

# BIOS 6301: Assignment 3

Charles Rhea

*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.

$5^{n=\text{day}}$  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.

*The p-value is greater than 0.05*

```
n <- 25
treatment <- rbinom(n, size = 1, prob = 0.5)
outcome <- rnorm(n, 60, 20)
outcome_2 <- ifelse(treatment == 1, outcome+5, outcome)

model <- lm(outcome_2 ~ treatment)
summary(model)
```

```
##
## Call:
## lm(formula = outcome_2 ~ treatment)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -32.709  -5.969  -1.151   5.427  43.799
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   57.789     4.524  12.775 6.28e-12 ***
## treatment      8.595     6.529   1.316   0.201
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.31 on 23 degrees of freedom
## Multiple R-squared:  0.07006,    Adjusted R-squared:  0.02963
## F-statistic: 1.733 on 1 and 23 DF,  p-value: 0.201
```

```
model.c <- coef(summary(model))
p <- (pt(model.c[2,3],8,lower.tail = FALSE))*2
p
```

```
## [1] 0.2245156
```

```
p < 0.05
```

```
## [1] FALSE
```

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(1)

n <- 100
mean(replicate(1000, {
  treatment <- rbinom(n, size = 1, prob = 0.5)
  outcome <- rnorm(n, 60, 20)
  outcome_2 <- ifelse(treatment == 1, outcome+5, outcome)

  model <- lm(outcome_2 ~ treatment)
  model.c <- coef(summary(model))
  p <- (pt(model.c[2,3],8,lower.tail = FALSE))*2
}) < 0.05 )
```

```
## [1] 0.19
```

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

```
set.seed(2)

n <- 1000
mean(replicate(1000, {
  treatment <- rbinom(n, size = 1, prob = 0.5)
  outcome <- rnorm(n, 60, 20)
  outcome_2 <- ifelse(treatment == 1, outcome+5, outcome)

  model <- lm(outcome_2 ~ treatment)
  model.c <- coef(summary(model))
  p <- (pt(model.c[2,3],8,lower.tail = FALSE))*2
}) < 0.05 )
```

```
## [1] 0.951
```

## 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.

```
library(readr)
df <- read_csv("~/Desktop/BIOS 6301 - Introduction to Statistical Computing/datasets//proj_wr23.csv")
```

```
## Rows: 256 Columns: 10
## -- Column specification -----
## Delimiter: ","
## chr (2): PlayerName, Team
## dbl (8): rec_att, rec_yds, rec_tds, rush_att, rush_yds, rush_tds, fumbles, fpts
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

```
head(df)
```

```
## # A tibble: 6 x 10
##   PlayerName Team rec_att rec_yds rec_tds rush_att rush_yds rush_tds fumbles
##   <chr>      <chr>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Justin Jeffe~ MIN     110.   1529.     8      3.7    18.1     0.3     0.4
## 2 Ja'Marr Chase CIN     107    1377.    10.1    5.2    24.6     0.1     1.1
## 3 Cooper Kupp LAR      104.    1325.     8.7    7.1    74.7     0.9     0.6
## 4 Tyreek Hill MIA      106.    1402.     7.8    9.5    57.1     0.6     0.5
## 5 CeeDee Lamb DAL       99.7   1307.     7.9    8.5    55.4     0.2     0.5
## 6 Stefon Diggs BUF       96.8   1286.     8.5    0.8     3.9      0      0.6
## # i 1 more variable: fpts <dbl>
```

```
wr23.df <- df[-c(1,2)]
head(wr23.df)
```

```
## # A tibble: 6 x 8
##   rec_att rec_yds rec_tds rush_att rush_yds rush_tds fumbles fpts
##   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>
## 1 110.   1529.     8      3.7    18.1     0.3     0.4 204.
## 2 107    1377.    10.1    5.2    24.6     0.1     1.1 199.
## 3 104.    1325.     8.7    7.1    74.7     0.9     0.6 196.
## 4 106.    1402.     7.8    9.5    57.1     0.6     0.5 195.
## 5 99.7    1307.     7.9    8.5    55.4     0.2     0.5 184.
## 6 96.8    1286.     8.5    0.8     3.9      0      0.6 179.
```

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

```
wr23.corr <- cor(wr23.df)
wr23.corr
```

```
##           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)

```
library(MASS)

var.wr23.df <- var(wr23.df)
mean.wr23.df <- colMeans(wr23.df)

sim <- mvrnorm(30, mu = mean.wr23.df, Sigma = var.wr23.df)
sim.df <- as.data.frame(sim)

sim.matrix <- cor(sim.df)
sim.matrix
```

```
## rec_att rec_yds rec_tds rush_att rush_yds rush_tds fumbles
## rec_att 1.0000000 0.9765594 0.9768014 0.5541054 0.5206232 0.4359737 0.6444594
## rec_yds 0.9765594 1.0000000 0.9711872 0.4753867 0.4329781 0.3949890 0.6293757
## rec_tds 0.9768014 0.9711872 1.0000000 0.4358826 0.4097618 0.3541964 0.5966012
## rush_att 0.5541054 0.4753867 0.4358826 1.0000000 0.9491220 0.8128254 0.5538708
## rush_yds 0.5206232 0.4329781 0.4097618 0.9491220 1.0000000 0.9100954 0.5390258
## rush_tds 0.4359737 0.3949890 0.3541964 0.8128254 0.9100954 1.0000000 0.5206031
## fumbles 0.6444594 0.6293757 0.5966012 0.5538708 0.5390258 0.5206031 1.0000000
## fpts 0.9861085 0.9964053 0.9821103 0.5122496 0.4800669 0.4365000 0.6318189
## fpts
## rec_att 0.9861085
## rec_yds 0.9964053
## rec_tds 0.9821103
## rush_att 0.5122496
## rush_yds 0.4800669
## rush_tds 0.4365000
## fumbles 0.6318189
## fpts 1.0000000
```

```
keep.1=0
loops=1000
```

```

for (i in 1:loops){
  sim2 <- mvrnorm(30, mu = mean.wr23.df, Sigma = var.wr23.df)
  sim2.matrix <- keep.1+cor(sim2)/loops
}

sim2.matrix

```

```

##          rec_att      rec_yds      rec_tds      rush_att      rush_yds
## rec_att  1.000000e-03  9.748363e-04  9.733791e-04  9.366844e-05  7.517513e-05
## rec_yds  9.748363e-04  1.000000e-03  9.756412e-04  9.123550e-05  6.484852e-05
## rec_tds  9.733791e-04  9.756412e-04  1.000000e-03  4.090452e-05  1.094207e-05
## rush_att 9.366844e-05  9.123550e-05  4.090452e-05  1.000000e-03  9.494793e-04
## rush_yds 7.517513e-05  6.484852e-05  1.094207e-05  9.494793e-04  1.000000e-03
## rush_tds -1.873136e-05 -2.522219e-05 -8.317690e-05 8.313376e-04 9.092258e-04
## fumbles  7.718322e-04  7.675817e-04  7.273057e-04 1.974602e-04 1.746754e-04
## fpts      9.782027e-04  9.967942e-04  9.813811e-04 1.439216e-04 1.208035e-04
##          rush_tds      fumbles      fpts
## rec_att  -1.873136e-05 7.718322e-04 9.782027e-04
## rec_yds  -2.522219e-05 7.675817e-04 9.967942e-04
## rec_tds  -8.317690e-05 7.273057e-04 9.813811e-04
## rush_att  8.313376e-04 1.974602e-04 1.439216e-04
## rush_yds  9.092258e-04 1.746754e-04 1.208035e-04
## rush_tds  1.000000e-03 7.903137e-05 2.776676e-05
## fumbles  7.903137e-05 1.000000e-03 7.642516e-04
## fpts      2.776676e-05 7.642516e-04 1.000000e-03

```

### 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       5.34       9.79       1.05       1.14
##      gamma    geometric hypergeometric    lognormal    negbinomial
##      1.01       2.02       2.64       1.72       31.49
##      normal    poisson      t      uniform      weibull
##      -0.03     25.06     -0.03     0.50       1.00
```

Here, ‘sapply’ is running the `nDist` function (defined above) on the value of 500. Further, we have specified that ‘sapply’ return the mean values of each output (e.g., `beta`, `binomial`, `chisquared`) and round the numerical value to 2 decimal places.

2. What about this? (3 points)

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

```
##      beta      uniform      t      f hypergeometric
##      0.000000000  0.002236068  0.006958524  0.007451598  0.009445132
##      normal      weibull    exponential      gamma      binomial
##      0.009880869  0.010208356  0.011367081  0.011367081  0.018750439
##      lognormal    geometric    chisquared    poisson    negbinomial
##      0.018890265  0.026051568  0.043585971  0.053692595  0.085075818
```

We are using ‘sapply’ to run the `nDist` function on the value of 10000 and specifying it return the mean values of each output rounded to 2 decimal places. Additionally, we are asking this command to be replicated 20 times and to sort the outputted means in a matrix. Finally, we are asking to find the standard deviation of each mean, then sort the results from lowest to highest.

In the output above, a small value would indicate that  $N=10,000$  would provide a sufficient 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)

```
#sizeDist = c()
#n=100000
#
# while(length(sizeDist) != 15){
#   sdDist = rowMeans(replicate(100, {
#     apply(replicate(20, round(sapply(nDist(n), mean), 2)), 1, sd)
#   }))
#   x = sdDist[which(sdDist < 0.02)]
#   sizeDist[setdiff(names(x), names(sizeDist))]= n
#   n=n+1000
# }
#sizeDist
```

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	100
binomial	8,000
chisquared	50,000
exponential	2,500
f	1,200
gamma	2,500
geometric	15,000
hypergeometric	4,300
lognormal	12,000
negbinomial	200,000
normal	2,500
poisson	70,000
t	3,000
uniform	300
weibull	2,600