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

Jan Vogler (jan.vogler@duke.edu)

October 23, 2015

Today's Agenda

1. Data management I: reading 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-Stages Least Squares (2SLS)

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

Don't forget that in most cases you will need to

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.

```
load(filename.Rdata)
```

```
## Error in load(filename.Rdata): object 'filename.Rdata' not found
```

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

```
### .csv files
csvdata = read.csv("filename.csv", stringsAsFactors = FALSE)
```

```
## Warning in file(file, "rt"): cannot open file 'filename.csv': No such file
## or directory
```

```
## Error in file(file, "rt"): cannot open the connection
```

```
# 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")
```

```
## Warning in file(file, "rt"): cannot open file 'filename.txt': No such file
## or directory
```

```
## Error in file(file, "rt"): cannot open the connection
```

Additionally, there are many other data formats. Some of them require the foreign package.

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

```
## Error in read.spss("filename", to.data.frame = TRUE): unable to open file: 'No such file or director
```

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

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

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

```
## Error in eval(expr, envir, enclos): could not find function "read.dta13"
```

It is most likely that you will need to use more than one dataset for your empirical analysis. You can merge two datasets by using the merge command.

```
merge(dataset1, dataset2, by = c("Country", "Year"))
```

```
## Error in merge(dataset1, dataset2, by = c("Country", "Year")): object 'dataset1' not found
```

To delete data, we simply use the following command:

```
LDC$ecris2 = NULL
```

We can also only keep some data that we really need:

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

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)  #383 observations remaining
```

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
```

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. What happens if we merge these two dataframes?

```
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.

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?

```
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
##  Min.   :1    Min.   : 0.00
##  1st Qu.:1    1st Qu.:17.05
##  Median :1    Median :24.69
##  Mean   :1    Mean   :22.28
##  3rd Qu.:1    3rd Qu.:28.25
##  Max.   :1    Max.   :30.52
##  NA's   :12
```

```
### We are only left with all observations for Angola.
```

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

```
ind.files <- dir(pattern = "ind\\d+\\.sav")  # Grab file names
ind.data <- lapply(ind.files, read.spss)  # Create list with each file's data
names(ind.data) <- gsub("\\.sav", "", ind.files)  # Name list elements by survey wave
```

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.

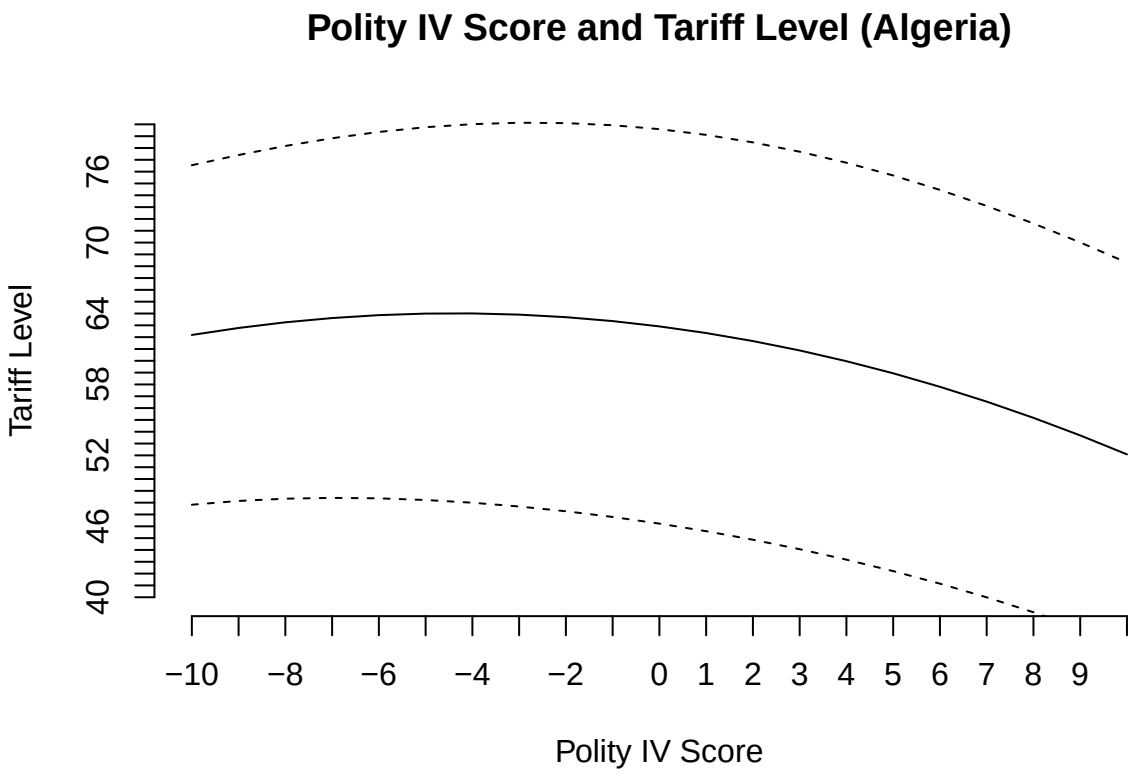
```
LDC$polityiv_squared = (LDC$polityiv_update2)^2
LDC$l1polity_squared = (LDC$l1polity)^2
```

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 +
# 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?

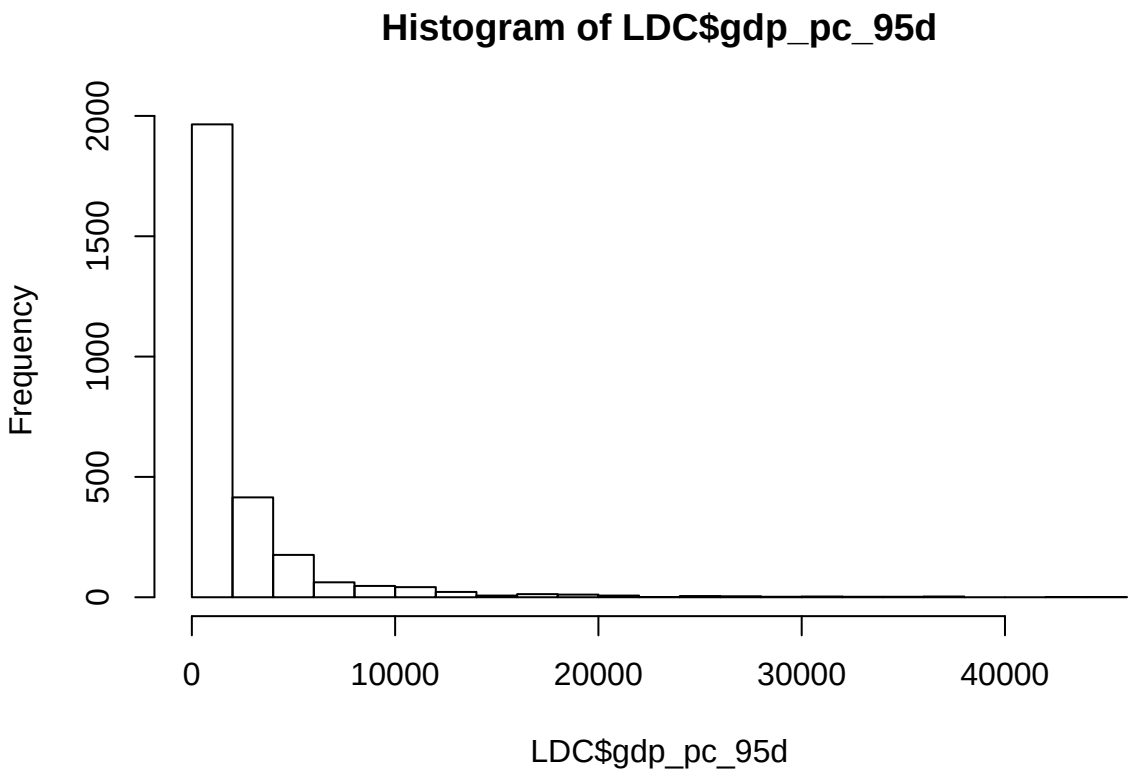
```
nd <- data.frame(l1polity = seq(-10, 10, by = 1), l1polity_squared = seq(-10,
    10, by = 1)^2, l1signed = rep(0.1511, 21), l1office = rep(8.431, 21), l1gdp_pc = rep(2888,
    21), l1lnpop = rep(15.1, 21), l1ecris2 = rep(0.0641, 21), l1bpc1 = rep(0.5909,
    21), l1avnewtar = rep(14.91, 21), ctylabel = 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(40, 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(40, 80), labels = seq(40, 80))
matlines(cis[, c(2, 3)], lty = 2, col = "black")
```



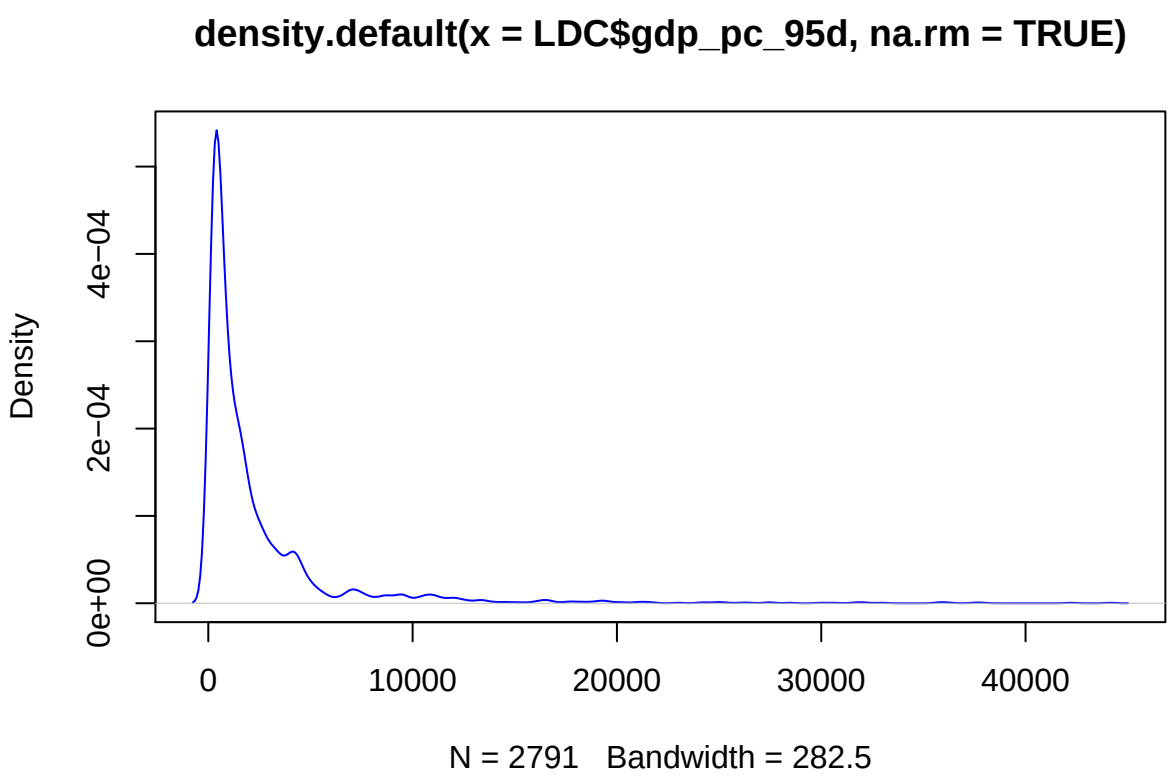
This relationship looks slightly different than the relationship estimated by Milner and Kubota.

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.

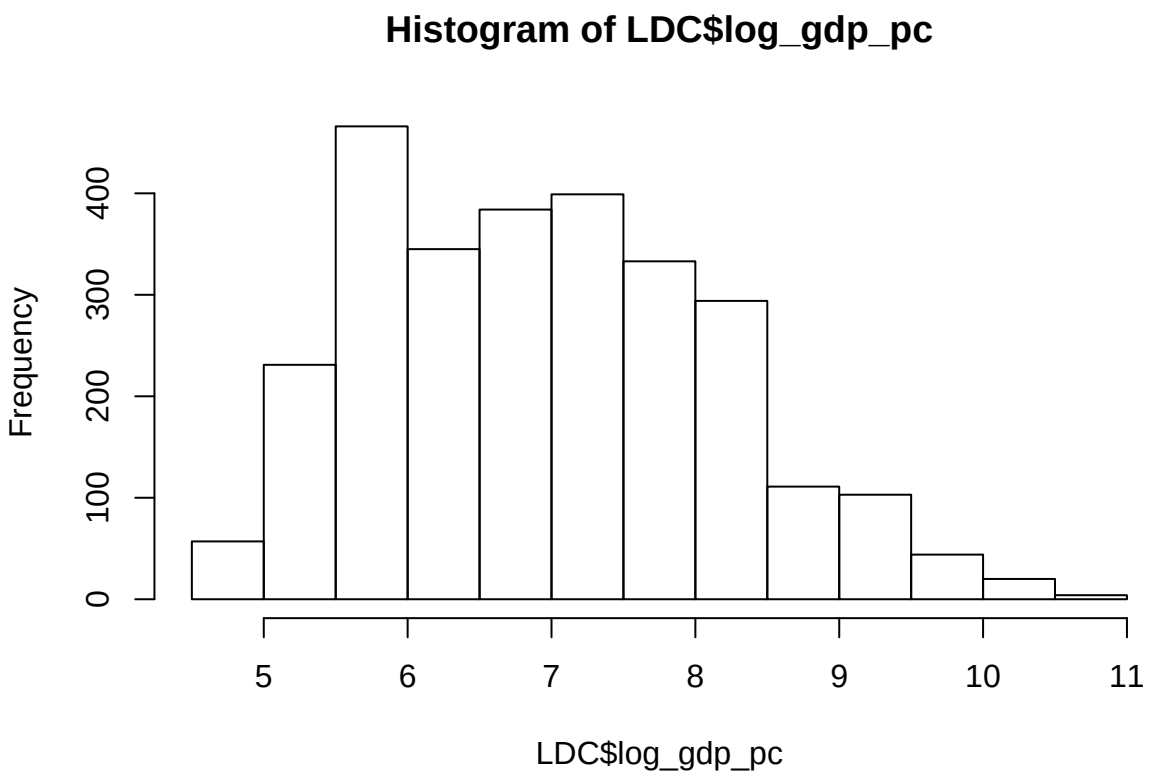
```
hist(LDC$gdp_pc_95d, breaks = 21)
```



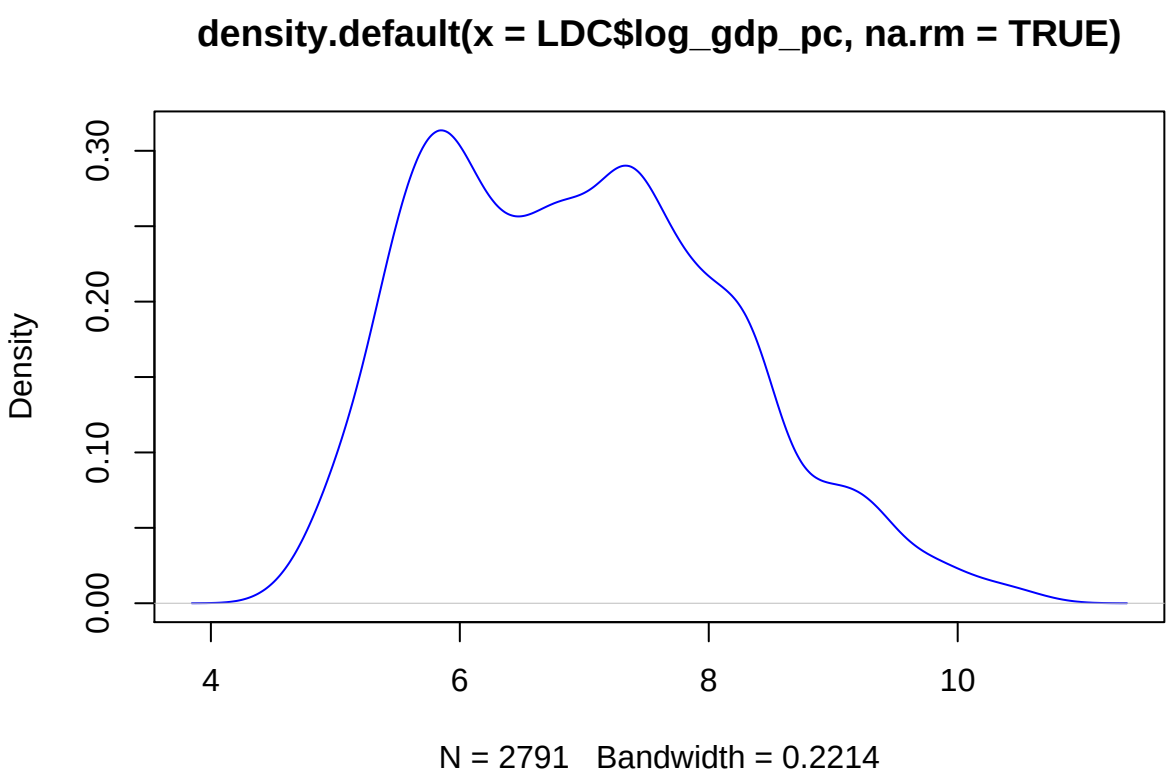

```
dens = density(LDC$gdp_pc_95d, na.rm = TRUE)
plot(dens, col = "blue")
```




```
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
```

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"    "Middle-income country" NA
```

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

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

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

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

```r
### 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.

```r
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)
```

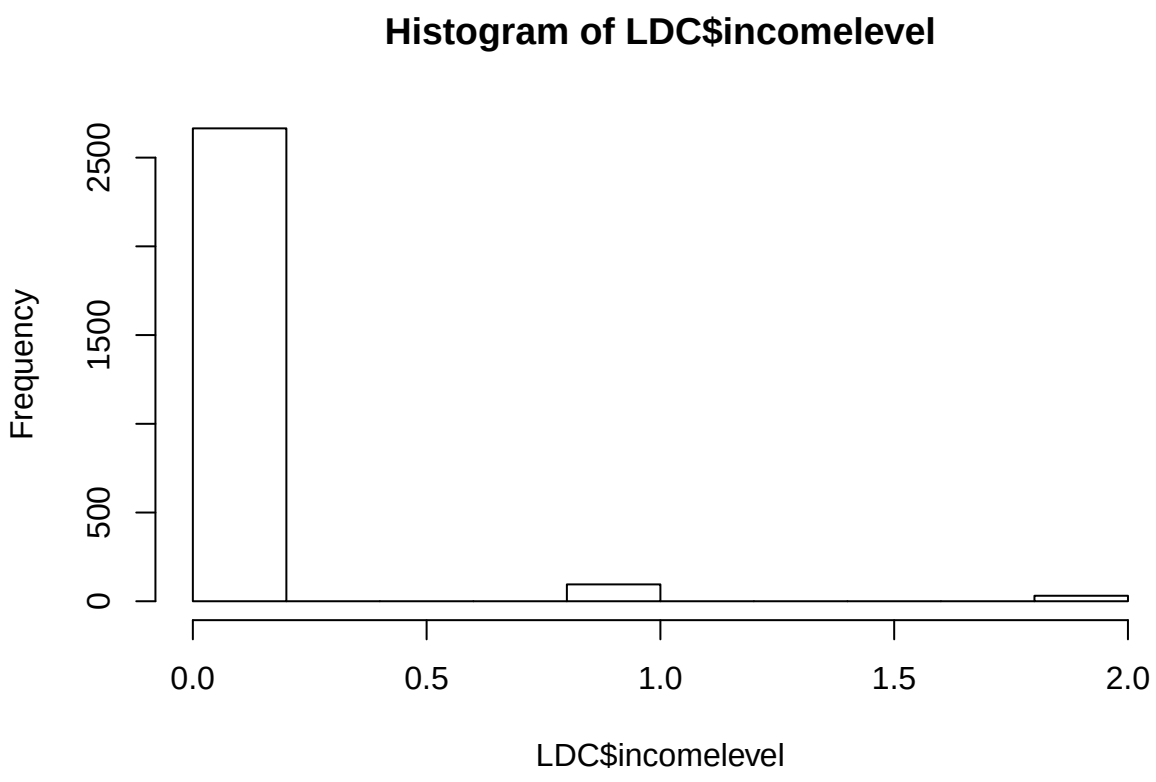
```
## Error in hist.default(LDC$incomelevel): 'x' must be numeric
```

```r
### 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:

```r
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 == 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?

```
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
        }
    }
}
```

The "car" package has a recode command that you might find useful for your own work. Let us recode every economic crisis from a value of "1" to a value of "2". Quiz question: in a linear model, which effect would that have on the coefficient of the variable? (Note: the variable is binary)

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.2.2
```

```
# LDC$ecris2 = recode(LDC$ecris2,'1='2'')
```

4. Data management IV: useful commands

One of the most useful functions in R is the apply function. This function allows you to apply a function to all rows (=1) or all columns of a dataframe (=2). Let's see how this could be useful for recoding data.

```
vector1 = c("A", "B", "B", "B")
vector2 = c("B", "A", "C", "A")
matrix = rbind(vector1, vector2)
```

Next we create a function that has three inputs: a vector x, an old variable value, and a new value.

```
recode = function(x, old = "A", new = "fun") {
    x[x == old] = new
}
```

Now let's apply this function to all columns.

```
apply(matrix, 2, recode)
```

```
## [1] "fun" "fun" "fun" "fun"
```

```
matrix
```

```
##         [,1] [,2] [,3] [,4]
## vector1 "A"  "B"  "B"  "B"
## vector2 "B"  "A"  "C"  "A"
```

Here are some more useful commands.

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

```
##   [1] "Turkey"                "SouthAfrica"
##   [3] "Argentina"             "Bolivia"
##   [5] "Brazil"                "Chile"
##   [7] "Colombia"              "CostaRica"
##   [9] "DominicanRepublic"     "Ecuador"
##  [11] "ElSalvador"            "Guatemala"
##  [13] "Haiti"                 "Honduras"
##  [15] "Mexico"                "Nicaragua"
##  [17] "Panama"                "Paraguay"
##  [19] "Peru"                  "Uruguay"
##  [21] "Venezuela"             "Guyana"
##  [23] "Jamaica"               "Trinidad&Tobago"
##  [25] "Bahrain"               "Cyprus"
##  [27] "Iran"                  "Israel"
##  [29] "Jordan"                "Kuwait"
##  [31] "Oman"                  "SaudiArabia"
##  [33] "Syria"                 "UnitedArabEmirates"
##  [35] "Egypt"                 "Bangladesh"
##  [37] "Bhutan"                "Cambodia"
##  [39] "SriLanka"              "India"
##  [41] "Indonesia"             "Korea"
##  [43] "Laos"                  "Malaysia"
##  [45] "Nepal"                 "Pakistan"
##  [47] "Philippines"           "Singapore"
##  [49] "Thailand"              "Djibouti"
##  [51] "Algeria"               "Angola"
##  [53] "Botswana"              "Burundi"
##  [55] "Cameroon"              "CentralAfricanRepublic"
##  [57] "Chad"                  "Comoros"
##  [59] "Congo"                 "Zaire"
##  [61] "Benin"                 "EquatorialGuinea"
##  [63] "Ethiopia"              "Gabon"
##  [65] "Gambia"                "Ghana"
##  [67] "GuineaBissau"          "Guinea"
##  [69] "Coted'Ivoire"          "Kenya"
##  [71] "Lesotho"               "Madagascar"
##  [73] "Malawi"                "Mali"
##  [75] "Mauritania"            "Mauritius"
```

```
##  [77] "Morocco"                "Mozambique"
##  [79] "Niger"                  "Nigeria"
##  [81] "Zimbabwe"               "Rwanda"
##  [83] "Senegal"                "SierraLeone"
##  [85] "Namibia"                "Swaziland"
##  [87] "Tanzania"               "Togo"
##  [89] "Tunisia"                "Uganda"
##  [91] "BurkinaFaso"            "Zambia"
##  [93] "Fiji"                   "PapuaNewGuinea"
##  [95] "Armenia"                "Azerbaijan"
##  [97] "Belarus"                "Albania"
##  [99] "Georgia"                "Kazakhstan"
## [101] "KyrgyzRepublic"         "Bulgaria"
## [103] "Moldova"                "Russia"
## [105] "Tajikistan"             "China"
## [107] "Turkmenistan"           "Ukraine"
## [109] "Uzbekistan"             "Estonia"
## [111] "Latvia"                 "Hungary"
## [113] "Lithuania"              "Mongolia"
## [115] "Croatia"                "Slovenia"
## [117] "Poland"                 "Romania"
```

```
# Displays all of the empirically observed values

head(LDC$fdignp)
```

```
## NULL
```

```
# Returns the first five values

names(LDC)
```

```
##  [1] "ctylabel"         "date"             "polityiv_update2"
##  [4] "gdp_pc_95d"       "newtar"           "l1polity"
##  [7] "l1polity.1"       "l1signed"         "l1office"
## [10] "l1gdp_pc"         "l1lnpop"          "l1ecris2"
## [13] "l1bpc1"           "l1avnewtar"       "polityiv_squared"
## [16] "l1polity_squared" "log_gdp_pc"       "incomelevel"
## [19] "major_crisis"     "politychange"
```

```
# Returns the variable names of a data frame

class(LDC$ctylabel)
```

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

```
# 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)
```

```r
# Changes a variable to a numeric variable

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

```
##  10%  90%
##  8.5 40.0
```

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

5. Two-Stages Least Squares (2SLS)