R Users Guide to Stat 201: Chapter 8 and 9

Michael Shyne, 2017

Chapters 8 and 9: Hypothesis Testing

Chapters 8 and 9 both concern hypothesis testing, specifically proportion tests and mean tests. Chapter 8 introduces one sample tests and chapter 9 continues with two sample tests. However, the R functions are the same for one sample and two sample tests with only small changes to parameters. Thus, we will tackle both chapters together in this guide.

Proportion tests

One sample proportion tests are conducted to determine whether a sample is drawn from a population that has a particular proportion of "successes." In R, they are conducted using the `prop.test()` function, which takes the parameters `x` (number of successes) and `n` (sample size). This is equivalent to the StatCrunch test "with summary". To conduct a test of a sample with 125 successes out of a sample size of 300,

```
prop.test(125, 300, correct=FALSE )
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  125 out of 300, null probability 0.5
## X-squared = 8.3333, df = 1, p-value = 0.003892
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.3622761 0.4731644
## sample estimates:
##         p
## 0.4166667
```

When running an hypothesis test, often the first thing we are interest in is the p-value. This can be found on the second line of output (not counting the title and blank lines). For this test, `p-value = 0.003892`.

Next, we would like to know the test statistic. This is also found in the second line. However, we are not given a z value as we might expect. Instead, R reports `X-squared = 8.3333`. The `X` here is not really an "X", but rather represents the Greek letter "chi" (χ). Thus, the test statistic is the χ^2 (chi-squared) statistic. The chi^2 distribution will be introduced later in the course. For now, it suffices to note that chi^2 is related to the z distribution squared. In fact, for our one sample proportion test, the the chi^2 statistic is the square of the z statistic. To find the z test statistic, simply take the square root of chi^2 value keeping in mind the z statistic will be negative is the sample proportion is less than the null proportion (more on this later).

The test output gives us a lot more information. The first line reports the data used to conduct the test, as well as the proportion test against as the null probability. Since we did not specify a proportion to test, the function used the default value of 0.5. The third line tells us the alternative hypothesis used. Again, since we did not specify a value, it used the default alternative hypothesis of "not equal" to the null proportion. Lines four and five give us a confidence interval, defaulting to a 95% interval. Lines six and seven report the sample proportion.

You should have noticed the `correct=FALSE` parameter we added to this first function call. By default, R will apply a Yates' continuity correct to the test. The details of this correct are beyond the scope of this class. In order to get the same results as StatCrunch, and the results MyStatLab expects, we need to always include the `correct=FALSE` option to prevent the correction from being used.

Of course, we will often want to change the proportion we are testing against and the alternative hypothesis test. To change the null proportion, include the `p=` parameter. It can take any valid proportion ($0 < p < 1$). To use a different alternative hypothesis, include the parameter `alternative=` (this can be shorted to `alt=`). It can take one of three strings, " "two.sided", "less" or "greater" (these can also be abbreviated, i.e "two" or "great"). Thus, to test whether a population proportion is greater then 0.45, using the same data as above,

```
prop.test(125, 300, p=0.45, alt='greater', correct=FALSE)   # Don't forget the correction
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  125 out of 300, null probability 0.45
## X-squared = 1.3468, df = 1, p-value = 0.8771
## alternative hypothesis: true p is greater than 0.45
## 95 percent confidence interval:
##  0.3707965 1.0000000
## sample estimates:
##         p
## 0.4166667
```

A note about confidence intervals and one-sided tests: As stated above the `prop.test()` function, as well as many other R functions, defaults to a 95% confidence interval. This can be adjusted by including the `conf.int=` parameter in the function call (for example, `confi.int=0.9`). However, when testing a one-sided alternative hypothesis ($p > 0.035$), R will report a "proper" confidence interval. Such intervals are not symmetric, putting the whole rejection region on one side or the other of the distribution. Thus, it is not necessary to employ to trick necessary in StatCrunch of calculating a confidence interval with a $2 \times \alpha$ rejection region.

Test results as objects

While the output from `prop.test()` provides a lot of useful information, sometimes we wish to use the results in a different manner. If we want to automatically store the results in some fashion, such as logging the results in an automated process, or perhaps we want to do further calculations with the results. The output of the proportion test function and, as we will see, many other R functions can be stored in a variable.

```
p.test.results <- prop.test(125, 300, p=0.45, alt='greater', correct=FALSE)
```

As you can see, when we do this we don't get any kind of output. If we want to see the original output, we can simply type the variable name.

```
p.test.results
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  125 out of 300, null probability 0.45
## X-squared = 1.3468, df = 1, p-value = 0.8771
## alternative hypothesis: true p is greater than 0.45
## 95 percent confidence interval:
##  0.3707965 1.0000000
## sample estimates:
##         p
## 0.4166667
```

To see what information the variable contains, we can examine its structure.

```
str(p.test.results)
```

```
## List of 9
##  $ statistic  : Named num 1.35
##   ..- attr(*, "names")= chr "X-squared"
##  $ parameter  : Named int 1
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 0.877
##  $ estimate   : Named num 0.417
##   ..- attr(*, "names")= chr "p"
##  $ null.value : Named num 0.45
##   ..- attr(*, "names")= chr "p"
##  $ conf.int   : atomic [1:2] 0.371 1
##   ..- attr(*, "conf.level")= num 0.95
##  $ alternative: chr "greater"
##  $ method     : chr "1-sample proportions test without continuity correction"
##  $ data.name  : chr "125 out of 300, null probability 0.45"
##  - attr(*, "class")= chr "htest"
```

The output of the `str()` function can be intimidating, but if we look carefully, we can see out result object contains all the relevant information provided in the default output. By referencing the named parameters, we display specific values or do further calculations.

```
# What is the p-value?
p.test.results$p.value
```

```
## [1] 0.877081
```

```
# What is the test statistic?
p.test.results$statistic
```

```
## X-squared
##  1.346801
```

```
# The z statistic is the square root of the chi-squared value
z <- sqrt(p.test.results$statistic)
names(z)<- "z statistic"   # Change the "name" of the statistic
                           #  This is only for display purposes

# If the sample proportion is less than the null proportion.
#   then z will have a negative value
if (p.test.results$estimate < p.test.results$null.value)
    z <- -1 * z
z
```

```
## z statistic
##   -1.160518
```

Two sample proportion tests

Two sample proportion tests are conducted much in the same way as one sample tests. The function `prop.test()` is still used, but now vectors of successes and sample sizes are passed to the function. For example, to test our original sample of 125 successes out of 300 against another sample with 142 successes out of a sample size of 290,

```
prop.test(c(125,142), c(300,290), correct=FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(125, 142) out of c(300, 290)
## X-squared = 3.1708, df = 1, p-value = 0.07497
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.153128869  0.007151858
## sample estimates:
##    prop 1    prop 2
## 0.4166667 0.4896552
```

Much of what we did with one sample proportions can be done with two sample proportions. One important difference is the the null proportion parameter `p=` works differently and, for our purposes, be ignored in two proportion tests.

Proportion tests "with data"

Often proportion data will be reported as successes out of sample size as we've been working with. Sometimes, however, you have a full data set you wish to use for hypothesis tests. While you could count values that correspond to successes and then conduct test as we have been, there are some sort cuts available to us in R.

Consider the `mtcars` data set. The variable `am` refers to the presence of automatic transmission (0 = automatic, 1 = manual). To summarize this data, we can use the `table()` function.

```
table(mtcars$am)
```

```
##
##  0  1
## 19 13
```

We can see that there are 19 cars with automatic transmissions and 13 with manual transmissions. We can pass this table straight into `prop.test()`.

```
prop.test(table(mtcars$am), correct=F)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  table(mtcars$am), null probability 0.5
## X-squared = 1.125, df = 1, p-value = 0.2888
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.4226002 0.7448037
## sample estimates:
##       p
## 0.59375
```

If we conduct the same test with the data is summary form (19 successes out of 32 sample size), we get identical results.

```
prop.test(19, 32, correct=F)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  19 out of 32, null probability 0.5
## X-squared = 1.125, df = 1, p-value = 0.2888
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.4226002 0.7448037
## sample estimates:
##       p
## 0.59375
```

Keep in mind, R considers the first item in the table, alphabetically, the "successes". In this example, the automatic transmissions with a 0 value were the successes. It can take some effort if you want to consider a different value the successes. If the variable you are testing has named values stored as character strings, it might require renaming the categories so that the category of interest comes first alphabetically. In our case, since the values are stored as 1 and 0, we can test `1 - am` to "flip" the values.

```r
# Test 1 - am, so manual transmissions (am=1) come first
prop.test(table(1-mtcars$am), correct=FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  table(1 - mtcars$am), null probability 0.5
## X-squared = 1.125, df = 1, p-value = 0.2888
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.2551963 0.5773998
## sample estimates:
##       p
## 0.40625
```

Mean tests (t tests)

One sample mean tests are conducted to determine whether a sample is drawn from a population that has a particular mean parameter (μ). In R, we will use the function `t.test()`. This works similarly to `prop.test()` with a few notable exception. First, `t.test()` expects sample data. It does not work with summary data as StatCrunch can. Thus, the first parameter in the function call should be a vector of numeric data. Second, the function parameter for the null value is `mu=` rather than `p=`.

Using the `mtcars` data set again, we can test if the mean fuel efficiency of the cars is less than 25 mpg.

```r
t.test(mtcars$mpg, mu=25, alt='less')
```

```
##
##  One Sample t-test
##
## data:  mtcars$mpg
## t = -4.6079, df = 31, p-value = 3.293e-05
## alternative hypothesis: true mean is less than 25
## 95 percent confidence interval:
##      -Inf 21.89707
## sample estimates:
## mean of x
##  20.09062
```

We can see that the output is very similar to the output for proportion tests. The p-value, test statistic (t-value) and confidence intervals are available. Like proportion tests, results can be stored in a variable for further manipulation.

Two independent sample t tests

Two sample t tests are conducted by passing two numeric vectors to `t.test()`. For example, suppose we wish to compare the fuel efficiency of cars with automatic transmissions and manual transmissions.

```
mpg.auto <- mtcars$mpg[mtcars$am==0]     # mpg of auto trans cars
mpg.manual <- mtcars$mpg[mtcars$am==1]   # mpg of manual trans cars

# Are the mean mpgs the same?
t.test(mpg.auto, mpg.manual)
```

```
## 
##  Welch Two Sample t-test
## 
## data:  mpg.auto and mpg.manual
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean of x mean of y 
##  17.14737  24.39231
```

Paired data t tests

Paired data t tests are conducted by including `paired=TRUE` in the function call. The `immer` data set contains data from an experiment on barley yields. Various varieties of barley were planted in different locations over two years. We can test whether the yields changed from year 1 to year 2.

```
require(MASS)    # The dataset is in the MASS library
```

```
## Loading required package: MASS
```

```
# Is the mean difference in yeilds (Y1 - Y2) zero?
t.test(immer$Y1, immer$Y2, paired=TRUE)
```

```
## 
##  Paired t-test
## 
## data:  immer$Y1 and immer$Y2
## t = 3.324, df = 29, p-value = 0.002413
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   6.121954 25.704713
## sample estimates:
## mean of the differences 
##                15.91333
```

t tests with summary data

While there are not built-in functions to do t tests with summary data, it is not difficult to calculate test statistics "by hand" and then calculate p-values. Recall, given a sample mean $\bar{x}$ and sample standard deviation s, the t test statistic for a test with a null hypothesis $\mu = \mu_0$ is calculated by

$$t = \frac{\bar{x} - \mu_0}{s}.$$

Then, the p-value for a t test is the probability of a value equal to or more extreme than t in a t distribution. This can be calculated with the `pt()` function.

For example, to test a sample with mean $\bar{x} = 15.6$, standard deviation on $s = 2.1$ and sample size $n = 20$ against $H_0 : \mu = 13$,

```
x.bar <- 15.6
s <- 2.1
n <- 20
mu.0 <- 13

t <- (x.bar - mu.0)/s
t
```

```
## [1] 1.238095
```

```
# Since t > 0, we want the upper tail
p.val <- pt(t, df=n-1, lower.tail=FALSE)
p.val
```

```
## [1] 0.1153808
```

Since this is the probability of just one tail, if you are conducting a one-sided test ($H_a : \mu > 13$), this is your p-value. However, if you are conducting a two-sided test ($H_a : \mu \neq 13$), remember to multiply this value by 2 to account for both tails.

License