432 Class 04

https://thomaselove.github.io/432-2024/

2024-01-25

Today's Agenda

- Fitting two-factor ANOVA/ANCOVA models with `lm`
  - Incorporating an interaction between factors
  - Incorporating a quantitative covariate
  - Using a quadratic polynomial fit
- Regression Diagnostics via Residual Plots
- Validating / evaluating results with `yardstick`

Appendix

How the `c4im` data were created from `smart_ohio.csv`

Today's R Setup

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

library(janitor)
library(broom)
library(gt)
library(car)
library(mosaic)
library(patchwork)
library(naniar)
library(simputation)    ## single imputation of missing data
library(rsample)        ## data splitting
library(yardstick)      ## evaluating fits
library(rms)            ## regression tools (Frank Harrell)
library(tidyverse)

theme_set(theme_bw())
```

Section 1

The c4im data

The c4im 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 the past month: $1 =$ yes, $0 =$ no
`genhealth`	self-reported overall health (5 levels)
`fruit_day`	average fruit servings consumed per day

Data Load

```r
c4im <- read_rds("c04/data/c4im.Rds")
c4im |> n_miss()
```

```
[1] 0
```

```r
identical(nrow(c4im), n_distinct(c4im$ID))
```

```
[1] TRUE
```

Our covariate, `fruit_day`

Our main interest is in the factors `exerany` and `genhealth`.

Later, we'll adjust for the (quantitative) covariate `fruit_day`. Here, we'll be including the covariate to help account for some nuisance variation, rather than being deeply interested in the impact of `fruit_day` on `bmi`. A common approach, then, is centering the predictor prior to including it.

response	min	Q1	median	Q3	max	mean	sd
fruit_day	0.000	0.710	1.135	2.000	10.000	1.438	1.100
fruit_c	−1.438	−0.728	−0.303	0.562	8.562	0.000	1.100

Splitting the Sample

```r
set.seed(432)    ## for future replication
c4im_split <- initial_split(c4im, prop = 3/4)
train_c4im <- training(c4im_split)
test_c4im <- testing(c4im_split)
c(nrow(c4im), nrow(train_c4im), nrow(test_c4im))
```

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

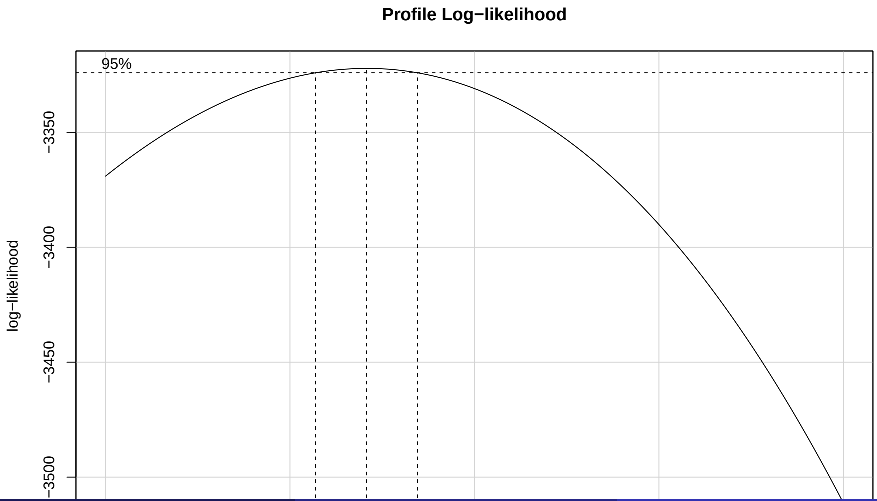
Models We'll Build Today

1. Predict `bmi` using `exer_any` and `genhealth` (both categorical)
    - without then with an interaction between the predictors
2. Add in a (centered) quantitative covariate, `fruit_c`.
3. Incorporate `fruit_c` using a quadratic polynomial.

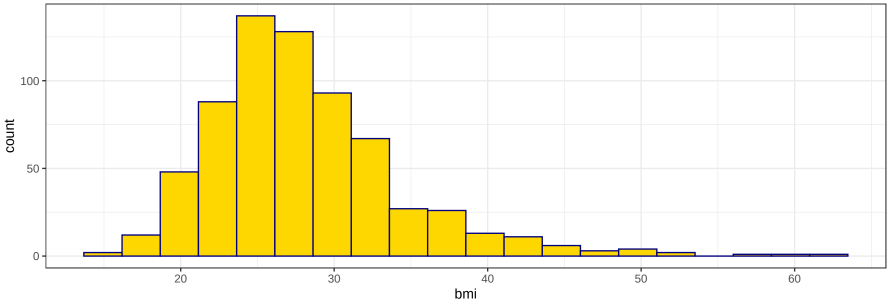
We'll fit all of these models with `lm`, and assess them in terms of in-sample (training) fit and out-of-sample (testing) performance.

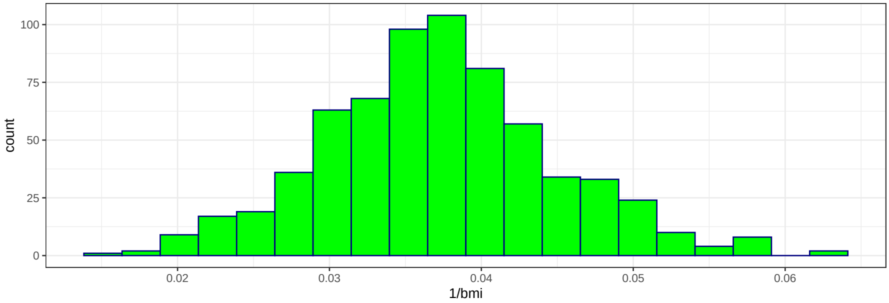
Consider transforming bmi?

```r
m0 <- lm(bmi ~ exerany + health, data = train_c4im)
boxCox(m0)
```



Should we transform `bmi`?





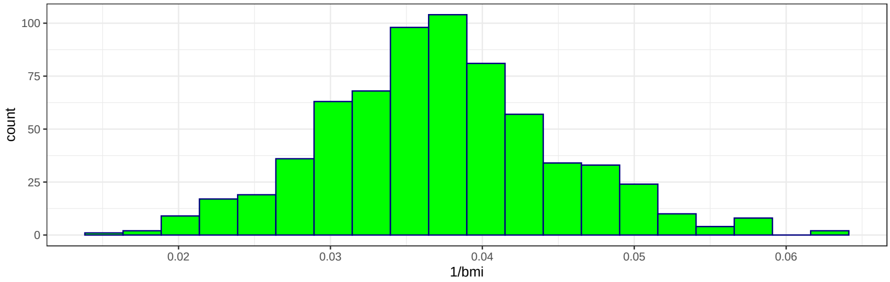
Re-scaling the transformation

```
bind_rows( favstats(~ 1/bmi, data = train_c4im),
           favstats(~ 1000/bmi, data = train_c4im)) |>
  mutate(outcome = c("1/bmi", "1000/bmi")) |>
  relocate(outcome) |>
  gt() |> fmt_number(columns = min:sd, decimals = 3) |>
  tab_options(table.font.size = 20)
```

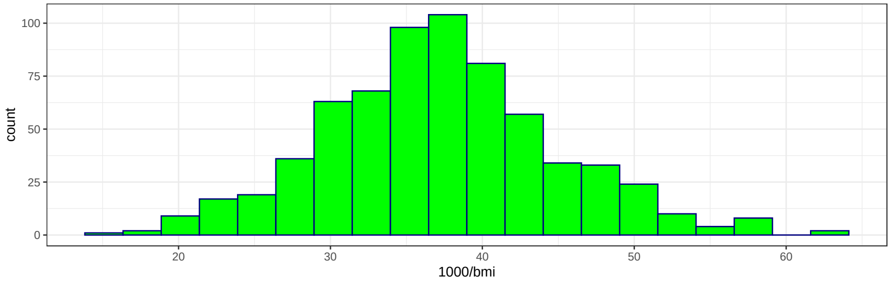
outcome	min	Q1	median	Q3	max	mean	sd
1/bmi	0.016	0.032	0.037	0.042	0.064	0.037	0.008
1000/bmi	15.873	32.248	36.839	41.806	63.654	37.240	7.606

Shape doesn't change

1/BMI



1000/BMI



Means by exerany and health

```
summaries_1 <- train_c4im |>
    group_by(exerany, health) |>
    summarise(n = n(), mean = mean(1000/bmi), stdev = sd(1000/
summaries_1
```

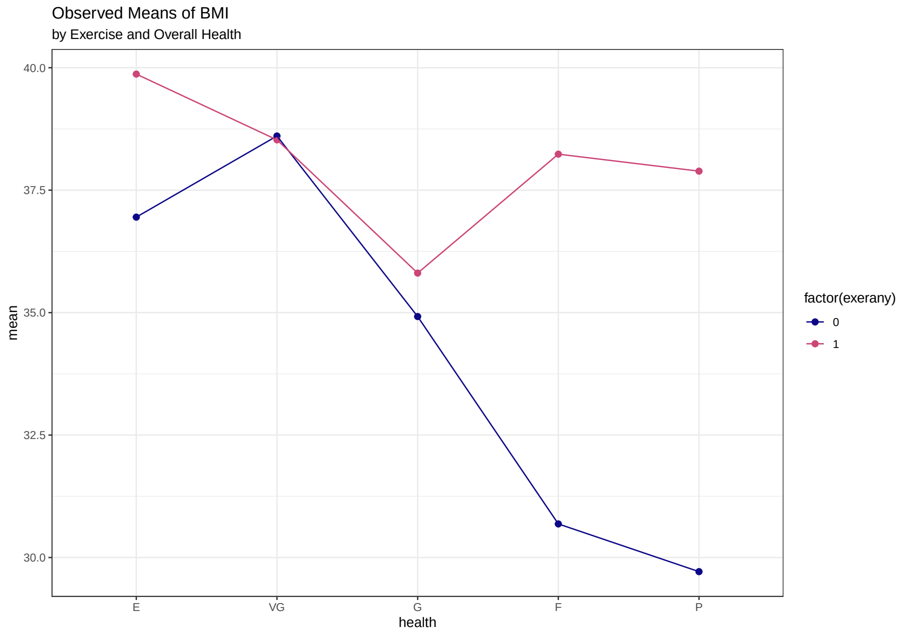
```
# A tibble: 10 x 5
# Groups:   exerany [2]
   exerany health     n  mean stdev
     <int> <fct>  <int> <dbl> <dbl>
 1       0 E         18  36.9  4.70
 2       0 VG        54  38.6  7.50
 3       0 G         58  34.9  8.51
 4       0 F         31  30.7  8.49
 5       0 P          8  29.7  7.24
 6       1 E         92  39.9  6.50
 7       1 VG       191  38.5  6.87
 8       1 G        152  35.8  7.20
```

Code for Interaction Plot

```
ggplot(summaries_1, aes(x = health, y = mean,
                        col = factor(exerany))) +
  geom_point(size = 2) +
  geom_line(aes(group = factor(exerany))) +
  scale_color_viridis_d(option = "C", end = 0.5) +
  labs(title = "Observed Means of 1000/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 on the next slide that results from this code?

Resulting Interaction Plot



Section 2

Fitting a Two-Way ANOVA model for 1000/BMI

Model `m1` without interaction

```r
m1 <- lm(1000/bmi ~ exerany + health, data = train_c4im)
```

- How well does this model fit the training data?

```r
glance(m1) |>
    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 = 20)
```

r.squared	adj.r.squared	sigma	nobs	df	df.residual	AIC	B
0.064	0.057	7.385	670	5	664	4,588.7	4,62

Tidied ANOVA for `m1`

```r
tidy(anova(m1)) |> 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	859.17	859.17	15.75	0.0001
health	4	1,624.63	406.16	7.45	0.0000
Residuals	664	36,217.09	54.54	NA	NA

Tidied summary of `m1` coefficients

```
tidy(m1, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals =
  tab_options(table.font.size = 20)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	37.593	0.897	41.909	0.000	36.116	39.071
exerany	2.150	0.664	3.237	0.001	1.056	3.245
healthVG	−0.725	0.848	−0.855	0.393	−2.123	0.672
healthG	−3.588	0.872	−4.112	0.000	−5.025	−2.151
healthF	−3.601	1.095	−3.287	0.001	−5.405	−1.796
healthP	−3.784	1.640	−2.308	0.021	−6.485	−1.083

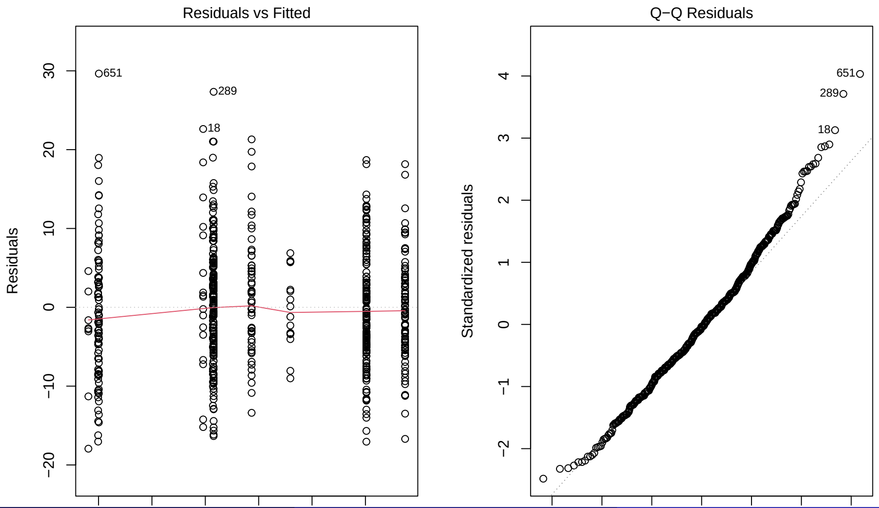
Interpreting `m1`

Name	`exerany`	`health`	predicted `1000/bmi`
Harry	0	Excellent	37.59
Sally	1	Excellent	37.59 + 2.15 = 39.74
Billy	0	Fair	37.59 - 3.60 = 33.99
Meg	1	Fair	37.59 + 2.15 - 3.60 = 36.14

- Effect of `exerany`?
- Effect of `health` = Fair instead of Excellent?

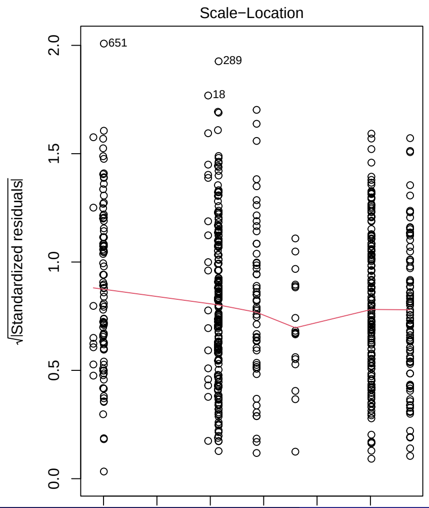
`m1` Residual Plots ($n = 670$)

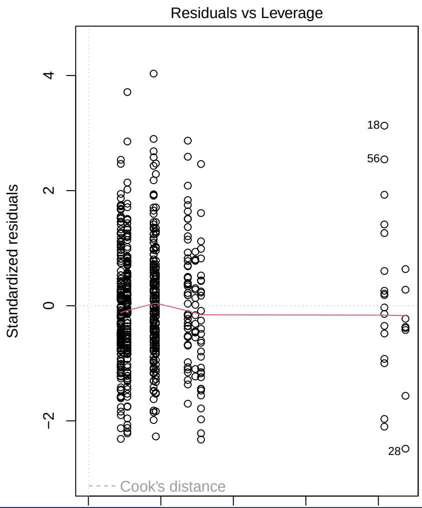
```r
par(mfrow = c(1,2)); plot(m1, which = c(1,2))
```



m1 Residual Plots ($n = 670$)

```
par(mfrow = c(1,2)); plot(m1, which = c(3,5))
```





Section 3

Fitting ANOVA model `m1int` including interaction

Adding the interaction term to m1

```
m1int <- lm(1000/bmi ~ exerany * health, data = train_c4im)
```

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

```
bind_rows(glance(m1), glance(m1int)) |>
  mutate(mod = c("m1", "m1int")) |>
  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 = 20)
```

mod	r.sq	adj.r.sq	sigma	nobs	df	df.res	AIC	BIC
m1	0.064	0.057	7.385	670	5	664	4,588.7	4,620.2
m1int	0.091	0.079	7.301	670	9	660	4,577.2	4,626.8

ANOVA for the `m1int` model

```r
tidy(anova(m1int)) |> 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	859.17	859.17	16.12	0.0001
health	4	1,624.63	406.16	7.62	0.0000
exerany:health	4	1,036.15	259.04	4.86	0.0007
Residuals	660	35,180.94	53.30	NA	NA

ANOVA test comparing `m1` to `m1int`

```r
anova(m1, m1int)
```

```
Analysis of Variance Table

Model 1: 1000/bmi ~ exerany + health
Model 2: 1000/bmi ~ exerany * health
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    664 36217
2    660 35181  4    1036.2 4.8596 0.0007223 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 
```

m1int coefficients

```
tidy(m1int, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals =
  tab_options(table.font.size = 20)
```

term	estimate	std.error	statistic	p.value	conf.low	co
(Intercept)	36.950	1.721	21.472	0.000	34.115	
exerany	2.920	1.882	1.552	0.121	−0.179	
healthVG	1.656	1.987	0.834	0.405	−1.617	
healthG	−2.030	1.970	−1.030	0.303	−5.274	
healthF	−6.264	2.164	−2.895	0.004	−9.827	
healthP	−7.238	3.102	−2.333	0.020	−12.348	
exerany:healthVG	−2.999	2.192	−1.368	0.172	−6.610	
exerany:healthG	−2.033	2.193	−0.927	0.354	−5.646	
exerany:healthF	4.629	2.520	1.837	0.067	0.479	
exerany:healthP	5.256	3.652	1.439	0.151	−0.760	

Interpreting the `m1int` model

Name	`exerany`	`health`	predicted `1000/bmi`
Harry	0	Excellent	36.95
Sally	1	Excellent	36.95 + 2.92 = 39.87
Billy	0	Fair	36.95 - 6.26 = 30.69
Meg	1	Fair	36.95 + 2.92 - 6.26 + 4.63 = 38.24

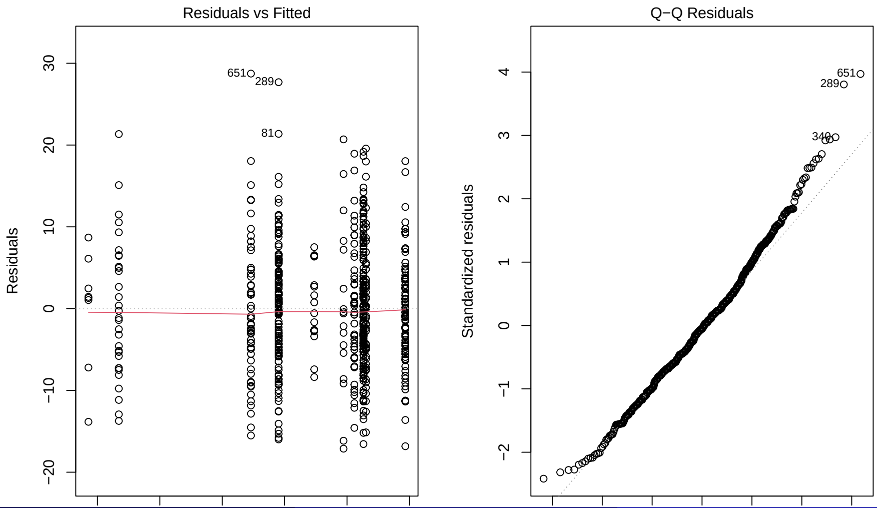
- How do we interpret effect sizes here? **It depends**.

Interpreting the `m1int` model

- Effect of `exerany` on predicted `1000/bmi`?
  - If `health` = Excellent, effect is +2.92
  - If `health` = Fair, effect is (2.92 + 4.63) = +7.55
- Effect of `health` = Fair instead of Excellent?
  - If `exerany` = 0 (no), effect is -6.26
  - If `exerany` = 1 (yes), effect is (-6.26 + 4.63) = -1.63

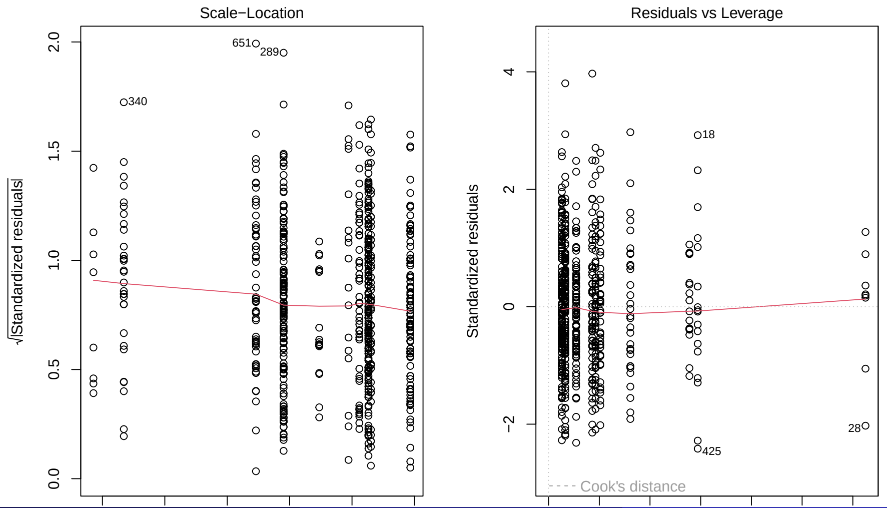
Residuals from `m1int`? ($n = 670$)

```r
par(mfrow = c(1,2)); plot(m1int, which = c(1,2))
```



Residuals from m1int? ($n = 670$)

```
par(mfrow = c(1,2)); plot(m1int, which = c(3,5))
```



Section 4

Incorporating a Covariate into our two-way ANOVA models

Add fruit_c to m1

```
m2 <- lm(1000/bmi ~ fruit_c + exerany + health, data = train_c
```

- How well does this model fit the training data?

```
bind_rows(glance(m1), glance(m2)) |>
  mutate(mod = c("m1", "m2")) |>
  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 = 20)
```

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m1	0.064	0.057	7.385	5	664	4,588.7	4,620.2
m2	0.075	0.066	7.349	6	663	4,583.1	4,619.2

ANOVA for the m2 model

```
tidy(anova(m2)) |> 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
fruit_c	1	692.07	692.07	12.81	0.0004
exerany	1	697.12	697.12	12.91	0.0004
health	4	1, 499.96	374.99	6.94	0.0000
Residuals	663	35, 811.73	54.01	NA	NA

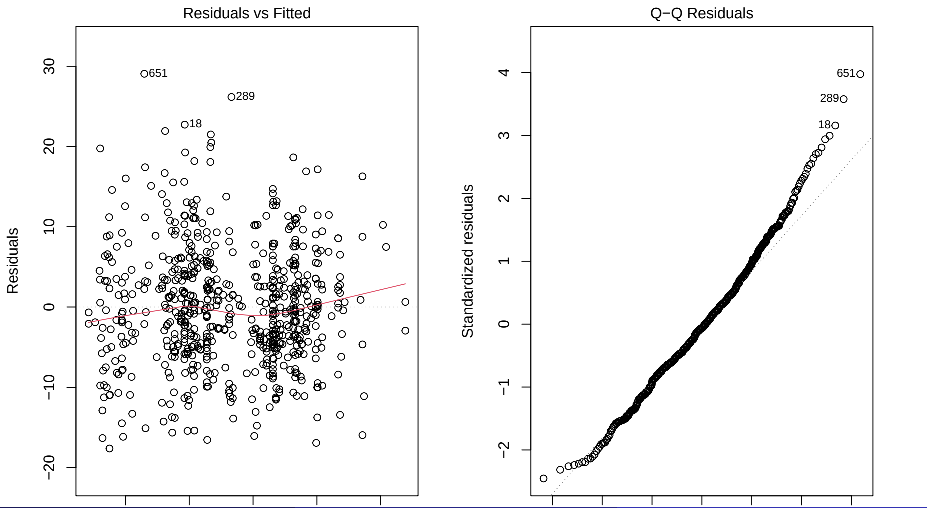
m2 coefficients

```
tidy(m2, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals =
  tab_options(table.font.size = 20)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	37.613	0.893	42.134	0.000	36.142	39.083
fruit_c	0.702	0.256	2.739	0.006	0.280	1.124
exerany	1.957	0.665	2.943	0.003	0.862	3.052
healthVG	−0.642	0.845	−0.760	0.447	−2.034	0.749
healthG	−3.413	0.871	−3.921	0.000	−4.847	−1.979
healthF	−3.390	1.093	−3.102	0.002	−5.190	−1.590
healthP	−3.803	1.632	−2.331	0.020	−6.491	−1.115

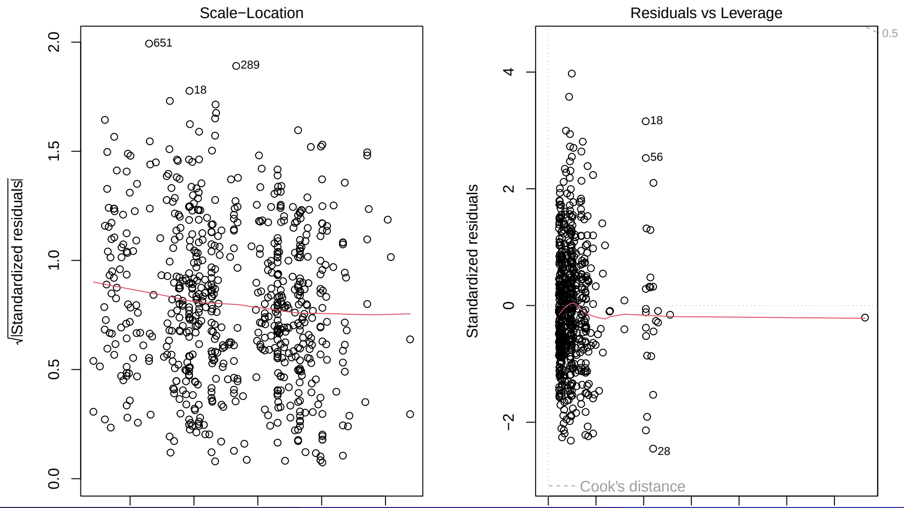
m2 Residuals

```r
par(mfrow = c(1,2)); plot(m2, which = c(1,2))
```



m2 Residuals

```
par(mfrow = c(1,2)); plot(m2, which = c(3,5))
```



Include the interaction term?

```
m2int <- lm(1000/bmi ~ fruit_c + exerany * health,
            data = train_c4im)
```

ANOVA for the m2int model

```
tidy(anova(m2int)) |> 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
fruit_c	1	692.07	692.07	13.14	0.0003
exerany	1	697.12	697.12	13.24	0.0003
health	4	1,499.96	374.99	7.12	0.0000
exerany:health	4	1,110.99	277.75	5.27	0.0003
Residuals	659	34,700.74	52.66	NA	NA

m2int coefficients

```
tidy(m2int, conf.int = TRUE, conf.level = 0.90) |>
  gt() |> fmt_number(columns = estimate:conf.high, decimals =
  tab_options(table.font.size = 18)
```

term	estimate	std.error	statistic	p.value	conf.low	co
(Intercept)	37.064	1.711	21.665	0.000	34.246	
fruit_c	0.766	0.254	3.020	0.003	0.348	
exerany	2.598	1.873	1.387	0.166	−0.488	
healthVG	1.711	1.975	0.866	0.387	−1.543	
healthG	−1.920	1.958	−0.981	0.327	−5.146	
healthF	−6.205	2.150	−2.885	0.004	−9.747	−
healthP	−7.734	3.088	−2.505	0.012	−12.821	−
exerany:healthVG	−2.961	2.179	−1.359	0.175	−6.550	
exerany:healthG	−1.938	2.180	−0.889	0.374	−5.529	
exerany:healthF	4.867	2.505	1.943	0.052	0.741	
exerany:healthP	5.930	3.637	1.631	0.103	−0.060	

ANOVA: Compare m2 & m2int

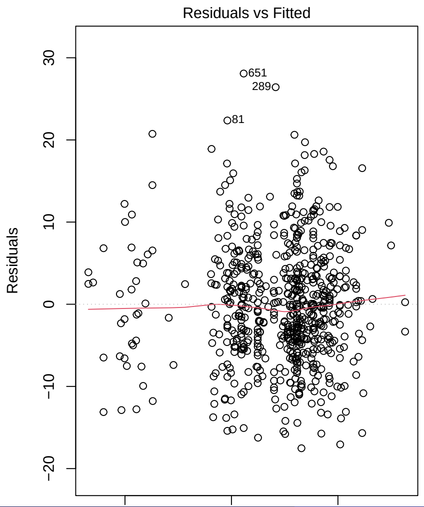
```r
anova(m2, m2int)
```

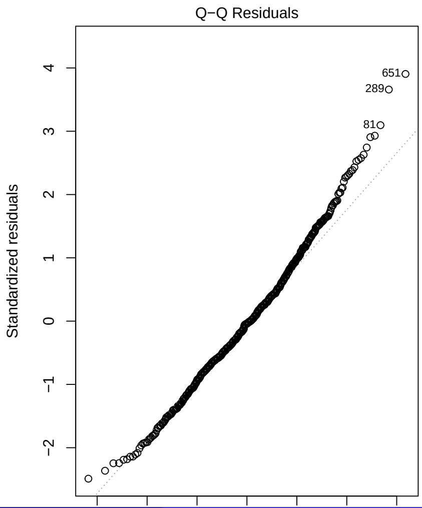
```
Analysis of Variance Table

Model 1: 1000/bmi ~ fruit_c + exerany + health
Model 2: 1000/bmi ~ fruit_c + exerany * health
  Res.Df   RSS Df Sum of Sq      F   Pr(>F)
1    663 35812
2    659 34701  4      1111 5.2747 0.000347 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

m2int Residuals

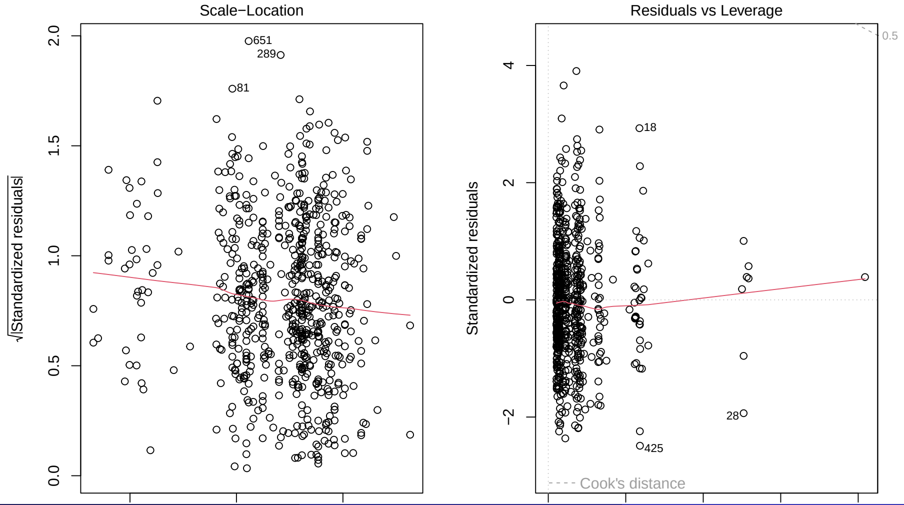
```r
par(mfrow = c(1,2)); plot(m2int, which = c(1,2))
```





m2int Residuals

```
par(mfrow = c(1,2)); plot(m2int, which = c(3,5))
```



Section 5

Comparing Our Models

Which of the four models fits best?

In the **training** sample, we have...

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m1	0.064	0.057	7.385	5	664	$4,588.7$	$4,620.2$
m2	0.075	0.066	7.349	6	663	$4,583.1$	$4,619.2$
m1int	0.091	0.079	7.301	9	660	$4,577.2$	$4,626.8$
m2int	0.103	0.090	7.256	10	659	$4,570.0$	$4,624.1$

- Adjusted R^2, σ and AIC all improve as we move down from `m1` towards `m2_int`. BIC likes `m2`.
- BUT the training sample cannot judge between models accurately. Our models have already *seen* that data.

What does augment() give us?

```
m1_test_aug <- augment(m1, newdata = test_c4im) |>
  mutate(out = 1000/bmi)
m1_test_aug |> select(ID, bmi, out, .fitted, .resid, health, e
  slice(198:202) |> gt() |>
  fmt_number(columns = bmi:.resid, decimals = 2) |>
  tab_options(table.font.size = 20)
```

ID	bmi	out	.fitted	.resid	health	exerany
1016	28.44	35.16	33.81	1.35	P	0
1018	26.68	37.48	39.74	−2.26	E	1
1019	25.74	38.85	36.14	2.71	F	1
1020	20.57	48.61	39.02	9.60	VG	1
1024	24.52	40.78	34.01	6.78	G	0

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

What to do?

Our models predict `1000/bmi`, but we want to assess predictions of `bmi`. How do we convert predicted `1000/bmi` to predicted `bmi`?

Note that $1000/(1000/\texttt{bmi}) = \texttt{bmi}$, so we need

- $1000/$`.fitted` for our predicted `bmi`, and
- observed `bmi` - predicted `bmi` for our residuals

Adjusting augment() appropriately

```r
m1_test_aug <- augment(m1, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)
m1_test_aug |>
  select(ID, bmi, bmi_fit, bmi_res, health, exerany, .fitted,
  slice(198:202) |> gt() |>
  fmt_number(columns = bmi:bmi_res, decimals = 2) |>
  fmt_number(columns = .fitted:.resid, decimals = 2) |>
  tab_options(table.font.size = 20)
```

ID	bmi	bmi_fit	bmi_res	health	exerany	.fitted	.resid
1016	28.44	29.58	−1.14	P	0	33.81	1.35
1018	26.68	25.16	1.52	E	1	39.74	−2.26
1019	25.74	27.67	−1.93	F	1	36.14	2.71
1020	20.57	25.63	−5.06	VG	1	39.02	9.60
1024	24.52	29.41	−4.89	G	0	34.01	6.78

Augment all four models so far...

```
m1_test_aug <- augment(m1, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)

m1int_test_aug <- augment(m1int, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)

m2_test_aug <- augment(m2, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)

m2int_test_aug <- augment(m2int, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)
```

Section 6

Using the yardstick package

The `yardstick` package

For each subject in the testing set, we will need:

- estimate = model's prediction of that subject's `bmi`
- truth = the `bmi` value observed for that subject

Calculate a summary of the predictions across the n test subjects

Summaries from `yardstick`

- R^2 = square of the correlation between truth and estimate
- `mae` = mean absolute error …

$$mae = \frac{1}{n} \sum |truth - estimate|$$

- `rmse` = root mean squared error …

$$rmse = \sqrt{\frac{1}{n} \sum (truth - estimate)^2}$$

Testing Results (Validated R^2)

We can use the `yardstick` package and its `rsq()` function.

```r
testing_r2 <- bind_rows(
    rsq(m1_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m1int_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m2_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m2int_test_aug, truth = bmi, estimate = bmi_fit)) |>
    mutate(model = c("m1", "m1int", "m2", "m2int"))
testing_r2 |>
  gt() |> fmt_number(.estimate, decimals = 3) |>
  tab_options(table.font.size = 20)
```

.metric	.estimator	.estimate	model
rsq	standard	0.078	m1
rsq	standard	0.036	m1int
rsq	standard	0.069	m2
rsq	standard	0.032	m2int

Mean Absolute Error?

Consider the mean absolute prediction error …

```
testing_mae <- bind_rows(
    mae(m1_test_aug, truth = bmi, estimate = bmi_fit),
    mae(m1int_test_aug, truth = bmi, estimate = bmi_fit),
    mae(m2_test_aug, truth = bmi, estimate = bmi_fit),
    mae(m2int_test_aug, truth = bmi, estimate = bmi_fit)) |>
    mutate(model = c("m1", "m1int", "m2", "m2int"))
testing_mae |>
  gt() |> fmt_number(.estimate, decimals = 3) |>
  tab_options(table.font.size = 20)
```

.metric	.estimator	.estimate	model
mae	standard	4.296	m1
mae	standard	4.463	m1int
mae	standard	4.309	m2
mae	standard	4.485	m2int

Root Mean Squared Error?

How about the square root of the mean squared prediction error, or RMSE?

```
testing_rmse <- bind_rows(
   rmse(m1_test_aug, truth = bmi, estimate = bmi_fit),
   rmse(m1int_test_aug, truth = bmi, estimate = bmi_fit),
   rmse(m2_test_aug, truth = bmi, estimate = bmi_fit),
   rmse(m2int_test_aug, truth = bmi, estimate = bmi_fit)) |>
   mutate(model = c("m1", "m1int", "m2", "m2int"))
testing_rmse |>
  gt() |> fmt_number(.estimate, decimals = 3) |>
  tab_options(table.font.size = 20)
```

.metric	.estimator	.estimate	model
rmse	standard	5.628	m1
rmse	standard	5.842	m1int
rmse	standard	5.650	m2

Other `yardstick` summaries (1)

- `rsq_trad()` $=$ defines R^2 using sums of squares.
  - The `rsq()` measure we showed a few slides ago is a squared correlation coefficient guaranteed to be in (0, 1).
- `mape()` $=$ mean absolute percentage error
- `mpe()` $=$ mean percentage error

Other yardstick summaries (2)

- huber_loss() = Huber loss (often used in robust regression), which is less sensitive to outliers than rmse().
- ccc() = concordance correlation coefficient, which attempts to measure both consistency/correlation (like rsq()) and accuracy (like rmse()).

See the yardstick home page for more details.

Section 7

Incorporating Non-Linearity into our models

Polynomial Regression

A polynomial in the variable `x` of degree D is a linear combination of the powers of `x` up to D.

For example:

- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Quartic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$
- Quintic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

Fitting such a model creates a **polynomial regression**.

Adding a polynomial in fruit_c

Can we predict 1000/bmi with a polynomial in fruit_c?

```
lm(1000/bmi ~ fruit_c, data = train_c4im)
lm(1000/bmi ~ poly(fruit_c, 2), data = train_c4im)
lm(1000/bmi ~ poly(fruit_c, 3), data = train_c4im)
```

Plotting the Polynomials

```
p1 <- ggplot(train_c4im, aes(x = fruit_c, y = 1000/bmi)) +
    geom_point(alpha = 0.3) +
    geom_smooth(formula = y ~ x, method = "lm",
                col = "red", se = FALSE) +
    labs(title = "Linear Fit")

p2 <- ggplot(train_c4im, aes(x = fruit_c, y = 1000/bmi)) +
    geom_point(alpha = 0.3) +
    geom_smooth(formula = y ~ poly(x, 2), method = "lm",
                col = "blue", se = FALSE) +
    labs(title = "2nd order Polynomial")

p3 <- ggplot(train_c4im, aes(x = fruit_c, y = 1000/bmi)) +
    geom_point(alpha = 0.3) +
    geom_smooth(formula = y ~ poly(x, 3), method = "lm",
                col = "purple", se = FALSE) +
    labs(title = "3rd order Polynomial")
```

Raw vs. Orthogonal Polynomials

Predict 1000/bmi using fruit_c with a "raw polynomial of degree 2."

```r
(temp1 <- lm(1000/bmi ~ fruit_c + I(fruit_c^2),
             data = train_c4im))
```

```

Call:
lm(formula = 1000/bmi ~ fruit_c + I(fruit_c^2), data = train_c

Coefficients:
 (Intercept)       fruit_c  I(fruit_c^2)
     37.3653        1.1640       -0.1201
```

Predicted 1000/bmi for fruit_c = 0.5 is

```
1000/bmi = 37.3653 + 1.1640 (fruit_c) - 0.1201 (fruit_c^2)
         = 37.3653 + 1.1640 (0.5) - 0.1201 (0.25)
         = 37.91727
```

Does the raw polynomial match our expectations?

```
temp1 <- lm(1000/bmi ~ fruit_c + I(fruit_c^2),
            data = train_c4im)

augment(temp1, newdata = tibble(fruit_c = 0.5)) |>
  gt() |> tab_options(table.font.size = 20)
```

fruit_c	.fitted
0.5	37.91727

This matches our "by hand" calculation.

- But it turns out most regression models use *orthogonal* rather than raw polynomials...

Fitting an Orthogonal Polynomial

Predict `1000/bmi` using `fruit_c` with an *orthogonal* polynomial of degree 2.

```
(temp2 <- lm(1000/bmi ~ poly(fruit_c,2), data = train_c4im))
```

```
Call:
lm(formula = 1000/bmi ~ poly(fruit_c, 2), data = train_c4im)

Coefficients:
      (Intercept)  poly(fruit_c, 2)1  poly(fruit_c, 2)2  
            37.24              26.31              -9.15  
```

This looks very different from our previous version of the model. What happens when we make a prediction, though?

Prediction in the Orthogonal Polynomial Model

Remember that in our raw polynomial model, our "by hand" and "using R" calculations each predicted `1000/bmi` for a subject with `fruit_c` = 0.5 to be 37.91727.

What happens with the orthogonal polynomial model `temp2`?

```
augment(temp2, newdata = data.frame(fruit_c = 0.5)) |>
  gt() |> tab_options(table.font.size = 20)
```

fruit_c	.fitted
0.5	37.91727

- No change in the prediction.

Fits of raw vs orthogonal polynomials

```
temp1_aug <- augment(temp1, train_c4im)
temp2_aug <- augment(temp2, train_c4im)

p1 <- ggplot(temp1_aug, aes(x = fruit_c, y = 1000/bmi)) +
    geom_point(alpha = 0.3) +
    geom_line(aes(x = fruit_c, y = .fitted), col = "red", size
    labs(title = "temp1: Raw fit, degree 2")

p2 <- ggplot(temp2_aug, aes(x = fruit_c, y = 1000/bmi)) +
    geom_point(alpha = 0.3) +
    geom_line(aes(x = fruit_c, y = .fitted), col = "blue", siz
    labs(title = "temp2: Orthogonal fit, degree 2")

p1 + p2 +
    plot_annotation(title = "Comparing Two Methods of Fitting
```

Comparing Two Methods of Fitting a Quadratic Polynomial

temp1: Raw fit, degree 2

temp2: Orthogonal fit, degree 2

Why use orthogonal polynomials?

- The main reason is to avoid having to include powers of our predictor that are highly collinear.
- Variance Inflation Factor assesses collinearity…

```
rms::vif(temp1)         ## from rms package
```

```
    fruit_c I(fruit_c^2) 
   1.665243     1.665243 
```

- Orthogonal polynomial terms are uncorrelated…

```
rms::vif(temp2)
```

```
poly(fruit_c, 2)1 poly(fruit_c, 2)2 
                1                 1 
```

Why orthogonal polynomials?

The tradeoff is that the raw polynomial is a lot easier to explain in terms of a single equation in the simplest case.

Actually, we'll often use splines instead of polynomials, which are more flexible and require less maintenance, but at the cost of pretty much requiring you to focus on visualizing their predictions rather than their equations. We'll talk about splines next time.

Adding a Second Order Polynomial

```
m3 <- lm(1000/bmi ~ poly(fruit_c,2) + exerany + health,
         data = train_c4im)
```

- Comparison to other models without the interaction…

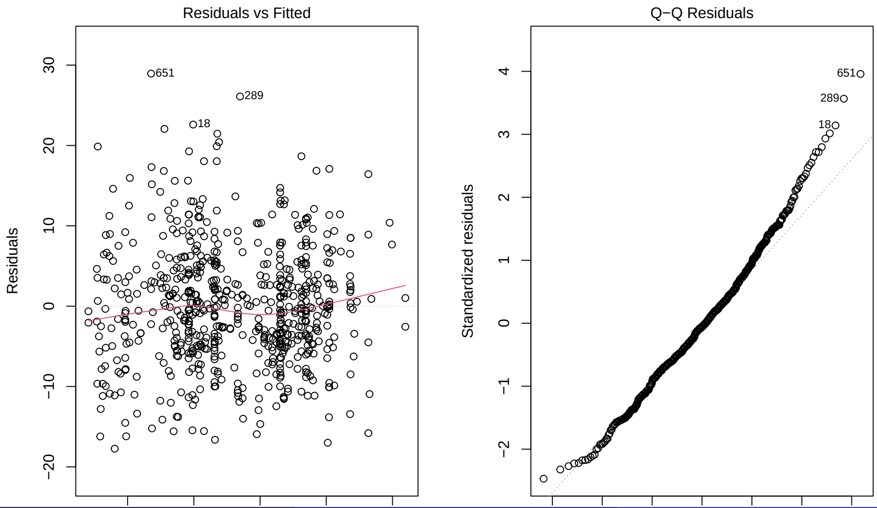
mod	r.squared	adj.r.squared	sigma	df	df.residual	nobs	AIC
m1	0.0642	0.0571	7.385	5	664	670	4,588.7
m2	0.0747	0.0663	7.349	6	663	670	4,583.1
m3	0.0749	0.0651	7.354	7	662	670	4,585.0

`m3` coefficients

term	estimate	std.error	statistic	p.value	conf.low	cor
(Intercept)	37.655	0.896	42.028	0.000	36.179	3
poly(fruit_c, 2)1	20.411	7.439	2.744	0.006	8.158	3
poly(fruit_c, 2)2	−2.908	7.529	−0.386	0.699	−15.309	
exerany	1.915	0.674	2.840	0.005	0.804	
healthVG	−0.642	0.845	−0.759	0.448	−2.034	
healthG	−3.406	0.871	−3.908	0.000	−4.841	−
healthF	−3.393	1.093	−3.103	0.002	−5.194	−
healthP	−3.723	1.646	−2.263	0.024	−6.434	−

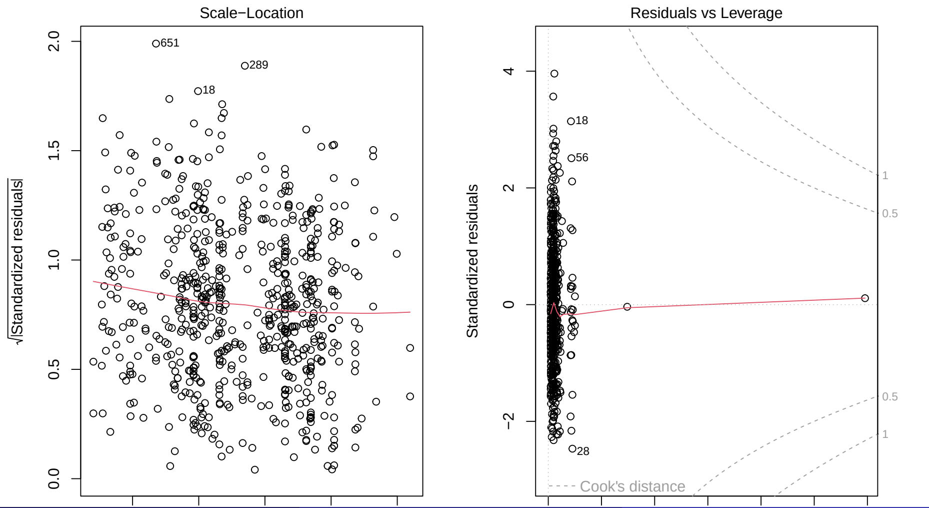
m3 Residuals

```r
par(mfrow = c(1,2)); plot(m3, which = c(1,2))
```



m3 Residuals

```r
par(mfrow = c(1,2)); plot(m3, which = c(3,5))
```



Add in the interaction

```r
m3int <- lm(1000/bmi ~ poly(fruit_c,2) + exerany * health,
           data = train_c4im)
```

- Comparison to other models with the interaction...

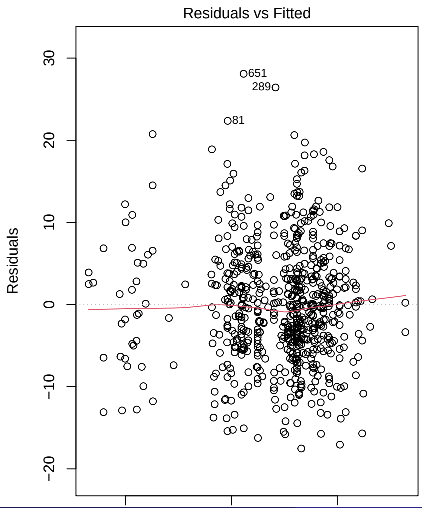
mod	r.squared	adj.r.squared	sigma	df	df.residual	nobs	AI
m1int	0.0910	0.0786	7.301	9	660	670	4,577.
m2int	0.1034	0.0898	7.256	10	659	670	4,570.
m3int	0.1034	0.0884	7.262	11	658	670	4,572.

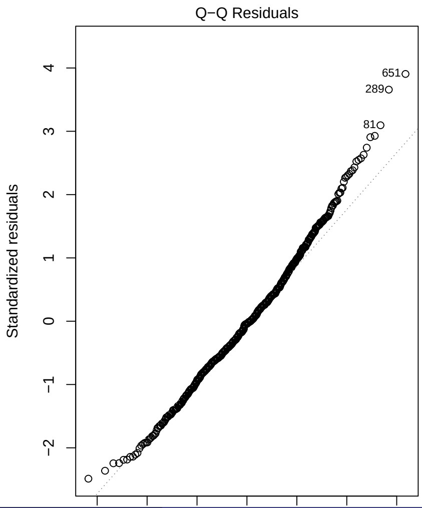
`m3int` coefficients

term	estimate	std.error	statistic	p.value	conf.low	co
(Intercept)	37.079	1.713	21.640	0.000	34.257	
poly(fruit_c, 2)1	22.219	7.364	3.017	0.003	10.090	
poly(fruit_c, 2)2	0.216	7.639	0.028	0.977	−12.368	
exerany	2.601	1.878	1.385	0.167	−0.492	
healthVG	1.711	1.977	0.866	0.387	−1.545	
healthG	−1.920	1.960	−0.980	0.328	−5.148	
healthF	−6.203	2.153	−2.882	0.004	−9.749	−
healthP	−7.754	3.168	−2.448	0.015	−12.972	−
exerany:healthVG	−2.961	2.181	−1.358	0.175	−6.553	
exerany:healthG	−1.940	2.182	−0.889	0.374	−5.534	
exerany:healthF	4.866	2.508	1.940	0.053	0.735	
exerany:healthP	5.951	3.709	1.604	0.109	−0.160	

m3int Residuals

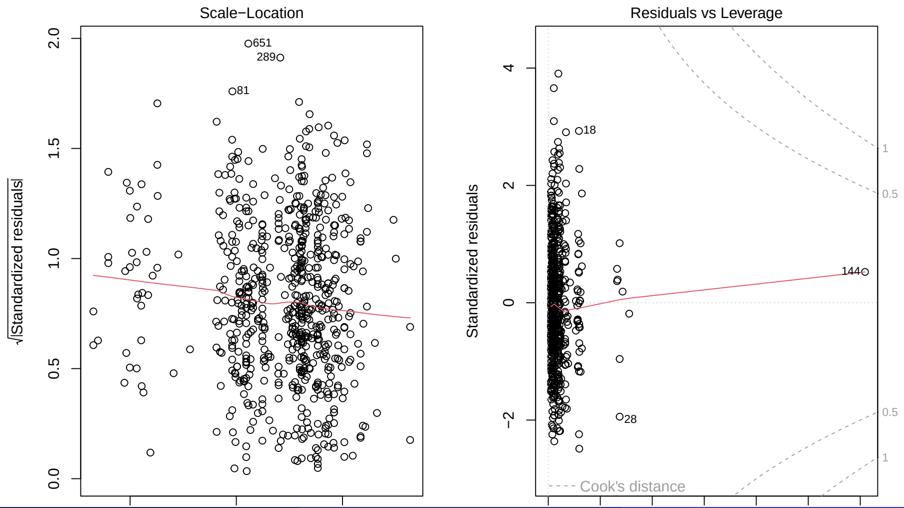
```r
par(mfrow = c(1,2)); plot(m3int, which = c(1,2))
```





`m3int` Residuals

```r
par(mfrow = c(1,2)); plot(m3int, which = c(3,5))
```



Testing Sample for `m3` and `m3int`?

```
m3_test_aug <- augment(m3, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)
m3int_test_aug <- augment(m3int, newdata = test_c4im) |>
  mutate(bmi_fit = 1000/.fitted, bmi_res = bmi - bmi_fit)

testing_r2 <- bind_rows(
    rsq(m1_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m2_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m3_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m1int_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m2int_test_aug, truth = bmi, estimate = bmi_fit),
    rsq(m3int_test_aug, truth = bmi, estimate = bmi_fit)) |>
    mutate(mod = c("m1", "m2", "m3", "m1int", "m2int", "m3int'
```

- I've hidden my calculations for RMSE and MAE here.

Test Results for all six models

```
bind_cols(testing_r2 |> select(mod, rsquare = .estimate),
          testing_rmse |> select(rmse = .estimate),
          testing_mae |> select(mae = .estimate)) |>
  mutate(elements = c("exerany + health", "add fruit_c", "add
  gt() |> fmt_number(columns = rsquare:mae, decimals = 4) |>
  tab_options(table.font.size = 20)
```

mod	rsquare	rmse	mae	elements
m1	0.0779	5.6277	4.2962	exerany + health
m2	0.0692	5.6497	4.3087	add fruit_c
m3	0.0698	5.6477	4.3052	add polynomial
m1int	0.0358	5.8417	4.4628	m1 + interaction
m2int	0.0318	5.8878	4.4850	m2 + interaction
m3int	0.0317	5.8887	4.4855	m3 + interaction

- Did the polynomial in `m3` and `m3int` improve predictions?

Next Week

- Fitting splines, as well as polynomial terms.
- Using the `ols` function from the **rms** package to fit linear regression models with non-linear terms.
- Submit Lab 2 to Canvas by Tuesday 2024-01-30 at Noon.

Section 8

Appendix

Creating Today's Data Set

```
url1 <- "https://raw.githubusercontent.com/THOMASELOVE/432-dat

smart_ohio <- read_csv(url1)

c4 <- 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) |>
    type.convert(as.is = FALSE) |>
    mutate(ID = as.character(SEQNO - 2017000000)) |>
    relocate(ID)
```

Codebook for useful c4 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 alcoholic drinks consumed per week
`female`	sex: $1 =$ female, $0 =$ male

Codebook for useful c4 variables (2)

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

Variable	Description
`exerany`	any exercise in the 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 on BRFSS SMART data

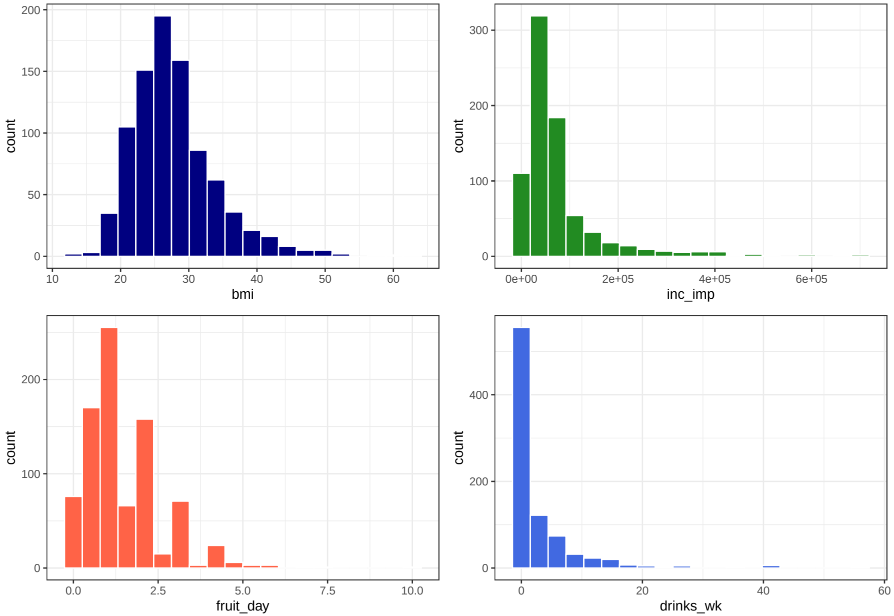
Basic Data Summaries

Available approaches include:

- `summary`
- `mosaic` package's `inspect()`
- `Hmisc` package's `describe`

all of which can work nicely in an HTML presentation, but none of them fit well on one of these slides.

Quick Histogram of each quantitative variable



Code for previous slide

```r
p1 <- ggplot(c4, aes(x = bmi)) +
    geom_histogram(fill = "navy", col = "white", bins = 20)
p2 <- ggplot(c4, aes(x = inc_imp)) +
    geom_histogram(fill = "forestgreen", col = "white",
                   bins = 20)
p3 <- ggplot(c4, aes(x = fruit_day)) +
    geom_histogram(fill = "tomato", col = "white", bins = 20)
p4 <- ggplot(c4, aes(x = drinks_wk)) +
    geom_histogram(fill = "royalblue", col = "white",
                   bins = 20)
(p1 + p2) / (p3 + p4)
```

I also used `#| warning: false` in the plot's code chunk label to avoid warnings about missing values, like this one for `inc_imp`:

```
Warning: Removed 120 rows containing non-finite values
```

Binary variables in raw c4

```
c4 |> 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 c4

```r
c4 |> tabyl(genhealth)
```

```
   genhealth   n     percent valid_percent
 1_Excellent 148 0.165548098    0.16573348
  2_VeryGood 324 0.362416107    0.36282195
      3_Good 274 0.306487696    0.30683091
      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

```
c4 <- c4 |>
    mutate(health =
               fct_recode(genhealth,
                          E = "1_Excellent",
                          VG = "2_VeryGood",
                          G = "3_Good",
                          F = "4_Fair",
                          P = "5_Poor"))
```

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

Checking health vs. genhealth in c4

```
c4 |> tabyl(genhealth, health) |> adorn_title()
```

```
             health                          
   genhealth      E  VG   G   F  P NA_
 1_Excellent    148   0   0   0  0   0
  2_VeryGood      0 324   0   0  0   0
      3_Good      0   0 274   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 preserved the order and we have much shorter labels. Sometimes, that's helpful.

Multicategorical race_eth in raw c4

```
c4 |> count(race_eth)
```

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

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

- What is this variable actually about?

Multicategorical race_eth in raw c4

```r
c4 |> count(race_eth)
```

```
# A tibble: 6 x 2
  race_eth                      n
  <fct>                     <int>
1 Black non-Hispanic          167
2 Hispanic                     27
3 Multiracial non-Hispanic     19
4 Other race non-Hispanic      22
5 White non-Hispanic          646
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 of the categories.
- 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...

```r
c4 <- c4 |>
    mutate(race_eth = fct_infreq(race_eth))

c4 |> 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

```
c4 <- c4 |>
    select(ID, bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, health, race_eth, everything())

miss_var_summary(c4)
```

```
# A tibble: 13 x 3
   variable    n_miss pct_miss
   <chr>        <int>    <dbl>
 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 genhealth        1    0.112
 8 ID               0    0
```

Single Imputation Approach?

```r
set.seed(43203)
c4im <- c4 |>
    select(ID, bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, health, race_eth) |>
    data.frame() |>
    impute_cart(health ~ bmi + female) |>
    impute_pmm(exerany ~ female + health + bmi) |>
    impute_rlm(inc_imp + drinks_wk + fruit_day ~
                   bmi + female + health + exerany) |>
    impute_cart(race_eth ~ health + inc_imp + bmi) |>
    tibble()

prop_miss_case(c4im)
```

```
[1] 0
```

Saving the tidied data

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

```
write_rds(c4, "c04/data/c4.Rds")

write_rds(c4im, "c04/data/c4im.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.