ps5

Sijun Zhang

2019/12/1

Question 1

```r
library(data.table)

# read in the  data: -----------------------------------------------------------
## This data will be used in the question.
url_base <- 'https://www.eia.gov/consumption/residential/data/2015/csv/'

recs_file <- './recs2015_public_v4.csv'
if ( !file.exists(recs_file) ) {
  recs_url <- sprintf('%s/recs2015_public_v4.csv', url_base)
  recs <- readr::read_delim(recs_url, delim = ',')
  readr::write_delim(recs, path = recs_file, delim = ',')
} else {
  recs <- readr::read_delim(recs_file, delim = ',')
}
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   METROMICRO = col_character(),
##   UATYP10 = col_character(),
##   CLIMATE_REGION_PUB = col_character(),
##   IECC_CLIMATE_PUB = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```r
recs = data.table(recs)

decode_division = function(x) {
  if(!is.numeric(x)) stop('decode_divsion expects numeric input indexed from 1!')
  y <- x
  re <- switch (y,
                'New England',
                'Middle Atlantic',
                'East North Central',
                'West North Central',
                'South Atlantic',
                'East South Central',
                'West South Central',
                'Mountain North',
                'Mountain South',
                'Pacific')
  return(re)
}

decode_all_division = function(x) {
  return(sapply(x,decode_division))
}

collapse_UC = function(x) {
  if (x == "C") {re <- "U"}
  else {re <- x}
  return(re)
}

collapse_all_UC = function(x) {
  sapply(x, collapse_UC)
}

# Key values for each observation: --------------------------------------------
internet_recs <- recs[, .(DOEID = DOEID, dvi = decode_all_division(DIVISION),
                          utype = collapse_all_UC(UATYP10),
                          internet = INTERNET, weight = NWEIGHT)]

# Convert weights to long: ----------------------------------------------------
brrwts = names(recs)[which(names(recs) == "BRRWT1"):
                       which(names(recs) == "BRRWT96")]

weights_long <- recs[, .SD, .SDcols = c(1, which(names(recs) == "BRRWT1"):
                                          which(names(recs) == "BRRWT96"))]
weights_long <- melt(weights_long, id.vars = c("DOEID"), measure.vars = brrwts)

# Join home type to weights: --------------------------------------------------
internet_weighted <- merge(weights_long, internet_recs, by='DOEID', all.x = TRUE)

if( nrow(weights_long) != nrow(internet_weighted) ) {
  stop("DOEID mismatch!")
}

internet_weighted <- internet_weighted[, `:=`(brrwt_with_internet = internet*value,
                                              nw_with_internet = internet*weight)
                     ][, .(brrwt = sum(value), nweight = sum(weight),
                       nw_with_internet = sum(nw_with_internet),
                       brrwt_with_internet = sum(brrwt_with_internet),
                       repl = variable), by=.(dvi, utype, variable)
                     ][, `:=`(prop_repl = brrwt_with_internet/brrwt,
                       prop = nw_with_internet/nweight)
                     ][, .(dvi, utype, repl, prop_repl, prop)]


# calc the confidence interval using brrwt replications
internet_weighted <- dcast(internet_weighted, dvi+repl~utype, value.var=c("prop_repl","prop") )

internet_weighted_ci <-  internet_weighted[, `:=`(diff_prop = prop_U-prop_R,
                                                  diff_prop_repl = prop_repl_U-prop_repl_R)
                         ][, `:=`(rsq_prop1 = (prop_U-prop_repl_U)^2/(1-0.5)^2,
                           rsq_prop2 = (prop_R-prop_repl_R)^2/(1-0.5)^2,
                           rsq_prop_diff = (diff_prop-diff_prop_repl)^2/(1-0.5)^2)
                         ][, .(prop_U = mean(prop_U),  prop_R = mean(prop_R),
                           diff_prop = mean(diff_prop), stderr_prop1 = sqrt(mean(rsq_prop1)),
                           stderr_prop2 = sqrt(mean(rsq_prop2)),
                           stderr_prop_diff = sqrt(mean(rsq_prop_diff))), by = .(dvi)
                         ][, `:=`(diff_prop_lwr = diff_prop - 1.96*(stderr_prop_diff),
                           diff_prop_upr = diff_prop + 1.96*(stderr_prop_diff))
                         ][order(-diff_prop)]

# tabling
options(digits = 4)
knitr::kable(internet_weighted_ci)
```

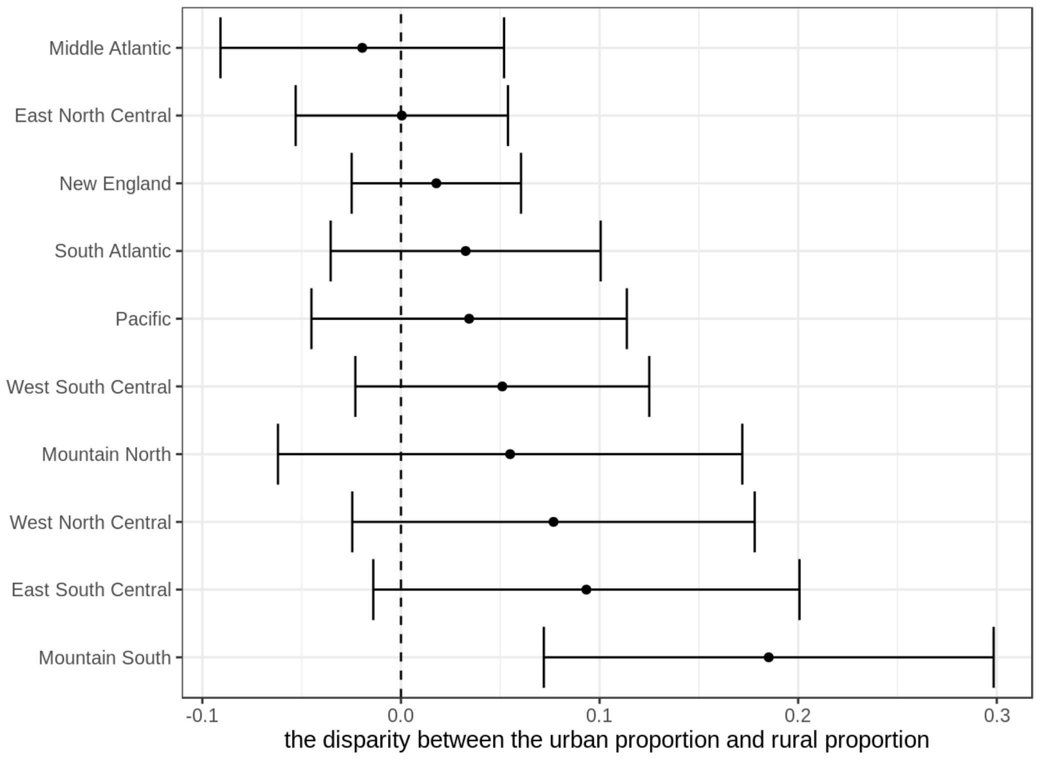
dvi	prop_U	prop_R	diff_prop	stderr_prop1	stderr_prop2	stderr_prop_diff	diff_prop_lwr	diff_prop_upr
Mountain South	0.8527	0.6675	0.1852	0.0201	0.0433	0.0578	0.0719	0.2984
East South Central	0.7836	0.6903	0.0933	0.0399	0.0282	0.0547	-0.0140	0.2006
West North Central	0.8800	0.8033	0.0768	0.0172	0.0451	0.0517	-0.0245	0.1781
Mountain North	0.8742	0.8193	0.0550	0.0277	0.0414	0.0596	-0.0619	0.1719
West South Central	0.8161	0.7650	0.0510	0.0265	0.0223	0.0377	-0.0230	0.1250
Pacific	0.8871	0.8528	0.0343	0.0129	0.0400	0.0405	-0.0451	0.1137
South Atlantic	0.8530	0.8204	0.0326	0.0136	0.0294	0.0347	-0.0354	0.1005
New England	0.8757	0.8579	0.0178	0.0259	0.0175	0.0218	-0.0248	0.0604
East North Central	0.8625	0.8621	0.0004	0.0127	0.0233	0.0273	-0.0530	0.0539
Middle Atlantic	0.8934	0.9129	-0.0195	0.0280	0.0305	0.0364	-0.0909	0.0519

```r
internet_weighted_ci[which( internet_weighted_ci[,"diff_prop"] == max(internet_weighted_ci[,"diff_prop"])
),1]
```

```
##                dvi
## 1: Mountain South
```

```r
# graphing
library(ggplot2)
graph_tmp = internet_weighted_ci[,`:=`(measure = factor(dvi, levels = unique(dvi) ))]

ggplot(graph_tmp, aes( y = measure, x = diff_prop) ) +
  geom_point() +
  geom_errorbar( aes(xmin = diff_prop_lwr, xmax = diff_prop_upr) ) +
  geom_vline( xintercept = 0, lty = 'dashed') +
  xlab('the disparity between the urban proportion and rural proportion') +
  ylab('') +
  theme_bw()
```



From the above table, prop_urb shows the proportion of home with internet in urban area, prop_rur shows the proportion of home with internet in rural area and diff_prop shows the disparity between the urban proportion and rural proportion. Each of them has been obtained 95% confidence inerval which is shown above. From the diff_prop columns, we can find the “Mountain South” census division has the largest disparity between urban and rural areas in terms of the proportion of homes with internet access.

Question 2

Part a

```
#!/bin/bash
#----------------------------------------------
# author: Sijun Zhang umid:89934761 randyz@umich.edu
# last change date: 12/1/2019
#----------------------------------------------

# Download the data and inspect the first 100 rows at the command line.
file="GSE138311_series_matrix.txt"
if [ ! -f "$file" ]; then
    wget ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE138nnn/GSE138311/matrix/GSE138311_series_matrix.txt.gz
    gunzip GSE138311_series_matrix.txt.gz
fi

head --line=100 $file

# Write the commands for doing so in your solution. How many lines of header information are there?
head --line=100 $file > GSE138311_series_matrix_head100.txt
grep -rn -o -E "\"ID_REF\"" GSE138311_series_matrix_head100.txt | tr ":" "\n" | awk 'NR%2 ==1'
```

The result shows that there are 69 lines used for header information that includes one line for colnames. In other words, if the colnames line is not treated as the header, then we have 68 lines of header.

Part b

```
library(data.table)
library(ggplot2)
gse = fread("GSE138311_series_matrix.txt", skip = 68)
```

```
## Warning in fread("GSE138311_series_matrix.txt", skip = 68): Discarded single-
## line footer: <<!series_matrix_table_end>>
```

```
gse_b = gse[grep("^ch", ID_REF),
        ][, -("GSM4105199"), with = FALSE]
gse_b = melt(gse_b, id.vars = c("ID_REF"))

head(gse_b)
```

```
##              ID_REF   variable   value
## 1: ch.1.101940785F GSM4105187 0.04272
## 2:   ch.1.1021960F GSM4105187 0.11388
## 3:   ch.1.1026209F GSM4105187 0.06742
## 4: ch.1.103396251R GSM4105187 0.10374
## 5:   ch.1.1047298R GSM4105187 0.04844
## 6: ch.1.107099706F GSM4105187 0.06505
```

We can see each row reperesent a sample-probe pair.

Part c

```
#c. label the diseased sample as sample_group 1 and the other as 0
label_disease = function(x) {
  if (x %in% c("GSM4105187", "GSM4105188", "GSM4105189",
               "GSM4105190", "GSM4105191", "GSM4105192",
               "GSM4105193")) { re = 1}
  else {re = 0}
  return(re)
}

label_disease_all = function(x) {
  return(sapply(x,label_disease))
}

gse_c = gse_b[, `:=`(sample_group = label_disease_all(variable))]
head(gse_c)
```

```
##              ID_REF   variable   value sample_group
## 1: ch.1.101940785F GSM4105187 0.04272            1
## 2:   ch.1.1021960F GSM4105187 0.11388            1
## 3:   ch.1.1026209F GSM4105187 0.06742            1
## 4: ch.1.103396251R GSM4105187 0.10374            1
## 5:   ch.1.1047298R GSM4105187 0.04844            1
## 6: ch.1.107099706F GSM4105187 0.06505            1
```

Label the diseased sample as sample_group 1 and the other as 0

Part d

```
calc_t = function(mu,n,s) {
  sp = sqrt( ((n[1]-1)*(s[1])^2 + (n[2]-1)*(s[2])^2 ) / (sum(n)-2) )
  se = sp * sqrt(1/n[1] + 1/n[2])
  t = (mu[1] - mu[2]) / se
  return(t)
}

gse_d = gse_c[, .(mu = mean(value), n = .N, s = sd(value)), by=.(ID_REF, sample_group)
        ][, .(t_score = calc_t(mu,n,s)), by=.(ID_REF) ]
head(gse_d)
```

```
##              ID_REF t_score
## 1: ch.1.101940785F -0.5317
## 2:   ch.1.1021960F  1.0065
## 3:   ch.1.1026209F  0.9559
## 4: ch.1.103396251R  2.2946
## 5:   ch.1.1047298R  0.9773
## 6: ch.1.107099706F  1.1423
```

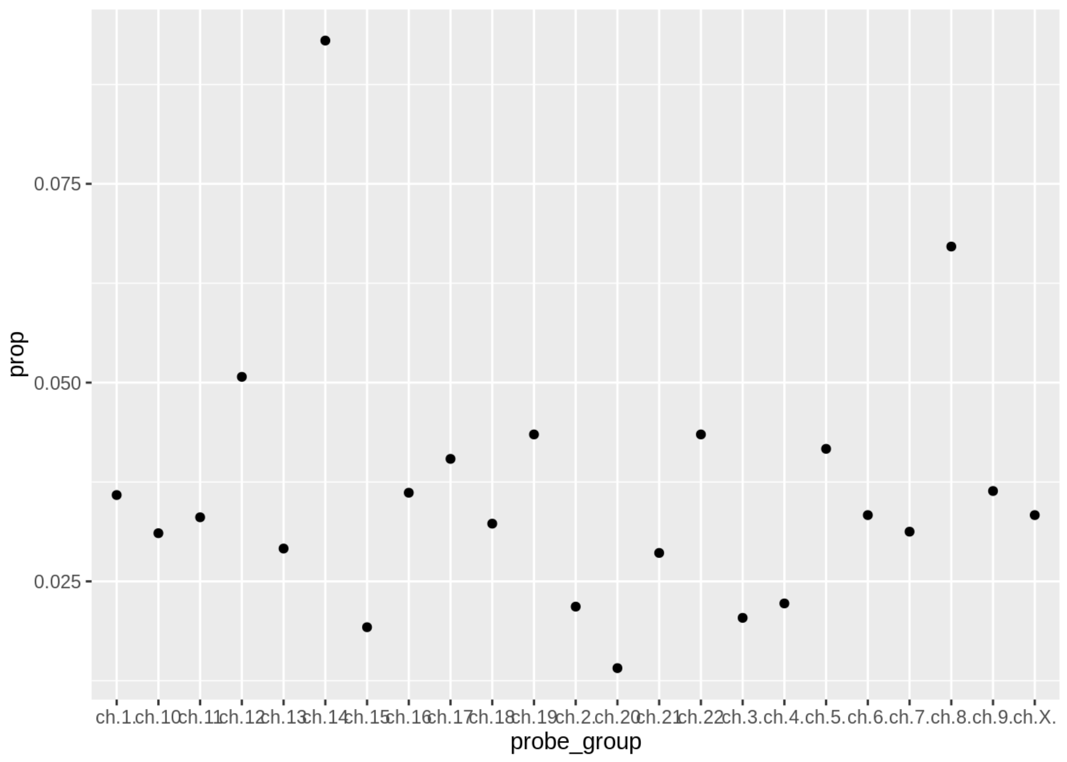
Part e

```
gse_e = gse_d[,`:=`(probe_group = substr(ID_REF,  1, 5))]
```

Part f

```
gse_f = gse_e[, `:=`(critical_abs_value = qt(0.975, df=10))
        ][, `:=`(index = (abs(t_score) > critical_abs_value))
        ][, .(prop = sum(index) / .N), by=.(probe_group)]

ggplot(gse_f, aes(x = probe_group, y = prop)) +
  geom_point()
```



From the graph, we can find the probe_group “ch.14”, “ch.12” and “ch.8” stands out as potentially over-represented as the proportions of them are greater than 0.05

Part g

```r
# declare the sample_id for reference
crohn_ind = append(rep(1,7), rep(0,5))
sample_id = c("GSM4105187", "GSM4105188", "GSM4105189",
              "GSM4105190", "GSM4105191", "GSM4105192","GSM4105193",
              "GSM4105194", "GSM4105195", "GSM4105196", "GSM4105197", "GSM4105198")

permutation_test = function(df, type = "two-tailed",
                            permuate = TRUE, alpha = 0.05){
  if (!permuate){
    # same process in d.
    df_est = df[, .(mu = mean(value), n = .N, s = sd(value)), by=.(ID_REF, sample_group)
               ][, .(t_score = calc_t(mu,n,s)), by=.(ID_REF)
                 ][,`:=`(probe_group = substr(ID_REF,  1, 5))]
  }
  if (permuate){
    # permutate the sample_group using merge
    sample_group = sample(crohn_ind, size = 12, replace = FALSE)
    sample_group_table = data.table(cbind(sample_id ,sample_group))
    df_est = df[, .(ID_REF, sample_id = variable,value)]
    # give the df permuated sample_group
    df_est = merge(df_est,sample_group_table, by = 'sample_id', all.x = TRUE)
    df_est = df_est[, .(mu = mean(value), n = .N, s = sd(value)), by=.(ID_REF, sample_group)
                   ][, .(t_score = calc_t(mu,n,s)), by=.(ID_REF)
                     ][,`:=`(probe_group = substr(ID_REF,  1, 5))]
  }
  if(type == "two-tailed"){
    df_est = df_est[, `:=`(index = (abs(t_score) > qt(1-alpha/2, df=10)))
                   ][, .(T_abs = mean(index*abs(t_score))), by=.(probe_group)]
  } else if (type == "greater"){
    df_est = df_est[, `:=`(index = ((t_score) > qt(1-alpha, df=10)))
                   ][, .(T_up = mean(index*(t_score))), by=.(probe_group)]
  } else {
    df_est = df_est[, `:=`(index = ((t_score) < qt(alpha, df=10)))
                   ][, .(T_down = mean(index*(t_score))), by=.(probe_group)]
  }
  return(df_est)
}
```

The basic idea in the permuating process is to shuffle (sample w/o replacement) either the 12 sample ids or the 12 Crohn’s/Not-Crohn’s labels and use merge to give the permuated group label to each sample.

Part h

In the following part, we apply the guide in resampling techniques and add one to both the numerator and denominator.

```
set.seed(5)
T_abs_original = permutation_test(gse_c, type = "two-tailed", permuate = FALSE)
p_value_index = T_abs_original[, .(probe_group, ind = 0)]
start_time=Sys.time()
for (i in 1:1000) {
  T_abs_permuated = permutation_test(gse_c, type = "two-tailed", permuate = TRUE)
  p_value_index_i = merge(T_abs_permuated, T_abs_original,
                          all.x = TRUE, by='probe_group')
  # calc whether the observed Tabs score for each group is larger than the expectation
  p_value_index_i = p_value_index_i[,.(probe_group, ind_i = T_abs.x >= T_abs.y)]
  # cumulating the larger or not result
  p_value_index = merge(p_value_index, p_value_index_i, all.x = TRUE, by='probe_group')[
    ,.(probe_group,ind = ind+ind_i)]
}
end_time=Sys.time()
for_loop_time = end_time-start_time
p_value_index = p_value_index[, .(probe_group, p_value = (ind+1)/1001)]
cat("The time taken to compute 1,000 permutation in for-loop is shown below \n")
```

```
## The time taken to compute 1,000 permutation in for-loop is shown below
```

```
for_loop_time
```

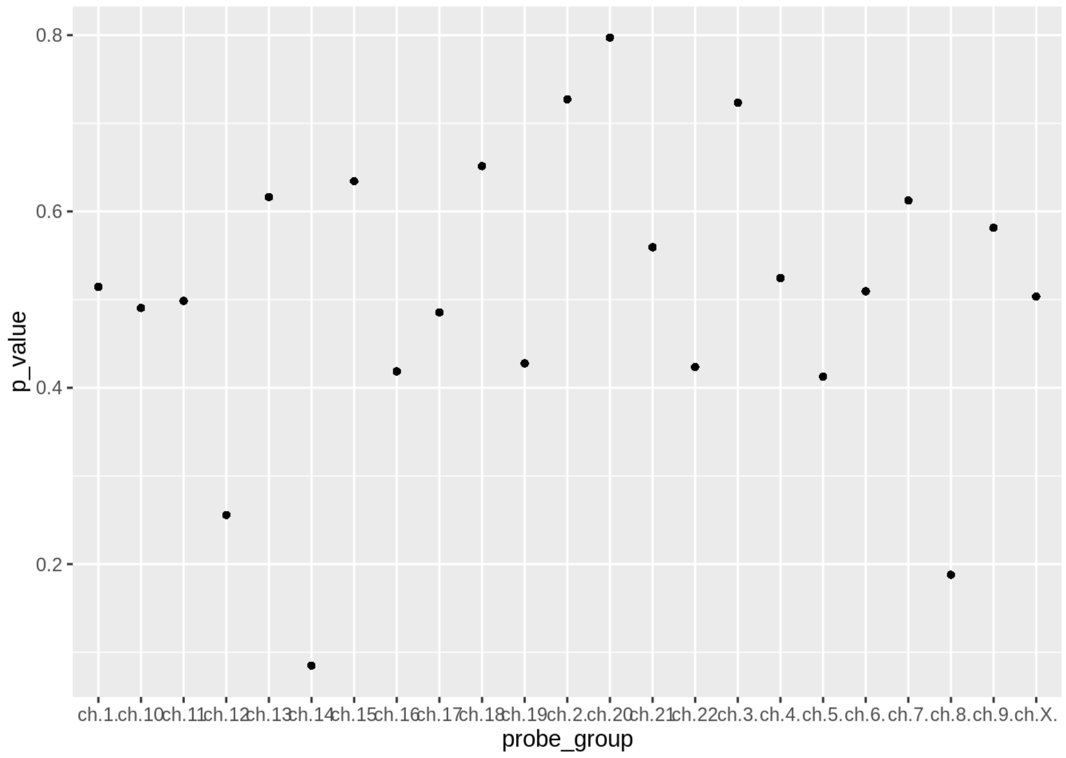
```
## Time difference of 22.23 secs
```

```
knitr::kable(p_value_index)
```

probe_group	p_value
ch.1.	0.5145
ch.10	0.4905
ch.11	0.4985
ch.12	0.2557
ch.13	0.6164
ch.14	0.0849
ch.15	0.6344
ch.16	0.4186
ch.17	0.4855
ch.18	0.6513
ch.19	0.4276
ch.2.	0.7273
ch.20	0.7972
ch.21	0.5594
ch.22	0.4236
ch.3.	0.7233
ch.4.	0.5245
ch.5.	0.4126
ch.6.	0.5095

probe_group	p_value
ch.7.	0.6124
ch.8.	0.1878
ch.9.	0.5814
ch.X.	0.5035

```
ggplot(p_value_index, aes(x = probe_group, y = p_value)) +
  geom_point(shape=16)
```



From the graph, we can find the ch.14 gene group has the p-value = 0.0849151, where we can approximately reject the null hypothesis at most significant level = 0.0849. However, at 0.05 level, we can not reject null hypothesis in any group.

Part i

```r
library(parallel)
set.seed(6925)
p_value_index = T_abs_original[, .(probe_group, ind = 0)]
T_up_original = permutation_test(gse_c, type = "greater", permuate = FALSE)
permutating_T_up = function(i, x) {
  T_up_permuated = permutation_test(x, type = "greater", permuate = TRUE)
  return(T_up_permuated)
}
# parallel computing the T_up
start_time=Sys.time()
T_up_permuated_all = mclapply(1:1000, permutating_T_up, x=gse_c)
end_time=Sys.time()
# calc the p_value
for (i in 1:1000) {
  T_up_permuated = T_up_permuated_all[[i]]
  p_value_index_i = merge(T_up_permuated, T_up_original,
                          all.x = TRUE, by='probe_group')
  # calc whether the observed Tup score for each group is larger than the expectation
  p_value_index_i = p_value_index_i[,.(probe_group, ind_i = T_up.x >= T_up.y)]
  # cumulating the larger or not result
  p_value_index = merge(p_value_index, p_value_index_i, all.x = TRUE, by='probe_group')[
    ,.(probe_group,ind = ind+ind_i)]
}
mclapply_time = end_time-start_time
p_value_index = p_value_index[, .(probe_group, p_value = (ind+1)/1001)]
cat("The time taken to compute 1,000 permutation in mclapply_time is shown below \n")
```

```
## The time taken to compute 1,000 permutation in mclapply_time is shown below
```

```r
mclapply_time
```

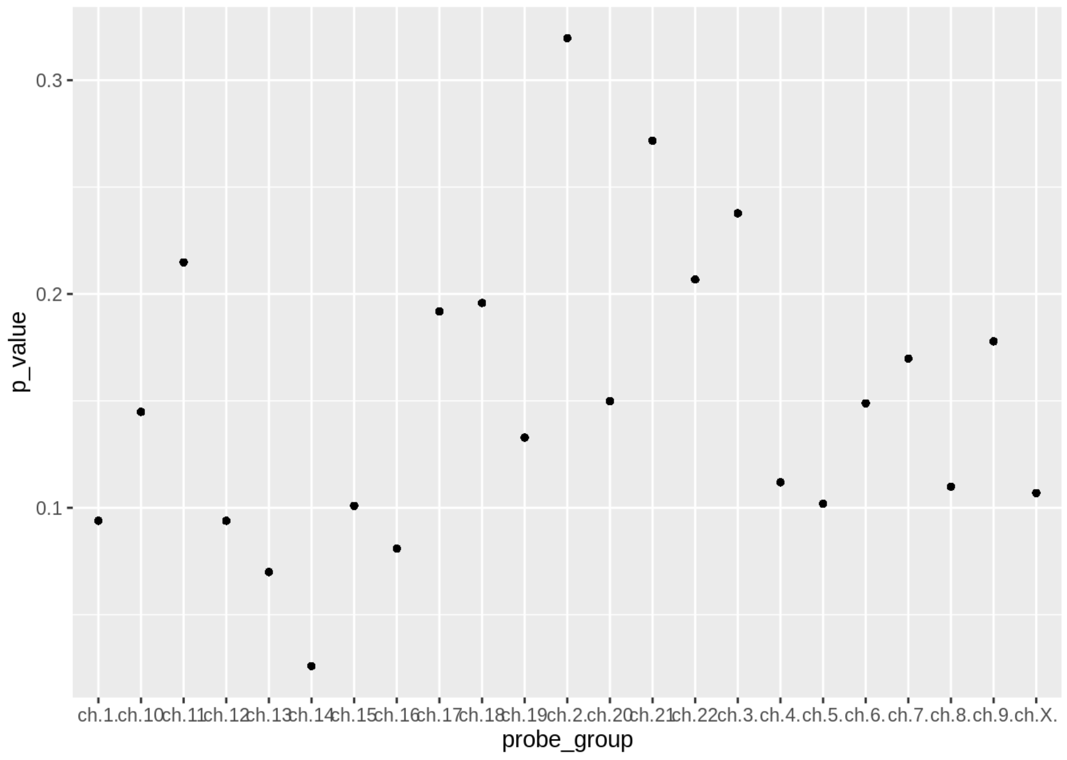
```
## Time difference of 8.169 secs
```

```r
knitr::kable(p_value_index)
```

probe_group	p_value
ch.1.	0.0939
ch.10	0.1449
ch.11	0.2148
ch.12	0.0939
ch.13	0.0699
ch.14	0.0260
ch.15	0.1009
ch.16	0.0809
ch.17	0.1918
ch.18	0.1958
ch.19	0.1329
ch.2.	0.3197
ch.20	0.1499
ch.21	0.2717
ch.22	0.2068
ch.3.	0.2378
ch.4.	0.1119
ch.5.	0.1019

probe_group	p_value
ch.6.	0.1489
ch.7.	0.1698
ch.8.	0.1099
ch.9.	0.1778
ch.X.	0.1069

```
ggplot(p_value_index, aes(x = probe_group, y = p_value)) +
  geom_point(shape=16)
```



Comparing to the for-loop method, the mclapply is approximately 3 times faster than the single-thread method. And in the p-value of the T_up, the ch.14. gene group has the smallest p-value = 0.03696304, where we can reject the null hypothsis.

Part j

```r
#j. split the task into 2 sub-task using future
library(future)
plan(multisession)
set.seed(6925)
p_value_index = T_abs_original[, .(probe_group, ind = 0)]
T_down_original = permutation_test(gse_c, type = "lesser", permuate = FALSE)
permutating_T_down = function(x) {
  T_down_permuated = permutation_test(x, type = "lesser", permuate = TRUE)
  return(T_down_permuated)
}
# parallel computing the T_down
start_time=Sys.time()
T_up_permuated_all_1 %<-% {
  c = list()
  for (i in 1:500) {c[[i]] = permutating_T_down(gse_c)}
  c
}
T_up_permuated_all_2 %<-% {
  c = list()
  for (i in 1:500) {c[[i]] = permutating_T_down(gse_c)}
  c
}
T_down_permuated_all = c(T_up_permuated_all_1, T_up_permuated_all_2)
end_time=Sys.time()
future_time = end_time - start_time
# calc the p_value
for (i in 1:1000) {
  T_down_permuated = T_down_permuated_all[[i]]
  p_value_index_i = merge(T_down_permuated, T_down_original,
                          all.x = TRUE, by='probe_group')
  # calc whether the observed Tup score for each group is larger than the expectation
  # according to the resampling method, as the value of T_down all smaller than or equal to 0
  # and the p_est compare the absolute value of statistics, we change the >= to <= here.
  p_value_index_i = p_value_index_i[,.(probe_group, ind_i = T_down.x <= T_down.y)]
  # cumulating the larger or not result
  p_value_index = merge(p_value_index, p_value_index_i, all.x = TRUE, by='probe_group')[
    ,.(probe_group,ind = ind+ind_i)]
}
p_value_index = p_value_index[, .(probe_group, p_value = (ind+1)/1001)]
cat("The time taken to compute 1,000 permutation in mclapply_time is shown below \n")
```

```
## The time taken to compute 1,000 permutation in mclapply_time is shown below
```

```r
future_time
```

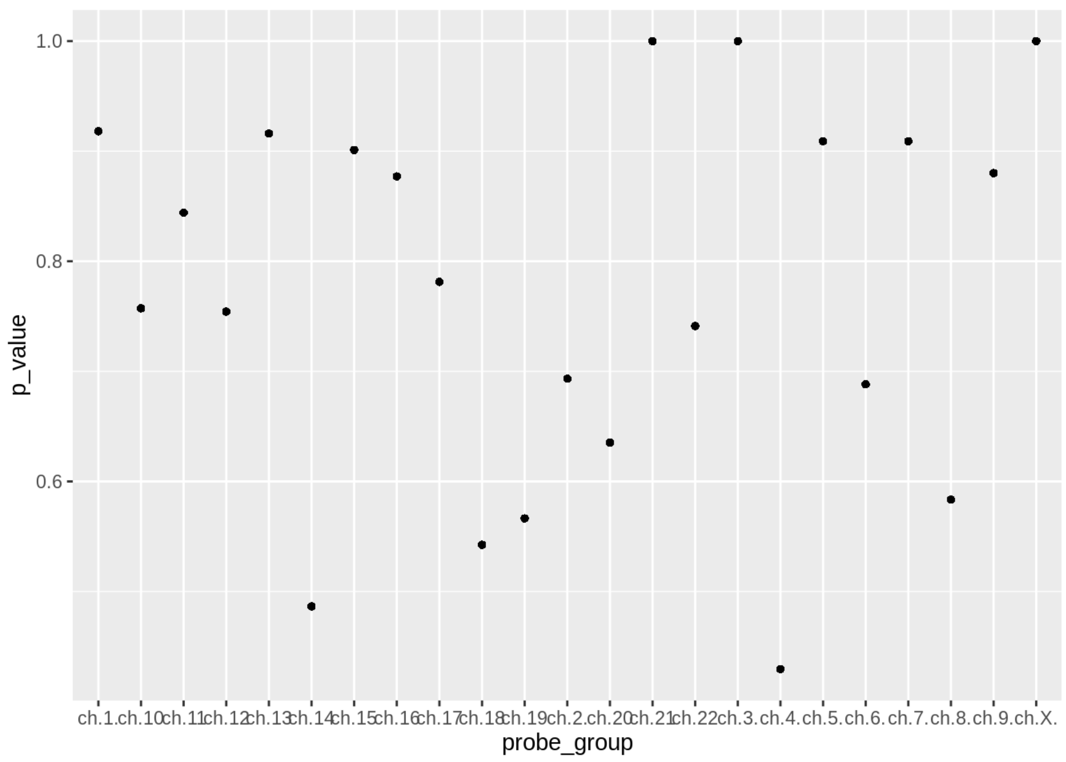
```
## Time difference of 12.08 secs
```

```r
knitr::kable(p_value_index)
```

probe_group	p_value
ch.1.	0.9181
ch.10	0.7572
ch.11	0.8442
ch.12	0.7542
ch.13	0.9161
ch.14	0.4865
ch.15	0.9011
ch.16	0.8771
ch.17	0.7812
ch.18	0.5425

probe_group	p_value
ch.19	0.5664
ch.2.	0.6933
ch.20	0.6354
ch.21	1.0000
ch.22	0.7413
ch.3.	1.0000
ch.4.	0.4296
ch.5.	0.9091
ch.6.	0.6883
ch.7.	0.9091
ch.8.	0.5834
ch.9.	0.8801
ch.X.	1.0000

```
ggplot(p_value_index, aes(x = probe_group, y = p_value)) +
  geom_point(shape=16)
```



In this part j. I split the task into 2 sub-task using future

Comparing to the for-loop method, the future is approximately 2 times faster than the single-thread method. And in the p-value of the T_down, the ch.4. gene group has the smallest p-value, but all of them are greater 0.05, so we can not reject any null hypothesis here.