Bios 6301: Assignment 3

Jonathan Lifferth

Due Tuesday, 26 September, 1:00 PM

50 points total.

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

Submit a single knitr file (named `homework3.rmd`) by email to marisa.h.blackman@vanderbilt.edu. Place your R code in between the appropriate chunks for each question. Check your output by using the `Knit HTML` button in RStudio.

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

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 assignment1). 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)

```
library(MASS)
?mvrnorm()
set.seed(1234)
# sample size for each of the two categories (so this number is doubled in the simulation)
sample_size <- 50
alpha <- 0.05
below_alpha <- list()

# I know that it would be better to also incorporate a binomial function to randomly determine the trea
for (i in seq(1:1000)) {
    df0 <- data.frame(value = rnorm(sample_size, mean = 60, sd = 20), treatment = rep(0, sample_size))
    df1 <- data.frame(value = rnorm(sample_size, mean = 60, sd = 20) + 5, treatment = rep(1, sample_siz

    df_combined <- rbind(df0, df1)
    model <- lm(value ~ treatment, dat=df_combined)
    below_alpha[i] <- summary(model)$coefficients[,4][2] < alpha
}
```

```r
below_alpha_unlisted <- unlist(below_alpha)
power = sum(below_alpha_unlisted) / length(below_alpha_unlisted)
power
```

```
## [1] 0.252
```

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

```r
set.seed(1234)
# sample size for each of the two categories (so this number is doubled in the simulation)
sample_size <- 500
alpha <- 0.05
below_alpha <- list()


# I know that it would be better to also incorporate a binomial function to randomly determine the trea
for (i in seq(1:1000)) {
    df0 <- data.frame(value = rnorm(sample_size, mean = 60, sd = 20), treatment = rep(0, sample_size))
    df1 <- data.frame(value = rnorm(sample_size, mean = 60, sd = 20) + 5, treatment = rep(1, sample_siz

    df_combined <- rbind(df0, df1)
    model <- lm(value ~ treatment, dat=df_combined)
    below_alpha[i] <- summary(model)$coefficients[,4][2] < alpha
}

below_alpha_unlisted <- unlist(below_alpha)
power = sum(below_alpha_unlisted) / length(below_alpha_unlisted)
power
```

```
## [1] 0.982
```

Question 2

14 points

Obtain a copy of the football-values lecture. Save the `2023/proj_wr23.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)

```r
football_df <- read.csv('proj_wr23.csv')
football_df <- subset(football_df, select = -c(PlayerName, Team))
football_correlation <- cor(football_df)
football_correlation
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9751004 0.9632402 0.3472074 0.3268797 0.2727289 0.6500961
## rec_yds  0.9751004 1.0000000 0.9690563 0.3275376 0.3063227 0.2651483 0.6464774
## rec_tds  0.9632402 0.9690563 1.0000000 0.2656168 0.2542211 0.2277144 0.6148910
## rush_att 0.3472074 0.3275376 0.2656168 1.0000000 0.9527671 0.8333155 0.3881654
```

```
## rush_yds 0.3268797 0.3063227 0.2542211 0.9527671 1.0000000 0.8994103 0.3619078
## rush_tds 0.2727289 0.2651483 0.2277144 0.8333155 0.8994103 1.0000000 0.3048482
## fumbles  0.6500961 0.6464774 0.6148910 0.3881654 0.3619078 0.3048482 1.0000000
## fpts     0.9777955 0.9961858 0.9791021 0.3704589 0.3565869 0.3183841 0.6445244
##               fpts
## rec_att  0.9777955
## rec_yds  0.9961858
## rec_tds  0.9791021
## rush_att 0.3704589
## rush_yds 0.3565869
## rush_tds 0.3183841
## fumbles  0.6445244
## 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)

```
vcov.football=var(football_df)
means.football=colMeans(football_df)

correlation_matrices <- list()

for (i in (1:1000)) {
   football.sim = mvrnorm(30, mu = means.football, Sigma = vcov.football)
   football.sim = as.data.frame(football.sim)
   rho.sim=cor(football.sim)   ## Simulated correlation matrix
   correlation_matrices[[i]] <- rho.sim
}

mean_matrix <- matrix(0, nrow = nrow(football_correlation), ncol=ncol(football_correlation))
for (matrix_ in correlation_matrices) {
    mean_matrix <- mean_matrix + matrix_
}

mean_matrix <- mean_matrix / length(correlation_matrices)
mean_matrix
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9744290 0.9623343 0.3400570 0.3193799 0.2659945 0.6447610
## rec_yds  0.9744290 1.0000000 0.9684401 0.3202136 0.2992387 0.2590554 0.6429468
## rec_tds  0.9623343 0.9684401 1.0000000 0.2610690 0.2497338 0.2230735 0.6106926
## rush_att 0.3400570 0.3202136 0.2610690 1.0000000 0.9511628 0.8320070 0.3821362
## rush_yds 0.3193799 0.2992387 0.2497338 0.9511628 1.0000000 0.8983688 0.3558863
## rush_tds 0.2659945 0.2590554 0.2230735 0.8320070 0.8983688 1.0000000 0.2986598
## fumbles  0.6447610 0.6429468 0.6106926 0.3821362 0.3558863 0.2986598 1.0000000
## fpts     0.9771365 0.9960555 0.9787362 0.3629977 0.3491859 0.3116957 0.6408016
##               fpts
## rec_att  0.9771365
## rec_yds  0.9960555
## rec_tds  0.9787362
## rush_att 0.3629977
## rush_yds 0.3491859
## rush_tds 0.3116957
```

```
## fumbles  0.6408016
## fpts     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           5.13          10.15           0.97           1.18
##          gamma      geometric hypergeometric      lognormal    negbinomial
##           1.11           1.87           2.76           1.77          32.47
##         normal        poisson              t        uniform        weibull
##           0.04          25.39           0.01           0.50           0.99
```

answer here
The nDist function takes a number "n" and calculates a series of summary statistics for various types of distributions of size "n". The sapply((_, mean), 2) then takes the mean of the values from each distribution.

2. What about this? (3 points)

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

```
##           beta        uniform              f        weibull              t
##    0.000000000    0.003940345    0.006069770    0.008271702    0.009445132
##          gamma    exponential         normal hypergeometric      lognormal
##    0.009679060    0.010208356    0.010259784    0.013138934    0.020774478
##       binomial      geometric     chisquared        poisson    negbinomial
##    0.025833475    0.029607076    0.054990430    0.063401395    0.086284413
```

```
answer here
Here, the mean for each distribution is calculated (as above) but the results are rounded (to the r
```

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)

```
apply(replicate(20, round(sapply(nDist(2000), mean), 2)), 1, sd)
```

```
##           beta       binomial     chisquared    exponential              f
##    0.000000000    0.031805329    0.082812407    0.023946212    0.015078741
##          gamma      geometric hypergeometric      lognormal    negbinomial
##    0.023708759    0.060164249    0.032163235    0.053160630    0.224271335
##         normal        poisson              t        uniform        weibull
##    0.017947291    0.156819541    0.028172402    0.005619515    0.021878853
```

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	6
binomial	10000
chisquared	50000
exponential	4000
f	1300
gamma	2000
geometric	10000
hypergeometric	3800
lognormal	10000
negbinomial	30000
normal	3100
poisson	59000
t	4700
uniform	300
weibull	2000