Tutorial 9: Data Management and Two-Stage Least Squares (2SLS) Regression

Jan Vogler (jan.vogler@duke.edu)

October 23, 2015

Today's Agenda

1. Data management I: reading/subsetting data, keeping/deleting variables
2. Data management II: data transformation
3. Data management III: creating new variables
4. Data management IV: useful commands
5. Two-stage least squares (2SLS) regression

1. Data management I: reading/subsetting data, keeping/deleting variables

R can save its own datafiles. Those will have the format ".Rdata". In order to load such a dataset, you can simply use the load command.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/w9/")
load("filename.Rdata")
```

R can also read some other file formats, including .csv and .txt files.

```
### .csv files
csvdata = read.csv("filename.csv", stringsAsFactors = FALSE)
# stringsAsFactors=FALSE is important because otherwise R will load
# character variables as factors, treating them as numerical under the
# surfae, which can lead to complications later on

### .txt files
textdata = read.table("filename.txt", header = TRUE)

# header=TRUE indicates that the first row contains the names of the
# variables (see later for example)
```

Additionally, there are many other data formats (SPSS .sav and Stata .dta files). Some of them require the foreign package.

Use the following command to install it:

install.packages("foreign")

```
library(foreign)
spssdata = read.spss("filename.sav", to.data.frame = TRUE)

library(foreign)
statadata = read.dta("filename.dta")
```

For data files that were saved by the most recent version of Stata (13), you will need another package called “readstata13”. Please use the following command to install it:

install.packages(“readstata13”)

```
library(readstata13)
read.dta13("filename.dta")
```

```
## Error in eval(expr, envir, enclos): First byte: Not a version 13/14 dta-file.
```

Let us now read our LDC dataset.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/")
LDC = read.dta("LDC_IO_replication.dta")
```

To delete a variable in a data frame, we simply use the following command:

```
LDC$ecris2 = NULL
```

We can also only keep the data that we actually need for our analysis:

```
LDC = LDC[, c("ctylabel", "date", "polityiv_update2", "gdp_pc_95d", "newtar",
    "l1polity", "l1polity", "l1signed", "l1office", "l1gdp_pc", "l1lnpop", "l1ecris2",
    "l1bpc1", "l1avnewtar", "l1fdi")]
```

The following command allows us to only look at cases on which we have all observations of specific variables.

```
complete = with(LDC, complete.cases(polityiv_update2, gdp_pc_95d))
# The 'with' command allows you to evaluate a file for certain expressions,
# such as complete.cases
LDC = LDC[complete, ]
# Then we subset the file and only take the complete cases
```

This is often a better solution than na.omit because na.omit will remove all rows with missing values, which can result in the loss of too many values

```
LDComit = na.omit(LDC)
# only 383 observations remaining if we use this method
```

Before doing any analysis, you should inspect your variables closely. Make sure that you are aware of the properties of your most important variables.

```
summary(LDC$polityiv_update2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -10.000  -7.000  -4.000  -1.121   7.000  10.000
```

```
summary(LDC$gdp_pc_95d)
```

```
##     Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
##    91.02   380.70   978.00  2320.00  2437.00 44160.00
```

```
summary(LDC$newtar)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    0.10   11.70   17.80   21.73   27.50  102.20    2014
```

You have learned how to merge data in one of the previous tutorials. But what happens if we want to merge data with different numbers of observations?

```
a = c("Household A", "Household B")
b = c(2, 2)
data1 = data.frame(a, b)
colnames(data1) = c("Name", "Number of people")

data1
```

```
##          Name Number of people
## 1 Household A                2
## 2 Household B                2
```

```
c = c("Household A", "Household A", "Household B", "Household B")
d = c("Individual 1", "Individual 2", "Individual 3", "Individual 4")

data2 = data.frame(c, d)
colnames(data2) = c("Name", "Person")

data2
```

```
##          Name       Person
## 1 Household A Individual 1
## 2 Household A Individual 2
## 3 Household B Individual 3
## 4 Household B Individual 4
```

As we can see, the data has different numbers of observations on the household level.

```
merge(data1, data2, by = ("Name"))
```

```
##          Name Number of people       Person
## 1 Household A                2 Individual 1
## 2 Household A                2 Individual 2
## 3 Household B                2 Individual 3
## 4 Household B                2 Individual 4
```

As we can see, the data will expand.

Merge has an option that can be helpful. If you have two data frames "x" (here: data1) and "y" (here: data2), you can decide to keep all parts of either data frame or both. This is useful because the merge command normally deletes observations that are not

```r
merge(x = data1, y = data2, all = TRUE, all.x = TRUE, all.y = TRUE)
```

```
##          Name Number of people       Person
## 1 Household A                2 Individual 1
## 2 Household A                2 Individual 2
## 3 Household B                2 Individual 3
## 4 Household B                2 Individual 4
```

When merging we can think of our data frames as the "master" and "merging" data frame. Often our master data frame contains the crucial variables of our analysis. We then combine it with additional "merging" data frames. If we consider the variables in our master data frame to be most important for the analysis, it often makes sense to keep all entries of this data frame (by using all.x=TRUE). In many cases, one has to be careful when merging data frames because it might be the case that several observations in the merging data frame match an entry in the master data frame, even though the researcher does not want to match one with multiple observations.

Let's assume we are only interested in a subset of the data. For example, we are only interested in the country Angola. How can we subset the data that we have?

```r
LDC2 = subset(LDC, ctylabel == "Angola")
summary(LDC2)
```

```
##    ctylabel              date      polityiv_update2   gdp_pc_95d
##  Length:14          Min.   :1980   Min.   :-7.000   Min.   :519.6
##  Class :character   1st Qu.:1983   1st Qu.:-7.000   1st Qu.:643.8
##  Mode  :character   Median :1986   Median :-7.000   Median :675.3
##                     Mean   :1988   Mean   :-6.143   Mean   :651.0
##                     3rd Qu.:1990   3rd Qu.:-7.000   3rd Qu.:707.4
##                     Max.   :1999   Max.   :-3.000   Max.   :732.4
##
##      newtar       l1polity        l1polity.1        l1signed
##  Min.   : NA   Min.   :-7.000   Min.   :-7.000   Min.   :0
##  1st Qu.: NA   1st Qu.:-7.000   1st Qu.:-7.000   1st Qu.:0
##  Median : NA   Median :-7.000   Median :-7.000   Median :0
##  Mean   :NaN   Mean   :-6.385   Mean   :-6.385   Mean   :0
##  3rd Qu.: NA   3rd Qu.:-7.000   3rd Qu.:-7.000   3rd Qu.:0
##  Max.   : NA   Max.   :-3.000   Max.   :-3.000   Max.   :0
##  NA's   :14    NA's   :1        NA's   :1
##     l1office         l1gdp_pc        l1lnpop         l1ecris2
##  Min.   : 1.000   Min.   :504.1   Min.   :15.74   Min.   :0.0000
##  1st Qu.: 4.000   1st Qu.:641.4   1st Qu.:15.82   1st Qu.:0.0000
##  Median : 6.000   Median :672.5   Median :15.91   Median :0.0000
##  Mean   : 7.231   Mean   :646.8   Mean   :15.96   Mean   :0.1429
##  3rd Qu.: 9.000   3rd Qu.:713.1   3rd Qu.:16.00   3rd Qu.:0.0000
##  Max.   :18.000   Max.   :732.4   Max.   :16.30   Max.   :1.0000
##  NA's   :1        NA's   :1
##      l1bpc1     l1avnewtar        l1fdi
##  Min.   :1    Min.   : 0.00   Min.   : 1.718
##  1st Qu.:1    1st Qu.:17.05   1st Qu.: 2.421
##  Median :1    Median :24.69   Median : 4.211
##  Mean   :1    Mean   :22.28   Mean   : 8.115
##  3rd Qu.:1    3rd Qu.:28.25   3rd Qu.: 5.726
##  Max.   :1    Max.   :30.52   Max.   :36.361
##  NA's   :12                   NA's   :6
```

```
# We are left with observations for Angola only.  We can also apply this to
# numerical variables and set conditions for subsetting.
```

What would you do if you have multiple datafiles that all have a similar name. How can you load this data very efficiently?

Credit to Brett Gall for this chunk of the code.

Let's assume we have 1000 different individuals and their files have similar names. How can we load those files without writing 1000 separate commands for loading the data?

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/w9/")
ind.files = dir(pattern = "ind\\d+.sav")
# This command allows us to get the names of the datafiles Note that \\d+
# is a so-called 'regular expression' \\ initiates the regular expression
# 'd' stands for a digit + stands for one or more (digit) Google 'regular
# expressions' to learn how to construct more regular expressions

ind.data = lapply(ind.files, read.spss)
# This loads the data via the 'lapply' command Here we apply the command
# 'read.spss' to each element of our list read.spss requires the foreign
# package

names(ind.data) = gsub(".sav", "", ind.files)
# Rename list elements, remove '.sav' and replace it with '' (empty)

names(ind.data)
```

```
## [1] "ind1" "ind2" "ind3" "ind4" "ind5"
```

2. Data management II: data transformation and recoding

We can transform variables in a number of ways. One of the most common ways to transform data is to square it to estimate curvilinear relationships. Let us construct a squared version of the Polity IV Score.

In order to make the square meaningful, we first have to add +10 to all Polity Scores (otherwise negative squares would get the same positive values as the positive square).

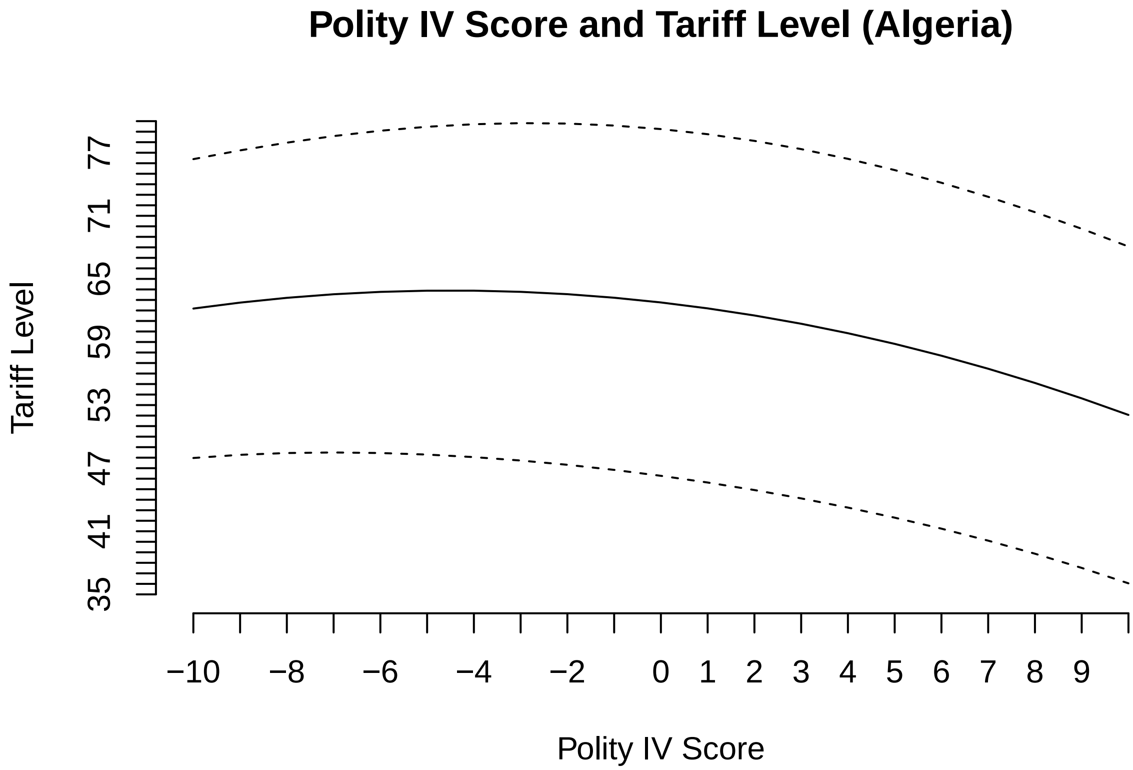
```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/")
LDC = read.dta("LDC_IO_replication.dta")
LDC$polityiv_update2 = LDC$polityiv_update2 + 10
LDC$l1polity = LDC$l1polity + 10
LDC$polityiv_squared = (LDC$polityiv_update2)^2
LDC$l1polity_squared = (LDC$l1polity)^2
```

Let us see whether there is a curvilinear relationship between the Polity IV Score and tariff levels.

```
main_int = lm(newtar ~ l1polity + l1polity_squared + l1signed + l1office + l1gdp_pc +
    l1lnpop + l1ecris2 + l1bpc1 + l1avnewtar + factor(ctylabel) - 1, data = LDC)
# summary(main_int)
```

Interestingly, there appears to be a curvilinear relationship between the two variables! How would we interpret the results of the linear regression? How would we plot this curvilinear relationship?

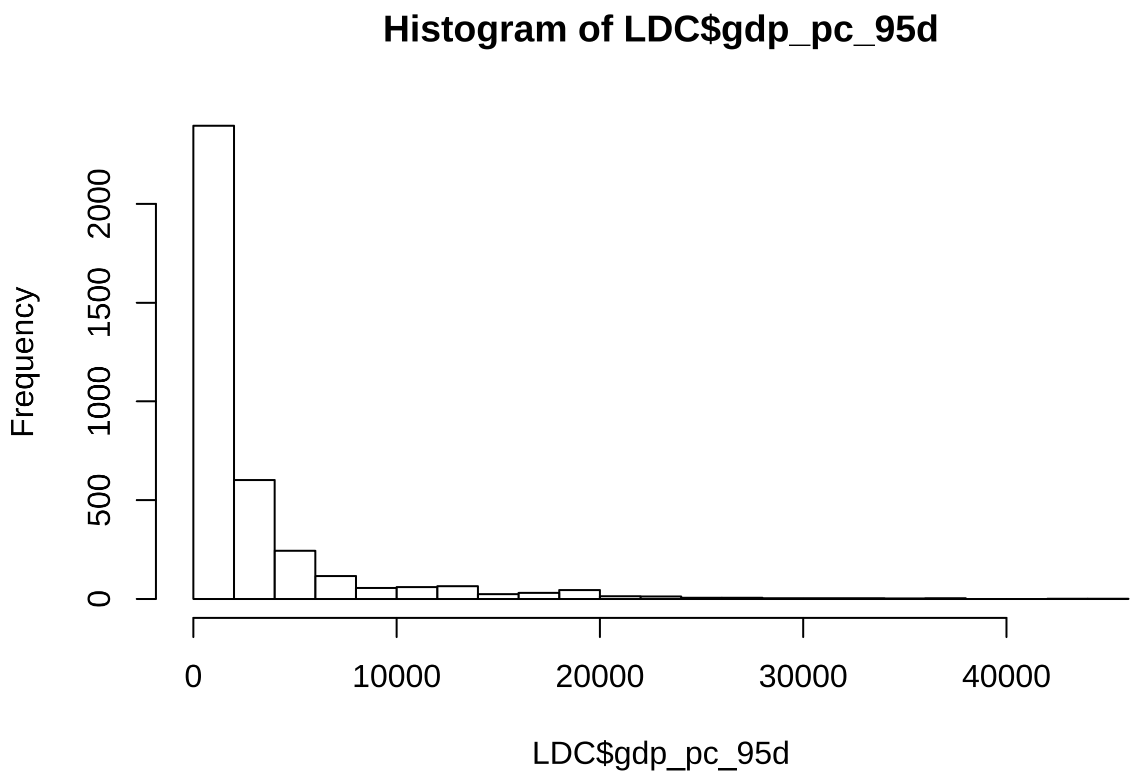
```
nd <- data.frame(l1polity = seq(0, 20, by = 1), l1polity_squared = seq(0, 20,
    by = 1)^2, l1signed = rep(0.1511, 21), l1office = rep(8.431, 21), l1gdp_pc = rep(2888,
    21), l1lnpop = rep(15.1, 21), l1ecris2 = rep(0.0641, 21), l1bpc1 = rep(0.5909,
    21), l1avnewtar = rep(14.91, 21), ctylabel = rep("Algeria", 21))

pred.p1 <- predict(main_int, type = "response", se.fit = TRUE, newdata = nd)
pred.table <- cbind(pred.p1$fit, pred.p1$se.fit)

fit <- pred.p1$fit
low <- pred.p1$fit - 2 * pred.p1$se.fit
high <- pred.p1$fit + 2 * pred.p1$se.fit
cis <- cbind(fit, low, high)

plot(pred.p1$fit, type = "l", ylim = c(35, 80), main = "Polity IV Score and Tariff Level (Algeria)",
    xlab = "Polity IV Score", ylab = "Tariff Level", axes = FALSE)
axis(1, at = seq(1, 21), labels = seq(-10, 10, 1))
axis(2, at = seq(35, 80), labels = seq(35, 80))
matlines(cis[, c(2, 3)], lty = 2, col = "black")
```



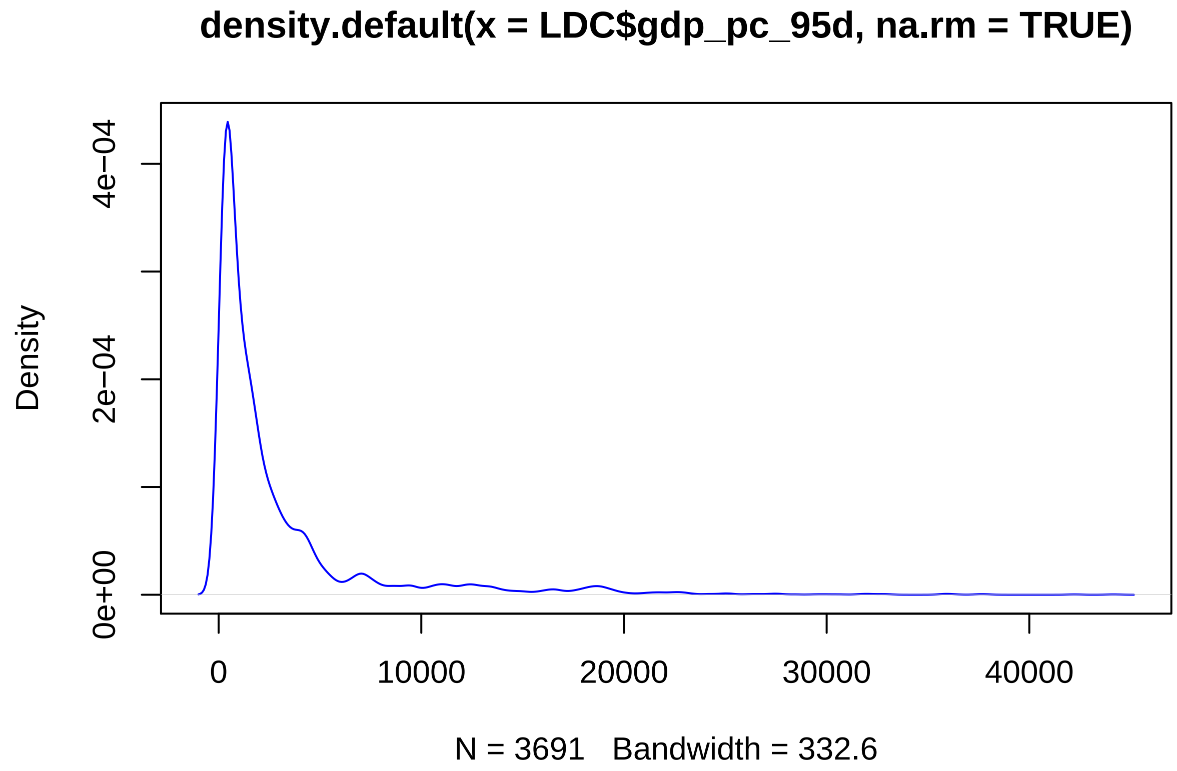
This relationship looks slightly different than the relationship estimated by Milner and Kubota. How is it different?

Moreover, another frequently used way to transform data is to take the natural logarithm. In many cases, researchers do this because the distribution of the data is skewed to the right. The goal is to reduce the skewness.

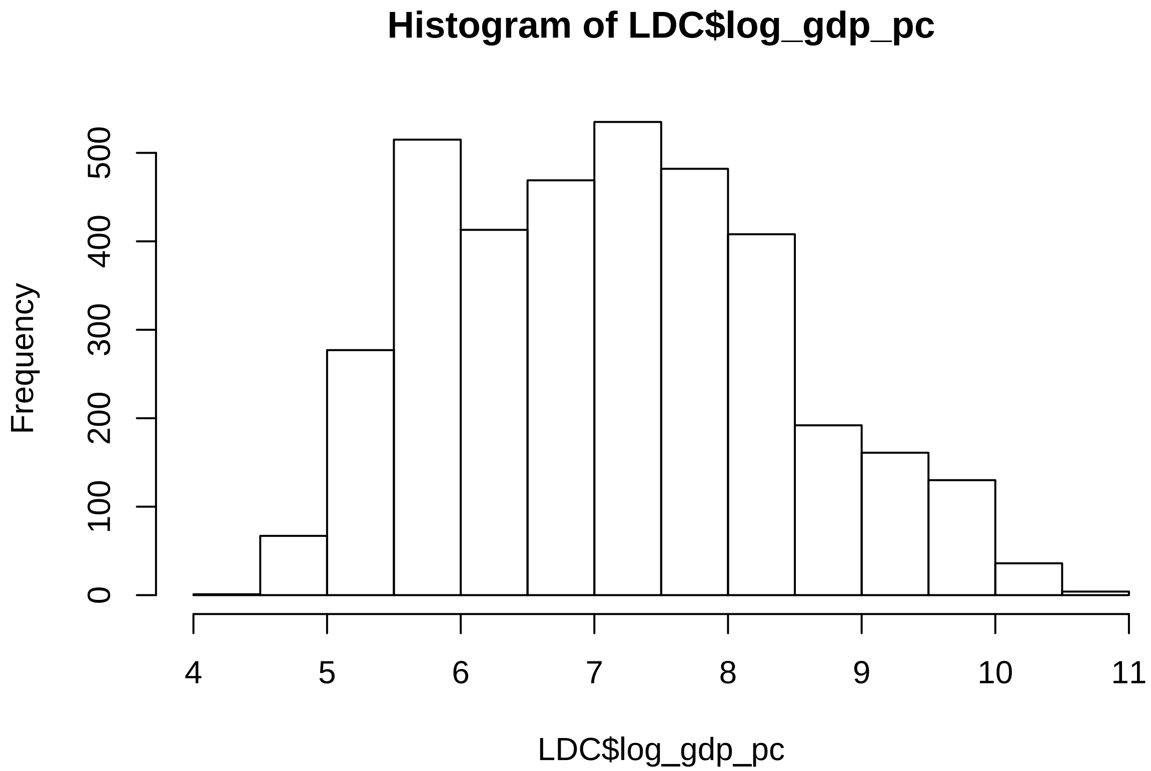
```
hist(LDC$gdp_pc_95d, breaks = 21)
```



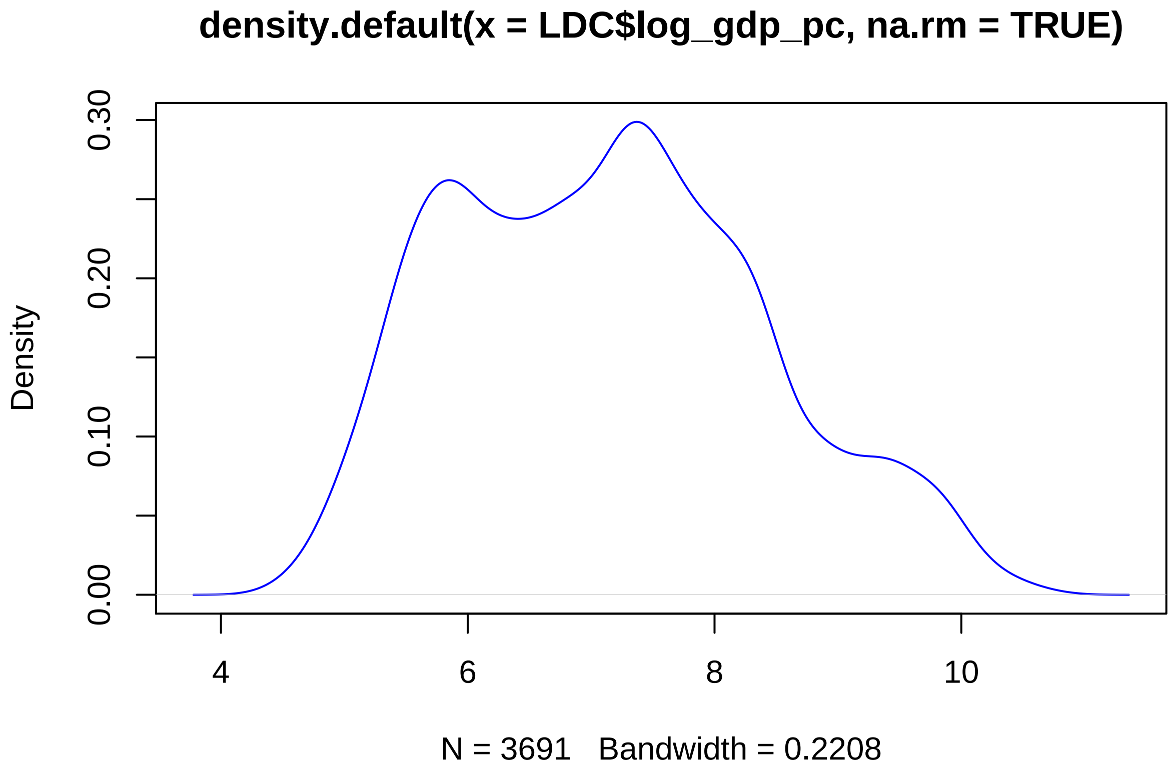
```
dens = density(LDC$gdp_pc_95d, na.rm = TRUE)
plot(dens, col = "blue")
```



```
LDC$log_gdp_pc = log(LDC$gdp_pc_95d)

hist(LDC$log_gdp_pc, breaks = 21)
```



```
dens = density(LDC$log_gdp_pc, na.rm = TRUE)
plot(dens, col = "blue")
```



```
### We have reduced the skewness and enforced a distribution that is closer to
### a unimodal distribution
```

What do we do if the data is not in the format in which we want it to be?

Let us look at a dataset that deals with lotteries in different states.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/w9/")
prize = read.csv("Lottery Prize Amounts Awarded.csv", stringsAsFactors = FALSE)
tickets = read.csv("Lottery Ticket Sales.csv", stringsAsFactors = FALSE)

# Let's look at the datasets

# Let's get rid of missing values

prize = prize[93:103, ]
tickets = tickets[93:103, ]
```

We want to have the data in a format in which we have both the tickets sold and the prizes awarded for each year and state. How do we get there? The “reshape” package is very useful in this respect.

install.packages(“reshape”)

```
library(reshape)
prize <- melt(prize, id = c("Year"))
prize$Prize = prize$value
```

```
prize$value = NULL

tickets <- melt(tickets, id = c("Year"))
tickets$TSales = tickets$value
tickets$value = NULL
```

Now let's merge the two dataframes.

```
full = merge(prize, tickets, by = c("Year", "variable"))
View(full)
```

More information on the reshape package can be found here: http://had.co.nz/reshape/

3. Data management III: creating new variables

Often we can create new variables from existing ones. Let us create three categories for a country's wealth. (1) Low income countries, (2) middle income countries, and (3) high income countries.

```
LDC$incomelevel = NA
LDC$incomelevel[LDC$gdp_pc_95d <= 10000] = "Low-income country"
unique(LDC$incomelevel)
```

```
## [1] "Low-income country" NA
```

```
LDC$incomelevel[LDC$gdp_pc_95d > 10000 & LDC$gdp_pc_95d <= 20000] = "Middle-income country"
unique(LDC$incomelevel)
```

```
## [1] "Low-income country"    NA                      "Middle-income country"
```

```
LDC$incomelevel[LDC$gdp_pc_95d > 20000] = "High-income country"
unique(LDC$incomelevel)
```

```
## [1] "Low-income country"    NA                      "Middle-income country"
## [4] "High-income country"
```

```
hist(LDC$incomelevel)
```

```
## Error in hist.default(LDC$incomelevel): 'x' must be numeric
```

```
### Does not work because it is not numeric!
```

If we want a histogram of our data, our argument needs to be numeric. We can do a simple data transformation to turn the variable into a numeric variable.

```
LDC$incomelevel[LDC$incomelevel == "Low-income country"] = 0
LDC$incomelevel[LDC$incomelevel == "Middle-income country"] = 1
LDC$incomelevel[LDC$incomelevel == "High-income country"] = 2

hist(LDC$incomelevel)
```

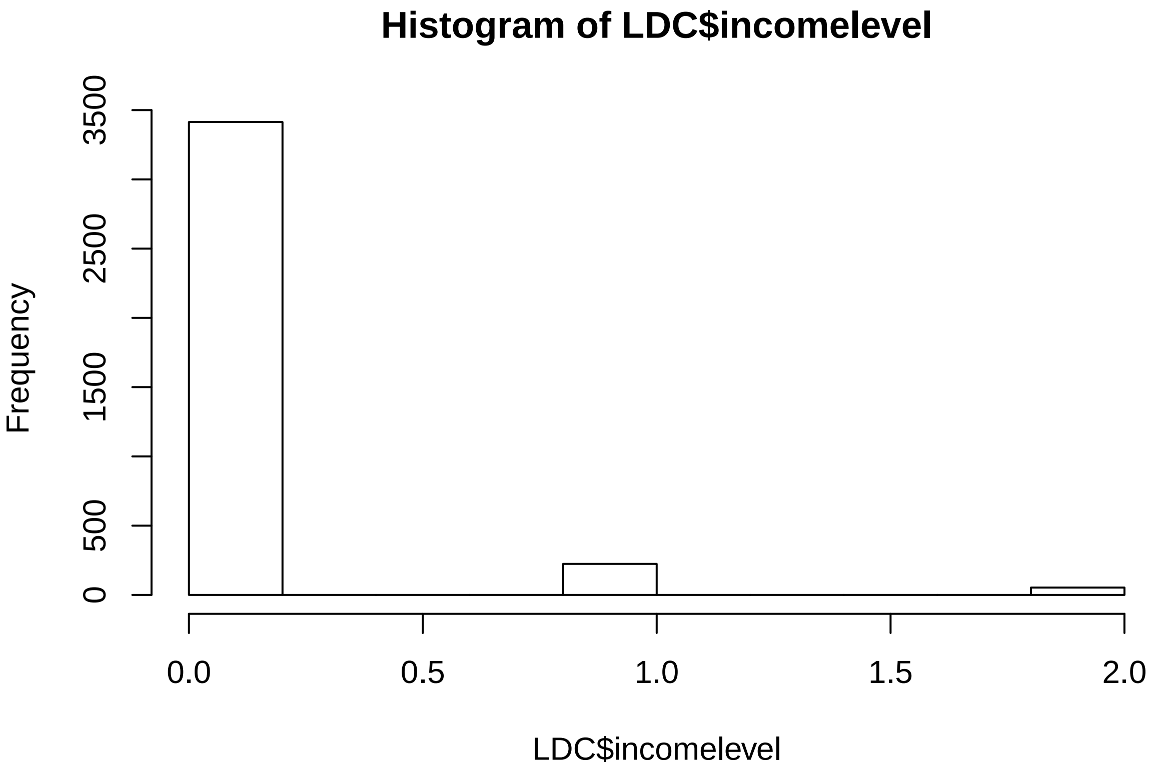
```
## Error in hist.default(LDC$incomelevel): 'x' must be numeric
```

```
### Why does this not work? We just assigned numeric values.
```

We just assigned numeric values to our variable but R still treats this a character variable. What is the reason for this?

In order to change the coding of the variable to numeric, we can use the following command:

```
LDC$incomelevel = as.numeric(LDC$incomelevel)
hist(LDC$incomelevel)
```



As we can see, the vast majority of our sample are low-income countries.

Now let us try to create a new variable that depends on two other variables. We want to have a major economic crises when we have both an economic crisis and a balance-of-payment crisis. How can we do that in R?

```
LDC$major_crisis = NA
LDC$major_crisis[LDC$ecris2 == 1 & LDC$bpcris == 0] <- 0
LDC$major_crisis[LDC$ecris2 == 0 & LDC$bpcris == 1] <- 0
LDC$major_crisis[LDC$ecris2 == 1 & LDC$bpcris == 1] <- 1
```

Let's say you want to create a variable that shows you if the Polity IV Score changed in any given year, with 0 indicating no change and 1 indicating a change. How would we do that?

```
complete = with(LDC, complete.cases(polityiv_update2))
# Necessary because otherwise we will get NAs
LDC = LDC[complete, ]
LDC$politychange = NA
for (i in 2:length(LDC$polityiv_update2)) {
    if (LDC$ctylabel[i] == LDC$ctylabel[i - 1]) {
        if (LDC$polityiv_update2[i] != LDC$polityiv_update2[i - 1]) {
            LDC$politychange[i] = 1
        } else {
            LDC$politychange[i] = 0
        }
    }
}
```

4. Data management IV: useful commands

Here are some more useful commands for data management:

```
unique(LDC$ctylabel)
```

```
##   [1] "Turkey"                 "Yugoslavia"
##   [3] "SouthAfrica"            "Argentina"
##   [5] "Bolivia"                "Brazil"
##   [7] "Chile"                  "Colombia"
##   [9] "CostaRica"              "DominicanRepublic"
##  [11] "Ecuador"                "ElSalvador"
##  [13] "Guatemala"              "Haiti"
##  [15] "Honduras"               "Mexico"
##  [17] "Nicaragua"              "Panama"
##  [19] "Paraguay"               "Peru"
##  [21] "Uruguay"                "Venezuela"
##  [23] "Guyana"                 "Jamaica"
##  [25] "Trinidad&Tobago"        "Bahrain"
##  [27] "Cyprus"                 "Iran"
##  [29] "Iraq"                   "Israel"
##  [31] "Jordan"                 "Kuwait"
##  [33] "Lebanon"                "Oman"
##  [35] "Qatar"                  "SaudiArabia"
##  [37] "Syria"                  "UnitedArabEmirates"
##  [39] "Egypt"                  "Yemen"
##  [41] "RepublicofYemen"        "Afghanistan"
##  [43] "Bangladesh"             "Bhutan"
##  [45] "Myanmar"                "Cambodia"
##  [47] "SriLanka"               "India"
##  [49] "Indonesia"              "Korea"
##  [51] "Laos"                   "Malaysia"
##  [53] "Nepal"                  "Pakistan"
##  [55] "Philippines"            "Singapore"
##  [57] "Thailand"               "Vietnam"
##  [59] "Djibouti"               "Algeria"
##  [61] "Angola"                 "Botswana"
##  [63] "Burundi"                "Cameroon"
##  [65] "CentralAfricanRepublic" "Chad"
##  [67] "Comoros"                "Congo"
##  [69] "Zaire"                  "Benin"
##  [71] "EquatorialGuinea"       "Ethiopia"
##  [73] "Gabon"                  "Gambia"
##  [75] "Ghana"                  "GuineaBissau"
##  [77] "Guinea"                 "Coted'Ivoire"
##  [79] "Kenya"                  "Lesotho"
##  [81] "Liberia"                "Libya"
##  [83] "Madagascar"             "Malawi"
##  [85] "Mali"                   "Mauritania"
##  [87] "Mauritius"              "Morocco"
##  [89] "Mozambique"             "Niger"
##  [91] "Nigeria"                "Zimbabwe"
##  [93] "Rwanda"                 "Senegal"
##  [95] "SierraLeone"            "Somalia"
##  [97] "Namibia"                "Sudan"
##  [99] "Swaziland"              "Tanzania"
## [101] "Togo"                   "Tunisia"
## [103] "Uganda"                 "BurkinaFaso"
## [105] "Zambia"                 "Fiji"
## [107] "PapuaNewGuinea"         "Armenia"
## [109] "Azerbaijan"             "Belarus"
## [111] "Albania"                "Georgia"
## [113] "Kazakhstan"             "KyrgyzRepublic"
## [115] "Bulgaria"               "Moldova"
## [117] "Russia"                 "Tajikistan"
## [119] "China"                  "Turkmenistan"
## [121] "Ukraine"                "Uzbekistan"
## [123] "Cuba"                   "Czechoslovakia"
## [125] "Estonia"                "Latvia"
## [127] "Hungary"                "Lithuania"
## [129] "Mongolia"               "Croatia"
## [131] "Slovenia"               "Poland"
## [133] "Romania"
```

```
# Displays all of the empirically observed unique values Is this useful for
# continuous variables?

head(LDC$polityiv_update2)
```

```
## [1] 18  8  8 19 19 19
```

```
# Returns the first five values of a vector Also, tail() returns the last
# five values

names(LDC)
```

```
##  [1] "country"           "ctylabel"          "date"
##  [4] "gatt_wto_new"      "aclpn"             "bpc1"
##  [7] "dopen_wacz2"       "ecris2"            "fdignp"
## [10] "gdp_pc_95d"        "l1aclpn"           "l1bpc1"
## [13] "l1ecris2"         "newtar"           "polityiv_update2"
## [16] "signed"           "yrsoffic"         "usheg"
## [19] "l1usheg"          "l1fiveop"         "l1gdp_pc"
## [22] "avsw"             "avnewtar"         "l1avsw"
## [25] "l1avnewtar"       "lnpop"            "l1lnpop"
## [28] "l1office"         "l1partyage2000"   "l1fdi"
## [31] "l1polity"         "l2polity"         "l3polity"
## [34] "l1signed"         "milit2"           "sp2"
## [37] "pers2"            "l1milit2"         "l1sp2"
## [40] "dictator1"        "l1dictator1"      "yr70"
## [43] "yr80"             "l1ssch"           "closedyr"
## [46] "_spline1"         "_spline2"         "_spline3"
## [49] "l1gatt_wto_new"   "polityiv_squared" "l1polity_squared"
## [52] "log_gdp_pc"       "incomelevel"      "major_crisis"
## [55] "politychange"
```

```r
# Returns the variable names of a data frame colnames() does the same

class(LDC$ctylabel)
```

```
## [1] "character"
```

```r
# Show the classification of a variable

char_newtar = as.character(LDC$newtar)
# Changes a variable to a character variable

val_newtar = as.numeric(LDC$char_newtar)
# Changes a variable to a numeric variable

quantile(LDC$newtar, p = c(0.1, 0.9), na.rm = T)
```

```
##   10%   90%
##  8.38 40.06
```

```r
# Displays the 10th and 90th percentile of a variable
```

5. Two-Stage Least Squares (2SLS) Regression

Two-stage least squares regression is an important tool when we believe that there is a high level of endogeneity in our model. Endogeneity refers to a situation in which there is a mutual influence of our dependent and independent variables (feedback loop).

In one of our last sessions, we looked at the influence of economic development on the level of democracy. But what if there is (on average) better economic growth in democratic political systems? This would result in a feedback loop as described above.

Let us use the LDC dataset to illustrate 2SLS regression.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/")
library(foreign)
LDC = read.dta("LDC_IO_replication.dta")
model1 = lm(polityiv_update2 ~ l1gdp_pc, data = LDC)
summary(model1)
```

```
##
## Call:
## lm(formula = polityiv_update2 ~ l1gdp_pc, data = LDC)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.161  -5.841  -2.886   8.051  11.178
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.205e+00  1.588e-01  -7.584   4.6e-14 ***
## l1gdp_pc     6.274e-05  3.404e-05   1.843   0.0654 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.157 on 2696 degrees of freedom
##   (2672 observations deleted due to missingness)
## Multiple R-squared:  0.001258,   Adjusted R-squared:  0.000888
## F-statistic: 3.397 on 1 and 2696 DF,  p-value: 0.06543
```

In this regression we might have the problem of endogeneity.

When we use 2SLS regression, we identify an instrument that we believe to not be correlated with our dependent variable of interest, but to have an influence on our independent variable. With this method, we can isolate the effect that our independent variable has through our instruments.

We will use a package called "Zelig" to estimate a 2SLS regression.

Y = Democracy Level (dependent variable)

X = GDP per capita (explanatory variable)

I = FDI inflows (instrument variable)

Note that these claims contradict the theoretical claims made in tutorial 7. So please just treat this as a demonstration of how to set up such a regression.

install.packages("Zelig")

library(Zelig)

```
formula_inst <- list(mu1 = polityiv_update2 ~ l1gdp_pc, mu2 = l1gdp_pc ~ l1fdi,
    inst = ~l1fdi)
model2 = zelig(formula = formula_inst, model = "twosls", data = LDC)
```

```
## Loading required package: systemfit
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
```

```
## The following object is masked from 'package:reshape':
##
##     expand
##
## Loading required package: car
##
## Attaching package: 'car'
##
## The following object is masked from 'package:boot':
##
##     logit
##
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Warning in systemfit(data = data, formula = formula, method = method, inst
## = inst, : the estimation of systems of equations with unequal numbers of
## observations has not been thoroughly tested yet
```

```
##
##
##  How to cite this model in Zelig:
##   Ferdinand Alimadhi, Ying Lu, and Elena Villalon. 2015.
##   "twosls"
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://gking.harvard.edu/zelig
##
```

```
summary(model2)
```

```
##
## systemfit results
## method: 2SLS
##
##           N   DF        SSR   detRCov   OLS-R2 McElroy-R2
## system 5217 5213 7452859889 112809845 0.006978   0.543059
##
##        N   DF        SSR         MSE      RMSE        R2    Adj R2
## mu1 2376 2374     502127     211.511   14.5434 -3.228237 -3.230018
## mu2 2841 2839 7452357762 2624993.928 1620.1833  0.007029  0.006679
##
## The covariance matrix of the residuals
##           mu1       mu2
## mu1    211.511  -19734.4
## mu2 -19734.354 2374603.4
##
## The correlations of the residuals
##           mu1       mu2
## mu1  1.000000 -0.882904
## mu2 -0.882904  1.000000
##
##
## 2SLS estimates for 'mu1' (equation 1)
## Model Formula: polityiv_update2 ~ l1gdp_pc
## Instruments: ~l1fdi
##
##                 Estimate   Std. Error  t value Pr(>|t|)
## (Intercept) -14.73013136   9.48794685 -1.55251  0.12067
## l1gdp_pc      0.00964792   0.00664071  1.45284  0.14640
##
## Residual standard error: 14.543415 on 2374 degrees of freedom
## Number of observations: 2376 Degrees of Freedom: 2374
## SSR: 502126.957721 MSE: 211.510934 Root MSE: 14.543415
## Multiple R-Squared: -3.228237 Adjusted R-Squared: -3.230018
##
##
## 2SLS estimates for 'mu2' (equation 2)
## Model Formula: l1gdp_pc ~ l1fdi
## Instruments: ~l1fdi
##
##               Estimate Std. Error  t value   Pr(>|t|)
## (Intercept) 1500.47988   30.69836 48.87817 < 2.22e-16 ***
## l1fdi         26.37012    5.59478  4.71334 2.5543e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1620.183301 on 2839 degrees of freedom
## Number of observations: 2841 Degrees of Freedom: 2839
## SSR: 7452357762.23601 MSE: 2624993.928227 Root MSE: 1620.183301
## Multiple R-Squared: 0.007029 Adjusted R-Squared: 0.006679
```

How would we interpret these results? How are they different from the results obtained by OLS?

An alternative approach is using the ivreg function in R. For this we need the AER package.

install.packages("AER")

```r
library(AER)
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:boot':
##
##     aml
```

```r
model3 = ivreg(polityiv_update2 ~ l1gdp_pc | l1fdi, data = LDC)
summary(model3)
```

```
##
```

```
## Call:
## ivreg(formula = polityiv_update2 ~ l1gdp_pc | l1fdi, data = LDC)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.324  -4.546   3.837   7.467  21.753
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.730131   9.487947  -1.553    0.121
## l1gdp_pc      0.009648   0.006641   1.453    0.146
##
## Residual standard error: 14.54 on 2374 degrees of freedom
## Multiple R-Squared: -3.228,  Adjusted R-squared: -3.23
## Wald test: 2.111 on 1 and 2374 DF,  p-value: 0.1464
```

The instrument here is on the right (l1fdi).