LAB 4

STAT 131

Sep 17, 2021

Read in data.

```
craigslist <- read.csv("craigslist.csv",
                       header = TRUE, stringsAsFactors = FALSE)
one.bedrooms <- craigslist[craigslist$brs == 1, ]
```

Simulation: when will the t-test fail

Following four functions generate four types of sample (You can ignore the details concerning how the samples are generated.)

- **generate.sample1**: Small sample size (5) and normal distributed samples.
- **generate.sample2**: Small sample size (5) and non-normal distributed samples.
- **generate.sample3**: Large sample size (100) and non-normal distributed samples.
- **generate.sample4**: Large sample size (100), normal distirbuted but sample not independent.

```
generate.sample1 <- function(){
  sample <- rnorm(5)
  return(sample)
}

generate.sample2 <- function(){
  sample <- rgamma(5, 1.5, 2)
  return(sample)
}

generate.sample3 <- function(){
  sample <- rcauchy(100, 0, 1)
  return(sample)
}

generate.sample4 <- function(){
  sample <- arima.sim(n = 100, list(ar = c(1, -0.5), ma = c(0.3)))
  return(sample)
}
```

If you run the function, such as `generate.sample1`, it will generate the corresponding samples.

```
samples <- generate.sample1()
samples
```

```
## [1] -1.4811898 -0.5565788  0.8557158  0.8094467 -0.7422694
```

Exercise 1.

Please complete the function `simulation`. It accept the name of the above sample generation functions and return the estimated type I error. Please follow the following steps for completing the function `simulation`:

1. generate two samples from the function `FUN`.
2. calculate the p-value using t-test.
3. repeat 1 and 2 for N times.
4. calculate the chance of rejecting the null hypothesis $H_0 : \mu_1 = \mu_2$ when the significant level is set to 0.05. (that is, the estimated Type I Error) HINT: The chance of rejecting the null can be estimated by ((The number of rejections) / N) = ((The number of p values less or equal than 0.05) / N).

```
simulation <- function(FUN, N = 10000){
  # FUN: function name for sample generation
  # N: number of repetitions
  p_value  <- replicate(N, {sample1 <- FUN()
                            sample2 <- FUN()
                            t.test(sample1, sample2)$p.value})
  mean(p_value <= 0.05)
}
```

After complete your function above, you can run the following code:

```r
set.seed(123456) # please do not modify the seed
simulation(generate.sample1)
```

```
## [1] 0.0491
```

```r
set.seed(123456) # please do not modify the seed
simulation(generate.sample2)
```

```
## [1] 0.0309
```

```r
set.seed(123456) # please do not modify the seed
simulation(generate.sample3)
```

```
## [1] 0.0217
```

```r
set.seed(123456) # please do not modify the seed
simulation(generate.sample4)
```

```
## [1] 0.1663
```

Since the null hypothesis is true when we generate the data, the estimated Type I error should be close to 0.05 as we increase the number of repetitions. However, you may observe that t-test fails except for the first sample.

From the simulation, which test do you think is better for the Craigslist dataset? why?

```
Write your non-coding answer here.
```

Bonferroni correction

Exercise 2.

(a) Perform t-test on the mean price difference for each pairwise city combinations. Save the p-values in vector `p.values`. For which city pairs we can not reject the null hypothesis that mean rent price is equal? (above level 0.05) Again, you can use `apply` or for loops, but `apply` is recommended.

```r
cities <- unique(craigslist$location) # get the vector of unique city names
cities.combn <- combn(cities, 2) # this will give you all the possible combinations of two cities

# 'p.values'
p.values <- apply(cities.combn, 2, function(x){
  # The price vector for city 1 (x[1])
  city1data <- one.bedrooms[one.bedrooms$location == x[1], "price"]
  # The price vector for city 2 (x[2])
  city2data <- one.bedrooms[one.bedrooms$location == x[2], "price"]
  # return the p value of t test
  p <- t.test(city1data, city2data)$p.value
return(p)
})
```

```r
# You can not reject the null hypothesis that mean rent price for the following city pairs.
# (Subset the `cities.combn`.)
# save the your answer as
# 'not.reject'
not.reject <- cities.combn[, which(p.values > 0.05)]
```

(b) Perform Bonferroni corrections on the p-values. For which city pairs we can not reject the null hypothesis that mean rent price is equal? (under level 0.05)

```r
# insert code here save you answer as
# 'p.values.adj'
p.values.adj <- sapply(p.values, function(x) min(1, x*length(p.values)))
```

```r
# You can not reject the null hypothesis that mean rent price for the following city pairs.
# (Subset the `cities.combn`.)
# (After Bonferroni corrections)
# insert code here save you answer as
# 'not.reject.adj'
not.reject.adj <- cities.combn[, which(p.values.adj > 0.05)]
```

```r
p.values
```

```
##  [1] 5.542522e-08 7.136154e-11 4.812387e-04 9.027882e-03 8.132916e-06
##  [6] 1.676094e-03 6.676098e-01 4.094762e-02 1.946747e-08 1.948115e-02
## [11] 8.641691e-05 1.968442e-01 2.654283e-16 3.546448e-01 5.150725e-20
## [16] 3.785839e-12 1.163662e-07 9.123619e-23 2.861464e-18 1.213019e-05
## [21] 1.963060e-14 1.276837e-02 1.543595e-15 1.773589e-10 8.805651e-09
## [26] 3.899818e-20 4.099075e-14 1.637967e-09 9.020201e-02 5.218336e-11
## [31] 1.522581e-04 1.728144e-02 1.179865e-16 2.509165e-09 2.383418e-10
## [36] 5.530550e-01 2.923214e-04 1.225353e-07 3.265775e-04 6.940014e-01
## [41] 1.916160e-11 3.792002e-06 2.636801e-04 1.527701e-16 4.823605e-10
## [46] 1.961275e-05 1.704923e-09 2.095027e-03 3.166190e-01 2.237653e-02
## [51] 3.628235e-11 6.681972e-04 7.591992e-15 1.134889e-07 6.116978e-05
```

```r
not.reject
```

```
##      [,1]            [,2]       [,3]      [,4]       [,5]
## [1,] "berkeley"      "oakland"  "oakland" "richmond" "emeryville"
## [2,] "mountain view" "richmond" "alameda" "alameda"  "palo alto"
##      [,6]           [,7]
## [1,] "emeryville"   "palo alto"
## [2,] "redwood city" "redwood city"
```

```r
p.values.adj
```

```
##  [1] 3.048387e-06 3.924885e-09 2.646813e-02 4.965335e-01 4.473104e-04
##  [6] 9.218518e-02 1.000000e+00 1.000000e+00 1.070711e-06 1.000000e+00
## [11] 4.752930e-03 1.000000e+00 1.459855e-14 1.000000e+00 2.832899e-18
## [16] 2.082211e-10 6.400139e-06 5.017990e-21 1.573805e-16 6.671607e-04
## [21] 1.079683e-12 7.022604e-01 8.489772e-14 9.754738e-09 4.843108e-07
## [26] 2.144900e-18 2.254491e-12 9.008819e-08 1.000000e+00 2.870085e-09
## [31] 8.374193e-03 9.504794e-01 6.489257e-15 1.380041e-07 1.310880e-08
## [36] 1.000000e+00 1.607768e-02 6.739443e-06 1.796176e-02 1.000000e+00
## [41] 1.053888e-09 2.085601e-04 1.450240e-02 8.402353e-15 2.652983e-08
## [46] 1.078701e-03 9.377074e-08 1.152265e-01 1.000000e+00 1.000000e+00
## [51] 1.995529e-09 3.675085e-02 4.175596e-13 6.241892e-06 3.364338e-03
```

```r
not.reject.adj
```

```
##      [,1]         [,2]        [,3]            [,4]        [,5]
## [1,] "berkeley"   "berkeley"  "berkeley"      "berkeley"  "berkeley"
## [2,] "emeryville" "palo alto" "mountain view" "sunnyvale" "redwood city"
##      [,6]       [,7]      [,8]                  [,9]       [,10]
## [1,] "oakland"  "oakland" "albany / el cerrito" "richmond" "richmond"
## [2,] "richmond" "alameda" "alameda"             "alameda"  "sunnyvale"
##      [,11]        [,12]          [,13]        [,14]          [,15]
## [1,] "emeryville" "emeryville"   "palo alto"  "palo alto"    "mountain view"
## [2,] "palo alto"  "redwood city" "menlo park" "redwood city" "sunnyvale"
```