Crimtab Data (with noise) for Simulation of T-Distribution

데이터테크전공 20173204 곽명빈

2020-10-25

Data Loading

```
load("./crimtab_noise.RData")
ls()
```

```
## [1] "crimtab_2"            "crimtab_df"           "crimtab_long"
## [4] "crimtab_long_df"      "crimtab_long_df_noise" "r_noise"
```

```
ls.str()
```

```
## crimtab_2 :  'table' int [1:42, 1:22] 0 0 0 0 0 0 1 0 0 0 ...
## crimtab_df : 'data.frame':   924 obs. of  3 variables:
##  $ finger: num  9.4 9.5 9.6 9.7 9.8 9.9 10 10.1 10.2 10.3 ...
##  $ height: num  56 56 56 56 56 56 56 56 56 56 ...
##  $ Freq  : int  0 0 0 0 0 0 1 0 0 0 ...
## crimtab_long :  num [1:3000, 1:2] 10 10.3 9.9 10.2 10.2 10.3 10.4 10.7 10 10.1 ...
## crimtab_long_df : 'data.frame':  3000 obs. of  2 variables:
##  $ finger: num  10 10.3 9.9 10.2 10.2 10.3 10.4 10.7 10 10.1 ...
##  $ height: num  56 57 58 58 58 58 58 58 59 59 ...
## crimtab_long_df_noise : 'data.frame':    3000 obs. of  2 variables:
##  $ finger: num  9.98 10.29 9.91 10.24 10.17 ...
##  $ height: num  55.8 56.9 58.1 58.4 57.7 ...
## r_noise :  num [1:3000] -0.2345 -0.1279 0.0729 0.4082 -0.2983 ...
```

```
head(crimtab_long_df_noise,
     n = 10)
```

```
##       finger   height
## 1   9.976551 55.76551
## 2  10.287212 56.87212
## 3   9.907285 58.07285
## 4  10.240821 58.40821
## 5  10.170168 57.70168
## 6  10.339839 58.39839
## 7  10.444468 58.44468
## 8  10.716080 58.16080
## 9  10.012911 59.12911
## 10 10.056179 58.56179
```

Student 의 Simulation 재현

Sample t-values

3,000장의 카드를 잘 섞는 것은 `sample()` 이용.

```
# set.seed(113)
crimtab_shuffle_noise <-
   crimtab_long_df_noise[sample(1:3000), ]
head(crimtab_shuffle_noise,
     n = 10)
```

```
##        finger   height
## 2753 12.13674 69.36744
## 1451 11.90282 65.02817
## 114  10.79315 60.93147
## 253  11.09707 61.97068
## 2782 12.28416 68.84159
## 1150 11.16040 64.60402
## 1425 11.79061 64.90610
## 716  10.89315 63.93146
## 82   10.22125 61.21251
## 2170 11.66442 66.64423
```

표본의 크기가 4인 750개의 표본을 만드는 작업은 `rep()` 이용.

```
sample_id <- as.factor(rep(1:750, each = 4))
head(sample_id, n = 10)
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3
## 750 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 750
```

각 표본의 평균과 표준편차 계산에는 `tapply()` 이용.

```
finger_sample_mean <- tapply(crimtab_shuffle_noise[, "finger"],
                             INDEX = sample_id,
                             FUN = mean)
finger_sample_sd <- tapply(crimtab_shuffle_noise[, "finger"],
                           INDEX = sample_id,
                           FUN = sd)
str(finger_sample_mean)
```

```
##  num [1:750(1d)] 11.5 11.5 10.9 12.1 11.9 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ : chr [1:750] "1" "2" "3" "4" ...
```

```
str(finger_sample_sd)
```

```
##  num [1:750(1d)] 0.64 0.627 0.853 0.63 0.481 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ : chr [1:750] "1" "2" "3" "4" ...
```

t-통계량 계산. Student는 표준편차 계산에서 분모에 n을 사용하고 히스토그램을 그려 비교하였으나 자유도 3인 t-분포와 비교하기 위하여 $t = \frac{\bar{X}_n - \mu}{\hat{SD}/\sqrt{n}}$을 계산함. (여기서 $\hat{SD}$는 표본 표준편차)

```
sample_t <- (finger_sample_mean - mean(crimtab_long_df_noise[, "finger"])) / (finger_sample_sd/
sqrt(4))
str(sample_t)
```

```
##  num [1:750(1d)] -0.2008 -0.0466 -1.4396 1.746 1.3033 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ : chr [1:750] "1" "2" "3" "4" ...
```

계산한 t-통계량 값들의 평균과 표준편차, 히스토그램을 그리고 자유도 3인 t-분포의 밀도함수 및 표준정규곡선과 비교. 우선 모두 같은 값들이 나와서 분모가 0인 경우가 있는지 파악. 있으면 모평균과 비교하여 양수인 경우 +6, 음수인 경우 -6 값 부여(Student가 한 일)

```
t_inf <- is.infinite(sample_t)
sample_t[t_inf]
```

```
## named numeric(0)
```

```
sample_t[t_inf] <- 6 * sign(sample_t[t_inf])
```

문제되는 값이 없는 것을 확인하고, 평균과 표준편차 계산. 자유도 n인 t-분포의 평균과 표준편차는 각각 0과 $\sqrt{\frac{n}{n-2}}$임을 상기할 것. -6이나 +6보다 큰 값이 상당히 자주 나온다는 점에 유의.

```
mean(sample_t)
```

```
## [1] -0.00135926
```

```
sd(sample_t)
```

```
## [1] 1.566195
```

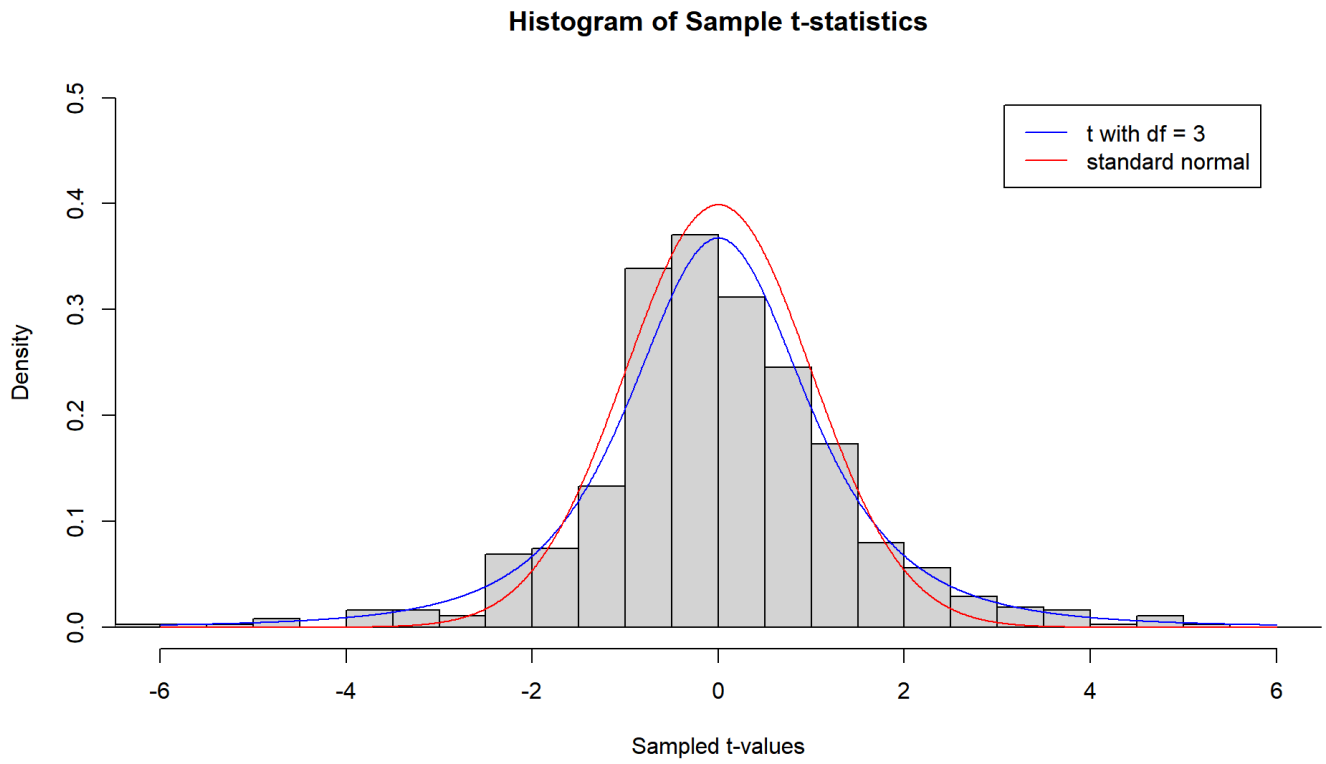
```
summary(sample_t)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -15.474107  -0.727005  -0.060401  -0.001359   0.766502   9.357463
```

Histogram and Density Curves

t-통계량들의 히스토그램을 그리고, 자유도 3인 t의 밀도함수, 표준정규분포 밀도함수와 비교.

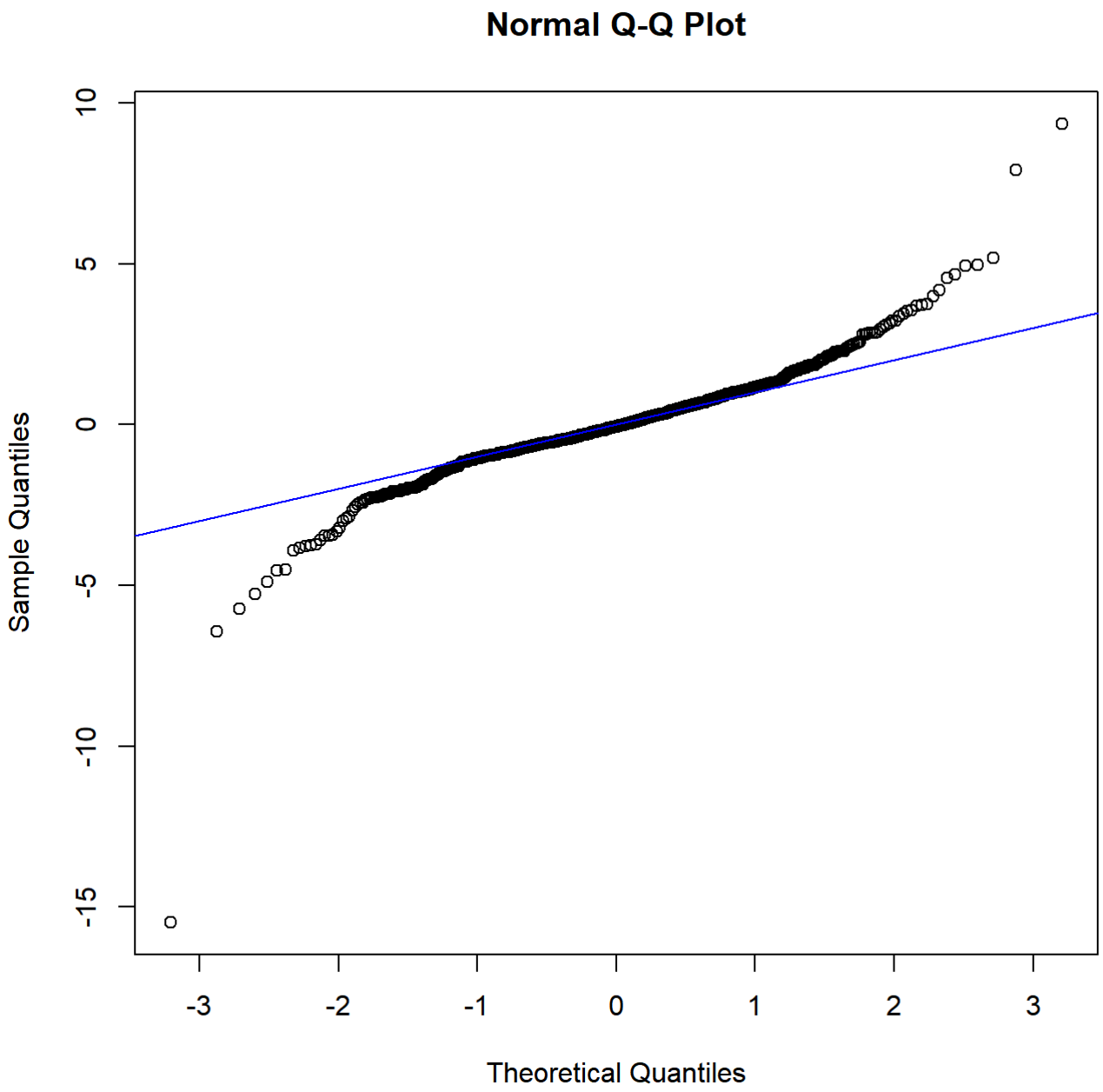
```
# hist(sample_t, prob = TRUE, ylim = c(0, 0.5))
# hist(sample_t, prob = TRUE, nclass = 20, xlim = c(-6, 6), ylim = c(0, 0.5), main = "Histogram
of Sample t-statistics", xlab = "Sampled t-values")
# hist(sample_t, prob = TRUE, nclass = 50, xlim = c(-6, 6), ylim = c(0, 0.5), main = "Histogram
of Sample t-statistics", xlab = "Sampled t-values")
hist(sample_t,
     prob = TRUE,
     breaks = seq(-20, 20, by = 0.5),
     xlim = c(-6, 6),
     ylim = c(0, 0.5),
     main = "Histogram of Sample t-statistics",
     xlab = "Sampled t-values")
lines(seq(-6, 6, by = 0.01),
      dt(seq(-6, 6, by = 0.01), df = 3),
      col = "blue")
lines(seq(-6, 6, by = 0.01),
      dnorm(seq(-6, 6, by = 0.01)),
      col = "red")
legend("topright",
       inset = 0.05,
       legend = c("t with df = 3", "standard normal"),
       lty = 1,
       col = c("blue", "red"))
```



QQnorm

`qqnorm()` 을 그려보면 정규분포와 꼬리에서 큰 차이가 난다는 것을 알 수 있음.

```
qqnorm(sample_t)
abline(a = 0, b = 1, col = "blue")
```



Comments

이번 시간에는 t-test에대해 알아보게 되었습니다. t-test는 두개가 유의한 차이가 있는지에 대한 검사란 것을자료를 통해 알게 되었습니다.student선생님의 t-test과정을 배우게 되었습니다. 모집단에서 표본이 작을 경우 모집단의 분포를 알고 표준편차를 모를경우 t- test를 사용한다는 것을 알게되었습니다. 3천개의 데이터가 정규분포에 가까운걸 알고 있으니 4개씩 묶어서 750개의 샘플로 만들어 자유도가 3인 t분포와 닮은지 알아보는 과정들이였습니다.sample을통해 3000개의 데이터를 섞은후, rep를 이용하여 750개의 표본을 만들수 있었습니다. tapply를 이용하여 평균과 표준편차를 계산하여 t-통계량을 계산할수 있게 되었습니다. 히스토그램을 그려본 결과 꼬리부분의 차이가 있다는것을 눈으로 확인할 수 있었습니다.