432 Class 17

https://thomaselove.github.io/432-2025

2025-03-18

## Today’s Agenda

**Regression Models for Count Outcomes**

* Modeling approaches illustrated in these slides
  + Poisson Regression & Zero-Inflated Poisson (ZIP)
  + Negative Binomial Regression & Zero-Inflated Negative Binomial (ZINB)

Chapters 24-26 of the Course Notes describe this material, as well as hurdle models (next class) and tobit regression, and some additional issues with certain types of count models.

## countreg and topmodels packages

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 and topmodels packages, which are currently available only on R-Forge. To install, type:

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

into the R Console within R Studio.

## Today’s R Setup

knitr::opts\_chunk$set(comment=NA)  
  
library(janitor); library(gt); library(broom)   
library(mosaic); library(Hmisc); library(patchwork)  
library(rsample); library(yardstick); library(here)  
library(conflicted) ## resolve conflicts  
library(topmodels) ## for rootograms  
library(MASS) ## for glm.nb to fit NB models  
library(pscl) ## for zero-inflated and hurdle fits  
library(lmtest) ## for Vuong test  
library(easystats)  
library(tidyverse)  
  
conflicts\_prefer(dplyr::select(), dplyr::filter(), base::max(),   
 base::sum(), yardstick::rmse(), yardstick::mae(),  
 pscl::zeroinfl())  
  
theme\_set(theme\_bw())

# An Overview

## GLMs for Count Outcomes

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

Count data 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
* days in the past 30 where your mental health was poor

We’ll use the Poisson and Negative Binomial probability distributions.

## 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.
  + 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.

## Poisson regression

* 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.

We will use glm to fit Poisson models, by using family = "Poisson".

## Dealing with Overdispersion

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 for now.

Instead, we’ll look at other ways (especially zero-inflation and the negative binomial models) to address overdispersion.

## Negative Binomial Regression

* 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 the zero counts, 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’ll use zeroinfl() from pscl to fit ZIP and ZINB regressions.

## Hurdle models

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 NB model where no zeros are predicted)

We use hurdle() from pscl to fit these.

## 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 and topmodels packages 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.

## Comparing Models

1. 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

Source: NMES1988 data in R’s AER package, cleaned up to medicare.csv.

Essentially the same data are used in from the University of Virginia on hurdle models.

Data are a cross-section 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.

## Ingesting medicare data

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.

medicare <- read\_csv(here("c17/data/medicare.csv"),   
 show\_col\_types = FALSE) |>   
 mutate(across(where(is\_character), as\_factor),  
 subject = as.character(subject))

## The medicare code book

| Variable | Description |
| --- | --- |
| subject | subject number (code) |
| visits | outcome: # of physician office visits |
| hospital | # of hospital stays |
| health | self-rated health (poor, average, excellent) |
| chronic | # of chronic conditions |
| sex | male or female |
| school | years of education |
| insurance | subject (also) has private insurance? (yes/no) |

### Today’s Goal

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

## The medicare tibble

medicare |> select(-subject)

# A tibble: 4,406 × 7  
 visits hospital health chronic sex school insurance  
 <dbl> <dbl> <fct> <dbl> <fct> <dbl> <fct>   
 1 5 1 average 2 male 6 yes   
 2 1 0 average 2 female 10 yes   
 3 13 3 poor 4 female 10 no   
 4 16 1 poor 2 male 3 yes   
 5 3 0 average 2 female 6 yes   
 6 17 0 poor 5 female 7 no   
 7 9 0 average 0 female 8 yes   
 8 3 0 average 0 female 8 yes   
 9 1 0 average 0 female 8 yes   
10 0 0 average 0 female 8 yes   
# ℹ 4,396 more rows

## Quick Summary of medicare

medicare |> select(-subject) |> summary()

visits hospital health chronic   
 Min. : 0.000 Min. :0.000 average :3509 Min. :0.000   
 1st Qu.: 1.000 1st Qu.:0.000 poor : 554 1st Qu.:1.000   
 Median : 4.000 Median :0.000 excellent: 343 Median :1.000   
 Mean : 5.774 Mean :0.296 Mean :1.542   
 3rd Qu.: 8.000 3rd Qu.:0.000 3rd Qu.:2.000   
 Max. :89.000 Max. :8.000 Max. :8.000   
 sex school insurance   
 male :1778 Min. : 0.00 yes:3421   
 female:2628 1st Qu.: 8.00 no : 985   
 Median :11.00   
 Mean :10.29   
 3rd Qu.:12.00   
 Max. :18.00

### Adjust order of insurance

medicare <- medicare |>  
 mutate(insurance = fct\_relevel(insurance, "no", "yes"))

I want No first, then Yes, when building models.

## Our outcome, visits

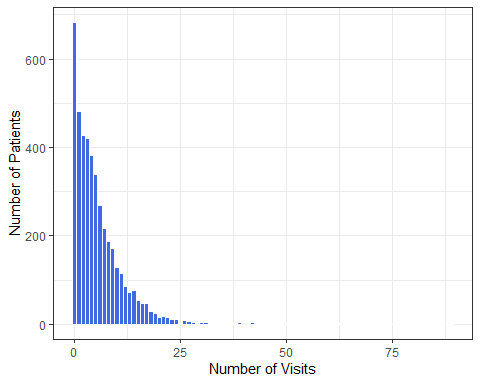
favstats(~ visits, data = medicare)

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

describe(medicare$visits) # from Hmisc

medicare$visits   
 n missing distinct Info Mean pMedian Gmd .05   
 4406 0 60 0.992 5.774 4.5 6.227 0   
 .10 .25 .50 .75 .90 .95   
 0 1 4 8 13 17   
  
lowest : 0 1 2 3 4, highest: 63 65 66 68 89

ggplot(medicare, aes(x = visits)) +  
 geom\_histogram(binwidth = 1, fill = "royalblue",   
 col = "white") +  
 labs(y = "Number of Patients", x = "Number of Visits")



## Partitioning the Data

Creating Training and Testing Samples with rsample functions…

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.

dim(med\_train); dim(med\_test)

[1] 3304 8

[1] 1102 8

## Reiterating the Goal

Predict visits using some combination of these 6 predictors…

| Predictor | Description |
| --- | --- |
| hospital | # of hospital stays |
| health | self-rated health (poor, average, excellent) |
| chronic | # of chronic conditions |
| sex | male or female |
| school | years of education |
| insurance | subject (also) has private insurance? (yes/no) |

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

# 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.

## Complete mod\_1 Summary

summary(mod\_1)

Call:  
glm(formula = visits ~ hospital + health + chronic + sex + school +   
 insurance, family = "poisson", data = med\_train)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.886576 0.028813 30.770 < 2e-16 \*\*\*  
hospital 0.163555 0.006710 24.374 < 2e-16 \*\*\*  
healthpoor 0.309610 0.020244 15.294 < 2e-16 \*\*\*  
healthexcellent -0.358758 0.034875 -10.287 < 2e-16 \*\*\*  
chronic 0.137349 0.005266 26.082 < 2e-16 \*\*\*  
sexfemale 0.098325 0.014805 6.641 3.11e-11 \*\*\*  
school 0.031258 0.002111 14.808 < 2e-16 \*\*\*  
insuranceyes 0.200249 0.019484 10.278 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for poisson family taken to be 1)  
  
 Null deviance: 20618 on 3303 degrees of freedom  
Residual deviance: 17598 on 3296 degrees of freedom  
AIC: 27232  
  
Number of Fisher Scoring iterations: 5

## mod\_1 (Poisson) model coefficients

tidy(mod\_1) |> gt() |> fmt\_number(decimals = 3)

| term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- |
| (Intercept) | 0.887 | 0.029 | 30.770 | 0.000 |
| hospital | 0.164 | 0.007 | 24.374 | 0.000 |
| healthpoor | 0.310 | 0.020 | 15.294 | 0.000 |
| healthexcellent | -0.359 | 0.035 | -10.287 | 0.000 |
| chronic | 0.137 | 0.005 | 26.082 | 0.000 |
| sexfemale | 0.098 | 0.015 | 6.641 | 0.000 |
| school | 0.031 | 0.002 | 14.808 | 0.000 |
| insuranceyes | 0.200 | 0.019 | 10.278 | 0.000 |

Harry and Larry have the same values for all other predictors but only Harry has private insurance. mod\_1 estimates Harry’s log(visits) to be 0.2 larger than Larry’s log(visits).

## Dealing with log transformations

OK, you ran a regression/fit a linear model and some of your variables are log-transformed.

* We fit a linear model to predict the log of an outcome. How might we most effectively interpret the coefficients of that model?
* How does our thinking change if we only take the log of a predictor?
* How about if we log both the outcome and the predictor?

Source [is here](https://library.virginia.edu/data/articles/interpreting-log-transformations-in-a-linear-model) for further reference (also see today’s README.)

## Only the outcome is log-transformed

Exponentiate the coefficient. This gives the multiplicative factor for every one-unit increase in the independent variable.

* Example: the coefficient is 0.198. exp(0.198) = 1.218962. For every one-unit increase in the independent variable, our dependent variable increases by a factor of about 1.22, or 22%. Recall that multiplying a number by 1.22 is the same as increasing the number by 22%.
* Likewise, multiplying a number by, say 0.84, is the same as decreasing the number by 1 – 0.84 = 0.16, or 16%.

from [this source](https://library.virginia.edu/data/articles/interpreting-log-transformations-in-a-linear-model)

## Only a predictor is log-transformed

* Divide the coefficient by 100. This tells us that a 1% increase in the independent variable increases (or decreases) the dependent variable by (coefficient/100) units.
* Example: the coefficient is 0.198. 0.198/100 = 0.00198. For every 1% increase in the independent variable, our dependent variable increases by about 0.002.
* For x percent increase, multiply the coefficient by log(1.x).
* Example: For every 10% increase in the independent variable, our dependent variable increases by about 0.198 \* log(1.10) = 0.02

from [this source](https://library.virginia.edu/data/articles/interpreting-log-transformations-in-a-linear-model)

## Both y and x are log-transformed

Interpret the coefficient as the percent increase in the dependent variable for every 1% increase in the independent variable.

* Example: the coefficient is 0.198. For every 1% increase in the independent variable, our dependent variable increases by about 0.20%.
* For x percent increase, calculate 1.x to the power of the coefficient, subtract 1, and multiply by 100.
* Example: For every 20% increase in the independent variable, our dependent variable increases by about (1.20 0.198 - 1) \* 100 = 3.7 percent.

from [this source](https://library.virginia.edu/data/articles/interpreting-log-transformations-in-a-linear-model)

## mod\_1 parameters

* Note the exponentiation here. IRR = incidence rate ratio

model\_parameters(mod\_1, exponentiate = TRUE, ci = 0.90)

Parameter | IRR | SE | 90% CI | z | p  
---------------------------------------------------------------------  
(Intercept) | 2.43 | 0.07 | [2.31, 2.54] | 30.77 | < .001  
hospital | 1.18 | 7.90e-03 | [1.16, 1.19] | 24.37 | < .001  
health [poor] | 1.36 | 0.03 | [1.32, 1.41] | 15.29 | < .001  
health [excellent] | 0.70 | 0.02 | [0.66, 0.74] | -10.29 | < .001  
chronic | 1.15 | 6.04e-03 | [1.14, 1.16] | 26.08 | < .001  
sex [female] | 1.10 | 0.02 | [1.08, 1.13] | 6.64 | < .001  
school | 1.03 | 2.18e-03 | [1.03, 1.04] | 14.81 | < .001  
insurance [yes] | 1.22 | 0.02 | [1.18, 1.26] | 10.28 | < .001

Uncertainty intervals (profile-likelihood) and p-values (two-tailed)  
 computed using a Wald z-distribution approximation.

* A change from no to yes in insurance (holding other predictors constant) is associated with a 22% increase in our outcome, visits.

## mod\_1 performance summaries

model\_performance(mod\_1)

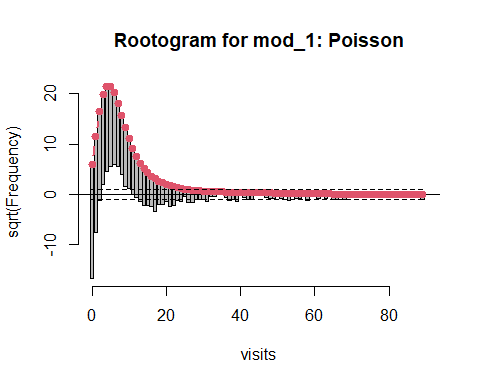
# Indices of model performance  
  
AIC | AICc | BIC | Nagelkerke's R2 | RMSE | Sigma | Score\_log | Score\_spherical  
-------------------------------------------------------------------------------------------------  
27231.869 | 27231.912 | 27280.692 | 0.600 | 6.594 | 1.000 | -4.119 | 0.013

glance(mod\_1)

# A tibble: 1 × 8  
 null.deviance df.null logLik AIC BIC deviance df.residual nobs  
 <dbl> <int> <dbl> <dbl> <dbl> <dbl> <int> <int>  
1 20618. 3303 -13608. 27232. 27281. 17598. 3296 3304

## Visualize fit: (Hanging) Rootogram

plot(rootogram(mod\_1, plot = FALSE), xlim = c(0, 90),   
 main = "Rootogram for mod\_1: Poisson")



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

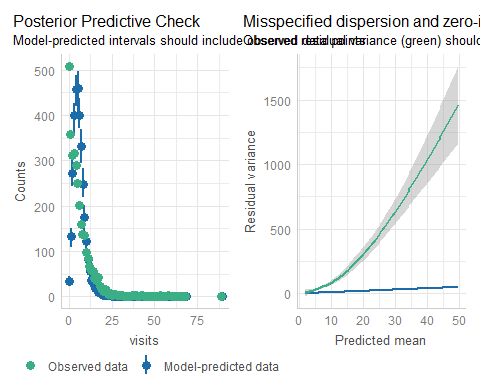
## Interpreting the Rootogram

* 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.
* <https://arxiv.org/pdf/1605.01311> has more on rootograms.
* 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.

## Checking mod\_1 (plots 1-2)

check\_model(mod\_1, check = c("pp\_check", "overdispersion"))

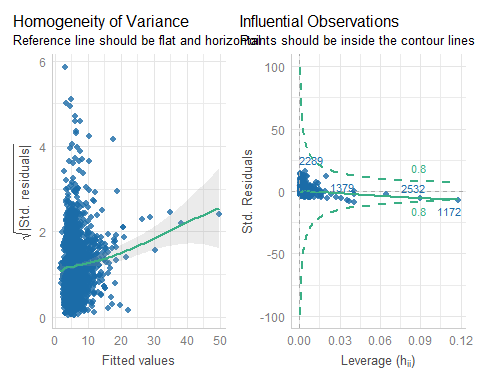
Cannot simulate residuals for models of class `glm`. Please try  
 `check\_model(..., residual\_type = "normal")` instead.



## Checking mod\_1 (plots 3-4)

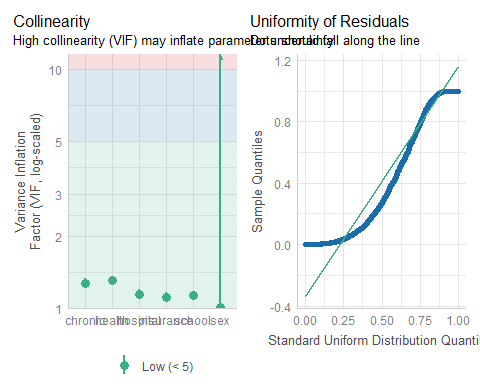
check\_model(mod\_1, check = c("homogeneity", "outliers"))

Cannot simulate residuals for models of class `glm`. Please try  
 `check\_model(..., residual\_type = "normal")` instead.



## Checking mod\_1 (plots 5-6)

check\_model(mod\_1, check = c("vif", "qq"))



## Store 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 × 3  
 subject visits .fitted  
 <chr> <dbl> <dbl>  
1 355 19 5.02  
2 2661 3 4.21  
3 2895 0 4.65

## 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 |> gt() |> fmt\_number(decimals = 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 (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 is…

summary(mod\_2)$theta

[1] 1.21109

The Poisson model is essentially the negative binomial model assuming a known .

## Complete mod\_2 summary

summary(mod\_2)

Call:  
MASS::glm.nb(formula = visits ~ hospital + health + chronic +   
 sex + school + insurance, data = med\_train, init.theta = 1.211089878,   
 link = log)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.765746 0.065602 11.673 < 2e-16 \*\*\*  
hospital 0.224205 0.022745 9.857 < 2e-16 \*\*\*  
healthpoor 0.360067 0.055608 6.475 9.47e-11 \*\*\*  
healthexcellent -0.335591 0.070353 -4.770 1.84e-06 \*\*\*  
chronic 0.169070 0.013887 12.174 < 2e-16 \*\*\*  
sexfemale 0.109443 0.035920 3.047 0.00231 \*\*   
school 0.030763 0.005037 6.107 1.02e-09 \*\*\*  
insuranceyes 0.237080 0.045780 5.179 2.23e-07 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for Negative Binomial(1.2111) family taken to be 1)  
  
 Null deviance: 4341.7 on 3303 degrees of freedom  
Residual deviance: 3783.1 on 3296 degrees of freedom  
AIC: 18328  
  
Number of Fisher Scoring iterations: 1  
  
 Theta: 1.2111   
 Std. Err.: 0.0388   
  
 2 x log-likelihood: -18309.8280

## mod\_2 (NB) coefficients

tidy(mod\_2) |> gt() |> fmt\_number(decimals = 3)

| term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- |
| (Intercept) | 0.766 | 0.066 | 11.673 | 0.000 |
| hospital | 0.224 | 0.023 | 9.857 | 0.000 |
| healthpoor | 0.360 | 0.056 | 6.475 | 0.000 |
| healthexcellent | -0.336 | 0.070 | -4.770 | 0.000 |
| chronic | 0.169 | 0.014 | 12.174 | 0.000 |
| sexfemale | 0.109 | 0.036 | 3.047 | 0.002 |
| school | 0.031 | 0.005 | 6.107 | 0.000 |
| insuranceyes | 0.237 | 0.046 | 5.179 | 0.000 |

## mod\_2 parameters

model\_parameters(mod\_2, exponentiate = TRUE, ci = 0.90)

Parameter | IRR | SE | 90% CI | z | p  
--------------------------------------------------------------------  
(Intercept) | 2.15 | 0.14 | [1.93, 2.40] | 11.67 | < .001  
hospital | 1.25 | 0.03 | [1.20, 1.30] | 9.86 | < .001  
health [poor] | 1.43 | 0.08 | [1.31, 1.57] | 6.48 | < .001  
health [excellent] | 0.71 | 0.05 | [0.64, 0.80] | -4.77 | < .001  
chronic | 1.18 | 0.02 | [1.16, 1.21] | 12.17 | < .001  
sex [female] | 1.12 | 0.04 | [1.05, 1.18] | 3.05 | 0.002   
school | 1.03 | 5.19e-03 | [1.02, 1.04] | 6.11 | < .001  
insurance [yes] | 1.27 | 0.06 | [1.17, 1.37] | 5.18 | < .001

Uncertainty intervals (profile-likelihood) and p-values (two-tailed)  
 computed using a Wald z-distribution approximation.

## mod\_2 performance summaries

model\_performance(mod\_2)

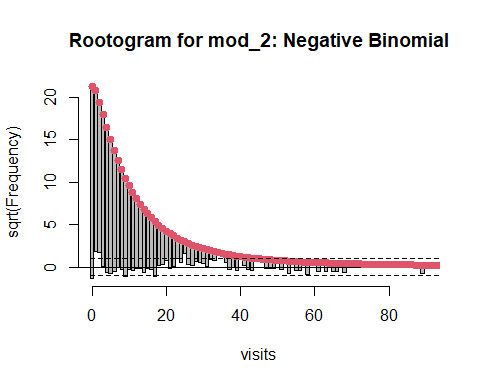
# Indices of model performance  
  
AIC | AICc | BIC | Nagelkerke's R2 | RMSE | Sigma | Score\_log | Score\_spherical  
-------------------------------------------------------------------------------------------------  
18327.828 | 18327.883 | 18382.754 | 0.213 | 6.941 | 1.000 | -2.938 | 0.015

glance(mod\_2)

# A tibble: 1 × 8  
 null.deviance df.null logLik AIC BIC deviance df.residual nobs  
 <dbl> <int> <logLik> <dbl> <dbl> <dbl> <int> <int>  
1 4342. 3303 -9154.914 18328. 18383. 3783. 3296 3304

## Rootogram for NB Model

plot(rootogram(mod\_2, plot = FALSE), xlim = c(0, 90),   
 main = "Rootogram for mod\_2: Negative Binomial")

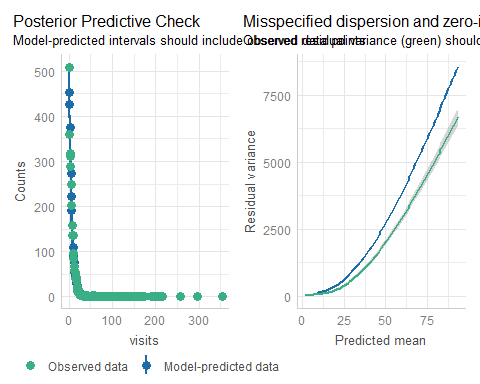


Does this look better than the Poisson rootogram?

## Checking mod\_2 (plots 1-2)

check\_model(mod\_2, check = c("pp\_check", "overdispersion"))

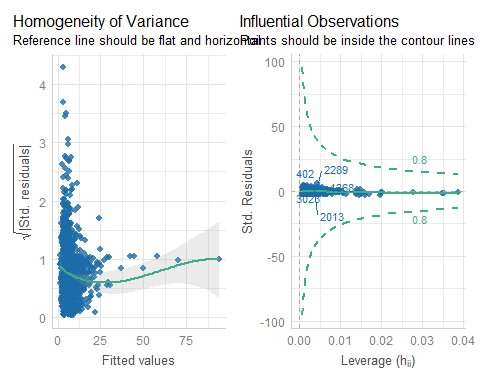
Cannot simulate residuals for models of class `negbin`. Please try  
 `check\_model(..., residual\_type = "normal")` instead.



## Checking mod\_2 (plots 3-4)

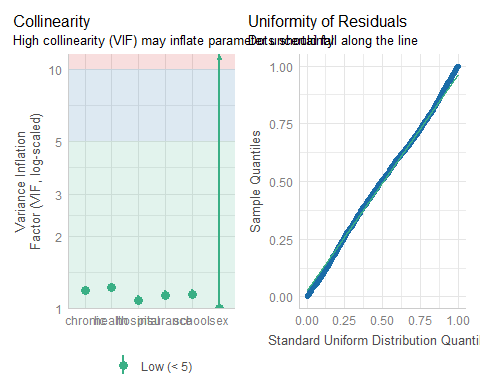
check\_model(mod\_2, check = c("homogeneity", "outliers"))

Cannot simulate residuals for models of class `negbin`. Please try  
 `check\_model(..., residual\_type = "normal")` instead.



## Checking mod\_2 (plots 5-6)

check\_model(mod\_2, check = c("vif", "qq"))



## Store 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 × 3  
 subject visits .fitted  
 <chr> <dbl> <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 silence it, as I have here.

## Training Fit for mod\_2

mod\_2\_aug has actual (visits) and predicted counts (in .fitted.)

mod\_2\_summary <-   
 mets(mod\_2\_aug, truth = visits, estimate = .fitted) |>  
 mutate(model = "mod\_2") |> relocate(model)  
mod\_2\_summary |> gt() |> fmt\_number(decimals = 3)

| model | .metric | .estimator | .estimate |
| --- | --- | --- | --- |
| mod\_2 | rsq | standard | 0.078 |
| mod\_2 | rmse | standard | 6.941 |
| mod\_2 | mae | standard | 4.252 |

## Training Sample So Far

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

| Model | Rootogram impressions |
| --- | --- |
| mod\_1 (P) | Many problems. Data appear overdispersed. |
| mod\_2 (NB) | Still not enough zeros; some big predictions. |

## Training Sample Summaries

bind\_rows(mod\_1\_summary, mod\_2\_summary) |>   
 pivot\_wider(names\_from = model,   
 values\_from = .estimate) |>   
 gt() |> fmt\_number(decimals = 3) |>   
 tab\_options(table.font.size = 20)

| .metric | .estimator | mod\_1 | mod\_2 |
| --- | --- | --- | --- |
| rsq | standard | 0.100 | 0.078 |
| rmse | standard | 6.594 | 6.941 |
| mae | standard | 4.189 | 4.252 |

# 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)

Call:  
pscl::zeroinfl(formula = visits ~ hospital + health + chronic + sex +   
 school + insurance, data = med\_train)  
  
Pearson residuals:  
 Min 1Q Median 3Q Max   
-5.4401 -1.1618 -0.4769 0.5699 24.3630   
  
Count model coefficients (poisson with log link):  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 1.298380 0.029608 43.853 < 2e-16 \*\*\*  
hospital 0.160449 0.006780 23.664 < 2e-16 \*\*\*  
healthpoor 0.302326 0.020022 15.099 < 2e-16 \*\*\*  
healthexcellent -0.281826 0.035775 -7.878 3.33e-15 \*\*\*  
chronic 0.097090 0.005420 17.913 < 2e-16 \*\*\*  
sexfemale 0.056219 0.014934 3.765 0.000167 \*\*\*  
school 0.023367 0.002139 10.924 < 2e-16 \*\*\*  
insuranceyes 0.093169 0.019794 4.707 2.52e-06 \*\*\*  
  
Zero-inflation model coefficients (binomial with logit link):  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.25623 0.16807 1.525 0.127379   
hospital -0.29538 0.10252 -2.881 0.003962 \*\*   
healthpoor -0.09642 0.19067 -0.506 0.613089   
healthexcellent 0.32851 0.16898 1.944 0.051891 .   
chronic -0.49292 0.05183 -9.510 < 2e-16 \*\*\*  
sexfemale -0.36437 0.10295 -3.539 0.000401 \*\*\*  
school -0.06165 0.01410 -4.373 1.22e-05 \*\*\*  
insuranceyes -0.66566 0.11996 -5.549 2.87e-08 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
  
Number of iterations in BFGS optimization: 22   
Log-likelihood: -1.222e+04 on 16 Df

## mod\_3 parameters

model\_parameters(mod\_3, ci = 0.90)

# Fixed Effects  
  
Parameter | Log-Mean | SE | 90% CI | z | p  
--------------------------------------------------------------------------  
(Intercept) | 1.30 | 0.03 | [ 1.25, 1.35] | 43.85 | < .001  
hospital | 0.16 | 6.78e-03 | [ 0.15, 0.17] | 23.66 | < .001  
health [poor] | 0.30 | 0.02 | [ 0.27, 0.34] | 15.10 | < .001  
health [excellent] | -0.28 | 0.04 | [-0.34, -0.22] | -7.88 | < .001  
chronic | 0.10 | 5.42e-03 | [ 0.09, 0.11] | 17.91 | < .001  
sex [female] | 0.06 | 0.01 | [ 0.03, 0.08] | 3.76 | < .001  
school | 0.02 | 2.14e-03 | [ 0.02, 0.03] | 10.92 | < .001  
insurance [yes] | 0.09 | 0.02 | [ 0.06, 0.13] | 4.71 | < .001  
  
# Zero-Inflation  
  
Parameter | Log-Odds | SE | 90% CI | z | p  
----------------------------------------------------------------------  
(Intercept) | 0.26 | 0.17 | [-0.02, 0.53] | 1.52 | 0.127   
hospital | -0.30 | 0.10 | [-0.46, -0.13] | -2.88 | 0.004   
health [poor] | -0.10 | 0.19 | [-0.41, 0.22] | -0.51 | 0.613   
health [excellent] | 0.33 | 0.17 | [ 0.05, 0.61] | 1.94 | 0.052   
chronic | -0.49 | 0.05 | [-0.58, -0.41] | -9.51 | < .001  
sex [female] | -0.36 | 0.10 | [-0.53, -0.20] | -3.54 | < .001  
school | -0.06 | 0.01 | [-0.08, -0.04] | -4.37 | < .001  
insurance [yes] | -0.67 | 0.12 | [-0.86, -0.47] | -5.55 | < .001

The model has a log- or logit-link. Consider using `exponentiate =  
 TRUE` to interpret coefficients as ratios.

## mod\_3 performance summaries

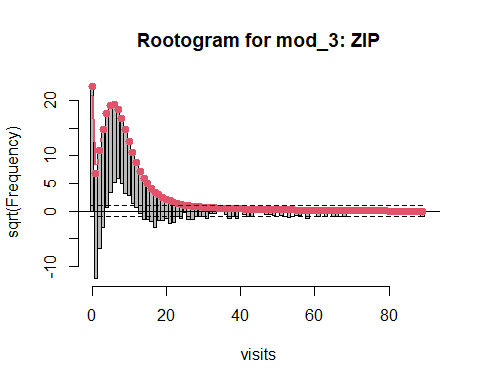
model\_performance(mod\_3)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------------  
24467.627 | 24467.792 | 24565.273 | 0.656 | 0.656 | 6.560 | 6.576  
  
AIC | Score\_log | Score\_spherical  
---------------------------------------  
24467.627 | -3.698 | 0.013

* No glance() available.

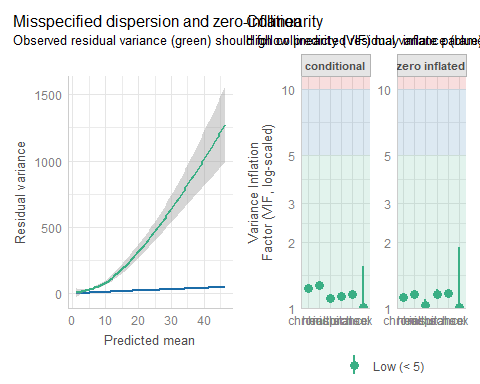
## Rootogram for ZIP model

plot(rootogram(mod\_3, plot = FALSE), xlim = c(0, 90),   
 main = "Rootogram for mod\_3: ZIP")



## Check mod\_3 model

check\_model(mod\_3, check = c("vif", "overdispersion"),  
 residual\_type = "normal", detrend = FALSE)



## Store 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 × 4  
 subject visits .fitted .resid  
 <chr> <dbl> <dbl> <dbl>  
1 355 19 5.31 13.7   
2 2661 3 4.21 -1.21  
3 2895 0 4.59 -4.59

## Training: 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 |> gt() |> fmt\_number(decimals = 3)

| model | .metric | .estimator | .estimate |
| --- | --- | --- | --- |
| mod\_3 | rsq | standard | 0.108 |
| mod\_3 | rmse | standard | 6.560 |
| mod\_3 | mae | standard | 4.164 |

## Training: Through mod\_3

bind\_rows(mod\_1\_summary, mod\_2\_summary, mod\_3\_summary) |>   
 pivot\_wider(names\_from = model,   
 values\_from = .estimate) |>   
 gt() |> fmt\_number(decimals = 3) |>   
 tab\_options(table.font.size = 20)

| .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 and smaller values of RMSE and MAE.

## Comparing models with Vuong

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 better than mod\_1 (Poisson) in our training sample.

# 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)

## mod\_4 summary

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   
-1.1943 -0.7072 -0.2773 0.3347 17.2775   
  
Count model coefficients (negbin with log link):  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 1.059477 0.070188 15.095 < 2e-16 \*\*\*  
hospital 0.207087 0.023460 8.827 < 2e-16 \*\*\*  
healthpoor 0.333854 0.052720 6.333 2.41e-10 \*\*\*  
healthexcellent -0.288168 0.073627 -3.914 9.08e-05 \*\*\*  
chronic 0.126107 0.013737 9.180 < 2e-16 \*\*\*  
sexfemale 0.069573 0.035975 1.934 0.05312 .   
school 0.025015 0.004983 5.020 5.18e-07 \*\*\*  
insuranceyes 0.151514 0.048328 3.135 0.00172 \*\*   
Log(theta) 0.389048 0.040783 9.539 < 2e-16 \*\*\*  
  
Zero-inflation model coefficients (binomial with logit link):  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.45740 0.31049 1.473 0.14071   
hospital -0.84887 0.57525 -1.476 0.14004   
healthpoor -0.33729 0.70242 -0.480 0.63110   
healthexcellent 0.27654 0.33753 0.819 0.41262   
chronic -1.16806 0.20536 -5.688 1.29e-08 \*\*\*  
sexfemale -0.56382 0.24204 -2.329 0.01983 \*   
school -0.09233 0.03157 -2.925 0.00345 \*\*   
insuranceyes -1.00819 0.27195 -3.707 0.00021 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
  
Theta = 1.4756   
Number of iterations in BFGS optimization: 48   
Log-likelihood: -9102 on 17 Df

## mod\_4 parameters

model\_parameters(mod\_4, ci = 0.90)

# Fixed Effects  
  
Parameter | Log-Mean | SE | 90% CI | z | p  
--------------------------------------------------------------------------  
(Intercept) | 1.06 | 0.07 | [ 0.94, 1.17] | 15.09 | < .001  
hospital | 0.21 | 0.02 | [ 0.17, 0.25] | 8.83 | < .001  
health [poor] | 0.33 | 0.05 | [ 0.25, 0.42] | 6.33 | < .001  
health [excellent] | -0.29 | 0.07 | [-0.41, -0.17] | -3.91 | < .001  
chronic | 0.13 | 0.01 | [ 0.10, 0.15] | 9.18 | < .001  
sex [female] | 0.07 | 0.04 | [ 0.01, 0.13] | 1.93 | 0.053   
school | 0.03 | 4.98e-03 | [ 0.02, 0.03] | 5.02 | < .001  
insurance [yes] | 0.15 | 0.05 | [ 0.07, 0.23] | 3.14 | 0.002   
  
# Zero-Inflation  
  
Parameter | Log-Odds | SE | 90% CI | z | p  
----------------------------------------------------------------------  
(Intercept) | 0.46 | 0.31 | [-0.05, 0.97] | 1.47 | 0.141   
hospital | -0.85 | 0.58 | [-1.80, 0.10] | -1.48 | 0.140   
health [poor] | -0.34 | 0.70 | [-1.49, 0.82] | -0.48 | 0.631   
health [excellent] | 0.28 | 0.34 | [-0.28, 0.83] | 0.82 | 0.413   
chronic | -1.17 | 0.21 | [-1.51, -0.83] | -5.69 | < .001  
sex [female] | -0.56 | 0.24 | [-0.96, -0.17] | -2.33 | 0.020   
school | -0.09 | 0.03 | [-0.14, -0.04] | -2.92 | 0.003   
insurance [yes] | -1.01 | 0.27 | [-1.46, -0.56] | -3.71 | < .001

## mod\_4 performance summaries

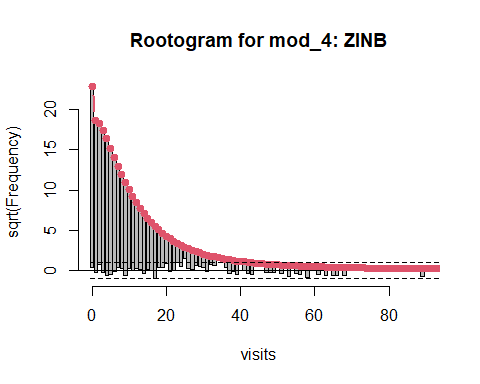
model\_performance(mod\_4)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------------  
18237.904 | 18238.090 | 18341.653 | 0.895 | 0.895 | 6.709 | 6.726  
  
AIC | Score\_log | Score\_spherical  
---------------------------------------  
18237.904 | -2.853 | 0.014

* No glance() available.

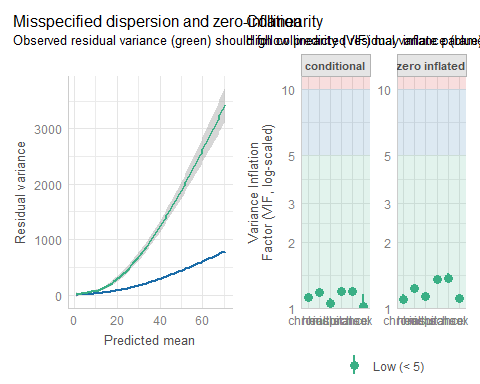
## Rootogram for ZINB model

plot(rootogram(mod\_4, plot = FALSE), xlim = c(0, 90),   
 main = "Rootogram for mod\_4: ZINB")



## Check mod\_4 model

check\_model(mod\_4, check = c("vif", "overdispersion"),  
 residual\_type = "normal", detrend = FALSE)



## Store 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 × 4  
 subject visits .fitted .resid  
 <chr> <dbl> <dbl> <dbl>  
1 355 19 5.29 13.7   
2 2661 3 4.47 -1.47  
3 2895 0 4.57 -4.57

## 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 |> gt() |> fmt\_number(decimals = 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 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) |>   
 gt() |> fmt\_number(decimals = 3) |>   
 tab\_options(table.font.size = 20)

| .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

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 4.808304 model1 > model2 7.6108e-07  
AIC-corrected 4.082004 model1 > model2 2.2325e-05  
BIC-corrected 1.865741 model1 > model2 0.031039

The large positive z-statistics indicate mod\_4 (ZINB) fits 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.

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.

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" "chronic" "sex"   
 [7] "school" "insurance" "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: 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) |>   
 gt() |> fmt\_number(decimals = 3) |>   
 tab\_options(table.font.size = 20)

| .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 looks best? Is it an obvious choice?