PS3 R Notes and PS3 Practice Problems

Matthew Alampay Davis

October 12, 2021

The preamble

```
setwd('~/Documents/Grad School/Columbia/Y3/Metrics TA/Recitation 4')
library(readstata13)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(estimatr)
library(ggplot2)
library(magrittr)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

A couple of 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 coding script or notebook. These lines run the commands that set the working directory (where to look for files) and loads all the packages we'll use for the R Notebook. The 'setwd()' function is needed for your R environment to know where to look for files you refer to in case your Notebook is saved in a folder different from your default working directory (which should usually be the case). You should run this in the Console panel rather than in the notebook since a Notebook will always use the folder the Notebook is saved to as its working directory. That is to say, 'setwd()' is not needed for your Notebook to be converted to a pdf.

Some of these packages we're loading are ones we've used before: in order, the first few let us load Stata files, manipulate data, and implement robust standard errors in our regression models.

The last two lines are new so install those packages if you haven't already. 'magrittr' (named after the artist Rene Magritte) enables to use 'piping' grammar, useful for easy data manipulation. 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 below) or 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 (covered in a future recitation). It's pretty flexible.

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
```

Transforming variables

Now suppose we want to create a new variable in our dataset that is a transformation of an already existing one. Using the airquality dataset again, suppose we want to create a variable for the logarithm of the temperature. We know one way of doing this already:

```
airquality$logtemperature <- log(airquality$Temp)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day logtemperature
## 1    41     190  7.4   67     5   1       4.204693
## 2    36     118  8.0   72     5   2       4.276666
## 3    12     149 12.6   74     5   3       4.304065
## 4    18     313 11.5   62     5   4       4.127134
## 5    NA      NA 14.3   56     5   5       4.025352
## 6    28      NA 14.9   66     5   6       4.189655
```

But suppose you wanted to make several new variables quickly. Here's one way using the 'mutate' function, also stemming from the dplyr package:

```
complete.airquality <- mutate(complete.airquality,
                              logtemperature = log(Temp),
                              logwind = log(Wind),
                              Ozonemonth = Ozone*Month,
                              Country = 'United States',
                              tempwind = logtemperature/logwind)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day logtemperature
```

```
## 1    41      190  7.4    67      5    1        4.204693
## 2    36      118  8.0    72      5    2        4.276666
## 3    12      149 12.6    74      5    3        4.304065
## 4    18      313 11.5    62      5    4        4.127134
## 5    NA       NA 14.3    56      5    5        4.025352
## 6    28       NA 14.9    66      5    6        4.189655
```

The first argument is the dataset we want to 'mutate' and every ensuing argument follows the familiar format of "name of new variable = some function of existing variables". Now we have a bunch of different variables defined accordingly. I don't think log temperature divided by log wind really means anything but that's just there to show you how flexible this command is.

Pulling regression results

Let's run a maybe meaningless regression: Ozone regressed on logtemperature and logwind on the subset of airquality with no NAs:

```
air.model <- lm_robust(Ozone ~ logtemperature + logwind, data = complete.airquality, se_type = 'stata')
summary(air.model)
```

```
##
## Call:
## lm_robust(formula = Ozone ~ logtemperature + logwind, data = complete.airquality,
##     se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##                Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper  DF
## (Intercept)     -383.68     85.591  -4.483 1.843e-05  -553.33  -214.02 108
## logtemperature   117.84     16.801   7.014 2.118e-10    84.54   151.14 108
## logwind          -38.81      8.457  -4.589 1.209e-05   -55.57   -22.04 108
##
## Multiple R-squared:  0.6195 ,    Adjusted R-squared:  0.6125
## F-statistic: 79.79 on 2 and 108 DF,  p-value: < 2.2e-16
```

Sometimes we'll want to refer to the coefficient estimate, t-statistic, or p-value associated with one of the regressors. We've done something like this before, but here's an easy way to isolate specific ones:

```
air.model$coefficients['logtemperature']
```

```
## logtemperature
##         117.84
```

```
air.model$std.error['logwind']
```

```
##  logwind
## 8.456927
```

```
air.model$statistic['(Intercept)']
```

```
## (Intercept)
##   -4.482706
```

```
air.model$fstatistic
```

```
##     value     numdf     dendf
##  79.78879   2.00000 108.00000
```

Residuals

lm_robust objects apparently are incapable of letting you pull residuals directly. Our options are usually to pull them from an equivalent non-robust lm model object or to produce them ourselves as the difference between true and predicted values. Here's an example using the same complete dataset:

```
complete.airquality <- mutate(complete.airquality,
                              ozone.predictions = air.model$fitted.values,
                              residuals = Ozone-ozone.predictions)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day logtemperature
## 1    41     190  7.4   67     5   1       4.204693
## 2    36     118  8.0   72     5   2       4.276666
## 3    12     149 12.6   74     5   3       4.304065
## 4    18     313 11.5   62     5   4       4.127134
## 5    NA      NA 14.3   56     5   5       4.025352
## 6    28      NA 14.9   66     5   6       4.189655
```

Root Mean Squared Error

Similarly, lm_robust models do not report the RMSE in their output. The RMSE is independent of the robust standard errors you use since coefficient estimates are independent of the standard errors you use, which means predicted/fitted values will be exactly the same, which means residuals will be exactly the same. All this is to say, we can use the non-robust lm model to figure out the RMSE.

T produce the RMSE, you can just extract the residuals from the non-robust model and then calculate them that way. For example, using our air.model defined above:

```
air.model2 <- lm(Ozone ~ logtemperature + logwind, data = complete.airquality) # a non-robust lm object
summary(air.model2)
```

```
##
## Call:
## lm(formula = Ozone ~ logtemperature + logwind, data = complete.airquality)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -41.939 -12.420  -2.366  11.093  81.327
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -383.679     86.396  -4.441 2.17e-05 ***
## logtemperature  117.840     18.195   6.477 2.86e-09 ***
## logwind         -38.806      5.811  -6.678 1.08e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.71 on 108 degrees of freedom
## Multiple R-squared:  0.6195, Adjusted R-squared:  0.6125
## F-statistic: 87.93 on 2 and 108 DF,  p-value: < 2.2e-16
```

You'll notice a statistic here in the third-to-last line called the "Residual standard error". This is just what R calls the RMSE. To pull it you can just run the following command:

```
sigma(air.model2)
```

```
## [1] 20.71451
```

If you wanted to compute it manually (maybe as a sanity check), we'd need to account for degrees of freedom:

```
sqrt(sum(air.model2$residuals^2)/df.residual(air.model2))
```

```
## [1] 20.71451
```

Joint hypothesis tests

We know how to conduct tests of significance for individual coefficients: just look at the t-statistic or p-value or confidence interval for a particular regressor. We'll also be interested in joint hypothesis tests, which test hypotheses of the following variety:

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

It'll come up in this week's problem set and definitely several future ones as well so do install the *car* package. In particular, we'll want to use the function *linearHypothesis*.

Suppose we wanted to test whether logtemperature and logwind are jointly significant as regressors in the above regression. We'd implement that as follows:

```
linearHypothesis(air.model, c('logtemperature = 0', 'logwind = 0'))
```

```
## Linear hypothesis test
##
## Hypothesis:
## logtemperature = 0
## logwind = 0
##
```

```
## Model 1: restricted model
## Model 2: Ozone ~ logtemperature + logwind
##
##   Res.Df Df  Chisq Pr(>Chisq)
## 1    110
## 2    108  2 159.58  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This gives us a Chi-squared statistic of 165.73 with 113 degrees of freedom and a p-value very close to zero.

We can also specify it to conduct an F-test instead:

```r
linearHypothesis(air.model, c('logtemperature = 0', 'logwind = 0'), test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## logtemperature = 0
## logwind = 0
##
## Model 1: restricted model
## Model 2: Ozone ~ logtemperature + logwind
##
##   Res.Df Df      F    Pr(>F)
## 1    110
## 2    108  2 79.789 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

which gives us an F-statistic of 82.865. These are still equivalent tests as you can tell from both tests producing the exact same p-value.

Finally, we might have a model that assumes homoskedastic errors:

```r
air.model2 <- lm(Ozone ~ logtemperature + logwind, data = complete.airquality)
```

This will give us non robust standard errors and the test statistics will come out slightly different:

```r
linearHypothesis(air.model2, c('logtemperature = 0', 'logwind = 0'), test = 'F')
```

```
## Linear hypothesis test
##
## Hypothesis:
## logtemperature = 0
## logwind = 0
##
## Model 1: restricted model
## Model 2: Ozone ~ logtemperature + logwind
##
##   Res.Df    RSS Df Sum of Sq     F    Pr(>F)
## 1    110 121802
## 2    108  46342  2     75460 87.93 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-statistic is slightly different (though not that different). If we wanted to recover the original result, we could just add another argument to the linearHypothesis function to tell it to use the robust standard errors (i.e. HC1, White-adjusted standard errors):

```
linearHypothesis(air.model2, c('logtemperature = 0', 'logwind = 0'), white.adjust = 'hc1')
```

```
## Linear hypothesis test
##
## Hypothesis:
## logtemperature = 0
## logwind = 0
##
## Model 1: restricted model
## Model 2: Ozone ~ logtemperature + logwind
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    110
## 2    108  2 79.789 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

HC1 is the technical name of the standard errors Stata uses. You can even use se_type='HC1' instead of se_type='stata' as an argument in our lm_robust command, they'll come out the same.

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.

Non-Practice Problem: Stock-Watson Empirical Exercise 5.1

```
earnings <- read.dta13('Earnings_and_Height.dta')
```

```
## Warning in read.dta13("Earnings_and_Height.dta"):
##    Factor codes of type double or float detected in variables
##
##    race
##
##    No labels have been assigned.
##    Set option 'nonint.factors = TRUE' to assign labels anyway.
```

```
## Warning in read.dta13("Earnings_and_Height.dta"):
##    Missing factor labels for variables
##
##    age
##
##    No labels have beend assigned.
##    Set option 'generate.factors=TRUE' to generate labels.
```

```r
summary(earnings)
```

```
##        sex             age                                     mrd       
##  0:female:9974   Min.   :25.00   0:< 14 yrs                 :    0  
##  1:male  :7896   1st Qu.:33.00   1:Married, sps in hh     :11422  
##                  Median :40.00   2:Married, sps not in hh:  219  
##                  Mean   :40.92   3:Widowed                  :  432  
##                  3rd Qu.:48.00   3:Divorced                 : 2582  
##                  Max.   :65.00   5:Separated                :  572  
##                                  6:Never Married            : 2643  
##       educ                       cworker              region          race      
##  Min.   : 0.00   1:Private company :12475   1:Northeast:3636   Min.   :1.000  
##  1st Qu.:12.00   4:Local Govt emp  : 1913   2:Midwest  :4593   1st Qu.:1.000  
##  Median :13.00   6:Self-employed   : 1487   3:South    :5794   Median :1.000  
##  Mean   :13.54   3:State Govt emp  :  984   4:West     :3847   Mean   :1.386  
##  3rd Qu.:16.00   2:Fed Govt emp    :  656                      3rd Qu.:1.000  
##  Max.   :19.00   5:Incorporated bus:  355                      Max.   :4.000  
##                  (Other)           :    0                                     
##     earnings          height          weight         occupation    
##  Min.   : 4726   Min.   :48.00   Min.   : 80.0   Min.   : 1.000  
##  1st Qu.:23363   1st Qu.:64.00   1st Qu.:140.0   1st Qu.: 2.000  
##  Median :38925   Median :67.00   Median :163.0   Median : 5.000  
##  Mean   :46875   Mean   :66.96   Mean   :170.4   Mean   : 6.011  
##  3rd Qu.:84055   3rd Qu.:70.00   3rd Qu.:190.0   3rd Qu.: 8.000  
##  Max.   :84055   Max.   :84.00   Max.   :501.0   Max.   :15.000  
## 
```

```r
summary(earnings$sex)
```

```
## 0:female   1:male 
##     9974     7896
```

The sex variable is in an inconvenient form so let's just make it a binary 0 and 1, where 1 indicates a male observation:

```r
earnings$sex <- earnings$sex=='1:male'
summary(earnings$sex)
```

```
##    Mode   FALSE    TRUE 
## logical    9974    7896
```

The logic in the first line here is that the right-hand side of the assignment is a statement. If an observation for sex was equal to ("==") the character string '1:male', the statement is "TRUE" which is coded as a 1 in most programming languages (including Stata and R). Otherwise, it is given a 0.

Part a: Regress earnings on height

```r
all.mod <- lm_robust(earnings ~ height, data = earnings, se_type = 'stata')
summary(all.mod)
```

```
##
## Call:
## lm_robust(formula = earnings ~ height, data = earnings, se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper    DF
## (Intercept)   -512.7     3379.9 -0.1517 8.794e-01  -7137.6   6112.1 17868
## height         707.7       50.4 14.0425 1.478e-44    608.9    806.5 17868
##
## Multiple R-squared:  0.01088 ,   Adjusted R-squared:  0.01082
## F-statistic: 197.2 on 1 and 17868 DF,  p-value: < 2.2e-16
```

The p-value on the height variable is 1.478e-44 (which means 1.478×10^{-44}), which we may just as well consider zero. This slope coefficient is thus statistically significant at any meaningful power level.

This regression output also gives you the 95% confidence interval by default: 608.9 to 806.5. If we wanted to, we could also compute the confidence interval for any degree of confidence. For example, 99%:

```
confint(all.mod, level = 0.99)
```

```
##                  0.5 %    99.5 %
## (Intercept) -9219.6158 8194.1487
## height        577.8487  837.4944
```

Part b: Same regression but on the subsample for women

```
women.mod <- lm_robust(earnings ~ height,
                       data = filter(earnings, sex ==0),
                       se_type = 'stata')
summary(women.mod)
```

```
##
## Call:
## lm_robust(formula = earnings ~ height, data = filter(earnings,
##     sex == 0), se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)  12650.9    6299.15   2.008 4.463e-02    303.2  24998.5 9972
## height         511.2      97.58   5.239 1.650e-07    319.9    702.5 9972
##
## Multiple R-squared:  0.002672 ,  Adjusted R-squared:  0.002572
## F-statistic: 27.44 on 1 and 9972 DF,  p-value: 1.65e-07
```

Again, the p-value is pretty much zero so the relationship is statistically significant. The confidence interval is now 319.9-702.5

Part c: Same regression but on the subsample for men

```
men.mod <- lm_robust(earnings ~ height,
                     data = filter(earnings, sex == 1),
                     se_type = 'stata')
summary(men.mod)
```

```
##
## Call:
## lm_robust(formula = earnings ~ height, data = filter(earnings,
##     sex == 1), se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)   -43130    6925.01  -6.228 4.960e-10   -56705   -29555 7894
## height          1307      98.86  13.220 1.771e-39     1113     1501 7894
##
## Multiple R-squared:  0.02086 ,   Adjusted R-squared:  0.02074
## F-statistic: 174.8 on 1 and 7894 DF,  p-value: < 2.2e-16
```

Again, the p-value is pretty much zero so the relationship is statistically significant. The confidence interval is now 1113-1501.

Part d: Test the null hypothesis that the effect of height on earnings is the same for men and women

See iPad notes

Practice Problem 1: Stock-Watson Empirical Exercise 5.2

```
growth <- read.dta13('Growth.dta')
growth.model <- lm_robust(growth ~ tradeshare,
                          data = filter(growth, tradeshare < 1.5),
                          se_type = 'stata')
summary(growth.model)
```

```
##
## Call:
## lm_robust(formula = growth ~ tradeshare, data = filter(growth,
##     tradeshare < 1.5), se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|) CI Lower CI Upper DF
## (Intercept)   0.9574     0.5361   1.786  0.07899 -0.11415    2.029 62
```

```
## tradeshare    1.6809     0.8656   1.942  0.05670 -0.04944    3.411 62
##
## Multiple R-squared:  0.04466 ,   Adjusted R-squared:  0.02925
## F-statistic: 3.771 on 1 and 62 DF,  p-value: 0.0567
```

Part a

The p-value on tradeshare is 0.0567. This measn we can reject the null hypothesis $H_0 : \beta_1 = 0$ vs. a two-sided alternative hypothesis at the 10% level, but not at either a 5% or 1% significance level, at least using robust standard errors.

Part b

The p-value associated with the coefficient's t-statistic is 0.0567, as mentioned above.

Part c

```
confint(growth.model, level = 0.9)
```

```
##                     5 %     95 %
## (Intercept) 0.06229901 1.852522
## tradeshare  0.23549365 3.126316
```

The 90% confidence interval is reported in the regression output is (0.235, 3.13)

Practice Problem 2: Stock-Watson Empirical Exercise 5.3

```
smoking <- read.dta13('birthweight_smoking.dta')
head(smoking)
```

```
##   nprevist alcohol tripre1 tripre2 tripre3 tripre0 birthweight smoker unmarried
## 1       12       0       1       0       0       0        4253      1         1
## 2        5       0       0       1       0       0        3459      0         0
## 3       12       0       1       0       0       0        2920      1         0
## 4       13       0       1       0       0       0        2600      0         0
## 5        9       0       1       0       0       0        3742      0         0
## 6       11       0       1       0       0       0        3420      0         0
##   educ age drinks
## 1   12  27      0
## 2   16  24      0
## 3   11  23      0
## 4   17  28      0
## 5   13  27      0
## 6   16  33      0
```

Part a

```
# Part a-i
mean(smoking$birthweight)
```

```
## [1] 3382.934
```

```
# Part a-ii and a-iii
# Method 1
filter(smoking, smoker == 1) %$%
  birthweight %>%
  mean
```

```
## [1] 3178.832
```

```
filter(smoking, smoker == 0) %$%
  birthweight %>%
  mean
```

```
## [1] 3432.06
```

```
# Method 2
t.test(birthweight ~ smoker, data = smoking)
```

```
##
##  Welch Two Sample t-test
##
## data:  birthweight by smoker
## t = 9.4414, df = 887.15, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  200.5882 305.8685
## sample estimates:
## mean in group 0 mean in group 1
##        3432.060        3178.832
```

Part b

We can use the same t-test function to answer part b-i.

```
weight.test <- t.test(birthweight ~ smoker, data = smoking)
# Differences
diff(weight.test$estimate)
```

```
## mean in group 1
##       -253.2284
```

```r
# Standard error of difference
weight.test$stderr
```

```
## [1] 26.82106
```

```r
# 95% confidence interval on differences
weight.test$conf.int
```

```
## [1] 200.5882 305.8685
## attr(,"conf.level")
## [1] 0.95
```

Part c

```r
weight.model <- lm_robust(birthweight ~ smoker, data = smoking)
summary(weight.model)
```

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker, data = smoking)
##
## Standard error type:  HC2
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper   DF
## (Intercept)   3432.1      11.89 288.675 0.000e+00   3408.7   3455.4 2998
## smoker        -253.2      26.82  -9.441 7.148e-21   -305.8   -200.6 2998
##
## Multiple R-squared:  0.0286 ,    Adjusted R-squared:  0.02828
## F-statistic: 89.14 on 1 and 2998 DF,  p-value: < 2.2e-16
```

Part c-i

The intercept is the average birthweight for non-smokers

The slope is the difference between average birthweights for smokers and non-smokers

Part c-ii

The standard errors are the same (if we use the robust standard errors)

Part c-iii

```r
confint(weight.model)
```

```
##                 2.5 %    97.5 %
## (Intercept) 3408.7485 3455.3714
## smoker      -305.8179 -200.6388
```

Practice Problem 3: Stock-Watson Empirical Exercise 6.1 (a,b,d only)

```
smoking <- read.dta13('birthweight_smoking.dta')
```

Part a

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

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker, data = smoking, se_type = "stata")
##
## 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 = 'stata'
summary(birth.model.b)
```

```
##
## Call:
## lm_robust(formula = birthweight ~ smoker + alcohol + nprevist,
##     data = smoking, se_type = "stata")
##
## 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)

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

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

(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

(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.

Practice Problem 4: Stock-Watson Empirical Exercise 6.2

```
growth <- read.dta13('Growth.dta') %>%
  filter(tradeshare < 1.5)
```

Part a

```
summary(growth)
```

```
##  country_name           growth             oil        rgdp60
##  Length:64          Min.   :-2.8119   Min.   :0   Min.   : 367
##  Class :character   1st Qu.: 0.8057   1st Qu.:0   1st Qu.:1144
##  Mode  :character   Median : 1.9745   Median :0   Median :2028
##                     Mean   : 1.8691   Mean   :0   Mean   :3131
##                     3rd Qu.: 2.8283   3rd Qu.:0   3rd Qu.:5180
##                     Max.   : 7.1569   Max.   :0   Max.   :9895
##    tradeshare       yearsschool       rev_coups       assasinations
##  Min.   :0.1405   Min.   : 0.200   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.3847   1st Qu.: 1.880   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.5390   Median : 3.550   Median :0.08333   Median :0.1000
##  Mean   :0.5424   Mean   : 3.959   Mean   :0.17007   Mean   :0.2819
##  3rd Qu.:0.6588   3rd Qu.: 5.343   3rd Qu.:0.26667   3rd Qu.:0.2333
##  Max.   :1.1279   Max.   :10.070   Max.   :0.97037   Max.   :2.4667
```

Part b

```
growth.model <- lm_robust(growth ~ tradeshare + yearsschool + rev_coups + assasinations + rgdp60, data =
summary(growth.model)
```

```
##
## Call:
## lm_robust(formula = growth ~ tradeshare + yearsschool + rev_coups +
##     assasinations + rgdp60, data = growth, se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##                 Estimate Std. Error t value  Pr(>|t|)   CI Lower   CI Upper DF
## (Intercept)    0.6268915  0.8690927  0.7213 4.736e-01 -1.1127867  2.3665696 58
## tradeshare     1.3408193  0.8819886  1.5202 1.339e-01 -0.4246728  3.1063114 58
## yearsschool    0.5642445  0.1294907  4.3574 5.442e-05  0.3050408  0.8234482 58
## rev_coups     -2.1504256  0.8746010 -2.4588 1.695e-02 -3.9011297 -0.3997214 58
## assasinations  0.3225844  0.3803478  0.8481 3.999e-01 -0.4387644  1.0839333 58
## rgdp60        -0.0004613  0.0001215 -3.7968 3.529e-04 -0.0007045 -0.0002181 58
##
## Multiple R-squared:  0.2911 ,    Adjusted R-squared:   0.23
## F-statistic: 7.001 on 5 and 58 DF,  p-value: 3.54e-05
```

The coefficient on Rev_Coups is -2.15. An additional coup in a five year period, reduces the average year growth rate by $(2.15/5) = 0.43\%$ over this 25 year period. This means the GDP in 1995 is expected to be approximately $.43 \times 25 = 10.75\%$ lower. This is a large effect.

Part c

Obviously, you can take the mean values from part a and manually multiply them by the regression estimates to produce a prediction. Here's a semi-advanced way of doing that in one line. Don't worry if you don't understand what it's doing, I was just lazy. We'll learn more about the predict function here next week.

```
mean.vals <- sapply(growth[,-1], FUN = mean, na.rm = T) %>% t %>% data.frame
predict(growth.model, mean.vals)
```

```
##       1
## 1.86912
```

Part d

```
mean.vals$tradeshare <- mean.vals$tradeshare + sd(growth$tradeshare)
predict(growth.model, data.frame(mean.vals))
```

```
##        1
## 2.175273
```

Part e

The variable "oil" takes on the value of 0 for all 64 countries in the sample. This would generate perfect multicollinearity since the variable is a linear combination of one of the regressors, namely the constant.