432 Class 04

https://thomaselove.github.io/432-2023/

2023-01-26

Today's Agenda

- Fitting two-factor ANOVA/ANCOVA models with `lm`
    - Incorporating an interaction between factors
    - Incorporating polynomial terms
    - Incorporating restricted cubic splines
- Regression Diagnostics via Residual Plots
- Validating / evaluating results with `yardstick`

Appendix

How the `class4im` data were created from `smart_ohio.csv`

- Data Ingest, Cleaning, and Single Imputation to deal with missingness (assumes MAR)

Chapters 6-13 of the Course Notes are relevant here.

Today's R Setup

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

library(janitor)
library(broom)
library(knitr)
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 `class4im` data (creation described in some detail in Appendix)

Codebook for useful `class4im` variables

- 894 subjects in Cleveland-Elyria with `bmi` and no history of diabetes (missing values singly imputed: assume MAR)
- There are 9 variables in the data but we'll use only these five 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

- All subjects have `hx_diabetes` (all 0), and `MMSA` (Cleveland-Elyria).
- See Course Notes Chapter on BRFSS SMART data for variable details
- Appendix provides details on data development.

Data Load

```
class4im <- read_rds("c04/data/class4im.Rds")
class4im |> n_miss()
```

```
[1] 0
```

```
identical(nrow(class4im), n_distinct(class4im$ID))
```

```
[1] TRUE
```

Splitting the Sample

```
set.seed(432)    ## for future replication
class4im_split <- initial_split(class4im, prop = 3/4)
train_c4im <- training(class4im_split)
test_c4im <- testing(class4im_split)
c(nrow(class4im), nrow(train_c4im), nrow(test_c4im))
```

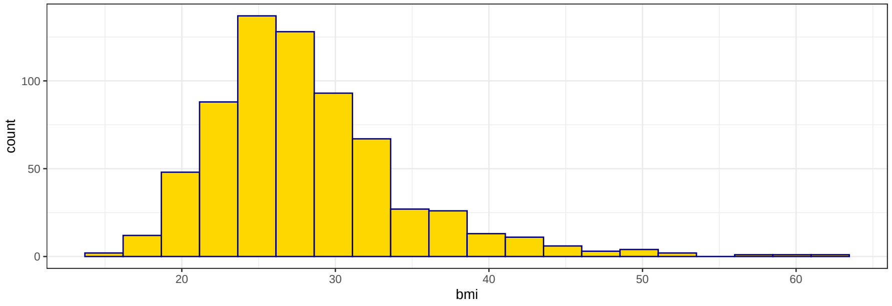
```
[1] 894 670 224
```

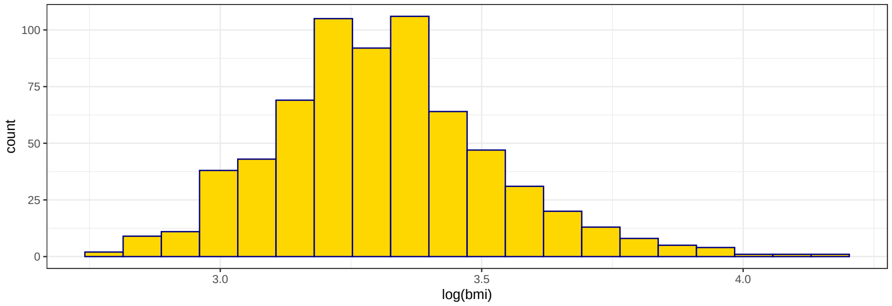
Models We'll Build Today

1. Predict `bmi` using `exer_any` and `genhealth` (both categorical)
   - without and then with an interaction between the two predictors
2. Add in a quantitative covariate, `fruit_day`, first simply as a main (and linear) effect
3. Incorporate the `fruit_day` information using a quadratic polynomial instead.
4. Incorporate the `fruit_day` information using a restricted cubic spline with 4 knots instead.

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.

We could, but won't transform our outcome.





bmi means by exerany and health

```
summaries_1 <- train_c4im |>
    group_by(exerany, health) |>
    summarise(n = n(), mean = mean(bmi), stdev = sd(bmi))
summaries_1 |> kable(digits = 2)
```

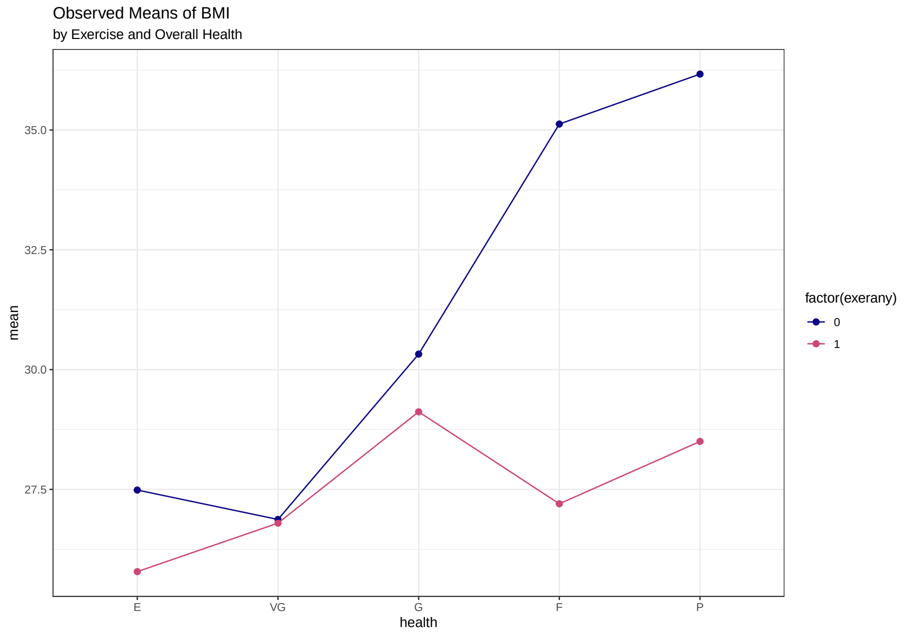
exerany	health	n	mean	stdev
0	E	18	27.49	3.56
0	VG	54	26.87	5.27
0	G	58	30.32	7.45
0	F	31	35.12	9.95
0	P	8	36.17	12.13
1	E	92	25.78	4.49
1	VG	191	26.80	4.89
1	G	152	29.12	6.27
1	F	49	27.20	5.52
1	P	17	28.50	8.61

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 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 BMI

Building a Model (m_1) without interaction

```
m_1 <- lm(bmi ~ exerany + health,
          data = train_c4im)
```

- 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
0.089	0.082	6.12	670	5	664	4335.8	4367.3

Tidied ANOVA for `m_1`

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

term	df	sumsq	meansq	statistic	p.value
exerany	1	896.51	896.51	23.97	0
health	4	1526.49	381.62	10.21	0
Residuals	664	24830.10	37.39	NA	NA

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))
```

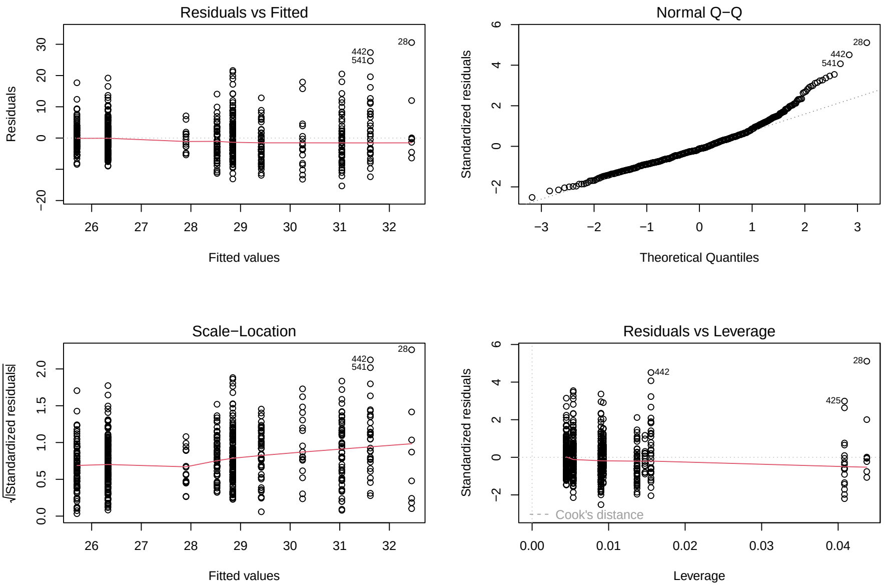
term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	27.90	0.74	37.57	0.000	26.68	29.12
exerany	-2.20	0.55	-4.00	0.000	-3.10	-1.29
healthVG	0.63	0.70	0.89	0.374	-0.53	1.78
healthG	3.14	0.72	4.35	0.000	1.95	4.33
healthF	3.72	0.91	4.10	0.000	2.22	5.21
healthP	4.55	1.36	3.35	0.001	2.31	6.78

Interpreting `m_1`

Name	`exerany`	`health`	predicted `bmi`
Harry	0	Excellent	27.91
Sally	1	Excellent	27.91 - 2.20 = 25.71
Billy	0	Fair	27.91 + 3.71 = 31.62
Meg	1	Fair	27.91 - 2.20 + 3.71 = 29.42

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

`m_1` Residual Plots (conclusions?)



Section 3

Fitting ANOVA model `m_1int` including interaction

Adding the interaction term to m_1

```
m_1int <- lm(bmi ~ exerany * health,
             data = train_c4im)
```

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

```
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	0.089	0.082	6.12	670	5	664	4335.8	4367.3
m_1int	0.126	0.114	6.01	670	9	660	4315.7	4365.3

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	896.51	896.51	24.85	0
health	4	1526.49	381.62	10.58	0
exerany:health	4	1019.61	254.90	7.07	0
Residuals	660	23810.49	36.08	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    664 24830
2    660 23811  4    1019.6 7.0656 1.424e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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))
```

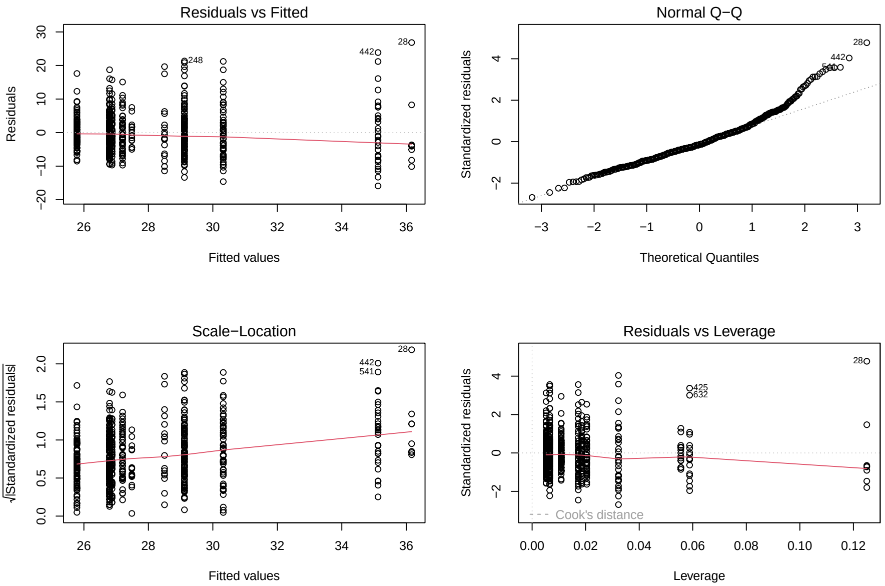
term	estimate	se	t	p	conf.low	conf.high
(Intercept)	27.49	1.42	19.42	0.000	25.16	29.82
exerany	-1.70	1.55	-1.10	0.272	-4.25	0.85
healthVG	-0.61	1.63	-0.38	0.707	-3.31	2.08
healthG	2.84	1.62	1.75	0.081	0.17	5.51
healthF	7.64	1.78	4.29	0.000	4.70	10.57
healthP	8.68	2.55	3.40	0.001	4.48	12.88
exerany:healthVG	1.63	1.80	0.90	0.368	-1.35	4.60
exerany:healthG	0.50	1.80	0.28	0.783	-2.47	3.47
exerany:healthF	-6.22	2.07	-3.00	0.003	-9.64	-2.81
exerany:healthP	-5.96	3.00	-1.98	0.048	-10.91	-1.01

Interpreting the `m_1int` model

Name	`exerany`	`health`	predicted `bmi`
Harry	0	Excellent	27.49
Sally	1	Excellent	27.49 - 1.69 = 25.80
Billy	0	Fair	27.49 + 7.64 = 35.13
Meg	1	Fair	27.49 - 1.69 + 7.64 - 6.22 = 27.22

- How do we interpret effect sizes here? **It depends**.
- Effect of `exerany`?
  - If `health` = Excellent, effect is -1.69
  - If `health` = Fair, effect is (-1.69 - 6.22) = -7.91
- Effect of `health` = Fair instead of Excellent?
  - If `exerany` = 0 (no), effect is 7.64
  - If `exerany` = 1 (yes), effect is (7.64 - 6.22) = 1.42

Plot the Residuals from model `m_1int`?



Section 4

Incorporating a Covariate (as a main and linear effect) into our two-way ANOVA models

Adding in the covariate `fruit_day` to `m_1`

```r
m_2 <- lm(bmi ~ fruit_day + exerany + health,
          data = train_c4im)
```

- How well does this model fit the training data?

```r
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	0.089	0.082	6.12	5	664	4335.8	4367.3
m_2	0.098	0.090	6.09	6	663	4331.1	4367.2

ANOVA for the `m_2` model

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

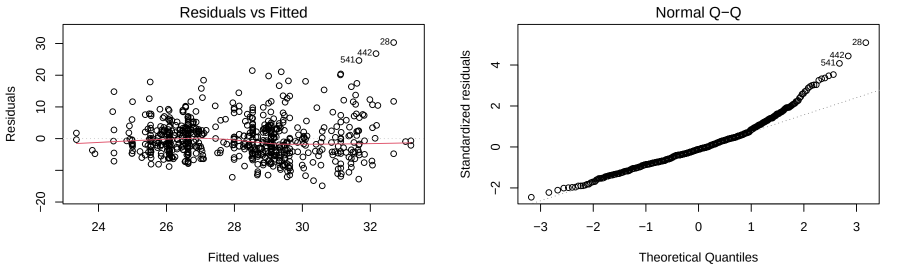
term	df	sumsq	meansq	statistic	p.value
fruit_day	1	467.13	467.13	12.60	0
exerany	1	761.38	761.38	20.53	0
health	4	1439.70	359.93	9.71	0
Residuals	663	24584.90	37.08	NA	NA

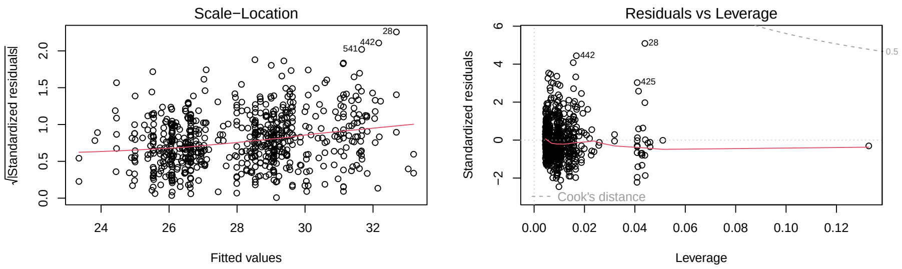
Tidied summary of `m_2` coefficients

```
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)	28.67	0.80	35.93	0.000	27.36	29.99
fruit_day	-0.55	0.21	-2.57	0.010	-0.90	-0.20
exerany	-2.05	0.55	-3.72	0.000	-2.96	-1.14
healthVG	0.56	0.70	0.80	0.423	-0.59	1.71
healthG	3.01	0.72	4.17	0.000	1.82	4.19
healthF	3.55	0.91	3.92	0.000	2.06	5.04
healthP	4.56	1.35	3.37	0.001	2.34	6.79

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





Who is that poorest fit case?

Plot suggests we look at row 28

```
train_c4im |> slice(28) |>
    select(ID, bmi, fruit_day, exerany, health) |> kable()
```

ID	bmi	fruit_day	exerany	health
320	63	1	0	P

What is unusual about this subject?

```
train_c4im |> arrange(desc(bmi))
```

```
# A tibble: 670 x 9
   ID      bmi inc_imp fruit_day drinks_wk female exerany heal
   <chr> <dbl>   <dbl>     <dbl>     <dbl>  <int>   <int> <fct
 1 320    63     20581     1           0.7      1       0 P
 2 959    59.0    5720     0.1         0        1       0 F
```

What if we included the interaction term?

```
m_2int <- lm(bmi ~ fruit_day + exerany * health,
             data = train_c4im)
```

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	467.13	467.13	13.10	0
exerany	1	761.38	761.38	21.35	0
health	4	1439.70	359.93	10.09	0
exerany:health	4	1079.20	269.80	7.56	0
Residuals	659	23505.69	35.67	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))
```

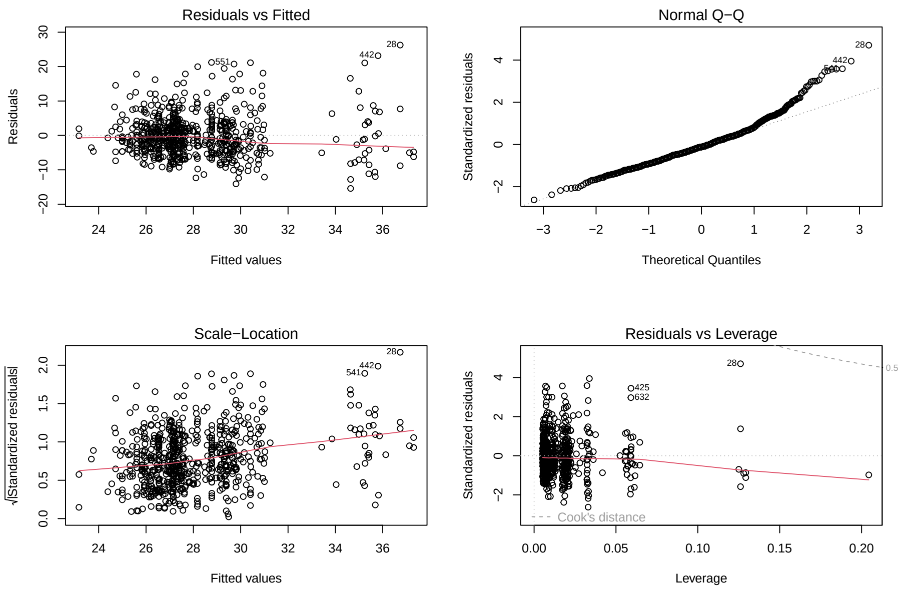
term	estimate	se	t	p	conf.low	conf.high
(Intercept)	28.27	1.43	19.73	0.000	25.91	30.63
fruit_day	-0.61	0.21	-2.92	0.004	-0.95	-0.27
exerany	-1.45	1.54	-0.94	0.349	-3.99	1.09
healthVG	-0.66	1.63	-0.40	0.686	-3.33	2.02
healthG	2.75	1.61	1.71	0.089	0.09	5.40
healthF	7.59	1.77	4.29	0.000	4.67	10.50
healthP	9.07	2.54	3.57	0.000	4.89	13.26
exerany:healthVG	1.60	1.79	0.89	0.374	-1.36	4.55
exerany:healthG	0.42	1.79	0.24	0.814	-2.53	3.38
exerany:healthF	-6.41	2.06	-3.11	0.002	-9.81	-3.01
exerany:healthP	-6.50	2.99	-2.17	0.030	-11.43	-1.57

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    663 24585                                   
2    659 23506  4    1079.2 7.5641 5.829e-06 ***
---
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	0.089	0.082	6.12	5	664	4335.8	4367.3
m_2	0.098	0.090	6.09	6	663	4331.1	4367.2
m_1int	0.126	0.114	6.01	9	660	4315.7	4365.3
m_2int	0.138	0.124	5.97	10	659	4309.1	4363.1

- Adjusted R^2, σ, AIC and BIC all improve as we move down from `m1` towards `m2_int`.
- BUT the testing sample cannot judge between models accurately. Our models have already *seen* that data.
- For fairer comparisons, we'll need to also consider the (held out) testing sample.

Model predictions of `bmi` in the test sample

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

```
m1_test_aug <- augment(m_1, newdata = test_c4im)
m1int_test_aug <- augment(m_1int, newdata = test_c4im)
m2_test_aug <- augment(m_2, newdata = test_c4im)
m2int_test_aug <- augment(m_2int, newdata = test_c4im)
```

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

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

ID	bmi	.fitted	.resid
4	26.51	28.84456	-2.334562
5	24.25	28.84456	-4.594562

What does the `yardstick` package do?

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, such as:

- 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 (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	0.0716	m_1
rsq	standard	0.0397	m_1int
rsq	standard	0.0652	m_2
rsq	standard	0.0364	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 = 2)
```

.metric	.estimator	.estimate	model
mae	standard	4.43	m_1
mae	standard	4.62	m_1int
mae	standard	4.48	m_2
mae	standard	4.71	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	5.729	m_1
rmse	standard	6.025	m_1int
rmse	standard	5.769	m_2
rmse	standard	6.082	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.

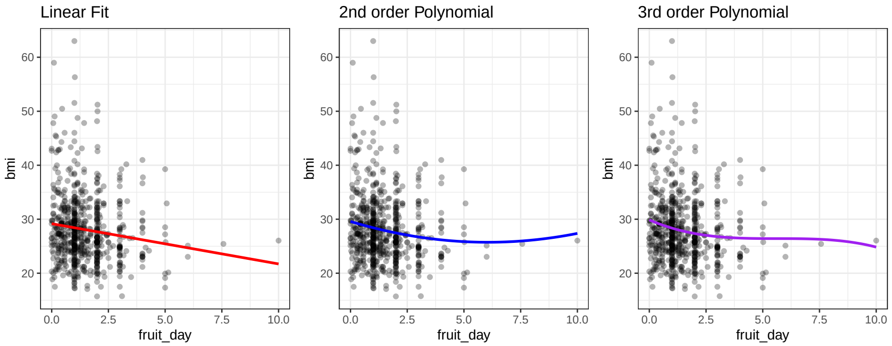
Section 5

Incorporating Non-Linearity into our models

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_c4im)
lm(bmi ~ poly(fruit_day, 2), data = train_c4im)
lm(bmi ~ poly(fruit_day, 3), data = train_c4im)
```



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**.

Raw Polynomials vs. Orthogonal Polynomials

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

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

```

Call:
lm(formula = bmi ~ fruit_day + I(fruit_day^2), data = train_c4

Coefficients:
   (Intercept)       fruit_day  I(fruit_day^2)
       29.5866         -1.2726          0.1052
```

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

```
bmi = 29.5925 - 1.2733 (fruit_day) + 0.1051 (fruit_day^2)
    = 29.5925 - 1.2733 (2) + 0.1051 (4)
    = 27.466
```

Does the raw polynomial match our expectations?

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

```r
augment(temp1, newdata = tibble(fruit_day = 2)) |>
    kable(digits = 3)
```

fruit_day	.fitted
2	27.462

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_c4im))
```

```

Call:
lm(formula = bmi ~ poly(fruit_day, 2), data = train_c4im)

Coefficients:
        (Intercept)  poly(fruit_day, 2)1  poly(fruit_day, 2)2
             28.084              -21.613                8.011
```

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 27.466.

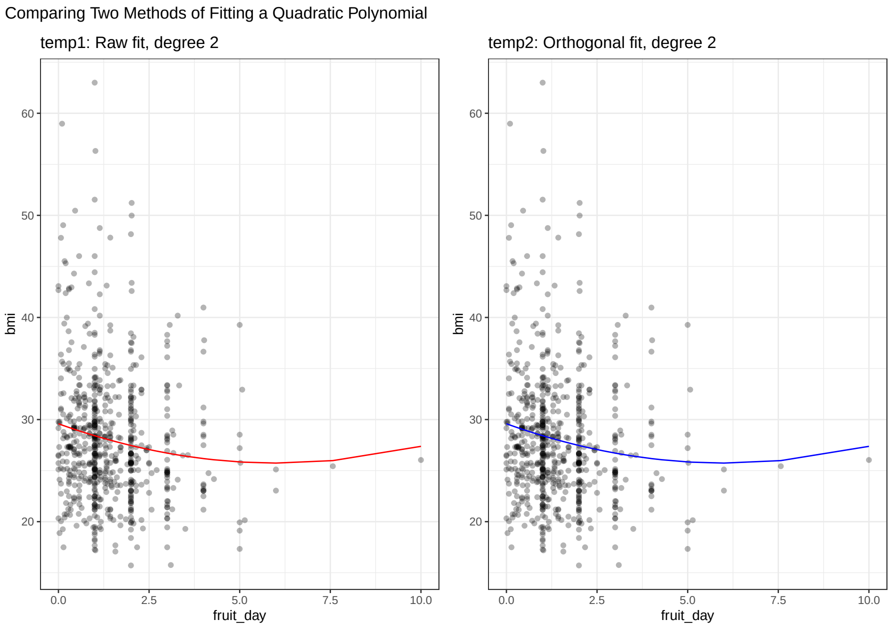
Now, what happens with the orthogonal polynomial model temp2 we just fit?

```
augment(temp2, newdata = data.frame(fruit_day = 2)) |>
    kable(digits = 3)
```

fruit_day	.fitted
2	27.462

- No change in the prediction.

Fits of raw vs orthogonal polynomials



Why do we 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…

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

```
    fruit_day I(fruit_day^2) 
     4.652178       4.652178 
```

- Orthogonal polynomial terms are uncorrelated with one another, easing the process of identifying which terms add value to our model.

```r
vif(temp2)
```

```
poly(fruit_day, 2)1 poly(fruit_day, 2)2 
                  1                   1 
```

Why orthogonal rather than raw 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 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_c4im)
```

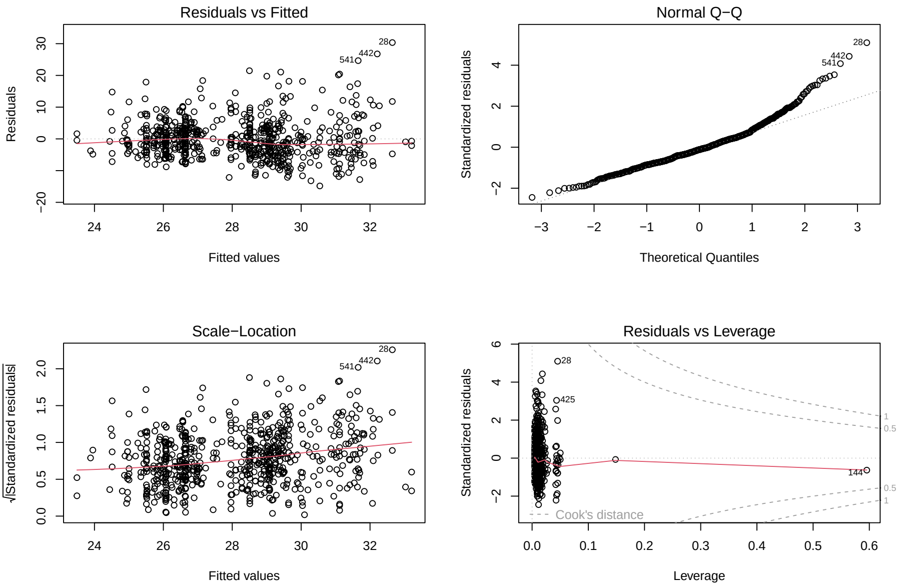
- Comparison to other models without the interaction…

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1	0.0889	0.0820	6.12	5	664	4335.8	4367.3
m_2	0.0979	0.0897	6.09	6	663	4331.1	4367.2
m_3	0.0979	0.0884	6.09	7	662	4333.1	4373.7

Tidied summary of `m_3` coefficients

term	est	se	t	p	conf.low	conf.high
(Intercept)	27.86	0.74	37.53	0.000	26.64	29.09
poly(fruit_day, 2)1	-15.86	6.16	-2.57	0.010	-26.01	-5.70
poly(fruit_day, 2)2	1.08	6.24	0.17	0.862	-9.19	11.36
exerany	-2.03	0.56	-3.64	0.000	-2.95	-1.11
healthVG	0.56	0.70	0.80	0.424	-0.59	1.71
healthG	3.00	0.72	4.16	0.000	1.81	4.19
healthF	3.55	0.91	3.92	0.000	2.06	5.05
healthP	4.53	1.36	3.32	0.001	2.29	6.78

m_3 Residual Plots



Add in the interaction

```
m_3int <- lm(bmi ~ poly(fruit_day,2) + exerany * health,
             data = train_c4im)
```

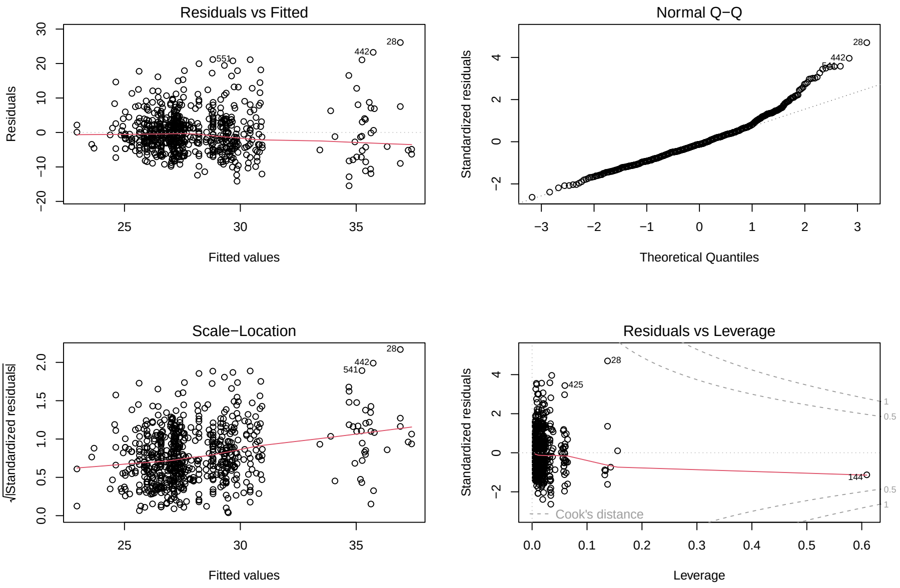
- Comparison to other models with the interaction...

mod	r.sq	adj.r.sq	sigma	df	df.res	AIC	BIC
m_1int	0.1263	0.1144	6.01	9	660	4315.7	4365.3
m_2int	0.1375	0.1244	5.97	10	659	4309.1	4363.1
m_3int	0.1376	0.1232	5.98	11	658	4311.0	4369.6

Tidied summary of `m_3int` coefficients

term	est	se	t	p	conf.low	conf.high
(Intercept)	27.40	1.41	19.43	0.000	25.07	29.72
poly(fruit_day, 2)1	-17.70	6.06	-2.92	0.004	-27.68	-7.71
poly(fruit_day, 2)2	-1.63	6.29	-0.26	0.795	-11.99	8.73
exerany	-1.47	1.55	-0.95	0.342	-4.01	1.08
healthVG	-0.66	1.63	-0.40	0.686	-3.34	2.02
healthG	2.75	1.61	1.70	0.089	0.09	5.40
healthF	7.58	1.77	4.28	0.000	4.66	10.50
healthP	9.22	2.61	3.54	0.000	4.93	13.52
exerany:healthVG	1.60	1.79	0.89	0.374	-1.36	4.55
exerany:healthG	0.43	1.80	0.24	0.809	-2.53	3.39
exerany:healthF	-6.40	2.06	-3.10	0.002	-9.80	-3.00
exerany:healthP	-6.65	3.05	-2.18	0.030	-11.68	-1.62

m_3int Residual Plots



How do models m_3 and m_3int do in testing?

```
m3_test_aug <- augment(m_3, newdata = test_c4im)
m3int_test_aug <- augment(m_3int, newdata = test_c4im)

testing_r2 <- bind_rows(
    rsq(m1_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2_test_aug, truth = bmi, estimate = .fitted),
    rsq(m3_test_aug, truth = bmi, estimate = .fitted),
    rsq(m1int_test_aug, truth = bmi, estimate = .fitted),
    rsq(m2int_test_aug, truth = bmi, estimate = .fitted),
    rsq(m3int_test_aug, truth = bmi, estimate = .fitted)) |>
    mutate(model = c("m_1", "m_2", "m_3", "m_1int",
                     "m_2int", "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	0.0716	5.729	4.428
m_2	0.0652	5.769	4.476
m_3	0.0656	5.768	4.476
m_1int	0.0397	6.025	4.624
m_2int	0.0364	6.082	4.707
m_3int	0.0357	6.090	4.707

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

Splines

- A **linear spline** is a continuous function formed by connecting points (called **knots** of the spline) by line segments.
- A **restricted cubic spline** is a way to build highly complicated curves into a regression equation in a fairly easily structured way.
- A restricted cubic spline is a series of polynomial functions joined together at the knots.
  - Such a spline gives us a way to flexibly account for non-linearity without over-fitting the model.
  - Restricted cubic splines can fit many different types of non-linearities.
  - Specifying the number of knots is all you need to do in R to get a reasonable result from a restricted cubic spline.

The most common choices are 3, 4, or 5 knots.

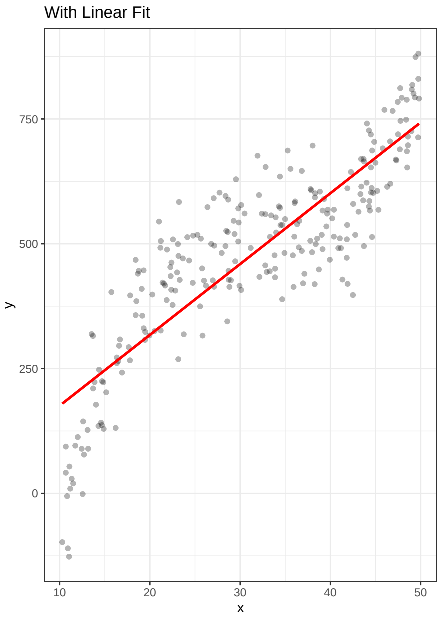
- 3 Knots, 2 degrees of freedom, allows the curve to “bend” once.
- 4 Knots, 3 degrees of freedom, lets the curve “bend” twice.
- 5 Knots, 4 degrees of freedom, lets the curve “bend” three times.

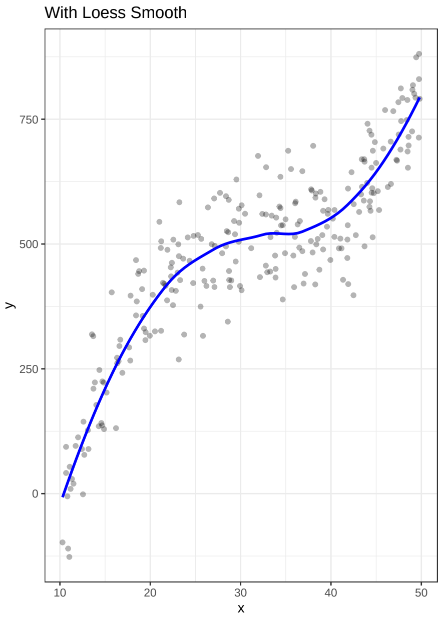
A simulated data set

```r
set.seed(4322021)

sim_data <- tibble(
    x = runif(250, min = 10, max = 50),
    y = 3*(x-30) - 0.3*(x-30)^2 + 0.05*(x-30)^3 +
        rnorm(250, mean = 500, sd = 70)
)

head(sim_data, 2)
```

```
# A tibble: 2 x 2
      x     y
  <dbl> <dbl>
1  42.5  397.
2  35.9  414.
```

The `sim_data`, plotted.



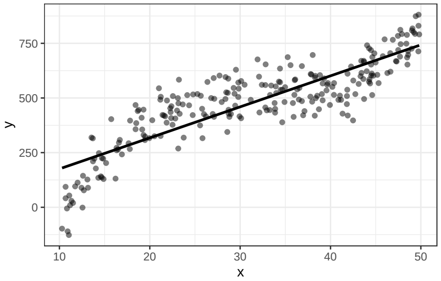


Fitting Restricted Cubic Splines with lm and rcs

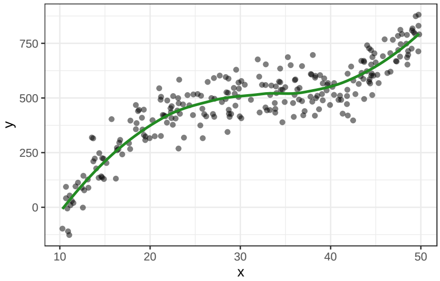
```
sim_linear <- lm(y ~ x, data = sim_data)
sim_poly2  <- lm(y ~ poly(x, 2), data = sim_data)
sim_poly3  <- lm(y ~ poly(x, 3), data = sim_data)
sim_rcs3   <- lm(y ~ rcs(x, 3), data = sim_data)
sim_rcs4   <- lm(y ~ rcs(x, 4), data = sim_data)
sim_rcs5   <- lm(y ~ rcs(x, 5), data = sim_data)
```

Looking at the Polynomial Fits

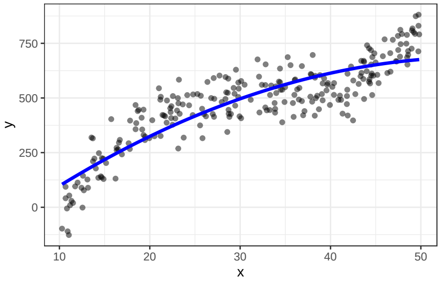
Linear Fit



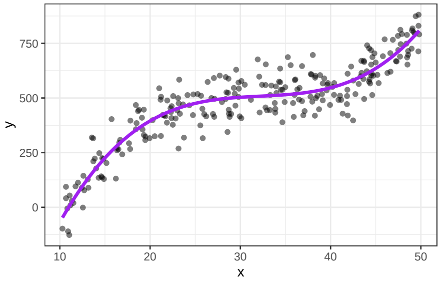
Loess Smooth



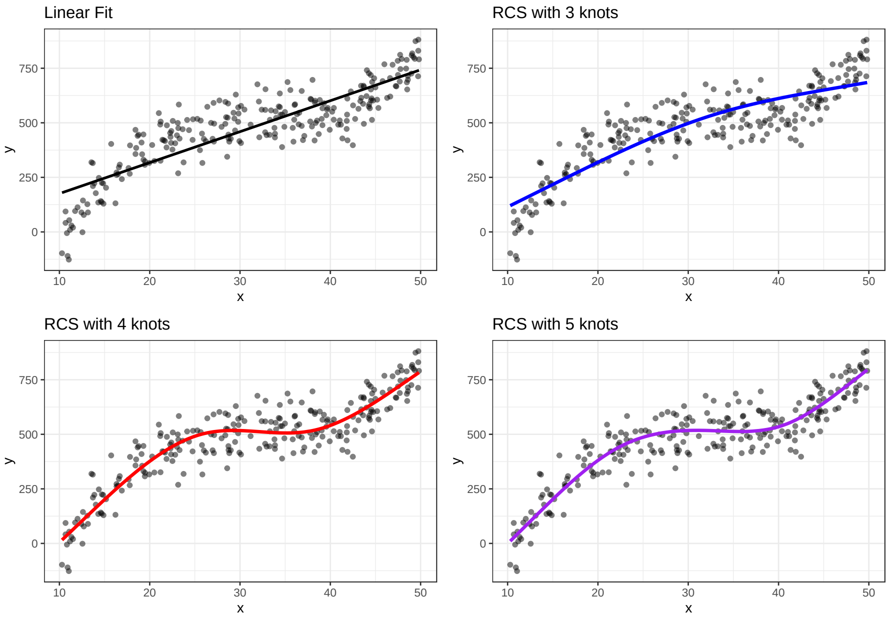
Quadratic Polynomial



Cubic Polynomial



Looking at the Restricted Cubic Spline Fits



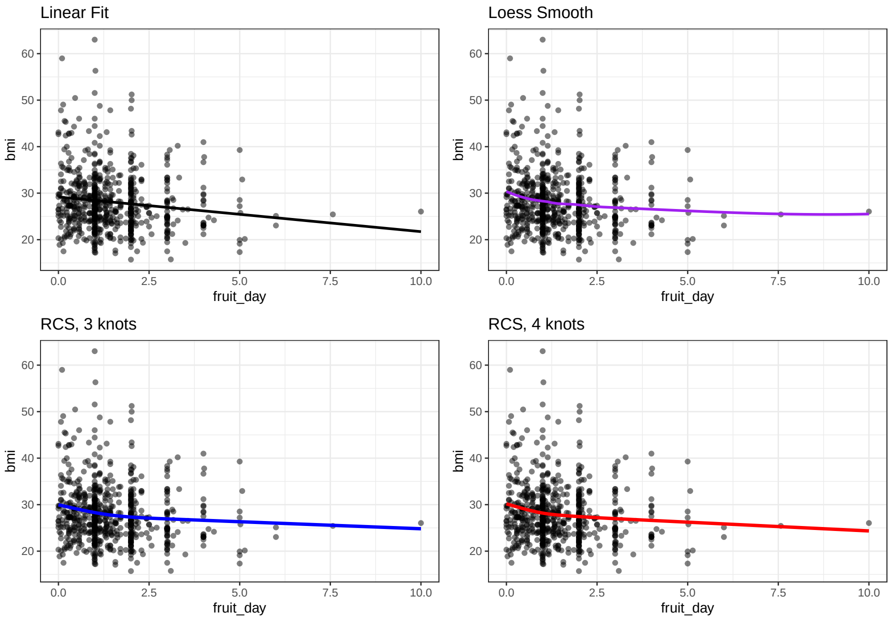
Fitting Restricted Cubic Splines with lm and rcs

For most applications, three to five knots strike a nice balance between complicating the model needlessly and fitting data pleasingly. Let's consider a restricted cubic spline model for bmi based on fruit_day again, but now with:

- in temp3, 3 knots, and
- in temp4, 4 knots,

```
temp3 <- lm(bmi ~ rcs(fruit_day, 3), data = train_c4im)
temp4 <- lm(bmi ~ rcs(fruit_day, 4), data = train_c4im)
```

Spline models for `bmi` and `fruit_day`



Let's try an RCS with 4 knots

```
m_4 <- lm(bmi ~ rcs(fruit_day, 4) + exerany + health,
          data = train_c4im)

m_4int <- lm(bmi ~ rcs(fruit_day, 4) + exerany * health,
          data = train_c4im)
```

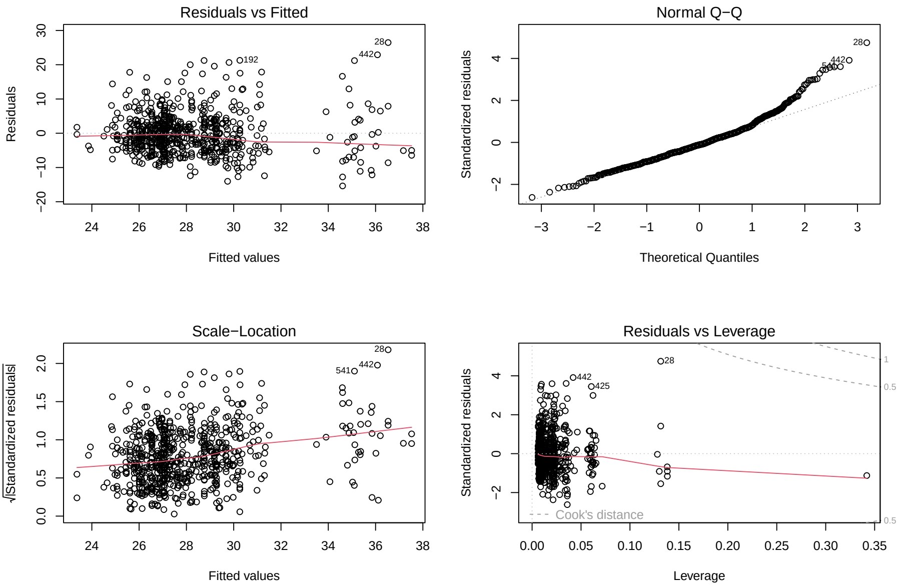
Comparing 4 models including the `exerany*health` interaction…

mod	fruit	r.sq	adj.r.sq	sigma	df	AIC	BIC
m_1int	not in	0.1263	0.1144	6.006	9	4315.7	4365.3
m_2int	linear	0.1375	0.1244	5.972	10	4309.1	4363.1
m_3int	poly(2)	0.1376	0.1232	5.977	11	4311.0	4369.6
m_4int	rcs(4)	0.1379	0.1221	5.980	12	4312.8	4375.9

Tidied summary of `m_4int` coefficients

term	est	se	t	p	lo90	hi90
(Intercept)	28.56	1.54	18.50	0.000	26.02	31.11
rcs(fruit_day, 4)fruit_day	-1.21	1.27	-0.96	0.339	-3.30	0.87
rcs(fruit_day, 4)fruit_day'	2.31	5.82	0.40	0.691	-7.28	11.91
rcs(fruit_day, 4)fruit_day''	-5.95	16.58	-0.36	0.720	-33.26	21.37
exerany	-1.36	1.55	-0.87	0.383	-3.92	1.20
healthVG	-0.64	1.63	-0.39	0.696	-3.32	2.04
healthG	2.77	1.61	1.72	0.086	0.11	5.43
healthF	7.64	1.77	4.30	0.000	4.71	10.56
healthP	9.05	2.56	3.53	0.000	4.82	13.27
exerany:healthVG	1.57	1.80	0.87	0.382	-1.39	4.53
exerany:healthG	0.37	1.80	0.21	0.836	-2.59	3.34
exerany:healthF	-6.47	2.07	-3.13	0.002	-9.87	-3.06
exerany:healthP	-6.50	3.02	-2.15	0.032	-11.47	-1.52

`m_4int` Residual Plots



How do models `m_4` and `m_4int` do in testing?

model	rsquare	rmse	mae
m_1	0.0716	5.729	4.428
m_2	0.0652	5.769	4.476
m_3	0.0656	5.768	4.476
m_4	0.0686	5.763	4.470
m_1int	0.0397	6.025	4.624
m_2int	0.0364	6.082	4.707
m_3int	0.0357	6.090	4.707
m_4int	0.0395	6.091	4.703

I'll note that there's a fair amount of very repetitive code in the Quarto file to create that table.

- What are our conclusions?

Next Week

- Using the `ols` function from the **rms** package to fit linear regression models with non-linear terms.
- Be sure to submit Lab 2 to Canvas by Monday 2023-01-30 at 9 PM.

Section 6

Appendix: How The `class4` and `class4im` data were built from the `smart_ohio.csv` data created in the Course Notes

Creating Today's Data Set

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

smart_ohio <- read_csv(url1)

class4 <- 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 `class4` 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 Course Notes Chapter on BRFSS SMART data for variable details

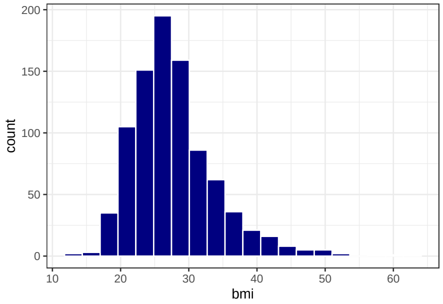
Basic Data Summaries

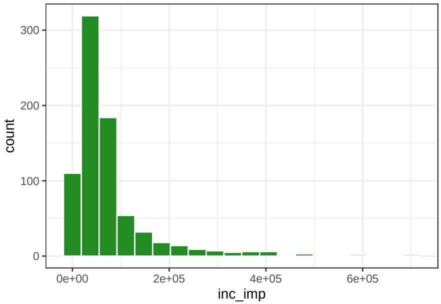
Available approaches include:

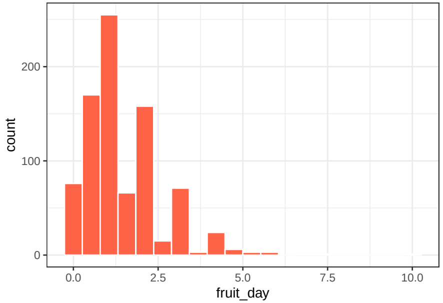
- `summary`
- `mosaic` package's `inspect()`
- `Hmisc` package's `describe`

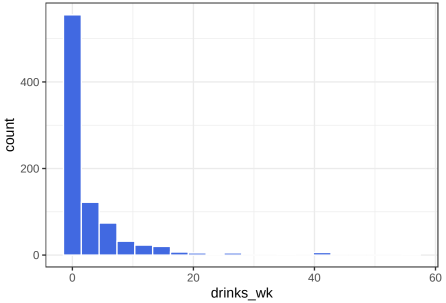
all of which can work nicely in an HTML presentation, but none of them fit well on one of these slides.

Quick Histogram of each quantitative variable









Code for previous slide

```r
p1 <- ggplot(class4, aes(x = bmi)) +
    geom_histogram(fill = "navy", col = "white", bins = 20)
p2 <- ggplot(class4, aes(x = inc_imp)) +
    geom_histogram(fill = "forestgreen", col = "white",
                   bins = 20)
p3 <- ggplot(class4, aes(x = fruit_day)) +
    geom_histogram(fill = "tomato", col = "white", bins = 20)
p4 <- ggplot(class4, 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 class4

```
class4 |> 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 `class4`

```
class4 |> 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 class4

```
class4 |> 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

```
class4 <- class4 |>
    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 class4

```r
class4 |> 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 class4

```
class4 |> 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 class4

```
class4 |> 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
class4 <- class4 |>
    mutate(race_eth = fct_infreq(race_eth))
```

```r
class4 |> 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

```
class4 <- class4 |>
    select(ID, bmi, inc_imp, fruit_day, drinks_wk,
           female, exerany, health, race_eth, everything())

miss_var_summary(class4)
```

```
# 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)
class4im <- class4 |>
    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(class4im)
```

```
[1] 0
```

Saving the tidied data

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

```r
write_rds(class4, "c04/data/class4.Rds")

write_rds(class4im, "c04/data/class4im.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.