exc: 343
2 sex                   0             1 FALSE          2 fem: 2628, mal: 1778
3 insurance             0             1 FALSE          2 yes: 3421, no: 985

-- Variable type: numeric -----------------------------------------------------------
# A tibble: 5 x 11
  skim_variable n_missing complete_rate     mean       sd    p0   p25   p50   p75  p100 hist
* <chr>             <int>         <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 subject               0             1 2204.    1272.        1 1102. 2204. 3305.  4406 ▇▇▇▇▇
2 visits                0             1    5.77     6.76      0     1     4     8    89 ▇▁▁▁▁
3 hospital              0             1    0.296    0.746     0     0     0     0     8 ▇▁▁▁▁
4 chronic               0             1    1.54     1.35      0     1     1     2     8 ▇▃▁▁▁
5 school                0             1   10.3      3.74      0     8    11    12    18 ▁▁▃▇▁
```

Our outcome, `visits`



```
mosaic::favstats(~ visits, data = medicare)
```

```
 min Q1 median Q3 max     mean       sd    n missing
   0  1      4  8  89 5.774399 6.759225 4406       0
```

visits numerical summaries

```
medicare %$% Hmisc::describe(visits)
```

```
visits
       n  missing distinct     Info     Mean      Gmd
    4406        0       60    0.992    5.774    6.227
     .05      .10      .25      .50      .75      .90
       0        0        1        4        8       13
     .95
      17

lowest :  0  1  2  3  4, highest: 63 65 66 68 89
```

Reiterating the Goal

Predict `visits` using some combination of these 6 predictors...

Predictor	Description
`hospital`	number of hospital stays
`health`	self-perceived health status (poor, average, excellent)
`chronic`	number of chronic conditions
`sex`	male or female
`school`	number of years of education
`insurance`	is the subject (also) covered by private insurance? (yes or no)

We'll build separate training and test samples to help us validate.

Partitioning the Data into Training vs. Test Samples

```r
set.seed(432)
med_split <- initial_split(medicare, prop = 0.75)

med_train = training(med_split)
med_test = testing(med_split)
```

I've held out 25% of the `medicare` data for the test sample.

```r
dim(med_train)
```

```
[1] 3304    8
```

```r
dim(med_test)
```

```
[1] 1102    8
```

`mod_1`: A Poisson Regression

Poisson Regression

Assume our count data (`visits`) follows a Poisson distribution with a mean conditional on our predictors.

```
mod_1 <- glm(visits ~ hospital + health + chronic +
                    sex + school + insurance,
                data = med_train, family = "poisson")
```

The Poisson model uses a logarithm as its link function, so the model is actually predicting log(`visits`).

Note that we're fitting the model here using the training sample alone.

`mod_1` (Poisson) model coefficients

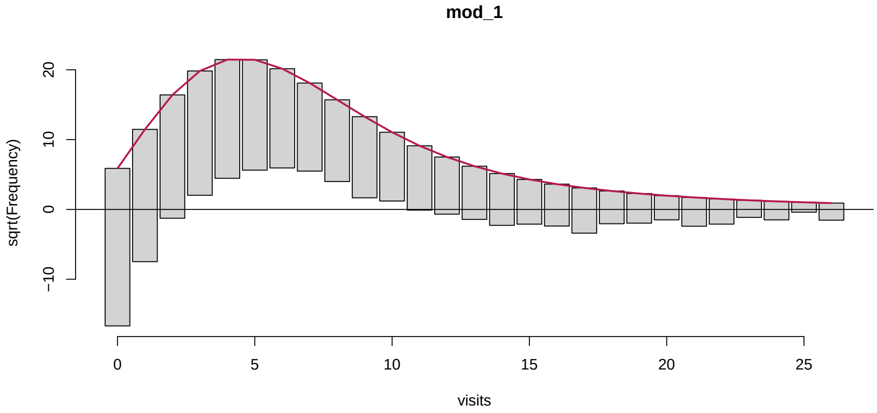
```
tidy(mod_1) %>% kable(digits = c(0, 3, 3, 1, 3))
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.985	0.027	36.1	0
hospital	0.164	0.007	24.4	0
healthexcellent	-0.359	0.035	-10.3	0
healthpoor	0.310	0.020	15.3	0
chronic	0.137	0.005	26.1	0
sexmale	-0.098	0.015	-6.6	0
school	0.031	0.002	14.8	0
insuranceyes	0.200	0.019	10.3	0

If Harry and Larry have the same values for all other predictors but only Harry has private insurance, the model predicts Harry to have a value of log(`visits`) that is 0.2 larger than Larry's log(`visits`).

Visualize fit with a (Hanging) Rootogram

```
countreg::rootogram(mod_1)
```



See the next slide for details on how to interpret this. . .

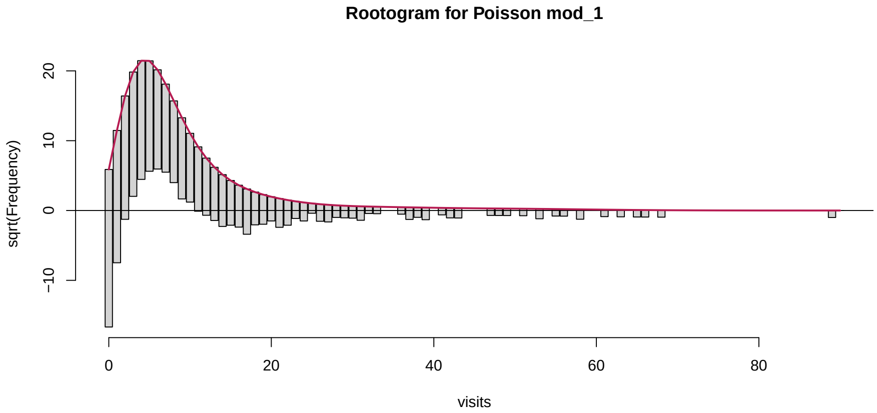
Interpreting the Rootogram

- The red curved line is the theoretical Poisson fit.
- "Hanging" from each point on the red line is a bar, the height of which represents the observed counts.
  - A bar hanging below 0 indicates that the model under-predicts that value. (Model predicts fewer values than the data show.)
  - A bar hanging above 0 indicates over-prediction of that value. (Model predicts more values than the data show.)
- The counts have been transformed with a square root transformation to prevent smaller counts from getting obscured and overwhelmed by larger counts.

For more information on rootograms, check out https://arxiv.org/pdf/1605.01311.

The Complete Rootogram for `mod_1`

```
countreg::rootogram(mod_1, max = 90,
                    main = "Rootogram for Poisson mod_1")
```



This shows what happens with the subject with 89 visits.

Interpreting the Rootogram for `mod_1`

In `mod_1`, we see a great deal of under-prediction for counts of 0 and 1, then over-prediction for visit counts in the 3-10 range, with some under-prediction again at more than a dozen or so visits.

- Our Poisson model (`mod_1`) doesn't fit enough zeros or ones, and fits too many 3-12 values, then not enough of the higher values.

Store Training Sample `mod_1` Predictions

We'll use the `augment` function to store the predictions within our training sample. Note the use of `"response"` to predict `visits`, not log(`visits`).

```
mod_1_aug <- augment(mod_1, med_train,
                     type.predict = "response")

mod_1_aug %>% select(subject, visits, .fitted) %>%
  head(3)
```

```
# A tibble: 3 x 3
  subject visits .fitted
    <int>  <int>   <dbl>
1     355     19    5.02
2    2661      3    4.21
3    2895      0    4.65
```

Summarizing Training Sample `mod_1` Fit

Within our training sample, `mod_1_aug` now contains both the actual counts (`visits`) and the predicted counts (in `.fitted`) from `mod_1`. We'll summarize the fit. . .

```
mets <- metric_set(rsq, rmse, mae)
mod_1_summary <-
  mets(mod_1_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_1") %>% relocate(model)
mod_1_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_1	rsq	standard	0.100
mod_1	rmse	standard	6.594
mod_1	mae	standard	4.189

These will become interesting as we build additional models.

`mod_2`: A Negative Binomial Regression

Fitting the Negative Binomial Model

The negative binomial model requires the estimation of an additional parameter, called θ (theta). The default link for this generalized linear model is also a logarithm, like the Poisson.

```
mod_2 <- MASS::glm.nb(visits ~ hospital + health + chronic +
                      sex + school + insurance,
                  data = med_train)
```

The estimated dispersion parameter value θ is...

```
summary(mod_2)$theta
```

```
[1] 1.21109
```

The Poisson model is essentially the negative binomial model assuming a known $\theta = 1$.

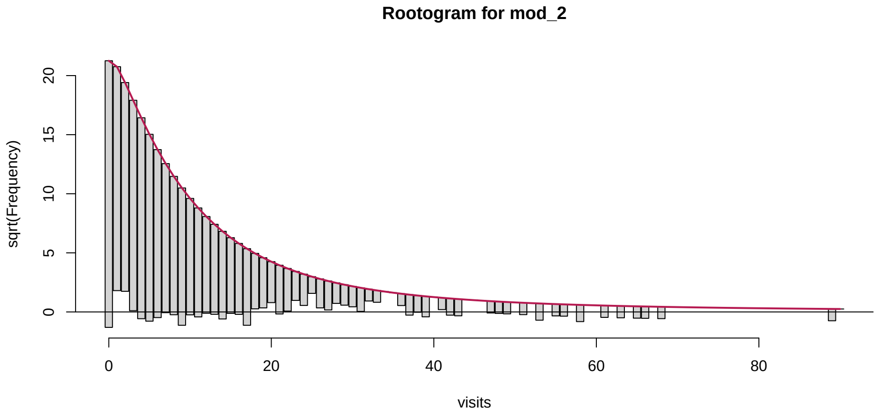
`mod_2` (Negative Binomial) coefficients

```
tidy(mod_2) %>% kable(digits = c(0, 3, 3, 1, 3))
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.875	0.063	14.0	0.000
hospital	0.224	0.023	9.9	0.000
healthexcellent	-0.336	0.070	-4.8	0.000
healthpoor	0.360	0.056	6.5	0.000
chronic	0.169	0.014	12.2	0.000
sexmale	-0.109	0.036	-3.0	0.002
school	0.031	0.005	6.1	0.000
insuranceyes	0.237	0.046	5.2	0.000

Rootogram for Negative Binomial Model

```
countreg::rootogram(mod_2, max = 90,
                    main = "Rootogram for mod_2")
```



Does this look better than the Poisson rootogram?

Store Training Sample `mod_2` Predictions

```
mod_2_aug <- augment(mod_2, med_train,
                     type.predict = "response")

mod_2_aug %>% select(subject, visits, .fitted) %>%
  head(3)
```

```
# A tibble: 3 x 3
  subject visits .fitted
    <int>  <int>   <dbl>
1     355     19    5.22
2    2661      3    4.08
3    2895      0    4.39
```

- Note that this may throw a warning about who maintains tidiers for `negbin` models. I'd ignore it.

Summarizing Training Sample `mod_2` Fit

As before, `mod_2_aug` now has actual (`visits`) and predicted counts (in `.fitted`) from `mod_2`.

```
mod_2_summary <-
  mets(mod_2_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_2") %>% relocate(model)
mod_2_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_2	rsq	standard	0.078
mod_2	rmse	standard	6.941
mod_2	mae	standard	4.252

So Far in our Training Sample

The reasonable things to summarize in sample look like the impressions from the rootograms and the summaries we've prepared so far.

```
bind_rows(mod_1_summary, mod_2_summary) %>%
  pivot_wider(names_from = model,
              values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2
rsq	standard	0.100	0.078
rmse	standard	6.594	6.941
mae	standard	4.189	4.252

Model	Rootogram impressions
mod_1 (P)	Many problems. Data appear overdispersed.
mod_2 (NB)	Still not enough zeros; some big predictions.

`mod_3`: Zero-Inflated Poisson (ZIP) Model

Zero-Inflated Poisson (ZIP) model

The zero-inflated Poisson model describes count data with an excess of zero counts.

The model posits that there are two processes involved:

- a logistic regression model is used to predict excess zeros
- while a Poisson model is used to predict the counts

We'll use the `pscl` package to fit zero-inflated models.

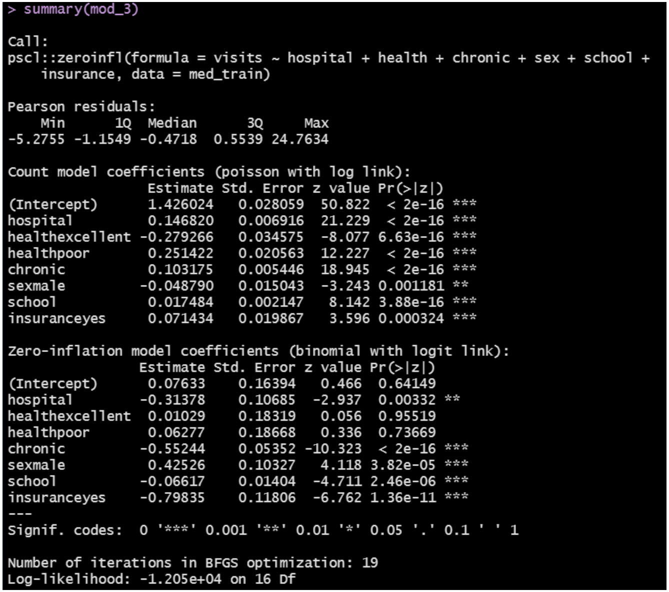
```
mod_3 <- pscl::zeroinfl(visits ~ hospital + health +
                    chronic + sex + school + insurance,
                    data = med_train)
```

mod_3 ZIP coefficients

Sadly, there's no broom tidying functions for these zero-inflated models.

```
summary(mod_3)
```

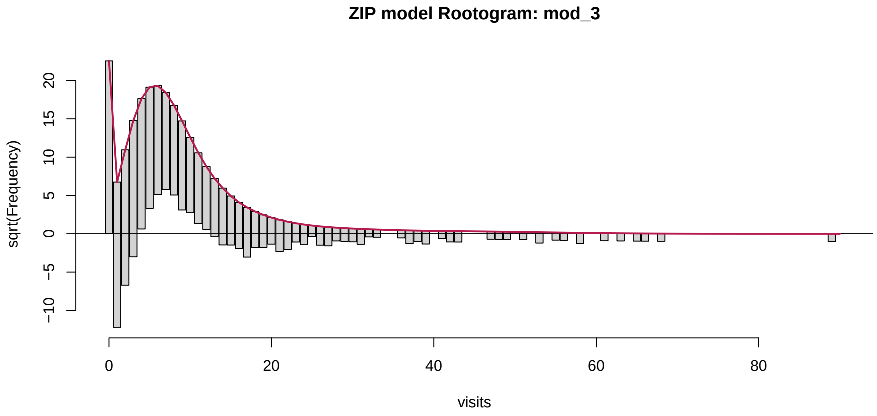
Screenshot on next slide...



```
> summary(mod_3)

Call:
pscl::zeroinfl(formula = visits ~ hospital + health + chronic + sex + school +
    insurance, data = med_train)

Pearson residuals:
    Min      1Q  Median      3Q     Max
-5.2755 -1.1549 -0.4718  0.5539 24.7634

Count model coefficients (poisson with log link):
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.426024   0.028059  50.822  < 2e-16 ***
hospital         0.146820   0.006916  21.229  < 2e-16 ***
healthexcellent -0.279266   0.034575  -8.077 6.63e-16 ***
healthpoor       0.251422   0.020563  12.227  < 2e-16 ***
chronic          0.103175   0.005446  18.945  < 2e-16 ***
sexmale         -0.048790   0.015043  -3.243 0.001181 **
school           0.017484   0.002147   8.142 3.88e-16 ***
insuranceyes     0.071434   0.019867   3.596 0.000324 ***

Zero-inflation model coefficients (binomial with logit link):
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      0.07633    0.16394   0.466  0.64149
hospital        -0.31378    0.10685  -2.937  0.00332 **
healthexcellent  0.01029    0.18319   0.056  0.95519
healthpoor       0.06277    0.18668   0.336  0.73669
chronic         -0.55244    0.05352 -10.323  < 2e-16 ***
sexmale          0.42526    0.10327   4.118 3.82e-05 ***
school          -0.06617    0.01404  -4.711 2.46e-06 ***
insuranceyes    -0.79835    0.11806  -6.762 1.36e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 19
Log-likelihood: -1.205e+04 on 16 Df
```

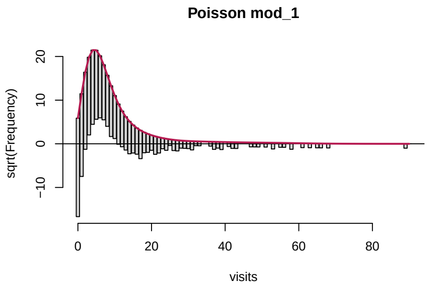
Rootogram for ZIP model

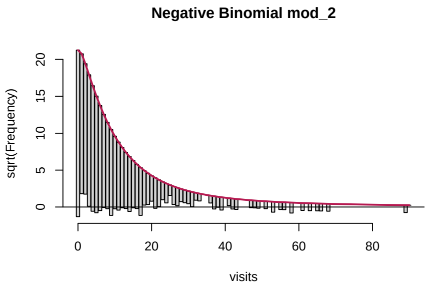
```
countreg::rootogram(mod_3, max = 90,
                    main = "ZIP model Rootogram: mod_3")
```

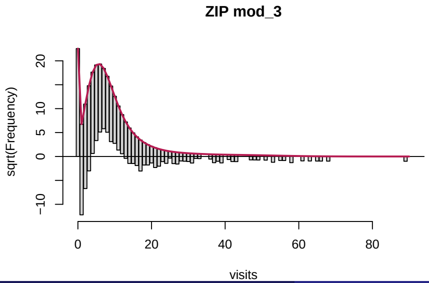


What do you think? Next slide shows all models so far.

First Three Rootograms - Which Looks Best?







Store Training Sample `mod_3` Predictions

We have no `augment` or other `broom` functions available for zero-inflated models, so . . .

```
mod_3_aug <- med_train %>%
    mutate(".fitted" = predict(mod_3, type = "response"),
           ".resid" = resid(mod_3, type = "response"))

mod_3_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
    <int>  <int>   <dbl>  <dbl>
1     355     19    5.31  13.7
2    2661      3    4.21  -1.21
3    2895      0    4.59  -4.59
```

Summarizing Training Sample `mod_3` Fit

`mod_3_aug` now has actual (`visits`) and predicted counts (in `.fitted`) from `mod_3`, just as we set up for the previous two models.

```
mod_3_summary <-
  mets(mod_3_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_3") %>% relocate(model)
mod_3_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_3	rsq	standard	0.108
mod_3	rmse	standard	6.560
mod_3	mae	standard	4.164

Training Sample Results through `mod_3`

```
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary) %>%
  pivot_wider(names_from = model,
              values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3
rsq	standard	0.100	0.078	0.108
rmse	standard	6.594	6.941	6.560
mae	standard	4.189	4.252	4.164

Remember we want a larger R^2 and smaller values of RMSE and MAE.

Comparing models with Vuong's procedure

Vuong's test compares predicted probabilities (for each count) in two non-nested models. How about Poisson vs. ZIP?

```
vuong(mod_1, mod_3)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A    p-value
Raw                   -14.59671 model2 > model1 < 2.22e-16
AIC-corrected         -14.51271 model2 > model1 < 2.22e-16
BIC-corrected         -14.25638 model2 > model1 < 2.22e-16
```

The large negative z-statistic indicates `mod_3` (ZIP) fits detectably better than `mod_1` (Poisson) in our training sample.

Reference: Vuong, QH (1989) Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57:307-333.

`mod_4`: Zero-Inflated Negative Binomial (ZINB) Model

Zero-Inflated Negative Binomial (ZINB) model

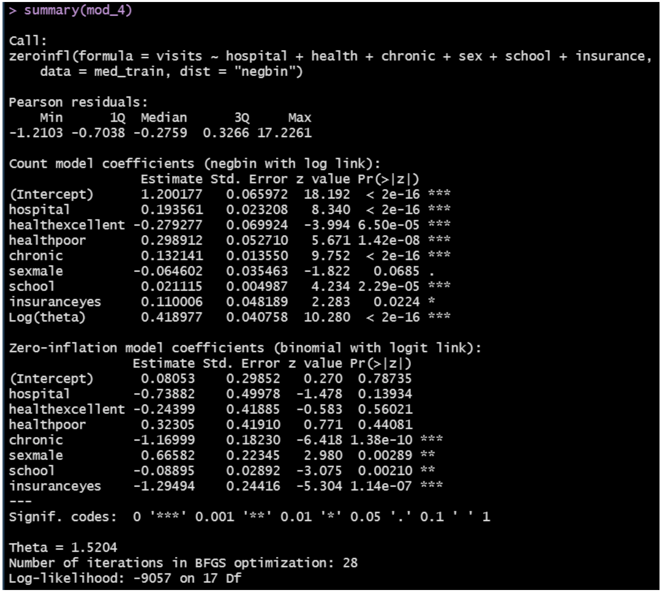
As in the ZIP, we assume there are two processes involved:

- a logistic regression model is used to predict excess zeros
- while a negative binomial model is used to predict the counts

We'll use the `pscl` package again and the `zeroinfl` function.

```
mod_4 <- zeroinfl(visits ~ hospital + health + chronic +
                    sex + school + insurance,
                dist = "negbin", data = med_train)
```

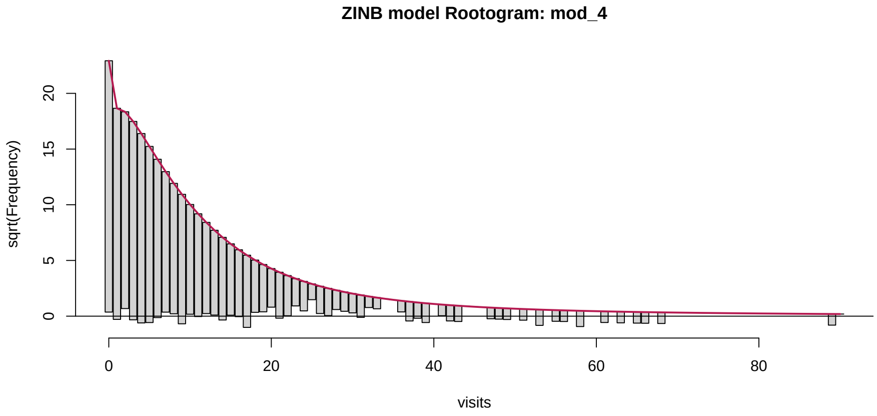
`summary(mod_4)` results on next slide. . .



```
> summary(mod_4)

Call:
zeroinfl(formula = visits ~ hospital + health + chronic + sex + school + insurance,
    data = med_train, dist = "negbin")

Pearson residuals:
    Min      1Q  Median      3Q     Max 
-1.2103 -0.7038 -0.2759  0.3266 17.2261 

Count model coefficients (negbin with log link):
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      1.200177   0.065972  18.192  < 2e-16 ***
hospital         0.193561   0.023208   8.340  < 2e-16 ***
healthexcellent -0.279277   0.069924  -3.994 6.50e-05 ***
healthpoor       0.298912   0.052710   5.671 1.42e-08 ***
chronic          0.132141   0.013550   9.752  < 2e-16 ***
sexmale         -0.064602   0.035463  -1.822   0.0685 .  
school           0.021115   0.004987   4.234 2.29e-05 ***
insuranceyes     0.110006   0.048189   2.283   0.0224 *  
Log(theta)       0.418977   0.040758  10.280  < 2e-16 ***

Zero-inflation model coefficients (binomial with logit link):
                Estimate Std. Error z value Pr(>|z|)    
(Intercept)      0.08053    0.29852   0.270  0.78735    
hospital        -0.73882    0.49978  -1.478  0.13934    
healthexcellent -0.24399    0.41885  -0.583  0.56021    
healthpoor       0.32305    0.41910   0.771  0.44081    
chronic         -1.16999    0.18230  -6.418 1.38e-10 ***
sexmale          0.66582    0.22345   2.980  0.00289 ** 
school          -0.08895    0.02892  -3.075  0.00210 ** 
insuranceyes    -1.29494    0.24416  -5.304 1.14e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Theta = 1.5204 
Number of iterations in BFGS optimization: 28 
Log-likelihood: -9057 on 17 Df
```

Rootogram for ZIP model

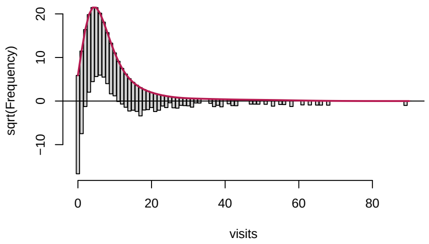
```
countreg::rootogram(mod_4, max = 90,
                    main = "ZINB model Rootogram: mod_4")
```



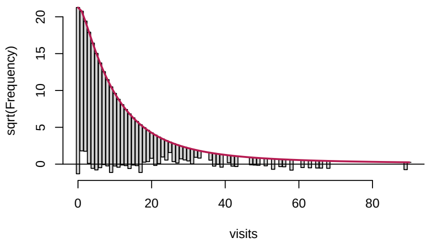
Again, next slide shows all models so far.

First Four Rootograms - Which Looks Best?

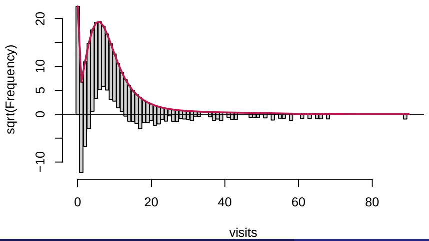
Poisson mod_1



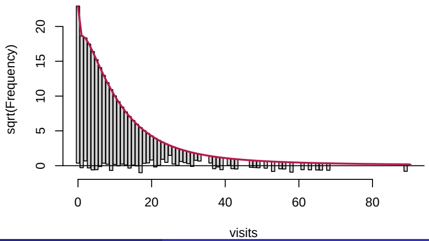
Negative Binomial mod_2



ZIP mod_3



ZINB mod_4



Store Training Sample `mod_4` Predictions

Again, there is no `augment` or other `broom` functions available for zero-inflated models, so . . .

```
mod_4_aug <- med_train %>%
    mutate(".fitted" = predict(mod_4, type = "response"),
           ".resid" = resid(mod_4, type = "response"))

mod_4_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
    <int>  <int>   <dbl>  <dbl>
1     355     19    5.29  13.7
2    2661      3    4.47  -1.47
3    2895      0    4.57  -4.57
```

Summarizing Training Sample `mod_4` Fit

`mod_4_aug` now has actual (`visits`) and predicted counts (in `.fitted`) from `mod_4`.

```
mod_4_summary <-
  mets(mod_4_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_4") %>% relocate(model)
mod_4_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_4	rsq	standard	0.094
mod_4	rmse	standard	6.709
mod_4	mae	standard	4.191

Training Sample Results through `mod_4`

```
bind_rows(mod_1_summary, mod_2_summary,
          mod_3_summary, mod_4_summary) %>%
  pivot_wider(names_from = model,
              values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3	mod_4
rsq	standard	0.100	0.078	0.108	0.094
rmse	standard	6.594	6.941	6.560	6.709
mae	standard	4.189	4.252	4.164	4.191

What do you think?

Comparing models with Vuong's procedure

Vuong's test compares predicted probabilities (for each count) in two non-nested models. How about Negative Binomial vs. ZINB?

```
vuong(mod_4, mod_2)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A    p-value
Raw                    4.808258 model1 > model2 7.6126e-07
AIC-corrected          4.081965 model1 > model2 2.2328e-05
BIC-corrected          1.865724 model1 > model2    0.03104
```

The large positive z-statistics indicate `mod_4` (ZINB) fits detectably better than `mod_2` (Negative Binomial) in our training sample.

Validation in the Test Sample for our Four Models?

Validation: Test Sample Predictions

Predict the `visit` counts for each subject in our test sample.

```r
test_1 <- predict(mod_1, newdata = med_test,
                  type.predict = "response")
test_2 <- predict(mod_2, newdata = med_test,
                  type.predict = "response")
test_3 <- predict(mod_3, newdata = med_test,
                  type.predict = "response")
test_4 <- predict(mod_4, newdata = med_test,
                  type.predict = "response")
```

Create a Tibble with Predictions

Combine the various predictions into a tibble with the original data.

```r
test_res <- bind_cols(med_test,
              pre_m1 = test_1, pre_m2 = test_2,
              pre_m3 = test_3, pre_m4 = test_4)

names(test_res)
```

```
 [1] "subject"   "visits"    "hospital"  "health"
 [5] "chronic"   "sex"       "school"    "insurance"
 [9] "pre_m1"    "pre_m2"    "pre_m3"    "pre_m4"
```

Summarize fit in test sample for each model

```
m1_sum <- mets(test_res, truth = visits, estimate = pre_m1) %>
  mutate(model = "mod_1")
m2_sum <- mets(test_res, truth = visits, estimate = pre_m2) %>
  mutate(model = "mod_2")
m3_sum <- mets(test_res, truth = visits, estimate = pre_m3) %>
  mutate(model = "mod_3")
m4_sum <- mets(test_res, truth = visits, estimate = pre_m4) %>
  mutate(model = "mod_4")

test_sum <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum)
```

Validation Results in Test Sample: Four Models

```
test_sum <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum) %>%
  pivot_wider(names_from = model,
              values_from = .estimate)

test_sum %>%
  select(-.estimator) %>% kable(dig = 3)
```

.metric	mod_1	mod_2	mod_3	mod_4
rsq	0.103	0.108	0.099	0.097
rmse	7.212	7.205	5.907	5.967
mae	4.455	4.450	3.994	4.009

- Which model would you choose based on test sample performance?
- Is there an obvious choice?

Hurdle Models (optional: see Chapter 19 for details)

The Hurdle Model

The hurdle model is a two-part model that specifies one process for zero counts and another process for positive counts. The idea is that positive counts occur once a threshold is crossed, or put another way, a hurdle is cleared. If the hurdle is not cleared, then we have a count of 0.

- The first part of the model is typically a **binary logistic regression** model. This models whether an observation takes a positive count or not.
- The second part of the model is usually a truncated Poisson or Negative Binomial model. Truncated means we're only fitting positive counts, and not zeros.

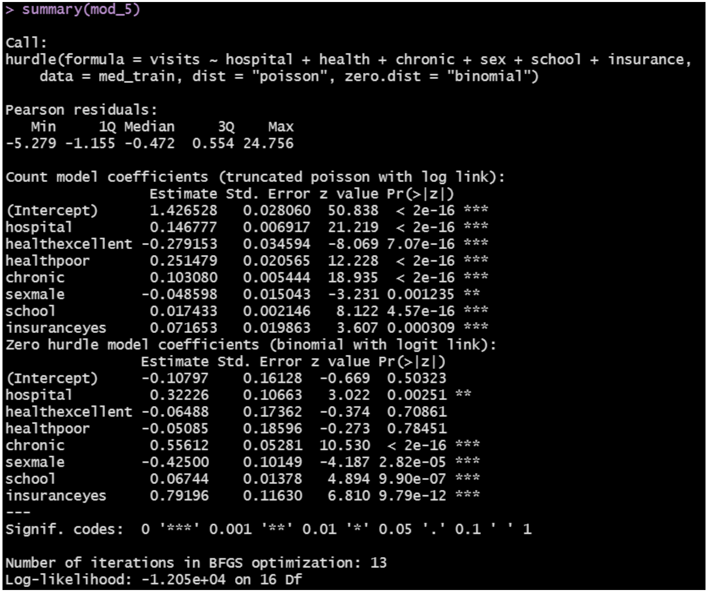
`mod_5`: Poisson-Logistic Hurdle Model

Fitting a Hurdle Model / Poisson-Logistic

In fitting a hurdle model to our medicare training data, the interpretation would be that one process governs whether a patient visits a doctor or not, and another process governs how many visits are made.

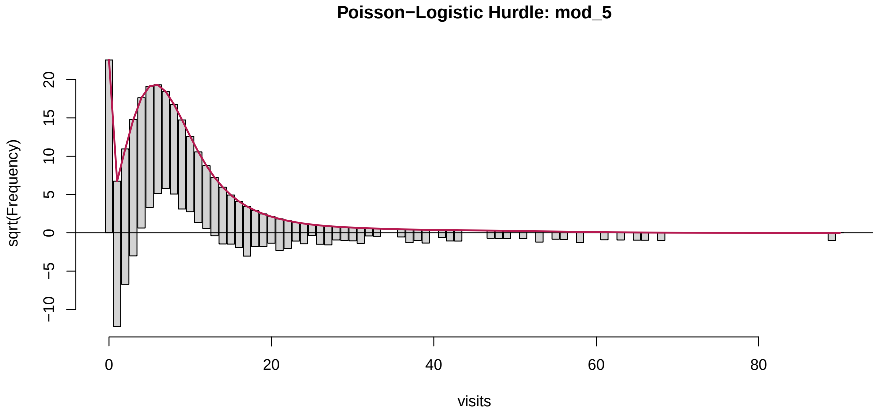
```r
mod_5 <- hurdle(visits ~ hospital + health + chronic +
                    sex + school + insurance,
                dist = "poisson", zero.dist = "binomial",
                data = med_train)
```

`summary(mod_5)` results follow. . .

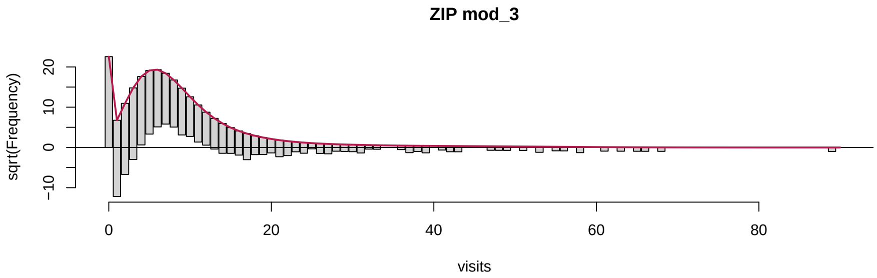

```
> summary(mod_5)

Call:
hurdle(formula = visits ~ hospital + health + chronic + sex + school + insurance,
    data = med_train, dist = "poisson", zero.dist = "binomial")

Pearson residuals:
   Min      1Q Median      3Q     Max
-5.279 -1.155 -0.472  0.554 24.756

Count model coefficients (truncated poisson with log link):
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.426528   0.028060  50.838  < 2e-16 ***
hospital         0.146777   0.006917  21.219  < 2e-16 ***
healthexcellent -0.279153   0.034594  -8.069 7.07e-16 ***
healthpoor       0.251479   0.020565  12.228  < 2e-16 ***
chronic          0.103080   0.005444  18.935  < 2e-16 ***
sexmale         -0.048598   0.015043  -3.231 0.001235 **
school           0.017433   0.002146   8.122 4.57e-16 ***
insuranceyes     0.071653   0.019863   3.607 0.000309 ***
Zero hurdle model coefficients (binomial with logit link):
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.10797    0.16128  -0.669  0.50323
hospital         0.32226    0.10663   3.022  0.00251 **
healthexcellent -0.06488    0.17362  -0.374  0.70861
healthpoor      -0.05085    0.18596  -0.273  0.78451
chronic          0.55612    0.05281  10.530  < 2e-16 ***
sexmale         -0.42500    0.10149  -4.187 2.82e-05 ***
school           0.06744    0.01378   4.894 9.90e-07 ***
insuranceyes     0.79196    0.11630   6.810 9.79e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 13
Log-likelihood: -1.205e+04 on 16 Df
```

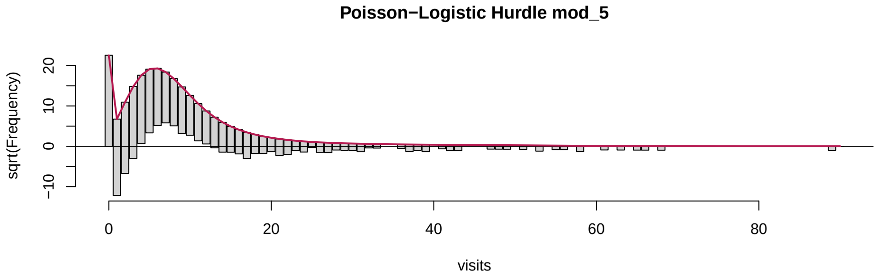
Rootogram for Poisson-Logistic Hurdle model

```
countreg::rootogram(mod_5, max = 90,
                    main = "Poisson-Logistic Hurdle: mod_5")
```



Poisson-Based Rootograms - Which Looks Best?





Store Training Sample `mod_5` Predictions

No augment or other broom functions for hurdle models, so . . .

```
mod_5_aug <- med_train %>%
    mutate(".fitted" = predict(mod_5, type = "response"),
           ".resid" = resid(mod_5, type = "response"))

mod_5_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
    <int>  <int>   <dbl>  <dbl>
1     355     19    5.32  13.7
2    2661      3    4.20  -1.20
3    2895      0    4.59  -4.59
```

Summarizing Training Sample `mod_5` Fit

```
mod_5_summary <-
  mets(mod_5_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_5") %>% relocate(model)
mod_5_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_5	rsq	standard	0.108
mod_5	rmse	standard	6.560
mod_5	mae	standard	4.164

Training Sample Results through `mod_5`

```r
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary,
          mod_4_summary, mod_5_summary) %>%
  pivot_wider(names_from = model,
              values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3	mod_4	mod_5
rsq	standard	0.100	0.078	0.108	0.094	0.108
rmse	standard	6.594	6.941	6.560	6.709	6.560
mae	standard	4.189	4.252	4.164	4.191	4.164

What do you think?

Are ZIP and Poisson-Logistic Hurdle the Same?

```r
temp_check <- tibble(
  subject = mod_3_aug$subject,
  visits = mod_3_aug$visits,
  pred_zip = mod_3_aug$.fitted,
  pred_hur = mod_5_aug$.fitted,
  diff = pred_hur - pred_zip)

mosaic::favstats(~ diff, data = temp_check)
```

```
        min            Q1       median            Q3
 -0.02412392 -0.0004090414 0.0003037249 0.0009281458
        max        mean          sd    n missing
 0.03270803 0.000330143 0.003049981 3304       0
```

Vuong test: Comparing mod_3 and mod_5

```
vuong(mod_3, mod_5)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A  p-value
Raw                    1.913251 model1 > model2 0.027858
AIC-corrected          1.913251 model1 > model2 0.027858
BIC-corrected          1.913251 model1 > model2 0.027858
```

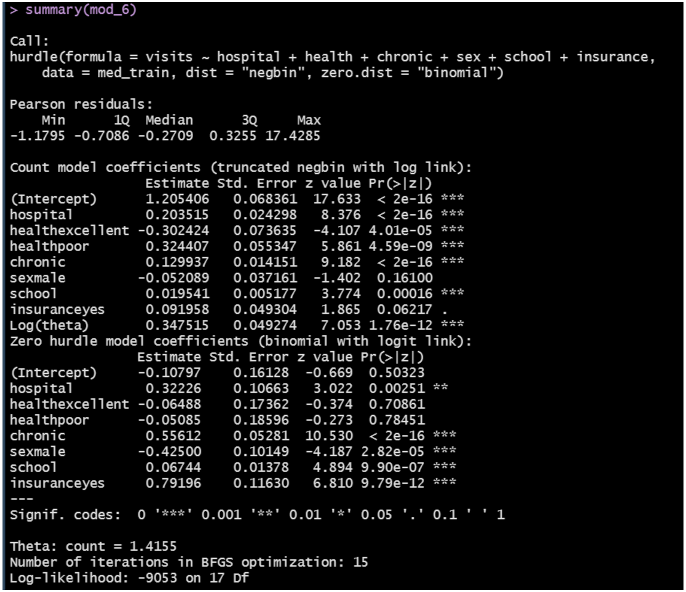
There's some evidence `mod_3` (ZIP) fits a bit better than `mod_5` (Hurdle) in our training sample, though the p value (barely) exceeds 0.05.

`mod_6`: Negative Binomial-Logistic Hurdle Model

Fitting a Hurdle Model / NB-Logistic

```
mod_6 <- hurdle(visits ~ hospital + health + chronic +
                  sex + school + insurance,
              dist = "negbin", zero.dist = "binomial",
              data = med_train)
```

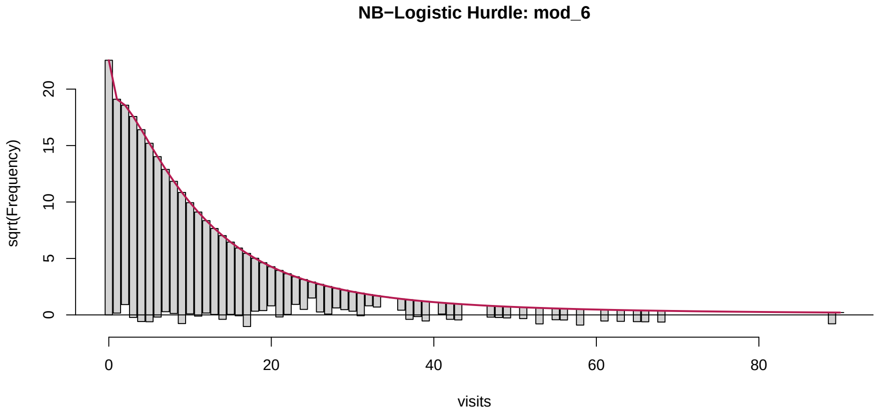
`summary(mod_6)` results follow. . .



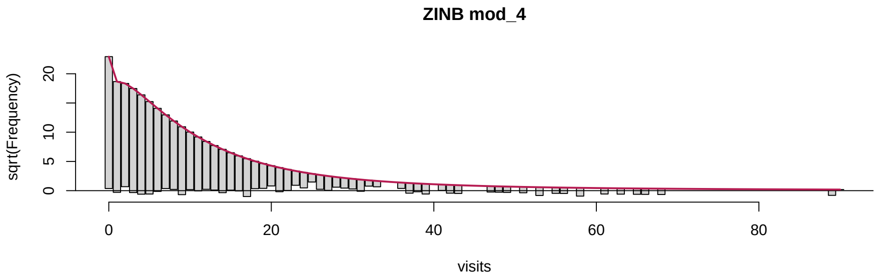
```
> summary(mod_6)

Call:
hurdle(formula = visits ~ hospital + health + chronic + sex + school + insurance,
    data = med_train, dist = "negbin", zero.dist = "binomial")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-1.1795 -0.7086 -0.2709  0.3255 17.4285

Count model coefficients (truncated negbin with log link):
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.205406   0.068361  17.633  < 2e-16 ***
hospital         0.203515   0.024298   8.376  < 2e-16 ***
healthexcellent -0.302424   0.073635  -4.107 4.01e-05 ***
healthpoor       0.324407   0.055347   5.861 4.59e-09 ***
chronic          0.129937   0.014151   9.182  < 2e-16 ***
sexmale         -0.052089   0.037161  -1.402  0.16100
school           0.019541   0.005177   3.774  0.00016 ***
insuranceyes     0.091958   0.049304   1.865  0.06217 .
Log(theta)       0.347515   0.049274   7.053 1.76e-12 ***
Zero hurdle model coefficients (binomial with logit link):
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.10797    0.16128  -0.669  0.50323
hospital         0.32226    0.10663   3.022  0.00251 **
healthexcellent -0.06488    0.17362  -0.374  0.70861
healthpoor      -0.05085    0.18596  -0.273  0.78451
chronic          0.55612    0.05281  10.530  < 2e-16 ***
sexmale         -0.42500    0.10149  -4.187 2.82e-05 ***
school           0.06744    0.01378   4.894 9.90e-07 ***
insuranceyes     0.79196    0.11630   6.810 9.79e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 1.4155
Number of iterations in BFGS optimization: 15
Log-likelihood: -9053 on 17 Df
```

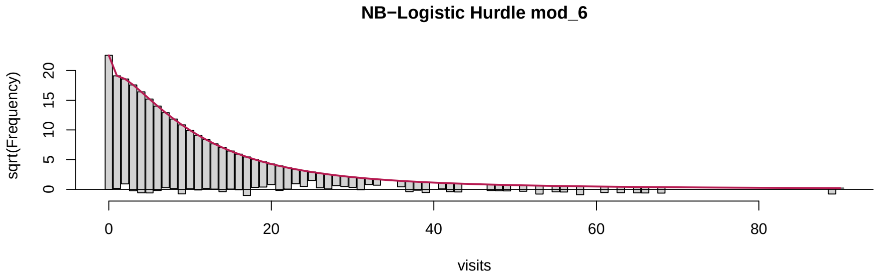
Rootogram for NB-Logistic Hurdle model

```
countreg::rootogram(mod_6, max = 90,
                    main = "NB-Logistic Hurdle: mod_6")
```



NB-Based Rootograms - Which Looks Best?





Store Training Sample mod_6 Predictions

```r
mod_6_aug <- med_train %>%
    mutate(".fitted" = predict(mod_6, type = "response"),
           ".resid" = resid(mod_6, type = "response"))

mod_6_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
    <int>  <int>   <dbl>  <dbl>
1     355     19    5.35  13.6
2    2661      3    4.16  -1.16
3    2895      0    4.49  -4.49
```

Summarizing Training Sample mod_6 Fit

```
mod_6_summary <-
  mets(mod_6_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_6") %>% relocate(model)
mod_6_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_6	rsq	standard	0.089
mod_6	rmse	standard	6.772
mod_6	mae	standard	4.209

Training Sample Results through `mod_6`

```
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary,
          mod_4_summary, mod_5_summary, mod_6_summary) %>%
  pivot_wider(names_from = model, values_from = .estimate) %>
  select(-.estimator) %>% kable(dig = 3)
```

.metric	mod_1	mod_2	mod_3	mod_4	mod_5	mod_6
rsq	0.100	0.078	0.108	0.094	0.108	0.089
rmse	6.594	6.941	6.560	6.709	6.560	6.772
mae	4.189	4.252	4.164	4.191	4.164	4.209

Vuong test: Comparing `mod_4` and `mod_6`

```
vuong(mod_4, mod_6)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
-------------------------------------------------------------
              Vuong z-statistic             H_A p-value
Raw                  0.02994935 model1 > model2 0.48805
AIC-corrected        0.02994935 model1 > model2 0.48805
BIC-corrected        0.02994935 model1 > model2 0.48805
```

There's some evidence `mod_4` (ZINB) fits better than `mod_6` (NB Hurdle) in our training sample, but not to a statistically detectable degree, based on the large *p* value.

Validation including Hurdle Models

Validation: Test Sample Predictions

Predict the `visit` counts for each subject in our test sample.

```
test_5 <- predict(mod_5, newdata = med_test,
                  type.predict = "response")
test_6 <- predict(mod_6, newdata = med_test,
                  type.predict = "response")
```

Create a Tibble with Predictions

Combine the various predictions into a tibble with the original data.

```r
test_res6 <- bind_cols(med_test,
                pre_m1 = test_1, pre_m2 = test_2,
                pre_m3 = test_3, pre_m4 = test_4,
                pre_m5 = test_5, pre_m6 = test_6)

names(test_res6)
```

```
 [1] "subject"   "visits"    "hospital"  "health"
 [5] "chronic"   "sex"       "school"    "insurance"
 [9] "pre_m1"    "pre_m2"    "pre_m3"    "pre_m4"
[13] "pre_m5"    "pre_m6"
```

Summarize fit in test sample for each model

```
m1_sum <- mets(test_res6, truth = visits, estimate = pre_m1) %
  mutate(model = "mod_1")
m2_sum <- mets(test_res6, truth = visits, estimate = pre_m2) %
  mutate(model = "mod_2")
m3_sum <- mets(test_res6, truth = visits, estimate = pre_m3) %
  mutate(model = "mod_3")
m4_sum <- mets(test_res6, truth = visits, estimate = pre_m4) %
  mutate(model = "mod_4")
m5_sum <- mets(test_res6, truth = visits, estimate = pre_m5) %
  mutate(model = "mod_5")
m6_sum <- mets(test_res6, truth = visits, estimate = pre_m6) %
  mutate(model = "mod_6")

test_sum6 <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum,
                       m5_sum, m6_sum)
```

Validation Results in Test Sample: All Models

```
test_sum6 <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum,
                       m5_sum, m6_sum) %>%
  pivot_wider(names_from = model,
              values_from = .estimate)

test_sum6 %>%
  select(-.estimator) %>% kable(dig = 4)
```

.metric	mod_1	mod_2	mod_3	mod_4	mod_5	mod_6
rsq	0.1032	0.1082	0.0993	0.0970	0.0992	0.0937
rmse	7.2122	7.2051	5.9069	5.9674	5.9072	6.0009
mae	4.4550	4.4496	3.9943	4.0095	3.9946	4.0265

- Now which model would you choose based on test sample performance?