432 Class 4 Slides

github.com/THOMASELOVE/432-2018

2018-01-25

Setup

```
library(skimr)
library(simputation)
library(broom)
library(modelr)
library(tidyverse)

smartcle1 <- read.csv("data/smartcle1.csv")
```

Today's Materials

- Prediction and Confidence Intervals
- Centering and Rescaling Predictors
- Two-Factor Analysis of Variance
- More to come. . .

Last time, we built smartcle3 and two models. . .

```
set.seed(20180123)

smartcle3 <- smartcle1 %>%
  select(SEQNO, bmi, sleephrs, female, alcdays, exerany) %>%
  impute_rhd(exerany ~ 1) %>%
  impute_pmm(sleephrs ~ 1) %>%
  impute_rlm(bmi ~ female + sleephrs) %>%
  impute_cart(alcdays ~ .) %>%
  tbl_df()

model_int <- lm(bmi ~ female * sleephrs, data = smartcle3)
model_noint <- lm(bmi ~ female + sleephrs, data = smartcle3)
```

Building Predictions for New Data (Individual Subjects)

What do we predict for the `bmi` of a female subject who gets 10 hours of sleep per night? What if the subject was male, instead?

```
new1 <- data_frame(female = c(1, 0), sleephrs = c(10,10))

predict(model_int, newdata = new1,
        interval = "prediction", level = 0.95)
```

```
       fit      lwr      upr
1 26.33333 14.13710 38.52955
2 28.35049 16.13121 40.56977
```

Building Predictions for New Data (Average Predictions)

What do we predict for the average `bmi` of a population of female subjects who sleep for 10 hours? What about the population of male subjects?

```
new1 <- data_frame(female = c(1, 0), sleephrs = c(10,10))

predict(model_int, newdata = new1,
        interval = "confidence", level = 0.95)
```

```
       fit      lwr      upr
1 26.33333 25.25921 27.40744
2 28.35049 27.04027 29.66071
```

Centering and Rescaling Predictors (See Notes sections 2.13, 2.14 and 4.7)

Centering sleephrs to ease interaction description

```
smartcle3 <- smartcle3 %>%
  mutate(sleep_c = sleephrs - mean(sleephrs))

model_int_c <- lm(bmi ~ female * sleep_c, data = smartcle3)
model_int_c
```

```
Call:
lm(formula = bmi ~ female * sleep_c, data = smartcle3)

Coefficients:
   (Intercept)          female         sleep_c
      28.23061        -0.67926         0.04019
female:sleep_c
      -0.44857
```

Interpreting Interaction: Centered `sleephrs`

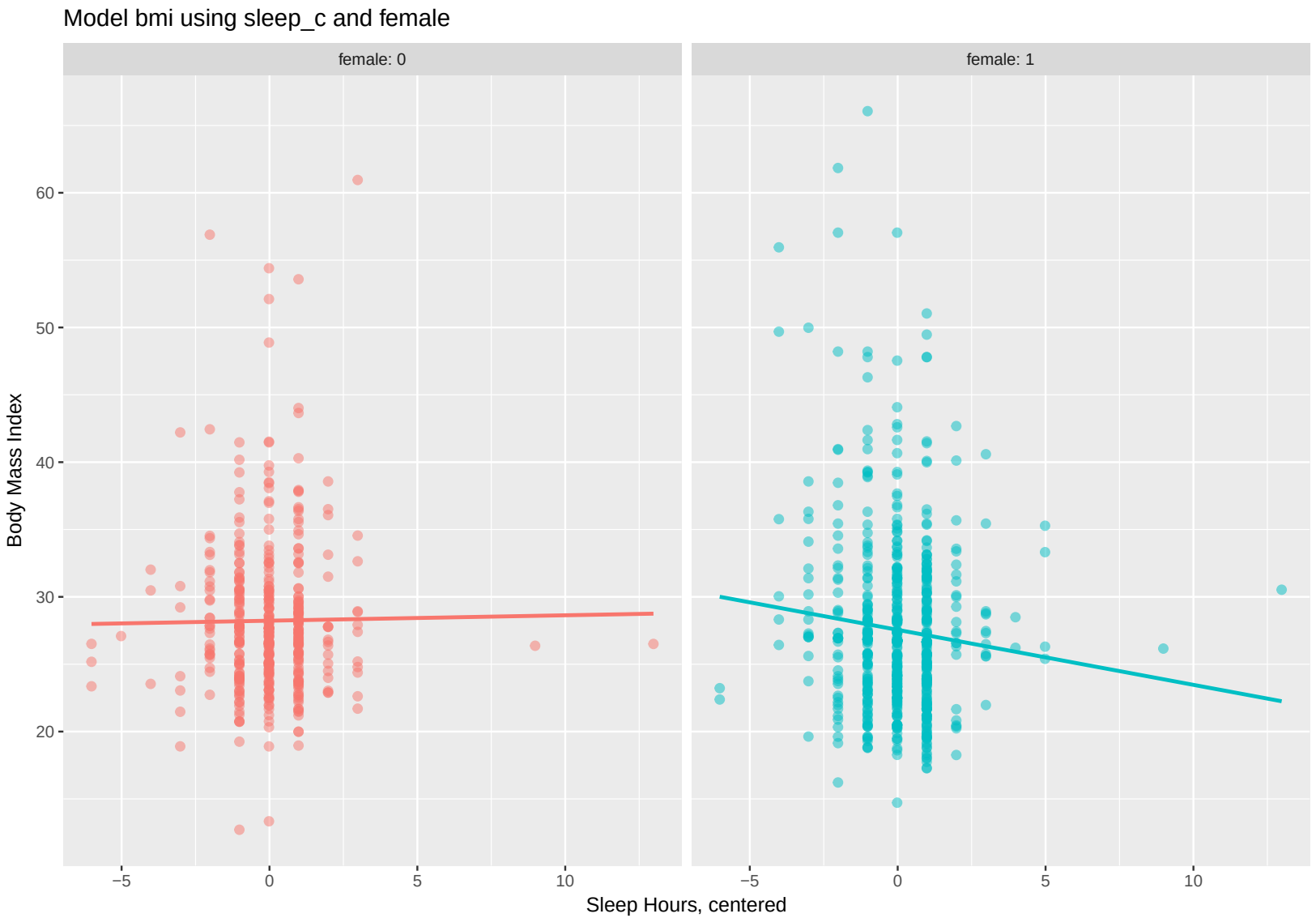
`bmi` = 28.23 - 0.68 `female` + 0.04 centered `sleep_c` - 0.45 `female` x centered `sleep_c`

- Now, 28.23 is the predicted `bmi` for a male who gets the average amount of sleep (7.02 hours)
- And 28.23 - 0.68 = 27.55 is the predicted `bmi` for a female who gets the average amount of sleep.
- So, the main effect of `female` is the predictive difference (female - male) in `bmi` for mean `sleephrs`,
- the product term is the change in the slope of centered `sleephrs_c` on `bmi` for a female rather than a male, and
- the residual standard deviation and the R-squared values remain unchanged from the model before centering.

```
glance(model_int_c) %>% round(., 3)
```

```
  r.squared adj.r.squared sigma statistic p.value df
1     0.009         0.006 6.191      3.08   0.027  4
```

Plotting `bmi` on centered `sleep_c` by `female`



Rescaling?

Centering helped us interpret the main effects in the regression, but it still leaves a scaling problem.

- The female coefficient estimate is much larger than that of sleephrs, but this is misleading, considering that we are comparing the complete change in one variable (sex = female or not) to a 1-hour change in average sleep.
- Gelman and Hill (2007) recommend all continuous predictors be scaled by dividing by 2 standard deviations
    - A 1-unit change in the rescaled predictor corresponds to a change from 1 standard deviation below the mean, to 1 standard deviation above.
    - An unscaled binary (1/0) predictor with 50% probability of occurring will be exactly comparable

Rescaling to sleep_z and re-fitting the model

```
smartcle3 <- smartcle3 %>%
    mutate(sleep_z = (sleephrs - mean(sleephrs)) /
             (2*sd(sleephrs)))

model_int_z <- lm(bmi ~ female * sleep_z, data = smartcle3)

model_int_z
```

```
Call:
lm(formula = bmi ~ female * sleep_z, data = smartcle3)

Coefficients:
   (Intercept)          female          sleep_z
       28.2306         -0.6793          0.1224
female:sleep_z
       -1.3660
```

Comparing our Interaction Models

Original Model

- bmi = 27.95 + 2.47 female + 0.04 sleephrs - 0.45 female x sleephrs

Centered Model

- bmi = 28.23 - 0.68 female + 0.04 sleep_c - 0.45 female x sleep_c

Centered, Rescaled Model

- bmi = 28.23 - 0.68 female + 0.12 sleep_z - 1.37 female x sleep_z

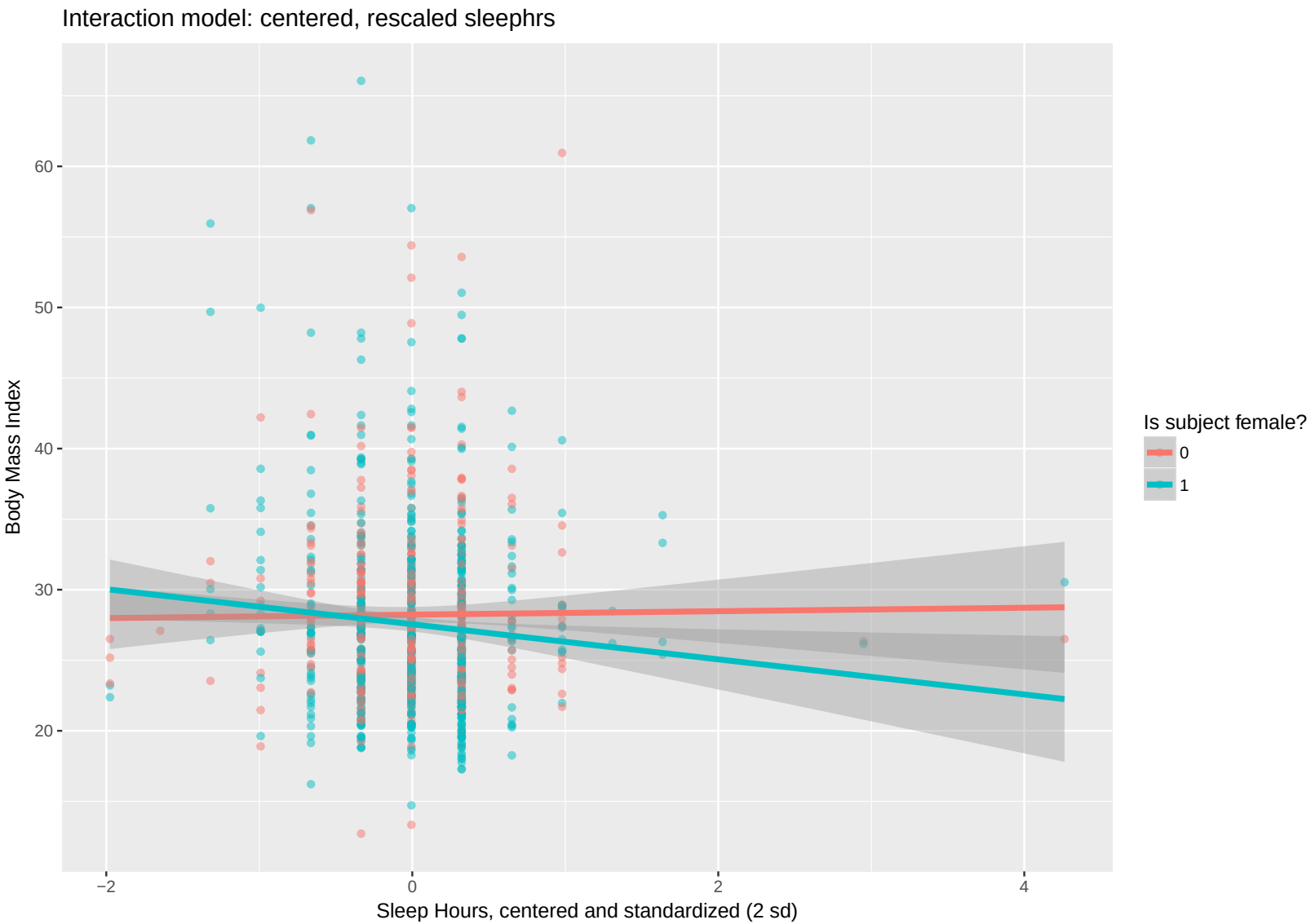
Interpreting the Centered, Rescaled Model

- Main effect of `female`, -0.68, is still the predictive difference (female - male) in `bmi` with `sleephrs` at its mean, 7.02 hours,
- Intercept (28.23) is still the predicted `bmi` for a male who sleeps the mean number of hours, and
- the residual standard deviation and the R-squared values remain unchanged

but now we also have:

- the coefficient of `sleep_z` is the predictive difference in bmi associated with a change in `sleephrs` of 2 standard deviations (from one standard deviation below the mean of 7.02 to one standard deviation above 7.02.)
    - Since sd(sleephrs) is 1.52, this corresponds to a change from 5.50 hours per night to 8.54 hours per night.
- the coefficient of the product term (-1.37) corresponds to the change in the coefficient of `sleep_z` for females as compared to males.

Plotting the Rescaled, Centered Model



Two-Factor Analysis of Variance (see Notes Chapter 3)

How do female and exerany relate to bmi?

```
smart3_sum <- smartcle3 %>%
  group_by(female, exerany) %>%
  summarize(mean.bmi = mean(bmi), sd.bmi = sd(bmi))
```

Resulting tibble for `smart3_sum`

```
smart3_sum
```

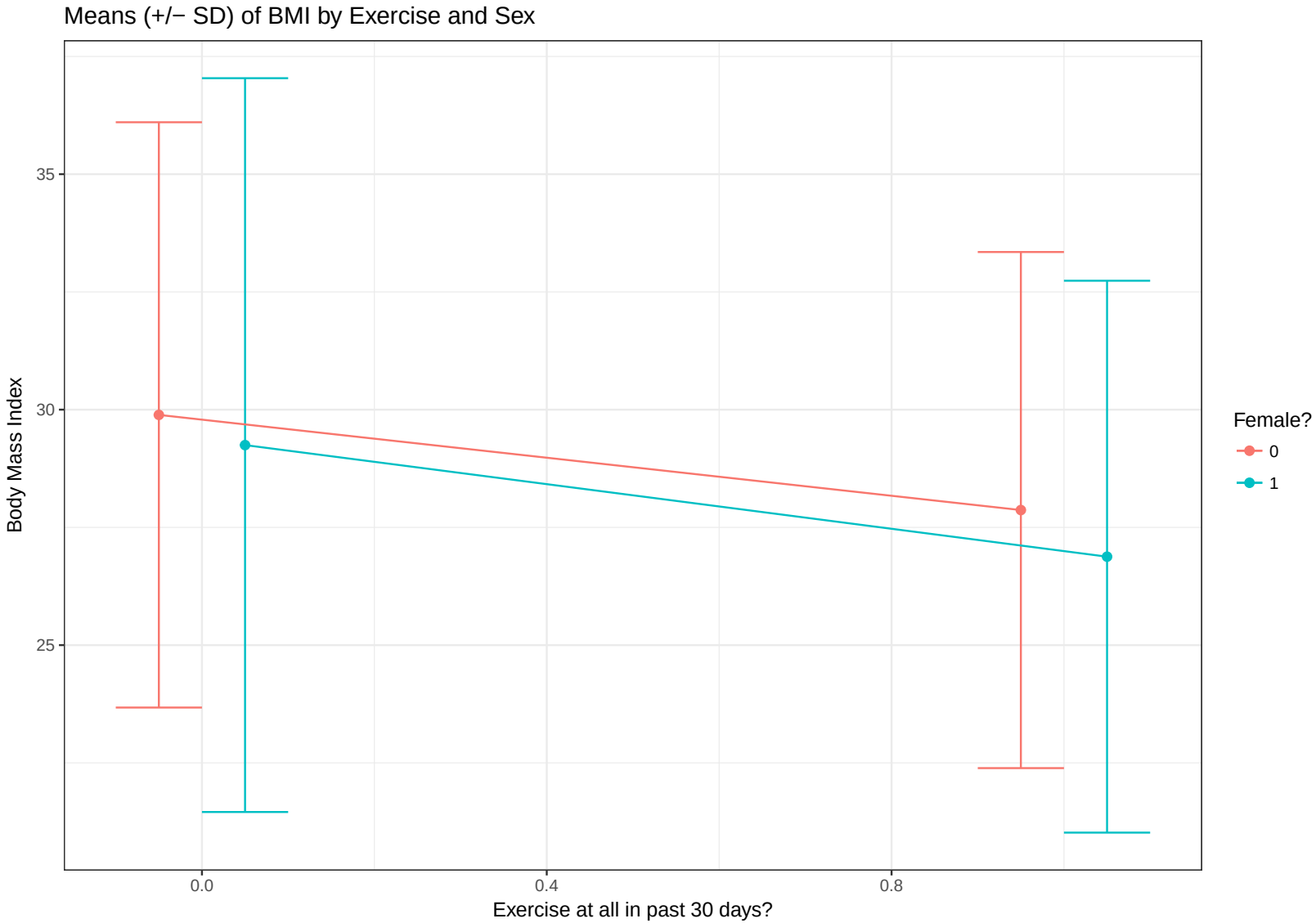
```
# A tibble: 4 x 4
# Groups: female [?]
  female exerany mean.bmi sd.bmi
   <int>   <dbl>    <dbl>  <dbl>
1      0       0     29.9   6.21
2      0       1     27.9   5.48
3      1       0     29.2   7.79
4      1       1     26.9   5.86
```

This would be more useful as a plot.

Building a Means Plot (result on next slide)

```
pd <- position_dodge(0.2)

ggplot(smart3_sum, aes(x = exerany, y = mean.bmi,
                       col = factor(female))) +
  geom_errorbar(aes(ymin = mean.bmi - sd.bmi,
                    ymax = mean.bmi + sd.bmi),
                width = 0.2, position = pd) +
  geom_point(size = 2, position = pd) +
  geom_line(aes(group = female), position = pd) +
  scale_color_discrete(name = "Female?") +
  theme_bw() +
  labs(y = "Body Mass Index",
       x = "Exercise at all in past 30 days?",
       title = "Means (+/- SD) of BMI by Exercise and Sex")
```

Means Plot (Do we have a strong interaction effect?)



Two-Way ANOVA model with Interaction

```
model2 <- lm(bmi ~ female * exerany, data = smartcle3)

anova(model2)
```

```
Analysis of Variance Table

Response: bmi
                 Df Sum Sq Mean Sq F value    Pr(>F)
female            1    118  117.76  3.1288   0.07722 .
exerany           1    947  946.71 25.1530 6.231e-07 ***
female:exerany    1      5    4.97  0.1320   0.71642
Residuals      1032  38843   37.64
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Does it seem like we need the interaction term in this case?

Summary of Two-Factor ANOVA with Interaction

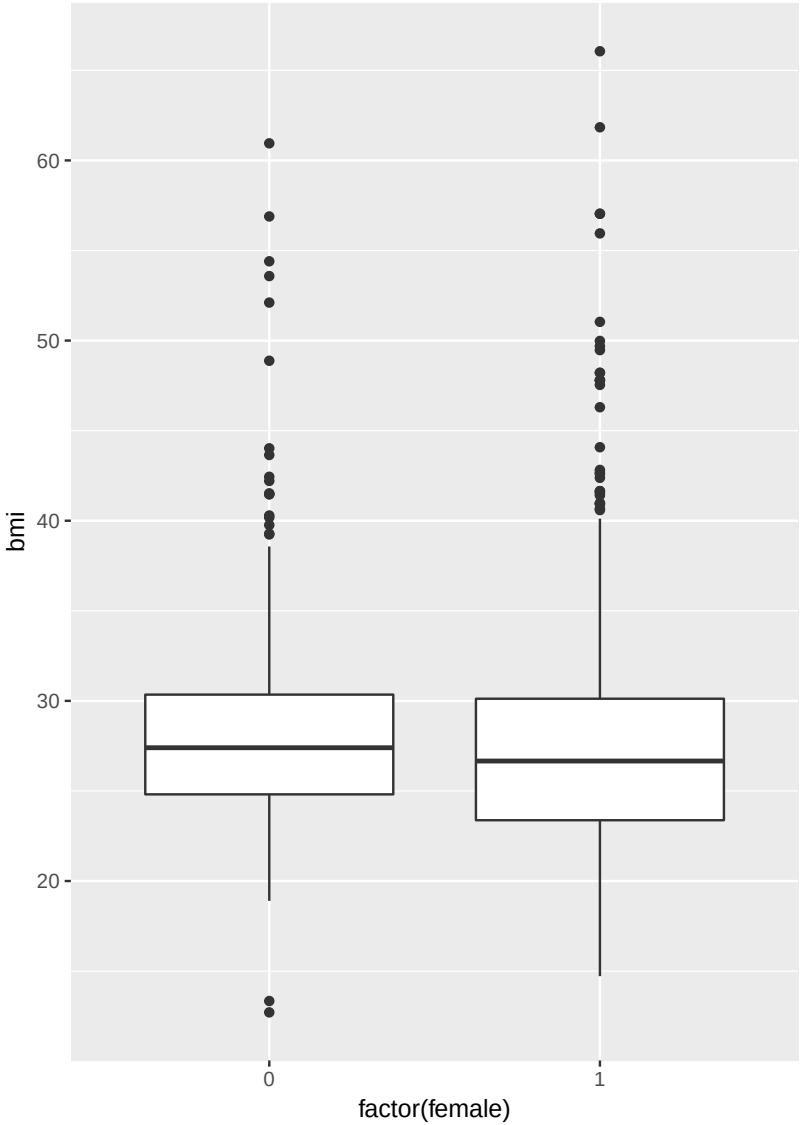
```
> summary(model2)

Call:
lm(formula = bmi ~ female * exerany, data = smartcle3)

Residuals:
    Min      1Q  Median      3Q     Max
-15.158  -3.830  -0.763   2.145  36.813

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    29.8887     0.7132  41.909   <2e-16 ***
female         -0.6414     0.8514  -0.753   0.4514
exerany        -2.0208     0.7870  -2.568   0.0104 *
female:exerany -0.3484     0.9590  -0.363   0.7164
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.135 on 1032 degrees of freedom
Multiple R-squared:  0.0268,    Adjusted R-squared:  0.02397
F-statistic: 9.471 on 3 and 1032 DF,  p-value: 3.557e-06
```

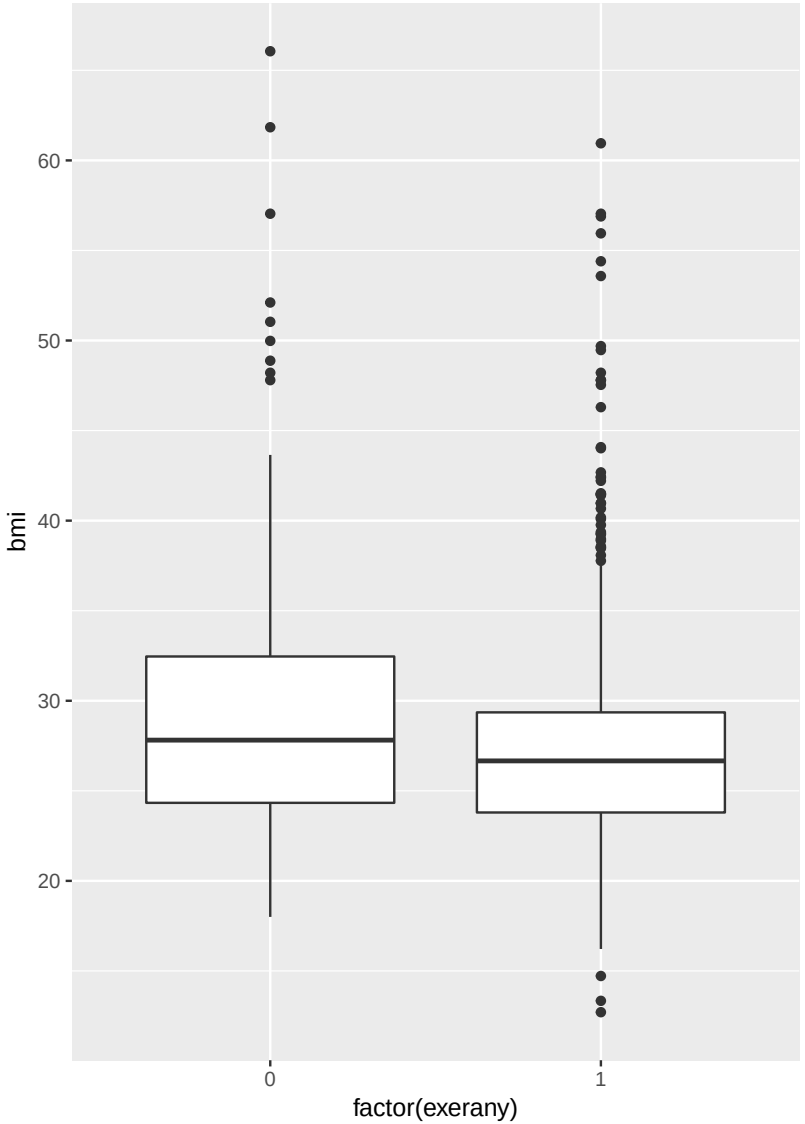
What if we wanted the model with no interaction?

Here's the key plot, then...

```
p1 <- ggplot(smartcle3, aes(x = factor(female), y = bmi)) +
    geom_boxplot()
p2 <- ggplot(smartcle3, aes(x = factor(exerany), y = bmi)) +
    geom_boxplot()

gridExtra::grid.arrange(p1, p2, nrow = 1)
```

Key Plot for Two-Way ANOVA, no interaction





Two-Way ANOVA model without Interaction

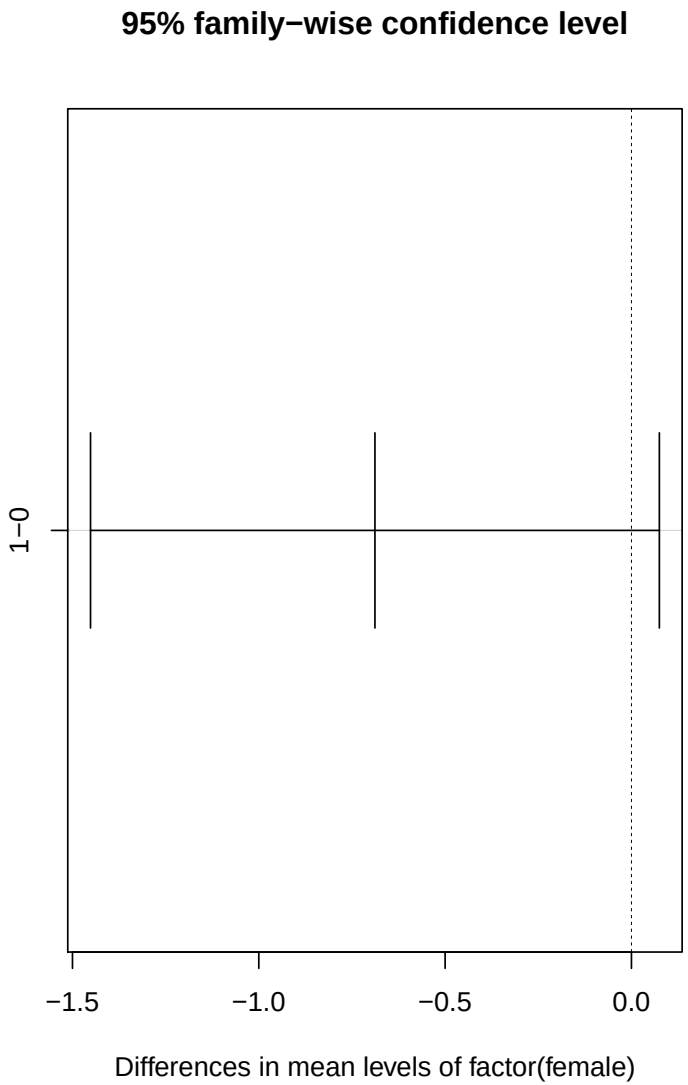
```
model2_noint <- lm(bmi ~ female + exerany, data = smartcle3)

anova(model2_noint)
```

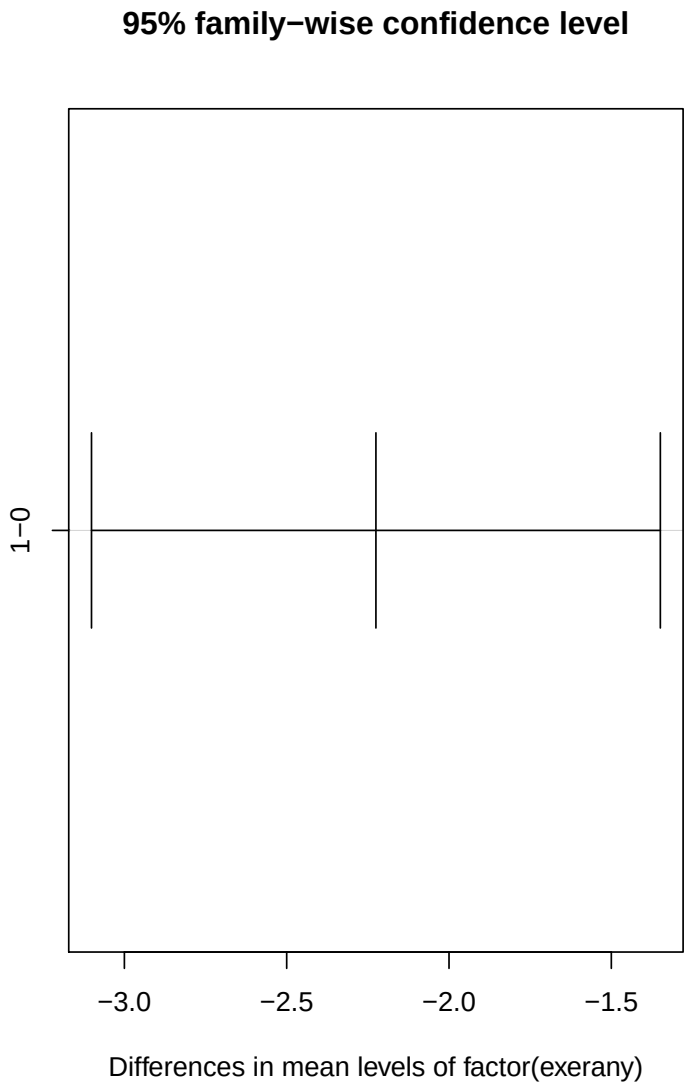
```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value    Pr(>F)
female       1    118  117.76  3.1314   0.07709 .
exerany      1    947  946.71 25.1742 6.164e-07 ***
Residuals 1033  38848   37.61
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary of Two-Factor No Interaction ANOVA

```
> summary(model2_noint)

Call:
lm(formula = bmi ~ female + exerany, data = smartcle3)

Residuals:
    Min      1Q  Median      3Q     Max
-15.116  -3.860  -0.736   2.124  36.895

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.0814     0.4766  63.119  < 2e-16 ***
female       -0.9161     0.3916  -2.339   0.0195 *
exerany      -2.2555     0.4495  -5.017 6.16e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.132 on 1033 degrees of freedom
Multiple R-squared:  0.02667,   Adjusted R-squared:  0.02479
F-statistic: 14.15 on 2 and 1033 DF,  p-value: 8.634e-07
```

Tukey HSD Comparisons (no interaction)

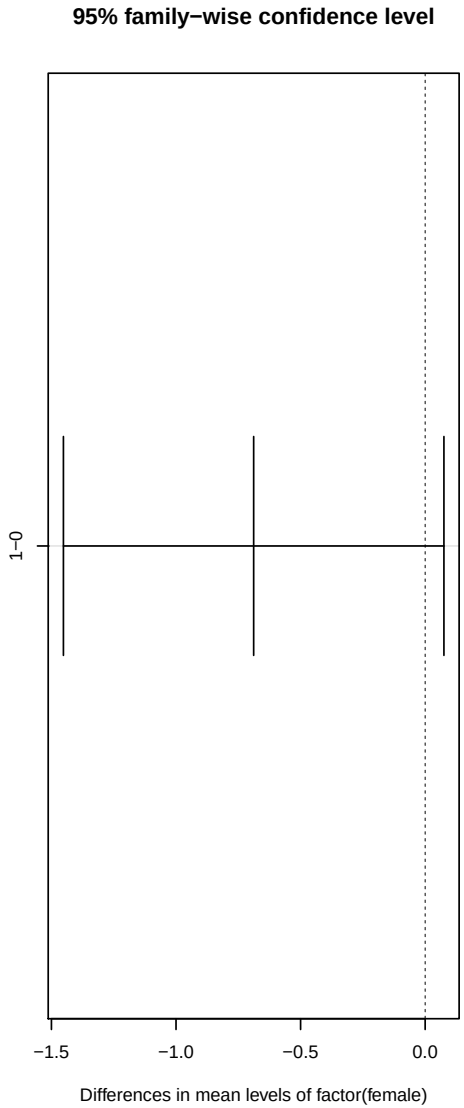


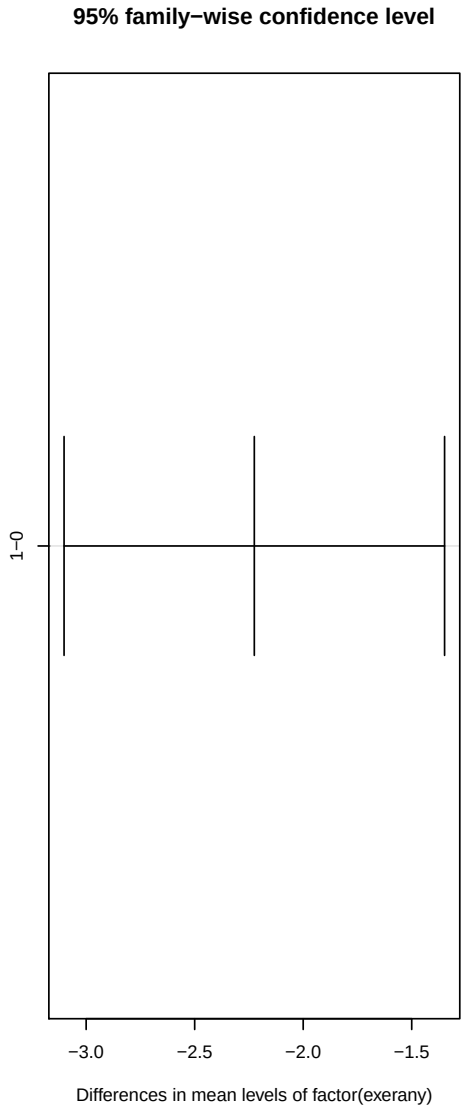


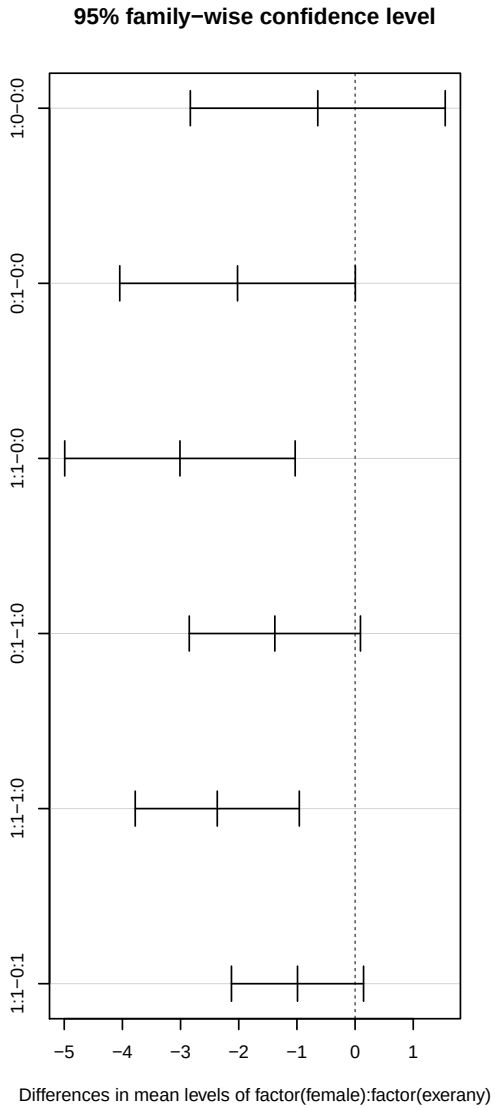
Tukey HSD Comparisons (without interaction)

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = bmi ~ factor(female) + factor(exerany), dat

$`factor(female)`
          diff       lwr        upr     p adj
1-0 -0.6883146 -1.451577 0.07494728 0.0770918

$`factor(exerany)`
          diff       lwr       upr p adj
1-0 -2.225162 -3.101315 -1.349009 7e-07
```

Tukey HSD comparisons WITH interaction







Tukey HSD comparisons WITH interaction

```
> TukeyHSD(aov(bmi ~ factor(female) * factor(exerany), data = smartcle3))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = bmi ~ factor(female) * factor(exerany), data = smartcle3)

$`factor(female)`
          diff       lwr        upr     p adj
1-0 -0.6883146 -1.451898 0.07526902 0.0772162

$`factor(exerany)`
         diff       lwr      upr p adj
1-0 -2.225162 -3.101685 -1.34864 7e-07

$`factor(female):factor(exerany)`
              diff       lwr          upr     p adj
1:0-0:0 -0.6414435 -2.832366  1.549478791 0.8752356
0:1-0:0 -2.0208224 -4.045876  0.004230988 0.0507142
1:1-0:0 -3.0107133 -4.991656 -1.029770182 0.0005667
0:1-1:0 -1.3793789 -2.850875  0.092117170 0.0754115
1:1-1:0 -2.3692698 -3.779445 -0.959094236 0.0000992
1:1-0:1 -0.9898909 -2.125362  0.145580643 0.1124126
```

Indicator Variables

What if I used (1 = yes, 2 = no) instead of (1 = yes, 0 = no) for exerany? What if I tell R that exerany is a factor?

```
smartcle3 <- smartcle3 %>%
  mutate(exer_12 = 2 - exerany,
    exer_yn = fct_recode(factor(exerany), Y = "1", N = "0"))

smartcle3 %>% count(exerany, exer_12, exer_yn)
```

```
# A tibble: 2 x 4
  exerany exer_12 exer_yn     n
    <dbl>   <dbl> <fct>   <int>
1    0       2.00 N         248
2    1.00    1.00 Y         788
```

Two-Predictor model with exerany (1 = yes, 0 = no)

```
lm(bmi ~ exerany * alcdays, data = smartcle3)
```

```
Call:
lm(formula = bmi ~ exerany * alcdays, data = smartcle3)

Coefficients:
    (Intercept)          exerany          alcdays
       29.79211         -2.10499         -0.10141
exerany:alcdays
        0.02546
```

Two-Predictor model with exer_12 (1 = yes, 2 = no)

```
lm(bmi ~ exer_12 * alcdays, data = smartcle3)
```

```
Call:
lm(formula = bmi ~ exer_12 * alcdays, data = smartcle3)

Coefficients:
    (Intercept)          exer_12          alcdays
       25.58214          2.10499          -0.05049
exer_12:alcdays
       -0.02546
```

Compare to

```
(Intercept)     exerany     alcdays  exerany:alcdays
29.79211     -2.10499    -0.10141          0.02546
```

Two-Predictor model with exer_yn (factor)

```
lm(bmi ~ exer_yn * alcdays, data = smartcle3)
```

```
Call:
lm(formula = bmi ~ exer_yn * alcdays, data = smartcle3)

Coefficients:
     (Intercept)          exer_ynY           alcdays
         29.79211          -2.10499          -0.10141
exer_ynY:alcdays
          0.02546
```

Compare to

```
(Intercept)    exerany    alcdays  exerany:alcdays
29.79211    -2.10499   -0.10141          0.02546
```

Fitting Linear Regressions, and then Validating Them

A Linear Regression for bmi from smartcle3

```
mod_ks <- lm(bmi ~ female + sleephrs + alcdays + exerany,
             data = smartcle3)
round(coef(mod_ks),2)
```

```
(Intercept)      female    sleephrs     alcdays     exerany
      32.32       -1.19       -0.24       -0.10       -2.15
```

```
glance(mod_ks)
```

```
   r.squared adj.r.squared    sigma statistic      p.value
1 0.04422908    0.04052095 6.082745  11.92759 1.774063e-09
  df    logLik      AIC      BIC deviance df.residual
1  5 -3337.967 6687.934 6717.592 38146.78        1031
```

tidy(mod_ks)

```
         term    estimate  std.error statistic
1 (Intercept) 32.32268299 1.00240361 32.245178
2      female -1.18547540 0.39507596 -3.000626
3    sleephrs -0.24394812 0.12430035 -1.962570
4     alcdays -0.09690421 0.02446772 -3.960492
5     exerany -2.14510628 0.44678217 -4.801235
         p.value
1 2.601681e-158
2  2.759053e-03
3  4.996479e-02
4  7.993482e-05
5  1.809953e-06
```

ANOVA for sequential testing of predictors

```
anova(mod_ks)
```

```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value    Pr(>F)
female       1    118  117.76  3.1828   0.07471 .
sleephrs     1    119  119.37  3.2263   0.07276 .
alcdays      1    675  675.22 18.2494 2.117e-05 ***
exerany      1    853  852.91 23.0519 1.810e-06 ***
Residuals 1031  38147   37.00
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Different order but the same model?

```
anova(lm(bmi ~ exerany + alcdays + female + sleephrs,
         data = smartcle3))
```

```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value    Pr(>F)
exerany      1    859  858.67 23.2075 1.672e-06 ***
alcdays      1    425  425.22 11.4926 0.0007252 ***
female       1    339  338.87  9.1586 0.0025369 **
sleephrs     1    143  142.51  3.8517 0.0499648 *
Residuals 1031  38147   37.00
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Does Order Matter? Comparing Slopes

model_ks Order	Estimate
Intercept	32.32
female	-1.19
sleephrs	-0.24
alcdays	-0.10
exerany	-2.15

Revised Order	Estimate
Intercept	32.32
exerany	-2.15
alcdays	-0.10
female	-1.19
sleephrs	-0.24

Does Order Matter? Comparing *t* test and CI results

- t tests in `summary` and `tidy` test value as "last predictor in"

`model_ks` Order	Estimate	t test p	95% CI
Intercept	32.32	< 2e-16	(30.3, 34.3)
`female`	-1.19	0.0028	(-2.0, -0.4)
`sleephrs`	-0.24	0.0499	(-0.5, -0.0)
`alcdays`	-0.10	7.9e-05	(-0.14, -0.05)
`exerany`	-2.15	1.8e-06	(-3.0, -1.2)

Revised Order	Estimate	t test p	95% CI
Intercept	32.32	< 2e-16	(30.3, 34.3)
`exerany`	-2.15	1.8e-06	(-2.0, -0.4)
`alcdays`	-0.10	7.9e-05	(-0.5, -0.0)
`female`	-1.19	0.0028	(-0.14, -0.05)
`sleephrs`	-0.24	0.0499	(-3.0, -1.2)

Does Order Matter? Comparing Slopes and *p* values

- t tests in `summary` and `tidy` test value as "last predictor in"
- anova tests of a single `lm` consider predictive value "in sequence"

`model_ks` Order	Estimate	t test *p*	ANOVA *p*
Intercept	32.32	< 2e-16	-
`female`	-1.19	0.0028	0.075
`sleephrs`	-0.24	0.0499	0.073
`alcdays`	-0.10	7.9e-05	2.1e-05
`exerany`	-2.15	1.8e-06	1.8e-06

Revised Order	Estimate	t test *p*	ANOVA *p*
Intercept	32.32	< 2e-16	-
`exerany`	-2.15	1.8e-06	1.7e-06
`alcdays`	-0.10	7.9e-05	0.0007
`female`	-1.19	0.0028	0.0025
`sleephrs`	-0.24	0.0499	0.0499

Do we need all of those variables in `mod_ks`? (Sections 7-8)

Stepwise Regression (backwards elimination)

```
step(mod_ks)
```

```
Start:  AIC=3745.89
bmi ~ female + sleephrs + alcdays + exerany

           Df Sum of Sq   RSS    AIC
<none>                  38147 3745.9
- sleephrs  1    142.51 38289 3747.8
- female    1    333.14 38480 3752.9
- alcdays   1    580.36 38727 3759.5
- exerany   1    852.91 39000 3766.8


Call:
lm(formula = bmi ~ female + sleephrs + alcdays + exerany, data

Coefficients:
```

Stepwise Regression (forwards selection)

```
with(smartcle3,
     step(lm(bmi ~ 1),
          scope = (~ exerany + alcdays + female + sleephrs),
          direction = "forward"))
```

Forward Selection Stepwise Regression, Results: 1

```
Start:  AIC=3784.76
bmi ~ 1

           Df Sum of Sq   RSS    AIC
+ exerany   1    858.67 39053 3764.2
+ alcdays   1    528.88 39383 3772.9
+ sleephrs  1    124.34 39788 3783.5
+ female    1    117.76 39794 3783.7
<none>                  39912 3784.8

Step:  AIC=3764.23
bmi ~ exerany

           Df Sum of Sq   RSS    AIC
+ alcdays   1    425.22 38628 3754.9
+ female    1    205.80 38848 3760.8
+ sleephrs  1    126.97 38926 3762.9
<none>                  39053 3764.2
```

Forward Selection Stepwise Regression, Results: 2

```
Step:  AIC=3754.88
bmi ~ exerany + alcdays

           Df Sum of Sq   RSS    AIC
+ female    1    338.87 38289 3747.8
+ sleephrs  1    148.24 38480 3752.9
<none>                  38628 3754.9

Step:  AIC=3747.76
bmi ~ exerany + alcdays + female

           Df Sum of Sq   RSS    AIC
+ sleephrs  1    142.51 38147 3745.9
<none>                  38289 3747.8
```

Forward Selection Stepwise Regression, Results: 3

```
Step:  AIC=3745.89
bmi ~ exerany + alcdays + female + sleephrs


Call:
lm(formula = bmi ~ exerany + alcdays + female + sleephrs)

Coefficients:
(Intercept)      exerany      alcdays       female     sleephrs
    32.3227      -2.1451      -0.0969      -1.1855      -0.2439
```

Conclusions?

- Forward selection and backwards elimination show the same model, which is also the kitchen sink model.
  - Does that mean that the model is right?
  - Does that mean that the model is good?
  - Does that mean that the model is the best possible combination of these predictors?

- Should we feel substantially more confident about the above statements when the forward selection result = the backwards elimination result, as in our model for `bmi` using `smartcle3`?

Conclusions?

- Forward selection and backwards elimination show the same model, which is also the kitchen sink model.
  - Does that mean that the model is right?
  - Does that mean that the model is good?
  - Does that mean that the model is the best possible combination of these predictors?

- Should we feel substantially more confident about the above statements when the forward selection result = the backwards elimination result, as in our model for `bmi` using `smartcle3`?

- No.

Validating the Model (See Section 6)

Training and Test Samples (as in 431)

Suppose we want to evaluate whether our `model_ks` predicts effectively in new data.

One approach (used, for instance, in 431) would be to split our sample into a separate training (perhaps 70% of the data) and test (perhaps 30% of the data) samples, and then:

1. fit the model in the training sample,
2. use the resulting model to make predictions for `bmi` in the test sample, and
3. evaluate the quality of those predictions, perhaps by comparing the results to what we'd get using a different model.

But there are problems with this approach, especially if n is small.

What else could we do?

Suppose we're afraid that our model building and testing will be hampered by a small sample size.

- A potential solution is the idea of **cross-validation**, which involves partitioning our data into a series of training-test subsets, multiple times, and then combining the results.
- So, in the next slides, I'll show you how to do something called **10-fold cross validation** using some tools from the `modelr` package, which is a non-core part of the `tidyverse`.

10-fold cross validation: The idea

1. Split the 1,036 observations in our `smartcle3` data frame into a partition of about 90% (so about 932 observations) for a training sample, leaving the remaining 10% (about 104 observations) for a test sample. Label the test sample `.id` = 1 in R.
2. Refit our model (here, kitchen sink) to the training sample, and use it to predict our outcome (`bmi`) in the test sample.
3. Store the prediction results for the subjects in the test sample.
4. Split the observations again, ensuring that a completely new 10% gets held out for the test sample, labeling this new test sample `.id` = 2 in R. Then redo parts 2 and 3. Now you have prediction results for 20% of the subjects in the original data.
5. Repeat the process (10x in total) until you have prediction results for all 100% of the subjects in the original data. Thus, each observation is used 9 times in the training sample, and once in the test sample.

10-fold cross-validation of mod_ks

```
set.seed(432021)

sink_models <- smartcle3 %>%
    crossv_kfold(k = 10) %>%
    mutate(model = map(train, ~
                       lm(bmi ~ female + sleephrs +
                            alcdays + exerany, data = .)))

sink_predictions <- sink_models %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))
```

The first few cross-validated predictions

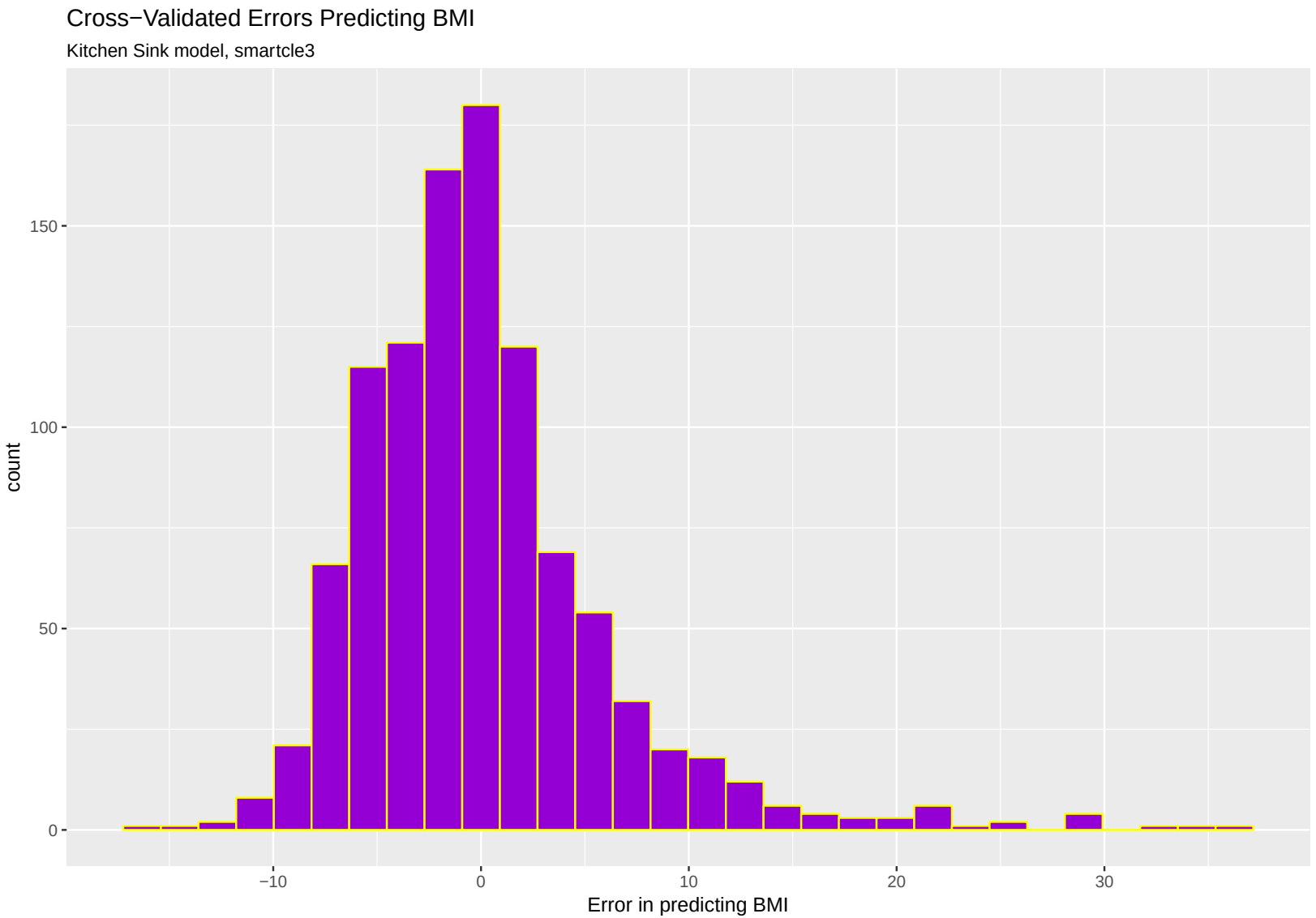
```
head(sink_predictions, 3)
```

```
# A tibble: 3 x 13
  .id         SEQNO   bmi sleephrs female alcdays exerany
  <chr>       <dbl> <dbl>    <int>  <int>   <dbl>   <dbl>
1 01    2016000003  26.9         8      0    4.00       0
2 01    2016000025  21.0         8      1    0          1.00
3 01    2016000028  21.2         7      1    4.00       1.00
# ... with 6 more variables: sleep_c <dbl>, sleep_z <dbl>,
#   exer_12 <dbl>, exer_yn <fct>, .fitted <dbl>, .se.fit
#   <dbl>
```

Graphing the Cross-Validated Prediction Errors of `bmi` (code)

```
sink_predictions %>%
    mutate(errors = bmi - .fitted) %>%
    ggplot(., aes(x = errors)) +
    geom_histogram(bins = 30, fill = "darkviolet",
                   col = "yellow") +
    labs(title = "Cross-Validated Errors Predicting BMI",
         subtitle = "Kitchen Sink model, smartcle3",
         x = "Error in predicting BMI")
```

Cross-Validated Prediction Errors of `bmi`



Summary Statistics Based on Cross-Validated Prediction Errors

We'll look at the **root mean squared prediction error** or RMSE, and the **mean absolute error**, too.

```
sink_predictions %>%
    summarize(RMSE_sink = sqrt(mean((bmi - .fitted) ^2)),
              MAE_sink = mean(abs(bmi - .fitted)))
```

```
# A tibble: 1 x 2
  RMSE_sink MAE_sink
      <dbl>    <dbl>
1      6.11     4.27
```

Comparison to a Model with the Intercept Only (predict mean BMI)?

```
sink_predictions %>%
    summarize(RMSE_sink = sqrt(mean((bmi - .fitted) ^2)),
        RMSE_intercept = sqrt(mean((bmi - mean(bmi))^2)),
        MAE_sink = mean(abs(bmi - .fitted)),
        MAE_intercept = mean(abs(bmi - mean(bmi)))) %>%
  round(., 3)
```

```
# A tibble: 1 x 4
  RMSE_sink RMSE_intercept MAE_sink MAE_intercept
      <dbl>          <dbl>    <dbl>         <dbl>
1      6.11           6.21     4.26          4.35
```

Next Week

- Homework 1 discussion in class Tuesday
- Stepwise Regression via the Allen-Cady Procedure
- Best Subsets approaches to Variable Selection
- Making Decisions about Non-Linearity in Y or in the Xs