pMaxTestForMike

Tom Booker

20/04/2021

Replicate the results from Mike's Mathematica Notebook

```r
p_value_adjustment <- function(p_vec){
  p_adj = 1 - (1 - min(p_vec))^length(p_vec)
  return(p_adj)
}

p_jth <- function(plist,j) {
  # get the list of the p-value vector
  k=length(plist)
  # return an NA if the j parameter is bigger than the length of the p vector
  if(j>k) {return(NA)}
  # sort the p-values
  pjth=sort(plist)[j]
  # scale the probabilities
  prob = if(j<k) (pjth /sort(plist)[j+1]) else max(plist)
  # perform the test
  dbinom(x = j, size = j, prob=prob )
}

pMax <- function(pVec){
  outVec <- rep(1, length(pVec))
  for (r in 1:length(pVec)){
    outVec[r] =  p_jth(pVec, r)
  }
  return(outVec)
}

pMaxAll <- function(pVec){
  p_value_adjustment(pMax(pVec))
}

pMaxIdent <- function(pVec){
  pMax_results <- pMax(pVec)
  if (min(pMax_results) == pMax_results[1]){
    return(0)
  }
  else{
    return(which.min(pMax_results))
  }
}

FisherCombProb <- function( pVec ){
  testStat = -2*sum(log( pVec ))
  pVal = 1 - pchisq( testStat, 2*length( pVec ) )
  return( pVal )
}
```

Now I'll test the exact same case as Mike to compare p-vals

```r
pValues <- c(0.001,0.2,0.5,0.8, 0.5)

FisherCombProb(pValues)
```

```
## [1] 0.02694623
```

```r
p_value_adjustment( pMax(pValues) )
```

```
## [1] 0.02475125
```

Yep, those work. I find that there I get the exact same p-values as Mike did

Now I'll do lots of replicate simulations to see if this holds

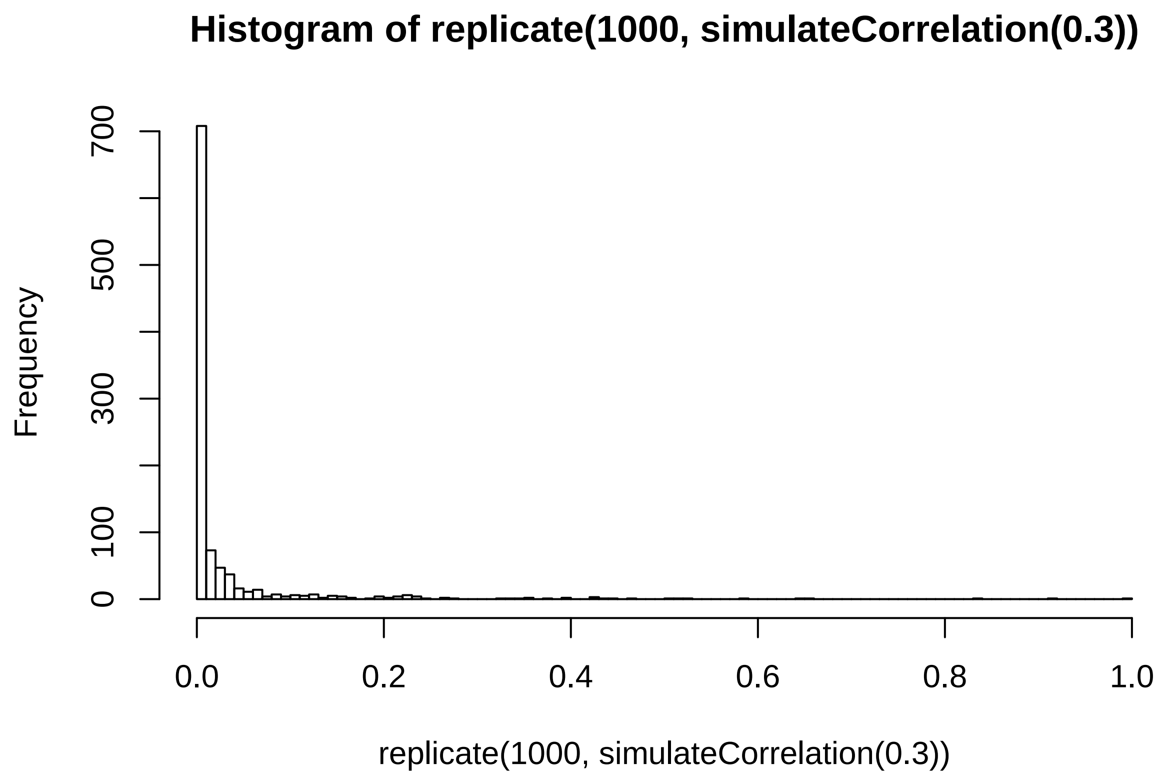
```r
## Now let's simulate some GEAs
# This is not how Mike did his, but I think it's an Ok strategy

simulateCorrelation <- function(r, n = 100){
  # r is the population correlation coefficient
  # n is the number of demes

  # x1 is the sample data
  x1 = rnorm(n)
  # x2 is the data that will be transformed so as to be correlated with x1
  x2 = rnorm(n)

  # transform x2 to Y, a vector of values with population correlation r to x1
  Y = r*x1+sqrt(1-r*r)*x2
  return( cor.test(Y, x1)$p.value )
}

## Plot the histogram for a single correlation to see the distribution of p-values
hist( replicate(1000, simulateCorrelation( 0.3 )), breaks = 100)
```



```
# Simulate data with 4 false nulls, 1000 times
oneFalseNull <- cbind(  replicate(1000, simulateCorrelation(0.35) ) ,
                        replicate(1000, simulateCorrelation(0.35) ) ,
                        replicate(1000, simulateCorrelation(0.35) ) ,
                        replicate(1000, simulateCorrelation(0.35) ),
                        replicate(1000, simulateCorrelation(0.0) ))
## This is what the input data for the WZA looks like:
head(oneFalseNull, n=1)
```

```
##              [,1]       [,2]         [,3]        [,4]      [,5]
## [1,] 0.0005200511 0.02080678 8.329021e-05 0.001879311 0.1993723
```

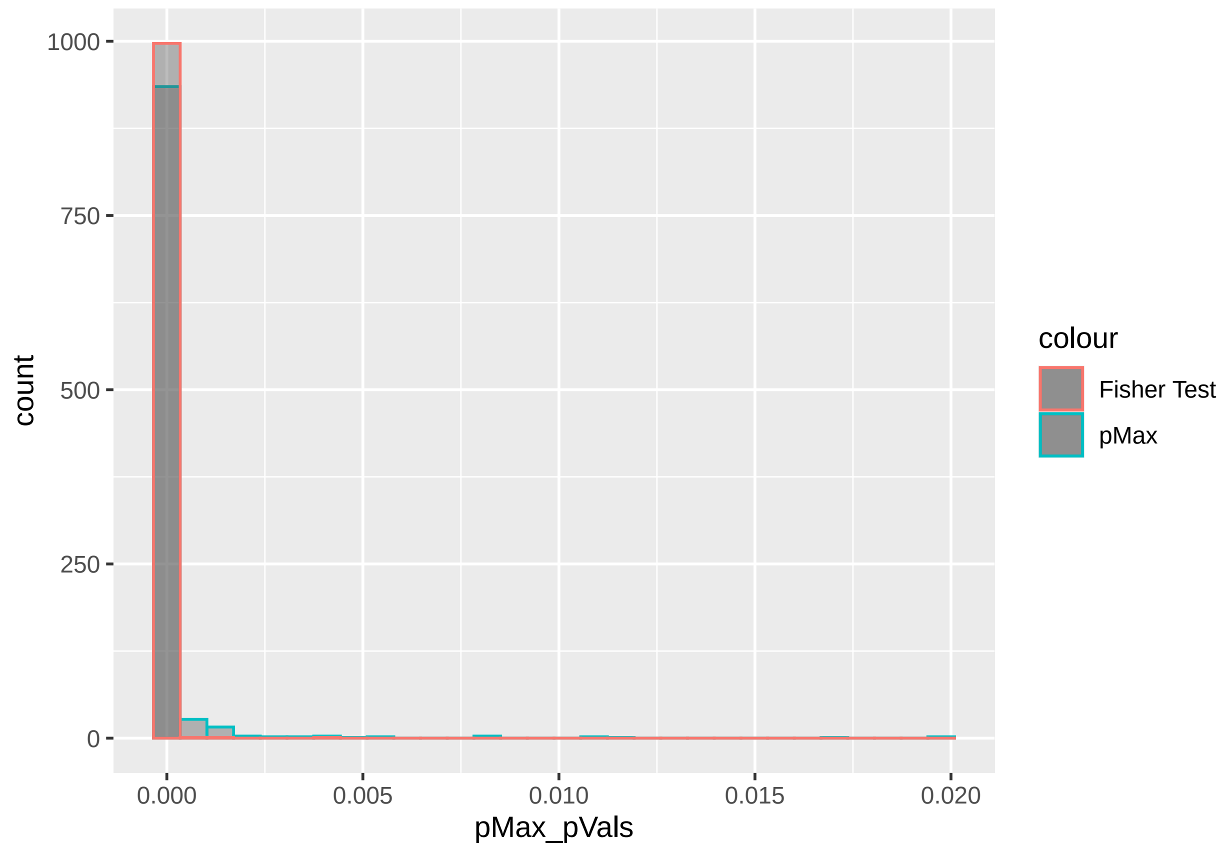
Now plot a histogram comparing the Fisher test to the pMax for this case with 4 false nulls and a single true null

```
pMax_pVals <- apply(oneFalseNull, 1, pMaxAll )

Fisher_pVals <- apply(oneFalseNull, 1, FisherCombProb )

library(ggplot2)

ggplot(data = data.frame(pMax_pVals), aes( x = pMax_pVals))+
  geom_histogram(alpha = 0.4, aes(col = "pMax"))+
  geom_histogram(data = data.frame(Fisher_pVals), aes( x = Fisher_pVals, col = "Fisher Test"), alpha =
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Unlike Mike's result, we find that the Fisher test is more powerful.

Obviously there is a difference in what Mike did and what I've done, he used a beta distribution to generate p-values , I simulated a correlation. They are both kind of hacky, but if the pMax were uniformly better we'd expect it to outperform Fisher here.