GRADUATE STUDENT STAT 840 A2

Vsevolod Ladtchenko 20895137

Problem 4

a)

```
get_DT = function(x, y)
{
  m = length(x)
  n = length(y)

  xbar = mean(x)
  ybar = mean(y)

  D_obs = xbar - ybar

  Sp2 = (sum((x - xbar)^2) + sum((y - ybar)^2)) / (n + m - 2)
  Sp = sqrt(Sp2)

  T_obs = (xbar - ybar) / (Sp * sqrt(1/m + 1/n))

  return(c(D_obs, T_obs))
}

a = 0.05 # alpha
m = 10 # x samples
n = 10 # y samples
N = 1000 # number of simulations
B = 1000 # number of bootstrap/permutation runs

results = data.frame(matrix(ncol = 4, nrow = N))
colnames(results) = c('boot_D', 'boot_T', 'perm_D', 'perm_T')

for (j in 1:N)
{
  # initial sample

  x = rnorm(m, 0, 1)
  y = rnorm(n, 0, 1)

  DT_ = get_DT(x, y)
  D_obs = DT_[1]
  T_obs = DT_[2]

  # resampling: bootstrap & permutation
```

```
  DT_stars =  data.frame(matrix(ncol = 4, nrow = B))
  colnames(DT_stars) = c('boot_D', 'boot_T', 'perm_D', 'perm_T')

  xy = c(x,y)

  for (i in 1:B)
  {
    xsb = sample(xy, m, replace=T) # x star bootstrap
    ysb = sample(xy, n, replace=T)
    xsp = sample(xy, m, replace=F) # x star permutation
    ysp = sample(xy, n, replace=F)

    DT_stars[i,] = c(get_DT(xsb, ysb), get_DT(xsp, ysp))
  }

  #D_boot = sum(abs(DT_stars[,1]) >= abs(D_obs)) / B
  #T_boot = sum(abs(DT_stars[,2]) >= abs(T_obs)) / B
  #D_perm = sum(abs(DT_stars[,3]) >= abs(D_obs)) / B
  #T_perm = sum(abs(DT_stars[,4]) >= abs(T_obs)) / B
  #results[j,] = c(D_boot, T_boot, D_perm, T_perm) <= a

  # take a transpose here because comparison is done along vertical vectors
  results[j,] = (rowSums(t(abs(DT_stars)) >= abs(c(D_obs, T_obs))) / B) <= a
}

colSums(results) / N
```

```
## boot_D boot_T perm_D perm_T
##  0.054  0.052  0.179  0.161
```

We see that with the bootstrap procedure, both D and T have approximate type 1 error probability around 0.05, as would be expected, while D and T for the permutation tests yield much higher probability of around 17%.

b)

```
results = data.frame(matrix(ncol = 4, nrow = N))
colnames(results) = c('boot_D', 'boot_T', 'perm_D', 'perm_T')

for (j in 1:N)
{
  # initial sample

  x = rnorm(m, 0, 1)
  y = rnorm(n, 1, 1)

  DT_ = get_DT(x, y)
  D_obs = DT_[1]
  T_obs = DT_[2]

  # resampling: bootstrap & permutation

  DT_stars =  data.frame(matrix(ncol = 4, nrow = B))
```

```r
  colnames(DT_stars) = c('boot_D', 'boot_T', 'perm_D', 'perm_T')

  xy = c(x,y)

  for (i in 1:B)
  {
    xsb = sample(xy, m, replace=T) # x star bootstrap
    ysb = sample(xy, n, replace=T)
    xsp = sample(xy, m, replace=F) # x star permutation
    ysp = sample(xy, n, replace=F)

    DT_stars[i,] = c(get_DT(xsb, ysb), get_DT(xsp, ysp))
  }

  # take a transpose here because comparison is done along vertical vectors
  results[j,] = (rowSums(t(abs(DT_stars)) >= abs(c(D_obs, T_obs))) / B) <= a
}

colSums(results) / N
```

```
## boot_D boot_T perm_D perm_T
##  0.564  0.547  0.775  0.759
```

Now that the null hypothesis went from being true to being false, this simulation is now testing the power of the test, meaning the probability of correctly rejecting a false null hypothesis. We see that for bootstrap, both D and T yield around 58%, while the permutation tests yield around 79%. It is hard to say which of D or T is better since they are similar. However, there is a large difference between Bootstrap and Permutation tests.