432 Class 03

https://thomaselove.github.io/432-2025/

2025-01-21

## 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 check\_model()
* Validating / evaluating results with yardstick

### Appendix

How the c3im data were created from smart\_ohio.csv

## Today’s R Setup

knitr::opts\_chunk$set(comment = NA)  
  
library(janitor)  
library(naniar)  
library(broom)  
library(car)  
library(gt)  
library(mosaic) ## for df\_stats and favstats  
library(mice) ## imputation of missing data  
library(patchwork)   
library(rms) ## regression tools (Frank Harrell)  
library(rsample) ## data splitting  
library(yardstick) ## evaluating fits  
library(easystats)  
library(tidyverse)   
  
theme\_set(theme\_lucid())

# The c3im data

## The c3im 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

c3im <- read\_rds("c03/data/c3im.Rds")  
c3im

# A tibble: 894 × 9  
 ID bmi inc\_imp fruit\_day drinks\_wk female exerany health race\_eth   
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <fct>   
 1 2 23.0 86865 4 0 1 0 E White non-Hisp…  
 2 3 26.9 138916 3 0 1 1 G Other race non…  
 3 4 26.5 57333 2 4.67 1 1 G White non-Hisp…  
 4 5 24.2 58311 0.57 0.93 0 1 G White non-Hisp…  
 5 7 23.0 2318 2 2 0 1 G White non-Hisp…  
 6 8 28.4 79667 1 0 0 1 VG Other race non…  
 7 9 30.1 47880 0.23 0 0 1 F Black non-Hisp…  
 8 10 19.8 100136 0.77 0.47 1 1 E White non-Hisp…  
 9 11 27.2 73145 0.71 0 0 1 E White non-Hisp…  
10 12 24.6 76917 1.07 0 1 1 E Other race non…  
# ℹ 884 more rows

c3im |> n\_miss()

[1] 0

identical(nrow(c3im), n\_distinct(c3im$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.

c3im <- c3im |>  
 mutate(fruit\_c = fruit\_day - mean(fruit\_day))  
  
df\_stats(~ fruit\_day + fruit\_c, data = c3im) |> 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 | 1.070 | 2.000 | 10.000 | 1.439 | 1.156 | 894 | 0 |
| fruit\_c | -1.439 | -0.729 | -0.369 | 0.561 | 8.561 | 0.000 | 1.156 | 894 | 0 |

## Splitting the Sample

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

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

set.seed(432) ## for future replication  
  
c3im\_split <- initial\_split(c3im, prop = 3/4)  
  
train\_c3im <- training(c3im\_split)  
test\_c3im <- testing(c3im\_split)  
  
c(nrow(c3im), nrow(train\_c3im), nrow(test\_c3im))

[1] 894 670 224

# Building Models

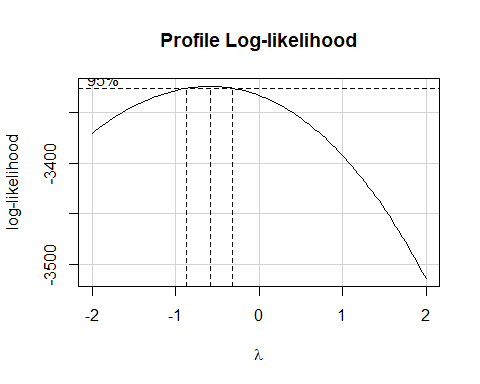
## Models We’ll Build Today

1. Predict bmi using exer\_any and genhealth (both categorical)
   * without (and 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 first in-sample (training) fit and then out-of-sample (testing) performance.

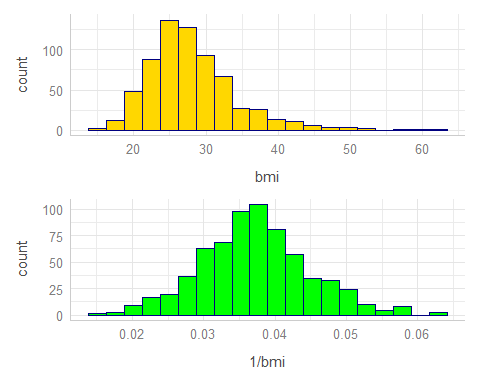
## Consider transforming bmi?

m0 <- lm(bmi ~ exerany + health, data = train\_c3im)  
boxCox(m0)



## Should we transform bmi?

p1 <- ggplot(train\_c3im, aes(x = bmi)) +   
 geom\_histogram(col = "navy", fill = "gold", bins = 20)  
  
p2 <- ggplot(train\_c3im, aes(x = 1/bmi)) +   
 geom\_histogram(col = "navy", fill = "green", bins = 20)  
  
p1 / p2



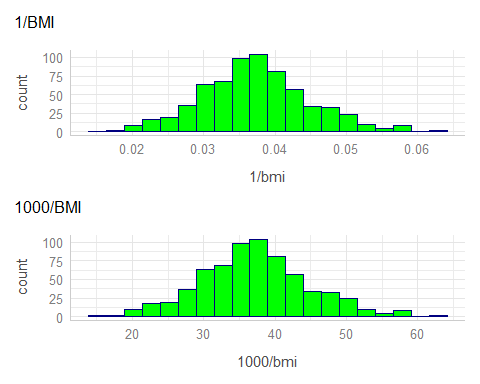
## Re-scaling the transformation

bind\_rows( favstats(~ 1/bmi, data = train\_c3im),  
 favstats(~ 1000/bmi, data = train\_c3im)) |>  
 mutate(outcome = c("1/bmi", "1000/bmi")) |>   
 relocate(outcome) |>  
 gt() |> fmt\_number(columns = min:sd, decimals = 3) |>   
 tab\_options(table.font.size = 24)

| outcome | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1/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\_c3im, aes(x = 1/bmi)) +   
 geom\_histogram(col = "navy", fill = "green", bins = 20) +  
 labs(title = "1/BMI")  
  
p3 <- ggplot(train\_c3im, 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\_c3im |>  
 group\_by(exerany, health) |>  
 summarise(n = n(), mean = mean(1000/bmi), stdev = sd(1000/bmi))

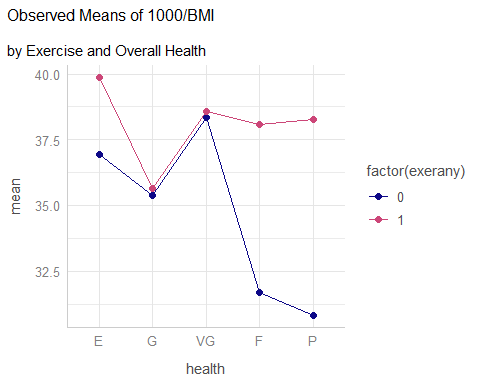
`summarise()` has grouped output by 'exerany'. You can override using the  
`.groups` argument.

summaries\_1

# A tibble: 10 × 5  
# Groups: exerany [2]  
 exerany health n mean stdev  
 <dbl> <fct> <int> <dbl> <dbl>  
 1 0 E 18 36.9 4.70  
 2 0 G 62 35.4 8.47  
 3 0 VG 54 38.3 7.61  
 4 0 F 34 31.7 8.87  
 5 0 P 10 30.8 6.90  
 6 1 E 92 39.9 6.50  
 7 1 G 148 35.6 7.19  
 8 1 VG 190 38.6 6.84  
 9 1 F 47 38.1 7.65  
10 1 P 15 38.3 11.1

## 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 (see next slide) that results from this code?

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

## Model m1 without interaction

m1 <- lm(1000/bmi ~ exerany + health, data = train\_c3im)

Using the tidy() function from broom:

tidy(m1, conf.int = TRUE, conf.level = 0.90) |>   
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 tab\_options(table.font.size = 24)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 37.841 | 0.895 | 42.278 | 0.000 | 36.367 | 39.316 |
| exerany | 1.854 | 0.658 | 2.817 | 0.005 | 0.770 | 2.938 |
| healthG | -3.586 | 0.876 | -4.096 | 0.000 | -5.028 | -2.144 |
| healthVG | -0.755 | 0.851 | -0.888 | 0.375 | -2.157 | 0.646 |
| healthF | -3.522 | 1.097 | -3.211 | 0.001 | -5.329 | -1.716 |
| healthP | -3.682 | 1.648 | -2.235 | 0.026 | -6.396 | -0.968 |

## Model Parameters for m1

model\_parameters(m1, ci = 0.90)

Parameter | Coefficient | SE | 90% CI | t(664) | p  
-------------------------------------------------------------------  
(Intercept) | 37.84 | 0.90 | [36.37, 39.32] | 42.28 | < .001  
exerany | 1.85 | 0.66 | [ 0.77, 2.94] | 2.82 | 0.005   
health [G] | -3.59 | 0.88 | [-5.03, -2.14] | -4.10 | < .001  
health [VG] | -0.76 | 0.85 | [-2.16, 0.65] | -0.89 | 0.375   
health [F] | -3.52 | 1.10 | [-5.33, -1.72] | -3.21 | 0.001   
health [P] | -3.68 | 1.65 | [-6.40, -0.97] | -2.23 | 0.026

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

## Model Parameters for m1 (with gt())

Reformatting with gt()…

model\_parameters(m1, ci = 0.90) |>   
 gt() |> fmt\_number(columns = -c(CI, df\_error), decimals = 3) |>  
 tab\_options(table.font.size = 20)

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 37.841 | 0.895 | 0.9 | 36.367 | 39.316 | 42.278 | 664 | 0.000 |
| exerany | 1.854 | 0.658 | 0.9 | 0.770 | 2.938 | 2.817 | 664 | 0.005 |
| healthG | -3.586 | 0.876 | 0.9 | -5.028 | -2.144 | -4.096 | 664 | 0.000 |
| healthVG | -0.755 | 0.851 | 0.9 | -2.157 | 0.646 | -0.888 | 664 | 0.375 |
| healthF | -3.522 | 1.097 | 0.9 | -5.329 | -1.716 | -3.211 | 664 | 0.001 |
| healthP | -3.682 | 1.648 | 0.9 | -6.396 | -0.968 | -2.235 | 664 | 0.026 |

## How well does m1 fit the training data?

model\_performance(m1)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
------------------------------------------------------------------  
4591.820 | 4591.990 | 4623.371 | 0.060 | 0.053 | 7.369 | 7.403

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 = 24)

| r.squared | adj.r.squared | sigma | nobs | df | df.residual | AIC | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.060 | 0.053 | 7.403 | 670 | 5 | 664 | 4,591.8 | 4,623.4 |

## 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 | 748.17 | 748.17 | 13.65 | 0.0002 |
| health | 4 | 1,565.65 | 391.41 | 7.14 | 0.0000 |
| Residuals | 664 | 36,387.06 | 54.80 | NA | NA |

## Interpreting m1

| Name | exerany | health | predicted 1000/bmi |
| --- | --- | --- | --- |
| Harry | 0 | Excellent | 37.84 |
| Sally | 1 | Excellent | 37.84 + 1.85 = 39.69 |
| Billy | 0 | Fair | 37.84 - 3.52 = 34.32 |
| Meg | 1 | Fair | 37.84 + 1.85 - 3.52 = 36.17 |

* Effect of exerany?
* Effect of health = Fair instead of Excellent?

## Model Checks

We’ll be checking assumptions related to:

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

## A Note about my approach

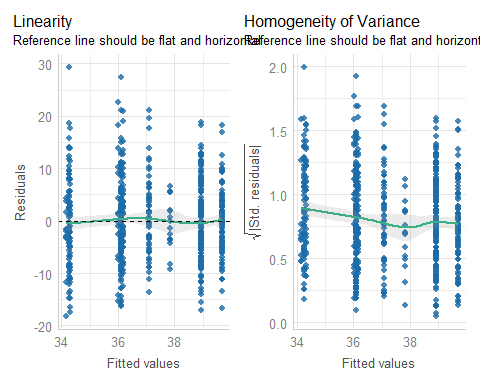
In your work, you’d just use:

check\_model(m1, detrend = FALSE)

with #| fig-height: 9 at the start of the code chunk so that the plots are easier to read, but I need to do more here so the slides look nice…

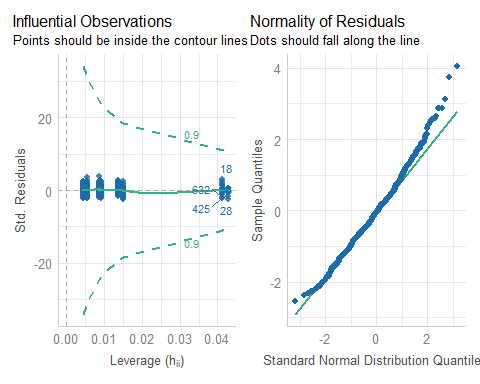
## Checking model m1 (*n* = 670)

check\_model(m1, check = c("linearity", "homogeneity"))



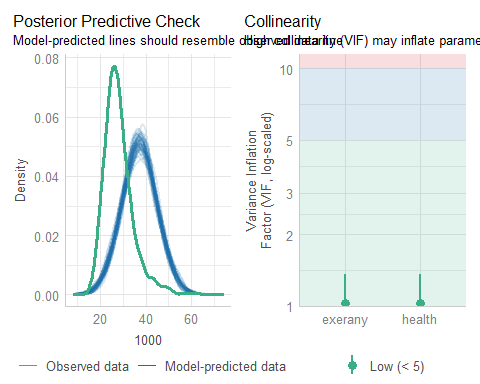
## Checking model m1 (*n* = 670)

check\_model(m1, check = c("outliers", "qq"), detrend = FALSE)



## Checking model m1 (*n* = 670)

check\_model(m1, check = c("pp\_check", "vif"))



# Fitting ANOVA model m1int including interaction

## Adding the interaction term to m1

m1int <- lm(1000/bmi ~ exerany \* health, data = train\_c3im)

* 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 | 0.060 | 0.053 | 7.403 | 670 | 5 | 664 | 4,591.8 | 4,623.4 |
| m1int | 0.081 | 0.069 | 7.340 | 670 | 9 | 660 | 4,584.4 | 4,634.0 |

## 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 | 748.17 | 748.17 | 13.89 | 0.0002 |
| health | 4 | 1,565.65 | 391.41 | 7.27 | 0.0000 |
| exerany:health | 4 | 828.80 | 207.20 | 3.85 | 0.0042 |
| Residuals | 660 | 35,558.26 | 53.88 | 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 36387   
2 660 35558 4 828.8 3.8459 0.004247 \*\*  
---  
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 | 1.730 | 21.357 | 0.000 | 34.100 | 39.799 |
| exerany | 2.920 | 1.892 | 1.544 | 0.123 | -0.196 | 6.036 |
| healthG | -1.559 | 1.965 | -0.793 | 0.428 | -4.796 | 1.678 |
| healthVG | 1.398 | 1.998 | 0.700 | 0.484 | -1.893 | 4.688 |
| healthF | -5.243 | 2.140 | -2.450 | 0.015 | -8.767 | -1.719 |
| healthP | -6.150 | 2.895 | -2.124 | 0.034 | -10.918 | -1.381 |
| exerany:healthG | -2.677 | 2.194 | -1.220 | 0.223 | -6.290 | 0.936 |
| exerany:healthVG | -2.686 | 2.205 | -1.218 | 0.224 | -6.317 | 0.945 |
| exerany:healthF | 3.436 | 2.512 | 1.368 | 0.172 | -0.702 | 7.573 |
| exerany:healthP | 4.533 | 3.544 | 1.279 | 0.201 | -1.304 | 10.370 |

## 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 - 5.24 = 31.71 |
| Meg | 1 | Fair | 36.95 + 2.92 - 5.24 + 3.44 = 38.07 |

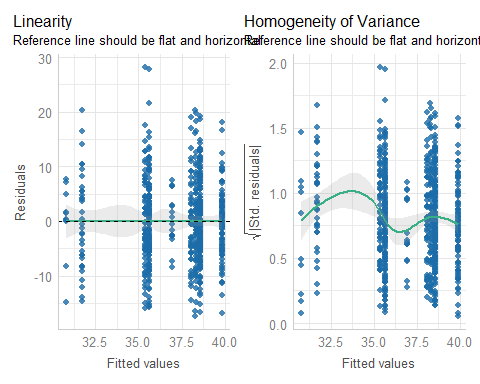
* How do we interpret effect sizes here? **It depends**.

## 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 + 3.44) = +6.36
* Effect of health = Fair instead of Excellent?
  + If exerany = 0 (no), effect is -5.24
  + If exerany = 1 (yes), effect is (-5.24 + 3.44) = -1.80

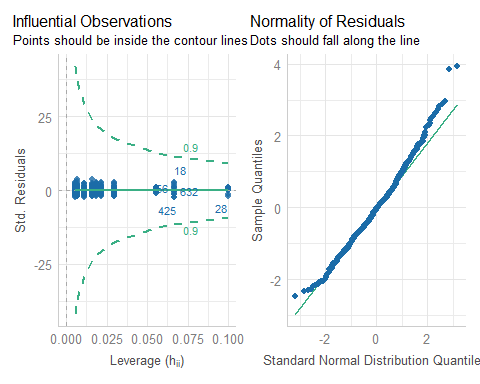
## Checking model m1int (*n* = 670)

check\_model(m1int, check = c("linearity", "homogeneity"))



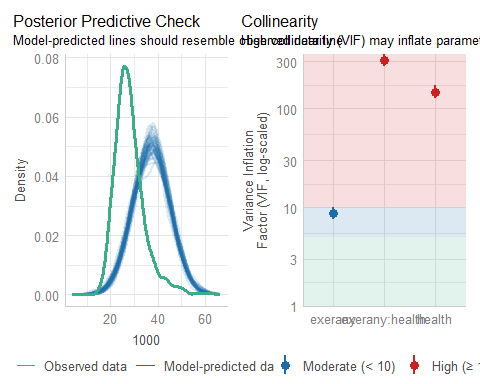
## Checking model m1int (*n* = 670)

check\_model(m1int, check = c("outliers", "qq"), detrend = FALSE)



## Checking model m1int (*n* = 670)

check\_model(m1int, check = c("pp\_check", "vif"))



# Incorporating a Covariate into our two-way ANOVA models

## Add fruit\_c to m1

m2 <- lm(1000/bmi ~ fruit\_c + exerany + health, data = train\_c3im)

* 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 | 0.060 | 0.053 | 7.403 | 5 | 664 | 4,591.8 | 4,623.4 |
| m2 | 0.069 | 0.060 | 7.372 | 6 | 663 | 4,587.3 | 4,623.4 |

## 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 | 632.07 | 632.07 | 11.63 | 0.0007 |
| exerany | 1 | 595.84 | 595.84 | 10.96 | 0.0010 |
| health | 4 | 1,437.22 | 359.31 | 6.61 | 0.0000 |
| Residuals | 663 | 36,035.75 | 54.35 | NA | NA |

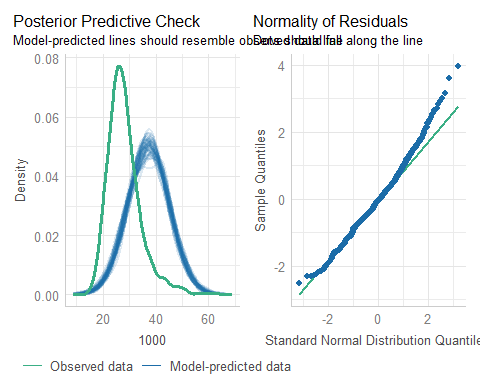
## m2 coefficients

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) | 37.851 | 0.891 | 42.461 | 0.000 | 36.382 | 39.319 |
| fruit\_c | 0.644 | 0.253 | 2.542 | 0.011 | 0.227 | 1.061 |
| exerany | 1.668 | 0.659 | 2.529 | 0.012 | 0.582 | 2.754 |
| healthG | -3.401 | 0.875 | -3.886 | 0.000 | -4.842 | -1.959 |
| healthVG | -0.665 | 0.848 | -0.784 | 0.433 | -2.062 | 0.732 |
| healthF | -3.311 | 1.096 | -3.022 | 0.003 | -5.116 | -1.506 |
| healthP | -3.690 | 1.641 | -2.249 | 0.025 | -6.393 | -0.987 |

## Checking model m2 (*n* = 670)

check\_model(m2, detrend = FALSE, check = c("pp\_check", "qq"))



## Include the interaction term?

m2int <- lm(1000/bmi ~ fruit\_c + exerany \* health,   
 data = train\_c3im)

### 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 | 632.07 | 632.07 | 11.85 | 0.0006 |
| exerany | 1 | 595.84 | 595.84 | 11.17 | 0.0009 |
| health | 4 | 1,437.22 | 359.31 | 6.74 | 0.0000 |
| exerany:health | 4 | 881.14 | 220.28 | 4.13 | 0.0026 |
| Residuals | 659 | 35,154.62 | 53.35 | 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) | 37.053 | 1.722 | 21.518 | 0.000 | 34.217 | 39.890 |
| fruit\_c | 0.692 | 0.252 | 2.751 | 0.006 | 0.278 | 1.107 |
| exerany | 2.608 | 1.886 | 1.383 | 0.167 | -0.498 | 5.714 |
| healthG | -1.446 | 1.956 | -0.739 | 0.460 | -4.668 | 1.775 |
| healthVG | 1.444 | 1.988 | 0.726 | 0.468 | -1.830 | 4.718 |
| healthF | -5.154 | 2.129 | -2.420 | 0.016 | -8.661 | -1.647 |
| healthP | -6.552 | 2.884 | -2.272 | 0.023 | -11.303 | -1.801 |
| exerany:healthG | -2.575 | 2.183 | -1.180 | 0.239 | -6.171 | 1.021 |
| exerany:healthVG | -2.629 | 2.194 | -1.198 | 0.231 | -6.242 | 0.985 |
| exerany:healthF | 3.624 | 2.500 | 1.449 | 0.148 | -0.495 | 7.743 |
| exerany:healthP | 5.145 | 3.533 | 1.456 | 0.146 | -0.675 | 10.964 |

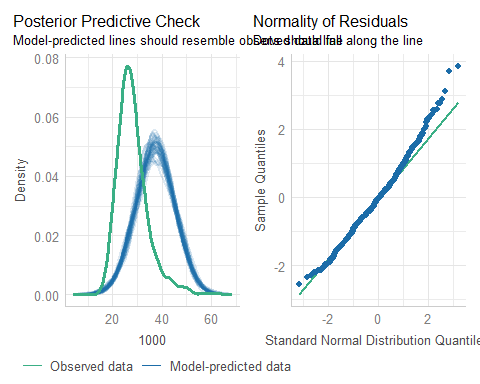
## ANOVA: Compare m2 & m2int

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 36036   
2 659 35155 4 881.14 4.1294 0.002597 \*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Checking model m2 (*n* = 670)

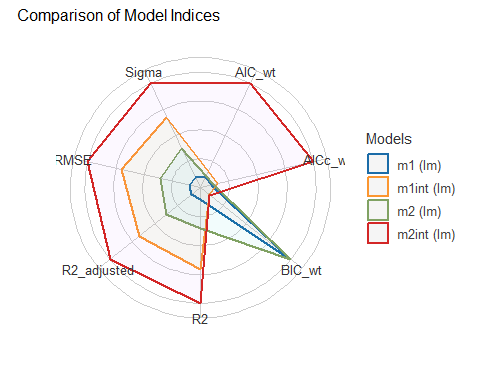
check\_model(m2int, detrend = FALSE, check = c("pp\_check", "qq"))



# Comparing Our Models

## Compare In-Sample Performance

plot(compare\_performance(m1, m1int, m2, m2int))



## 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 | 0.060 | 0.053 | 7.403 | 5 | 664 | 4,591.8 | 4,623.4 |
| m2 | 0.069 | 0.060 | 7.372 | 6 | 663 | 4,587.3 | 4,623.4 |
| m1int | 0.081 | 0.069 | 7.340 | 9 | 660 | 4,584.4 | 4,634.0 |
| m2int | 0.092 | 0.078 | 7.304 | 10 | 659 | 4,578.7 | 4,632.8 |

* Adjusted , and AIC all improve as we move down from m1 towards m2\_int. BIC likes m1 and m2.
* 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\_c3im) |>   
 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 | 34.16 | 1.00 | P | 0 |
| 1018 | 26.68 | 37.48 | 39.70 | -2.21 | E | 1 |
| 1019 | 25.74 | 38.85 | 34.32 | 4.53 | F | 0 |
| 1020 | 20.57 | 48.61 | 38.94 | 9.67 | VG | 1 |
| 1024 | 24.52 | 40.78 | 34.26 | 6.53 | 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\_c3im) |>   
 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 | 29.27 | -0.83 | P | 0 | 34.16 | 1.00 |
| 1018 | 26.68 | 25.19 | 1.49 | E | 1 | 39.70 | -2.21 |
| 1019 | 25.74 | 29.14 | -3.40 | F | 0 | 34.32 | 4.53 |
| 1020 | 20.57 | 25.68 | -5.11 | VG | 1 | 38.94 | 9.67 |
| 1024 | 24.52 | 29.19 | -4.67 | G | 0 | 34.26 | 6.53 |

## Augment all four models so far…

m1\_test\_aug <- augment(m1, newdata = test\_c3im) |>  
 mutate(bmi\_fit = 1000/.fitted, bmi\_res = bmi - bmi\_fit)  
  
m1int\_test\_aug <- augment(m1int, newdata = test\_c3im) |>  
 mutate(bmi\_fit = 1000/.fitted, bmi\_res = bmi - bmi\_fit)  
  
m2\_test\_aug <- augment(m2, newdata = test\_c3im) |>  
 mutate(bmi\_fit = 1000/.fitted, bmi\_res = bmi - bmi\_fit)  
  
m2int\_test\_aug <- augment(m2int, newdata = test\_c3im) |>  
 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

* = squared correlation of 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(  
 yardstick::rsq(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::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 | 0.075 | m1 |
| rsq | standard | 0.035 | m1int |
| rsq | standard | 0.070 | m2 |
| rsq | standard | 0.033 | m2int |

## Mean Absolute Error?

Consider the mean absolute prediction error …

testing\_mae <- bind\_rows(  
 yardstick::mae(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(model = c("m1", "m1int", "m2", "m2int"))  
testing\_mae |>   
 gt() |> fmt\_number(.estimate, decimals = 4) |>  
 tab\_options(table.font.size = 20)

| .metric | .estimator | .estimate | model |
| --- | --- | --- | --- |
| mae | standard | 4.3019 | m1 |
| mae | standard | 4.4593 | m1int |
| mae | standard | 4.3022 | m2 |
| mae | standard | 4.4588 | m2int |

## Root Mean Squared Error?

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

testing\_rmse <- bind\_rows(  
 yardstick::rmse(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::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 | 5.637 | m1 |
| rmse | standard | 5.809 | m1int |
| rmse | standard | 5.647 | m2 |
| rmse | standard | 5.835 | 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.

* Linear:
* Quadratic:
* Cubic:
* Quartic:

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

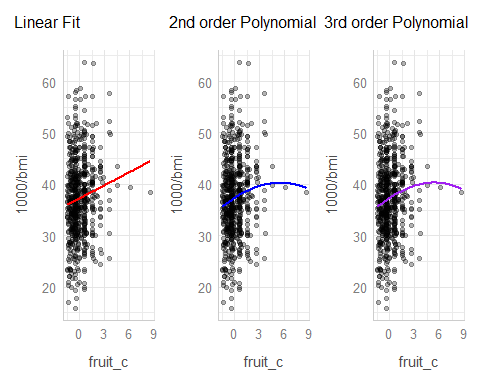
## Adding a polynomial in fruit\_c

Can we predict 1000/bmi with a polynomial in fruit\_c?

lm(1000/bmi ~ fruit\_c, data = train\_c3im)  
lm(1000/bmi ~ poly(fruit\_c, 2), data = train\_c3im)  
lm(1000/bmi ~ poly(fruit\_c, 3), data = train\_c3im)

## Plotting the Polynomials

p1 <- ggplot(train\_c3im, 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\_c3im, 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\_c3im, 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\_c3im)  
temp1$coefficients

(Intercept) fruit\_c I(fruit\_c^2)   
 37.34842948 1.05552458 -0.09700152

Predicted 1000/bmi for fruit\_c = 0.5 is

1000/bmi = 37.34842948 + 1.05552458 (fruit\_c) - 0.09700152 (fruit\_c^2)  
 = 37.34842948 + 1.05552458 (0.5) - 0.09700152 (0.25)  
 = 37.85194

## Does the raw polynomial match our expectations?

temp1 <- lm(1000/bmi ~ fruit\_c + I(fruit\_c^2),   
 data = train\_c3im)  
  
augment(temp1, newdata = tibble(fruit\_c = 0.5)) |>   
 gt() |> tab\_options(table.font.size = 20)

| fruit\_c | .fitted |
| --- | --- |
| 0.5 | 37.85194 |

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\_c3im))

Call:  
lm(formula = 1000/bmi ~ poly(fruit\_c, 2), data = train\_c3im)  
  
Coefficients:  
 (Intercept) poly(fruit\_c, 2)1 poly(fruit\_c, 2)2   
 37.240 25.141 -7.429

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

## Orthogonal Polynomial Model Fit

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 37.85194.

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 | 37.85194 |

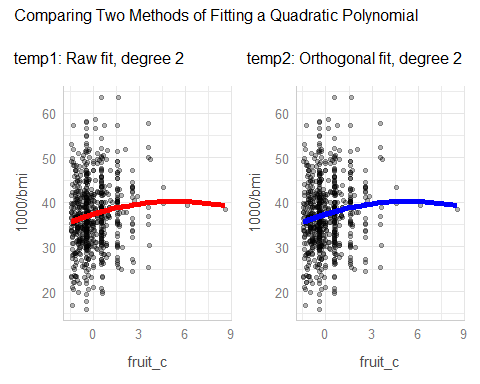
* No change in the prediction.

## Fits of raw vs orthogonal polynomials

temp1\_aug <- augment(temp1, train\_c3im)  
temp2\_aug <- augment(temp2, train\_c3im)  
  
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)   
 1.647484 1.647484

* 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.

|  |
| --- |
| Note |
| 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 later. |

## Adding a Second Order Polynomial

m3 <- lm(1000/bmi ~ poly(fruit\_c,2) + exerany + health,  
 data = train\_c3im)

model\_parameters(m3, ci = 0.9)

Parameter | Coefficient | SE | 90% CI | t(662) | p  
-----------------------------------------------------------------------------  
(Intercept) | 37.88 | 0.89 | [ 36.40, 39.35] | 42.35 | < .001  
fruit c [1st degree] | 19.02 | 7.48 | [ 6.71, 31.33] | 2.54 | 0.011   
fruit c [2nd degree] | -1.76 | 7.53 | [-14.17, 10.65] | -0.23 | 0.816   
exerany | 1.65 | 0.67 | [ 0.55, 2.74] | 2.47 | 0.014   
health [G] | -3.40 | 0.88 | [ -4.84, -1.95] | -3.88 | < .001  
health [VG] | -0.67 | 0.85 | [ -2.06, 0.73] | -0.78 | 0.433   
health [F] | -3.31 | 1.10 | [ -5.12, -1.51] | -3.02 | 0.003   
health [P] | -3.64 | 1.65 | [ -6.37, -0.92] | -2.20 | 0.028

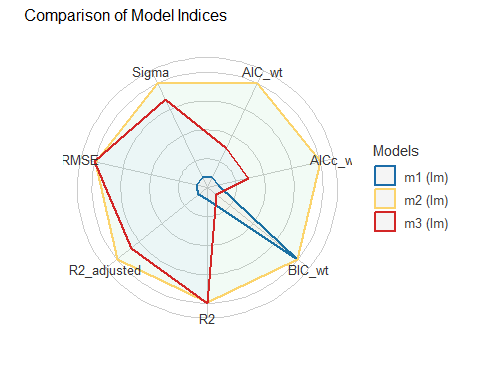
Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

## m3 vs. m1 and m2

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)

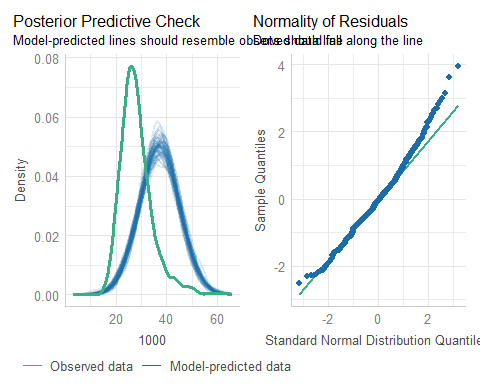
| mod | r.squared | adj.r.squared | sigma | df | df.residual | nobs | AIC | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m1 | 0.0598 | 0.0527 | 7.403 | 5 | 664 | 670 | 4,591.8 | 4,623.4 |
| m2 | 0.0689 | 0.0604 | 7.372 | 6 | 663 | 670 | 4,587.3 | 4,623.4 |
| m3 | 0.0689 | 0.0591 | 7.378 | 7 | 662 | 670 | 4,589.3 | 4,629.8 |

plot(compare\_performance(m1, m2, m3))



## m3 model check

check\_model(m3, detrend = FALSE, check = c("pp\_check", "qq"))



## Add in the interaction

m3int <- lm(1000/bmi ~ poly(fruit\_c,2) + exerany \* health,  
 data = train\_c3im)  
  
model\_parameters(m3int, ci = 0.90)

Parameter | Coefficient | SE | 90% CI | t(658) | p  
------------------------------------------------------------------------------  
(Intercept) | 37.06 | 1.72 | [ 34.22, 39.90] | 21.49 | < .001  
fruit c [1st degree] | 20.40 | 7.42 | [ 8.17, 32.63] | 2.75 | 0.006   
fruit c [2nd degree] | 0.82 | 7.59 | [-11.68, 13.32] | 0.11 | 0.914   
exerany | 2.62 | 1.89 | [ -0.49, 5.73] | 1.39 | 0.166   
health [G] | -1.44 | 1.96 | [ -4.67, 1.78] | -0.74 | 0.461   
health [VG] | 1.44 | 1.99 | [ -1.83, 4.72] | 0.73 | 0.468   
health [F] | -5.15 | 2.13 | [ -8.66, -1.64] | -2.42 | 0.016   
health [P] | -6.61 | 2.93 | [-11.44, -1.78] | -2.25 | 0.025   
exerany × health [G] | -2.58 | 2.19 | [ -6.18, 1.02] | -1.18 | 0.238   
exerany × health [VG] | -2.63 | 2.20 | [ -6.24, 0.99] | -1.20 | 0.232   
exerany × health [F] | 3.62 | 2.50 | [ -0.50, 7.74] | 1.45 | 0.149   
exerany × health [P] | 5.20 | 3.58 | [ -0.69, 11.09] | 1.45 | 0.146

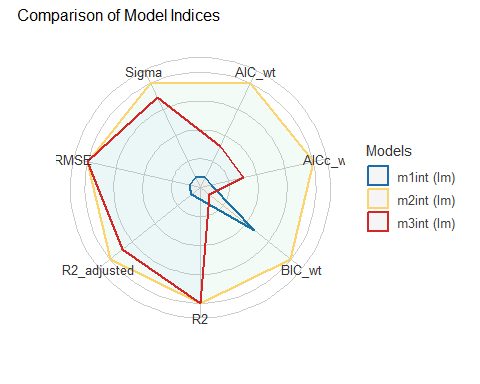
Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

## Comparison of interaction models

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)

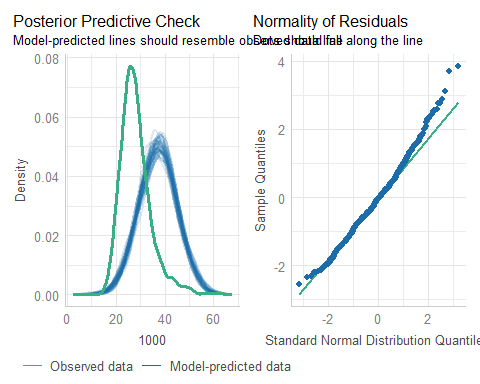
| mod | r.squared | adj.r.squared | sigma | df | df.residual | nobs | AIC | BIC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m1int | 0.0812 | 0.0687 | 7.340 | 9 | 660 | 670 | 4,584.4 | 4,634.0 |
| m2int | 0.0916 | 0.0778 | 7.304 | 10 | 659 | 670 | 4,578.7 | 4,632.8 |
| m3int | 0.0916 | 0.0765 | 7.309 | 11 | 658 | 670 | 4,580.7 | 4,639.3 |

plot(compare\_performance(m1int, m2int, m3int))



## m3int model check

check\_model(m3int, detrend = FALSE, check = c("pp\_check", "qq"))



## Testing Sample for m3 and m3int?

m3\_test\_aug <- augment(m3, newdata = test\_c3im) |>  
 mutate(bmi\_fit = 1000/.fitted, bmi\_res = bmi - bmi\_fit)  
m3int\_test\_aug <- augment(m3int, newdata = test\_c3im) |>  
 mutate(bmi\_fit = 1000/.fitted, bmi\_res = bmi - bmi\_fit)  
  
testing\_r2 <- bind\_rows(  
 yardstick::rsq(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m3\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rsq(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::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(  
 yardstick::rmse(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m3\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::rmse(m3int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(mod = c("m1", "m2", "m3", "m1int",  
 "m2int", "m3int"))  
  
testing\_mae <- bind\_rows(  
 yardstick::mae(m1\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m2\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m3\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m1int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::mae(m2int\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 yardstick::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 | 0.0753 | 5.6373 | 4.3019 | exerany + health |
| m2 | 0.0704 | 5.6469 | 4.3022 | add fruit\_c |
| m3 | 0.0703 | 5.6469 | 4.3030 | add polynomial |
| m1int | 0.0348 | 5.8090 | 4.4593 | m1 + interaction |
| m2int | 0.0330 | 5.8352 | 4.4588 | m2 + interaction |
| m3int | 0.0328 | 5.8373 | 4.4597 | m3 + interaction |

* Did the polynomial in m3 and m3int improve predictions?

## What’s Next?

Basics of logistic regression fitting and evaluation

# Appendix

## Creating Today’s Data Set

url1 <- "https://raw.githubusercontent.com/THOMASELOVE/432-data/master/data/smart\_ohio.csv"  
  
smart\_ohio <- read\_csv(url1)  
  
c3 <- smart\_ohio |>  
 filter(hx\_diabetes == 0, mmsa == "Cleveland-Elyria",  
 complete.cases(bmi)) |>  
 select(bmi, inc\_imp, fruit\_day, drinks\_wk,   
 female, exerany, genhealth, race\_eth,   
 hx\_diabetes, mmsa, SEQNO) |>   
 mutate(across(where(is.character), as\_factor)) |>  
 mutate(ID = as.character(SEQNO - 2017000000)) |>  
 relocate(ID)

## Codebook for useful c3 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 weekly alcoholic drinks consumed |
| female | sex: 1 = female, 0 = male |

## Codebook for useful c3 variables (2)

* 894 subjects in Cleveland-Elyria without diabetes

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

* plus ID, SEQNO, hx\_diabetes (all 0), MMSA
* See [Course Notes Chapter 6](https://thomaselove.github.io/432-notes/smart.html) on BRFSS SMART data

## Basic Data Summaries

Available approaches include:

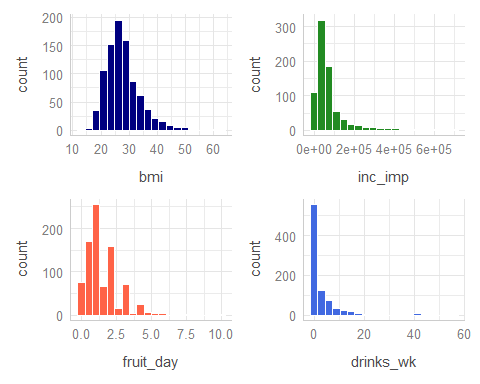
* data\_codebook() from datawizard in easystats
* Hmisc package’s describe(), or
* summary()

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

## Histogram of each quantity

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

p1 <- ggplot(c3, aes(x = bmi)) +   
 geom\_histogram(fill = "navy", col = "white", bins = 20)  
p2 <- ggplot(c3, aes(x = inc\_imp)) +   
 geom\_histogram(fill = "forestgreen", col = "white", bins = 20)  
p3 <- ggplot(c3, aes(x = fruit\_day)) +   
 geom\_histogram(fill = "tomato", col = "white", bins = 20)  
p4 <- ggplot(c3, aes(x = drinks\_wk)) +   
 geom\_histogram(fill = "royalblue", col = "white", bins = 20)  
  
(p1 + p2) / (p3 + p4)



## Binary variables in raw c3

c3 |> 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 c3

c3 |> tabyl(genhealth)

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

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

## Changing the levels for genhealth

c3 <- c3 |>  
 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 c3

c3 |> tabyl(genhealth, health) |> adorn\_title()

health   
 genhealth E G VG F P NA\_  
 1\_Excellent 148 0 0 0 0 0  
 3\_Good 0 274 0 0 0 0  
 2\_VeryGood 0 0 324 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 c3

c3 |> count(race\_eth)

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

“Don’t know”, “Not sure”, and “Refused” were treated as missing.

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

## What is the question you are asking?

Collapsing race\_eth levels *might* be rational for *some* questions.

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

## Is combining race and Hispanic/Latinx ethnicity helpful?

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

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

## Resorting the factor for race\_eth

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

c3 <- c3 |>  
 mutate(race\_eth = fct\_infreq(race\_eth))  
  
c3 |> 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

c3 <- c3 |>  
 select(ID, bmi, inc\_imp, fruit\_day, drinks\_wk,   
 female, exerany, health, race\_eth)  
  
miss\_var\_summary(c3)

# A tibble: 9 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
1 inc\_imp 120 13.4   
2 exerany 42 4.70   
3 fruit\_day 41 4.59   
4 drinks\_wk 39 4.36   
5 race\_eth 13 1.45   
6 health 1 0.112  
7 ID 0 0   
8 bmi 0 0   
9 female 0 0

## Single Imputation with mice

c3im <- mice(c3, m = 1, seed = 20250121, print = FALSE) |>  
 complete() |>  
 tibble()

Warning: Number of logged events: 1

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

prop\_miss\_case(c3im)

[1] 0

dim(c3im)

[1] 894 9

## Saving the tidied data

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

write\_rds(c3, "c03/data/c3.Rds")  
  
write\_rds(c3im, "c03/data/c3im.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.