Tutorial 7: Data management

Johannes Karreth

RPOS 517, Day 7

Contents

This tutorial shows you:

- a shortcut to import data in various formats into R
- how to clean raw data from a public source
- how to process data for analysis
- how to merge separate data sources into one
- how to collapse data sets
- how to create new variables, including time-series operators
- how to read in data from a public Dropbox URL

Note on copying & pasting code from the PDF version of this tutorial: Please note that you may run into trouble if you copy & paste code from the PDF version of this tutorial into your R script. When the PDF is created, some characters (for instance, quotation marks or indentations) are converted into non-text characters that R won't recognize. To use code from this tutorial, please type it yourself into your R script or you may copy & paste code from the *source file* for this tutorial which is posted on my website.

Note on R functions discussed in this tutorial: I don't discuss many functions in detail here and therefore I encourage you to look up the help files for these functions or search the web for them before you use them. This will help you understand the functions better. Each of these functions is well-documented either in its help file (which you can access in R by typing `?ifelse`, for instance) or on the web. The *Companion to Applied Regression* (see our syllabus) also provides many detailed explanations.

Entering your own data into a spreadsheet

Since many of you work with data you collect yourself, you will often enter data directly into spreadsheets. I don't have any specific recommendations for this, but I strongly encourage you to use the following workflow:

- enter raw data into spreadsheets, then save them as .csv files
- do all and any processing (as you learn below) in R with a documented script
- This makes your data cleaning and processing reproducible for yourself and for others in the future
- It will also prevent you from being unable to reconstruct what you did down the road

Remember, one of the goals of our seminar is to help you build a better relationship with your future self. Following this workflow will help with that.

Keep data management and analysis separate

In all you work, you should keep a separate script file for data management and data analysis. This will also help you maintain a reproducible workflow and keep your code manageable. For instance, in my projects, I typically have at least two R scripts in my project folder:

- `project.datamgmt.R`, which starts by importing the original source data and cleans and prepares it for analysis
- `project.analysis.R`, which conducts all analysis and creates tables and graphs

Importing and exporting data

So far, you've used the following commands to read data into R; all of them were explained in the tutorial for Day 1:

- `read.csv()` to read a comma-separated values spreadsheet
- `read.xls()` (from the "gdata" package) to read an Excel spreadsheet
- `read.dta()` (from the "foreign" package) to read a dataset in Stata format

The rio package

You might also encounter data in other formats when you do your own research. For this purpose, the R package "rio" is particularly useful. Its developer describes it as "a set of tools aims to simplify the process of importing/exporting data." The package has two main functions, `import()` and `export()`. It allows you to import and export data from/to the following formats:

- Tab-separated data (.tsv)
- Comma-separated data (.csv)
- Pipe-separated data (.psv)
- Fixed-width format data (.fwf)
- Saved R objects (.RData)
- Stata (.dta)
- SPSS and SPSS portable (.sav and .por) | Yes | Yes (.sav only) |
- "XBASE" database files (.dbf)
- Excel (.xlsx)
- SAS and SAS XPORT (.sas7bdat and .xpt)
- Minitab (.mtp)
- Epiinfo (.rec)
- Clipboard (as tab-separated)

For more information, see a readme page for the "rio" package on Github: https://github.com/leeper/rio.

Example: importing an SPSS data set

In this example, we import a dataset from the Afrobarometer project into R. The Afrobarometer is an African-led series of national public attitude surveys on democracy and governance in Africa, and you can find more information on it at http://www.afrobarometer.org/. The survey data are provided to scholars in SPSS format. SPSS is a statistical software package akin to Stata or R. At http://www.afrobarometer.org/data/data-rounds-merged, you can find a download link for the fourth round of the Afrobarometer. The file is called "merged_r4_data.sav". Let's use this link to read this dataset into R using the `import()` function from the "rio" package.

First, install the "rio" package. You only have to do this once.

```
install.packages("rio")
```

Next, load the package.

```
library("rio")
```

Now you can import the Afrobarometer dataset in your R environment. I'll call it `ab`. This will take a few seconds since the dataset is over 15MB big. **Note: as always, replace the file path below with the location of the file on your personal computer.**

```
ab <- import(file = "/Users/johanneskarreth/Documents/Uni/Teaching/POS 517/Tutorials/Day 7 - Data manag
```

Before actually looking at the dataset itself, you should look at its dimensions.

```
dim(ab)
```

```
## [1] 27713   294
```

This is a fairly large dataset, so let's trim it down a bit. From the codebook at http://www.afrobarometer.org/files/documents/codebook/merged_r4_codebook3.pdf, I determine that I'll be interested only in the following variables:

- COUNTRY
- URBRUR: Urban or Rural Primary Sampling Unit
- Q42A: In your opinion how much of a democracy is [Ghana/Kenya/etc.]? today?
- Q89: What is the highest level of education you have completed?
- Q101: Respondent's gender
- Q1: Respondent's age

So we use R's indexing structure (see the tutorial for Day 2) to create a new object, `ab.small`, that contains only these six variables.

```
ab.small <- ab[, c("COUNTRY", "URBRUR", "Q42A", "Q89", "Q101", "Q1")]
dim(ab.small)
```

```
## [1] 27713     6
```

```
summary(ab.small)
```

```
##          COUNTRY        URBRUR
##  Uganda      : 2431   Urban:10521
##  South Africa: 2400   Rural:17192
##  Nigeria     : 2324
##  Madagascar  : 1350
##  Cape Verde  : 1264
##  Mali        : 1232
##  (Other)     :16712
##                                    Q42A
##  A democracy, but with minor problems:8249
##  A democracy, with major problems    :7338
##  A full democracy                    :7310
##  Not a democracy                     :1875
##  Don't know                          :1724
##  Do not understand question/democracy:1202
##  (Other)                             :  15
##                                          Q89            Q101
##  Some secondary school/high school         :5950   Missing:    0
##  Some primary schooling                    :5111   Male   :13837
##  No formal schooling                       :4365   Female :13876
##  Secondary school completed/high school    :4165
```

```
##  Primary school completed                      :3897
##  Post-secondary qualifications, not university:1674
##  (Other)                                       :2551
##        Q1
##  Min.   : -1.00
##  1st Qu.: 25.00
##  Median : 33.00
##  Mean   : 47.68
##  3rd Qu.: 45.00
##  Max.   :999.00
##
```

Now that we've read the dataset into R, we can process it for further analysis - which we'll do further below in this tutorial.

Example: importing a Stata dataset

For this example, we use the European Social Survey, an academically driven cross-national survey that has been conducted every two years across Europe since 2001. You can find more information on the ESS at http://www.europeansocialsurvey.org/ (under Data and Documentation > Round 6) after you register on the site to access data and codebooks. Let's download the Stata version of the 2012 round of the ESS, called "ESS6e02_1.dta", and use the `import()` function again to read the dataset into R.

```
ess <- import(file = "/Users/johanneskarreth/Documents/Uni/Teaching/POS 517/Tutorials/Day 7 - Data manag
```

Before actually looking at the dataset itself, you should look at its dimensions.

```
dim(ess)
```

```
## [1] 54673   626
```

This is again a fairly large dataset, so let's trim it down a bit. From the variable list at http://www.europeansocialsurvey.org/docs/round6/survey/ESS6_appendix_a7_e02_1.pdf, I decide that I'll be interested only in the following variables:

- cntry: Country
- trstlgl: Trust in the legal system, 0 means you do not trust an institution at all, and 10 means you have complete trust.
- lrscale: Placement on left right scale, where 0 means the left and 10 means the right
- fairelc: How important R thinks it is for democracy in general that national elections are free and fair
- yrbrn: Year of birth
- gndr: Gender
- hinctnta: Household's total net income, all sources

Again, we use R's indexing structure to create a new object, `ess.small`, that contains only these seven variables.

```
ess.small <- ess[, c("cntry", "trstlgl", "lrscale", "fairelc", "yrbrn", "gndr", "hinctnta")]
dim(ess.small)
```

```
## [1] 54673     7
```

```
summary(ess.small)
```

```
##     cntry                        trstlgl            lrscale
##  Length:54673       5              : 7893   5         :15400
##  Class :character   8              : 6311   Don't know: 7489
##  Mode  :character   7              : 6171   7         : 4942
##                     No trust at all: 5959   6         : 4323
##                     3              : 5178   4         : 4269
##                     6              : 5130   8         : 3987
##                     (Other)        :18031   (Other)   :14263
##                                          fairelc          yrbrn
##  Extremely important for democracy in general:31800   Min.   :1909
##  9                                             : 6834   1st Qu.:1950
##  8                                             : 6396   Median :1964
##  7                                             : 3213   Mean   :1981
##  5                                             : 1945   3rd Qu.:1980
##  6                                             : 1718   Max.   :9999
##  (Other)                                       : 2767
##         gndr                 hinctnta
##  Male     :24929   Refusal       : 6211
##  Female   :29727   R - 2nd decile: 5492
##  No answer:   17   J - 1st decile: 5063
##                    C - 3rd decile: 5011
##                    M - 4th decile: 4888
##                    F - 5th decile: 4538
##                    (Other)       :23470
```

Now that we've read the dataset into R, we can process it for further analysis. We won't revisit the ESS data in this tutorial, but you will encounter it in your in-class assignment on Day 7.

Data cleaning

Before any analysis, you will often, if not always, need to process data that you obtained from elsewhere or that you collected yourself. In this section, we'll go over some typical scenarios for this.

Often, you need to make sure that the variables have the correct numerical or character values. Different data sources often use different codes for missing values, for instance. Let's work through this with the `ab.small` dataset. First, I check the structure of the dataset:

```
str(ab.small)
```

```
## 'data.frame':    27713 obs. of  6 variables:
##  $ COUNTRY: Factor w/ 20 levels "Benin","Botswana",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ URBRUR : Factor w/ 2 levels "Urban","Rural": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Q42A   : Factor w/ 8 levels "Missing","Not a democracy",..: 5 5 5 4 3 4 3 4 4 4 ...
##  $ Q89    : Factor w/ 13 levels "Missing","No formal schooling",..: 6 4 6 5 6 6 7 4 6 6 ...
##  $ Q101   : Factor w/ 3 levels "Missing","Male",..: 3 2 3 2 3 2 3 2 2 3 ...
##  $ Q1     : atomic  38 46 28 30 23 24 40 50 24 36 ...
##   ..- attr(*, "value.labels")= Named chr  "999" "998" "-1"
##   .. ..- attr(*, "names")= chr  "Don't know" "Refused" "Missing"
```

It looks like we're dealing with a number of factor variables here. Let's check each of them individually. Because they are factors, the best way to look at all their values is to use the `table()` function.

```
table(ab.small$COUNTRY)
```

```
##
##        Benin     Botswana Burkina Faso   Cape Verde        Ghana
##         1200         1200         1200         1264         1200
##        Kenya      Lesotho      Liberia   Madagascar       Malawi
##         1104         1200         1200         1350         1200
##         Mali   Mozambique      Namibia      Nigeria      Senegal
##         1232         1200         1200         2324         1200
## South Africa     Tanzania       Uganda       Zambia     Zimbabwe
##         2400         1208         2431         1200         1200
```

These are all proper country names, so we can move on to the next variable.

1. What do the numbers under the country names tell you?

```
table(ab.small$URBRUR)
```

```
##
## Urban Rural
## 10521 17192
```

This is a factor with 2 levels and no missing values, so we can move on to the next variable.

2. How would you calculate the percentage of respondents live in urban versus rural areas?

```
table(ab.small$Q42A)
```

```
##
##                              Missing                      Not a democracy
##                                   15                                 1875
##     A democracy, with major problems A democracy, but with minor problems
##                                 7338                                 8249
##                     A full democracy Do not understand question/democracy
##                                 7310                                 1202
##                           Don't know                              Refused
##                                 1724                                    0
```

This variable needs some work. First, we need to recode all values that are currently listed as "Missing" to R's default code for missing, `NA`. For this, we can use the `ifelse()` function (see the tutorial for Day 4). Note that we create a new object for our recoded variable. This is good practice: always leave the source variable untouched and document your recoding command. This way, you can always retrace your steps later and share your work with other researchers. I name the new variable `perceivedDem`. Below, I use several steps to code all answers that do not rank the country's democracy as missing. Depending on your analysis, of course, you may want to use the "Don't know" answers in a different way. In the code below, you see the `|` symbol: this stands for "or", just as `&` stands for "and".

```
ab.small$perceivedDem <- ifelse(ab.small$Q42A == "Missing" |
                                  ab.small$Q42A == "Don't know" |
                                  ab.small$Q42A == "Refused" |
                                  ab.small$Q42A == "Do not understand question/democracy",
                                  NA, ab.small$Q42A)
table(ab.small$perceivedDem)
```

```
##
##    2    3    4    5
## 1875 7338 8249 7310
```

Note that `NA` is not set in quotation marks because it is a specific value and not a string. This is important. We now have numeric values, but we may want to recode them so that 0 is the lowest value and 3 the highest.

```
ab.small$perceivedDem <- ab.small$perceivedDem - 2
table(ab.small$perceivedDem)
```

```
##
##    0    1    2    3
## 1875 7338 8249 7310
```

Now let's make the education variable an ordinal numerical variable. First, let's have a look at the variable in its current form:

```
table(ab.small$Q89)
```

```
##
##                                          Missing
##                                               10
##                                No formal schooling
##                                             4365
##                             Informal schooling only
##                                             1260
##                              Some primary schooling
##                                             5111
##                            Primary school completed
##                                             3897
##                 Some secondary school/high school
##                                             5950
##            Secondary school completed/high school
##                                             4165
## Post-secondary qualifications, not university
##                                             1674
##                                   Some university
##                                              649
##                               University completed
##                                              506
##                                    Post-graduate
##                                               92
##                                       Don't know
##                                               34
##                                          Refused
##                                                0
```

We should first recode "Missing", "Don't know", and "Refused" to `NA`. Again, we create a new variable, education.

```
ab.small$education <- ifelse(ab.small$Q89 == "Missing" |
                                  ab.small$Q89 == "Don't know" |
                                  ab.small$Q89 == "Refused",
                                  NA, ab.small$Q89)
table(ab.small$education)
```

```
##
##    2    3    4    5    6    7    8    9   10   11
## 4365 1260 5111 3897 5950 4165 1674  649  506   92
```

We now have numeric values, but we may want to recode them so that 0 is the lowest value and 3 the highest.

```
ab.small$education <- ab.small$education - 2
table(ab.small$education)
```

```
##
##    0    1    2    3    4    5    6    7    8    9
## 4365 1260 5111 3897 5950 4165 1674  649  506   92
```

If we wanted to put labels on the variable, we can use the `factor` function:

```
levels(ab.small$Q89)
```

```
##  [1] "Missing"
##  [2] "No formal schooling"
##  [3] "Informal schooling only"
##  [4] "Some primary schooling"
##  [5] "Primary school completed"
##  [6] "Some secondary school/high school"
##  [7] "Secondary school completed/high school"
##  [8] "Post-secondary qualifications, not university"
##  [9] "Some university"
## [10] "University completed"
## [11] "Post-graduate"
## [12] "Don't know"
## [13] "Refused"
```

```
ab.small$educationLevel <- factor(ab.small$education)
levels(ab.small$educationLevel) <- levels(ab.small$Q89)[2:11]
table(ab.small$educationLevel)
```

```
##
##                          No formal schooling
##                                         4365
##                       Informal schooling only
##                                         1260
##                        Some primary schooling
##                                         5111
```

```
##                       Primary school completed
##                                           3897
##             Some secondary school/high school
##                                           5950
##        Secondary school completed/high school
##                                           4165
## Post-secondary qualifications, not university
##                                           1674
##                                 Some university
##                                            649
##                            University completed
##                                            506
##                                   Post-graduate
##                                             92
```

Next, let's check the `Q101` variable for gender:

```
table(ab.small$Q101)
```

```
##
## Missing    Male  Female
##       0   13837   13876
```

We can't use the Missing value and want this variable to be numerical so that males are 0 and females 1. Here, we load the "car" library (the companion library to our textbook) and use the `recode()` function.

```
library(car)
ab.small$female <- recode(ab.small$Q101, "'Missing' = NA; 'Male' = 0; 'Female' = 1")
table(ab.small$female)
```

```
##
##     0     1
## 13837 13876
```

Lastly, let's check the age variable. Because this is already a numerical variable, let's use `str()` here and not make a table (because it would be too long.)

```
str(ab.small$Q1)
```

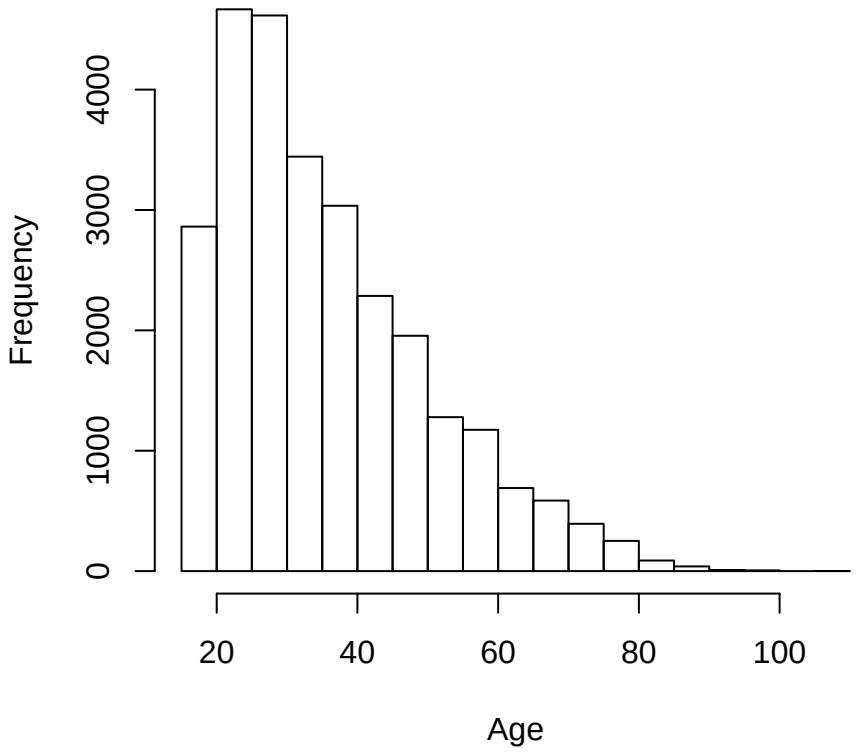
```
##  atomic [1:27713] 38 46 28 30 23 24 40 50 24 36 ...
##  - attr(*, "value.labels")= Named chr [1:3] "999" "998" "-1"
##   ..- attr(*, "names")= chr [1:3] "Don't know" "Refused" "Missing"
```

We can see that the values `999`, `998`, and `-1` correspond to "Don't know", "Refused", and "Missing". So let's recode those again to `NA` as we did with the democracy variable above:

```
ab.small$age <- ifelse(ab.small$Q1 == 999 |
                    ab.small$Q1 == 998 |
                    ab.small$Q1 == -1,
                    NA, ab.small$Q1)
summary(ab.small$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   18.00   25.00   33.00   36.33   45.00  110.00     333
```

```
hist(ab.small$age, main = "", xlab = "Age")
```



Now let's look at our dataset:

```
str(ab.small)
```

```
## 'data.frame':    27713 obs. of  11 variables:
##  $ COUNTRY       : Factor w/ 20 levels "Benin","Botswana",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ URBRUR        : Factor w/ 2 levels "Urban","Rural": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Q42A          : Factor w/ 8 levels "Missing","Not a democracy",..: 5 5 5 4 3 4 3 4 4 4 ...
##  $ Q89           : Factor w/ 13 levels "Missing","No formal schooling",..: 6 4 6 5 6 6 7 4 6 6 ...
##  $ Q101          : Factor w/ 3 levels "Missing","Male",..: 3 2 3 2 3 2 3 2 2 3 ...
##  $ Q1            : atomic  38 46 28 30 23 24 40 50 24 36 ...
##   ..- attr(*, "value.labels")= Named chr  "999" "998" "-1"
##   .. ..- attr(*, "names")= chr  "Don't know" "Refused" "Missing"
##  $ perceivedDem  : num  3 3 3 2 1 2 1 2 2 2 ...
##  $ education     : num  4 2 4 3 4 4 5 2 4 4 ...
##  $ educationLevel: Factor w/ 10 levels "No formal schooling",..: 5 3 5 4 5 5 6 3 5 5 ...
##  $ female        : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 1 2 ...
##  $ age           : num  38 46 28 30 23 24 40 50 24 36 ...
```

For our further analyses, let's keep only the variables we just created. For this, we create a new object, `ab.work` by keeping only the columns we need:

```
ab.work <- ab.small[, c("COUNTRY", "URBRUR", "perceivedDem", "female", "age",
                        "education", "educationLevel")]
```

Alternatively, we could also drop the columns we don't need. This might make more sense if you want to keep many and drop few variables:

```
ab.work <- ab.small
ab.work$Q42A <- NULL
ab.work$Q89 <- NULL
ab.work$Q101 <- NULL
ab.work$Q1 <- NULL
```

Finally, let's summarize our dataset:

```
summary(ab.work)
```

```
##          COUNTRY        URBRUR       perceivedDem     education
##  Uganda      : 2431   Urban:10521   Min.   :0.000   Min.   :0.000
##  South Africa: 2400   Rural:17192   1st Qu.:1.000   1st Qu.:2.000
##  Nigeria     : 2324                 Median :2.000   Median :3.000
##  Madagascar  : 1350                 Mean   :1.847   Mean   :3.154
##  Cape Verde  : 1264                 3rd Qu.:3.000   3rd Qu.:5.000
##  Mali        : 1232                 Max.   :3.000   Max.   :9.000
##  (Other)     :16712                 NA's   :2941    NA's   :44
##                                  educationLevel female         age
##  Some secondary school/high school     :5950   0:13837   Min.   : 18.00
##  Some primary schooling                :5111   1:13876   1st Qu.: 25.00
##  No formal schooling                   :4365             Median : 33.00
##  Secondary school completed/high school:4165             Mean   : 36.33
##  Primary school completed              :3897             3rd Qu.: 45.00
##  (Other)                               :4181             Max.   :110.00
##  NA's                                  :  44             NA's   :333
```

Data structure

In today's article, *Tidy Data*, you learned that the default structure of dataset consists of rows and columns. However, datasets typically have more underlying structure than just rows and columns. For instance, the `ab.small` data frame is properly structured so that each observation is one row and each variable is one column. But each observation comes from one of the 20 countries in the survey. You can therefore think as respondent i being also nested in country j.

Collapsing a dataset

This can become important when you want to summarize information across a group indicator j, such as countries in this case. What if you were interested in creating a country-level variable that measure the average perception of the country's political system as democratic, based on the `perceivedDem` variable we created from the Afrobarometer. To do this, you need to "collapse" the dataset by country and create a new dataset with only country names and the average value of `perceivedDem` by country.

For this operation, we will begin to make use of the "dplyr" package in R. This package is a great tool for "data wrangling" and will become on of your standard tools. On my website, I link to the Github page for this package, where you can find more information on how this package works.

First, you need to install the package (only once), and then load it (in every script where you use it). For collapsing the data, we will use the `summarize` and `group_by` functions.

3. If we collapse the `ab.work` data set by countries, how many rows will the resulting dataset have?

We'll create a new dataset named `ab.country`.

```
library(dplyr)
ab.country <- summarize(group_by(ab.work, COUNTRY),
                        perceivedDem = mean(perceivedDem, na.rm = TRUE))
```

Note the use of `na.rm = TRUE` within the `mean` function. The `perceivedDem` variable has missing values, and R cannot calculate a mean on a vector with missing values unless `na.rm` is set to `TRUE`.

4. What command/s would you use if you wanted to summarize this variable with the median instead of the mean? What if you wanted to calculate its range?

This is the dataset we just created:

```
ab.country
```

```
## Source: local data frame [20 x 2]
##
##         COUNTRY perceivedDem
## 1         Benin     2.190941
## 2      Botswana     2.488889
## 3  Burkina Faso     1.848024
## 4    Cape Verde     2.107989
## 5         Ghana     2.435315
## 6         Kenya     1.568047
## 7       Lesotho     1.509953
## 8       Liberia     1.912811
## 9    Madagascar     1.797376
## 10       Malawi     1.816530
## 11         Mali     1.947186
## 12   Mozambique     1.972603
## 13      Namibia     2.148464
## 14      Nigeria     1.427995
## 15      Senegal     1.574279
## 16 South Africa     1.816717
## 17     Tanzania     2.231054
## 18       Uganda     1.738606
## 19       Zambia     1.690975
## 20     Zimbabwe     1.057171
```

Adding a summarized variable to a dataset

If you wanted to simply add the average `perceivedDem` variable to your original survey data without collapsing it, you can do this using the exact same language, but substitute `mutate` for `summarize`.

```
ab.work <- mutate(group_by(ab.work, COUNTRY), perceivedDemAvg = mean(perceivedDem, na.rm = TRUE))
```

If we look at the data, you will notice a new variable on the right:

```
head(ab.work)
```

```
## Source: local data frame [6 x 8]
## Groups: COUNTRY
##
##   COUNTRY URBRUR perceivedDem education                   educationLevel
## 1   Benin  Urban            3         4 Some secondary school/high school
## 2   Benin  Urban            3         2            Some primary schooling
## 3   Benin  Urban            3         4 Some secondary school/high school
## 4   Benin  Urban            2         3          Primary school completed
## 5   Benin  Urban            1         4 Some secondary school/high school
## 6   Benin  Urban            2         4 Some secondary school/high school
##   female age perceivedDemAvg
## 1      1  38        2.190941
## 2      0  46        2.190941
## 3      1  28        2.190941
## 4      0  30        2.190941
## 5      1  23        2.190941
## 6      0  24        2.190941
```

Time-series cross-sectional data

In cross-country survey data, you can think of respondents i nested in countries j. When you work with macro-economic or macro-political data from several countries over multiple time points, you can similarly think of countries i and years t. One row in your data set is then a country-year it. In this context, you can use the same operations as above.

As an example, we'll introduce a handy data compendium from the University of Gothenburg, Sweden. The *Quality of Government* project, online at http://qog.pol.gu.se/data/, has put together one master data set containing dozens of political and economic indicators for a large number of countries over long time spans. If you work with cross-country data, this makes your life much easier.

First, have a look at the codebook at http://qog.pol.gu.se/data/datadownloads/qogbasicdata. This codebook is a great example of how to document a dataset. When you collect your own data and share it with others, you should aim for a similarly clear style of documentation.

Let's now use the "QoG Basic Time-Series Data (version January 2015)". I first downloaded the CSV file from http://www.qogdata.pol.gu.se/data/qog_bas_ts_jan15.csv to my working directory, and then read it into R with the familiar `read.csv()` function. (I could use `import()` as well.)

```
qog <- read.csv("/Users/johanneskarreth/Documents/Uni/Teaching/POS 517/Tutorials/Day 7 - Data managemen
str(qog)
```

```
## 'data.frame':    14559 obs. of  220 variables:
##  $ ccode                : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ cname                : Factor w/ 211 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year                 : int  1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 ...
##  $ ccodealp             : Factor w/ 202 levels "AFG","AGO","ALB",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ cname_year           : Factor w/ 14559 levels "Afghanistan 1946",..: 1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ ccodealp_year       : Factor w/ 13938 levels "AFG00","AFG01",..: 16 17 18 19 20 21 22 23 24 25 .
##  $ ccodecow            : int  700 700 700 700 700 700 700 700 700 700 ...
##  $ ccodewb             : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ version             : Factor w/ 1 level "QoGBasTSJan15": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ajr_settmort        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ al_ethnic           : num  0.769 0.769 0.769 0.769 0.769 ...
##  $ al_language         : num  0.614 0.614 0.614 0.614 0.614 ...
##  $ al_religion         : num  0.272 0.272 0.272 0.272 0.272 ...
##  $ bdm_s               : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ bdm_w               : num  0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
##  $ bdm_w_s             : num  0.25 0.25 0.25 0.25 0.25 ...
##  $ bl_asy25f           : num  NA NA NA NA 0.016 ...
##  $ bl_asy25mf          : num  NA NA NA NA 0.122 ...
##  $ bnr_dem             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cam_contest         : num  NA NA NA NA -1.01 ...
##  $ cam_inclusive       : num  NA NA NA NA -0.741 ...
##  $ chga_demo           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ chga_hinst          : int  5 5 5 5 5 5 5 3 3 3 ...
##  $ ciri_assn           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_disap          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_dommov         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_empinx_new     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_empinx_old     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_formov         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_injud          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_kill           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_move_old       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_physint        : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_polpris        : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_relfre_new     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_speech         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_tort           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_wecon          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_wopol          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_worker         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ciri_wosoc          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_cemo            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_checks          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_fraud           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_gprlc1          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_gps1            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_gs              : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_housesys        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_lipc            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_plurality       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_polariz         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_seats           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_system          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dpi_yio             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ dr_eg               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ dr_ig               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ dr_pg               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ dr_sg               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ el_avelf            : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ epi_epi              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ eu_spr_exp_pens_TOTAL: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ fe_etfra             : num  0.751 0.751 0.751 0.751 0.751 ...
##  $ fh_cl                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ fh_fotpsc5           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ fh_ipolity2          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ fh_polity2           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ fh_pr                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ fh_status            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ fi_index             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ fi_index_cl          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gd_ptsa              : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ gd_ptss              : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ gle_cgdpc            : num  NA NA NA NA 131 ...
##  $ gle_rgdpc            : num  NA NA NA NA 893 ...
##  $ gol_est_spec         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ h_j                  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ h_polcon3            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ hf_corrupt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hf_efiscore          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hf_govt              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hf_prights           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ hf_trade             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ ht_colonial          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht_partsz            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ ht_region            : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ ht_regtype           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ iaep_bp              : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ iaep_evp             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ iaep_lvp             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ iaep_npa             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ iaep_osp             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ icrg_qog             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_exp              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_gd               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_gdp              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_gdpgr            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_infl             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_pop              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ imf_rev              : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

This dataset is organized in the structure I described above: countries i and years t, with one row being one country-year observation. Countries are identified by names and a set of country codes.

As in the earlier examples, let's assume we're working on a project where we only need a limited set of indicators. In this case, let's focus on measures for democracy and economic development. After consulting the codebook, we decide we want to work with the following variables:

- gle_cgdpc: GDP per Capita (Current Prices)
- uds_mean: Unified Democracy Scores

For country identifiers, we'll keep the Correlates of War country codes `ccodecow` and the country names `cname` and also the regional identifier `ht_region`; we'll only look at the years 2000-2010 for now. This means

we'll be subsetting the dataset on two dimensions: we only need 5 columns, and we only need the rows for which the condition `year >= 2000 & year <= 2010` apply. You will remember that R operates by rows first, and columns second, with a comma separating the two indices within hard brackets `[ , ]`. That way, we can use one line of code to create our desired subset, which we call `qog.small`:

```
qog.small <- qog[qog$year >= 2000 & qog$year <= 2010,
                 c("cname", "ccodecow", "ht_region", "year", "gle_cgdpc", "uds_mean")]
str(qog.small)
```

```
## 'data.frame':    2321 obs. of  6 variables:
##  $ cname    : Factor w/ 211 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ccodecow : int  700 700 700 700 700 700 700 700 700 700 ...
##  $ ht_region: int  8 8 8 8 8 8 8 8 8 8 ...
##  $ year     : int  2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 ...
##  $ gle_cgdpc: num  694 637 790 870 917 ...
##  $ uds_mean : num  -1.363 -1.632 -0.878 -0.885 -0.702 ...
```

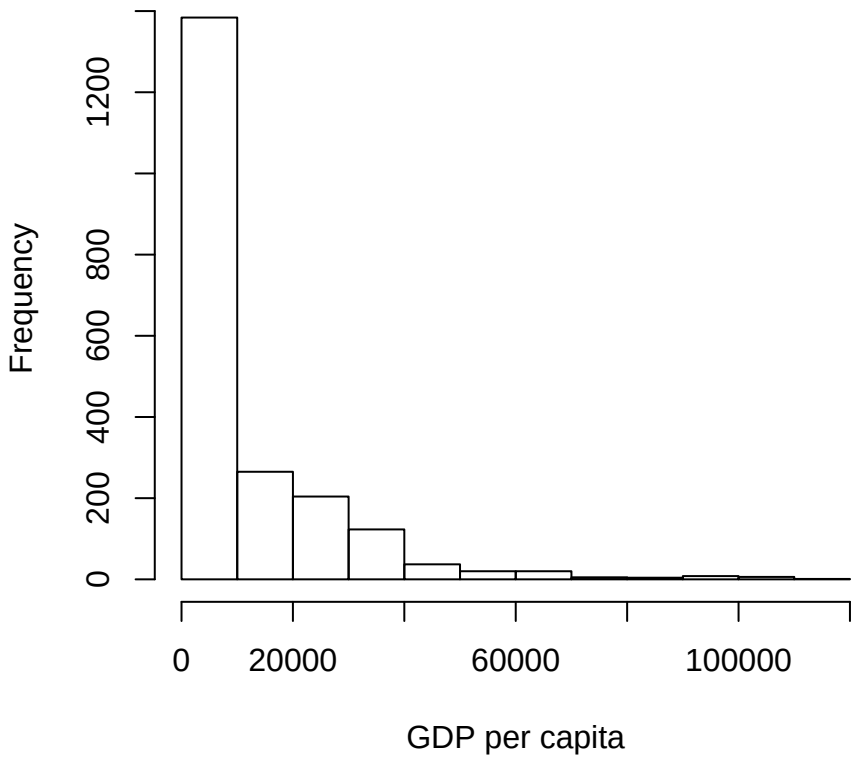
Transforming variables

You've already worked on transforming variables before, so let's briefly reiterate one type of transformation and introduce a few new ones.

Logarithmic transformation

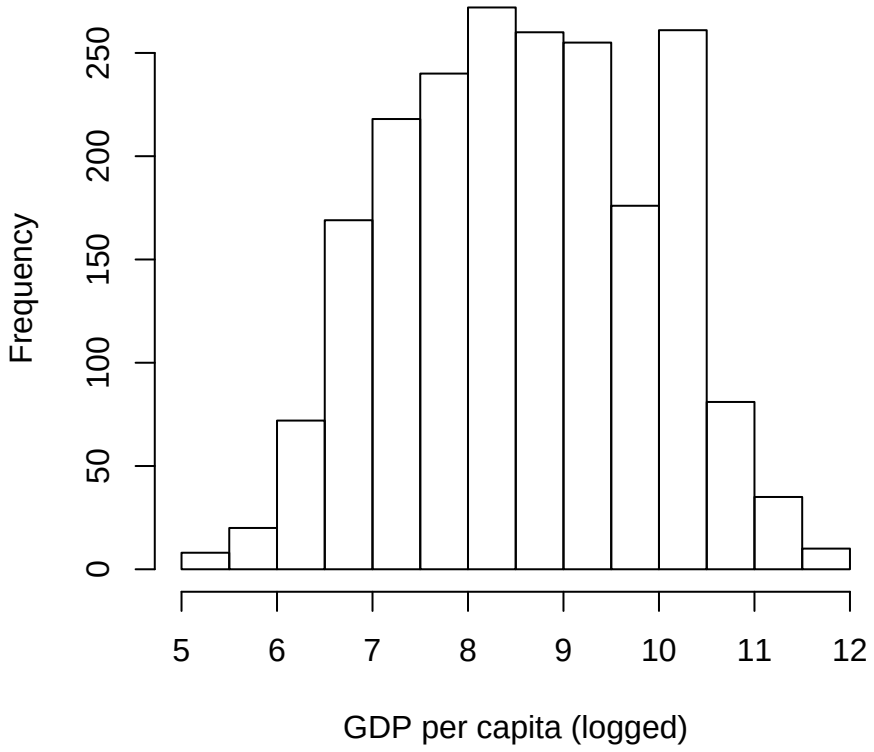
Have a quick look at the GDP per capita variable.

```
hist(qog.small$gle_cgdpc, main = "", xlab = "GDP per capita")
```



This variable has a strong right skew, and we may consider using the logarithmic transformation for it. For this, we create a new variable, `gle_cgdpc_ln`:

```
qog.small$gle_cgdpc_ln <- log(qog.small$gle_cgdpc)
hist(qog.small$gle_cgdpc_ln, main = "", xlab = "GDP per capita (logged)")
```



Time-series operators

We don't spend a separate day on so-called time-series cross-sectional data: data with multiple observations over time ("time-series") for multiple units ("cross-sectional"). But you should be familiar with some basic data manipulation that applies specifically to time-series cross-sectional data. This includes the creation of time-series operators.

To start, let's focus on one single time series though: we'll only pick the United States from our QoG dataset. In the Correlates of War (COW) country code system, the United States is assigned the ID 2 (see http://www.jkarreth.net/files/countrycodes.html for a sortable list of all COW country codes).

```
usa <- qog.small[qog.small$ccodecow == 2, ]
usa
```

```
##                cname ccodecow ht_region year gle_cgdpc uds_mean
## NA              <NA>       NA        NA   NA        NA       NA
## NA.1            <NA>       NA        NA   NA        NA       NA
## NA.2            <NA>       NA        NA   NA        NA       NA
## NA.3            <NA>       NA        NA   NA        NA       NA
## NA.4            <NA>       NA        NA   NA        NA       NA
## NA.5            <NA>       NA        NA   NA        NA       NA
## NA.6            <NA>       NA        NA   NA        NA       NA
## NA.7            <NA>       NA        NA   NA        NA       NA
```

```
## NA.8            <NA>       NA        NA   NA        NA       NA
## NA.9            <NA>       NA        NA   NA        NA       NA
## NA.10           <NA>       NA        NA   NA        NA       NA
## NA.11           <NA>       NA        NA   NA        NA       NA
## NA.12           <NA>       NA        NA   NA        NA       NA
## NA.13           <NA>       NA        NA   NA        NA       NA
## NA.14           <NA>       NA        NA   NA        NA       NA
## NA.15           <NA>       NA        NA   NA        NA       NA
## NA.16           <NA>       NA        NA   NA        NA       NA
## NA.17           <NA>       NA        NA   NA        NA       NA
## NA.18           <NA>       NA        NA   NA        NA       NA
## NA.19           <NA>       NA        NA   NA        NA       NA
## NA.20           <NA>       NA        NA   NA        NA       NA
## NA.21           <NA>       NA        NA   NA        NA       NA
## NA.22           <NA>       NA        NA   NA        NA       NA
## NA.23           <NA>       NA        NA   NA        NA       NA
## NA.24           <NA>       NA        NA   NA        NA       NA
## NA.25           <NA>       NA        NA   NA        NA       NA
## NA.26           <NA>       NA        NA   NA        NA       NA
## NA.27           <NA>       NA        NA   NA        NA       NA
## NA.28           <NA>       NA        NA   NA        NA       NA
## NA.29           <NA>       NA        NA   NA        NA       NA
## NA.30           <NA>       NA        NA   NA        NA       NA
## NA.31           <NA>       NA        NA   NA        NA       NA
## NA.32           <NA>       NA        NA   NA        NA       NA
## NA.33           <NA>       NA        NA   NA        NA       NA
## NA.34           <NA>       NA        NA   NA        NA       NA
## NA.35           <NA>       NA        NA   NA        NA       NA
## NA.36           <NA>       NA        NA   NA        NA       NA
## NA.37           <NA>       NA        NA   NA        NA       NA
## NA.38           <NA>       NA        NA   NA        NA       NA
## NA.39           <NA>       NA        NA   NA        NA       NA
## NA.40           <NA>       NA        NA   NA        NA       NA
## NA.41           <NA>       NA        NA   NA        NA       NA
## NA.42           <NA>       NA        NA   NA        NA       NA
## NA.43           <NA>       NA        NA   NA        NA       NA
## 13372  United States        2         5 2000  35412.48 1.561573
## 13373  United States        2         5 2001  36122.87 1.559541
## 13374  United States        2         5 2002  36804.33 1.544341
## 13375  United States        2         5 2003  38098.34 1.567313
## 13376  United States        2         5 2004  40142.37 1.558781
## 13377  United States        2         5 2005  42329.65 1.549633
## 13378  United States        2         5 2006  44447.90 1.551548
## 13379  United States        2         5 2007  46208.62 1.553307
## 13380  United States        2         5 2008  46519.89 1.556611
## 13381  United States        2         5 2009  44845.30 1.560328
## 13382  United States        2         5 2010  46203.23 1.559407
## NA.44           <NA>       NA        NA   NA        NA       NA
## NA.45           <NA>       NA        NA   NA        NA       NA
## NA.46           <NA>       NA        NA   NA        NA       NA
## NA.47           <NA>       NA        NA   NA        NA       NA
## NA.48           <NA>       NA        NA   NA        NA       NA
## NA.49           <NA>       NA        NA   NA        NA       NA
## NA.50           <NA>       NA        NA   NA        NA       NA
```

```
## NA.51          <NA>       NA        NA   NA        NA       NA
## NA.52          <NA>       NA        NA   NA        NA       NA
## NA.53          <NA>       NA        NA   NA        NA       NA
## NA.54          <NA>       NA        NA   NA        NA       NA
## NA.55          <NA>       NA        NA   NA        NA       NA
## NA.56          <NA>       NA        NA   NA        NA       NA
## NA.57          <NA>       NA        NA   NA        NA       NA
## NA.58          <NA>       NA        NA   NA        NA       NA
## NA.59          <NA>       NA        NA   NA        NA       NA
## NA.60          <NA>       NA        NA   NA        NA       NA
## NA.61          <NA>       NA        NA   NA        NA       NA
## NA.62          <NA>       NA        NA   NA        NA       NA
## NA.63          <NA>       NA        NA   NA        NA       NA
## NA.64          <NA>       NA        NA   NA        NA       NA
## NA.65          <NA>       NA        NA   NA        NA       NA
## NA.66          <NA>       NA        NA   NA        NA       NA
## NA.67          <NA>       NA        NA   NA        NA       NA
## NA.68          <NA>       NA        NA   NA        NA       NA
## NA.69          <NA>       NA        NA   NA        NA       NA
## NA.70          <NA>       NA        NA   NA        NA       NA
## NA.71          <NA>       NA        NA   NA        NA       NA
## NA.72          <NA>       NA        NA   NA        NA       NA
## NA.73          <NA>       NA        NA   NA        NA       NA
## NA.74          <NA>       NA        NA   NA        NA       NA
## NA.75          <NA>       NA        NA   NA        NA       NA
## NA.76          <NA>       NA        NA   NA        NA       NA
## NA.77          <NA>       NA        NA   NA        NA       NA
## NA.78          <NA>       NA        NA   NA        NA       NA
## NA.79          <NA>       NA        NA   NA        NA       NA
## NA.80          <NA>       NA        NA   NA        NA       NA
## NA.81          <NA>       NA        NA   NA        NA       NA
## NA.82          <NA>       NA        NA   NA        NA       NA
## NA.83          <NA>       NA        NA   NA        NA       NA
## NA.84          <NA>       NA        NA   NA        NA       NA
## NA.85          <NA>       NA        NA   NA        NA       NA
## NA.86          <NA>       NA        NA   NA        NA       NA
## NA.87          <NA>       NA        NA   NA        NA       NA
## NA.88          <NA>       NA        NA   NA        NA       NA
## NA.89          <NA>       NA        NA   NA        NA       NA
## NA.90          <NA>       NA        NA   NA        NA       NA
## NA.91          <NA>       NA        NA   NA        NA       NA
## NA.92          <NA>       NA        NA   NA        NA       NA
## NA.93          <NA>       NA        NA   NA        NA       NA
## NA.94          <NA>       NA        NA   NA        NA       NA
## NA.95          <NA>       NA        NA   NA        NA       NA
## NA.96          <NA>       NA        NA   NA        NA       NA
## NA.97          <NA>       NA        NA   NA        NA       NA
## NA.98          <NA>       NA        NA   NA        NA       NA
## NA.99          <NA>       NA        NA   NA        NA       NA
## NA.100         <NA>       NA        NA   NA        NA       NA
## NA.101         <NA>       NA        NA   NA        NA       NA
## NA.102         <NA>       NA        NA   NA        NA       NA
## NA.103         <NA>       NA        NA   NA        NA       NA
## NA.104         <NA>       NA        NA   NA        NA       NA
```

```
## NA.105          <NA>       NA        NA   NA        NA       NA
## NA.106          <NA>       NA        NA   NA        NA       NA
## NA.107          <NA>       NA        NA   NA        NA       NA
## NA.108          <NA>       NA        NA   NA        NA       NA
## NA.109          <NA>       NA        NA   NA        NA       NA
## NA.110          <NA>       NA        NA   NA        NA       NA
## NA.111          <NA>       NA        NA   NA        NA       NA
## NA.112          <NA>       NA        NA   NA        NA       NA
## NA.113          <NA>       NA        NA   NA        NA       NA
## NA.114          <NA>       NA        NA   NA        NA       NA
## NA.115          <NA>       NA        NA   NA        NA       NA
## NA.116          <NA>       NA        NA   NA        NA       NA
## NA.117          <NA>       NA        NA   NA        NA       NA
## NA.118          <NA>       NA        NA   NA        NA       NA
## NA.119          <NA>       NA        NA   NA        NA       NA
## NA.120          <NA>       NA        NA   NA        NA       NA
##        gle_cgdpc_ln
## NA               NA
## NA.1             NA
## NA.2             NA
## NA.3             NA
## NA.4             NA
## NA.5             NA
## NA.6             NA
## NA.7             NA
## NA.8             NA
## NA.9             NA
## NA.10            NA
## NA.11            NA
## NA.12            NA
## NA.13            NA
## NA.14            NA
## NA.15            NA
## NA.16            NA
## NA.17            NA
## NA.18            NA
## NA.19            NA
## NA.20            NA
## NA.21            NA
## NA.22            NA
## NA.23            NA
## NA.24            NA
## NA.25            NA
## NA.26            NA
## NA.27            NA
## NA.28            NA
## NA.29            NA
## NA.30            NA
## NA.31            NA
## NA.32            NA
## NA.33            NA
## NA.34            NA
## NA.35            NA
## NA.36            NA
```

```
## NA.37           NA
## NA.38           NA
## NA.39           NA
## NA.40           NA
## NA.41           NA
## NA.42           NA
## NA.43           NA
## 13372     10.47482
## 13373     10.49468
## 13374     10.51337
## 13375     10.54793
## 13376     10.60019
## 13377     10.65324
## 13378     10.70207
## 13379     10.74092
## 13380     10.74764
## 13381     10.71097
## 13382     10.74080
## NA.44           NA
## NA.45           NA
## NA.46           NA
## NA.47           NA
## NA.48           NA
## NA.49           NA
## NA.50           NA
## NA.51           NA
## NA.52           NA
## NA.53           NA
## NA.54           NA
## NA.55           NA
## NA.56           NA
## NA.57           NA
## NA.58           NA
## NA.59           NA
## NA.60           NA
## NA.61           NA
## NA.62           NA
## NA.63           NA
## NA.64           NA
## NA.65           NA
## NA.66           NA
## NA.67           NA
## NA.68           NA
## NA.69           NA
## NA.70           NA
## NA.71           NA
## NA.72           NA
## NA.73           NA
## NA.74           NA
## NA.75           NA
## NA.76           NA
## NA.77           NA
## NA.78           NA
## NA.79           NA
```

```
## NA.80            NA
## NA.81            NA
## NA.82            NA
## NA.83            NA
## NA.84            NA
## NA.85            NA
## NA.86            NA
## NA.87            NA
## NA.88            NA
## NA.89            NA
## NA.90            NA
## NA.91            NA
## NA.92            NA
## NA.93            NA
## NA.94            NA
## NA.95            NA
## NA.96            NA
## NA.97            NA
## NA.98            NA
## NA.99            NA
## NA.100           NA
## NA.101           NA
## NA.102           NA
## NA.103           NA
## NA.104           NA
## NA.105           NA
## NA.106           NA
## NA.107           NA
## NA.108           NA
## NA.109           NA
## NA.110           NA
## NA.111           NA
## NA.112           NA
## NA.113           NA
## NA.114           NA
## NA.115           NA
## NA.116           NA
## NA.117           NA
## NA.118           NA
## NA.119           NA
## NA.120           NA
```

In this case, we run into a problem: the variable `ccodecow` contains `NA` values and therefore these observations are included in our new subset. We can get around this by specifying an additional condition for our subset:

```
usa <- qog.small[qog.small$ccodecow == 2 & is.na(qog.small$ccodecow) == FALSE, ]
usa
```

```
##               cname ccodecow ht_region year gle_cgdpc uds_mean
## 13372 United States        2         5 2000  35412.48 1.561573
## 13373 United States        2         5 2001  36122.87 1.559541
## 13374 United States        2         5 2002  36804.33 1.544341
## 13375 United States        2         5 2003  38098.34 1.567313
## 13376 United States        2         5 2004  40142.37 1.558781
```

```
## 13377 United States        2         5 2005  42329.65 1.549633
## 13378 United States        2         5 2006  44447.90 1.551548
## 13379 United States        2         5 2007  46208.62 1.553307
## 13380 United States        2         5 2008  46519.89 1.556611
## 13381 United States        2         5 2009  44845.30 1.560328
## 13382 United States        2         5 2010  46203.23 1.559407
##       gle_cgdpc_ln
## 13372     10.47482
## 13373     10.49468
## 13374     10.51337
## 13375     10.54793
## 13376     10.60019
## 13377     10.65324
## 13378     10.70207
## 13379     10.74092
## 13380     10.74764
## 13381     10.71097
## 13382     10.74080
```

Now, let's say we want to create a variable that expresses the absolute change in GDP per capita from year to year. We can do this in two steps. First, we create a "lag" of GDP per capita, using the `lag()` function. This function takes two arguments. The first is the vector from which we are creating the lag, and the second, `k`, is the number of time units (here: years) by which we want to lag the original variable. Before we use the `lag` function, I sort the data by years to make sure that the lag is taken from the previous year. I do this using the `arrange` function, which is part of the "dplyr" package that you loaded above.

```
usa <- arrange(usa, year)
usa$gle_cgdpc_lag <- lag(usa$gle_cgdpc, k = 1)
usa
```

```
##            cname ccodecow ht_region year gle_cgdpc uds_mean gle_cgdpc_ln
## 1  United States        2         5 2000  35412.48 1.561573     10.47482
## 2  United States        2         5 2001  36122.87 1.559541     10.49468
## 3  United States        2         5 2002  36804.33 1.544341     10.51337
## 4  United States        2         5 2003  38098.34 1.567313     10.54793
## 5  United States        2         5 2004  40142.37 1.558781     10.60019
## 6  United States        2         5 2005  42329.65 1.549633     10.65324
## 7  United States        2         5 2006  44447.90 1.551548     10.70207
## 8  United States        2         5 2007  46208.62 1.553307     10.74092
## 9  United States        2         5 2008  46519.89 1.556611     10.74764
## 10 United States        2         5 2009  44845.30 1.560328     10.71097
## 11 United States        2         5 2010  46203.23 1.559407     10.74080
##    gle_cgdpc_lag
## 1             NA
## 2       35412.48
## 3       36122.87
## 4       36804.33
## 5       38098.34
## 6       40142.37
## 7       42329.65
## 8       44447.90
## 9       46208.62
## 10      46519.89
## 11      44845.30
```

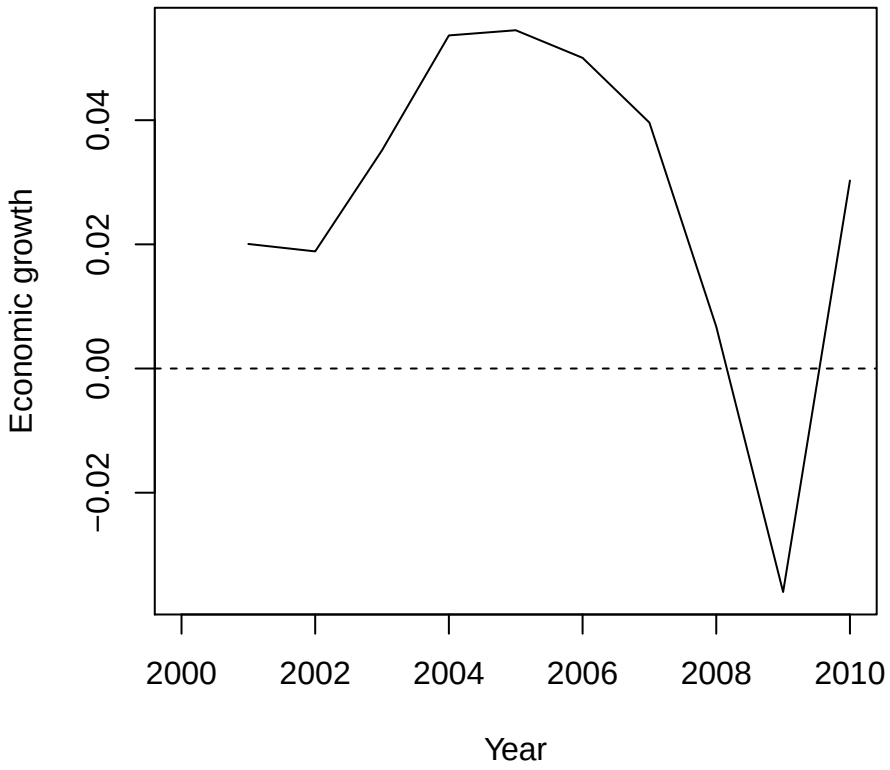
I can now create the "change in GDP per capita" variable by simply subtracting the previous from the current value:

```
usa$gle_cgdpc_ch <- usa$gle_cgdpc - usa$gle_cgdpc_lag
usa
```

```
##            cname ccodecow ht_region year gle_cgdpc uds_mean gle_cgdpc_ln
## 1  United States        2         5 2000  35412.48 1.561573     10.47482
## 2  United States        2         5 2001  36122.87 1.559541     10.49468
## 3  United States        2         5 2002  36804.33 1.544341     10.51337
## 4  United States        2         5 2003  38098.34 1.567313     10.54793
## 5  United States        2         5 2004  40142.37 1.558781     10.60019
## 6  United States        2         5 2005  42329.65 1.549633     10.65324
## 7  United States        2         5 2006  44447.90 1.551548     10.70207
## 8  United States        2         5 2007  46208.62 1.553307     10.74092
## 9  United States        2         5 2008  46519.89 1.556611     10.74764
## 10 United States        2         5 2009  44845.30 1.560328     10.71097
## 11 United States        2         5 2010  46203.23 1.559407     10.74080
##    gle_cgdpc_lag gle_cgdpc_ch
## 1             NA           NA
## 2       35412.48      710.391
## 3       36122.87      681.457
## 4       36804.33     1294.012
## 5       38098.34     2044.031
## 6       40142.37     2187.277
## 7       42329.65     2118.250
## 8       44447.90     1760.723
## 9       46208.62      311.270
## 10      46519.89    -1674.590
## 11      44845.30     1357.929
```

And now we can create a quick plot of percentage growth in GDP per capita, using another new variable, `gle_cgdpc_growth`, that we create as well:

```
usa$gle_cgdpc_growth <- usa$gle_cgdpc_ch / usa$gle_cgdpc_lag
plot(x = usa$year, y = usa$gle_cgdpc_growth, type = "l", main = "",
     xlab = "Year", ylab = "Economic growth")
abline(h = 0, lty = "dashed")
```



We can also create lags in time-series cross-sectional data as in the qog.small dataset, where we want lags for individual units. For this, we return to the `mutate` function from the "dplyr" package that you already used above. We also need to take into account the grouping structure and use the `group_by` function for that purpose.

First, I eliminate all observations that do not have COW country codes, assuming that these are duplicate observations. In your research, you should carefully check such cases by hand.

```
qog.small <- qog.small[is.na(qog.small$ccodecow) == FALSE, ]
```

Next, I sort the dataset by country codes and then years. I do this in order to have the data ready for creating lagged variables further below.

5. Which country and year will be first (and last) in this sorted dataset?

```
qog.small <- arrange(qog.small, ccodecow, year)
```

Now, I use again the `mutate()` function to add a variable to this dataset without changing the structure of the dataset.

```
qog.small <- mutate(group_by(qog.small, ccodecow),
                    gle_cgdpc_lag = lag(gle_cgdpc, k = 1))
tail(qog.small)
```

```
## Source: local data frame [6 x 8]
## Groups: ccodecow
##
##   cname ccodecow ht_region year gle_cgdpc  uds_mean gle_cgdpc_ln
```

```
## 1 Samoa      990         9 2005   2923.50 0.3877878     7.980537
## 2 Samoa      990         9 2006   3029.97 0.2429549     8.016308
## 3 Samoa      990         9 2007   2828.93 0.2434975     7.947654
## 4 Samoa      990         9 2008   2863.70 0.2411200     7.959870
## 5 Samoa      990         9 2009   3142.23 0.6412778     8.052688
## 6 Samoa      990         9 2010   3154.86 0.6409692     8.056699
##   gle_cgdpc_lag
## 1       2786.59
## 2       2923.50
## 3       3029.97
## 4       2828.93
## 5       2863.70
## 6       3142.23
```

You can use all sorts of other variable creation commands like you did above for the `usa` time series.

Merging data

Often in your work, you will find yourself having to combine datasets from different sources. We'll work through a quick example returning to the Afrobarometer survey data from the beginning of this tutorial. Let's say we are interested in the relationship between a common measure for democracy, the Unified Democracy Score, and survey respondents' perception of how democratic their country is. We already have both pieces of information. Recall that we created a country-level version of the Afrobarometer data that contains the average perception of democracy:

```
ab.country
```

```
## Source: local data frame [20 x 2]
##
##         COUNTRY perceivedDem
## 1         Benin     2.190941
## 2      Botswana     2.488889
## 3  Burkina Faso     1.848024
## 4    Cape Verde     2.107989
## 5         Ghana     2.435315
## 6         Kenya     1.568047
## 7       Lesotho     1.509953
## 8       Liberia     1.912811
## 9    Madagascar     1.797376
## 10       Malawi     1.816530
## 11         Mali     1.947186
## 12   Mozambique     1.972603
## 13      Namibia     2.148464
## 14      Nigeria     1.427995
## 15      Senegal     1.574279
## 16 South Africa     1.816717
## 17     Tanzania     2.231054
## 18       Uganda     1.738606
## 19       Zambia     1.690975
## 20     Zimbabwe     1.057171
```

And, we have the `qog.small` dataset containing the Unified Democracy Scores variable.

To bring the two together, we need two things: first, matching identifiers that help our software package assign the appropriate observations to each other. The QoG data contains COW country codes, but our Afrobarometer data only has country names. We can use the `countrycode()` function from the package with the same name to add a variable `ccodecow` to our Afrobarometer data. Install the "countrycode" package once, and then load it every R session where you use it. Have a look at the help file for the `countrycode()` function before using it here.

```
library(countrycode)
ab.country$ccodecow <- countrycode(ab.country$COUNTRY, origin = "country.name",
                                   destination = "cown")
ab.country
```

```
## Source: local data frame [20 x 3]
##
##         COUNTRY perceivedDem ccodecow
## 1         Benin     2.190941      434
## 2      Botswana     2.488889      571
## 3  Burkina Faso     1.848024      439
## 4    Cape Verde     2.107989      402
## 5         Ghana     2.435315      452
## 6         Kenya     1.568047      501
## 7       Lesotho     1.509953      570
## 8       Liberia     1.912811      450
## 9    Madagascar     1.797376      580
## 10       Malawi     1.816530      553
## 11         Mali     1.947186      432
## 12   Mozambique     1.972603      541
## 13      Namibia     2.148464      565
## 14      Nigeria     1.427995      475
## 15      Senegal     1.574279      433
## 16 South Africa     1.816717      560
## 17     Tanzania     2.231054      510
## 18       Uganda     1.738606      500
## 19       Zambia     1.690975      551
## 20     Zimbabwe     1.057171      552
```

We can now prepare the QoG data for the merge. Recall that the Afrobarometer survey was conducted in 2008, so limiting ourselves to the QoG data from that year probably makes sense.

```
qog.2008 <- qog.small[qog.small$year == 2008, ]
```

Now we use the `merge()` function to create a new object that contains both the `ab.country` and `qog.2008` datasets. Because we are only interested in the observations for which we have survey data, we only keep those observations, setting `all.y` to `FALSE`.

6. How many columns will the new merged dataset have? How many observations?

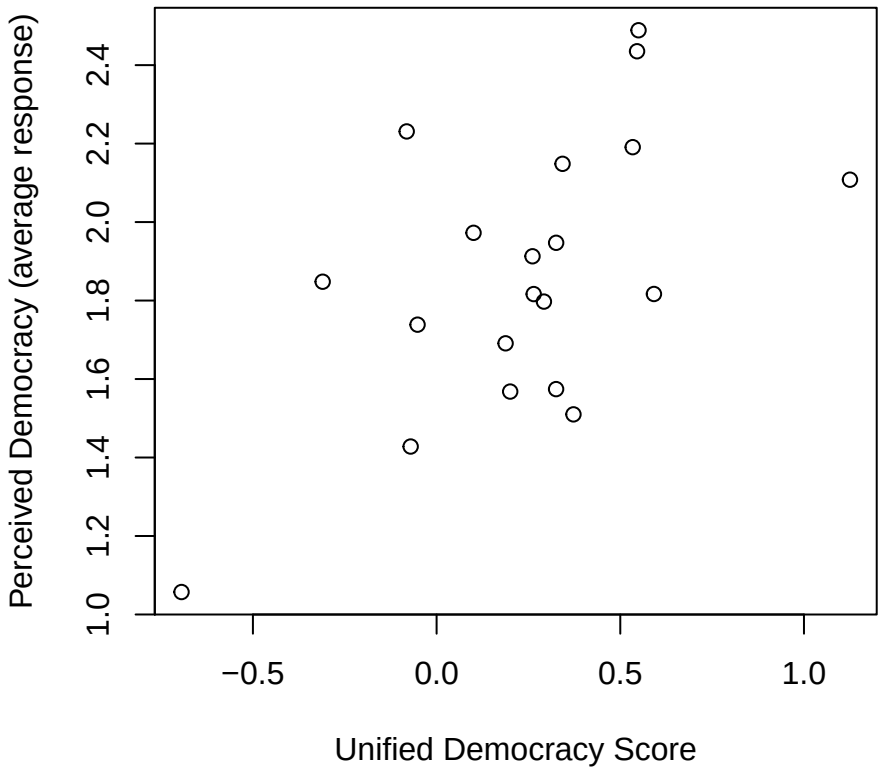
```
ab.qog <- merge(x = ab.country, y = qog.2008, by = "ccodecow", all.x = TRUE, all.y = FALSE)
ab.qog
```

```
##    ccodecow      COUNTRY perceivedDem        cname ht_region year
## 1       402   Cape Verde     2.107989   Cape Verde         4 2008
```

```
## 2       432         Mali     1.947186         Mali         4 2008
## 3       433      Senegal     1.574279      Senegal         4 2008
## 4       434        Benin     2.190941        Benin         4 2008
## 5       439 Burkina Faso     1.848024 Burkina Faso         4 2008
## 6       450      Liberia     1.912811      Liberia         4 2008
## 7       452        Ghana     2.435315        Ghana         4 2008
## 8       475      Nigeria     1.427995      Nigeria         4 2008
## 9       500       Uganda     1.738606       Uganda         4 2008
## 10      501        Kenya     1.568047        Kenya         4 2008
## 11      510     Tanzania     2.231054     Tanzania         4 2008
## 12      541   Mozambique     1.972603   Mozambique         4 2008
## 13      551       Zambia     1.690975       Zambia         4 2008
## 14      552     Zimbabwe     1.057171     Zimbabwe         4 2008
## 15      553       Malawi     1.816530       Malawi         4 2008
## 16      560 South Africa     1.816717 South Africa         4 2008
## 17      565      Namibia     2.148464      Namibia         4 2008
## 18      570      Lesotho     1.509953      Lesotho         4 2008
## 19      571     Botswana     2.488889     Botswana         4 2008
## 20      580   Madagascar     1.797376   Madagascar         4 2008
##    gle_cgdpc    uds_mean gle_cgdpc_ln gle_cgdpc_lag
## 1    3497.61  1.12520100     8.159835       3129.26
## 2     924.45  0.32543170     6.829199        887.64
## 3    1482.40  0.32528931     7.301418       1447.38
## 4    1387.37  0.53380263     7.235165       1373.85
## 5     992.79 -0.31005162     6.900519        950.35
## 6     416.60  0.26105148     6.032127        420.57
## 7    2055.85  0.54564589     7.628445       1887.85
## 8    1867.95 -0.07064825     7.532597       1965.47
## 9    1190.36 -0.05178989     7.082011       1139.39
## 10   1367.24  0.20031281     7.220549       1320.71
## 11   1209.36 -0.08134177     7.097847       1121.24
## 12    731.01  0.10057697     6.594427        691.23
## 13   1657.15  0.18778080     7.412855       1628.14
## 14   3581.84 -0.69430882     8.183632       3831.50
## 15    876.58  0.26429641     6.776028        774.30
## 16   8158.52  0.59160823     9.006818       7865.17
## 17   4559.66  0.34284481     8.425003       4645.27
## 18   1398.08  0.37254864     7.242855       1362.55
## 19  12935.19  0.54960614     9.467707      10689.01
## 20    830.19  0.29212907     6.721655        799.07
```

And now we can investigate the relationship between respondents' average perception of democracy and the Unified Democracy Scores, which are based on information about political institutions and political processes.

```
plot(y = ab.qog$perceivedDem, x = ab.qog$uds_mean,
     ylab = "Perceived Democracy (average response)",
     xlab = "Unified Democracy Score")
```



Of course we can also estimate a bivariate regression model of this relationship:

```
mod <- lm(perceivedDem ~ uds_mean, data = ab.qog)
summary(mod)
```

```
##
## Call:
## lm(formula = perceivedDem ~ uds_mean, data = ab.qog)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42451 -0.24345 -0.01489  0.19484  0.53863
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.73580    0.07814  22.213 1.56e-14 ***
## uds_mean     0.53326    0.17764   3.002  0.00765 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2926 on 18 degrees of freedom
## Multiple R-squared:  0.3336, Adjusted R-squared:  0.2966
## F-statistic: 9.012 on 1 and 18 DF,  p-value: 0.007653
```

Creating new variables through recoding

This tutorial concludes with two common ways of creating new variables: recoding a variable into a binary variable, and cutting a dichotomous variable into percentiles.

Dichotomizing variables

Sometimes you may be interested in condensing information into a binary yes/no indicator. For instance, instead of including an education measure with 10 categories (which we have in our survey data, as `education` and `educationLevel`), you may want to know whether someone finished high school or not. To do this, we return to using the `ifelse()` function. Let's create a variable `hs` that is set to 1 for all survey respondents that finished high school and to 0 for all who did not. For this, we need to recall the levels of the `education` variable.

```
table(ab.small$educationLevel, ab.small$education)
```

```
##
##                                              0    1    2    3    4
##   No formal schooling                     4365    0    0    0    0
##   Informal schooling only                    0 1260    0    0    0
##   Some primary schooling                     0    0 5111    0    0
##   Primary school completed                   0    0    0 3897    0
##   Some secondary school/high school          0    0    0    0 5950
##   Secondary school completed/high school     0    0    0    0    0
##   Post-secondary qualifications, not university    0    0    0    0    0
##   Some university                            0    0    0    0    0
##   University completed                       0    0    0    0    0
##   Post-graduate                              0    0    0    0    0
##
##                                              5    6    7    8    9
##   No formal schooling                        0    0    0    0    0
##   Informal schooling only                    0    0    0    0    0
##   Some primary schooling                     0    0    0    0    0
##   Primary school completed                   0    0    0    0    0
##   Some secondary school/high school          0    0    0    0    0
##   Secondary school completed/high school  4165    0    0    0    0
##   Post-secondary qualifications, not university    0 1674    0    0    0
##   Some university                            0    0  649    0    0
##   University completed                       0    0    0  506    0
##   Post-graduate                              0    0    0    0   92
```

So, values of 5 and above were assigned to respondents who completed high school. We use this cutoff for our new binary variable `hs`.

```
ab.small$hs <- ifelse(ab.small$education >= 5, 1, 0)
table(ab.small$educationLevel, ab.small$hs)
```

```
##
##                                              0    1
##   No formal schooling                     4365    0
##   Informal schooling only                 1260    0
##   Some primary schooling                  5111    0
##   Primary school completed                3897    0
##   Some secondary school/high school       5950    0
##   Secondary school completed/high school     0 4165
##   Post-secondary qualifications, not university    0 1674
##   Some university                            0  649
##   University completed                       0  506
##   Post-graduate                              0   92
```

7. What would you do if you wanted to create a "trichotomous" variable with values of 0, 1, and 2 for respondents whose education ended before high school (0), after high school (1), and who graduated from a post-secondary institution (2)?

Creating percentiles

For many variables, percentiles are often more useful to interpret than absolute values. Let's take a slightly challenging example and say we want to create an indicator for groups of 10% of GDP per capita for countries. That is, the bottom 10% of countries receive a 0, the next 10% a 1, and so forth. To do this, we combine a couple of R functions:

- `cut()` divides the range of a continuous variable into intervals and codes the values according to which interval they fall.
- `quantile()` calculates percentiles (and requires the `na.rm` argument to be set to `TRUE`!)
- `as.numeric()` converts a factor into a numeric variable

We work with the qog.small data again.

```
deciles <- quantile(qog.small$gle_cgdpc, probs = seq(0, 1, by = 0.1), na.rm = TRUE)
qog.small$gle_cgdpc_dec <- cut(qog.small$gle_cgdpc,
                               breaks = deciles, include.lowest = TRUE)
table(qog.small$gle_cgdpc_dec)
```

```
##
##           [173,937]      (937,1.54e+03] (1.54e+03,2.51e+03]
##                 208                 207                 207
## (2.51e+03,3.58e+03] (3.58e+03,5.28e+03] (5.28e+03,7.94e+03]
##                 207                 207                 207
## (7.94e+03,1.16e+04] (1.16e+04,2.07e+04]  (2.07e+04,3.1e+04]
##                 207                 207                 207
##  (3.1e+04,1.13e+05]
##                 207
```

If we want to make this new variable a numeric variable, we can do the following:

```
qog.small$gle_cgdpc_dec2 <- as.numeric(qog.small$gle_cgdpc_dec)
table(qog.small$gle_cgdpc_dec, qog.small$gle_cgdpc_dec2)
```

```
##
##                         1   2   3   4   5   6   7   8   9  10
##   [173,937]           208   0   0   0   0   0   0   0   0   0
##   (937,1.54e+03]        0 207   0   0   0   0   0   0   0   0
##   (1.54e+03,2.51e+03]   0   0 207   0   0   0   0   0   0   0
##   (2.51e+03,3.58e+03]   0   0   0 207   0   0   0   0   0   0
##   (3.58e+03,5.28e+03]   0   0   0   0 207   0   0   0   0   0
##   (5.28e+03,7.94e+03]   0   0   0   0   0 207   0   0   0   0
##   (7.94e+03,1.16e+04]   0   0   0   0   0   0 207   0   0   0
##   (1.16e+04,2.07e+04]   0   0   0   0   0   0   0 207   0   0
##   (2.07e+04,3.1e+04]    0   0   0   0   0   0   0   0 207   0
##   (3.1e+04,1.13e+05]    0   0   0   0   0   0   0   0   0 207
```

Reshaping data

You read about wide and long formats in the *Tidy Data* article. Sometimes you will encounter data in wide format, where different measurements of the same variable (e.g., measurements of income over several years) are entered in a spreadsheet as columns, not rows. This is the case, for instance, for economic indicators released by statistics offices.

R offers some easy options to convert these data into long format. We'll briefly work along two examples.

Clean data

The first example that is provided by UCLA's Stat Consulting Group at http://www.ats.ucla.edu/stat/stata/modules/reshapel.htm. First, we read in a dataset in wide format.

```
dat.wide <- import(file = "http://www.ats.ucla.edu/stat/stata/modules/faminc.dta")
dat.wide
```

```
##   famid faminc96 faminc97 faminc98
## 1     3    75000    76000    77000
## 2     1    40000    40500    41000
## 3     2    45000    45400    45800
```

Now, we use the `melt()` function from the "reshape2" package that you encountered in the *Tidy Data* article. Install the package (only once), and then load it (in every script where you use it)

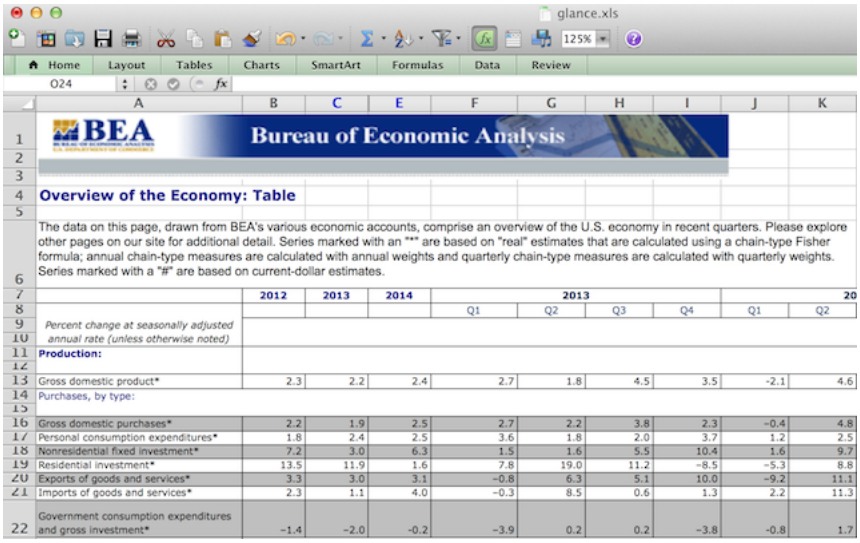
```
library(reshape2)
dat.long <- melt(dat.wide,
                 id.vars = c("famid"),
                 variable.name = "year",
                 value.name = "faminc")
dat.long$year <- as.numeric(gsub("faminc","",dat.long$year))
arrange(dat.long, famid, year)
```

```
##   famid year faminc
## 1     1   96  40000
## 2     1   97  40500
## 3     1   98  41000
## 4     2   96  45000
## 5     2   97  45400
## 6     2   98  45800
## 7     3   96  75000
## 8     3   97  76000
## 9     3   98  77000
```

Messier data

We can also use these functions for "messier" datasets. For instance, let's say we want to download some economic indicators from the U.S. Bureau of Economic Analysis. The website "U.S. Economy at a Glance" offers these data as an Excel spreadsheet (link on the top right: http://www.bea.gov/newsreleases/xls/glance.xls).

After downloading this spreadsheet, we get the following data:



glance.xls

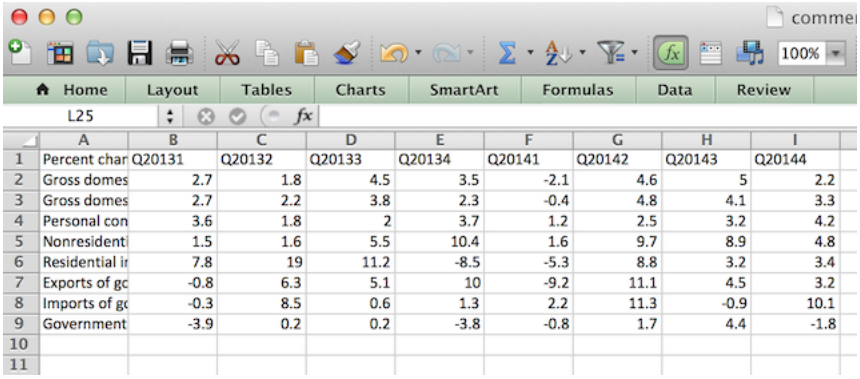
Bureau of Economic Analysis

Overview of the Economy: Table

The data on this page, drawn from BEA's various economic accounts, comprise an overview of the U.S. economy in recent quarters. Please explore other pages on our site for additional detail. Series marked with an "*" are based on "real" estimates that are calculated using a chain-type Fisher formula; annual chain-type measures are calculated with annual weights and quarterly chain-type measures are calculated with quarterly weights. Series marked with a "#" are based on current-dollar estimates.

	2012	2013	2014	2013 Q1	2013 Q2	2013 Q3	2013 Q4	20 Q1	20 Q2
Percent change at seasonally adjusted annual rate (unless otherwise noted)									
Production:									
Gross domestic product*	2.3	2.2	2.4	2.7	1.8	4.5	3.5	-2.1	4.6
Purchases, by type:									
Gross domestic purchases*	2.2	1.9	2.5	2.7	2.2	3.8	2.3	-0.4	4.8
Personal consumption expenditures*	1.8	2.4	2.5	3.6	1.8	2.0	3.7	1.2	2.5
Nonresidential fixed investment*	7.2	3.0	6.3	1.5	1.6	5.5	10.4	1.6	9.7
Residential investment*	13.5	11.9	1.6	7.8	19.0	11.2	-8.5	-5.3	8.8
Exports of goods and services*	3.3	3.0	3.1	-0.8	6.3	5.1	10.0	-9.2	11.1
Imports of goods and services*	2.3	1.1	4.0	-0.3	8.5	0.6	1.3	2.2	11.3
Government consumption expenditures and gross investment*	-1.4	-2.0	-0.2	-3.9	0.2	0.2	-3.8	-0.8	1.7

I clean these data up by hand in Excel so that they can be read into R:



	A	B	C	D	E	F	G	H	I
1	Percent char	Q20131	Q20132	Q20133	Q20134	Q20141	Q20142	Q20143	Q20144
2	Gross domes	2.7	1.8	4.5	3.5	-2.1	4.6	5	2.2
3	Gross domes	2.7	2.2	3.8	2.3	-0.4	4.8	4.1	3.3
4	Personal con	3.6	1.8	2	3.7	1.2	2.5	3.2	4.2
5	Nonresident	1.5	1.6	5.5	10.4	1.6	9.7	8.9	4.8
6	Residential ir	7.8	19	11.2	-8.5	-5.3	8.8	3.2	3.4
7	Exports of gc	-0.8	6.3	5.1	10	-9.2	11.1	4.5	3.2
8	Imports of gc	-0.3	8.5	0.6	1.3	2.2	11.3	-0.9	10.1
9	Government	-3.9	0.2	0.2	-3.8	-0.8	1.7	4.4	-1.8
10									
11									

```
com.wide <- read.csv("/Users/johanneskarreth/Documents/Uni/Teaching/POS 517/Tutorials/Day 7 - Data manag
com.wide
```

```
##   Percent.change.at.seasonally.adjusted.annual.rate..unless.otherwise.noted.
## 1                                                    Gross domestic product*
## 2                                                  Gross domestic purchases*
## 3                                        Personal consumption expenditures*
## 4                                          Nonresidential fixed investment*
## 5                                                  Residential investment*
## 6                                            Exports of goods and services*
## 7                                            Imports of goods and services*
## 8                         Government consumption expenditures and gross investment*
##   Q20131 Q20132 Q20133 Q20134 Q20141 Q20142 Q20143 Q20144
## 1    2.7    1.8    4.5    3.5   -2.1    4.6    5.0    2.2
## 2    2.7    2.2    3.8    2.3   -0.4    4.8    4.1    3.3
## 3    3.6    1.8    2.0    3.7    1.2    2.5    3.2    4.2
## 4    1.5    1.6    5.5   10.4    1.6    9.7    8.9    4.8
## 5    7.8   19.0   11.2   -8.5   -5.3    8.8    3.2    3.4
## 6   -0.8    6.3    5.1   10.0   -9.2   11.1    4.5    3.2
## 7   -0.3    8.5    0.6    1.3    2.2   11.3   -0.9   10.1
## 8   -3.9    0.2    0.2   -3.8   -0.8    1.7    4.4   -1.8
```

```
com.long <- melt(com.wide)
names(com.long) <- c("indicator", "quarter", "change")
com.long <- dcast(com.long, quarter ~ indicator)
com.long
```

```
##   quarter Exports of goods and services*
## 1  Q20131                           -0.8
## 2  Q20132                            6.3
## 3  Q20133                            5.1
## 4  Q20134                           10.0
## 5  Q20141                           -9.2
## 6  Q20142                           11.1
## 7  Q20143                            4.5
## 8  Q20144                            3.2
##   Government consumption expenditures and gross investment*
## 1                                                      -3.9
## 2                                                       0.2
## 3                                                       0.2
## 4                                                      -3.8
## 5                                                      -0.8
## 6                                                       1.7
## 7                                                       4.4
## 8                                                      -1.8
##   Gross domestic product* Gross domestic purchases*
## 1                     2.7                       2.7
## 2                     1.8                       2.2
## 3                     4.5                       3.8
## 4                     3.5                       2.3
## 5                    -2.1                      -0.4
## 6                     4.6                       4.8
## 7                     5.0                       4.1
## 8                     2.2                       3.3
##   Imports of goods and services* Nonresidential fixed investment*
## 1                           -0.3                               1.5
## 2                            8.5                               1.6
## 3                            0.6                               5.5
## 4                            1.3                              10.4
## 5                            2.2                               1.6
## 6                           11.3                               9.7
## 7                           -0.9                               8.9
## 8                           10.1                               4.8
##   Personal consumption expenditures* Residential investment*
## 1                                3.6                     7.8
## 2                                1.8                    19.0
## 3                                2.0                    11.2
## 4                                3.7                    -8.5
## 5                                1.2                    -5.3
## 6                                2.5                     8.8
## 7                                3.2                     3.2
## 8                                4.2                     3.4
```

Now all I need to do is change the variable names. For this, I use the `gsub()` function, which is very helpful for the purpose of changing each element of a character vector:

```
names(com.long) <- gsub(pattern = "[/*]",
                        replacement = "",
                        x = names(com.long))
com.long
```

```
##   quarter Exports of goods and services
## 1  Q20131                          -0.8
## 2  Q20132                           6.3
## 3  Q20133                           5.1
## 4  Q20134                          10.0
## 5  Q20141                          -9.2
## 6  Q20142                          11.1
## 7  Q20143                           4.5
## 8  Q20144                           3.2
##   Government consumption expenditures and gross investment
## 1                                                     -3.9
## 2                                                      0.2
## 3                                                      0.2
## 4                                                     -3.8
## 5                                                     -0.8
## 6                                                      1.7
## 7                                                      4.4
## 8                                                     -1.8
##   Gross domestic product Gross domestic purchases
## 1                    2.7                     2.7
## 2                    1.8                     2.2
## 3                    4.5                     3.8
## 4                    3.5                     2.3
## 5                   -2.1                    -0.4
## 6                    4.6                     4.8
## 7                    5.0                     4.1
## 8                    2.2                     3.3
##   Imports of goods and services Nonresidential fixed investment
## 1                          -0.3                             1.5
## 2                           8.5                             1.6
## 3                           0.6                             5.5
## 4                           1.3                            10.4
## 5                           2.2                             1.6
## 6                          11.3                             9.7
## 7                          -0.9                             8.9
## 8                          10.1                             4.8
##   Personal consumption expenditures Residential investment
## 1                               3.6                    7.8
## 2                               1.8                   19.0
## 3                               2.0                   11.2
## 4                               3.7                   -8.5
## 5                               1.2                   -5.3
## 6                               2.5                    8.8
## 7                               3.2                    3.2
## 8                               4.2                    3.4
```

For more info on how to use `gsub()`, see http://www.endmemo.com/program/R/gsub.php.

Generalizing this approach

This general approach will work the same way for bigger datasets. You will need to use a slightly different workflow if you have For more examples, have a look at the following tutorials:

- Using the `melt()` function (as above):
  - Converting data between wide and long format, by Winston Chang
  - An Introduction to reshape2, by Sean C. Anderson
- Using the `reshape()` function (different, but equally helpful):
  - reshape (from base) Explained: Part I, reshape (from base) Explained: Part II, by Tyler Rinker
  - How can I reshape my data in R?, by the UCLA Stat Consulting Group
  - Reshape using R, by Oscar Torres-Reyna
- Automatic Conversion of Tables to LongForm Dataframes, by Jimmy Oh

Reading in a dataset data from a public Dropbox folder

Lastly, some of you asked about how to read a dataset from a public Dropbox folder directly into R. This is not as straightforward as simply providing the URL, as you did when reading a dataset from my website into R. The reason for this is that Dropbox URLs start with `https://`. There is an easy solution for this, though, thanks to Christopher Gandrud, who wrote the book *Reproducible Research with R and RStudio* which is listed on my syllabus. For a brief walkthrough, see http://aaronbaggett.com/notes/2014/03/28/reading-secure-data-into-r-from-dropbox/ or http://christophergandrud.blogspot.com/2013/04/dropbox-r-data.html.

Let's work with the "commerce" spreadsheet I just used above. I saved it in my Dropbox and its public link is https://www.dropbox.com/s/27b5y87sviq4g0z/commerce.csv?dl=0. This means that the key I need to enter in the `source_DropboxData()` function is "27b5y87sviq4g0z". You need to first install (just once) the "repmis" package. Then, load the package. Now, you can use the following code to read in a file from a (public) dropbox link:

```
library(repmis)
commerce.dat <- source_DropboxData(file = "commerce.csv",
  key = "27b5y87sviq4g0z", sep = ",", header = TRUE)
```

Note that this function might ask you once whether you want to create a Cache folder, the very first time you use it. You'll have to answer that with "Y". If you knit a document, this might prevent the document from being compiled, so use this function interactively the very first time you use it.

Final thoughts and more resources

Data management and data processing will almost always eat up much of the overall time you spend on a quantitative research project. Often, you might end up spending 80% of your time on data management before getting to the remaining 20% that it takes you to analyze your data. Developing good habits and an easy-to-use toolset will therefore go long ways to make you a more efficient (and effective) researcher.

Here are some more tutorials on data cleaning and data processing if you find some time and want to practice your skills:

- dplyr package vignette, by Hadley Wickham

- dplyr tutorial, by Kevin Markham
- Hands-on dplyr tutorial for faster data manipulation in R, by dataschool.io
- Video: Hands-on dplyr tutorial for faster data manipulation in R, by dataschool.io
- dplyr: How to do data manipulation with R, by by Sharp Sight Labs