R Users Guide to Stat 201: Chapter 7

Michael Shyne, 2017

Chapter 7: Estimates and Sample Sizes

Chapter 7 introduces confidence intervals and sample size calculations. R does not have much to add. There are no built-in functions for the kind of confidence intervals we are interested. You may notice if you search the documentation that there is a `confint()` function. That will calculated confidence intervals for linear models, which we will cover in later chapters. For now, if we want a function, we'll have to write it ourselves.

Functions

This is a very basic overview of creating your own functions. Functions are created with the `function()` function, which consists of passed parameters and a code block contained in curly braces {}. Values which are returned by the function are passed with the `return()` function.

```
# Create a function which squares a value
sqr <- function(x){
    return(x^2)
}

sqr(4)
```

```
## [1] 16
```

Note that the parameter is not typed in anyway. R will attempt to carry out the instructions on whatever is passed to it.

```
sqr(1:5)
```

```
## [1]  1  4  9 16 25
```

```
sqr(data.frame(a=rnorm(3), b=rnorm(3)))
```

```
##              a           b
## [1,] 1.1148268 0.372097410
## [2,] 1.3814831 0.004169123
## [3,] 0.3736483 0.082676845
```

Confidence intervals

To calculate confidence intervals, we need a point estimate, a standard deviation, a sample size and a confidence level.

```
# Function to calculate confidence intervals
conf.int.1 <- function(x, sd, n, level){
    alpha = 1 - level
    ci <- x + c(-1,1) * qnorm(alpha/2, lower=F) * sd / sqrt(n)
    return(ci)
}

# From the slides
conf.int.1(0.36, sqrt(0.36*0.64), 100, 0.95)
```

```
## [1] 0.2659217 0.4540783
```

Note: in the calculation, `... + c(-1,1) * ...` is an easy way to implement plus/minus.

Optional paramters

In our first function, no default values for parameters are given. Thus, they are all required and an error will occur when calling the function if values are not given for all of them. We can make parameters optional by providing default values. If a value is passed, it will override the default.

```r
# Function to calculate confidence intervals
conf.int.2 <- function(x, sd, n, level=0.95){
    alpha = 1 - level
    ci <- x + c(-1,1) * qnorm(alpha/2, lower=F) * sd / sqrt(n)
    return(ci)
}

# From the slides
conf.int.2(0.36, sqrt(0.36*0.64), 100)
```

```
## [1] 0.2659217 0.4540783
```

```r
# Same problem, 99% confidence level
conf.int.2(0.36, sqrt(0.36*0.64), 100, .99)
```

```
## [1] 0.2363602 0.4836398
```

Critical values from t distribution

This function is now fine as far as it goes, but we learned that, if the population standard deviation is unknown, we should use critical values from the *t* distribution. We could create a separate function for those cases, but that seems inefficient. Let's just add an optional parameter.

```r
# Function to calculate confidence intervals
conf.int.3 <- function(x, sd, n, level=0.95, use.t=FALSE){
    alpha = 1 - level

    if (use.t){
        crit.value <- qt(alpha/2, df=n-1, lower=F)
    } else {
        crit.value <- qnorm(alpha/2, lower=F)
    }

    ci <- x + c(-1,1) * crit.value * sd / sqrt(n)

    return(ci)
}

# From the slides, unknown sigma
conf.int.3(5.4, 2.7, 36, 0.9, use.t = TRUE)
```

```
## [1] 4.639692 6.160308
```

```
# From the slides, known sigma
conf.int.3(66.3, 5.79, 40)
```

```
## [1] 64.50569 68.09431
```

Confidence intervals from samples

As one further enhancement, let's add the ability to handle samples, rather than just summary statistics. If sample size is not provided, we will treat `x` as a sample, rather than the point estimate. We will also, finally, properly comment our function.

```
# Function to calculate confidence intervals
conf.int.4 <- function(x, sd=0, n=0, level=0.95, use.t=FALSE, ...){
    # Define values to be used in calculation (may be changed)
    x.mean <- x
    x.sd <- sd
    x.n <- n

    # Alpha is 1 - confidence level
    alpha = 1 - level

    # If sample size is not provided, x is a sample
    if (n==0){
        x.mean <- mean(x,...)
        x.n <- length(x)

        # If sd is not provided, calculate from sample
        #  and force use of t critical values
        if (sd==0){
            x.sd <- sd(x, ...)
            use.t <- TRUE
        }
    }

    # Find proper critical value
    if (use.t){
        crit.value <- qt(alpha/2, df=x.n-1, lower=F)
    } else {
        crit.value <- qnorm(alpha/2, lower=F)
    }

    # Calculate confidence interval
    ci <- x.mean + c(-1,1) * crit.value * x.sd / sqrt(x.n)

    return(ci)
}

# Should work the same if n is provided
conf.int.4(5.4, 2.7, 36, 0.9, use.t = TRUE)
```

```
## [1] 4.639692 6.160308
```

```
# Confidence interval from a sample
x.sam <- rnorm(30, mean=20, sd=3)
```

```
conf.int.4(x.sam)
```

```
## [1] 19.03122 21.42426
```

Ellipsis parameter

You may have noticed the ellipsis (`...`) in the parameter list. This allows any named parameter to be passed to a function, which may be passed on to functions called within the function. For example, if we had missing values in our sample, we could include `na.rm=TRUE` as a parameter to our confidence interval function, which would then be passed to the `mean()` and `sd()` functions (because we included the ellipsis in the function calls) to allow proper calculations.

```
# Add some "missing values" to the sample
x.sam[c(12, 23)] <- NA

# We get an error, because mean and sd give errors with missing data
conf.int.4(x.sam)
```

```
## [1] NA NA
```

```
# Tell mean and sd to ignore missing values
conf.int.4(x.sam, na.rm=TRUE)
```

```
## [1] 18.79782 21.16878
```

Sample sizes

Creating a function to calculate sample sizes is left as an exercise for the reader.

Student's t distribution

The *t* distribution has similar functions as the other distributions, `rt()`, `pt()`, `qt()`, etc. The only parameter needed is degrees of freedom (df). Otherwise, they work identically to analogous functions for other distributions.

License

(cc) BY-NC-SA

This document is distributed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.