432 Class 03 Slides

thomaselove.github.io/432

2021-02-09

Today's Agenda

- Create a data set for week 2 analyses from `smart_ohio`
- Making cleaning / tidying decisions, then saving our work
- Simple imputation
- Splitting the sample with `rsample` tools
- Fitting a model (and then several more models) with `lm`
    - Incorporating an interaction between factors
    - Incorporating polynomial terms
- Regression Diagnostics via Residual Plots
- Evaluating results in holdout sample with `yardstick`

Setup

```
knitr::opts_chunk$set(comment = NA)
options(width = 60)

library(here)
library(knitr)
library(skimr)
library(patchwork)
library(janitor)
library(naniar)
library(simputation)
library(broom)
library(rsample)        ## new today: data splitting
library(yardstick)      ## new today: evaluating fits
library(tidyverse)

theme_set(theme_bw())
options(dplyr.summarise.inform = FALSE)  ## avoid message
```

Similar approach as last time...

```
smart_ohio <- read_csv(here("data/smart_ohio.csv"))

week2 <- 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() %>%
    mutate(ID = as.character(SEQNO - 2017000000)) %>%
    relocate(ID)
```

```
week2
```

```
# A tibble: 894 x 12
   ID      bmi inc_imp fruit_day drinks_wk female exerany
   <chr> <dbl>   <int>     <dbl>     <dbl>  <int>   <int>
 1 2      23.0   86865     4          0         1       0
 2 3      26.9      NA     3          0         1       1
 3 4      26.5      NA     2          4.67      1       1
 4 5      24.2   58311     0.570      0.93      0       1
 5 7      23.0    2318     2          2         0       1
 6 8      28.4   79667     1          0         0       1
 7 9      30.1   47880     0.23       0         0       1
 8 10     19.8  100136     0.77       0.47      1       1
 9 11     27.2   73145     0.71       0         0       1
10 12     24.6   76917     1.07       0         1       1
# ... with 884 more rows, and 5 more variables:
#   genhealth <fct>, race_eth <fct>, hx_diabetes <int>,
#   mmsa <fct>, SEQNO <int>
```

Codebook for useful `week2` variables

- 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
`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` (all Cleveland-Elyria)
- See Chapter 2 of the Course Notes for details on the variables

Basic Data Summaries

Available approaches include:

- `summary`
- `mosaic` package's `inspect()`
- `skimr` package's `skim_without_charts()`
- `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.

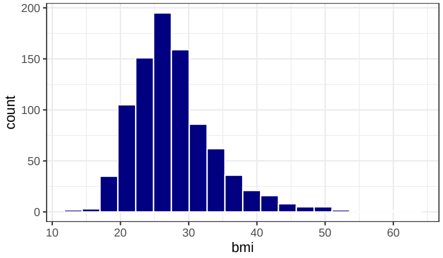
Summarizing the Quantities (Raw `week2`)

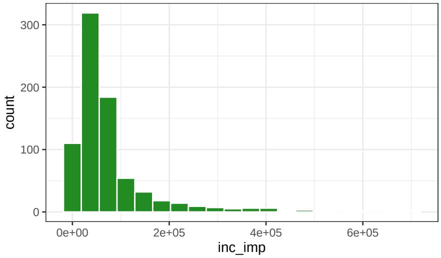
```
week2 %>% select(bmi, inc_imp, fruit_day, drinks_wk) %>%
    skim_without_charts() %>%
    yank(., "numeric") %>%
    select(var = skim_variable, n_missing, min = p0,
           median = p50, max = p100, mean, sd) %>%
    kable(digits = 1)
```

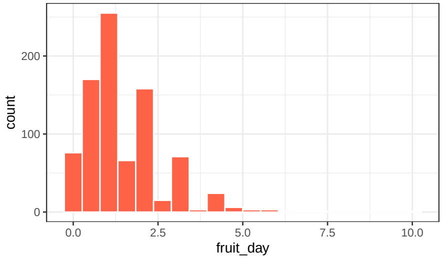
var	n_missing	min	median	max	mean	sd
bmi	0	13.3	26.8	63	27.9	6.3
inc_imp	120	216.0	48224.5	700676	75673.5	90695.8
fruit_day	41	0.0	1.1	10	1.4	1.1
drinks_wk	39	0.0	0.5	56	3.0	6.1

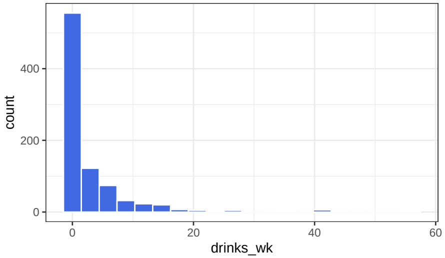
- Any signs of trouble? (What are we looking for?)

Quick Histogram of each quantitative variable









Code for previous slide

```r
p1 <- ggplot(week2, aes(x = bmi)) +
    geom_histogram(fill = "navy", col = "white", bins = 20)
p2 <- ggplot(week2, aes(x = inc_imp)) +
    geom_histogram(fill = "forestgreen", col = "white",
                   bins = 20)
p3 <- ggplot(week2, aes(x = fruit_day)) +
    geom_histogram(fill = "tomato", col = "white", bins = 20)
p4 <- ggplot(week2, 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 `week2`

```
week2 %>% 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?

Binary variables in raw `week2`

```
week2 %>% 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?
- I think the 1/0 values and names are OK choices.

Multicategorical genhealth in raw week2

```
week2 %>% 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 were each treated as missing.
- How might we manage this variable?

Changing the levels for `genhealth`

```
week2 <- week2 %>%
    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 week2

```
week2 %>% 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 `week2`

```r
week2 %>% 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 `week2`

```
week2 %>% 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. . .

```
week2 <- week2 %>%
    mutate(race_eth = fct_infreq(race_eth))
```

```
week2 %>% 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

```
week2 <- week2 %>%
    select(ID, bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, health, race_eth, everything())

miss_var_summary(week2)
```

```
# 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)
week2im <- week2 %>%
    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(week2im)
```

```
[1] 0
```

Saving the tidied data

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

```
saveRDS(week2, here("data", "week2.Rds"))

saveRDS(week2im, here("data", "week2im.Rds"))
```

To reload these files, we'd use `readRDS`.

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

Splitting the Sample

Use `initial_split` from `rsample` to partition the data into:

- Model development (training) sample where we'll build models
- Model evaluation (testing) sample which we'll hold out for a while

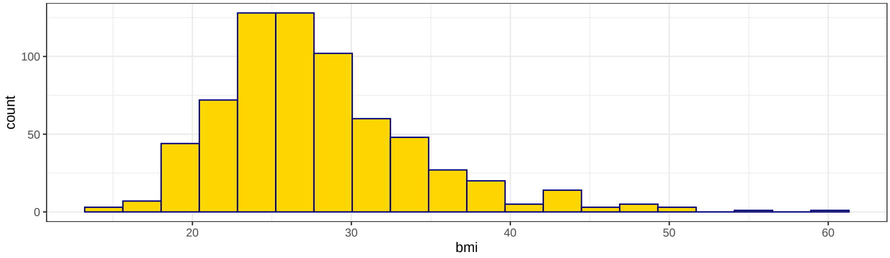
```
set.seed(432)    ## to make the work replicable in the future
week2im_split <- initial_split(week2im, prop = 3/4)

train_w2im <- training(week2im_split)
test_w2im <- testing(week2im_split)
```

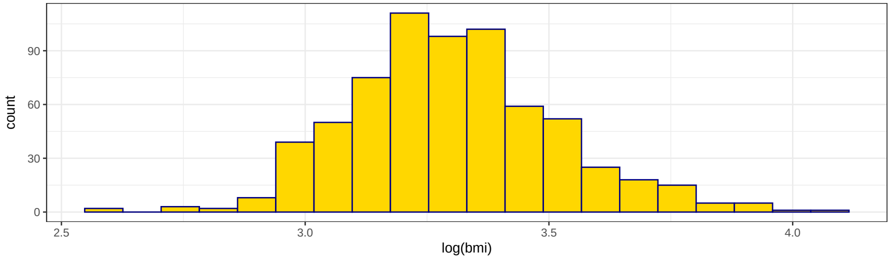
```
dim(train_w2im); dim(test_w2im)
```

```
[1] 671   9
```

```
[1] 223   9
```

Should we transform our outcome?





bmi means by exerany and health

```
summaries_1 <- train_w2im %>%
    group_by(exerany, health) %>%
    summarise(n = n(), mean = mean(bmi), stdev = sd(bmi))
summaries_1 %>% kable(digits = 2)
```

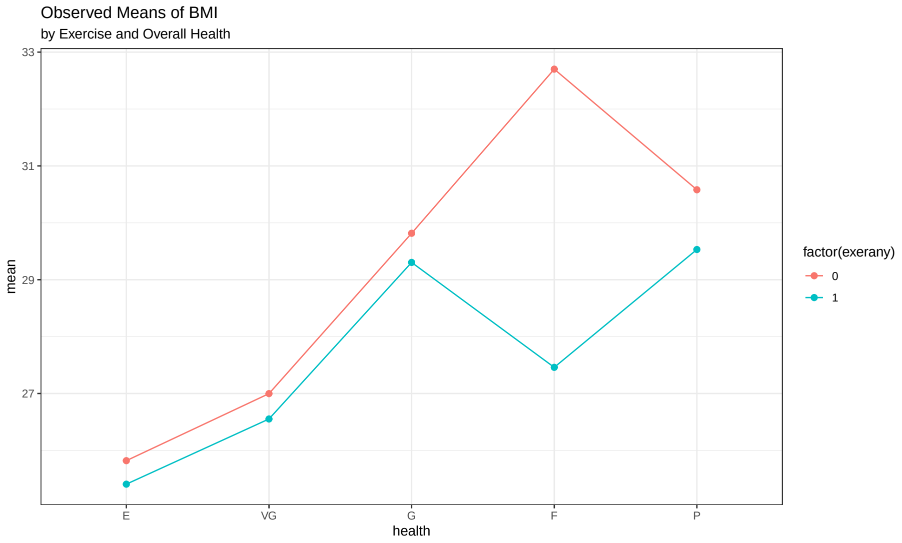
exerany	health	n	mean	stdev
0	E	13	25.82	4.99
0	VG	49	27.00	5.35
0	G	57	29.82	6.78
0	F	38	32.70	9.79
0	P	12	30.58	7.86
1	E	95	25.41	4.47
1	VG	195	26.55	4.63
1	G	147	29.30	6.31
1	F	51	27.46	6.07
1	P	14	29.53	10.21

Code for Interaction Plot

```r
ggplot(summaries_1, aes(x = health, y = mean,
                        col = factor(exerany))) +
    geom_point(size = 2) +
    geom_line(aes(group = factor(exerany))) +
    labs(title = "Observed Means of 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



Building a Model without interaction

```
m_1 <- lm(bmi ~ exerany + health,
          data = train_w2im)
```

- How well does this model fit the training data?

```
glance(m_1) %>%
    select(r.squared, adj.r.squared, sigma, nobs,
           df, df.residual, AIC, BIC) %>%
    kable(digits = c(3, 3, 2, 0, 0, 0, 1, 1))
```

r.squared	adj.r.squared	sigma	nobs	df	df.residual	AIC	BIC
0.082	0.075	6	671	5	665	4316	4347.5

ANOVA for the `m_1` model

```r
anova(m_1)
```

```
Analysis of Variance Table

Response: bmi
           Df  Sum Sq Mean Sq F value    Pr(>F)
exerany     1   546.0  546.04  15.185 0.0001073 ***
health      4  1599.6  399.90  11.121  9.75e-09 ***
Residuals 665 23912.2   35.96
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tidied ANOVA for the `m_1` model

```
tidy(anova(m_1)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

term	df	sumsq	meansq	statistic	p.value
exerany	1	546.04	546.04	15.19	0
health	4	1599.60	399.90	11.12	0
Residuals	665	23912.24	35.96	NA	NA

A summary of `m_1` coefficients

```r
summary(m_1)$coeff
```

```
              Estimate Std. Error   t value      Pr(>|t|)
(Intercept) 26.660139  0.7516549 35.468590 1.760701e-155
exerany     -1.368895  0.5476208 -2.499712  1.266926e-02
healthVG     1.075617  0.6944477  1.548882  1.218860e-01
healthG      3.773133  0.7188867  5.248578  2.064177e-07
healthF      3.821700  0.8747353  4.368978  1.447837e-05
healthP      4.092343  1.3232015  3.092759  2.066074e-03
```

Tidied summary of `m_1` coefficients

```
tidy(m_1, conf.int = TRUE, conf.level = 0.90) %>%
    kable(digits = c(0,2,2,2,3,2,2))
```

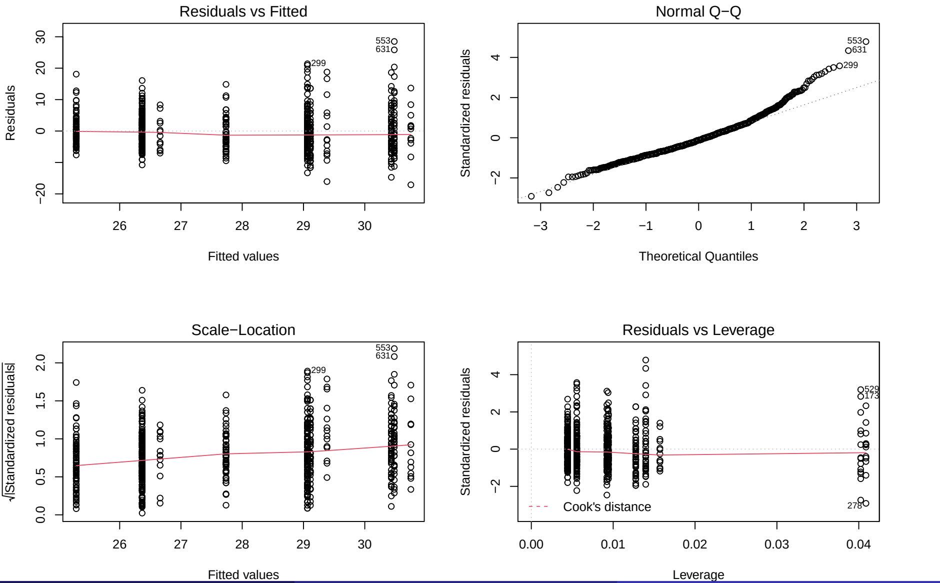
term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	26.66	0.75	35.47	0.000	25.42	27.90
exerany	-1.37	0.55	-2.50	0.013	-2.27	-0.47
healthVG	1.08	0.69	1.55	0.122	-0.07	2.22
healthG	3.77	0.72	5.25	0.000	2.59	4.96
healthF	3.82	0.87	4.37	0.000	2.38	5.26
healthP	4.09	1.32	3.09	0.002	1.91	6.27

Plot the Residuals from model `m_1`?

```r
par(mfrow = c(2,2))
plot(m_1)
par(mfrow = c(1,1))
```

That's the simplest code to get the four key plots to show up in the most familiar pattern, as shown on the next slide. . .

m_1 Residual Plots (conclusions?)



Adding the interaction term to `m_1`

```r
m_1int <- lm(bmi ~ exerany * health,
             data = train_w2im)
```

- How does this model compare in terms of fit to the training data?

```r
bind_rows(glance(m_1), glance(m_1int)) %>%
    mutate(mod = c("m_1", "m_1int")) %>%
    select(mod, r.sq = r.squared, adj.r.sq = adj.r.squared,
       sigma, nobs, df, df.res = df.residual, AIC, BIC) %>%
    kable(digits = c(0, 3, 3, 2, 0, 0, 0, 1, 1))
```

mod	r.sq	adj.r.sq	sigma	nobs	df	df.res	AIC	BIC
m_1	0.082	0.075	6.00	671	5	665	4316.0	4347.5
m_1int	0.098	0.085	5.96	671	9	661	4312.6	4362.2

ANOVA for the `m_1int` model

```
tidy(anova(m_1int)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

term	df	sumsq	meansq	statistic	p.value
exerany	1	546.04	546.04	15.35	0.000
health	4	1599.60	399.90	11.24	0.000
exerany:health	4	401.12	100.28	2.82	0.024
Residuals	661	23511.12	35.57	NA	NA

ANOVA test comparing m_1 to m_1int

```
anova(m_1, m_1int)
```

```
Analysis of Variance Table

Model 1: bmi ~ exerany + health
Model 2: bmi ~ exerany * health
  Res.Df   RSS Df Sum of Sq      F  Pr(>F)
1    665 23912
2    661 23511  4    401.12 2.8193 0.02442 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A summary of m_1int coefficients

```
summary(m_1int)$coeff
```

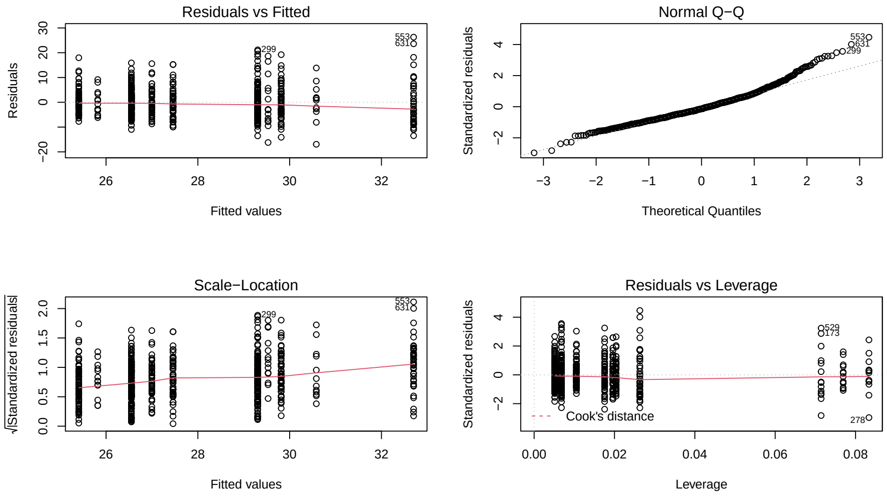
```
                      Estimate Std. Error     t value
(Intercept)        25.81923077   1.654109 15.60914361
exerany            -0.41291498   1.763658 -0.23412422
healthVG            1.17872841   1.860639  0.63350745
healthG             3.99690958   1.833056  2.18046193
healthF             6.88155870   1.916274  3.59111495
healthP             4.76160256   2.387501  1.99438748
exerany:healthVG -0.03278779   2.004692 -0.01635552
exerany:healthG  -0.09955190   1.994109 -0.04992299
exerany:healthF  -4.82826665   2.178060 -2.21677363
exerany:healthP  -0.63720407   2.935169 -0.21709281
                      Pr(>|t|)
(Intercept)        5.435751e-47
exerany            8.149610e-01
healthVG           5.266215e-01
```

Tidied summary of `m_1int` coefficients

```
tidy(m_1int, conf.int = TRUE, conf.level = 0.90) %>%
    rename(se = std.error, t = statistic, p = p.value) %>%
    kable(digits = c(0,2,2,2,3,2,2))
```

term	estimate	se	t	p	conf.low	conf.high
(Intercept)	25.82	1.65	15.61	0.000	23.09	28.54
exerany	-0.41	1.76	-0.23	0.815	-3.32	2.49
healthVG	1.18	1.86	0.63	0.527	-1.89	4.24
healthG	4.00	1.83	2.18	0.030	0.98	7.02
healthF	6.88	1.92	3.59	0.000	3.73	10.04
healthP	4.76	2.39	1.99	0.047	0.83	8.69
exerany:healthVG	-0.03	2.00	-0.02	0.987	-3.33	3.27
exerany:healthG	-0.10	1.99	-0.05	0.960	-3.38	3.19
exerany:healthF	-4.83	2.18	-2.22	0.027	-8.42	-1.24
exerany:healthP	-0.64	2.94	-0.22	0.828	-5.47	4.20

Plot the Residuals from model `m_1int`?



Adding in the covariate `fruit_day` to `m_1`

```
m_2 <- lm(bmi ~ fruit_day + exerany + health,
          data = train_w2im)
```

- How well does this model fit the training data?

```
bind_rows(glance(m_1), glance(m_2)) %>%
    mutate(mod = c("m_1", "m_2")) %>%
    select(mod, r.sq = r.squared, adj.r.sq = adj.r.squared,
        sigma, df, df.res = df.residual, AIC, BIC) %>%
    kable(digits = c(0, 3, 3, 2, 0, 0, 1, 1))
```

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1	0.082	0.075	6.00	5	665	4316.0	4347.5
m_2	0.090	0.081	5.98	6	664	4312.6	4348.7

- Also available in `glance` for a model fit with `lm` are `statistic`, `p.value`, `logLik`, and `deviance`.

ANOVA for the `m_2` model

```
tidy(anova(m_2)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

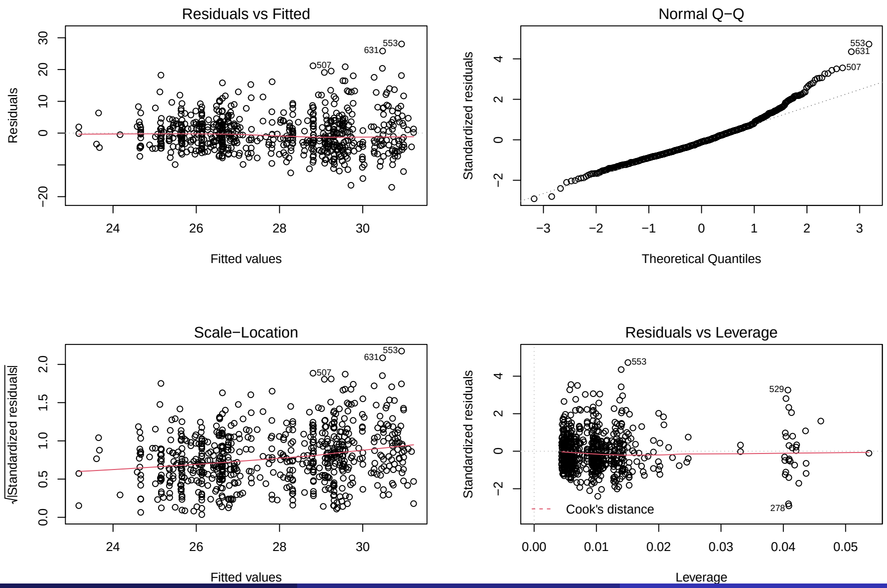
term	df	sumsq	meansq	statistic	p.value
fruit_day	1	413.34	413.34	11.57	0.001
exerany	1	411.23	411.23	11.51	0.001
health	4	1509.31	377.33	10.56	0.000
Residuals	664	23724.00	35.73	NA	NA

Tidied summary of `m_2` coefficients

```
tidy(m_2, conf.int = TRUE, conf.level = 0.90) %>%
    kable(digits = c(0,2,2,2,3,2,2))
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	27.34	0.81	33.93	0.000	26.01	28.67
fruit_day	-0.50	0.22	-2.30	0.022	-0.85	-0.14
exerany	-1.19	0.55	-2.15	0.032	-2.10	-0.28
healthVG	0.97	0.69	1.40	0.162	-0.17	2.11
healthG	3.65	0.72	5.09	0.000	2.47	4.84
healthF	3.64	0.88	4.16	0.000	2.20	5.08
healthP	3.92	1.32	2.96	0.003	1.74	6.09

`m_2` Residual Plots (non-constant variance?)



What if we included the interaction term?

```
m_2int <- lm(bmi ~ fruit_day + exerany * health,
             data = train_w2im)
```

Compare m_2int fit to previous models...

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1	0.082	0.075	6.00	5	665	4316.0	4347.5
m_2	0.090	0.081	5.98	6	664	4312.6	4348.7
m_1int	0.098	0.085	5.96	9	661	4312.6	4362.2
m_2int	0.106	0.093	5.94	10	660	4308.2	4362.3

- m_1 = no fruit_day, no exerany*health interaction
- m_2 = fruit_day, but no interaction
- m_1int = no fruit_day, with interaction
- m_2int = both fruit_day and interaction

ANOVA for the m_2int model

```
tidy(anova(m_2int)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

term	df	sumsq	meansq	statistic	p.value
fruit_day	1	413.34	413.34	11.71	0.001
exerany	1	411.23	411.23	11.65	0.001
health	4	1509.31	377.33	10.69	0.000
exerany:health	4	436.03	109.01	3.09	0.016
Residuals	660	23287.97	35.28	NA	NA

Tidied summary of `m_2int` coefficients

```
tidy(m_2int, conf.int = TRUE, conf.level = 0.90) %>%
    rename(se = std.error, t = statistic, p = p.value) %>%
    kable(digits = c(0,2,2,2,3,2,2))
```

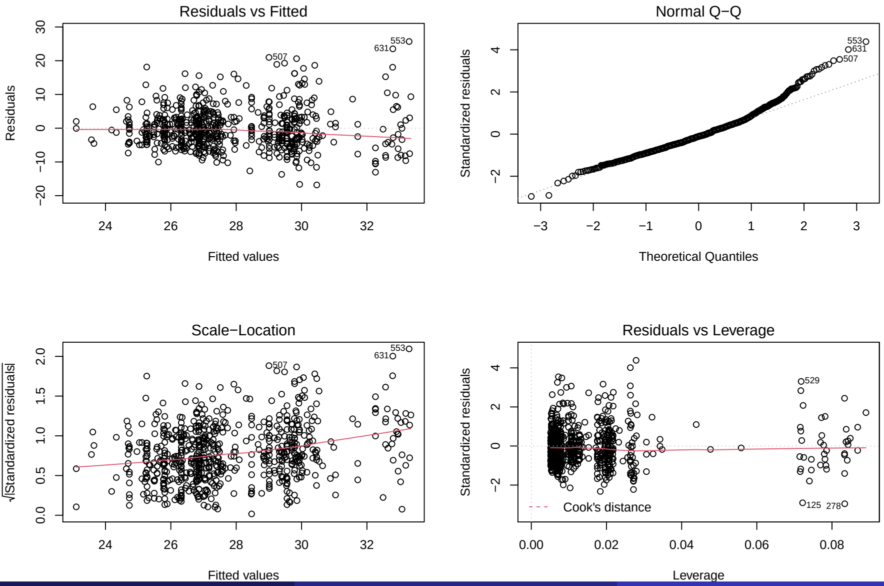
term	estimate	se	t	p	conf.low	conf.high
(Intercept)	26.50	1.67	15.87	0.000	23.75	29.25
fruit_day	-0.54	0.22	-2.51	0.012	-0.90	-0.19
exerany	-0.14	1.76	-0.08	0.935	-3.04	2.76
healthVG	1.03	1.85	0.56	0.578	-2.02	4.08
healthG	3.98	1.83	2.18	0.030	0.97	6.98
healthF	6.85	1.91	3.59	0.000	3.70	9.99
healthP	4.58	2.38	1.92	0.055	0.66	8.50
exerany:healthVG	0.02	2.00	0.01	0.993	-3.27	3.31
exerany:healthG	-0.23	1.99	-0.12	0.906	-3.51	3.04
exerany:healthF	-5.07	2.17	-2.33	0.020	-8.65	-1.49
exerany:healthP	-0.61	2.92	-0.21	0.835	-5.42	4.21

ANOVA comparison of m_2 and m_2int

```r
anova(m_2, m_2int)
```

```
Analysis of Variance Table

Model 1: bmi ~ fruit_day + exerany + health
Model 2: bmi ~ fruit_day + exerany * health
  Res.Df   RSS Df Sum of Sq      F  Pr(>F)
1    664 23724
2    660 23288  4    436.03 3.0893 0.01551 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual plots for model `m_2int`?



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
m_1	0.082	0.075	6.00	5	665	4316.0	4347.5
m_2	0.090	0.081	5.98	6	664	4312.6	4348.7
m_1int	0.098	0.085	5.96	9	661	4312.6	4362.2
m_2int	0.106	0.093	5.94	10	660	4308.2	4362.3

- The interaction models look better by Adjusted R^2 and σ; AIC likes `m_2int` while BIC likes `m1`. What to do?
- More importantly, the testing sample cannot judge between models accurately. Our models have already *seen* that data.
- For fairer comparisons, consider the (held out) testing sample. . .

Model predictions of `bmi` in the testing sample

We'll use `augment` from the `broom` package...

```
m1_test_aug <- augment(m_1, newdata = test_w2im)
m1int_test_aug <- augment(m_1int, newdata = test_w2im)
m2_test_aug <- augment(m_2, newdata = test_w2im)
m2int_test_aug <- augment(m_2int, newdata = test_w2im)
```

This adds fitted values (predictions) and residuals (errors) ...

```
m1_test_aug %>% select(ID, bmi, .fitted, .resid) %>%
    slice(1:2) %>% kable()
```

ID	bmi	.fitted	.resid
11	27.17	25.29124	1.878756
15	27.09	29.06438	-1.974377

Testing Results (using R^2)

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

```
testing_r2 <- bind_rows(
    rsq(m1_test_aug, truth = bmi, estimate = .fitted),
    rsq(m1int_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2int_test_aug, truth = bmi, estimate = .fitted)) %>%
    mutate(model = c("m_1", "m_1int", "m_2", "m_2int"))
testing_r2 %>% kable(dig = 4)
```

.metric	.estimator	.estimate	model
rsq	standard	0.0828	m_1
rsq	standard	0.0881	m_1int
rsq	standard	0.0782	m_2
rsq	standard	0.0829	m_2int

Mean Absolute Error?

Consider the mean absolute prediction error . . .

```
testing_mae <- bind_rows(
    mae(m1_test_aug, truth = bmi, estimate = .fitted),
    mae(m1int_test_aug, truth = bmi, estimate = .fitted),
    mae(m2_test_aug, truth = bmi, estimate = .fitted),
    mae(m2int_test_aug, truth = bmi, estimate = .fitted)) %>%
    mutate(model = c("m_1", "m_1int", "m_2", "m_2int"))
testing_mae %>% kable(dig = 3)
```

.metric	.estimator	.estimate	model
mae	standard	4.447	m_1
mae	standard	4.458	m_1int
mae	standard	4.411	m_2
mae	standard	4.425	m_2int

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 = .fitted),
   rmse(m1int_test_aug, truth = bmi, estimate = .fitted),
   rmse(m2_test_aug, truth = bmi, estimate = .fitted),
   rmse(m2int_test_aug, truth = bmi, estimate = .fitted)) %>%
   mutate(model = c("m_1", "m_1int", "m_2", "m_2int"))
testing_rmse %>% kable(digits = 3)
```

.metric	.estimator	.estimate	model
rmse	standard	6.095	m_1
rmse	standard	6.079	m_1int
rmse	standard	6.110	m_2
rmse	standard	6.096	m_2int

Other Summaries for Numerical Predictions

Within the `yardstick` package, there are several other summaries, including:

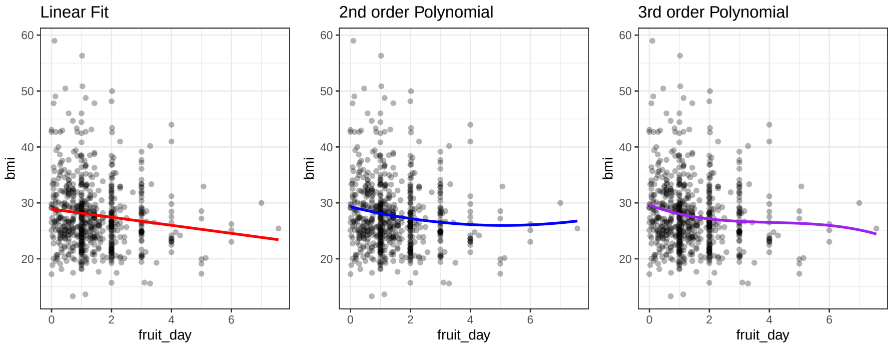
- `rsq_trad()` = defines R^2 using sums of squares.
  - The `rsq()` measure we showed a few slides ago is a squared correlation coefficient and is guaranteed to fall in (0, 1).
- `mape()` = mean absolute percentage error
- `mpe()` = mean percentage error
- `huber_loss()` = Huber loss (often used in robust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

Incorporating a non-linear term for `fruit_day`

Suppose we wanted to include a polynomial term for `fruit_day`:

```
lm(bmi ~ fruit_day, data = train_w2im)
lm(bmi ~ poly(fruit_day, 2), data = train_w2im)
lm(bmi ~ poly(fruit_day, 3), data = train_w2im)
```



Raw Polynomials vs. Orthogonal Polynomials

Predict `bmi` using `fruit_day` with a polynomial of degree 2.

```
(temp1 <- lm(bmi ~ fruit_day + I(fruit_day^2),
             data = train_w2im))
```

```

Call:
lm(formula = bmi ~ fruit_day + I(fruit_day^2), data = train_w2

Coefficients:
   (Intercept)       fruit_day  I(fruit_day^2)
       29.2991         -1.3079          0.1284
```

This uses raw polynomials. Predicted `bmi` for `fruit_day = 2` is

```
bmi = 29.2991 - 1.3079 (fruit_day) + 0.1284 (fruit_day^2)
    = 29.2991 - 1.3079 (2) + 0.1284 (4)
    = 27.1969
```

Does the raw polynomial match our expectations?

```
temp1 <- lm(bmi ~ fruit_day + I(fruit_day^2),
            data = train_w2im)
```

```
augment(temp1, newdata = data.frame(fruit_day = 2)) %>%
    kable(digits = 4)
```

fruit_day	.fitted
2	27.1969

and this matches our "by hand" calculation. But it turns out most regression models use orthogonal rather than raw polynomials...

Fitting an Orthogonal Polynomial

Predict `bmi` using `fruit_day` with an **orthogonal** polynomial of degree 2.

```
(temp2 <- lm(bmi ~ poly(fruit_day,2), data = train_w2im))
```

```
Call:
lm(formula = bmi ~ poly(fruit_day, 2), data = train_w2im)

Coefficients:
        (Intercept)  poly(fruit_day, 2)1
              27.84               -20.33
poly(fruit_day, 2)2
               7.21
```

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 both concluded that the predicted `bmi` for a subject with `fruit_day` = 2 was 27.1969.

- Now, what happens with the orthogonal polynomial model `temp2` we just fit?

```
augment(temp2, newdata = data.frame(fruit_day = 2)) %>%
    kable(digits = 4)
```

fruit_day	.fitted
2	27.1969

- No change in the prediction.

Why do we use orthogonal polynomials?

- The main reason is to avoid having to include powers of our predictor that are highly collinear. (x, x^2 and x^3, for instance, are often highly correlated.)
- Instead, the orthogonal polynomial terms are uncorrelated with one another, so it's relatively easy to identify which of the polynomial terms are actually valuable in our model.

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 usually avoid polynomials in our practical work, and instead use splines, 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.

Adding a Second Order Polynomial to our Models

```
m_3 <- lm(bmi ~ poly(fruit_day,2) + exerany + health,
          data = train_w2im)
```

- Comparison to other models without the interaction…

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1	0.0823	0.0754	6.00	5	665	4316.0	4347.5
m_2	0.0896	0.0813	5.98	6	664	4312.6	4348.7
m_3	0.0903	0.0807	5.98	7	663	4314.1	4354.7

ANOVA for the m_3 model

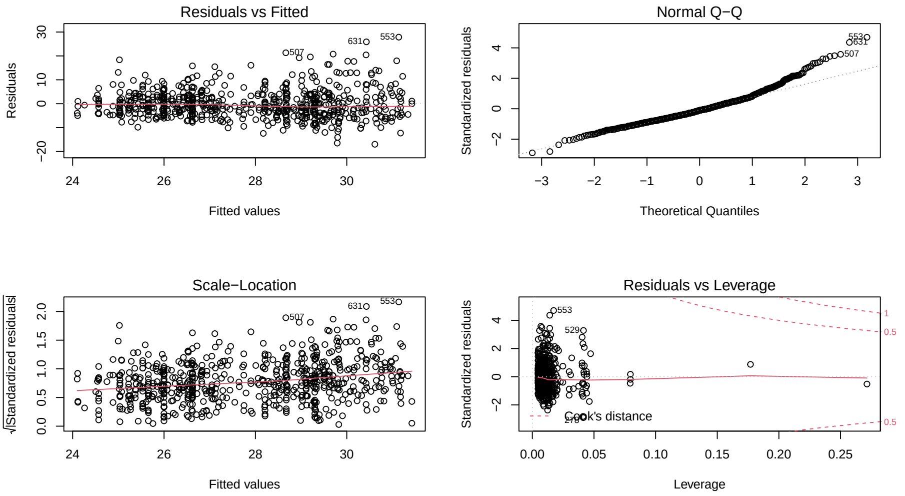
```
tidy(anova(m_3)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

term	df	sumsq	meansq	statistic	p.value
poly(fruit_day, 2)	2	465.32	232.66	6.51	0.002
exerany	1	376.32	376.32	10.53	0.001
health	4	1511.16	377.79	10.57	0.000
Residuals	663	23705.07	35.75	NA	NA

Tidied summary of `m_3` coefficients

term	est	se	t	p	conf.low	conf.high
(Intercept)	26.58	0.75	35.35	0.000	25.35	27.82
poly(fruit_day, 2)1	-14.08	6.09	-2.31	0.021	-24.12	-4.05
poly(fruit_day, 2)2	4.41	6.06	0.73	0.467	-5.58	14.40
exerany	-1.12	0.56	-2.01	0.045	-2.04	-0.20
healthVG	0.96	0.69	1.39	0.165	-0.18	2.11
healthG	3.64	0.72	5.07	0.000	2.46	4.83
healthF	3.66	0.88	4.18	0.000	2.22	5.11
healthP	3.92	1.32	2.97	0.003	1.75	6.10

m_3 Residual Plots



Add in the interaction

```
m_3int <- lm(bmi ~ poly(fruit_day,2) + exerany * health,
             data = train_w2im)
```

- Comparison to other models with the interaction...

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1int	0.0977	0.0854	5.96	9	661	4312.6	4362.2
m_2int	0.1063	0.0928	5.94	10	660	4308.2	4362.3
m_3int	0.1074	0.0925	5.94	11	659	4309.4	4368.0

ANOVA for the m_3int model

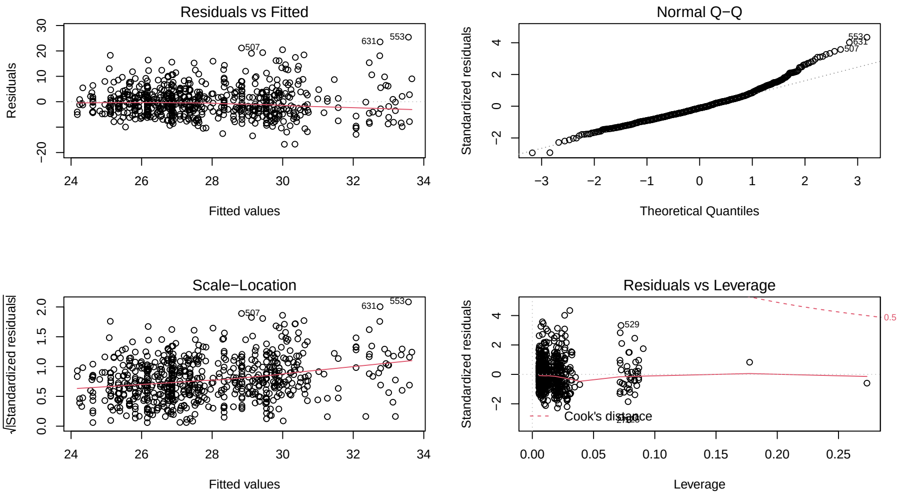
```
tidy(anova(m_3int)) %>%
    kable(dig = c(0, 0, 2, 2, 2, 3))
```

term	df	sumsq	meansq	statistic	p.value
poly(fruit_day, 2)	2	465.32	232.66	6.59	0.001
exerany	1	376.32	376.32	10.66	0.001
health	4	1511.16	377.79	10.70	0.000
exerany:health	4	444.77	111.19	3.15	0.014
Residuals	659	23260.30	35.30	NA	NA

Tidied summary of `m_3int` coefficients

term	est	se	t	p	conf.low	conf.high
(Intercept)	25.64	1.65	15.53	0.000	22.92	28.36
poly(fruit_day, 2)1	-15.42	6.08	-2.54	0.011	-25.43	-5.41
poly(fruit_day, 2)2	5.34	6.03	0.89	0.376	-4.59	15.28
exerany	-0.03	1.76	-0.02	0.987	-2.94	2.88
healthVG	1.04	1.85	0.56	0.574	-2.01	4.10
healthG	3.99	1.83	2.19	0.029	0.99	7.00
healthF	6.93	1.91	3.62	0.000	3.78	10.07
healthP	4.60	2.38	1.93	0.054	0.68	8.52
exerany:healthVG	-0.01	2.00	0.00	0.997	-3.30	3.28
exerany:healthG	-0.27	1.99	-0.14	0.891	-3.55	3.00
exerany:healthF	-5.15	2.17	-2.37	0.018	-8.73	-1.57
exerany:healthP	-0.61	2.92	-0.21	0.835	-5.42	4.21

`m_3int` Residual Plots



How do models m_3 and m_3int do in testing?

```
m3_test_aug <- augment(m_3, newdata = test_w2im)
m3int_test_aug <- augment(m_3int, newdata = test_w2im)

testing_r2 <- bind_rows(
    rsq(m1_test_aug, truth = bmi, estimate = .fitted),
    rsq(m1int_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2int_test_aug, truth = bmi, estimate = .fitted),
    rsq(m3_test_aug, truth = bmi, estimate = .fitted),
    rsq(m3int_test_aug, truth = bmi, estimate = .fitted)) %>%
    mutate(model = c("m_1", "m_1int", "m_2", "m_2int",
                     "m_3", "m_3int"))
```

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

Results comparing all six models (testing)

```
bind_cols(testing_r2 %>% select(model, rsquare = .estimate),
          testing_rmse %>% select(rmse = .estimate),
          testing_mae %>% select(mae = .estimate)) %>%
    kable(digits = c(0, 4, 3, 3))
```

model	rsquare	rmse	mae
m_1	0.0828	6.095	4.447
m_1int	0.0881	6.079	4.458
m_2	0.0782	6.110	4.411
m_2int	0.0829	6.096	4.425
m_3	0.0764	6.116	4.430
m_3int	0.0806	6.105	4.444

- Did the polynomial term in `m_3` and `m_3int` improve our predictions?

Next Time

- Feedback from the Minute Paper after Class 03, due tomorrow at Noon, please.
- Incorporating splines into linear regression models
- Using the `ols` modeling structure (from the `rms` package) to fit and assess linear regression models
- The Spearman ρ^2 plot, and some thoughts on how to spend data / degrees of freedom on nonlinearity