Recitation 2: Multicollinearity and Joint Hypotheses

Matthew Alampay Davis

June 6, 2023

Let's get into the habit of loading all our packages in the opening "preamble" chunk:

A few things to note here. The chunk above is what we might call the 'preamble', basically just the preliminary stuff we run at the start of a script or notebook. Here, we're using it to load all packages and all datasets we'll use throughout so that we don't have to worry about them later; much more convenient to do them all at once at the beginning than think about the ordering of commmands in different parts of the Notebook.

Since the preamble often produces a lot of output, we might want to omit it from the pdf file we eventually produce. We can do this by beginning the chunk with {r, include = FALSE} instead of just {r}.

Some of the packages loaded above are ones we've used before. The last two packages are new so install those packages if you haven't already. 'magrittr' (named after the artist Rene Magritte) expands on the 'piping' grammar of tidyverse packages, very useful for convenient data manipulation and cleaning. We'll get to this later.

The 'car' package allows us to perform linear hypothesis tests of the type that will be useful in this week's problem set and future ones. So we will use this package whenever we are given a question that asks us to use a regression result to conduct a joint hypothesis test like $H_0 : \beta_1 = \beta_2 = 0$ (covered here) or even more complicated linear hypotheses like $H_0 : \beta_1 = -\beta_2$ or $H_0 : \beta_1 = \beta_3$ or even $H_0 : \beta_1 = 2\beta_3 + 4\beta_5$ if we wanted (subject of coming lectures).

More R Basics

Missing data

Occasionally we'll run into a dataset that is incomplete because it has missing values. One example of this is the in-built *airquality* dataset:

```
summary(airquality)
```

```
##      Ozone           Solar.R          Wind             Temp
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
##  NA's   :37       NA's   :7
##      Month            Day
##  Min.   :5.000   Min.   : 1.0
```

```
##  1st Qu.:6.000   1st Qu.: 8.0
##  Median :7.000   Median :16.0
##  Mean   :6.993   Mean   :15.8
##  3rd Qu.:8.000   3rd Qu.:23.0
##  Max.   :9.000   Max.   :31.0
##
```

Look at the variables *Ozone* and *Solar.R*. The last row says these variables have 37 NA's and 7 NA's respectively. NA is how most coding languages refer to missing data. This means that when we want to compute a statistic of some of that variable, we'll get an error:

```
mean(airquality$Ozone)
```

```
## [1] NA
```

```
sd(airquality$Ozone)
```

```
## [1] NA
```

There are a couple of ways around this. For statistics like mean, standard deviation, variance, etc., these functions come with an argument that ignores all NA's in the vector we're looking at. *na.rm* is short for "NA remove":

```
mean(airquality$Ozone, na.rm = TRUE) # make sure to use all caps for TRUE and FALSE
```

```
## [1] 42.12931
```

```
sd(airquality$Solar.R, na.rm = T) # capital T is shorthand for TRUE, same with F for FALSE
```

```
## [1] 90.05842
```

We may want to create an object that is the same as airquality, but with all observations that have an NA for any of the variables gets removed:

```
complete.airquality <- na.omit(airquality) # Omit all rows with an NA
summary(complete.airquality)
```

```
##      Ozone           Solar.R           Wind             Temp
##  Min.   :  1.0   Min.   :  7.0   Min.   : 2.30   Min.   :57.00
##  1st Qu.: 18.0   1st Qu.:113.5   1st Qu.: 7.40   1st Qu.:71.00
##  Median : 31.0   Median :207.0   Median : 9.70   Median :79.00
##  Mean   : 42.1   Mean   :184.8   Mean   : 9.94   Mean   :77.79
##  3rd Qu.: 62.0   3rd Qu.:255.5   3rd Qu.:11.50   3rd Qu.:84.50
##  Max.   :168.0   Max.   :334.0   Max.   :20.70   Max.   :97.00
##      Month            Day
##  Min.   :5.000   Min.   : 1.00
##  1st Qu.:6.000   1st Qu.: 9.00
##  Median :7.000   Median :16.00
##  Mean   :7.216   Mean   :15.95
##  3rd Qu.:9.000   3rd Qu.:22.50
##  Max.   :9.000   Max.   :31.00
```

Notice that the two datasets have different numbers of rows: all the rows with an NA in it have been removed:

```
nrow(airquality)
```

```
## [1] 153
```

```
nrow(complete.airquality)
```

```
## [1] 111
```

We can also use the familiar *filter* function from the dplyr package to remove all observations that have NAs for a specific variable

```
# Use filtering to keep only observations for which Solar.R variable is not NA
airquality2 <- dplyr::filter(airquality, is.na(`Solar.R`)==FALSE)
summary(airquality2)
```

```
##      Ozone           Solar.R           Wind           Temp           Month
##  Min.   :  1.0   Min.   :  7.0   Min.   : 1.7   Min.   :57.00   Min.   :5.000
##  1st Qu.: 18.0   1st Qu.:115.8   1st Qu.: 7.4   1st Qu.:73.00   1st Qu.:6.000
##  Median : 31.0   Median :205.0   Median : 9.7   Median :79.00   Median :7.000
##  Mean   : 42.1   Mean   :185.9   Mean   :10.0   Mean   :78.12   Mean   :7.027
##  3rd Qu.: 62.0   3rd Qu.:258.8   3rd Qu.:11.5   3rd Qu.:84.00   3rd Qu.:8.000
##  Max.   :168.0   Max.   :334.0   Max.   :20.7   Max.   :97.00   Max.   :9.000
##  NA's   :35
##       Day
##  Min.   : 1.00
##  1st Qu.: 9.00
##  Median :16.00
##  Mean   :16.12
##  3rd Qu.:23.75
##  Max.   :31.00
##
```

```
nrow(airquality2)
```

```
## [1] 146
```

Indexing

We can pick out the nth element of a vector or dataframe just by using square brackets:

```
cps$age2[1]
```

```
## Warning: Unknown or uninitialised column: 'age2'.
```

```
## NULL
```

gives us the squared age of the first observation. This index works because we are looking at a vector, which is one dimensional. In comparison a data.frame is two-dimensional:

```
head(cps)
```

```
## # A tibble: 6 x 5
##    year   ahe bachelor female   age
##   <dbl> <dbl>    <dbl>  <dbl> <dbl>
## 1  2015 11.8         0      0    26
## 2  2015  9.62        0      1    33
## 3  2015 12.0         0      0    31
## 4  2015 18.4         0      0    32
## 5  2015 41.8         0      0    28
## 6  2015 19.2         0      1    31
```

```
dim(cps)
```

```
## [1] 7098    5
```

If we wanted to pick the element in the fifth row and third column, we'd just use [5,3] as so:

```
cps[5,3]
```

```
## # A tibble: 1 x 1
##   bachelor
##      <dbl>
## 1        0
```

Just remember the first number is the row and the second is the column. We can also say

```
cps[5, 'age']
```

```
## # A tibble: 1 x 1
##     age
##   <dbl>
## 1    28
```

And if we wanted, for example, every column pertaining to the fifth observation, we would just leave the second argument blank to include all columns available:

```
cps[5,]
```

```
## # A tibble: 1 x 5
##    year   ahe bachelor female   age
##   <dbl> <dbl>    <dbl>  <dbl> <dbl>
## 1  2015  41.8        0      0    28
```

Data manipulation and cleaning: Across the "tidyverse"

We're here introducing some basic data-cleaning functions from a variety of packages which are designed to be mutually compatible. This ecosystem of packages can be loaded all at once as one umbrella package called the "tidyverse". Here are some of its most useful functions:

Subsetting: filter() and select()

We already know how to subset data by rows using the filter function:

```
table(earnings$educ)
```

```
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##   16   11   23   33   34   41  104   63  209  290  395  579 6495 1547 2081  815
##   16   17   18   19
## 2714  680 1705   35
```

```
test <- filter(earnings, educ >= 12)
table(test$educ)
```

```
##
##   12   13   14   15   16   17   18   19
## 6495 1547 2081  815 2714  680 1705   35
```

This reduces the earnings dataset to only include observations with at least a high school education. Similarly, we can reduce the dataset by removing variables while retaining all observations:

```
names(earnings)
```

```
##  [1] "sex"        "age"        "mrd"        "educ"       "cworker"
##  [6] "region"     "race"       "earnings"   "height"     "weight"
## [11] "occupation"
```

```
test <- select(earnings,
               age, race, earnings, height, weight, educ)
head(test)
```

```
## # A tibble: 6 x 6
##   age       race                    earnings height weight  educ
##   <dbl+lbl> <dbl+lbl>                  <dbl>  <dbl>  <dbl> <dbl>
## 1 48        1 [non-hispanic white]   84055.     65    133    13
## 2 41        1 [non-hispanic white]   14021.     65    155    12
## 3 26        1 [non-hispanic white]   84055.     60    108    16
## 4 37        1 [non-hispanic white]   84055.     67    150    16
## 5 35        1 [non-hispanic white]   28560.     68    180    16
## 6 25        1 [non-hispanic white]   23363.     63    101    15
```

The first argument is the dataset we want to subset, then every argument after that separated by commas are the variables we want to keep. This also allows us to change the ordering of our columns.

Creating and transforming variables: mutate()

Now suppose we want to create a new variable in our dataset that is a transformation of already existing ones. Suppose we wanted to create a new variable "college" which is going to be a 1 for any observation with an education of at least 16 years and a 0 otherwise.

We might know one way of doing this already:

```
# Creates a dummy variable that's TRUE or FALSE for education > 16
test$college <- test$educ >= 16
# Converts this to a number where TRUE == 1 and FALSE == 0
test$college <- as.numeric(test$college)
table(test$college)
```

```
##
##     0     1
## 12736  5134
```

A more convenient way to do this and create many such transformed variables at a time is the 'mutate' function. We can create the same variable but let's also add similar ones corresponding to different education levels:

```
earnings <- mutate(earnings,
                   lt.hs = as.numeric(educ < 12),
                   hs = as.numeric(educ == 12),
                   some.college = as.numeric(12 < educ & educ < 16),
                   college = as.numeric(educ >= 16)) %>%
  # Some more examples
  mutate(log.age = log(age),
         age.sq = age^2,
         no.college = 1-college)
head(earnings)
```

```
## # A tibble: 6 x 18
##   sex         age   mrd      educ cworker region  race    earnings height weight
##   <dbl+lbl>   <dbl> <dbl+l> <dbl> <dbl+l> <dbl+l> <dbl+l>    <dbl>  <dbl>  <dbl>
## 1 0 [0:femal~ 48    1 [1:M~    13 1 [1:P~ 3 [3:S~ 1 [non~   84055.     65    133
## 2 0 [0:femal~ 41    6 [6:N~    12 1 [1:P~ 2 [2:M~ 1 [non~   14021.     65    155
## 3 0 [0:femal~ 26    1 [1:M~    16 1 [1:P~ 1 [1:N~ 1 [non~   84055.     60    108
## 4 0 [0:femal~ 37    1 [1:M~    16 1 [1:P~ 2 [2:M~ 1 [non~   84055.     67    150
## 5 0 [0:femal~ 35    6 [6:N~    16 1 [1:P~ 1 [1:N~ 1 [non~   28560.     68    180
## 6 0 [0:femal~ 25    6 [6:N~    15 1 [1:P~ 4 [4:W~ 1 [non~   23363.     63    101
## # i 8 more variables: occupation <dbl>, lt.hs <dbl>, hs <dbl>,
## #   some.college <dbl>, college <dbl>, log.age <dbl>, age.sq <dbl>,
## #   no.college <dbl>
```

Notice the "%>%" connecting the two mutate commands. This is called a "pipe" as it conveniently joins successive functions. Essentially it takes the output of whatever's on the left of the "%>%" and immediately uses it as the first argument of the function on the right of the "%>%.

Here's another example: suppose we want to identify the observations corresponding to white women between the ages of 40 and 50 and then among this group:

```
earnings.example <- earnings %>%
  # Create a dummy variable that's 1 for all women (sex == 1) who are white (race == 1)
  mutate(ww = sex == 1 & race ==1) %>%
  # Create a dummy that's a 1 if they are in the desired age range
  mutate(age.4050 = (age >= 40 & age < 50)) %>%
  # Subset to the people for whom both these dummies are == 1
  filter(ww == 1 & age.4050 == 1)
```

We can take it even further by using the "%$%" pipe, which is used to refer to variables within the object to its left. Here's an example tabulating the education level of the group defined above:

```
# With pipes
earnings.example %$%
  # Select the college variable in the earnings.example data:
  college %>%
  # Use the table function to count how many people have a college degree
  table() %>%
  # Convert these counts to a percentage
  prop.table()
```

```
## .
##        0        1
## 0.638479 0.361521
```

```
# Without pipes, as one command
prop.table(table(earnings.example$college))
```

```
##
##        0        1
## 0.638479 0.361521
```

More with regressions

Multicollinearity and dropping a regressor/covariate

We know from lectures that multicollinearity occurs when one regressor is a linear combination of the other regressors. For example, above we created four dummy variables corresponding to those will less than a high school education, those with only a high school education, those who went to college but didn't graduate, and those with at least a bachelor's degree. Clearly, all observations will have a 1 for one of the variables and a 0 for all the others; they are mutually exclusive and exhaustive.

If we wanted to look at the relative effects of these different levels of education while controlling for age, we would have to remove a regressor from our regression equation. By default, R and Stata prefer to drop one of the collinear education variables:

```
intercept.model <- lm_robust(earnings ~ lt.hs + hs + some.college + college + age,
                             se_type = 'HC1',
                             earnings)
summary(intercept.model)
```

```
## 1 coefficient  not defined because the design matrix is rank deficient
```

```
##
## Call:
## lm_robust(formula = earnings ~ lt.hs + hs + some.college + college +
##     age, data = earnings, se_type = "HC1")
##
## Standard error type:  HC1
##
```

```
## Coefficients: (1 not defined because the design matrix is rank deficient)
##              Estimate Std. Error t value   Pr(>|t|) CI Lower CI Upper    DF
## (Intercept)   13569.4     908.82   14.93  4.167e-50  11788.0  15350.8 17865
## lt.hs              NA         NA      NA         NA       NA       NA    NA
## hs            12331.1     562.30   21.93 3.281e-105  11229.0  13433.3 17865
## some.college  20699.5     614.99   33.66 7.395e-241  19494.1  21905.0 17865
## college       33300.4     590.93   56.35  0.000e+00  32142.1  34458.7 17865
## age             344.8      18.39   18.75  1.047e-77    308.8    380.9 17865
##
## Multiple R-squared:  0.1637 ,    Adjusted R-squared:  0.1635
## F-statistic:    NA on 4 and 17865 DF,  p-value: NA
```

Alternatively, we could drop the intercept and retain both dummy variables. This is because the intercept (which is just a constant) is collinear with the sum of the dummy variables (also a constant: these sum to $1+0+0+0 = 1$ for all observations).

We can implement this by simply adding a "+0" or "-1" to our formula:

```
no.intercept.1 <- lm_robust(earnings ~ 0 + lt.hs + hs + some.college + college + age,
                            se_type = 'HC1',
                            earnings)
summary(no.intercept.1)
```

```
##
## Call:
## lm_robust(formula = earnings ~ 0 + lt.hs + hs + some.college +
##     college + age, data = earnings, se_type = "HC1")
##
## Standard error type:  HC1
##
## Coefficients:
##              Estimate Std. Error t value   Pr(>|t|) CI Lower CI Upper    DF
## lt.hs         13569.4     908.82   14.93  4.167e-50  11788.0  15350.8 17865
## hs            25900.5     792.48   32.68 1.325e-227  24347.2  27453.8 17865
## some.college  34268.9     816.96   41.95  0.000e+00  32667.6  35870.2 17865
## college       46869.8     833.86   56.21  0.000e+00  45235.3  48504.2 17865
## age             344.8      18.39   18.75  1.047e-77    308.8    380.9 17865
##
## Multiple R-squared:  0.7926 ,    Adjusted R-squared:  0.7925
## F-statistic: 1.323e+04 on 5 and 17865 DF,  p-value: < 2.2e-16
```

```
no.intercept.2 <- lm_robust(earnings ~ -1 + height + lt.hs + hs + some.college + college,
                            se_type = 'HC1',
                            earnings)
summary(no.intercept.2)
```

```
##
## Call:
## lm_robust(formula = earnings ~ -1 + height + lt.hs + hs + some.college +
##     college, data = earnings, se_type = "HC1")
##
## Standard error type:  HC1
##
```

```
## Coefficients:
##              Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper    DF
## height          453.8      47.03  9.6492 5.605e-22    361.6      546 17865
## lt.hs         -1483.0    3134.22 -0.4732 6.361e-01  -7626.4     4660 17865
## hs             9843.7    3150.60  3.1244 1.785e-03   3668.2    16019 17865
## some.college  17590.1    3177.68  5.5355 3.147e-08  11361.5    23819 17865
## college       30233.1    3199.42  9.4495 3.813e-21  23961.9    36504 17865
##
## Multiple R-squared:  0.7896 ,    Adjusted R-squared:  0.7895
## F-statistic: 1.301e+04 on 5 and 17865 DF,  p-value: < 2.2e-16
```

Even though they result in different coefficient estimates, the methods are equivalent; the resulting coefficients are only different because they are measuring different relative effects. But with one regression's estimates, you can always back out the other regression's estimates.

Root Mean Squared Error

We encountered one minor difference between lm and lm_robust models last week: you can directly pull the residuals form an lm model but you have to manually construct them for lm_robust models:

```
earnings.lm <- lm(earnings ~ -1 + college + no.college, earnings)
summary(earnings.lm)
```

```
##
## Call:
## lm(formula = earnings ~ -1 + college + no.college, data = earnings)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -56116 -20940  -7532  23212  42810
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## college     60842.4      354.8   171.5   <2e-16 ***
## no.college  41245.1      225.3   183.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25420 on 17868 degrees of freedom
## Multiple R-squared:  0.7789, Adjusted R-squared:  0.7788
## F-statistic: 3.147e+04 on 2 and 17868 DF,  p-value: < 2.2e-16
```

```
earnings.lmrobust <- lm_robust(earnings ~ -1 + college + no.college,
                               se_type = 'HC1',
                               earnings)
summary(earnings.lmrobust)
```

```
##
## Call:
## lm_robust(formula = earnings ~ -1 + college + no.college, data = earnings,
##     se_type = "HC1")
##
```

```
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|) CI Lower CI Upper    DF
## college        60842      361.4   168.4        0    60134    61551 17868
## no.college     41245      223.6   184.5        0    40807    41683 17868
##
## Multiple R-squared:  0.7789 ,    Adjusted R-squared:  0.7788
## F-statistic: 3.119e+04 on 2 and 17868 DF,  p-value: < 2.2e-16
```

Relatedly, the other difference you'll notice between lm and lm_robust models is that RMSE appears in the output for lm models ("Residual standard error: 25420") but not in lm_robust models. To get the RMSE for the latter, you can simply take the square root of its residual mean variance:

```
sqrt(earnings.lmrobust$res_var)
```

```
## [1] 25421.6
```

You can also use the sigma function on the lm model:

```
sigma(earnings.lm)
```

```
## [1] 25421.6
```

You could also compute it manually (maybe as a sanity check), accounting for degrees of freedom "df.residual":

```
sqrt(sum(earnings.lm$residuals^2)/df.residual(earnings.lm))
```

```
## [1] 25421.6
```

R-squared

```
# For lm_robust models:
earnings.lmrobust$r.squared
```

```
## [1] 0.7788642
```

```
earnings.lmrobust$adj.r.squared
```

```
## [1] 0.7788394
```

```
# For lm models:
summ.lm <- summary(earnings.lm)
summ.lm$r.squared
```

```
## [1] 0.7788642
```

```
summ.lm$adj.r.squared
```

```
## [1] 0.7788394
```

F-statistic

```
# For lm_robust models:
earnings.lmrobust$fstatistic
```

```
##    value    numdf    dendf
## 31192.25     2.00 17868.00
```

```
# For lm models:
summ.lm$fstatistic
```

```
##    value    numdf    dendf
## 31466.51     2.00 17868.00
```

Keep in mind the F-statistics found in regression outputs correspond to a specific F-statistic: the one jointly testing the significance of *all* regressors. We may be interested in testing the joint significance of only a subset of regressors.

Joint hypothesis tests

This week, we'll be interested in joint hypothesis tests, which test hypotheses of the following variety:

$$H_0 : \beta_1 = \beta_2 = \beta_3 = 0$$
$$H_1 : \text{At least one of these coefficients is non-zero}$$

Like our standard single-regressor significance test, testing this hypothesis requires estimating a test statistic and evaluating its significance. For this, we'll want to install the *car* package (apparently short for "companion to applied regression"). In particular, we'll want to use the function *linearHypothesis*.

Imagine we ran the following regression to predict future earnings:

```
joint.model <- lm_robust(earnings ~ college + height + weight + race + age,
                         se_type = 'HC1',
                         earnings)
summary(joint.model)
```

```
##
## Call:
## lm_robust(formula = earnings ~ college + height + weight + race +
##     age, data = earnings, se_type = "HC1")
##
## Standard error type:  HC1
##
## Coefficients:
```

```
##              Estimate Std. Error  t value  Pr(>|t|) CI Lower  CI Upper    DF
## (Intercept)  2824.355   3490.465   0.8092 4.184e-01 -4017.29  9666.004 17864
## college     19197.896    421.243  45.5744 0.000e+00 18372.22 20023.572 17864
## height        482.202     51.847   9.3005 1.556e-20   380.58   583.827 17864
## weight         -4.264      3.957  -1.0776 2.812e-01   -12.02     3.492 17864
## race        -3386.093    243.451 -13.9088 9.478e-44 -3863.28 -2908.907 17864
## age           285.093     18.842  15.1308 2.113e-51   248.16   322.025 17864
##
## Multiple R-squared:  0.137 , Adjusted R-squared:  0.1368
## F-statistic: 601.9 on 5 and 17864 DF,  p-value: < 2.2e-16
```

Now suppose we wanted to test whether college education and height are jointly significant as predictors of future earnings. We'd run the following command to derive the corresponding F-statistic:

```
linearHypothesis(joint.model, c('college = 0', 'height = 0'))
```

```
## Linear hypothesis test
##
## Hypothesis:
## college = 0
## height = 0
##
## Model 1: restricted model
## Model 2: earnings ~ college + height + weight + race + age
##
##   Res.Df Df Chisq Pr(>Chisq)
## 1  17866
## 2  17864  2  2297  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This gives us a Chi-squared statistic of 2297, corresponding to a p-value very close to zero.

In this course, we will typically test joint significance using the F-statistic instead:

```
linearHypothesis(joint.model, c('college = 0', 'height = 0'),
                 test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## college = 0
## height = 0
##
## Model 1: restricted model
## Model 2: earnings ~ college + height + weight + race + age
##
##   Res.Df Df      F    Pr(>F)
## 1  17866
## 2  17864  2 1148.5 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our F-statistic is 1148.5 and the p-value is identical to the Chi-squared test

We could also do the same with non-robust standard errors:

```
joint.model.homo <- lm(earnings ~ college + height + weight + race + age, earnings)
```

Because of the assumption of homoskedasticity, the corresponding test statistics will come out slightly different:

```
linearHypothesis(joint.model.homo, c('college = 0', 'height = 0'),
                 test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## college = 0
## height = 0
##
## Model 1: restricted model
## Model 2: earnings ~ college + height + weight + race + age
##
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  17866 1.2627e+13
## 2  17864 1.1178e+13  2 1.4495e+12 1158.3 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-statistic is slightly larger but the conclusion is unchanged. We can also use this lm model to recover the original robust result by specifying the difference in standard errors as an argument to the linearHypothesis function:

```
linearHypothesis(joint.model.homo, c('college = 0', 'height = 0'), white.adjust = 'hc1',
                 test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## college = 0
## height = 0
##
## Model 1: restricted model
## Model 2: earnings ~ college + height + weight + race + age
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1  17866
## 2  17864  2 1148.5 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It should then be straightforward to extend this exercise if you want to jointly test even more variables: just add them to the vector of equations in the linearHypothesis function.

Practice Problem 1: Stock-Watson Empirical Exercise 6.1

Here, we're just doing the subquestions a, b, and d. Part c is a bit time-consuming but a good exercise for understanding the concept of control variables using the getting an understanding of what control variables do using the Frisch-Waugh Theorem.

Part a

```
birth.model.a <- lm_robust(birthweight ~ smoker,
                           data = smoking, se_type = 'HC1')
summary(birth.model.a)
```

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker, data = smoking, se_type = "HC1")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)   3432.1      11.89 288.638 0.000e+00   3408.7   3455.4 2998
## smoker        -253.2      26.81  -9.445 6.903e-21   -305.8   -200.7 2998
##
## Multiple R-squared:  0.0286 ,    Adjusted R-squared:  0.02828
## F-statistic: 89.21 on 1 and 2998 DF,  p-value: < 2.2e-16
```

The estimated effect of smoking on birthweight is -253.2 grams for every unit increase in the smoker variable

Part b

```
birth.model.b <- lm_robust(birthweight ~ smoker + alcohol + nprevist,
                           data = smoking, se_type = 'HC1')
summary(birth.model.b)
```

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker + alcohol + nprevist,
##     data = smoking, se_type = "HC1")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)  3051.25     43.714  69.800 0.000e+00  2965.54  3136.96 2996
## smoker       -217.58     26.108  -8.334 1.175e-16  -268.77  -166.39 2996
## alcohol       -30.49     72.597  -0.420 6.745e-01  -172.84   111.85 2996
## nprevist       34.07      3.608   9.442 7.109e-21    26.99    41.14 2996
##
## Multiple R-squared:  0.07285 ,   Adjusted R-squared:  0.07192
## F-statistic: 59.48 on 3 and 2996 DF,  p-value: < 2.2e-16
```

(i) Smoking may be correlated with both alcohol and the number of pre-natal doctor visits, thus satisfying (1) in Key Concept 6.1. Moreover, both alcohol consumption and the number of doctor visits may have their own independent affects on birthweight, thus satisfying (2) in Key Concept 6.1.

(ii) The estimate is somewhat smaller: it has fallen to 217 grams from 253 grams, so the regression in (a) may suffer from omitted variable bias.

(iii) Predicting Jane's baby's birth weight

```
3051.25-217.58*1-30.49*0+34.07*8
```

```
## [1] 3106.23
```

Alternatively, we know what values she has for all the covariates in order:

```
# Jane smoked (x1 = 1), did not drink (x2 = 0), had eight visits (x3 = 8)
jane <- data.frame(smoker = 1,
                   alcohol = 0,
                   nprevist = 8)
# Use the regression to predict her child's birthweight:
predict(birth.model.b, newdata = jane)
```

```
##        1
## 3106.228
```

(iv)

```
birth.model.b$r.squared
```

```
## [1] 0.0728503
```

```
birth.model.b$adj.r.squared
```

```
## [1] 0.07192191
```

They are nearly identical because the sample size is very large; the degrees of freedom adjustment will be small in comparison

(v) Nprevist is a control variable. It captures, for example, mother's access to healthcare and health. Because Nprevist is a control variable, its coefficient does not have a causal interpretation.

Part d

```
birth.model.d <- lm_robust(birthweight ~ smoker + alcohol + tripre0 + tripre2 + tripre3, data = smoking
summary(birth.model.d)
```

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker + alcohol + tripre0 +
##     tripre2 + tripre3, data = smoking, se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error  t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)   3454.5      12.48 276.7697 0.000e+00   3430.1  3479.02 2994
## smoker        -228.8      26.55  -8.6199 1.068e-17   -280.9  -176.79 2994
## alcohol        -15.1      69.70  -0.2166 8.285e-01   -151.8   121.57 2994
## tripre0       -698.0     146.58  -4.7617 2.011e-06   -985.4  -410.56 2994
## tripre2       -100.8      31.55  -3.1958 1.409e-03   -162.7   -38.97 2994
## tripre3       -137.0      67.70  -2.0231 4.315e-02   -269.7    -4.22 2994
##
## Multiple R-squared:  0.04647 ,   Adjusted R-squared:  0.04487
## F-statistic: 23.22 on 5 and 2994 DF,  p-value: < 2.2e-16
```

(i) Tripre1 is omitted to avoid perfect multicollinearity. (Tripre0+ Tripre1+ Tripre2+ Tripre3 = 1, the value of the "constant" regressor that determines the intercept). The regression would not run, or the software will report results from an arbitrary normalization if Tripre0, Tripre1, Tripre2, Tripre3, and the constant term all included in the regression.

(ii) Babies born to women who had no prenatal doctor visits (Tripre0 = 1) had birthweights that on average were 698.0 grams (1.5 lbs) lower than babies from others who saw a doctor during the first trimester (Tripre1 = 1).

(iii) Babies born to women whose first doctor visit was during the second trimester (Tripre2 = 1) had birthweights that on average were 100.8 grams (0.2 lbs) lower than babies from others who saw a doctor during the first trimester (Tripre1 = 1). Babies born to women whose first doctor visit was during the third trimester (Tripre3 = 1) had birthweights that on average were 137 grams (0.3 lbs) lower than babies from others who saw a doctor during the first trimester (Tripre1 = 1).

(iv) No. The multiple R^2 has decreased from 0.073 to 0.046.

Bonus: Stock-Watson Empirical Exercise 7.2 (b and c)

In the empirical exercises on earning and height in Chapters 4 and 5, you estimated a relatively large and statistically significant effect of a worker's height on his or her earnings. One explanation for this result is omitted variable bias: Height is correlated with an omitted factor that affects earnings. For example, Case and Paxson (2008) suggest that cognitive ability (or intelligence) is the omitted factor. The mechanism they describe is straightforward: Poor nutrition and other harmful environmental factors in utero and in early childhood have, on average, deleterious effects on both cognitive and physical development. Cognitive ability affects earnings later in life and thus is an omitted variable in the regression.

If the mechanism described above is correct, the estimated effect of height on earnings should disappear if a variable measuring cognitive ability is included in the regression. Unfortunately, there isn't a direct measure of cognitive ability in the data set, but the data set does include years of education for each individual. Because students with higher cognitive ability are more likely to attend school longer, years of education might serve as a control variable for cognitive ability; in this case, including education in the regression will eliminate, or at least attenuate, the omitted variable bias problem.

Use the years of education variable (educ) to construct four indicator variables for whether a worker has less than a high school diploma, a high school diploma , some college , a bachelor's degree or higher.

We already did this in the notes above

ii) For women only, run a regression of (1) Earnings on Height and (2) Earnings on Height, including LT_HS, HS, and Some_Col as control variables.

```
earnings.w <- filter(earnings, sex == 1)
mod1 <- lm(earnings ~ height, earnings.w)
mod2 <- lm(earnings ~ height + lt.hs + hs + some.college, earnings.w)
```

iii) Compare the estimated coefficient on Height in regressions (1) and (2). Is there a large change in the coefficient? Has it changed in a way consistent with the cognitive ability explanation? Explain.

```
# Only works for lm models
stargazer(mod1, mod2, type = 'text')
```

```
##
## ===============================================================================
##                                       Dependent variable:
##                     ------------------------------------------------------
##                                            earnings
##                                  (1)                        (2)
## -------------------------------------------------------------------------------
## height                      1,306.860***                744.681***
##                               (100.766)                  (94.699)
##
## lt.hs                                                  -31,400.500***
##                                                           (978.439)
##
## hs                                                     -20,345.850***
##                                                           (692.439)
##
## some.college                                           -12,610.920***
##                                                           (758.065)
##
## Constant                   -43,130.340***                9,862.740
##                              (7,068.481)                (6,697.041)
##
## -------------------------------------------------------------------------------
## Observations                    7,896                      7,896
## R2                              0.021                      0.166
## Adjusted R2                     0.021                      0.165
## Residual Std. Error  26,671.290 (df = 7894)    24,623.220 (df = 7891)
## F Statistic         168.201*** (df = 1; 7894) 392.038*** (df = 4; 7891)
## ===============================================================================
## Note:                                        *p<0.1; **p<0.05; ***p<0.01
```

There is a large change in the coefficient, strongly suggestive of the presence of omitted variable bias in the first regression.

iv) The regression omits the control variable College. Why?

Multicollinearity: all education variables would be a linear combination of the other three. Unless you remove the constant, there is no excluded group to be compared to. You could run a regression with college, but then you'd have to remove the intercept.

v) Test the joint null hypothesis that the coefficients on the education variables are equal to 0

```
linearHypothesis(mod2,
                 c('lt.hs = 0', 'hs = 0', 'some.college = 0'),
                 test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## lt.hs = 0
## hs = 0
## some.college = 0
##
## Model 1: restricted model
## Model 2: earnings ~ height + lt.hs + hs + some.college
##
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1   7894 5.6155e+12
## 2   7891 4.7843e+12  3 8.3112e+11 456.94 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-statistic is very large: 456.94 corresponding to an essentially 0 p-value. We reject the null hypothesis that the coefficients on all these education variables are 0.

vi) Discuss the values of the estimated coefficients on LT_HS, HS, and Some_Col. Explain their relative values.)

```
stargazer(mod2, type = 'text')
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                               earnings
## ------------------------------------------------
## height                       744.681***
##                               (94.699)
##
```

```
## lt.hs                      -31,400.500***
##                               (978.439)
##
## hs                         -20,345.850***
##                               (692.439)
##
## some.college               -12,610.920***
##                               (758.065)
##
## Constant                      9,862.740
##                              (6,697.041)
##
## ------------------------------------------------
## Observations                    7,896
## R2                              0.166
## Adjusted R2                     0.165
## Residual Std. Error    24,623.220 (df = 7891)
## F Statistic           392.038*** (df = 4; 7891)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

The omitted category are those with at least a college education. The coefficients on the other variables are then interpreted as the additional expected salary relative to college-educated people. We see these values are negative and increasingly so for lower levels of education. This suggests expected earnings increase with educational attainment.