STA 242 Assignment 1

Race Times - The Cherry Blossom Run

Xueting Shao

SID: 912457262

1. Data Description
   1.1. Data Profile

The data is collected separately in women and men from 1999 to 2010. 113288 observations are recorded. In each year, the data contain different variables. After cleaning the data, the possibly meaningful variables are all shown below:

```
10km    5mile    age         div       Guntime   hometown
name    Nettime  num         pace      pace.1    pace.2
place   s        signal[*]   signal#   split     tot
```

Variable `10km, 5mile, Guntime, Nettime`, and `split` are of time format and they are all transferred to the amount of seconds after data cleaning. Variable `div` and `tot` records the place and total runners. Variable `age, div, num, pace, pace.1, pace.2, place, tot` are numerical. Variable `hometown` and `name` are in string form. Variable `signal[*]` and `signal#` are logical, which indicates whether there is a ! or * symbol in some of the variable.

1.2. Overall Runner Profile

From Table 1, the number of participants is increasing from 1999 to 2010 and the total number of participants in 2010 is about three times of that in 1999. In addition, female participants increase more drastically than male participants. This may suggest The Cherry Blossom Run is getting popular, especially with women.

Table 1 Number of Runners in 1999-2010

Year	Men	Women	Total
1999	3190	2358	5548
2000	3017	2167	5184
2001	3623	2973	6596
2002	3724	3335	7059
2003	3948	3544	7492
2004	4156	3899	8055
2005	4327	4336	8663
2006	5237	5437	10674

```
2007  5276  5692     10968
2008  5908  6399     12307
2009  6651  8325     14976
2010  6911  8855     15766
```

From Figure 1, the median of gun time is used to describe the overall runners' performance in that year. The gun time increases slowly until 2008 and it seems runners' performance regress a lot since 2008. At the same time, the increasing patterns of male and female runners are similar. This may suggest both men and women suffer from the regression in running, which may be induced by fast and unhealthy lifestyle in recent years.

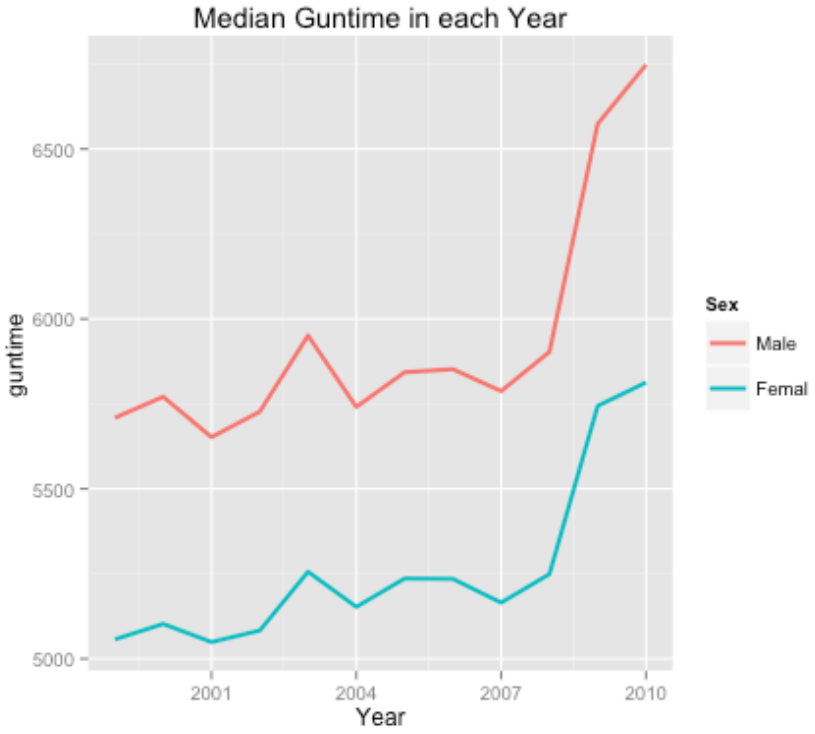

Figure 1 Median Gun Time in each Year

In Figure 2, ages of participants do not change much over years. The range of age is quite wide: maximum is around 80 and minimum is around 12.

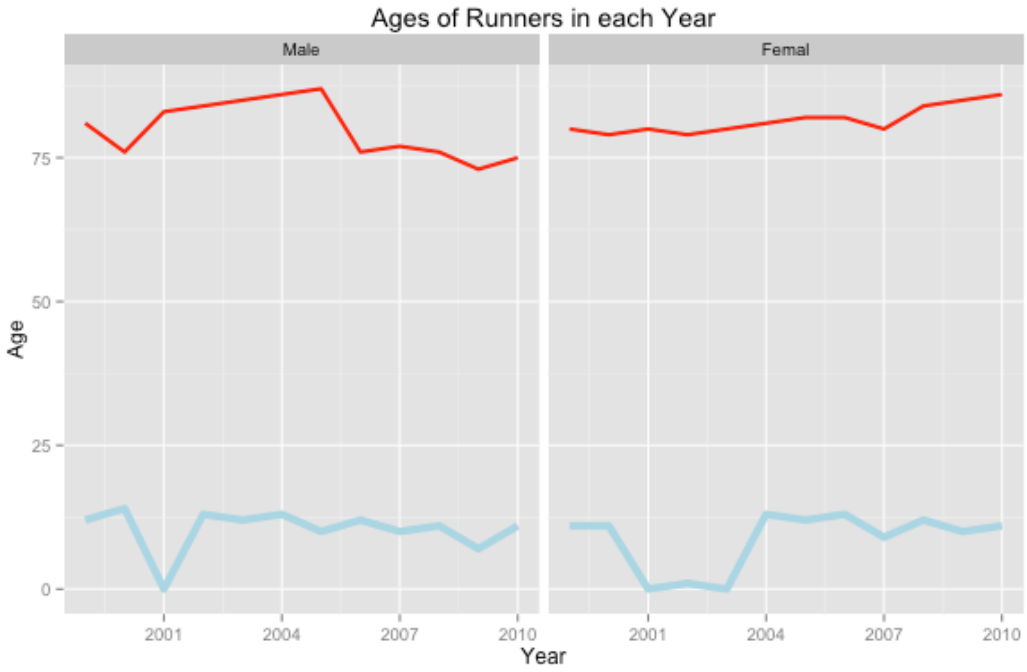

Figure 2 Age of Runners in each Year

2. Interest Aspect: Hometown Diversity

As a person who runs as slow as a turtle, I always tend to blame my slowness for my hometown when my friends joke on me. Chinese don't like running although my Chinese friends are definitely outliers. However, I really want to use this data to prove my excuse is right. People from some places are not good at running!

2.1. Runners from United States

Most of the participants are from Virginia, Maryland, District of Columbia, Pennsylvania, and New York, which are shown in Figure 3.

Furthermore, Virginia, Maryland, and District of Columbia have much more participants than other states. It may be related to the close distance between these states to Washington D.C, where the run is held.

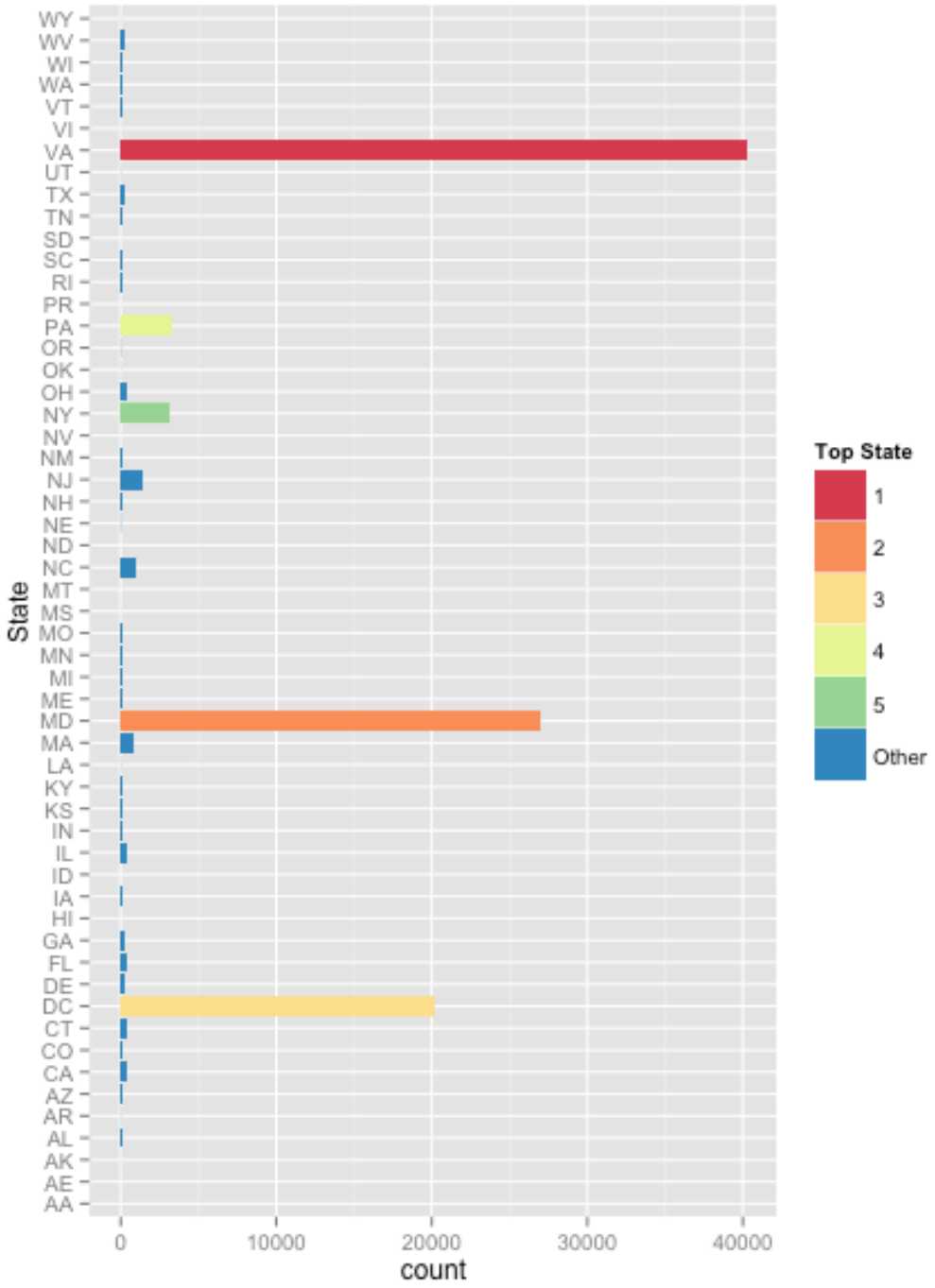

Figure 3 Amounts of Participants in States over Years

The amounts of participants from those states are high, and the rates of runners who rank the first ten are not low compared with other states, as it is shown in Table 2.

New York has a lot of participants, but none of those participants rank top ten, which is not surprising.

Table 2 Rate of Top Runners

DC	MD	NY	PA	VA
0.00005	0.00004	0.00000	0.00060	0.00012

From Figure 4, Virginia has the most top-ten runners. However, none of the winner comes from America. The highest rank is three, which is achieved by a runner from California.

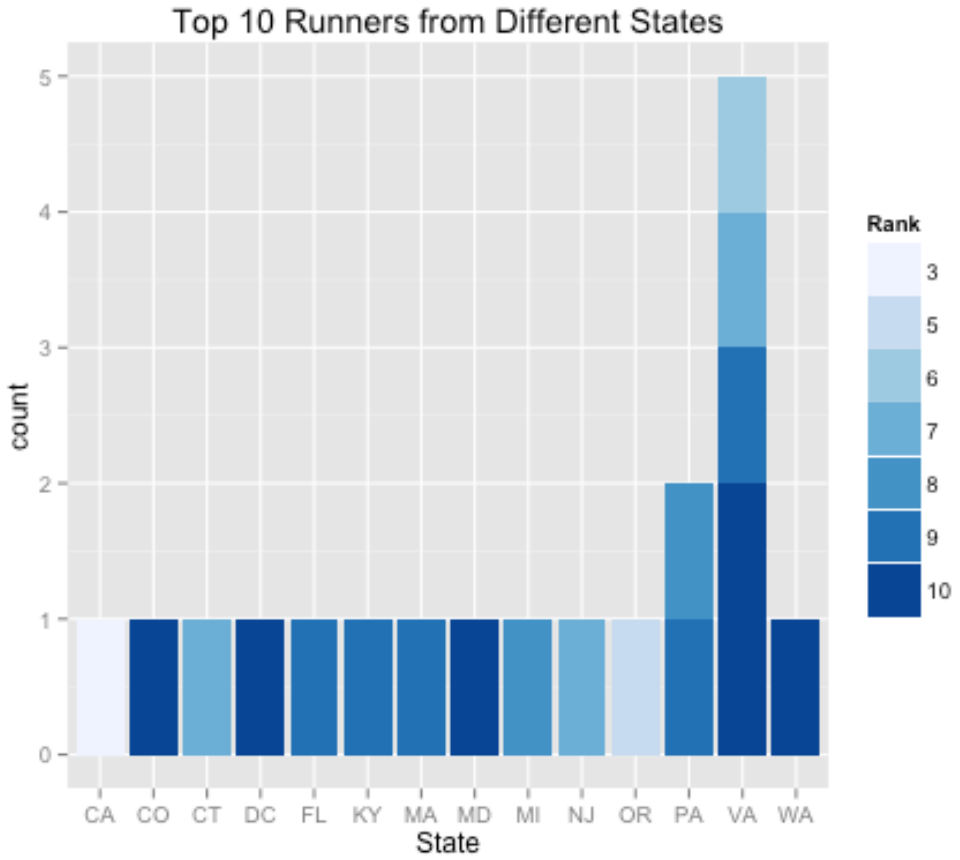

Figure 4 Top 10 Runners from Different States

2.2. Runners from Other Countries in the World

Besides runners from America, there are some runners from other countries, especially Kenya. Canadian, Ethiopian and Russian also show strong interests in the run.

Table 3 Runners from Different Countries

Country	Total	Country	Total
Australia	3	Kenya	157
Austria	1	Koengen	1
Brazil	1	Mexico	7
Canada	48	Morocco	5
Colombia	3	New Zealand	2
Denmark	1	Romania	10

Ethiopia	45	Russia	26
France	4	Switzerland	2
Germany	7	Tanzania	2
Holland	1	Ukraine	2
Japan	13	United Kingdom	7
Jordan	1	Uruguay	1

Among those countries, runners from Kenya have much better performance than those from any other countries. In Figure 5, it is obvious that most winners are Kenyan for these twelve years. And the rate of top 10 runners is 0.64! It is consistent with the fact that Kenya is famous for the world's best runners.

Ethiopian and Russian runners also have quite good performance, but Romania has a better number and rate of winners. This also agrees with the common sense that people from these countries tend to be considered as strong and fast.

In general, most foreign runners tend to have a better performance than Americans. It should be resulted from large number of slow participants since the Cherry Blossom Run is held in America. It is more possible that some Americans who are not good at running will be interested in the Cherry Blossom Run.

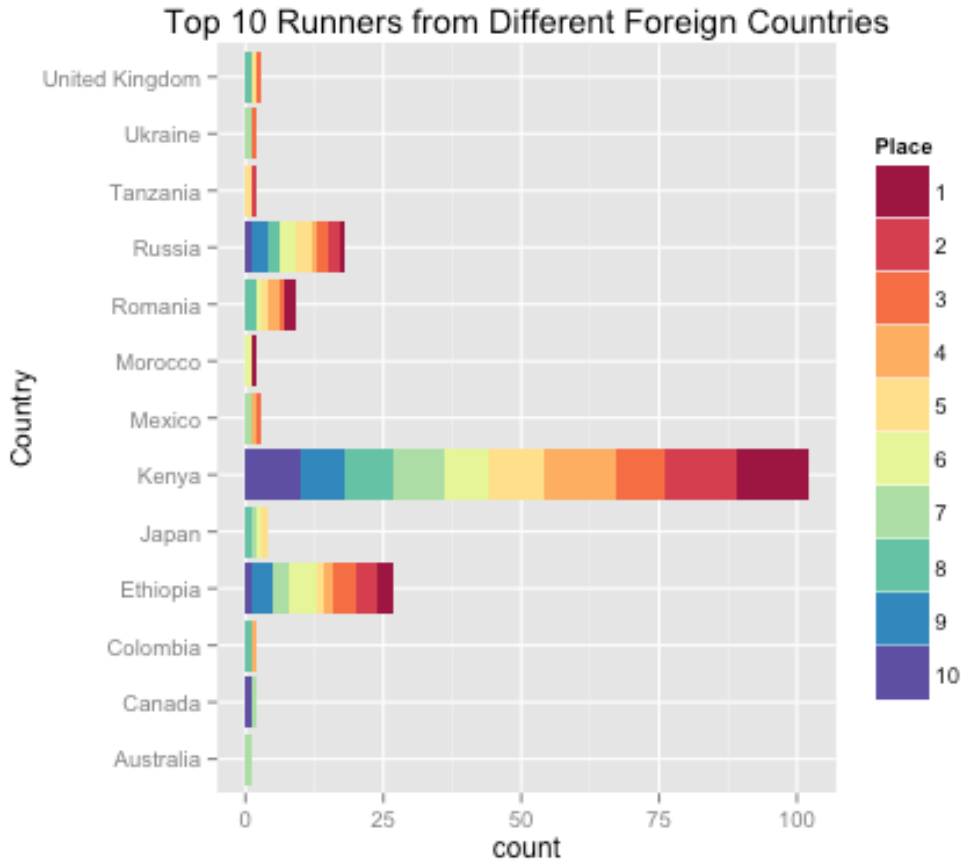

Figure 5 Top 10 Runners from Different Countries

2.3. Runners from Kenya

Since most winners are from Kenya, it is meaningful to take a look at Kenyan runners overs years.

In Figure 6, number of Kenyan participants fluctuates over years but there seems no obvious pattern. In 2006, the numbers are extremely low for both female and male runners. It may be related to the serious drought in Kenya at that year, but it needs further research.

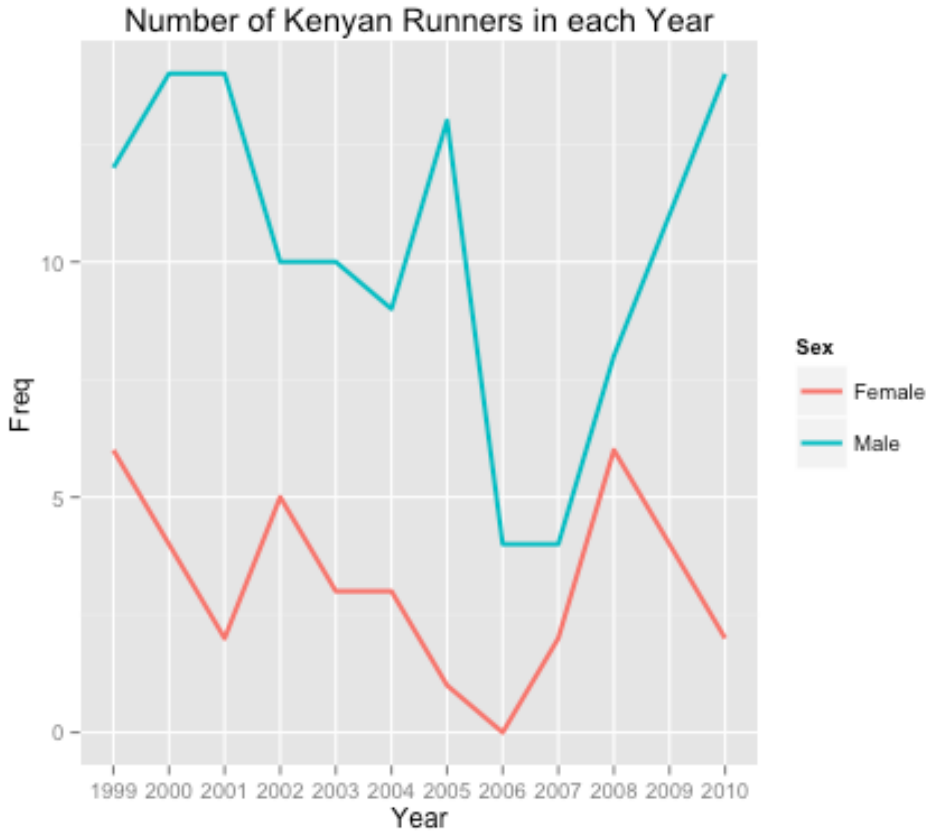

Figure 6 Numbers of Kenyan Runners over Years

Figure 7 shows neither increasing nor decreasing pattern of gun time. It suggests the performance of Kenyan Runners is stable over years. They are good from 1999 to 2010!

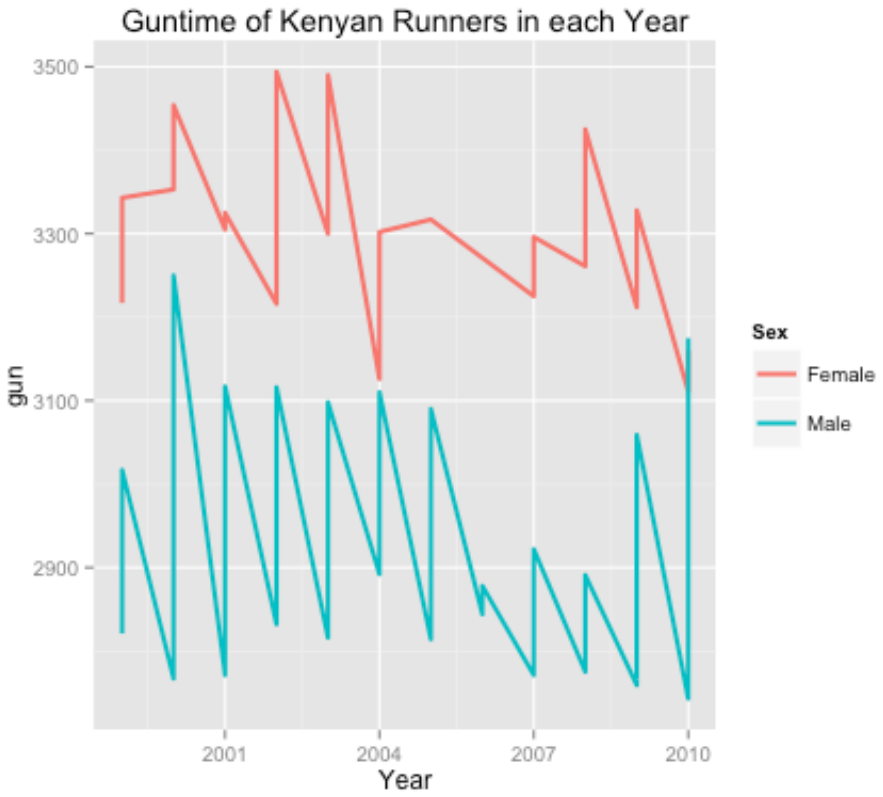

Figure 7 Gun time of Kenyan Runners over Years

3. Conclusion

From analysis on runners' hometown, it can be concluded that participant number and performance have certain relationship with hometown. However, it is hard to say whether there is a cause-result relationship.

In general, Kenyan, Ethiopian and Russian runners have better performance than others from 1999 to 2010. But this doesn't mean American run slower than them. Since the Cherry Blossom Run is held in America, participants from other countries may be professional runners or very interested in running. People living near Washington. D. C. may participate not because of running but in terms of having fun. It can be proved by further analysis of variable `age, s, signal[*],signal#`.

Inside America, Virginia, Maryland, District of Columbia, Pennsylvania, and New York have a lot of participants. Part of the reason may be the close distance from those states to Washington. Runners from Virginia have the best overall performance, but none of New Yorkers run into the top 10. It is consistent with common sense that people in New York do have less exercise.

Appendix 1

```
#Read fILE

setwd('/Users/Shawn/Dropbox/242/proj/Assignment1/data')
flist = list.files()

#define a function to read file and close the connection, in case of warnings
re = function( f){
  #f is the file name
  con = file(f, open = 'r')
  txt = readLines(con)
  close(con)
  return(txt)
}

#put all 24 files txt in alltxt, so alltxt is a large list which contain 24 elements
alltxt = sapply( 1:length(flist), function(i) re(flist[i]))
#relate the elements(txt) with orignal file names
names(alltxt) = flist

#take a look at how "===" locates
for (i in 1:length(flist)){print(alltxt[[i]][9])}
#not all 8th line is "==="

#find which line begins with "=="
#bounindex is a list of 24 elements,
#and each element is a int representing the ?th line of "=="
bounindex = sapply( 1:length(flist), function(i) grep('=', alltxt[[i]]))


#alltxt[[15]][1:20]
#flist[15]
#> flist[15]
#[1] "women10Mile_2001"
#the file is different because it doesn't have header
#get the pattern of width of columns, then use read.fwf

#strlocate will give the distance between patterns in the string
strlocate = function(string, pattern){
  sp = gregexpr( pattern, string)
  l = nchar(string)
  #first idenfity if there is pattern in string
  if (length(sp) != 0 && sp[[1]] > 0) {
    #loc contain every locations of pattern in the string
```

```
   loc = as.numeric(sp[[1]])
   tmp = loc
   n = length(loc)
   loc[2:n] = tmp[2:n] - tmp[1:n-1]
   #if there are another characters following the last pattern in the string
   #eg "PATTERN dsdf"
   #then the last distance should be increased to cover the following characters
   if (tmp[n] != l) loc[n+1] = l-loc[n]
  }
  else loc = NA
  #if there is no pattern in the string, return "NA"
  #return value is either "NA" or a vector contain numbers
  return(loc)
}

#wid is a list contain 24 elements , each elements contain the width we need to put
in read.fwf
wid = sapply(1:length(alltxt), function(i) strlocate(alltxt[[i]][bounindex[[i]]], " {1,}")
)

#wid&bounindex has NA in 15th at this step, because the file doesn't contain header
#find the wid of 15th

getwid15 = function( txt15, start){
  #start is the line which the header should have be
  pattern15 = "[0-9] {1,}"
  loc15 = gregexpr( pattern15 , txt15[start])
  loc15 = as.numeric(loc15[[1]])
  tmp = loc15
  loc15[4:5] = tmp[3:4]
  loc15[6] = tmp[4]
  loc15[3] = loc15[3] - 3
  loc15[5] = loc15[5] - 7
  loc15[7] = nchar(txt15[start])
  wid15 = loc15
  tmp = loc15
  wid15[2:7] = tmp[2:7] - tmp[1:6]
  return(wid15)
}
bounindex[[15]] = grep("    1", alltxt[[15]])[1] + 1
wid[[15]] = getwid15(alltxt[[15]], bounindex[[15]] - 1)

#8th&20th file, the "===" dosen't split the the varible Hometown and Net tim
#find the wid of 8th
tmp = wid[[8]]
wid[[8]][6] = tmp[6] - 8
```

```
wid[[8]][7] = 8
wid[[8]][8:10] = tmp[7:9]
rm(tmp)
#find the wid of 20
tmp = wid[[20]]
wid[[20]][6] = tmp[6] - 8
wid[[20]][7] = 8
wid[[20]][8:10] = tmp[7:9]
rm(tmp)

#dat is a large list of 24 elements, and each element is the data.frame
dat = sapply(1:length(flist),
        function(i) read.fwf(flist[i], widths = wid[[i]] ,
                    comment.char = '', skip = bounindex[[i]] - 2, stringsAsFactors = F))

#Warning messages:
#1: In readLines(file, n = thisblock) :
#  incomplete final line found on 'men10Mile_2006'
#2: In readLines(file, n = thisblock) :
#  incomplete final line found on 'men10Mile_2008'
#1: In readLines(file, n = thisblock) :
#  incomplete final line found on 'women10Mile_2004'
#2: In readLines(file, n = thisblock) :
#  incomplete final line found on 'women10Mile_2008'
names(dat) = flist

#tableHeader is trying to give the header of a table
tableHeader = function(table, p, t){
  #p is the row number of header, if there is no header then p=0
  #t are the row numbers needed to be dropped
  if( p != 0){
    name = as.character(unlist(table[p,]))
    #delete the blanket
    name = sapply(1:length(name), function(i) gsub(' ', '' , name[i]) )
    #transfer all the names into lower case
    name = sapply(1:length(name), function(i) tolower(name[i]))
    colnames(table) = name
  }
  #delete certain rows
  if( t !=0) table = table[-t, ]
  return(table)
}

#since no header in 15th file, so p of 15th should be 0
pv = rep(1,length(dat))
pv[15] = 0
```

```
#so, there is no need to delte row in 15th file, so the t of 15th should be 0
tv = lapply(1:length(dat), function(i) c(1,2))
tv[[15]] = 0

#dat is refined with all data.frame has their header
dat = sapply(1:length(dat), function(i) tableHeader(dat[[i]], pv[i], tv[[i]] ))
save.image("~/Dropbox/242/proj/Assignment1/data/AS1.RData")

#===========
test1 = lapply(1:length(dat), function(i) colnames(dat[[i]]))
#however 20th ,11th still have blank in colnames, it may be caused by the txt type
test2 = dat[[11]]
colnames(test2) = sapply(1:ncol(test2), function(i) gsub(' ', '' , colnames(test2)[i]) )
colnames(test2)
#===========

#define the colname by myself LOL
colnames(dat[[11]]) = c('place', 'div/tot', 'num', 'name', 'age', 'hometown', 'guntim',
'nettim', 'pace')
colnames(dat[[15]]) = c('place', 'tot', 'name', 'age', 'hometown', 'guntim', 'nettim')

#need to split div/tot into two columns
#val is a list of 24, each element contains the colnames of that dataframe
val = sapply(1:length(dat), function(i) colnames(dat[[i]]))
#divtot is a list of 24, each element contains the index of div/tot column
#if there is no div/tot, then 0
divtot = sapply( 1:length(val), function(i) which(val[[i]] == 'div/tot'))

#define a function to divide column
divCol = function(table, col, seg, name1, name2){
 #col is the index of column needed to be partitioned
 #seg is the segment sign in that column
 #name1 and name2 are the colnames of new columns
 if (length(col) != 0){
  n = nrow(table)
  div1 = sapply(1:n, function(i) strsplit(as.character(table[i,col]), seg)[[1]][1] )
  div2 = sapply(1:n, function(i) strsplit(as.character(table[i,col]), seg)[[1]][2] )
  table_t = cbind(table[,-col], div1, div2)
  cn = ncol(table_t)
  colnames(table_t)[c(cn-1 , cn)] = c(name1, name2)
 }
 else table_t = table

 return(table_t)
}
```

```
#then parition it in every dataframe in dat
dat = sapply(1:length(val), function(i) divCol(dat[[i]], divtot[[i]], '/', 'div', 'tot'))
save.image("~/Dropbox/242/proj/Assignment1/data/AS1.RData")

#checkheader return overall levels of colnames in the list which contain table
frames
checkheader = function(list){
  #list: a list contains data.frames
  t = sapply(1:length(list), function(i) colnames(list[[i]]))
  t = unlist(t)
  n = levels(factor(t))
  return(n)
}

checkheader(dat)
#some headers have the same meaning but different value

transf = function(data, old, newname){
  #if the data has some colnames which is contain in old,
  #then change them into corresponding ones in newname

  #old, newname  are vectors contain characters
  #tmp : index of colnames of data which is also in old one
  #tmp2: index of old
  tmp = colnames(data) %in% old
  tmp2 = old %in% colnames(data)
  if (sum(tmp) != 0) colnames(data)[tmp] = newname[tmp2]
  return(data)
}


dat = sapply(1:length(dat), function(i)
  transf(dat[[i]],
       c('ag',   '5mi',   'gun',   'net',    'guntim', 'nettim', 'time'),
       c('age', '5mile','guntime','nettime'  ,'guntime','nettime' , 'guntime') )  )
checkheader(dat)
#the colnames are cleaned


#deal with the # * in some time variables
#target contain the potential variables which may have # and *
target = c("10km","5mile", "guntime", "nettime", "pace", "pace.1", "pace.2", "split" )

mark = function(table, range, signal){
  #range is the colunman index we are looking at
  #signal is a special remark we want to specifiy: #,*
```

```
  coln = colnames(table)
  tmp = coln %in% range
  #then the colnames we need to deal with are coln[tmp]
  #colv is a dataframe which only contain coln[tmp] colnums
  colv = table [coln[tmp] ]
  #sig contains the logical value which identify whether there is signal in that coln
  sig = apply(colv, 2, grepl , pattern = signal)
  #sigindex is the index of colnum which have signal
  sigindex = which(apply(sig, 2, sum) != 0 )
  if(length(sigindex) !=0){
    #newcolnum is a colnum which indicate if this obs has signal
    newcolnum = sig[, sigindex]
    table_t = cbind(table, newcolnum)
    colnames(table_t)[ncol(table_t)] =paste0('signal',signal)
    #table_t has a new logical coln which indicate the signal appearance
    #then delete the signal in the coln[tmp]
    unchangev = table_t[coln[tmp][sigindex]]
    change = sapply(unchangev, function(i) gsub( signal, '', i))
    table_t[coln[tmp][sigindex]] = change
  }
  else table_t = table
  return(table_t)
}

dat1 = lapply(dat, function(i) mark(i, target, "#"))
dat1 = lapply(dat1, function(i) mark(i, target, "[*]"))

#deal with the time stuff: conver them into seconds
#->0km,5mile,guntim, nettime,pace,pace.1,pace.2,

toSeconds = function(time){
  time = as.character(time)
  time = gsub("[^0-9:]", '', time)
  tmp = strsplit(time, ":")
  #check how many : in time, then we can know if there is hour, minute, second
  tmp = as.numeric(tmp[[1]])
  n = length(tmp)
  if (n == 2) sum(tmp*c(60 , 1))->time
  if (n == 3) sum(tmp*c(3600 , 60 , 1))->time
  return(time)
}

toSecondsv = function(timev){
  #timev is a vector containing times needed to be formatted
  timev = sapply(timev, function(i) toSeconds(i))
  return(timev)
```

```
}

toSecondstable = function(table, colname){
  #colname is a vector that contain potential time col names in the table
  coln = colnames(table)
  tmp = coln %in% colname
  #coln[tmp] are the colnames
  #sec are the dataframe that contains those transfered colnums
  sec = apply(table[coln[tmp]] , 2, toSecondsv)
  table[coln[tmp]] = sec
  return(table)
}

dat2 = sapply(dat1, function(i) toSecondstable(i, target))
names(dat2) = flist
save.image("~/Dropbox/242/proj/Assignment1/AS1.RData")
```

Appendix 2

```
#1.1Data Profile
va = checkheader(dat2)
va = matrix(va, nrow = 3, ncol = 6, byrow = T)
#number of runner
runnern = sapply(dat2, function(i) nrow(i))
runntable = cbind(runnern[1:12], runnern[13:24])
colnames(runntable) = c("Men", 'Women')
rownames(runntable) = c(1999:2010)
sumn = apply(runntable, 1, sum)
runntable = cbind(runntable, total = sumn)


for (i in 1:length(dat2)){
  dat2[[i]]$guntime = as.numeric(dat2[[i]]$guntime)
  dat2[[i]]$age = as.numeric(dat2[[i]]$age)
}
#median guntime in a  year
guntime = sapply(dat2, function(i) median(i$guntime, na.rm = T))
#max, min age in a year
agemax = sapply(dat2, function(i) max(i$age, na.rm = T))
agemin = sapply(dat2, function(i) min(i$age, na.rm = T))
guntimetable = cbind(guntime = guntime, year = c(1999:2010,1999:2010), sex =
c(rep(1,12),rep(0,12) ),
                     maxa = agemax, mina = agemin)
guntimetable = as.data.frame(guntimetable)
guntimetable$sex = factor(guntimetable$sex)
```

```
levels(guntimetable$sex) = c('Male', 'Femal')

library(ggplot2)
library(RColorBrewer)
#plot median guntime over year
ggplot(guntimetable, aes(x = year, y = guntime, group = sex, color = sex))+
  geom_line(size = 1) +
  labs(title = "Median Guntime in each Year", x = 'Year', color = 'Sex')

#plot agemin, max over year
ggplot() + geom_line( aes(x = year, y = maxa ), guntimetable, colour = "red", size = 1)
+
  geom_line( aes(x = year, y = mina), colour = "lightblue" ,size = 2, guntimetable)+
  facet_wrap(~ sex) + labs(title = "Ages of Runners in each Year", x = 'Year', y =
"Age")

#2Hometown
placev = lapply(dat2, function(i) i$place)
gunv = lapply(dat2, function(i) i$guntime)
placevv = unlist(placev)
gunvv = unlist(gunv)
hometown = lapply(dat2, function(i) i$hometown)
homeinfo = data.frame(place = as.numeric(placevv),gun =as.numeric(gunvv),
hometown = unlist(hometown))
#homeinfo is a dataframe contain all the runners guntime, place, and theri
hometown

#clean hometown variable ( delete "blanks")
homeinfo$hometown = sapply(homeinfo$hometown, function(i) gsub("^[^a-zA-
Z]|[^a-zA-Z]{2,}", "", i))
#levels(factor(homeinfo$hometown))


#then start to get state from homeinfo

getPattern = function(string, pattern) {
  #getPattern return the substring of those pattern in the string
  t = regmatches(string, regexpr(pattern, string))
  t = gsub(" ", "", t)
  #if there is no pattern in the string, then return "Unknown"
  if(length(t) == 0) t= 'Unknown'
  return(t)
}

getState = function(table){
  #get state name from hometown variable
```

```
  sapply(1:nrow(table), function(i) getPattern(table$hometown[i], " [A-Z][A-Z]"))
  #return a chr [1:34234]
}

statev = getState(homeinfo)
homeinfo = transform(homeinfo, state = statev)

#get all the abbreviation of usa states from Internet
library(XML)
usastate =
readHTMLTable("http://www.50states.com/abbreviations.htm#.VSsXHFxWKfQ",
which = 1)
colnames(usastate) = c('state', 'Abb')

#levels(factor(statev))
#checkstate is named logical vector,
#to identify whether those abbrevations we pull out are real state abbrevations
checkstate = sapply(levels(factor(statev)) , function(i) i%in%usastate$Abb)

fakestate = names(which(checkstate == FALSE))

replaceindata = function(ttable, coln, old, newval){
  #replaceindata replace the old value in certain coln of ttable, with newval
  #coln is the index of target colunm
  #old is the old value, newval is the new value
  ttable[which( tolower(ttable[,coln]) == tolower(old)), coln] = newval
  return(ttable)
}

for (i in 1:length(fakestate)){
  homeinfo = replaceindata(homeinfo, 4, fakestate[i], "Unknown")
}
homeinfo$state = droplevels(homeinfo$state)
levels(homeinfo$state)


#total number of runners from each state
##first delete those obs which doesn't have state
state.na = which(homeinfo$state == "Unknown")
statecount = summary(droplevels(homeinfo$state[-state.na] ))
statecounttable = data.frame( count = as.numeric(statecount), state =
names(statecount))
#find top 5 numers of runners of states
topindex = order(statecounttable$count, decreasing = T)[1:5]
top = rep( "Other",nrow(statecounttable))
top[topindex] = c(1:5)
```

```
statecounttable = cbind(statecounttable, top = top )
#statecounttable[topindex,]
#count state top
#49 40212    VA   1
#23 27014    MD   2
#10 20198    DC   3
#41  3315    PA   4
#37  3083    NY   5

#plot it
ggplot(statecounttable , aes( x = state, y = count, fill = factor(top), order =
desc(top)))  +
  geom_bar(stat = 'identity') + coord_flip() + scale_fill_brewer(palette = "Spectral") +
  labs( x = 'State', fill = 'Top State')

#get the top10 runners
top10 = homeinfo[which( homeinfo$place >=1 & homeinfo$place <= 10 ), ]
#total number of top 10 runners from each state
statecount_10 = summary(top10$state)
statecount_10 = statecount_10[which (names(statecount_10) != "Unknown")]
#the rate of top10 runners from each state
state_10_rate = statecount_10/statecount
round(state_10_rate[which(names(state_10_rate) %in% c('VA','MD','DC','PA','NY'))]
, digit = 5)


#plot the detail top10 runners of each state
ggplot(stateplace_10 , aes(factor(state), fill = factor(place), order = desc(place) )) +
  geom_bar() +
  scale_fill_brewer() +
  labs(title = "Top 10 Runners from Different States", x = 'State', fill = 'Rank')


#analysis based on country
#delete the data in 2006, so all the US cities have their state
homeinfo2 = homeinfo[! grepl("2006", rownames(homeinfo) )  ,]
state.na2 = homeinfo2$state == "Unknown"
homeinfo2.nonusa = homeinfo2[state.na2, ]
summary( factor(homeinfo2.nonusa$hometown ))

#clean the country value
oldv = c("Eth",      "Fra",     "Ken",  "Rus", "Tornoto",     "Berlin", "Bri",
        "Mex",    "Rom",       "Us" ,        "Usa",  "London"  ,  "Naruto" ,
        "Scarboroontario", "Toronto", "Uhwiesen" , "Vienna A", "Wellington",
        "AUS", "Baldock Hertfor UK ","Berlin GE", "CAN", "GER", "JAP","Rep Of S.africa",
        "COL","Toronto ON")
```

```
newv = c("Ethiopia" , "France", "Kenya", "Russia", "Canada", "Germany", "United
Kingdom",
       "Mexico", "Romania", "United States", "United States", "United Kindom",
"Japan",
       "Canada",        "Canada", "Switzerland" , "Austria", "New Zealand",
       "Australia", "United Kingdom", "Germany", "Canada", "Germany", "Japan",
"RSA",
       "Colombia", "Canada")

#Personally, i think if a person is from Naruto... we can locate Japan for him/her.

for (i in 1:length(oldv)){
  homeinfo2.nonusa = replaceindata( homeinfo2.nonusa, 3, oldv[i], newv[i])
}

levels(factor(homeinfo2.nonusa$hometown ))

tmp = homeinfo2.nonusa$hometown %in%
  c("Australia" , "Austria" , "Brazil", "Canada", "Colombia", "Denmark" , "Ethiopia",
    "France", "Germany", "Holland", "Japan", "Jordan", "Kenya", "Koengen", "Mexico",
    "Morocco", "New Zealand", "Romania", "Russia", "Switzerland", "Tanzania",
"Ukraine",
    "United Kingdom", "Uruguay" )

homeinfo2.nonusa  = homeinfo2.nonusa[tmp, ]
test = data.frame(summary( factor(homeinfo2.nonusa$hometown)))
top10foreign = homeinfo2.nonusa[which(homeinfo2.nonusa$place %in% c(1:10)), ]
summary(factor(top10foreign$hometown))

#plot the detail top10 runners of each foreign country
ggplot(top10foreign , aes(factor(hometown), fill = factor(place), order = desc(place)
)) + geom_bar() +
  scale_fill_brewer( palette = "Spectral") + coord_flip() +
  labs(title = "Top 10 Runners from Different Foreign Countries", x = 'Country', fill =
'Place')

#analysis Kenya
#Ken, KEN, Kenya
ken = homeinfo[homeinfo$hometown %in% c("Ken", "KEN", "Kenya") , 1:3 ]
year_ken = getPattern(rownames(ken), "[0-9]{4}")
#1 for men, 0 for woman
sex_ken = as.numeric( !grepl("women",rownames(ken)) )
ken = transform(ken, year = as.numeric(year_ken), sex = sex_ken )
ken$sex = factor(ken$sex)
levels(ken$sex) = c("Female", "Male")
ggplot(ken, aes(x = year, y = gun, group = sex, color = sex))+
```

```
  geom_line(size = 1) +
  labs(title = "Guntime of Kenyan Runners in each Year", x = 'Year', color = 'Sex')

countKen = with(ken, table( year, sex ))
countKen = data.frame(countKen)

ggplot(countKen, aes(x = year, y = Freq , group = sex, color = sex))+
  geom_line(size = 1) +
  labs(title = "Number of Kenyan Runners in each Year", x = 'Year', color = 'Sex')
```