Bios 6301: Assignment 3

Yeji Ko

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)

```r
treatment <- rbinom(100,1, 0.5)
outcome <- rnorm(100,60,20)
x <- data.frame(treatment, outcome)
x$outcome <- ifelse(x$treatment == 1, x$outcome + 5, x$outcome) # add 5 if treatment == 1
mod <- lm(outcome ~ treatment, data = x) # create a linear model
coef(summary(mod))[2,4] # extract p-value
```

```
## [1] 0.05775862
```

```r
# repeat 1000 times
set.seed(1234)
mean(replicate(1e3, {
treatment <- rbinom(100,1, 0.5)
outcome <- rnorm(100,60,20)
x <- data.frame(treatment, outcome)
x$outcome <- ifelse(x$treatment == 1, x$outcome + 5, x$outcome) # add 5 if treatment == 1
mod <- lm(outcome ~ treatment, data = x) # create a linear model
coef(summary(mod))[2,4]
}) < 0.05)
```

```
## [1] 0.228
```

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

```
set.seed(1234)
mean(replicate(1e3, {
treatment <- rbinom(1000,1, 0.5)
outcome <- rnorm(1000,60,20)
x <- data.frame(treatment, outcome)
x$outcome <- ifelse(x$treatment == 1, x$outcome + 5, x$outcome) # add 5 if treatment == 1
mod <- lm(outcome ~ treatment, data = x) # create a linear model
coef(summary(mod))[2,4]
}) < 0.05)
```

```
## [1] 0.975
```

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.

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

```
proj_wr21 <- read.csv('proj_wr21.csv')
x <- proj_wr21[-c(1:2)]
cor(x)
```

```
##             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)

```
(rho=cor(x))
```

```
##             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
```

```
(vcov=var(x))
```

```
##               rec_att      rec_yds      rec_tds    rush_att    rush_yds
## rec_att    951.881525  12217.36997   79.1418829  34.3791104  209.750458
## rec_yds  12217.369967 160006.01993 1036.3760093 416.9163803 2560.876948
## rec_tds     79.141883   1036.37601    7.0657915   2.7431819   16.752191
## rush_att    34.379110    416.91638    2.7431819   9.1158251   52.895823
## rush_yds   209.750458   2560.87695   16.7521907  52.8958226  314.274732
## rush_tds     1.476326     17.93203    0.1225012   0.3731026    2.217875
## fumbles     10.033727    130.57146    0.8256371   0.3952694    2.377153
## fpts      1706.434040  22324.11601  146.8061269  64.9067870  396.738969
##              rush_tds      fumbles        fpts
## rec_att    1.47632636  10.03372690 1706.434040
## rec_yds   17.93202699 130.57145717 22324.116005
## rec_tds    0.12250125   0.82563710  146.806127
## rush_att   0.37310256   0.39526938   64.906787
## rush_yds   2.21787535   2.37715253  396.738969
## rush_tds   0.01908700   0.01600594    2.831126
## fumbles    0.01600594   0.16602480   18.019506
## fpts       2.83112648  18.01950577 3134.263774
```

```
(means=colMeans(x))
```

```
##      rec_att      rec_yds      rec_tds     rush_att     rush_yds     rush_tds
##  27.76765799 356.78289963   2.23271375   1.25650558   7.30260223   0.04200743
##      fumbles         fpts
##   0.30446097  49.45204461
```

```
require(MASS)
```

```
## Loading required package: MASS
```

```
sim = mvrnorm(30, mu = means, Sigma = vcov)
sim = as.data.frame(sim)

(rho.sim=cor(sim))   # create a simulated correlation matrix
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9939044 0.9711742 0.2676815 0.3011769 0.1732277 0.9125134
## rec_yds  0.9939044 1.0000000 0.9791311 0.2318710 0.2659958 0.1324026 0.9201598
## rec_tds  0.9711742 0.9791311 1.0000000 0.2413144 0.2758702 0.1352836 0.8855415
## rush_att 0.2676815 0.2318710 0.2413144 1.0000000 0.9938791 0.8943585 0.2948170
## rush_yds 0.3011769 0.2659958 0.2758702 0.9938791 1.0000000 0.9058955 0.3201236
## rush_tds 0.1732277 0.1324026 0.1352836 0.8943585 0.9058955 1.0000000 0.1484751
```

```
## fumbles  0.9125134 0.9201598 0.8855415 0.2948170 0.3201236 0.1484751 1.0000000
## fpts     0.9922568 0.9972796 0.9890606 0.2731595 0.3076606 0.1717309 0.9143673
##               fpts
## rec_att  0.9922568
## rec_yds  0.9972796
## rec_tds  0.9890606
## rush_att 0.2731595
## rush_yds 0.3076606
## rush_tds 0.1717309
## fumbles  0.9143673
## fpts     1.0000000
```

```
rho                  # compare with the original correlation matrix
```

```
##            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
```

```
keep.1 = 0
loops=1000

for (i in 1:loops) {
      sim = mvrnorm(30, mu = means, Sigma = vcov)
      keep.1=keep.1+ cor(sim)/loops
}

rho
```

```
##            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
```

```r
keep.1
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9892925 0.9635208 0.3719545 0.3854190 0.3485786 0.7901616
## rec_yds  0.9892925 1.0000000 0.9737131 0.3491755 0.3641673 0.3277934 0.7941538
## rec_tds  0.9635208 0.9737131 1.0000000 0.3452179 0.3581375 0.3363856 0.7548461
## rush_att 0.3719545 0.3491755 0.3452179 1.0000000 0.9878733 0.8913456 0.3279481
## rush_yds 0.3854190 0.3641673 0.3581375 0.9878733 1.0000000 0.9020138 0.3355087
## rush_tds 0.3485786 0.3277934 0.3363856 0.8913456 0.9020138 1.0000000 0.2907549
## fumbles  0.7901616 0.7941538 0.7548461 0.3279481 0.3355087 0.2907549 1.0000000
## fpts     0.9872113 0.9967550 0.9859345 0.3873684 0.4022347 0.3687656 0.7830540
##               fpts
## rec_att  0.9872113
## rec_yds  0.9967550
## rec_tds  0.9859345
## rush_att 0.3873684
## rush_yds 0.4022347
## rush_tds 0.3687656
## fumbles  0.7830540
## fpts     1.0000000
```

Question 3

21 points

Here's some code:

```r
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)

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

```
##          beta       binomial     chisquared    exponential              f 
##          0.10           5.39          10.31           1.01           1.19 
##         gamma      geometric hypergeometric      lognormal    negbinomial 
##          1.00           2.19           2.66           1.63          31.88 
##        normal        poisson              t        uniform        weibull 
##          0.01          25.24          -0.05           0.52           0.92
```

```
This prints out the means of 500-times-replicated test statistics
from 15 different distributions, whch is rounded with two decimal places.
```

2. What about this? (3 points)

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

```
##           beta        uniform              f    exponential              t 
##    0.000000000    0.003077935    0.006863327    0.009233805    0.009880869 
##         normal        weibull          gamma hypergeometric      lognormal 
##    0.010208356    0.010711528    0.011821034    0.011909748    0.019303667 
##       binomial      geometric     chisquared        poisson    negbinomial 
##    0.019303667    0.020900768    0.034625819    0.060808154    0.072712484
```

```
This replicates the means of 10000-times-replicated test statistics,
each from 15 different distributions, rounded with two decimal places, 20 times.
Then it calculates the standard deviation for each distribution and orders from lowest to highest.
```

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	16
binomial	21
chisquared	32
exponential	34
f	35
gamma	41
geometric	53
hypergeometric	68
lognormal	76
negbinomial	83
normal	85
poisson	102
t	107
uniform	110
weibull	119

```
```r
n <- 10
i <- 1
size <- c()
```
```

~~~
set.seed(1234)
for(i in 1:15){repeat{
truth <- data.frame(matrix(rep(round(sapply(nDist(10000), mean), 2),n), ncol = 15, byrow = TRUE))
x <- nDist(n)
p <- t.test(x[[i]],truth[[i]], alternative='two.sided')$p.value
  if(p < 0.05) break
  n <- n + 1
  size[i] <- n
}
  i <- i + 1
}

size
```

```
## [1] 16 21 32 34 35 41 53 68 76 83 85 102 107 110 119
```
~~~