PS5

Cali Li

11/27/2019

Question 1

```
# load packages: ------------------------------------------------------------
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# load data
recs = fread('C:/Users/wenji/Downloads/Stats 506/recs2015_public_v4.csv')

# compute proportion based on nweight
prop_nweight = recs[, uatyp := ifelse(UATYP10 == "R",
                                      "R_nweight", "U_nweight")] %>%
  .[, .(DOEID, uatyp, NWEIGHT, DIVISION, INTERNET)] %>%
  .[, nweight_internet := ifelse(INTERNET == 1, NWEIGHT, 0)] %>%
  .[, .(sum_nweight_internet = sum(nweight_internet),
        sum_nweight = sum(NWEIGHT)), by = .(uatyp, DIVISION)] %>%
  .[, prop := sum_nweight_internet / sum_nweight] %>%
  .[, .(uatyp, division = DIVISION, prop)] %>%
  melt(id = 1:2) %>%
  dcast(division ~ uatyp)

# compute proportion based on brrwt*
prop_brrwt = recs[,uatyp := ifelse(UATYP10 == "R", "R_brrwt", "U_brrwt")] %>%
  .[, .(DOEID, uatyp, DIVISION, INTERNET, .SD),
    .SDcols = paste("BRRWT", 1:96, sep = "")] %>%
  melt(id = 1:4) %>%
  .[, .(sum_brrwt_internet = sum(value*INTERNET),
        sum_brrwt = sum(value)), by = .(uatyp, DIVISION, variable)] %>%
  .[, prop := sum_brrwt_internet / sum_brrwt] %>%
  .[, .(uatyp, division = DIVISION, variable, prop)] %>%
  dcast(division + variable ~ uatyp) %>%
  .[prop_nweight, on = 'division'] %>%

  # compute se and CI based on brrwt*
  .[, se := sqrt(sum((abs(R_nweight - U_nweight) - abs(R_brrwt-U_brrwt))^2)
                 *4/96), by = division] %>%
  .[, `:=`(disparsity = round(abs(R_nweight - U_nweight), 3),
           lwr = round(abs(R_nweight - U_nweight) - se, 3),
           upr = round(abs(R_nweight - U_nweight) + se, 3))] %>%

  # output a table
  .[, .(Division = replace(unique(division), 1 : 10,
                           c('New England', 'Middle Atlantic',
                             'East North Central', 'West North Central',
                             'South Atlantic', 'East South Central',
                             'West South Central', 'Mountain North',
                             'Mountain South', 'Pacific')),
        Disparsity = unique(disparsity),
        Lwr = unique(lwr),
        Upr = unique(upr))] %>%
  .[order(-Disparsity)] %>%
  knitr::kable(align = 'c')
```

```
## Using 'prop' as value column. Use 'value.var' to override
```

```
prop_brrwt
```

Division	Disparsity	Lwr	Upr
Mountain South	0.185	0.127	0.243
East South Central	0.093	0.039	0.148
West North Central	0.077	0.025	0.128
Mountain North	0.055	-0.005	0.115
West South Central	0.051	0.013	0.089
Pacific	0.034	-0.004	0.072
South Atlantic	0.033	-0.002	0.067
Middle Atlantic	0.019	-0.013	0.052
New England	0.018	0.001	0.035
East North Central	0.000	-0.026	0.027

Question 2

```
# question 2.a
cat('gunzip -c GSE138311_series_matrix.txt.gz | head -n 100')
```

```
## gunzip -c GSE138311_series_matrix.txt.gz | head -n 100
```

```
cat('gunzip -c GSE138311_series_matrix.txt.gz | tail -n +69 > total_data.txt')
```

```
## gunzip -c GSE138311_series_matrix.txt.gz | tail -n +69 > total_data.txt
```

```
# load packages: -------------------------------------------------------------
library(dplyr)
library(tidyr)
library(data.table)

# question b
## load data into R
data = fread('C:/Users/wenji/Downloads/total_data.txt')
```

```
## Warning in fread("C:/Users/wenji/Downloads/total_data.txt"): Discarded
## single-line footer: <<!series_matrix_table_end>>
```

```
data = data[grep('^ch', ID_REF),
            which(unlist(lapply(data, function(x) !all(is.na(x))))),
            with = FALSE] %>%
  melt(id=1)

# question c
## From the information, we can get that GSM4105187-GSM4105193 are samples for
## Crohn's disease; while GSM4105194-GSM4105198 are samples for no Crohn's
## disease.
data = data[, `:=`(sample_group = ifelse(variable %in% unique(variable)[1:7],
                                         "Crohn", "noCrohn"))]

# question d
## cmpute t-stats
t_stats = dcast(data, ID_REF+variable~sample_group) %>%
  .[, .(mean = mean(Crohn, na.rm = TRUE) - mean(noCrohn, na.rm = TRUE),
           std_Crohn = sd(Crohn, na.rm = TRUE),
           std_noCrohn = sd(noCrohn, na.rm = TRUE)),
    by = .(ID_REF)] %>%
  .[, .(ID_REF,
        t_stats = round(mean / (sqrt((6 * (std_Crohn) ^ 2 +
                                     4 * (std_noCrohn) ^ 2) / 10) *
                               sqrt(1/7 + 1/5)),3))]

# question e
data_e = data[, `:=`(probe_group = substr(ID_REF, 0, 5))]

# question f
## compute proportion based on significance
data_f = data_e[t_stats, on = 'ID_REF'] %>%
  .[, p := 2 * pt(t_stats, df = 10)] %>%
  .[, `:=`(sig = 1L * (p < 0.05))] %>%
  .[, .(prop = sum(sig) / .N), by = probe_group]

# question g
t_stat = function(data, type, permute=TRUE) {
```

```r
  if (permute) {
    data = data[, sample_group1 := sample_group[sample(1:.N,
                                                       replace = FALSE)]]
  }
  ## Compute t-stats
  data_1 = data[sample_group1 == 'Crohn', .(ccount = .N, cmean = mean(value),
                                           x1 = sum(value^2) -
                                             .N * ((mean(value))^2)),
                by = ID_REF]
  data_2 = data[sample_group1 == 'noCrohn', .(ncount = .N,
                                              nmean = mean(value),
                                              x2 = sum(value^2) -
                                                .N * ((mean(value))^2)),
                by = ID_REF]
  data_t = data_1[data_2, on = 'ID_REF'] %>%
    .[, .(t_stats = (cmean - nmean) / sqrt((x1 + x2)/(ccount + ncount - 2))
          / sqrt(1 / ccount + 1 / ncount)), by = ID_REF] %>%
    .[, `:=`(probe_group = substr(ID_REF, 0, 5))]

  ## compute three different computation methods
  if (type == "two_tailed") {
    t = data_t[, p := 2 * pt(t_stats, df = 10)] %>%
      .[, `:=`(sig = 1L * (p < 0.05))] %>%
      .[, .(prop = sum(sig) / .N), by = probe_group]
  }
  if (type == "greater") {
    t = data_t[, p := qt(1-0.05, df = 10)] %>%
      .[, `:=`(sig = 1L * (t_stats > p))] %>%
      .[, .(prop = sum(sig) / .N), by = probe_group]
  }
  if (type == "lesser") {
    t = data_t[, p := qt(0.05, df = 10)] %>%
      .[, `:=`(sig = 1L * (t_stats < p))] %>%
      .[, .(prop = sum(sig) / .N), by = probe_group]
  }
  return(t)
}

# question h
## compute on the original data
t_stat(data, type = "two_tailed")
```

```
##     probe_group        prop
##  1:       ch.1. 0.011952191
##  2:       ch.10 0.043478261
##  3:       ch.11 0.008333333
##  4:       ch.12 0.021897810
##  5:       ch.13 0.029126214
##  6:       ch.14 0.023255814
##  7:       ch.15 0.028846154
##  8:       ch.16 0.012048193
##  9:       ch.17 0.030303030
## 10:       ch.18 0.000000000
## 11:       ch.19 0.057971014
## 12:       ch.2. 0.029090909
## 13:       ch.20 0.028571429
## 14:       ch.21 0.028571429
## 15:       ch.22 0.000000000
## 16:       ch.3. 0.020408163
## 17:       ch.4. 0.050000000
## 18:       ch.5. 0.025000000
## 19:       ch.6. 0.013422819
## 20:       ch.7. 0.050000000
## 21:       ch.8. 0.033557047
## 22:       ch.9. 0.009090909
## 23:       ch.X. 0.011111111
##     probe_group        prop
```

```
## define a new function stat_h
stat_h = function(n) {
  t = t_stat(data, type = "two_tailed")
  return(t)
}
p = function(n) {
  mean(n$prop) / (sd(n$prop)/sqrt(22))
}

## compute time over 1000 permutation
system.time({
  test_h = lapply(1:1000, stat_h)
  p_h = lapply(test_h, p)
})
```

```
##    user  system elapsed
##  120.28   47.47  232.50
```

```
# question j
## mclapply
## define a new function stat_j using greater method.
stat_j = function(n) {
  t = t_stat(data, type = "greater")
  return(t)
}

## load package and compute time over 1000 permutation
library(parallel)
system.time({
  test_j = mclapply(1:1000, stat_j)
  p_j = mclapply(test_j, p)
})
```

```
##    user  system elapsed
##  123.39   51.64  245.05
```

```
# question i
## future
## define a new function stat_i using lesser method.
stat_i = function(n) {
  t = t_stat(data, type = "lesser")
  return(t)
}

## load package and compute time over 1000 permutation
library(future)
plan(multisession)
system.time({
  test_h = lapply(1:1000, stat_i)
  p_j = lapply(test_j, p)
})
```

```
##    user  system elapsed
##  146.12   53.46  261.16
```