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

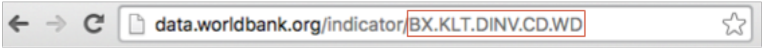
- What is the **difference** between R and RStudio?
- In R, what are **packages** and how do you **install** and load them?
- What are **objects** and what are **functions**? Give examples.
- What is the **assignment operator**? What is **component selection**?

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.

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 *Foreign Direct Investment*:



data.worldbank.org/indicator/BX.KLT.DINV.CD.WD

Download WDI

Now use the WDI function to download the indicator:

```
# Load WDI package
library(WDI)

# Download data. Place in new object called fdi
fdi <- WDI(indicator = 'BX.KLT.DINV.CD.WD', start = 1990, e

head(fdi)
```

```
##   iso2c    country BX.KLT.DINV.CD.WD year
## 1    1A Arab World       43135000919 2014
## 2    1A Arab World       46598273225 2013
## 3    1A Arab World       49144078588 2012
## 4    1A Arab World       44755312886 2011
## 5    1A Arab World       66667048123 2010
## 6    1A Arab World       79522853477 2009
```

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 variables

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

```
library(dplyr)

fdi <- rename(fdi, foreign_direct_investment = BX.KLT.DINV.

head(fdi)
```

```
##   iso2c    country foreign_direct_investment year
## 1    1A Arab World               43135000919 2014
## 2    1A Arab World               46598273225 2013
## 3    1A Arab World               49144078588 2012
## 4    1A Arab World               44755312886 2011
## 5    1A Arab World               66667048123 2010
## 6    1A Arab World               79522853477 2009
```

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)

```
# Create vector of all iso2c codes for units we don't want
regions <- unique(fdi$iso2c[grep('[0-9]', fdi$iso2c)])
regions <- c(regions, fdi$iso2c[grep('^[XZ]', fdi$iso2c)])
regions <- c(regions, 'EU', 'OE')

# Keep rows with iso2c codes that are not regions
fdi <- subset(fdi, !(iso2c %in% regions))

head(fdi)
```

```
##     iso2c     country foreign_direct_investment year
## 851    AF Afghanistan                  48756005 2014
## 852    AF Afghanistan                  39663686 2013
## 853    AF Afghanistan                  61525860 2012
## 854    AF Afghanistan                  57620844 2011
## 855    AF Afghanistan                  54200551 2010
## 856    AF Afghanistan                 197512727 2009
```

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.

Download multiple WDI indicators

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

```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
## [7] "SP.POP.GROW"       "EG.USE.COMM.CL.ZS"
```

Cleaning code for reference

```
library(dplyr) # contains rename and %>% functions

# 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))

# Rename indicators
wdi <- wdi %>% rename(methane_emissions = EN.ATM.METH.KT.C
               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.
```

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.

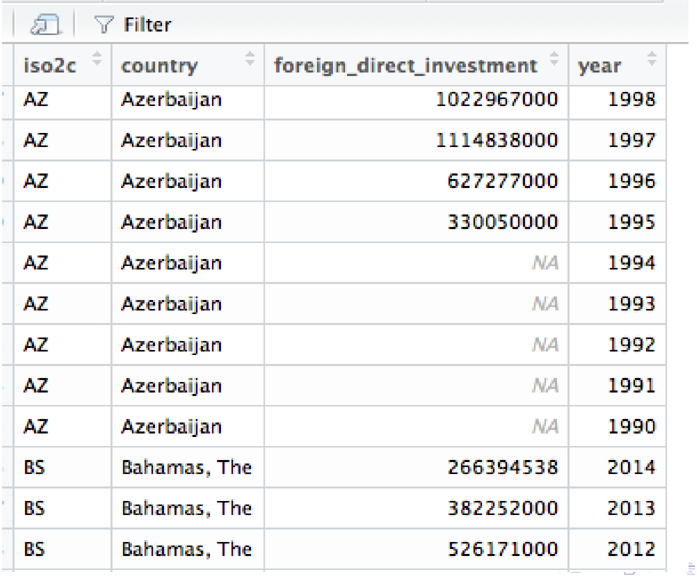
A good first step for exploring missing data is to use the `summary` function, which gives you a count of the number of NA's. It will also help you identify if there are any -999 codes, i.e. likely by showing unintuitive `min` and `max` values.

```
summary(fdi$foreign_direct_investment)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.
## -3.582e+10  3.140e+07  2.662e+08  5.955e+09  2.005e+09
##       NA's
##        685
```

Look at the data

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



iso2c	country	foreign_direct_investment	year
AZ	Azerbaijan	1022967000	1998
AZ	Azerbaijan	1114838000	1997
AZ	Azerbaijan	627277000	1996
AZ	Azerbaijan	330050000	1995
AZ	Azerbaijan	NA	1994
AZ	Azerbaijan	NA	1993
AZ	Azerbaijan	NA	1992
AZ	Azerbaijan	NA	1991
AZ	Azerbaijan	NA	1990
BS	Bahamas, The	266394538	2014
BS	Bahamas, The	382252000	2013
BS	Bahamas, The	526171000	2012

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, imagine we have a data frame called survey and we want to recode all -999 values of some variable called trust as NA

```r
data$trust[data$trust == -999] <- NA
```

Dropping observations with missing data (1)

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

```r
fdi <- subset(fdi, is.na(foreign_direct_investment))
```

Dropping observations with missing data (2)

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

```
library(DataCombine)

# Indicators to create complete cases on
indicators_environ <- c('electricity_use', 'co2_emissions'
                        'population_growth', 'alternative_

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

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

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`:

```
fdi$foreign_direct_investment[
                        is.na(fdi$foreign_direct_investment
                    mean(fdi$foreign_direct_investment, na
```

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:

```r
cor(wdi$electricity_use, wdi$co2_emissions, use = 'complete
```

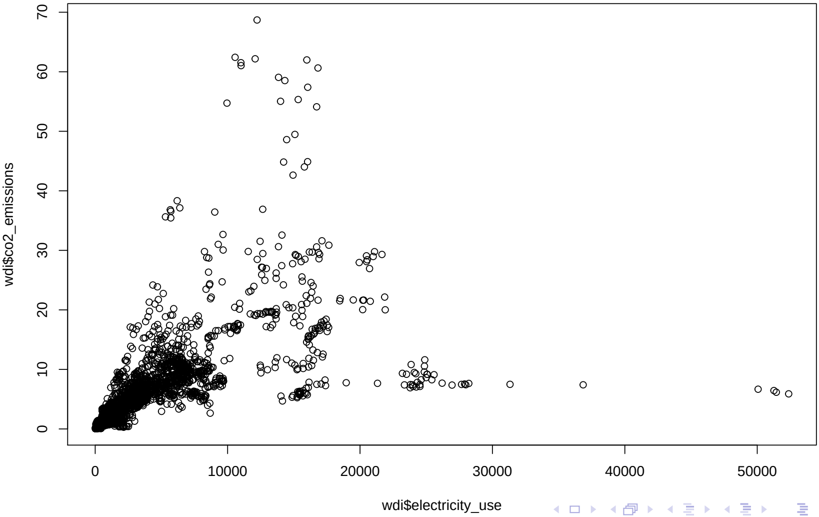
```
## [1] 0.6301933
```

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

Bi-variate plots

Another view with a bi-variate plot.

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



Correlation matrix

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

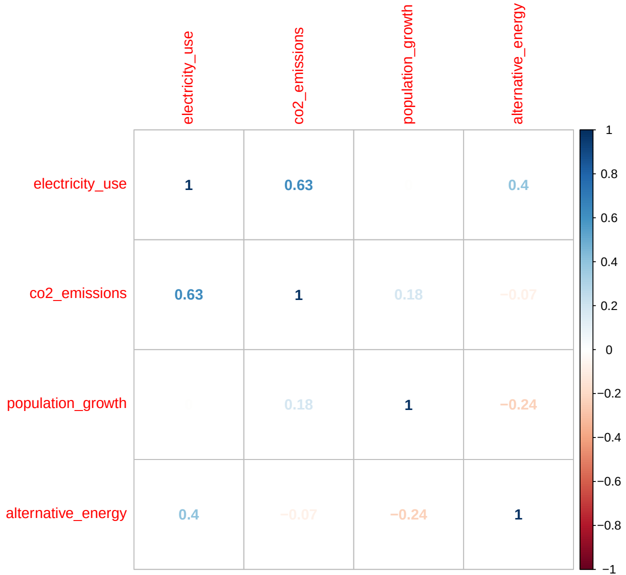
```
# Remember we created a vector of indicator names earlier
environ_cor <- cor(wdi[, indicators_environ], use = 'comple

environ_cor
```

```
##                      electricity_use co2_emissions populat
## electricity_use         1.0000000000    0.63019334    -0.
## co2_emissions           0.6301933356    1.00000000     0.
## population_growth      -0.0006567439    0.18341256     1.
## alternative_energy      0.4010761472   -0.07324789    -0.
##                      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.

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

## Find the mean
fun_mean(x = swiss$Examination)
```

```
## [1] 16.48936
```

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:

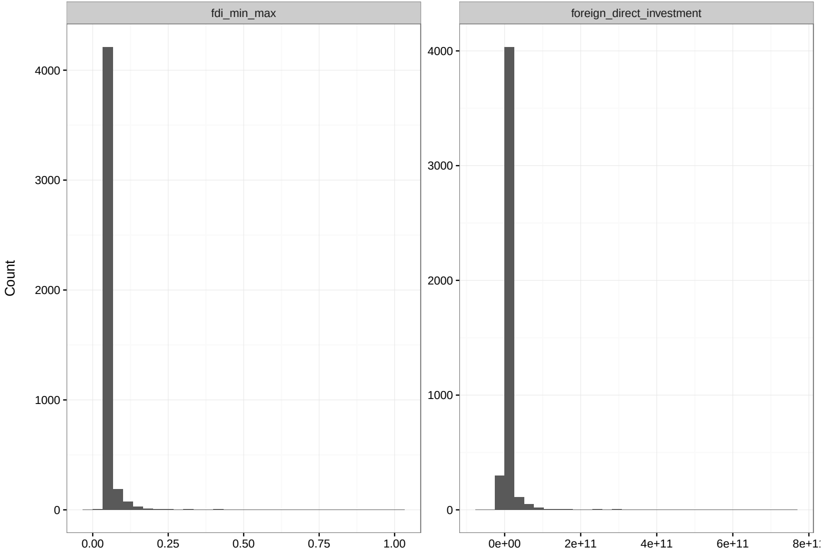
```r
min_max <- function(x) {
            (x - min(x, na.rm = T))/
            (max(x, na.rm = T) - min(x, na.rm = T))
}
```

Min-Max rescale

Now use the function:

```r
fdi$fdi_min_max <- min_max(fdi$foreign_direct_investment)
```

Compare original to 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:

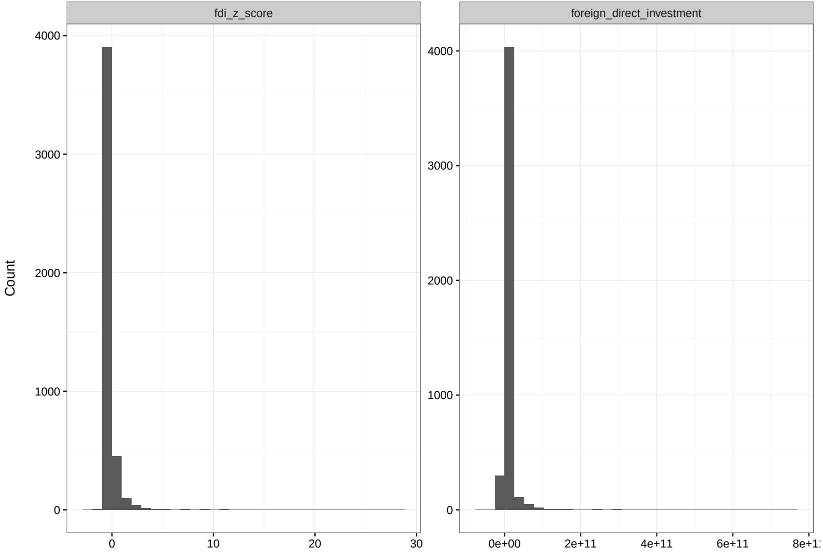
```r
z_score <- function(x) {
                (x - mean(x, na.rm = T)) /
                sd(x, na.rm = T)
}
```

Z-Score rescale

Now use the function:

```r
fdi$fdi_z_score <- z_score(fdi$foreign_direct_investment)
```

Compare original to Z-Score



Reverse a variable's direction

The equation to reverse a variable's direction:

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

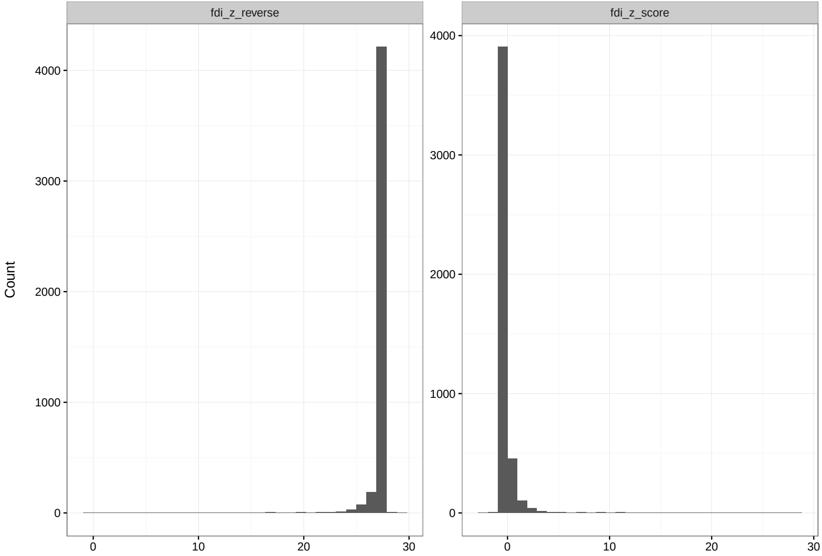
So the function would be:

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

Now use the function:

```
fdi$fdi_z_reverse <- reverse_direction(fdi$fdi_z_score)
```

Reverse


fdi_z_reverse
fdi_z_score
Count
4000
3000
2000
1000
0
0
10
20
30
4000
3000
2000
1000
0
0
10
20
30

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`.

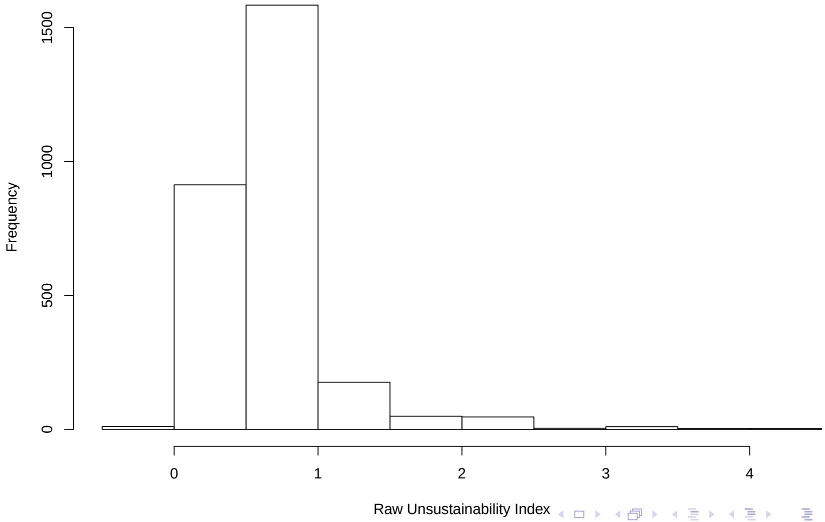
Weight/Aggregate example

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$electricity_use
                                  wdi_sub$population_growth
                                  wdi_sub$alternative_energ
```

Component indicator results

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



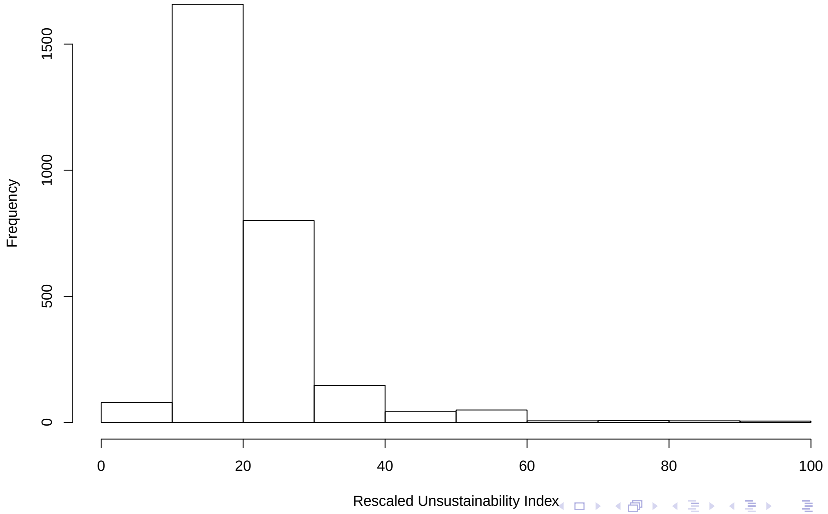
Rescale the index

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

```r
wdi_sub$unsustainability <- min_max(wdi_sub$unsustainabilit
```

Rescaled index

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



Map the index

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

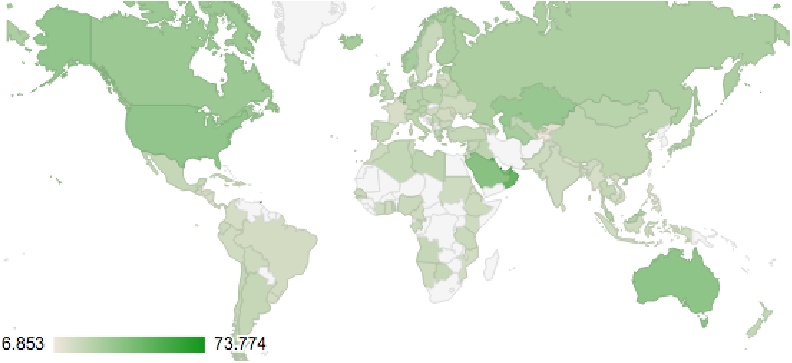
```
# Subset data for only 2011
wdi_2011 <- subset(wdi_sub, year == 2011)

# 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

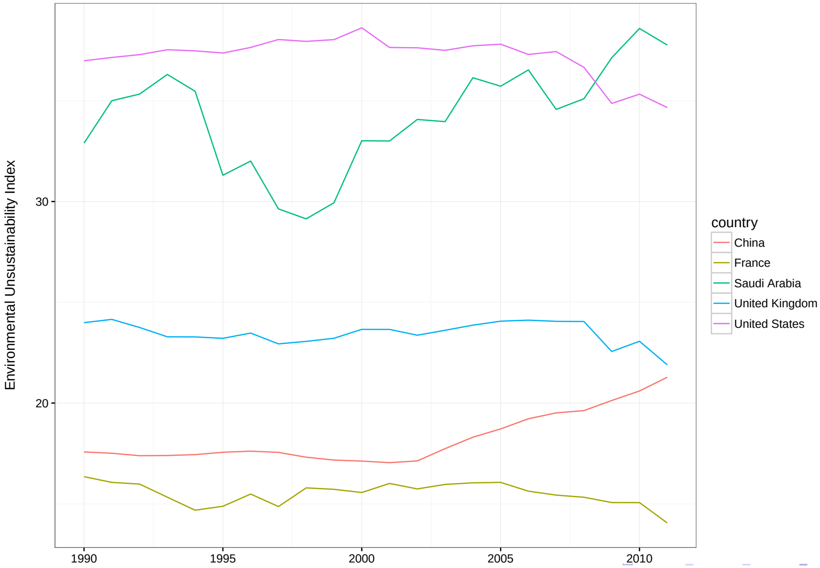
When you create an index for units (e.g. countries) over time (e.g. years) it is useful to also plot these changes.

```
# Select specific countries
keep <- c('China', 'Saudi Arabia', 'France', 'United State
          '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\
    theme_bw()
```

Index over time

```
index_plot
```



Experiment

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

Seminar: Make an Index

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 (histogram and map)