Bios 6301: Assignment 3

Haley Yaremych

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.seed(95)
n=100
less.counter = 0

for (i in 1:1000) {
    df = data.frame(trt = rbinom(n, 1, .5), out = rnorm(n, mean=60, sd=20))
    df[df$trt==1,2] = df[df$trt==1,2] + 5

    mod1 = lm(out ~ trt, df)
    pval = summary(mod1)$coefficients[2,4]

    if (pval < .05) {less.counter =  less.counter + 1}
}

less.counter/1000
```

```
## [1] 0.263
```

When N = 100, the power is 0.263.

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

```
set.seed(95)
n=1000
less.counter = 0

for (i in 1:1000) {
    df = data.frame(trt = rbinom(n, 1, .5), out = rnorm(n, mean=60, sd=20))
    df[df$trt==1,2] = df[df$trt==1,2] + 5

    mod1 = lm(out ~ trt, df)
    pval = summary(mod1)$coefficients[2,4]

    if (pval < .05) {less.counter =  less.counter + 1}
}

less.counter/1000
```

```
## [1] 0.988
```

When N = 1000, the power is 0.988.

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.

```
wr = read.csv("~/Documents/Fall 2021/Statistical Computing/homework/proj_wr21.csv", header=TRUE)
wr = wr[,-c(1,2)]
```

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

```
(cor.wr = cor(wr))
```

```
##            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(MBESS)
library(MASS)
```

```
sim.cor=0

for (i in 1:1000) {
    wr.sim = mvrnorm(30,
                mu = numeric(dim(wr)[2]),
                Sigma = cor2cov(cor.wr, sd=unname(apply(wr, MARGIN=2, FUN=sd))))

    sim.cor = sim.cor + cor(wr.sim)/1000
}

sim.cor
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9897202 0.9642338 0.3595168 0.3750958 0.3391914 0.7957220
## rec_yds  0.9897202 1.0000000 0.9742356 0.3360849 0.3529172 0.3178428 0.7985202
## rec_tds  0.9642338 0.9742356 1.0000000 0.3330515 0.3476751 0.3265359 0.7604502
## rush_att 0.3595168 0.3360849 0.3330515 1.0000000 0.9878067 0.8898503 0.3113654
## rush_yds 0.3750958 0.3529172 0.3476751 0.9878067 1.0000000 0.9010135 0.3200349
## rush_tds 0.3391914 0.3178428 0.3265359 0.8898503 0.9010135 1.0000000 0.2784986
## fumbles  0.7957220 0.7985202 0.7604502 0.3113654 0.3200349 0.2784986 1.0000000
## fpts     0.9876153 0.9967866 0.9862422 0.3745423 0.3912189 0.3587734 0.7873606
##               fpts
## rec_att  0.9876153
## rec_yds  0.9967866
## rec_tds  0.9862422
## rush_att 0.3745423
## rush_yds 0.3912189
## rush_tds 0.3587734
## fumbles  0.7873606
## 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           5.34           9.76           0.96           1.11
##          gamma      geometric hypergeometric      lognormal    negbinomial
##           1.04           1.97           2.66           1.67          32.26
##         normal        poisson              t        uniform        weibull
##          -0.06          24.81           0.04           0.49           0.99
```

Here we are running the `nDist` function and specifying n = 500. Given this sample size and the other relevant parameters that were written into the function (e.g., df, shape parameters, etc., whatever parameters that define each distribution), we are then using `sapply` to run the function (which samples `n` times from each of the various distributions with the given parameters) and to calculate the mean of each of those samples. Then, we take that answer and round it to 2 decimal places.

1. What about this? (3 points)

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

```
##           beta        uniform        weibull              f         normal
##    0.000000000    0.003940345    0.006708204    0.007677719    0.010052494
##    exponential              t          gamma hypergeometric      lognormal
##    0.010711528    0.011742859    0.012523662    0.014464112    0.016670175
##       binomial      geometric        poisson     chisquared    negbinomial
##    0.017006191    0.028022547    0.039590004    0.054231473    0.099603160
```

Working from the inside out:

`round(sapply(nDist(10000), mean), 2)` is doing the same thing as above, but this time for n = 10,000.

This process (i.e., finding the mean of each distribution when n=10,000) is being replicated 20 times.

Then, we're taking the results of these replications, and for each row (i.e., `MARGIN=1`), we are calculating the SD. Here, each row contains the 20 replications for a certian distribution.

Then, we are sorting those SDs from smallest to largest.

Ultimately, we're simulating sampling variability of the means of each distribution. Across 20 samples each of size 10,000, the means of the beta distributions are the least variable whereas the means of the negative binominal distributions are the most variable.

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".

1. 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	5
binomial	190

distribution	N
chisquared	60,000
exponential	3,000
f	1,000
gamma	2,500
geometric	15,000
hypergeometric	4,200
lognormal	15,000
negbinomial	200,000
normal	3,500
poisson	60,000
t	4,500
uniform	300
weibull	3,000