431 Class 21

https://thomaselove.github.io/431-2024/

2024-11-12

## Today’s Agenda

1. Reviewing (**and Fixing**) What We Have Done So Far
2. Considering Bayesian alternative fits with weakly informative priors
   * What must we do differently (Bayes vs. OLS)?
3. Assessing the candidate models more thoroughly, in both the training and test samples
   * MAPE, RMSPE, Maximum Prediction Error, Validated
4. Incorporating multiple imputation in building a final model

## 431 strategy: “most useful” model?

1. Split the data into a development (model training) sample of about 70-80% of the observations, and a holdout (model test) sample, containing the remaining observations.
2. Develop candidate models using the development sample.
3. Assess the quality of fit for candidate models within the development sample.
4. Check adherence to regression assumptions in the development sample.

## 431 strategy: “most useful” model?

1. When you have candidates, assess them based on the accuracy of the predictions they make for the data held out (and thus not used in building the models.)
2. Select a “final” model for use based on the evidence in steps 3, 4 and especially 5.

## R Packages and Data Load

knitr::opts\_chunk$set(comment = NA)  
library(janitor)

Attaching package: 'janitor'

The following objects are masked from 'package:stats':  
  
 chisq.test, fisher.test

library(mice)

Attaching package: 'mice'

The following object is masked from 'package:stats':  
  
 filter

The following objects are masked from 'package:base':  
  
 cbind, rbind

library(naniar)  
library(patchwork)  
library(car) ## for vif function as well as Box-Cox

Loading required package: carData

library(GGally) ## for ggpairs scatterplot matrix

Loading required package: ggplot2

Registered S3 method overwritten by 'GGally':  
 method from   
 +.gg ggplot2

library(broom) ## for predictions, residuals with augment  
library(rstanarm) ## fitting Bayesian regressions

Loading required package: Rcpp

This is rstanarm version 2.32.1

- See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

- Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.

- For execution on a local, multicore CPU with excess RAM we recommend calling

options(mc.cores = parallel::detectCores())

Attaching package: 'rstanarm'

The following object is masked from 'package:car':  
  
 logit

library(gt) ## some prettier tables  
library(easystats)

# Attaching packages: easystats 0.7.3  
✔ bayestestR 0.15.0 ✔ correlation 0.8.6   
✔ datawizard 0.13.0 ✔ effectsize 0.8.9   
✔ insight 0.20.5 ✔ modelbased 0.8.9   
✔ performance 0.12.4 ✔ parameters 0.23.0  
✔ report 0.5.9 ✔ see 0.9.0

library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.5  
✔ forcats 1.0.0 ✔ stringr 1.5.1  
✔ lubridate 1.9.3 ✔ tibble 3.2.1  
✔ purrr 1.0.2 ✔ tidyr 1.3.1

── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks mice::filter(), stats::filter()  
✖ dplyr::lag() masks stats::lag()  
✖ dplyr::recode() masks car::recode()  
✖ purrr::some() masks car::some()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

source("c21/data/Love-431.R")  
theme\_set(theme\_bw())  
dm500 <- readRDS("c21/data/dm500.Rds")

# What We’ve Done So Far

## Imputation and Partitioning

set.seed(20241031)  
  
dm500\_tenimps <- mice(dm500, m = 10, printFlag = FALSE)

Warning: Number of logged events: 1

dm500\_i <- complete(dm500\_tenimps, 7) |> tibble()  
  
set.seed(4312024)  
dm500\_i\_train <- dm500\_i |>   
 slice\_sample(prop = 0.7, replace = FALSE)  
dm500\_i\_test <-   
 anti\_join(dm500\_i, dm500\_i\_train, by = "subject")

# Fixing My Mistake from Classes 19-20

## Three Regression Models We’ve Fit

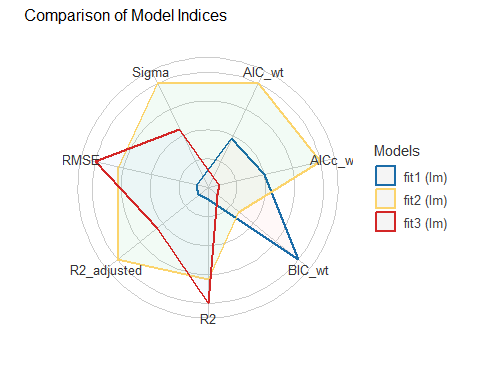
* Using the model training sample, and a (100/a1c) outcome transformation.
* My mistake: once you decide on a transformation, **create it before fitting**.

dm500\_i\_train <- dm500\_i\_train |> mutate(transa1c = 100/a1c)  
  
fit1 <- lm(transa1c ~ a1c\_old, data = dm500\_i\_train)  
  
fit2 <- lm(transa1c ~ a1c\_old + age, data = dm500\_i\_train)  
  
fit3 <- lm(transa1c ~ a1c\_old + age + income,   
 data = dm500\_i\_train)

## Performance Indices for 3 Models

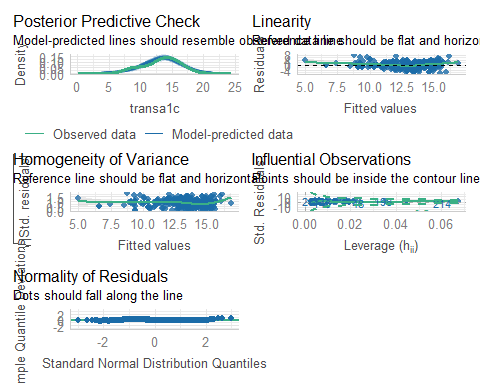
* in the training sample

plot(compare\_performance(fit1, fit2, fit3))



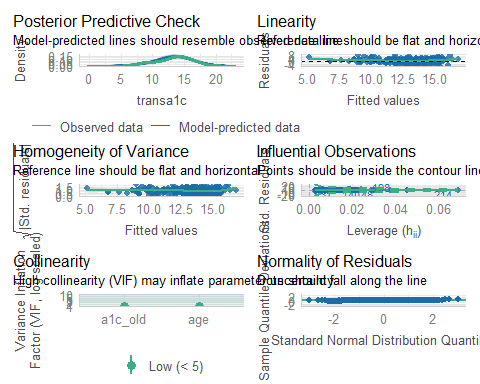
## OLS Model fit1 Checking

check\_model(fit1)



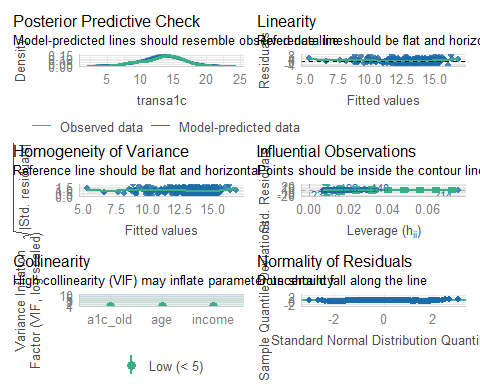
## OLS Model fit2 Checking

check\_model(fit2)



## OLS Model fit3 Checking

check\_model(fit3)



## augment training samples

aug1 <- augment(fit1, data = dm500\_i\_train)  
aug2 <- augment(fit2, data = dm500\_i\_train)  
aug3 <- augment(fit3, data = dm500\_i\_train)

augment results for fit2 in our first four subjects…

aug2 |> head() |> gt()

| a1c | a1c\_old | age | income | subject | transa1c | .fitted | .resid | .hat | .sigma | .cooksd | .std.resid |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11.6 | 5.6 | 54 | Below\_30K | S-168 | 8.62069 | 15.38293 | -6.7622361 | 0.007351110 | 2.276248 | 2.145934e-02 | -2.9484254 |
| 6.5 | 6.4 | 68 | Higher\_than\_50K | S-359 | 15.38462 | 14.96010 | 0.4245154 | 0.008948617 | 2.305194 | 1.032820e-04 | 0.1852435 |
| 7.6 | 7.0 | 60 | Between\_30-50K | S-421 | 13.15789 | 14.18858 | -1.0306848 | 0.003869153 | 2.304640 | 2.605607e-04 | -0.4486063 |
| 7.3 | 8.0 | 56 | Between\_30-50K | S-037 | 13.69863 | 13.13196 | 0.5666705 | 0.002935108 | 2.305107 | 5.963655e-05 | 0.2465282 |
| 7.7 | 7.8 | 63 | Between\_30-50K | S-030 | 12.98701 | 13.49557 | -0.5085564 | 0.004906357 | 2.305146 | 8.060899e-05 | -0.2214648 |
| 6.4 | 7.0 | 34 | Below\_30K | S-055 | 15.62500 | 13.54996 | 2.0750381 | 0.020785932 | 2.302550 | 5.871383e-03 | 0.9109298 |

# Bayesian fits instead?

## Refit with Bayesian models?

What must we change to use Bayesian (stan\_glm) fits?

1. Must create transformed outcome in data prior to fitting with rstanarm().
2. Results are a bit different for model\_parameters()
3. Results are a bit different for model\_performance()
4. No real change for check\_models()
5. There is no augment() function for rstanarm() fits.

## Refit with Bayesian models?

with default weakly informative priors

set.seed(20241112)  
  
fit1B <- stan\_glm(transa1c ~ a1c\_old,   
 data = dm500\_i\_train, refresh = 0)  
fit2B <- stan\_glm(transa1c ~ a1c\_old + age,   
 data = dm500\_i\_train, refresh = 0)  
fit3B <- stan\_glm(transa1c ~ a1c\_old + age + income,   
 data = dm500\_i\_train, refresh = 0)

## Estimating fit1 Coefficients?

* Bayesian fit

model\_parameters(fit1B, ci = 0.95) |> gt()

| Parameter | Median | CI | CI\_low | CI\_high | pd | Rhat | ESS | Prior\_Distribution | Prior\_Location | Prior\_Scale |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 20.97661 | 0.95 | 19.910242 | 22.0855930 | 1 | 1.000731 | 4048.198 | normal | 13.36034 | 7.270914 |
| a1c\_old | -0.98244 | 0.95 | -1.118648 | -0.8463795 | 1 | 1.000882 | 3928.896 | normal | 0.00000 | 4.028203 |

* OLS fit

model\_parameters(fit1, ci = 0.95) |> gt() |> fmt\_number(decimals = 3)

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 20.968 | 0.545 | 0.950 | 19.896 | 22.039 | 38.490 | 348.000 | 0.000 |
| a1c\_old | -0.982 | 0.068 | 0.950 | -1.117 | -0.847 | -14.338 | 348.000 | 0.000 |

## Estimating fit2 Coefficients?

model\_parameters(fit2B, ci = 0.95) |> gt() |> fmt\_number(decimals = 3)

| Parameter | Median | CI | CI\_low | CI\_high | pd | Rhat | ESS | Prior\_Distribution | Prior\_Location | Prior\_Scale |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.406 | 0.950 | 17.316 | 21.372 | 1.000 | 1.001 | 4,749.509 | normal | 13.360 | 7.271 |
| a1c\_old | -0.957 | 0.950 | -1.087 | -0.817 | 1.000 | 1.001 | 4,638.566 | normal | 0.000 | 4.028 |
| age | 0.025 | 0.950 | -0.002 | 0.052 | 0.962 | 1.000 | 5,188.851 | normal | 0.000 | 0.794 |

model\_parameters(fit2, ci = 0.95) |> gt() |> fmt\_number(decimals = 3)

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.423 | 1.019 | 0.950 | 17.420 | 21.427 | 19.064 | 347.000 | 0.000 |
| a1c\_old | -0.958 | 0.070 | 0.950 | -1.095 | -0.822 | -13.786 | 347.000 | 0.000 |
| age | 0.025 | 0.014 | 0.950 | -0.002 | 0.052 | 1.791 | 347.000 | 0.074 |

## Estimating fit3 Coefficients?

model\_parameters(fit3B, ci = 0.95) |> gt() |> fmt\_number(decimals = 3)

| Parameter | Median | CI | CI\_low | CI\_high | pd | Rhat | ESS | Prior\_Distribution | Prior\_Location | Prior\_Scale |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.487 | 0.950 | 17.457 | 21.626 | 1.000 | 1.000 | 3,032.152 | normal | 13.360 | 7.271 |
| a1c\_old | -0.953 | 0.950 | -1.088 | -0.815 | 1.000 | 1.000 | 4,061.029 | normal | 0.000 | 4.028 |
| age | 0.023 | 0.950 | -0.004 | 0.051 | 0.953 | 1.000 | 3,711.976 | normal | 0.000 | 0.794 |
| incomeBetween\_30-50K | 0.058 | 0.950 | -0.580 | 0.697 | 0.566 | 1.001 | 2,710.381 | normal | 0.000 | 14.803 |
| incomeBelow\_30K | -0.209 | 0.950 | -0.831 | 0.454 | 0.734 | 1.000 | 2,758.186 | normal | 0.000 | 15.100 |

model\_parameters(fit3, ci = 0.95) |> gt() |> fmt\_number(decimals = 3)

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.492 | 1.069 | 0.950 | 17.390 | 21.594 | 18.240 | 345.000 | 0.000 |
| a1c\_old | -0.952 | 0.070 | 0.950 | -1.090 | -0.815 | -13.608 | 345.000 | 0.000 |
| age | 0.023 | 0.014 | 0.950 | -0.004 | 0.051 | 1.687 | 345.000 | 0.092 |
| incomeBetween\_30-50K | 0.052 | 0.323 | 0.950 | -0.585 | 0.688 | 0.160 | 345.000 | 0.873 |
| incomeBelow\_30K | -0.214 | 0.334 | 0.950 | -0.870 | 0.442 | -0.641 | 345.000 | 0.522 |

## Model Performance with fit1B?

model\_performance(fit1B) |> gt() |> tab\_options(table.font.size = 24)

| ELPD | ELPD\_SE | LOOIC | LOOIC\_SE | WAIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -792.0265 | 15.45717 | 1584.053 | 30.91434 | 1584.023 | 0.3699966 | 0.3672415 | 2.302677 | 2.312659 |

* R2 = “unadjusted” Bayesian (see [r2\_bayes](https://easystats.github.io/performance/reference/r2_bayes.html) for details)
* R2\_adjusted = leave-one-out cross-validation (LOO) adjusted , which is conceptually closer to our OLS adjusted than some other options.
* RMSE = root mean squared error (standard deviation of the unexplained variance) and lower values mean better fit.

## Bayesian Model Performance

model\_performance(fit2B) |> gt() |> tab\_options(table.font.size = 24)

| ELPD | ELPD\_SE | LOOIC | LOOIC\_SE | WAIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -791.2411 | 15.42323 | 1582.482 | 30.84646 | 1582.476 | 0.3752328 | 0.3691855 | 2.292108 | 2.301049 |

* Sigma = residual standard deviation (interpret in same way as RMSE)
* WAIC = widely applicable information criterion. Lower WAIC values mean better fit.

## Bayesian Model Performance

model\_performance(fit3B) |> gt() |> tab\_options(table.font.size = 24)

| ELPD | ELPD\_SE | LOOIC | LOOIC\_SE | WAIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -792.8554 | 15.42613 | 1585.711 | 30.85225 | 1585.686 | 0.3782747 | 0.3614431 | 2.289063 | 2.306763 |

* LOOIC = leave-one-out cross-validation (LOO) information criterion. Lower LOOIC values mean better fit.
* LOOIC\_SE = standard error of LOOIC

See [this link](https://easystats.github.io/performance/reference/model_performance.stanreg.html) on “Performance of Bayesian Models” for still more options.

## Performance within Training Sample?

compare\_performance(fit1B, fit2B, fit3B, rank = TRUE) |>   
 gt() |> fmt\_number(decimals = 3)

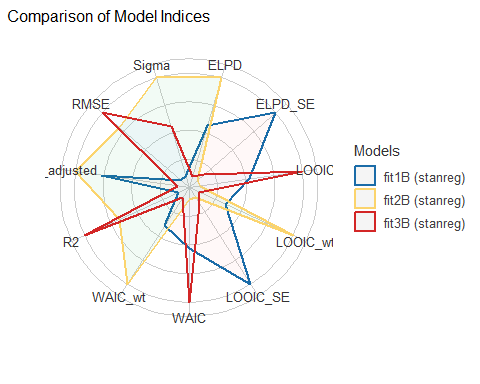
| Name | Model | R2 | R2\_adjusted | RMSE | Sigma | WAIC\_wt | LOOIC\_wt | Performance\_Score |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| fit2B | stanreg | 0.375 | 0.369 | 2.292 | 2.301 | 0.602 | 0.781 | 0.901 |
| fit3B | stanreg | 0.378 | 0.361 | 2.289 | 2.307 | 0.121 | 0.000 | 0.418 |
| fit1B | stanreg | 0.370 | 0.367 | 2.303 | 2.313 | 0.277 | 0.219 | 0.226 |

compare\_performance(fit1, fit2, fit3, rank = TRUE) |>   
 gt() |> fmt\_number(decimals = 3)

| Name | Model | R2 | R2\_adjusted | RMSE | Sigma | AIC\_wt | AICc\_wt | BIC\_wt | Performance\_Score |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| fit2 | lm | 0.377 | 0.374 | 2.292 | 2.302 | 0.568 | 0.568 | 0.211 | 0.831 |
| fit3 | lm | 0.379 | 0.372 | 2.289 | 2.306 | 0.123 | 0.116 | 0.001 | 0.433 |
| fit1 | lm | 0.371 | 0.370 | 2.303 | 2.309 | 0.308 | 0.316 | 0.788 | 0.265 |

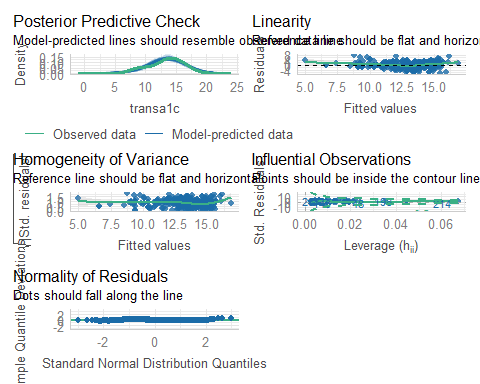
## Bayes Performance Indicators?

plot(compare\_performance(fit1B, fit2B, fit3B))



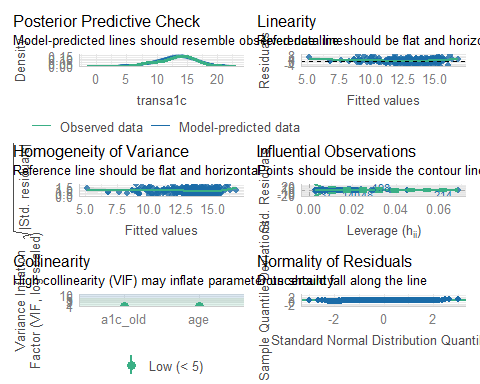
## Checking Bayesian fit1B

check\_model(fit1B)



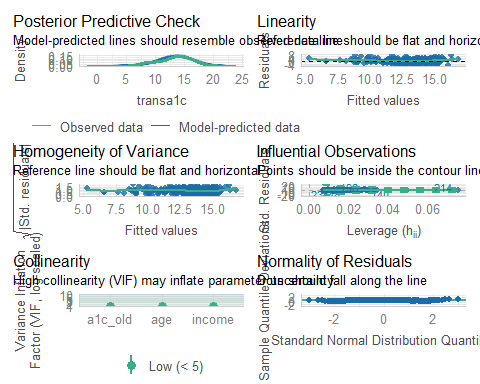
## Checking Bayesian fit2B

check\_model(fit2B)



## Checking Bayesian fit3B

check\_model(fit3B)



## No augment from Bayes fits

The augment() function from **broom** doesn’t work with Bayesian fits using rstanarm().

We still want to get our predicted (fitted) values and residuals for each observation in our training sample.

aug\_1B <- dm500\_i\_train |>  
 mutate(.fitted = predict(fit1B),  
 .resid = transa1c - predict(fit1B))  
aug\_2B <- dm500\_i\_train |>  
 mutate(.fitted = predict(fit2B),  
 .resid = transa1c - predict(fit2B))  
aug\_3B <- dm500\_i\_train |>  
 mutate(.fitted = predict(fit3B),  
 .resid = transa1c - predict(fit3B))

## fit2 model (OLS vs Bayes)

aug2 |> select(1:8) |> head(3)

# A tibble: 3 × 8  
 a1c a1c\_old age income subject transa1c .fitted .resid  
 <dbl> <dbl> <dbl> <fct> <chr> <dbl> <dbl> <dbl>  
1 11.6 5.6 54 Below\_30K S-168 8.62 15.4 -6.76   
2 6.5 6.4 68 Higher\_than\_50K S-359 15.4 15.0 0.425  
3 7.6 7 60 Between\_30-50K S-421 13.2 14.2 -1.03

aug\_2B |> head(3)

# A tibble: 3 × 8  
 a1c a1c\_old age income subject transa1c .fitted .resid  
 <dbl> <dbl> <dbl> <fct> <chr> <dbl> <dbl> <dbl>  
1 11.6 5.6 54 Below\_30K S-168 8.62 15.4 -6.76   
2 6.5 6.4 68 Higher\_than\_50K S-359 15.4 15.0 0.419  
3 7.6 7 60 Between\_30-50K S-421 13.2 14.2 -1.04

# Extending to the Test Sample

## Making Predictions: Test Sample

* Create transformed outcome in test sample.
* For OLS fits, apply augment() to new data = test sample.

dm500\_i\_test <- dm500\_i\_test |> mutate(transa1c = 100/a1c)  
  
aug1\_test <- augment(fit1, newdata = dm500\_i\_test) |>   
 mutate(mod = "fit1")  
aug2\_test <- augment(fit2, newdata = dm500\_i\_test) |>   
 mutate(mod = "fit2")  
aug3\_test <- augment(fit3, newdata = dm500\_i\_test) |>   
 mutate(mod = "fit3")

Results on next slide…

## augment for each model

aug1\_test |> head(2) |> gt()

| a1c | a1c\_old | age | income | subject | transa1c | .fitted | .resid | mod |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8.7 | 10.7 | 47 | Higher\_than\_50K | S-003 | 11.49425 | 10.46152 | 1.032736 | fit1 |
| 6.7 | 6.3 | 64 | Between\_30-50K | S-005 | 14.92537 | 14.78184 | 0.143537 | fit1 |

aug2\_test |> head(2) |> gt()

| a1c | a1c\_old | age | income | subject | transa1c | .fitted | .resid | mod |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8.7 | 10.7 | 47 | Higher\_than\_50K | S-003 | 11.49425 | 10.32330 | 1.17095515 | fit2 |
| 6.7 | 6.3 | 64 | Between\_30-50K | S-005 | 14.92537 | 14.95769 | -0.03231506 | fit2 |

aug3\_test |> head(2) |> gt()

| a1c | a1c\_old | age | income | subject | transa1c | .fitted | .resid | mod |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8.7 | 10.7 | 47 | Higher\_than\_50K | S-003 | 11.49425 | 10.40699 | 1.0872627 | fit3 |
| 6.7 | 6.3 | 64 | Between\_30-50K | S-005 | 14.92537 | 15.04821 | -0.1228401 | fit3 |

## Combine Augmented Results

temp12 <- bind\_rows(aug1\_test, aug2\_test)  
test\_res <- bind\_rows(temp12, aug3\_test) |>  
 relocate(subject, mod, a1c, everything()) |>  
 arrange(subject, mod)  
  
test\_res |> head() |> gt() |> tab\_options(table.font.size = 24)

| subject | mod | a1c | a1c\_old | age | income | transa1c | .fitted | .resid |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S-003 | fit1 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.46152 | 1.03273565 |
| S-003 | fit2 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.32330 | 1.17095515 |
| S-003 | fit3 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.40699 | 1.08726268 |
| S-005 | fit1 | 6.7 | 6.3 | 64 | Between\_30-50K | 14.92537 | 14.78184 | 0.14353701 |
| S-005 | fit2 | 6.7 | 6.3 | 64 | Between\_30-50K | 14.92537 | 14.95769 | -0.03231506 |
| S-005 | fit3 | 6.7 | 6.3 | 64 | Between\_30-50K | 14.92537 | 15.04821 | -0.12284005 |

## Calculating Prediction Errors

For each of our (OLS) models, we have:

* a1c, the outcome we actually care about
* transa1c = 100/a1c, the transformed outcome
* .fitted, a predicted transa1c from the model

What we want is

* a1c\_pred, the predicted a1c using this model, and
* a1c\_error, the error made in predicting a1c

## Back-Transformation

Each .fitted value is a prediction of 100/a1c.

$$
\frac{100}{a1c} = \mbox{.fitted}, \mbox{ so } \frac{1}{a1c} = \frac{\mbox{.fitted}}{100}, \mbox{ and so}\\
a1c = \frac{100}{\mbox{.fitted}}
$$

To get a1c\_pred we need to back-transform our .fitted values, with 100/.fitted, it seems.

## Adding predicted A1c and error

* We add a1c\_pred, the predicted a1c using this model, and
* a1c\_error = error for this model in predicting a1c

test\_res <- test\_res |>   
 mutate(a1c\_pred = 100/.fitted,  
 a1c\_error = a1c - a1c\_pred)  
  
test\_res |> head(4) |> gt()

| subject | mod | a1c | a1c\_old | age | income | transa1c | .fitted | .resid | a1c\_pred | a1c\_error |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S-003 | fit1 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.46152 | 1.032736 | 9.558843 | -0.85884294 |
| S-003 | fit2 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.32330 | 1.170955 | 9.686827 | -0.98682708 |
| S-003 | fit3 | 8.7 | 10.7 | 47 | Higher\_than\_50K | 11.49425 | 10.40699 | 1.087263 | 9.608926 | -0.90892613 |
| S-005 | fit1 | 6.7 | 6.3 | 64 | Between\_30-50K | 14.92537 | 14.78184 | 0.143537 | 6.765059 | -0.06505944 |

## Summarize Errors for Each Fit

We’ll look at four summary measures in 431…

1. MAPE = mean absolute prediction error
2. Max\_APE = maximum absolute prediction error
3. RMSPE = square root of mean squared prediction error
4. Validated = squared correlation of predictions and outcome in the new data

Measures 1-3 are all in the same units as a1c, our outcome.

## Building our Four Summaries

The test\_res tibble contains, for each model,

* a1c, the actual outcome value
* a1c\_pred, the predicted outcome value
* a1c\_error = a1c - a1c\_pred = prediction error

test\_summary <- test\_res |>   
 group\_by(mod) |>  
 summarize(MAPE = mean(abs(a1c\_error)),  
 Max\_APE = max(abs(a1c\_error)),  
 RMSE = sqrt(mean(a1c\_error^2)),  
 R2\_val = cor(a1c, a1c\_pred)^2)

## Table for Test-Sample Prediction

test\_summary |> gt() |>   
 fmt\_number(decimals = 4) |> tab\_options(table.font.size = 30)

| mod | MAPE | Max\_APE | RMSE | R2\_val |
| --- | --- | --- | --- | --- |
| fit1 | 1.0825 | 4.9541 | 1.5404 | 0.4350 |
| fit2 | 1.0770 | 5.0353 | 1.5310 | 0.4430 |
| fit3 | 1.0822 | 5.0564 | 1.5315 | 0.4409 |

* Which model is best, by these metrics?
* fit2 has smallest MAPE, RMSPE, and largest validated
* fit1 has smallest MaxAPE

## What about a Bayesian fit?

Everything is the same as OLS, except we’d have to work around the use of augment in our test data, but predict() can handle what we need, as below.

test\_1B <- dm500\_i\_test |>  
 mutate(a1c\_pred = predict(fit1B, newdata = dm500\_i\_test),  
 a1c\_error = transa1c - a1c\_pred)  
test\_1B |> head(4) |> gt()

| a1c | a1c\_old | age | income | subject | transa1c | a1c\_pred | a1c\_error |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8.7 | 10.7 | 47 | Higher\_than\_50K | S-003 | 11.49425 | 10.45906 | 1.0351920 |
| 6.7 | 6.3 | 64 | Between\_30-50K | S-005 | 14.92537 | 14.78261 | 0.1427673 |
| 8.4 | 8.6 | 44 | Between\_30-50K | S-014 | 11.90476 | 12.52257 | -0.6178090 |
| 5.7 | 5.7 | 52 | Between\_30-50K | S-015 | 17.54386 | 15.37218 | 2.1716795 |

# Putting it all together

## Comparing fit1, fit2, fit3

* fit1 includes a1c\_old, fit2 adds age, fit3 adds income
* Similar model assumption problems (hard-to-ignore problem with linearity, maybe some non-constant variance)
* **training** sample: fit2 performed better than the others on adjusted , AIC and , while fit1 was best on BIC, and fit3 was best on RMSE.
* **test** sample: fit2 performed better on MAPE, RMSPE and validated , while fit1 had the smallest maximum APE.
* Differences between models on most metrics were modest.

# Let’s Choose Model fit2

## Complete Case Analysis

* **Project B Study 2**: Report the model building process, displaying conclusions from training sample (transformation decisions, fitted parameters with CIs and interpretations, comparison of performance metrics, checks of model assumptions) and from test sample (comparison of predictive error)
* Of course, here we did some imputation, so we’d need to do this all over again to get these results.

## Simple Imputation

* **Project B Study 2**: Report the model building process, displaying conclusions from training sample (transformation decisions, fitted parameters with CIs and interpretations, comparison of performance metrics, checks of model assumptions) and from test sample (comparison of predictive error)
* Sometimes, we do some of the steps above without reporting them to the public, of course. But that’s not the goal here.

## Model fit2 using Single Imputation

Estimated using a single imputation in training sample.

n\_obs(fit2)

[1] 350

model\_parameters(fit2, ci = 0.95) |> gt() |>   
 fmt\_number(decimals = 3) |> tab\_options(table.font.size = 24)

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.423 | 1.019 | 0.950 | 17.420 | 21.427 | 19.064 | 347.000 | 0.000 |
| a1c\_old | -0.958 | 0.070 | 0.950 | -1.095 | -0.822 | -13.786 | 347.000 | 0.000 |
| age | 0.025 | 0.014 | 0.950 | -0.002 | 0.052 | 1.791 | 347.000 | 0.074 |

## Model fit2 using Single Imputation

glance(fit2) |> select(1:6) |> gt() |>   
 tab\_options(table.font.size = 24)

| r.squared | adj.r.squared | sigma | statistic | p.value | df |
| --- | --- | --- | --- | --- | --- |
| 0.3771109 | 0.3735207 | 2.301984 | 105.0407 | 2.139262e-36 | 2 |

glance(fit2) |> select(7:12) |> gt() |>   
 tab\_options(table.font.size = 24)

| logLik | AIC | BIC | deviance | df.residual | nobs |
| --- | --- | --- | --- | --- | --- |
| -786.942 | 1581.884 | 1597.316 | 1838.799 | 347 | 350 |

## Incorporating Multiple Imputations

* **Same as Project B Study 2**: Report the model building process, displaying conclusions from training sample (transformation decisions, fitted parameters with CIs and interpretations, comparison of performance metrics, checks of model assumptions) and from test sample (comparison of predictive error)
* **New** Then **add** information on fitted parameters across the whole data set (not split into training and testing) after multiple (in this case, 10) imputations.

## Using Ten Imputations

dm500\_tenimps contained mice results across 10 imputations, then we used the 7th in our work. Let’s build a new set of results across the model we’ve settled on, with a fresh set of 10 imputations.

* The original data (with missing values) are in dm500 - we need only to add our transformed outcome, 100/a1c.

dm500 <- dm500 |> mutate(transa1c = 100/a1c)  
  
imp2\_ests <- dm500 |>  
 mice(m = 10, seed = 431, print = FALSE) |>  
 with(lm(transa1c ~ a1c\_old + age)) |>  
 pool()

Warning: Number of logged events: 1

## Estimates across 10 imputations

model\_parameters(imp2\_ests, ci = 0.95) |> gt() |>   
 fmt\_number(decimals = 3) |> tab\_options(table.font.size = 24)

Warning: Number of logged events: 1  
Warning: Number of logged events: 1  
Warning: Number of logged events: 1  
Warning: Number of logged events: 1

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 19.933 | 0.848 | 0.950 | 18.266 | 21.600 | 23.495 | 485.520 | 0.000 |
| a1c\_old | -0.998 | 0.060 | 0.950 | -1.115 | -0.881 | -16.771 | 465.427 | 0.000 |
| age | 0.021 | 0.012 | 0.950 | -0.002 | 0.043 | 1.786 | 482.672 | 0.075 |

glance(imp2\_ests) |> gt() |> tab\_options(table.font.size = 24)

| nimp | nobs | r.squared | adj.r.squared |
| --- | --- | --- | --- |
| 10 | 500 | 0.3842468 | 0.3817684 |

n\_obs(imp2\_ests)

[1] 500 500 500 500 500 500 500 500 500 500

## What’s next?

* Do most of this again, including some additional bells and whistles, and some new data.
* Multiple imputation with Bayesian linear models is a topic for 432.

## Session Information

xfun::session\_info()

R version 4.4.1 (2024-06-14 ucrt)  
Platform: x86\_64-w64-mingw32/x64  
Running under: Windows 11 x64 (build 22631)  
  
Locale:  
 LC\_COLLATE=English\_United States.utf8   
 LC\_CTYPE=English\_United States.utf8   
 LC\_MONETARY=English\_United States.utf8  
 LC\_NUMERIC=C   
 LC\_TIME=English\_United States.utf8   
  
Package version:  
 abind\_1.4-8 askpass\_1.2.1 backports\_1.5.0   
 base64enc\_0.1-3 bayesplot\_1.11.1 bayestestR\_0.15.0   
 BH\_1.84.0.0 bigD\_0.3.0 bit\_4.5.0   
 bit64\_4.5.2 bitops\_1.0.9 blob\_1.2.4   
 boot\_1.3-31 broom\_1.0.7 bslib\_0.8.0   
 cachem\_1.1.0 callr\_3.7.6 car\_3.1-3   
 carData\_3.0-5 cellranger\_1.1.0 checkmate\_2.3.2   
 cli\_3.6.3 clipr\_0.8.0 coda\_0.19-4.1   
 codetools\_0.2-20 colorspace\_2.1-1 colourpicker\_1.3.0   
 commonmark\_1.9.2 compiler\_4.4.1 conflicted\_1.2.0   
 correlation\_0.8.6 cowplot\_1.1.3 cpp11\_0.5.0   
 crayon\_1.5.3 crosstalk\_1.2.1 curl\_5.2.3   
 data.table\_1.16.2 datasets\_4.4.1 datawizard\_0.13.0   
 DBI\_1.2.3 dbplyr\_2.5.0 Deriv\_4.1.6   
 desc\_1.4.3 digest\_0.6.37 distributional\_0.5.0  
 doBy\_4.6.24 dplyr\_1.1.4 DT\_0.33   
 dtplyr\_1.3.1 dygraphs\_1.1.1.6 easystats\_0.7.3   
 effectsize\_0.8.9 emmeans\_1.10.5 estimability\_1.5.1   
 evaluate\_1.0.1 fansi\_1.0.6 farver\_2.1.2   
 fastmap\_1.2.0 fontawesome\_0.5.2 forcats\_1.0.0   
 foreach\_1.5.2 Formula\_1.2-5 fs\_1.6.5   
 gargle\_1.5.2 generics\_0.1.3 GGally\_2.2.1   
 ggplot2\_3.5.1 ggrepel\_0.9.6 ggridges\_0.5.6   
 ggstats\_0.7.0 glmnet\_4.1-8 glue\_1.8.0   
 googledrive\_2.1.1 googlesheets4\_1.1.1 graphics\_4.4.1   
 grDevices\_4.4.1 grid\_4.4.1 gridExtra\_2.3   
 gt\_0.11.1 gtable\_0.3.6 gtools\_3.9.5   
 haven\_2.5.4 highr\_0.11 hms\_1.1.3   
 htmltools\_0.5.8.1 htmlwidgets\_1.6.4 httpuv\_1.6.15   
 httr\_1.4.7 ids\_1.0.1 igraph\_2.1.1   
 inline\_0.3.19 insight\_0.20.5 isoband\_0.2.7   
 iterators\_1.0.14 janitor\_2.2.0 jomo\_2.7-6   
 jquerylib\_0.1.4 jsonlite\_1.8.9 juicyjuice\_0.1.0   
 knitr\_1.49 labeling\_0.4.3 later\_1.3.2   
 lattice\_0.22-6 lazyeval\_0.2.2 lifecycle\_1.0.4   
 lme4\_1.1-35.5 loo\_2.8.0 lubridate\_1.9.3   
 magrittr\_2.0.3 markdown\_1.13 MASS\_7.3-61   
 Matrix\_1.7-0 MatrixModels\_0.5.3 matrixStats\_1.4.1   
 memoise\_2.0.1 methods\_4.4.1 mgcv\_1.9-1   
 mice\_3.16.0 microbenchmark\_1.5.0 mime\_0.12   
 miniUI\_0.1.1.1 minqa\_1.2.8 mitml\_0.4-5   
 modelbased\_0.8.9 modelr\_0.1.11 multcomp\_1.4-26   
 munsell\_0.5.1 mvtnorm\_1.3-1 naniar\_1.1.0   
 nlme\_3.1-164 nloptr\_2.1.1 nnet\_7.3-19   
 norm\_1.0.11.1 numDeriv\_2016.8.1.1 openssl\_2.2.2   
 ordinal\_2023.12.4.1 pan\_1.9 parallel\_4.4.1   
 parameters\_0.23.0 patchwork\_1.3.0 pbkrtest\_0.5.3   
 performance\_0.12.4 pillar\_1.9.0 pkgbuild\_1.4.5   
 pkgconfig\_2.0.3 plyr\_1.8.9 posterior\_1.6.0   
 prettyunits\_1.2.0 processx\_3.8.4 progress\_1.2.3   
 promises\_1.3.0 ps\_1.8.1 purrr\_1.0.2   
 quantreg\_5.99 QuickJSR\_1.4.0 R6\_2.5.1   
 ragg\_1.3.3 rappdirs\_0.3.3 RColorBrewer\_1.1-3   
 Rcpp\_1.0.13 RcppEigen\_0.3.4.0.2 RcppParallel\_5.1.9   
 reactable\_0.4.4 reactR\_0.6.1 readr\_2.1.5   
 readxl\_1.4.3 rematch\_2.0.0 rematch2\_2.1.2   
 report\_0.5.9 reprex\_2.1.1 reshape2\_1.4.4   
 rlang\_1.1.4 rmarkdown\_2.29 rpart\_4.1.23   
 rstan\_2.32.6 rstanarm\_2.32.1 rstantools\_2.4.0   
 rstudioapi\_0.17.1 rvest\_1.0.4 sandwich\_3.1-1   
 sass\_0.4.9 scales\_1.3.0 see\_0.9.0   
 selectr\_0.4.2 shape\_1.4.6.1 shiny\_1.9.1   
 shinyjs\_2.1.0 shinystan\_2.6.0 shinythemes\_1.2.0   
 snakecase\_0.11.1 sourcetools\_0.1.7.1 SparseM\_1.84.2   
 splines\_4.4.1 StanHeaders\_2.32.10 stats\_4.4.1   
 stats4\_4.4.1 stringi\_1.8.4 stringr\_1.5.1   
 survival\_3.7-0 sys\_3.4.3 systemfonts\_1.1.0   
 tensorA\_0.36.2.1 textshaping\_0.4.0 TH.data\_1.1-2   
 threejs\_0.3.3 tibble\_3.2.1 tidyr\_1.3.1   
 tidyselect\_1.2.1 tidyverse\_2.0.0 timechange\_0.3.0   
 tinytex\_0.54 tools\_4.4.1 tzdb\_0.4.0   
 ucminf\_1.2.2 UpSetR\_1.4.0 utf8\_1.2.4   
 utils\_4.4.1 uuid\_1.2.1 V8\_6.0.0   
 vctrs\_0.6.5 viridis\_0.6.5 viridisLite\_0.4.2   
 visdat\_0.6.0 vroom\_1.6.5 withr\_3.0.2   
 xfun\_0.48 xml2\_1.3.6 xtable\_1.8-4   
 xts\_0.14.1 yaml\_2.3.10 zoo\_1.8-12