International Food Security

Put your name here

Put due date here

## About the data

[Data.gov](https://www.data.gov) bills itself as "the home of the U.S. Government's open data." One of the many data sources publicly available from this site is a data set from The Economic Research Service (ERS) of the U.S. Department of Agriculture titled "International Food Security."

The data set was used to produce the ERS International Food Security Assessment 2013-2023 report and was released in June 2013. (Documentation for the most recent 2015-2025 report, which is still helpful for this past data set, can be found [here](http://www.ers.usda.gov/data-products/international-food-security/documentation.aspx)).

You can download the CSV data file from [this webpage](https://catalog.data.gov/dataset/international-food-security). The file name should be "gfa25.csv" when you download this file; keep this file name, or rename the file to "gfa25.csv" if it was downloaded with a different file name.

The food security data set has annual time series data for several variables (we'll focus on just a few, including GDP, food aid, and population). However, it's missing another variable we'd like: land area. The [World Bank Open Data](http://data.worldbank.org/) site has land area data (in square kilometers) available to download [here](http://data.worldbank.org/indicator/AG.LND.TOTL.K2?name_desc=false). Download this data, and rename the file with the country-specific data as "worldbank\_landarea.csv". Of the files included with the download from the World Bank site, this is the only file that is *not* metadata.

We'll read in and clean up both data sets separately before joining them together.

## Food security

To begin, read in and clean the International Food Security data set. The raw data looks like this:

## # A tibble: 3 × 6  
## Country Commodity Item Unit Year  
## <chr> <chr> <chr> <chr> <int>  
## 1 Algeria Total Grains/Cereals Area Harvested 1000 Ha 1980  
## 2 Algeria Total Grains/Cereals Yield MT / Ha 1980  
## 3 Algeria Total Grains/Cereals Production Quantity 1000 MT 1980  
## # ... with 1 more variables: Amount <dbl>

We are interested in data on each country's total food aid (Grain Equivalent, 1000 MT), the country's total population, and the country's gross domestic product (in 2005 dollars). Currently, these three variables are listed within the column "Item" (values "Total", "Total Population - Both Sexes", and "GDP (constant 2005 US$)"), with observed values in the "Amount" column.

We need to clean this data up to make it easier to use for plotting and analysis. First, filter the data, so that it only includes rows where the value of "Item" is one of the three variables of interest.

Next, the columns "Commodity" and "Unit" seem to be constant for a given value of "Item". Therefore, we'll make a separate dataframe called meta\_food that has columns for "Item", "Commodity", and "Unit". It should include only columns for "Item", "Commodity", and "Unite", and give only the unique rows. (*Hint*: The distinct function from dplyr is useful for this step.)

## # A tibble: 3 × 3  
## Item Commodity Unit  
## <chr> <chr> <chr>  
## 1 GDP (constant 2005 US$) Economic Data Million  
## 2 Total Population - Both Sexes Population Million  
## 3 Total Food Aid Grain Equiv. 1000 MT

Now that we have saved this information in a separate data set, we'll clean up the main dataframe (food). First, we remove the columns for "Commodity" and "Unit", then we convert the data set so that "Total", "Total Population - Both Sexes", and "GDP (constant 2005 US$)" are each separate columns, with the values given in "Amount" as the cell values. Finally, rename the variables "Country" and "Year" to be lowercase and use the column names "food\_aid" for "Total", "pop" for "Total Population - Both Sexes", and "gdp" for "GDP (constant 2005 US$)". After these changes, the first ten rows of the dataframe look like this:

## # A tibble: 10 × 5  
## country year gdp food\_aid pop  
## <chr> <int> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1980 NA NA 14.186  
## 2 Afghanistan 1981 NA NA 13.984  
## 3 Afghanistan 1982 NA NA 13.673  
## 4 Afghanistan 1983 NA NA 13.300  
## 5 Afghanistan 1984 NA NA 12.932  
## 6 Afghanistan 1985 NA NA 12.625  
## 7 Afghanistan 1986 NA NA 12.372  
## 8 Afghanistan 1987 NA NA 12.183  
## 9 Afghanistan 1988 NA 0.0000 12.157  
## 10 Afghanistan 1989 NA 378.2602 12.415

## Land area

Next, we can add another variable to our data set (land area) that exists in a separate data set. We'll download and read in the World Bank's land area data. Here are the column names of the raw data:

## [1] "Country Name" "Country Code" "Indicator Name" "Indicator Code"  
## [5] "1960" "1961" "1962" "1963"   
## [9] "1964" "1965" "1966" "1967"   
## [13] "1968" "1969" "1970" "1971"   
## [17] "1972" "1973" "1974" "1975"   
## [21] "1976" "1977" "1978" "1979"   
## [25] "1980" "1981" "1982" "1983"   
## [29] "1984" "1985" "1986" "1987"   
## [33] "1988" "1989" "1990" "1991"   
## [37] "1992" "1993" "1994" "1995"   
## [41] "1996" "1997" "1998" "1999"   
## [45] "2000" "2001" "2002" "2003"   
## [49] "2004" "2005" "2006" "2007"   
## [53] "2008" "2009" "2010" "2011"   
## [57] "2012" "2013" "2014" "2015"

We are not going to use the column named "Country Code", and the columns "Indicator Name" and "Indicator Code" are always the same (you can use unique to confirm this if you'd like). Therefore, we'll clean this data by removing these three columns. Next, we'll rename "Country Name" to "country". Next, a variable ("year") is currently in the column names ("1960", "1961", etc.). We will convert the dataframe to bring the year into a column named "year" and the observed value for each of these years into a column called "land\_area". Then, we'll filter out any rows with missing data for "land\_area". Finally, we'll change the class of "year" to integer class (check out as.integer for this step) and the class of "land\_area" to numeric. After this cleaning, the first ten rows of the dataframe look like this:

## # A tibble: 10 × 3  
## country year land\_area  
## <chr> <int> <dbl>  
## 1 Aruba 1961 180  
## 2 Andorra 1961 470  
## 3 Afghanistan 1961 652860  
## 4 Angola 1961 1246700  
## 5 Albania 1961 27400  
## 6 Arab World 1961 13624031  
## 7 United Arab Emirates 1961 83600  
## 8 Argentina 1961 2736690  
## 9 Armenia 1961 28470  
## 10 American Samoa 1961 200

## Join the dataframes

Now that both dataframes are cleaned up, we can join them together by country and year. Since we're only interested in countries that have both land\_area **and** food security data available, we'll do an inner\_join. The first six rows of our joined dataframe should look like this:

## # A tibble: 6 × 6  
## country year land\_area gdp food\_aid pop  
## <chr> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1980 652860 NA NA 14.186  
## 2 Angola 1980 1246700 NA NA 7.638  
## 3 Burundi 1980 25680 791.4031 NA 4.130  
## 4 Benin 1980 112760 1665.8932 NA 3.611  
## 5 Burkina Faso 1980 273600 1678.9617 NA 7.212  
## 6 Bangladesh 1980 130170 20089.2691 NA 80.624

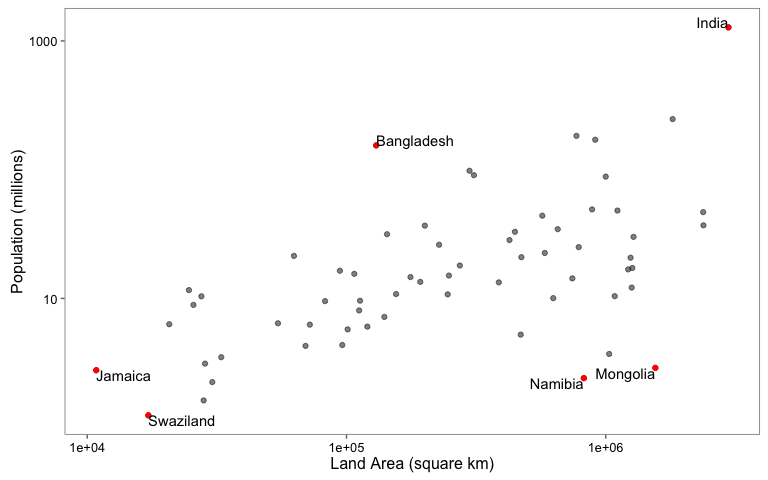
## Summary of GDP

The following table shows the average, minimum, and maximum values of GDP between 2000 and 2009 (inclusive) for each the ten countries with highest mean GDP over this time period. (*Hint:* Set na.rm = FALSE, or leave this as the default, in calculating all these functions, so the final table will only include countries with complete data from 2000 to 2009.)

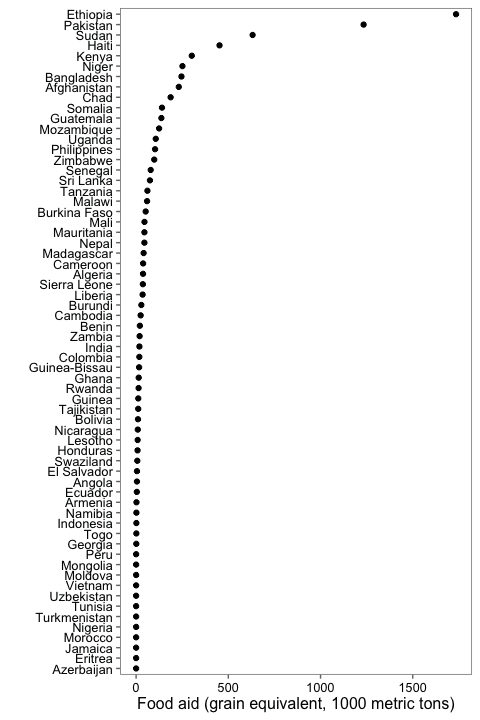
|  |  |  |  |
| --- | --- | --- | --- |
| Country | GDP (mean) | GDP (minimum) | GDP (maximum) |
| India | 827504.92 | 602653.71 | 1127948.41 |
| Indonesia | 283982.82 | 226918.06 | 355757.09 |
| Colombia | 146692.99 | 122659.08 | 175906.78 |
| Nigeria | 109889.17 | 83382.45 | 144926.44 |
| Pakistan | 105920.35 | 85822.30 | 127469.47 |
| Philippines | 101493.92 | 82353.67 | 121832.44 |
| Algeria | 97940.54 | 78895.34 | 112573.62 |
| Peru | 80597.54 | 64653.62 | 103147.79 |
| Bangladesh | 59866.49 | 46268.66 | 76809.88 |
| Morocco | 59370.35 | 46686.02 | 72868.90 |

## Visualization

The following plot shows the association between land area and population for the year 2013 for countries in this data set. Countries with unusual values (India, Bangladesh, Jamaica, Swaziland, Mongolia, and Namibia) are highlighted. (*Hint:* Use these country names to do the subsetting necessary to create this highlighting. To make sure the labels aren't directly over the points, use the vjust = "outward" and hjust = "inward" arguments.)



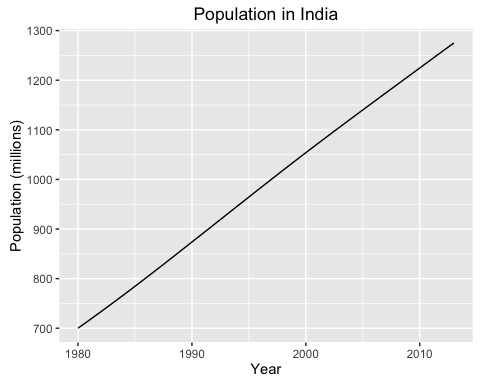
The next figure shows the total food aid received by the countries in this data set in 2010. Note that the points in this plot are arranging by total food aid in 2010. (*Hint*: Adjust the fig.height option in your code chunk so that the country labels are readable in the rendered figure.)



## Plotting function

Finally, we've written a function called plot\_pop that will plot population over time for a selected country. It has two arguments, datafr and which\_country. If you run the function, it will subset data for a single country from the dataframe specified with datafr and will plot year versus population for that country. For example, running plot\_pop(datafr = food\_land, which\_country = "India") would plot population over time in India:

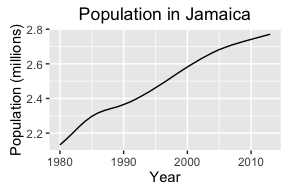
plot\_pop(datafr = food\_land, which\_country = "India")



(*Hint*: Look closely at the plot and make sure your function creates a plot that looks the same, in terms of elements like axis labels and the plot title. Also, you can assume that datafr, the dataframe input to the function, always has a column named country with countries and a column named pop with yearly population. You can also assume that the country name input for which\_country is always a country that has data in the dataframe, so you don't need to include any code for error checking.)

Here are a few more examples of running this function:

plot\_pop(datafr = food\_land, which\_country = "Jamaica")



plot\_pop(datafr = food\_land, which\_country = "Angola")

