431 Class 19

Thomas E. Love, Ph.D.

2022-11-10

Today's Agenda

- Finish the regression analysis on the complete cases from `dm1` which we started in Class 18.
  - Regression Assumptions?
  - Making Predictions in the Holdout (Test) Sample?
  - Selecting and presenting a final model

Today's Packages

```
options(dplyr.summarise.inform = FALSE)

library(simputation) # for single impuation
library(car) # for boxCox
library(GGally) # for ggpairs
library(glue) # for enhancing labels with code results
library(ggrepel) # help with residual plots
library(equatiomatic) # help with equation extraction
library(broom) # for tidying model output
library(kableExtra) # formatting tables
library(janitor); library(naniar); library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

431 strategy: "most useful" model?

We went through these three steps in Class 18.

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.

431 strategy: "most useful" model?

We'll walk through these three steps today.

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.

From Class 18

```
dm1 <- readRDS("c19/data/dm1.Rds")
dm1_cc <- dm1 |> drop_na()

set.seed(202211)
dm1_cc_train <- dm1_cc |>
  slice_sample(prop = 0.7, replace = FALSE)
dm1_cc_test <-
  anti_join(dm1_cc, dm1_cc_train, by = "subject")
```

Three Regression Models We've Fit

- Model development sample: using the (1/a1c) transformation of our outcome.

```
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)
```

Could we have fit other predictor sets?

Three predictor candidates, so we could have used…

- `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`)

Would Stepwise Regression Help?

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.

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

Stepwise Regression on `mod_3`

```
1  step(mod_3)
```

Would Stepwise Regression Help?

```
Start:  AIC=-2507.78
(1/a1c) ~ a1c_old + age + income

           Df Sum of Sq     RSS     AIC
- income    2  0.000646 0.18303 -2510.6
- age       1  0.000703 0.18309 -2508.5
<none>                  0.18239 -2507.8
- a1c_old   1  0.082383 0.26477 -2384.9

Step:  AIC=-2510.59
(1/a1c) ~ a1c_old + age

           Df Sum of Sq     RSS     AIC
- age       1  0.001018 0.18405 -2510.7
<none>                  0.18303 -2510.6
- a1c_old   1  0.082647 0.26568 -2386.5

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

Coefficients:
(Intercept)      a1c_old
    0.20741     -0.00952
```

An Important Point

Stepwise regression lands on our `mod_2`, as it turns out.

- There is a **huge** amount of evidence that variable selection causes severe problems in estimation and inference.
- Stepwise regression is an especially 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.

Summarizing Fit (Training Sample)

Which Model looks best? Does this depend on the summary?

```
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)) |> kable_minimal(font_size = 28
```

model_vars	r2	adj_r2	sigma	AIC	BIC	df	df_res
1_a1c_old	0.3248	0.3228	0.02351	-1558.0	-1547	1	333
2_+age	0.3286	0.3245	0.02348	-1557.9	-1543	2	332
3_+income	0.3309	0.3228	0.02351	-1555.1	-1532	4	330

Which Model Looks Best?

Check adherence to regression assumptions in the development sample.

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

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

`aug1` includes all variables in `dm_cc_train` and also:

- `inv_a1c` = 1/`a1c`, transformed outcome `mod_1` predicts
- `.fitted` = fitted (predicted) values of 1/`a1c`
- `.resid` = residual (observed - fitted outcome) values; larger residuals (positive or negative) mean poorer fit
- `.std.resid` = standardized residuals (residuals scaled to SD = 1, remember residual mean is already 0)

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

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

`aug1` also includes:

`augment` results for the first 2 subjects

```
1 aug1 |> select(subject, a1c:income, inv_a1c) |>
2   tail(2) |> kbl(dig = 3) |> kable_classic(full_width = F)
```

subject	a1c	a1c_old	age	income	inv_a1c
S-467	6.5	7.7	42	Below_30K	0.154
S-172	6.9	11.1	48	Below_30K	0.145

```
1 aug1 |> select(subject, .fitted:.cooksd) |>
2   tail(2) |> kbl(dig = 3) |> kable_classic(full_width = F)
```

subject	.fitted	.resid	.hat	.sigma	.cooksd
S-467	0.134	0.020	0.003	0.024	0.001
S-172	0.102	0.043	0.015	0.023	0.026

`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:

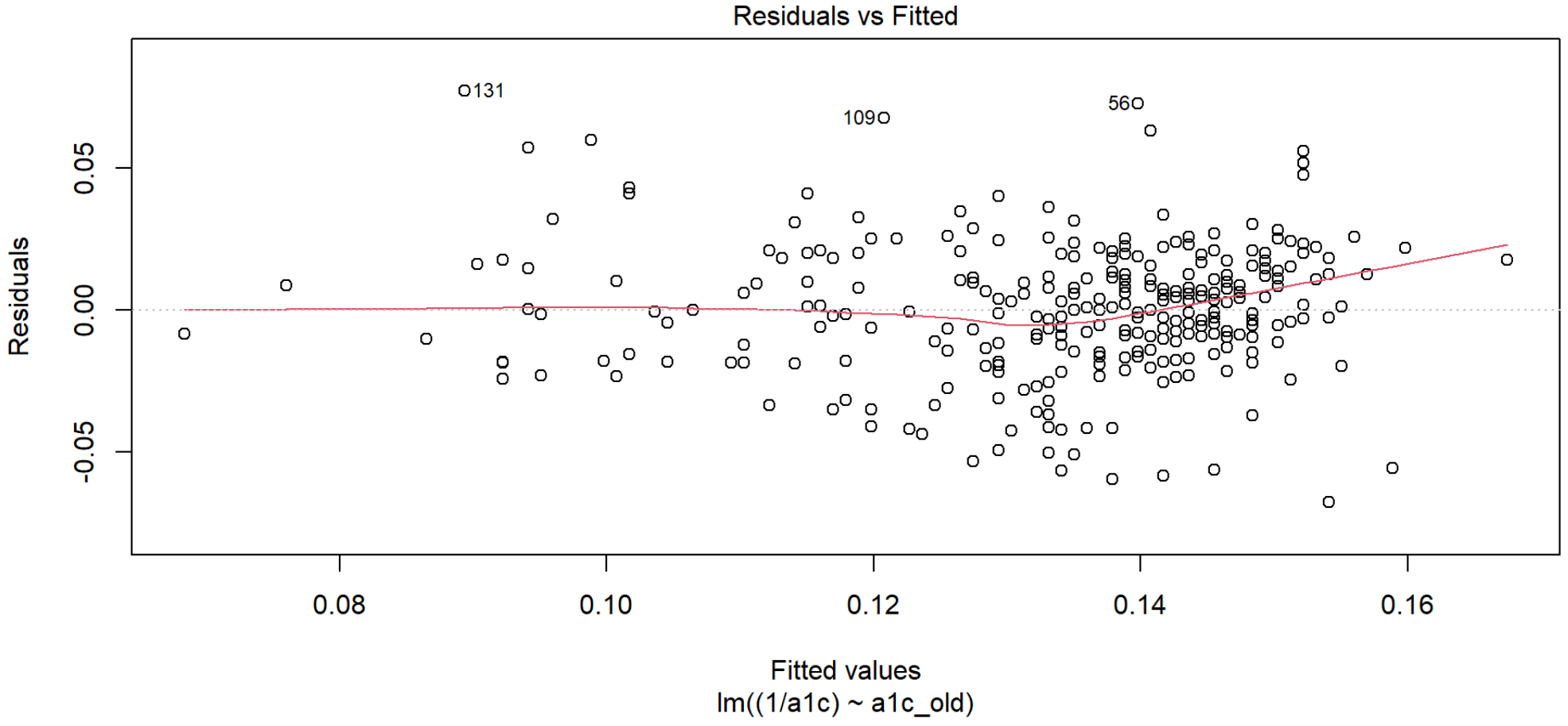
1. Linearity
2. Constant Variance (Homoscedasticity)
3. Normality
4. Independence

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 (mod_1)

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



Which points are highlighted in that plot?

Note that the points labeled 56, 109 and 131 are the 56th, 109th and 131st rows in our `dm1_cc_train` data file, or, equivalently, in our `aug1` file.

```
1 aug1 |> slice(c(56, 109, 131)) |> select(a1c:.resid, inv_a1c)
```

```
# A tibble: 3 × 8
    a1c a1c_old   age income          subject .fitted .resid inv_a1c
  <dbl>   <dbl> <dbl> <fct>           <chr>     <dbl>  <dbl>   <dbl>
1   4.7     7.1    59 Between_30-50K S-164     0.140  0.0730   0.213
2   5.3     9.1    48 Below_30K      S-071     0.121  0.0679   0.189
3   6      12.4    59 Below_30K      S-105     0.0894 0.0773   0.167
```

These are subjects `S-164`, `S-071`, and `S-105`, respectively.

Another way to confirm who the plot is identifying

As mentioned, we think the identifiers (56, 109 and 131) of the points with the largest residual (in absolute value) describe subjects `S-164`, `S-071`, and `S-105`, respectively. Does this make sense?

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

```
# A tibble: 6 × 2
  subject  .resid
  <chr>     <dbl>
1 S-105    0.0773
2 S-164    0.0730
3 S-071    0.0679
4 S-168   -0.0679
5 S-052    0.0633
6 S-001    0.0599
```

Normal Q-Q: `mod_1` Standardized Residuals

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

How troublesome are these outliers?

```
1 nrow(aug1)
```

```
[1] 335
```

```
1 aug1 |> select(subject, .std.resid) |>
2   arrange(desc(abs(.std.resid)))
```

```
# A tibble: 335 × 2
   subject .std.resid
   <chr>         <dbl>
 1 S-105          3.33
 2 S-164          3.11
 3 S-168         -2.90
 4 S-071          2.90
 5 S-052          2.70
 6 S-001          2.57
 7 S-214         -2.55
 8 S-258         -2.49
 9 S-141          2.47
10 S-497         -2.41
# … with 325 more rows
```

Testing the largest outlier?

```
1 outlierTest(mod_1)
```

```
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
    rstudent unadjusted p-value Bonferroni p
131 3.383089         0.00080246      0.26882
```

A studentized residual is just another way to standardize the residuals that has some useful properties here.

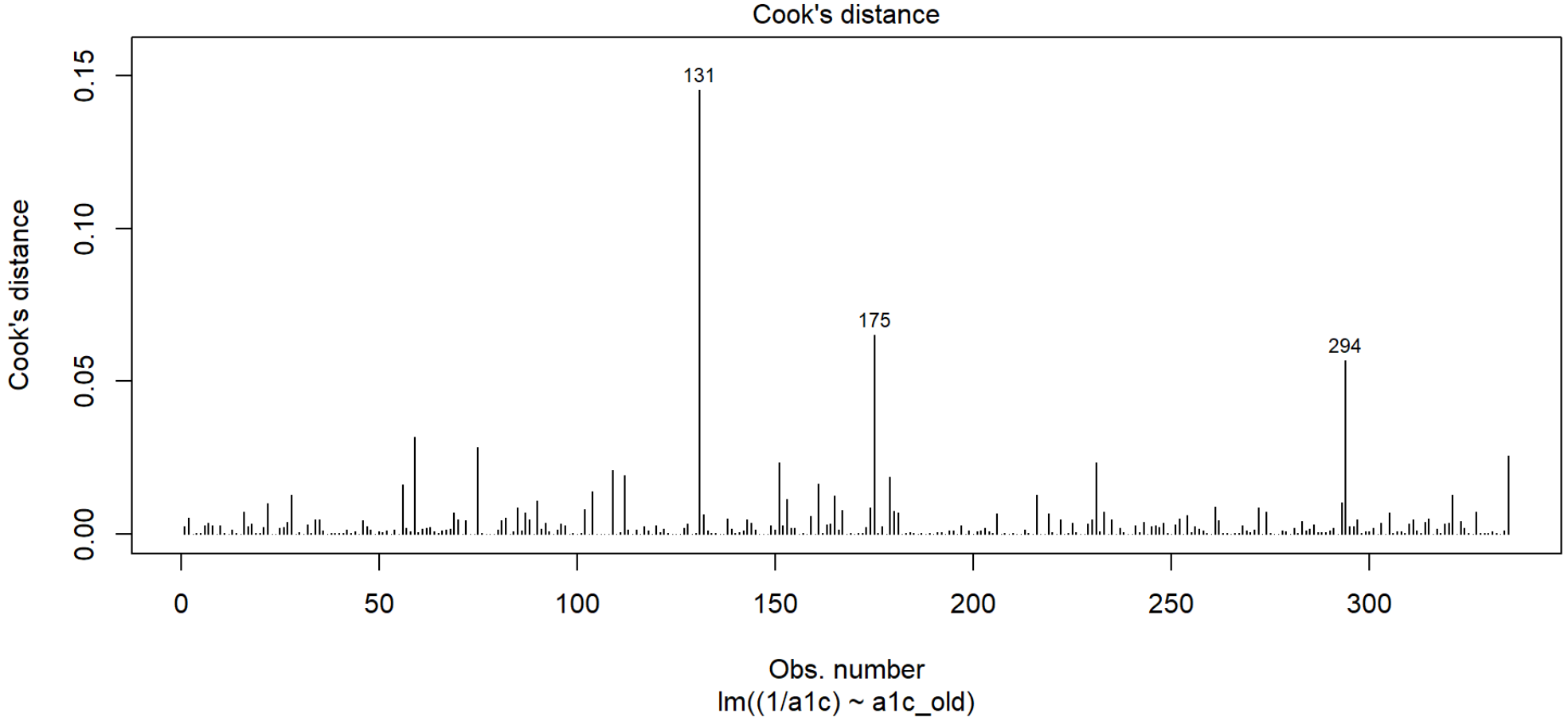
- No indication that having a maximum absolute value of 3.38 in a sample of 335 studentized residuals is a major concern about the Normality assumption, given the Bonferroni p- value = 0.27.

Scale-Location for heteroscedasticity (mod_1)

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

Cook's distance for influence (mod_1)

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

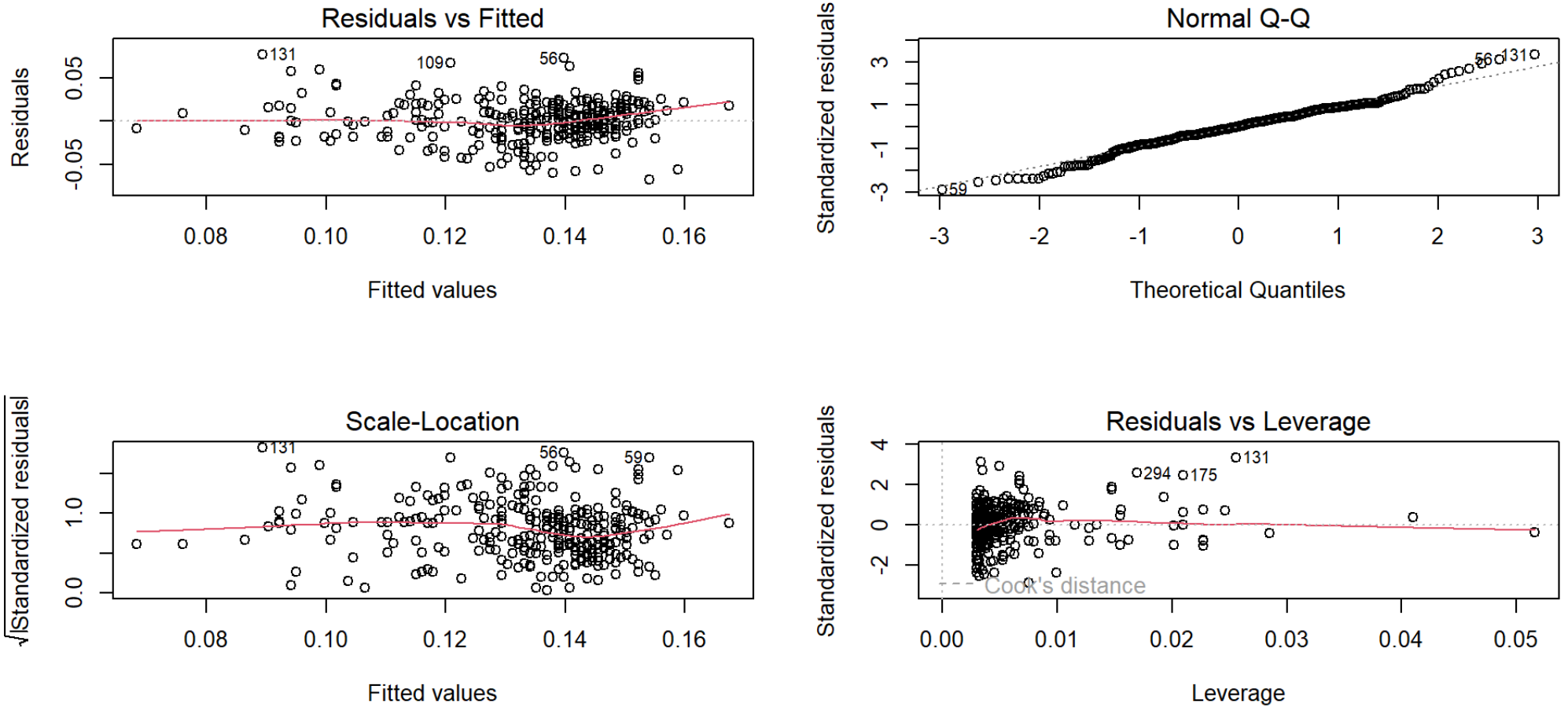


Residuals, Leverage and Influence (mod_1)

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

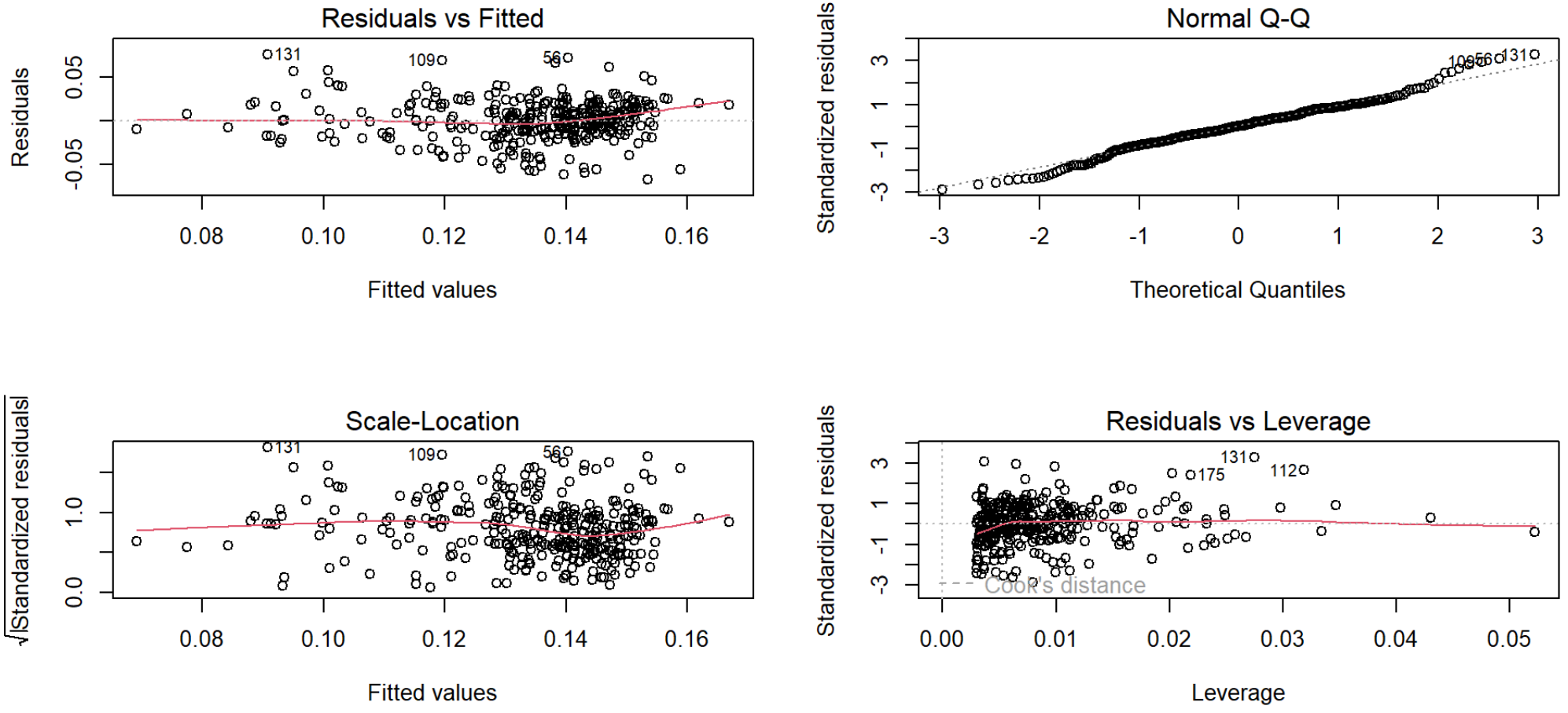
Residual Plots for Model mod_1

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



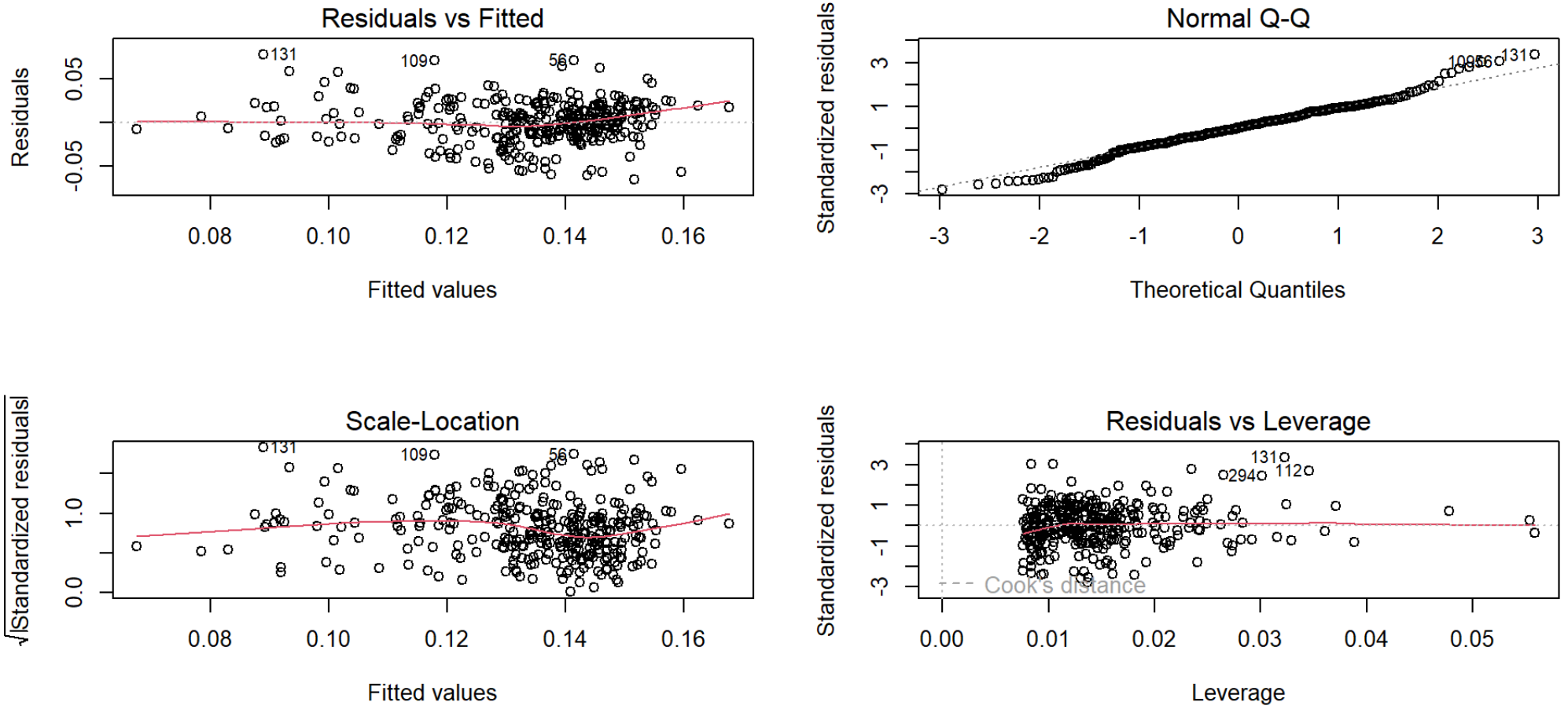
Residual Plots for Model mod_2

```
1 par(mfrow = c(2,2)); plot(mod_2); par(mfrow = c(1,1))
```



Residual Plots for Model mod_3

```
1  par(mfrow = c(2,2)); plot(mod_3); par(mfrow = c(1,1))
```



Is collinearity a serious issue here?

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

```
            GVIF Df GVIF^(1/(2*Df))
a1c_old 1.030377  1        1.015075
age     1.092349  1        1.045155
income  1.081374  2        1.019751
```

- Collinearity = correlated predictors
  - Remember that the scatterplot matrix didn't suggest any strong correlations between our predictors.

Is collinearity a serious issue here?

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

```
            GVIF Df GVIF^(1/(2*Df))
a1c_old 1.030377  1        1.015075
age     1.092349  1        1.045155
income  1.081374  2        1.019751
```

- (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.

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. `mod_2` looks better in terms of adjusted \(R^2\) and \(\sigma\), but `mod_1` looks better on AIC and BIC. There's 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 build 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.

Using `ggplot2` to build residual plots?

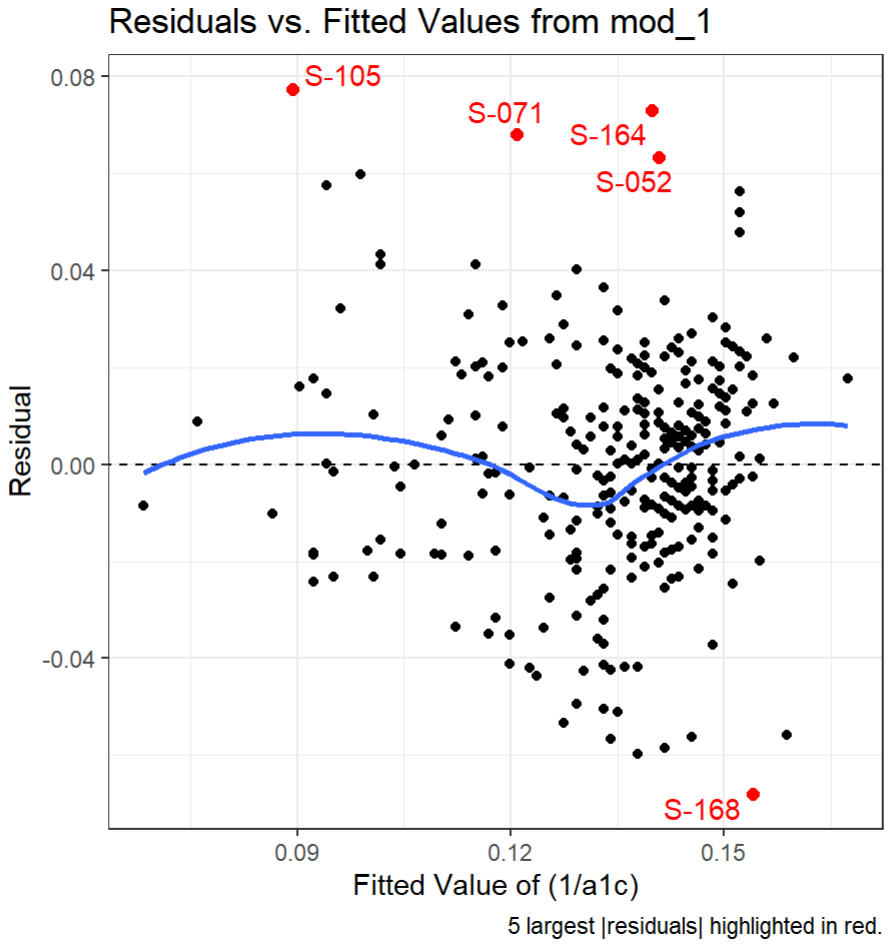
3. The scale-location plot of the square root of the standardized residuals vs. the fitted values is also pretty straightforward.
4. The `augment` function can be used to obtain Cook's distance, standardized residuals and leverage values, so we can mimic both the index plot (of Cook's distance) as well as the residuals vs. leverage plot with Cook's distance contours, if we like.

Demonstrations on the next few slides.

Residuals vs. Fitted Values: ggplot2

```
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)
```

Residuals vs. Fitted Values: `ggplot2`



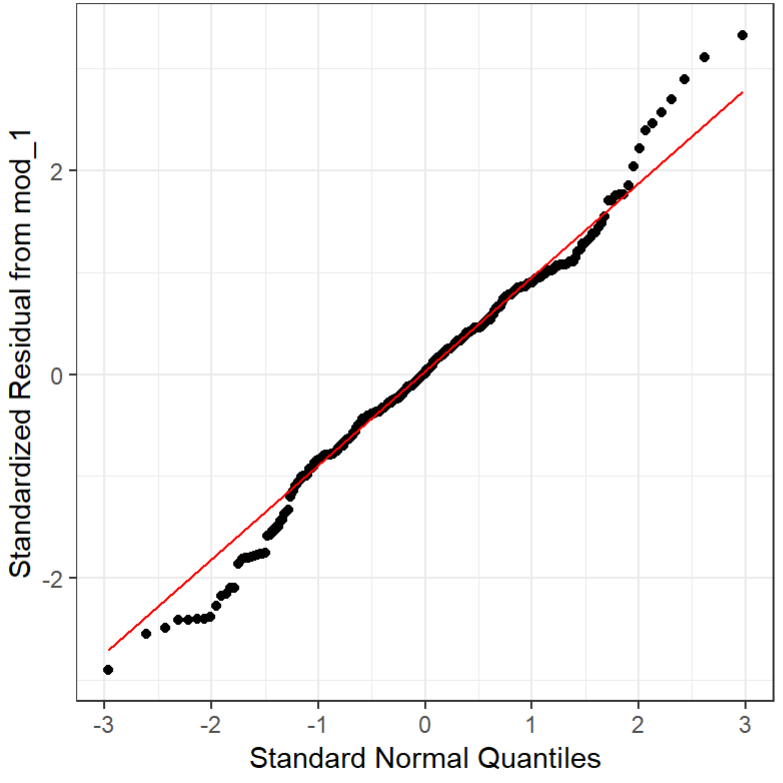
Standardized Residuals: ggplot2

```
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")

p1 + p2 +
  plot_layout(widths = c(2, 1)) +
  plot_annotation(
    title = "Normality of Standardized Residuals from mod_1"
```

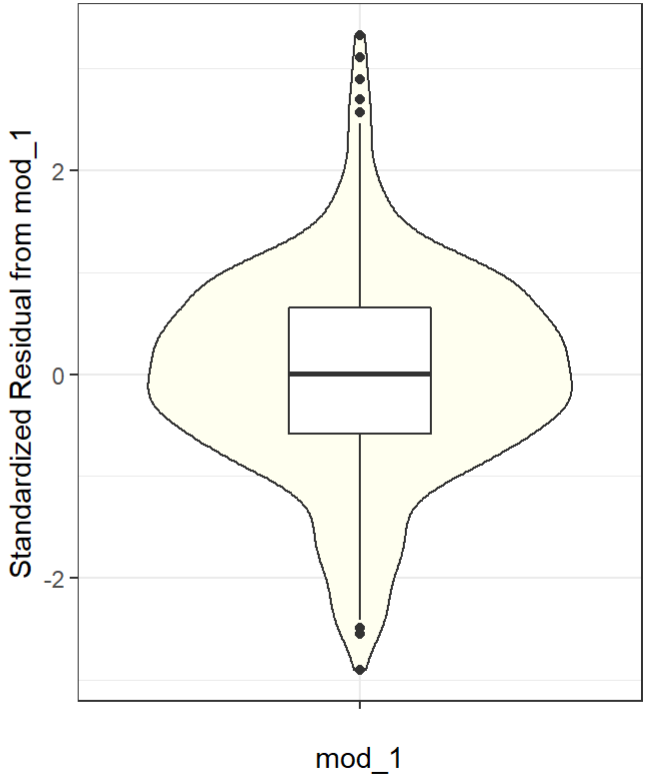
Standardized Residuals: ggplot2

Normality of Standardized Residuals from mod_1

Normal Q-Q plot



Box and Violin Plots

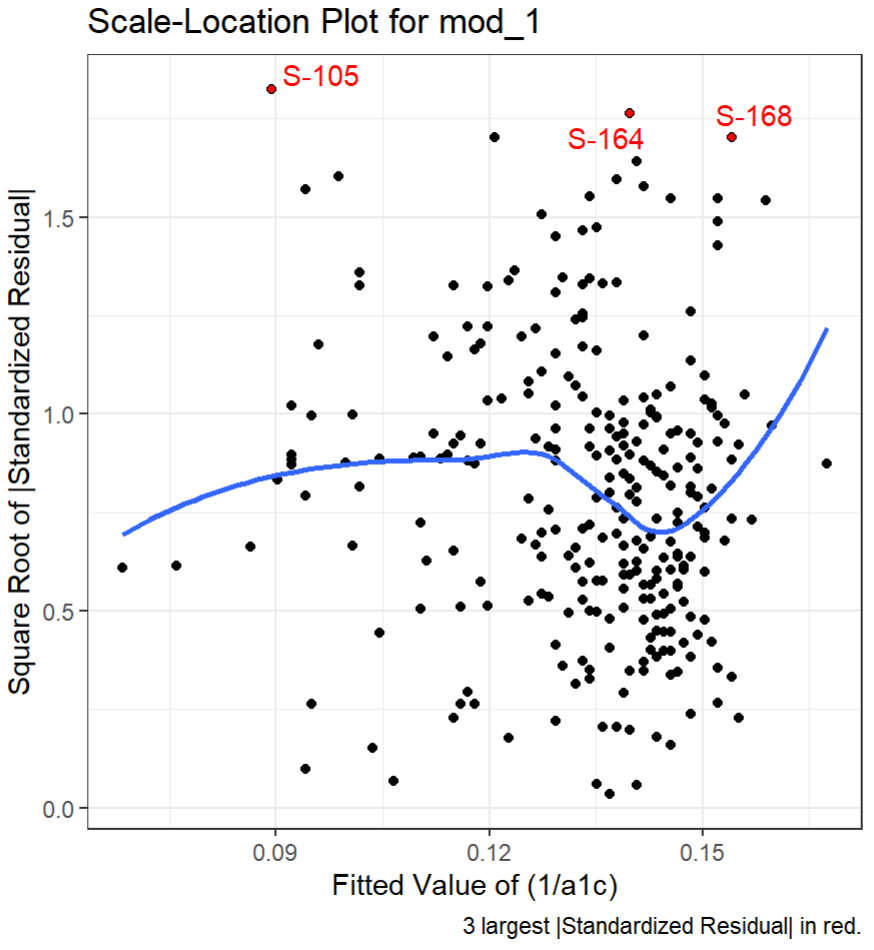


n = 335 residual values are plotted here.

Scale-Location Plot via ggplot2

```
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)
```

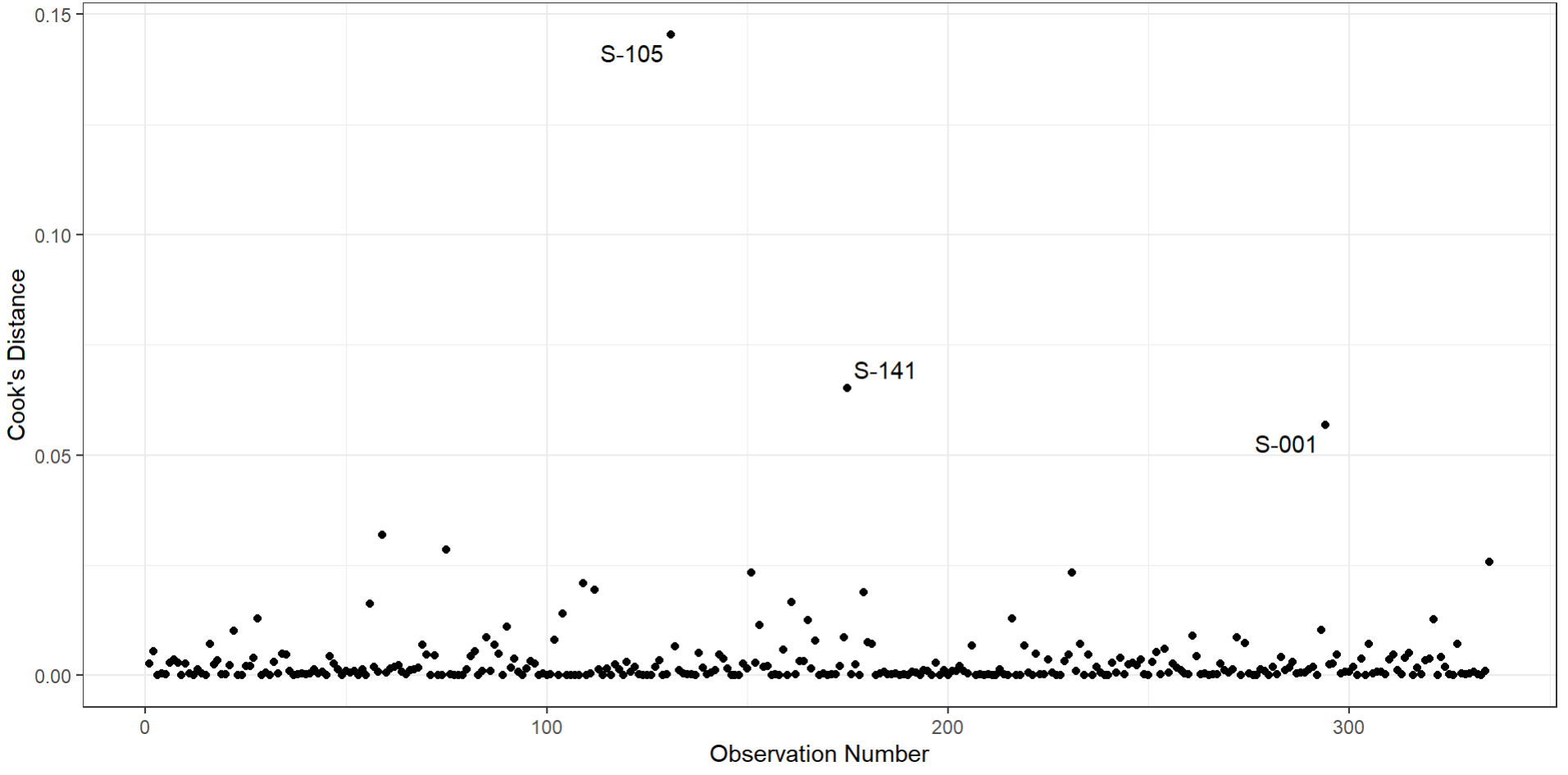
Scale-Location Plot via `ggplot2`



Cook's Distance Index Plot via ggplot2

```
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")
```

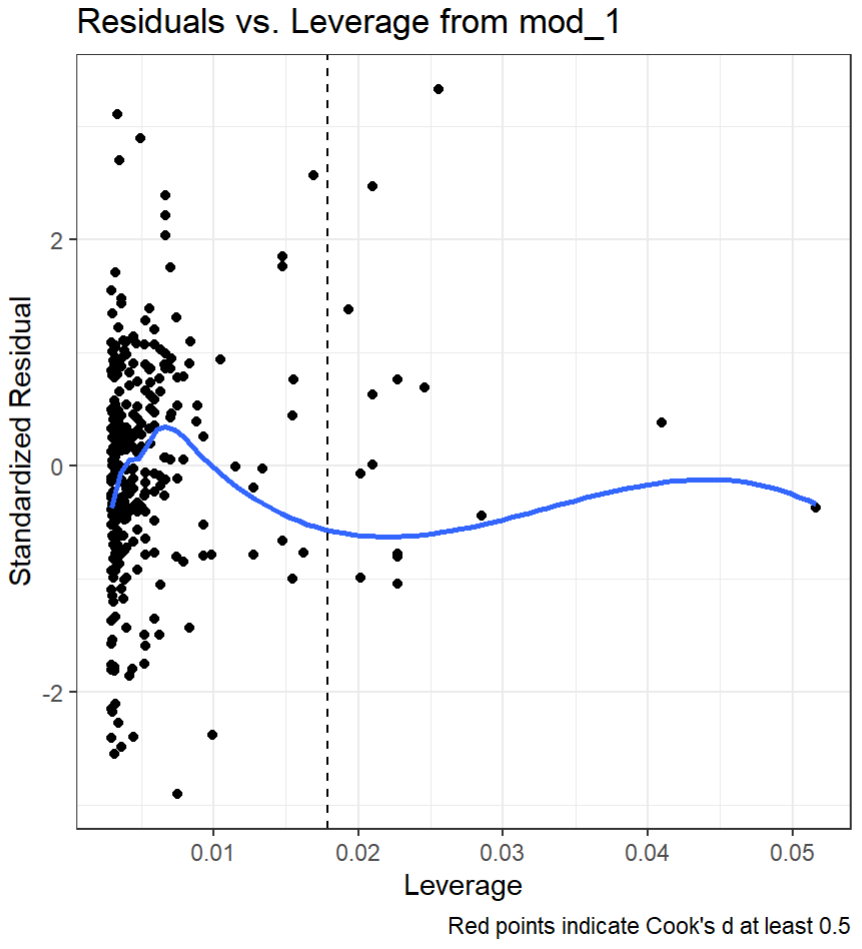
Cook's Distance Index Plot via `ggplot2`



Residuals vs. Leverage Plot via ggplot2

```
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)
```

Residuals vs. Leverage Plot via `ggplot2`



Some Notes on the Residuals vs. Leverage Plot

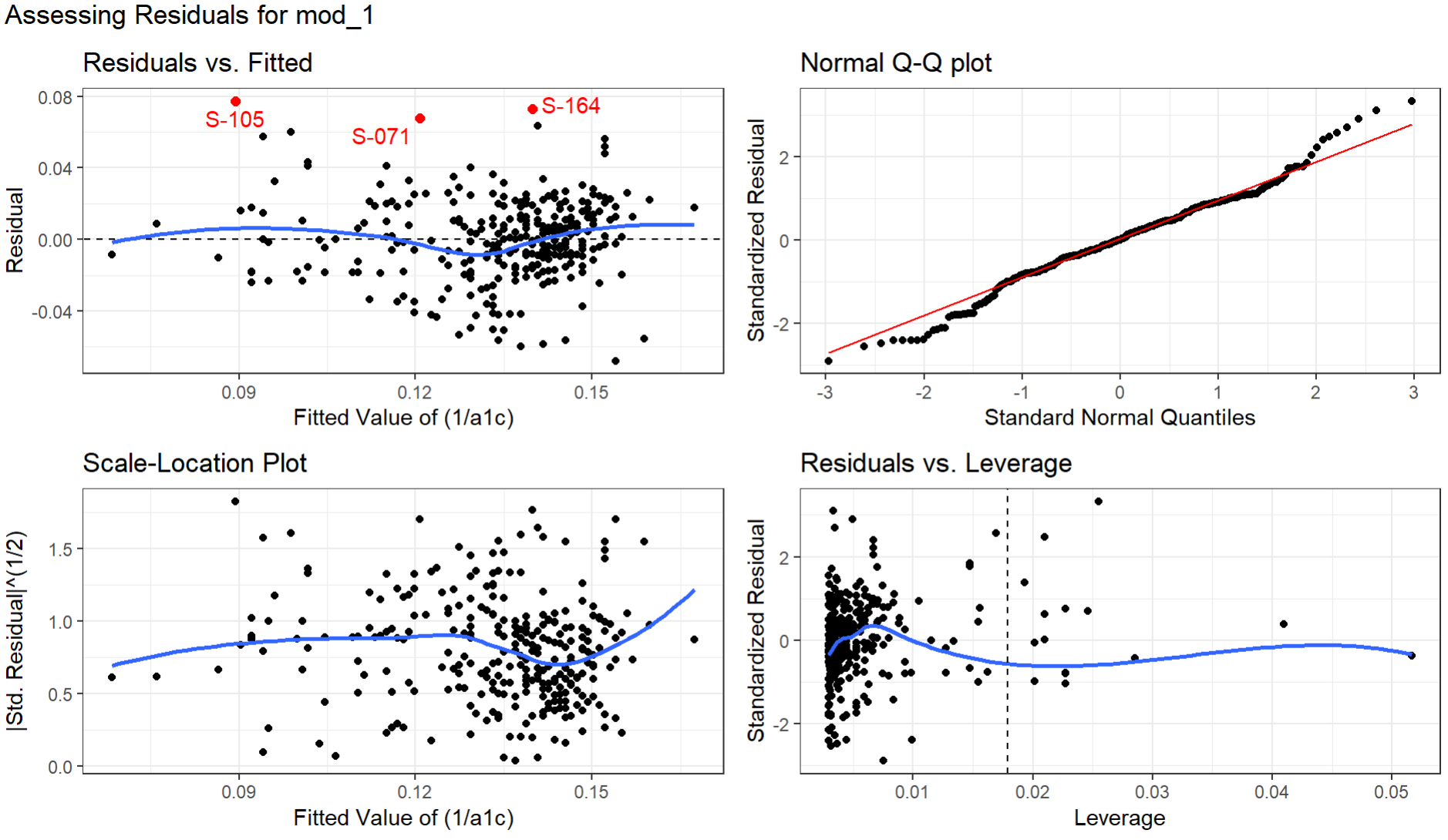
In this `ggplot()` approach,

- Points with Cook's d >= 0.5 would be highlighted and in red, if there were any.
- Points right of the dashed line have high leverage, by one standard.
- Points with more than 3 times the average leverage are identified as highly leveraged by some people, hence my dashed vertical line.

Residual Plots for mod_1 (via ggplot2)

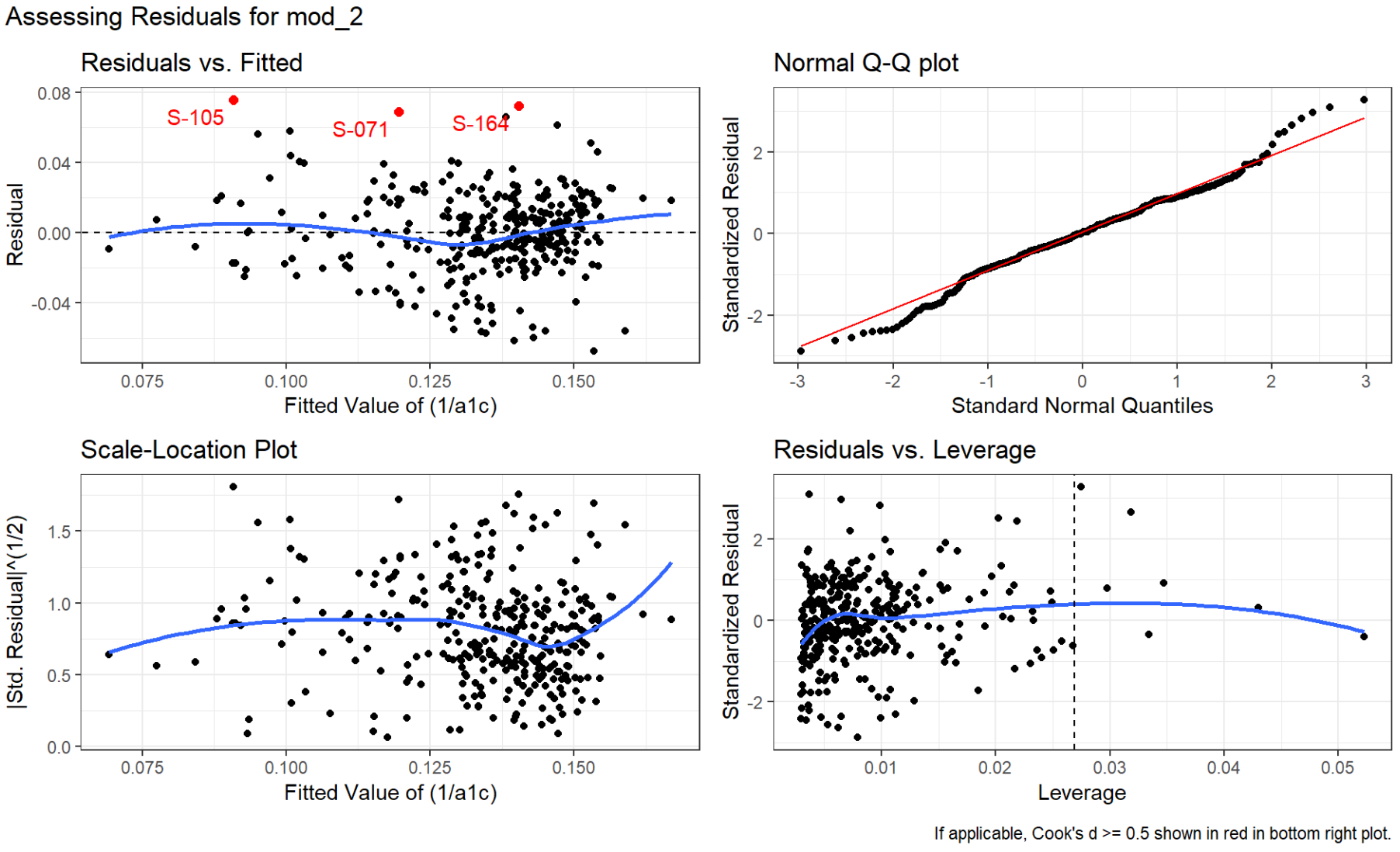
```
p1 <- ggplot(aug1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_point(data = aug1 |>
               slice_max(abs(.resid), n = 3),
             col = "red", size = 2) +
  geom_text_repel(data = aug1 |>
               slice_max(abs(.resid), n = 3),
               aes(label = subject), col = "red") +
  geom_abline(intercept = 0, slope = 0, lty = "dashed") +
  geom_smooth(method = "loess", formula = y ~ x, se = F) +
  labs(title = "Residuals vs. Fitted",
       x = "Fitted Value of (1/a1c)", y = "Residual")

p2 <- ggplot(aug1, aes(sample = .std.resid)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(title = "Normal Q-Q plot",
       y = "Standardized Residual",
       x = "Standard Normal Quantiles")
```

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

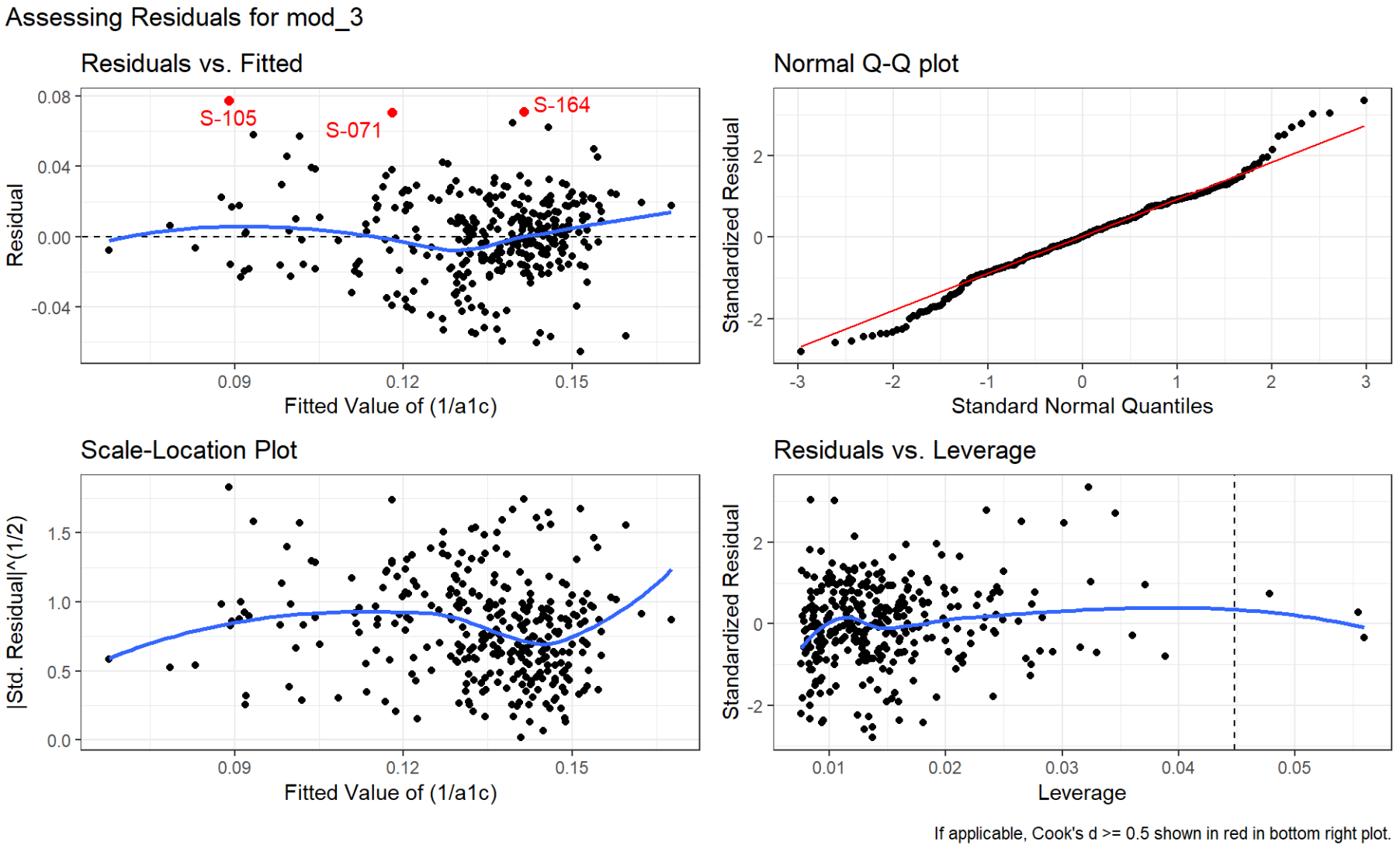


If applicable, Cook's d >= 0.5 shown in red in bottom right plot.

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



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.

Make predictions into the test sample using these 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...

mod_1 prediction errors in test sample

```
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?

```
test_m1 |>
  select(subject, a1c, fit_a1c, res_a1c, a1c_old,
         age, income) |>
  head() |>
  kbl(digits = c(0, 1, 2, 2, 1, 0, 0)) |> kable_classic(font_size = 28)
```

subject	a1c	fit_a1c	res_a1c	a1c_old	age	income
S-002	11.0	19.15	-8.15	16.3	54	Between_30-50K
S-005	6.7	6.78	-0.08	6.3	64	Between_30-50K
S-013	8.1	6.96	1.14	6.7	55	Higher_than_50K
S-019	7.9	7.06	0.84	6.9	70	Between_30-50K
S-023	6.9	6.74	0.16	6.2	58	Higher_than_50K
S-025	7.6	7.25	0.35	7.3	56	Below_30K

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)
```

Test sample results: all 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) |>
  kbl(digits = c(0, 1, 2, 2, 1, 0, 0)) |> kable_classic(font_size = 28)
```

Test sample results: all three models

name	subject	a1c	fit_a1c	res_a1c	a1c_old	age	income
mod_1	S-002	11.0	19.15	-8.1	16	54	Between_30-50K
mod_2	S-002	11.0	18.77	-7.8	16	54	Between_30-50K
mod_3	S-002	11.0	18.23	-7.2	16	54	Between_30-50K
mod_1	S-013	8.1	6.96	1.1	7	55	Higher_than_50K
mod_2	S-013	8.1	6.98	1.1	7	55	Higher_than_50K
mod_3	S-013	8.1	6.94	1.2	7	55	Higher_than_50K

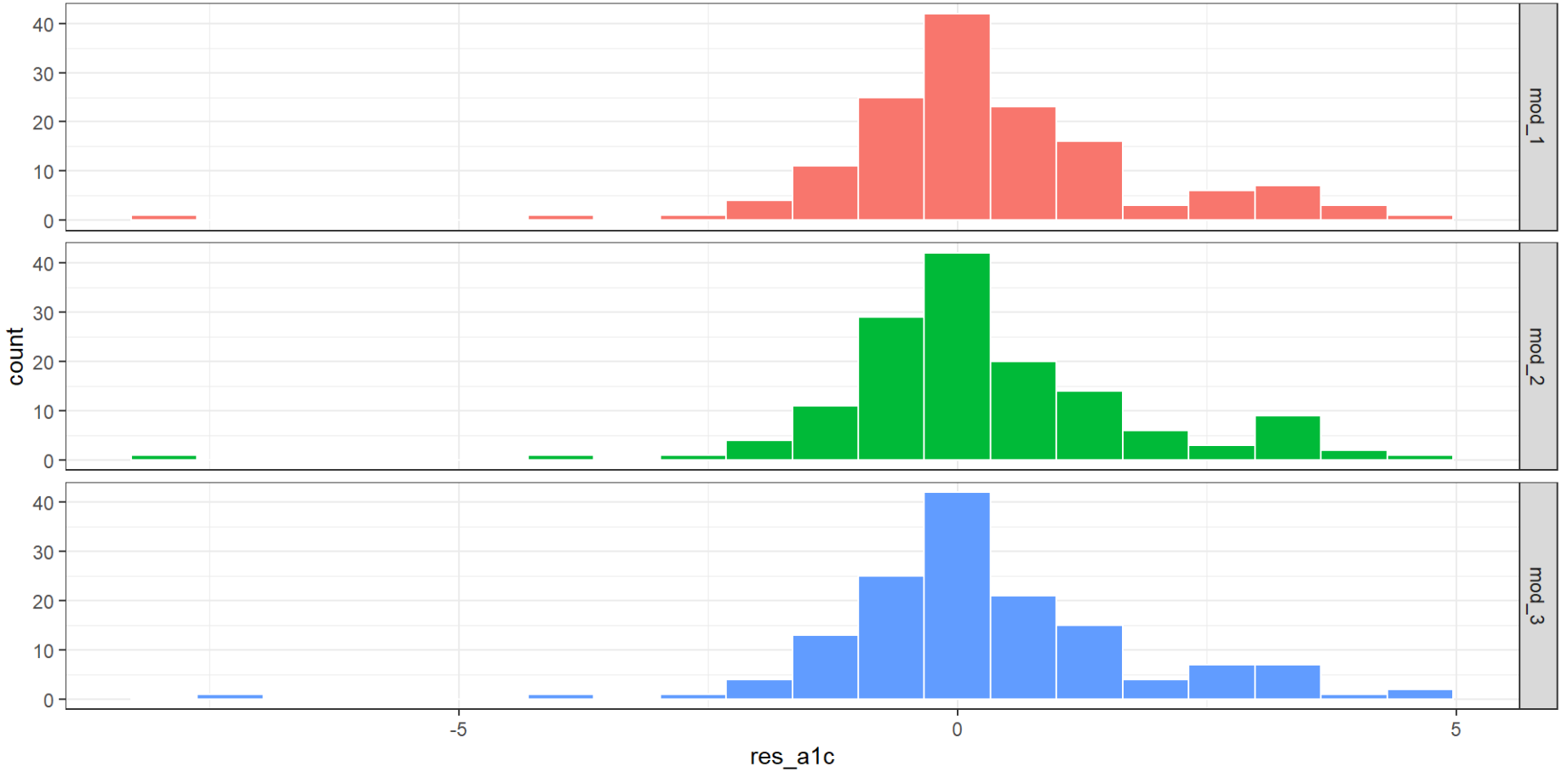
What do we do to compare the test-sample errors?

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.

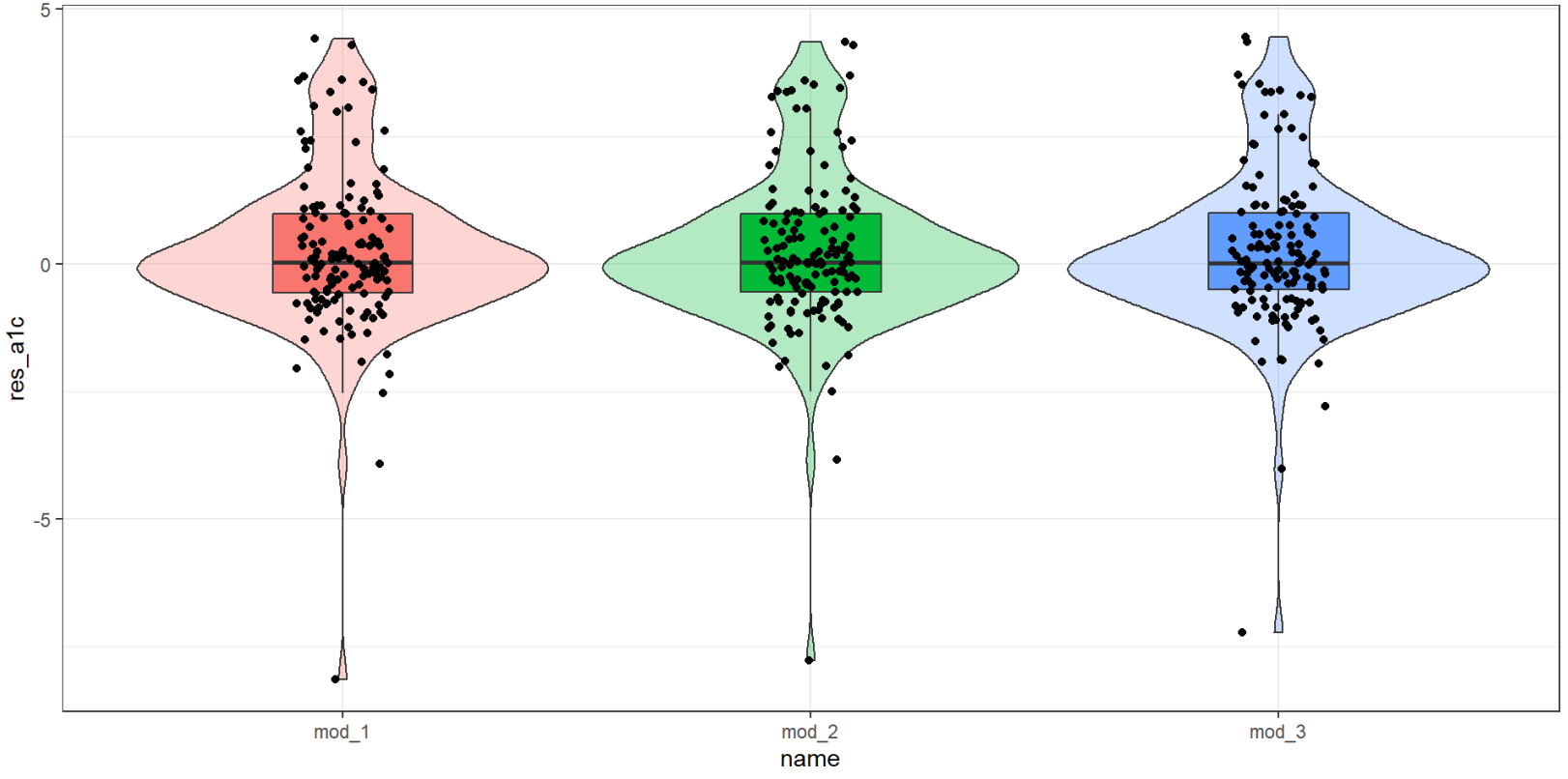
Visualize the prediction errors

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



Alternate Plot

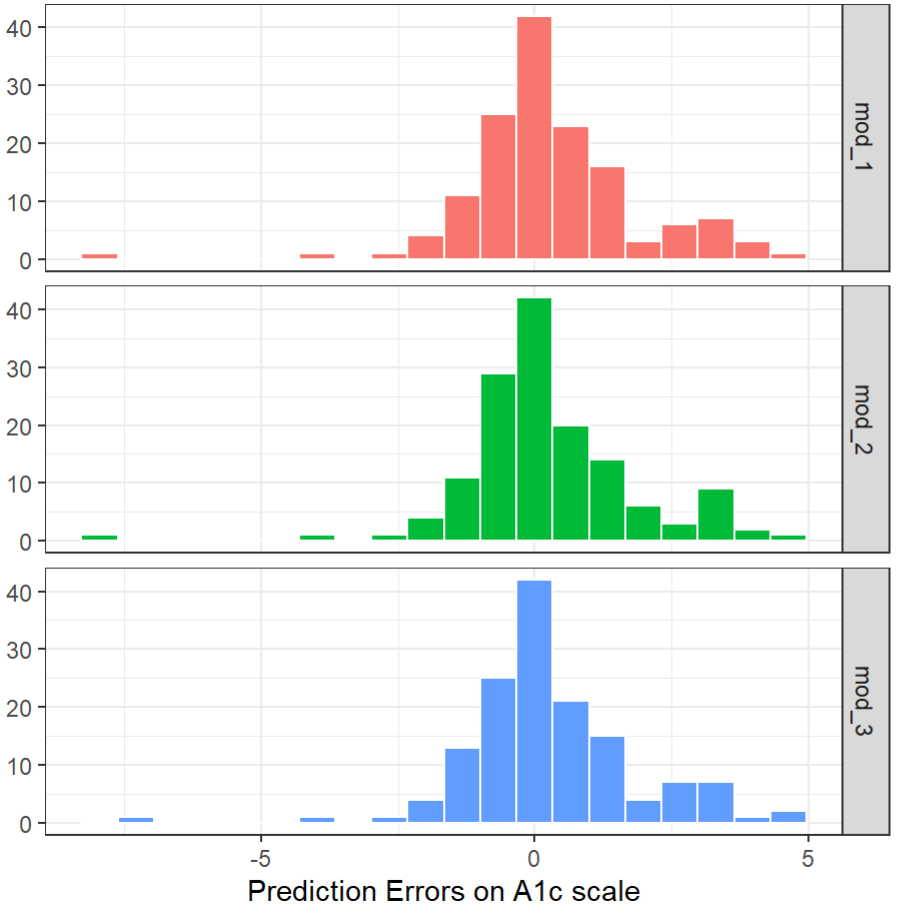
```
ggplot(test_comp, aes(x = name, y = res_a1c, fill = name)) +
  geom_violin(alpha = 0.3) +
  geom_boxplot(width = 0.3, outlier.shape = NA) +
  geom_jitter(height = 0, width = 0.1) +
  guides(fill = "none")
```



Test-Sample Prediction Errors

```
p1 <- ggplot(test_comp, aes(x = res_a1c, fill = name)) +
  geom_histogram(bins = 20, col = "white") +
  labs(x = "Prediction Errors on A1c scale", y = "") +
  facet_grid (name ~ .) + guides(fill = "none")

p2 <- ggplot(test_comp, aes(x = factor(name), y = res_a1c,
                            fill = name)) +
  geom_violin(alpha = 0.3) +
  geom_boxplot(width = 0.3, notch = TRUE) +
  scale_x_discrete(position = "top",
                   limits =
                     rev(levels(factor(test_comp$name)))) +
  guides(fill = "none") +
  labs(x = "", y = "Prediction Errors on A1c scale") +
  coord_flip()

p1 + p2 + plot_layout(ncol = 2)
```

Test-Sample Prediction Errors



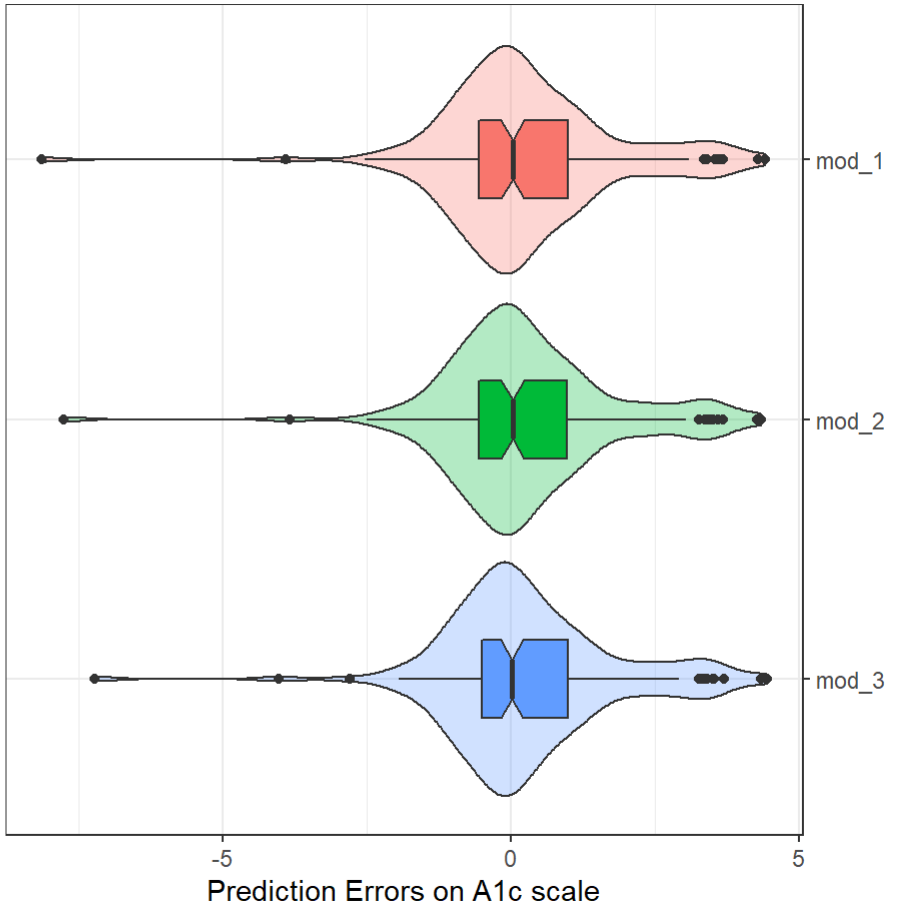


Table Comparing Model Prediction Errors

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))) |>
  kbl(digits = c(0, 0, 4, 3, 2, 3)) |> kable_classic(font_size = 28)
```

Table Comparing Model Prediction Errors

name	n	MAPE	RMSPE	max_error	median_APE
mod_1	144	1.0714	1.595	8.15	0.757
mod_2	144	1.0558	1.566	7.77	0.739
mod_3	144	1.0586	1.555	7.23	0.714

Conclusions from Table of Errors

- Model `mod_2` has the smallest MAPE (mean APE)
- Model `mod_3` has the smallest maximum error and root mean squared prediction error (RMSPE) and median absolute prediction error.

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)))
```

Identifying the Largest Errors

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

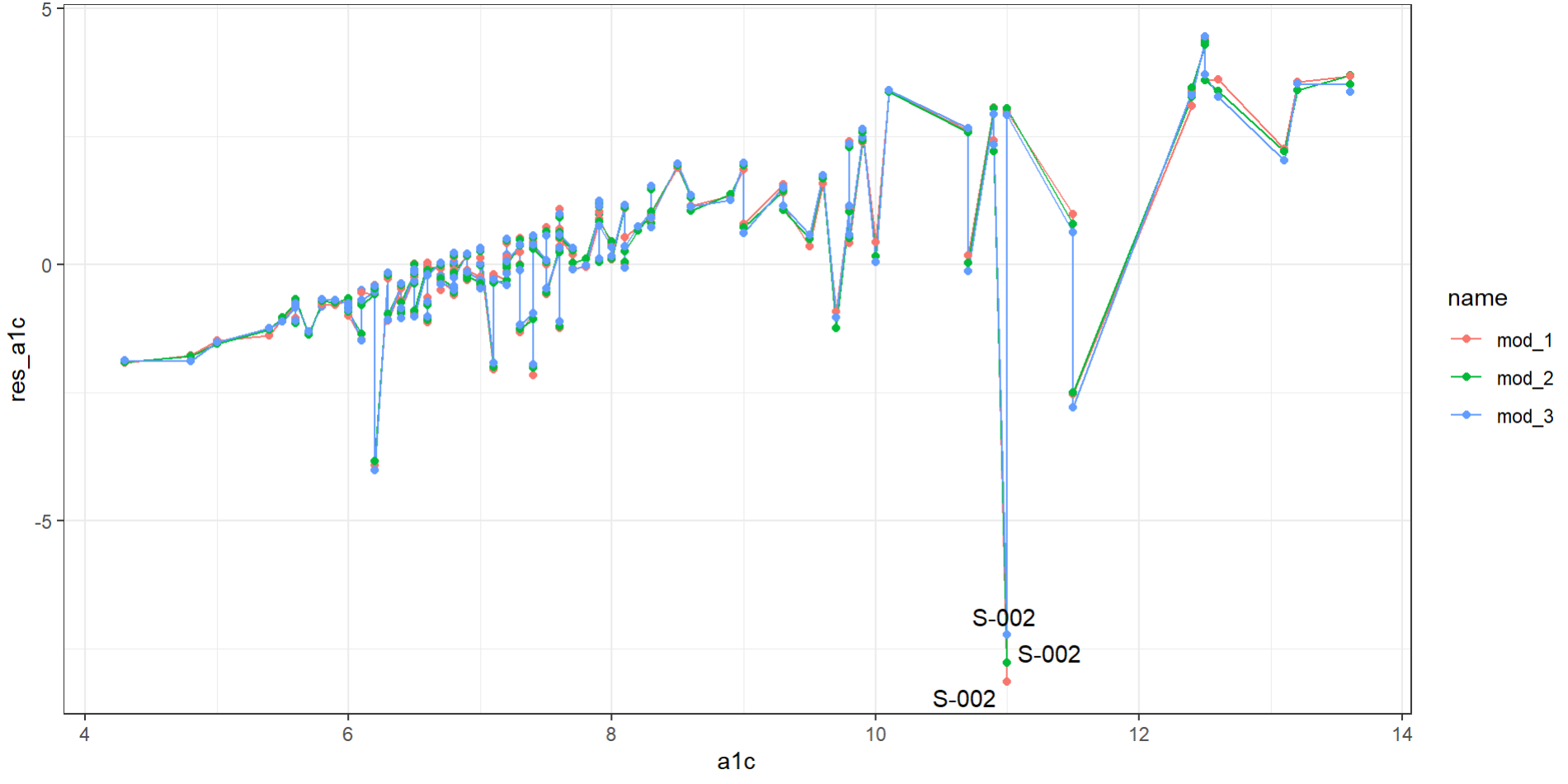
```
# A tibble: 3 × 5
  subject name    a1c fit_a1c res_a1c
  <chr>   <chr> <dbl>   <dbl>   <dbl>
1 S-002   mod_1    11    19.1   -8.15
2 S-002   mod_2    11    18.8   -7.77
3 S-002   mod_3    11    18.2   -7.23
```

Line Plot of the Errors?

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

```
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-002"),
                  aes(label = subject))
```

Line Plot of the Errors?



What if we ignored S-002 for a moment?

```
test_comp |> filter(subject != "S-002") |>
  group_by(name) |>
  summarize(n = n(),
            MAPE = mean(abs(res_a1c)),
            RMSPE = sqrt(mean(res_a1c^2)),
            max_error = max(abs(res_a1c))) |>
  kbl(digits = c(0, 0, 3, 4, 2)) |> kable_classic(font_size = 28)
```

name	n	MAPE	RMSPE	max_error
mod_1	143	1.022	1.4479	4.41
mod_2	143	1.009	1.4310	4.35
mod_3	143	1.015	1.4382	4.44

Excluding subject S-002, mod_2 wins all three summaries.

"Complete Case" Conclusions?

1. In-sample model predictions are about equally accurate for each of the three models. `mod_2` looks better in terms of adjusted \(R^2\) and \(\sigma\), but `mod_1` looks better on AIC and 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, `mod_2` has the smallest MAPE (mean APE), while `mod_3` has the best results for RMSPE and maximum error, although again all three models are pretty comparable. Excluding a bad miss on one subject in the test sample suggests that `mod_2` may in fact be a bit better than the others.

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

Clean Up

```
rm(aug1, aug1_extra, aug2, aug3,
   mod_1, mod_2, mod_3,
   p1, p2, p3, p4,
   temp1, temp2, temp3,
   test_comp, test_m1, test_m2, test_m3)
```

Session Information

```
1 sessionInfo()
```

```
R version 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```