STAT30850 Homework 2

Sarah Adilijiang

Problem 1

(a) BH functions

```
# general modified BH method for each group
modified_BH = function(P, alpha, gamma){
    n = length(P)
    order_index = order(P, decreasing=FALSE)
    pi_hat = sum(P>gamma)/n/(1-gamma)
    P[P>gamma] = Inf    # never reject any p > gamma
    ks = which(P[order_index] <= (1:n)*alpha/n/pi_hat)
    if(length(ks)==0){
        return(NULL)
    }else{
        k = max(ks)
        rej_index = order_index[1:k]
        return(rej_index)
    }
}

# group adaptive BH function
group_adaptive_BH = function(P, group_sizes, alpha, gamma) {
    n = length(P)
    K = length(group_sizes)
    rej_index = NULL
    inx = 1
    for (k in 1:K) {
        group_rej_index = modified_BH(P[inx:(inx+group_sizes[k]-1)], alpha, gamma)
        rej_index = c(rej_index, c(inx:(inx+group_sizes[k]-1))[group_rej_index])
        inx = inx + group_sizes[k]
    }
    return(rej_index)
}
```

(b) Simulations

```
# function for simulating data
simulate_data = function(group_sizes, sig_proportions){
    n = sum(group_sizes)
    K = length(group_sizes)
    Signal = P = NULL
    for (k in 1:K) {
        n_sig = round(group_sizes[k]*sig_proportions[k], digits=0)
        p_sig = 2*pnorm(abs(rnorm(n_sig,2,1)),lower.tail=FALSE)
        P = c(P, p_sig, runif(group_sizes[k]-n_sig))
        Signal = c(Signal, rep(1,n_sig), rep(0,group_sizes[k]-n_sig)) # 0:null, 1:signal
    }
    return(list(Signal=Signal, P=P))
}
```

```r
# function for calculating the average FDP & average power
average_FDP_and_power = function(group_sizes, sig_proportions, alpha, gamma, sim_num){
    FDP = power = rep(0,sim_num)
    for (i in 1:sim_num) {
        data = simulate_data(group_sizes, sig_proportions)
        rej_index = group_adaptive_BH(data$P, group_sizes, alpha, gamma)
        FDP[i] = sum(data$Signal[rej_index]==0) / length(rej_index)
        power[i] = sum(data$Signal[rej_index]==1) / sum(data$Signal)
    }
    return(list(average_FDP=mean(FDP), average_power=mean(power)))
}
```

```r
set.seed(1)
group = average_FDP_and_power(group_sizes=rep(50,20),
                              sig_proportions=c(rep(0,10),rep(0.5,10)),
                              alpha=0.1, gamma=0.5, sim_num=1000)
single = average_FDP_and_power(group_sizes=1000,
                              sig_proportions=0.25,
                              alpha=0.1, gamma=0.5, sim_num=1000)
# report results
results = matrix(NA,2,2)
colnames(results) = c("average_FDP","average_power")
rownames(results) = c("group-adaptive BH","single-group BH")
results["group-adaptive BH",] = c(group$average_FDP, group$average_power)
results["single-group BH",] = c(single$average_FDP, single$average_power)
results
```

```
##                   average_FDP average_power
## group-adaptive BH  0.11425627      0.520292
## single-group BH    0.09410266      0.279840
```

Results:

We can see that the average FDP values of the two methods are similar with each other, though this value of the group-adaptive BH is a little bit higher.

However, the average power values of the two methods are quite different. The average power of the group-adaptive BH method is nearly twice of that of the single-group BH method. This is because the group-adaptive BH method will calculate the $\hat{\pi}_0$ for each group which makes it perform better in discovering the true signals without increasing too much of the FDR (average FDP).

(c) Different settings

(1) Question 1

What is the effect of non-uniform groups, i.e. groups where the proportion of signals is varied from one group to another?

Let's set the proportion of signals to be varied from one group to another. Here I use the randomly generated signal proportions for each group, repeat the experiment with ten sets of random proportions and calculate the average of the results for the two BH methods.

```r
set.seed(1)
group_averages = single_averages = matrix(NA,10,2)
for (i in 1:10) {
    sig_proportions = sample(10, 20, replace=TRUE)/10
    group = average_FDP_and_power(group_sizes=rep(50,20),
```

```
                                  sig_proportions,
                                  alpha=0.1, gamma=0.5, sim_num=1000)
    single = average_FDP_and_power(group_sizes=1000,
                                  mean(sig_proportions),
                                  alpha=0.1, gamma=0.5, sim_num=1000)
    group_averages[i,] = c(group$average_FDP, group$average_power)
    single_averages[i,] = c(single$average_FDP, single$average_power)
}

# report results
results["group-adaptive BH",] = colMeans(group_averages)
results["single-group BH",] = colMeans(single_averages)
results
```

```
##                   average_FDP average_power
## group-adaptive BH  0.06487176     0.6718992
## single-group BH    0.08157015     0.5529946
```

Results:

Here we can see that, when the true proportion of signals is varied from one group to another, the group-adaptive BH method performs obviously better than the single-group BH method in having lower average FDP value and higher average power value than those of the single-group BH method.

(2) Question 2

What is the effect of group size for both power and FDR control-in particular, for smaller groups, are the $\hat{\pi}_0^g$'s reliable enough for FDR control?

In previous question (b), there are 20 groups and each group has same group size of 50 p-values. Now Let's try different group sizes from 10 to 200, while still keeping the number of groups to be 20.

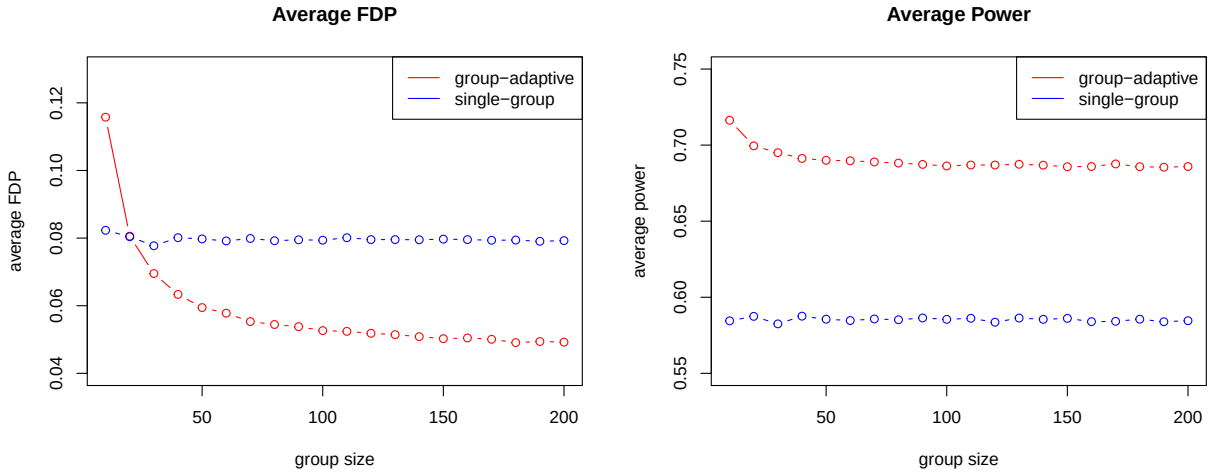
Notice that here I use a randomly generated proportion of signals.

```
set.seed(1)
sizes = seq(10,200,10)
sig_proportions = sample(10, 20, replace=TRUE)/10

group_size_averages = matrix(NA,length(sizes),4)
for (i in 1:length(sizes)) {
    n = sizes[i] * 20
    group = average_FDP_and_power(group_sizes = rep(sizes[i],20),
                                  sig_proportions,
                                  alpha=0.1, gamma=0.5, sim_num=1000)
    single = average_FDP_and_power(group_sizes = n,
                                  mean(sig_proportions),
                                  alpha=0.1, gamma=0.5, sim_num=1000)
    group_size_averages[i,] = c(group$average_FDP, group$average_power,
                                single$average_FDP, single$average_power)
}
```

```
# report results by plotting
par(mfrow=c(1,2))
plot(sizes, group_size_averages[,1], col=2, type="b", ylim=c(0.04,0.13),
     xlab="group size",ylab="average FDP",main="Average FDP")
lines(sizes, group_size_averages[,3], col=4, type="b")
legend("topright", legend=c("group-adaptive","single-group"), col=c(2,4), lty=1)
```

```
plot(sizes, group_size_averages[,2], col=2, type="b", ylim=c(0.55,0.75),
     xlab="group size",ylab="average power",main="Average Power")
lines(sizes, group_size_averages[,4], col=4, type="b")
legend("topright", legend=c("group-adaptive","single-group"), col=c(2,4), lty=1)
```



Results:

We can see that the single-group BH method generates almost similar average FDP values when using different group sizes. Same results for the average power values. Therefore, the group size almost has no effect on the single-group BH method.

However, the group-adaptive BH method is more sensitive to the group size changes. As the group size increases, the average FDP value decreases dramatically when the group size is small and then decreases slowly after the group size is larger than 50. On the other hand, the average power value decrease a little when group size is small and then almost keep the same value after group size is larger than 50. Therefore, the group-adaptive BH method is very sensitive to the group size especially when group size is small, where the estimated $\hat{\pi}_0^g$'s are not reliable enough for FDR control. Especially, when the group size is 10, the average FDP value is even larger than 0.1.

Furthermore, when group size is larger than 50, the group-adaptive BH method always performs better than the single-group BH method in having lower average FDP values and higher average power values than those of the single-group BH method.

(3) Question 3

What is the effect of γ value for both power and FDR control?

Now Let's try different values of γ from 0.1 to 0.9, while still keeping the number of groups to be 20. Notice that here I use a randomly generated proportion of signals and a group size of 50.

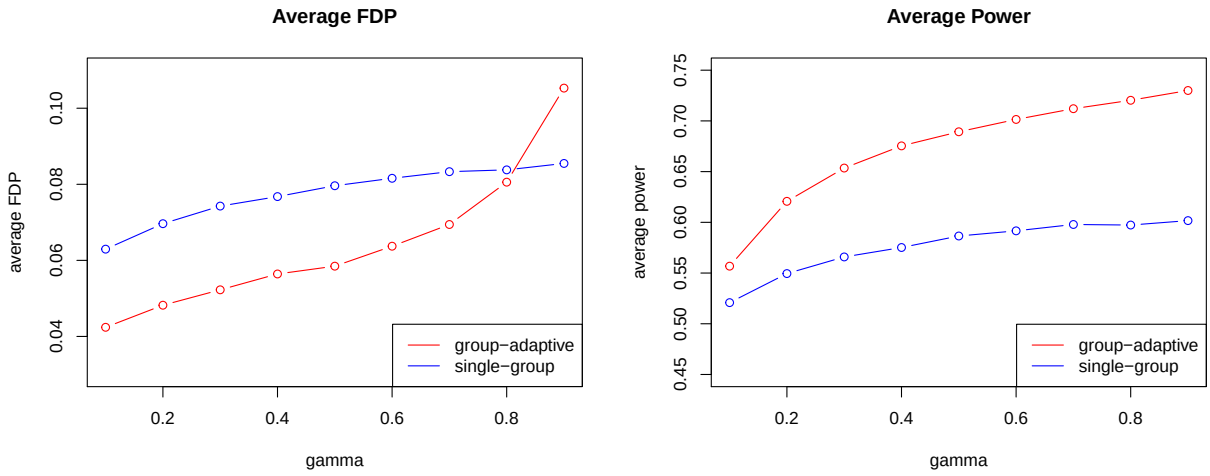
```
set.seed(1)
gammas = seq(0.1,0.9,0.1)
sig_proportions = sample(10, 20, replace=TRUE)/10

gamma_averages = matrix(NA,length(gammas),4)
for (i in 1:length(gammas)) {
    group = average_FDP_and_power(group_sizes = rep(50,20),
                                  sig_proportions,
                                  alpha=0.1, gammas[i], sim_num=1000)
```

```
    single = average_FDP_and_power(group_sizes = 1000,
                                   mean(sig_proportions),
                                   alpha=0.1, gammas[i], sim_num=1000)
    gamma_averages[i,] = c(group$average_FDP, group$average_power,
                                  single$average_FDP, single$average_power)
}
```

```
# report results by plotting
par(mfrow=c(1,2))
plot(gammas, gamma_averages[,1], col=2, type="b", ylim=c(0.03,0.11),
     xlab="gamma",ylab="average FDP",main="Average FDP")
lines(gammas, gamma_averages[,3], col=4, type="b")
legend("bottomright", legend=c("group-adaptive","single-group"), col=c(2,4), lty=1)

plot(gammas, gamma_averages[,2], col=2, type="b", ylim=c(0.45,0.75),
     xlab="gamma",ylab="average power",main="Average Power")
lines(gammas, gamma_averages[,4], col=4, type="b")
legend("bottomright", legend=c("group-adaptive","single-group"), col=c(2,4), lty=1)
```



Results:

For the single-group BH method, we can see that as the γ value increases, both the average FDP value and average power value increase, though the increasing rate is slowing down as well.

For the group-adaptive BH method, generally, the group-adaptive BH method is more sensitive to the γ value changes. And again, as the γ value increases, both the average FDP value and average power value increase. However, the increasing rate is slowing down for the power value, while it is first slowing down before $\gamma = 0.5$ then getting larger after $\gamma = 0.5$. Therefore, it's better to choose $\gamma = 0.5$ for the group-adaptive BH method.

Furthermore, when γ is between 0.1 and 0.8, the group-adaptive BH method always performs better than the single-group BH method in having lower average FDP values and higher average power values than those of the single-group BH method.

These two values both increase in the two methods because as the γ value increases, the estimated $\hat{\pi}_0$ or $\hat{\pi}_0^g$'s will decrease, thus more signals and nulls will be rejected, which results in larger FDP values and power values.

Problem 2

(a)

$$\begin{aligned}\text{BayesFDR}(t) &= \mathbb{P}(P \text{ came from null } | P \le t) \;=\; \frac{\mathbb{P}(P \text{ came from null},\; P \le t)}{\mathbb{P}(P \le t)} \\ &= \frac{\mathbb{P}(P \le t \mid P \text{ came from null}) \times \pi_0}{\mathbb{P}(P \le t \mid P \text{ came from null}) \times \pi_0 + \mathbb{P}(P \le t \mid P \text{ came from non-null}) \times (1-\pi_0)} \\ &= \frac{\pi_0 t}{\pi_0 t + (1-\pi_0) t^{\epsilon}}\end{aligned}$$

Then, to control FDR at the level α, we set:

$$\text{BayesFDR}(t) = \frac{\pi_0 t}{\pi_0 t + (1-\pi_0) t^{\epsilon}} = \alpha$$

So:

$$\hat{t}_{\text{Bayes}} = \left[\frac{\pi_0(1-\alpha)}{\alpha(1-\pi_0)}\right]^{1/(\epsilon-1)}$$

(b)

$$\begin{aligned}N\{\text{p-values} \le t\} \;&\approx E\left[N\{\text{p-values} \le t\}\right] = \sum_{i=1}^{n} E[I_{P_i \le t}] = \sum_{i=1}^{n} \mathbb{P}(P_i \le t) \\ &= \sum_{i=1}^{n} [\pi_0 t + (1-\pi_0) t^{\epsilon}] \;=\; n\,[\pi_0 t + (1-\pi_0) t^{\epsilon}]\end{aligned}$$

$$\Rightarrow \qquad \widehat{\text{FDP}}(t) = \frac{nt}{N\{\text{p-values} \le t\}} \approx \frac{nt}{n[\pi_0 t + (1-\pi_0) t^{\epsilon}]} = \frac{1}{\pi_0 + (1-\pi_0) t^{\epsilon-1}}$$

By setting $\widehat{\text{FDP}}(t) = \alpha$, we get that:

$$\hat{t}_{\text{BH}} = \left[\frac{1-\alpha\pi_0}{\alpha(1-\pi_0)}\right]^{1/(\epsilon-1)}$$

(c)

According to the Storey's modification of the BH procedure, we have:

$$\hat{\pi}_0 = \frac{N\{\text{p-values} > 1-\gamma\}}{n\gamma} = \frac{N\{\text{p-values} > 0.5\}}{0.5\; n} = \frac{2}{n} \times N\{\text{p-values} > 0.5\}$$

So:

$$\begin{aligned}E[\hat{\pi}_0] &= \frac{2}{n} \times E\left[N\{\text{p-values} > 0.5\}\right] = \frac{2}{n} \sum_{i=1}^{n} E[I_{P_i > 0.5}] = \frac{2}{n} \sum_{i=1}^{n} \mathbb{P}(P_i > 0.5) \\ &= \frac{2}{n} \sum_{i=1}^{n} [\pi_0(1-0.5) + (1-\pi_0)(1-0.5^{\epsilon})] \;=\; 2\,[0.5\pi_0 + (1-\pi_0)(1-0.5^{\epsilon})] \\ &= 2\,[1 - 0.5\pi_0 - (1-\pi_0)0.5^{\epsilon}] \\ &= 2 - \pi_0 - (1-\pi_0)0.5^{\epsilon-1}\end{aligned}$$

(d)

Use the results from question (b), we get that:

$$\widehat{\text{FDP}}(t) = \frac{\hat{\pi}_0\ nt}{N\{\text{p-values} \leq t\}} \approx \frac{E[\hat{\pi}_0]}{\pi_0 + (1-\pi_0)t^{\epsilon-1}} = \frac{2 - \pi_0 - (1-\pi_0)0.5^{\epsilon-1}}{\pi_0 + (1-\pi_0)t^{\epsilon-1}}$$

By setting $\widehat{\text{FDP}}(t) = \alpha$, we get that:

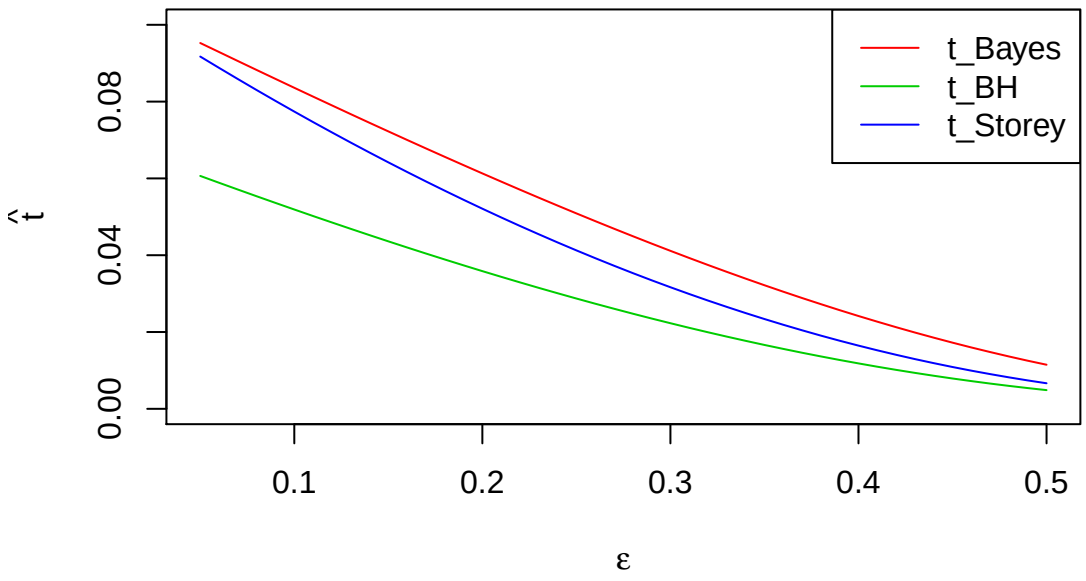
$$\hat{t}_{\text{Storey}} = \left[\frac{2 - \pi_0(1+\alpha) - (1-\pi_0)0.5^{\epsilon-1}}{\alpha(1-\pi_0)}\right]^{1/(\epsilon-1)}$$

(e)

```
t_Bayes = function(alpha, pi, epsilon){
    t = ( pi*(1-alpha)/alpha/(1-pi) )^(1/(epsilon-1))
    return(t)
}
t_BH = function(alpha, pi, epsilon){
    t = ( (1-alpha*pi)/alpha/(1-pi) )^(1/(epsilon-1))
    return(t)
}
t_Storey = function(alpha, pi, epsilon){
    numerator = 2 - pi*(1+alpha) - (1-pi)*0.5^(epsilon-1)
    t = (numerator/alpha/(1-pi))^(1/(epsilon-1))
    return(t)
}
```

```
alpha = 0.2
pi = 0.7
epsilons = seq(0.05,0.5,0.01)
Bayes = BH = Storey = rep(NA,length(epsilons))
for (i in 1:length(epsilons)) {
    Bayes[i] = t_Bayes(alpha, pi, epsilons[i])
    BH[i] = t_BH(alpha, pi, epsilons[i])
    Storey[i] = t_Storey(alpha, pi, epsilons[i])
}

plot(epsilons, Bayes, col=2, type="l", ylim=c(0,0.1),
     xlab=expression(epsilon),ylab=expression(hat(t)))
lines(epsilons, BH, col=3, type="l")
lines(epsilons, Storey, col=4, type="l")
legend("topright", legend=c("t_Bayes","t_BH","t_Storey"), col=c(2,3,4), lty=1)
```



Results:

We can see that as the ϵ value increases, the $\hat{t}$ values calculated from the three methods will all decrease. And at the same ϵ value, we have that: $\hat{t}_{\text{Bayes}} > \hat{t}_{\text{Storey}} > \hat{t}_{\text{BH}}$.

(f)

Since:

$$\text{FDR}(t) = \frac{\pi_0 t}{\pi_0 t + (1-\pi_0)t^{\epsilon}} = \frac{\pi_0}{\pi_0 + (1-\pi_0)t^{\epsilon-1}}$$

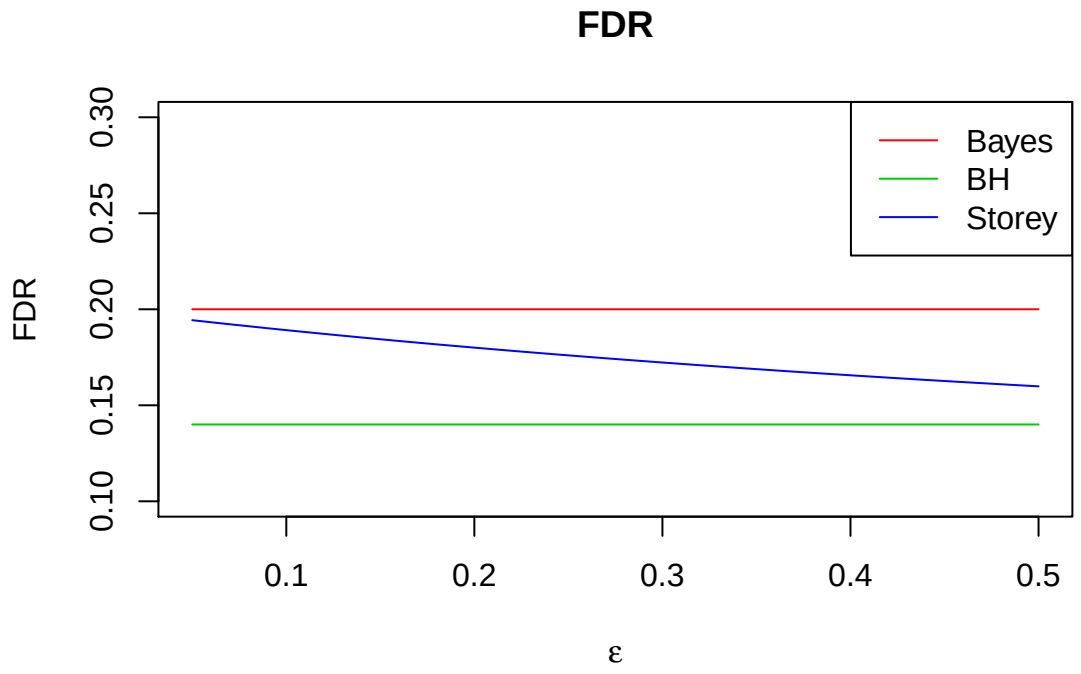
$$\text{Power}(t) = \mathbb{P}(P \leq t \mid P \text{ came from non-null}) = t^{\epsilon}$$

So we can plot the FDR and power values.

```
FDR = function(t, pi, epsilon){
    return(pi / ( pi+(1-pi)*t^(epsilon-1) ))
}
power = function(t, epsilon){
    return(t^epsilon)
}
```

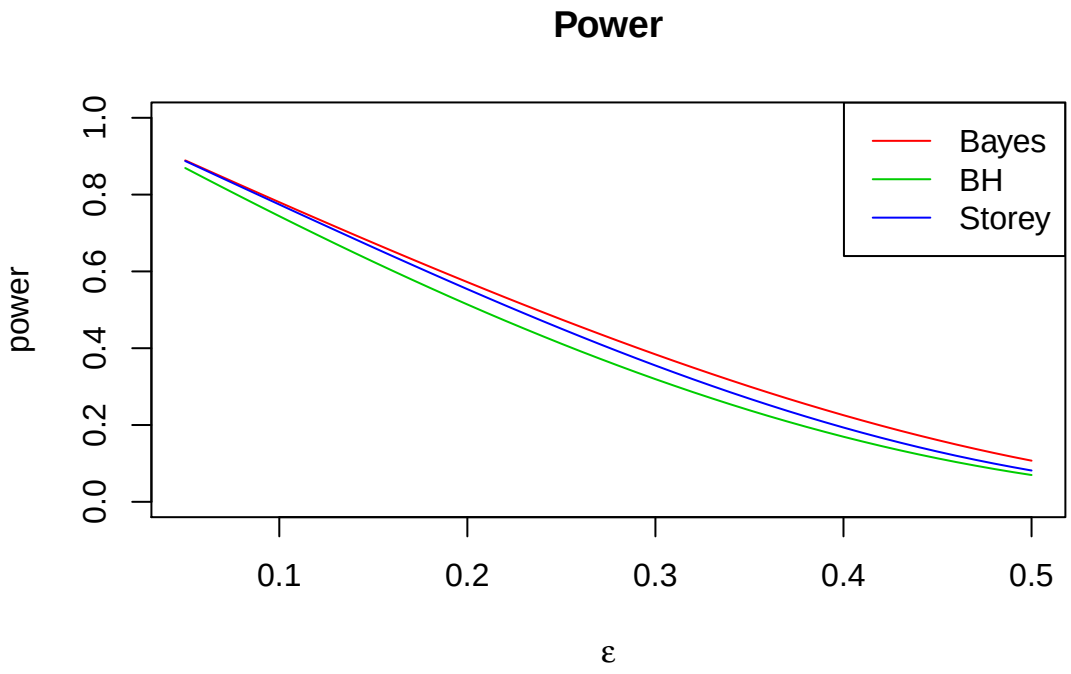
```
alpha = 0.2
pi = 0.7
epsilons = seq(0.05,0.5,0.01)
Bayes = BH = Storey = rep(NA,length(epsilons))
for (i in 1:length(epsilons)) {
    Bayes[i] = FDR(t_Bayes(alpha, pi, epsilons[i]), pi, epsilons[i])
    BH[i] = FDR(t_BH(alpha, pi, epsilons[i]), pi, epsilons[i])
    Storey[i] = FDR(t_Storey(alpha, pi, epsilons[i]), pi, epsilons[i])
}
```

```r
plot(epsilons, Bayes, col=2, type="l", ylim=c(0.1,0.3),
     xlab=expression(epsilon),ylab="FDR",main="FDR")
lines(epsilons, BH, col=3, type="l")
lines(epsilons, Storey, col=4, type="l")
legend("topright", legend=c("Bayes","BH","Storey"), col=c(2,3,4), lty=1)
```




```r
alpha = 0.2
pi = 0.7
epsilons = seq(0.05,0.5,0.01)
Bayes = BH = Storey = rep(NA,length(epsilons))
for (i in 1:length(epsilons)) {
    Bayes[i] = power(t_Bayes(alpha, pi, epsilons[i]), epsilons[i])
    BH[i] = power(t_BH(alpha, pi, epsilons[i]), epsilons[i])
    Storey[i] = power(t_Storey(alpha, pi, epsilons[i]), epsilons[i])
}

plot(epsilons, Bayes, col=2, type="l", ylim=c(0,1),
     xlab=expression(epsilon),ylab="power",main="Power")
lines(epsilons, BH, col=3, type="l")
lines(epsilons, Storey, col=4, type="l")
legend("topright", legend=c("Bayes","BH","Storey"), col=c(2,3,4), lty=1)
```



Results:

(1)

The FDR value of Bayes method does not change with ϵ, whose value is the same as level α. And the FDR value of BH method also does not change with ϵ, whose value is smaller than the level α. However, the FDR value of Storey method decreases when the ϵ value increases, though it is also always lower than the level α.

Furthermore, at the same ϵ value, we have that: $\text{FDR}_{\text{Bayes}} > \text{FDR}_{\text{Storey}} > \text{FDR}_{\text{BH}}$.

(2)

We can see that as the ϵ value increases, the power values from the three methods will all decrease. And at the same ϵ value, we have that: $\text{Power}_{\text{Bayes}} > \text{Power}_{\text{Storey}} > \text{Power}_{\text{BH}}$.

Problem 3

```
coverage_rate = function(sim_num, N, p, alpha){
    p_hats = rmultinom(sim_num, size=N, prob=p) / N
    coverages = rep(0,sim_num)
    for (i in 1:sim_num) {
        max_p_inx = which.max(p_hats[,i])
        max_p = p_hats[,i][max_p_inx]
        z = qnorm(alpha/2, 0, 1,lower.tail=FALSE)
        CI_left =  max_p - z * sqrt(max_p*(1-max_p)/N)
        CI_right = max_p + z * sqrt(max_p*(1-max_p)/N)
        true_p = p[max_p_inx]
        if((true_p>=CI_left) & (true_p<=CI_right)){ coverages[i] = 1}
    }
    return(mean(coverages))
}
```

(a)

```
set.seed(1)
p1 = rep(1/20,20)
coverage_rate(sim_num=10000, N=1000, p=p1, alpha=0.1)
```

```
## [1] 0.4132
```

```
set.seed(1)
p2 = 1/(1:20)
coverage_rate(sim_num=10000, N=1000, p=p2/sum(p2), alpha=0.1)
```

```
## [1] 0.8951
```

Results:

For $p_1 = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$, the coverage rate is about 41.32%, which is much smaller than the target 90% coverage rate. In this case, all the probablities of voting for each candidate are the same, i.e. the true $p_{\hat{i}}$ for any winning candidate will always be $1/20 = 0.05$. But according to how we pick the winning candidate, the maximun value of the votes is picked, so the selected $\hat{p}_i$ will be the largest one that is probably the most far away one from the true value 0.05. As a result, the 90% confidence interval of this quite biased $\hat{p}_i$ will have a lower chance of containing the true value $p_{\hat{i}}$.

For $p_2 = (1, \frac{1}{2}, \cdots, \frac{1}{n})$/(normalizing constant), the coverage rate is about 89.51%, which is very close to the target 90% coverage rate. In this case, all the probablities of voting for each candidate are different from each other, so the candidate with highest p will most likely to win, and its estimation $p_{\hat{i}}$ should be closer to its true value $p_{\hat{i}}$.

(b)

```
set.seed(1)
p1 = rep(1/20,20)
coverage_rate(sim_num=10000, N=1000, p=p1, alpha=0.1/20)
```

```
## [1] 0.9868
```

```
set.seed(1)
p2 = 1/(1:20)
coverage_rate(sim_num=10000, N=1000, p=p2/sum(p2), alpha=0.1/20)
```

```
## [1] 0.9932
```

Results:

We can see that for both $p_1 = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$ and $p_2 = (1, \frac{1}{2}, \cdots, \frac{1}{n})$/(normalizing constant), the coverage rates are as high as about 99%, which is much higher than the target 90% coverage rate and pretty close to 100%. Therefore, this corrected CI is too conservative.