R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
#Import Tidyverse
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(dplyr)
library(matlib)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(stringr)
library(sigmoid)
library(knitr)

rm(list=ls()) #remove all variable

setwd("C:\\Users\\kstan\\Documents\\A1\\dat\\")
```

```
#################
#Part 1
#################

####Exercise 1######

#Import datstu
datstu<-read.csv("datstu.csv")
#Import datsss
datsss<-read.csv("datsss.csv")
#Import datjss
datjss<-read.csv("datjss.csv")


#Q1: Count number of students using nrow: 340823
nrow(datstu)
```

```
## [1] 340823
```

```
#Q2: Count number of schools using ndistinct: 898
n_distinct(datsss['schoolcode'])
```

```
## [1] 898
```

```
#Q3: Count number of distinct programs. With blank: 33, without blank: 32
n_distinct(union(union(union(union(union(unique(datstu$choicepgm1),unique(datstu$choicepgm2)),unique(da
```

```
## [1] 33
```

```
#Q4: Count number of distinct program and school combos:3086
datstu1=datstu %>% select('schoolcode1','choicepgm1')
datstu2=datstu %>% select('schoolcode2','choicepgm2')
names(datstu2)[1] <- "schoolcode1"
names(datstu2)[2] <- "choicepgm1"
datstu3=datstu %>% select('schoolcode3','choicepgm3')
names(datstu3)[1] <- "schoolcode1"
names(datstu3)[2] <- "choicepgm1"
datstu4=datstu %>% select('schoolcode4','choicepgm4')
names(datstu4)[1] <- "schoolcode1"
names(datstu4)[2] <- "choicepgm1"
datstu5=datstu %>% select('schoolcode5','choicepgm5')
names(datstu5)[1] <- "schoolcode1"
names(datstu5)[2] <- "choicepgm1"
datstu6=datstu %>% select('schoolcode6','choicepgm6')
names(datstu6)[1] <- "schoolcode1"
names(datstu6)[2] <- "choicepgm1"
new <- rbind(datstu1, datstu2)
new <- rbind(new, datstu3)
new <- rbind(new, datstu4)
new <- rbind(new, datstu5)
new <- rbind(new, datstu6)
a<-unique(new)
nrow(a)
```

```
## [1] 3086
```

```
#Q5: Missing test Score: 179887
sum(is.na(datstu$score))
```

```
## [1] 179887
```

```
#Q6: Applied to the same school more than once:120071
b=(as.numeric(datstu['schoolcode1']==datstu['schoolcode2']))

b[is.na(b)]<-0
c=(as.numeric(datstu['schoolcode1']==datstu['schoolcode3']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode1']==datstu['schoolcode4']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode1']==datstu['schoolcode5']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode1']==datstu['schoolcode6']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode2']==datstu['schoolcode3']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode2']==datstu['schoolcode4']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode2']==datstu['schoolcode5']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode2']==datstu['schoolcode6']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode3']==datstu['schoolcode4']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode3']==datstu['schoolcode5']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode3']==datstu['schoolcode6']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode4']==datstu['schoolcode5']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode4']==datstu['schoolcode6']))
c[is.na(c)]<-0
b=b+c
c=(as.numeric(datstu['schoolcode5']==datstu['schoolcode6']))
c[is.na(c)]<-0
b=b+c
sum(as.numeric(b>0))
```

```
## [1] 120071
```

```
#Q7: Apply to less than 6 choices:17734

sum(as.numeric(is.na(datstu$schoolcode1)|is.na(datstu$schoolcode2)|is.na(datstu$schoolcode3)|is.na(datst
```

```
## [1] 17734
```

```
######Exercise 2########

#Step 1a: First, create a unique code for each school program
datstu2 = datstu %>%
  mutate(choice1=paste0(schoolcode1,choicepgm1),
         choice2=paste0(schoolcode2,choicepgm2),
         choice3=paste0(schoolcode3,choicepgm3),
         choice4=paste0(schoolcode4,choicepgm4),
         choice5=paste0(schoolcode5,choicepgm5),
         choice6=paste0(schoolcode6,choicepgm6))
schools <- datstu2 %>%
  select(choice1:choice6) %>%
  gather(key = "choice2", value = "choice",choice1:choice6)
schools=unique(schools['choice'])

#Step 1B: separate the primary key into school code and program code
schools = schools %>%
  mutate(schoolcode= gsub('\\D','', choice),pgm=gsub('\\d','', choice))


schools['schoolcode']=as.numeric(unlist(schools['schoolcode']))


#Step 2: Cleaning school codes

#get unique values for each school, district, and longitude
datsss2=unique(datsss %>% select(schoolcode:ssslat))

#sort these values
datsss3=arrange(datsss2,'ssslong',by_group='schoolcode')
#check for any instance that has a missing longitude when another
#observation of the same school code does not
datsss3['num']=as.numeric(is.na(datsss3['ssslong']))
temp=datsss3 %>%
  group_by(schoolcode) %>%
  summarize(min=min(num))
datsss3=left_join(datsss3,temp,by='schoolcode')
datsss3=datsss3[(datsss3['num']==datsss3['min']),]

#do the same as above for school district
datsss3['num']=as.numeric(datsss3['sssdistrict']=='')
temp=datsss3 %>%
  group_by(schoolcode) %>%
  summarize(min2=min(num))
datsss3=left_join(datsss3,temp,by='schoolcode')
```

```
datsss3=datsss3[(datsss3['num']==datsss3['min2']),]

#Keep only the desired columns
datsss3=select(datsss3,schoolcode:ssslat)
#Join this column of schools and the new dataset
schools2=left_join(schools,datsss3,by='schoolcode')

#Create a dataset with all of the school choice program and scores
temp=datstu[(datstu['rankplace']==1)&(!is.na(datstu['rankplace'])),]
temp2=temp %>% select('schoolcode1','choicepgm1','score')
names(temp2)[1] <- "schoolcode"
names(temp2)[2] <- "pgm"
temp=datstu[(datstu['rankplace']==2)&(!is.na(datstu['rankplace'])),]
temp3=temp %>% select('schoolcode2','choicepgm2','score')
names(temp3)[1] <- "schoolcode"
names(temp3)[2] <- "pgm"
temp=datstu[(datstu['rankplace']==3)&(!is.na(datstu['rankplace'])),]
temp4=temp %>% select('schoolcode3','choicepgm3','score')
names(temp4)[1] <- "schoolcode"
names(temp4)[2] <- "pgm"
temp=datstu[(datstu['rankplace']==4)&(!is.na(datstu['rankplace'])),]
temp5=temp %>% select('schoolcode4','choicepgm4','score')
names(temp5)[1] <- "schoolcode"
names(temp5)[2] <- "pgm"
temp=datstu[(datstu['rankplace']==5)&(!is.na(datstu['rankplace'])),]
temp6=temp %>% select('schoolcode5','choicepgm5','score')
names(temp6)[1] <- "schoolcode"
names(temp6)[2] <- "pgm"
temp=datstu[(datstu['rankplace']==6)&(!is.na(datstu['rankplace'])),]
temp7=temp %>% select('schoolcode6','choicepgm6','score')
names(temp7)[1] <- "schoolcode"
names(temp7)[2] <- "pgm"
new <- rbind(temp2, temp3)
new <- rbind(new, temp4)
new <- rbind(new, temp5)
new <- rbind(new, temp6)
new <- rbind(new, temp7)

new2=new %>%
  group_by(schoolcode,pgm) %>%
  summarize(cutoff=min(score),quality=mean(score),size=n(),)
```

```
## `summarise()` has grouped output by 'schoolcode'. You can override using the `.groups` argument.
```

```
#Join schools programs to score variables
schools3=left_join(schools2,new2,by=c('schoolcode','pgm'))

head(schools3, 20)
```

```
##                   choice schoolcode             pgm                sssdistrict
## 1    50112Home Economics      50112  Home Economics               Kumasi Metro
## 2      70102General Arts      70102    General Arts               Ho Municipal
```

```
## 3          50702Business      50702        Business         Kwabre (Mamponteng)
## 4       90501Visual Arts      90501     Visual Arts  Kassena/Nankani (Navrongo)
## 5    51802Home Economics      51802  Home Economics    Sekyere East (Effiduase)
## 6      10102General Arts      10102    General Arts          Accra Metropolitan
## 7      80301General Arts      80301    General Arts         East Gonja (Salaga)
## 8      40301General Arts      40301    General Arts           Nzema East (Axim)
## 9          21303Business      21303        Business            East Akim (Kibi)
## 10     80101General Arts      80101    General Arts                      Tamale
## 11 100201General Science     100201 General Science                       Lawra
## 12         30603Business      30603        Business Awutu/Efutu/Senya (Winneba)
## 13         80101Business      80101        Business                      Tamale
## 14        90301Technical      90301       Technical           Builsa (Sandema)
## 15     40903General Arts      40903    General Arts         Wassa West (Tarkwa)
## 16     80102General Arts      80102    General Arts                      Tamale
## 17     10401General Arts      10401    General Arts        Dangme West (Dodowa)
## 18      60301Agriculture      60301     Agriculture                    Berekum
## 19    100102General Arts     100102    General Arts               Wa Municipal
## 20   50501Home Economics      50501  Home Economics      Sekyere West (Mampong)
##       ssslong    ssslat cutoff  quality size
## 1  -1.5971872  6.682060    293 312.3200   50
## 2   0.5261422  6.717607    345 366.1250  120
## 3  -1.5414201  6.806778    242 272.3952  210
## 4  -1.2174410 10.909423    243 273.3333   45
## 5  -0.8442360  7.210829    282 301.3625   80
## 6  -0.1971153  5.607396    388 404.9773   88
## 7  -0.5339396  8.729157    256 283.0500  100
## 8  -2.3118021  5.141226    251 277.9590  195
## 9  -0.4543442  6.178558    316 337.5686  153
## 10 -0.7843482  9.383351    331 347.5000  100
## 11 -2.8009412 10.546398    334 356.1125   80
## 12 -0.5086389  5.544896    248 271.4154   65
## 13 -0.7843482  9.383351    308 333.5600  100
## 14 -1.3374945 10.557073    211 237.0500   40
## 15 -1.9888532  5.276049    283 299.3091  110
## 16 -0.7843482  9.383351    310 327.0000  149
## 17  0.5123865  5.786251    317 337.4250  160
## 18 -2.6317439  7.503565    297 315.6964   56
## 19 -2.2850304 10.030622    291 311.1111   90
## 20 -1.1800768  7.199565    221 257.0800   50
```

```r
######Exercise 3########
#To be clear, I assume you want this for all choices not a
#student's accepted choice
#first, reduce the size of the schools file
schools4=schools3 %>% select(schoolcode,pgm,ssslong,ssslat,cutoff,quality,size)
#merge junior high data onto the student file
merged1=left_join(datstu,datjss,by="jssdistrict")

#merge the file from part 2 onto each school program combo and rename accordingly
merged2=left_join(merged1,schools4,by=c("schoolcode1"="schoolcode",'choicepgm1'='pgm'))
merged2=merged2 %>% rename(ssslong1=ssslong,ssslat1=ssslat,cutoff1=cutoff,quality1=quality,size1=size)
merged3=left_join(merged2,schools4,by=c("schoolcode2"="schoolcode",'choicepgm2'='pgm'))
merged3=merged3 %>% rename(ssslong2=ssslong,ssslat2=ssslat,cutoff2=cutoff,quality2=quality,size2=size)
merged4=left_join(merged3,schools4,by=c("schoolcode3"="schoolcode",'choicepgm3'='pgm'))
```

```
merged4=merged4 %>% rename(ssslong3=ssslong,ssslat3=ssslat,cutoff3=cutoff,quality3=quality,size3=size)
merged5=left_join(merged4,schools4,by=c("schoolcode4"="schoolcode",'choicepgm4'='pgm'))
merged5=merged5 %>% rename(ssslong4=ssslong,ssslat4=ssslat,cutoff4=cutoff,quality4=quality,size4=size)
merged6=left_join(merged5,schools4,by=c("schoolcode5"="schoolcode",'choicepgm5'='pgm'))
merged6=merged6 %>% rename(ssslong5=ssslong,ssslat5=ssslat,cutoff5=cutoff,quality5=quality,size5=size)
merged7=left_join(merged6,schools4,by=c("schoolcode6"="schoolcode",'choicepgm6'='pgm'))
merged7=merged7 %>% rename(ssslong6=ssslong,ssslat6=ssslat,cutoff6=cutoff,quality6=quality,size6=size)

#calculate distance for each school program choice
merged7['dist1']=((69.172*(merged7['ssslong1']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.
merged7['dist2']=((69.172*(merged7['ssslong2']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.
merged7['dist3']=((69.172*(merged7['ssslong3']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.
merged7['dist4']=((69.172*(merged7['ssslong4']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.
merged7['dist5']=((69.172*(merged7['ssslong5']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.
merged7['dist6']=((69.172*(merged7['ssslong6']-merged7['point_x'])*cos(merged7['point_x']/57.3))^2+(69.

head(merged7,20)
```

```
##    X.x score agey male schoolcode1 schoolcode2 schoolcode3 schoolcode4
## 1    1    NA   16    0       50112       50107       50202       50202
## 2    2    NA   17    0       70102       70602       70107       70105
## 3    3    NA   19    0       50702       50705       50115       50706
## 4    4    NA   23    1       90501       90403       90101     9090401
## 5    5    NA   15    0       51802       51701       50205       50207
## 6    6    NA   15    0       10102       50103       51701       50202
## 7    7    NA   22    1       80301       80401       80302       80402
## 8    8    NA   19    1       40301       40401       40402       40302
## 9    9    NA   19    1       21303       21303       21201       21201
## 10  10    NA   16    0       80101       90401       50503       50901
## 11  11    NA   17    0       51802       50601       50503       50602
## 12  12    NA   17    1      100201       90505       80107       90501
## 13  13    NA   16    1       30603       30603       30904       30904
## 14  14    NA   16    0       80101       80102       80104       80103
## 15  15    NA   21    1       90301       90602       80106       90501
## 16  16    NA   17    1       40903       40904       40901       41102
## 17  17    NA   17    1       80102       80101       80108       80105
## 18  18    NA   18    1       10401       70503       10504       20304
## 19  19    NA   19    1       60301       60502       60504       60301
## 20  20    NA   18    1      100102       80701       80201       80701
##    schoolcode5 schoolcode6      choicepgm1      choicepgm2      choicepgm3
## 1        50702       50901  Home Economics    General Arts     Visual Arts
## 2        70605       70603    General Arts        Business    General Arts
## 3        51603       50703        Business  Home Economics        Business
## 4        90102       90303     Visual Arts    General Arts     Agriculture
## 5        51602       50204  Home Economics    General Arts  Home Economics
## 6        50601       51603    General Arts    General Arts    General Arts
## 7        80501       80902    General Arts    General Arts    General Arts
## 8        40202       40304    General Arts    General Arts    General Arts
## 9        20203       20106        Business        Business General Science
## 10       50501       50504    General Arts    General Arts    General Arts
## 11       50603       50901  Home Economics     Visual Arts  Home Economics
## 12       90201       90602 General Science General Science General Science
## 13       30602       30903        Business       Technical    General Arts
```

```
## 14       80401       81001        Business        Business  Home Economics
## 15       10109       10119       Technical       Technical       Technical
## 16       41101       40202    General Arts        Business        Business
## 17       50901       50901    General Arts    General Arts     Agriculture
## 18       20801       21102    General Arts    General Arts    General Arts
## 19       40902       40903     Agriculture    General Arts     Agriculture
## 20       80108       80802    General Arts  Home Economics    General Arts
##            choicepgm4      choicepgm5      choicepgm6
## 1          Visual Arts  Home Economics    General Arts
## 2         General Arts  Home Economics    General Arts
## 3       Home Economics  Home Economics        Business
## 4  Motor Vehicle Mech.     Agriculture    General Arts
## 5         General Arts    General Arts  Home Economics
## 6         General Arts  Home Economics  Home Economics
## 7         General Arts    General Arts    General Arts
## 8          Agriculture     Agriculture     Agriculture
## 9      General Science    General Arts    General Arts
## 10        General Arts    General Arts    General Arts
## 11      Home Economics    General Arts    General Arts
## 12     General Science General Science     Agriculture
## 13            Business        Business    General Arts
## 14      Home Economics     Agriculture     Agriculture
## 15         Agriculture       Technical     Visual Arts
## 16            Business     Agriculture General Science
## 17            Business    General Arts    General Arts
## 18        General Arts     Agriculture       Technical
## 19         Agriculture     Agriculture     Agriculture
## 20        General Arts  Home Economics  Home Economics
##                             jssdistrict rankplace X.y    point_x   point_y
## 1  Bosomtwe/Atwima/Kwanwoma (Kuntanase)        NA  23 -1.5627517  6.559323
## 2                          Ho Municipal        NA 117  0.5261422  6.717607
## 3                 Kwabre (Mamponteng)        NA  27 -1.5414201  6.806778
## 4           Kassena/Nankani (Navrongo)        NA 105 -1.2174410 10.909423
## 5             Atwima Mponua (Nyinahin)        NA  12 -2.1771805  6.549507
## 6                         Kumasi Metro        NA  26 -1.5971872  6.682060
## 7             Nanumba North (Bimbilla)        NA  92 -0.1417642  8.816774
## 8                 Jomoro (Half Assini)        NA 132 -2.8032203  5.069508
## 9                     East Akim (Kibi)        NA  67 -0.4543442  6.178558
## 10           Ejura/Sekyedumase (Ejura)        NA  25 -1.3679653  7.462874
## 11              Sekyere West (Mampong)        NA  30 -1.1800768  7.199565
## 12          Kassena/Nankani (Navrongo)        NA 105 -1.2174410 10.909423
## 13                        Agona Swedru        NA  51 -0.7552425  5.617353
## 14              Tolon Kunbungu (Tolon)        NA  96 -1.1097199  9.527246
## 15                  Accra Metropolitan        NA  78 -0.1971153  5.607396
## 16         Mpohor-Wassa East (Daboase)        NA 134 -1.6975694  5.330796
## 17           Ejura/Sekyedumase (Ejura)        NA  25 -1.3679653  7.462874
## 18                  Ga West (Amasaman)        NA  82 -0.3975105  5.664688
## 19           Wassa Amenfi (Asankragwa)        NA 138 -2.3020179  5.725518
## 20                               Bole        NA  88 -2.2666752  8.629696
##      ssslong1   ssslat1 cutoff1 quality1 size1   ssslong2   ssslat2 cutoff2
## 1  -1.5971872  6.682060     293 312.3200    50 -1.5971872  6.682060     375
## 2   0.5261422  6.717607     345 366.1250   120  0.2673851  6.896852     337
## 3  -1.5414201  6.806778     242 272.3952   210 -1.5414201  6.806778     241
## 4  -1.2174410 10.909423     243 273.3333    45 -0.8802326 10.742456     302
```

```
## 5  -0.8442360  7.210829     282 301.3625    80 -1.5627517  6.559323     352
## 6  -0.1971153  5.607396     388 404.9773    88 -1.5971872  6.682060     357
## 7  -0.5339396  8.729157     256 283.0500   100 -0.1417642  8.816774     223
## 8  -2.3118021  5.141226     251 277.9590   195 -2.8032203  5.069508     241
## 9  -0.4543442  6.178558     316 337.5686   153 -0.4543442  6.178558     316
## 10 -0.7843482  9.383351     331 347.5000   100 -0.8802326 10.742456     271
## 11 -0.8442360  7.210829     282 301.3625    80 -1.5486143  7.001996     284
## 12 -2.8009412 10.546398     334 356.1125    80 -1.2174410 10.909423     339
## 13 -0.5086389  5.544896     248 271.4154    65 -0.5086389  5.544896     239
## 14 -0.7843482  9.383351     308 333.5600   100 -0.7843482  9.383351     286
## 15 -1.3374945 10.557073     211 237.0500    40 -0.7877043 10.924120     234
## 16 -1.9888532  5.276049     283 299.3091   110 -1.9888532  5.276049     309
## 17 -0.7843482  9.383351     310 327.0000   149 -0.7843482  9.383351     331
## 18  0.5123865  5.786251     317 337.4250   160  0.8530558  5.907464     213
## 19 -2.6317439  7.503565     297 315.6964    56 -2.7593036  7.683096     257
## 20 -2.2850304 10.030622     291 311.1111    90 -2.2666752  8.629696     220
##    quality2 size2   ssslong3   ssslat3 cutoff3 quality3 size3   ssslong4
## 1  386.1778   135 -1.8087571  6.681337     321 333.7000    50 -1.8087571
## 2  350.4500    40  0.5261422  6.717607     216 256.8704    54  0.5261422
## 3  261.0500    60 -1.5971872  6.682060     302 326.8625   160 -1.5414201
## 4  322.0750   120 -0.1881377 11.036352     263 279.2000    90 -0.8802326
## 5  364.6582    79 -1.8087571  6.681337     265 277.0400    50 -1.8087571
## 6  369.2182   110 -1.5627517  6.559323     352 364.6582    79 -1.8087571
## 7  245.0583   120 -0.5339396  8.729157     218 260.0917   120 -0.1417642
## 8  268.4444    90 -2.8032203  5.069508     199 236.3023    43 -2.3118021
## 9  337.5686   153 -0.3560941  6.436071     238 287.8750    40 -0.3560941
## 10 290.6267   150 -1.1800768  7.199565     289 309.4840   250 -1.3679653
## 11 303.5000    60 -1.1800768  7.199565     300 315.9000    50 -1.5486143
## 12 366.3913    46 -0.7843482  9.383351     326 354.1375    80 -1.2174410
## 13 260.7692    65 -0.7552425  5.617353     215 246.1933   119 -0.7552425
## 14 307.6333   150 -0.7843482  9.383351     262 283.2000    60 -0.7843482
## 15 257.0000    25 -0.7843482  9.383351     208 243.7889    90 -1.2174410
## 16 330.0667    90 -1.9888532  5.276049     223 252.6000   100 -1.6975694
## 17 347.5000   100 -0.7843482  9.383351     241 260.8778    90 -0.7843482
## 18 244.3000   120 -0.3975105  5.664688     277 297.9375    80 -0.2682494
## 19 289.8467   150 -2.7593036  7.683096     215 268.1154    26 -2.6317439
## 20 238.2619    42 -1.7004058  9.505651     244 267.7944   180 -2.2666752
##      ssslat4 cutoff4 quality4 size4    ssslong5   ssslat5 cutoff5 quality5
## 1   6.681337     321 333.7000    50 -1.54142010  6.806778     272 289.2833
## 2   6.717607     210 249.4854   103  0.26738513  6.896852     221 247.5500
## 3   6.806778     216 249.7000    20 -1.38873518  6.707927     301 316.7800
## 4  10.742456     204 236.1739    23 -0.18813774 11.036352     207 239.4222
## 5   6.681337     205 258.0950   179 -1.56275165  6.559323     289 309.3800
## 6   6.681337     324 336.4440   250 -1.54861426  7.001996     281 300.8250
## 7   8.816774     203 228.5636    55 -0.41415736 10.471273     235 254.8538
## 8   5.141226     246 272.0667    90 -2.63786101  6.258390     220 246.5778
## 9   6.436071     238 287.8750    40 -0.47498974  5.944515     209 258.5455
## 10  7.462874     217 254.4417   120 -1.18007684  7.199565     238 269.3333
## 11  7.001996     295 312.2200    50 -1.54861426  7.001996     236 269.2857
## 12 10.909423     304 320.8444    45 -0.47885746 10.818527     234 255.1250
## 13  5.617353     217 247.5000    80 -0.50863892  5.544896     276 300.0500
## 14  9.383351     261 277.5000    60 -0.14176416  8.816774     218 232.7667
## 15 10.909423     271 294.3000    90 -0.19711526  5.607396     257 279.0850
## 16  5.330796     200 235.1220    41 -1.69756937  5.330796     224 253.6400
```

```
## 17  9.383351     247 263.4889    90 -1.36796534  7.462874     217 254.4417
## 18  5.826003     285 303.2625    80  0.08832825  6.189229     209 257.8667
## 19  7.503565     297 315.6964    56 -1.98885322  5.276049     249 272.8200
## 20  8.629696     196 238.6420   162 -0.78434825  9.383351     239 256.5889
##    size5   ssslong6   ssslat6 cutoff6 quality6 size6      dist1      dist2
## 1     60 -1.3679653  7.462874     217 254.4417   120   8.817538   8.817538
## 2     40  0.2673851  6.896852     261 284.6250   120   0.000000  21.773055
## 3     50 -1.5414201  6.806778     212 257.1477   149   0.000000   0.000000
## 4     90 -1.3374945 10.557073     234 257.5900   100   0.000000  26.023378
## 5    200 -1.8087571  6.681337     288 306.7875    80 102.867013  42.476025
## 6     80 -1.3887352  6.707927     301 316.7800    50 122.056374   0.000000
## 7    130  0.1662941  9.916286     198 233.2750    80  27.796246   0.000000
## 8     90 -2.3118021  5.141226     207 241.5000    40  34.312224   0.000000
## 9     33 -0.2975123  6.112613     284 306.0417   120   0.000000   0.000000
## 10   150 -1.1800768  7.199565     239 274.6400   250 138.838530 229.348828
## 11   140 -1.3679653  7.462874     217 254.4417   120  23.238924  28.919828
## 12    40 -0.7877043 10.924120     281 290.8222    45 112.351348   0.000000
## 13    20 -0.7552425  5.617353     310 325.0583   120  17.777720  17.777720
## 14    30  0.1662941  9.916286     208 230.2500     8  24.605457  24.605457
## 15   200 -0.1971153  5.607396     272 290.9388    49 351.348494 370.030431
## 16    25 -2.6378610  6.258390     231 267.8222    45  20.492784  20.492784
## 17   120 -1.3679653  7.462874     217 254.4417   120 138.838530 138.838530
## 18    45 -0.7990373  6.133319     273 289.8800    50  63.497120  88.117124
## 19    50 -1.9888532  5.276049     271 290.6500    40 125.084615 139.049230
## 20    90 -0.2008451  9.352131     208 224.0000    11  96.913132   0.000000
##         dist3     dist4     dist5     dist6
## 1   18.989094  18.98909  17.18043  63.93522
## 2    0.000000   0.00000  21.77306  21.77306
## 3    9.449630   0.00000  12.57856   0.00000
## 4   71.722324  26.02338  71.72232  25.74807
## 5   27.049629  27.04963  42.47603  27.04963
## 6    8.817528  14.62911  22.38403  14.52407
## 7   27.796246   0.00000 115.98568  78.98413
## 8    0.000000  34.31222  83.02714  34.31222
## 9   19.065069  19.06507  16.25206  11.76808
## 10  22.373010   0.00000  22.37301  22.37301
## 11   0.000000  28.91983  28.91983  22.37356
## 12 109.728226   0.00000  51.46329  29.73642
## 13   0.000000   0.00000  17.77772   0.00000
## 14  24.605457  24.60546  83.04548  92.25981
## 15 264.330096 373.48105   0.00000   0.00000
## 16  20.492784   0.00000   0.00000  91.34381
## 17 138.838530 138.83853   0.00000   0.00000
## 18   0.000000  14.29873  49.45542  42.68710
## 19 139.049230 125.08461  37.88302  37.88302
## 20  72.133419   0.00000 114.95573 151.27794
```

```
#####Exercise 4######
#Take the mean. Too many to list in comment.See output
apply(X=merged7[c('cutoff1','cutoff2','cutoff3','cutoff4','cutoff5','cutoff6')], MARGIN=2, FUN=mean, na
```

```
##  cutoff1  cutoff2  cutoff3  cutoff4  cutoff5  cutoff6
## 294.2372 281.3566 272.6860 263.0766 250.0849 246.1803
```

```
apply(X=merged7[c('quality1','quality2','quality3','quality4','quality5','quality6')], MARGIN=2, FUN=mea
```

```
## quality1 quality2 quality3 quality4 quality5 quality6
## 316.6898 304.5832 296.7897 288.5767 277.5041 274.0787
```

```
apply(X=merged7[c('dist1','dist2','dist3','dist4','dist5','dist6')], MARGIN=2, FUN=mean, na.rm=TRUE)
```

```
##    dist1    dist2    dist3    dist4    dist5    dist6
## 28.33968 28.27001 27.40812 24.44267 28.77922 29.59148
```

```
#take the standard deviation. Too many to list in comment.See output
apply(X=merged7[c('cutoff1','cutoff2','cutoff3','cutoff4','cutoff5','cutoff6')], MARGIN=2, FUN=sd, na.rm
```

```
##  cutoff1  cutoff2  cutoff3  cutoff4  cutoff5  cutoff6
## 54.12556 49.57892 46.96337 45.13990 32.06955 31.44553
```

```
apply(X=merged7[c('quality1','quality2','quality3','quality4','quality5','quality6')], MARGIN=2, FUN=sd
```

```
## quality1 quality2 quality3 quality4 quality5 quality6
## 48.53860 43.82404 41.16625 39.26620 26.73123 26.23097
```

```
apply(X=merged7[c('dist1','dist2','dist3','dist4','dist5','dist6')], MARGIN=2, FUN=sd, na.rm=TRUE)
```

```
##    dist1    dist2    dist3    dist4    dist5    dist6
## 44.32713 42.66886 41.19427 39.10815 28.41479 28.51486
```

```
#ensure that score is not missing or else they aren't in a quantile
merged8=merged7[!is.na(merged7['score']),]

#generate quantiles
merged8['quant']=cut(merged8$score, breaks=quantile(merged8$score,na.rm=T),labels=1:4,include.lowest=TRU
```

```
#create a groupby for the quantile and place it in each.
#means are contained in df, and sd in df2 for each variable,

df <-  merged8 %>%
  group_by(quant) %>%
  summarise_all((mean),na.rm=T)
```

```
## Warning in mean.default(choicepgm1, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm1, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm1, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm1, na.rm = TRUE): argument is not numeric or
## logical: returning NA
```

```
## Warning in mean.default(choicepgm2, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm2, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm2, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm2, na.rm = TRUE): argument is not numeric or
## logical: returning NA


## Warning in mean.default(choicepgm3, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm3, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm3, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm3, na.rm = TRUE): argument is not numeric or
## logical: returning NA


## Warning in mean.default(choicepgm4, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm4, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm4, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm4, na.rm = TRUE): argument is not numeric or
## logical: returning NA


## Warning in mean.default(choicepgm5, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm5, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm5, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm5, na.rm = TRUE): argument is not numeric or
## logical: returning NA


## Warning in mean.default(choicepgm6, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm6, na.rm = TRUE): argument is not numeric or
## logical: returning NA
```

```
## Warning in mean.default(choicepgm6, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(choicepgm6, na.rm = TRUE): argument is not numeric or
## logical: returning NA


## Warning in mean.default(jssdistrict, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(jssdistrict, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(jssdistrict, na.rm = TRUE): argument is not numeric or
## logical: returning NA

## Warning in mean.default(jssdistrict, na.rm = TRUE): argument is not numeric or
## logical: returning NA
```

```
df=df[,c('quant','cutoff1','cutoff2','cutoff3','cutoff4','cutoff5','cutoff6','quality1','quality2','qua
df
```

```
## # A tibble: 4 x 19
##   quant cutoff1 cutoff2 cutoff3 cutoff4 cutoff5 cutoff6 quality1 quality2
##   <fct>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
## 1 1        283.    270.    261.    251.    247.    242.     307.     294.
## 2 2        301.    285.    274.    262.    253.    249.     323.     308.
## 3 3        323.    304.    291.    276.    260.    255.     343.     325.
## 4 4        362.    341.    324.    307.    266.    261.     380.     359.
## # ... with 10 more variables: quality3 <dbl>, quality4 <dbl>, quality5 <dbl>,
## #   quality6 <dbl>, dist1 <dbl>, dist2 <dbl>, dist3 <dbl>, dist4 <dbl>,
## #   dist5 <dbl>, dist6 <dbl>
```

```
df2 <-  merged8 %>%
  group_by(quant) %>%
  summarise_all((sd),na.rm=T)
```

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```

```
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =
## na.rm): NAs introduced by coercion
```

```
df2=df2[,c('quant','cutoff1','cutoff2','cutoff3','cutoff4','cutoff5','cutoff6','quality1','quality2','q

df2
```

```
## # A tibble: 4 x 19
##   quant cutoff1 cutoff2 cutoff3 cutoff4 cutoff5 cutoff6 quality1 quality2
##   <fct>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
## 1 1        44.6    41.4    40.4    39.8    31.2    30.7     38.9     35.9
## 2 2        45.0    42.4    41.4    40.9    31.4    31.1     39.3     36.6
## 3 3        43.5    42.4    41.7    41.3    31.2    31.2     38.3     36.6
## 4 4        38.1    38.2    39.8    41.7    30.3    30.5     34.9     33.9
## # ... with 10 more variables: quality3 <dbl>, quality4 <dbl>, quality5 <dbl>,
## #   quality6 <dbl>, dist1 <dbl>, dist2 <dbl>, dist3 <dbl>, dist4 <dbl>,
## #   dist5 <dbl>, dist6 <dbl>
```

```
#######
#Part 2
#######

##Exercise 5

#Set seed
set.seed(1)
#gen x1-3 an intercept and epsilon
x1 = runif(10000,1,3)
x2=rgamma(10000,3,2)
x3=rbinom(10000,1,0.3)
x0=rep(1,10000)
eps=rnorm(10000,2,1)
#create y
y=0.5+1.2*x1-0.9*x2+0.1*x3+eps
#create y dummy
ybar=mean(y)
ydum=replace(y,y>ybar,1)
ydum=replace(ydum,y<=ybar,0)
```

```r
##Exercise 6

#here is the correlation. It is not similar to 1.2.: 0.4840753
cor(y,x1)
```

```
## [1] 0.4840753
```

```r
#create matrix
X=matrix(c(x0,x1,x2,x3),ncol=4)
#solve for the betas
beta=solve(t(X) %*% X) %*% t(X) %*% y
#here are the betas. They are pretty close: 2.49,1.196,-.89,.125
beta
```

```
##            [,1]
## [1,]  2.4918794
## [2,]  1.1964958
## [3,] -0.8912185
## [4,]  0.1250952
```

```r
#solve for the standard errors
e=y-X%*%beta
sigma=t(e)%*%e/(nrow(X)-ncol(X))
cov=sigma[1]*solve(t(X)%*%X)
#here are the standard errors. They are pretty close: 0.04071 0.01729 0.01149 0.02196
sqrt(diag(cov))
```

```
## [1] 0.04071021 0.01729093 0.01148825 0.02196299
```

```r
##Exercise 7

#Linear probability model
#solve for beta
beta=solve(t(X) %*% X) %*% t(X) %*% ydum
#solve for the standard errors
e=ydum-X%*%beta
sigma=t(e)%*%e/(nrow(X)-ncol(X))
cov=sigma[1]*solve(t(X)%*%X)
#here are the betas and standard errors. They are significant.
#The coefficient says that an increase of 1 in x1 increases the probability
#of ydudm=1 by 34%
beta
```

```
##             [,1]
## [1,]  0.14293129
## [2,]  0.34963841
## [3,] -0.23006264
## [4,]  0.04674943
```

```r
sqrt(diag(cov))
```

```
## [1] 0.016511557 0.007012985 0.004659492 0.008907915
```

```r
#Probit
#create initial function
flike = function(par,x1,x2,x3,yvar)
{
  xbeta           = par[1] + par[2]*x1 + par[3]*x2 + par[4]*x3
  pr              = pnorm(xbeta)
  pr[pr>0.999999] = 0.999999
  pr[pr<0.000001] = 0.000001
  like           = yvar*log(pr) + (1-yvar)*log(1-pr)
  return(-sum(like))
}
#use the truth as our initial guess to speed convergence
truth=c(.5,1.2,-.9,.1)

#solve with hessian
res  = optim(truth,fn=flike,method="BFGS",control=list(trace=6,REPORT=1,maxit=1000),x1=x1,x2=x2,x3=x3,y
```

```
## initial  value 10590.468118
## iter   2 value 6923.664270
## iter   3 value 6620.914729
## iter   4 value 6528.724202
## iter   5 value 6504.493432
## iter   6 value 4955.533511
## iter   7 value 4856.330851
## iter   8 value 4853.447619
## iter   9 value 4853.412220
## iter   9 value 4853.412196
## iter   9 value 4853.412196
## final  value 4853.412196
## converged
```

```r
fisher_info = solve(res$hessian)       # standard formula is -res$hessian but flike is return -like
prop_sigma  = sqrt(diag(fisher_info))
#coefficients: -1.1018124  1.2152102 -0.9194743  0.1721529
probitcoef=res$par
res$par
```

```
## [1] -1.1018124  1.2152102 -0.9194743  0.1721529
```

```r
#standard errors:0.05736692 0.02800771 0.02199492 0.03200662
probitstd=prop_sigma
prop_sigma
```

```
## [1] 0.05736692 0.02800771 0.02199492 0.03200662
```

```r
#at this point these coefficients are uninterpretable, all they tell
# us is that x1 and x3 have a positive effect, x2 has a negative effect
# all are statistically significant


#####logit
flike = function(par,x1,x2,x3,yvar)
{
  xbeta           = par[1] + par[2]*x1 + par[3]*x2 + par[4]*x3
  pr              = exp(xbeta)/(1+exp(xbeta))
  pr[pr>0.999999] = 0.999999
  pr[pr<0.000001] = 0.000001
  like           = yvar*log(pr) + (1-yvar)*log(1-pr)
  return(-sum(like))
}

#solve for logit
res  = optim(truth,fn=flike,method="BFGS",control=list(trace=6,REPORT=1,maxit=1000),x1=x1,x2=x2,x3=x3,y
```

```
## initial  value 7421.795702
## iter   2 value 5274.799933
## iter   3 value 5146.169208
## iter   4 value 4998.004972
## iter   5 value 4997.668968
## iter   6 value 4878.593356
## iter   7 value 4862.372864
## iter   8 value 4861.957120
## iter   9 value 4861.955258
## iter   9 value 4861.955187
## iter   9 value 4861.955187
## final  value 4861.955187
## converged
```

```r
fisher_info = solve(res$hessian)       # standard formula is -res$hessian but flike is return -like
prop_sigma  = sqrt(diag(fisher_info))
#coefficients:-1.8496536  2.0529346 -1.5627390  0.2900699
logitcoef=res$par
res$par
```

```
## [1] -1.8496536  2.0529346 -1.5627390  0.2900699
```

```r
#standard errors: 0.09744291 0.05015588 0.03970207 0.05467439
logitstd=prop_sigma
prop_sigma
```

```
## [1] 0.09744291 0.05015588 0.03970207 0.05467439
```

```r
# the interpretation on coefficients for logit are same as for probit
#see above.
```

```r
#Exercise 8
#calculate marginal effect of probit: -0.3007497  0.3317026 -0.2509788  0.0469907
mean(dnorm(X%*%probitcoef))*probitcoef
```

```
## [1] -0.3007497  0.3317026 -0.2509788  0.0469907
```

```r
#derivative of sigmoid
dsigmoid <- function(x) {
  s <- sigmoid(x)
  s * (1 - s)
}
#marginal effect of logit: -0.29703555  0.32968040 -0.25095998  0.04658228
mean(dsigmoid(X%*%logitcoef))*logitcoef
```

```
## [1] -0.29703555  0.32968040 -0.25095998  0.04658228
```

```r
#For standard errors I use bootstrap
#Take a random sample of the x's and use those to calculate
#the marginal effects standard errors
outs = mat.or.vec(20,4)
for (i in 1:20)
{
  samp     = sample(1:10000,10000,rep=TRUE)
  dat_samp = X[samp,]
  res  = optim(truth,fn=flike,method="BFGS",control=list(trace=6,REPORT=1,maxit=1000),x1=dat_samp[,2],x
  probitcoef=res$par
  outs[i,] = mean(dnorm(dat_samp%*%probitcoef))*probitcoef
}
```

```
## initial  value 10459.540961
## iter   2 value 8215.805602
## iter   3 value 7409.922223
## iter   4 value 7401.098944
## iter   5 value 7375.567775
## iter   6 value 7075.049993
## iter   7 value 6930.077846
## iter   8 value 6929.706732
## iter   9 value 6929.686477
## iter  10 value 6929.686079
## iter  10 value 6929.686078
## final  value 6929.686078
## converged
## initial  value 10452.748075
## iter   2 value 8201.262279
## iter   3 value 7397.688371
## iter   4 value 7387.354057
## iter   5 value 7352.409362
## iter   6 value 7042.795130
## iter   7 value 6931.934493
## iter   8 value 6929.656927
## iter   9 value 6929.654801
## iter   9 value 6929.654711
```

```
## iter   9 value 6929.654711
## final  value 6929.654711
## converged
## initial  value 10423.698755
## iter   2 value 8305.206516
## iter   3 value 7409.695081
## iter   4 value 7400.701032
## iter   5 value 7372.220708
## iter   6 value 7224.058096
## iter   7 value 6930.496099
## iter   8 value 6930.321960
## iter   9 value 6930.298984
## iter   9 value 6930.298933
## iter   9 value 6930.298933
## final  value 6930.298933
## converged
## initial  value 10487.039122
## iter   2 value 8274.417449
## iter   3 value 7415.705045
## iter   4 value 7405.422820
## iter   5 value 7361.701955
## iter   6 value 7010.699801
## iter   7 value 6931.345250
## iter   8 value 6930.047163
## iter   9 value 6930.044304
## iter  10 value 6930.043917
## iter  10 value 6930.043917
## final  value 6930.043917
## converged
## initial  value 10459.436617
## iter   2 value 8249.974674
## iter   3 value 7409.233552
## iter   4 value 7391.575116
## iter   5 value 7326.687655
## iter   6 value 6937.042337
## iter   7 value 6930.529069
## iter   8 value 6930.090436
## iter   9 value 6930.083544
## iter  10 value 6930.083391
## iter  10 value 6930.083391
## final  value 6930.083391
## converged
## initial  value 10471.424404
## iter   2 value 8211.573021
## iter   3 value 7412.576255
## iter   4 value 7370.675910
## iter   5 value 7305.488257
## iter   6 value 7072.500477
## iter   7 value 6932.092214
## iter   8 value 6929.731920
## iter   9 value 6929.728132
## iter  10 value 6929.727894
## iter  10 value 6929.727893
## final  value 6929.727893
```

```
## converged
## initial  value 10551.691996
## iter   2 value 8356.474450
## iter   3 value 7426.616733
## iter   4 value 7412.075768
## iter   5 value 7401.129446
## iter   6 value 6942.658295
## iter   7 value 6927.059720
## iter   8 value 6926.562112
## iter   9 value 6926.542680
## iter  10 value 6926.542261
## iter  10 value 6926.542260
## final  value 6926.542260
## converged
## initial  value 10528.079580
## iter   2 value 8306.348499
## iter   3 value 7416.301333
## iter   4 value 7408.211054
## iter   5 value 7221.759822
## iter   6 value 7027.055819
## iter   7 value 6930.616630
## iter   8 value 6929.046709
## iter   9 value 6929.045788
## iter   9 value 6929.045787
## iter   9 value 6929.045787
## final  value 6929.045787
## converged
## initial  value 10591.739170
## iter   2 value 8281.081206
## iter   3 value 7419.761027
## iter   4 value 7419.083050
## iter   5 value 7078.582530
## iter   6 value 6973.278866
## iter   7 value 6929.996348
## iter   8 value 6929.028526
## iter   9 value 6929.020697
## iter  10 value 6929.019744
## iter  10 value 6929.019744
## final  value 6929.019744
## converged
## initial  value 10433.812181
## iter   2 value 8222.179111
## iter   3 value 7402.684590
## iter   4 value 7394.693164
## iter   5 value 7224.432252
## iter   6 value 7034.057079
## iter   7 value 6930.483361
## iter   8 value 6929.108101
## iter   9 value 6929.104405
## iter  10 value 6929.104108
## iter  10 value 6929.104108
## final  value 6929.104108
## converged
## initial  value 10523.313201
```

```
## iter   2 value 8339.503307
## iter   3 value 7424.830227
## iter   4 value 7413.414242
## iter   5 value 7385.346905
## iter   6 value 7338.587786
## iter   7 value 6929.728057
## iter   8 value 6929.671224
## iter   9 value 6929.665892
## iter   9 value 6929.665844
## iter   9 value 6929.665844
## final  value 6929.665844
## converged
## initial  value 10413.789409
## iter   2 value 8224.791558
## iter   3 value 7389.695502
## iter   4 value 7387.985528
## iter   5 value 7330.301463
## iter   6 value 7042.656788
## iter   7 value 6930.916470
## iter   8 value 6928.791933
## iter   9 value 6928.744912
## iter  10 value 6928.741454
## iter  10 value 6928.741453
## final  value 6928.741453
## converged
## initial  value 10473.223906
## iter   2 value 8303.559749
## iter   3 value 7402.620165
## iter   4 value 7400.835356
## iter   5 value 7381.522882
## iter   6 value 7059.099334
## iter   7 value 6931.899682
## iter   8 value 6929.563612
## iter   9 value 6929.536043
## iter  10 value 6929.533254
## iter  10 value 6929.533253
## final  value 6929.533253
## converged
## initial  value 10544.470754
## iter   2 value 8275.998371
## iter   3 value 7408.482684
## iter   4 value 7401.964662
## iter   5 value 7272.997913
## iter   6 value 7124.053753
## iter   7 value 6929.672737
## iter   8 value 6929.547561
## iter   9 value 6929.540335
## iter  10 value 6929.540169
## iter  10 value 6929.540169
## final  value 6929.540169
## converged
## initial  value 10441.463887
## iter   2 value 8231.956982
## iter   3 value 7397.728451
```

```
## iter   4 value 7374.581039
## iter   5 value 7090.582971
## iter   6 value 6973.907419
## iter   7 value 6928.696470
## iter   8 value 6928.023745
## iter   9 value 6927.997548
## iter  10 value 6927.996945
## iter  10 value 6927.996945
## final  value 6927.996945
## converged
## initial  value 10495.039008
## iter   2 value 8286.294837
## iter   3 value 7425.550501
## iter   4 value 7412.813093
## iter   5 value 7395.779018
## iter   6 value 6981.215276
## iter   7 value 6930.011859
## iter   8 value 6929.508554
## iter   9 value 6929.476227
## iter  10 value 6929.475898
## iter  10 value 6929.475898
## final  value 6929.475898
## converged
## initial  value 10547.788508
## iter   2 value 8300.931792
## iter   3 value 7418.202699
## iter   4 value 7413.131853
## iter   5 value 6984.857514
## iter   6 value 6933.772156
## iter   7 value 6927.281798
## iter   8 value 6926.971039
## iter   9 value 6926.963905
## iter  10 value 6926.963726
## iter  10 value 6926.963726
## final  value 6926.963726
## converged
## initial  value 10472.441969
## iter   2 value 8287.229140
## iter   3 value 7411.367822
## iter   4 value 7398.574813
## iter   5 value 7383.433460
## iter   6 value 6977.774798
## iter   7 value 6930.235788
## iter   8 value 6929.652738
## iter   9 value 6929.615659
## iter  10 value 6929.615210
## iter  10 value 6929.615210
## final  value 6929.615210
## converged
## initial  value 10489.896265
## iter   2 value 8336.907139
## iter   3 value 7405.570211
## iter   4 value 7365.264121
## iter   5 value 7096.349424
```

```
## iter   6 value 7003.234010
## iter   7 value 6930.863080
## iter   8 value 6929.703279
## iter   9 value 6929.700005
## iter  10 value 6929.699486
## iter  10 value 6929.699486
## final  value 6929.699486
## converged
## initial  value 10456.515283
## iter   2 value 8278.133684
## iter   3 value 7412.107257
## iter   4 value 7401.579538
## iter   5 value 7378.020085
## iter   6 value 6992.761202
## iter   7 value 6930.457499
## iter   8 value 6930.064507
## iter   9 value 6930.039280
## iter  10 value 6930.038966
## iter  10 value 6930.038966
## final  value 6930.038966
## converged
```

```
#se of marginal effect of probit
apply(outs,2,sd)
```

```
## [1] 0.037225921 0.014840699 0.008127311 0.010971294
```

```
outs = mat.or.vec(20,4)
for (i in 1:20)
{
  samp     = sample(1:10000,10000,rep=TRUE)
  dat_samp = X[samp,]
  res  = optim(truth,fn=flike,method="BFGS",control=list(trace=6,REPORT=1,maxit=1000),x1=dat_samp[,2],x
  logitcoef=res$par
  outs[i,] = mean(dsigmoid(X%*%logitcoef))*logitcoef
}
```

```
## initial  value 10446.144722
## iter   2 value 8271.184804
## iter   3 value 7403.370004
## iter   4 value 7383.808829
## iter   5 value 7351.362875
## iter   6 value 6977.155332
## iter   7 value 6931.098033
## iter   8 value 6930.251570
## iter   9 value 6930.222989
## iter  10 value 6930.221900
## iter  10 value 6930.221899
## final  value 6930.221899
## converged
## initial  value 10438.865353
## iter   2 value 8207.244457
## iter   3 value 7388.140882
```

```
## iter   4 value 7380.043591
## iter   5 value 7166.525074
## iter   6 value 7004.242978
## iter   7 value 6929.687185
## iter   8 value 6928.144710
## iter   9 value 6928.143401
## iter   9 value 6928.143334
## iter   9 value 6928.143334
## final  value 6928.143334
## converged
## initial  value 10461.805825
## iter   2 value 8247.761536
## iter   3 value 7409.660697
## iter   4 value 7400.174543
## iter   5 value 7367.136514
## iter   6 value 7074.355383
## iter   7 value 6933.283227
## iter   8 value 6930.040223
## iter   9 value 6930.037356
## iter   9 value 6930.037332
## iter   9 value 6930.037332
## final  value 6930.037332
## converged
## initial  value 10410.412058
## iter   2 value 8219.807145
## iter   3 value 7400.036580
## iter   4 value 7395.387393
## iter   5 value 7190.416356
## iter   6 value 7024.964436
## iter   7 value 6930.019365
## iter   8 value 6928.749038
## iter   9 value 6928.699520
## iter  10 value 6928.698139
## iter  10 value 6928.698139
## final  value 6928.698139
## converged
## initial  value 10491.164782
## iter   2 value 8262.196194
## iter   3 value 7405.676325
## iter   4 value 7396.963095
## iter   5 value 7351.162846
## iter   6 value 6985.349285
## iter   7 value 6930.013043
## iter   8 value 6928.999759
## iter   9 value 6928.977302
## iter  10 value 6928.975893
## iter  10 value 6928.975893
## final  value 6928.975893
## converged
## initial  value 10484.746349
## iter   2 value 8308.150943
## iter   3 value 7412.701830
## iter   4 value 7402.472280
## iter   5 value 7384.418604
```

```
## iter   6 value 6936.582469
## iter   7 value 6928.850973
## iter   8 value 6928.479195
## iter   9 value 6928.469333
## iter  10 value 6928.469128
## iter  10 value 6928.469128
## final  value 6928.469128
## converged
## initial  value 10526.581067
## iter   2 value 8283.091603
## iter   3 value 7411.632598
## iter   4 value 7405.009718
## iter   5 value 7334.746716
## iter   6 value 6955.163764
## iter   7 value 6929.341562
## iter   8 value 6928.880872
## iter   9 value 6928.858259
## iter  10 value 6928.857881
## iter  10 value 6928.857881
## final  value 6928.857881
## converged
## initial  value 10537.556840
## iter   2 value 8366.289727
## iter   3 value 7432.995913
## iter   4 value 7423.949598
## iter   5 value 7232.047664
## iter   6 value 7030.060686
## iter   7 value 6928.810996
## iter   8 value 6928.009522
## iter   9 value 6927.993495
## iter  10 value 6927.992382
## iter  10 value 6927.992382
## final  value 6927.992382
## converged
## initial  value 10486.903450
## iter   2 value 8274.262324
## iter   3 value 7417.304891
## iter   4 value 7405.418547
## iter   5 value 7382.428931
## iter   6 value 7083.702435
## iter   7 value 6933.457197
## iter   8 value 6930.284679
## iter   9 value 6930.281158
## iter  10 value 6930.280946
## iter  10 value 6930.280944
## final  value 6930.280944
## converged
## initial  value 10396.069288
## iter   2 value 8225.827694
## iter   3 value 7392.810591
## iter   4 value 7381.194778
## iter   5 value 7365.294504
## iter   6 value 6940.037051
## iter   7 value 6929.095723
```

```
## iter   8 value 6928.688345
## iter   9 value 6928.674462
## iter  10 value 6928.674242
## iter  10 value 6928.674242
## final  value 6928.674242
## converged
## initial  value 10511.814418
## iter   2 value 8284.112274
## iter   3 value 7410.880912
## iter   4 value 7406.631781
## iter   5 value 7384.852490
## iter   6 value 7071.033747
## iter   7 value 6932.821144
## iter   8 value 6930.254971
## iter   9 value 6930.229840
## iter  10 value 6930.227282
## iter  10 value 6930.227282
## final  value 6930.227282
## converged
## initial  value 10447.939874
## iter   2 value 8298.925696
## iter   3 value 7417.191557
## iter   4 value 7405.502947
## iter   5 value 7381.233484
## iter   6 value 7099.285080
## iter   7 value 6933.401112
## iter   8 value 6930.078643
## iter   9 value 6930.073762
## iter  10 value 6930.073388
## iter  10 value 6930.073387
## final  value 6930.073387
## converged
## initial  value 10605.554632
## iter   2 value 8380.067643
## iter   3 value 7425.505923
## iter   4 value 7382.376120
## iter   5 value 7297.662478
## iter   6 value 7146.106048
## iter   7 value 6928.753716
## iter   8 value 6928.675834
## iter   9 value 6928.662539
## iter   9 value 6928.662533
## iter   9 value 6928.662533
## final  value 6928.662533
## converged
## initial  value 10470.722256
## iter   2 value 8296.889145
## iter   3 value 7421.215516
## iter   4 value 7402.153432
## iter   5 value 7358.444067
## iter   6 value 7067.862460
## iter   7 value 6932.163530
## iter   8 value 6929.759648
## iter   9 value 6929.731487
```

```
## iter  10 value 6929.729073
## iter  10 value 6929.729072
## final  value 6929.729072
## converged
## initial  value 10414.144940
## iter   2 value 8206.401145
## iter   3 value 7388.050986
## iter   4 value 7381.001382
## iter   5 value 6997.849072
## iter   6 value 6995.254153
## iter   7 value 6957.120325
## iter   8 value 6927.320509
## iter   9 value 6927.315300
## iter  10 value 6927.315009
## iter  10 value 6927.315007
## final  value 6927.315007
## converged
## initial  value 10456.810288
## iter   2 value 8301.755203
## iter   3 value 7408.246448
## iter   4 value 7396.603097
## iter   5 value 7373.302470
## iter   6 value 7085.365164
## iter   7 value 6933.054058
## iter   8 value 6929.657458
## iter   9 value 6929.654594
## iter   9 value 6929.654540
## iter   9 value 6929.654540
## final  value 6929.654540
## converged
## initial  value 10597.963812
## iter   2 value 8340.850962
## iter   3 value 7434.958175
## iter   4 value 7427.793667
## iter   5 value 7273.130126
## iter   6 value 7065.706146
## iter   7 value 6930.168259
## iter   8 value 6927.804330
## iter   9 value 6927.802807
## iter   9 value 6927.802793
## iter   9 value 6927.802793
## final  value 6927.802793
## converged
## initial  value 10461.966536
## iter   2 value 8255.436426
## iter   3 value 7393.282343
## iter   4 value 7383.929945
## iter   5 value 7342.976692
## iter   6 value 7084.029957
## iter   7 value 6930.151137
## iter   8 value 6929.457993
## iter   9 value 6929.399872
## iter  10 value 6929.399448
## iter  10 value 6929.399447
```

```
## final  value 6929.399447
## converged
## initial  value 10414.984279
## iter   2 value 8267.448045
## iter   3 value 7413.365960
## iter   4 value 7403.086748
## iter   5 value 7378.638030
## iter   6 value 7036.210861
## iter   7 value 6929.924736
## iter   8 value 6929.538951
## iter   9 value 6929.512398
## iter  10 value 6929.512021
## iter  10 value 6929.512019
## final  value 6929.512019
## converged
## initial  value 10475.116862
## iter   2 value 8298.503630
## iter   3 value 7422.333655
## iter   4 value 7412.600018
## iter   5 value 7393.188267
## iter   6 value 6934.914383
## iter   7 value 6928.668902
## iter   8 value 6928.297270
## iter   9 value 6928.290201
## iter  10 value 6928.290044
## iter  10 value 6928.290044
## final  value 6928.290044
## converged
```

```
#se of marginal effect of logit
apply(outs,2,sd)
```

```
## [1] 0.019747765 0.008198305 0.005785904 0.009935170
```