Intro to R - Session 2

Daniel Viana and Jessica Zamborain Mason

October, 2022

Table of Contents

In this session, we will learn how to manipulate and visualize data using the dplyr and ggplot packages

##Install packages First, we need to intall the tidyverse package, which contains a siut of packages that makes data wrangling and visualization a lot easier!

```
#Install the package. You can use the function below (install.packages) to
install any package. Alternativelly, if you are using R studio, you can go to
Tools>Install Packages and just select the package you which to install
#install.packages(tidyverse)

#Next, you need to upload the package to the library to make all functions
available to use
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.0 ──

## ✓ ggplot2 3.3.2     ✓ purrr   0.3.3
## ✓ tibble  2.1.3     ✓ dplyr   0.8.4
## ✓ tidyr   1.0.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0

## ── Conflicts ────────────────────────────────────────── 
tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

##Download data

First, lets download a database that we can work with. We will use a database from the world bank that estimates the prevalence of child malnutrition in the world. This data provides an estimate of the proportion of children under-five whose weight for age is less than minus two standard deviations from the median for the international reference population ages 0 to 59 months.

```r
#First, lets download the WDI (World development Indicators) package
#install.packages(WDI)
library(WDI)

#Dowload the data we want. Lets name the database "dat"
dat = WDI(country = "all", indicator = c("malnut" = "5.51.01.02.malnut"),
start=2005, end=2011, extra=TRUE, cache=NULL)

##If you do not have good internet connection, you can also read the csv data
file we provided (WDI_malnut_dat.csv). I have stored the data in a "data""
folder inside my project folder. To read this data, you can use the
read_csv() function:
dat = read_csv("~/Rbootcamp_Madagascar/data/WDI_malnut_dat.csv")
```

```
## Parsed with column specification:
## cols(
##   iso2c = col_character(),
##   country = col_character(),
##   malnut = col_double(),
##   year = col_double(),
##   iso3c = col_character(),
##   region = col_character(),
##   capital = col_character(),
##   longitude = col_double(),
##   latitude = col_double(),
##   income = col_character(),
##   lending = col_character()
## )
```

```r
#Now lets inspect the data
names(dat)
```

```
##  [1] "iso2c"     "country"   "malnut"    "year"      "iso3c"     "region"
##  [7] "capital"   "longitude" "latitude"  "income"    "lending"
```

```r
head(dat)
```

```
## # A tibble: 6 x 11
##   iso2c country malnut  year iso3c region capital longitude latitude
income
##   <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl> <chr>
## 1 4E    East A…     NA  2011 EAP   Aggre… <NA>           NA       NA
Aggre…
## 2 4E    East A…     NA  2010 EAP   Aggre… <NA>           NA       NA
Aggre…
## 3 4E    East A…     NA  2009 EAP   Aggre… <NA>           NA       NA
Aggre…
## 4 4E    East A…     NA  2008 EAP   Aggre… <NA>           NA       NA
Aggre…
## 5 4E    East A…     NA  2007 EAP   Aggre… <NA>           NA       NA
Aggre…
```

```
## 6 4E    East A…     NA  2006 EAP   Aggre… <NA>           NA       NA
Aggre…
## # … with 1 more variable: lending <chr>

str(dat)

## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 1078 obs. of  11
variables:
##  $ iso2c    : chr  "4E" "4E" "4E" "4E" ...
##  $ country  : chr  "East Asia & Pacific (excluding high income)" "East
Asia & Pacific (excluding high income)" "East Asia & Pacific (excluding high
income)" "East Asia & Pacific (excluding high income)" ...
##  $ malnut   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ year     : num  2011 2010 2009 2008 2007 ...
##  $ iso3c    : chr  "EAP" "EAP" "EAP" "EAP" ...
##  $ region   : chr  "Aggregates" "Aggregates" "Aggregates" "Aggregates" ...
##  $ capital  : chr  NA NA NA NA ...
##  $ longitude: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ latitude : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ income   : chr  "Aggregates" "Aggregates" "Aggregates" "Aggregates" ...
##  $ lending  : chr  "Aggregates" "Aggregates" "Aggregates" "Aggregates" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   iso2c = col_character(),
##   ..   country = col_character(),
##   ..   malnut = col_double(),
##   ..   year = col_double(),
##   ..   iso3c = col_character(),
##   ..   region = col_character(),
##   ..   capital = col_character(),
##   ..   longitude = col_double(),
##   ..   latitude = col_double(),
##   ..   income = col_character(),
##   ..   lending = col_character()
##   .. )

#As we can see, this is a data frame containing 1078 observations and 11
variables, where "malnut" is the main variable that we are interested in.
```

#Tidy data

To work with tidyverse packages, we need to use a data frame in the long (tidy) format, where we put variables in the columns and observations in the rows. Many databases will be in the wide format, so it is important to make sure that you are working with a dataset in the long format.

An examples of wide format data is:

```
##      ozone      wind     temp
## 1 23.61538 11.622581 65.54839
## 2 29.44444 10.266667 79.10000
## 3 59.11538  8.941935 83.90323
## 4 59.96154  8.793548 83.96774
```

The same data in the long format would be:

```
##    variable     value
## 1     ozone 23.615385
## 2     ozone 29.444444
## 3     ozone 59.115385
## 4     ozone 59.961538
## 5      wind 11.622581
## 6      wind 10.266667
## 7      wind  8.941935
## 8      wind  8.793548
## 9      temp 65.548387
## 10     temp 79.100000
## 11     temp 83.903226
## 12     temp 83.967742
```

There are many ways to transform databases from wide to long and long to wide formats. You can use the dplyr package functions gather() for transforming from wide to long and spread() to transform from long to wide. Another option (which I find easier) is to use the reshape2 package functions melt() and dcast().

Since our data is already in the long (tidy) format, we will move on. But see the supplemental material if you would like to explore this further.

##Dplyr basic data wrangling functions

There are six main functions the we can use to do the majority of data manipulations:

- **`filter()`**: pick observations by their values
- **`select()`**: pick variables by their names
- **`mutate()`**: create new variables with functions of existing variables
- **`group_by()`**: changes the scope of each function from operating on the entire dataset to operating on it group-by-group
- **`summarise()`**: collapse many values down to a single summary
- **`arrange()`**: reorder the rows

The first argument of any function is the data frame and the subsequent arguments describe what to do with the data frame.

#Filter

This function allows you to subset your data for specific countries, years, etc.

First, lets filter only data from Madagascar:

```
filter(dat, country %in% "Madagascar")
```

```
## # A tibble: 7 x 11
##   iso2c country malnut  year iso3c region capital longitude latitude
income
##   <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl> <chr>
## 1 MG    Madaga…  0.333  2011 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 2 MG    Madaga…  0.667  2010 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 3 MG    Madaga…  0.667  2009 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 4 MG    Madaga…  1      2008 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 5 MG    Madaga…  0.667  2007 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 6 MG    Madaga…  1      2006 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 7 MG    Madaga…  1      2005 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## # … with 1 more variable: lending <chr>
```

```
#now we could take the mean Malnutrition proportion from 2005 to 2011
x = filter(dat, country %in% "Madagascar")
mean(x$malnut)
```

```
## [1] 0.7619057
```

Lets filter data only from Madagascar and Mozambique. For this we will have to use the %in% operator:

```
filter(dat, country %in% c("Madagascar", "Mozambique"))
```

```
## # A tibble: 14 x 11
##    iso2c country malnut  year iso3c region capital longitude latitude
income
##    <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl>
<chr>
##  1 MG    Madaga…  0.333  2011 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  2 MG    Madaga…  0.667  2010 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  3 MG    Madaga…  0.667  2009 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  4 MG    Madaga…  1      2008 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  5 MG    Madaga…  0.667  2007 MDG   Sub-S… Antana…      45.7    -20.5 Low
```

```
i…
##  6 MG    Madaga…  1      2006 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  7 MG    Madaga…  1      2005 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
##  8 MZ    Mozamb…  1      2011 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
##  9 MZ    Mozamb…  0.667  2010 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## 10 MZ    Mozamb…  1      2009 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## 11 MZ    Mozamb…  1      2008 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## 12 MZ    Mozamb…  1      2007 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## 13 MZ    Mozamb…  1      2006 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## 14 MZ    Mozamb…  1      2005 MOZ   Sub-S… Maputo       32.6    -26.0 Low
i…
## # … with 1 more variable: lending <chr>
```

Now, lets filter for Madagascar and for values greater than 0.8

```
filter(dat, country == "Madagascar", malnut>0.8)
```

```
## # A tibble: 3 x 11
##   iso2c country malnut  year iso3c region capital longitude latitude
income
##   <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl> <chr>
## 1 MG    Madaga…      1  2008 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 2 MG    Madaga…      1  2006 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 3 MG    Madaga…      1  2005 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## # … with 1 more variable: lending <chr>
```

#Select

This funciton allows you to subset the data on variables or columns.

```
#Lets subset the data so that it shows only columns iso3c, year and malnut
x = select(dat, iso3c, year, malnut)
head(x)
```

```
## # A tibble: 6 x 3
##   iso3c  year malnut
##   <chr> <dbl>  <dbl>
## 1 EAP    2011     NA
## 2 EAP    2010     NA
## 3 EAP    2009     NA
```

```
## 4 EAP    2008     NA
## 5 EAP    2007     NA
## 6 EAP    2006     NA

#See that our dataset has now only 3 variables

#Now, lets remove longitude and latitude (we might not be interested in those
columns)
x = select(dat, -longitude, -latitude)
head(x)

## # A tibble: 6 x 9
##   iso2c country              malnut  year iso3c region  capital income
lending
##   <chr> <chr>                 <dbl> <dbl> <chr> <chr>   <chr>   <chr>
<chr>
## 1 4E    East Asia & Pacific…     NA  2011 EAP   Aggreg… <NA>    Aggreg…
Aggrega…
## 2 4E    East Asia & Pacific…     NA  2010 EAP   Aggreg… <NA>    Aggreg…
Aggrega…
## 3 4E    East Asia & Pacific…     NA  2009 EAP   Aggreg… <NA>    Aggreg…
Aggrega…
## 4 4E    East Asia & Pacific…     NA  2008 EAP   Aggreg… <NA>    Aggreg…
Aggrega…
## 5 4E    East Asia & Pacific…     NA  2007 EAP   Aggreg… <NA>    Aggreg…
Aggrega…
## 6 4E    East Asia & Pacific…     NA  2006 EAP   Aggreg… <NA>    Aggreg…
Aggrega…

#See now that we have 9 columns (instead of 11)
```

#Using the pipe %>% operator

This new syntax leads to code that is much easier to write and to read. You can think "and then" whenever you see the pipe operator, %>% The RStudio keyboard shortcut: Ctrl + Shift + M (Windows), Cmd + Shift + M (Mac).

```
#Lets do the same example as above. Now instead of including the data frame
inside the function, we will use the pipe operator:

#Instead of doing this:
x = filter(dat, country == "Madagascar", malnut>0.8)

#You can now do this:
x = dat %>%
  filter(country == "Madagascar",
         malnut>0.8)
head(x)

## # A tibble: 3 x 11
##   iso2c country malnut  year iso3c region capital longitude latitude
```

```
income
##   <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl> <chr>
## 1 MG    Madaga…      1  2008 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 2 MG    Madaga…      1  2006 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 3 MG    Madaga…      1  2005 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## # … with 1 more variable: lending <chr>
```

Now, you can combine two different functions:

```
dat %>%
  filter(country == "Madagascar") %>%
  select(iso3c, year, malnut)

## # A tibble: 7 x 3
##   iso3c  year malnut
##   <chr> <dbl>  <dbl>
## 1 MDG    2011  0.333
## 2 MDG    2010  0.667
## 3 MDG    2009  0.667
## 4 MDG    2008  1
## 5 MDG    2007  0.667
## 6 MDG    2006  1
## 7 MDG    2005  1

##This is an important, since most manipulations will have more than one
operations.
#You can read this as: "from the data frame "dat", filter for Madagascar and
then select variables iso3c, year and malnut"
```

#Mutate

Now lets say we want to add a new column to our data frame. This function allows you to change existing variables or create new ones.

```
#Lets say we want a column that is the percentage of child malnutrition
instead of proportions. To do that, we need to multiply the malnut variable
by 100. Let call this new variable perc_malnut

dat %>%
  mutate(perc_malnut = malnut*100) %>%
  filter(country == "Madagascar")

## # A tibble: 7 x 12
##   iso2c country malnut  year iso3c region capital longitude latitude
income
##   <chr> <chr>    <dbl> <dbl> <chr> <chr>  <chr>       <dbl>    <dbl> <chr>
## 1 MG    Madaga…  0.333  2011 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
```

```
## 2 MG    Madaga…  0.667  2010 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 3 MG    Madaga…  0.667  2009 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 4 MG    Madaga…  1      2008 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 5 MG    Madaga…  0.667  2007 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 6 MG    Madaga…  1      2006 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## 7 MG    Madaga…  1      2005 MDG   Sub-S… Antana…      45.7    -20.5 Low
i…
## # … with 2 more variables: lending <chr>, perc_malnut <dbl>

#Now we have one more column that is perc_malnut
```

#Group by and summarise What if we wanted to know the average "malnut" value per year for all countries? Answering this question requires a grouping variable.

By using `group_by()` we can set our grouping variable to year and create a new column called `mean_malnut` that will calculate the mean malnut value for each year accross all countries in our dataset.

The function `summarize()` will only keep the columns that are grouped_by or summarized. You can also use this to sum accross rows by using `sum()` instead of `mean()`

```
dat %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(malnut, na.rm = T)) #you need to include na.rm
= TRUE when you have NA's in the data.

## # A tibble: 7 x 2
##    year mean_malnut
##   <dbl>       <dbl>
## 1  2005       0.559
## 2  2006       0.562
## 3  2007       0.548
## 4  2008       0.543
## 5  2009       0.565
## 6  2010       0.584
## 7  2011       0.589

#Now we can see the world average child malnutrition estimates per year!
```

We can also group by more than one variable. Lets say we want to know thw average malnut value per year by income level.

```
dat %>%
  group_by(year, income) %>%
  summarise(mean_malnut = mean(malnut, na.rm = T)) #you need to include na.rm
= TRUE when you have NA's in the data.
```

```
## # A tibble: 35 x 3
## # Groups:   year [7]
##     year income              mean_malnut
##    <dbl> <chr>                     <dbl>
##  1  2005 Aggregates              NaN
##  2  2005 High income               0.267
##  3  2005 Low income                0.667
##  4  2005 Lower middle income       0.609
##  5  2005 Upper middle income       0.512
##  6  2006 Aggregates              NaN
##  7  2006 High income               0.300
##  8  2006 Low income                0.655
##  9  2006 Lower middle income       0.616
## 10  2006 Upper middle income       0.515
## # … with 25 more rows

#Now we can see the world average child malnutrition estimates per year!
```

#Arrange

This funciton allows you to arrange the data according to any specific variable. For example, you might want to arrange the output from the previous example by the mean_malnut value, instead of years.

```
dat %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(malnut, na.rm = T)) %>%
  arrange(mean_malnut)
```

```
## # A tibble: 7 x 2
##    year mean_malnut
##   <dbl>       <dbl>
## 1  2008       0.543
## 2  2007       0.548
## 3  2005       0.559
## 4  2006       0.562
## 5  2009       0.565
## 6  2010       0.584
## 7  2011       0.589
```

You can also arrange the data in descending order:

```
dat %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(malnut, na.rm = T)) %>%
  arrange(desc(mean_malnut))
```

```
## # A tibble: 7 x 2
##    year mean_malnut
##   <dbl>       <dbl>
## 1  2011       0.589
```

```
## 2  2010       0.584
## 3  2009       0.565
## 4  2006       0.562
## 5  2005       0.559
## 6  2007       0.548
## 7  2008       0.543
```

Lets now use all functions together!

Your turn: lets say we want to know the average malnut value (in percentage) in Sub-Saharan Africa countries from 2005 to 2011 in ascending order.

```
dat %>%
  filter(region == "Sub-Saharan Africa") %>%
  mutate(perc_malnut = 100*malnut) %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(perc_malnut, na.rm = T)) %>%
  arrange(mean_malnut)
```

```
## # A tibble: 7 x 2
##    year mean_malnut
##   <dbl>       <dbl>
## 1  2008        57.4
## 2  2007        58.9
## 3  2009        60.3
## 4  2006        60.3
## 5  2005        63.1
## 6  2010        63.8
## 7  2011        67.4
```

Plotting with `ggplot2`

`ggplot2` is a plotting package that makes it simple to produce high quality figures from data frames. To use **`ggplot2`**, you will need the data frame in the "long" format.

Lets start by doing a simple plot of the average child malnutrition estimates for Sub-Saharan Africa countries (from the last exercise)

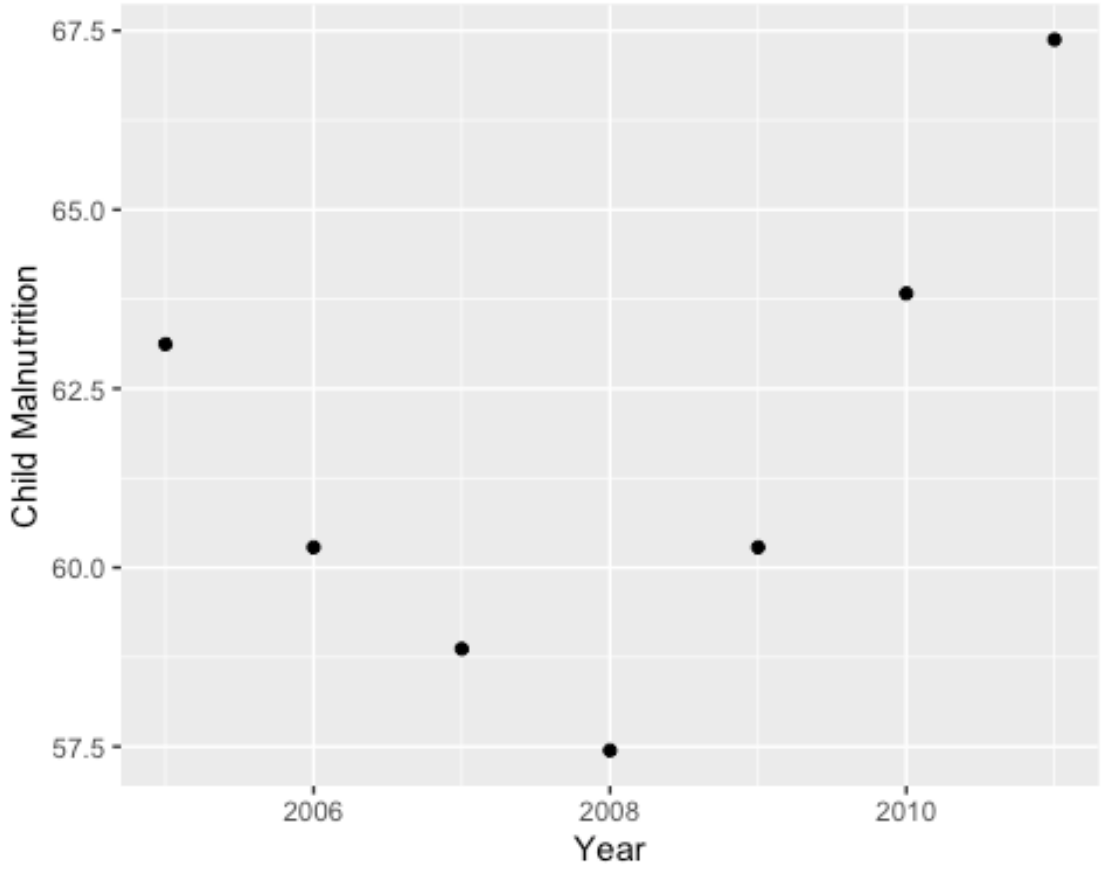
To build a ggplot, we need to:

- use the `ggplot()` function and specify the data frame using the `data` argument
- add `geoms` – graphical representation of the data in the plot (points, lines, bars). Examples include: * `geom_point()` for scatter plots, dot plots, etc. * `geom_bar()` for bar charts * `geom_line()` for trend lines, time-series, etc.
  * `geom_boxplot()` for boxplot charts

To add a geom to the plot use + operator and assign x and y aesthetics (`aes`)

```
#fist, lets create a dataframe with the data we want to plot. Lets call it
dat_SSA
dat_SSA = dat %>%
```

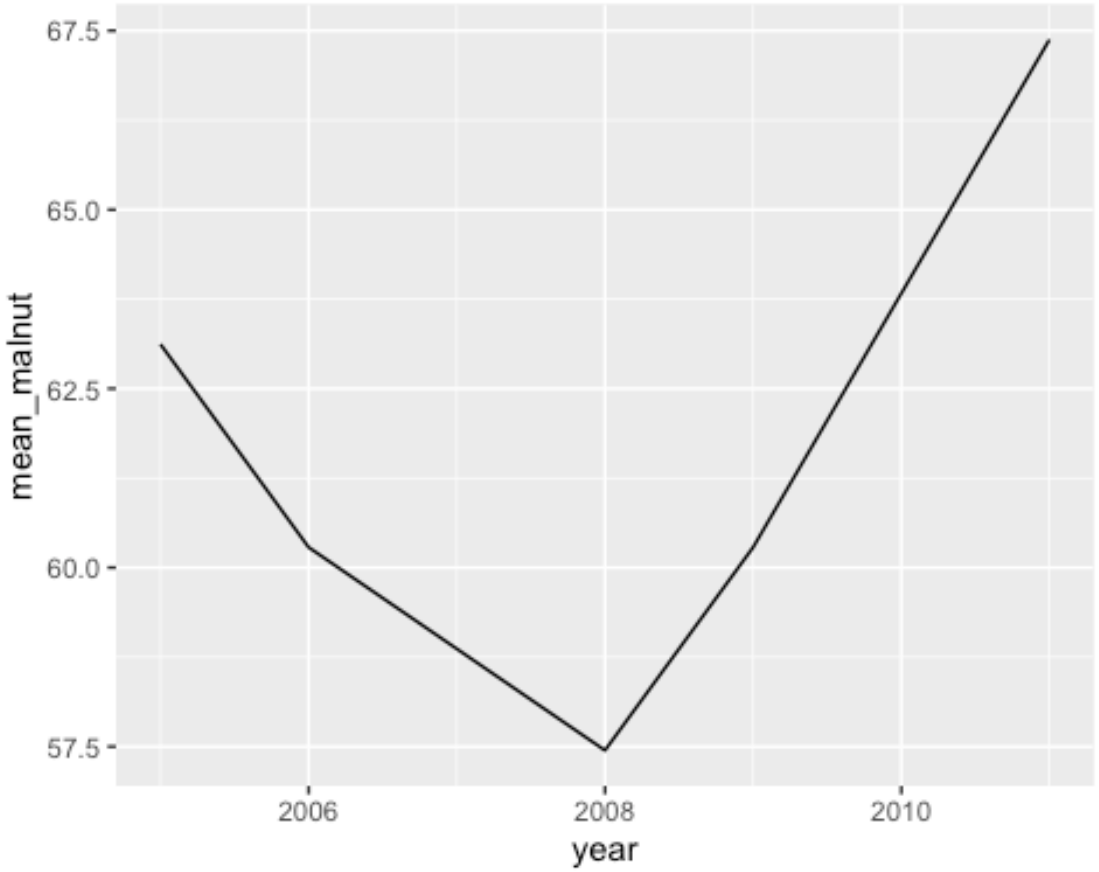
```
  filter(region == "Sub-Saharan Africa") %>%
  mutate(perc_malnut = 100*malnut) %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(perc_malnut, na.rm = T)) %>%
  arrange(mean_malnut)

#Now lets do a simple scater plot of the data
ggplot(data = dat_SSA) +
  geom_point(aes(x = year, y = mean_malnut)) +
  labs(x = "Year", y = "Child Malnutrition")
```



We can also plot this as a line:

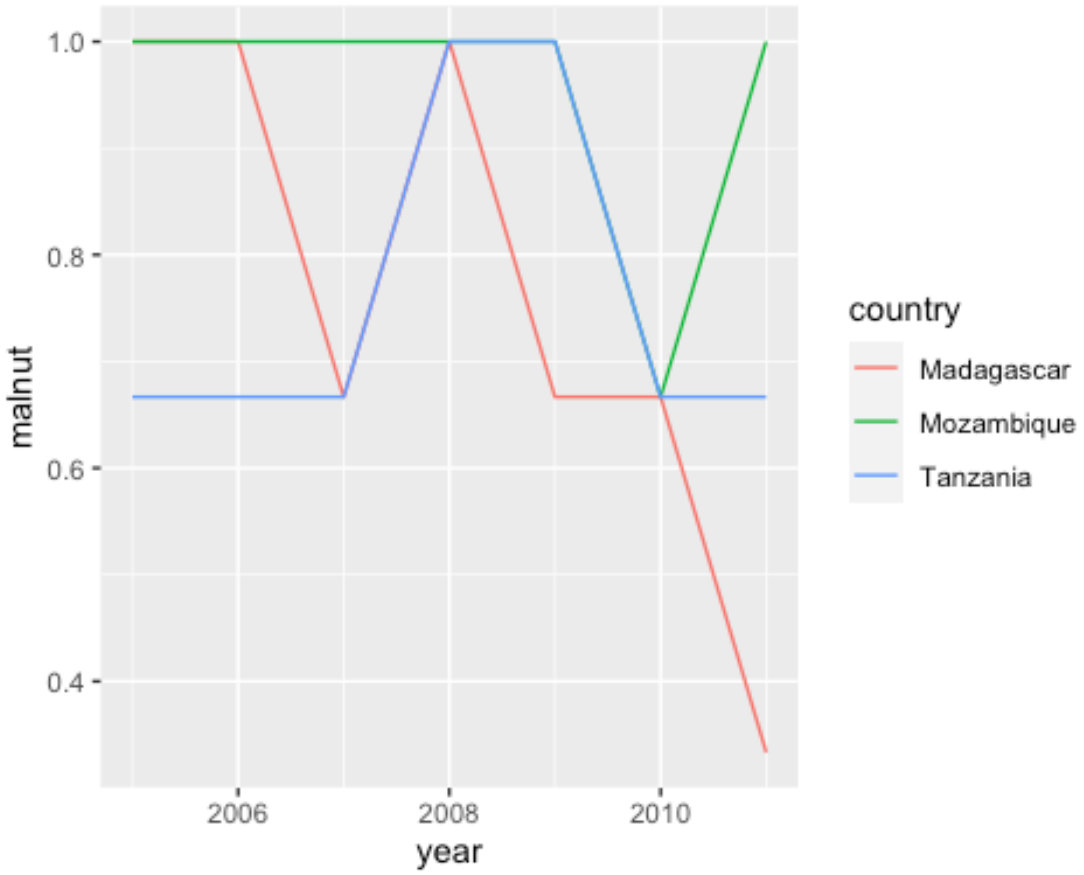
```
#fist, lets create a dataframe with the data we want to plot. Lets call it
dat_SSA
dat_SSA = dat %>%
  filter(region == "Sub-Saharan Africa") %>%
  mutate(perc_malnut = 100*malnut) %>%
  group_by(year) %>%
  summarise(mean_malnut = mean(perc_malnut, na.rm = T)) %>%
  arrange(mean_malnut)

#Now lets do a simple scater plot of the data
```

```r
ggplot(data = dat_SSA) +
  geom_line(aes(x = year, y = mean_malnut))
```



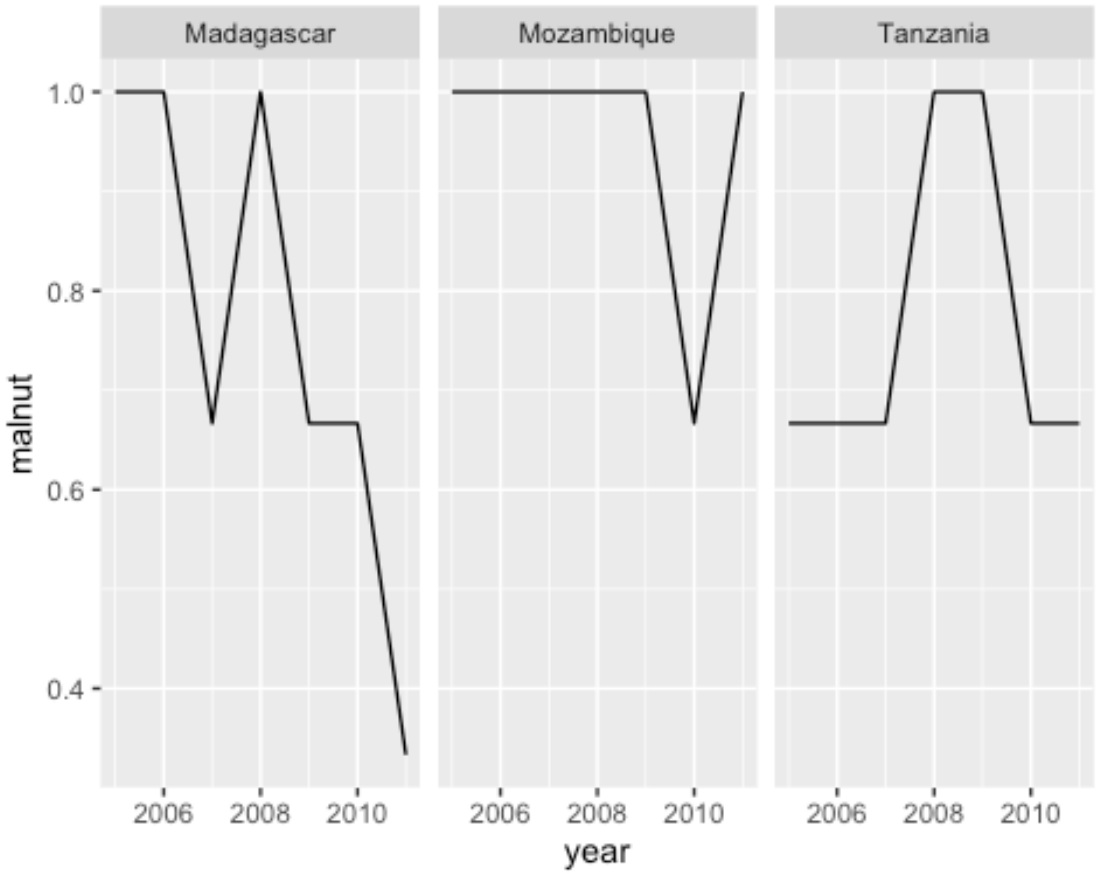
Now, lets say we are interested in lookng at trends from specific countries. We could have one line (or set of points for each country). Lets look at Madagascar, Mozambique and Tanzania.

```r
#fist, lets create a dataframe with the data we want to plot. Lets call it
dat_SSA
dat_SSA = dat %>%
  filter(country %in% c("Madagascar", "Mozambique", "Tanzania")) %>%
  mutate(perc_malnut = 100*malnut)

#Now lets do a simple scater plot of the data
ggplot(data = dat_SSA) +
  geom_line(aes(x = year, y = malnut, color = country))
```



Because the lines are overlapping, we can also plot each country in a seperate figure, using the *facet_wrap* function:

```
#fist, lets create a dataframe with the data we want to plot. Lets call it
dat_SSA
dat_SSA = dat %>%
  filter(country %in% c("Madagascar", "Mozambique", "Tanzania")) %>%
  mutate(perc_malnut = 100*malnut)

#Now lets do a simple scater plot of the data
ggplot(data = dat_SSA) +
  geom_line(aes(x = year, y = malnut))+
  facet_wrap(~ country)
```



Customizing plots

Take a look at the **ggplot2** cheat sheet, and think of ways you could improve the plot.

Now, let's change x and y axis labels and add a title to the figure. We can also remove the gray background using `theme`. `ggplot2` has many built in themes, such as `theme_bw()`, but you type `theme_` to see the many `theme` options. We can also use the `theme`

We can also change the size and color of the line:

```
#fist, lets create a dataframe with the data we want to plot. Lets call it
dat_SSA
dat_SSA = dat %>%
  filter(country %in% c("Madagascar", "Mozambique", "Tanzania")) %>%
  mutate(perc_malnut = 100*malnut)

#Now lets do a simple scater plot of the data
ggplot(data = dat_SSA) +
  geom_line(aes(x = year, y = malnut), size = 3, color = "red")+
  facet_wrap(~ country) +
  labs(x = "Year",
       y = "Child Malnutrition",
```

```
      title = "Prevalence of Child Malnutrition") +
 theme_bw()
```