Bios 6301: Assignment 3

Haoyang Yi

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 treatmenat 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(414)
size = 100
times = 1000
res_pvalue = NULL
for(i in 1:times){
group = sample(c(0,1),size,replace = T)
y = rnorm(size,mean = 60,sd = 20)
y[group==1]=y[group==1]+5
mod = lm(y~group)
res_pvalue[i] = coef(summary(mod))[2,4]
}
sum(res_pvalue < 0.05)/times # power when size = 100
```

```
## [1] 0.231
```

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

```
set.seed(414)
size = 1000
times = 1000
res_pvalue = NULL
for(i in 1:times){
group = sample(c(0,1),size,replace = T)
y = rnorm(size,mean = 60,sd = 20)
y[group==1]=y[group==1]+5
mod = lm(y~group)
res_pvalue[i] = coef(summary(mod))[2,4]
}
sum(res_pvalue < 0.05)/times # power when size = 1000
```

```
## [1] 0.974
```

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)

```
library(MASS)
df = read.csv('proj_wr21.csv')
df = df[,-c(1,2)]
cor(df) # correlation matrix
```

```
##             rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att   1.0000000 0.9889836 0.9620513 0.2242480 0.2810831 0.2312038 0.6423627
## rec_yds   0.9889836 1.0000000 0.9720400 0.2062038 0.2614786 0.2115013 0.6487247
## rec_tds   0.9620513 0.9720400 1.0000000 0.2004448 0.2540571 0.2151580 0.6021914
## rush_att  0.2242480 0.2062038 0.2004448 1.0000000 0.9779751 0.9308512 0.1446322
## rush_yds  0.2810831 0.2614786 0.2540571 0.9779751 1.0000000 0.9298581 0.1761579
## rush_tds  0.2312038 0.2115013 0.2151580 0.9308512 0.9298581 1.0000000 0.1809564
## fumbles   0.6423627 0.6487247 0.6021914 0.1446322 0.1761579 0.1809564 1.0000000
## fpts      0.9863078 0.9957911 0.9842850 0.2610623 0.3162080 0.2677821 0.6288445
##                fpts
## rec_att   0.9863078
## rec_yds   0.9957911
## rec_tds   0.9842850
## rush_att  0.2610623
## rush_yds  0.3162080
## rush_tds  0.2677821
## fumbles   0.6288445
## 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)

```
dfcor = cor(df)
dfcov = var(df)
dfmean = colMeans(df)
rescor = 0
set.seed(414)
# generate data from similar correlation structure
for (i in 1:1000){
    dt = mvrnorm(30,dfmean,dfcov)
    rescor = rescor+cor(dt)/1000
}
rescor
```

```
##             rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att   1.0000000 0.9884243 0.9609830 0.2219991 0.2781960 0.2294294 0.6353030
## rec_yds   0.9884243 1.0000000 0.9711987 0.2055199 0.2600256 0.2106236 0.6424439
## rec_tds   0.9609830 0.9711987 1.0000000 0.1997415 0.2527589 0.2145637 0.5957682
## rush_att  0.2219991 0.2055199 0.1997415 1.0000000 0.9768849 0.9271844 0.1485957
## rush_yds  0.2781960 0.2600256 0.2527589 0.9768849 1.0000000 0.9263911 0.1797179
## rush_tds  0.2294294 0.2106236 0.2145637 0.9271844 0.9263911 1.0000000 0.1822552
## fumbles   0.6353030 0.6424439 0.5957682 0.1485957 0.1797179 0.1822552 1.0000000
## fpts      0.9856832 0.9956520 0.9838195 0.2591613 0.3136727 0.2658371 0.6226126
##                fpts
```

```
## rec_att  0.9856832
## rec_yds  0.9956520
## rec_tds  0.9838195
## rush_att 0.2591613
## rush_yds 0.3136727
## rush_tds 0.2658371
## fumbles  0.6226126
## fpts     1.0000000
```

```r
rescor-cor(df) # The difference between average cor matrix and true cor matrix is small.
```

```
##                rec_att       rec_yds       rec_tds      rush_att      rush_yds
## rec_att   6.661338e-16 -5.592727e-04 -1.068322e-03 -2.248962e-03 -2.887092e-03
## rec_yds  -5.592727e-04  6.661338e-16 -8.412918e-04 -6.839453e-04 -1.452956e-03
## rec_tds  -1.068322e-03 -8.412918e-04  6.661338e-16 -7.032827e-04 -1.298142e-03
## rush_att -2.248962e-03 -6.839453e-04 -7.032827e-04  6.661338e-16 -1.090128e-03
## rush_yds -2.887092e-03 -1.452956e-03 -1.298142e-03 -1.090128e-03  6.661338e-16
## rush_tds -1.774428e-03 -8.777031e-04 -5.942986e-04 -3.666769e-03 -3.467047e-03
## fumbles  -7.059659e-03 -6.280755e-03 -6.423238e-03  3.963492e-03  3.559944e-03
## fpts     -6.245745e-04 -1.391585e-04 -4.654498e-04 -1.900973e-03 -2.535274e-03
##               rush_tds       fumbles          fpts
## rec_att  -1.774428e-03 -7.059659e-03 -6.245745e-04
## rec_yds  -8.777031e-04 -6.280755e-03 -1.391585e-04
## rec_tds  -5.942986e-04 -6.423238e-03 -4.654498e-04
## rush_att -3.666769e-03  3.963492e-03 -1.900973e-03
## rush_yds -3.467047e-03  3.559944e-03 -2.535274e-03
## rush_tds  6.661338e-16  1.298764e-03 -1.945003e-03
## fumbles   1.298764e-03  6.661338e-16 -6.231894e-03
## fpts     -1.945003e-03 -6.231894e-03  6.661338e-16
```

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),
```

```r
        weibull = rweibull(n, shape)
    )
}
as.numeric(sapply(nDist(500), mean)[1])
```

```
## [1] 0.1004172
```

1. What does this do? (3 points)

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

```
##          beta       binomial     chisquared    exponential              f
##          0.10           5.29           9.88           0.99           1.10
##         gamma      geometric hypergeometric      lognormal    negbinomial
##          0.99           1.84           2.56           1.69          31.37
##        normal        poisson              t        uniform        weibull
##          0.06          24.93           0.12           0.51           0.95
```

It return 2 digit means of each generated sample from beta,uniform…distributions, like mean of a generated sample with size 500 from beta distribution (5,45)

1. 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    0.002236068    0.007677719    0.008750940    0.009947229
##        normal        weibull          gamma hypergeometric       binomial
##   0.010990426    0.012565617    0.012732057    0.013562720    0.015217718
##     geometric      lognormal     chisquared        poisson    negbinomial
##   0.021740092    0.024381831    0.046732723    0.047225662    0.079795792
```

It runs the simulation 20 times and calculates the standard deviation of 20 simulated means of each sample from beta,uniform..distributions .

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".

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

```r
b= nDist(50)
t = data.frame(distribution = names(b),true_mean = c(0.1,16/3,10,1,17/15,1,2,400/150,exp(1/2),32,0,25,0
n = seq(10,2000,10) # starts with sample size = 10 and set gap = 10
set.seed(414)
diff = matrix(0,length(n),length(names(b)))
for (i in 1:length(names(b))){
    for(j in 1:length(n))
        {diff[j,i]=as.numeric(sapply(nDist(n[j]), mean)[i])-t[i,2]
}
}
diff = cbind(n,diff)
minval = NULL
for (i in 2:ncol(diff)){
    minval[i-1]=min(which(abs(diff[,i])<=0.02))
}
t$sizerequired = 10+10*(minval-1)
t
```

```
##      distribution true_mean sizerequired
```

```
## 1            beta  0.100000            10
## 2        binomial  5.333333           130
## 3      chisquared 10.000000           130
## 4     exponential  1.000000            90
## 5               f  1.133333            50
## 6           gamma  1.000000           200
## 7       geometric  2.000000           240
## 8  hypergeometric  2.666667           160
## 9       lognormal  1.648721            20
## 10    negbinomial 32.000000           100
## 11         normal  0.000000            80
## 12        poisson 25.000000           360
## 13              t  0.000000            70
## 14        uniform  0.500000            90
## 15        weibull  1.000000            10
```

#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	130
chisquared	130
exponential	90
f	50
gamma	200
geometric	240
hypergeometric	160
lognormal	20
negbinomial	100
normal	80
poisson	360
t	70
uniform	90
weibull	10