Assignment 2: Linear and Logistic Regression
Introduction to Applied Data Science
2022-2023

Bas Machielsen
a.h.machielsen@uu.nl

April 2023

Assignment 2: Linear and Logistic Regression to Predict Poverty

1. Preparation

In this assignment, you will build your own linear and logistic regression models predicting poverty, using the gradient descent algorithms we talked about in the lecture. Similarly to the previous assignment, you will fill up the code chunks left empty in this document, and you will interpret them in this document. To start with, please replace my name and e-mail address with yours. Then, remove the lines:

```
output:
  pdf_document:
    includes:
      in_header: "preamble.tex"
    latex_engine: xelatex
```

from the document, and replace them with:

```
output: pdf_document
```

To start building statistical models that predict poverty, we need to proceed in the following way. We will predict poverty in a simple setting, and predict poverty on the basis of educational attainment and democracy scores. We will start from scratch, meaning that we ourselves will look for relevant data sources, put them together, and build some simple statistical models. Finally, we will check our solutions against standard benchmarks, and will attempt to interpret them.

2. Obtain the required poverty data

Next, we are going to make use of the `wbstats` package, which you have seen before. If you haven't done so already, you can install the package with `install.packages('wbstats')`. Make sure not to put this in your Rmarkdown document, as R will then attempt to install this package every time you knit your document.

The `wbstats` package allows you to navigate the World Bank database, and download datasets without having to visit the World Bank website, download the data into a spreadsheet, and subsequently load it into R. With the help of this package, we just download the data into R right away.

Question 1: Read this so-called vignette to find out how to navigate the World Bank data using the `wbstats` package. Download the variable that measures the "*Poverty headcount ratio at national poverty lines (% of population)*", the proportion of the population with daily per capita income below the national poverty line.

```
library(wbstats)
all_poverty_variables <- wbstats::wb_search("poverty")

View(all_poverty_variables)
```

Change `eval = FALSE` to `eval = TRUE` below once you've found the right answer.

```
poverty_data <- wb_data("SI.POV.NAHC")
```

Now, we will also load the `tidyverse` package, as we're going to do some data wrangling to glue various pieces of data together:

```
library(tidyverse)
```

Question x: Remove the `NA` observations from the dataset. Hint: you can use `na.omit`, but you have to `select` only a couple of columns.

```
poverty_data <- poverty_data %>%
  select(1:5) %>%
  na.omit()
```

Question 2: For which countries is this data available? Print a vector of the first 10 unique country names. Do not type a string variable of all the country names yourself, but use code to extract this.

```
poverty_data %>% select(country) %>% pull() %>% unique() %>% head(10)
```

```
##  [1] "Afghanistan" "Albania"     "Algeria"     "Angola"      "Argentina"
##  [6] "Armenia"     "Austria"     "Azerbaijan"  "Bangladesh"  "Belarus"
```

Next, save the *unique* `iso2c` and `iso3c` codes in two separate vectors:

```
iso2c <- unique(poverty_data %>% select(iso2c)) %>% pull()

iso3c <- unique(poverty_data %>% select(iso3c)) %>% pull()
```

For convenience, we might want to rename our poverty variable. One way to do this is as follows:

```
poverty_data <- poverty_data %>%
  rename(poverty = `SI.POV.NAHC`)
```

The `rename` function follows the `new = old` syntax, which you can verify by typing `?rename` in the console.

3. Obtain the required democracy data

Next, we will download datasets related to varieties of democracy. This data is contained in the Github package `democracyData`. The package is not available on the official R package repository CRAN, but it a custom package which is downloadable from somebody's github repository. To download packages from Github, we need the `devtools` package, and then, we can install the package using the `install_github()` function.

```
#install.packages("devtools")
devtools::install_github("xmarquez/democracyData")
```

After installing the package, we want to load a dataset from the package. To get a sense of what the data looks like, you can have a look here. The particular dataset we will use is called the `anckar` dataset, and contains data used in Anckar and Fredriksson (2018).

```
anckar <- democracyData::anckar
```

Have a look at the dataset:

```
head(anckar)
```

```
## # A tibble: 6 x 13
##   anckar~1 ancka~2 abbre~3  year democ~4 monar~5 regim~6 regim~7 popel~8 exten~9
##   <chr>       <dbl> <chr>   <dbl>   <dbl>   <dbl> <fct>   <fct>     <dbl> <chr>
## 1 UNITED ~       2 USA      1800       1       0 Presid~ Presid~       1 United~
## 2 UNITED ~       2 USA      1801       1       0 Presid~ Presid~       0 United~
## 3 UNITED ~       2 USA      1802       1       0 Presid~ Presid~       0 United~
## 4 UNITED ~       2 USA      1803       1       0 Presid~ Presid~       0 United~
## 5 UNITED ~       2 USA      1804       1       0 Presid~ Presid~       0 United~
## 6 UNITED ~       2 USA      1805       1       0 Presid~ Presid~       1 United~
## # ... with 3 more variables: GWn <dbl>, cown <int>, in_GW_system <lgl>, and
## #   abbreviated variable names 1: anckar_country, 2: anckar_ccode,
## #   3: abbreviation, 4: democracy, 5: monarchy, 6: regimebroadcat,
## #   7: regimenarrowcat, 8: popelection, 9: extended_country_name
```

The dataset contains a couple of indicators whether a country is a democracy at a given point in time as judged by the authors mentioned above. There is also some miscellaneous information. However, if we compare this dataset to the `poverty_data` dataset, we notice that some countries have different names. It will therefore be difficult to bring them together. Fortunately, the `democracyData` dataset contains a function called `country_year_coder`, which allows us to look up country code. As you may have noticed, we extracted

```
anckar <- democracyData::country_year_coder(anckar,
                                            country_col = anckar_country,
                                            date_col = year,
                                            code_col = anckar_ccode,
                                            include_in_output = "iso2c")
```

Now, we want to select the variables `iso2c` and `democracy`, and merge them to the `poverty_data` dataframe.

```
to_be_merged <- anckar %>% select(iso2c, year, democracy)
```

Question x: Use the `left_join` command to merge these two variables to the `poverty_data` dataframe. Read `?left_join` to correctly specify the `by` argument in this function.

```
data <- left_join(poverty_data, to_be_merged, by = c("iso2c" = "iso2c", "date" = "year"))
```

Question x: Find out what the median is of the poverty variable in the dataset you downloaded. Make a new dummy variable `povyesno`, meaning:

$$\text{povyesno}_i = \begin{cases} 1 & \text{if Poverty } > \text{median(Poverty)} \\ 0 & \text{otherwise} \end{cases}$$

You can easily do so using the `if_else` function from `dplyr`. Check `?if_else` for its syntax.

```
data <- data %>%
  mutate(povyesno = if_else(poverty > median(poverty, na.rm = TRUE), 1, 0))
```

4. Obtain the required educational data

Next, we will want to integrate information about educational attainment. Fortunately, the *World Bank* also collects many different kinds of data on educational attainment. For an impression, have a look at:

```
test <- wbstats::wb_search("educational attainment")
```

For this setting, we'll take the variable `SE.PRM.CUAT.ZS`, which is the fraction of the population aged 25 and over that at least complete primary education.

```
education <- wb_data('SE.PRM.CUAT.ZS')
```

Question x: Select the first five variables and then omit the `NA` observations from this dataset. Also rename the variable to `educ`.

```
education <- education %>% select(1:5) %>% na.omit() %>% rename(educ = `SE.PRM.CUAT.ZS`)
```

Question x: Now use `left_join` again to merge your already assembled `data` with the `education` data:

```
data <- left_join(data, education, by = c("iso2c" = "iso2c", "date" = "date"))
```

There are a couple of unnecessary variables in your dataset. They can be removed using the `select` function in the following way:

```
data <- data %>%
  select(-iso3c.x, -country.x, -iso3c.y, -country.y)
```

Now we are ready to train a statistical model.

5. Building a linear regression model

Now, we want to predict poverty by means of two factors: (i) historical educational investment, and (ii) democracy. We have just undertaken various efforts to collect all these data in the data.frame called `data`. Next, we'll proceed to estimate a statistical model to predict poverty. In particular, we will estimate the following model:

$$\text{poverty}_i = \alpha + \beta_1 \cdot \text{democracy}_i + \beta_2 \cdot \text{education}_i + \epsilon_i$$

This model estimates the coefficients α, β_1 and β_2. Do you think β_1 and β_2 will be greater than zero, smaller than zero, or zero, and why?

It turns out that this linear model can be solved using the gradient descent algorithm. This means that we are describing a loss function, randomly initialize parameters, and then compute the gradient, and move the parameters in the direction *opposite* the largest increase. Below is an implementation of such an algorithm. You do not have to change much, but you have to implement the gradients for β_1 and β_2 yourself.

6. Building a logistic regression model

Finally, we'll analyze the same data using a logistic regression model. A key element in a logistic regression model is the so-called *sigmoid* function. The sigmoid function $S(x)$ is defined as follows:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Question x: Implement the sigmoid function as a function in R

```r
sigmoid <- function(x){
  1/(1+exp(-x))
}
```

Question x: Differentiate the sigmoid function with respect to x. Rewrite this derivative to show that it is equal to $S(x)(1 - S(x))$ using the definition of $S(x)$ given above.

Question x: Implement the derivative of the sigmoid function in R.

```r
sigmoid_derivative <- function(x){

  sigmoid(x) * (1-sigmoid(x))

}
```

A nice thing about this sigmoid function is that its outcomes always fall between 0 and 1. We'll use the sigmoid function to build a logistic regression model. In particular, instead of a simple argument x, we are predicting the outcome, poverty, on the basis of the same two independent variables as before, and two corresponding coefficients. Our sigmoid function would thus look like this:

$$\text{Poverty}_i = \frac{1}{1 + e^{-[\alpha+\beta_1 \cdot \text{educ}_i+\beta_2 \cdot \text{stab}_i]}}$$

Question x: Use the chain rule from calculus to identify the derivatives of the above particular sigmoid function with respect to education$_i$ and stability$_i$ and α. Hint: use the result from above, and use that:

$$\frac{\partial S(.)}{\partial \text{education}_i} = \frac{\partial S(x)}{\partial x} \cdot \frac{\partial x}{\partial \text{education}_i}$$

In the next few steps, we want to implement these functions in our gradient descent algorithm. We can largely use the same setup as we used in the linear model, with the exception of the derivates.

Question x: fill the derivatives in this algorithm yourself.

```r
gradient_descent <- function(x, y,
                             learn_rate,
                             conv_threshold = 1e-10,
                             n = length(y),
                             max_iter = 5000) {
  alpha <- runif(1, 0, 1)
  beta_1 <- runif(1, 0, 1)
  beta_2 <- runif(1, 0, 1)

  yhat <- sigmoid(alpha + beta_1 * x[,1] + beta_2 * x[,2])
  MSE <- -(1 / n) * sum(y * log(yhat) + (1-y)* log(1-yhat))
  converged = F
  iterations = 0

  while(converged == F) {

    MSE <- -(1 / n) * sum(y * log(yhat) + (1-y)* log(1-yhat))
    alpha_new <- alpha - learn_rate * (1 / n) * (sum((yhat - y)))
```

```
    beta_1_new <- beta_1 - learn_rate * (1 / n) *
      (sum((yhat - y)*x[,1])) # fill in here
    beta_2_new <- beta_2 - learn_rate * (1 / n) *
      (sum((yhat - y)*x[,2])) # fill in here

    alpha <- alpha_new
    beta_1 <- beta_1_new
    beta_2 <- beta_2_new

    yhat <- sigmoid(alpha + beta_1 * x[,1] + beta_2 * x[,2])
    MSE_new <- -(1 / n) * sum(y * log(yhat) + (1-y)* log(1-yhat))

    if(abs(MSE - MSE_new) <= conv_threshold) {
      converged = T
      return(paste("Converged. Optimal intercept:", alpha,
                   "Optimal slope beta1:", beta_1,
                   "Optimal slope beta2:", beta_2))
    }

    iterations = iterations + 1
    if(iterations > max_iter) {
      converged = T
      return(paste("Optimal intercept:", alpha,
                   "Optimal slope beta1:", beta_1,
                   "Optimal slope beta2:", beta_2))
    }
  }
}
```

```
library(AER)
```

```
## Loading required package: car
## Loading required package: carData
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
## The following object is masked from 'package:purrr':
##
##     some
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
```

```
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: survival
```

```
data("GrowthDJ")

GrowthDJ <- GrowthDJ %>%
  na.omit() %>%
  mutate(gdp60 = gdp60/10000, popgrowth = popgrowth/10, growth = if_else(gdpgrowth > median(gdpgrowth, r

glm(growth ~ gdp60 + popgrowth, data = GrowthDJ) %>% summary()
```

```
##
## Call:
## glm(formula = growth ~ gdp60 + popgrowth, data = GrowthDJ)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.70020  -0.48068  -0.06625   0.39573   0.88058
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.12403    0.11513   1.077 0.284041
## gdp60       -0.16899    0.06126  -2.758 0.006943 **
## popgrowth    1.85055    0.48244   3.836 0.000223 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2177264)
##
##     Null deviance: 24.960  on 99  degrees of freedom
## Residual deviance: 21.119  on 97  degrees of freedom
## AIC: 136.29
##
## Number of Fisher Scoring iterations: 2
```

```
x = cbind(GrowthDJ$gdp60, GrowthDJ$popgrowth)
y = GrowthDJ$growth
```

```
glm(y ~ x, family="binomial") %>% summary()
```

```
##
## Call:
## glm(formula = y ~ x, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5740  -1.1284  -0.4742   0.9927   1.9966
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.7659     0.6613  -2.670  0.00758 **
## x1           -0.8603     0.4498  -1.912  0.05582 .
## x2            8.6005     2.6543   3.240  0.00119 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.47  on 99  degrees of freedom
## Residual deviance: 121.74  on 97  degrees of freedom
## AIC: 127.74
##
## Number of Fisher Scoring iterations: 4
```

Now, we will check the outcome of the gradient descent algorithm.

```
gradient_descent(x = x, y = y, 0.5, conv_threshold = 1e-10, max_iter = 50000)
```

```
## [1] "Converged. Optimal intercept: -1.76355229499617 Optimal slope beta1: -0.86012590220448 Optimal s
```

7. Evaluate your output and compare it to standard solutions

Both linear regression and logistic regression can also be performed using an analytical solution for the minimum loss. For linear regression, this function is implemented in R in the `lm` function. Check the syntax for the `lm` method using `?lm`.

Question x: Verify that the solution from the analytical model (`lm(y ~ x1 + x2, data = dataset)`) is identical to the solution you obtained when performing gradient descent.

Question x: you can also report a summary of the model estimated using the `modelsummary` package and function. Report a summary of the model below.

```
model1 <- lm(y ~ x1 + x2, data = dataset)

modelsummary::modelsummary(model1)
```

Finally, we also want to use what we've learned about poverty.

Question x: Give an interpretation of the model. What happens when a country becomes more democratic, and what would happen to poverty if a country had invested more in education?

Question x: Generate the predicted values from the model. Find the *highest* predicted value. For which country is this? And what is the predicted value? Use code to find all of this information.