431 Classes 18-20

thomaselove.github.io/431

2021-10-26

Agenda for Classes 18-20

- Power / Sample Size Decisions and The March of Science
- Multiple Regresssion using the `dm1` data
    - Using `df_stats` to get `favstats` for multiple variables at once
    - Using the `naniar` package to identify and summarizing missingness
    - Complete Cases and Simple imputation to deal with missingness
    - Partitioning our data into training/test samples
    - Outcome transformation: what to consider
    - Assessing the fit in the sample where we build the model
        - Using `tidy` to describe model coefficients
        - Using `glance` to study fit quality
        - Using `augment` to obtain predicted values and residuals
        - Residual plots to check assumptions with `plot` and with `ggplot2`
    - Testing the model in new data (a holdout sample)
        - Assessing the predictions and prediction errors
        - Back-transformation and MAPE, root MSPE and max error

Schedule

We will discuss this material over Classes 18-20.

Our R Setup

```r
knitr::opts_chunk$set(comment = NA)
options(dplyr.summarise.inform = FALSE)

library(simputation) # for single impuation
library(car) # for boxCox
library(GGally) # for ggpairs
library(ggrepel) # help with residual plots
library(equatiomatic) # help with equation extraction
library(knitr); library(janitor); library(magrittr)
library(patchwork); library(broom); library(naniar)
library(tidyverse)

theme_set(theme_bw())
```

On Power / Sample Size Decisions and the March of Science

The March of Science and Power Calculations

I mentioned last time that the most common scenario was to identify:

- desired significance level α
- desired power (1 - β)

and the details of the plan in terms of what comparison is to be made, and how will the data be collected to support that comparison.

This will then permit the calculation of a minimum necessary sample size to achieve these desires.

I also mentioned that $\alpha = 0.05$ and $\beta = 0.2$ were the most common selections.

- Neither 95% confidence nor 80% power is a magical choice.
- Anything below 80% power will be hard to justify in real work.

A useful metaphor?

Sometimes I like to think of science as a march towards a destination.

Actually, I suppose it's an infinitely long march towards an ever-receding destination, but let's leave the philosophy out of it for a moment.

Suppose, for example, that we're trying to make a meaningful change in the world, perhaps to treat an infection.

What we're trying to do is related to where we are in the March of Science.

Early vs. Late in the March of Science

In **early** work, we're focused more on discovery than making final decisions.

- We don't have a lot of past experience, so we bring little relevant data to the table.
- We're (often) most concerned about discovering new possibilities, and we don't have a very clear sense of where to go next.
- We're (often) less concerned about false starts than we are about missed opportunities.

In **late** work, we're focused more on making a decision about how to treat.

- We have a fair amount of relevant history to draw on, sometimes quite detailed.
- We're more concerned about testing the limits of our current knowledge than we are about missing opportunities to consider a new pathway.
- We're often concerned about doing harm if we implement the strategy that looks most promising.

How does this relate to power and significance?

If we treat our sample size and study design as fixed strategies, then there is a tradeoff between:

- reducing α, the rate of Type I error (increasing our confidence) and
- reducing β, the rate of Type II error (increasing our power)

Suppose we are testing a new treatment for some condition.

- A Type I error means we conclude this treatment is helpful, when it actually isn't.
- A Type II error means we conclude this treatment is not helpful, when it actually is.

Early Work: Power and Sample Size

In early work, we are searching for treatments of promise, and our initial study will inevitably not be the last word on the subject, but rather will be followed up by confirmatory studies. In such a setting, it is often the case that:

- We're not so concerned about getting results that cause us to continue to explore a treatment that doesn't actually do what we need it to do.
- We're really concerned about ruling out a treatment that is promising before we should.

This implies we should prioritize reducing Type II error rates (we want more power to detect small but real effects, even if this means we will occasionally identify something as promising when it isn't.)

This means setting lower confidence levels and higher power levels, potentially, than the standard 95% confidence and 80% power.

Late Work: Power and Sample Size

In late work, we have already identified promising treatments, and we are trying to confirm those results. The current study may actually be the last word on the subject, and we want to be sure we do no harm.

- We're very concerned about getting results that cause us to continue to explore a treatment that doesn't actually do what we need it to do.
- We're less concerned about ruling out a treatment that is promising but doesn't actually work.

This implies we should prioritize reducing Type I error rates (we want greater confidence, even at the expense of power, that the effect we claim based on past data holds up.)

This kind of confirmatory work is usually well suited to studies set up with higher confidence (perhaps 95% or 99% or more) and lower power (80% is the minimum I would recommend) against reasonable alternatives.

Conclusions

1. If you're early in the March of Science (perhaps just one pilot study has been done) then I would emphasize Type II error (power) more than usual, perhaps pushing required power to 90%, at least for a reasonably substantial "minimum scientifically important difference" δ.
2. If you're late in the March of Science (perhaps confirming the results of multiple prior studies) then I would be happier with 80% power and higher levels of confidence.
3. If you're in the middle, trading off Type I and Type II error is worth some thought, but I'd never recommend being under 80% power for an effect that matters.
4. If it's feasible to run a study large enough to have strong performance on both α and β, that's obviously ideal. Typically that doesn't happen in early work.

Multiple Regression with the `dm1` data

The dm1 data: Four Variables (+ Subject)

Suppose we want to consider predicting the a1c values of 500 diabetes subjects now, based on these three predictors:

- a1c_old: subject's Hemoglobin A1c (in %) two years ago
- age: subject's age in years
- income: median income of subject's home neighborhood (3 categories)

```
dm1 <- readRDS("data/dm1.Rds")

head(dm1, 3)
```

```
# A tibble: 3 x 5
    a1c a1c_old   age income          subject
  <dbl>   <dbl> <dbl> <fct>           <chr>
1   6.3    11.4    62 Higher_than_50K S-001
2  11      16.3    54 Between_30-50K  S-002
3   8.7    10.7    47 <NA>            S-003
```

Summarizing the dm1 tibble

```
summary(dm1)
```

```
      a1c            a1c_old              age       
 Min.   : 4.300   Min.   : 4.200   Min.   :31.00  
 1st Qu.: 6.500   1st Qu.: 6.500   1st Qu.:49.00  
 Median : 7.300   Median : 7.300   Median :56.00  
 Mean   : 7.898   Mean   : 7.693   Mean   :55.41  
 3rd Qu.: 8.600   3rd Qu.: 8.300   3rd Qu.:62.00  
 Max.   :16.700   Max.   :16.300   Max.   :70.00  
 NA's   :4        NA's   :15                      
              income       subject         
 Higher_than_50K:123   Length:500         
 Between_30-50K :194   Class :character  
 Below_30K      :178   Mode  :character  
 NA's           :  5                      
```

What roles will these variables play?

`a1c` is our outcome, which we'll predict using three models . . .

1. Model 1: Use `a1c_old` alone to predict `a1c`
2. Model 2: Use `a1c_old` and `age` together to predict `a1c`
3. Model 3: Use `a1c_old`, `age`, and `income` together to predict `a1c`

df_stats to get favstats on multiple quantities

Suppose we want favstats results on all 3 quantitative variables.

```
dm1 %>%
  mosaicCore::df_stats(~ a1c + a1c_old + age) %>%
  rename(na = missing) %>% kable(dig = 2)
```

response	min	Q1	median	Q3	max	mean	sd	n	na
a1c	4.3	6.5	7.3	8.6	16.7	7.90	2.06	496	4
a1c_old	4.2	6.5	7.3	8.3	16.3	7.69	1.75	485	15
age	31.0	49.0	56.0	62.0	70.0	55.41	9.02	500	0

- The `df_stats` function is part of the `mosaicCore` package.
- Either use `library(mosaic)` to make `df_stats()` available, or specify it with `mosaicCore::df_stats()`.

What will we do about missing data?

```
dm1 %>%
  summarize(across(everything(), ~ sum(is.na(.)))) %>%
  kable()
```

a1c	a1c_old	age	income	subject
4	15	0	5	0

- We're missing 4 values of `a1c`, our outcome
- and 15 values of `a1c_old`, a predictor (Models 1-3)
- and 5 values of `income`, another predictor (Model 3)

But what if we have other questions, like:

- How many observations are missing at least one of these variables?
- How many subjects (cases) are missing multiple variables?

Counting missingness by subject with naniar

The naniar package provides several useful functions for identifying and summarizing missingness which work within a tidy workflow.

miss_case_table() for instance, provides a summary table describing the number of subjects missing 0, 1, 2, ... of the variables in our tibble.

```
miss_case_table(dm1)
```

```
# A tibble: 3 x 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0     479      95.8
2              1      18       3.6
3              2       3       0.6
```

So, there are 18 subjects missing one variable, and 3 missing two.

Can we identify these cases?

miss_case_summary lists missingness for each subject

```
miss_case_summary(dm1)
```

```
# A tibble: 500 x 3
    case n_miss pct_miss
   <int>  <int>    <dbl>
 1   280      2       40
 2   319      2       40
 3   488      2       40
 4     3      1       20
 5     7      1       20
 6    16      1       20
 7    20      1       20
 8    41      1       20
 9    49      1       20
10    67      1       20
# ... with 490 more rows
```

Can we summarize missingness by variable?

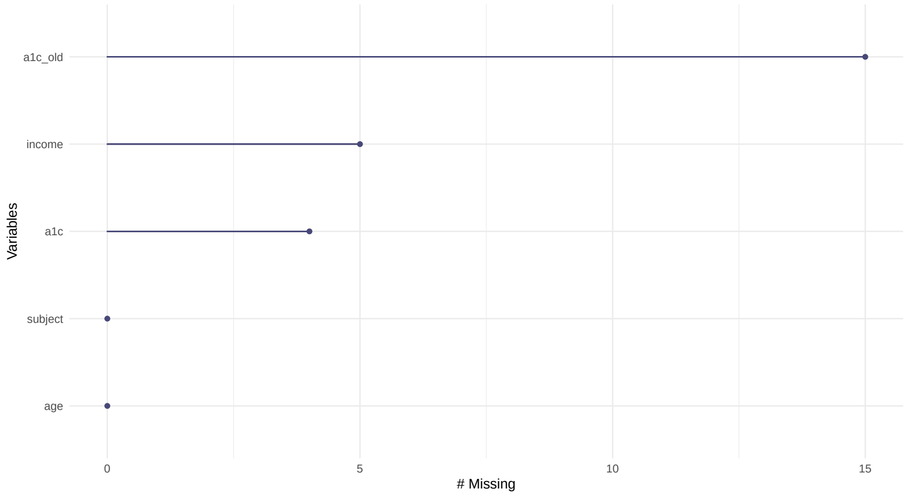
```
miss_var_summary(dm1)
```

```
# A tibble: 5 x 3
  variable n_miss pct_miss
  <chr>     <int>    <dbl>
1 a1c_old      15      3
2 income        5      1
3 a1c           4      0.8
4 age           0      0
5 subject       0      0
```

There's a `miss_var_table()` function, too, if that's useful.

naniar also has helpers for plots

```r
gg_miss_var(dm1)
```



Option 1: Complete Cases Only

We might assume that all of our missing values are Missing Completely At Random (MCAR) and thus that we can safely drop all observations with missing data from our data set.

```r
dm1_cc <- dm1 %>% filter(complete.cases(.))


nrow(dm1)
```

```
[1] 500
```

```r
nrow(dm1_cc)
```

```
[1] 479
```

- For today, I want to use the same observations in each of my 3 models.
- So we would drop 21 subjects, and fit models with the 479 subjects who have complete data on all four variables.

Simple Imputation with the `simputation` package

Option 2: Simple Imputation

Suppose I don't want to impute the outcome. I think people missing my outcome shouldn't be included in my models.

- We'll drop the 4 observations missing `a1c` from our data set.

Perhaps I'd be OK with assuming the missing values of `income` or `a1c_old` are MAR (so that we could use variables in our data to predict them.)

- This would allow us to use imputation methods to "fill in" or "impute" missing predictor values so that we can still use all of the other 496 subjects in our models.
- The `simputation` package provides a straightforward method to do this, while maintaining a tidy workflow.
- There are dangers in assuming everything is MCAR, so this looks helpful (MAR is a lesser assumption) but it introduces the issue of "creating" data where it didn't exist.

Simple Imputation of Missing `a1c_old` Values

We could use a robust linear model method to impute our quantitative `a1c_old` values on the basis of age, which is missing no observations in common with `a1c_old` (in fact, age is missing no observations.)

```
tempA <- impute_rlm(dm1, a1c_old ~ age)

tempA %>% miss_var_summary()
```

```
# A tibble: 5 x 3
  variable n_miss pct_miss
  <chr>     <int>    <dbl>
1 income        5      1
2 a1c           4      0.8
3 a1c_old       0      0
4 age           0      0
5 subject       0      0
```

Simple Imputation of Missing `income` Values

We could use a decision tree (CART) method to impute our missing categorical `income` values, also on the basis of `age`.

```
tempB <- impute_cart(dm1, income ~ age)

tempB %>% miss_var_summary()
```

```
# A tibble: 5 x 3
  variable n_miss pct_miss
  <chr>     <int>    <dbl>
1 a1c_old      15      3
2 a1c           4      0.8
3 age           0      0
4 income        0      0
5 subject       0      0
```

Chaining our Simple Imputations

Or we could put all of our imputations together in a chain.

- In 431, I encourage you to try `rlm` for imputing quantitative variables, and `cart` for categorical variables.
- Were I imputing a binary categorical variable, I would present it as a factor to `impute_cart`.

```
dm1_imp <- dm1 %>%
  filter(complete.cases(a1c, subject)) %>%
  impute_rlm(a1c_old ~ age) %>%
  impute_cart(income ~ age + a1c_old)
```

- I imputed `a1c_old` using `age` and then imputed `income` using both `age` and `a1c_old`.

What is the result?

Summary of imputed tibble

`dm1_imp` has 496 observations (since we dropped the 4 subjects with missing `a1c`: our *outcome*) but no missing values left.

```
dm1_imp %>% summary()
```

```
      a1c             a1c_old            age       
 Min.   : 4.300   Min.   : 4.200   Min.   :31.00  
 1st Qu.: 6.500   1st Qu.: 6.500   1st Qu.:49.00  
 Median : 7.300   Median : 7.300   Median :56.00  
 Mean   : 7.898   Mean   : 7.691   Mean   :55.35  
 3rd Qu.: 8.600   3rd Qu.: 8.300   3rd Qu.:62.00  
 Max.   :16.700   Max.   :16.300   Max.   :70.00  
              income        subject         
 Higher_than_50K:121   Length:496          
 Between_30-50K :193   Class :character  
 Below_30K      :182   Mode  :character  
```

Two approaches for dealing with missing data

1. We could assume MCAR for all variables, and then work with the complete cases (n = 479) in `dm1_cc`.
2. We could assume MAR for the predictors, and work with the simply imputed (n = 496) in `dm1_imp`

Neither of these, as it turns out, will be 100% satisfactory, but for now, we'll compare the impact of these two approaches on the results of our models.

OK. We'll do the complete case analysis now, and return to the imputed data later.

How will we decide which of the models is "best"?

Our goal is accurate prediction of `a1c` values.

Which of these models gives us the "best" result?

1. Model 1: Use `a1c_old` alone to predict `a1c`
2. Model 2: Use `a1c_old` and age together to predict `a1c`
3. Model 3: Use `a1c_old`, age, and `income` together to predict `a1c`

and does our answer change depending on whether we start our work with the complete cases (`dm1_cc`: n = 479) or our simply imputed data (`dm1_imp`: n = 496)?

How shall we be guided by our data?

It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience. (A. Einstein)

- often this is reduced to "make everything as simple as possible but no simpler"

Entities should not be multiplied without necessity. (Occam's razor)

- often this is reduced to "the simplest solution is most likely the right one"

George Box's aphorisms

On Parsimony: Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.

On Worrying Selectively: Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad.

- and, the most familiar version...

  ... all models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind.

431 approach: Which model is “most useful”?

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.
5. 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.)
6. Select a “final” model for use based on the evidence in steps 3, 4 and especially 5.

Split the data into a model development (training) sample of about 70-80% of the observations, and a model test (holdout) sample, containing the remaining observations.

Partitioning the 479 Complete Cases

- We'll select a random sample (without replacement) of 70% of the data (70-80% is customary) for model training.
- We'll hold out the remaining 30% for model testing, using `anti_join()` to identify all `dm1_cc` subjects not in `dm1_cc_train`.

```
set.seed(20211026)

dm1_cc_train <- dm1_cc %>%
  slice_sample(prop = 0.7, replace = FALSE)

dm1_cc_test <-
  anti_join(dm1_cc, dm1_cc_train, by = "subject")

c(nrow(dm1_cc_train), nrow(dm1_cc_test), nrow(dm1_cc))
```

```
[1] 335 144 479
```

Develop candidate models using the development sample.

A look at the outcome (a1c) distribution

We'll study the outcome variable (a1c) in the development sample, to consider whether a transformation might be in order.

I did a little fancy work with the code (continues next slide)...

```r
p1 <- ggplot(dm1_cc_train, aes(x = a1c)) +
  geom_histogram(binwidth = 0.5,
                 fill = "slateblue", col = "white")

p2 <- ggplot(dm1_cc_train, aes(sample = a1c)) +
  geom_qq(col = "slateblue") + geom_qq_line(col = "red")

p3 <- ggplot(dm1_cc_train, aes(x = "", y = a1c)) +
  geom_violin(fill = "slateblue", alpha = 0.3) +
  geom_boxplot(fill = "slateblue", width = 0.3,
               outlier.color = "red") +
  labs(x = "") + coord_flip()
```

A look at the outcome (a1c) distribution

Putting the plots together, and titling them meaningfully...

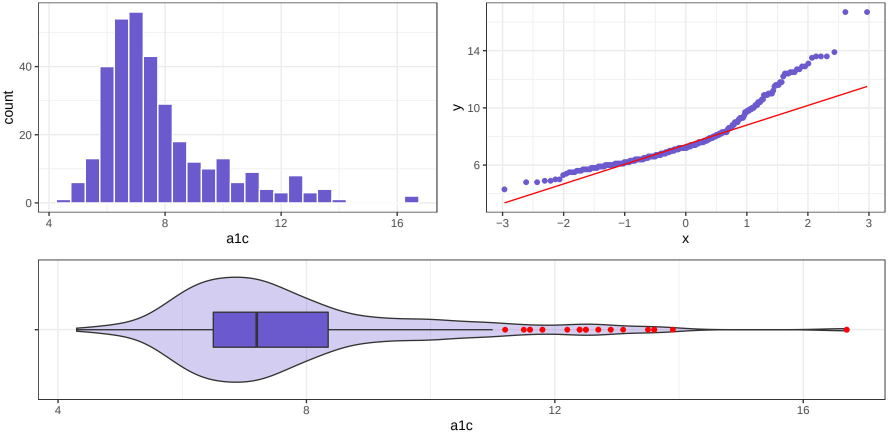
```r
p1 + p2 - p3 +
  plot_layout(ncol = 1, height = c(3, 2)) +
  plot_annotation(title = "Hemoglobin A1c values (%)",
        subtitle = paste0("Model Development Sample: ",
                          nrow(dm1_cc_train),
                          " adults with diabetes"))
```

Result on the next slide...

Outcome (a1c): Model Development Sample

Hemoglobin A1c values (%)

Model Development Sample: 335 adults with diabetes



Why Transform the Outcome?

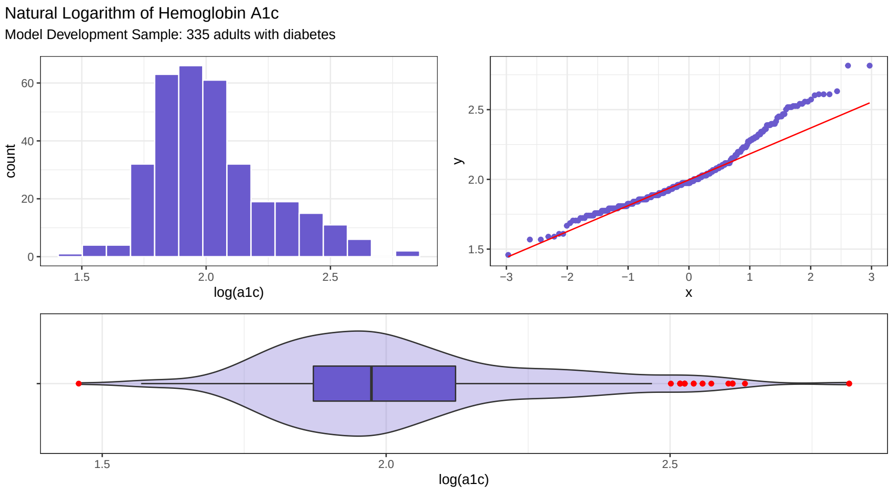
We want to try to identify a good transformation for the conditional distribution of the outcome, given the predictors, in an attempt to make the linear regression assumptions of linearity, Normality and constant variance more appropriate.

Ladder of Especially Useful (and often interpretable) transformations

Transformation	y^2	y	$\sqrt{y}$	log(y)	$1/y$	$1/y^2$
λ	2	1	0.5	0	-1	-2

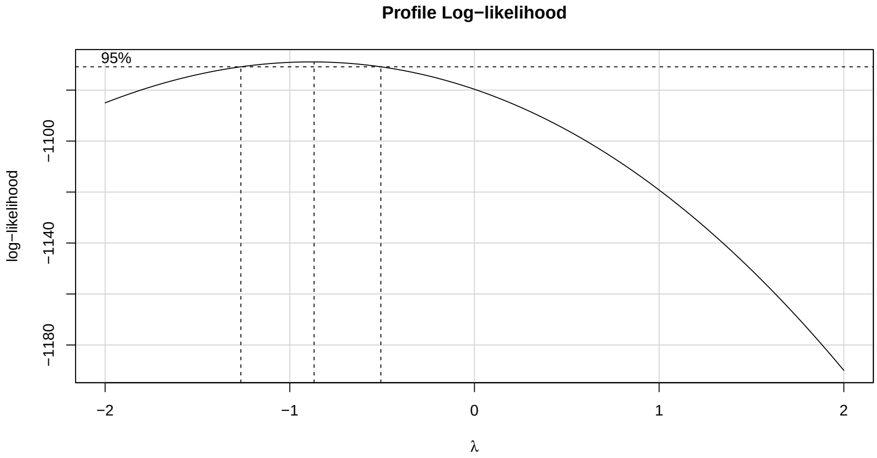
- We see some sign of right skew in the `a1c` data. Let's try a log transformation.

Consider a log transformation?



Using Box-Cox to help select a transformation?

```
mod_0 <- lm(a1c ~ a1c_old + age + income,
            data = dm1_cc_train)
boxCox(mod_0)
```

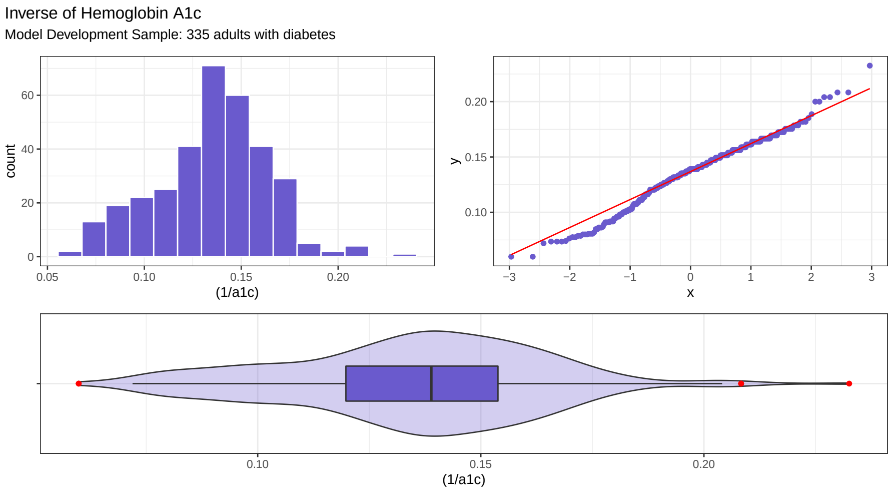


Using Box-Cox to help select a transformation?

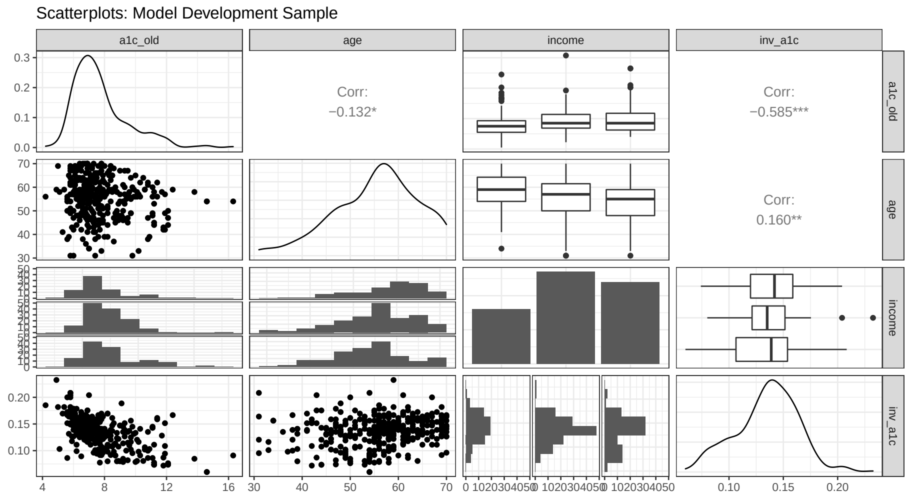
```
summary(powerTransform(mod_0))
```

```
bcPower Transformation to Normality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1   -0.8838          -1      -1.2628      -0.5048

Likelihood ratio test that transformation parameter is equal t
 (log transformation)
                           LRT df       pval
LR test, lambda = (0) 21.48071  1 3.5741e-06

Likelihood ratio test that no transformation is needed
                           LRT df       pval
LR test, lambda = (1) 100.5543  1 < 2.22e-16
```

Consider the inverse?



Scatterplot Matrix (code on next slide)



Scatterplot Matrix (Code)

```
dm1_cc_train %>%
  mutate(inv_a1c = 1/a1c) %>%
  select(a1c_old, age, income, inv_a1c) %>%
  ggpairs(.,
    title = "Scatterplots: Model Development Sample",
    lower = list(combo = wrap("facethist", bins = 10)))
```

Note that `ggpairs` comes from the `GGally` package.

- If you have more than 4-5 predictors, it's usually necessary to split this up into two or more scatterplot matrices, each of which should include the outcome.
- I'd always put the outcome last in my selection here. That way, the bottom row will show the most important scatterplots, with the outcome on the Y axis, and each predictor, in turn on the X.

Three Regression Models We'll Fit

- Remember we're using the model development sample here.
- Let's work with the (1/a1c) transformation.

```
mod_1 <- lm((1/a1c) ~ a1c_old, data = dm1_cc_train)

mod_2 <- lm((1/a1c) ~ a1c_old + age, data = dm1_cc_train)

mod_3 <- lm((1/a1c) ~ a1c_old + age + income,
            data = dm1_cc_train)
```

Assess the quality of fit for candidate models within the development sample.

Tidied coefficients (mod_1)

```
tidy_m1 <- tidy(mod_1, conf.int = TRUE, conf.level = 0.95)

tidy_m1 %>%
  select(term, estimate, std.error, p.value,
         conf.low, conf.high) %>%
  knitr::kable(digits = 4)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.2080	0.0057	0	0.1968	0.2192
a1c_old	-0.0094	0.0007	0	-0.0108	-0.0080

The Regression Equation (`mod_1`)

Use the `equatiomatic` package to help here. Note the use of **results = 'asis'** in the code chunk name.

```
extract_eq(mod_1, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE)
```

$$\widehat{(1/a1c)} = 0.208 - 0.0094(a1c\_old) \tag{1}$$

Summary of Fit Quality (mod_1)

```
glance(mod_1) %>%
  mutate(name = "mod_1") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_1	0.342	0.34	0.023	-1565	-1553

Tidied coefficients (mod_2)

```
tidy_m2 <- tidy(mod_2, conf.int = TRUE, conf.level = 0.95)

tidy_m2 %>%
  select(term, estimate, std.error, p.value,
         conf.low, conf.high) %>%
  knitr::kable(digits = 4)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.1913	0.0105	0.0000	0.1707	0.2120
a1c_old	-0.0093	0.0007	0.0000	-0.0107	-0.0078
age	0.0003	0.0001	0.0603	0.0000	0.0006

The Regression Equation (`mod_2`)

Again, we'll use the `equatiomatic` package, with **results = 'asis'**.

```
extract_eq(mod_2, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE)
```

$$\widehat{(1/a1c)} = 0.1913 - 0.0093(a1c\_old) + 3e - 04(age) \tag{2}$$

Summary of Fit Quality (mod_2)

```r
glance(mod_2) %>%
  mutate(name = "mod_2") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_2	0.349	0.345	0.023	-1566	-1551

Tidied coefficients (mod_3)

```
tidy_m3 <- tidy(mod_3, conf.int = TRUE, conf.level = 0.95)

tidy_m3 %>%
  select(term, estimate, se = std.error,
         low = conf.low, high = conf.high, p = p.value) %>%
  knitr::kable(digits = c(4,4,4,4,3))
```

term	estimate	se	low	high	p
(Intercept)	0.1922	0.0110	0.1707	0.214	0.0000
a1c_old	-0.0092	0.0007	-0.0107	-0.008	0.0000
age	0.0003	0.0001	0.0000	0.001	0.0771
incomeBetween_30-50K	0.0002	0.0033	-0.0063	0.007	0.9456
incomeBelow_30K	-0.0016	0.0034	-0.0084	0.005	0.6384

The Regression Equation (`mod_3`)

Again, we'll use the `equatiomatic` package.

```
extract_eq(mod_3, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE, wrap = TRUE, terms_per_line = 2)
```

$$\begin{aligned}\widehat{(1/a1c)} &= 0.1922 - 0.0092(a1c\_old)\ + \\ &\quad 3e-04(age) + 2e-04(income_{Between\_30-50K})\ - \\ &\quad 0.0016(income_{Below\_30K})\end{aligned} \qquad (3)$$

Summary of Fit Quality (mod_3)

```
glance(mod_3) %>%
  mutate(name = "mod_3") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_3	0.35	0.342	0.023	-1563	-1540

Could we have fit other predictor sets?

Perhaps an automated procedure, like stepwise regression, would suggest a better alternative?

- Three predictor candidates, so we could have used any of these predictor sets:
- `a1c_old` alone (our `mod_1`)
- `age` alone
- `income` alone
- `a1c_old` and `age` (our `mod_2`)
- `a1c_old` and `income`
- `age` and `income`
- `a1c_old`, `age` and `income` (our `mod_3`)

```
step(mod_3)
```

Stepwise Regression Results (part 1 of 2)

We'll try backwards elimination, where we let R's `step` function start with the full model (`mod_3`) including all three predictors, and then remove the predictor whose removal causes the largest drop in AIC, until we reach a point where eliminating another predictor will not improve the AIC.

- Remember the smaller (more negative, here) the AIC, the better.

```
step(mod_3)
```

```
Start:  AIC=-2515.5
(1/a1c) ~ a1c_old + age + income

          Df Sum of Sq     RSS     AIC
- income   2  0.000236 0.17847 -2519.1
<none>                 0.17823 -2515.5
- age      1  0.001698 0.17993 -2514.3
- a1c_old  1  0.086785 0.26502 -2384.6
```

Stepwise Regression Results (part 2 of 2)

```
Step:  AIC=-2519.05
(1/a1c) ~ a1c_old + age

          Df Sum of Sq     RSS     AIC
<none>                 0.17847 -2519.1
- age      1   0.00191 0.18038 -2517.5
- a1c_old  1   0.08858 0.26705 -2386.0

Call:
lm(formula = (1/a1c) ~ a1c_old + age, data = dm1_cc_train)

Coefficients:
(Intercept)      a1c_old          age
  0.1913429   -0.0092565    0.0002749
```

and we wind up here with just our `mod_2`.

An Important Point

- There is an **enormous** amount of evidence that variable selection causes severe problems in estimation and inference.
- Stepwise regression, in particular, is an egregiously bad choice.
- Disappointingly, there really isn't a good choice. The task itself just isn't one we can do well in a uniform way across all of the different types of regression models we'll build.

More on this in 432.

Comparing Summary Measures of Fit

in the development (model training) sample. . .

```
bind_rows(glance(mod_1), glance(mod_2), glance(mod_3)) %>%
  mutate(model_vars = c("1_a1c_old", "2_+age", "3_+income")) %
  select(model_vars, r2 = r.squared, adj_r2 = adj.r.squared,
         sigma, AIC, BIC, df, df_res = df.residual) %>%
  kable(digits = c(0, 4, 4, 5, 1, 0, 0, 0))
```

model_vars	r2	adj_r2	sigma	AIC	BIC	df	df_res
1_a1c_old	0.3418	0.3398	0.02327	-1564.8	-1553	1	333
2_+age	0.3487	0.3448	0.02319	-1566.4	-1551	2	332
3_+income	0.3496	0.3417	0.02324	-1562.8	-1540	4	330

In the data we used to build the model, these are our results.

Which Model Looks Best In-Sample?

For each of these summaries, which model looks best in the training sample?

model	vars	r2	adj_r2	sigma	AIC	BIC
mod_1	a1c_old	0.3418	0.3398	0.02327	-1564.8	-1553
mod_2	+ age	0.3487	0.3448	0.02319	-1566.4	-1551
mod_3	+ income	0.3496	0.3417	0.02324	-1562.8	-1540

- By r^2, the largest model will always look best (raw r^2 is greedy)
- Adjusted r^2 penalizes for lack of parsimony. Model 2 looks best there.
- For σ, AIC and BIC, we want small (more negative) values.
    - Model 2 looks best by σ and AIC, as well.
    - Model 1 looks a little better than Model 2 by BIC.
- Overall, what should we conclude about in-sample fit quality?

Check adherence to regression assumptions in the development sample.

Using `augment` to add fits, residuals, etc.

```r
aug1 <- augment(mod_1, data = dm1_cc_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome
```

`aug1` includes all variables in `dm_cc_train` to which we've added:

- `inv_a1c` = 1/`a1c`, the transformed outcome that `mod_1` predicts
- `.fitted` = fitted (predicted) values of 1/`a1c`
- `.resid` = residual (observed outcome - fitted outcome) values, so that larger values (positive or negative) mean poorer fit points
- `.std.resid` = standardized residuals (residuals scaled to SD = 1, remember that the residual mean is already 0)
- `.hat` statistic = measures *leverage* (larger values of `.hat` indicate unusual combinations of predictor values)
- `.cooksd` = Cook's distance (or Cook's d), a measure of the subject's *influence* on the model (larger Cook's d values indicate that removing the point will materially change the model's coefficients)
- plus `.sigma` = estimated σ if this point is dropped from the model

augment results for the first 2 subjects

```
aug1 %>% select(subject, a1c:income, inv_a1c) %>%
  tail(2) %>% kable(dig = 3)
```

subject	a1c	a1c_old	age	income	inv_a1c
S-060	7.0	7.3	45	Higher_than_50K	0.143
S-116	6.4	6.5	63	Between_30-50K	0.156

```
aug1 %>% select(subject, .fitted:.cooksd) %>%
  tail(2) %>% kable(dig = 3)
```

subject	.fitted	.resid	.hat	.sigma	.cooksd
S-060	0.139	0.004	0.003	0.023	0
S-116	0.147	0.010	0.004	0.023	0

augment for models mod_2 and mod_3

We need the augment results for our other two models: mod_2 and mod_3.

```
aug2 <- augment(mod_2, data = dm1_cc_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome
```

```
aug3 <- augment(mod_3, data = dm1_cc_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome
```

Checking Regression Assumptions

Four key assumptions we need to think about:

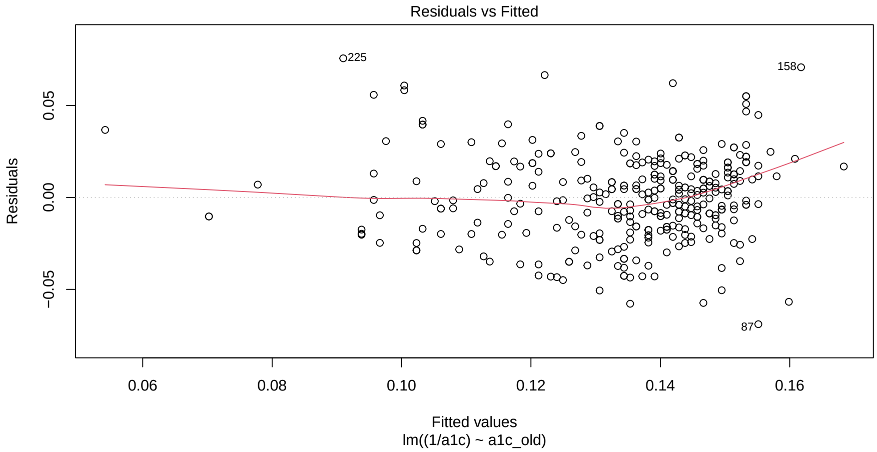
1. Linearity
2. Constant Variance (Homoscedasticity)
3. Normality
4. Independence

How do we assess 1, 2, and 3? Residual plots.

There are five automated ones that we could obtain using `plot(mod_1)`...

Residuals vs. Fitted Values Plot (Model `mod_1`)

```r
plot(mod_1, which = 1)
```



Which points are highlighted in that plot?

Note that the points labeled 87, 158 and 225 are the 87th, 158th and 225th rows in our `dm1_cc_train` data file, or, equivalently, in our `aug1` file.

```
aug1 %>% slice(c(87, 158, 225)) %>% select(a1c:.resid)
```

```
# A tibble: 3 x 7
    a1c a1c_old   age income     subject .fitted  .resid
  <dbl>   <dbl> <dbl> <fct>      <chr>     <dbl>   <dbl>
1  11.6     5.6    54 Below_30K S-168    0.155  -0.0689
2   4.3     4.9    59 Between_~ S-386    0.162   0.0708
3   6      12.4    59 Below_30K S-105    0.0910  0.0757
```

These are subjects `S-168`, `S-386`, and `S-105`, respectively.

Another way to confirm who the plot is identifying

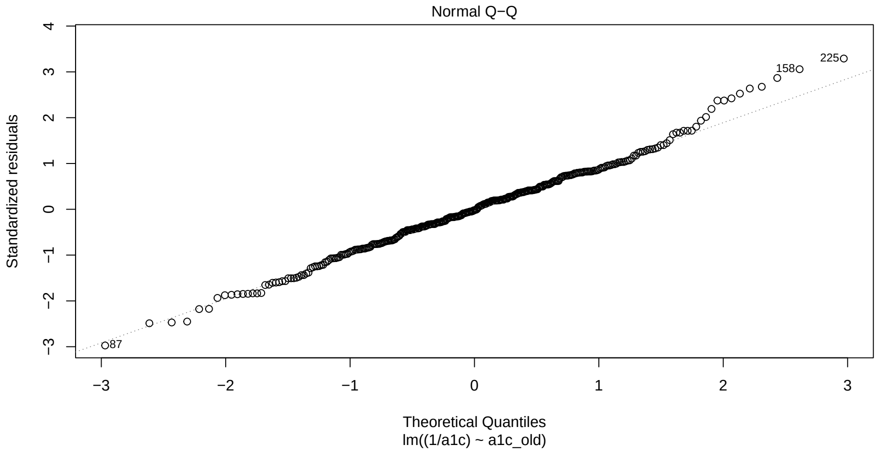
As mentioned, we think the identifiers (87, 158 and 225) of the points with the largest residual (in absolute value) describe subjects `S-168`, `S-386`, and `S-105`, respectively. Does this make sense?

```
aug1 %>% select(subject, .resid) %>%
  arrange(desc(abs(.resid))) %>% head()
```

```
# A tibble: 6 x 2
  subject  .resid
  <chr>     <dbl>
1 S-105    0.0757
2 S-386    0.0708
3 S-168   -0.0689
4 S-071    0.0666
5 S-052    0.0621
6 S-341    0.0609
```

Normal Q-Q of Standardized Residuals (mod_1)

```r
plot(mod_1, which = 2)
```



Are the outliers we see there completely out of line?

```
nrow(aug1)
```

```
[1] 335
```

```
aug1 %>% select(subject, .std.resid) %>%
  arrange(desc(abs(.std.resid)))
```

```
# A tibble: 335 x 2
   subject .std.resid
   <chr>        <dbl>
 1 S-105         3.29
 2 S-386         3.06
 3 S-168        -2.97
 4 S-071         2.87
 5 S-052         2.67
 6 S-341         2.64
 7 S-001         2.53
 8 S-009        -2.49
```

Is a Z score of 3.34 for the biggest outlier scary here?

```
outlierTest(mod_1)
```

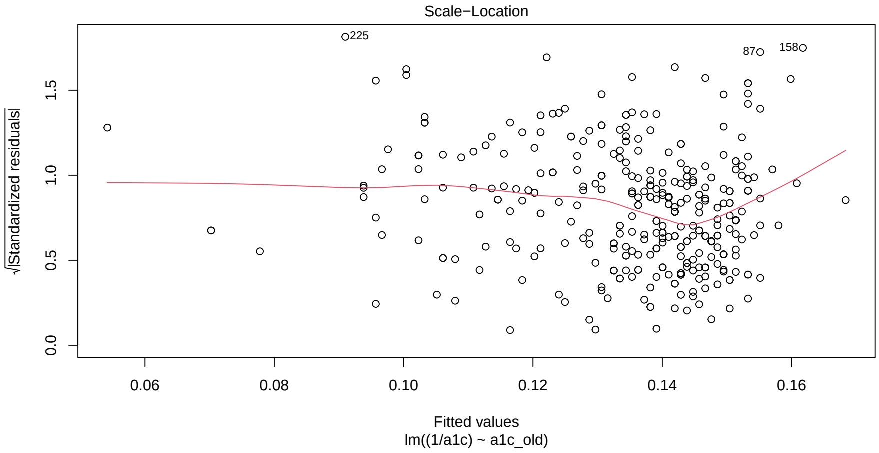
```
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
    rstudent unadjusted p-value Bonferroni p
225 3.340755         0.00093066      0.31177
```

For now, a studentized residual is just another way to standardize the residuals that has some useful properties in this setting.

- There's no indication that having a maximum absolute value of 3.34 in a sample of 335 studentized residuals is a major concern about the assumption of Normality, given the Bonferroni p value of 0.31.

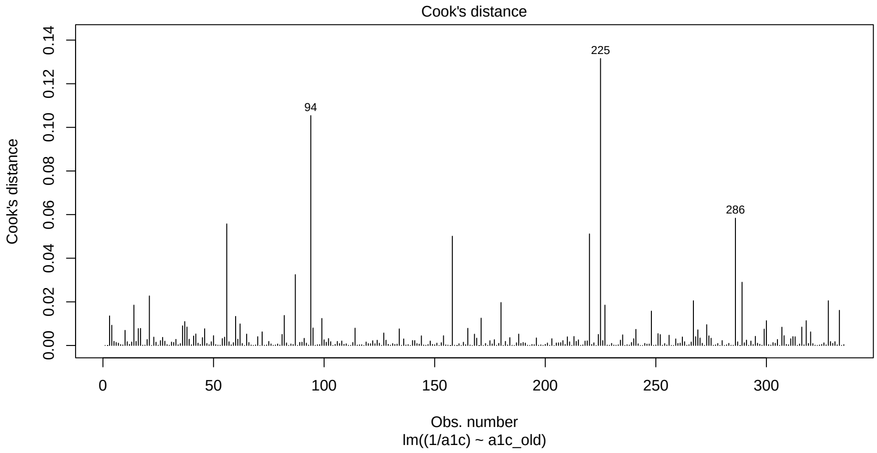
Scale-Location: Check for heteroscedasticity (`mod_1`)

```
plot(mod_1, which = 3)
```



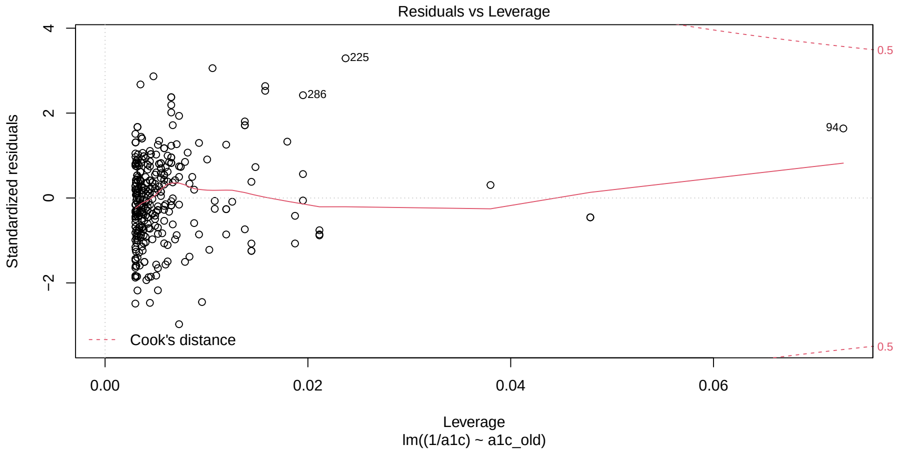
Index plot of Cook's distance for influence (`mod_1`)

```r
plot(mod_1, which = 4)
```



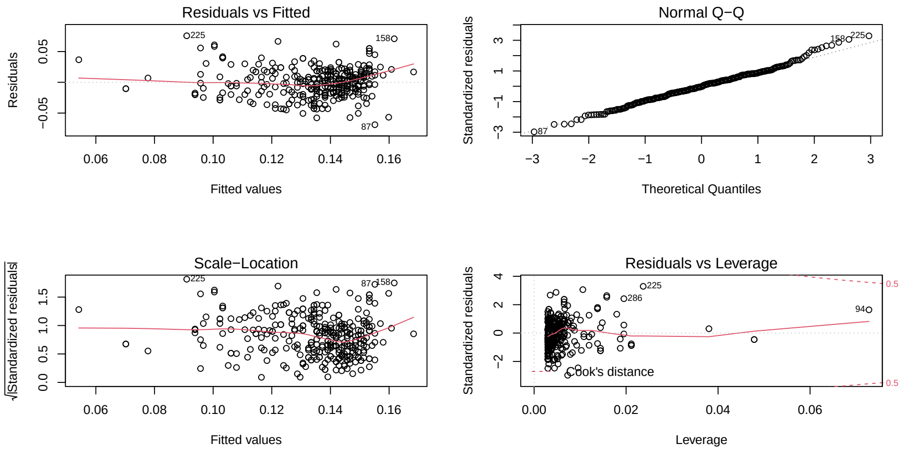
Residuals, Leverage and Influence plot (mod_1)

```
plot(mod_1, which = 5)
```

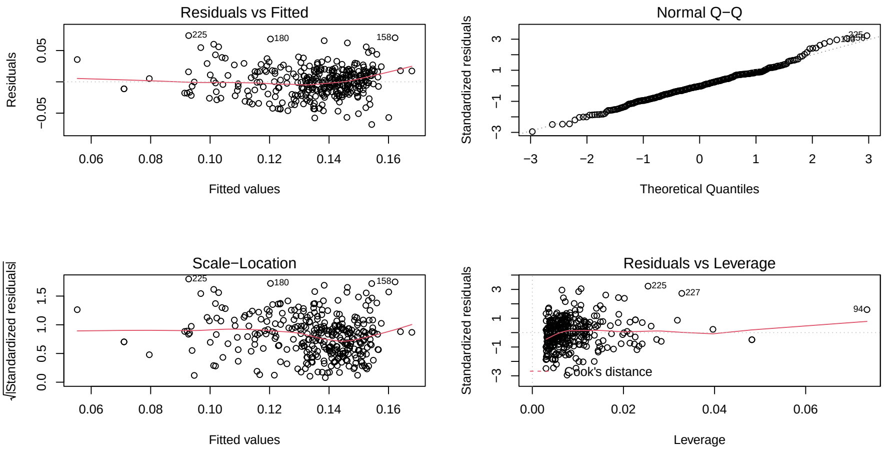


Residual Plots for Model `mod_1`

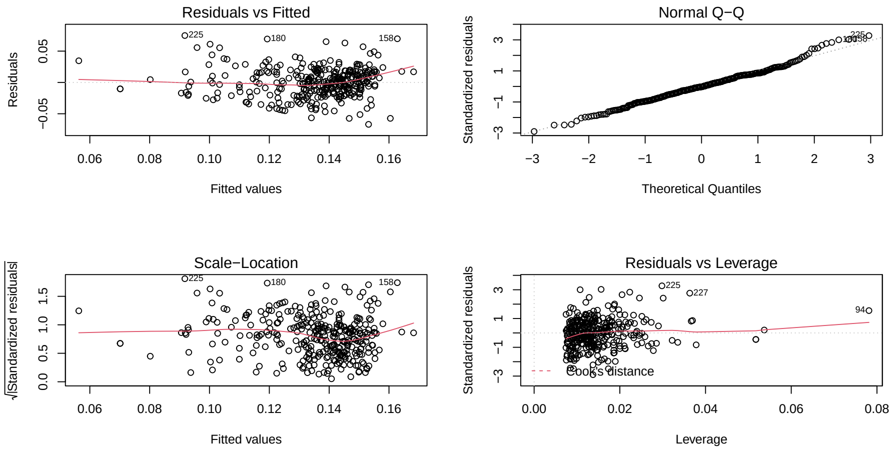
```r
par(mfrow = c(2,2)); plot(mod_1); par(mfrow = c(1,1))
```



Residual Plots for Model `mod_2`



Residual Plots for Model `mod_3`



Is collinearity a serious issue here?

```
car::vif(mod_3)
```

```
            GVIF Df GVIF^(1/(2*Df))
a1c_old 1.031609  1        1.015682
age     1.050249  1        1.024816
income  1.052156  2        1.012791
```

- Collinearity = correlated predictors
- (generalized) Variance Inflation Factor tells us something about how the standard errors of our coefficients are inflated as a result of correlation between predictors.
  - We tend to worry most about VIFs in this output that exceed 5.
  - Remember that the scatterplot matrix didn't suggest any strong correlations between our predictors.

What would we do if we had strong collinearity? Drop a predictor?

Conclusions so far?

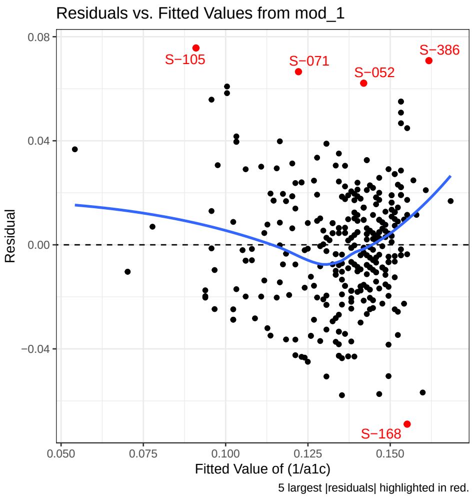
1. In-sample model predictions are about equally accurate for each of the three models. Model 2 looks better in terms of adjusted R^2 and AIC, but model 1 looks better on BIC. There's really not much to choose from there.
2. Residual plots look similarly reasonable for linearity, Normality and constant variance in all three models.

Using `ggplot2` to create these residual plots?

1. Residuals vs. Fitted Values plots are straightforward, with the use of the `augment` function from the `broom` package.
   - We can also plot residuals against individual predictors, if we like.
2. Similarly, plots to assess the Normality of the residuals, like a Normal Q-Q plot, are straightforward, and can use either raw residuals or standardized residuals.
3. The scale-location plot of the square root of the standardized residuals vs. the fitted values is also pretty straightforward.
4. The `augment` function can be used to obtain Cook's distance, standardized residuals and leverage values, so we can mimic both the index plot (of Cook's distance) as well as the residuals vs. leverage plot with Cook's distance contours, if we like.

Demonstrations on the next few slides, followed by the code.

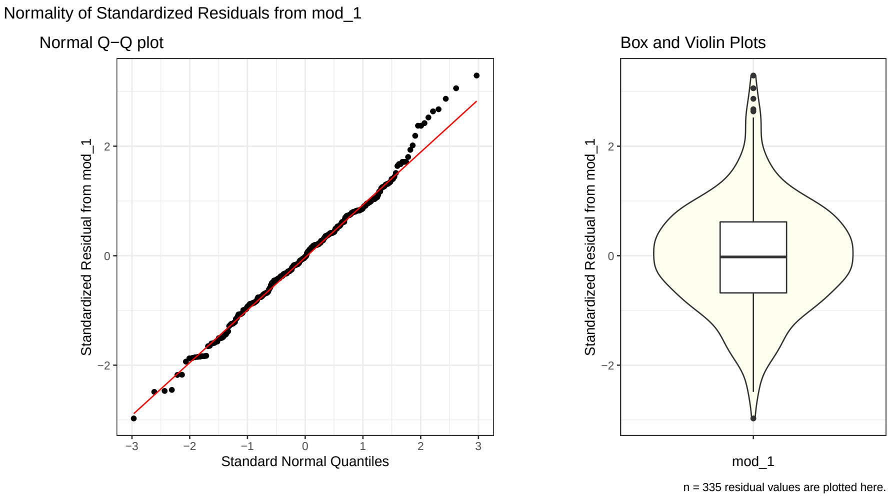
Residuals vs. Fitted Values Plot via `ggplot2`



Code for Residuals vs. Fitted Values

```
ggplot(aug1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_point(data = aug1 %>%
               slice_max(abs(.resid), n = 5),
             col = "red", size = 2) +
  geom_text_repel(data = aug1 %>%
               slice_max(abs(.resid), n = 5),
               aes(label = subject), col = "red") +
  geom_abline(intercept = 0, slope = 0, lty = "dashed") +
  geom_smooth(method = "loess", formula = y ~ x, se = F) +
  labs(title = "Residuals vs. Fitted Values from mod_1",
       caption = "5 largest |residuals| highlighted in red.",
       x = "Fitted Value of (1/a1c)", y = "Residual") +
  theme(aspect.ratio = 1)
```

Normality of Standardized Residuals via `ggplot2`



Code for Normality Checks (1 of 2)

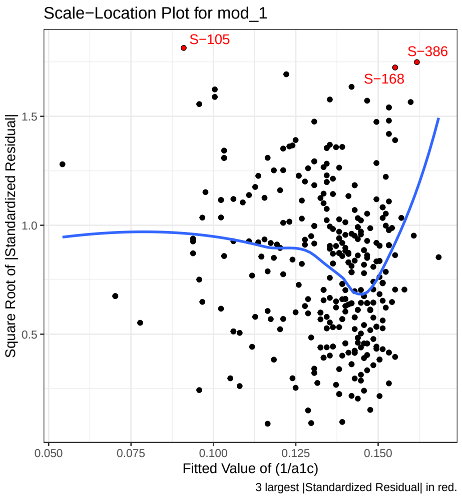
```r
p1 <- ggplot(aug1, aes(sample = .std.resid)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(title = "Normal Q-Q plot",
       y = "Standardized Residual from mod_1",
       x = "Standard Normal Quantiles") +
  theme(aspect.ratio = 1)

p2 <- ggplot(aug1, aes(y = .std.resid, x = "")) +
  geom_violin(fill = "ivory") +
  geom_boxplot(width = 0.3) +
  labs(title = "Box and Violin Plots",
       y = "Standardized Residual from mod_1",
       x = "mod_1")
```

. . . continues on next slide

Code for Normality Checks (2 of 2)

```r
p1 + p2 +
  plot_layout(widths = c(2, 1)) +
  plot_annotation(
    title = "Normality of Standardized Residuals from mod_1",
    caption = paste0("n = ",
                     nrow(aug1 %>% select(.std.resid)),
                     " residual values are plotted here."))
```

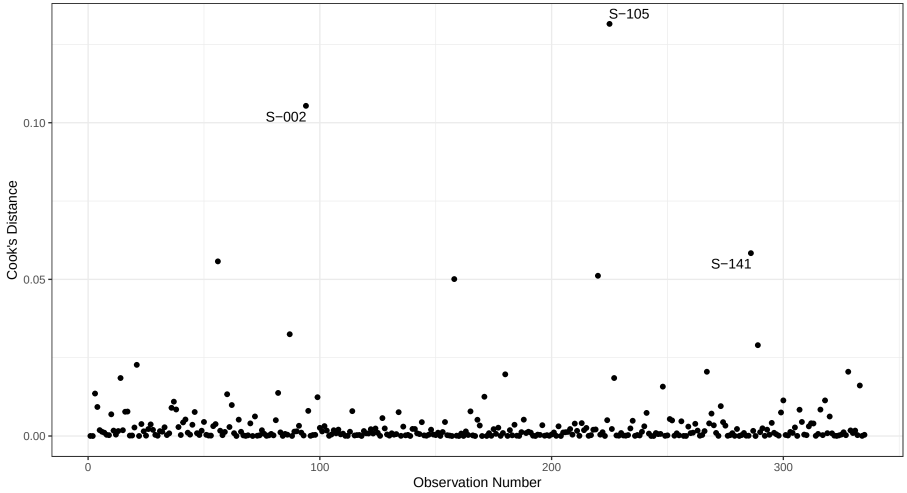
Scale-Location Plot via ggplot2



Code for Scale-Location Plot

```r
ggplot(aug1, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_point(data = aug1 %>%
                 slice_max(sqrt(abs(.std.resid)), n = 3),
             col = "red", size = 1) +
  geom_text_repel(data = aug1 %>%
                 slice_max(sqrt(abs(.std.resid)), n = 3),
                 aes(label = subject), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = F) +
  labs(title = "Scale-Location Plot for mod_1",
       caption = "3 largest |Standardized Residual| in red.",
       x = "Fitted Value of (1/a1c)",
       y = "Square Root of |Standardized Residual|") +
  theme(aspect.ratio = 1)
```

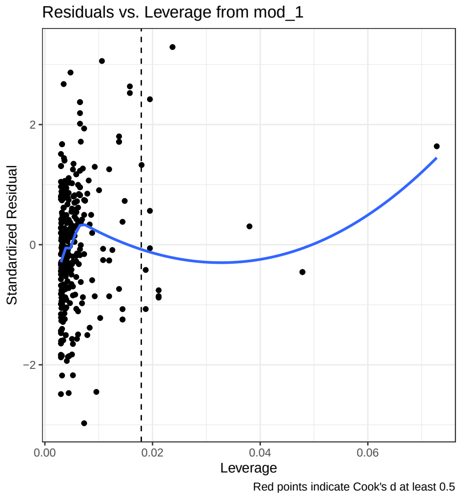
Cook's Distance Index Plot via `ggplot2`



Code for Cook’s Distance Index Plot

```r
aug1_extra <- aug1 %>%
  mutate(obsnum = 1:nrow(aug1 %>% select(.cooksd)))

ggplot(aug1_extra, aes(x = obsnum, y = .cooksd)) +
  geom_point() +
  geom_text_repel(data = aug1_extra %>%
                    slice_max(.cooksd, n = 3),
                    aes(label = subject)) +
  labs(x = "Observation Number",
       y = "Cook's Distance")
```

Residuals vs. Leverage Plot via `ggplot2`



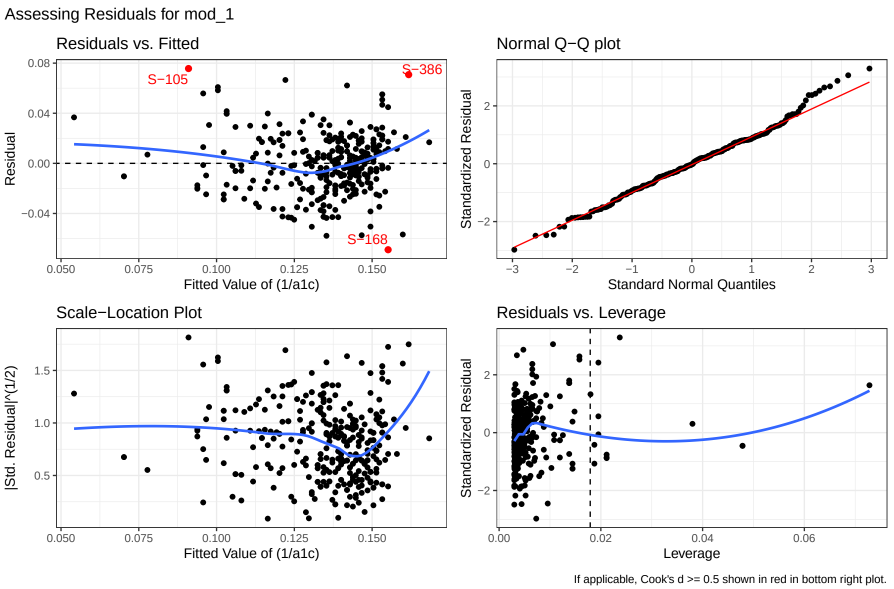
- Points with Cook's d >= 0.5 would be highlighted and in red.
- Points right of the dashed line have high leverage, by one standard.

Code for Residuals vs. Leverage Plot

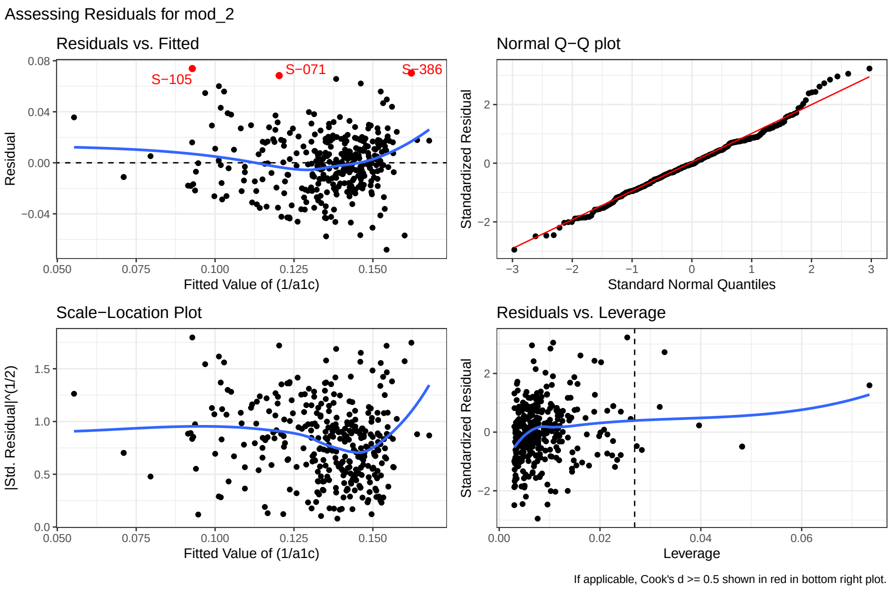
```
ggplot(aug1, aes(x = .hat, y = .std.resid)) +
  geom_point() +
  geom_point(data = aug1 %>% filter(.cooksd >= 0.5),
             col = "red", size = 2) +
  geom_text_repel(data = aug1 %>% filter(.cooksd >= 0.5),
               aes(label = subject), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = F) +
  geom_vline(aes(xintercept = 3*mean(.hat)), lty = "dashed") +
  labs(title = "Residuals vs. Leverage from mod_1",
       caption = "Red points indicate Cook's d at least 0.5",
       x = "Leverage", y = "Standardized Residual") +
  theme(aspect.ratio = 1)
```

- Points with more than 3 times the average leverage are identified as highly leveraged by some people, hence my dashed vertical line.

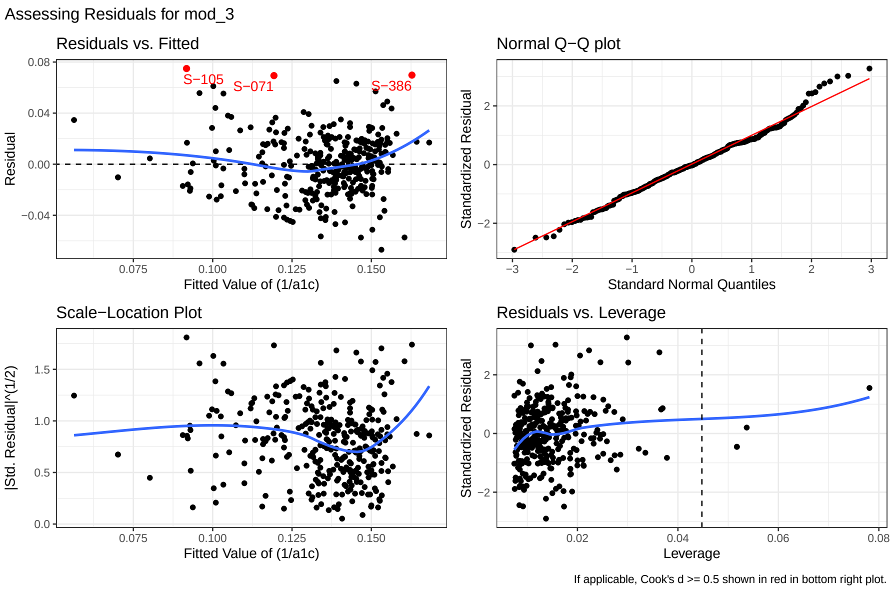
Main 4 Residual Plots for `mod_1` (via `ggplot2`)



Main 4 Residual Plots for `mod_2` (via `ggplot2`)



Main 4 Residual Plots for `mod_3` (via `ggplot2`)



Conclusions so far? (repeating what we said earlier)

1. In-sample model predictions are about equally accurate for each of the three models. Model 2 looks better in terms of adjusted R^2 and AIC, but model 1 looks better on BIC. There's really not much to choose from there.
2. Residual plots look similarly reasonable for linearity, Normality and constant variance in all three models.

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.)

Calculate prediction errors for `mod_1` in test sample

The `augment` function in the `broom` package will create predictions within our new sample, but we want to back-transform these predictions so that they are on the original scale (`a1c`, rather than our transformed regression outcome `1/a1c`). Since the way to back out of the inverse transformation is to take the inverse again, we will take the inverse of the fitted values provided by `augment` and then calculate residuals on the original scale, as follows. . .

```r
test_m1 <- augment(mod_1, newdata = dm1_cc_test) %>%
  mutate(name = "mod_1", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)
```

What does `test_m1` now include?

```r
test_m1 %>%
  select(subject, a1c, fit_a1c, res_a1c, a1c_old,
         age, income) %>%
  head() %>%
  knitr::kable(digits = c(0, 1, 2, 2, 1, 0, 0))
```

subject	a1c	fit_a1c	res_a1c	a1c_old	age	income
S-004	6.5	6.52	-0.02	5.8	53	Below_30K
S-005	6.7	6.73	-0.03	6.3	64	Between_30-50K
S-012	12.2	9.87	2.33	11.3	52	Below_30K
S-014	8.4	7.88	0.52	8.6	44	Between_30-50K
S-015	5.7	6.48	-0.78	5.7	52	Between_30-50K
S-021	11.4	7.60	3.80	8.1	51	Higher_than_50K

Gather test-sample prediction errors for models 2, 3

```
test_m2 <- augment(mod_2, newdata = dm1_cc_test) %>%
  mutate(name = "mod_2", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)

test_m3 <- augment(mod_3, newdata = dm1_cc_test) %>%
  mutate(name = "mod_3", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)
```

Combine test sample results from the three models

```
test_comp <- bind_rows(test_m1, test_m2, test_m3) %>%
  arrange(subject, name)

test_comp %>% select(name, subject, a1c, fit_a1c, res_a1c,
                     a1c_old, age, income) %>%
  slice(1:3, 7:9) %>%
  knitr::kable(digits = c(0, 0, 1, 2, 2, 1, 0, 0))
```

name	subject	a1c	fit_a1c	res_a1c	a1c_old	age	income
mod_1	S-004	6.5	6.52	-0.02	5.8	53	Below_30K
mod_2	S-004	6.5	6.57	-0.07	5.8	53	Below_30K
mod_3	S-004	6.5	6.62	-0.12	5.8	53	Below_30K
mod_1	S-012	12.2	9.87	2.33	11.3	52	Below_30K
mod_2	S-012	12.2	9.90	2.30	11.3	52	Below_30K
mod_3	S-012	12.2	9.99	2.21	11.3	52	Below_30K

What do we do to compare the test-sample errors?

Given this tibble, including predictions and residuals from the three models on our test data, we can now:

1. Visualize the prediction errors from each model.
2. Summarize those errors across each model.
3. Identify the "worst fitting" subject for each model in the test sample.

Visualize the prediction errors

```
ggplot(test_comp, aes(x = res_a1c, fill = name)) +
  geom_histogram(bins = 20, col = "white") +
  facet_grid (name ~ .) + guides(fill = "none")
```

or maybe

```
ggplot(test_comp, aes(x = name, y = res_a1c, fill = name)) +
  geom_violin(alpha = 0.3) +
  geom_boxplot(width = 0.3, outlier.shape = NA) +
  geom_jitter(height = 0, width = 0.1) +
  guides(fill = "none")
```

Test-Sample Prediction Errors

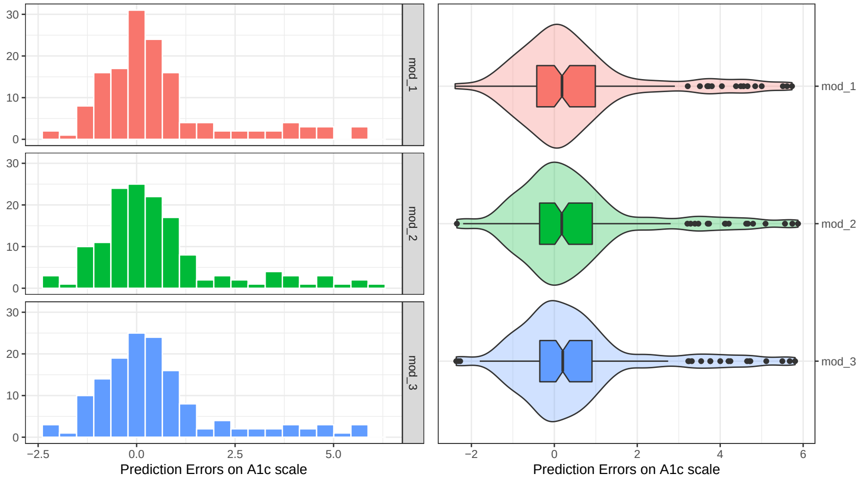


Table Comparing Model Prediction Errors

Calculate the mean absolute prediction error (MAPE), the square root of the mean squared prediction error (RMSPE) and the maximum absolute error across the predictions made by each model. Let's add the median absolute prediction error, too.

```
test_comp %>%
  group_by(name) %>%
  summarize(n = n(),
            MAPE = mean(abs(res_a1c)),
            RMSPE = sqrt(mean(res_a1c^2)),
            max_error = max(abs(res_a1c)),
            median_APE = median(abs(res_a1c))) %>%
  kable(digits = c(0, 0, 4, 3, 2, 3))
```

Table moved to the next slide.

Conclusions from Table of Errors

name	n	mean_APE	RMSPE	max_error	median_APE
mod_1	144	1.1153	1.731	5.73	0.651
mod_2	144	1.1201	1.723	5.88	0.658
mod_3	144	1.1242	1.722	5.81	0.699

- Model `mod_1` has the smallest MAPE (mean APE) and maximum error and median absolute prediction error.
- Model `mod_3` has the smallest root mean squared prediction error (RMSPE).

Identify the largest errors

Identify the subject(s) where that maximum prediction error was made by each model, and the observed and model-fitted values of `a1c` in each case.

```
temp1 <- test_m1 %>%
  filter(abs(res_a1c) == max(abs(res_a1c)))

temp2 <- test_m2 %>%
  filter(abs(res_a1c) == max(abs(res_a1c)))

temp3 <- test_m3 %>%
  filter(abs(res_a1c) == max(abs(res_a1c)))
```

Identify the largest errors (Results)

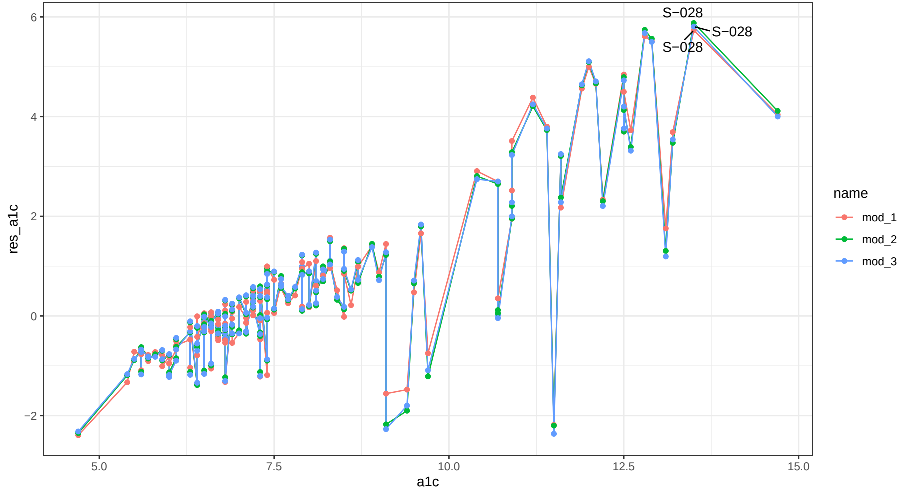
Identify the subject(s) where that maximum prediction error was made by each model, and the observed and model-fitted values of a1c in each case.

```
bind_rows(temp1, temp2, temp3) %>%
  select(subject, name, a1c, fit_a1c, res_a1c)
```

```
# A tibble: 3 x 5
  subject name    a1c fit_a1c res_a1c
  <chr>   <chr> <dbl>   <dbl>   <dbl>
1 S-028   mod_1  13.5    7.77    5.73
2 S-028   mod_2  13.5    7.62    5.88
3 S-028   mod_3  13.5    7.69    5.81
```

Line Plot of the Errors?

Compare the errors that are made at each level of observed A1c?



Code for the Line Plot of the Prediction Errors

```r
ggplot(test_comp, aes(x = a1c, y = res_a1c,
                      group = name)) +
  geom_line(aes(col = name)) +
  geom_point(aes(col = name)) +
  geom_text_repel(data = test_comp %>%
              filter(subject == "S-028"),
              aes(label = subject))
```

What if we ignored S-028 for a moment?

All three miss this subject substantially, but without S-028, we have:

```
test_comp %>% filter(subject != "S-028") %>%
  group_by(name) %>%
  summarize(n = n(),
            MAPE = mean(abs(res_a1c)),
            RMSPE = sqrt(mean(res_a1c^2)),
            max_error = max(abs(res_a1c))) %>%
  kable(digits = c(0, 0, 3, 4, 2))
```

name	n	MAPE	RMSPE	max_error
mod_1	143	1.083	1.6694	5.61
mod_2	143	1.087	1.6577	5.74
mod_3	143	1.092	1.6583	5.68

Excluding subject S-028, `mod_1` wins MAPE and maxE, but `mod_2` wins RMSPE

Conclusions based on complete case analysis?

1. In-sample model predictions are about equally accurate for each of the three models. Model 2 looks better in terms of adjusted R^2 and AIC, but model 1 looks better on BIC. There's really not much to choose from there.
2. Residual plots look similarly reasonable for linearity, Normality and constant variance in all three models.
3. In our holdout sample, model `mod_1` has the smallest MAPE (mean APE) and RMSPE and maximum error, while model `mod_2` has the smallest median absolute prediction error, although again all three models are pretty comparable. Excluding a bad miss on one subject in the test sample yields similar comparisons. Again, the three models do about equally well on these measures.

So, what should our "most useful" model be?

OK. Let’s do all of that again, using the (singly) imputed data.

Partition imputed data from `dm1_imp`

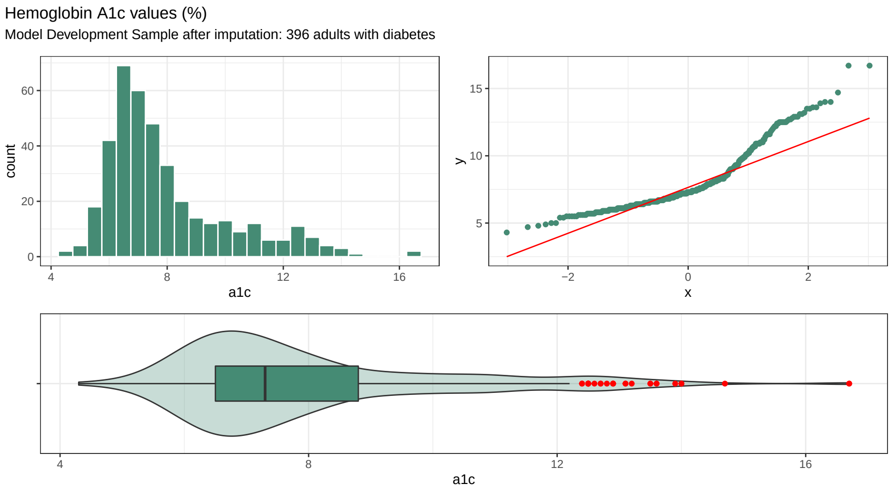
This time, we'll build an 80% development, 20% holdout partition of the `dm1_imp` data, and we'll also change our random seed, just for fun.

```
set.seed(20212021)

dm1_imp_train <- dm1_imp %>%
  slice_sample(prop = 0.8, replace = FALSE)

dm1_imp_test <-
  anti_join(dm1_imp, dm1_imp_train, by = "subject")

dim(dm1_imp_train); dim(dm1_imp_test)
```

```
[1] 396   5
```

```
[1] 100   5
```

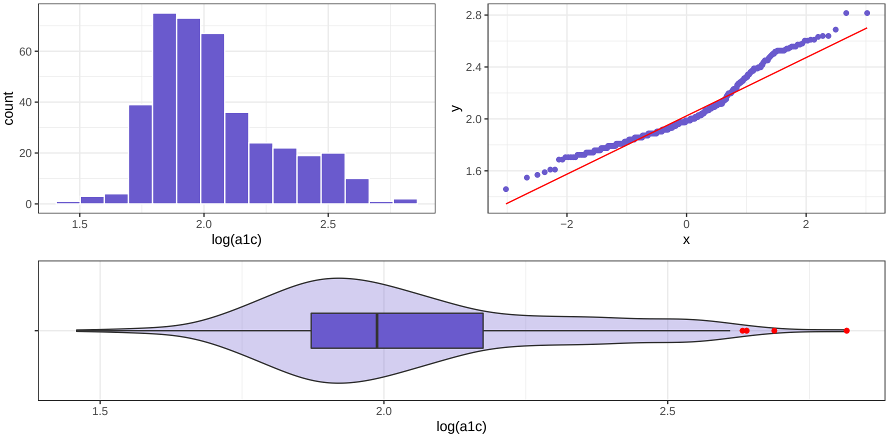
Distribution of a1c in training sample



Consider a log transformation?

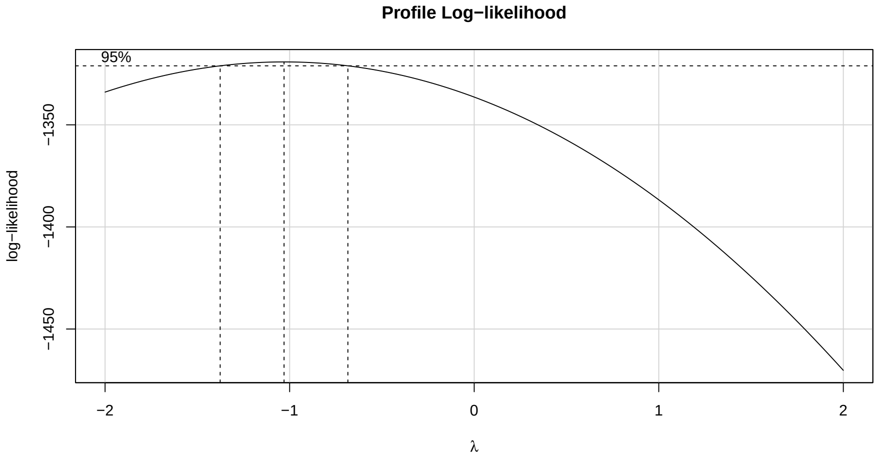
Natural Logarithm of Hemoglobin A1c

Model Development Sample: 396 adults with diabetes

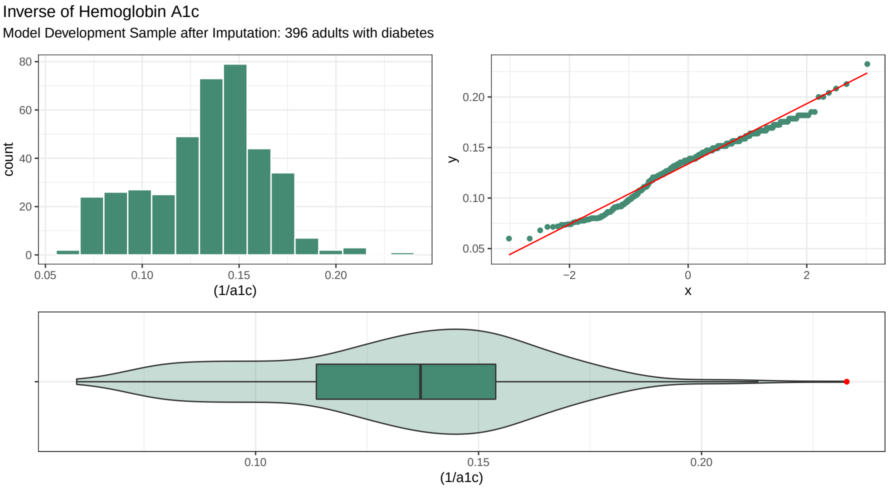


What does Box-Cox suggest?

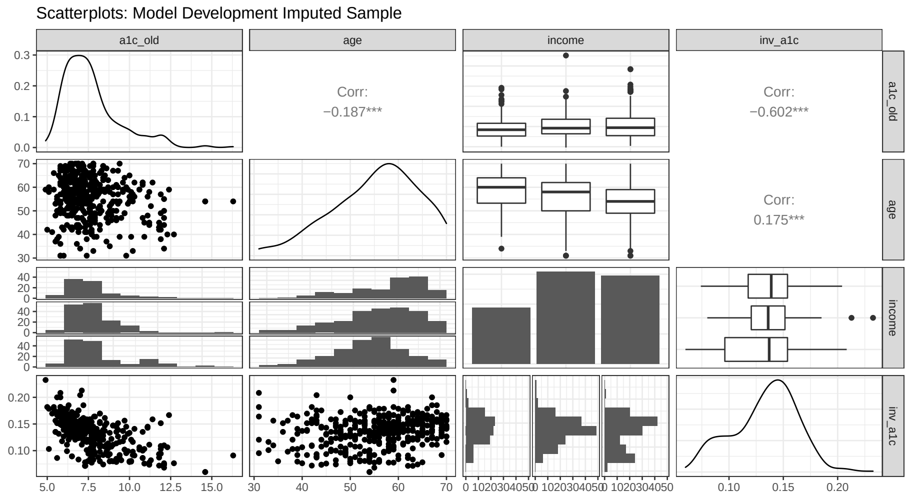
```
imod_0 <- lm(a1c ~ a1c_old + age + income,
             data = dm1_imp_train)
boxCox(imod_0)
```



Inverse of A1c again?



Scatterplot Matrix



Fitting the Same Three Models

- Remember we're using the model development sample here.

```
imod_1 <- lm((1/a1c) ~ a1c_old, data = dm1_imp_train)

imod_2 <- lm((1/a1c) ~ a1c_old + age, data = dm1_imp_train)

imod_3 <- lm((1/a1c) ~ a1c_old + age + income,
            data = dm1_imp_train)
```

Assess the quality of fit for candidate models within the development sample.

Tidied coefficients (imod_1)

```
tidy_im1 <- tidy(imod_1, conf.int = TRUE, conf.level = 0.95)

tidy_im1 %>%
  select(term, estimate, std.error, p.value,
         conf.low, conf.high) %>%
  knitr::kable(digits = 4)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.2126	0.0054	0	0.2019	0.2233
a1c_old	-0.0103	0.0007	0	-0.0117	-0.0090

The Regression Equation (imod_1)

Again, we'll use the equatiomatic package.

```
extract_eq(imod_1, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE, wrap = TRUE, terms_per_line = 3)
```

$$\widehat{(1/a1c)} = 0.2126 - 0.0103(a1c\_old) \tag{4}$$

Summary of Fit Quality (imod_1)

```
glance(imod_1) %>%
  mutate(name = "imod_1") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
imod_1	0.362	0.361	0.024	-1833	-1821

Tidied coefficients (`imod_2`)

```
tidy_im2 <- tidy(imod_2, conf.int = TRUE, conf.level = 0.95)

tidy_im2 %>%
  select(term, estimate, std.error, p.value,
         conf.low, conf.high) %>%
  knitr::kable(digits = 4)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.1991	0.0101	0.0000	0.1792	0.2189
a1c_old	-0.0101	0.0007	0.0000	-0.0115	-0.0087
age	0.0002	0.0001	0.1131	-0.0001	0.0005

The Regression Equation (`imod_2`)

Again, we'll use the `equatiomatic` package, and **results = 'asis'**.

```
extract_eq(imod_2, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE)
```

$$\widehat{(1/a1c)} = 0.1991 - 0.0101(a1c\_old) + 2e - 04(age) \quad (5)$$

Summary of Fit Quality (imod_2)

```
glance(imod_2) %>%
  mutate(name = "imod_2") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
imod_2	0.367	0.363	0.024	-1833	-1817

Tidied coefficients (imod_3)

```r
tidy_im3 <- tidy(imod_3, conf.int = TRUE, conf.level = 0.95)

tidy_im3 %>%
  select(term, estimate, se = std.error,
         low = conf.low, high = conf.high, p = p.value) %>%
  knitr::kable(digits = c(4,4,4,4,3))
```

term	estimate	se	low	high	p
(Intercept)	0.2002	0.0106	0.1795	0.221	0.0000
a1c_old	-0.0101	0.0007	-0.0115	-0.009	0.0000
age	0.0002	0.0001	-0.0001	0.000	0.1530
incomeBetween_30-50K	0.0010	0.0031	-0.0052	0.007	0.7590
incomeBelow_30K	-0.0023	0.0032	-0.0086	0.004	0.4764

The Regression Equation (imod_3)

Again, we'll use the equatiomatic package.

```
extract_eq(imod_3, use_coefs = TRUE, coef_digits = 4,
           ital_vars = TRUE, wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{(1/a1c)} &= 0.2002 - 0.0101(a1c\_old) + \\
&\quad 2e-04(age) + 0.001(income_{Between\_30-50K}) - \\
&\quad 0.0023(income_{Below\_30K})
\end{aligned} \tag{6}
$$

Summary of Fit Quality (imod_3)

```
glance(imod_3) %>%
  mutate(name = "imod_3") %>%
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) %>%
  knitr::kable(digits = c(0, 3, 3, 3, 0, 0))
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
imod_3	0.369	0.362	0.024	-1831	-1807

I checked stepwise regression again

- Even though variable selection **never** works, it is seductive.

What if we do forward selection in this situation?

```
min.model <- lm(a1c ~ 1, data = dm1_imp_train)
fwd.model <- step(min.model, direction = "forward",
                  scope = ~ a1c_old + age + income)
```

```
Start:  AIC=606.99
a1c ~ 1

          Df Sum of Sq    RSS    AIC
+ a1c_old  1    694.77 1129.9 419.20
+ age      1     64.26 1760.4 594.79
+ income   2     48.20 1776.5 600.39
<none>                 1824.7 606.99

Step:  AIC=419.2
```

Stepwise Regression Results

We wind up back at the model with all three predictors in this case (mod_3).

```
fwd.model$coefficients
```

```
        (Intercept)             a1c_old
         3.05112418          0.74107516
incomeBetween_30-50K     incomeBelow_30K
        -0.16321016          0.34246995
                age
        -0.01520655
```

- As we'll discuss in 432, there is an immense amount of evidence that variable selection causes severe problems in estimation and inference.

Which Model Looks Best In-Sample?

For each of these summaries, which model looks best in the training sample?

model	vars	r2	adj_r2	sigma	AIC	BIC
imod_1	a1c_old	0.362	0.361	0.02380	-1832.9	-1821
imod_2	+ age	0.367	0.363	0.02375	-1833.4	-1817
imod_3	+ income	0.369	0.362	0.02377	-1830.9	-1807

- `imod_3` (as it must, here) has the best R-square.
- `imod_2` wins on adjusted R-square and σ and AIC
- `imod_1` has the best BIC

Using augment to add fits, residuals, etc.

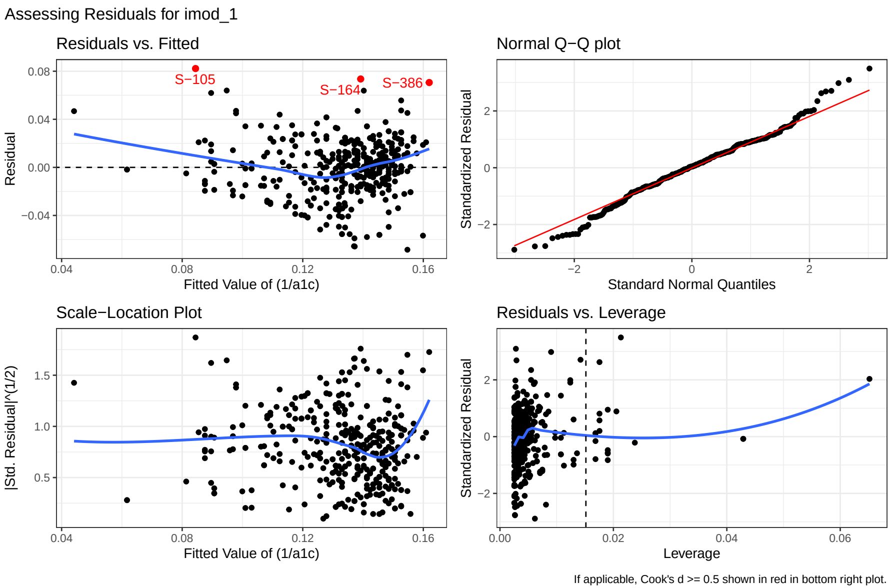
```r
augi1 <- augment(imod_1, data = dm1_imp_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome

augi2 <- augment(imod_2, data = dm1_imp_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome

augi3 <- augment(imod_3, data = dm1_imp_train) %>%
  mutate(inv_a1c = 1/a1c) # add in our model's outcome
```

Checking Regression Assumptions

Four key assumptions we need to think about:

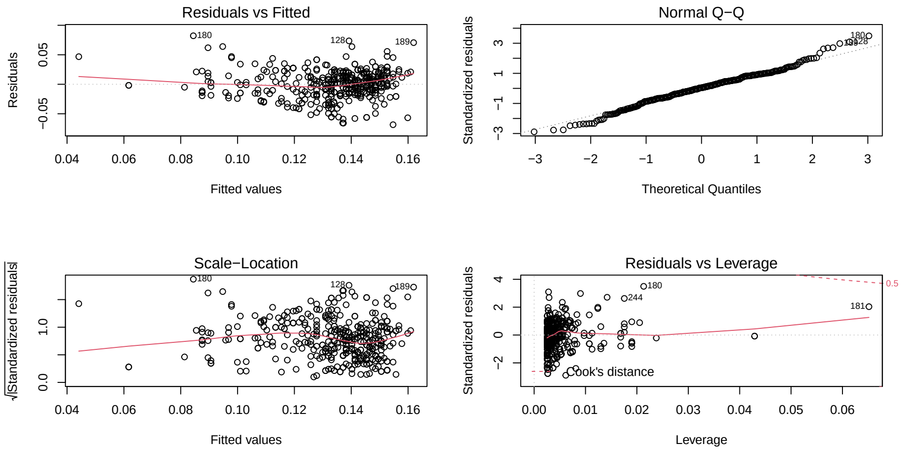
1. Linearity
2. Constant Variance (Homoscedasticity)
3. Normality
4. Independence

Main 4 Residual Plots for `imod_1` (via `ggplot2`)

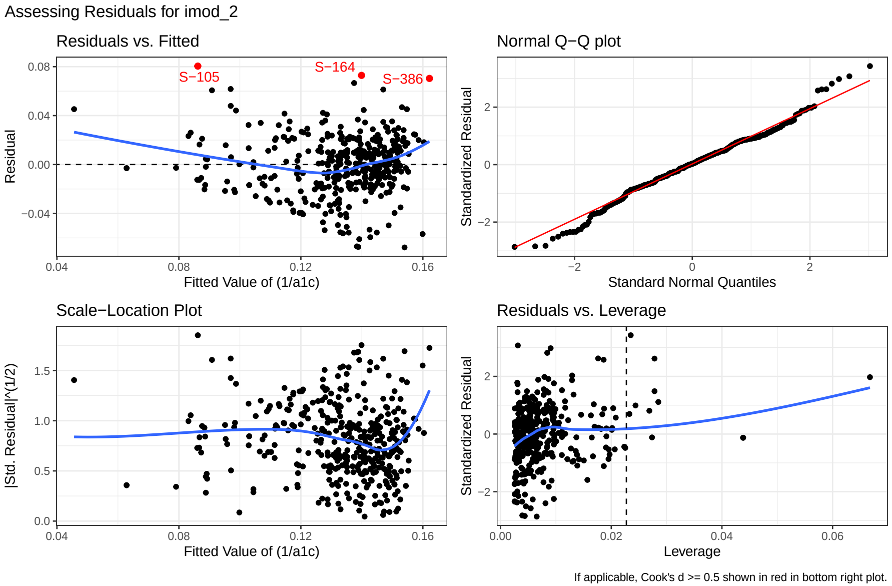


Base R Residual Plots for Model `imod_1`

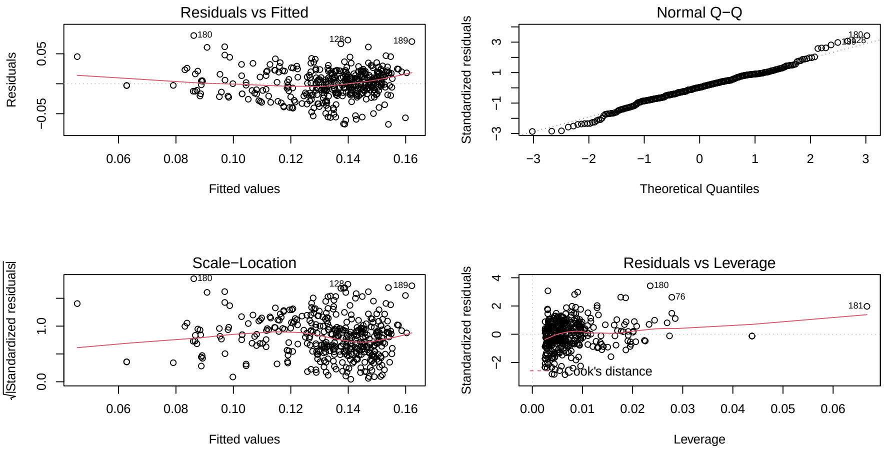
```
par(mfrow = c(2,2)); plot(imod_1); par(mfrow = c(1,1))
```



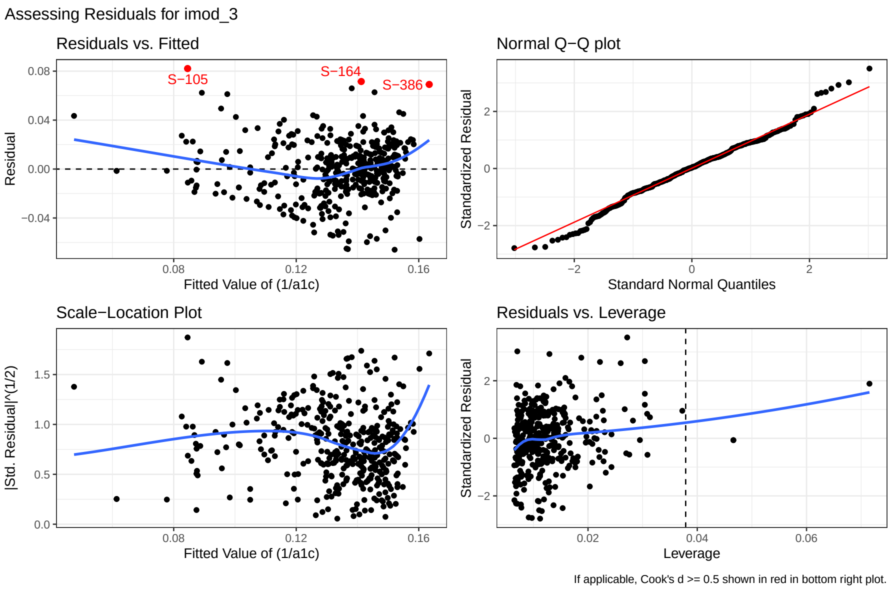
Main 4 Residual Plots for `imod_2` (via `ggplot2`)



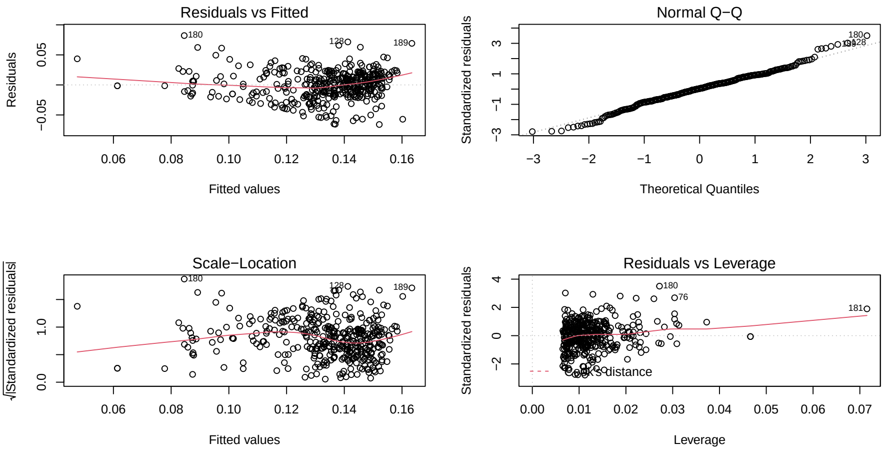
Base R Residual Plots for Model `imod_2`



Main 4 Residual Plots for `imod_3` (via `ggplot2`)



Base R Residual Plots for Model `imod_3`



Is collinearity a serious issue here?

```
car::vif(imod_3)
```

```
             GVIF Df GVIF^(1/(2*Df))
a1c_old 1.041113  1        1.020350
age     1.069426  1        1.034131
income  1.042549  2        1.010472
```

None of these values exceed 5, so it doesn't seem like there's any problem.

```
car::vif(imod_2)
```

```
a1c_old     age
1.03632 1.03632
```

Conclusions so far (in-sample)?

1. In-sample model predictions are not wildly different in terms of accuracy across the three models.
    - Model `imod_3` has the best R^2, while
    - Model `imod_2` wins on adjusted R^2, σ and AIC, and
    - Model `imod_1` has the best BIC.
2. Residual plots look similarly reasonable for linearity, Normality and constant variance in all three models after imputation.

Calculate prediction errors in test samples

```r
test_im1 <- augment(imod_1, newdata = dm1_imp_test) %>%
  mutate(name = "imod_1", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)

test_im2 <- augment(imod_2, newdata = dm1_imp_test) %>%
  mutate(name = "imod_2", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)

test_im3 <- augment(imod_3, newdata = dm1_imp_test) %>%
  mutate(name = "imod_3", fit_a1c = 1 / .fitted,
         res_a1c = a1c - fit_a1c)
```

```r
test_icomp <- bind_rows(test_im1, test_im2, test_im3) %>%
  arrange(subject, name)
```

Visualize Test-Sample Prediction Errors

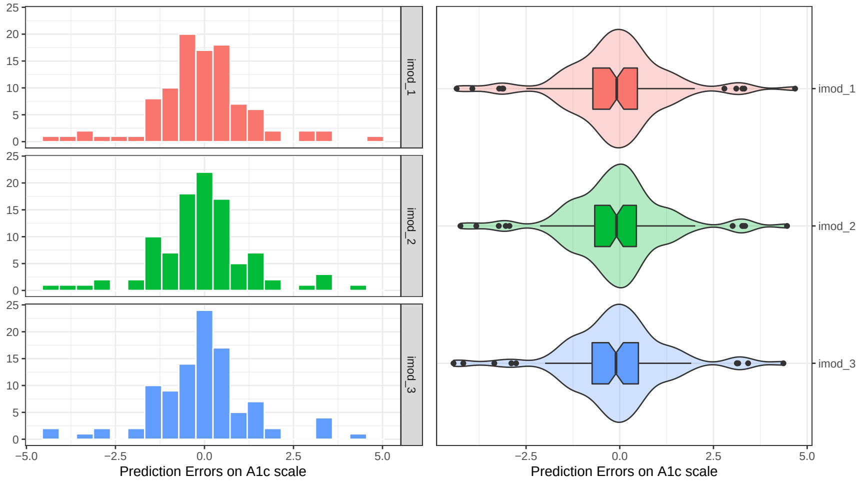


Table Comparing Model Prediction Errors

- Model `imod_2` has the best mean APE (MAPE) and RMSPE, while `imod_3` has the smallest maximum predictive error.

```
test_icomp %>%
  group_by(name) %>%
  summarize(n = n(),
            MAPE = mean(abs(res_a1c)),
            RMSPE = sqrt(mean(res_a1c^2)),
            max_error = max(abs(res_a1c))) %>%
  kable(digits = c(0, 0, 3, 3, 2))
```

name	n	MAPE	RMSPE	max_error
imod_1	100	0.971	1.396	4.68
imod_2	100	0.958	1.377	4.46
imod_3	100	0.964	1.384	4.43

Identify the largest errors (Results)

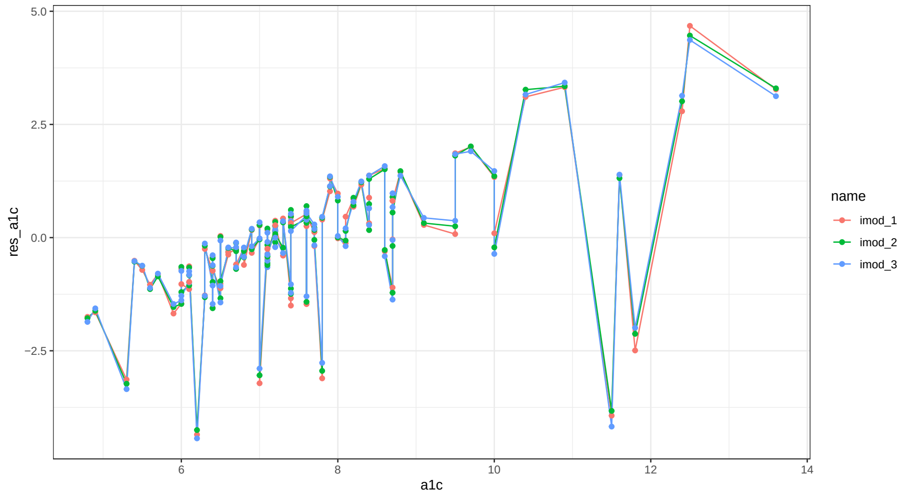
Identify the subject(s) where that maximum prediction error was made by each model, and the observed and model-fitted values of a1c in each case.

```
bind_rows(tempi1, tempi2, tempi3) %>%
  select(subject, name, a1c, fit_a1c, res_a1c)
```

```
# A tibble: 3 x 5
  subject name      a1c fit_a1c res_a1c
  <chr>   <chr>   <dbl>   <dbl>   <dbl>
1 S-471   imod_1  12.5     7.82    4.68
2 S-471   imod_2  12.5     8.04    4.46
3 S-341   imod_3   6.2    10.6    -4.43
```

Line Plot of the Errors?

Compare the errors that are made at each level of observed A1c?



Key Summaries

With complete cases,

- in-sample: all three models look OK on assumptions in residual plots, model 2 looks like it fits a little better by Adjusted R^2 and AIC, model 1 looks slightly better by BIC.
- out-of-sample: distributions of errors are similar. Model 1 has smallest MAPE, RMPSE and maximum error, while Model 2 has the smallest median error, but all three models are pretty similar.

With imputation,

- in-sample: nothing disastrous in residual plots, model 3 has the best R^2, Model 2 wins on adjusted R^2, σ, and AIC, and Model 1 has the best BIC.
- out-of-sample: Model 2 has the smallest MAPE, RMSE, but model 3 has the smallest maximum predictive error.

So what can we conclude? Does this particular imputation strategy have a big impact?

Again, this is our 431 Strategy

Which model is "most useful" in a prediction context?

1. Split the data into a model development (training) sample of about 70-80% of the observations, and a model test (holdout) 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.
5. 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.)
6. Select a "final" model for use based on the evidence in steps 3, 4 and especially 5.