R-work for Online Assessment

Instructions

You should work through the code below and complete it. Keep the completed code and all the resulting output. Next you should answer the questions in the online quiz. Every student will see a slightly different collection of questions (as we will randomly draw 10 questions from a pool of about 20 questions).

The questions are of four types.

1) Questions that merely ask you to report output from your analysis.
2) Some questions will ask you about R code. For example, you will see a lot of gaps (`XXXX`) in the code and questions may ask you how to complete the code to make the code work. Sometimes the `XXXX` will represent one word and on other occasions it will represent a full line (or two) of code. Other questions may ask you about the output to be produced by a particular bit of code. If you want to practice these sorts of questions you could practice on Datacamp.
3) The third type of questions will test your understanding of econometric issues. For example: "What is the meaning of an estimated coefficient?" "Is a particular coefficient statistically significant?"
4) The fourth type of question, if asked, will be on general programming issues. For example: what is the meaning of a particular error message, or, how would you search for a particular piece of information.

Preparing your workfile

We add the basic libraries needed for this week's work:

```r
library(tidyverse)    # for almost all data handling tasks
library(ggplot2)      # to produce nice graphiscs
library(stargazer)    # to produce nice results tables
library(haven)        # to import stata file
library(AER)          # access to HS robust standard errors
source("stargazer_HC.r")  # includes the robust regression display
```

Introduction

The data are an extract from the Understanding Society Survey (formerly the British Household Survey Panel).

Data Upload - and understanding data structure

Upload the data, which are saved in a STATA datafile (extension `.dta`). There is a function which loads STATA file. It is called `read_dta` and is supplied by the `haven` package.

```
data_USoc <- XXXX("20222_USoc_extract.dta")
data_USoc <- as.data.frame(XXXX)    # ensure data frame structure
names(XXXX)
```

```
data_USoc <- read_dta("20222_USoc_extract.dta")
data_USoc <- as.data.frame(data_USoc)    # ensure data frame structure
names(data_USoc)
```

```
##  [1] "pidp"    "age"     "jbhrs"   "paygu"   "wave"    "cpi"     "year"
##  [8] "region"  "urate"   "male"    "race"    "educ"    "degree"  "mfsize9"
```

Let us ensure that categorical variables are stored as `factor` variables. It is easiest to work with these in R.

```
data_USoc$region <- XXXX
data_USoc$male <- XXXX
data_USoc$degree <- XXXX
data_USoc$race <- XXXX
```

```
data_USoc$region <- as_factor(data_USoc$region)
data_USoc$male <- as_factor(data_USoc$male)
data_USoc$degree <- as_factor(data_USoc$degree)
data_USoc$race <- as_factor(data_USoc$race)
```

Check for which regions we have data. Either use the `levels` or the `unique` function.

```
levels(data_USoc$region)
```

```
##  [1] "north east"               "north west"
##  [3] "yorkshire and the humber" "east midlands"
##  [5] "west midlands"            "east of england"
##  [7] "london"                   "south east"
##  [9] "south west"               "wales"
## [11] "scotland"                 "northern ireland"
```

```
unique(data_USoc$region)
```

```
##  [1] south east               east midlands            north east
##  [4] north west               scotland                 wales
##  [7] east of england          northern ireland         london
## [10] west midlands            yorkshire and the humber south west
## 12 Levels: north east north west yorkshire and the humber ... northern ireland
```

Now check which `race` and `degree` categories exist in the data.

```
levels(data_USoc$degree)
```

```
## [1] "no degree"     "first degree"  "higher degree"
```

```
levels(data_USoc$race)
```

```
## [1] "white" "mixed" "asian" "black" "other"
```

As we defined the `male` variable as a factor it has levels `male` and `female` (check `levels(data_USoc$male)` to confirm). It would be better to relabel the variable to `gender`.

```
names(data_USoc)[names(data_USoc) == "male"] <- "gender"
```

The pay information (`paygu`) is provided as a measure of the (usual) gross pay per month. As workers work for varying numbers of hours per week (`jbhrs`) we divide the monthly pay by the approximate monthly hours (`4*jbhrs`). We shall also adjust for increasing price levels (as measured by `cpi`). These two adjustments leave us with an inflation adjusted hourly wage. We call this variable `hrpay` and also calculate the natural log of this variable (`lnhrpay`).

```
data_USoc <- data_USoc %>%
              XXXX(hrpay = XXXX/(cpi/100)) %>%
              XXXX(lnhrpay = XXXX)
```

```
data_USoc <- data_USoc %>%
              mutate(hrpay = paygu/(jbhrs*4)/(cpi/100)) %>%
              mutate(lnhrpay = log(hrpay))
```

As we wanted to save these additional variables we assign the result of the operation to `data_USoc`.

We also want to use a measure of annual pay (`paygu*12/(cpi/100))`) and add this variable (`annualpay`) to the dataframe (`data_USoc`). Also add the log of this variable as a variable to the dataframe and call it `lnannualpay`.

```
data_USoc <- XXXX %>%
              XXXX
```

```
data_USoc <- data_USoc %>%
              mutate(annualpay = paygu*12/(cpi/100))%>%
              mutate(lnannualpay = log(annualpay))
```

Let's first summarise all numerical variables in our dataset, using the `stargazer` function.

```
XXXX
```

```
stargazer(data_USoc,type="latex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Feb 21, 2020 - 16:44:26

Table 1:

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
pidp	133,272	839,218,358.000	467,699,610.000	280,165	410,528,927	1,225,328,047	1,639,568,724
age	133,272	46.172	18.295	9	31	60	103
jbhrs	64,217	32.594	11.614	0.100	25.000	40.000	97.000
paygu	59,216	1,823.574	1,475.064	0.083	850.000	2,400.000	15,000.000
wave	133,272	1.912	0.818	1	1	3	3
cpi	133,272	116.790	4.199	110.800	114.500	119.600	126.100
year	133,272	2,010.453	0.991	2,009	2,010	2,011	2,013
urate	133,272	7.955	1.311	5.800	6.700	9.100	10.800
educ	133,041	12.838	2.316	11.000	11.000	15.000	17.000
mfsize9	58,989	303.135	484.430	1.000	17.000	350.000	1,500.000
hrpay	58,960	12.268	45.140	0.0004	6.612	14.518	7,104.150
lnhrpay	58,960	2.283	0.631	−7.816	1.889	2.675	8.868
annualpay	59,216	18,761.600	15,185.830	0.813	8,695.652	24,665.560	162,454.900
lnannualpay	59,216	9.524	0.883	−0.207	9.071	10.113	11.998

You should find, for instance, that the mean value of the unemployment rate (`urate`) is 7.955 and the standard deviation for the `age` variable is 18.295.

For later purposes we will also need variables $age^2/100$ and $log(age)$. We now need to create these variables (`agesq` and `lnage`) and add them to the `data_USoc` dataframe.

```
XXXX
mean(data_USoc$lnage)
```

```
data_USoc <- data_USoc %>%
                mutate(agesq = age*age/100) %>%
                mutate(lnage = log(age))
mean(data_USoc$lnage)
```

[1] 3.744307

You should find the mean of `lnage` to be 3.744307.

Another variable needed later is a variable which indicates whether a respondent has a degree. We call this variable `grad`. It should be a factor variable with two levels, `degree` and `no degree`.

```
data_USoc <- data_USoc %>%
              mutate(grad = ifelse(degree %in% c("first degree","higher degree"),"degree",
                                   ifelse(degree == "no degree","no degree",NA)))
data_USoc$grad <- as_factor(data_USoc$grad)
```

Google to understand what the `ifelse()` function does.

Let's find out how many observations we have for some of our categorical variables.

```
table(data_USoc$degree)
```

```
##
##     no degree  first degree higher degree
##        102644          17509         12888
```

We can also create a frequency table for two variables, here `degree` as the row variable and `gender` as the column variable.

```
table(data_USoc$degree, data_USoc$gender)
```

```
##
##                   female  male
##   no degree        56475 46169
##   first degree      9464  8045
##   higher degree     6037  6851
```

Create a frequence table with `region` as the row variable and `race` as the column variable.

```
XXXX(data_USoc$XXXX, XXXX)
```

```
table(data_USoc$region, data_USoc$race)
```

```
                       white mixed asian black other
```

north east 4666 18 167 26 23 north west 11498 152 1144 267 120 yorkshire and the humber 8469 130 1159 206 141 east midlands 8086 146 902 211 106 west midlands 8094 214 1852 620 147 east of england 9556 141 736 321 153 london 6656 847 5772 4004 939 south east 13634 242 761 330 186 south west 9374 80 144 91 68 wales 5804 22 167 45 89 scotland 8366 42 156 42 70 northern ireland 5390 23 34 4 36

You want to see this information in proportions (by region). Run the following code:

```
options(digits = 2)
prop.table(table(region, race),margin = 1)
```

You should receive an error message. " object ‘region’ not found“. Fix the code!

```
options(digits = 2)
prop.table(table(data_USoc$region, data_USoc$race),margin = 1)
```

```
                                white   mixed   asian   black   other
```

north east 0.95224 0.00367 0.03408 0.00531 0.00469 north west 0.87232 0.01153 0.08679 0.02026 0.00910 yorkshire and the humber 0.83810 0.01286 0.11470 0.02039 0.01395 east midlands 0.85557 0.01545 0.09544 0.02233 0.01122 west midlands 0.74073 0.01958 0.16949 0.05674 0.01345 east of england 0.87613 0.01293 0.06748 0.02943 0.01403 london 0.36535 0.04649 0.31683 0.21978 0.05154 south east 0.89976 0.01597 0.05022 0.02178 0.01227 south west 0.96075 0.00820 0.01476 0.00933 0.00697 wales 0.94728 0.00359 0.02726 0.00734 0.01453 scotland 0.96427 0.00484 0.01798 0.00484 0.00807 northern ireland 0.98232 0.00419 0.00620 0.00073 0.00656

Data cleaning

We now remove (or "drop") unusable (or "missing") observations from our `data_USoc` dataframe. They are those observations which have missing (`NA`) data for `lnhrpay` (because the individual is not working) and we will remove observations for males who are 66 years or older and females who are 61 years or older.

```
data_USoc <- data_USoc %>%
                filter(!XXXX(lnhrpay)) %>%
                filter((gender == XXXX & age < 66) | (gender == "female" & XXXX < XXXX))
```

```
data_USoc <- data_USoc %>%
                filter(!is.na(lnhrpay)) %>%
                filter((gender == "male" & age < 66) | (gender == "female" & age < 61) )
```

You should end up with 56778 observations.

Estimate regression models - Version 1

We shall estimate the following regression models (`mod1`)

$$lnhrpay = \beta_0 + \beta_1\ age + \beta_2\ agesq + u$$

and (`mod2`)

$$lnhrpay = \alpha_0 + \alpha_1\ lnage + u$$

```
mod1 <- lm(XXXX ~ XXXX+XXXX, data = data_USoc)
mod2 <- lm(XXXX)
stargazer_HC(mod1,mod2)
```

```
mod1 <- lm(lnhrpay ~ age+agesq, data = data_USoc)
mod2 <- lm(lnhrpay ~ lnage, data = data_USoc)
stargazer_HC(mod1,mod2,type_out="latex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Feb 21, 2020 - 16:44:29

If you have done this correctly, you will find that that your estimated constant for `mod1` is 0.485.

Table 2:

	Dependent variable:	
	lnhrpay	
	(1)	(2)
age	0.087***	
	(0.001)	
agesq	−0.096***	
	(0.002)	
lnage		0.480***
		(0.008)
Constant	0.480***	0.530***
	(0.026)	(0.028)
Observations	56,778	56,778
R^2	0.098	0.066
Adjusted R^2	0.098	0.066
Residual Std. Error	0.590 (df = 56775)	0.600 (df = 56776)
F Statistic	3,101.000*** (df = 2; 56775)	3,984.000*** (df = 1; 56776)
Note:		*p<0.1; **p<0.05; ***p<0.01 Robust standard errors in parenthesis

We suggest that there are two ways of modelling the relationship between `age` and `lnhrpay` in a so-called parametric way, either as a quadratic relationship or as a logarithmic one.

As we have lots of data, there is a third more flexible approach. We do this be generating a dummy variable for every integer age (ages are reported in full years only). To do this we will first have to create an age variable which treats age as a categorical, or in R terms, a factor variable. We shall call this `age_f`.

```
data_USoc <- data_USoc %>% mutate(age_f = as.factor(age))
```

With `age_f` being a factor variable, it is now straightforward to include this factor variable into a regresion. We can either include a constant (`lnhrpay ~ age_f`) which will then use age = 16 as a base category, or we can estimate the model without a constant (`lnhrpay ~ age_f - 1`) in which case all age categories enter separately.

```
mod3 <- lm(lnhrpay ~ age_f, data = data_USoc)
mod4 <- lm(lnhrpay ~ age_f -1, data = data_USoc)
stargazer_HC(mod3,mod4)
```

```
mod3 <- lm(lnhrpay ~ age_f, data = data_USoc)
mod4 <- lm(lnhrpay ~ age_f -1, data = data_USoc)
stargazer_HC(mod3,mod4,type_out="latex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Feb 21, 2020 - 16:44:32

We now compare the fitted values for `mod1`, `mod2` and `mod4`. First we add the predicted values to the dataframe. There are several ways to achieve this and I recommend you ask Dr. Google. (Think carefully about the search terms.)

Table 3:

	Dependent variable:	
	lnhrpay	
	(1)	(2)
age_f16		1.500***
		(0.033)
age_f17	0.040	1.500***
	(0.041)	(0.024)
age_f18	0.140***	1.600***
	(0.038)	(0.019)
age_f19	0.230***	1.700***
	(0.039)	(0.021)
age_f20	0.330***	1.800***
	(0.038)	(0.019)
age_f21	0.410***	1.900***
	(0.038)	(0.018)
age_f22	0.470***	1.900***
	(0.037)	(0.017)
age_f23	0.530***	2.000***
	(0.036)	(0.014)
age_f24	0.600***	2.100***
	(0.036)	(0.012)
age_f25	0.660***	2.100***
	(0.037)	(0.015)
age_f26	0.700***	2.200***
	(0.036)	(0.015)
age_f27	0.750***	2.200***
	(0.036)	(0.014)
age_f28	0.790***	2.200***
	(0.036)	(0.015)
age_f29	0.830***	2.300***
	(0.037)	(0.016)
age_f30	0.880***	2.300***
	(0.036)	(0.014)
age_f31	0.900***	2.400***
	(0.037)	(0.015)
age_f32	0.920***	2.400***
	(0.036)	(0.014)
age_f33	0.940***	2.400***
	(0.037)	(0.015)

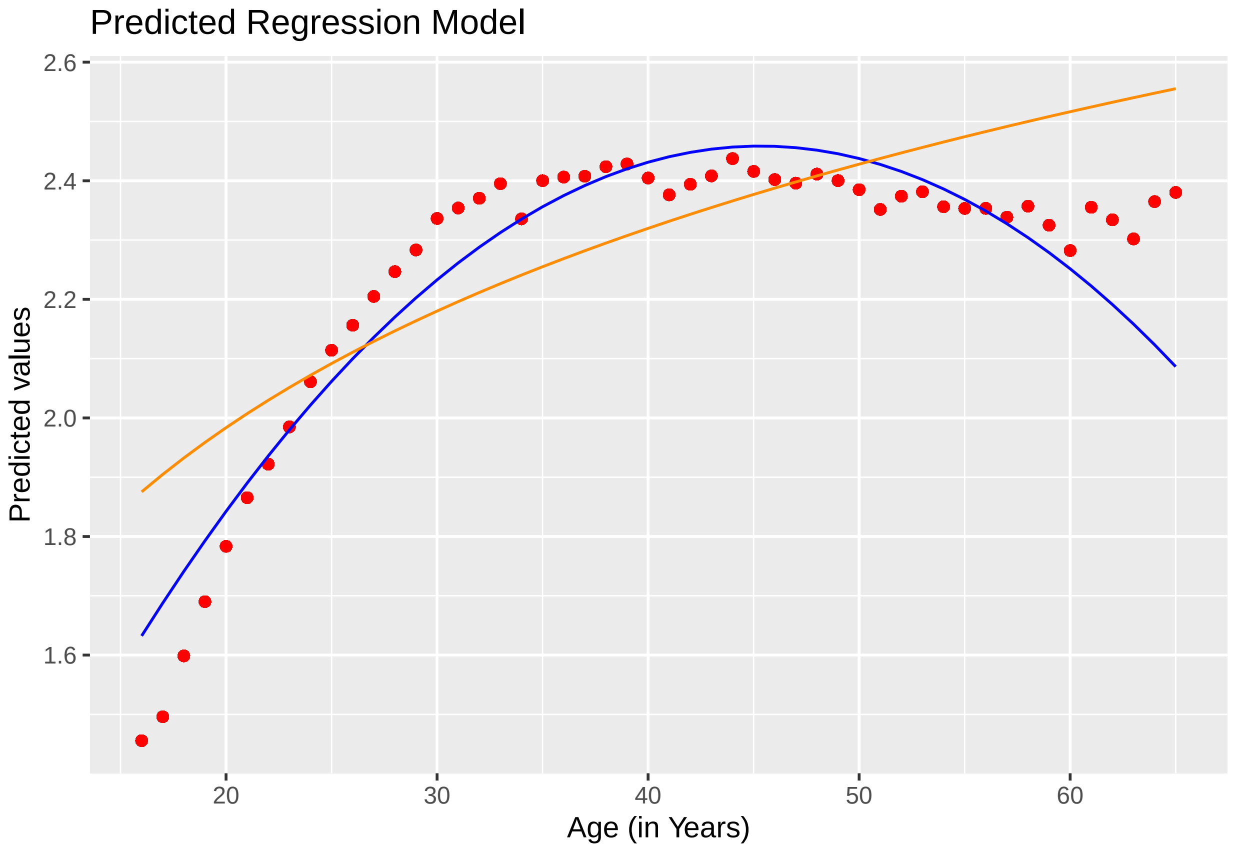
```
data_USoc$pred_mod1 <- XXXX
data_USoc$pred_mod2 <- XXXX
data_USoc$pred_mod4 <- XXXX
```

```
data_USoc$pred_mod1 <- mod1$fitted.values
data_USoc$pred_mod2 <- mod2$fitted.values
data_USoc$pred_mod4 <- mod4$fitted.values
```

Now we plot the predicted values for the three specifications. You should also change the Axis labels to "Predicted values" for the vertical axis, "Age (in Years)" for the horizontal axis and add a title ("Predicted Regression Model") to your picture. If you google you should find the appropriate commands. (Again, think carefully about the search terms.)

```
ggplot(data_USoc, aes(x=age,y=pred_mod4)) +
  geom_point(color = "red") +
  geom_line(aes(y=pred_mod1),color = "blue") +
  geom_line(aes(y=pred_mod2),color = "darkorange") +
  XXXX +    # add code to give your plot a title
  XXXX      # add code to change the axis labels
```

```
ggplot(data_USoc, aes(x=age,y=pred_mod4)) +
  geom_point(color = "red") +
  geom_line(aes(y=pred_mod1),color = "blue") +
  geom_line(aes(y=pred_mod2),color = "darkorange") +
  ggtitle("Predicted Regression Model") +
  ylab("Predicted values") +
  xlab("Age (in Years)")
```



The fit of `mod4` is the most flexible specification as it uses a coefficient for each year. Specifications `mod1` and `mod2` models model the relationship between `age` and `lnhrpay` with one and two parameters respectively.

Estimate regression models 2

Now we will estimate a quadratic model for `annualpay` (`annualpay ~ age + agesq`) on a subsets of data in order to compare these. When you know that you will be working with different subsets of data, the best way of doing that in R is to create a new factor variale (here `subset_ind`) which allows you to separate the data accordingly.

We will create two subgroups: 1) Males with a degree and 2) Males with no degree. You may want to check the values of the `grad` variable in order to define these correctly.

```
data_USoc$subset_ind <- "none"   # default group
data_USoc$subset_ind[data_USoc$gender == "male" & data_USoc$grad == "degree"] <- "Male with degree"
data_USoc$subset_ind[XXXX] <- "Male without degree"  # select all males with no degree
data_USoc$subset_ind <- as.factor(data_USoc$subset_ind)
table(data_USoc$subset_ind)
```

```
data_USoc$subset_ind <- "none"   # default group
data_USoc$subset_ind[data_USoc$gender == "male" & data_USoc$grad == "degree"] <- "Male with degree"
data_USoc$subset_ind[data_USoc$gender == "male" & data_USoc$grad == "no degree"] <- "Male without degre
data_USoc$subset_ind <- as.factor(data_USoc$subset_ind)
table(data_USoc$subset_ind)
```

Male with degree Male without degree none 8200 17626 30952

We will want to save the model predictions and for this purpose we pre-define a variable in which we will save the predictions.

```
data_USoc$pred_mod5 <- 0    # set the prediction to 0 by default
```

Now we estimate the model for the "Male with degree" subgroup. Note that the `lm` function accepts a `subset` argument which allows you to select a subset of observations, such as the group of all males with first degree.

```
mod5_md <- lm(XXXX ~ XXXX + XXXX, data = XXXX, subset = (subset_ind == XXXX))
stargazer_HC(XXXX)
data_USoc$pred_mod5[data_USoc$subset_ind==XXXX] <- mod5_md$fitted.values
```

```
mod5_md <- lm(annualpay ~ age + agesq, data = data_USoc, subset = (subset_ind == "Male with degree"))
stargazer_HC(mod5_md,type_out="latex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Feb 21, 2020 - 16:44:36

```
data_USoc$pred_mod5[data_USoc$subset_ind=="Male with degree"] <- mod5_md$fitted.values
```

Now we repeat the same just for the subgroup "Males without degree""

```
mod5_mnd <- XXXX
stargazer_HC(XXXX)
data_USoc$pred_mod5[XXXX] <- mod5_mnd$fitted.values
```

```
mod5_mnd <- lm(annualpay ~ age + agesq, data = data_USoc, subset = (subset_ind == "Male without degree"
stargazer_HC(mod5_mnd,type_out="latex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Feb 21, 2020 - 16:44:36

Table 4:

	Dependent variable:
	annualpay
age	3,931.000***
	(148.000)
agesq	$-$4,171.000***
	(177.000)
Constant	$-$53,843.000***
	(2,929.000)
Observations	8,200
R^2	0.120
Adjusted R^2	0.120
Residual Std. Error	20,125.000 (df = 8197)
F Statistic	559.000*** (df = 2; 8197)
Note:	*p<0.1; **p<0.05; ***p<0.01
	Robust standard errors in parenthesis

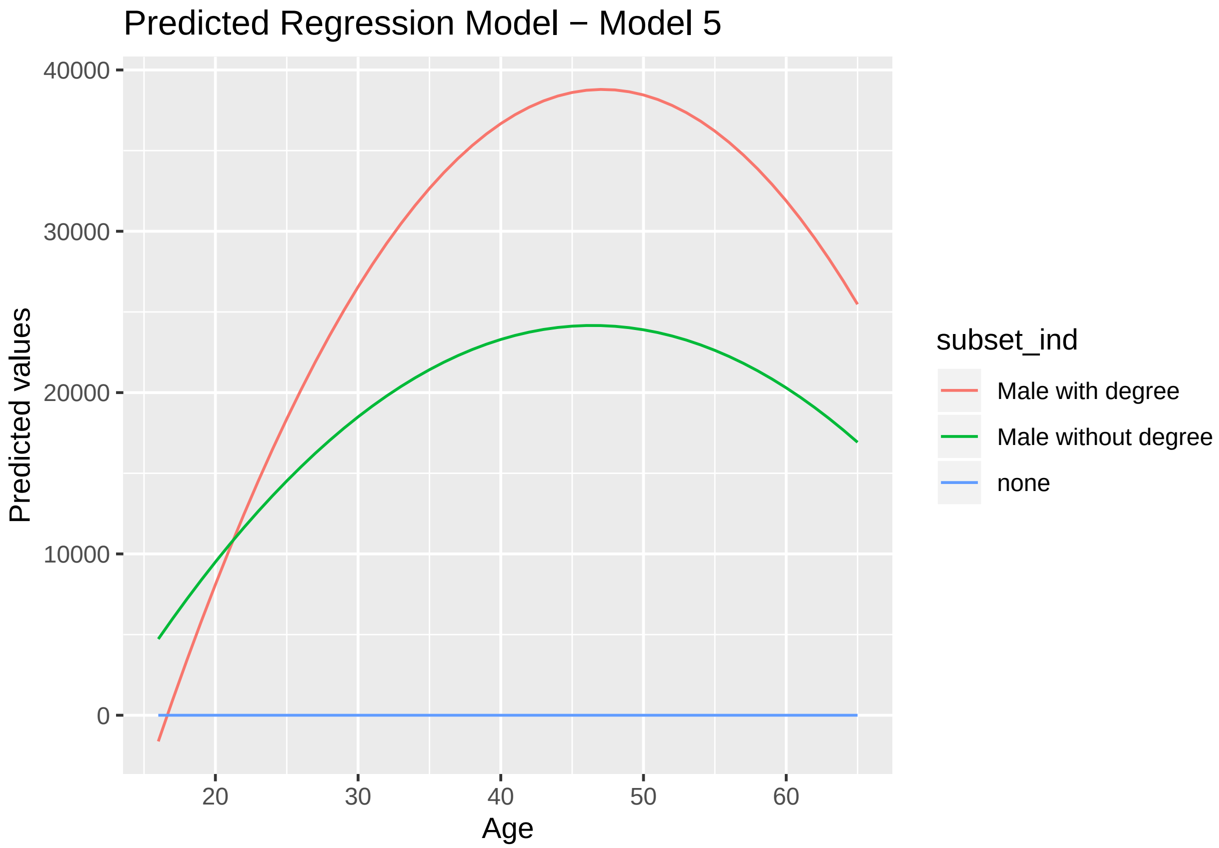
Table 5:

	Dependent variable:
	annualpay
age	1,949.000***
	(45.000)
agesq	$-$2,099.000***
	(56.000)
Constant	$-$21,088.000***
	(858.000)
Observations	17,626
R^2	0.140
Adjusted R^2	0.140
Residual Std. Error	12,593.000 (df = 17623)
F Statistic	1,398.000*** (df = 2; 17623)
Note:	*p<0.1; **p<0.05; ***p<0.01
	Robust standard errors in parenthesis

```
data_USoc$pred_mod5[data_USoc$subset_ind=="Male without degree"] <- mod5_mnd$fitted.values
```

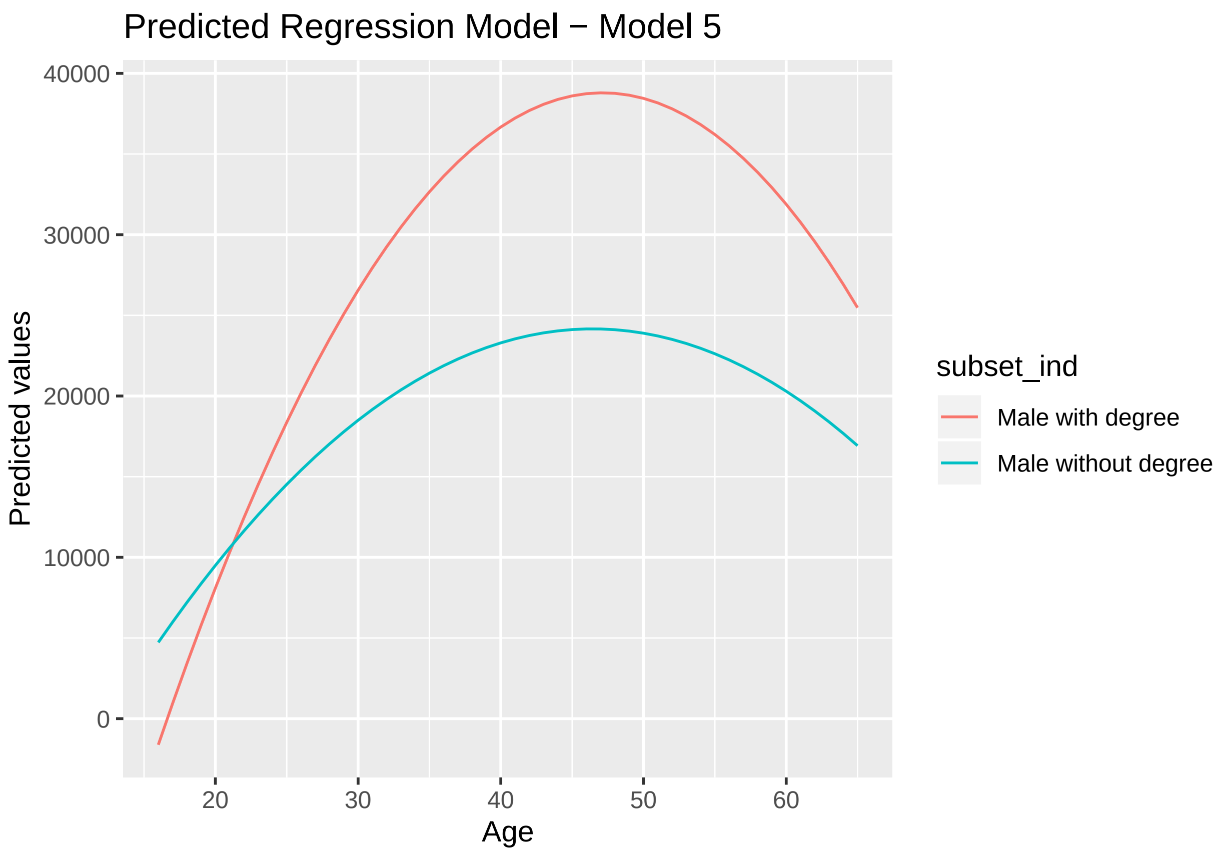
Now we plot the predicted values for the two specifications.

```
ggplot(data_USoc, aes(x=age,y=pred_mod5,color = subset_ind)) +
  geom_line() +
  ggtitle("Predicted Regression Model - Model 5") +
  ylab("Predicted values") +
  xlab("Age")
```



You will see that we have the "none" category plotted as well (of course we didn't estimate this). You could remove these data before plotting

```
# remove observations with subset_ind == "none"
data_temp <- data_USoc %>% filter(subset_ind != "none")
ggplot(data_temp, aes(x=age,y=pred_mod5,color = subset_ind)) +
  geom_line() +
  ggtitle("Predicted Regression Model - Model 5") +
  ylab("Predicted values") +
  xlab("Age")
```



END OF INSTRUCTIONS