Regression Analysis in R

1 Purpose

The purpose of this activity is to provide you with an understanding of regression analysis (for the purposes of improving your understanding of Problem Sets 3 and 4, the quantitative reading material, and/or your final essay) and to both develop and apply that knowledge to the use of the R statistical programming environment.

2 Overview

This lab can be completed during at home and everything you need to know is provided in this document.

3 Your Task

Using R as instructed, complete the following activities. If you are completely new to R and especially if you are completely new to quantitative data analysis, consider working through the Appendix to this document before proceeding. Before starting, you will need to install R:

1. Install R from: `https://cran.r-project.org/`
2. You may also find it more comfortable to work with R via RStudio, an "integrated development environment" for the software. You can download RStudio from: `https://www.rstudio.com/products/RStudio/`
3. If you don't use RStudio, you will need a text editor (e.g., Notepad, etc.) to store the analysis code that you write.

For this lab, we will use the UK portion of the European Social Survey (ESS) Round 7 data from 2014. You can download these data from here: `http://www.europeansocialsurvey.org/download.html?file=ESS7GB&c=GB&y=2014`. You may have to login with an email address and unzip the file. Save the file on your computer and unzip it.

3.1 Bivariate Regression

1. Open R. (You may want to use RStudio.)
2. Use `setwd()` to set your working directory to wherever you saved the file. For example, on Windows you might want to try something like `setwd("c:/users/thomas/desktop")`.
3. Start a new text document in the editor of your choice (again, could be in RStudio). Write all code that you generate in that file so you have a complete record of your analysis.
4. Read the data into R using:

```
# install a package to read in the data
install.packages("rio")

# load that package
library("rio")
d <- import("ESS7GB.dta")
```

5. How many observations are there in the dataset? How many variables?

6. Examine the `polintr` variable, which measures political interest. Tabulate the variable:

   ```
   table(d$polintr)
   ```

   Then, replace the values of "Refusal", "Don't know", and "No answer" as missing values:

   ```
   d$polintr[d$polintr %in% 5:7] <- NA
   ```

   Repeat the tabulation to confirm that this work. Using what you have learned already, attempt to answer the following questions:

   (a) What proportion respondents are "very interested" in politics?
   (b) What is the *modal* response category for this question?
   (c) What is the *median* level of interest?
   (d) Using the gender variable (`gndr`), assess who is more interested in politics: men or women?
   (e) Using a statistical test, assess whether the difference in political interest is statistically distinguishable from zero.

7. Now examine a different set of variables: happiness (`happy`) and religiosity (`rlgdgr`). First, remove the nonresponse categories from these variables:

   ```
   d$rlgdgr[d$rlgdgr %in% 12:14] <- NA
   d$happy[d$happy %in% 12:14] <- NA
   ```

8. How happy are people in Great Britain on average? How religious are they?

9. Draw a scatterplot of religiosity and happiness:

   ```
   library("ggplot2")
   ggplot(d, aes(x = as.integer(rlgdgr), y = as.integer(happy))) +
       geom_jitter(na.rm = TRUE)
   ```

   Looking at the plot, how correlated would you say these variables are? Does happiness go with religiosity?

10. Use `cov()` to calculate the covariance of these two variables and `cor()` to calculate the correlation coefficient for this relationship. Note: You will have to wrap the variable names in `as.integer()` (like in the above code) in order to do these calculations. Also, see `? cor` and read about the `use` argument to `cor()` and `cov()` in order to correctly calculate these statistics.

11. Recall that the correlation is simply a rescaled version of the covariation: it is the covariance of two variables divided by the product of their standard deviations. Calculate the standard deviation of each variable and — combined with the covariance you calculated above — use that to manually calculate the correlation coefficient for this relationship. Does it match the value returned by `cor()`?

12. Recall that the correlation coefficient says nothing about the "size" of the relationship between two variables. Instead, it captures the degree to which that relationship is well-represented by a straight-line. If the correlation is high (close to -1 or +1), we know that the data can be well-represented by a line but we do not know how much of an increase we would expect to see in one variable given an increase in the other variable. "Regression" is a method for representing the covariance between variables in another way, which provides us with exactly this way of describing the relationship.

13. In regression, we draw a line through the scatterplot of the two variables that represents the "line of best fit" and that we can summarize by the "slope" of the line: the expected change in the y-axis variable given a one-unit increase in the x-axis variable. Add the following line to your scatterplot from above to see this line of best fit:

```
+ geom_smooth(method = "lm")
```

14. To estimate the value of the slope, all we need to know is that the slope of the line is a rescaled version of the covariance of the two variables. Unlike the correlation coefficient, however, to estimate the slope, we rescale the covariance by the variance of the x-axis variable. To do this calculation, we have to be a little careful, so let's create a new data.frame to store our variables:

```
new <- na.omit(data.frame(religiosity = as.integer(d$rlgdgr),
                          happy = as.integer(d$happy)))
```

Note: We use `na.omit()` here to remove missing data from these two variables. How many observations did we lose by removing these missing values?

15. Now, using the `new` data.frame, calculate the covariance of the two variables. Then, calculate the variance of `religiosity`. Divide the covariance by the variance of religiosity. This is the slope of the regression line. How much happier is someone if they were to increase their religiosity by one point on this question scale? Does this correspond with the relationship in your scatterplot; can you see the slope of the line matches this calculated slope?

16. We do not have to manually calculate the slope in this way. Indeed, R provides a function called `lm()` that allows us to calculate this directly. In R, run the following to estimate this regression model and store the results as an object called `m`. View the results.

```
m <- lm(happy ~ religiosity, data = new)
m
```

Does the slope match what you calculated by hand?

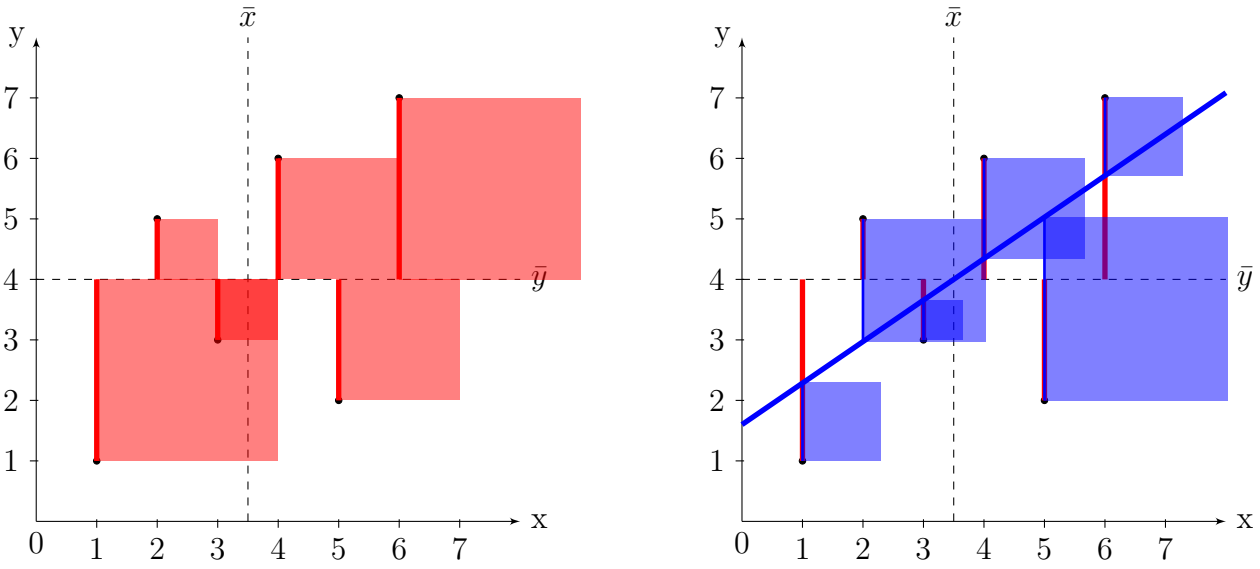
17. Another way of thinking about regression is in terms of the "sum of squares." We can think of the "sum of squares" as the total amount of variation in the y variable (happiness). You can calculate the total sum of squares for our data using:

```
sum((new$happy - mean(new$happy))^2)
```

This quantity (the total sum of squares) is literally a sum of the area of several squares. Figure 1(a) shows this for a simple example scatterplot. The total sum of squares is simply the sum of the individual deviation of each observed value of y from the mean of y, squared.

18. In regression, we are "explaining" some of that total variation in y with the covariation between x (religiosity) and y (happiness). This explained portion of the sum of squares is called the "sum of squares, explained" or "model sum of squares"; the amount of variation in y that is not "explained" by x is called the "residual sum of squares". This residual sum of squares is literally the sum of the squared areas formed by the distance between each point and the "predicted value" (or "fitted value") of y expected by the regression line given that observation's value of x. Figure 1(b) shows the residual sum of squares for the same example scatterplot.

19. If all of the variation in y is explained by the variation in x, then the residual sum of squares will be *small* while the explained sum of squares will be large. You can calculate the residual sum of squares using the following:



(a) Total Sum of Squares

(b) Residual Sum of Squares

Figure 1: Sum of Squares for a Simple Scatterplot

```
sum((new$happy - (coef(m)[1] + (coef(m)[2] * new$religiosity)))^2)
```

How large is the residual sum of squares compared to the total sum of squares you calculated above?

20. A common way to assess the "goodness of fit" of a regression estimation is whether the explained sum of squares is large. The proportion of the total sum of squares that is "explained" by the model is called the "R-squared" statistic, sometimes called the "coefficient of determination." It is simply 1 minus the ratio of the residual sum of squares to the total sum of squares. It is interpreted as a proportion: if it is 0, then religiosity explains *none* of the variation in happiness; if it is 1, then religiosity explains *all* of the variation in happiness. What is the R-squared for this relationship?

21. Recall from earlier that the regression slope is simply a scaled version of the covariance of the two variables. The correlation coefficient is also a scaled version of the covariance. Now, you can also note that there is a direct correspondence between the correlation coefficient and the R-squared value you just calculated. R-squared is simply the bivariate correlation coefficient, squared. Try it and confirm this is the case.

22. This focus on the residuals allows us to demonstrate a few R programming features that have useful practical applications:

    (a) `m$fitted` will return a vector containing the "fitted value" of y for each observation (given its value of x)

    (b) `m$residuals` will return a vector containing the residuals for each observation in your dataset. These residuals will be equal to `new$happy - m$fitted`

    (c) The `predict()` function allows you to calculate the fitted value of y for a given value of x. By default, it simply returns the same thing as doing `all.equal(predict(m), m$fitted)`. But, if you supply a data.frame as the value of the `newdata` argument, you can obtained the fitted value for an arbitrary value of x: `predict(m, newdata = data.frame(religiosity = 3))`

23. Returning to our main focus, another common measure of model fit is called the residual mean squared error, or sometimes simply $\hat{\sigma}$. This is simply the standard deviation of the residuals (the

unexplained variation in y). As religiosity explains more and more of the variation in happiness, then the residuals will become small and have a lower variance (and lower standard deviation).[1] There are three ways to calculate sigma: (a) by manually calculating the residuals and taking the standard deviation thereof, (b) extracting the residuals from the `m` object (returned by `lm()`), or (c) using R's calculation of sigma:

```
sqrt(sum((new$happy - (coef(m)[1] + (coef(m)[2] * new$religiosity)))^2)/nrow(new))
sd(residuals(m))
```

The reason this quantity is useful as a measure of model fit is that it is on the scale of the y-axis variable (happiness) and can be directly compared to the standard deviation of y. If religiosity explained a lot of the variation in happiness, then $\hat{\sigma}$ would be small compared to `sd(new$happy)`. Is it?

24. Returning to our estimated regression model, `m`, use `summary(m)` to see a more complete print out of the regression results. Can you make sense of all of the information that is displayed? What are the values of the following quantities:

    (a) Regression slope
    (b) $\hat{\sigma}$
    (c) R-squared
    (d) Mean residual
    (e) y-intercept (the value of happiness when religiosity = 0)

25. You will note that the table includes a number of other statistics that we haven't discussed, including the standard errors of the regression slope and y-intercept, and t-statistics and p-values for each of these. We are not going to cover the calculation of the standard errors, but can you see how the t-statistics are calculated given what we covered last week?

26. Is the estimated regression slope substantively large? Is the slope significantly different from zero?

27. In order to ensure your understanding, you may want to redo some of these exercises using other combinations of variables (e.g., political interest, etc.). One particularly illustrative example would be to use gender as an x-axis variable. (You will want to redefine `gender` as:

    ```
    d$gender <- as.integer(d$gndr)-1
    ```

    In that case, you will hopefully note some similarities between the conditional mean of an outcome variable given gender:

    ```
    aggregate(as.integer(polintr) ~ gender, data = d, FUN = mean, na.rm = TRUE)
    ```

    and the results of a corresponding regression analysis:

    ```
    lm(as.integer(polintr) ~ gender, data = d)
    ```

[1] Recall the standard deviation is simply the square root of the variance.

28. Recall that the p-value in a statistical significance test represents the probability of observing a t-statistic as large or larger than the one observed in a sample of the data from a population where the true t-statistic is zero (in this case, where the slope of the relationship between religiosity and happiness is non-existent).

    As an **advanced exercise**, you emulate that process of repeated sampling through a process known as "bootstrapping". In bootstrapping, you treat your sample data as if they were a population and draw repeated samples (with replacement) from your sample and apply an estimator (in this case, a regression estimator) to each resampled dataset. In essence, bootstrapping creates an approximate sampling distribution for your statistic.

    If you extract the estimated slope from each bootstrap sample, this bootstrap sampling distribution mimics the hypothetical distribution of slopes you would expect to observe in a population where the true slope was equal to that observed in your actual sample data. The proportion of slopes less than zero in this bootstrapped distribution can be interpreted as the p-value (the probability of observing a slope as large as the one you observed) from a population where the true slope was zero.

    Here's the code:

```
boot <- replicate(5000L,
          lm(happy ~ religiosity,
             data = new[sample(1:nrow(new), nrow(new), TRUE), ])$coef[2])
summary(boot)
sum(boot < 0)/length(boot) # p-value
```

    What is p-value of the regression slope, based on the bootstrap sampling distribution?

3.2 Multivariate Regression

1. Think again about political interest. How correlated are interest, happiness, gender, and religiosity? If they are correlated, can we interpret those correlations as causal relationships? What criteria for causal inference to we have to satisfy in order to interpret this correlation as a causal relationship?

2. One major barrier to causal inference here is confounding: the relationship between these two factors may be explained by some other variable or variables. We might be concerned that there is some earlier variable, that precedes both religiosity and (current) happiness in time that causes both variables, and in turn makes it appear that the two variables are related due to a causal relationship between religiosity and happiness.

    To think about this possibility, it can be useful to draw a "causal diagram" in the form of a "directed, acyclic graph" (DAG). Figure 2 shows an example DAG. We can imagine that religiosity is **X** and happiness is **Y**. We are concerned bout variables like **Z** and **W** that cause both. Draw a possible DAG, labeling possible confounding variables. (You might also want to include other variables in your graph, like **A** and **E** that cause happiness but that you do not think cause religiosity.)

3. Once you have a reasonable DAG showing possibly causal relationships, think about whether you can observe any of the variables in your DAG. Are all of the potentially confounding variables you named available in the ESS dataset? You can find a complete listing of variables in: `attributes(d)$var.labels` or in the complete documentation of the ESS questionnaire here: `http://www.europeansocialsurvey.org/docs/round7/fieldwork/united_kingdom/ESS7_questionn GB.pdf`.

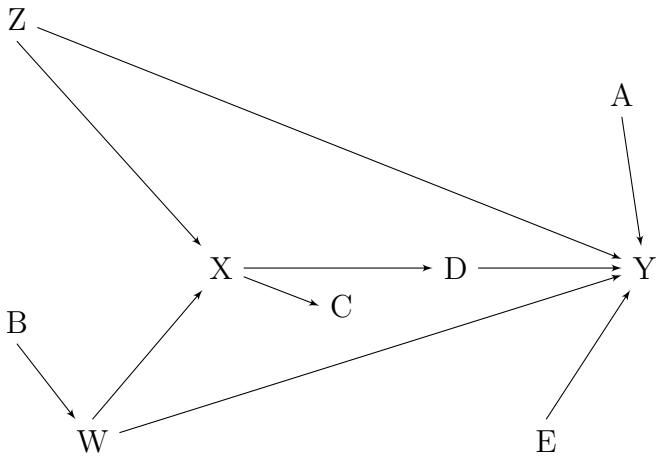


Figure 2: A Causal Graph

4. If you have access to potentially confounding variables, then we can use an elaboration of the bivariate regression models we just discussed to help to control for this confounding. If potentially confounding variables in your DAG are unobserved (either because they are not included in the ESS or because they are perhaps "unobservable" factors), there is nothing we can do to rule out the potential for confounding. We are, in that situation, completely unable to draw a causal inference from the observational data in front of us. We would need more data!

5. How can multiple regression help us to address confounding? The mathematics here is a bit complicated, but the following exercises are meant to provide some intuition. Before we begin, make sure you have a firm grasp on the variables you have identified. Use `summary()` or other functions to know the key statistics of each variable and `ggplot2` to produce scatterplots or other graphics comparing these variables to one another.

6. Multiple regression works by simultaneously accounting for the influence of multiple "right hand side" (RHS) variables on a single "outcome" variable (**Y** in the example DAG in Figure 2). How does it do this? Let's start with a simple example (which you can expand using the variables in your DAG). In the below model we are trying to see whether there is still an association between religiosity and happiness once we "control for" interest in politics:

```
new2 <- na.omit(data.frame(happy = as.integer(d$happy),
                  religiosity = as.integer(d$rlgdgr),
                  pinterest = as.integer(d$polintr),
                  gender = as.integer(d$gndr)))
m2 <- lm(happy ~ religiosity + pinterest, data = new2)
```

   Basically, this model says that we think happiness might be a function of religiosity, political interest, and gender, and we want to know how much each contributes to happiness. By including all variable in one regression equation, we estimate the influence of each variable on the outcome accounting for the simultaneous influence of the other RHS variables. Is there still a relationship between religiosity and happiness once we account for these other factors?

7. Try estimating bivariate relationships between each of these RHS variables and the outcome. How do the estimated slopes in the bivariate models compare to the estimated slopes in the multivariate specification?

```
p1 <- lm(happy ~ religiosity, data = new2)
p2 <- lm(happy ~ pinterest, data = new2)
```

8. When we regress an outcome on a variable, we aim to *break* the correlation between the residuals (the unexplained parts of y) and the RHS variable. TO see this, look at the correlations between the residuals from each of the bivariate models and the RHS variable we included:

```
cor(p1$residuals, new2$religiosity)
cor(p2$residuals, new2$pinterest)
```

This correlation is basically zero in each case.

9. But when we exclude a RHS variable (leaving it out of the regression equation), the residuals of y might still be associated with that variable:

```
round(cor(cbind(p1$residuals, new2$religiosity, new2$pinterest)), 3)
round(cor(cbind(p2$residuals, new2$religiosity, new2$pinterest)), 3)
```

10. We can see that the excluded variables appear to still be associated with the outcome (i.e., there is some residual some of squares that is still associated with the excluded RHS variables). If we include these, as in the multivariate model `m2`, we can account for the independent influences of each of these factors. But how does this work? To understand multiple regression we have to understand a simpler concept: the *partial correlation*.

    A partial correlation is the correlation between two variables controlling for the influence of a third variable. If we want to know the partial correlation between `happy` and `religiosity`, controlling for `pinterest`, we regress each variable on `pinterest` and correlate their residuals:

```
r1 <- lm(happy ~ pinterest, data = new2)$residuals
r2 <- lm(religiosity ~ pinterest, data = new2)$residuals
cor(r1, r2)
```

This removes the common influence of the third variable from the original correlation:

```
cor(new2$happy, new2$religiosity)
```

which in this case is small.

11. Multiple regression is simply a series of partial correlations. If we want to know the slope of the relationship between religiosity and happiness, we simply regress the residuals of the regression of `happy` on `pintrest` on the residuals from the regression of `religiosity` on `pinterest`:

```
lm(r1 ~ r2)
# or:
lm( lm(happy ~ pinterest, data = new2)$residuals ~
    lm(religiosity ~ pinterest, data = new2)$residuals )$coef
```

Similarly, if we want to know the influence of political interest controlling for religiosity, we do something analogous:

```
lm( lm(happy ~ religiosity, data = new2)$residuals ~
    lm(pinterest ~ religiosity, data = new2)$residuals )$coef
```

Note how the slope for `pinterest` and the slope for `religiosity` in the previous two partial correlation models are identical to the slopes from a multivariate regression of `happy` on both `pinterest` and `religiosity`:

```
lm(happy ~ pinterest + religiosity, data = new2)$coef
```

All a multiple regression model is doing is engaging a series of simultaneous regressions of the outcome on each RHS variable and each RHS variable on all the other RHS variables.

12. Everything that we learned above about bivariate regressions — residuals, fitted/predicted values, measures of model fit, statistical significance, etc. — applies in a multiple regression context. Explore some of these aspects of the multivariate regression models we have just estimated.

13. The only thing that is different about bivariate and multivariate regression is the substantive interpretation of estimated regression slopes. Rather than talk about these slopes as the "influence of X on Y" we instead interpret them as the "influence of X on Y controlling for Z" or the "influence of X on Y all else constant". This is a subtle change but it's important to remember that in a multivariate regression we are not estimating a *raw* relationship but rather than relationship accounting for the variance in X and Y explained by the other RHS variables in the model. For this reason, the slopes in a multivariate regression are sometimes called "marginal effects" due to their representing the change of influence of one variable while the other variables remain constant.

3.3 Survey-Weighted Analyses

14. Now that you have a sense of how to estimate and interpret regression models, we need to handle a very important final caveat to all of the above. When we work with survey data, we need to acknowledge that such data are often *weighted*. That is, they do not represent a simple random sample from a population but instead come from a more complex survey design and the *representativeness* of the sample data hinges on the estimates being re-weighted to make them match the target population.

15. To do this, we need to install and use a package specifically designed to work with survey data, called (logically) "survey". Install and load it.

16. We then need to tell R that our data are supposed to be survey-weighted by creating a "survey design" object using `svydesign()`:

```
library("survey")
# weighted version of data
weighted <- svydesign(id = ~ 0, weights = ~ pspwght, data = d)
# unweighted version of data (for examples below)
unweighted <- svydesign(id = ~ 0, weights = ~ 0, data = d)
```

We then need to use survey-specific analogues to all of our previous functions and use our new `weighted` object rather than `d`. A simple example of this is as follows:

```
# calculate unweighted means
svymean(~ happy + polintr + rlgdgr, design = unweighted)

# calculate weighted means
svymean(~ happy + polintr + rlgdgr, design = weighted)
```

How do the unweighted and weighted means differ? (The unweighted means should be the same as those found by `colMeans(d[c("happy", "polintr", "rlgdgr")])` on our original data.)

17. The difference is that in the unweighted analysis, every row in the dataset counts equally whereas in the weighted analysis, some rows count for more than others. Individuals that are underrepresented in the sample relative to the population are given weights larger than 1 and individuals that are overrepresented in the sample relative to the population are given weights smaller than 1. You might want to examine the distribution of weights using some of the following:

```
ggplot(d, aes(x = pspwght)) + geom_histogram()
summary(d$pspwght)

aggregate(pspwght ~ gndr, data = d, FUN = mean, na.rm = TRUE)
aggregate(pspwght ~ region, data = d, FUN = mean, na.rm = TRUE)
aggregate(pspwght ~ yrbrn, data = d, FUN = mean, na.rm = TRUE)
```

18. For regression analysis, we need to use `svyglm()` rather than `lm()`:

```
# unweighted lm() behaves as normal
summary(lm(happy ~ polintr + rlgdgr, data = d))

# unweighted svyglm() gives same results
summary(svyglm(happy ~ polintr + rlgdgr, design = unweighted))

# survey-weighted model
summary(svyglm(happy ~ polintr + rlgdgr, design = weighted))
```

The last sets of results should differ substantively from either of the former two.

4 Appendix: Introduction to Quantitative Data Analysis

This appendix covers a variety of basic issues in quantitative data analysis through R, thus also providing an introductory course in R programming. It does this through a heavy focus on data visualization.

1. Open the R console or R studio. Install the "gapminder" package, which contains a simple dataset of country-level variables, using `install.packages("gapminder")`. Assuming you encountered no issues, the "gapminder" package is now installed on your machine but not loaded.

2. Load the gapminder package using `library("gapminder")` . The package is now loaded and attached, so that you can use the data therein. Type `? gapminder` to open the documentation of the dataset.

3. Using the documentation and the Rconsole, answer the following:

   - How many variables are contained in the dataset? Confirm this in R using `ncol(gapminder)` .
   - How many "observations" or cases are contained in the dataset? Confirm this in R using `nrow(gapminder)` .
   - What is the "unit of analysis" of these data? Is an observation a country, a year, a country-year, or something else?

4. Use some R functions to obtain glances at the data.

```
> head(gapminder) # first few rows of the data
      country continent year lifeExp      pop gdpPercap
1 Afghanistan      Asia 1952  28.801  8425333  779.4453
2 Afghanistan      Asia 1957  30.332  9240934  820.8530
3 Afghanistan      Asia 1962  31.997 10267083  853.1007
4 Afghanistan      Asia 1967  34.020 11537966  836.1971
5 Afghanistan      Asia 1972  36.088 13079460  739.9811
6 Afghanistan      Asia 1977  38.438 14880372  786.1134

> str(gapminder) # a detailed summary of the 'str'ucture of the data
Classes tbl_df, tbl and 'data.frame':       1704 obs. of  6 variables:
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
 $ pop      : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 138679
 $ gdpPercap: num  779 821 853 836 740 ...

> summary(gapminder) # a quantitative summary of the data
        country        continent        year         lifeExp           pop
 Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60   Min.   :6.001e+04   M
 Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20   1st Qu.:2.794e+06   1
 Algeria    :  12   Asia    :396   Median :1980   Median :60.71   Median :7.024e+06   M
 Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47   Mean   :2.960e+07   M
 Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85   3rd Qu.:1.959e+07   3
 Australia  :  12                  Max.   :2007   Max.   :82.60   Max.   :1.319e+09   M
 (Other)    :1632
```

5. How many countries are represented in the dataset? There are many ways to answer this. Two possibilities are:

   - `length(table(gapminder[["country"]]))`
   - `length(unique(gapminder[["country"]]))`

   You should get 142.

6. In the above code, what is the `table()` function doing? Using the help documentation `? table` to find out.

7. In the above code, what is the `length()` function doing? Using the help documentation `? length` to find out.

8. How many *years* are represented in the dataset? Modify the above code examples to find the answer. The correct answer is 12.

9. What is the life expectancy at birth in years for Algeria in 1967? You can find this out visually by printing the entire dataset by just typing `gapminder` at the console, or by typing `fix(gapminder)` to view the data in a spreadsheet format.

   But if we want to find the answer *computationally*, we need to *subset* the data. We already did some subsetting above with `gapminder[["country"]]`, which uses `[[` bracket notation to extract one variable from the dataset. There are other ways to extract one variable from a dataset:

   (a) `gapminder[["country"]]`

   (b) `gapminder$country`

   (c) `gapminder[, "country"]`

   All of these are (basically) equivalent. We can also subset so that we only see certain rows of a dataset. To do that, we'll use the third form of subsetting to find out the answer to our question about life expectancy in Algeria in 1967. We'll do this in several steps to see what is going on but only the last step is needed to find the answer:

   (a) Extract only data for Algeria:
   `gapminder[gapminder[["country"]] == "Algeria", ]`

   (b) Extract only the `lifeExp` variable:
   `gapminder[, "lifeExp"]`

   (c) Extract all life expectancy data for Algeria:
   `gapminder[gapminder[["country"]] == "Algeria", "lifeExp", drop = FALSE]` [2]

   (d) Subset data to only Algeria for only 1967:
   `gapminder[gapminder[["country"]] == "Algeria" & gapminder[["year"]] == 1967, ]`

   (e) Extract life expectancy data for Algeria for only 1967:
   `gapminder[gapminder[["country"]] == "Algeria" & gapminder[["year"]] == 1967, "lifeExp"]`

[2]Note: the use of `drop = FALSE` means that our code returns a "data frame" rather than a "vector." If you set `drop = TRUE` (or just leave that part of the code out), you will get a vector of data instead.

You should find that the answer is 51.407.

10. Now modify the above code to find the GDP per capita for Somalia in 1992. The correct answer should be 926.9603.

11. Now let's calculate some *statistics* about these data. For example, to find the mean (average) life expectancy across all countries in 1992, we can extract the vector of life expectancy data for that year:
    `gapminder[gapminder[["year"]] == 1992, "lifeExp"]`
    Then we can use the `mean()` function to take the mean of this vector:
    `mean(gapminder[gapminder[["year"]] == 1992, "lifeExp"])`
    The answer should be 64.16034.

12. R provides many other functions to calculate basic statistics. Try the following functions on the life expectancy variable and see what you find:
    - Median: `median()`
    - Variance: `variance()`
    - Standard deviation: `sd()`

13. Another useful way of understanding *categorical* or *ordinal* data is to tabulate it. This means to count how many observations in the data take a particular value of the categorical variable. For example, to see how many countries there are in the dataset that exist on each continent, we can use:

```
> table(gapminder[gapminder[["year"]] == 1992, "continent"])

  Africa Americas     Asia   Europe  Oceania
      52       25       33       30        2
```

    Note how we are using the `table()` function on a subset of data. Test your knowledge: how many European countries are represented in the data in 1967?

14. The previous exercise demonstrated a "one-way" tabulation. But we may also want to produce "two-way" or "multi-way" tabulations, which we will typically call a *crosstabulation* or *crosstab* for short. To do that, we will use the `ftable()` function. For example, to see how many cases we have on each continent for each year, we can do:

    `ftable(gapminder[["year"]], gapminder[["continent"]])`

    Crosstabs are only useful for categorical data. Why is this? What happens when you try to tabulate a numerical/interval variable:
    `ftable(gapminder[["year"]], gapminder[["lifeExp"]])`

15. Tabulations only provide us with one statistic: a *count* (or "subgroup total"). If we want to describe the data in other ways — for example to know what the mean life expectancy was in each year — we need to transform our data through an *aggregation*. An aggregation is:

a new dataset created from the data where observations (rows) are combined by a function applied to subsets of the data defined by a "grouping factor" such that the new dataset has one observation per group.

For example, a simple aggregation would create a new dataset by applying a count function (`length()` in R notation) where country is the grouping factor:
`aggregate(. ~ country, data = gapminder, length)`

This code uses the `aggregate()` function to produce an aggregation. The `. ~country` portion of the code indicates that we want to aggregate every variable in our dataset (indicated by `.`) by `country` for the dataset `gapminder` using the aggregating function `length`. If we replace `.` with a variable name, we will only aggregate that variable as opposed to every variable in the dataset.

Modifying this code, answer our original question: what is the mean life expectancy (across all countries) by year? You should find, among other things, that the mean life expectancy in 2002 was 65.69492.

16. We can generalize this code further by aggregating not just by one variable but by many. For example, if we wanted to know the mean life expectancy by year *by continent*, we could say:
`aggregate(lifeExp ~ continent + year, data = gapminder, mean)`

    Using that information, can you confirm that average life expectancy in African countries in 2007 was 54.80604 years?

17. Now modify the above code to create an aggregated dataset that expresses the variability of life expectancy across countries by continent and year. Which continent had the most between-country variability in life expectancies in 2007? What about in 1952?

18. Now let's try graphing. R provides several different graphing libraries. Among the easiest to use is "ggplot2". You will need to install it using `install.packages("ggplot2")` and load it using `library("ggplot2")`. The article you read for this week by Hadley Wickham describes the theory or philosophy underlying the package and there is a useful cheatsheet available on Moodle. The package is very powerful, so we will only learn the basics at this point.

19. Every graph made with ggplot2 starts with the `ggplot()` function. This function expresses *what* data we want to visualize and then we add graphing elements to describe *how* we want to visualize those data.

    To begin, you can specify that we want to visualize the life expectancy variable:
    `ggplot(gapminder, aes(x = lifeExp))`

    You will get a plot but it will be blank because we haven't specified how we want those data visualized. For example, if we just want to see the general distribution of the data, we request a histogram:

    ```
    ggplot(gapminder, aes(x = lifeExp)) + geom_histogram()
    ```

    What is the most prevalent life expectancy? What is the general shape of the distribution? Now try drawing a histogram for the `gdpPercap` variable. How does this distribution compare to the last one?

20. ggplot2 is most useful for visualizing multivariate relationships. For example, if we want to

visualize the relationship between GDP per capita and life expectancy for all country–years, we can do:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) + geom_point()
```

What is the general shape of the relationship? Are the two variables closely related (correlated)? Calculate a Pearson's correlation coefficient for the relationship using:
`cor(gapminder[["gdpPercap"]], gapminder[["lifeExp"]])`

Recalling that the correlation coefficient ranges from -1 (perfectly negatively correlated) to 0 (linearly unrelated) to +1 (perfectly positively correlated), how strong is the relationship?

21. A common procedure in describing data is to transform it. Data that are highly *skewed*, as is GDP per capita, are often "log transformed" or "logged" by applying the logarithm function to the data in order to make them more easily interpretable. Try logging the GDP per capita variable and seeing how this changes the relationship between it and life expectancy:

```
ggplot(gapminder, aes(x = log(gdpPercap))) + geom_histogram()
ggplot(gapminder, aes(x = log(gdpPercap), y = lifeExp)) + geom_point()
cor(log(gapminder[["gdpPercap"]]), gapminder[["lifeExp"]])
```

How does your understanding of the relationship between the variables change?

22. Now let's try to look at even more complex relationships. Recall above that calculated mean life expectancy by country. Let's try to plot those results so that they are more easily understandable. Draw a scatterplot showing the relationship between year and life expectancy:

```
ggplot(gapminder, aes(x = year, y = lifeExp)) + geom_point()
```

How does the relationship here compare to the results we saw above from aggregating the data? One thing that can help clarify this relationship is to use a different "geom" to visualize the data. Try:

```
ggplot(gapminder, aes(x = year, y = lifeExp)) + geom_smooth()
```

What's the trend in life expectancy over time? Now practice a little more sophisticated ggplot2 skill by combining the scatterplot and smoothed trend line:

```
ggplot(gapminder, aes(x = year, y = lifeExp)) +
  geom_point() + geom_smooth()}
```

23. This visualization is helpful for describing a time trend, but we haven't yet expressed the between-continent variation that was perhaps the most interesting part of work so far. To do that, we have to instruct the `ggplot()` function to account for the continent variable. To do that, let's add a grouping factor:

```
ggplot(gapminder, aes(x = year, y = lifeExp, color = continent)) +
  geom_point() + geom_smooth()
```

What's going on here? How would you describe or characterize the continent-specific time trends? How much has life expectancy changed over the period of our data? Is the trend the same in all continents?

24. If you look closely at this last visualization, you should see that it is displaying an aggregation. The `geom_smooth()` component of our visualization code is aggregating our data by continent and year and then visualizing that pattern. To check that this is in fact what is happening, create an aggregation of this relationship and then visualize it.

    In case you're not sure how to do that, try the following:

    ```
    newdat <- aggregate(lifeExp ~ continent + year, data = gapminder, mean)
    ggplot(newdat, aes(x = year, y = lifeExp)) + geom_line(aes(color = continent))
    ```

    How do the two graphs compare?[3]

25. One of the interesting things about the data so far has been that the variance of life expectancy across countries within a continent seemed to change over time. To get a visual sense of the distribution of the data, let's use a boxplot (or "box and whiskers") visualization:

    ```
    ggplot(gapminder, aes(x = factor(year), y = lifeExp)) + geom_boxplot()
    ```

    This graph shows the "five number summary" (minimum, first quartile, median, third quartile, and maximum) of the data. Compare the visualization against a tabular representation of the same:

    ```
    aggregate(lifeExp ~ year, data = gapminder, fivenum)
    ```

    Do they match?

26. This interesting part of that relationship, though, was between-continent differences. To show them, we can try a couple of different things. one is to group the results by continent using colour:

    ```
    ggplot(gapminder, aes(x = factor(year), y = lifeExp)) +
        geom_boxplot(aes(color = continent))
    ```

    That's kind of ugly and unreadable though. Colour here is just like an axis, we're adding a dimension to the visualization captured by colour. Rather than using colour, we could show that dimension in another way. For example, we could swap the use of year and continent:

    ```
    ggplot(gapminder, aes(x = continent, y = lifeExp)) +
        geom_boxplot(aes(color = factor(year)))
    ```

    In that visualization, we see the separate trends for each continent, with year treated as a colour. This is better because it shows the continent-specific trends more clearly and it also shows how much between-country variation there is in Africa and Asia and how little there is in Asia.

    What's going on there? Why is the variation in Oceania so small? Create an subset of just the data by extracting just the observations for countries in Oceania. How many are there and which countries are they?

27. The last visualization used colour as a grouping factor. Another approach is to do "faceting" where several smaller graphs are made, and each group represents a subset of the data. To do that, we will add a `facet_wrap()` feature to the visualization:

[3]Note: If you want to compare graphs, it can be useful to save them to your computer's hard drive. To do this, plot the graph, then use the `ggsave()` function and specify a filename, like `ggsave("graph1.png")`. Note that graphs will be saved, by default, in your working directory, which you can identify by typing `getwd()` at the console

```
ggplot(gapminder, aes(x = factor(year), y = lifeExp)) +
    geom_boxplot() + facet_wrap(~continent)
```

If you don't like the look, you can customize it, for example, the following might be better:

```
ggplot(gapminder, aes(x = factor(year), y = lifeExp)) +
    geom_boxplot() + facet_wrap(~continent, nrow = 1)
```

Substantively, this is the same data as shown above but instead of using colours to represent year or continent, we are using facets. In this way, it should be clear that an axis, a colour, and a facet are at the most basic level just an axis — that is, a way to add dimensionality to a visualization.

28. We can add even more dimensions when we draw scatterplots. Let's go back to the relationship between GDP per capita and life expectancy. We can colour the graph to show continent-specific data:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point()
```

But we could also show that dimensionality using facets:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() + facet_wrap(~continent)
```

Or we could convey years as facets:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() + facet_wrap(~year)
```

Or see all of our data by facetting by year and continent:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() + facet_wrap(~year+continent)
```

That basically becomes unreadable, though. So we might be better of displaying one of those dimensions through colour and maybe log transform for clarity:

```
ggplot(gapminder, aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point(aes(colour = continent)) + facet_wrap(~year)
```

We can add further dimensions, such as population size, as well. Now not only axes, colour, and facet are used as dimensions, but so is the size of the points representing each country:

```
ggplot(gapminder, aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point(aes(colour = continent, size = pop)) +
  facet_wrap(~year)
```

What are the large outlier countries in Asia?

29. Finally, it is worth noting that ggplot2 visualizations are highly customizable. One of the simplest ways to customize them is to use "themes". These change the colours and other features of graphs in ways that may be attractive. To show how they work, we will save our graph as an object called `g` and then change the theme associated with it to see various options:

```
g <- ggplot(gapminder, aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point(aes(colour = continent, size = pop)) +
  facet_wrap(~year)

g + theme_bw()
g + theme_gray()
g + theme_minimal()
```

See `? theme_bw` for options. There is an another package called "ggthemes" that adds additional options. Install and load it to use these themes, such as:

```
library("ggthemes")
g + theme_economist()
g + theme_solarized(light = FALSE)
```

It is also possible to modify other features such as:

- Title, using `ggtitle()`
- x-axis and y-axis labels, using `xlab()` and `ylab()`
- x-axis and y-axis labels and dimensions, using `scale_x_continuous()`, `scale_x_continuous()`, etc.

See the documentation and the ggplot2 cheatsheet for guidance on these modifications.