Wk5_CorrectMultipleTests

Isaac Sowah Badu

2024-09-25

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.3.3
```

```
opts_knit$set(root.dir = "C:/Users/sowai23/Desktop/Year_2_First sem/R stats/Files/R-class-2024/WK05_Cor
```

```
# Defining a function
my.fun <- function(x){
y <- x*5 + 1
return(y)
}


# Using the function created
my.fun(10)
```

```
## [1] 51
```

```
# A function with two parameters
my.fun2 <- function(x, a){
y <- x*5 + 3^a
return(y)
}
my.fun2(10, 5)
```

```
## [1] 293
```

```
# What happens when you switch them around with, and without, specification? I got the same answers wit
my.fun2(a=5, x=10)
```

```
## [1] 293
```

```
## [1] 293
```

```
my.fun3 <- function(x){
y <- x*5 + 3^a
return(y)
}
```

#Checkpoint 1: Call my.fun2(10) and my.fun3(10): what happens? Now, specify a <- 5, and call my.fun2(10) and myfun3(10) again. Which one works and why? What if x is also an object in R?

```
##my.fun2(10)
```

```
#my.fun3(10)
```

#my.fun2(10) says argument "a" is missing, with no default and my.fun3(10) says object 'a' not found

```
##specifying
a <- 5
```

```
my.fun2(10, 5)
```

```
## [1] 293
```

```
##my.fun3(10, 5)
```

#my.fun2(10) now works but my.fun3(10) does not work. my.fun2(10) works now because we have given it an argument 5 which was initially missing. my.fun3(10) does not work because the function does not have an object a.

```
my.fun3 <- function(x){
y <- x*5 + 3^a
return(y)
}
```

```
#Writing the function called Bonf()
Bonf <- function(p.vect) {
  bonf1 <- 0.05 / length(p.vect)
  return(data.frame(p.vect, Is.Sig= p.vect < bonf1))
}

#usage
p.vect <- c(0.23, 0.02, 0.013, 0.045)
Bonf(p.vect)
```

```
##   p.vect Is.Sig
## 1  0.230  FALSE
## 2  0.020  FALSE
## 3  0.013  FALSE
## 4  0.045  FALSE
```

```
#Benjamini-Hotchberg
BenHotch <- function(p.vect){
rank.p <- rank(p.vect)
n <- length(p.vect)
rank.alpha<-rank.p*0.05/n
sig.p <- max(p.vect[p.vect<rank.alpha])
return(data.frame(p.vect, Is.Sig = p.vect<=sig.p))
}
```

```r
p.vect <- c(0.23, 0.02, 0.013, 0.045)
BenHotch(p.vect)
```

```
##   p.vect Is.Sig
## 1  0.230  FALSE
## 2  0.020   TRUE
## 3  0.013   TRUE
## 4  0.045  FALSE
```

```r
#if (!require("BiocManager", quietly = TRUE))
#install.packages("BiocManager")
#BiocManager::install("qvalue")
```

```r
library(qvalue)
```

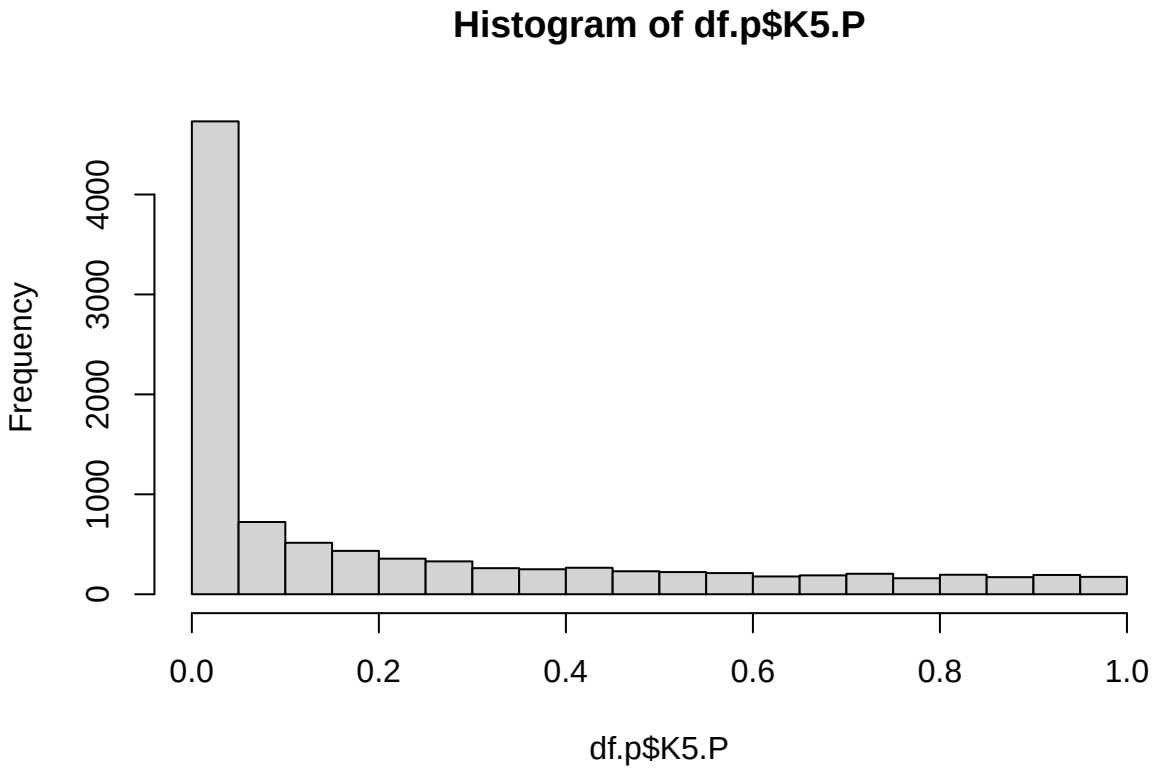
```r
library(knitr)
opts_knit$set(root.dir = "C:/Users/sowai23/Desktop/Year_2_First sem/R stats/Files/R-Class-2024/WK05_Cor
```

```r
###Uploading the data
df.p <- read.table("data/1R_P30_1351142954_453_1_NumPops=30_NumInd=20Bayenv2LFMMpca-1.Cpval", header=TR
```

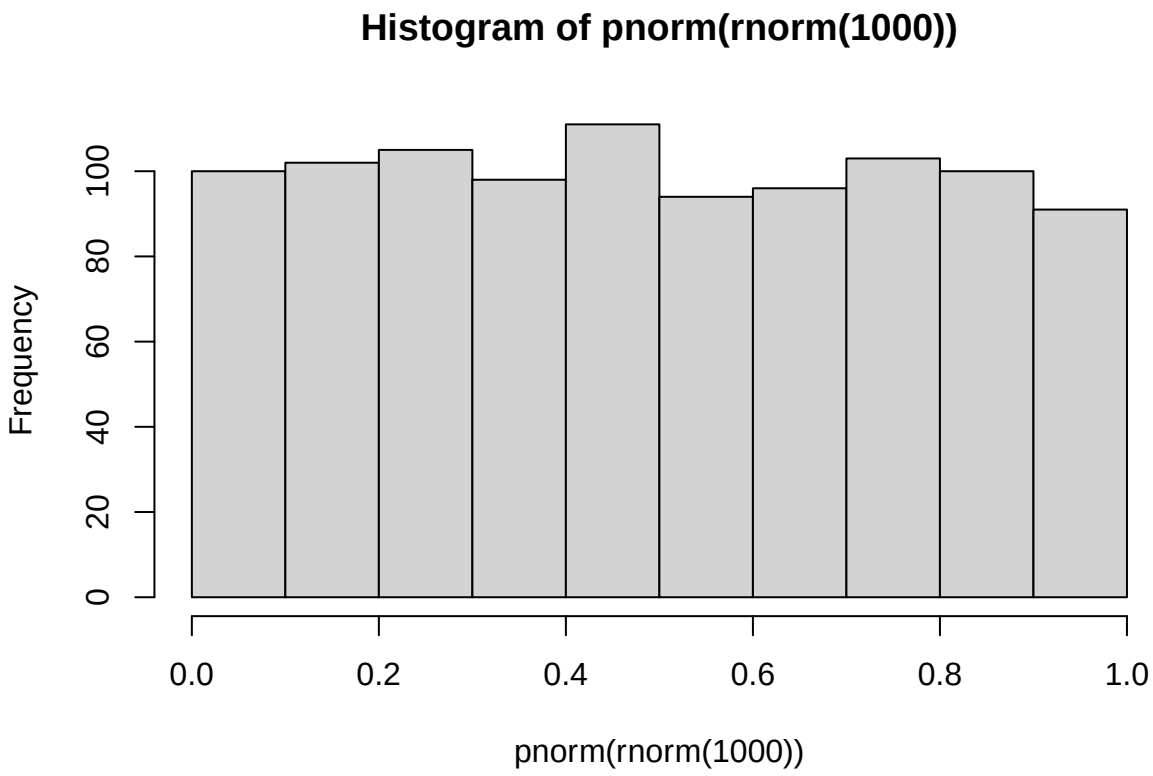
```r
# A logical indicating if a locus is neutral (TRUE)
IsNeut <- df.p$IsNeut[df.p$UseSNP]

# The p-value for each locus calculated by a program
p.list <- df.p$K5.P[df.p$UseSNP]

### We can check the assumption that our p-values are uniformly distributed
### by plotting a histogram:
hist(df.p$K5.P)
```




```
##Comparing figure 1 in Storey and Tibshirani (2003), where there is only a small inflation at p<0.01.
### Compare it to a histogram of p-values from a normal distribution:
hist(pnorm(rnorm(1000)))
```




```
### Making a logical list of whether the q-values are <0.05
q.out <- qvalue(p.list)$qvalue<0.05
```

```
#Creating a function called CorrectMultP() that takes a vector of p-values as input and outputs columns
CorrectMultP <- function(p.vect){
return(data.frame(
Bonf = Bonf(p.vect),
BenHotch = BenHotch(p.vect),
Qval = qvalue(p.vect)$qvalue<0.05))
}

out <- CorrectMultP(p.list)
head(out)
```

```
##   Bonf.p.vect Bonf.Is.Sig BenHotch.p.vect BenHotch.Is.Sig  Qval
## 1 9.13690e-08        TRUE     9.13690e-08            TRUE  TRUE
## 2 1.87369e-02       FALSE     1.87369e-02            TRUE  TRUE
## 3 1.59969e-01       FALSE     1.59969e-01           FALSE FALSE
## 4 3.40020e-10        TRUE     3.40020e-10            TRUE  TRUE
## 5 4.08625e-01       FALSE     4.08625e-01           FALSE FALSE
## 6 1.19112e-04       FALSE     1.19112e-04            TRUE  TRUE
```

```
### Calculate the number of significant tests
sum(out$Bonf.Is.Sig)
```

```
## [1] 1112
```

```
sum(out$BenHotch.Is.Sig)
```

```
## [1] 3972
```

```
sum(out$Qval)
```

```
## [1] 5102
```

#Checkpoint 3: Which test is the most conservative? least conservative? The true dataset only had 100 loci under selection. #Bonferroni is the most conservative with the fewest significant result and q-value is the least conservative with most significant results.