Bios 6301: Assignment 3

Jeongwon Choi

Due Tuesday, 27 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 tianyi.sun@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
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
set.seed(2022)
n_simul <- 1000
n_patient <- 100
p_values <- rep(NaN, n_simul)
```

```r
for (i in 1:n_simul){
data <- tibble(treatment_group = rep(0, n_patient), outcome = rep(0, n_patient))

data$treatment_group <- sample(x=c(0,1), size=n_patient, replace=TRUE)

data[data$treatment_group==0, 2] <- rnorm(n=sum(data$treatment_group==0), mean=60, sd=20)
data[data$treatment_group==1, 2] <- rnorm(n=sum(data$treatment_group==1), mean=65, sd=20)

model <- lm(data, formula=outcome~treatment_group)
summary(model)
summary.model <- summary(model)
p_values[i] <- summary.model$coefficients[2,4]
}

alpha <- 0.05
sum(p_values <= alpha)/n_simul
```

```
## [1] 0.22
```

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

```r
library(tidyverse)
set.seed(2022)
n_simul <- 1000
n_patient <- 1000
p_values <- rep(NaN, n_simul)

for (i in 1:n_simul){
data <- tibble(treatment_group = rep(0, n_patient), outcome = rep(0, n_patient))

data$treatment_group <- sample(x=c(0,1), size=n_patient, replace=TRUE)

data[data$treatment_group==0, 2] <- rnorm(n=sum(data$treatment_group==0), mean=60, sd=20)
data[data$treatment_group==1, 2] <- rnorm(n=sum(data$treatment_group==1), mean=65, sd=20)

model <- lm(data, formula=outcome~treatment_group)
summary(model)
summary.model <- summary(model)
p_values[i] <- summary.model$coefficients[2,4]
}

alpha <- 0.05
sum(p_values <= alpha)/n_simul
```

```
## [1] 0.979
```

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.

```r
proj <- read.csv("proj_wr21.csv")
proj <- proj[,3:dim(proj)[2]]
```

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

```r
cor.mat<-cor(proj)
```

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)

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
n_simul <- 1000

keep1<-0
for(i in 1:n_simul){
  sampled_data <- mvrnorm(n=30, mu=colMeans(proj), Sigma=cor.mat)
  sample_cormat <- cor(sampled_data)
  keep1 <- keep1 + sample_cormat/n_simul
}
keep1
```

```
##            rec_att   rec_yds   rec_tds  rush_att  rush_yds  rush_tds   fumbles
## rec_att  1.0000000 0.9885826 0.9603984 0.2231754 0.2780962 0.2304710 0.6365114
## rec_yds  0.9885826 1.0000000 0.9712581 0.2052482 0.2583291 0.2109358 0.6427768
## rec_tds  0.9603984 0.9712581 1.0000000 0.1990163 0.2504049 0.2148496 0.5973134
## rush_att 0.2231754 0.2052482 0.1990163 1.0000000 0.9775997 0.9302699 0.1405305
## rush_yds 0.2780962 0.2583291 0.2504049 0.9775997 1.0000000 0.9296253 0.1713254
## rush_tds 0.2304710 0.2109358 0.2148496 0.9302699 0.9296253 1.0000000 0.1763029
## fumbles  0.6365114 0.6427768 0.5973134 0.1405305 0.1713254 0.1763029 1.0000000
## fpts     0.9857014 0.9956304 0.9837721 0.2595979 0.3126007 0.2669880 0.6228609
##               fpts
## rec_att  0.9857014
## rec_yds  0.9956304
## rec_tds  0.9837721
## rush_att 0.2595979
## rush_yds 0.3126007
## rush_tds 0.2669880
## fumbles  0.6228609
## 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           5.38          10.28           1.00           1.13
##          gamma      geometric hypergeometric      lognormal    negbinomial
##           1.00           1.96           2.65           1.55          32.29
##         normal        poisson              t        uniform        weibull
##          -0.01          24.83          -0.05           0.49           1.06
```

```
It calculates means (and round to 2 decimal points) from 500 random samples that
were taken from different kinds of distributions given above.
```

2. What about this? (3 points)

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

```
##           beta        uniform              f    exponential         normal
##    0.000000000    0.002236068    0.005871429    0.008013147    0.008506963
## hypergeometric          gamma        weibull              t       binomial
##    0.010809353    0.010990426    0.010990426    0.013562720    0.020641042
##      lognormal      geometric     chisquared        poisson    negbinomial
##    0.021244194    0.022618111    0.054219340    0.054733806    0.117916116
```

```
This calculates mean of 10000 samples that are sampled from different kinds of
distributions, 20 times repeatedly and round to two decimal places. For these sample
means, we calculate standard deviations and sort these sds from the smaller ones.
```

```
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)

```r
set.seed(106)
pop.mean <- round(sapply(nDist(10000), mean), 4)
  thresh <- 0.02
  n_simul <- 10000

  n_sufficient <- rep(NaN, length(pop.mean))

  idx_update <- rep(TRUE, length(pop.mean)) # update idx_sufficient if it is true.

  for (i in 1:n_simul){

    sample.mean <- round(sapply(nDist(i), mean),4)
    absdiff <- abs(sample.mean - pop.mean)

    idx_sufficient <- absdiff <= thresh # True if it is close enough to the pop mean.
    n_sufficient[ idx_sufficient & idx_update  ] = rep(i, sum(idx_sufficient & idx_update))

    # after updating n_sufficient, make idx_update for the corresponding elements false to prevent unne
    idx_update[ idx_sufficient ] = rep(FALSE, sum(idx_sufficient))

    if (sum(!idx_update)==length(pop.mean)){
      break
    }
  }

  # show results
  print(n_sufficient)
```

```
##  [1]   3  35  23  11   7  13  37  33  23 114  34  42  33   7  94
```

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	3
binomial	35
chisquared	23
exponential	11
f	7
gamma	13
geometric	37
hypergeometric	33
lognormal	23

distribution	N
negbinomial	114
normal	34
poisson	42
t	33
uniform	7
weibull	94