Using data from Round 9 of the 2018 European Social Survey, the exercises below will show you how to perform one-sample significance tests in R.

To download the data, visit the ESS Website and navigate to the *Data and Documentation* menu and select **ESS Data Portal** from the drop-down menu. You will now be taken to a new page; click on ESS round 9 (2018). On the new page, scroll down to **Data Files** and click on the *Download* button next to the **ESS9 - integrated file, edition 3.1** heading. You will then be taken to a new page which will require you to provide your registration details. After providing this information, you will then be able to select the download format.

Download the data in **Stata** format.

1. Preparing and Exploring Data

For this practical, you will need the following packages: `haven` and `tidyverse`.

```
library(tidyverse)
library(haven)
```

We import the the ESS9 data file into an object called `ess9`.

To facilitate your work in this course unit, remember to set up a folder for this unit and create an R project or set your working directory prior to beginning the practicals. It is recommended that you download and place the data for the practicals in a separate sub-folder as I have done here (my data file is in a folder called `data`.)

```
ess9 <- read_dta("data/ESS9e03_1.dta")
```

We find that the dataset has 49519 observations of 572 variables; as you'll remember, we can find this information by looking in our Environment tab or using functions such as `dim()`. To obtain a summary view of the data object, `glimpse()` is not recommended in this case because the dataset is very large so a more suitable compact view can be achieved with the `head()` function.

```
head(ess9)
```

```
## # A tibble: 6 x 572
##   name    essround edition proddate  idno cntry dweight pspwght pweight anweight
##   <chr>      <dbl> <chr>   <chr>    <dbl> <chr>   <dbl>   <dbl>   <dbl>    <dbl>
## 1 ESS9e0~        9 3.1     17.02.2~    27 AT      0.581   0.218   0.302   0.0659
## 2 ESS9e0~        9 3.1     17.02.2~   137 AT      1.06    0.413   0.302   0.125
## 3 ESS9e0~        9 3.1     17.02.2~   194 AT      1.38    2.27    0.302   0.686
## 4 ESS9e0~        9 3.1     17.02.2~   208 AT      0.993   0.386   0.302   0.117
## 5 ESS9e0~        9 3.1     17.02.2~   220 AT      0.377   1.03    0.302   0.312
## 6 ESS9e0~        9 3.1     17.02.2~   254 AT      1.48    0.576   0.302   0.174
## # i 562 more variables: prob <dbl>, stratum <dbl>, psu <dbl>, nwspol <dbl+lbl>,
## #   netusoft <dbl+lbl>, netustm <dbl+lbl>, ppltrst <dbl+lbl>,
## #   pplfair <dbl+lbl>, pplhlp <dbl+lbl>, polintr <dbl+lbl>, psppsgva <dbl+lbl>,
## #   actrolga <dbl+lbl>, psppipla <dbl+lbl>, cptppola <dbl+lbl>,
## #   trstprl <dbl+lbl>, trstlgl <dbl+lbl>, trstplc <dbl+lbl>, trstplt <dbl+lbl>,
## #   trstprt <dbl+lbl>, trstep <dbl+lbl>, trstun <dbl+lbl>, vote <dbl+lbl>,
## #   prtvtcat <dbl+lbl>, prtvtdbe <dbl+lbl>, prtvtdbg <dbl+lbl>, ...
```

Let's say we are interested in the `trstprl` variable. We can use the `attributes()` function to find out that it measures degree of trust in parliament on an ordinal scale (0, or not trust at all, to 10, or complete trust). We also find out that the "Refusal", "Don't know", and "No answer" categories are denoted as `NA`. Remember that understanding your variables is crucial prior to analysis to ensure that you apply the correct statistical tests and draw appropriate conclusions.

```
attributes(ess9$trstprl)
```

```
## $label
## [1] "Trust in country's parliament"
##
## $format.stata
## [1] "%4.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
## No trust at all               1               2               3               4
##               0               1               2               3               4
##               5               6               7               8               9
##               5               6               7               8               9
##  Complete trust         Refusal      Don't know      No answer
##              10              NA              NA              NA
```

It is also important to find out the number of observations in each category when dealing with this type of variables, and the number of many missing values (if any). One way to do this is to use the `count()` function. There are two principal reasons why this is important. Firstly, you must determine

if there are a sufficient number of observations and/or whether missing values predominate in comparison to the total sample size of the survey. If missing values predominate and only a small number of observations is available for the categories, then that might indicate that the variable may not be representative. Also, missing values will affect the results of descriptive statistics functions, and so they must be removed when performing the calculations.

In this particular example, we can see that by comparison, the number of missing values is small and each category of the variable has a large number of observations.

```
count(ess9, trstprl)
```

```
## # A tibble: 12 x 2
##    trstprl                     n
##    <dbl+lbl>               <int>
##  1     0 [No trust at all]  5414
##  2     1 [1]                2561
##  3     2 [2]                3937
##  4     3 [3]                5121
##  5     4 [4]                4737
##  6     5 [5]                8461
##  7     6 [6]                5632
##  8     7 [7]                5830
##  9     8 [8]                4128
## 10     9 [9]                1396
## 11    10 [Complete trust]   1158
## 12 NA(b) [Don't know]       1144
```

To learn more about the `trstprl` variable, we can use the `summarise()` function to find out the total number of observations, mean, median, and standard deviation. We have a large sample size (over 48,000 observations!). We can also see that the median and the mean are not very different, but are not equal either. This is an expected phenomenon for ordinal variables.

```
ess9 %>%
  drop_na(trstprl) %>%
  summarise(n = n(),
            mean = mean(trstprl),
            median = median(trstprl),
            sd = sd(trstprl)
  )
```

```
## # A tibble: 1 x 4
##       n  mean median    sd
##   <int> <dbl>  <dbl> <dbl>
## 1 48375  4.52      5  2.66
```

2. Significance Test for a Mean

We will now perform a two-sided significance test. Previous research has shown that trust in Parliament was 5 on average and we want to find out whether this still the case or whether it has changed and we want to base our decision using a 0.95 confidence level.

We first write down the null and alternative hypotheses for this test.

$$H_0 : \mu = 5$$

$$H_a : \mu \neq 5$$

Base R does not have a dedicated function to calculate either the standard error or the z-statistic. As a result, you must either create your own functions, perform the calculations in steps, or both. This is another reason why you must ensure that you understand how the formulae discussed in the lectures work and how they are constructed.

As you already know, the standard error is calculated by dividing the sample standard deviation by the square root of the sample size. One way is to use the base R approach to first calculate the sample size and standard deviation separately, and store them in separate objects.

```
n_trstprl <- sum(!is.na(ess9$trstprl))

n_trstprl
```

```
## [1] 48375
```

```
sd_trstprl <- sd(ess9$trstprl, na.rm = TRUE)

sd_trstprl
```

```
## [1] 2.664302
```

Now we use these two data objects to calculate the standard error by dividing the `sd_trstprl` by the square root of the `n_trstprl` object and store it in an object called `se`.

```
se_trstprl <- sd_trstprl/sqrt(n_trstprl)

se_trstprl
```

```
## [1] 0.01211359
```

Since the sample size is large, we can use the z-score as our test statistic. We calculate our z-statistic by dividing the difference between our predicted value (i.e. the score for neutrality) from the calculated mean by the standard error.

First, we calculate the mean and store it in an object.

```
mean_trstprl <- mean(ess9$trstprl, na.rm = T)
```

Now we use this object and the `se_trstprl` object to calculate the z-score and store is it in an object called `z_trstprl`. Note how the numerator is placed between brackets; remember that R will not know the order in which to perform mathematical operations so the absence of the bracket will produce an incorrect result.

```
z_trstprl <- (mean_trstprl - 5)/se_trstprl

z_trstprl
```

```
## [1] -39.34162
```

Our z-score is -39.3416 which seems to be quite a large negative value but how do we compare it?

Now let's consider what the critical region for a 5% two-tailed z-test is.

Remember that we have to consider both tail probabilities, the right and the left; for a normal distribution, 95% of the observations fall above -1.96 standard deviations and below 1.96 standard deviations from the mean. Since the standard normal uses the standardised values (i.e. the z-scores), then for a two-tailed test, there will be two critical (rejection) regions, *above a z score of 1.96* and *below a z-score of -1.96*. Conversely, if the z score is between -1.96 and 1.96, then it falls within the **acceptance region** and there is no evidence to reject the null hypothesis.

Hence, if the calculated z score is **larger** than 1.96 OR **smaller than** -1.96, this means that the value is not part of the 95% of observations than fall between -1.96 and 1.96 from the mean; instead, it falls in the rejection region(s). Therefore, we have evidence to reject the null hypothesis. The -1.96 and 1.96 values are also referred to as "cut-off points" or "critical values".

The cut-off point(s) can also be calculated in R using the `qnorm()` function. This function provides percentiles for the standard normal distribution given a user-defined probability. In this case, we have a two-tailed probability which is equal to the alpha value (0.05). As you'll remember from the lectures, this alpha value is the total probability for both tails (the sum of 0.025 in the right tail and 0.025 in the left tail) that the null hypothesis is rejected when in fact it is true (or the Type I error).

Hence, what we doing is finding the z-score for 0.025 probability so, we divide 0.05 by 2. By default, the `qnorm()` function will provide a z-score for the lower tail and so the z-score will be negative.

```
qnorm(0.05/2)
```

```
## [1] -1.959964
```

To produce a positive z-score, we can set the `lower.tail` argument to FALSE which will then yield the right-tail z-score. As you can see, the z-score is now positive 1.96 and therefore, our cut-off point is ±1.96 for a two-tailed test. To find the cut-off point for a two-tailed test, you do not need to run both functions since the absolute value of the z-score is the same. You can simply use the below and specify the cut-off point as ±1.96.

```
qnorm(0.05/2, lower.tail = FALSE)
```

```
## [1] 1.959964
```

Our calculated z-score of -39.3416 does not fall between ±1.96. Therefore, there is evidence to reject the null hypothesis. However, let's formalise our conclusion with the p-value.

Let's find the p-value for the z-score we calculated (-39.3416). To do so, we can use the `pnorm()` base R function which provides the probabilities of variable values occurring under the standard normal

distribution assumption. In other words, the function provides the probability of z-scores and provides the same results you would obtain from a standard normal distribution table. The function can be used for both one-tailed and two-tailed tests and it can be considered to be the inverse of the `qnorm()` function. Whereas `qnorm()` can provide z-scores, the `pnorm()` function provides probabilities.

As with `qnorm(0)`, the `pnorm()` function also provides left-tail probabilities by default. To obtain the correct p-value for a two-tailed test, then there are a couple of things to add to the function.

Fist, for two two-tailed probabilities, you must multiply the obtained p-value by 2 since there are two tails (remember the symmetry feature of the normal distribution).

Second, the `lower.tail` argument must be set to FALSE.

Third, the z-score must be wrapped which the `abs()` function. This provides the absolute value for any value (i.e. removes the negative sign if the value is negative).

```
2 * pnorm(abs(z_trstprl), lower.tail = F)
```

```
## [1] 0
```

The p-value is 0. Since this is less than our significance level (0.05), we reject the null hypothesis. This means that the data does not support the claim that the mean trust in the parliament was on average 5.

3. Significance Test for a Proportion

We are now going to explore how to perform a two-sided significance test for a proportion at 0.95 confidence level. Let's say that we know from previous studies that the population mean vote attendance in the EU in 75% and we want to find out if this is still the case.

If we explore the `vote` variable using the `attributes()` function, we find out that this is a nominal variable with three main categories, "Yes", "No", and "Not eligible to vote", designated by numeric labels 1 to 3 respectively, and that three categories are designated for `NA` values.

```
attributes(ess9$vote)
```

```
## $label
## [1] "Voted last national election"
##
## $format.stata
## [1] "%3.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##                   Yes                    No Not eligible to vote
##                     1                     2                     3
##               Refusal           Don't know            No answer
##                    NA                    NA                    NA
```

We also find out that the variable is of type "double" when it should be factor (given that it is nominal). However, we do not need to transform it into a factor for significance tests (but you may want to do so for visualisations).

For our analysis, we are only interested in those survey participants who voted or did not vote, so we will leave the "Yes" and "No" categories labelled numerically as 1 and 2 respectively and drop all other categories (including any missing values) to create a new variable called `vote_binary`. Note: to keep the 1 and 2 values numeric, do not place them between double inverted commas (remember that if you do, they will be transformed to character values.)

```
ess9 <- ess9 %>%
  mutate(vote_binary = case_when(vote == 1 ~ 1,
                                 vote == 2 ~ 2)) %>%
  drop_na(vote_binary)
```

As you can see, the `vote_binary` is now a binary variable with numeric labels.

```
count(ess9, vote_binary)
```

```
## # A tibble: 2 x 2
##   vote_binary     n
##         <dbl> <int>
## 1           1 35505
## 2           2  9728
```

Now let's specify our hypotheses.

$$H_0 : \pi = 0.75$$

$$H_a : \pi \neq 0.75$$

There are many ways to calculate proportions in R, as you already know (e.g. using the `janitor` package). Another way is of course, using base R.

To create a proportions table with base R, wrap the `table()` function within the `prop.table()` function. The `table()` function simply displays the frequencies of each category of the variable and is the equivalent of the `count()` function except that the former does not display missing data. The `prop.table()` simply calculates the conditional proportions.

```
prop.table(table(ess9$vote_binary))
```

```
##
##         1         2
## 0.7849358 0.2150642
```

For our significance test, we are only interested in the proportion for the "Yes" category. This category represents the first column of the table, so we subset this value by placing the index value of one between square brackets at the end of the argument and save this value in an object called `prop_vote`. Note: remember that when specifying indices which represent column or row numbers, these are integers so we do not add double inverted commas. We will use this value later when we calculate the z statistic.

```
prop_vote <- prop.table(table(ess9$vote_binary))[1]
```

For our calculations, we also need the total number of observations.

```
n_vote <- sum(ess9$vote_binary)

n_vote
```

```
## [1] 54961
```

We now calculate the standard error under the presumption that the null hypothesis is true, by dividing the product of the null hypothesis proportion and 1 minus the null hypothesis proportion, by the total number of observations.

```
se_vote <- sqrt((0.75 * (1 - 0.75))/n_vote)

se_vote
```

```
## [1] 0.001847027
```

To then calculate the z-statistic, we subtract our null hypothesis proportion from the sample proportion we calculated earlier, and divide this result by the standard error.

```
z_vote <- (prop_vote - 0.75)/se_vote

z_vote
```

```
##       1
## 18.9146
```

Finally, we find the p-value associated with the calculated z-score. Note that since we know that the z-score is positive, we no longer need to use the `abs()` function.

```
2 * pnorm(z_vote, lower.tail = F)
```

```
##            1
## 8.647964e-80
```

The p-value is extremely small (8.647964e-80). Since this is less than our significance level (0.05), we reject the null hypothesis that the population mean vote attendance in the EU is 75%.

When numbers are extremely small or large, R provides them in exponential (scientific) notation; in this case, the p-value is translated as 0.0000.....and many other zeroes.....205988 (or in other words, 2.05988 to minus 88 power).

If you are unfamiliar with this type of notation, please have a look at the following resource https://www.graphpad.com/support/faq/what-does-it-mean-when-some-results-have-e-in-the-number/

If you want to find out how numbers are represented in different formats, you can use an online number converter such as: https://www.calculatorsoup.com/calculators/math/scientific-notation-converter.php