```
---
  title: "GV300 Assignment 2: Understanding Electoral Participation in the UK"
author: "231"
date: "`r Sys.Date()`"
output:
  pdf_document:
  toc: false
number_sections: true
---
# Preparation ------------------------------------------------------------
library(foreign)
library(ggplot2)
library(stargazer)
library(effects)


# 1. Model with a dummy variable -----------------------------------------

# Load the Putnam 1994 data
pd <- read.csv("c:/users/ryan/desktop/gv207/putnam.csv")

# Take a look at the NorthSouth dummy

table(pd $ NorthSouth) # frequency table

pd $ NorthSouth # actual values
str(pd $ NorthSouth) # data type

# We can see that this is stored as a factor variable.

#----------------------------------#
# Re-coding the NorthSouth variable
#----------------------------------#

# As we did two weeks ago, let's first create a dummy variable coded 1
# for North and 0 for South.

pd $ North <- ifelse(pd $ NorthSouth == "North", 1, 0)

# Make sure that we did this correctly.

table(pd $ NorthSouth) # Original factor
table(pd $ North)      # Dummy we created above

table(pd $ North, pd $ NorthSouth) # Compare the two to make sure that we did this correctly.


#----------------------------------#
# Regression with one dummy X
#----------------------------------#
# Let's regress InstPerform on the newly created dummy instead.

fit.ns <- lm(InstPerform ~ North, data = pd)
stargazer(fit.ns, type = "text")

# notice how this is exactly the same thing as the difference of means test
# we learned a few weeks ago:

t.test(pd$InstPerform~pd$North)
# 11.83333-5.125 = 6.708
# So, we see that the difference between the means for the 2 groups is the
# same as the coefficient estimate for the North dummy variable
# further, we see that the intercept from the regression is 5.125,
# which is the mean for group 0 from the t.test and 5.125 + 6.708 = 11.8333,
# which is the mean for group 1 from the t.test
# So, a bivariate regression with a dummy variable predictor is exactly the same
# as a difference of means test.  After all, a regression is predicting the mean
# of some y based on a value of some x.  In this case, the mean of y when x = 0 vs
# when x equals 1. The conditional mean of y when x = 0 is just the intercept from
```

```
# the regression model.  when x = 1, it is the intercept + the coefficient for North


#----------------------------------#
# Plot the effect
#  (incorrect version)
#----------------------------------#

plot(effect(term = "North", mod = fit.ns))


#pdf(file = "North_me.pdf", width = 8, height = 5)
plot(effect(term = "North", mod = fit.ns),
     main = "Effect of North on Institutional Performance",
     xlab = "Region", ylab = "Institutional Performance")
#dev.off()


# As we learned in the lecture, this graph is not ideal for presenting the
# marginal effect of a dummy variable.

# To create a better graph, we need to estimate a model using the original
# factor variable.


#----------------------------------#
# Regression with one dummy X (factor)
#----------------------------------#

# Regress InstPerform on the original NorthSouth (factor variable)
fit.ns.fac <- lm(InstPerform ~ NorthSouth, data = pd)

stargazer(fit.ns.fac, type = "text")

#  Te regression results will be a little bit counterintuitive.
# "NorthSouthSouth" means the value South from the variable
# NorthSouth, which looks a little weird in the table.


# Compare the two regression results.

stargazer(fit.ns, fit.ns.fac, type = "text")

# Note that the two models are mathematically equivalent.
# They represent the exact same results but in a different way.

# The First model uses South as the baseline, so the estimated
# Constant (5.125) shows the predicted value of Y for South.
# The predicted value of Y for North is 5.125 + 6.708 = 11.833.

# We get: 5.125 for South and 11.833 for North.

# On the other hand, the second model uses North as the baseline,
# so the estimated Constant (11.833) shows the predicted value
# of Y for North.
# The predicted value of Y for South is obtained by adding the
# value of the estimated coefficient for NorthSouthSouth (which
# means "NorthSouth" dummy being equal to "South"),
# 11.833 + (-6.708) * 1 = 5.125.

# We get: 11.833 for North and 5.125 for South.

# So, both models give us the exact same answers!


# You can present either one of the two
# results in a regression table (there is absolutely no need to
# present both at the same time). The important point is that, whether you
# choose models (1) or (2), you need to be crystal clear about how the
```

```
# dummy variable is coded. With model (1), you won't need to provide
# much explanation because it's obvious from the variable name. If you
# choose model (2), you'd better change the variable name "NorthSouthSouth"
# to "South" in the regression table.


# In order to create the correct effect plot,
# Model (2) (i.e., the model with the original factor variable)
# is required.


#----------------------------------#
# Plot the effect
#  (correct version)
#----------------------------------#

# Model that uses the original factor:
plot(effect(term = "NorthSouth", mod = fit.ns.fac))

# Change the axis labels and graph title to improve the appearance.
#pdf(file = "North_me2.pdf", width = 8, height = 5)
plot(effect(term = "NorthSouth", mod = fit.ns.fac),
     main = "Effect of North on Institutional Performance",
     xlab = "Region", ylab = "Institutional Performance")
#dev.off()


# That is, you should NOT use the model that is based on the recoded dummy.
# Let's see what happens if you do.

# Model that uses the recoded dummy:
plot(effect(term = "North", mod = fit.ns))

# This is a very misleading graph. Readers may mistakenly think that
# X here is a numeric variable. Don't do this.



# Summary:
# When the main independent variable of our interest is a dummy
# variable, we may want to estimate two versions of one model.
#
# Version 1: a model that uses a recoded dummy variable.
#
# Version 2: a model that uses the original dummy variable that
#            is stored as a factor variable.
#
# Version 1 is usually easier to interpret. Therefore, we may want to present
#  the Version 1 results in a regression table. However, an effect
#  plot based on Version 1 is quite misleading.
#
# Version 2 is estimated so that we can create an intuitive effect
#  plot. However, we should not present the Version 2 results
#  in a regression table as is. You will have to change the variable name
#  manually (i.e., from NorthSouthSouth to South) in the table once you copied
# and pasted the table into a MS Word file.


# Remember that Version 1 and Version 2 are mathematically equivalent.
# It's just that we estimate 1 to create a table, and we estimate 2
# to create a graph.




# 2. Model with no X ------------------------------------------------
# (constant-only model)

# To estimate a model with no X, we put the number 1 in the right-hand-side
```

```
# of the equation, as follows.

fit.nox <- lm(InstPerform ~ 1, data = pd)
stargazer(fit.nox, type = "text")


# the estimated Constant is equal to the
# mean of Y. This is the "best guess" (best predicted value) of Y when
# we have no X.

mean( pd $ InstPerform )

# R^2 from a constant-only model is always 0, because R^2 measures how
# much variation in Y is explained by X. Since there is no X, NO variation
# in Y is explained.


# Residual Std. Error (Root Mean Squared Error) from a constant-only
# model is always equal to the standard deviation of Y, by definition.

sd( pd $ InstPerform )




# 3. Model with a non-binary nominal variable ----------------------------

wd <- read.csv("c:/users/ryan/desktop/gv207/world.csv")

# Suppose we are interested in the effect of region on per capita GDP

# Y (per capita GDP in $ 10,000)
summary(wd $ gdp_10_thou)

# X (region)
table(wd $ region)

# R does assign a numeric value to each category such that
# Africa = 1, Asia-Pacific = 2, C&E Europe = 3, ..., W. Europe = 8

table( as.numeric(as.factor(wd $ region) ))

# However, these numbers do not have a substantive meaning.
# They simply distinguish between categories.


#-----------------------------------------------#
# Regression with one nominal variable (factor)
#-----------------------------------------------#

# One way to analyze the relationship between Y and a non-binary nominal variable
# is to estimate a regression of Y on the original factor variable (i.e., include
# the original factor variable as is).

fit.gr <- lm(gdp_10_thou ~ region, data = wd)

stargazer(fit.gr, type = "text")


#  R will internally create 8 dummy variables and
# include 7 of them, dropping the first dummy variable (regionAfrica).


#-----------------------------------------------#
# Plot the effect of region
#-----------------------------------------------#

plot(effect(term = "region", mod = fit.gr))
```

```
# Change the axis labels and graph title
#pdf(file = "Region_me.pdf", width = 8, height = 5)
plot(effect(term = "region", mod = fit.gr),
     main = "Per capita GDP by region",
     xlab = "Region", ylab = "Per capita GDP (in $10,000)")
#dev.off()



#---------------------------------------------#
# Manually create k dummy variables
#---------------------------------------------#
# Alternatively, we can manually create k dummy variables ourselves and include
# them in a regression.

# Let's first create a series of dummy variables

wd $ Africa <- ifelse(wd $ region == "Africa", 1, 0)
wd $ AsiaPacific <- ifelse(wd $ region == "Asia-Pacific", 1, 0)
wd $ CEEurope <- ifelse(wd $ region == "C&E Europe", 1, 0)
wd $ MiddleEast <- ifelse(wd $ region == "Middle East", 1, 0)
wd $ NAmerica <- ifelse(wd $ region == "N. America", 1, 0)
wd $ SAmerica <- ifelse(wd $ region == "S. America", 1, 0)
wd $ Scandinavia <- ifelse(wd $ region == "Scandinavia", 1, 0)
wd $ WEurope <- ifelse(wd $ region == "W. Europe", 1, 0)

# Let's see how these dummy variables are structured:

wd[1:10, c("region", "Africa", "AsiaPacific", "CEEurope", "MiddleEast",
"NAmerica","SAmerica","Scandinavia","WEurope")]


# For the first row, region is equal to Middle East and so only the
# Middle East dummy takes the value of 1 and all the other dummy variables
# are coded as 0. Likewise, for the second observation, only the CEEurope
# dummy is 1 and everything else is 0.



# As we learned in the lecture, since there are 8 categories, we
# actually need only 7 dummy variables to completely describe the values
# of the region variable. In running a regression model, we thus need to include
# 7 of the 8 dummy variables. The omitted category will be used as the baseline.


# Let's now estimate a regression model where 7 dummy variables are
# included in the RHS. We will drop the Africa dummy.

fit.gr.1 <- lm(gdp_10_thou ~
                 AsiaPacific +
                 CEEurope +
                 MiddleEast +
                 NAmerica +
                 SAmerica +
                 Scandinavia +
                 WEurope,
               data = wd )

stargazer(fit.gr.1, type = "text")

# The results are exactly the same as the one we obtained before.

stargazer(fit.gr, fit.gr.1, type = "text")


# In both of the models, the Africa dummy is excluded, so Africa is
# used as the baseline.
# This means that the estimated Constant represents the predicted value
```

```
# of Y for Africa, because this is the predicted value of Y when ALL the
# independent variables are equal to 0.

# To obtain the predicted value of Y for other regions, we can simply
# add Constant (0.093) to the estimated coefficient for the corresponding
# region. For example, the predicted Y for NAmerica is 0.093 + 2.077 = 2.17.


# An even better way to obtain the
# same results is to estimate a model that includes all 8 dummies
# but drops the constant. To drop a constant, we add "+ 0" in the
# right-hand-side of the equation, as follows.

fit.gr.2 <- lm(gdp_10_thou ~
                 Africa +
                 AsiaPacific +
                 CEEurope +
                 MiddleEast +
                 NAmerica +
                 SAmerica +
                 Scandinavia +
                 WEurope + 0,
               data = wd )

stargazer(fit.gr.2, type = "text")

# Let's compare the two results to see that these two models are
# mathematically equivalent.

stargazer(fit.gr.1, fit.gr.2, type = "text")

# Even though the estimated coefficients are different, they
# really are the same model. To see how, try calculating the
# predicted value of Y for NAmerica with model (1) and with
# model (2).


# If we use the first model, the predicted value of Y for N. America is
# 0.093 (which is the intercept) + 2.077 (which is the slope for NAmerica)
# as we saw above. It is equal to 0.093 + 2.077 = 2.17.

# On the other hand, if we use the second model, the predicted value of Y
# for N. America is simply 2.170 (slope for NAmerica).

# So both models say that it's 2.17. This is not a coincidence. We would
# obtain the same predicted values for other regions as well.
# This is because, again, the two models are equivalent. These two are
# simply using two different ways to express the same pattern.

# Notice that Residual Std. Error from both models are exactly the same.

# However, R^2 from these models are different. This is because one model
# has a Constant and the other model doesn't.


# What would happen if we fail to drop one dummy variable NOR drop the intercept...?

fit.gr.n <- lm(gdp_10_thou ~  Africa + AsiaPacific + CEEurope + MiddleEast +
                 NAmerica + SAmerica + Scandinavia + WEurope, data = wd )

stargazer(fit.gr.n, type = "text")

# What happens is that R will just simply drop the LAST dummy variable
# (WEurope) automatically. This is not wrong, but it is not ideal, either.
# As I mentioned above, the best approach here is to include ALL dummy variables
# and drop the constant.
```

```
#-----------------------------------------------#
# Incorrect approach
#-----------------------------------------------#

# As I mentioned in the lecture, we may sometimes find ourselves dealing with a data set
# where a nominal variable is stored as a numeric variable rather than as a factor variable.


# Let's imagine that we only have a numeric version of the region variable that only assigns
# numbers to different regions:
# Africa: 1
# Asia-Pacific: 2
# C&E Europe: 3
# ...


# Let's first create such a variable by forcing our region variable (factor) to be a numeric
# variable using the as.numeric function.

wd $ num.region <- as.numeric(as.factor(wd $ region))

# To take a look at the values...
table(wd $ region, wd $ num.region)

head(wd[c("region", "num.region")], 10)


# So, let's imagine that we only have this num.region variable but not region variable.


# Our natural instinct may be to include this variable in a regression as is, but doing so
# is WRONG!!!

##### BAD REGRESSION MODEL #####
fit.gdp.reg <- lm(gdp_10_thou ~ num.region, data = wd)
stargazer(fit.gdp.reg, type = "text")
##### BAD REGRESSION MODEL #####


# As I said in the lecture, this is nonsensical.
# This model is based on three unreasonable assumptions (1. there is an order
# in the values of X, 2. the effect of X on Y is directional, 3. one unit change
# in X produces the same change in Y across different values of X.)


# Running such a model means we choose a line to summarize the following scatterplot.
g <- ggplot(wd, aes(x = region, y = gdp_10_thou)) + geom_point()
g <- g + ylab("Per capita GDP") + xlab("Region")
g

# As we saw above, the following plot is a better visual summary of the relationship
# between region and GDP.

plot(effect(term = "region", mod = fit.gr))




# 4. Model with an ordinal X ----------------------------------------------

# Y: Government effectiveness
summary(wd $ effectiveness)

# X: Ethnic fractionalization level
table(wd $ frac_eth3)

# Note 1: Notice that the frequency table above has a mysterious blank
# column (with 3 observations). This indicates that there are 3 observations
```

```
# that are not categorized as any of the three values (High, Medium, or Low).

# In other words, the value of this variable is missing for these 3 observations,
# but they are not properly coded as NA. This can cause a big problem in running
# a regression, so let's fix this.

# We fix this by recoding these 3 observations as NA.

wd $ frac_eth3[wd $ frac_eth3 == ""] <- NA

table(wd $ frac_eth3)
# We still have a blank column, but there is 0 observation now.


#-----------------------------------------------#
# Automatic approach 1 (let R handle the issue)
#-----------------------------------------------#
# The frac_eth3 is an ordinal variable. Ordinal variables are also
# categorical variables, and so they are stored as a factor variable
# in R.

# As we saw above, there are automatic and manual approaches when
# it comes to handling factor variables in regression.

# Let's first see the automatic approach.
fit.ge.0 <- lm(effectiveness ~ frac_eth3, data = wd)
stargazer(fit.ge.0, type = "text")


# The results are not really intuitive.
# R stores this variable as a factor variable, but it doesn't
# recognize that there is a natural order in the values (Low --> High) or
# (High --> Low).
# R simply orders the three values (High, Medium, Low) alphabetically.

# We will fix this later, but let's first see what the effect graph would
# look like if we simply adopt the automatic approach.

plot(effect(term = "frac_eth3", mod = fit.ge.0))

#pdf(file = "frac3_me_auto1.pdf", width = 8, height = 5)
plot(effect(term = "frac_eth3", mod = fit.ge.0),
     main = "",
     xlab = "Ethnic fractionalization levels",
     ylab = "Government effectiveness")
#dev.off()


#-----------------------------------------------#
# Automatic approach 2 (fix the ordering first)
#-----------------------------------------------#

# We will now fix the ordering of the frac_eth3 variable.

wd $ frac_eth3_ord <- factor(wd $ frac_eth3,
                             levels = c("Low", "Medium", "High"))

# In writing down the values for the levels option, arrange them
# in the "natural" order (either Low to High or High to Low).


# Whenever we create a new variable by recoding an existing one,
# we should check if it worked correctly.

table(wd $ frac_eth3)
table(wd $ frac_eth3_ord)

table(wd $ frac_eth3_ord, wd $ frac_eth3)
```

```
# Now let's estimate a regression with this variable

fit.ge.1 <- lm(effectiveness ~ frac_eth3_ord, data = wd)
stargazer(fit.ge.1, type = "text")


# Compare the two models (automatic 1 and automatic 2)

stargazer(fit.ge.0, fit.ge.1, type = "text")

# Notice that the two models are equivalent, despite the difference in appearances.

# The two are equivalent in the sense that
#  (a) model fit statistics are the same;
#  (b) predicted values for Low, Medium, High are the same.

# We can easily see (a) by looking at the table.
# You can convince yourself about (b) by calculating the predicted values
# for Low, Medium, and High.

# From model (1)
#   High:   34.736
#   Low:    34.736 + 20.443 = 55.179
#   Medium: 34.736 + 13.347 = 48.083

# From model (2)
#   Low:    55.180
#   Medium: 55.180 - 7.096 = 48.084
#   High:   55.180 - 20.443 = 34.737

# Except for differences due to rounding, the two sets of three
#  predicted are identical.

# These two models are just different representations of the same relationship.



# Effect plot

plot(effect(term = "frac_eth3_ord", mod = fit.ge.1))

#pdf(file = "frac3_me_auto2.pdf", width = 8, height = 5)
plot(effect(term = "frac_eth3_ord", mod = fit.ge.1),
     main = "",
     xlab = "Ethnic fractionalization levels",
     ylab = "Government effectiveness")
#dev.off()


#-----------------------------------------------#
# Manual approach
#-----------------------------------------------#
# Let's now apply the manual approach.

# We create a series of dummy variables first
wd $ frac_eth3_Low <- ifelse(wd $ frac_eth3 == "Low", 1, 0)
wd $ frac_eth3_Medium <- ifelse(wd $ frac_eth3 == "Medium", 1, 0)
wd $ frac_eth3_High <- ifelse(wd $ frac_eth3 == "High", 1, 0)

# and include all of them AND drop the constant

fit.ge.2 <- lm(effectiveness ~ frac_eth3_Low + frac_eth3_Medium +
                 frac_eth3_High + 0,
               data = wd)

stargazer(fit.ge.2, type = "text")

# We can see that ethnic fractionalization level has a negative effect
```

```
# on government effectiveness.


# Compare the tables from automatic and manual approaches.

stargazer(fit.ge.0, fit.ge.2, type = "text", omit.stat = "f")



#-----------------------------------------------#
# Third approach: treat it as numerical
#-----------------------------------------------#

# First, we need to convert the ordinal frac_eth3_ord into numeric
wd $ num.frac_eth3 <- as.numeric(wd $ frac_eth3_ord)

table(wd $ num.frac_eth3, wd $ frac_eth3_ord)

# We can see that "Low" is now changed into 1, "Medium" is 2, and "High" is 3.


# It is really important that we use here the ORDERED version of frac_eth3
# that we manually created above, not the original frac_eth3.
# Recall that the original frac_eth3 has the "wrong" ordering: High, Low, Medium.


fit.ge.3 <- lm(effectiveness ~ num.frac_eth3, data = wd)

stargazer(fit.ge.3, type = "text")


plot(effect(term = "num.frac_eth3", mod = fit.ge.3))

#pdf(file = "frac_num_me.pdf", width = 8, height = 5)
plot(effect(term = "num.frac_eth3", mod = fit.ge.3),
     main = "",
     xlab = "Ethnic fractionalization levels",
     ylab = "Government effectiveness")
#dev.off()



#-----------------------------------------------#
# Compare the Method 1 and Method 2 results
#-----------------------------------------------#

stargazer(fit.ge.2, fit.ge.3, type = "text", omit.stat = "f")

# Here, the two models are NOT mathematically equivalent.
# The first model (dummy approach) is more flexible.
# If the more flexible model fits the data better (has a bigger R2),
# you should choose the more flexible model as the best model.

# However, it may be OK to choose the less flexible
# model (i.e., the numeric approach) when one or more of the following conditions hold:

# 1. RMSE for the less flexible model is not much smaller compared with the
#    more flexible one;
# 2. Your ordinal variable has too many (more than 5) categories;
# 3. The ordinal variable is NOT your main independent variable (i.e., it's just
#    a control variable).

# In our current example, RMSE is actually smaller for the less flexible model.
# Also, the flexible model suggests that the relationship is almost linear, as
# we saw in the plot.
# So, it is OK to use the numeric one, if frac_eth3 is just a control variable
# (i.e., not your main independent variable).
```

```
# end of file
```