Problem Set 4

Marc Eskew

4/26/2022

Problem 1

1a

In order to use a Gaussian copula to sample (S, T) we have to establish functions and parameters to search for the correct value of ρ to satisfy $corr(S, T) = 0.2$.

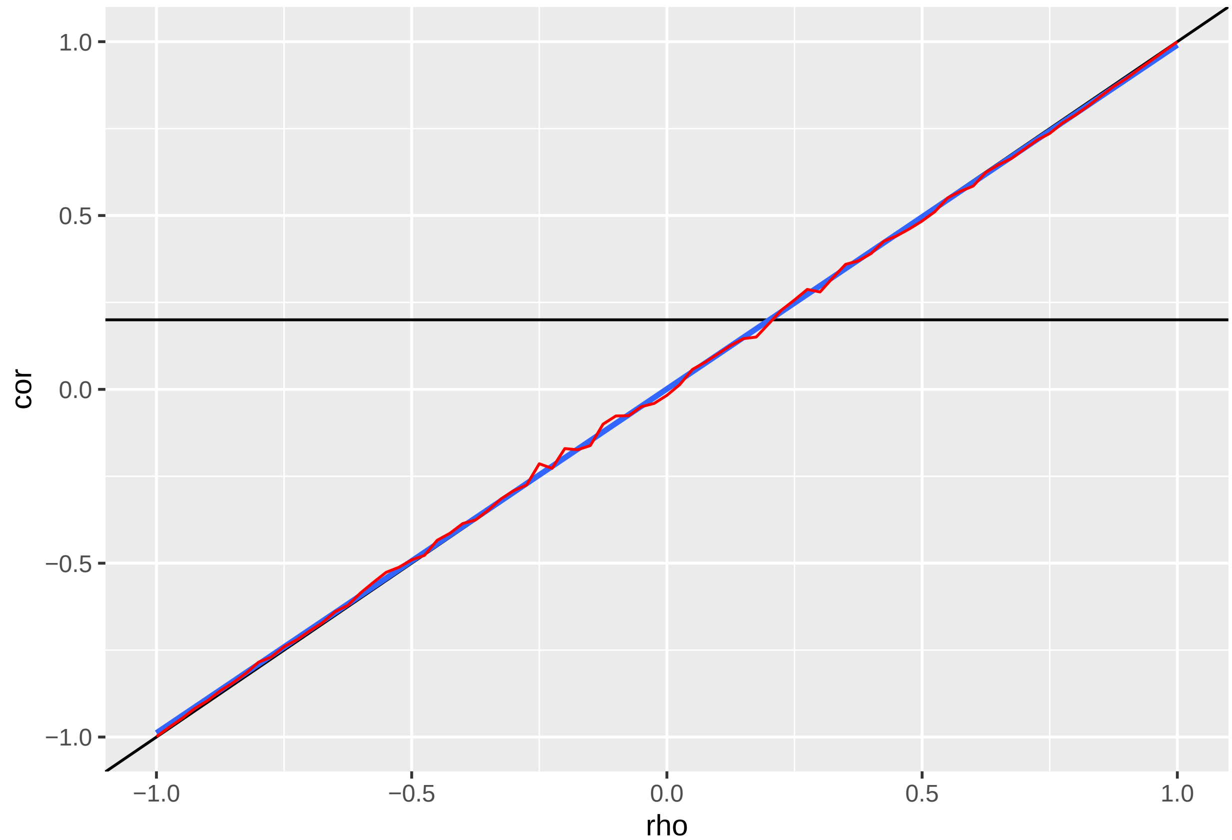
```
Y_fun <- function(rho,X,Z) {
  rho*X + sqrt(1-rho^2)*Z
}



T_fun <- function(Y){
  (990/(10*log(1-(Y))-11))+90
}

S_fun <-  function(Z){
  (90/(log(1-(Z))-1))+90
}

samples <- 10000
rho <- seq(-1,1, by = .025)
```

Iterating over our sequence of $\rho \in [-1, 1]$ and taking 10,000 samples using the Gaussian copula.

```
df_out <- data.frame()

for(i in rho) {
  df <- data.frame(x = rnorm(samples), z = rnorm(samples))

  df$Y <- mapply(Y_fun, i, df$x, df$z)
  df$U <- mapply(pnorm, df$Y)
  df$V <- mapply(pnorm, df$x)
  df$Tv <- mapply(T_fun, df$U)
  df$S <- mapply(S_fun, df$V)

  covdf <- cov(df$Tv,df$S)
  cov_T <- var((df$Tv))
  cov_S <- var((df$S))

  cor_df <- covdf/(sqrt(cov_S*cov_T))


  df_temp <- data.frame(rho=i,cor = cor_df)
```

```
  df_out <- bind_rows(df_out,df_temp)
}

ggplot(df_out, aes(x = rho, y = cor)) +
  geom_abline() +
  geom_hline(aes(yintercept = 0.2)) +
  geom_smooth(method = "lm") +
  geom_path(color = "red")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



$corr(S, T) = 0.2$ increases at a slightly lower rate than ρ, however the values are extremely close to be able to **approximate $\hat{\rho}$ at 0.2** to meet our conditions.

1b

Using $\hat{\rho} = 0.2$ from part a., we can estimate the price within 5% at 95% confidence.

First, as we have an unknown variance and error, we will dynamically generate those values. To start, a burn in period establishes initial values for ϵ and $\hat{\sigma}$. For each calculation of the error throughout this problem set, $\epsilon = \frac{1.96\hat{\sigma}}{\sqrt{n}}$.

```
pay_fun <- function(t,s){
  50000*exp(-.05*min(t,s))
}

err <- .05
```

```r
burn <- 100
rho <- .2
N <- burn
df_main <- data.frame()

df <- data.frame(x = rnorm(burn), z = rnorm(burn))

df$Y <- mapply(Y_fun, rho, df$x, df$z)
df$U <- mapply(pnorm, df$Y)
df$V <- mapply(pnorm, df$x)
df$Tv <- mapply(T_fun, df$U)
df$S <- mapply(S_fun, df$V)
df$val <-  mapply(pay_fun,df$Tv,df$S)

sum_val <- sum(df$val)
sum_val2 <- sum(df$val^2)
std_z <- (sum_val2/N - (sum_val/N)^2)^.5
err_cal <- 1.96*std_z/(N^.5)
err <- (sum_val/N)*.05
```

With these values, we calculate ϵ and $\hat{\sigma}$ in order to determine if we have an acceptable estimate. If not, we iterate and update values with additional simulations.

```r
while(err_cal > err) {
  N <- N + 1

  df <- data.frame(x = rnorm(1), z = rnorm(1))

  df$Y <- mapply(Y_fun, rho, df$x, df$z)
  df$U <- mapply(pnorm, df$Y)
  df$V <- mapply(pnorm, df$x)
  df$Tv <- mapply(T_fun, df$U)
  df$S <- mapply(S_fun, df$V)
  df$val <- mapply(pay_fun,df$Tv,df$S)

  sum_val <- sum_val + df$val
  sum_val2 <- sum_val2 + df$val^2
  std_z <- (sum_val2/N - (sum_val/N)^2)^.5
  err_cal <- 1.96*std_z/(N^.5)
  err <- (sum_val/N)*.05

}

sum_val/N
```

```
## [1] 19300.67
```

The price of the policy is 1.9300671×10^4 $\pm$ **5% with 95% confidence**.

Problem 2

2a

For this problem, our challenge is that there are a lot of different ways the states could vote as a collection. Given that each state could go R or D and there are 52 different states/provinces that vote, that means there are $2^{52} \approx 4.50e15$ different combinations. Iterating through every possible combination would not be entirely pleasant. So we will use simulation to estimate the value. Given a large enough sample, we can calculate the probability of a tie, P_t. Since the probability of a tie is equal to the number of possible ties by the number of total combinations, $P_t = \frac{N_t}{N}$, the number of ties is $N_t = P_t N$.

We initiate by generating a matrix of the value of each states electoral votes. Then we randomly assign outcomes with $p = 0.5$ to each state. The results are summed and checked for a tie and assigned a value 1, and 0 otherwise. For each set of X we calculate $P_t = \frac{1}{n}\sum_i^n X_i$.

Using CLT and the procedure described in question one, simulate values to get an estimate of N_t.

```r
half <- sum(elect$Votes)/2

burn <- 10000

N <- burn

mat_votes <- matrix(rep(elect$Votes,N),nrow = N,byrow = TRUE)

mat_p <- matrix(purrr::rbernoulli(N*52),nrow = N)

mat_t3 <- mat_p* mat_votes

df <- data.frame(votes=rowSums(mat_t3)) %>%
  mutate(val = ifelse(votes==half,1,0))

sum_val <- sum(df$val)
sum_val2 <- sum(df$val^2)
std_z <- (sum_val2/N - (sum_val/N)^2)^.5
err_cal <- 1.96*std_z/(N^.5)
err <- (sum_val/N)*.05

mat_votes <- matrix(elect$Votes,nrow = 1,byrow = TRUE)

while(err_cal > err) {
  N <- N + 1

  mat_p <- matrix(purrr::rbernoulli(52),nrow = 1)

  mat_t3 <- mat_p* mat_votes

  df <- data.frame(votes=rowSums(mat_t3)) %>%
    mutate(val = ifelse(votes==half,1,0))

  sum_val <- sum_val + df$val
  sum_val2 <- sum_val2 + df$val^2
  std_z <- (sum_val2/N - (sum_val/N)^2)^.5
  err_cal <- 1.96*std_z/(N^.5)
  err <- (sum_val/N)*.05

}

2^52*(sum_val/N)
```

```
## [1] 3.38017e+13
```

The number of combinations that result in ties is 3.3801695×10^{13} **± 5% with 95% confidence**.

2b

To determine the probability of a Republican victory by greater than or equal to 30 electoral votes, we slightly modify the procedure from part a. Instead of an outcome based on a coinflip for each state, we use the poll probability values provided in the dataset. As such, for each simulation the outcome is a Bernoulli RV with probability equal to the provided value for each state.

Additionally, instead of checking for $V_R = 269$, check for $V_R \geq 284$ which would be a 30 point victory. We are also looking for a probability, not the number of outcomes.

```
burn <- 100

N <- burn

mat <- matrix(rep(elect$PollProb,N),ncol = 52,byrow=TRUE)
mat_votes <- matrix(rep(elect$Votes,N),nrow = N,byrow=TRUE)

mat2 <- apply(mat,1:2,function(x){purrr::rbernoulli(1,x)})

mat_t3 <- mat2 * mat_votes

df <- data.frame(votes=rowSums(mat_t3)) %>%
  mutate(val = ifelse(votes>=half+15,1,0))

sum_val <- sum(df$val)
sum_val2 <- sum(df$val^2)
std_z <- (sum_val2/N - (sum_val/N)^2)^.5
err_cal <- 1.96*std_z/(N^.5)
err <- (sum_val/N)*.05

mat <- matrix(elect$PollProb,ncol = 52,byrow=TRUE)
mat_votes <- matrix(elect$Votes,nrow = 1,byrow=TRUE)

while(err_cal > err) {
  N <- N + 1

  mat2 <- apply(mat,1:2,function(x){purrr::rbernoulli(1,x)})

  mat_t3 <- mat2 * mat_votes

  df <- data.frame(votes=rowSums(mat_t3)) %>%
    mutate(val = ifelse(votes>=half+15,1,0))


  sum_val <- sum_val + df$val
  sum_val2 <- sum_val2 + df$val^2
  std_z <- (sum_val2/N - (sum_val/N)^2)^.5
  err_cal <- 1.96*std_z/(N^.5)
  err <- (sum_val/N)*.05

}

sum_val/N
```

```
## [1] 0.1599554
```

The number of combinations that result in ties is **0.1599554 ± 5% with 95% confidence**.