Bios 6301: Assignment 3

Yan Yan

Due Thursday, 27 September, 1:00 PM

50 points total.

Submit a single knitr file (named `homework3.rmd`) by email to coleman.r.harris@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.seed(10)
pv <- NULL
repeat {
n <- 100                          # sample size
treat <- numeric(n)
outcome <- numeric(n)
for (i in seq(n)) {
treat[i] <- sample(c(0,1),1)    # assign treatment to each patient
outcome[i] <- rnorm(1,mean = 60,sd = 20)  # assign treatment to each patient
if (treat[i] == 1) outcome[i] = outcome[i] + 5
}
z <- data.frame(treat,outcome)
mod.3 <- lm(outcome~treat, data = z)   # linear model
pv1 <- summary(mod.3)$coefficients[2,4] # extract p value
pv <- c(pv,pv1)
if(length(pv)==1000){break}  ## repeat 1000 times
}
mean(pv<=0.05)
```

```
## [1] 0.239
```

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

```
set.seed(11)
pv2 <- NULL
pv2 <- replicate(1e3,{          #repeat 1000 times
n <- 1000                         # sample size 1000
treat <- numeric(n)
outcome <- numeric(n)
for (i in seq(n)) {
treat[i] <- sample(c(0,1),1)    # assign treatment to each patient
outcome[i] <- rnorm(1,mean = 60,sd = 20)  # assign treatment to each patient
if (treat[i] == 1) outcome[i] = outcome[i] + 5
}
z <- data.frame(treat,outcome)
mod.3 <- lm(outcome~treat, data = z)    # linear model
summary(mod.3)$coefficients[2,4] # extract p value
})
mean(pv2<=0.05)
```

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

The power increased with sample size increasing.

Question 2

14 points

Obtain a copy of the football-values lecture. Save the `2018/proj_wr18.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)

```
my.data <- read.csv("proj_wr18.csv")
my.data2 <- my.data[,-1:-2]
res <- cor(my.data2)
round(res,2)
```

```
##          rush_att rush_yds rush_tds rec_att rec_yds rec_tds fumbles fpts
## rush_att     1.00     0.98     0.96    0.21    0.19    0.10    0.41 0.98
## rush_yds     0.98     1.00     0.98    0.18    0.17    0.09    0.40 1.00
## rush_tds     0.96     0.98     1.00    0.16    0.14    0.07    0.36 0.99
## rec_att      0.21     0.18     0.16    1.00    0.99    0.87    0.49 0.21
## rec_yds      0.19     0.17     0.14    0.99    1.00    0.88    0.46 0.20
## rec_tds      0.10     0.09     0.07    0.87    0.88    1.00    0.33 0.12
## fumbles      0.41     0.40     0.36    0.49    0.46    0.33    1.00 0.39
## fpts         0.98     1.00     0.99    0.21    0.20    0.12    0.39 1.00
```

1. Generate a data set with 30 rows that has a similar correlation structure. Repeat the procedure 10,000 times and return the mean correlation matrix. (10 points)

```r
library(MASS)
mod.wr=lm(rush_att~.,data=my.data2)
print(summary(mod.wr))
```

```
##
## Call:
## lm(formula = rush_att ~ ., data = my.data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.0953  -1.4041   0.8383   1.9931  18.9295
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.241712   0.492459  -2.521   0.0123 *
## rush_yds     0.188909   0.165184   1.144   0.2538
## rush_tds     4.337338   9.876563   0.439   0.6609
## rec_att      1.032355   0.665057   1.552   0.1218
## rec_yds     -0.005353   0.201455  -0.027   0.9788
## rec_tds     -0.948978  11.529343  -0.082   0.9345
## fumbles     -1.687990   3.641842  -0.463   0.6434
## fpts        -1.013912   1.646835  -0.616   0.5386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.027 on 261 degrees of freedom
## Multiple R-squared:  0.9644, Adjusted R-squared:  0.9635
## F-statistic:  1011 on 7 and 261 DF,  p-value: < 2.2e-16
```

```r
rho.wr <- cor(my.data2)
vcov.wr=var(my.data2)
means.wr=colMeans(my.data2)

keep.1=0
loops=10000

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

rho.wr ; keep.1
```

```
##           rush_att  rush_yds   rush_tds   rec_att   rec_yds    rec_tds
## rush_att 1.0000000 0.9811206 0.95797730 0.2053187 0.1899521 0.10202739
## rush_yds 0.9811206 1.0000000 0.98207194 0.1823581 0.1694193 0.08861680
## rush_tds 0.9579773 0.9820719 1.00000000 0.1587441 0.1440112 0.07327293
## rec_att  0.2053187 0.1823581 0.15874406 1.0000000 0.9852488 0.86528857
## rec_yds  0.1899521 0.1694193 0.14401118 0.9852488 1.0000000 0.88460235
## rec_tds  0.1020274 0.0886168 0.07327293 0.8652886 0.8846023 1.00000000
## fumbles  0.4082506 0.3953831 0.35752542 0.4917518 0.4638223 0.32748238
## fpts     0.9782576 0.9978996 0.98931462 0.2146497 0.2022450 0.12279565
##            fumbles      fpts
## rush_att 0.4082506 0.9782576
```

```
## rush_yds 0.3953831 0.9978996
## rush_tds 0.3575254 0.9893146
## rec_att  0.4917518 0.2146497
## rec_yds  0.4638223 0.2022450
## rec_tds  0.3274824 0.1227956
## fumbles  1.0000000 0.3925229
## fpts     0.3925229 1.0000000
```

```
##           rush_att   rush_yds   rush_tds   rec_att   rec_yds    rec_tds
## rush_att 1.0000000 0.98060144 0.95691012 0.2045091 0.1896517 0.10226235
## rush_yds 0.9806014 1.00000000 0.98152816 0.1816050 0.1691768 0.08880522
## rush_tds 0.9569101 0.98152816 1.00000000 0.1585364 0.1443346 0.07360816
## rec_att  0.2045091 0.18160500 0.15853638 1.0000000 0.9847652 0.86159241
## rec_yds  0.1896517 0.16917678 0.14433464 0.9847652 1.0000000 0.88148739
## rec_tds  0.1022623 0.08880522 0.07360816 0.8615924 0.8814874 1.00000000
## fumbles  0.4048678 0.39226732 0.35521450 0.4864942 0.4591651 0.32454223
## fpts     0.9777062 0.99783455 0.98898309 0.2133799 0.2014591 0.12231544
##            fumbles      fpts
## rush_att 0.4048678 0.9777062
## rush_yds 0.3922673 0.9978345
## rush_tds 0.3552145 0.9889831
## rec_att  0.4864942 0.2133799
## rec_yds  0.4591651 0.2014591
## rec_tds  0.3245422 0.1223154
## fumbles  1.0000000 0.3896068
## fpts     0.3896068 1.0000000
```

Question 3

21 points

Here's some code:

```
set.seed(890)
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           5.25          10.08           0.99           1.09
##          gamma      geometric hypergeometric      lognormal    negbinomial
##           1.05           2.10           2.63           1.78          32.10
##         normal        poisson              t        uniform        weibull
##          -0.05          25.11          -0.08           0.50           0.96
```

```
Generate 500 values for each of the distribution (with some fixed parameters), evaluate the mean fo
```

2. What about this? (3 points)

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

```
##           beta        uniform              f hypergeometric         normal
##    0.000000000    0.003077935    0.008255779    0.008506963    0.009665457
##              t        weibull          gamma    exponential       binomial
##    0.010500627    0.012343760    0.012513151    0.013327850    0.016733201
##      lognormal      geometric     chisquared        poisson    negbinomial
##    0.020365089    0.023395906    0.041927255    0.046506932    0.118164693
```

```
For each of the distribution, genarate 10000 values, evaluate and round the mean.Replicate 20 times
```

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	4
binomial	5700
chisquared	30000
exponential	1350
f	640
gamma	2000
geometric	8500
hypergeometric	3700
lognormal	8000
negbinomial	190000
normal	1950
poisson	49000
t	2300
uniform	100
weibull	1800

```
#### code used for finding the sample size
#### change initial value of i and increment for each distribution
hyp.sd <- NULL
```

```
i <- 3000
repeat{
  set.seed(890)
  mean.sd <- apply(replicate(20, round(sapply(nDist(i), mean), 2)), 1, sd)
  as.data.frame(mean.sd)
  hyp.sd[i] <- mean.sd["hypergeometric"]
  if(hyp.sd[i]< 0.02) {
    break}
  i <- i+100
}
i
```

```
## [1] 3700
```

I used the following initial values and increment for searching the best sample size.

Distribution	Initial Value	Increment	Result
Beta	2	1	4
binomial	5000	100	5700
chisquared	25000	1000	30000
exponential	800	50	1350
f	600	20	640
gamma	1000	100	2000
geometric	7000	500	8500
hypergeometric	3000	100	3700
lognormal	5000	500	8000
negbinomial	100000	10000	190000
normal	1000	50	1950
poisson	45000	1000	49000
t	2000	100	2300
uniform	50	10	100
weibull	1500	100	1800