Bios 6301: Assignment 3

Megan Taylor

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
set.seed(123)
# set number of subjects
patients <- 100
# create variable for treatment group
group <- rep(NA, patients)
# create variable for outcome
outcome <- rep(NA, patients)
# initialize data frame (each row is a patient)
study <- data.frame(group, outcome)
# set number of simulations
nsim = 1000
# initialize vector that will store p-values from each simulation
p.vec <- rep(NA, nsim)
# set loop for each simulation
for (i in 1:nsim){
    # set loop for each patient
    for (n in 1:patients){
        # assign group to each patient based on random number between 0 and 1 with equal probability
```

```r
        if (runif(1) <= 0.5) study[n,1] = 0
        else study[n,1] = 1
        # get outcome from normal variable
        study[n,2] = rnorm(1, 60, 20)
        # add 5 to those in the treatment group
        if (study[n,1] == 1)
            study[n,2] = study[n,2] + 5
    }
    # store the coefficients from the summary of the model
    mod <- coef(summary(lm(outcome ~ group, study)))
    # add the p-value from the simulation to the p.vec vector
    p.vec[i] <- mod[2,4]
}
# calculate power by finding the percent of simulations with p-value less than 0.05
power <- length(which(p.vec <= 0.05))/nsim
power
```

```
## [1] 0.234
```

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

```r
# same code as above, just changed patients to 1000
# set seed
set.seed(123)
# set number of subjects
patients <- 1000
# create variable for treatment group
group <- rep(NA, patients)
# create variable for outcome
outcome <- rep(NA, patients)
# initialize data frame (each row is a patient)
study <- data.frame(group, outcome)
# set number of simulations
nsim = 1000
# initialize vector that will store p-values from each simulation
p.vec <- rep(NA, nsim)
# set loop for each simulation
for (i in 1:nsim){
    # set loop for each patient
    for (n in 1:patients){
        # assign group to each patient based on random number between 0 and 1 with equal probability
        if (runif(1) <= 0.5) study[n,1] = 0
        else study[n,1] = 1
        # get outcome from normal variable
        study[n,2] = rnorm(1, 60, 20)
        # add 5 to those in the treatment group
        if (study[n,1] == 1)
            study[n,2] = study[n,2] + 5
    }
    # store the coefficients from the summary of the model
    mod <- coef(summary(lm(outcome ~ group, study)))
    # add the p-value from the simulation to the p.vec vector
    p.vec[i] <- mod[2,4]
}
# calculate power by finding the percent of simulations with p-value less than 0.05
```

```
power <- length(which(p.vec <= 0.05))/nsim
power
```

```
## [1] 0.98
```

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)

```
football <- read.csv("C:\\Users\\megan\\OneDrive\\Documents\\Fall 2021\\Statistical Computing\\Bios6301-
football <- football[-1]
football <- football[-1]
(rho.football=cor(football))
```

```
##             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(MASS)
vcov.football=var(football)
means.football=colMeans(football)

rhosim.avg = 0
nsim = 1000
for (i in 1:nsim){
    football.sim = mvrnorm(30, mu = means.football, Sigma = vcov.football)
    rhosim.avg = rhosim.avg + cor(football.sim)/nsim
}
rhosim.avg
```

```
##             rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att   1.0000000 0.9895969 0.9638967 0.3596202 0.3752461 0.3428140 0.7971398
## rec_yds   0.9895969 1.0000000 0.9740282 0.3357003 0.3528700 0.3204879 0.8003654
## rec_tds   0.9638967 0.9740282 1.0000000 0.3318861 0.3470541 0.3287513 0.7619927
## rush_att 0.3596202 0.3357003 0.3318861 1.0000000 0.9875468 0.8911090 0.3151148
## rush_yds 0.3752461 0.3528700 0.3470541 0.9875468 1.0000000 0.9026310 0.3238692
## rush_tds 0.3428140 0.3204879 0.3287513 0.8911090 0.9026310 1.0000000 0.2841717
## fumbles  0.7971398 0.8003654 0.7619927 0.3151148 0.3238692 0.2841717 1.0000000
## fpts     0.9875281 0.9968012 0.9861103 0.3737057 0.3907810 0.3610650 0.7894652
##               fpts
## rec_att  0.9875281
## rec_yds  0.9968012
## rec_tds  0.9861103
## rush_att 0.3737057
## rush_yds 0.3907810
## rush_tds 0.3610650
## fumbles  0.7894652
## 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           5.39           9.59           0.95           1.13
   ##          gamma      geometric hypergeometric      lognormal    negbinomial
   ##           0.91           2.17           2.63           1.63          31.76
   ##         normal        poisson              t        uniform        weibull
   ##          -0.07          25.28          -0.02           0.49           1.04
   ```

   ```
   This code is taking the mean of 500 observations from each of the given
   ```

```
distributions in the nDist function (which have specified parameters)
and rounding them to two decimal places.
```

2. What about this? (3 points)

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

```
##          beta        uniform              f          gamma         normal
##   0.000000000    0.003940345    0.008255779    0.008645047    0.008870412
##       weibull              t hypergeometric    exponential       binomial
##   0.010954451    0.011367081    0.011821034    0.014095539    0.019269556
##     lognormal      geometric        poisson     chisquared    negbinomial
##   0.021095023    0.021980853    0.041482400    0.041751142    0.080661737
```

```
This code takes the mean of 10000 observations from each of the
distributions given in the nDist function and repeats that process
20 times. It then takes the standard deviation of each of the rows
of 20 means (each row corresponds to one of the distributions).
Finally, it sorts the standard deviations labeled by distribution
in ascending order.
```

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	7
binomial	11000
chisquared	60000
exponential	3500
f	1800
gamma	3500
geometric	15000
hypergeometric	5000
lognormal	15000
negbinomial	250000
normal	3500
poisson	80000
t	4500
uniform	400
weibull	3500