

STA 250: STATISTICS
Lab 2

In this lab, we will run the experiments on testing rules you saw in lecture yesterday.

Drug effectiveness study

We have $X = (X_1, \dots, X_n)$ giving the increase in sleep hours for $n = 10$ patients who took the drug. Our testing rule was to reject H_0 if $T(x) = \sqrt{n}\bar{x}/s_x > 1.83$. We will first see how to calculate

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \text{ and } s_x = \left\{ \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \right\}^{1/2}$$

in R .

Calculating mean and standard deviation

Direct calculation can be done by using the function `sum()` that adds elements of a vector.

```
> x <- c(11, 0, -4, -29, 13) ## to calculate x.bar and s.x for these numbers
> n <- length(x)           ## length() returns number of elements
> x.bar <- sum(x) / n
> x.bar
[1] -1.8
> x.res <- x - x.bar        ## calculate each element's distance from x.bar
> x.res
[1] 12.8  1.8 -2.2 -27.2 14.8
> s.x <- sqrt(sum(x.res^2) / (n - 1))
> s.x
[1] 16.81368
```

But you do not have to do all this every time you need \bar{x} and s_x . R provides two functions `mean()` and `sd()` to do just this.

TASK 1. For the above vector of elements `x`, apply `mean()` and `sd()` on `x` and check whether they output `x.bar` and `s.x` as calculated above.

Generating random numbers from $\text{Normal}(\mu, \sigma^2)$

Next we will learn how to generate random numbers from a normal distribution. R provides a function `rnorm()` to simulate normal random variables. A quick look up of the function via either of the following commands

```
> ?rnorm
> help(rnorm)
```

shows the help page for this function, which is stored by R as `rnorm(n, mean = 0, sd = 1)`. The first argument `n` denote how many random numbers are to be generated. The other two arguments denote the mean and standard deviation of the normal distribution to be used. The default values are `mean = 0` and `sd = 1`.

For example to generate 10 random numbers from `Normal(0,1)` we could write either of these two commands

```
> rnorm(10)
[1] -0.1471863 -0.8869886  2.2181323  0.2239207 -0.4955731  1.9865856  0.4325436
[8] -1.3657901 -2.0858798 -0.3146318
> rnorm(10, 0, 1)
[1] -2.4454686 -0.4944930 -1.3538821  0.6416000  0.1226671 -1.4563393 -0.9939484
[8] -0.7122929 -0.5542696 -1.6866314
```

(The two strings are different because they are random numbers. You would get yet another different string if you called `rnorm(10)` again.) To store the numbers you generate into a vector `x` you could try

```
> x <- rnorm(10, 0, 1)
```

TASK 2. Generate 10 random numbers from `Normal(0,32)` and calculate their mean and standard deviation. Next generate 10 random numbers from `Normal(1,32)` and calculate their mean and standard deviation.

Replication

The following function

```
x.bar <- function(n, mu = 0, sigma = 1){
  x <- rnorm(n, mu, sigma)
  return(mean(x))
}
```

generates `n` random numbers from a normal distribution with mean `mu` and standard deviation `sigma` and returns the average of those `n` numbers, e.g.,

```
> x.bar(10, 0, 3)
[1] 0.4344835
```

To repeat this 20 times we could use the `replicate()` function in R and run the following:

```
> replicate(20, x.bar(10, 0, 3))
[1]  1.8691460 -0.2570218 -2.2213199 -0.1514526  1.6736048  0.8720125 -1.0159190
[8] -1.7213040 -1.1497875  1.8988286 -0.6949375 -0.2125251 -0.1242079  0.3699864
[15]  0.1490083  1.0075756 -1.1081866 -0.2352321 -0.9239403 -1.4795186
```

In these 20 replications, each time a different string of 10 numbers were generated from $\text{Normal}(0, 3^2)$ and their average was reported in the corresponding element of the output vector.

TASK 3. Modify the function `x.bar()` to write a function `s.x()` that will output the standard deviation of `n` numbers randomly generated from the normal distribution with mean = `mu` and standard deviation = `sigma`. Replicate this function 20 times for `n = 10`, `mu = 0` and `sigma = 3`. Then replicate again for 20 times but now with `sigma = 30`. Does s_x represent σ well?

Testing rule

The following function defines our testing procedure

```
test.rule <- function(x.bar, s.x, n = 10, c = 1.83){
  T.x <- sqrt(n) * x.bar / s.x
  return(T.x > c)
}
```

It takes \bar{x} , s_x , n and cut-off c as inputs and produces a TRUE/FALSE summary of whether $T(x) = \sqrt{n}\bar{x}/s_x > c$.

TASK 4. For the actual data, $\bar{x} = 2.33$, $s_x = 2$ and $n = 10$. What does `test.rule()` say about rejecting H_0 for this data with cut-off = 1.83?

Experiments with testing rules

The following function generates n data points $x = (x_1, \dots, x_n)$ from any chosen $\text{Normal}(\mu, \sigma^2)$ distribution and returns the verdict of `test.rule()` applies to this data

```
test.expt <- function(n, mu, sigma, c = 1.83){
  x <- rnorm(n, mu, sigma)
  x.bar <- mean(x)
  s.x <- sd(x)
  return(test.rule(x.bar, s.x, n, c))
}
```

TASK 5. Replicate `test.expt(10, 0, 3, 1.83)` 100 times and report how many times you saw a TRUE.

TASK 6. Replicate `test.expt(10, 0, 3, 1.83)` 5000 times and store the results in a vector `out`. Use the `mean(out)` to calculate what proportion of elements of `out` are TRUE. This is the proportion of times the test incorrectly rejected $H_0 : \mu \leq 0$. Is this proportion close to 5%?

TASK 7. Now make 5000 repeats of `test.expt()` where the $n = 10$ observations are generated from $\text{Normal}(1, 3^2)$. What proportion of times did the test correctly reject $H_0 : \mu \leq 0$?

Effect of changing the cut-off

Let's try two other choices of c :

TASK 8. Now repeat both task 6 and task 7 but with $c = 200$. Then repeat both again but with $c = 0.01$. Circle your choices from the following table

c	Tendency to incorrectly reject H_0	Tendency to correctly reject H_0
1.83	Adequate/Too much	Adequate/Too little
200	Adequate/Too much	Adequate/Too little
0.01	Adequate/Too much	Adequate/Too little