432 Class 02

Thomas E. Love, Ph.D.

2026-01-15

Today's Agenda

- Fitting linear regression models with `lm`
  - ANOVA: Incorporating an interaction between factors
  - ANCOVA: Incorporating a quantitative covariate
- Regression Diagnostics via `check_model()`
- Validating / evaluating results in a test sample

Appendix

How the `smt_im` data were created from `smart_ohio.csv`

Today's R Setup

```
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(naniar)
library(broom)
library(car)
library(gt)
library(mosaic)          ## for df_stats and favstats
library(mice)            ## imputation of missing data
library(patchwork)
library(rsample)         ## data splitting
library(easystats)
library(tidyverse)

theme_set(theme_lucid())
```


The `smt_im` data

The `smt_im` data

- 894 subjects in Cleveland-Elyria with `bmi` and no history of diabetes (missing values singly imputed: assume MAR)
- All subjects have `hx_diabetes` (all 0), and are located in the `MMSA` labeled Cleveland-Elyria.
- See Course Notes Chapter on BRFSS SMART data for variable details
- Appendix provides details on data development.

The Five Variables We'll Use Today

9 variables in the data but we'll use only these 5 today.

Variable	Description
`ID`	subject identifying code
`bmi`	(outcome) Body-Mass index in kg/m^2.
`exerany`	any exercise in past month: 1 = yes, 0 = no
`health`	self-reported overall health (5 levels)
`fruit_day`	average fruit servings consumed per day

Data Load

```
smt_im <- read_rds("c02/data/smt_im.Rds") |>
  select(ID, bmi, exerany, health, fruit_day, everything())

smt_im
```

```
# A tibble: 894 × 9
   ID      bmi exerany health fruit_day inc_imp drinks_wk female race_eth
   <chr> <dbl>   <dbl> <fct>      <dbl>   <dbl>     <dbl>  <dbl> <fct>
 1 2      23.0       0 E              4      86865      0          1 White non-Hisp…
 2 3      26.9       1 G              3      22921      0          1 Other race non…
 3 4      26.5       1 G              2      40128      4.67       1 White non-Hisp…
 4 5      24.2       1 G              0.57   58311      0.93       0 White non-Hisp…
 5 7      23.0       1 G              2       2318      2          0 White non-Hisp…
 6 8      28.4       1 VG             1      79667      0          0 Other race non…
 7 9      30.1       1 F              0.23   47880      0          0 Black non-Hisp…
 8 10     19.8       1 E              0.77  100136      0.47       1 White non-Hisp…
 9 11     27.2       1 E              0.71   73145      0          0 White non-Hisp…
10 12     24.6       1 E              1.07   76917      0          1 Other race non…
# ℹ 884 more rows
```


Checking our Data

Are there any missing values?

```
smt_im |> n_miss()
```

```
[1] 0
```

Does each row have a unique `ID` value?

```
identical(nrow(smt_im), n_distinct(smt_im$ID))
```

```
[1] TRUE
```

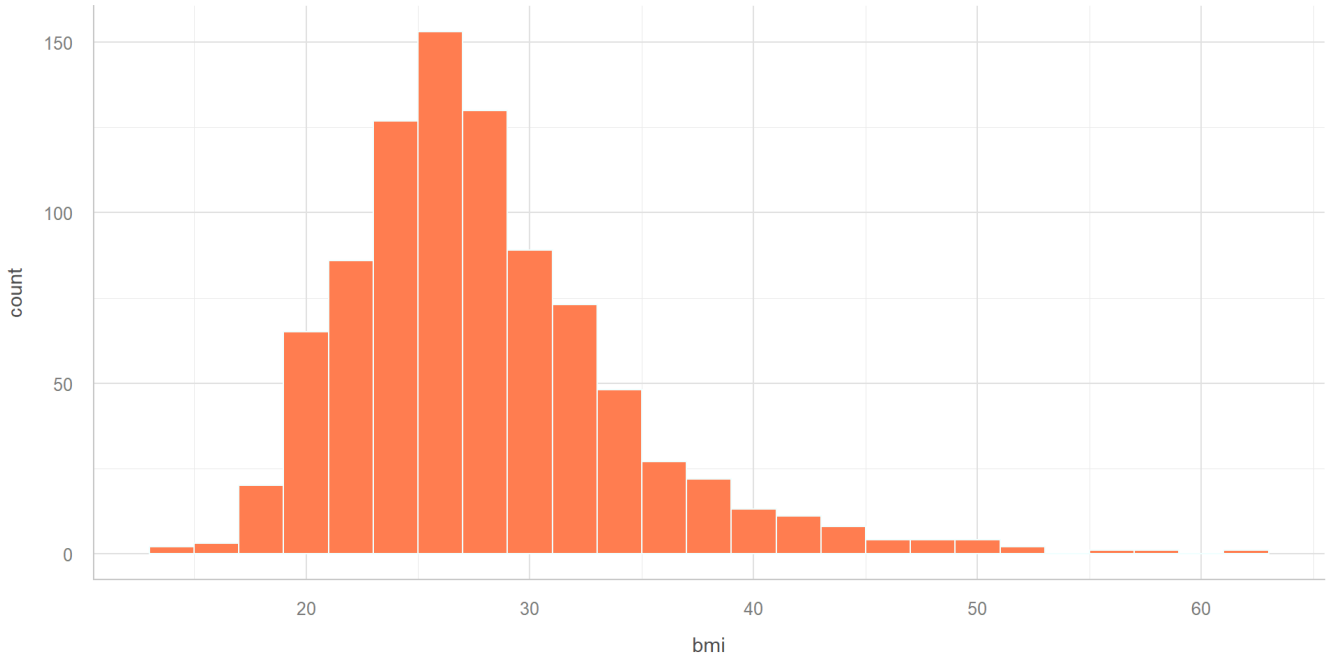
Range and Level Checks?

```
1 data_codebook(smt_im |> select(bmi, exerany, health, fruit_day))
```

```
select(smt_im, bmi, exerany, health, fruit_day) (894 rows and 4 variables, 4 shown)

ID | Name     | Type        | Missings |     Values |           N
---+----------+-------------+----------+------------+------------
1  | bmi      | numeric     | 0 (0.0%) | [13.3, 63] |         894
---+----------+-------------+----------+------------+------------
2  | exerany  | numeric     | 0 (0.0%) |          0 | 233 (26.1%)
   |          |             |          |          1 | 661 (73.9%)
---+----------+-------------+----------+------------+------------
3  | health   | categorical | 0 (0.0%) |          E | 148 (16.6%)
   |          |             |          |         VG | 324 (36.2%)
   |          |             |          |          G | 274 (30.6%)
   |          |             |          |          F | 113 (12.6%)
   |          |             |          |          P |  35 ( 3.9%)
```


Our outcome, bmi

```
1 ggplot(smt_im, aes(x = bmi)) +
2   geom_histogram(binwidth = 2, col = "azure", fill = "coral")
```



Key predictors: `exerany`, `health`

```
smt_im |> tabyl(exerany, health) |>
  adorn_totals(where = c("row", "col")) |>
  gt() |> tab_options(table.font.size = 28)
```

exerany	E	VG	G	F	P	Total
0	24	68	76	51	14	233
1	124	256	198	62	21	661
Total	148	324	274	113	35	894

Here, it doesn't matter much whether we store the 1/0 in `exerany` as numeric or as a two-level factor in R. For binary variables, sometimes the numeric version will be more useful and sometimes a factor will be more useful.

Our covariate, `fruit_day`

We are mostly interested in whether accounting for the quantitative covariate `fruit_day` changes the modeled association of our key predictors with `bmi`.

- Sometimes we center such a covariate (subtracting its mean.)

```
smt_im <- smt_im |>
  mutate(fruit_c = fruit_day - mean(fruit_day))
```

- Why? So that we can easily plug in the covariate's mean (which will now be 0) when making predictions.

Did we center `fruit_day` properly?

Here's a little "sanity check":

```
df_stats(~ fruit_day + fruit_c, data = smt_im) |> gt() |>
  fmt_number(columns = min:sd, decimals = 3) |>
  tab_options(table.font.size = 24)
```

response	min	Q1	median	Q3	max	mean	sd	n	missing
fruit_day	0.000	0.710	1.070	2.000	10.000	1.445	1.129	894	0
fruit_c	−1.445	−0.735	−0.375	0.555	8.555	0.000	1.129	894	0

The `df_stats()` function comes from the `mosaic` package, and can be used to apply `favstats()` to multiple variables at once.

Modeling Plan

1. Split `smt_im` into training and testing samples.
2. Predict `bmi` using `exer_any` and `health`
   - (`fit1`): without an interaction between the predictors
   - (`fit2`): and then with an interaction term
3. (`fit3`): Add in our (centered) covariate, `fruit_c` to `fit1`
4. (`fit4`): Add in our (centered) covariate, `fruit_c` to `fit2`
5. Assess all four models in training and testing samples.

Splitting the Sample

We'll partition our data set using some tools from the `rsample` package, into:

- a training sample containing 75% of the data
- a testing sample containing the remaining 25%

```
set.seed(432)    ## for future replication

smt_im_split <- initial_split(smt_im, prop = 3/4)

train_smt_im <- training(smt_im_split)
test_smt_im <- testing(smt_im_split)

c(nrow(smt_im), nrow(train_smt_im), nrow(test_smt_im))
```

```
[1] 894 670 224
```


Building Our Four Models

Modeling Plan

- Predict `bmi` using `exer_any` and `health`
  - (`fit1`): without an interaction between the predictors
  - (`fit2`): and then with an interaction term
  - (`fit3`): Add in our (centered) covariate, `fruit_c` to `fit1`
  - (`fit4`): Add in our (centered) covariate, `fruit_c` to `fit2`

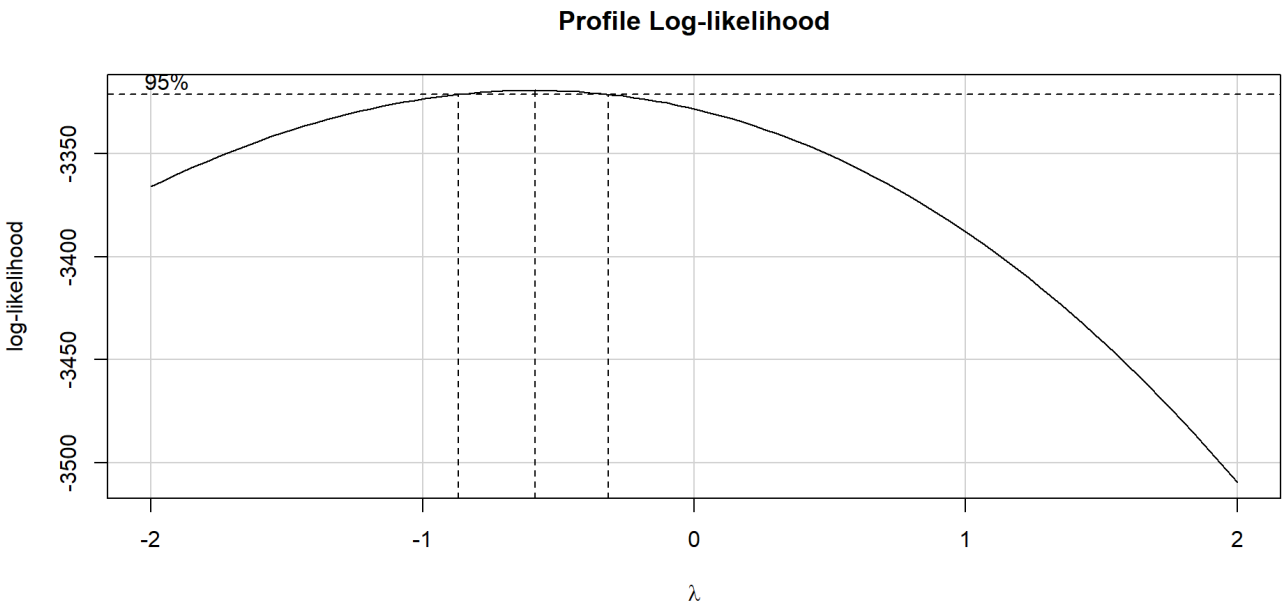
Tukey's transformation ladder

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

- Used in combination with a Box-Cox plot from `car`
- Requires the y variable to be **strictly positive**.
  - If desirable, you can add any constant to y or multiply y by any constant, before or after the transformation.
- Be sure you can back out of the transformation:
- $exp(log(y)) = y, \sqrt{y^2} = y, 1/\frac{1}{y} = y, [1/(\frac{1}{\sqrt{y}})]^2 = y$

Consider transforming `bmi`?

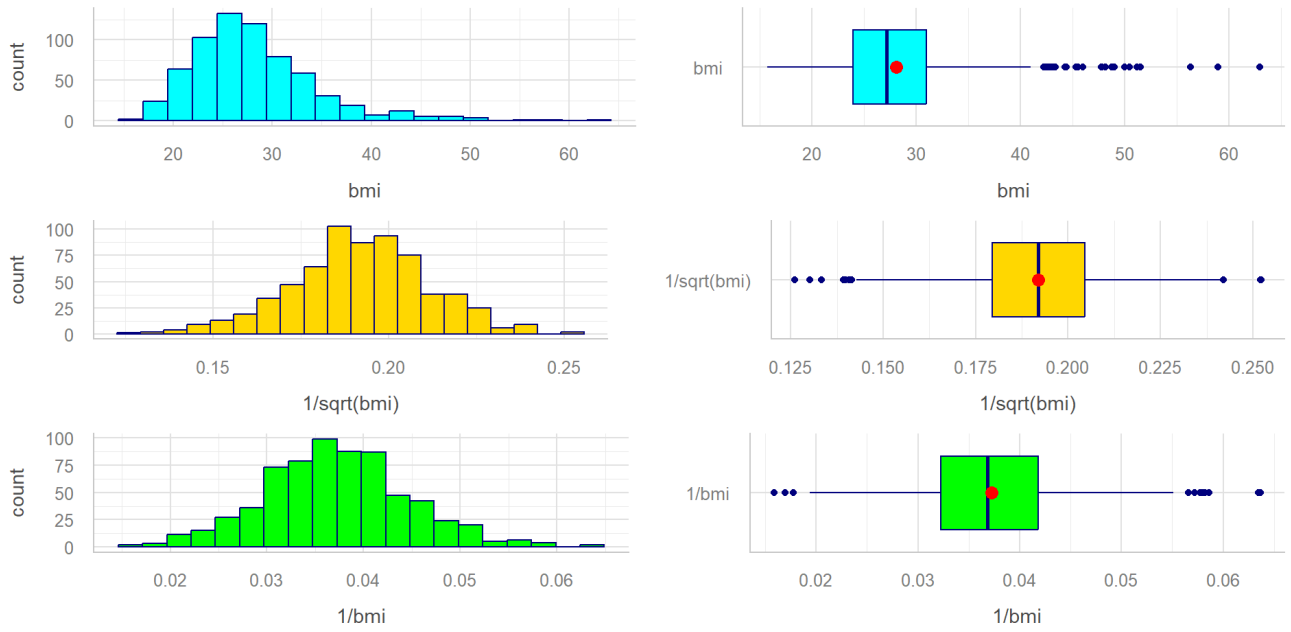
```
m0 <- lm(bmi ~ exerany + health + fruit_c, data = train_smt_im)
boxCox(m0)
```



Should we transform `bmi`? (n = 670)

```
p1a <- ggplot(train_smt_im, aes(x = bmi)) +
    geom_histogram(col = "navy", fill = "cyan", bins = 20)

p1b <- ggplot(train_smt_im, aes(x = bmi, y = "bmi")) +
    geom_boxplot(col = "navy", fill = "cyan") + labs(y = "") +
    stat_summary(geom = "point", col = "red", fun = mean, size = 3)

p2a <- ggplot(train_smt_im, aes(x = 1/sqrt(bmi))) +
    geom_histogram(col = "navy", fill = "gold", bins = 20)

p2b <- ggplot(train_smt_im, aes(x = 1/sqrt(bmi), y = "1/sqrt(bmi)")) +
    geom_boxplot(col = "navy", fill = "gold") + labs(y = "") +
    stat_summary(geom = "point", col = "red", fun = mean, size = 3)

p3a <- ggplot(train_smt_im, aes(x = 1/bmi)) +
    geom_histogram(col = "navy", fill = "green", bins = 20)

p3b <- ggplot(train_smt_im, aes(x = 1/bmi, y = "1/bmi")) +
```

Should we transform `bmi`? (n = 670)



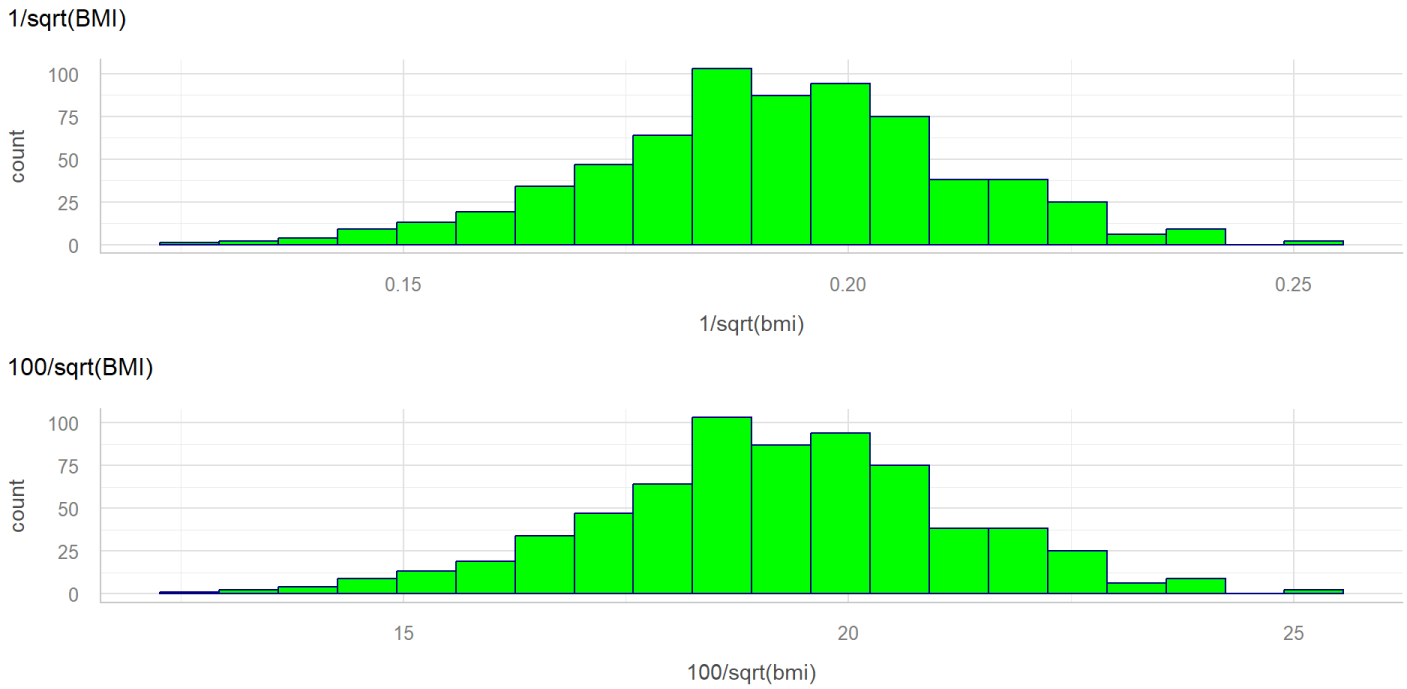
Re-scaling the transformation

To ease interpretation of coefficients, I sometimes scale an outcome transformation so that its values fall in (10, 100), rather than between 0 and 1.

```
bind_rows( favstats(~ 1/sqrt(bmi), data = train_smt_im),
           favstats(~ 100/sqrt(bmi), data = train_smt_im)) |>
  mutate(outcome = c("1/sqrt(bmi)", "100/sqrt(bmi)")) |>
  relocate(outcome) |>
  gt() |> fmt_number(columns = min:sd, decimals = 3) |>
  tab_options(table.font.size = 24)
```

outcome	min	Q1	median	Q3	max	mean	sd	n	missing
1/sqrt(bmi)	0.126	0.180	0.192	0.204	0.252	0.192	0.020	670	0
100/sqrt(bmi)	12.599	17.958	19.194	20.447	25.230	19.195	1.987	670	0

Shape doesn't change



Means by `exerany` and `health`

```
summaries_1 <- train_smt_im |>
    group_by(exerany, health) |>
    summarise(n = n(), mean = mean(100/sqrt(bmi)),
              stdev = sd(100/sqrt(bmi)))
summaries_1
```

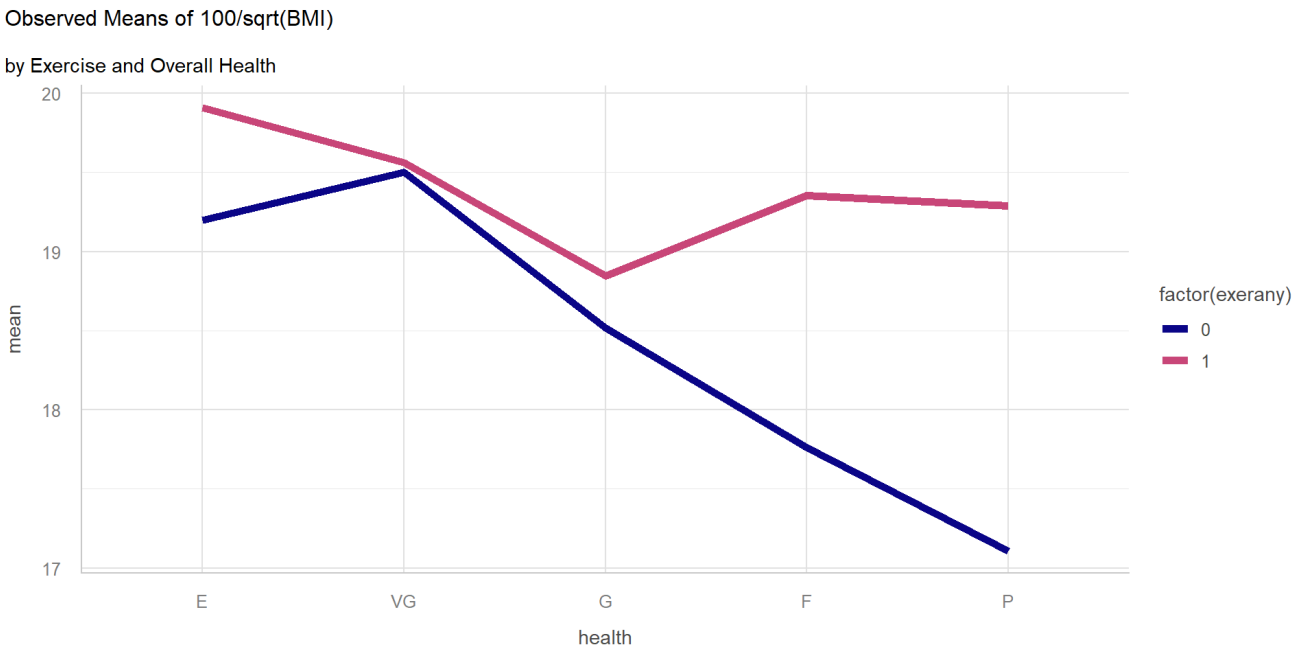
```
# A tibble: 10 × 5
# Groups:   exerany [2]
   exerany health     n  mean stdev
     <dbl> <fct>  <int> <dbl> <dbl>
 1       0 E         19  19.2  1.19
 2       0 VG        54  19.5  1.91
 3       0 G         60  18.5  2.21
 4       0 F         35  17.8  2.57
 5       0 P          8  17.1  2.26
 6       1 E         91  19.9  1.65
 7       1 VG       190  19.6  1.74
 8       1 G        150  18.8  1.91
 9       1 F         46  19.4  1.93
10       1 P         17  19.3  2.70
```

Code for Interaction Plot

```
ggplot(summaries_1, aes(x = health, y = mean, col = factor(exerany))) +
  geom_line(aes(group = factor(exerany)), linewidth = 2) +
  scale_color_viridis_d(option = "C", end = 0.5) +
  labs(title = "Observed Means of 100/sqrt(BMI)",
       subtitle = "by Exercise and Overall Health")
```

- Note the use of `factor` here since the `exerany` variable is in fact numeric, although it only takes the values 1 and 0.
  - Sometimes it's helpful to treat 1/0 as a factor, and sometimes not.
- Where is the evidence of serious non-parallelism (if any) in the plot (see next slide) that results from this code?

Code for Interaction Plot



Fitting a Two-Way ANOVA model for $100/\sqrt{BMI}$

Create our transformed outcome

We'll want to do this in both our training and test samples.

```
train_smt_im <- train_smt_im |> mutate(bmi_tr = 100 / sqrt(bmi))

test_smt_im <- test_smt_im |> mutate(bmi_tr = 100 / sqrt(bmi))
```

Model `fit1` without interaction

```
fit1 <- lm(bmi_tr ~ exerany + health, data = train_smt_im)
```

Using the `tidy()` function from `broom`:

```
tidy(fit1, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals = 3) |>
  tab_options(table.font.size = 24)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	19.315	0.232	83.255	0.000	18.933	19.697
exerany	0.566	0.172	3.302	0.001	0.284	0.849
healthVG	−0.207	0.221	−0.938	0.349	−0.572	0.157
healthG	−0.968	0.227	−4.256	0.000	−1.342	−0.593
healthF	−0.969	0.285	−3.396	0.001	−1.439	−0.499
healthP	−1.110	0.427	−2.599	0.010	−1.814	−0.407

Model Parameters for `fit1`

```
model_parameters(fit1, ci = 0.90)
```

```
Parameter    | Coefficient |   SE |         90% CI | t(664) |      p
-----------------------------------------------------------------------
(Intercept)  |       19.32 | 0.23 | [18.93, 19.70] |  83.26 | < .001
exerany      |        0.57 | 0.17 | [ 0.28,  0.85] |   3.30 | 0.001
health [VG]  |       -0.21 | 0.22 | [-0.57,  0.16] |  -0.94 | 0.349
health [G]   |       -0.97 | 0.23 | [-1.34, -0.59] |  -4.26 | < .001
health [F]   |       -0.97 | 0.29 | [-1.44, -0.50] |  -3.40 | < .001
health [P]   |       -1.11 | 0.43 | [-1.81, -0.41] |  -2.60 | 0.010
```

Model Parameters for `fit1`

Reformatting with `gt()`...

```
model_parameters(fit1, ci = 0.90) |>
  gt() |> fmt_number(columns = -c(CI, df_error), decimals = 3) |>
  tab_options(table.font.size = 24)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	19.315	0.232	0.9	18.933	19.697	83.255	664	0.000
exerany	0.566	0.172	0.9	0.284	0.849	3.302	664	0.001
healthVG	−0.207	0.221	0.9	−0.572	0.157	−0.938	664	0.349
healthG	−0.968	0.227	0.9	−1.342	−0.593	−4.256	664	0.000
healthF	−0.969	0.285	0.9	−1.439	−0.499	−3.396	664	0.001
healthP	−1.110	0.427	0.9	−1.814	−0.407	−2.599	664	0.010

The fit1 equation

`fit1`: $100/\sqrt{bmi}$ = 19.31 + .57 `exerany` - .20 (VG) - .96 (G) - .98 (F) - 1.08 (P)

Name	`exerany`	`health`	predicted $100/\sqrt{bmi}$
Harry	0	Excellent	19.31
Sally	1	Excellent	19.31 + .57 = 19.88
Billy	0	Fair	19.31 - .98 = 18.33
Meg	1	Fair	19.31 + .57 - .98 = 18.90

- Effect of `exerany` on $100/\sqrt{bmi}$?
- Effect of `health` = Fair instead of Excellent?

Does `fit1` fit the training data well?

```
n_obs(fit1)
```

```
[1] 670
```

```
model_performance(fit1)
```

```
# Indices of model performance

AIC    |   AICc |    BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
2786.9 | 2787.1 | 2818.5 | 0.069 |     0.062 | 1.916 | 1.925
```

```
glance(fit1) |>
    select(r.squared, adj.r.squared, sigma, nobs,
           df, df.residual, AIC, BIC) |>
  gt() |> fmt_number(columns = r.squared:sigma, decimals = 3) |>
  fmt_number(columns = AIC:BIC, decimals = 1) |>
  tab_options(table.font.size = 24)
```

r.squared	adj.r.squared	sigma	nobs	df	df.residual	AIC	BIC
0.069	0.062	1.925	670	5	664	2,786.9	2,818.5

Tidied ANOVA for `fit1`

```
tidy(anova(fit1)) |> gt() |>
  fmt_number(columns = sumsq:statistic, decimals = 2) |>
  fmt_number(columns = p.value, decimals = 4) |>
  tab_options(table.font.size = 24)
```

term	df	sumsq	meansq	statistic	p.value
exerany	1	63.76	63.76	17.21	0.0000
health	4	118.17	29.54	7.97	0.0000
Residuals	664	2,460.51	3.71	NA	NA

Model Checks

We'll be checking assumptions related to:

- linearity
- homoscedasticity (constant variance)
- influential observations (outliers, leverage and influence)
- whether the residuals follow a Normal distribution
- collinearity (variance inflation factor)
- and a posterior predictive check of our predictions

My slides and `check_model()`

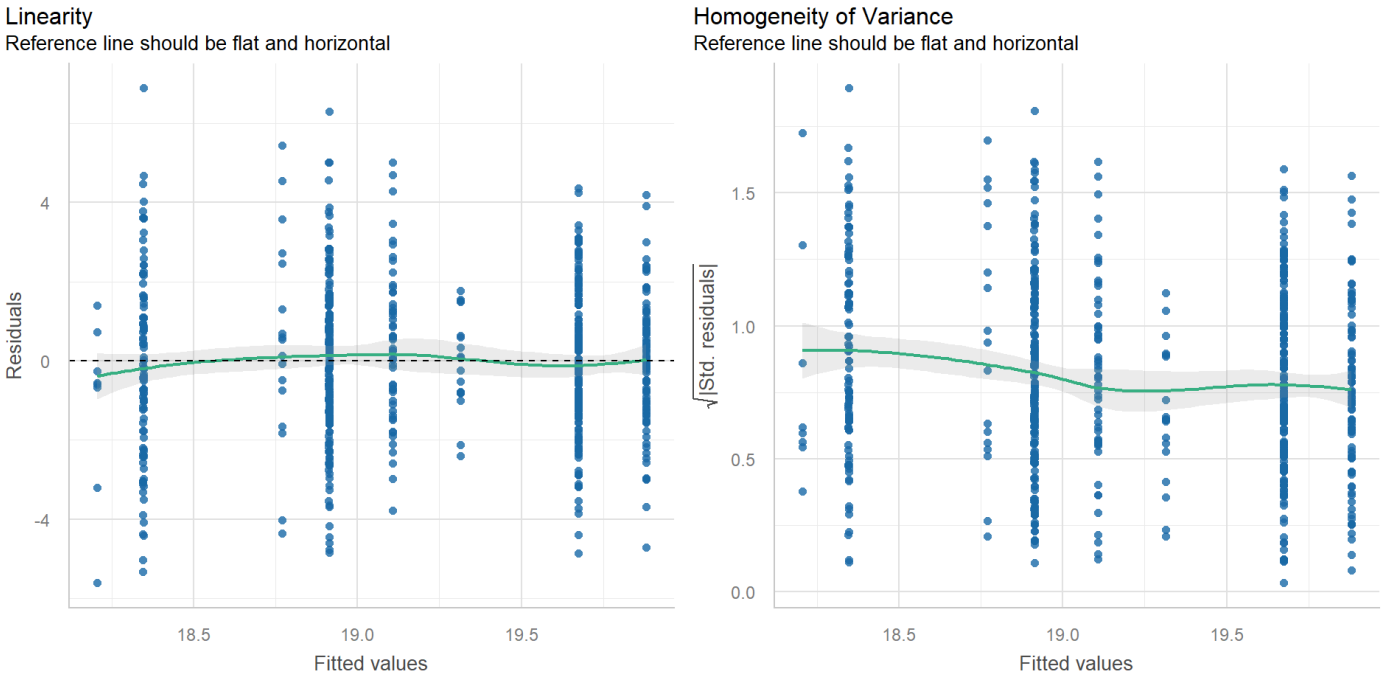
When building a regular HTML file, I would just use:

```
1 check_model(fit1, detrend = FALSE)
```

with `#| fig-height: 9` at the start of the code chunk so that the plots are taller than the default height (thus easier to read) but I will split out the plots for slides.

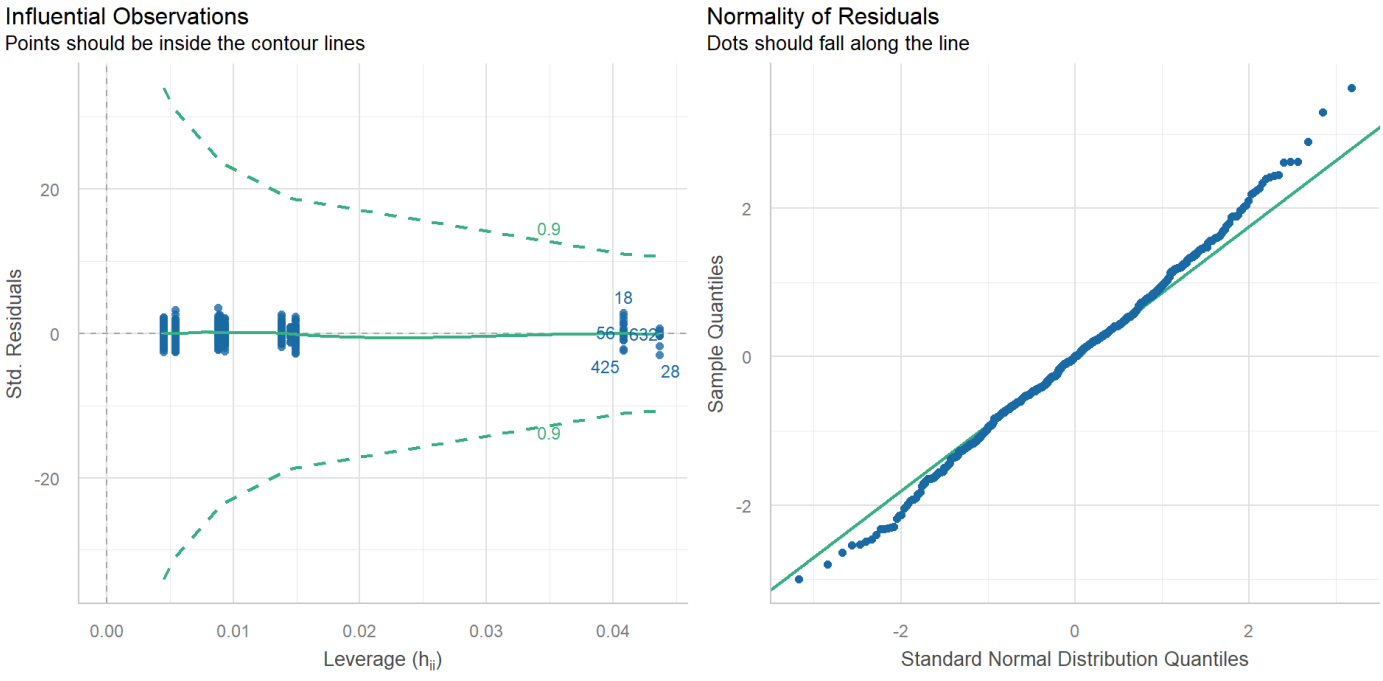
Checking model `fit1` (*n* = 670)

```
1 check_model(fit1, check = c("linearity", "homogeneity"))
```



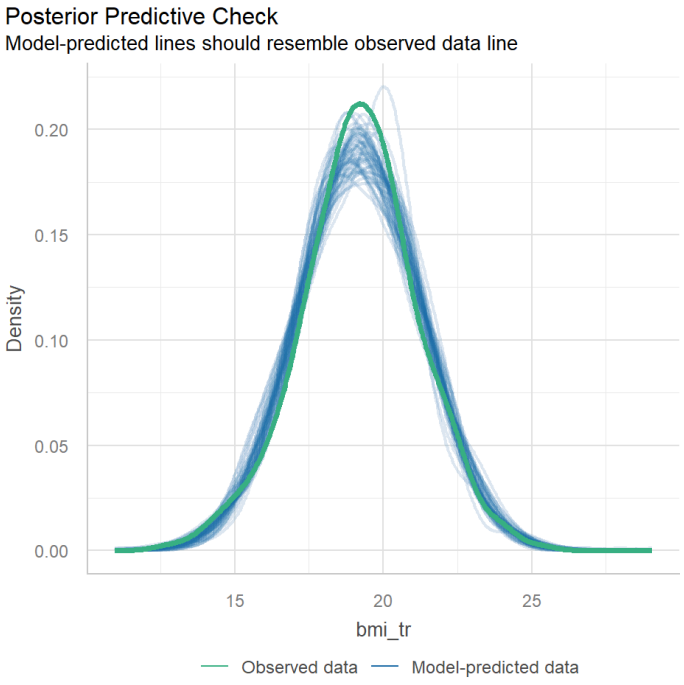
Checking model `fit1` (*n* = 670)

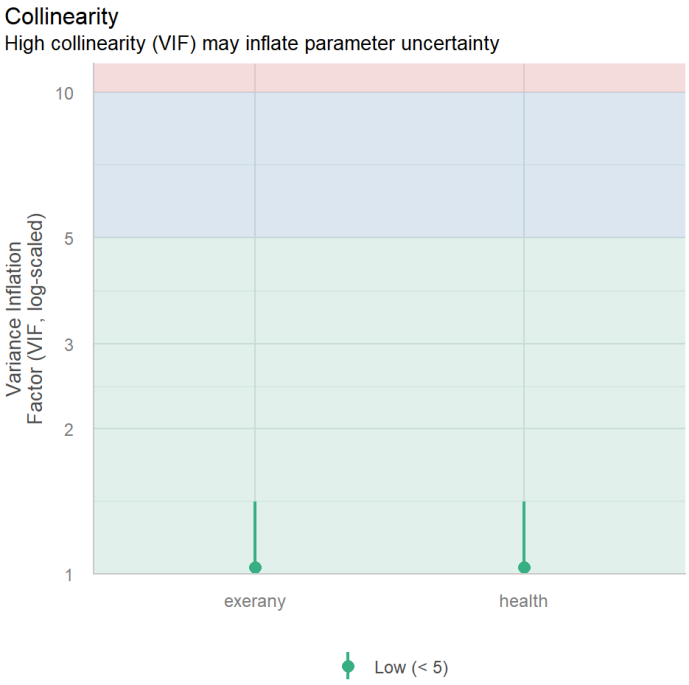
```
1 check_model(fit1, check = c("outliers", "qq"), detrend = FALSE)
```



Checking model `fit1` (*n* = 670)

```
1  check_model(fit1, check = c("pp_check", "vif"))
```





Standing Break

Fitting ANOVA model `fit2` including interaction

Adding the interaction term to `fit1`

```
fit2 <- lm(bmi_tr ~ exerany * health, data = train_smt_im)
```

- How do our models compare on fit to the training data?

```
bind_rows(glance(fit1), glance(fit2)) |>
  mutate(mod = c("fit1", "fit2")) |>
  select(mod, r.sq = r.squared, adj.r.sq = adj.r.squared,
         sigma, nobs, df, df.res = df.residual, AIC, BIC) |>
  gt() |> fmt_number(columns = r.sq:sigma, decimals = 3) |>
  fmt_number(columns = AIC:BIC, decimals = 1) |>
  tab_options(table.font.size = 24)
```

mod	r.sq	adj.r.sq	sigma	nobs	df	df.res	AIC	BIC
fit1	0.069	0.062	1.925	670	5	664	2,786.9	2,818.5
fit2	0.087	0.075	1.912	670	9	660	2,781.6	2,831.2

ANOVA for the `fit2` model

```
tidy(anova(fit2)) |> gt() |>
  fmt_number(columns = sumsq:statistic, decimals = 2) |>
  fmt_number(columns = p.value, decimals = 4) |>
  tab_options(table.font.size = 20)
```

term	df	sumsq	meansq	statistic	p.value
exerany	1	63.76	63.76	17.45	0.0000
health	4	118.17	29.54	8.08	0.0000
exerany:health	4	48.44	12.11	3.31	0.0106
Residuals	660	2,412.06	3.65	NA	NA

ANOVA test comparing `fit1` to `fit2`

```
anova(fit1, fit2)
```

```
Analysis of Variance Table

Model 1: bmi_tr ~ exerany + health
Model 2: bmi_tr ~ exerany * health
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1    664 2460.5
2    660 2412.1  4    48.444 3.3139 0.01059 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`fit2` coefficients

```
tidy(fit2, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals = 3) |>
  tab_options(table.font.size = 20)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	19.196	0.439	43.770	0.000	18.474	19.919
exerany	0.710	0.482	1.472	0.141	−0.084	1.504
healthVG	0.303	0.510	0.593	0.553	−0.537	1.142
healthG	−0.678	0.503	−1.347	0.179	−1.507	0.151
healthF	−1.432	0.545	−2.629	0.009	−2.330	−0.535
healthP	−2.089	0.806	−2.593	0.010	−3.416	−0.762
exerany:healthVG	−0.646	0.565	−1.143	0.253	−1.577	0.285
exerany:healthG	−0.383	0.564	−0.680	0.497	−1.312	0.545
exerany:healthF	0.881	0.645	1.366	0.173	−0.182	1.944
exerany:healthP	1.471	0.951	1.546	0.122	−0.096	3.037

Interpreting the `fit2` model

Name	`exerany`	`health`	predicted $100/\sqrt{bmi}$
Harry	0	Excellent	19.2
Sally	1	Excellent	19.2 + .71 = 19.91
Billy	0	Fair	19.2 - 1.43 = 17.77
Meg	1	Fair	19.2 + .71 - 1.43 + 0.88 = 19.36

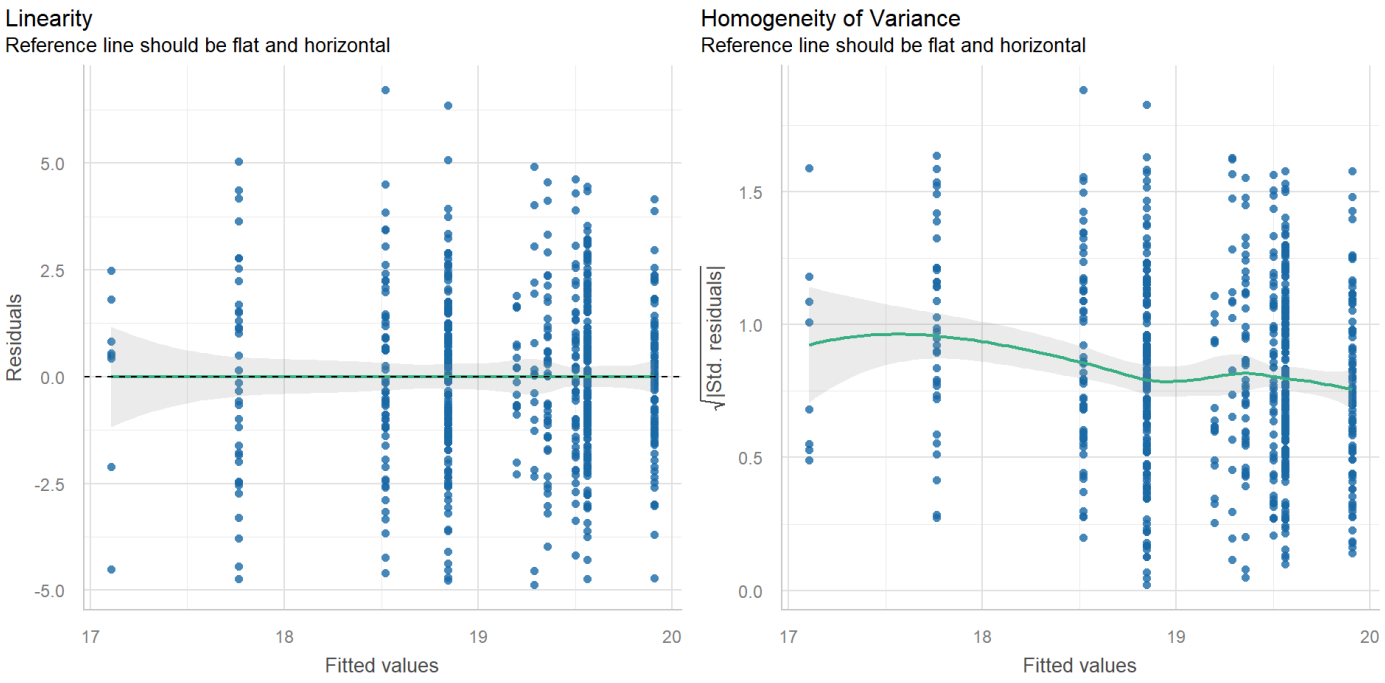
- How do we interpret effect sizes here? **It depends**…

Interpreting the `fit2` model

- Effect of `exerany` on predicted $100/\sqrt{bmi}$?
  - If `health` = Excellent, effect is +0.71
  - If `health` = Fair, effect is (0.71 + 0.88) = +1.59
- Effect of `health` = Fair instead of Excellent?
  - If `exerany` = 0 (no), effect is -1.43
  - If `exerany` = 1 (yes), effect is (-1.43 + 0.88) = -0.55

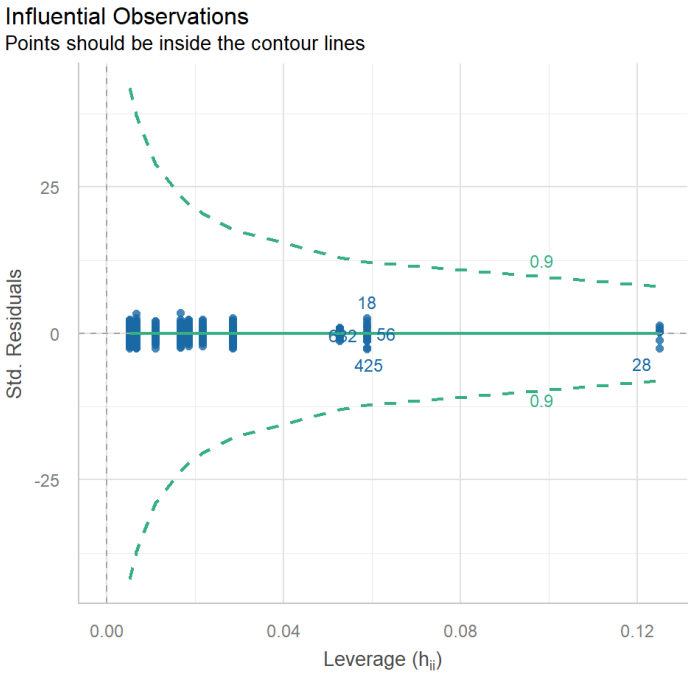
Checking model `fit2` (*n* = 670)

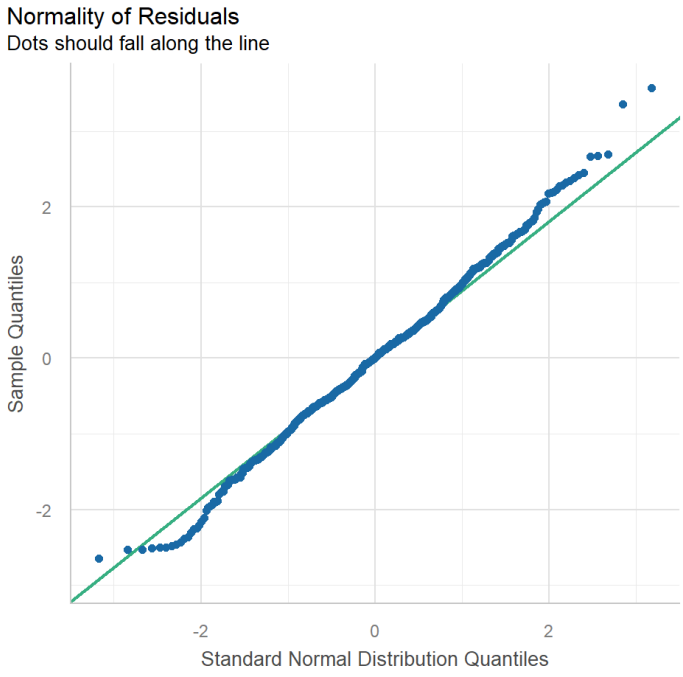
```
1  check_model(fit2, check = c("linearity", "homogeneity"))
```



Checking model `fit2` (*n* = 670)

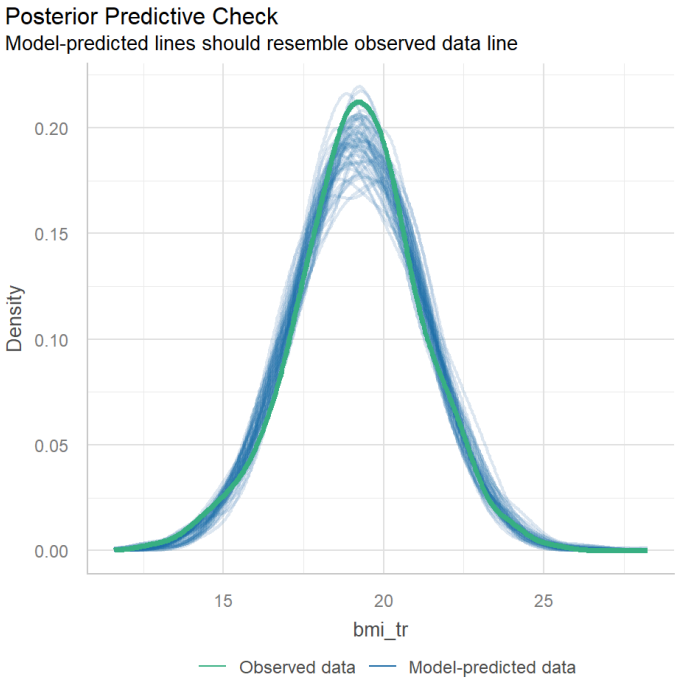
```
1 check_model(fit2, check = c("outliers", "qq"), detrend = FALSE)
```

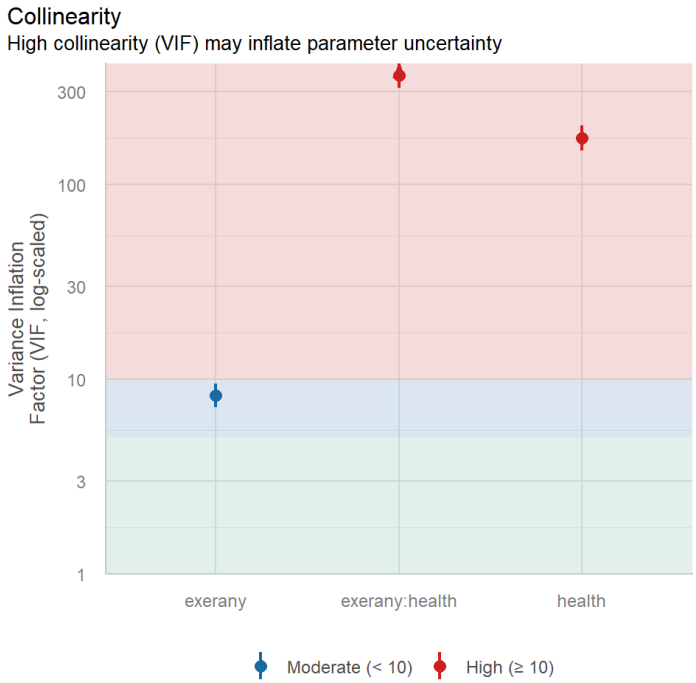




Checking model `fit2` (*n* = 670)

```
1 check_model(fit2, check = c("pp_check", "vif"))
```





Incorporating a Covariate into our two-way ANOVA models

Add `fruit_c` to `fit1`

```
fit3 <- lm(bmi_tr ~ fruit_c + exerany + health, data = train_smt_im)
```

- How well does this model fit the training data?

```
bind_rows(glance(fit1), glance(fit3)) |>
  mutate(mod = c("fit1", "fit3")) |>
  select(mod, r.sq = r.squared, adj.r.sq = adj.r.squared,
         sigma, df, df.res = df.residual, AIC, BIC) |>
  gt() |> fmt_number(columns = r.sq:sigma, decimals = 3) |>
  fmt_number(columns = AIC:BIC, decimals = 1) |>
  tab_options(table.font.size = 24)
```

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
fit1	0.069	0.062	1.925	5	664	2,786.9	2,818.5
fit3	0.079	0.071	1.916	6	663	2,781.4	2,817.4

ANOVA for the `fit3` model

```
tidy(anova(fit3)) |> gt() |>
  fmt_number(columns = sumsq:statistic, decimals = 2) |>
  fmt_number(columns = p.value, decimals = 4) |>
  tab_options(table.font.size = 24)
```

term	df	sumsq	meansq	statistic	p.value
fruit_c	1	48.95	48.95	13.34	0.0003
exerany	1	51.09	51.09	13.92	0.0002
health	4	109.59	27.40	7.47	0.0000
Residuals	663	2,432.81	3.67	NA	NA

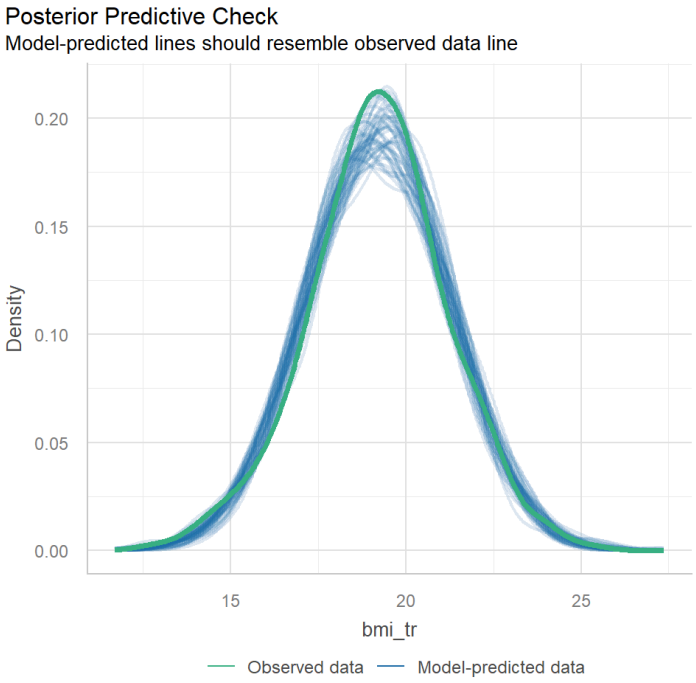
`fit3` coefficients

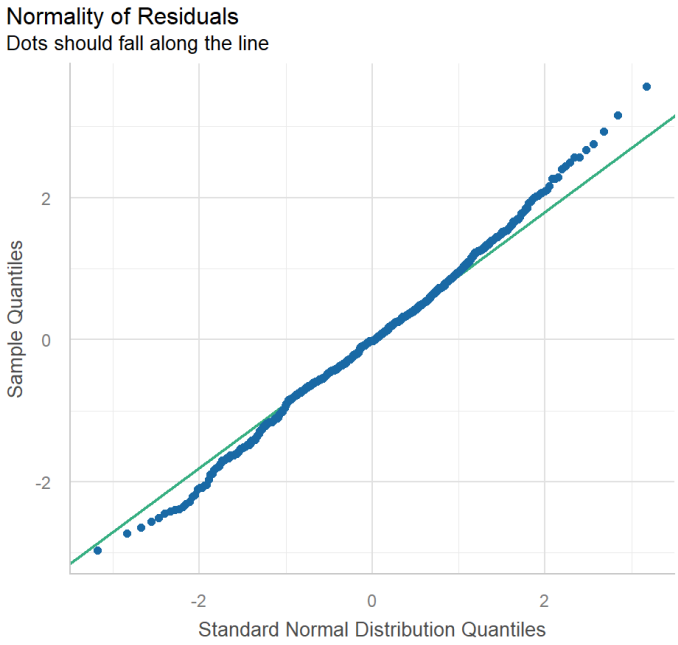
```
tidy(fit3, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals = 3) |>
  tab_options(table.font.size = 24)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	19.325	0.231	83.697	0.000	18.944	19.705
fruit_c	0.178	0.065	2.747	0.006	0.071	0.285
exerany	0.512	0.172	2.982	0.003	0.229	0.795
healthVG	−0.191	0.220	−0.870	0.385	−0.554	0.171
healthG	−0.926	0.227	−4.085	0.000	−1.300	−0.553
healthF	−0.914	0.285	−3.213	0.001	−1.383	−0.446
healthP	−1.119	0.425	−2.632	0.009	−1.819	−0.419

Checking model `fit3` (n = 670)

```
1 check_model(fit3, detrend = FALSE, check = c("pp_check", "qq"))
```





Include the interaction term?

```
1 fit4 <- lm(bmi_tr ~ fruit_c + exerany * health,
2            data = train_smt_im)
```

ANOVA for the `fit4` model

```
1 tidy(anova(fit4)) |> gt() |>
2   fmt_number(columns = sumsq:statistic, decimals = 2) |>
3   fmt_number(columns = p.value, decimals = 4) |>
4   tab_options(table.font.size = 20)
```

term	df	sumsq	meansq	statistic	p.value
fruit_c	1	48.95	48.95	13.55	0.0003
exerany	1	51.09	51.09	14.15	0.0002
health	4	109.59	27.40	7.59	0.0000
exerany:health	4	53.09	13.27	3.68	0.0057
Residuals	659	2,379.72	3.61	NA	NA

`fit4` coefficients

```
tidy(fit4, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals = 3) |>
  tab_options(table.font.size = 18)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	19.219	0.436	44.079	0.000	18.501	19.938
fruit_c	0.193	0.065	2.993	0.003	0.087	0.300
exerany	0.636	0.480	1.325	0.186	−0.155	1.427
healthVG	0.325	0.507	0.641	0.522	−0.510	1.160
healthG	−0.643	0.500	−1.285	0.199	−1.467	0.181
healthF	−1.403	0.542	−2.590	0.010	−2.295	−0.511
healthP	−2.207	0.802	−2.753	0.006	−3.528	−0.887
exerany:healthVG	−0.653	0.562	−1.163	0.245	−1.579	0.272
exerany:healthG	−0.372	0.560	−0.663	0.507	−1.295	0.551
exerany:healthF	0.927	0.642	1.444	0.149	−0.130	1.984
exerany:healthP	1.627	0.947	1.719	0.086	0.068	3.187

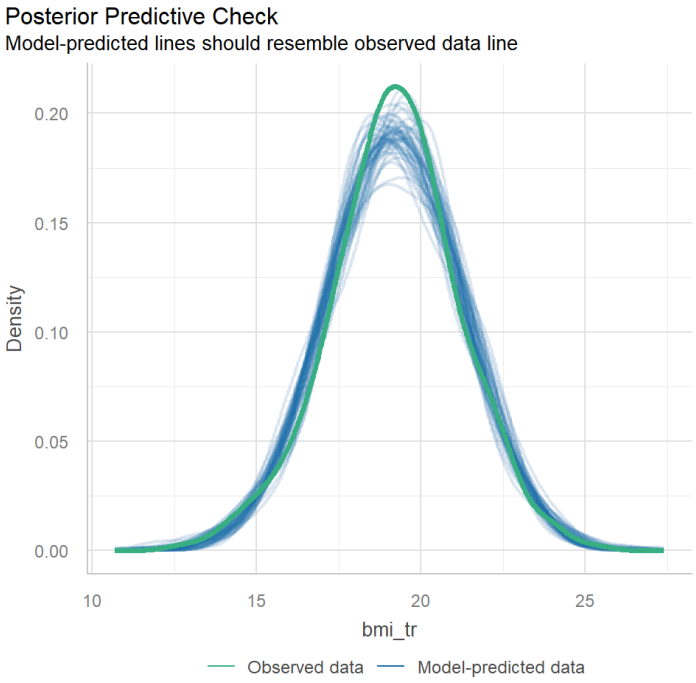
ANOVA: Compare `fit3` & `fit4`

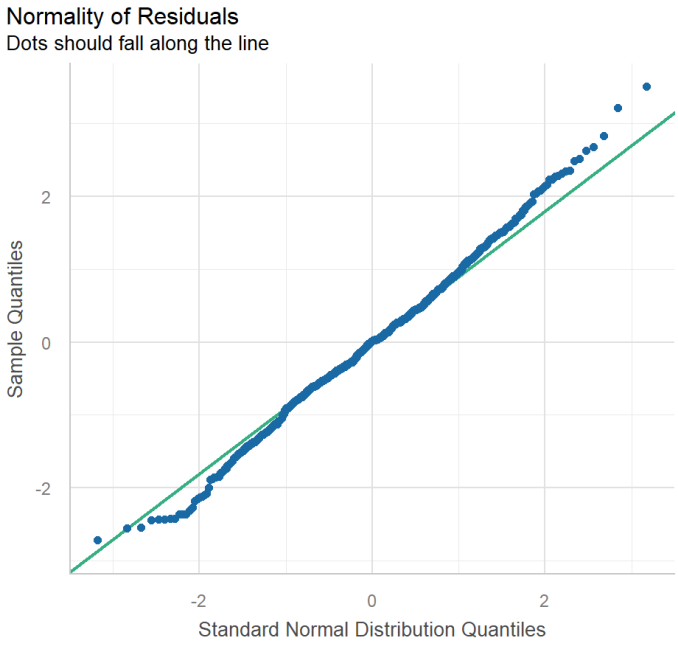
```
anova(fit3, fit4)
```

```
Analysis of Variance Table

Model 1: bmi_tr ~ fruit_c + exerany + health
Model 2: bmi_tr ~ fruit_c + exerany * health
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1    663 2432.8
2    659 2379.7  4    53.092 3.6756 0.005699 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Checking model `fit4` (*n* = 670)

```
1  check_model(fit4, detrend = FALSE, check = c("pp_check", "qq"))
```





Comparing Our Models

Which of the four models fits best?

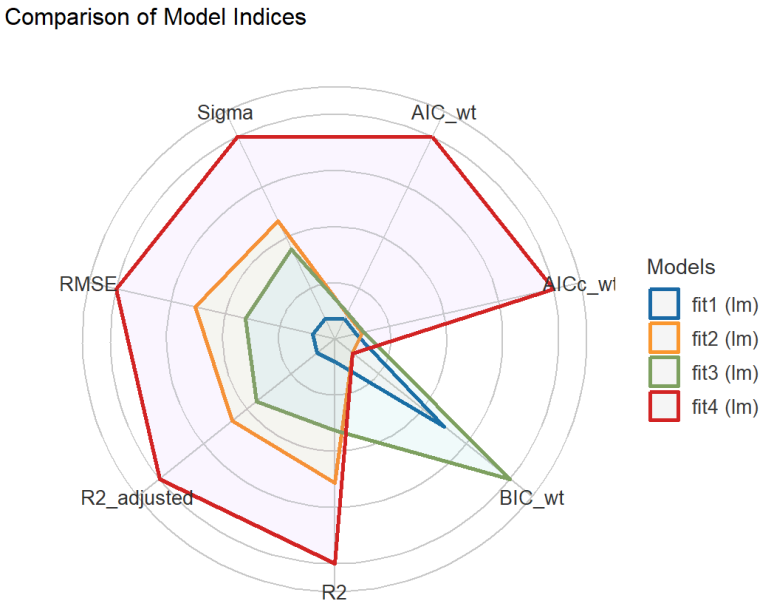
In the **training** sample, our model results (note ordering):

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
fit1	0.069	0.062	1.925	5	664	2,786.9	2,818.5
fit3	0.079	0.071	1.916	6	663	2,781.4	2,817.4
fit2	0.087	0.075	1.912	9	660	2,781.6	2,831.2
fit4	0.099	0.086	1.900	10	659	2,774.6	2,828.7

- Adjusted R^2, σ and AIC all improve as we move down this table. BIC likes `fit1` and `fit3`.
- The training sample is the data our models have *already seen*, so we should be cautious.

Comparison Plot: Training Sample

```
1 plot(compare_performance(fit1, fit2, fit3, fit4))
```



What does `augment()` give us?

```
fit1_test_aug <- augment(fit1, newdata = test_smt_im)
fit1_test_aug |> select(ID, bmi_tr, bmi, .fitted, .resid, health, exerany) |>
  slice(198:202) |> gt() |>
  fmt_number(columns = bmi_tr:.resid, decimals = 2) |>
  tab_options(table.font.size = 20)
```

ID	bmi_tr	bmi	.fitted	.resid	health	exerany
1016	18.75	28.44	18.20	0.55	P	0
1018	19.36	26.68	19.88	−0.52	E	1
1019	19.71	25.74	18.91	0.80	F	1
1020	22.05	20.57	19.67	2.37	VG	1
1024	20.19	24.52	18.35	1.85	G	0

Here, `.fitted` = predicted `bmi_tr` and `.resid` = `bmi_tr` - `.fitted`.

Back-Transformation of `bmi_tr`

Our models predict `bmi_tr` = $100/\sqrt{bmi}$, but we want to predict `bmi`. How do we convert predicted $100/\sqrt{bmi}$ to predicted `bmi`?

$$1/\left(\frac{100}{\sqrt{bmi}}\right) = \sqrt{bmi}/100,$$

$$\text{so } 100/\left(\frac{100}{\sqrt{bmi}}\right) = \sqrt{bmi},$$

$$\text{and so } \left[100/\left(\frac{100}{\sqrt{bmi}}\right)\right]^2 = bmi$$

`augment()` with results for `bmi`

We use $\left(\frac{100}{.fitted}\right)^2$ for predicted `bmi`, then errors are `bmi_res` = observed `bmi` - predicted `bmi`.

```
fit1_test_aug <- augment(fit1, newdata = test_smt_im) |>
  mutate(bmi_fit = (100/.fitted)^2, bmi_res = bmi - bmi_fit)

fit1_test_aug |> select(ID, bmi, bmi_fit, bmi_res,
      bmi_tr, .fitted, .resid, exerany, health, fruit_c) |>
  slice(5:6) |> gt() |>
  fmt_number(columns = c(bmi:.resid, fruit_c), decimals = 2) |>
  tab_options(table.font.size = 24)
```

ID	bmi	bmi_fit	bmi_res	bmi_tr	.fitted	.resid	exerany	health	fruit_c
21	18.83	25.84	−7.01	23.04	19.67	3.37	1	VG	−0.45
29	44.66	29.71	14.95	14.96	18.35	−3.38	0	G	−0.74

Augment all four models so far...

```
fit1_test_aug <- augment(fit1, newdata = test_smt_im) |>
  mutate(bmi_fit = (100/.fitted)^2, bmi_res = bmi - bmi_fit)

fit2_test_aug <- augment(fit2, newdata = test_smt_im) |>
  mutate(bmi_fit = (100/.fitted)^2, bmi_res = bmi - bmi_fit)

fit3_test_aug <- augment(fit3, newdata = test_smt_im) |>
  mutate(bmi_fit = (100/.fitted)^2, bmi_res = bmi - bmi_fit)

fit4_test_aug <- augment(fit4, newdata = test_smt_im) |>
  mutate(bmi_fit = (100/.fitted)^2, bmi_res = bmi - bmi_fit)
```

Four Key Error Summaries

We'll look at all four of these summaries when we do linear regression, usually.

- Mean absolute prediction error (MAPE)
- Maximum absolute prediction error (Max. Error)
- Square root of mean squared prediction error (RMSPE)
- Squared correlation of observed and predicted `bmi` (validated R^2)

Key Summaries for `fit1`

```
fit1_esum <- fit1_test_aug |>
  summarise(MAPE = mean(abs(bmi_res)),
            Max_E = max(abs(bmi_res)),
            RMSPE = sqrt(mean(bmi_res^2)),
            Val_R2 = cor(bmi, bmi_fit)^2) |>
  mutate(Model = "fit1")

fit1_esum
```

```
# A tibble: 1 × 5
   MAPE Max_E RMSPE Val_R2 Model
  <dbl> <dbl> <dbl>  <dbl> <chr>
1  4.30  21.1  5.62 0.0777 fit1
```

- I built the key summaries for `fit2`, `fit3` and `fit4` in the same way (included in code, not shown in slides.)

Compare Models in Test Sample

```
bind_rows(fit1_esum, fit2_esum, fit3_esum, fit4_esum) |>
  relocate(Model) |> gt() |> fmt_number(decimals = 3) |>
  tab_options(table.font.size = 24)
```

Model	MAPE	Max_E	RMSPE	Val_R2
fit1	4.300	21.140	5.621	0.078
fit2	4.430	20.530	5.792	0.042
fit3	4.325	21.107	5.650	0.068
fit4	4.465	21.162	5.844	0.036

Our Four Models

- `fit1`: `exerany` and `health` main effects; `fit2`: add interaction
- `fit3`: add `fruit_c` to `fit1`; `fit4`: add `fruit_c` to `fit2`

What's coming up next?

Basics of logistic regression fitting and evaluation

- What if we have a binary (yes/no or 1/0) outcome?
- Predict "whether or not BMI < 30", rather than BMI?
  - A linear probability model as a first idea
  - Using `glm()` rather than `lm()` to get a logistic model
  - Coefficients as log(odds ratios)
  - Changes in how we measure the model's performance
  - Changes in the assumptions we make

What to do before Class 3?

1. Lab 1 due to Canvas at noon Wednesday 2026-01-21.
2. Get started reading *How To Be a Modern Scientist* so you finish the little book by the end of January.
3. Be sure you have everything working properly in RStudio. Visit TA office hours (starting tomorrow, no hours this Monday) for help, or email **431-help** at case dot edu.
4. Review the Project A instructions.
5. The Calendar is the most important thing to look at.

Appendix: Creating Today's Data

Creating Today's Data Set

```
url1 <- "https://raw.githubusercontent.com/THOMASELOVE/432-data/master/data/smart_

smart_ohio <- read_csv(url1)

smt <- smart_ohio |>
    filter(hx_diabetes == 0, mmsa == "Cleveland-Elyria",
           complete.cases(bmi)) |>
    select(bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, genhealth, race_eth,
           hx_diabetes, mmsa, SEQNO) |>
    mutate(across(where(is.character), as_factor)) |>
    mutate(ID = as.character(SEQNO - 2017000000)) |>
    relocate(ID)
```


Codebook for useful `smt` variables (1)

- 894 subjects in Cleveland-Elyria with `bmi` and no history of diabetes

Variable	Description
`bmi`	(outcome) Body-Mass index in kg/m^2.
`inc_imp`	income (imputed from grouped values) in $
`fruit_day`	average fruit servings consumed per day
`drinks_wk`	average weekly alcoholic drinks consumed
`female`	sex: 1 = female, 0 = male

Codebook for useful `smt` variables (2)

- 894 subjects in Cleveland-Elyria without diabetes

Variable	Description
`exerany`	any exercise in past month: 1 = yes, 0 = no
`genhealth`	self-reported overall health (5 levels)
`race_eth`	race and Hispanic/Latinx ethnicity (5 levels)

- plus `ID`, `SEQNO`, `hx_diabetes` (all 0), `MMSA`
- See Course Notes Chapter 6 on BRFSS SMART data

Basic Data Summaries

Available approaches include:

- `data_codebook()` from `datawizard` in `easystats`
- `Hmisc` package's `describe()`, or
- `summary()`

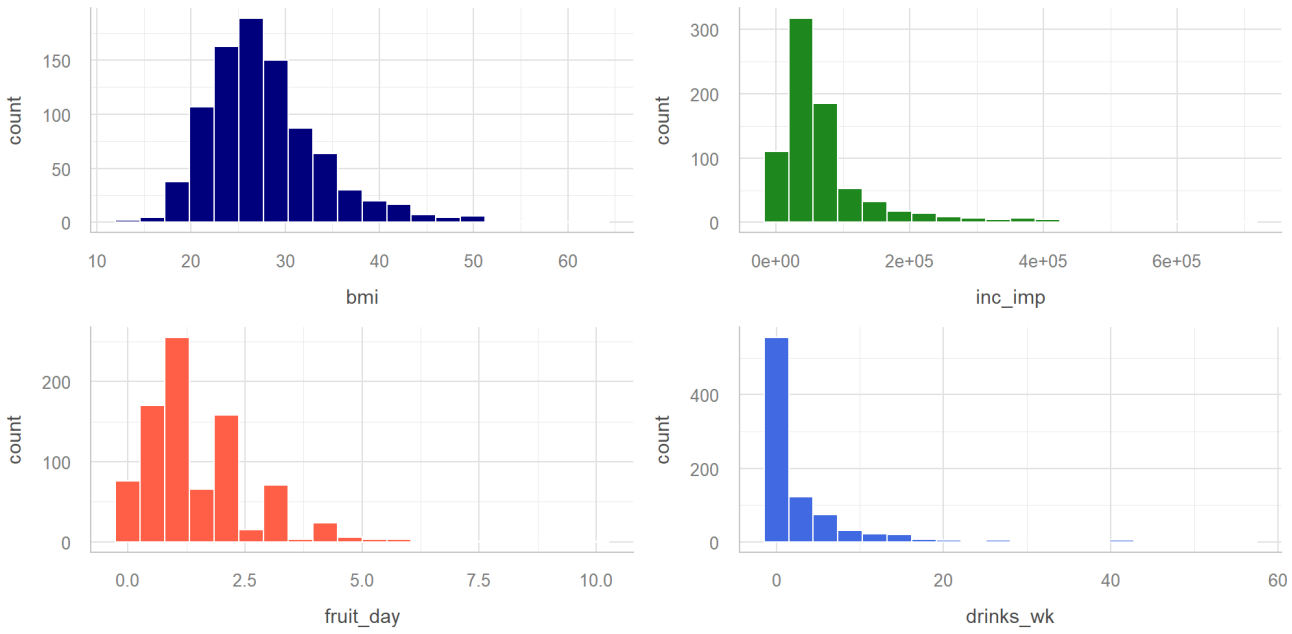
all of which can work nicely in an HTML presentation, but none of them fit well on a slide.

Histogram of each quantity

> **Note**
>
> I used `#| warning: false` in this code chunk to avoid warnings about missing values, like this one for `inc_imp`:
>
> ```
> Warning: Removed 120 rows containing non-finite values
> ```

```
p1 <- ggplot(smt, aes(x = bmi)) +
    geom_histogram(fill = "navy", col = "white", bins = 20)
p2 <- ggplot(smt, aes(x = inc_imp)) +
    geom_histogram(fill = "forestgreen", col = "white", bins = 20)
p3 <- ggplot(smt, aes(x = fruit_day)) +
    geom_histogram(fill = "tomato", col = "white", bins = 20)
p4 <- ggplot(smt, aes(x = drinks_wk)) +
    geom_histogram(fill = "royalblue", col = "white", bins = 20)

(p1 + p2) / (p3 + p4)
```


Histogram of each quantity



Binary variables in raw `smt`

```
1 smt |> tabyl(female, exerany) |> adorn_title()
```

```
       exerany
female       0   1 NA_
     0      95 268  20
     1     128 361  22
```

- `female` is based on biological sex (1 = female, 0 = male)
- `exerany` comes from a response to "During the past month, other than your regular job, did you participate in any physical activities or exercises such as running, calisthenics, golf, gardening, or walking for exercise?" (1 = yes, 0 = no, don't know and refused = missing)
- Any signs of trouble here?

Multicategorical `genhealth` in raw `smt`

```
1 smt |> tabyl(genhealth)
```

```
  genhealth   n     percent valid_percent
1_Excellent 148 0.165548098    0.16573348
     3_Good 274 0.306487696    0.30683091
 2_VeryGood 324 0.362416107    0.36282195
     4_Fair 112 0.125279642    0.12541993
     5_Poor  35 0.039149888    0.03919373
       <NA>   1 0.001118568            NA
```

- The variable is based on "Would you say that in general your health is …" using the five specified categories (Excellent -> Poor), numbered for convenience after data collection.
- Don't know / not sure / refused treated as missing.
- How might we manage this variable?

Changing the levels for genhealth

```
smt <- smt |>
    mutate(health =
               fct_recode(genhealth,
                          E = "1_Excellent",
                          VG = "2_VeryGood",
                          G = "3_Good",
                          F = "4_Fair",
                          P = "5_Poor"),
           health = fct_relevel(health, "E", "VG", "G", "F", "P"))
```

Might want to run a sanity check here, just to be sure…

Checking health vs. genhealth

```
smt |> tabyl(genhealth, health) |> adorn_title()
```

```
            health
  genhealth      E  VG   G   F  P NA_
1_Excellent    148   0   0   0  0   0
     3_Good      0   0 274   0  0   0
 2_VeryGood      0 324   0   0  0   0
     4_Fair      0   0   0 112  0   0
     5_Poor      0   0   0   0 35   0
       <NA>      0   0   0   0  0   1
```

- OK. We've adjusted the order to something more sensible, retained the missing value, and we have much shorter labels.

Multicategorical `race_eth` in raw `smt`

```
1 smt |> count(race_eth)
```

```
# A tibble: 6 × 2
  race_eth                    n
  <fct>                   <int>
1 White non-Hispanic        646
2 Other race non-Hispanic    22
3 Black non-Hispanic        167
4 Multiracial non-Hispanic   19
5 Hispanic                   27
6 <NA>                       13
```

"Don't know", "Not sure", and "Refused" were treated as missing.

> - What is this variable actually about?
> - What is the most common thing people do here?

What is the question you are asking?

Collapsing `race_eth` levels *might* be rational for *some* questions.

- We have lots of data from two categories, but only two.
- Systemic racism affects people of color in different ways across these categories, but also *within* them.

Is combining race and Hispanic/Latinx ethnicity helpful?

It's hard to see the justice in collecting this information and not using it in as granular a form as possible, though this leaves some small sample sizes. There is no magic number for "too small a sample size."

- Most people identified themselves in one category.
- These data are not ordered, and (I'd argue) ordering them isn't helpful.
- Regression models are easier to interpret, though, if the "baseline" category is a common one.

Resorting the factor for `race_eth`

Let's sort all five levels, from most observations to least…

```
smt <- smt |>
    mutate(race_eth = fct_infreq(race_eth))

smt |> tabyl(race_eth)
```

```
              race_eth   n    percent valid_percent
      White non-Hispanic 646 0.72259508    0.73325766
      Black non-Hispanic 167 0.18680089    0.18955732
               Hispanic  27 0.03020134    0.03064699
  Other race non-Hispanic  22 0.02460850    0.02497162
Multiracial non-Hispanic  19 0.02125280    0.02156640
                  <NA>  13 0.01454139            NA
```

- Not a perfect solution, certainly, but we'll try it out.

“Cleaned” Data and Missing Values

```
smt <- smt |>
    select(ID, bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, health, race_eth)

miss_var_summary(smt)
```

```
# A tibble: 9 × 3
  variable  n_miss pct_miss
  <chr>      <int>    <num>
1 inc_imp      120   13.4
2 exerany       42    4.70
3 fruit_day     41    4.59
4 drinks_wk     39    4.36
5 race_eth      13    1.45
6 health         1    0.112
7 ID             0    0
8 bmi            0    0
9 female         0    0
```


Single Imputation with `mice`

```
smt_im <- mice(smt, m = 1, seed = 20260115, print = FALSE) |>
  complete() |>
  tibble()
```

> **Note**
>
> You may get a logged event for the ID variable expressed as a character, and that can be ignored.

```
prop_miss_case(smt_im)
```

```
[1] 0
```

```
dim(smt_im)
```

```
[1] 894   9
```

Saving the tidied data

Let's save both the unimputed and the imputed tidy data as R data sets.

```
write_rds(smt, "c02/data/smt.Rds")

write_rds(smt_im, "c02/data/smt_im.Rds")
```

To reload these files, we'll use `read_rds()`.

- The main advantage here is that we've saved the whole R object, including all characteristics that we've added since the original download.

Session Information

```
xfun::session_info()
```

```
R version 4.5.2 (2025-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26200)

Locale:
  LC_COLLATE=English_United States.utf8
  LC_CTYPE=English_United States.utf8
  LC_MONETARY=English_United States.utf8
  LC_NUMERIC=C
  LC_TIME=English_United States.utf8

Package version:
  abind_1.4-8             askpass_1.2.1           backports_1.5.0
  base64enc_0.1.3         bayestestR_0.17.0       bigD_0.3.1
  bit_4.6.0               bit64_4.6.0-1           bitops_1.0.9
  blob_1.2.4              boot_1.3-32             broom_1.0.11
  bslib_0.9.0             cachem_1.1.0            callr_3.7.6
  car_3.1-3               carData_3.0-5           cellranger_1.1.0
  cli_3.6.5               clipr_0.8.0             coda_0.19-4.1
  codetools_0.2-20        colorspace_2.1.2        commonmark_2.0.0
```