Confidence Intervals in R

This document shows how to construct confidence intervals in R. We will show methods of finding confidence intervals for proportions and means of distributions. Two methods will be used; one is a direct calculation with an appropriate formula, second one using built-in functions.

Introduction to confidence intervals

In this section we introduce a very important discrete distribution which is Binomial. Suppose we are flipping a coin 10 times and that probability of a head is 50% (fair coin) then binomial distribution gives probabilities of getting x heads. Clearly in this case the possible values for x is 0,1,2 up to 10. Let's make a plot of probabilities. First we construct a vector of 0,1,2 up to 10:

```
x=c(0:10)
x
```

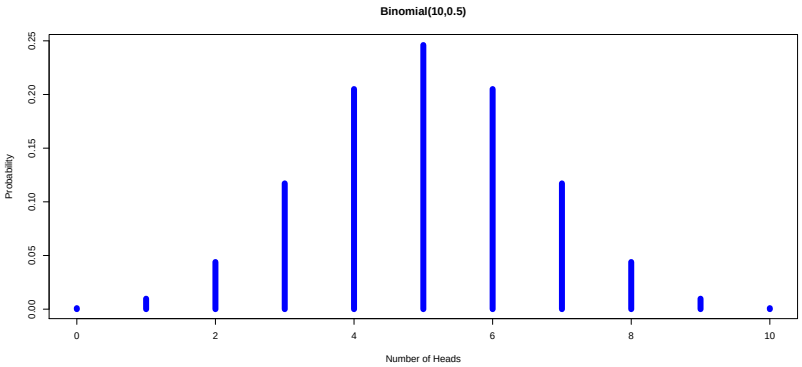
```
[1]  0  1  2  3  4  5  6  7  8  9 10
```

Now we use 'dbinom' function to find the probabilities. Here we use two arguments 'size' which is the number of trials and 'prob' which is probability of success:

```
y=dbinom(x,size=10,prob=0.5)
```

Now 'y' variable contains probabilities and we make a scatterplot of 'y' versus 'x':

```
plot(x,y,col='blue',type='h',lwd=10,xlab='Number of Heads',ylab='Probability',
  main='Binomial(10,0.5)')
```



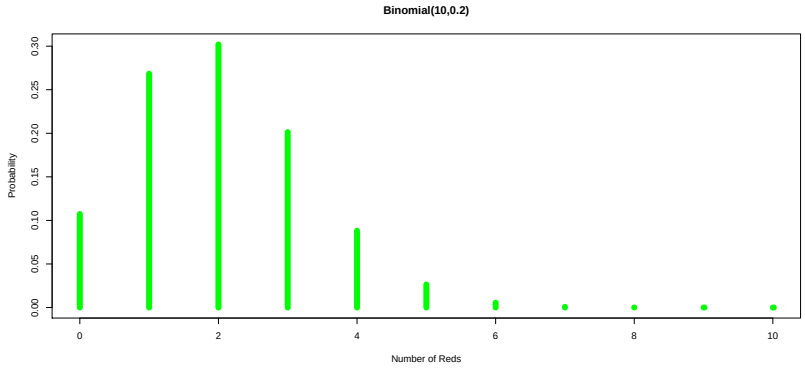
Here 'type='h'' makes vertical bars and 'lwd' determines the width of the bars. We note that the distribution is symmetric and unimodal. If we need for example probability of 3 successes then we can easily get it from the 'y' variable:

```
y[x==3]
```

```
1] 0.1171875
```

Now let's make a similar plot of probabilities but with probability of success equals to 0.2:

```
y=dbinom(x,size=10,prob=0.2)
plot(x,y,col='green',type='h',lwd=10,xlab='Number of Reds',ylab='Probability',
 main='Binomial(10,0.2)')
```



This distribution is no longer symmetric but shifted to the right.

Confidence intervals for proportions

In this section we show how to find confidence intervals for one proportion. We will do it using two methods. One by implementing the formula given in video lectures and second one by using build-in function.
First we consider the 'Bottle cap flipping' example. Here 1000 flips were done and we get Red 576 times. To make next calculations more general we will introduce variables 'N' (number of trials), 'num.success' (number of successes) and 'p.hat' which is sample proportion:

```
N=1000
num.success=576
p.hat=num.success/N
```

Next we want 95% confidence interval and hence for convenience we construct a new variable 'conf.level':

```
conf.level=0.95
```

In the next examples we will just change the value of this variable. Once we know how confident we want to be, we can find the critical value which is used in the construction of confidence intervals:

```
crit.val=qnorm( 1-(1-conf.level)/2 )
crit.val
```

```
[1] 1.959964
```

'$qnorm(p)$' is defined as the number, such that $p * 100\%$ of $\mathcal{N}(0,1)$ is below it. We see that this number is almost 1.96 and this critical value for 95% confidence level people frequently use. The next step is to find the margin of error and we use the usual 'conservative' formula with $\frac{1}{2}$ for unknown p:

```
ME=crit.val*sqrt(1/2*(1-1/2)/N)
```

At this stage we have all the ingredients for confidence interval. To get a nice looking output we use a 'cat' function which allows us to print words with different calculated variables ('$\backslash n$' in the end is needed to finish the line)

```
cat('CI for p is from ',p.hat-ME,' to ',p.hat+ME, '\n')
```

```
CI for p is from  0.5450102  to  0.6069898
```

Instead of using the formulas as we did above we can use the build-in function which calculates the confidence interval:

```
prop.test(x=num.success,n=N,conf.level=0.95,correct=FALSE)$conf.int
```

```
[1] 0.5451368 0.6062816
attr(,"conf.level")
[1] 0.95
```

To use this function we have to indicate the number of successes, total number of trials and confidence level. It can also perform some corrections to make confidence intervals more accurate but we do not need it here. We see that confidence intervals are almost the same. The first one is a little wider because we have implemented conservative margin of error.

Using the above procedure we can get for example 90% CI:

```
conf.level=0.90
crit.val=qnorm( 1-(1-conf.level)/2 )
ME=crit.val*sqrt(1/2*(1-1/2)/N)
cat('CI for p is from ',p.hat-ME,' to ',p.hat+ME, '\n')
```

```
CI for p is from  0.5499926  to  0.6020074
```

Equivalently we can use 'prop.test' function:

```
prop.test(x=num.success,n=N,conf.level=0.90,correct=FALSE)$conf.int
```

```
1] 0.5501236 0.6014663
attr(,"conf.level")
[1] 0.9
```

Note that these intervals are shorter than for 95% confidence level. Finally let's get 99% confidence intervals using both approaches:

```
conf.level=0.99
crit.val=qnorm( 1-(1-conf.level)/2 )
ME=crit.val*sqrt(1/2*(1-1/2)/N)
cat('CI for p is from ',p.hat-ME,' to ',p.hat+ME, '\n')
```

```
CI for p is from  0.5352726  to  0.6167274
```

```
prop.test(x=num.success,n=N,conf.level=0.99,correct=FALSE)$conf.int
```

```
[1] 0.5353746 0.6156235
attr(,"conf.level")
[1] 0.99
```

These confidence intervals are wider than 95% confidence intervals. Hence the more confident we want to be, the less precise the confidence interval is.

Sample size for estimating a proportion

In this section we investigate how to estimate the sample size based on the margin of error and confidence level. We should estimate the sample size before an experiment or survey. Suppose we are seeking 95% confidence interval and we want the margin of error to be 3%. Based on these conditions, the goal is to estimate the sample size. First we find critical value and introduce a new variable 'ME' (margin of error):

```
conf.level=0.95
crit.val=qnorm( 1-(1-conf.level)/2 )
ME=0.03
```

Now we use a standard formula with p equals to $\frac{1}{2}$ since before the experiment we do not know the actual proportion and hence we use a conservative value:

```
N=(crit.val*(0.5)/ME)^2
N
```

```
[1] 1067.072
```

Hence the sample size that we will need is around 1067. If we want 99% confidence interval than we repeat the above procedure with 'conf.level' equals to 0.99 instead if 0.95:

```
conf.level=0.99
crit.val=qnorm( 1-(1-conf.level)/2 )
ME=0.03
N=(crit.val*(0.5)/ME)^2
N
```

```
[1] 1843.027
```

In this case the sample size should be approximately 1843. So we see that the more confident we want to be, the larger the sample size should be (with the same margin of error).

Confidence Intervals for Means

We start this section with the 'Age change' data set. This data set includes 60 observations of people who made a plastic surgery. For each subject it is estimated by how many years a person looks younger than before the surgery. To download this data into R we use 'scan' function (since this is not a table but one column of numbers).

```
age.change=scan('agechange.txt')
```

Since the confidence interval for the mean of quantitative variable will involve the student-t distribution, let's compare it with a normal distribution. First we construct 'x' variable with equally spaced values from -4 to 4 with the step equals to 0.01, we can easily do it with the 'seq' function:

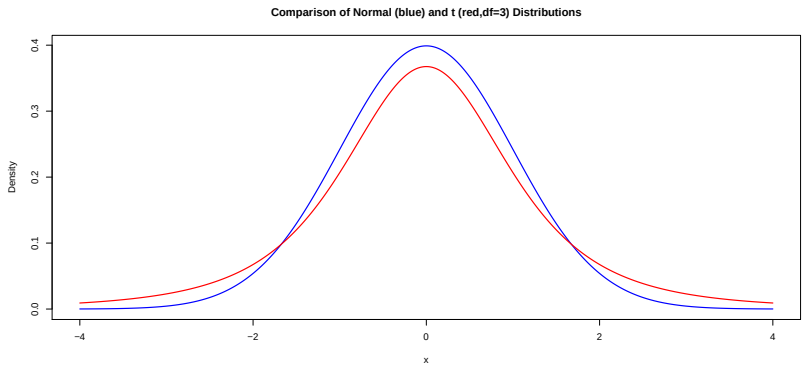
```
x=seq(-4,4,0.01)
```

Next we find the density of normal and student-t (with 3 degrees of freedom) distributions evaluated at the 'x' points:

```
y.norm=dnorm(x)
y.t=dt(x,df=3)
```

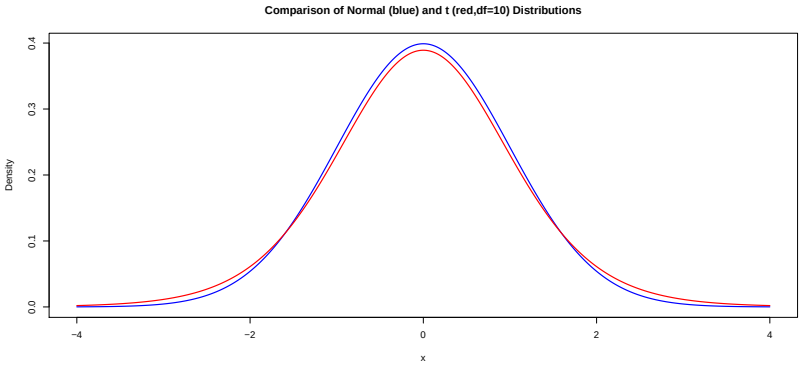
Note that 'dnorm' and 'dt' functions calculate the density of standard normal and student-t distributions respectively. Next we make two curves in one plot using 'points' function followed by 'plot':

```
plot(x,y.norm,col='blue',type='l',lwd=2,main='Comparison of Normal (blue)
  and t (red,df=3) Distributions',ylab='Density')
points(x,y.t,col='red',type='l',lwd=2)
```



Note that 'type='l'' produces lines between points. See that for large and small values of 'x' the density for student-t distribution is higher than for standard normal which shows that it has heavy tails. Next we do a similar plot but with 10 degrees of freedom:

```
y.t=dt(x,df=10)
plot(x,y.norm,col='blue',type='l',lwd=2,main='Comparison of Normal (blue)
  and t (red,df=10) Distributions',ylab='Density')
points(x,y.t,col='red',type='l',lwd=2)
```



Observe that in this plot the density of student-t is more similar to standard normal. Actually the theory tell us that as degrees of freedom increase the student-t approaches standard normal distribution.
Now we return to the 'Age change' example. The goal here is to produce a 95% confidence interval for the average age change. First we find the sample mean, sample variance and sample size ('length' function finds the number of elements in a variable):

```
x.bar=mean(age.change)
sam.var=var(age.change)
N=length(age.change)
```

Next we get critical value for 95

```
conf.level=0.95
crit.val=qt( 1-(1-conf.level)/2, df=(N-1) )
crit.val
```

```
[1] 2.000995
```

Note that it is no longer the usual 1.96 as for standard normal. Finally we obtain the margin of error and confidence interval using the standard formula:

```
ME=crit.val*sqrt(sam.var/N)
cat('CI for mu is from ',x.bar-ME,' to ',x.bar+ME, '\n')
```

```
CI for mu is from  6.415091  to  7.938242
```

Hence we are 95% sure that on average after a plastic surgery, a person looks younger from 6.4 to 7.9 years. Instead of using the above direct method, we can use 't.test' function:

```
t.test(x=age.change,conf.level=0.95)$conf.int
```

```
[1] 6.415091 7.938242
attr(,"conf.level")
[1] 0.95
```

We get exactly the same interval.

Next let's move to the 'Skeleton' data set:

```
Skeleton.data=read.table("SkeletonDataComplete.txt",header=TRUE)
head(Skeleton.data)
attach(Skeleton.data)
```

```
  Sex      BMIcat BMIquant Age DGestimate DGerror SBestimate SBerror
1   2 underweight    15.66  78         44     -34         60     -18
2   1      normal    23.03  44         32     -12         35      -9
3   1  overweight    27.92  72         32     -40         61     -11
4   1  overweight    27.83  59         44     -15         61       2
5   1      normal    21.41  60         32     -28         46     -14
6   1 underweight    13.65  34         25      -9         35       1
```

Here we need 95% CI for the DGerror variable. Using the above procedure we get:

```
x.bar=mean(DGerror)
sam.var=var(DGerror)
N=length(DGerror)
conf.level=0.95
crit.val=qt( 1-(1-conf.level)/2, df=(N-1) )
ME=crit.val*sqrt(sam.var/N)
cat('CI for mu is from ',x.bar-ME,' to ',x.bar+ME, '\n')
```

```
CI for mu is from  -15.53851  to  -12.76149
```

Hence we are 95% sure that the true mean of DGerror is between -15.53 and -12.76. We get the same interval using 't.test':

```
t.test(x=DGerror,conf.level=0.95)$conf.int
```

```
[1] -15.53851 -12.76149
attr(,"conf.level")
[1] 0.95
```

The 90% confidence interval is:

```
t.test(x=DGerror,conf.level=0.90)$conf.int
```

```
[1] -15.31444 -12.98556
attr(,"conf.level")
[1] 0.9
```

Summary of R Functions

We give a short summary of all new and/or important R functions [and arguments] that we used in this Module:

Distributions

```
dbinom() [x,size,prob]
dnorm()
dt() [df]
qnorm()
qt() [df]
```

Confidence Intervals

```
prop.test() [x,n,conf.level,correct]
t.test() [x,conf.level]
```

Miscellaneous

```
cat()
length()
points()
seq()
```