Appendix 1: Reading Data into R

1. Main Function

```
source('readFun.r')

#main codes to load data from all files
dirs = list.files('data',full.names = T)
dat = replicate(length(dirs), list())
names(dat) = dirs
for(i in 1:length(dirs)){
  dat[[i]] = readFile(dirs[i])
}


#merge all data together into one data frame
mileDat = Reduce(function(x,y) merge(x,y,all = T), dat)

#change the format of some variables
mileDat$sex = as.factor(mileDat$sex)
mileDat$year = as.factor(mileDat$year)
mileDat$name[mileDat$name == ''] = NA
mileDat$hometown[mileDat$hometown == ''] = NA
mileDat$s[mileDat$s == ''] = NA
mileDat$timeoguide = as.factor(mileDat$timeoguide)

mileDat$gender <- factor(mileDat$gender, levels=c("0", "1"), labels=c("Female", "Male"))
mileDat$name = tolower(mileDat$name)
mileDat$hometown = tolower(mileDat$hometown)

save(mileDat, file = 'data.rda')
```

This main function contains two parts. In the first part, read the data in each file and construct a data frame to store the data.**dat** is a list, in which each element is a data frame which contains information in each file. For example, **dat**[[**1**]] contains data in data/men10Mile_1999.

Since different data frames may have different colnames(feature), in part 2, we merge all data together to get one big dataframe(in mileDat).

2. Read each file(R Function **readFile**)

This function can be used to read information in each file. The input is the file name and the ouput is the data frame of this file.

```
readFile = function(filename){

  con = file(filename, open = "rt")
  dataFM = readLines(con, encoding = 'UTF-8')
  close(con)
```

```
  dataFM = preProcess(dataFM,filename)
  res = breakLines(dataFM)

  #add two columns records gender and year information from the file name
  subname = unlist(strsplit(filename, '/'))[2]
  if(substr(subname,1,1) == 'm'){
    res$gender = rep(1, dim(res)[1])
  }else{
    res$gender = rep(0, dim(res)[1])
  }
  res$year = as.numeric(substr(subname, nchar(subname)-3, nchar(subname)))

  return(res)
}
```

There are two subfunctions. They are **preProcess** and **breakLines**. **preProcess** can be used to make the header of each file in the same format. **breakLines** can be used to load the data in data frame.

2.1 R function **preProcess**

The input is the file name and s in the result of readLines. The output(s) is also a list but there in no useless information of header and it has the same format. s[1] is the column names. s[2] contains = and blanks to separate different cloumns. Other lines contain data.

```
#Preproess the data to make them in the same format(structured format)
preProcess = function(s,filename){

  #get the line with ======== which can be used to separate different columns
  offset = 1
  while(s[offset] =="" | substr(s[offset],1,1) != "="){
    offset = offset + 1
    if(offset == length(s)){
      break
    }
  }

  #is there is no offset line, use the data in the same year to add header information
  if(offset == length(s)){
    offset = 1
    while(!grepl('^[0-9]+',deleteBlank2(s[offset]))){
      offset = offset + 1
    }
    offset = offset - 1

    nm = unlist(strsplit(filename,'/'))[2]
    if(grepl('women',nm)){
      newfile = substr(nm, 3, nchar(filename))
    }else{
      newfile = paste('wo',nm,sep = '')
    }
    newfile = paste(unlist(strsplit(filename,'/'))[1],newfile, sep = '/')
    subcon = file(newfile, open = "rt")
```

```
  subdata = readLines(subcon,n = 50, encoding = 'UTF-8')
  close(subcon)

  sub_offset = 1
  while(subdata[sub_offset] =="" | substr(subdata[sub_offset],1,1) != "="){
    sub_offset = sub_offset + 1
  }
  add.data = subdata[(sub_offset-1):sub_offset]

  if(offset >=2)
  {
    s[(offset-1):offset] = add.data
  }else{
    s1 = add.data
    s1[3:(length(s) + 2 - offset)] = s[(offset+1):length(s)]
    s = s1
    offset = 2
  }
}

#find the position of '/', by which separate div and total
tag = unlist(gregexpr('/', s[offset + 1]))
if(tag[1] >0){
  substr(s[offset], tag, tag) = ' '
  substr(s[offset - 1], tag, tag) = ' '
}


#if there are two continuous blanks, change the first blank to =
tag = unlist(gregexpr('  ', s[offset]))
if (tag[1] > 0){
  for (i in 1:length(tag)){
    substr(s[offset], tag[i], tag[i]) = '='
  }
}

#combine digits and mi together in the head
#for example, 5 mi should be 5-mi
tmp = gregexpr('[0-9]+[ ]',s[offset - 1])
beg = unlist(tmp)
if(beg[1] > 0){
  len = attr(tmp[[1]],'match.length')
  pos = beg + len - 1
  for (k in 1:length(pos)){
    substr(s[offset - 1], pos[k], pos[k]) = '-'
  }
}


#if two columns are not divided by blanks in the offset line
#divide them into different columns
first = unlist(gregexpr('[ ][A-Z]+', s[offset - 1]))
second = unlist(gregexpr(' ', s[offset]))
```

```
  third = unlist(gregexpr(' ', s[offset + 1]))
  tmp = intersect(setdiff(first, second), third)
  if(length(tmp)> 0){
    for (i in 1:length(tmp)){
      substr(s[offset], tmp[i], tmp[i]) = ' '
    }
  }

  #if there are two blanks, change the first blank to =
  tag = unlist(gregexpr('  ', s[offset]))
  if (tag[1] > 0){
    for (i in 1:length(tag)){
      substr(s[offset], tag[i], tag[i]) = '='
    }

  }

  #delete the useless header information in the data
  if(offset > 2){
    s = s[-(1:(offset-2))]
  }

  #delete the unsless tail information in the data
  n = length(s)
  z = '1'
  if(nchar(s[n]) > 0){
    z = unlist(strsplit(s[n], ' '))
    z = z[z != '']
    z = z[1]
  }
  while(nchar(s[n]) == 0 | !grepl('^[0-9]+$', z)){
    n = n - 1
    if(nchar(s[n]) > 0){
      z = unlist(strsplit(s[n], ' '))
      z = z[z != '']
      z = deleteBlank2(z[1])
    }
  }
  s = s[1:n]

  #delete the blank lines
  tmp = which(deleteBlank(s) != '')
  s = s[tmp]
  return(s)
}
```

2.2 R function **breakLines**

This function can be used to put data in the dataframe and change their format. For example, change the time format to seconds.

The input is the result of function **preProcess**, the output(res) is a data frame.

```
#parse data and put them into a data frame
breakLines = function(s){

  #fieldsList can be used to separate different columns
  #fields records the begin and end of each column
  fieldsList = unlist(gregexpr(' ', s[2]))
  if(fieldsList[length(fieldsList)] < nchar(s[2])){
    fieldsList[length(fieldsList) + 1] = nchar(s[2])
  }
  fieldsList[length(fieldsList) + 1] = 0
  fieldsList = sort(fieldsList)


  #colnames for the data frame
  colnm = sapply(1:(length(fieldsList) - 1), function(i) {
    z = substr(s[1], fieldsList[i]+1, fieldsList[i+1])
    tolower(deleteBlank(z))
  })
  #use the same name if they record the same information
  colnm = sapply(colnm, function(i) {uniName(i)})

  #put data into the data frame
  m = sapply(3:length(s), function(j) {
    sapply(1:(length(fieldsList) - 1), function(i) {
      if (i<(length(fieldsList) - 1)){
        z = substr(s[j], fieldsList[i]+1, fieldsList[i+1])
      }else{
        z = substr(s[j], fieldsList[i]+1, nchar(s[j]))
      }

      deleteBlank2(z)
    })
  })
  res = as.data.frame(t(m))
  colnames(res) = colnm

  #add one column
  #if it is T, means that Under USATF OPEN guideline
  #F means Under USATF Age-Group guideline
  res$timeoguide = rep(NA, dim(res)[1])

  #transfer contents into differnent suitable format
  for (i in 1:(dim(res)[2] - 1)){
    tmp = paste(res[1:10,i], collapse = '')

    if (grepl('[A-Z]+|[a-z]+',tmp)){
      #transfer name and hometown data to character
      res[,i] = as.character(res[,i])
    }else if(grepl(':',tmp)){
      #transfer time to seconds and records the guideline information
      for (j in 1:dim(res)[1]){
        if(res[j,i] == ''){
          tmp[j] = NA
```

```
        }else{
          char = deleteBlank(res[j,i])
          if(grepl('#$',char)){
            res$timeoguide[j] = TRUE
            char = gsub('#','',char)
          }else if(grepl('\\*$',char)){
            res$timeoguide[j] = FALSE
            char = gsub('\\*','',char)
          }
          tmp[j] = toSeconds(char)
        }
      }
      res[,i] = tmp
      res[,i] = as.numeric(res[,i])

    }else if(grepl('[0-9]+',tmp)){

      res[,i] = as.numeric(as.character(res[,i]))
    }else{
      res[,i] = as.character(res[,i])
    }
  }

  #find whether there are several columns which have the same name
  tmp = which(colnm == 'pace')
  if(length(tmp) > 1)
  {
    res = subset(res, select = -tmp[1:(length(tmp) - 1)])
  }

  return(res)
}
```

3. Some Small Functions

3.1 Delete blanks and \

```
#delete all blanks
deleteBlank = function(char){
  return(gsub("\\s+", "", char))
}

#delete the leading and trailing blanks and / between div and total
deleteBlank2 = function(char){
  char = gsub("^\\s+|\\s+$", "", char)
  gsub('^/|/$','',char)
}
```

3.2 Use the same name for the column contains same information

In the data file, same columns may use different names. For example, 'GUN', 'GUN TIM','TIME' all stand for gun time. We must use the same name for further analysis. So, after change all names to lower case, use the following function to change them in the same name.

```
#Use the same name for the same column
uniName = function(nm){
  switch(nm,
         gun={
           nm = 'guntim'
         },
         time={
           nm = 'guntim'
         },
         net ={
           nm = 'nettim'
         },
         '5-mile'={
           nm = '5-mi'
         },
         'split' ={
           nm = '5-mi'
         },
         'ag' = {
           nm = 'age'
         },
         'div' = {
           nm = 'division'
         },
         'tot' = {
           nm = 'total'
         },
{
  nm = nm
}
  )
return(nm)
}
```

3.3 Change time format to seconds

The time format may has one or two :, for example, 1:05:06 and 45:06, we should change them to the same format(integer stands for seconds) which is each to compute.

```
#change the time to seconds
toSeconds = function(char){
  tmp = strsplit(char, ':')
  x = sapply(tmp, function(i){
    tmp2 = unlist(i)
    if(length(tmp2) == 2){
      as.numeric(tmp2[1])*60 + as.numeric(tmp2[2])
    }else if(length(tmp2) == 3){
```

```
      as.numeric(tmp2[1])*3600 + as.numeric(tmp2[2])*60 +  as.numeric(tmp2[3])
    }else{
      NA
    }
  })
  return(x)
}
```