10. Calculating *p* Values

Contents

- Calculating a Single *p* Value From a Normal Distribution
- Calculating a Single *p* Value From a t Distribution
- Calculating Many *p* Values From a t Distribution
- The Easy Way

Here we look at some examples of calculating *p* values. The examples are for both normal and t distributions. We assume that you can enter data and know the commands associated with basic probability. We first show how to do the calculations the hard way and show how to do the calculations. The last method makes use of the *t.test* command and demonstrates an easier way to calculate a *p* value.

10.1. Calculating a Single *p* Value From a Normal Distribution

We look at the steps necessary to calculate the *p* value for a particular test. In the interest of simplicity we only look at a two sided test, and we focus on one example. Here we want to show that the mean is not close to a fixed value, a.

$$H_o : \mu_x = a,$$

$$H_a : \mu_x \neq a,$$

The *p* value is calculated for a particular sample mean. Here we assume that we obtained a sample mean, x and want to find its *p* value. It is the probability that we would obtain a given sample mean that is greater than the absolute value of its Z-score or less than the negative of the absolute value of its Z-score.

For the special case of a normal distribution we also need the standard deviation. We will assume that we are given the standard deviation and call it *s*. The calculation for the *p* value can be done in several of ways. We will look at two ways here. The first way is to convert the sample means to their associated Z-score. The other way is to simply specify the standard deviation and let the computer do the conversion. At first glance it may seem like a no brainer, and we should just use the second method. Unfortunately, when using the t-distribution we need to convert to the t-score, so it is a good idea to know both ways.

We first look at how to calculate the *p* value using the Z-score. The Z-score is found by assuming that the null hypothesis is true, subtracting the assumed mean, and dividing by the theoretical standard deviation. Once the Z-score is found the probability that the value could be less the Z-score is found using the *pnorm* command.

This is not enough to get the *p* value. If the Z-score that is found is positive then we need to take one minus the associated probability. Also, for a two sided test we need to multiply the result by two. Here we avoid these issues and insure that the Z-score is negative by taking the negative of the absolute value.

We now look at a specific example. In the example below we will use a value of a of 5, a standard deviation of 2, and a sample size of 20. We then find the *p* value for a sample mean of 7:

```
> a <- 5
> s <- 2
> n <- 20
> xbar <- 7
> z <- (xbar-a)/(s/sqrt(n))
> z
[1] 4.472136
> 2*pnorm(-abs(z))
[1] 7.744216e-06
```

We now look at the same problem only specifying the mean and standard deviation within the *pnorm* command. Note that for this case we cannot so easily force the use of the left tail. Since the sample mean is more than the assumed mean we have to take two times one minus the probability:

```
> a <- 5
> s <- 2
> n <- 20
> xbar <- 7
> 2*(1-pnorm(xbar,mean=a,sd=s/sqrt(20)))
[1] 7.744216e-06
```

10.2. Calculating a Single *p* Value From a t Distribution

Finding the *p* value using a t distribution is very similar to using the Z-score as demonstrated above. The only difference is that you have to specify the number of degrees of freedom. Here we look at the same example as above but use the t distribution instead:

```
> a <- 5
> s <- 2
> n <- 20
> xbar <- 7
> t <- (xbar-a)/(s/sqrt(n))
> t
[1] 4.472136
> 2*pt(-abs(t),df=n-1)
[1] 0.0002611934
```

We now look at an example where we have a univariate data set and want to find the *p* value. In this example we use one of the data sets given in the data input chapter. We use the w1.dat data set:

```
> w1 <- read.csv(file="w1.dat",sep=",",head=TRUE)
> summary(w1)
      vals
 Min.   :0.130
 1st Qu.:0.480
 Median :0.720
 Mean   :0.765
 3rd Qu.:1.008
 Max.   :1.760
> length(w1$vals)
[1] 54
```

Here we use a two sided hypothesis test,

$$H_o : \mu_x = 0.7,$$

$$H_a : \mu_x \neq 0.7.$$

So we calculate the sample mean and sample standard deviation in order to calculate the p value:

```
> t <- (mean(w1$vals)-0.7)/(sd(w1$vals)/sqrt(length(w1$vals)))
> t
[1] 1.263217
> 2*pt(-abs(t),df=length(w1$vals)-1)
[1] 0.21204
```

10.3. Calculating Many p Values From a t Distribution

Suppose that you want to find the p values for many tests. This is a common task and most software packages will allow you to do this. Here we see how it can be done in R.

Here we assume that we want to do a one-sided hypothesis test for a number of comparisons. In particular we will look at three hypothesis tests. All are of the following form:

$$H_o : \mu_1 - \mu_2 = 0,$$

$$H_a : \mu_1 - \mu_2 \neq 0.$$

We have three different sets of comparisons to make:

Comparison	1		
	Mean	Std. Dev.	Number (pop.)
Group I	10	3	300
Group II	10.5	2.5	230

Comparison	2		
	Mean	Std. Dev.	Number (pop.)
Group I	12	4	210
Group II	13	5.3	340

Comparison	3		
	Mean	Std. Dev.	Number (pop.)
Group I	30	4.5	420
Group II	28.5	3	400

For each of these comparisons we want to calculate a *p* value. For each comparison there are two groups. We will refer to group one as the group whose results are in the first row of each comparison above. We will refer to group two as the group whose results are in the second row of each comparison above. Before we can do that we must first compute a standard error and a t-score. We will find general formulae which is necessary in order to do all three calculations at once.

We assume that the means for the first group are defined in a variable called *m1*. The means for the second group are defined in a variable called *m2*. The standard deviations for the first group are in a variable called *sd1*. The standard deviations for the second group are in a variable called *sd2*. The number of samples for the first group are in a variable called *num1*. Finally, the number of samples for the second group are in a variable called *num2*.

With these definitions the standard error is the square root of *(sd1^2)/num1+(sd2^2)/num2*. The associated t-score is *m1* minus *m2* all divided by the standard error. The R comands to do this can be found below:

```
> m1 <- c(10,12,30)
> m2 <- c(10.5,13,28.5)
> sd1 <- c(3,4,4.5)
> sd2 <- c(2.5,5.3,3)
> num1 <- c(300,210,420)
> num2 <- c(230,340,400)
> se <- sqrt(sd1*sd1/num1+sd2*sd2/num2)
> t <- (m1-m2)/se
```

To see the values just type in the variable name on a line alone:

```
> m1
[1] 10 12 30
> m2
[1] 10.5 13.0 28.5
> sd1
[1] 3.0 4.0 4.5
> sd2
[1] 2.5 5.3 3.0
> num1
[1] 300 210 420
> num2
[1] 230 340 400
> se
[1] 0.2391107 0.3985074 0.2659216
> t
[1] -2.091082 -2.509364  5.640761
```

To use the *pt* command we need to specify the number of degrees of freedom. This can be done using the *pmin* command. Note that there is also a command called *min*, but it does not work the same way. You need to use *pmin* to get the correct results. The numbers of degrees of freedom are *pmin(num1,num2)-1*. So the *p* values can be found using the following R command:

```
> pt(t,df=pmin(num1,num2)-1)
[1] 0.01881168 0.00642689 0.99999998
```

If you enter all of these commands into R you should have noticed that the last *p* value is not correct. The pt command gives the probability that a score is less that the specified t. The t-score for the last entry is positive, and we want the probability that a t-score is bigger. One way around this is to make sure that all of the t-scores are negative. You can do this by taking the negative of the absolute value of the t-scores:

```
> pt(-abs(t),df=pmin(num1,num2)-1)
[1] 1.881168e-02 6.426890e-03 1.605968e-08
```

The results from the command above should give you the *p* values for a one-sided test. It is left as an exercise how to find the *p* values for a two-sided test.

10.4. The Easy Way

The methods above demonstrate how to calculate the *p* values directly making use of the standard formulae. There is another, more direct way to do this using the *t.test* command. The *t.test* command takes a data set for an argument, and the default operation is to perform a two sided hypothesis test.

```
> x = c(9.0,9.5,9.6,10.2,11.6)
> t.test(x)

                                One Sample t-test

data:  x
t = 22.2937, df = 4, p-value = 2.397e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
        8.737095 11.222905
sample estimates:
mean of x
                      9.98

> help(t.test)
>
```

That was an obvious result. If you want to test against a different assumed mean then you can use the *mu* argument:

```
> x = c(9.0,9.5,9.6,10.2,11.6)
> t.test(x,mu=10)

                                One Sample t-test

data:  x
t = -0.0447, df = 4, p-value = 0.9665
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
        8.737095 11.222905
sample estimates:
mean of x
                        9.98
```

If you are interested in a one sided test then you can specify which test to employ using the *alternative* option:

```
> x = c(9.0,9.5,9.6,10.2,11.6)
> t.test(x,mu=10,alternative="less")

                                One Sample t-test

data:  x
t = -0.0447, df = 4, p-value = 0.4833
alternative hypothesis: true mean is less than 10
95 percent confidence interval:
                        -Inf 10.93434
sample estimates:
mean of x
                        9.98
```

The *t.test()* command also accepts a second data set to compare two sets of samples. The default is to treat them as independent sets, but there is an option to treat them as dependent data sets. (Enter *help(t.test)* for more information.) To test two different samples, the first two arguments should be the

data sets to compare:

```
> x = c(9.0,9.5,9.6,10.2,11.6)
> y=c(9.9,8.7,9.8,10.5,8.9,8.3,9.8,9.0)
> t.test(x,y)
        Welch Two Sample t-test

data:  x and y
t = 1.1891, df = 6.78, p-value = 0.2744
alternative hypothesis true difference in means is not equal to 0
95 percent confidence interval:
       -0.6185513  1.8535513
sample estimates:
mean of x mean of y
               9.9800    9.3625
```

Previous Next