Statistical Computing, Coursework B

Zhao Yifei, s2002004

Part1:Archaeology

Question1

The main problem for the confidence interval is we directly use arch_likelihood_estimation() to compute lower and upper bound of the interval without condisering N should be greater than max(y1,y2), which means, for considering about confidence interval, we should estimate the parameters in the true space, hence here for N between zero to max(y1,y2), it is not reasonable to compute lower, upper, param in arch_likelihood_estimation(c(y1,y2)) directly, which makes the negative region for N weird.
We can clearly see although we use log-transformation for N in the internal calculations, and transform (-51,827) to (125,1199), which seems also not reasonable since $125 < 256 = max(y1, y2)$. Thereforem, the main violated assumption is the reasonable space for N should be greater than max(y1,y2), which means the left bound of confidence interval should be greater than 256.
I think another problem is the size of observations, it is not big enough, hence we do not have enough evidence to say asymptotically Normal distribution for N.

Question2

In this question, we firstly need to get the sample for phi, since phi have a beta(a,b), here a=2, b=2, using 1000 monte-carlo samples we can achieve phi using rbeta(1000,a,b,ncp=0).
Then, we consider calculating monte-carlo integrand. The form is

$$P(N = n, Y = y) = \int_0^1 P_N(n) P_\phi(\phi) P(Y = y | N = n, \phi) d\phi$$

Integrating the joint distribution for the three things, We can clearly see it is integral of three joint probability density functions. It involves prior parameters and conditional observation distribution probabilities, hence we want to use arch_loglike() supplied in CWB2020 code to compute the log-likelihood for multiple Binom(N,phi) observations, and exponential of it multiplied by dgeom() is the addition for loop.
The posterior probability is P(N=n,Y=y)/P(Y=y), after computing P_NY, the P_NY/sum(P_NY) is the posterior probability for all N since we use these two equations:

$$P_{N|Y}(n) = \frac{P(N = n, Y = y)}{P(Y = y)}$$

,

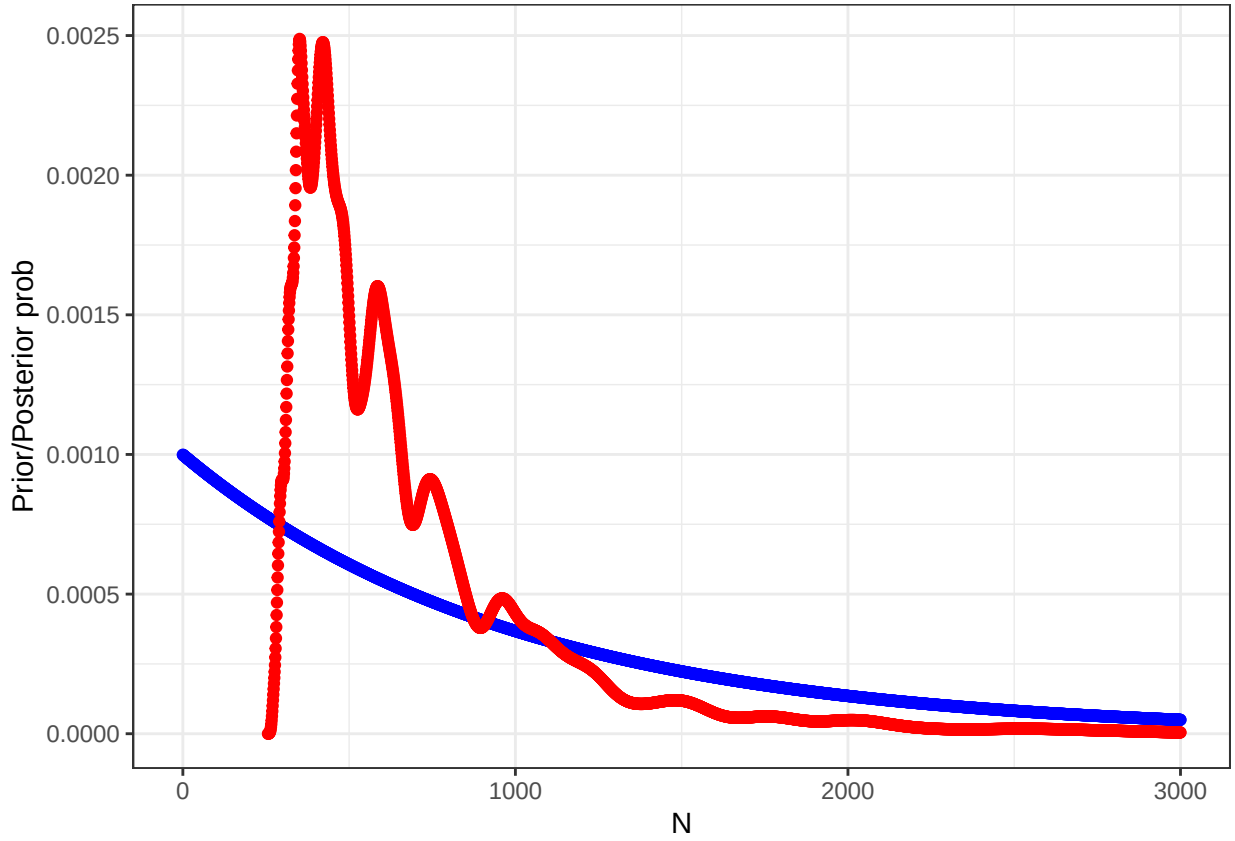
$$P(Y = y) = \sum_{n=max(y1,y2)}^{\infty} P(N = n, Y = y)$$

```
y <- c(256, 237)
N_xi <- 1/(1 + 1000)
phi_ab <- list(a = 2, b = 2)
N <- max(y):3000
n_mc <- 1000
```

```r
# Use the prior for phi for sampling:
phi <- rbeta(1000, 2, 2, ncp = 0)
P_NY <- numeric(length(N))
for (loop_phi in phi) {
# Compute and accumulate the integrand:
  P_NY <- P_NY + exp(arch_loglike(data.frame(N=N,phi=loop_phi), y))* dgeom(N, N_xi)
}
df <- data.frame(N = N,P_N_posterior = P_NY/sum(P_NY))
```

Question3

```r
library(ggplot2)
#set variable X different from N in Question2 because of the different support
X<-0:3000
ggplot()+
  geom_point(aes(X,dgeom(X,N_xi)),color="blue")+
  geom_point(aes(N,df$P_N_posterior),color="red")+
  xlab("N")+ylab("Prior/Posterior prob")
```



We use ggplot() to plot the posterior and prior probability, the blue graph seems smooth since the prior probability for N is geometry distribution. The red graph is for the posterior probability. We can clearly find as N becomes large, the red graph asymptotically tends to the blue one.
The support is the set of point where its probability is non-zero. In this question, the prior distribution

for N counts from N=0,1,2..., but for posterior distribution, N starts from 256, hence their supports are different.

Question4

For computing Bayesian 95% credible interval for N, consider the equation of sum in the question,

$$P(N \leq n|y_1, y_2) = \sum_{k=0}^{n} P(N = k|y_1, y_2)$$

Firstly, we use cumsum() to compute accumulated posterior probabilities of N, and then, for confidence level 95% in this question, we use which(T>0.975)[1]+256 to get the lowest index where the cumulative sum is greater than 0.975, the index shows the upper index of sum, by definition it is the right bound of credible interval. For the lower bound, we use length(which(T<0.025))+256 to get the highest index where the cumulative sum is less than 0.025, also by definition,it is the left bound of credible interval. Finally we claim these two index form the 95% credible interval of N.

```
#use cumsum to achieve accumulated probebilities for P_N_posterior
T<-cumsum(df$P_N_posterior)
a=length(which(T<0.025))+256
b=which(T>0.975)[1]+256
#show the results and test the credible level
data.frame(left_CI=a,right_CI=b,CI_level=T[b-256]-T[a-256])
```

```
##   left_CI right_CI  CI_level
## 1     309     1868 0.9502505
```

Therefore, looking at results, confidence intervals supplied in Question1 are $CI_N = (-51, 827)$, $CI_N = (125, 1199)$, credible interval we achieved is CI_N=(left_CI,right_CI). By comparing the result to the frequentistic confidence intervals the archaeologist computed, we find the credible interval is more reasonable since left_CI is greater than max(y1,y2).
What's more, in this real-world question, we want to say the 95% probability that the true parameter value N is in the interval, hence for Bayesian credible interval, it is more natural to say, there is a 95% probability that the total number of people buried is in the credible interval (left_CI,right_CI).

Part2:LOOCV and randomisation tests

Question1

We know it is reasonable to add two variables with Gaussian distribution of same size , for γ_i and e_i in this question, we can show the linear combination of these two variables equals to y_i with respect to Model B, $Y_i \sim N[\theta_1 + \theta_2 x_i, e^{\theta_3} + e^{\theta_4} x_i^2]$.To prove $Y_i = \alpha_0 + \gamma_i x_i + e_i$,we need to consider the expectation and variance of Y_i. Since here γ_i and e_i are parameters and x_i is observation, we get two equations,

$$E[y_i] = E[\alpha_0] + E[\gamma_i x_i] + E[e_i] = E[\alpha_0] + x_i E[\gamma_i] + 0 = \theta_1 + \theta_2 x_i$$

$$Var[y_i] = Var[\alpha_0] + Var[\gamma_i x_i] + Var[e_i] = 0 + x_i^2 Var[\gamma_i] + Var[e_i] = e^{\theta_3} + e^{\theta_4} x_i^2$$

By comparing both sides of two equations, we can claim Model B is mathematically equivalent to this error model and the parameters for γ_i, e_i are $\mu_\gamma = \theta_2$, $\sigma_\gamma^2 = e^{\theta_4}$, $\sigma_e^2 = e^{\theta_3}$

Question2

For the code filled in this question, we use estimate_and_predict() in CWB2020 to estimate the model and compute the predicition, for the input of this function, since we use leave-one-out cross validation, for this for-loop, we firstly split the 86 data into 86 subsets, then iterate into 86 subsets and treat each as a validation set, the remains 85 subsets are training data. Therefore, the inputs are data=filter(filament,Index!=ind), newdata=filter(filament,Index==ind), model=model, since $for(model\ in\ c("A","B"))$ shows model loops through two models, we can claim the double-loop achieves two estimates and computes two predictions with respect to model_A, model_B.

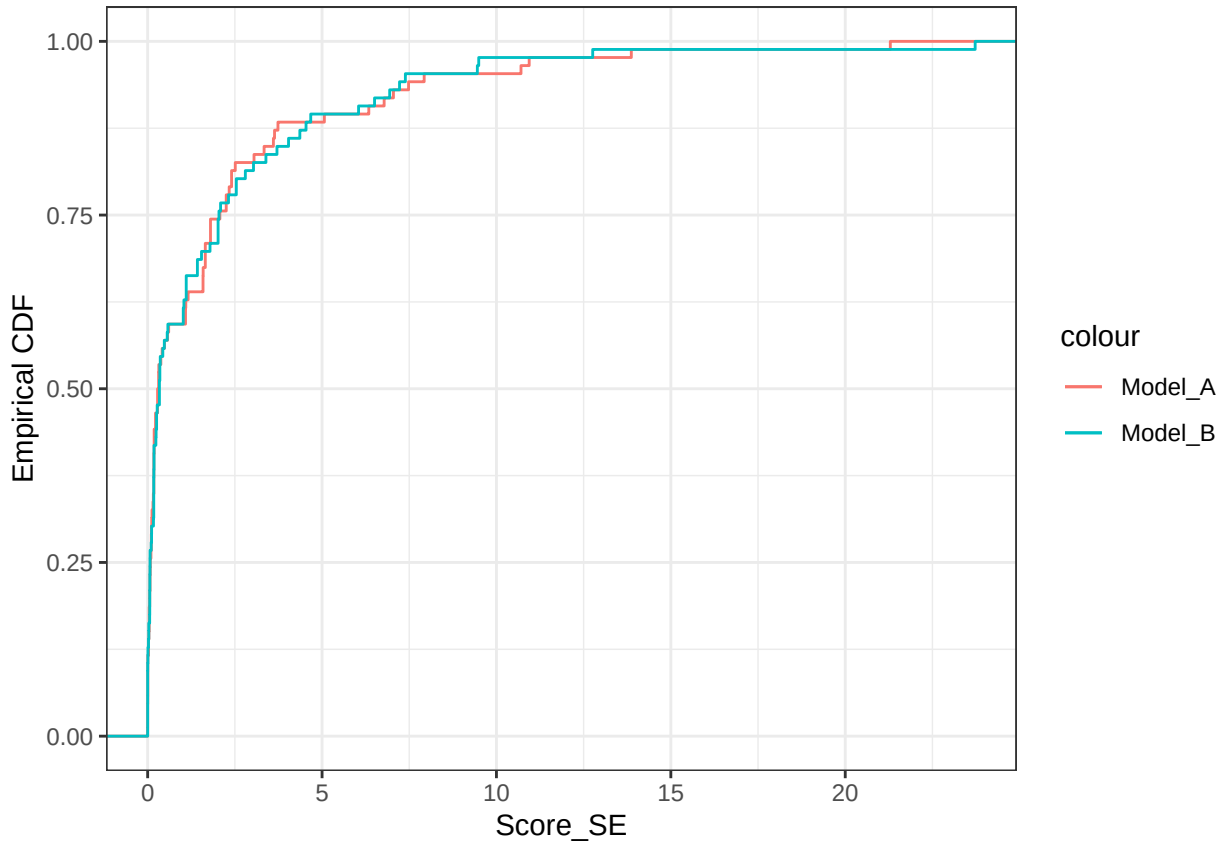
```
pred <- list(A = estimate_and_predict_output_template(nrow(filament)),
             B = estimate_and_predict_output_template(nrow(filament)))
for (ind in filament$Index) {
  for (model in c("A", "B")) {
# Leave out one observation, estimate the model, predict the left-out obs:
     pred[[model]][ind,] <- estimate_and_predict(filter(filament,Index!=ind),
                                                 filter(filament,Index==ind),
                                                 model = model,alpha = 0.1, df = Inf)
  }
}

scores <- list(A = calc_scores(pred[["A"]], filament[["Actual_Weight"]]),
               B = calc_scores(pred[["B"]], filament[["Actual_Weight"]]))
#compute scores SE,DS,Interval with respect to model_A and model_B
rbind(model_A=colMeans(scores$A), model_B=colMeans(scores$B))
```

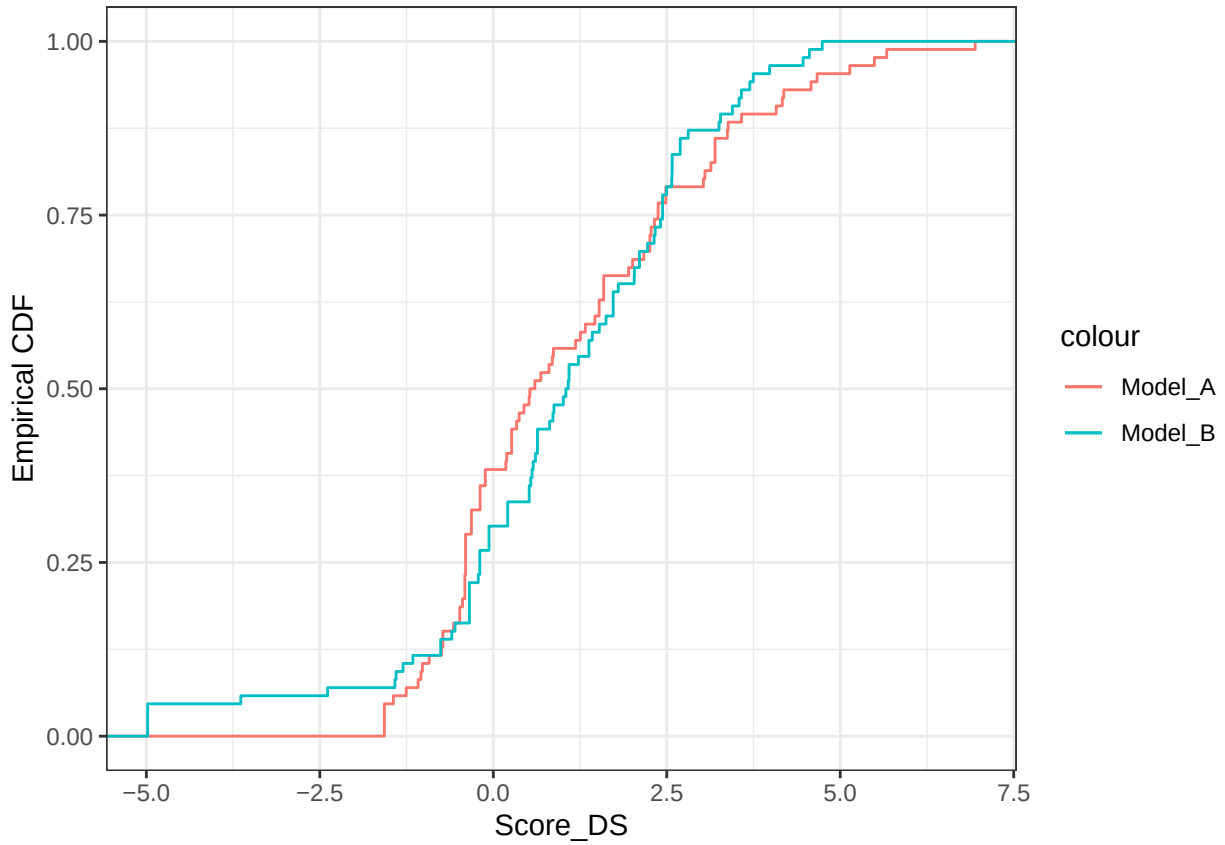
```
##               SE       DS Interval
## model_A 1.841734 1.128348 5.593457
## model_B 1.841995 0.939715 4.766243
```

Finally, we can compute two scores $S_i^A = S(F_i^A, y_i)$,$S_i^B = S(F_i^B, y_i)$ with respect to model_A and model_B. From the table above, we can see the insignificance of the differences for SE and DS between these two models, which were around 0.00026 and 0.19 respectively, but the score_interval difference is large, about 0.83, the detailed further test is in Question3. We can also show the detailed distributions ECDF of these scores as follows.

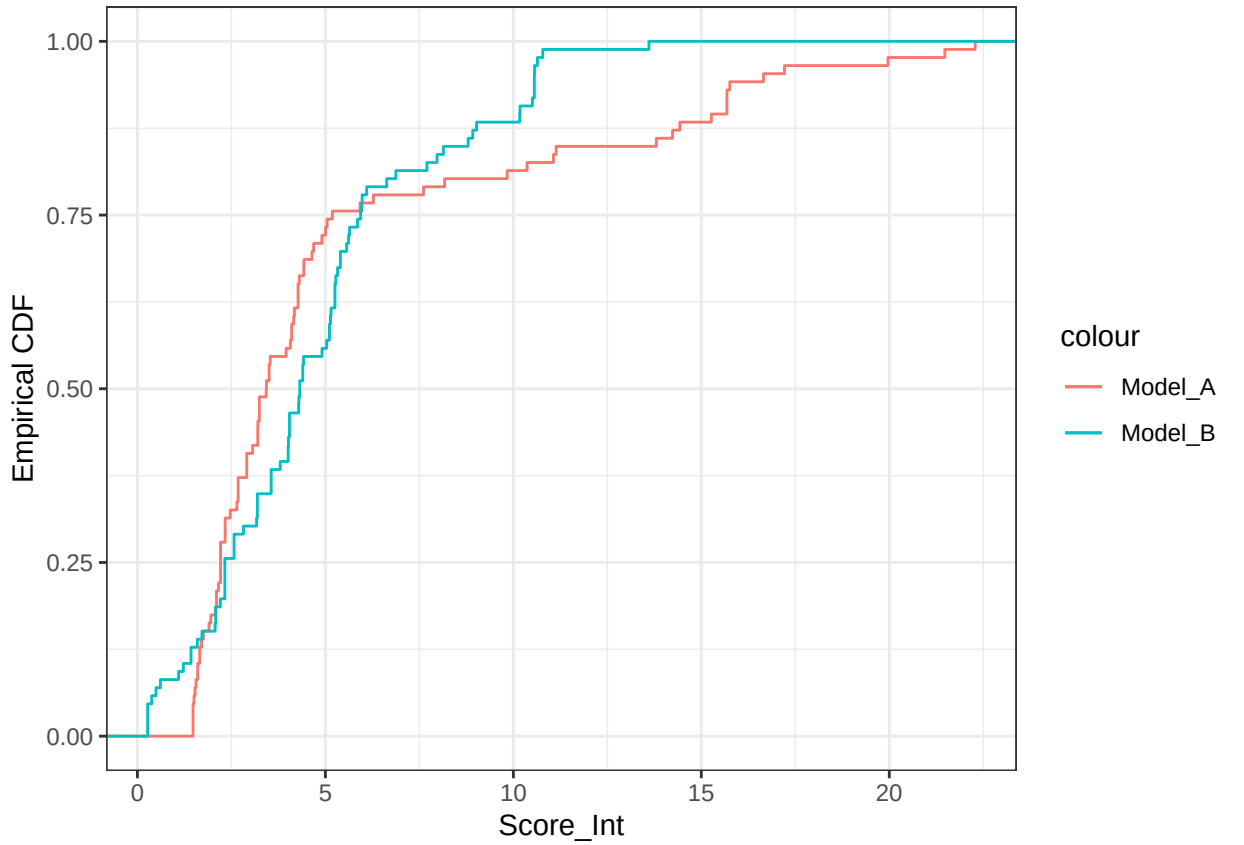
```
#ECDF for Score_SE
ggplot()+
  stat_ecdf(aes(x=as.vector(unlist(scores$A[1])),col="Model_A"))+
  stat_ecdf(aes(x=as.vector(unlist(scores$B[1])),col="Model_B"))+
  xlab("Score_SE")+ylab("Empirical CDF")
```




```
#ECDF for Score_DS
ggplot()+
  stat_ecdf(aes(x=as.vector(unlist(scores$A[2])),col="Model_A"))+
  stat_ecdf(aes(x=as.vector(unlist(scores$B[2])),col="Model_B"))+
  xlab("Score_DS")+ylab("Empirical CDF")
```




```
#ECDF for Score_Int
ggplot()+
  stat_ecdf(aes(x=as.vector(unlist(scores$A[3])),col="Model_A"))+
  stat_ecdf(aes(x=as.vector(unlist(scores$B[3])),col="Model_B"))+
  xlab("Score_Int")+ylab("Empirical CDF")
```



We can clearly see the plots are corresponding to the results in table. The two curves in the third plot are significantly different.

Question3

In this question, we need to compute the randomised test statistics. By looking at the 17th page of lec_notes6, for the first step, we use sample() to get 86 binomial indicators 1,-1, to achieve sign function for S_j^{Δ} over J loops, and then, we use the sample above multiplied by $S_j^{\Delta} = S_A^j - S_B^j$.Finally, we get the test statistics for each loop.
Assumptions for the test is based on the slides 15-17 in lec_notes6. In this question we test the score differences between model_A and model_B, we should assume under H_0 these two models have the same distribution, the joint sample of them is a collection of exchangeable variables. What's more, since J=10000 is large enough, so for the large test statistics rand_stat, we claim it can indicate the derivations from H_0.

```
# Compute the test statistics
stat <- colMeans(scores$A - scores$B)
J <- 10000
rand_stat <- data.frame(SE = numeric(J),
DS = numeric(J),
Interval = numeric(J))
# Compute the randomised test statistics
for (loop in seq_len(J)) {
rand_stat[loop,]<-colMeans(sample(c(1,-1),86,replace=TRUE,prob=c(1,1))*(scores$A-scores$B))
}
# Estimate the P-values for the tests
```

```r
p_value <- colMeans(rand_stat >= rep(stat, each = J))
#further investigation for comparing non-randomised data with randomised data
rbind(non_rand_data=stat,rand_data=colMeans(rand_stat))
```

```
##                            SE            DS     Interval
## non_rand_data -0.0002605523  0.1886325123  0.827213508
## rand_data      0.0010771275 -0.0007801381 -0.002979634
```

For the last line, we compare non-randomised data with randomised data to give further investigation about the pairwise differences between model_A and model_B with respect to different data. We find randomised data is better for testing H_0 since the scores are smaller. Therefore we can see, in this case, randomisation test is reasonable.

Question4

There are four sub-questions of Question4, the first one is for the proof, the remains are for computations and the code below shows the results of all values needed.
We just show the process of proof here. What we want to prove is $\sigma_T^2 = P_T(1-P_T)/J$. The basic theorem we use is De Moivre-Laplace Theorem, a special case of the central limit theorem. Applying theorem in this case, since Z^j are independent with Binomial(1,P_T), $E(Z^j) = P_T, Var(Z^j) = P_T(1-P_T)$, considering $\sum_{j=1}^{j=J} Z^j$, we get

$$\frac{\sum_{j=1}^{j=J} Z^j - J * P_T}{\sqrt{J * P_T(1-P_T)}} \sim N(0,1)$$

Now, we can get the distribution for $\sum_{j=1}^{j=J} Z^j$, it is $N(J * P_T, J * P_T(1-P_T)$. Then, we want to calculate the estimated variance of $\sum_{j=1}^{j=J} Z^j$ to get the expression of σ_T^2 of $\hat{P_T}$. We use the result

$$Var(\sum_{j=1}^{J} Z^j) = \sum_{i=1}^{J}\sum_{j=1}^{J} Cov(Z_i, Z_j) = \sum_{i=1}^{J} Var(Z^j) + 2 * \sum_{i,j:i<j} Cov(Z_i, Z_j)$$

and asymptotically we get$Var(\sum_{j=1}^{J} Z^j) = Var(J * \hat{P_T}) = J * P_T(1-P_T)$, since the estimate is σ_T^2 and $Z^{(j)} \sim^{i.i.d.} Bin(1, P_T)$. Finally, we can achieve $Var(\hat{P_T}) = \sigma_T^2 = P_T(1-P_T)/J$

```r
#compute standard deviation estimate for each test
std_dev<-sqrt(p_value*(1-p_value)/J)
#compute relative Monte Carlo error for each of the three p_value estimates
MC_error<-std_dev/p_value
#compute confidence interval for each test
#for p_value of SE
s1<-std_dev[1]
error1<-qnorm(0.975)*s1/sqrt(J)
SE_lwr<-p_value[1]-error1
SE_upr<-p_value[1]+error1
```

```r
#for p_value of DS
s2<-std_dev[2]
error2<-qnorm(0.975)*s2/sqrt(J)
DS_lwr<-p_value[2]-error2
DS_upr<-p_value[2]+error2
#for p_value of Interval
s3<-std_dev[3]
error3<-qnorm(0.975)*s3/sqrt(J)
Int_lwr<-p_value[3]-error3
Int_upr<-p_value[3]+error3
#plot the table about three Confidence Intervals
#Y1<-rbind(c(SE_lwr,SE_upr),c(DS_lwr,DS_upr),c(Int_lwr,Int_upr))
Y1<-rbind(SE_lwr,DS_lwr,Int_lwr)
Y2<-rbind(SE_upr,DS_upr,Int_upr)
X<-cbind(std_dev,MC_error,Y1,Y2,Y2-Y1,p_value)
colnames(X)<-c("std_dev","MC_error","95%CI_left","95%CI_right","Length_CI","p_value")
knitr::kable(X)
```

	std_dev	MC_error	95%CI_left	95%CI_right	Length_CI	p_value
SE	0.0049986	0.0097628	0.5119020	0.5120980	0.0001959	0.5120
DS	0.0020398	0.0468919	0.0434600	0.0435400	0.0000800	0.0435
Interval	0.0007261	0.1369961	0.0052858	0.0053142	0.0000285	0.0053

Looking at the table, we find lengths of three CI are all small which indicates it is mainly accurate to get the estimate $\hat{P}_T$. For the std_dev and MC_error, we can say all the three scores are proper scores.
The assumption of Part2 shows model_A and model_B have the same expectations but different variance. Considering the value of minimized scores, we can claim Score_SE has been minimised, but Score_DS can be minimised only when the variances are the same which also indicates these two models are equivalent. For Score_int, it is consistent for equal-tail error probability interval, here is the equal-tail 95% confidence interval, and it will be minimised when the length of coverage interval is narrowest. Considering Length_CI and p_value, we find the ratio 0.0000285/0.0053 is largest which is corresponding to the MC_error of Score_Int is biggest and std_dev of Score_Int is smallest.