431 Class 18

Thomas E. Love, Ph.D.

2022-11-03

Today's Agenda

Multiple Regression using the `dm1` data (Part 1 of 3)

- 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

Today's Packages

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

library(simputation) # for single impuation
library(car) # for boxCox
library(GGally) # for ggpairs
library(glue) # for labeling with live R code
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())
```

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)

The dm1 data

```
dm1 <- readRDS("c18/data/dm1.Rds")

dm1
```

```
# A tibble: 500 × 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
 4   6.5     5.8    53 Below_30K       S-004
 5   6.7     6.3    64 Between_30-50K  S-005
 6   5.8     6.5    48 Below_30K       S-006
 7   9.6    NA      49 Below_30K       S-007
 8   6.1     7.2    63 Between_30-50K  S-008
 9  12.9     7.7    55 Below_30K       S-009
10   6.7     6.8    63 Below_30K       S-010
# … with 490 more rows
```

Summarizing the dm1 tibble

```
1  summary(dm1)
```

```
      a1c             a1c_old           age                    income
 Min.   : 4.300   Min.   : 4.200   Min.   :31.00   Higher_than_50K:123
 1st Qu.: 6.500   1st Qu.: 6.500   1st Qu.:49.00   Between_30-50K :194
 Median : 7.300   Median : 7.300   Median :56.00   Below_30K      :178
 Mean   : 7.898   Mean   : 7.693   Mean   :55.41   NA's           :  5
 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
   subject
 Length:500
 Class :character
 Mode  :character
```

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`

`favstats` on multiple quantities?

```
dm1 |>
  mosaicCore::df_stats(~ a1c + a1c_old + age) |>
  rename(na = missing) |> kbl(digits = 2) |> kable_classic_2(font_size = 28
```

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

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

What will we do about missing data?

```
dm1 |>
  summarize(across(everything(), ~ sum(is.na(.)))) |>
  kbl() |> kable_classic_2(full_width = F)
```

a1c	a1c_old	age	income	subject
4	15	0	5	0

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

Missingness and naniar

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

```
  1  miss_case_table(dm1)
```

```
# A tibble: 3 × 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

```
1 miss_case_summary(dm1)
```

```
# A tibble: 500 × 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?

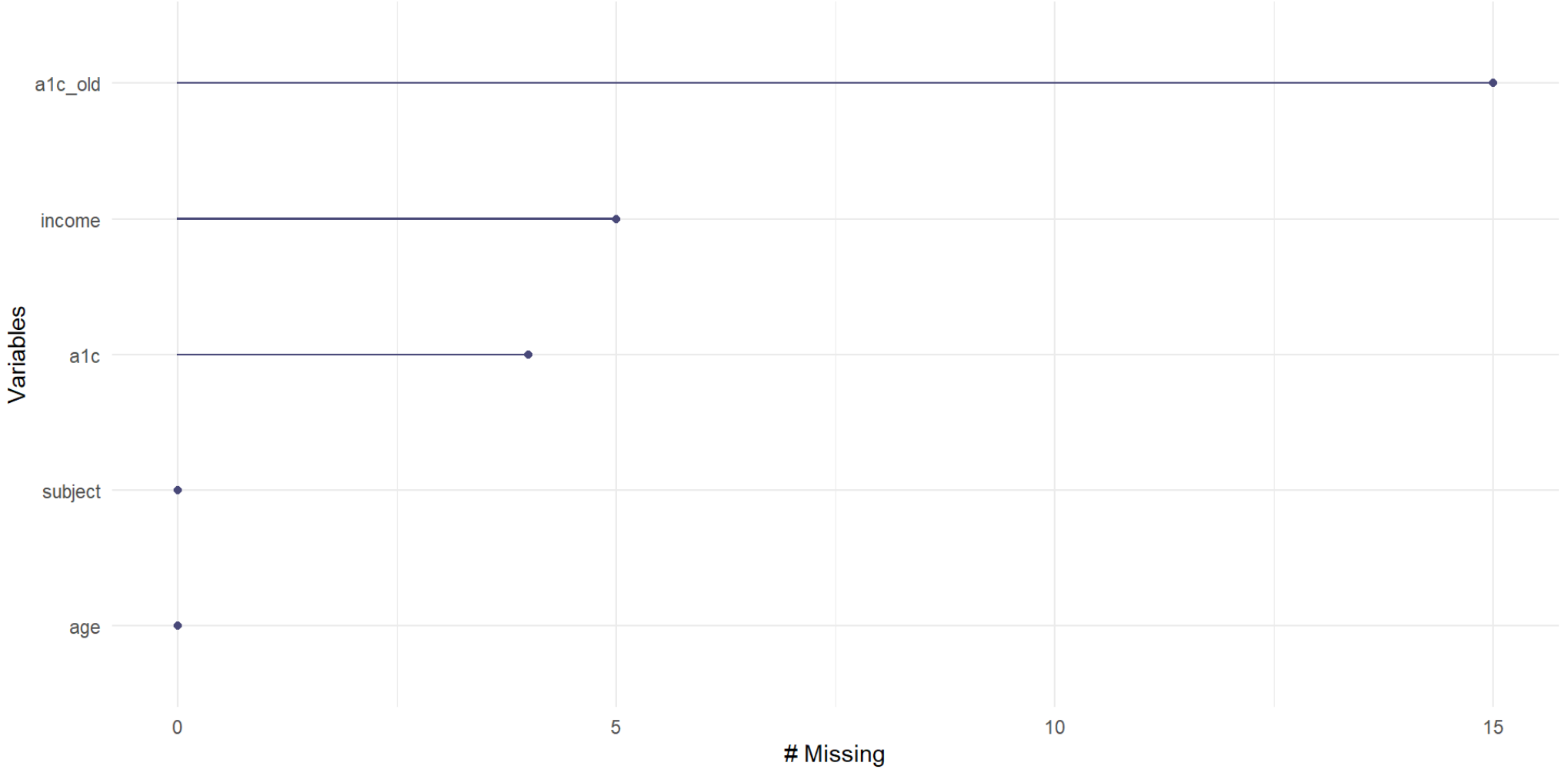
```
1 miss_var_summary(dm1)
```

```
# A tibble: 5 × 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

```
1 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.

```
1 dm1_cc <- dm1 |> drop_na()
2
3 nrow(dm1)
```

```
[1] 500
```

```
1 nrow(dm1_cc)
```

```
[1] 479
```

- In classes 18 and 19, we will drop these 21 subjects, and fit all three 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`.

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.)

Imputing Predictors

- 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

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.)

```
1 tempA <- impute_rlm(dm1, a1c_old ~ age)
2
3 tempA |> miss_var_summary()
```

```
# A tibble: 5 × 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`

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

```
1 tempB <- impute_cart(dm1, income ~ age)
2
3 tempB |> miss_var_summary()
```

```
# A tibble: 5 × 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

- 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`.

Summary of imputed tibble

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

```
1 dm1_imp |> summary()
```

```
      a1c            a1c_old           age                    income
 Min.   : 4.300   Min.   : 4.200   Min.   :31.00   Higher_than_50K:121
 1st Qu.: 6.500   1st Qu.: 6.500   1st Qu.:49.00   Between_30-50K :193
 Median : 7.300   Median : 7.300   Median :56.00   Below_30K      :182
 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
   subject
 Length:496
 Class :character
 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 in Classes 18-19, and return to the imputed data in Class 20.

Which of our three models is "best"?

Our goal is accurate prediction of `a1c` values.

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`

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"

How shall we be guided by our data?

> 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.

George Box's aphorisms

> 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 strategy: "most useful" model?

We'll get through these three steps today.

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 in Class 19.

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 and a model test (holdout) sample.

Partitioning the 479 Complete Cases

- We'll select a random sample (without replacement) of 70% of the data (60-80% 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`.

Partitioning the 479 Complete Cases

```
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")

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

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

Develop candidate models using the development sample.

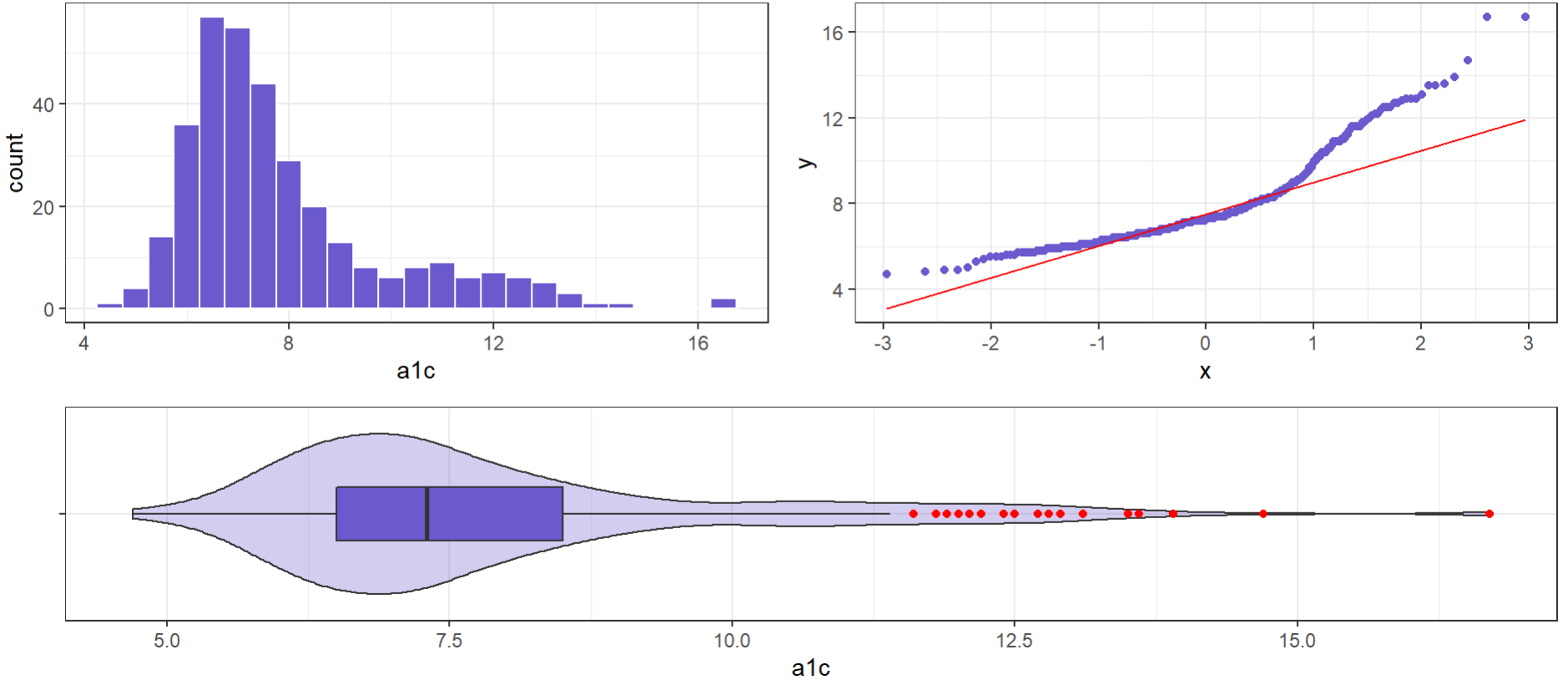
A look at the outcome (a1c) distribution

```
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()

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

A look at the outcome (a1c) distribution

Hemoglobin A1c values (%)

Model Development Sample: 335 adults with diabetes



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.

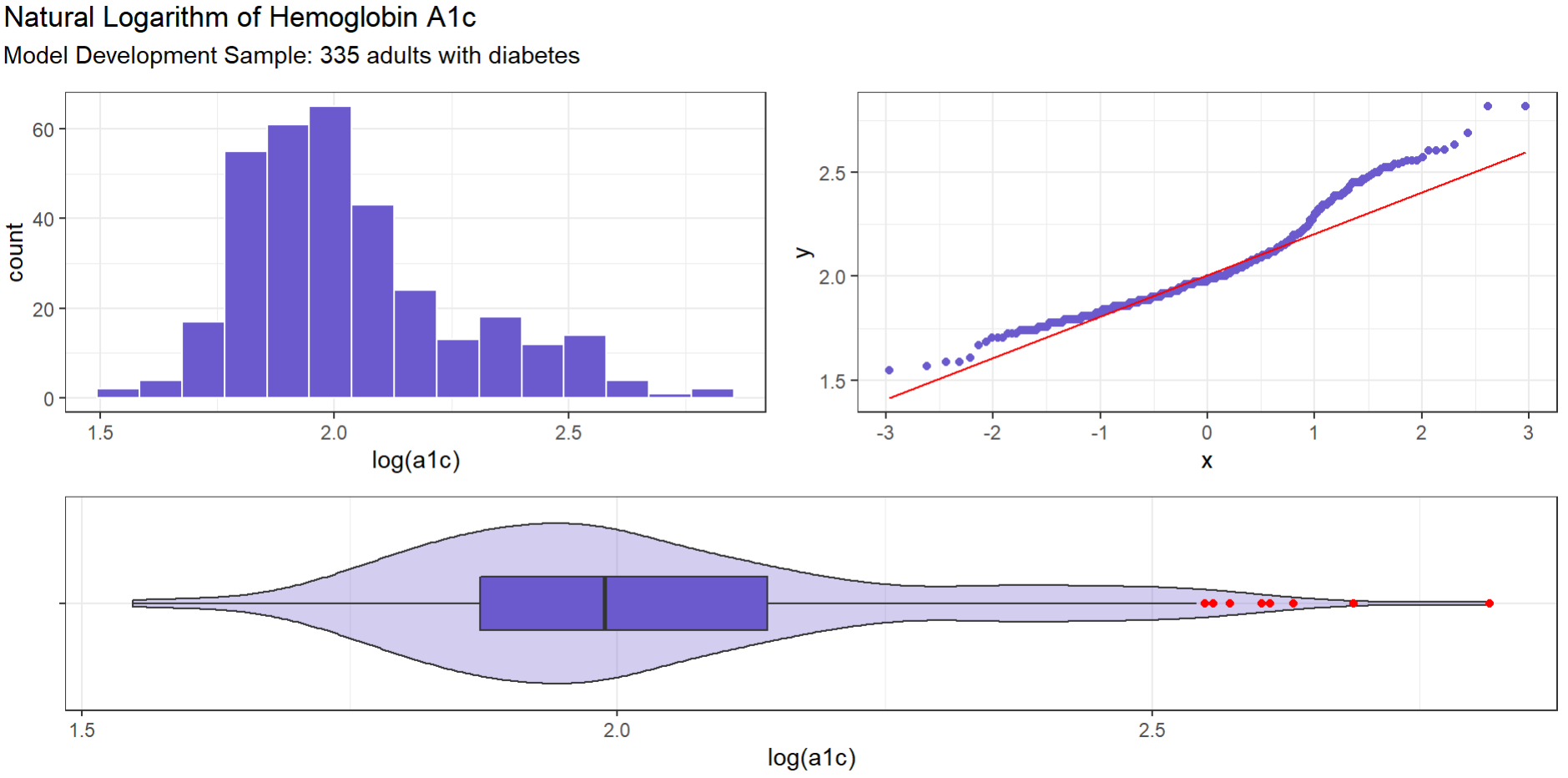
Ladder of Useful (interpretable) transformations

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

Consider a log transformation?

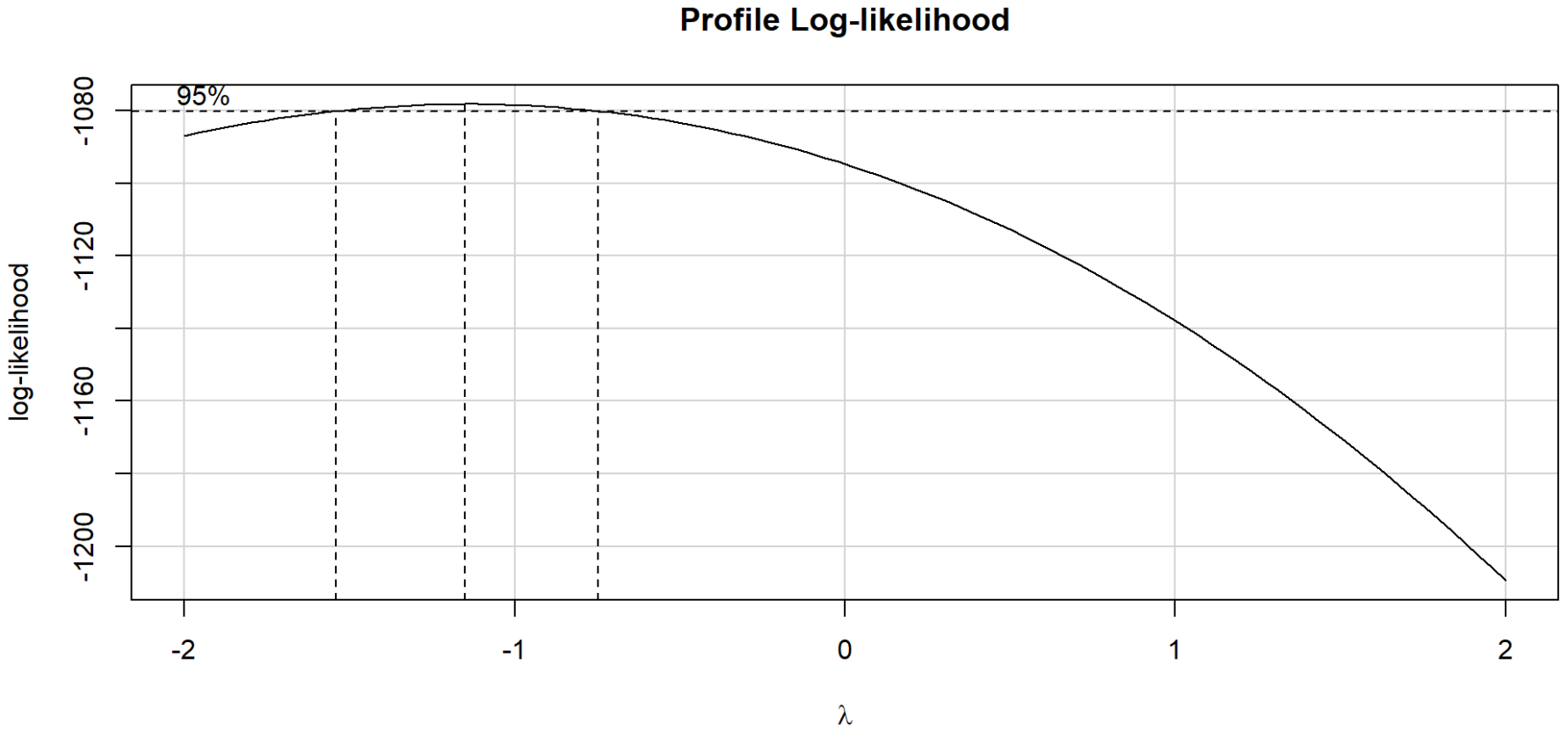
```
p1 <- ggplot(dm1_cc_train, aes(x = log(a1c))) +
  geom_histogram(bins = 15,
                 fill = "slateblue", col = "white")

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

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

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

Consider a log transformation?



Box-Cox to help pick a transformation?

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



Box-Cox to help pick a transformation?

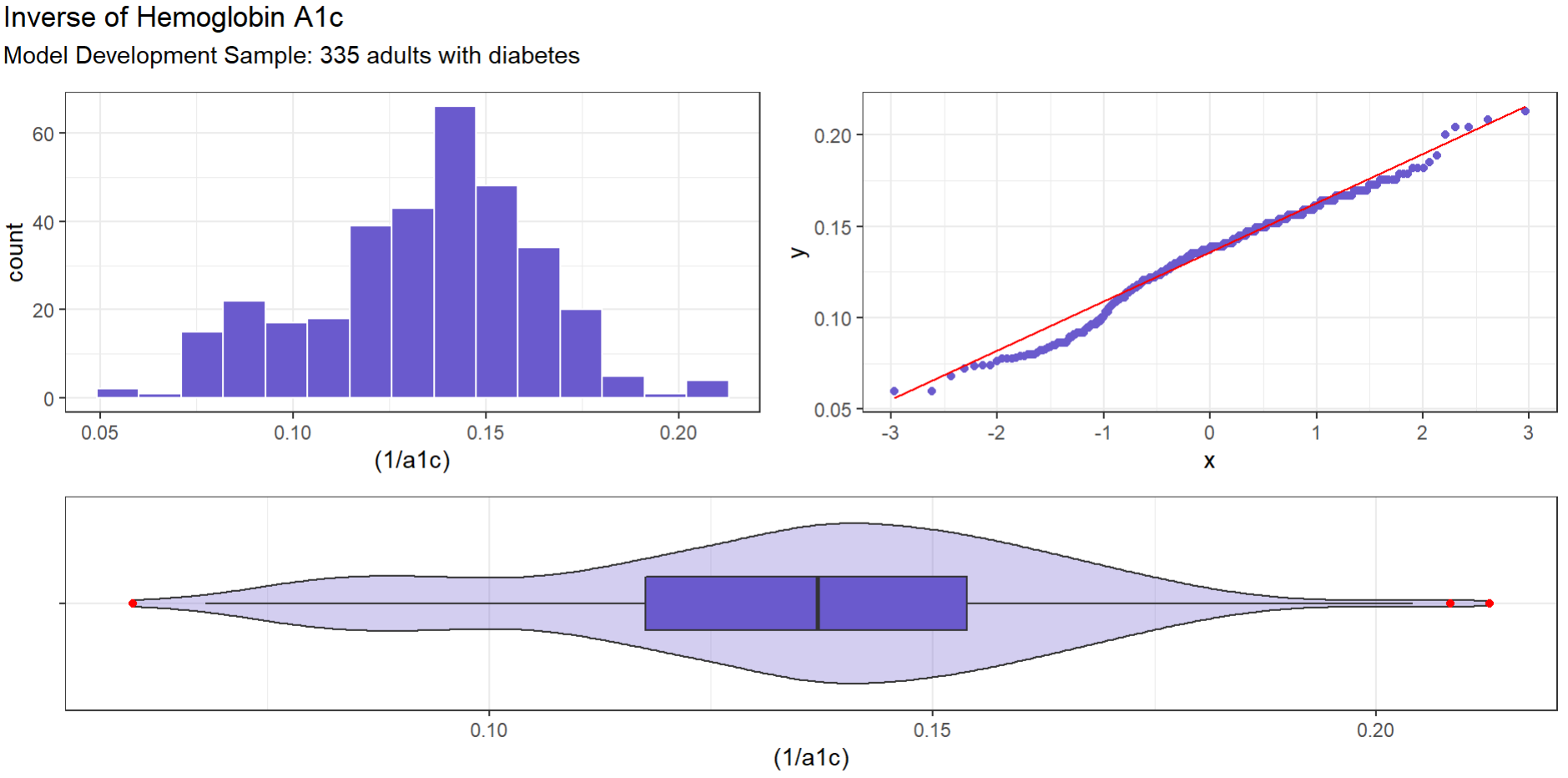
```
1 summary(powerTransform(mod_0))
```

```
bcPower Transformation to Normality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1   -1.1407           -1      -1.5373      -0.7441

Likelihood ratio test that transformation parameter is equal to 0
 (log transformation)
                            LRT df      pval
LR test, lambda = (0) 32.98128  1 9.305e-09

Likelihood ratio test that no transformation is needed
                           LRT df       pval
LR test, lambda = (1) 119.328  1 < 2.22e-16
```

Consider the inverse?

```
p1 <- ggplot(dm1_cc_train, aes(x = (1/a1c))) +
  geom_histogram(bins = 15,
                 fill = "slateblue", col = "white")

p2 <- ggplot(dm1_cc_train, aes(sample = (1/a1c))) +
  geom_qq(col = "slateblue") + geom_qq_line(col = "red")

p3 <- ggplot(dm1_cc_train, aes(x = "", y = (1/a1c))) +
  geom_violin(fill = "slateblue", alpha = 0.3) +
  geom_boxplot(fill = "slateblue", width = 0.3,
               outlier.color = "red") +
  labs(x = "") + coord_flip()

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

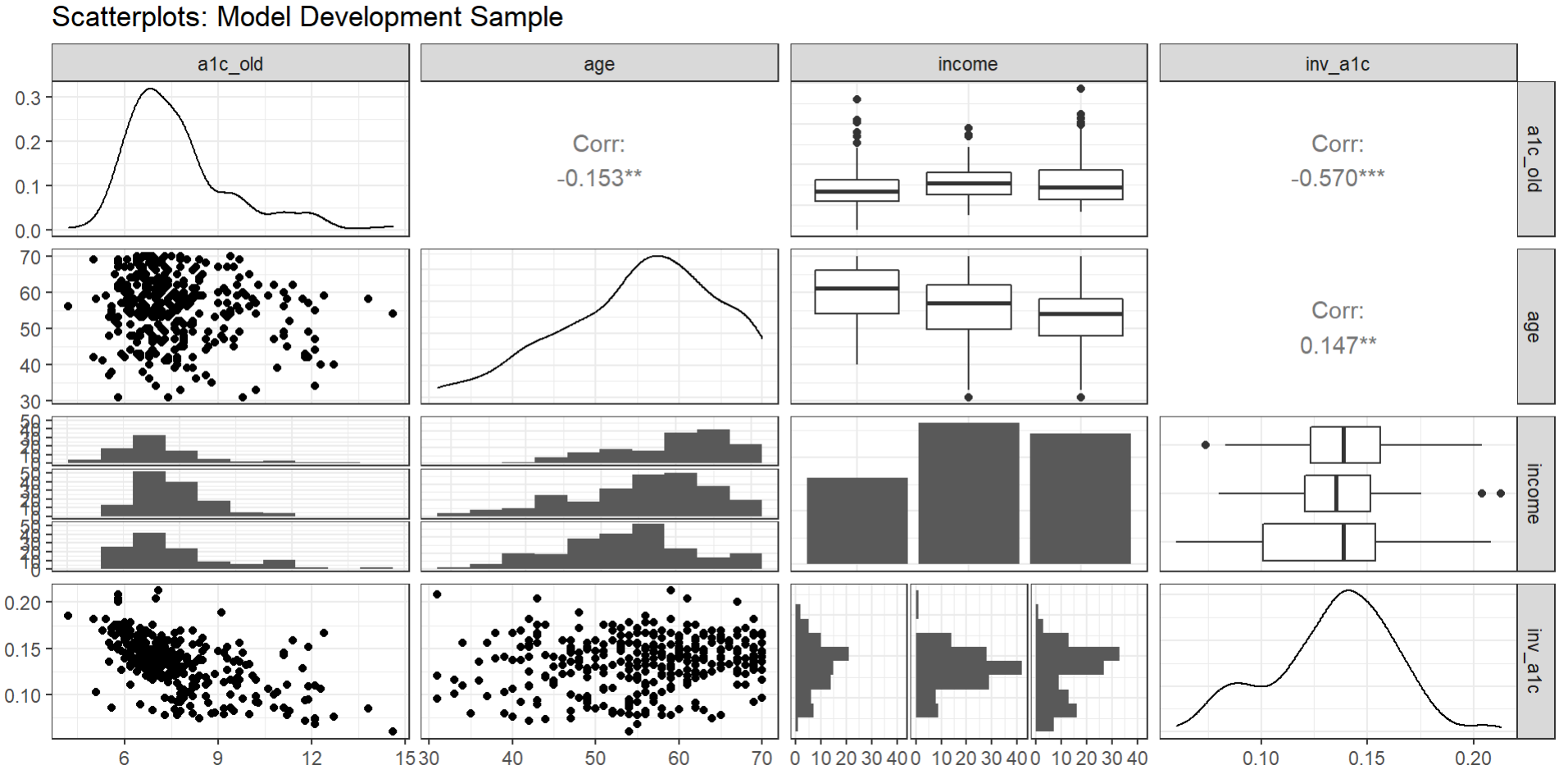
Consider the inverse?



Scatterplot Matrix

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

Scatterplot Matrix



`ggpairs()` for scatterplot matrices

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.
- 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 in 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) |>
  kbl(digits = 4) |>
  kable_classic_2(font_size = 28, full_width = F)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.2074	0.0059	0	0.1957	0.2191
a1c_old	-0.0095	0.0008	0	-0.0110	-0.0080

The Regression Equation (mod_1)

Use the equatiomatic package to help here.

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

$$\widehat{(1/a1c)} = 0.2074 - 0.0095(a1c\_old)$$

Use **results = 'asis'** in the code chunk name if you have trouble with this in R Markdown.

Summary of Fit Quality (mod_1)

```
glance(mod_1) |>
  mutate(name = "mod_1") |>
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) |>
  kbl(digits = c(0, 3, 3, 3, 0, 0)) |>
  kable_minimal(font_size = 28, full_width = F)
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_1	0.325	0.323	0.024	-1558	-1547

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) |>
  kbl(digits = 4) |>
  kable_classic_2(font_size = 28, full_width = F)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.1954	0.0106	0.0000	0.1745	0.2163
a1c_old	-0.0094	0.0008	0.0000	-0.0109	-0.0079
age	0.0002	0.0001	0.1752	-0.0001	0.0005

The Regression Equation (mod_2)

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

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

$$ \widehat{(1/a1c)} = 0.1954 - 0.0094(a1c\_old) + 2e-04(age) $$

Summary of Fit Quality (mod_2)

```
glance(mod_2) |>
  mutate(name = "mod_2") |>
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) |>
  kbl(digits = c(0, 3, 3, 3, 0, 0)) |>
  kable_minimal(font_size = 28, full_width = F)
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_2	0.329	0.325	0.023	-1558	-1543

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) |>
  kbl(digits = 4) |>
  kable_classic_2(font_size = 28, full_width = F)
```

term	estimate	se	low	high	p
(Intercept)	0.1975	0.0113	0.1752	0.2198	0.0000
a1c_old	-0.0093	0.0008	-0.0108	-0.0078	0.0000
age	0.0002	0.0001	-0.0001	0.0005	0.2604
incomeBetween_30-50K	0.0003	0.0034	-0.0063	0.0070	0.9250
incomeBelow_30K	-0.0027	0.0035	-0.0096	0.0042	0.4360

The Regression Equation (mod_3)

```
extract_eq(mod_3, use_coefs = TRUE, coef_digits = 5,
           ital_vars = TRUE, wrap = TRUE, terms_per_line = 1)
```

$$ \begin{aligned} \widehat{(1/a1c)} &= 0.1975\ - \\ &\quad 0.00932(a1c\_old)\ + \\ &\quad 0.00017(age)\ + \\ &\quad 0.00032(income_{Between\_30-50K})\ - \\ &\quad 0.00273(income_{Below\_30K}) \end{aligned} $$

Summary of Fit Quality (mod_3)

```
glance(mod_3) |>
  mutate(name = "mod_3") |>
  select(name, r.squared, adj.r.squared,
         sigma, AIC, BIC) |>
  kbl(digits = c(0, 3, 3, 3, 0, 0)) |>
  kable_minimal(font_size = 28, full_width = F)
```

name	r.squared	adj.r.squared	sigma	AIC	BIC
mod_3	0.331	0.323	0.024	-1555	-1532

Clean Up

```
rm(mod_0, mod_1, mod_2, mod_3,
   p1, p2, p3, temp, tempA, tempB,
   tidy_m1, tidy_m2, tidy_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
```