Bios 6301: Assignment 3

Yiqing Pan

50/50

Due Tuesday, 24 September, 1:00 PM

50 points total.

Add your name as `author` to the file's metadata section.

Submit a single quarto file (named `homework3.qmd`) by email to huiding.chen@vanderbilt.edu. Place your R code in between the appropriate chunks for each question. Check your output by using the `Render` button in RStudio.

$5^{n=day}$ points taken off for each day late.

```
library(MASS)
```

Question 1

15 points

Write a simulation to calculate the power for the following study design. The study has two variables, treatment group and outcome. There are two treatment groups (0, 1) and they should be assigned randomly with equal probability. The outcome should be a random normal variable with a mean of 60 and standard deviation of 20. If a patient is in the treatment group, add 5 to the outcome. 5 is the true treatment effect. Create a linear model for the outcome by the treatment group, and extract the p-value (hint: see assigment1). Test if the p-value is less than or equal to the alpha level, which should be set to 0.05.

Repeat this procedure 1000 times. The power is calculated by finding the percentage of times the p-value is less than or equal to the alpha level. Use the `set.seed` command so that the professor can reproduce your results.

1. Find the power when the sample size is 100 patients. (10 points)

```
set.seed(123)

sims = 1000;
mu = 60; sd = 20; treatment_effect = 5; sample_N = 100
a_level = 0.05

p_values <- NULL

for (i in (1:sims)) {
  treatment_group <- sample(c(0, 1), size = sample_N, replace = TRUE)
  outcome <- rnorm(sample_N, mu, sd)
  treatment_group = treatment_group*5
  outcome_adj = outcome + treatment_group
  treatment_group_category = ifelse(treatment_group==0, "0", "1")
```

```
  lm <- lm(outcome_adj ~ treatment_group_category)
  p <- coef(summary(lm))[, "Pr(>|t|)"]
  p_values[i] = p[2]
}

power = sum(p_values <= a_level)/ length(p_values)
print(power)
```

```
## [1] 0.241
```

1. Find the power when the sample size is 1000 patients. (5 points)

```
set.seed(123)

sims = 1000;
mu = 60; sd = 20; treatment_effect = 5; sample_N = 1000
a_level = 0.05

p_values <- NULL

for (i in (1:sims)) {
  treatment_group <- sample(c(0, 1), size = sample_N, replace = TRUE)
  outcome <- rnorm(sample_N, mu, sd)
  treatment_group = treatment_group*5
  outcome_adj = outcome + treatment_group
  treatment_group_category = ifelse(treatment_group==0, "0", "1")
  lm <- lm(outcome_adj ~ treatment_group_category)
  p <- coef(summary(lm))[, "Pr(>|t|)"]
  p_values[i] = p[2]
}

power = sum(p_values <= a_level)/ length(p_values)
print(power)
```

```
## [1] 0.973
```

Question 2

14 points

Obtain a copy of the football-values lecture. Save the `2024/proj_wr24.csv` file in your working directory. Read in the data set and remove the first two columns.

1. Show the correlation matrix of this data set. (4 points)

```
wr24 <- read.csv("proj_wr24.csv")
wr24 <- wr24[, -c(1:2)]
head(wr24)
```

```
##   rec_att rec_yds rec_tds rush_att rush_yds rush_tds fumbles  fpts
## 1   116.1  1488.5     9.7      9.8     68.2      0.8     1.1 216.6
```

```
## 2   110.2  1541.6     9.8      5.2     30.5      0.1     1.0 215.0
## 3   106.6  1358.0     9.1      3.0     12.6      0.1     0.7 190.7
## 4   104.8  1443.2     7.4      1.2      3.8      0.0     1.0 187.0
## 5   111.0  1354.4     8.2      4.5     31.1      0.1     0.9 186.4
## 6    94.1  1338.8     8.4      0.0      1.4      0.0     1.3 181.5
```

```
cor_matrix <- cor(wr24)
print(cor_matrix)
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9923777 0.9699488 0.3423003 0.3374507 0.2385212 0.8851834
## rec_yds  0.9923777 1.0000000 0.9802932 0.3333627 0.3263447 0.2302912 0.8846812
## rec_tds  0.9699488 0.9802932 1.0000000 0.3379421 0.3320431 0.2436018 0.8636770
## rush_att 0.3423003 0.3333627 0.3379421 1.0000000 0.9810253 0.8873316 0.3390530
## rush_yds 0.3374507 0.3263447 0.3320431 0.9810253 1.0000000 0.8990693 0.3278916
## rush_tds 0.2385212 0.2302912 0.2436018 0.8873316 0.8990693 1.0000000 0.2417073
## fumbles  0.8851834 0.8846812 0.8636770 0.3390530 0.3278916 0.2417073 1.0000000
## fpts     0.9890267 0.9968262 0.9887441 0.3839153 0.3783087 0.2850410 0.8804361
##              fpts
## rec_att  0.9890267
## rec_yds  0.9968262
## rec_tds  0.9887441
## rush_att 0.3839153
## rush_yds 0.3783087
## rush_tds 0.2850410
## fumbles  0.8804361
## fpts     1.0000000
```

1. Generate a data set with 30 rows that has a similar correlation structure. Repeat the procedure 1,000 times and return the mean correlation matrix. (10 points)

```
set.seed(123)
sims= 1000

generate_cor <- function(row, cor_matrix) {

  means <- rep(0, ncol(cor_matrix))
  simulated_data <- mvrnorm(n = row, mu = means, Sigma = cor_matrix)

  return(cor(simulated_data))
}

cor_matrices <- array(0, dim = c(ncol(cor_matrix), ncol(cor_matrix), sims))

for (i in (1:sims)) {
  cor_matrices[,,i] <- generate_cor(30, cor_matrix)
}

mean_cor_matrix <- apply(cor_matrices, c(1, 2), mean)
print(mean_cor_matrix)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
```

```
## [1,] 1.0000000 0.9921527 0.9692202 0.3460269 0.3403103 0.2425789 0.8815099
## [2,] 0.9921527 1.0000000 0.9798186 0.3374728 0.3296575 0.2344578 0.8811100
## [3,] 0.9692202 0.9798186 1.0000000 0.3419375 0.3356245 0.2482971 0.8595899
## [4,] 0.3460269 0.3374728 0.3419375 1.0000000 0.9804664 0.8842763 0.3427215
## [5,] 0.3403103 0.3296575 0.3356245 0.9804664 1.0000000 0.8962822 0.3309604
## [6,] 0.2425789 0.2344578 0.2482971 0.8842763 0.8962822 1.0000000 0.2460061
## [7,] 0.8815099 0.8811100 0.8595899 0.3427215 0.3309604 0.2460061 1.0000000
## [8,] 0.9887112 0.9967289 0.9884771 0.3872138 0.3809234 0.2884838 0.8768172
##           [,8]
## [1,] 0.9887112
## [2,] 0.9967289
## [3,] 0.9884771
## [4,] 0.3872138
## [5,] 0.3809234
## [6,] 0.2884838
## [7,] 0.8768172
## [8,] 1.0000000
```

Question 3

21 points

Here's some code:

```
nDist <- function(n = 100) {
    df <- 10
    prob <- 1/3
    shape <- 1
    size <- 16
    list(
        beta = rbeta(n, shape1 = 5, shape2 = 45),
        binomial = rbinom(n, size, prob),
        chisquared = rchisq(n, df),
        exponential = rexp(n),
        f = rf(n, df1 = 11, df2 = 17),
        gamma = rgamma(n, shape),
        geometric = rgeom(n, prob),
        hypergeometric = rhyper(n, m = 50, n = 100, k = 8),
        lognormal = rlnorm(n),
        negbinomial = rnbinom(n, size, prob),
        normal = rnorm(n),
        poisson = rpois(n, lambda = 25),
        t = rt(n, df),
        uniform = runif(n),
        weibull = rweibull(n, shape)
    )
}
```

1. What does this do? (3 points)

```
round(sapply(nDist(500), mean), 2)
```

```
##           beta       binomial     chisquared    exponential              f
```

```
##          0.10          5.21         10.19          1.07          1.16
##         gamma     geometric hypergeometric     lognormal    negbinomial
##          1.01          2.13          2.66          1.67         31.29
##        normal       poisson             t       uniform       weibull
##         -0.04         25.14          0.08          0.50          0.98
```

This code calls the defined function "nDist" with n=500 to generate 500 values for each distribution defned in the "nDist". Then, sapply is used to apply the mean() function to each distribution in the list returned by nDist(500). And finally, with round(), the mean of each distribution is rounded to 2 decimal points.

2. What about this? (3 points)

```
sort(apply(replicate(20, round(sapply(nDist(10000), mean), 2)), 1, sd))
```

```
##          beta       uniform        normal             f       weibull
##  0.000000000  0.002236068  0.006958524  0.009514532  0.009665457
##         gamma             t hypergeometric   exponential      binomial
##  0.012937095  0.012977714  0.013018206  0.013992479  0.021275140
##     lognormal     geometric       poisson    chisquared    negbinomial
##  0.027453310  0.033320612  0.044777226  0.046498444  0.107918098
```

The round(sapply(nDist(10000), mean), 2) produces the 2 decimal points rounded mean of each distribution specified in nDist, generated with 10000 values. Then, this process is repeated 20 times to get 20 rounded means for each distribution. apply() is then used to calculates the standard deviation across the 20 replications for each distribution means(1 means apply the function row-wise). Finally, sort is used to sort the standard deviation results ascendingly.

In the output above, a small value would indicate that `N=10,000` would provide a sufficent sample size as to estimate the mean of the distribution. Let's say that a value *less than 0.02* is "close enough".

3. For each distribution, estimate the sample size required to simulate the distribution's mean. (15 points)

Don't worry about being exact. It should already be clear that N < 10,000 for many of the distributions. You don't have to show your work. Put your answer to the right of the vertical bars (|) below.

distribution	N
beta	10
binomial	9000
chisquared	55000
exponential	2500
f	1250
gamma	2500
geometric	15000
hypergeometric	5000
lognormal	10000
negbinomial	30000
normal	2500
poisson	55000
t	4000
uniform	200
weibull	2500