1

# Using R to clean spreadsheet data: an example from CoreEcon

Lee C. Adkins
Professor of Economics
Oklahoma State University
Stillwater, OK 74078

October 9, 2018

[1]lee.adkins@okstate.edu

# Chapter 1

# Introduction

In this note, data from a spreadsheet is loaded into R, cleaned of all unnecessary rows and columns, and exported to a file suitable for use by econometric software. Naturally, one could also process the data in R.

This exercise was inspired by Doing Economics `https://www.core-econ.org/doing-economics/index.html`, which is part of the CoreEcon project. Specifically, the 4th unit, Measuring wellbeing, is employed. The R coding is taken mostly from Part 4.2 of that exercise.

First, you'll need a copy of R, which is available from the Comprehensive R Archive Network (CRAN) `https://cran.r-project.org/`. Download and install the version for your operating system. Here are a couple of YouTube videos to help you get setup.

`https://youtu.be/EHjakj38Nnw`

and if you are using RStudio:

`https://youtu.be/z9rH0RPuPMc`

## 1.1 New libraries

The CoreEcon project suggests installing three external R libraries to your system. There are:

- `readxl`: imports an Excel spreadsheet
- `tidyverse`: data manipulation
- `reshape2`: used to manipulate data
- `ggplot2`: produces graphs, but installed with `tidyverse`

To install these packages use:

```
install.packages(c("readxl", "tidyverse", "reshape2"))
```

Then import the libraries:

```
library(readxl)
library(tidyverse)
library(reshape2)
```

**To manipulate the spreadsheet in this example only the `readxl` library is required.** The `readxl` package will read `.xls` as well as `.xlsx` formats. It has no external dependencies, which makes it easy to use on different platforms. It also re-encodes ascii data to UTF-8, which is preferred by some econometric softwares.

In the next step, we'll go get some data to use.

## 1.2 Data

The data I'm working with in this example is from the United Nations Development Programme. Here is a brief summary of what is in the set:

> The 20 statistical tables in this annex provide an overview of key aspects of human development. The first five tables contain the family of composite human development indices and their components estimated by the Human Development Report Office (HDRO). The sixth table is produced in partnership with the Oxford Poverty and Human Development Initiative (OPHI) and will be added in due course. Remaining tables present a broader set of indicators related to human development. The five dashboards use colour coding to visualize partial groupings of countries according to performance on each indicator.
>
> Unless otherwise noted, tables use data available to the HDRO as of 15 July 2018. All indices and indicators, along with technical notes on the calculation of composite indices and additional source information, are available at http://hdr.undp.org/en/data. Countries and territories are ranked by 2017 Human Development Index (HDI) value. Robustness and reliability analysis has shown that for most countries differences in HDI are not statistically significant at the fourth decimal place. For this rea- son countries with the same HDI value at three decimal places are listed with tied ranks.

The table that interests us is Table 1. You can either download it or the entire Excel workbook that contains all of the tables and dashboards.

Select the Download all 2018 HDR data box on the left-hand side of the page. Save the file in an easily accessible location, and make sure to give it a suitable name. The 'Technical notes' give a diagrammatic presentation of how the HDI is constructed from four indicators.

The HDI indicators are measured in different units and have different ranges, so in order to put them together into a meaningful index, we need to normalize the indicators using the following formula:

$$Dimension\ index = \frac{actual\ value - minimum\ value}{maximum\ value - minimum\ value}$$

This produces a value in between 0 and 1 (inclusive), and allows different indicators to be compared.

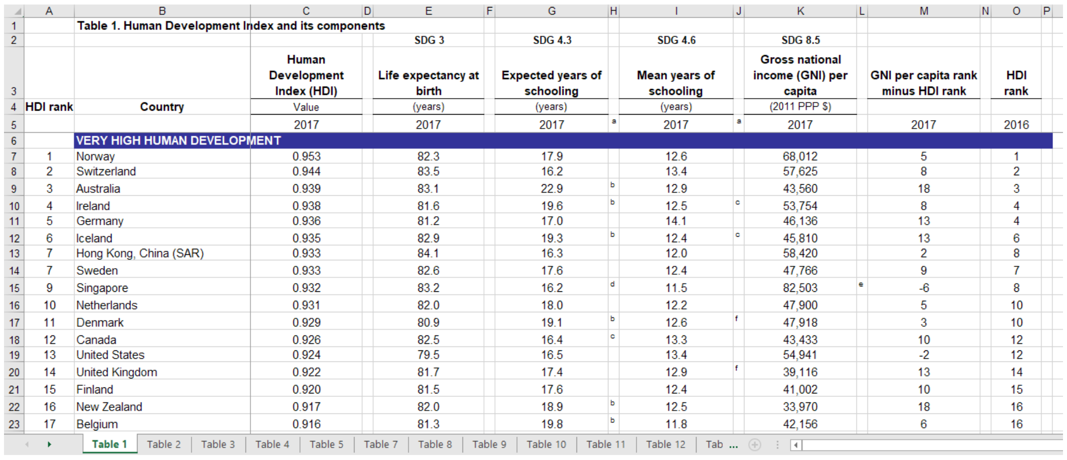
Now, we can combine these dimensional indices to give the HDI. The HDI is the geometric mean, a summary measure calculated by multiplying $N$ numbers together and then taking the $N$th root of this product. The geometric mean is useful when the items being averaged have different scoring indices or scales, because it is not sensitive to these differences, unlike the arithmetic mean. The geometric mean gives each variable the same influence over the value of the index, so doubling the value of one variable would have the same effect on the index as doubling the value of another variable.

In this case the HDI is the geometric mean of three dimension indices ($I_{Health}$ = Life expectancy index, $I_{Education}$ = Education index, and $I_{Income}$ = GNI index):

$$\text{HDI} = (I_{Health} \times I_{Education} \times I_{Income})^{1/3}$$

## 1.3 Read the data

The data are contained in an Excel workbook. We are interested in Table 1 which is shown below.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | Table 1. Human Development Index and its components | | | | | | | | | | | | | | |
| 2 | | | | | SDG 3 | | SDG 4.3 | | SDG 4.6 | | SDG 8.5 | | | | | |
| 3 | | | Human Development Index (HDI) | | Life expectancy at birth | | Expected years of schooling | | Mean years of schooling | | Gross national income (GNI) per capita | | GNI per capita rank minus HDI rank | | HDI rank | |
| 4 | HDI rank | Country | Value | | (years) | | (years) | | (years) | | (2011 PPP $) | | | | | |
| 5 | | | 2017 | | 2017 | | 2017 | a | 2017 | a | 2017 | | 2017 | | 2016 | |
| 6 | | VERY HIGH HUMAN DEVELOPMENT | | | | | | | | | | | | | | |
| 7 | 1 | Norway | 0.953 | | 82.3 | | 17.9 | | 12.6 | | 68,012 | | 5 | | 1 | |
| 8 | 2 | Switzerland | 0.944 | | 83.5 | | 16.2 | | 13.4 | | 57,625 | | 8 | | 2 | |
| 9 | 3 | Australia | 0.939 | | 83.1 | | 22.9 | b | 12.9 | | 43,560 | | 18 | | 3 | |
| 10 | 4 | Ireland | 0.938 | | 81.6 | | 19.6 | b | 12.5 | c | 53,754 | | 8 | | 4 | |
| 11 | 5 | Germany | 0.936 | | 81.2 | | 17.0 | | 14.1 | | 46,136 | | 13 | | 4 | |
| 12 | 6 | Iceland | 0.935 | | 82.9 | | 19.3 | b | 12.4 | c | 45,810 | | 13 | | 6 | |
| 13 | 7 | Hong Kong, China (SAR) | 0.933 | | 84.1 | | 16.3 | | 12.0 | | 58,420 | | 2 | | 8 | |
| 14 | 7 | Sweden | 0.933 | | 82.6 | | 17.6 | | 12.4 | | 47,766 | | 9 | | 7 | |
| 15 | 9 | Singapore | 0.932 | | 83.2 | | 16.2 | d | 11.5 | | 82,503 | e | -6 | | 8 | |
| 16 | 10 | Netherlands | 0.931 | | 82.0 | | 18.0 | | 12.2 | | 47,900 | | 5 | | 10 | |
| 17 | 11 | Denmark | 0.929 | | 80.9 | | 19.1 | b | 12.6 | f | 47,918 | | 3 | | 10 | |
| 18 | 12 | Canada | 0.926 | | 82.5 | | 16.4 | c | 13.3 | | 43,433 | | 10 | | 12 | |
| 19 | 13 | United States | 0.924 | | 79.5 | | 16.5 | | 13.4 | | 54,941 | | -2 | | 12 | |
| 20 | 14 | United Kingdom | 0.922 | | 81.7 | | 17.4 | | 12.9 | f | 39,116 | | 13 | | 14 | |
| 21 | 15 | Finland | 0.920 | | 81.5 | | 17.6 | | 12.4 | | 41,002 | | 10 | | 15 | |
| 22 | 16 | New Zealand | 0.917 | | 82.0 | | 18.9 | b | 12.5 | | 33,970 | | 18 | | 16 | |
| 23 | 17 | Belgium | 0.916 | | 81.3 | | 19.8 | b | 11.8 | | 42,156 | | 6 | | 16 | |

Some analysis of Table 1 shows that 189 countries are ranked by their HDI. This information will facilitate the clean-up. Notice that there are empty columns to eliminate, informational rows to delete, notes, footnotes and even some countries that are not ranked. There are also groups and regional aggregates of the data. All of these will be cleaned out of the data set before writing it to a .csv file.

The `read_excel` function is used to read the contents of the save spreadsheet into R. The sheet option specifies Table 1, and the table is read starting from line 3. Notice that the forward slash is used in the directory name. This is the Linux convention, not the Windows one. However, it is necessary here.

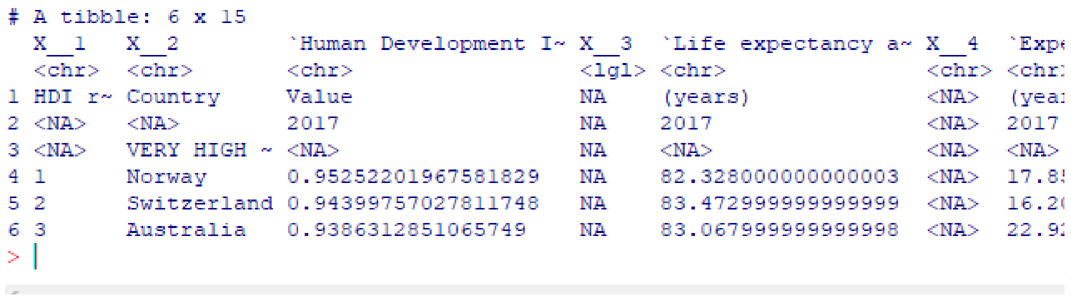
```
HDR2017 <- read_excel(
"C:/Users/leead/Documents/R/Data/2018_statistical_annex_all.xlsx",
                sheet="Table 1", # Worksheet to import
                skip = 2) # Number of rows to skip
```

One convention of R that deserves notice is that there is no line continuation command like the ones found in Stata or gretl. If a line is incomplete, R assumes that it is continued on the next line. Most of the time, this works fine. In the above case, each line ends in a comma and the open parenthesis closes the line 4.

The head and tail commands can be used to list the first (or last) few lines in the data frame as shown here:

```
head(HDR2017)
tail(HDR2017, n=6L)
```

Here are the first few rows of the data frame:



```
# A tibble: 6 x 15
  X__1   X__2        `Human Development I~ X__3  `Life expectancy a~ X__4  `Exp
  <chr>  <chr>       <chr>                 <lgl> <chr>               <chr> <chr
1 HDI r~ Country     Value                 NA    (years)             <NA>  (yea
2 <NA>   <NA>        2017                  NA    2017                <NA>  2017
3 <NA>   VERY HIGH ~ <NA>                  NA    <NA>                <NA>  <NA>
4 1      Norway      0.95252201967581829   NA    82.328000000000003  <NA>  17.8
5 2      Switzerland 0.94399757027811748   NA    83.472999999999999  <NA>  16.2
6 3      Australia   0.9386312851065749    NA    83.067999999999998  <NA>  22.9
> |
```

First not that the first two column names are X_1 and X_2. These should be changed. The names command is used for this.

```
names(HDR2017)[1] <- "HDI.rank"
names(HDR2017)[2] <- "Country"
```

The argument in parentheses is the name of the data frame (HDR2017). This is followed by the column number to name, and the name of the variable is listed to the right of the assignment operator ($<-$).

You can also use the existing name of a column as the basis for a name change. The last column in the frame is labeled **HDI rank**. We want to change this to HDI.rank.2016. The syntax is:

```
names(HDR2017)[names(HDR2017)=="HDI rank"] <- "HDI.rank.2016"
```

The command [names(HDR2017)==HDI rank"" replaces [1] in the previous use of names.

Also, the column titles appear as HDI, <NA>, and <NA>. These lines should be deleted. The subset command is useful here.

```
 HDR2017 <- subset(HDR2017,!is.na(HDI.rank) & HDI.rank != "HDI rank" )
```

The syntax is fairly straightforward. The first argument is the data frame that contains the full set. This is followed by a comma and then the condition which must be true in the subset assigned to HDR2017 which is to the left of the assignment operator.

The rows to be kept are: 1) those where `HDI.rank` is not equal to `<NA>` and (2) those where `HDI.rank` is not equal to `HDI rank`. Recall that column was renamed HDI.rank. In the first row the value of this variable is HDI rank. It will be dropped. The next two rows have values of `<NA>`, these will be dropped as well. The exclamation point (!) used before a statement negates it. So, `!is.na(HDI.rank)` means that the variable `HDI.rank` is not equal to na (which is the missing value code here).

Next, each of the remaining columns that begin with `X_` must be deleted. Again, the `subset` command can be used.

```
 sel_columns <- !startsWith(names(HDR2017),"X_")
```

In this case, we select all those columns that don't start with `X_`. These are put into `sel_columns`. Then `sel_columns` is used to identify the subset to retain using the select option.

```
 HDR2017 <- subset(HDR2017,select = sel_columns)
```

In some cases there will be more column headings and the generic formulation used above won't work. It is also possible to specify which columns to keep. For instance, to keep columns 1, 2, 3, 5, 7, 9, 11, 13, and 15 create the vector

```
sel_columns=c(1, 2, 3, 5, 7, 9, 11, 13, 15)
HDR2017 <- subset(HDR2017,select = sel_columns)
```

Finally, the remaining variables are given shorter names:

```
names(HDR2017)[3] <- "HDI"
names(HDR2017)[4] <- "LifeExp"
names(HDR2017)[5] <- "ExpSchool"
names(HDR2017)[6] <- "MeanSchool"
names(HDR2017)[7] <- "GNI.capita"
names(HDR2017)[8] <- "GNI.HDI.rank"
```

Though not strictly necessary, it's a good idea to identify what type of variables are in the data. Most of the series are numeric, but the country name is a factor variable. These are set using the as.numeric and as.factor functions.

```
HDR2017$HDI.rank <- as.numeric(HDR2017$HDI.rank)
HDR2017$Country <- as.factor(HDR2017$Country)
HDR2017$HDI <- as.numeric(HDR2017$HDI)
HDR2017$LifeExp <- as.numeric(HDR2017$LifeExp)
HDR2017$ExpSchool <- as.numeric(HDR2017$ExpSchool)
HDR2017$MeanSchool <- as.numeric(HDR2017$MeanSchool)
```

```
HDR2017$GNI.capita <- as.numeric(HDR2017$GNI.capita)
HDR2017$GNI.HDI.rank <- as.numeric(HDR2017$GNI.HDI.rank)
HDR2017$HDI.rank.2016 <- as.numeric(HDR2017$HDI.rank.2016)
str(HDR2017)
```

To write the data to a .csv file the write.table function works just fine.

```
write.table(HDR2017,
            file="C:/Users/leead/Documents/R/Data/HDI.csv",
            row.names=F,
            sep=",")
```

The data frame name is the first argument, then the file location and name of the output, `row.names=F` says that there are no row names in this data set and the separator is set to be a comma.

The `excel_sheets` command is used to identify sheet names in a workbook.

```
excel_sheets("C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx")
```

which yields:

```
> excel_sheets("C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx")
 [1] "Table 1"     "Table 2"     "Table 3"     "Table 4"     "Table 5"
 [6] "Table 7"     "Table 8"     "Table 9"     "Table 10"    "Table 11"
[11] "Table 12"    "Table 13"    "Table 15"    "Dashboard 1" "Dashboard 2"
[16] "Dashboard 3" "Dashboard 4" "Dashboard 5"
```

The complete workbook contains 18 spreadsheets.

# Chapter 2

# Importing the spreadsheet

Once the data are saved in a comma separated value spreadsheet, they are ready to import into other software for further processing. I'm using gretl in this example, but this works in Stata as well. Why not process in R? Well, you can. I just happen to be more familiar with gretl and Stata.

## 2.1 gretl

The spreadsheet is retrieved using the `open` command:

```
open C:/Users/ladkins/Documents/data/HDI.csv
```

Verify that the data were read in correctly by browsing or viewing the summary statistics. Also, check the contents of the main window to determine whether the sample size and variable names coincide with what you expect.

If all looks ok, save the data as a gretl dataset for future use.

```
store "H:\gretl\data\misc\HDI.gdt"
```

Next, one can create the indices and the HDI using gretl commands. First, we examing the summary statistics for two of the variables, expected schooling and average schooling. Then, a histogram of average schooling is generated.

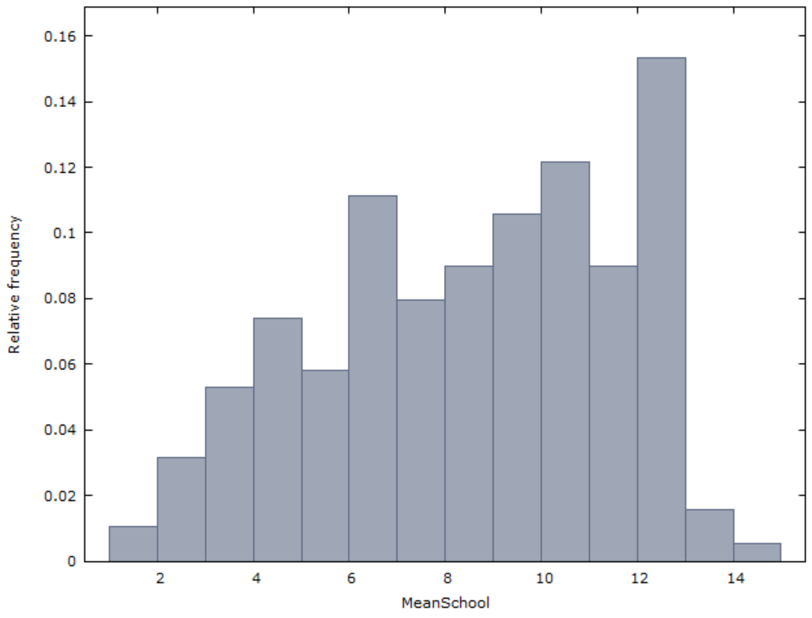
```
summary ExpSchool MeanSchool
freq MeanSchool --min=1 --binwidth=1 --plot=display
```

The summary statistics for the schooling variables are:

```
                    Mean      Median        S.D.         Min         Max
ExpSchool          13.24       13.19       2.927       4.872       22.92
MeanSchool         8.550       8.915       3.098       1.470       14.08
```

The maximum expected schooling is 22.92, which exceeds the theoretical maximum of 18. This will be addressed below before computing its index.

The histogram is:



The most frequent average schooling is 12 years. There are a few countries that exceed this.

Now the dimension indices are computed.

```
5 health=(LifeExp-20)/(85-20)
6 series censoredSchool = (ExpSchool>18) ? 18 : ExpSchool
7 series education = ((censoredSchool-0)/(18-0)+(MeanSchool-0)/(15-0))/2
8 series income = (ln(GNIcapita)-ln(100))/(ln(75000)-ln(100))
9 series HDIcalc = (health * education * income)^(1/3)
```

Since expected schooling exceeds 18 in a few cases, the variable must be censored when it exceeds the maximum. All observations that exceed 18 are censored at 18. This is done in two steps. In line 2 the `censoredSchool` variable is created and the `education` index based on it follows in the next line.

## 2.2 Stata

To import the .csv delimited data into Stata 15 use the import delimited command. The variable names reside in row 1 so use the varnames(1) option. As usual, the clear option allows you to replace the existing data with the imported.

```
import delimited C:\Users\leead\Documents\R\Data\HDI.csv, \\\
      varnames(1) clear numericcols(1,3:9)
```

Note, when Stata imports the .csv delimited data all variables are interpreted as strings. You must import numeric data using the numericcols(varlist) command. Here, the variable column numbers are listed (1 and 3 through 9). Column 2, the country name, remains a string.

# Chapter 3

# Scripts

## 3.1 R

```r
# install.packages(c("readxl","tidyverse","reshape2"))

library(readxl)
library(tidyverse)
library(reshape2)

HDR2017 <- read_excel(
"C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx",
                sheet="Table 1", # Worksheet to import
                skip = 2) # Number of rows to skip
excel_sheets("C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx")

head(HDR2017)
head(HDR2017, n=-6L)

# Rename the first two columns, currently named X_1 and X_2
names(HDR2017)[1] <- "HDI.rank"
names(HDR2017)[2] <- "Country"
# Rename the last column, which contains the 2016 rank
names(HDR2017)[names(HDR2017)=="HDI rank"] <- "HDI.rank.2016"
# Eliminate the row that contains the column title
HDR2017 <- subset(HDR2017,!is.na(HDI.rank) & HDI.rank != "HDI rank" )

# Check which variables do NOT (!) start with X_
sel_columns <- !startsWith(names(HDR2017),"X_")
# Select the columns that do not start with X_
HDR2017 <- subset(HDR2017,select = sel_columns)
str(HDR2017)

# Rename the other columns
names(HDR2017)[3] <- "HDI"
names(HDR2017)[4] <- "LifeExp"
names(HDR2017)[5] <- "ExpSchool"
names(HDR2017)[6] <- "MeanSchool"
names(HDR2017)[7] <- "GNI.capita"
names(HDR2017)[8] <- "GNI.HDI.rank"

# set the data as numeric or as factor variables
HDR2017$HDI.rank <- as.numeric(HDR2017$HDI.rank)
HDR2017$Country <- as.factor(HDR2017$Country)
HDR2017$HDI <- as.numeric(HDR2017$HDI)
HDR2017$LifeExp <- as.numeric(HDR2017$LifeExp)
HDR2017$ExpSchool <- as.numeric(HDR2017$ExpSchool)
HDR2017$MeanSchool <- as.numeric(HDR2017$MeanSchool)
HDR2017$GNI.capita <- as.numeric(HDR2017$GNI.capita)
HDR2017$GNI.HDI.rank <- as.numeric(HDR2017$GNI.HDI.rank)
HDR2017$HDI.rank.2016 <- as.numeric(HDR2017$HDI.rank.2016)
str(HDR2017)

# Write the data frame to a csv spreadsheet
write.table(HDR2017,
            file="C:/Users/ladkins/Documents/data/HDI.csv",
            row.names=F,
            sep=",")

# Manipulate variables
HDR2017$I.Health <- (HDR2017$LifeExp-20)/(85-20)
HDR2017$I.Education <- ((pmin(HDR2017$ExpSchool,18)-0)/(18-0) +
                           (HDR2017$MeanSchool-0)/(15-0))/2
HDR2017$I.Income <- (log(HDR2017$GNI.capita)-log(100))/
                    (log(75000)-log(100))
HDR2017$HDI.calc <- (HDR2017$I.Health *
                     HDR2017$I.Education * HDR2017$I.Income)^(1/3)
```

## 3.2 gretl

```
open C:/Users/ladkins/Documents/data/HDI.csv
summary --simple
store "H:\gretl\data\misc\HDI.gdt"

summary ExpSchool MeanSchool --simple
freq MeanSchool --min=1 --binwidth=1 --plot=display
health=(LifeExp-20)/(85-20)
series censoredSchool = (ExpSchool>18) ? 18 : ExpSchool
series education = ((censoredSchool-0)/(18-0)+(MeanSchool-0)/(15-0))/2
series income = (ln(GNIcapita)-ln(100))/(ln(75000)-ln(100))
series HDIcalc = (health * education * income)^(1/3)
```

## 3.3 Bonus Example

Here, I extracted the data from Table 9. This one is slightly more complicated since the column structure is nested.

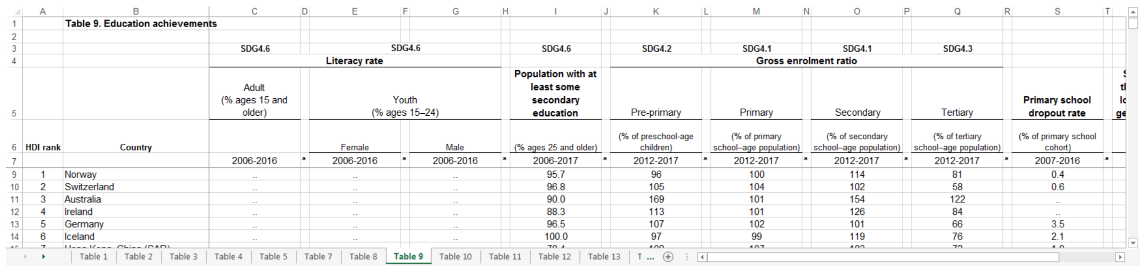


**Table 9. Education achievements**

| | | SDG4.6 | SDG4.6 | | SDG4.6 | SDG4.2 | SDG4.1 | SDG4.1 | SDG4.3 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Literacy rate | | | | Gross enrolment ratio | | | | |
| | | Adult (% ages 15 and older) | Youth (% ages 15–24) | | Population with at least some secondary education | Pre-primary | Primary | Secondary | Tertiary | Primary school dropout rate |
| HDI rank | Country | | Female | Male | (% ages 25 and older) | (% of preschool-age children) | (% of primary school–age population) | (% of secondary school–age population) | (% of tertiary school–age population) | (% of primary school cohort) |
| | | 2006-2016 | 2006-2016 | 2006-2016 | 2006-2017 | 2012-2017 | 2012-2017 | 2012-2017 | 2012-2017 | 2007-2016 |
| 1 | Norway | .. | .. | .. | 95.7 | 96 | 100 | 114 | 81 | 0.4 |
| 2 | Switzerland | .. | .. | .. | 96.8 | 105 | 104 | 102 | 58 | 0.6 |
| 3 | Australia | .. | .. | .. | 90.0 | 169 | 101 | 154 | 122 | .. |
| 4 | Ireland | .. | .. | .. | 88.3 | 113 | 101 | 126 | 84 | .. |
| 5 | Germany | .. | .. | .. | 96.5 | 107 | 102 | 101 | 66 | 3.5 |
| 6 | Iceland | .. | .. | .. | 100.0 | 97 | 99 | 119 | 76 | 2.1 |

```
HDR2017 <- read_excel(
"C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx",
                sheet="Table 9", # Worksheet to import
                skip = 2) # Number of rows to skip

excel_sheets("C:/Users/ladkins/Documents/data/statistical_annex_all.xlsx")

head(HDR2017)

names(HDR2017)[1] <- "HDI.rank" # Rename the first column, currently named X_1
names(HDR2017)[2] <- "Country" # Rename the second column, currently named X_2
HDR2017 <- subset(HDR2017,!is.na(Country))
HDR2017 <- subset(HDR2017,!is.na(HDI.rank) & HDI.rank != "HDI rank" )

sel_columns <- c(1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23)
HDR2017 <- subset(HDR2017,select = sel_columns)

str(HDR2017)
head(HDR2017)
tail(HDR2017)

names(HDR2017)[3] <- "adultLiteracy"
names(HDR2017)[4] <- "femaleYouthLiteracy"
names(HDR2017)[5] <- "maleYouthLiteracy"
names(HDR2017)[6] <- "someSecondary"
names(HDR2017)[7] <- "edPrePrimary"
names(HDR2017)[8] <- "edPrimary"
names(HDR2017)[9] <- "edSecondary"
names(HDR2017)[10] <- "edTertiary"
names(HDR2017)[11] <- "primaryDropOut"
names(HDR2017)[12] <- "survival"
names(HDR2017)[13] <- "govtExpenditure"

write.table(HDR2017,
            file="C:/Users/ladkins/Documents/data/edAchievement.csv",
            row.names=F,
            sep=",")
```