SG1022 Seminar 2: Composite Indicators

Christopher Gandrud

2016

Objectives

- Getting data into R from the World Bank Development Indicators with `WDI`
- Dealing with missing data
- Correlation (statistics and plots)
- Rescaling (with functions)
- Weighting and Aggregating

Pop Quiz

- What is the **difference** between R and RStudio?
- In R, what are **packages** and how do you **install** and load them?
- What are **objects** and what are **functions**? Give examples.
- What is the **assignment operator**? What is **component selection**?

World Development Indicators

You can also load data stored **remotely** (on another computer) into R. There are many ways to do this, depending on the data source.

Today we will download data from the World Bank's World Development Indicators using the WDI package.

Install packages.

Remember that to install a package use the `install.packages` function. **You only need to do this once.**

Today we will use six new packages that you need to install:

```r
# Create a vector of the packages to install
packages <- c('WDI', 'dplyr', 'DataCombine' 'Amelia',
              'corrplot', 'googleVis', 'ggplot2')

# Install packages
install.packages(packages)
```

Loading packages

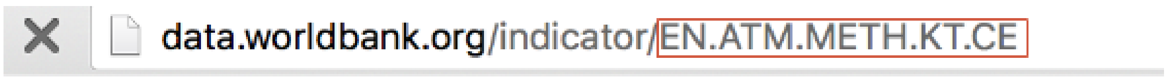
Each time you start R and want to use functions from a package, you need to load the package with the `library` function. So, for today use:

```r
library(WDI)
library(dplyr)
library(DataCombine)
library(Amelia)
library(corrplot)
library(googleVis)
library(ggplot2)
```

Remember to **include this code at the top** of your source code file to ensure that it runs correctly.

Find WDI Indicator ID

- **Go** the the World Bank's website: http://data.worldbank.org/indicator.
- **Click** on the indicator you are interested in.
- **Copy** the indicator ID. Example for *Methane Emissions*:



Download WDI (1 indicator)

Now use the `WDI` function from the WDI package to download the indicator:

```r
# Download data. Place in new object called methane emissions
methane <- WDI(indicator = 'EN.ATM.METH.KT.CE', start = 1990,
                end = 2014)

head(methane)
```

```
##   iso2c     country EN.ATM.METH.KT.CE year
## 1    1A Arab World                NA 2014
## 2    1A Arab World                NA 2013
## 3    1A Arab World                NA 2012
## 4    1A Arab World                NA 2011
## 5    1A Arab World            437574 2010
## 6    1A Arab World                NA 2009
```

Download WDI (multiple indicators)

We can download multiple indicators at once. To do this simply create a **vector of ID code strings**.

Let's download the following 5 indicators related to environmental sustainability:

```r
indicators <- c('EN.ATM.METH.KT.CE', 'EG.USE.ELEC.KH.PC',
                'EN.ATM.CO2E.PC', 'SP.POP.GROW',
                'EG.USE.COMM.CL.ZS')

wdi <- WDI(indicator = indicators, start = 1990, end = 2014)

names(wdi)
```

```
## [1] "iso2c"             "country"           "year"
## [4] "EN.ATM.METH.KT.CE" "EG.USE.ELEC.KH.PC" "EN.ATM.CO2E.PC"
## [7] "SP.POP.GROW"       "EG.USE.COMM.CL.ZS"
```

Some cleaning

We probably want to do some **cleaning** of this data set:

- **Rename** the indicator to something that is more intuitive.
- **Remove** units that are not countries (e.g. 'Arab World').

Renaming 1 variable

To rename variables in a data frame use the `rename` function from the dplyr package.

```r
methane <- rename(methane, methane_emissions = EN.ATM.METH.KT.CE)

names(methane)
```

```
## [1] "iso2c"             "country"           "methane_emissions"
## [4] "year"
```

Rename multiple variables

You can use the pipe `%>%` function (in dplyr) to help you rename multiple variables at the same time. (The pipe function takes one object and passes it to the first argument of the next function.)

```r
wdi <- wdi %>% rename(methane_emissions = EN.ATM.METH.KT.CE) %>%
               rename(electricity_use = EG.USE.ELEC.KH.PC) %>%
               rename(co2_emissions = EN.ATM.CO2E.PC) %>%
               rename(population_growth = SP.POP.GROW) %>%
               rename(alternative_energy = EG.USE.COMM.CL.ZS)

names(wdi)
```

```
## [1] "iso2c"             "country"            "year"
## [4] "methane_emissions" "electricity_use"    "co2_emissions"
## [7] "population_growth" "alternative_energy"
```

Removing non-countries (1)

All countries have an ISO 2 Letter Country Code. These include 2 letters. `iso2c` codes have patterns that we can use to select specific types of units:

- Regions (like 'Arab World') have `iso2c` codes that begin or end with a number.
- Economic groupings (Euroarea, Heavily indebted poor countries, etc) have `iso2c` letter codes beginning with X and Z (XC, XE, etc).
- Finally, we want to drop the EU (EU) and OECD (OE) in order to not double count units. . .

Removing non-countries (3)

```r
# Remove unwanted regions
regions <- unique(wdi$iso2c[grep('[0-9]', wdi$iso2c)])
regions <- c(regions, wdi$iso2c[grep('^[XZ]', wdi$iso2c)])
regions <- c(regions, 'EU', 'OE')

wdi <- subset(wdi, !(iso2c %in% regions))

head(wdi)[, 'country']
```

```
## [1] "Andorra" "Andorra" "Andorra" "Andorra" "Andorra" "Andorra"
```

Advanced: Regex

If you're interested: we use regular expressions to select character strings with certain characteristics (e.g. `[0-9]`, `[^[XZ]]`).

Note: regular expressions are very powerful, but also can take awhile to learn.

Missing Data

Remember that in R, missing data is usually coded `NA`. Note that sometimes data set creators also use other codes, such as `-999`.

A good first step for exploring missing data is to use the `summary` function, which gives you a count of the number of NA's. It will also help you identify if there are any `-999` codes, i.e. likely by showing unintuitive `min` and `max` values.

```r
summary(wdi$electricity_use)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's 
##    13.46   621.00  1976.00  3765.00  5125.00 53200.00     2232
```

Look at the data

Always take a look at your data to get a sense of the distribution of missing values.

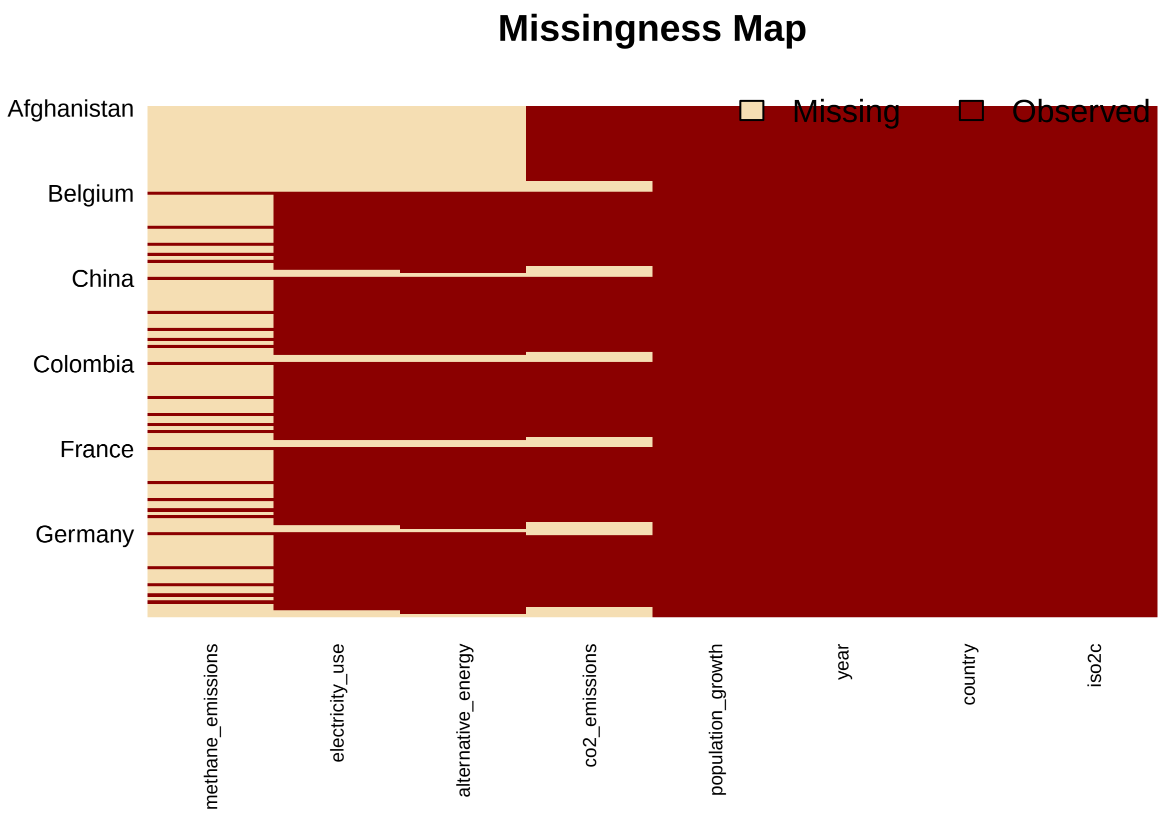
Why do you think values of the methane emissions variable missing?

Missingness map

One quick way to get an overview of the missing data across all of your variables is to create a **missingness map**. To do this use the `missmap` function from the Amelia package:

```r
# Subset the data (just to make the illustration easier to understand)
# Note: this is only for demonstration, you do not need to do this.
wdi_sub <- wdi %>% filter(country %in% c('Afghanistan', 'Belgium', 'China',
                                         'Colombia', 'France', 'Germany',
                                         'South Africa', 'Zambia'))

# Create missingness map
missmap(obj = wdi_sub,
        csvar = 'country', # country identifying variable
        tsvar = 'year',    # time identifying variable
        y.cex = 0.75,       # decrease y-axis labels so they fit
        x.cex = 0.55        # decrease x-axis labels so they fit
        )
```



We can quickly see from this plot that `methane_emmissions` has a lot of missing data and `population_growth` has few missing values.

Recode special values to NA

Special codes like `-999` often indicate specific reasons for missing data. You should take the time to **understand the substantive meaning** of these codes.

Ultimately, you may want to convert these into `NA` for analysis. For example:

```r
# NOTE: in this example nothing will change
# because there are no -999 values
wdi$electricity_use[wdi$electricity_use == -999] <- NA
```

Dropping observations with missing data (1 indicator)

You can drop observations with missing values on **one** variable with `subset`:

```r
wdi <- subset(wdi, !is.na(electricity_use))
```

Dropping observations with missing data (multiple indicators)

You can drop missing data on **multiple variables** with the `DropNA` function from the DataCombine package.

```r
# Indicators to create complete cases on
indicators_environ <- c('electricity_use', 'co2_emissions',
                        'population_growth', 'alternative_energy')

wdi <- DropNA(wdi, Var = indicators_environ)
```

```
## 2451 rows dropped from the data frame because of missing values.
```

Use this to get **complete cases** for your composite indicator.

Single impute missing values

Once you have analysed the reasons for your missing data, it **may** be reasonable to single impute values rather than drop cases.

For example, maybe it is reasonable to replace `NA` values with the variable `mean`:

```r
# Find mean methane
mean_methane <- mean(methane$methane_emissions,
                     na.rm = TRUE)

# Replace NAs with methane mean
# NOTE: nothing will change because we already dropped the NAs
methane$methane_emissions[
            is.na(methane$methane_emissions)] <- mean_methane
```

Note: these decisions need to be **fully justified**.

Correlation

One way to understand the structure of your components is to examine how they correlate with each other.

Use the `cor` function to find how two variables correlate with each other:

```
cor(wdi$electricity_use, wdi$co2_emissions, use = 'complete.obs')
```

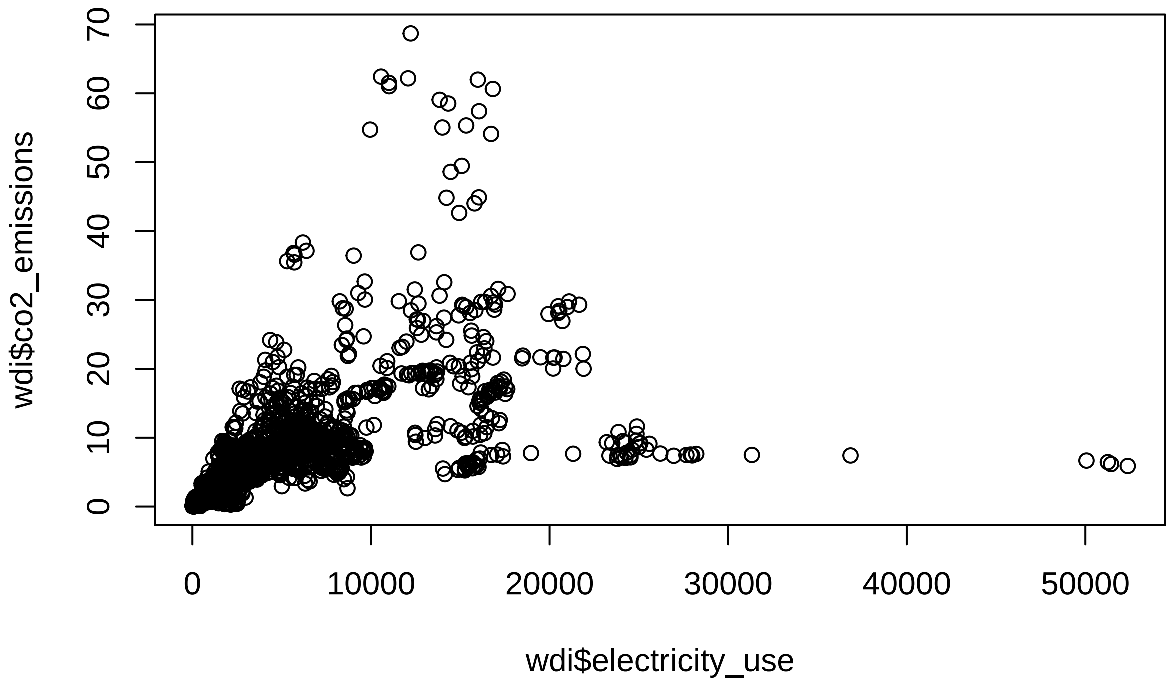
```
## [1] 0.6301933
```

This is the (linear) **correlation coefficient**.

Bi-variate plots

Another view with a bi-variate `plot`.

```
plot(wdi$electricity_use, wdi$co2_emissions)
```



Correlation matrix

You can create a correlation matrix to view multiple bi-variate correlations at once:

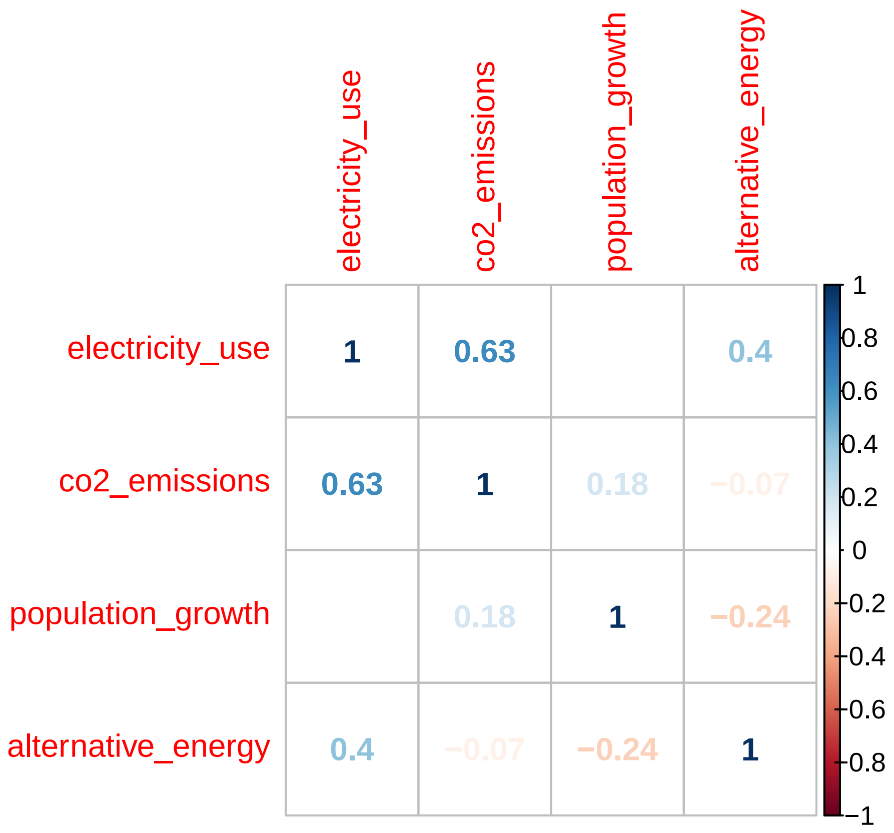
```
# Remember we created a vector of indicator names earlier
environ_cor <- cor(wdi[, indicators_environ], use = 'complete.obs')

environ_cor
```

```
##                   electricity_use co2_emissions population_growth
## electricity_use      1.0000000000    0.63019334     -0.0006567439
## co2_emissions        0.6301933356    1.00000000      0.1834125643
## population_growth   -0.0006567439    0.18341256      1.0000000000
## alternative_energy   0.4010761472   -0.07324789     -0.2398352393
##                   alternative_energy
## electricity_use           0.40107615
## co2_emissions              -0.07324789
## population_growth          -0.23983524
## alternative_energy          1.00000000
```

Easier view

```r
corrplot::corrplot(environ_cor, method = 'number')
```



Rescaling

As we discussed in the lecture, there are multiple ways you can rescale your component variables so that they are all on the same scale, e.g. **Min-Max**, and **Z-Scores**.

Before we learn these specific tools, let's learn a powerful new capability: creating your own functions.

Creating Functions

Use the `function` function to create new functions!

E.g. we can create a function to find the sample mean ($\bar{x} = \frac{\sum x}{n}$) of a vector.

```r
fun_mean <- function(x){
    sum(x) / length(x)
}
```

```r
## Find the mean
fun_mean(x = wdi$electricity_use)
```

```
## [1] 3749.494
```

Why create functions?

Functions:

- Simplify your code if you do repeated tasks.
- Lead to fewer mistakes.
- Are easier to understand.
- Save time over the long run–a general solution to problems in different contexts.

Min-Max function

To create a function to do Min-Max rescaling remember the equation:

$$I_{u,t} = \frac{x_{u,t} - \min(\mathrm{X})}{\max(\mathrm{X}) - \min(\mathrm{X})}$$

So the R function would be:

```r
min_max <- function(x) {
            (x - min(x, na.rm = T))/
            (max(x, na.rm = T) - min(x, na.rm = T))
}
```

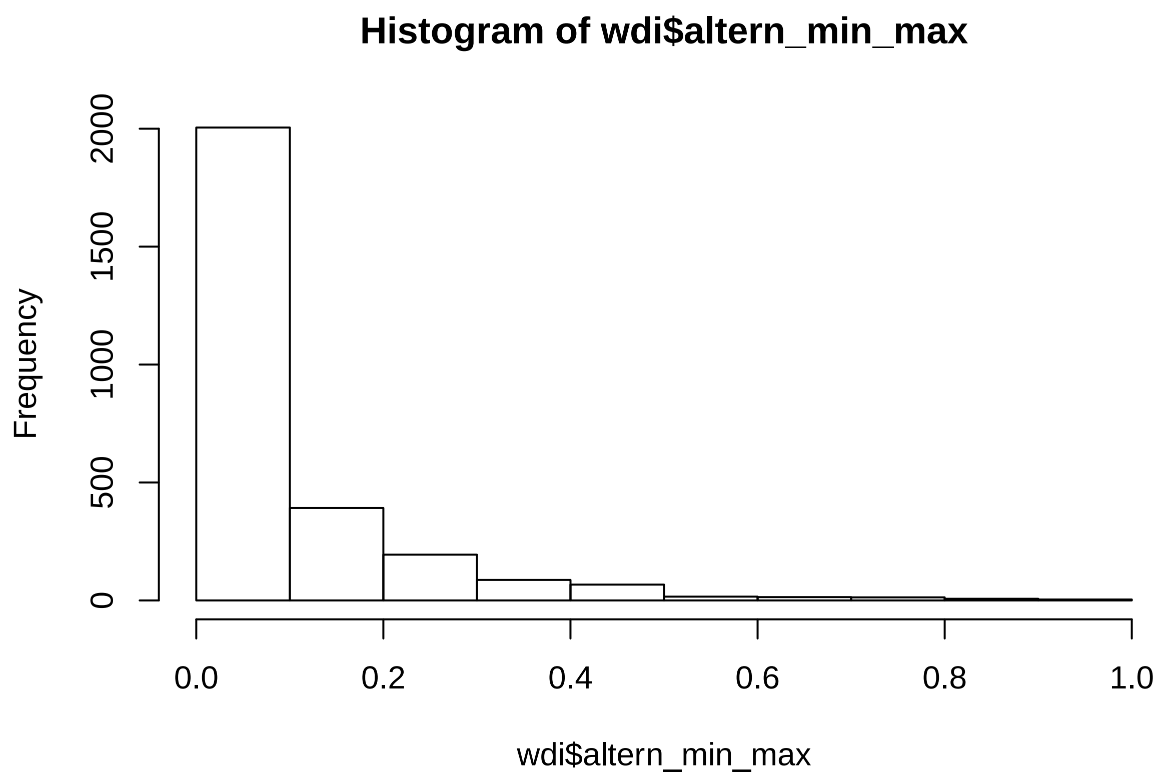
Min-Max rescale

Now use the function:

```r
wdi$altern_min_max <- min_max(wdi$alternative_energy)
```

Examine Min-Max distribution

```r
hist(wdi$altern_min_max)
```



Z-Score rescale

The equation for Z-Scores is:

$$I_{u,t} = \frac{x_{u,t} - \mu_{\mathrm{X}}}{\sigma_{\mathrm{X}}}$$

So, the R function would be:

```
z_score <- function(x) {
                (x - mean(x, na.rm = T)) /
                sd(x, na.rm = T)
}
```

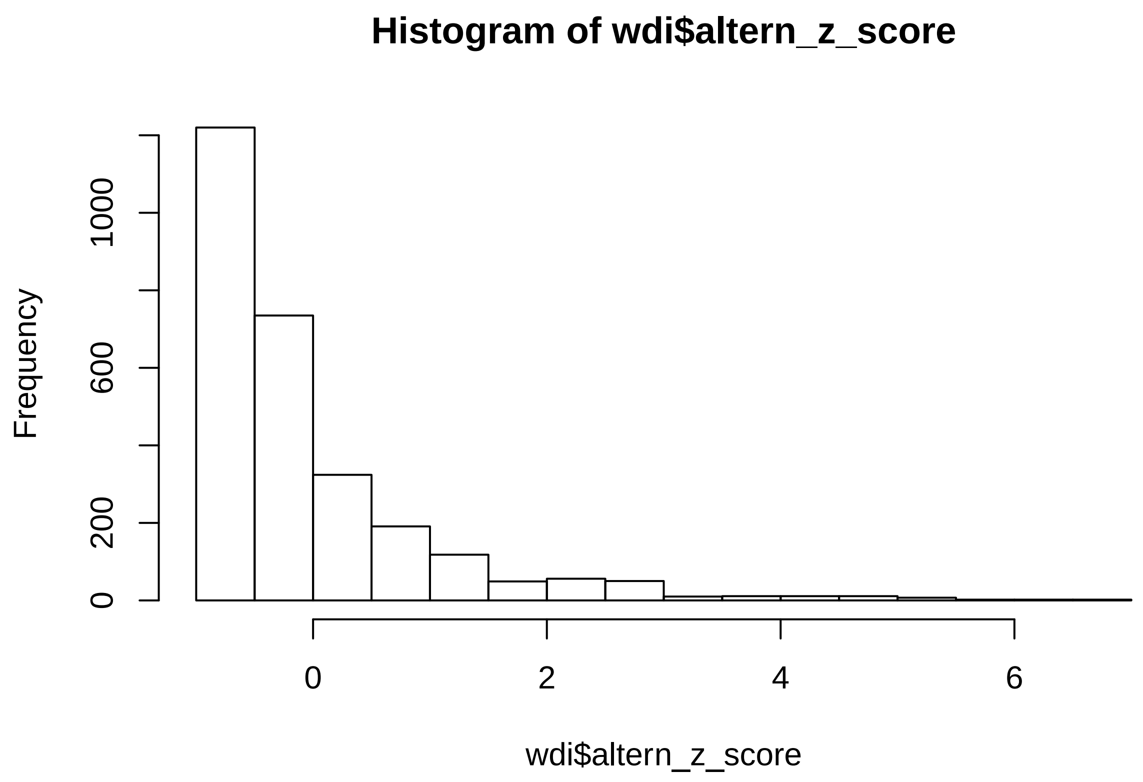
Z-Score rescale

Now use the function:

```
wdi$altern_z_score <- z_score(wdi$alternative_energy)
```

Examine Z-Score distribution

```
hist(wdi$altern_z_score)
```



Reverse a variable's direction

The equation to reverse a variable's direction:

$$I_{u,t} = \max(\mathrm{X}) - x_{u,t}$$

So the function would be:

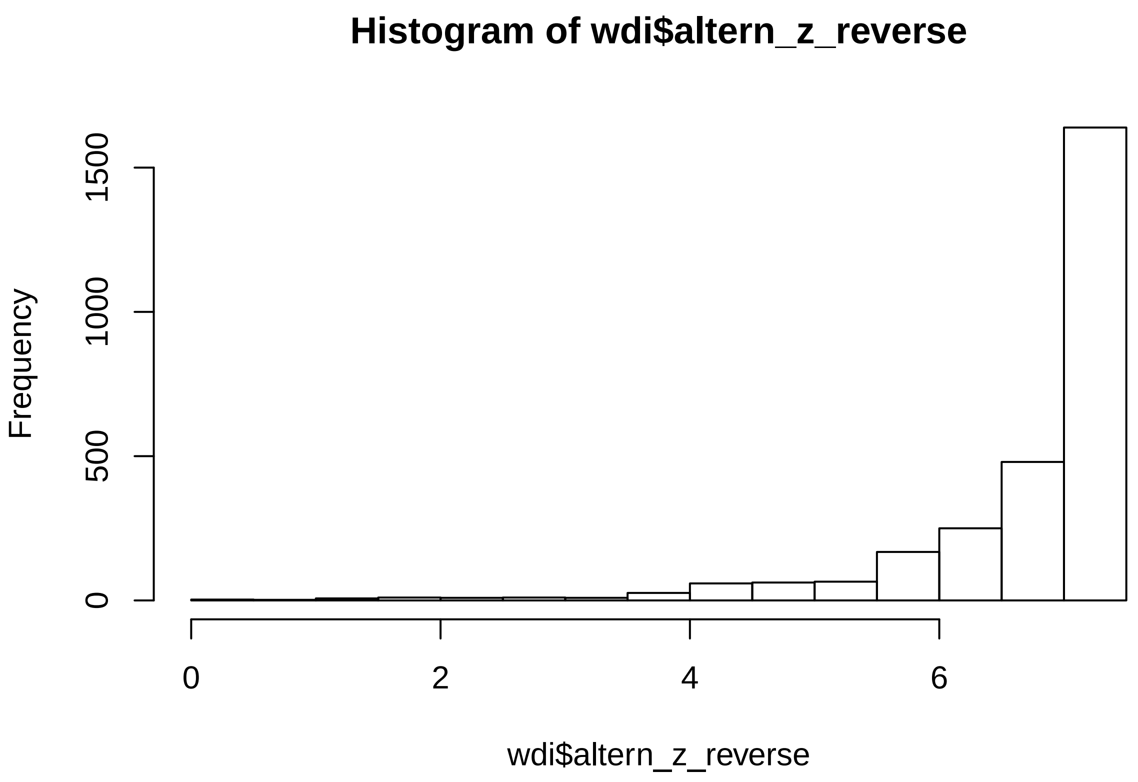
```
reverse_direction <- function(x) max(x, na.rm = T) - x
```

Now use the function:

```
wdi$altern_z_reverse <- reverse_direction(wdi$altern_z_score)
```

Examine reversed distribution

```
hist(wdi$altern_z_reverse)
```



You try

Put the following other variables on a Z-Score scale:

- `electricity_use`
- `co2_emissions`
- `population_growth`
- `alternative_energy`

Weight/Aggregate

Once we have our rescaled components, we then decide how to weight and aggregate our indicators.

For this course you will use '**expert-judgement**'.

Weight/Aggregate example

Imagine we have four variables that we want to combine into an Environmental Unsustainability index: `electricity_use`, `co2_emissions`, `population_growth`, and `alternative_energy`.

We have use z-scores to rescale them and reversed the direction of `alternative_energy`.

The results are in a data frame called `wdi_sub`.

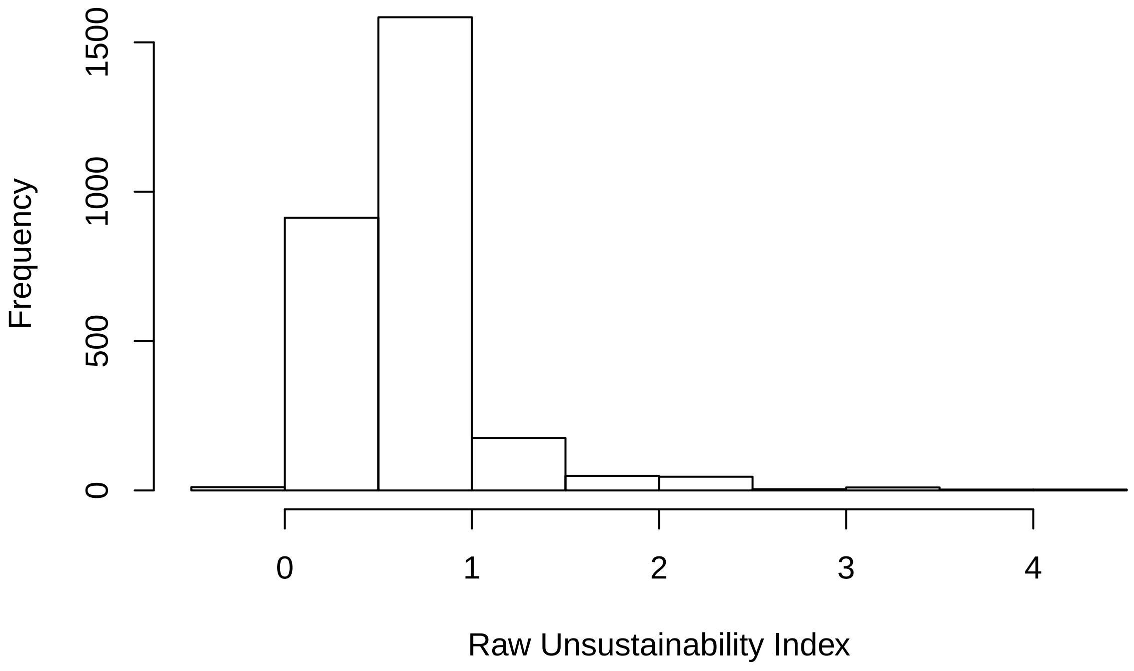
Weight/Aggregate example

We think that `co2_emissions` is particularly important so we give it a weighting of 0.3, the others have a weighting of 0.1:

```
wdi_sub$unsustainability <- wdi_sub$co2_emissions * 0.3 +
                                  wdi_sub$electricity_use * 0.1 +
                                  wdi_sub$population_growth * 0.1 +
                                  wdi_sub$alternative_energy * 0.1
```

Component indicator results

```
hist(wdi_sub$unsustainability, main = '',
     xlab = 'Raw Unsustainability Index')
```



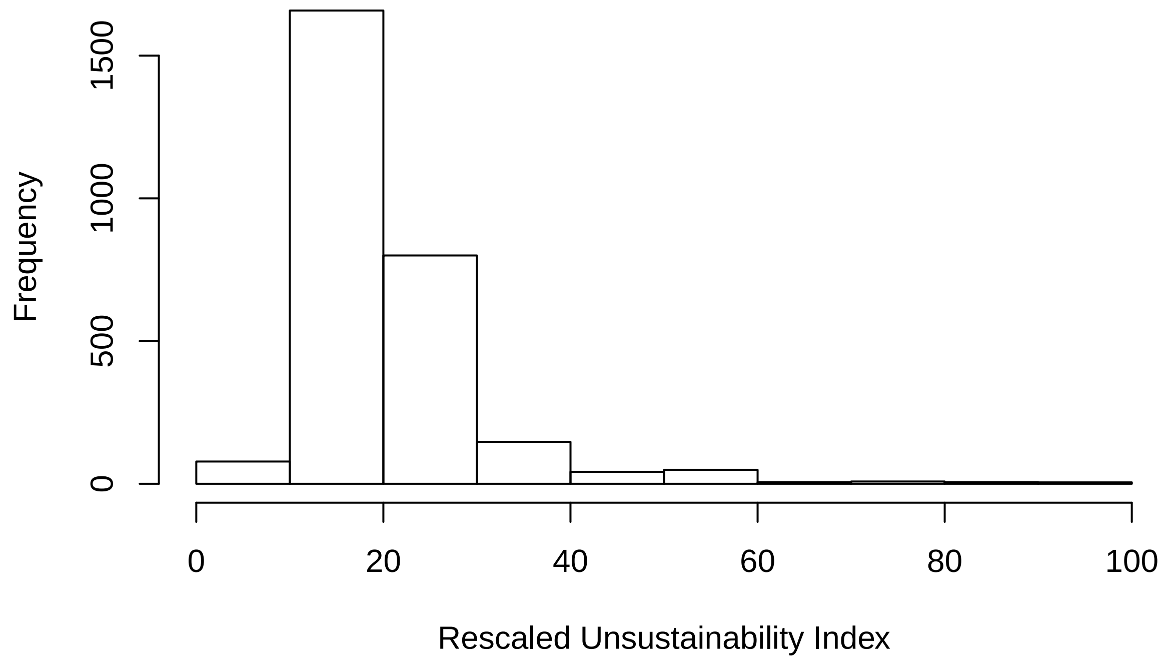
Rescale the index

We could of course rescale the index so that it is between 0 and 100.

```
wdi_sub$unsustainability <- min_max(wdi_sub$unsustainability) * 100
```

Rescaled index

```
hist(wdi_sub$unsustainability, main = '',
     xlab = 'Rescaled Unsustainability Index')
```



Map the index

You could also map the results (good sanity check):

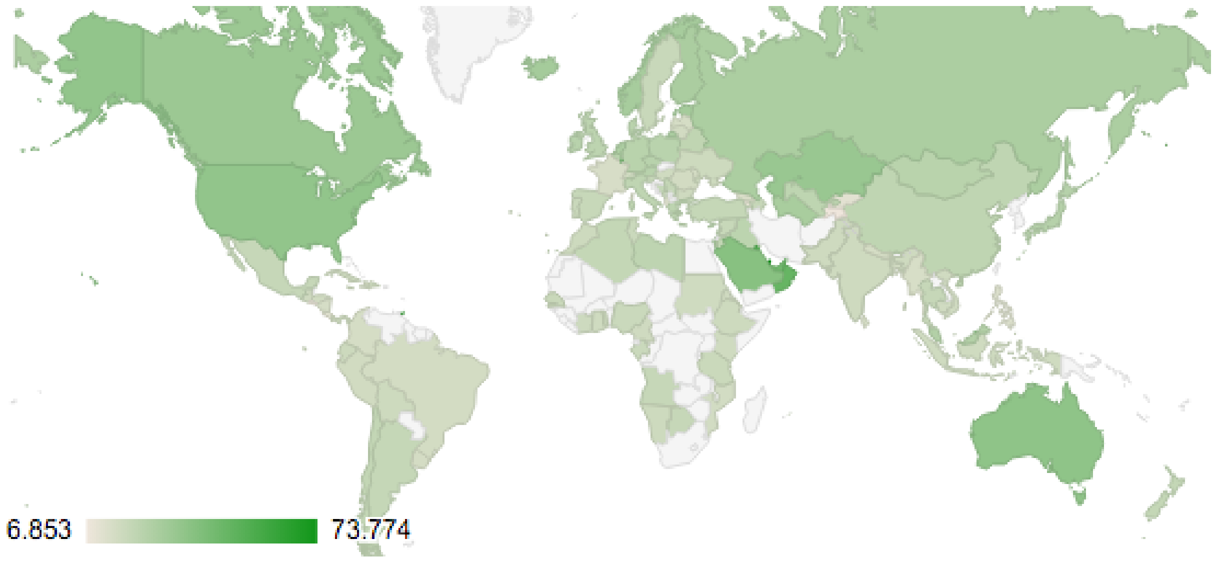
```r
# Subset data for only 2011
wdi_2011 <- subset(wdi_sub, year == 2011)

# Use the googleVis package to create the map
library(googleVis)

map <- gvisGeoChart(wdi_2011, locationvar = "country",
                    colorvar = "unsustainability")
```

Map the index

```r
plot(map)
```



Index over time

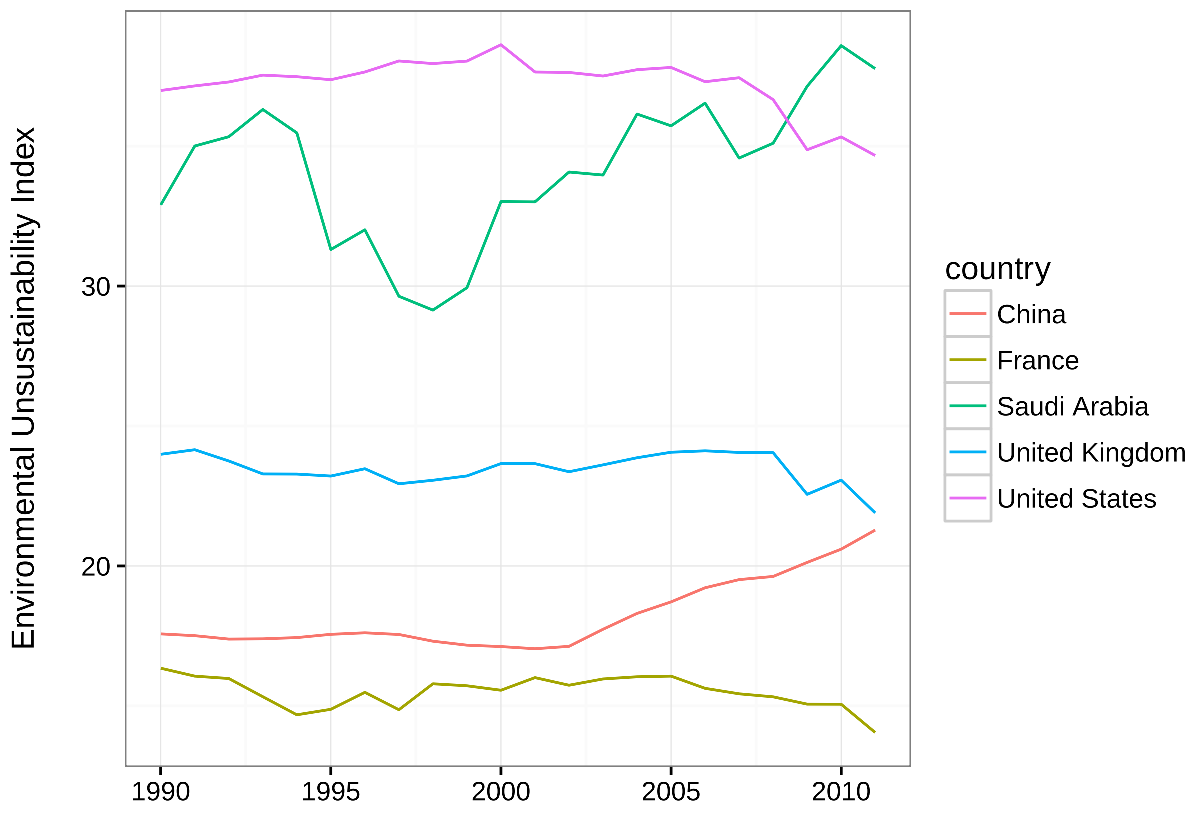
When you create an index for units (e.g. countries) over time (e.g. years) it is useful to also plot these changes.

```r
# Select specific countries
keep <- c('China', 'Saudi Arabia', 'France', 'United States',
          'South Africa', 'United Kingdom')
wdi_countries <- subset(wdi_sub, country %in% keep)

# Plot
library(ggplot2)
index_plot <- ggplot(wdi_countries,
                     aes(x = year, y = unsustainability,
                         colour = country)) +
    geom_line() + xlab('') +
    ylab('Environmental Unsustainability Index\n') +
    theme_bw()
```

Index over time

```r
index_plot
```



Experiment

It is important to **try and compare** multiple weighting schemes to examine how sensitive the index is to each one.

You do . . .

With a partner, using World Bank Development Indicators create an **Educational Achievement Index**:

- Select and download at least 4 indicators
- Examine and deal with missing values
- Explore the variables with a correlation matrix
- Put the variables on the same scale and reverse variable directions as need be.
- Weight and aggregate the variables into an composite index.
- Display the results (line chart and map)