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

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())

# 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](https://thomaselove.github.io/432-notes/06-smart.html) 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/m2. |
| 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

c4im <- read\_rds("c04/data/c4im.Rds")  
c4im |> n\_miss()

[1] 0

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 (subtracting its mean) prior to including it.

c4im <- c4im |>  
 mutate(fruit\_c = fruit\_day - mean(fruit\_day))  
  
df\_stats(~ fruit\_day + fruit\_c, data = c4im) |> gt() |>  
 fmt\_number(columns = min:sd, decimals = 3) |>  
 tab\_options(table.font.size = 20)

| response | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| fruit\_day | 0.000 | 0.710 | 1.135 | 2.000 | 10.000 | 1.438 | 1.100 | 894 | 0 |
| fruit\_c | -1.438 | -0.728 | -0.303 | 0.562 | 8.562 | 0.000 | 1.100 | 894 | 0 |

## Splitting the Sample

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

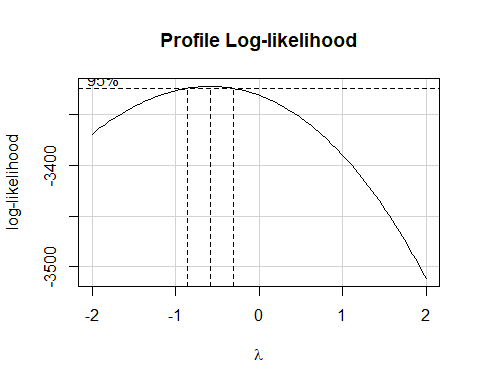
## Models We’ll Build Today

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.

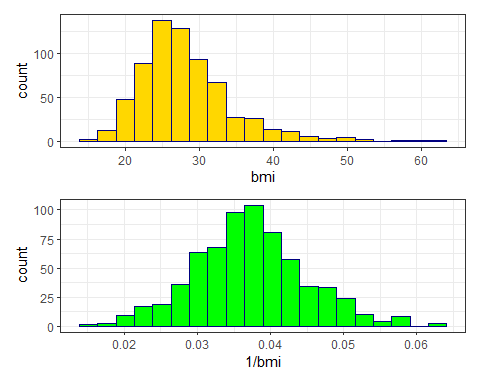
## Consider transforming bmi?

m0 <- lm(bmi ~ exerany + health, data = train\_c4im)  
boxCox(m0)



## Should we transform bmi?

p1 <- ggplot(train\_c4im, aes(x = bmi)) +   
 geom\_histogram(col = "navy", fill = "gold", bins = 20)  
  
p2 <- ggplot(train\_c4im, aes(x = 1/bmi)) +   
 geom\_histogram(col = "navy", fill = "green", bins = 20)  
  
p1 / p2



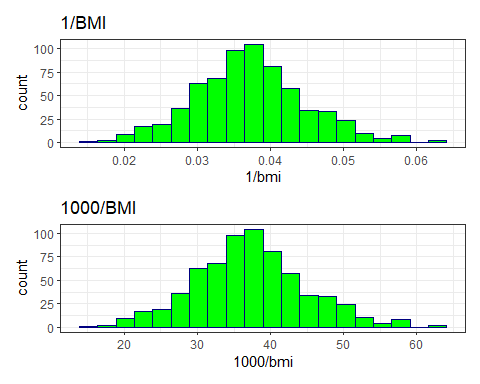
## Re-scaling the transformation

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 | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1/bmi | 0.016 | 0.032 | 0.037 | 0.042 | 0.064 | 0.037 | 0.008 | 670 | 0 |
| 1000/bmi | 15.873 | 32.248 | 36.839 | 41.806 | 63.654 | 37.240 | 7.606 | 670 | 0 |

## Shape doesn’t change

p2 <- ggplot(train\_c4im, aes(x = 1/bmi)) +   
 geom\_histogram(col = "navy", fill = "green", bins = 20) +  
 labs(title = "1/BMI")  
  
p3 <- ggplot(train\_c4im, aes(x = 1000/bmi)) +  
 geom\_histogram(col = "navy", fill = "green", bins = 20) +   
 labs(title = "1000/BMI")  
  
p2 / p3



## Means by exerany and health

summaries\_1 <- train\_c4im |>  
 group\_by(exerany, health) |>  
 summarise(n = n(), mean = mean(1000/bmi), stdev = sd(1000/bmi))

`summarise()` has grouped output by 'exerany'. You can override using the  
`.groups` argument.

summaries\_1

# A tibble: 10 × 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  
 9 1 F 49 38.2 7.63  
10 1 P 17 37.9 10.4

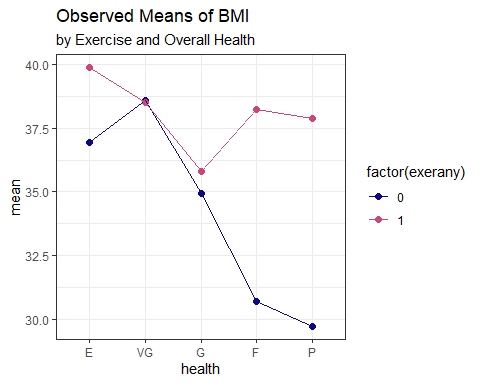
## Code for Interaction Plot

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

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")



# Fitting a Two-Way ANOVA model for 1000/BMI

## Model m1 without interaction

m1 <- lm(1000/bmi ~ exerany + health, data = train\_c4im)

* How well does this model fit the training data?

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 | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.064 | 0.057 | 7.385 | 670 | 5 | 664 | 4,588.7 | 4,620.2 |

## Tidied ANOVA for m1

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 = 3) |>  
 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 |

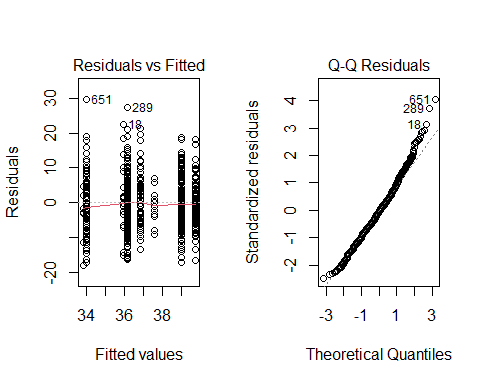
## Interpreting m1

| 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?

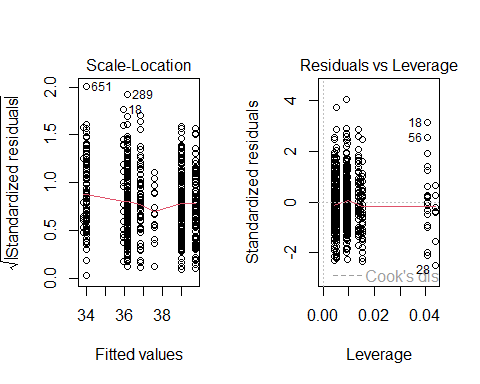
## m1 Residual Plots (*n* = 670)

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))



# 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

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

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 ' ' 1

## m1int coefficients

tidy(m1int, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 tab\_options(table.font.size = 20)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 36.950 | 1.721 | 21.472 | 0.000 | 34.115 | 39.784 |
| exerany | 2.920 | 1.882 | 1.552 | 0.121 | -0.179 | 6.020 |
| healthVG | 1.656 | 1.987 | 0.834 | 0.405 | -1.617 | 4.929 |
| healthG | -2.030 | 1.970 | -1.030 | 0.303 | -5.274 | 1.215 |
| healthF | -6.264 | 2.164 | -2.895 | 0.004 | -9.827 | -2.700 |
| healthP | -7.238 | 3.102 | -2.333 | 0.020 | -12.348 | -2.128 |
| exerany:healthVG | -2.999 | 2.192 | -1.368 | 0.172 | -6.610 | 0.612 |
| exerany:healthG | -2.033 | 2.193 | -0.927 | 0.354 | -5.646 | 1.579 |
| exerany:healthF | 4.629 | 2.520 | 1.837 | 0.067 | 0.479 | 8.779 |
| exerany:healthP | 5.256 | 3.652 | 1.439 | 0.151 | -0.760 | 11.272 |

## 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 |

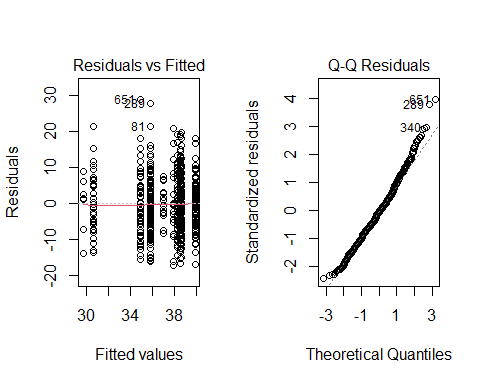
* How do we interpret effect sizes here? **It depends**.

## 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

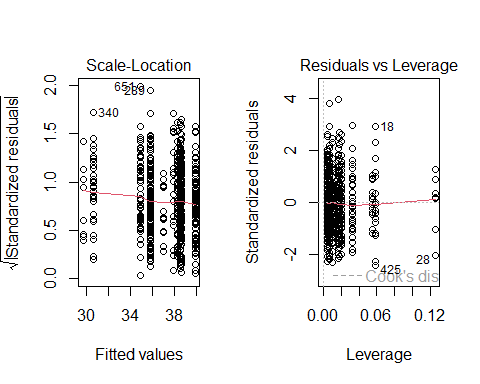
## Residuals from m1int? (*n* = 670)

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))



# Incorporating a Covariate into our two-way ANOVA models

## Add fruit\_c to m1

m2 <- lm(1000/bmi ~ fruit\_c + exerany + health, data = train\_c4im)

* 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 |

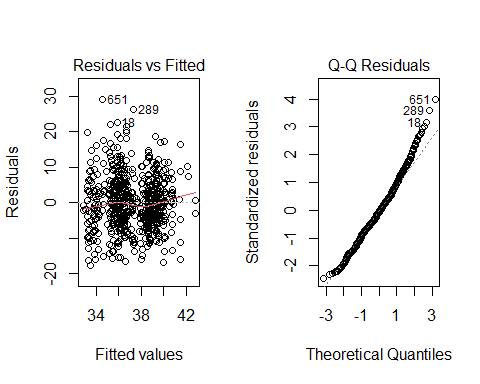
## m2 coefficients

tidy(m2, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 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 |

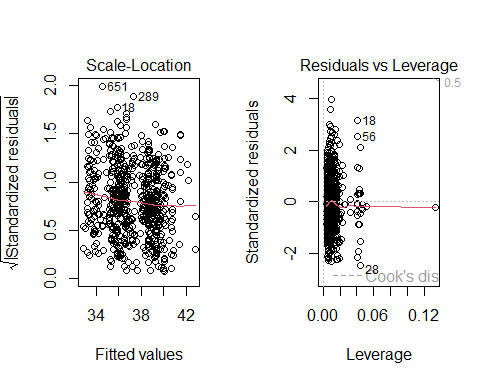
## m2 Residuals

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 = 3) |>  
 tab\_options(table.font.size = 18)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 37.064 | 1.711 | 21.665 | 0.000 | 34.246 | 39.882 |
| fruit\_c | 0.766 | 0.254 | 3.020 | 0.003 | 0.348 | 1.184 |
| exerany | 2.598 | 1.873 | 1.387 | 0.166 | -0.488 | 5.683 |
| healthVG | 1.711 | 1.975 | 0.866 | 0.387 | -1.543 | 4.964 |
| healthG | -1.920 | 1.958 | -0.981 | 0.327 | -5.146 | 1.305 |
| healthF | -6.205 | 2.150 | -2.885 | 0.004 | -9.747 | -2.662 |
| healthP | -7.734 | 3.088 | -2.505 | 0.012 | -12.821 | -2.648 |
| exerany:healthVG | -2.961 | 2.179 | -1.359 | 0.175 | -6.550 | 0.629 |
| exerany:healthG | -1.938 | 2.180 | -0.889 | 0.374 | -5.529 | 1.653 |
| exerany:healthF | 4.867 | 2.505 | 1.943 | 0.052 | 0.741 | 8.994 |
| exerany:healthP | 5.930 | 3.637 | 1.631 | 0.103 | -0.060 | 11.921 |

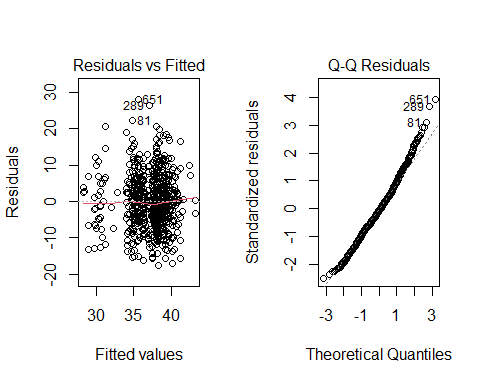
## ANOVA: Compare m2 & m2int

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

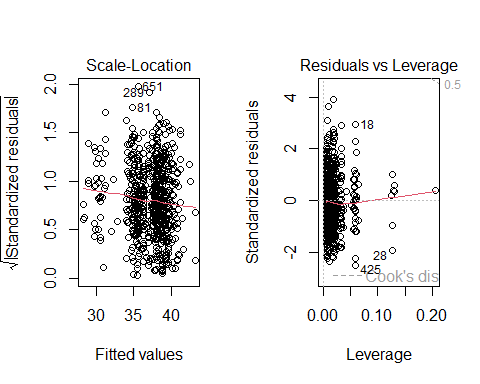
## m2int Residuals

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



## m2int Residuals

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



# Comparing Our Models

## Which of the four models fits best?

In the **training** sample, we have…

bind\_rows(glance(m1), glance(m2), glance(m1int), glance(m2int)) |>  
 mutate(mod = c("m1", "m2", "m1int", "m2int")) |>  
 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 |
| 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 , 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, exerany) |>  
 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/bmi) = bmi, so we need

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

## Adjusting augment() appropriately

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, .resid) |>  
 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)

# 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 test subjects

## Summaries from yardstick

* = square of the correlation between truth and estimate
* mae = mean absolute error …
* rmse = root mean squared error …

## Testing Results (Validated )

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

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 |
| rmse | standard | 5.888 | m2int |

## Other yardstick summaries (1)

* rsq\_trad() = defines 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](https://yardstick.tidymodels.org/index.html) for more details.

# 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:
* Quadratic:
* Cubic:
* Quartic:
* Quintic:

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

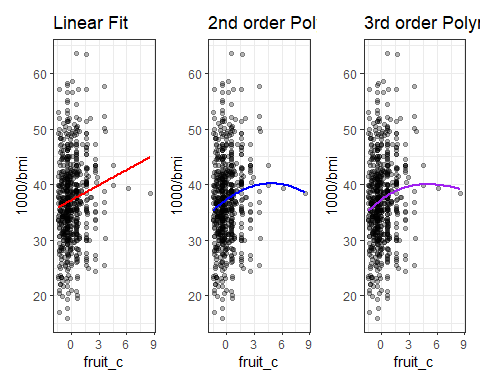
## Adding a polynomial in fruit\_c

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")  
  
p1 + p2 + p3



## Raw vs. Orthogonal Polynomials

Predict 1000/bmi using fruit\_c with a “raw polynomial of degree 2.”

(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\_c4im)  
  
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 |

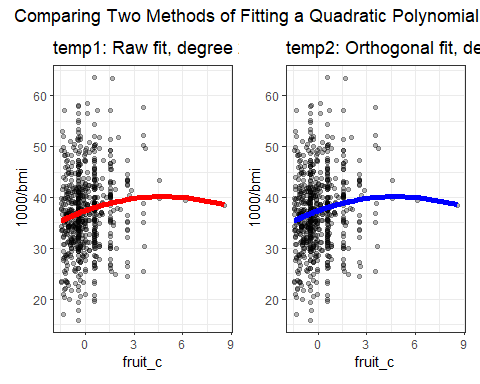
* No change in the prediction.

## 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 = 2) +  
 labs(title = "temp1: Raw fit, degree 2")

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.

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", size = 2) +  
 labs(title = "temp2: Orthogonal fit, degree 2")  
  
p1 + p2 +   
 plot\_annotation(title = "Comparing Two Methods of Fitting a Quadratic Polynomial")



* The two models are, in fact, identical.

## 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…

bind\_rows(glance(m1), glance(m2), glance(m3)) |>  
 mutate(mod = c("m1", "m2", "m3")) |>  
 select(mod, r.squared, adj.r.squared, sigma,   
 df, df.residual, nobs, AIC, BIC) |>   
 gt() |> fmt\_number(columns = r.squared:adj.r.squared, decimals = 4) |>  
 fmt\_number(columns = sigma, decimals = 3) |>  
 fmt\_number(columns = AIC:BIC, decimals = 1) |>  
 tab\_options(table.font.size = 20)

| mod | r.squared | adj.r.squared | sigma | df | df.residual | nobs | AIC | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m1 | 0.0642 | 0.0571 | 7.385 | 5 | 664 | 670 | 4,588.7 | 4,620.2 |
| m2 | 0.0747 | 0.0663 | 7.349 | 6 | 663 | 670 | 4,583.1 | 4,619.2 |
| m3 | 0.0749 | 0.0651 | 7.354 | 7 | 662 | 670 | 4,585.0 | 4,625.6 |

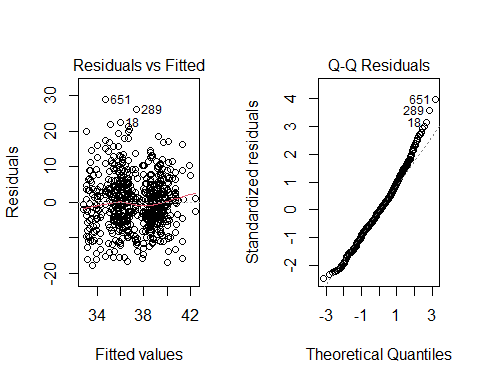
## m3 coefficients

tidy(m3, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 tab\_options(table.font.size = 20)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 37.655 | 0.896 | 42.028 | 0.000 | 36.179 | 39.131 |
| poly(fruit\_c, 2)1 | 20.411 | 7.439 | 2.744 | 0.006 | 8.158 | 32.664 |
| poly(fruit\_c, 2)2 | -2.908 | 7.529 | -0.386 | 0.699 | -15.309 | 9.493 |
| exerany | 1.915 | 0.674 | 2.840 | 0.005 | 0.804 | 3.026 |
| healthVG | -0.642 | 0.845 | -0.759 | 0.448 | -2.034 | 0.751 |
| healthG | -3.406 | 0.871 | -3.908 | 0.000 | -4.841 | -1.970 |
| healthF | -3.393 | 1.093 | -3.103 | 0.002 | -5.194 | -1.592 |
| healthP | -3.723 | 1.646 | -2.263 | 0.024 | -6.434 | -1.013 |

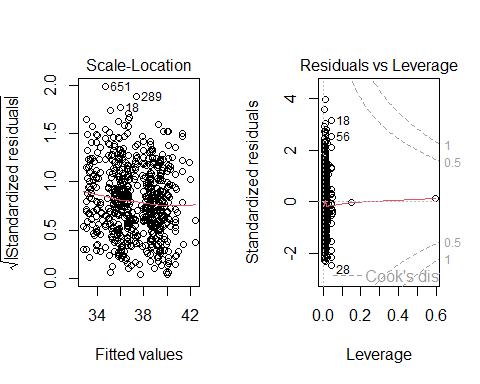
## m3 Residuals

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



## m3 Residuals

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



## Add in the interaction

m3int <- lm(1000/bmi ~ poly(fruit\_c,2) + exerany \* health,  
 data = train\_c4im)

* Comparison to other models with the interaction…

bind\_rows(glance(m1int), glance(m2int), glance(m3int)) |>  
 mutate(mod = c("m1int", "m2int", "m3int")) |>  
 select(mod, r.squared, adj.r.squared, sigma,   
 df, df.residual, nobs, AIC, BIC) |>   
 gt() |> fmt\_number(columns = r.squared:adj.r.squared, decimals = 4) |>  
 fmt\_number(columns = sigma, decimals = 3) |>  
 fmt\_number(columns = AIC:BIC, decimals = 1) |>  
 tab\_options(table.font.size = 20)

| mod | r.squared | adj.r.squared | sigma | df | df.residual | nobs | AIC | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m1int | 0.0910 | 0.0786 | 7.301 | 9 | 660 | 670 | 4,577.2 | 4,626.8 |
| m2int | 0.1034 | 0.0898 | 7.256 | 10 | 659 | 670 | 4,570.0 | 4,624.1 |
| m3int | 0.1034 | 0.0884 | 7.262 | 11 | 658 | 670 | 4,572.0 | 4,630.6 |

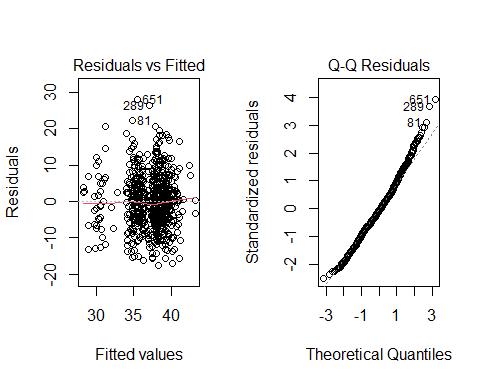
## m3int coefficients

tidy(m3int, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 tab\_options(table.font.size = 20)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 37.079 | 1.713 | 21.640 | 0.000 | 34.257 | 39.902 |
| poly(fruit\_c, 2)1 | 22.219 | 7.364 | 3.017 | 0.003 | 10.090 | 34.348 |
| poly(fruit\_c, 2)2 | 0.216 | 7.639 | 0.028 | 0.977 | -12.368 | 12.799 |
| exerany | 2.601 | 1.878 | 1.385 | 0.167 | -0.492 | 5.694 |
| healthVG | 1.711 | 1.977 | 0.866 | 0.387 | -1.545 | 4.966 |
| healthG | -1.920 | 1.960 | -0.980 | 0.328 | -5.148 | 1.308 |
| healthF | -6.203 | 2.153 | -2.882 | 0.004 | -9.749 | -2.658 |
| healthP | -7.754 | 3.168 | -2.448 | 0.015 | -12.972 | -2.536 |
| exerany:healthVG | -2.961 | 2.181 | -1.358 | 0.175 | -6.553 | 0.631 |
| exerany:healthG | -1.940 | 2.182 | -0.889 | 0.374 | -5.534 | 1.655 |
| exerany:healthF | 4.866 | 2.508 | 1.940 | 0.053 | 0.735 | 8.997 |
| exerany:healthP | 5.951 | 3.709 | 1.604 | 0.109 | -0.160 | 12.061 |

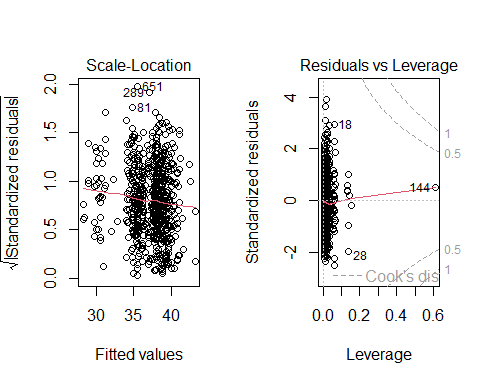
## m3int Residuals

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



## m3int Residuals

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.

testing\_rmse <- bind\_rows(  
 rmse(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m3\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m3int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(mod = c("m1", "m2", "m3", "m1int",  
 "m2int", "m3int"))  
  
testing\_mae <- bind\_rows(  
 mae(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m3\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m3int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(mod = c("m1", "m2", "m3", "m1int",  
 "m2int", "m3int"))

## 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 polynomial", "m1 + interaction", "m2 + interaction", "m3 + interaction")) |>  
 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](https://thomaselove.github.io/432-2024/lab2.html) to Canvas by Tuesday 2024-01-30 at Noon.

# Appendix

## Creating Today’s Data Set

url1 <- "https://raw.githubusercontent.com/THOMASELOVE/432-data/master/data/smart\_ohio.csv"  
  
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/m2. |
| 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

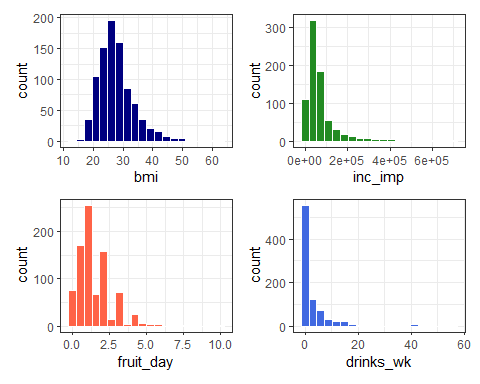
Available approaches include:

* 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

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)



## Code for previous slide

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

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 × 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…

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 × 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   
 9 bmi 0 0   
10 female 0 0   
11 hx\_diabetes 0 0   
12 mmsa 0 0   
13 SEQNO 0 0

## Single Imputation Approach?

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.