Bios 6301: Assignment 3

Lydia Yao

Due Tuesday, 28 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 michael.l.williams@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 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 the seed for predictability
set.seed(194842)
# Define alpha
alpha = 0.05
# Count the number of times the p-value is less than or equal to alpha
count = 0
# Define number of simulations
nsim = 1000
# Loop nsim times
for (k in 1:nsim){
    # Randomly select treatment and outcome
    treatment <- sample(c(0,1), replace=TRUE, size=100)
    outcome <- rnorm(100, 60, 20)
    d.frame = data.frame(treatment,outcome)
    colnames(d.frame) = c("treatment","outcome")
    # Add 5 to treatment group
    for (i in 1:nrow(d.frame)){
        if (d.frame[i,'treatment'] == 1) {
            d.frame[i, 'outcome'] = d.frame[i, 'outcome'] + 5
```

```
        }
    }
    # Create model and calculate p-value
    mod <- lm(outcome ~ treatment, dat=d.frame)
    mod.test <- t.test(outcome ~ treatment, dat=d.frame, var.equal=TRUE)
    # See if our p-value is less than or equal to alpha
    if (mod.test$p.value <= alpha){
        count = count + 1
    }
}
# Calculate Power
power = count / 1000
power
```

```
## [1] 0.219
```

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

```
# Define alpha
alpha = 0.05
# Count the number of times the p-value is less than or equal to alpha
count = 0
# Define number of simulations
nsim = 1000
# Loop nsim times
for (k in 1:nsim){
    # Randomly select treatment and outcome
    treatment <- sample(c(0,1), replace=TRUE, size=1000)
    outcome <- rnorm(1000, 60, 20)
    d.frame = data.frame(treatment,outcome)
    colnames(d.frame) = c("treatment","outcome")
    # Add 5 to treatment group
    for (i in 1:nrow(d.frame)){
        if (d.frame[i,'treatment'] == 1) {
            d.frame[i, 'outcome'] = d.frame[i, 'outcome'] + 5
        }
    }
    # Create model and calculate p-value
    mod <- lm(outcome ~ treatment, dat=d.frame)
    mod.test <- t.test(outcome ~ treatment, dat=d.frame, var.equal=TRUE)
    # See if our p-value is less than or equal to alpha
    if (mod.test$p.value <= alpha){
        count = count + 1
    }
}
# Calculate Power
power = count / 1000
power
```

```
## [1] 0.981
```

Question 2

14 points

Obtain a copy of the football-values lecture. Save the `2021/proj_wr21.csv` file in your working directory.

Read in the data set and remove the first two columns.

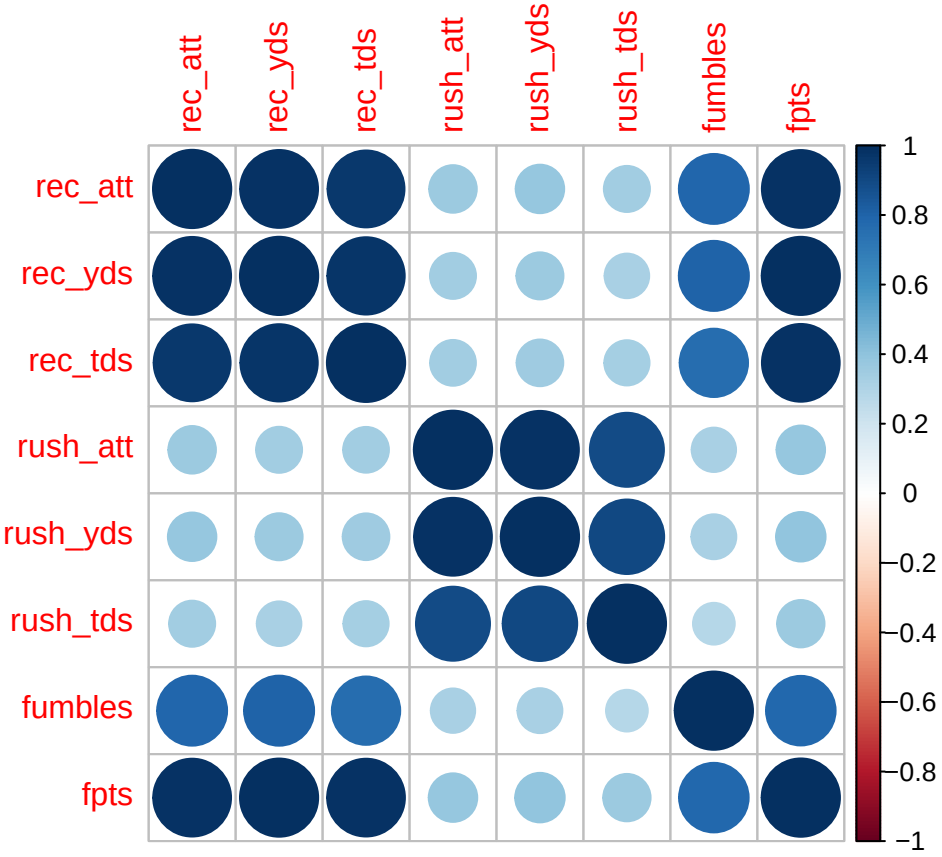
```
# Read data in
football <- read.csv("https://github.com/couthcommander/football-values/raw/main/2021/proj_wr21.csv", h
football <- subset(football, select = -c(1, 2))
```

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

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
# Get correlation plot from data and plot it
football.cor = cor(football, method = c("pearson"))
corrplot(football.cor)
```



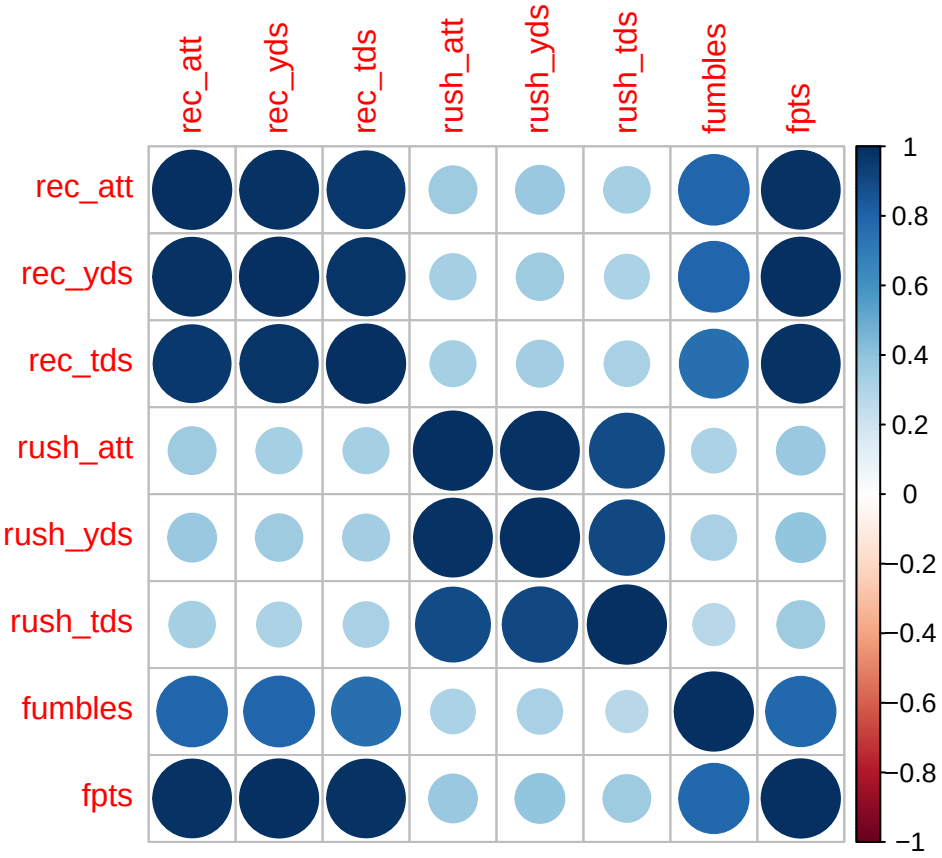

```
football.cor
```

```
##             rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att   1.0000000 0.9899611 0.9650160 0.3690670 0.3834924 0.3463555 0.7981497
## rec_yds   0.9899611 1.0000000 0.9746951 0.3452096 0.3611319 0.3244833 0.8011127
## rec_tds   0.9650160 0.9746951 1.0000000 0.3418033 0.3554974 0.3335733 0.7622937
## rush_att  0.3690670 0.3452096 0.3418033 1.0000000 0.9882542 0.8944610 0.3212985
## rush_yds  0.3834924 0.3611319 0.3554974 0.9882542 1.0000000 0.9055524 0.3290909
## rush_tds  0.3463555 0.3244833 0.3335733 0.8944610 0.9055524 1.0000000 0.2843320
## fumbles   0.7981497 0.8011127 0.7622937 0.3212985 0.3290909 0.2843320 1.0000000
## fpts      0.9879394 0.9968696 0.9864975 0.3839939 0.3997444 0.3660350 0.7899300
##                fpts
## rec_att   0.9879394
## rec_yds   0.9968696
```

```
## rec_tds  0.9864975
## rush_att 0.3839939
## rush_yds 0.3997444
## rush_tds 0.3660350
## fumbles  0.7899300
## 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)

```
library(MASS)
# Define initial resulting matrix of zeros
result <- matrix(0, 8, 8)
# Define our correlation structure that we want to imitate
sigma <- football.cor
mu <- colMeans(football)
# Loop through our simulations
for (i in 1:1000) {
    # Create 30 rows of sample
    sample <- mvrnorm(n=30, mu = mu, Sigma = sigma)
    # Get correlation and append to our result
    sample.cor = cor(sample, method = c("pearson"))
    result <- result + sample.cor
}
# Find mean of result and return
result = result/1000
corrplot(result)
```



```
result
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9896819 0.9643160 0.3591907 0.3748250 0.3392730 0.7916637
## rec_yds  0.9896819 1.0000000 0.9741429 0.3351173 0.3520972 0.3169393 0.7946523
## rec_tds  0.9643160 0.9741429 1.0000000 0.3315754 0.3461032 0.3247532 0.7552533
## rush_att 0.3591907 0.3351173 0.3315754 1.0000000 0.9878487 0.8919146 0.3138087
## rush_yds 0.3748250 0.3520972 0.3461032 0.9878487 1.0000000 0.9033177 0.3230552
## rush_tds 0.3392730 0.3169393 0.3247532 0.8919146 0.9033177 1.0000000 0.2772879
## fumbles  0.7916637 0.7946523 0.7552533 0.3138087 0.3230552 0.2772879 1.0000000
## fpts     0.9876783 0.9967931 0.9861594 0.3735342 0.3902763 0.3577869 0.7832390
##               fpts
## rec_att  0.9876783
## rec_yds  0.9967931
## rec_tds  0.9861594
## rush_att 0.3735342
## rush_yds 0.3902763
## rush_tds 0.3577869
## fumbles  0.7832390
## 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          5.36          9.62          1.11          1.12
##         gamma     geometric hypergeometric     lognormal    negbinomial
##          0.99          2.06          2.72          1.53         32.79
##        normal       poisson             t       uniform       weibull
##         -0.07         25.16          0.00          0.51          0.93
```

Here we are first calling the nDist(500) function to get a matrix of 500 results each of different

2. What about this? (3 points)

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

```
##          beta       uniform             f        normal         gamma
##  0.000000000  0.002236068  0.007591547  0.007880689  0.008645047
##             t   exponential       weibull hypergeometric      binomial
##  0.009333020  0.012343760  0.012814466  0.015719582  0.019432745
##    lognormal    geometric    chisquared      poisson   negbinomial
##  0.022594829  0.026635947  0.038865490  0.042289728  0.141182972
```

We are now sampling 10000 per distribution when we call the nDist function. Similar to the previou

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)

```
n <- 0
hold <- rep(NA,15)
s <- 1

while(TRUE){
    n = n + 10
    s <- apply(replicate(20,sapply(nDist(n), mean)), 1, sd)
    s2 <- which(s < 0.02)

    for (i in s2) {
        if (is.na(hold[i])){
            hold[i] = n
        }
    }
    if (sum(is.na(hold)) == 0){
        break
    }
}
```

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	2880
chisquared	19490
exponential	1080
f	540
gamma	1140
geometric	6060
hypergeometric	2250
lognormal	4580

distribution	N
negbinomial	206000
normal	1630
poisson	55010
t	1030
uniform	190
weibull	1130