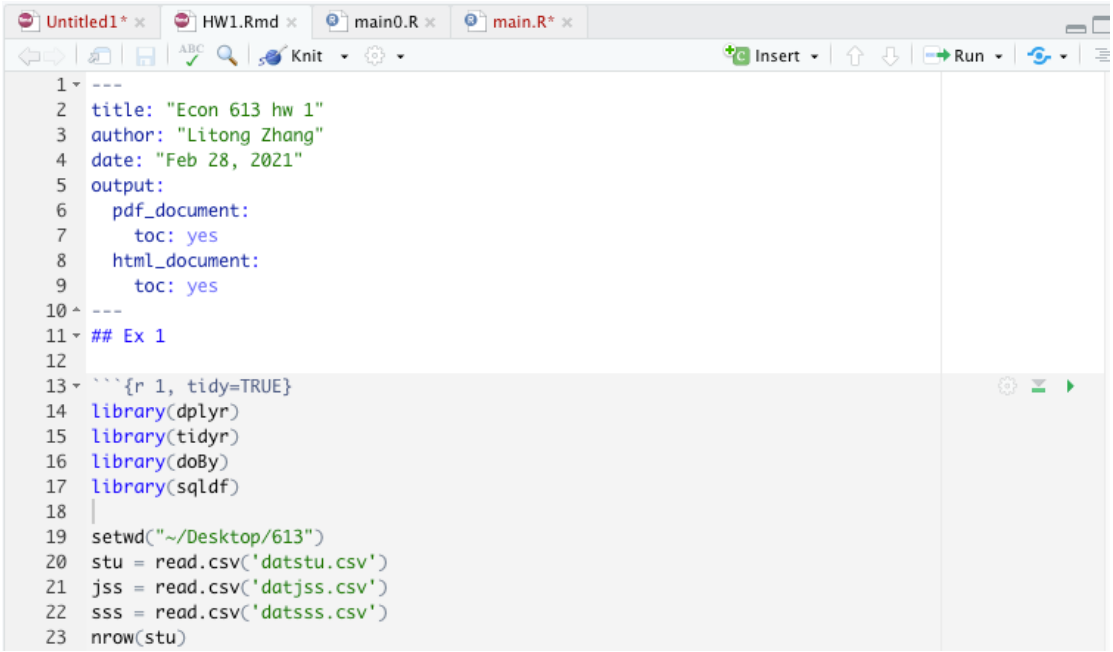
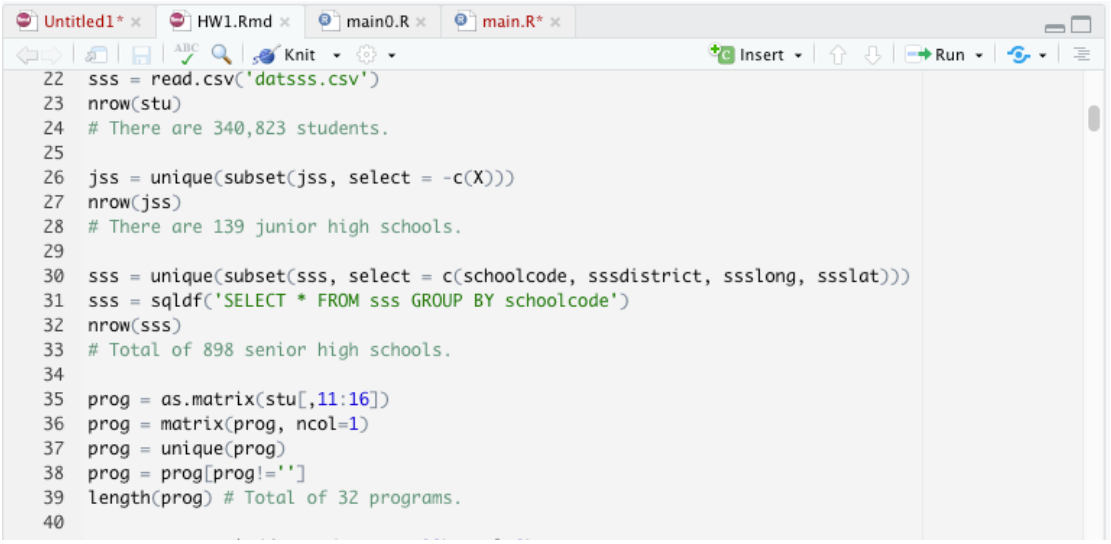


```
---
title: "Econ 613 hw 1"
author: "Litong Zhang"
date: "Feb 28, 2021"
output:
  pdf_document:
    toc: yes
  html_document:
    toc: yes
---
## Ex 1

```{r 1, tidy=TRUE}
library(dplyr)
library(tidyr)
library(doBy)
library(sqldf)

setwd("~/Desktop/613")
stu = read.csv('datstu.csv')
jss = read.csv('datjss.csv')
sss = read.csv('datsss.csv')
nrow(stu)
```

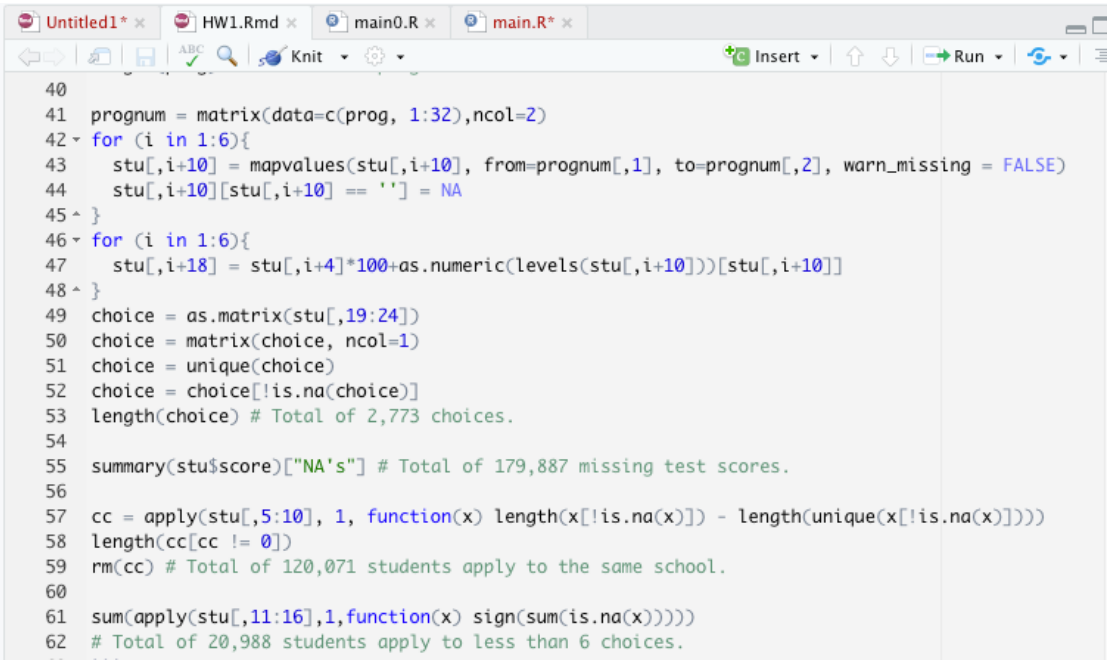


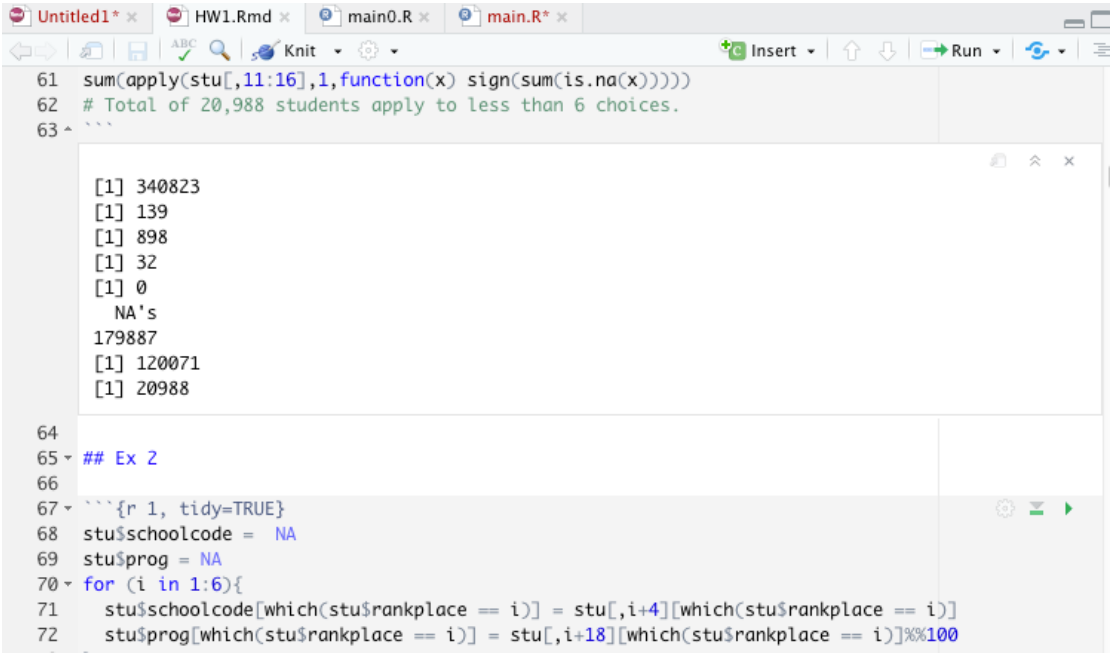
```
sss = read.csv('datsss.csv')
nrow(stu)
# There are 340,823 students.

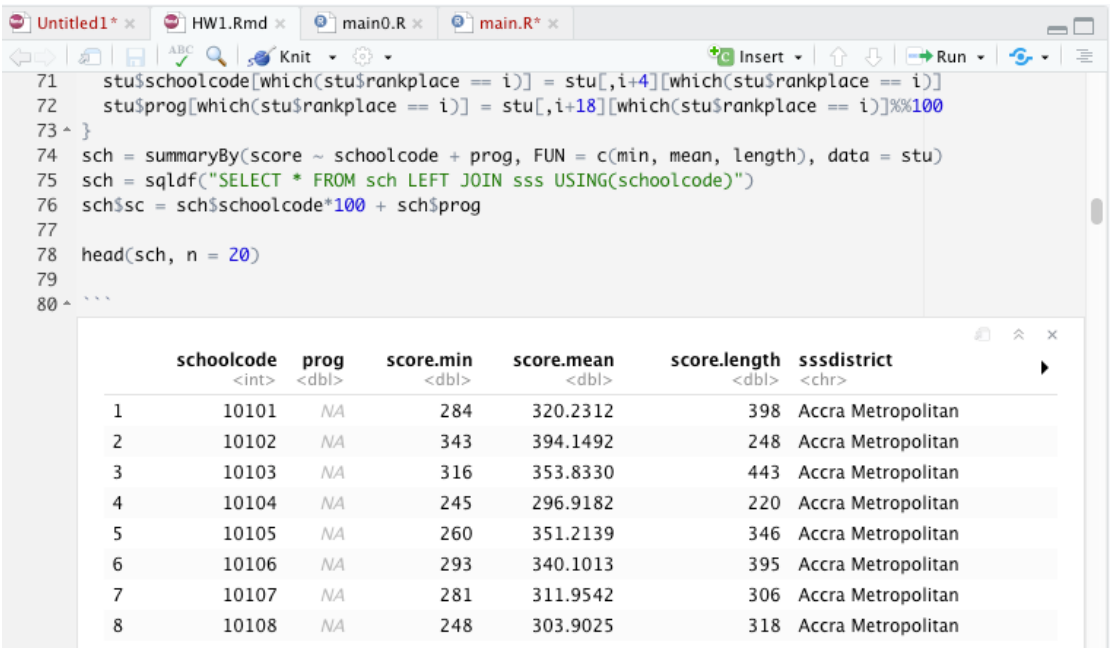
jss = unique(subset(jss, select = -c(X)))
nrow(jss)
# There are 139 junior high schools.

sss = unique(subset(sss, select = c(schoolcode, sssdistrict, ssslong, ssslat)))
sss = sqldf('SELECT * FROM sss GROUP BY schoolcode')
nrow(sss)
# Total of 898 senior high schools.

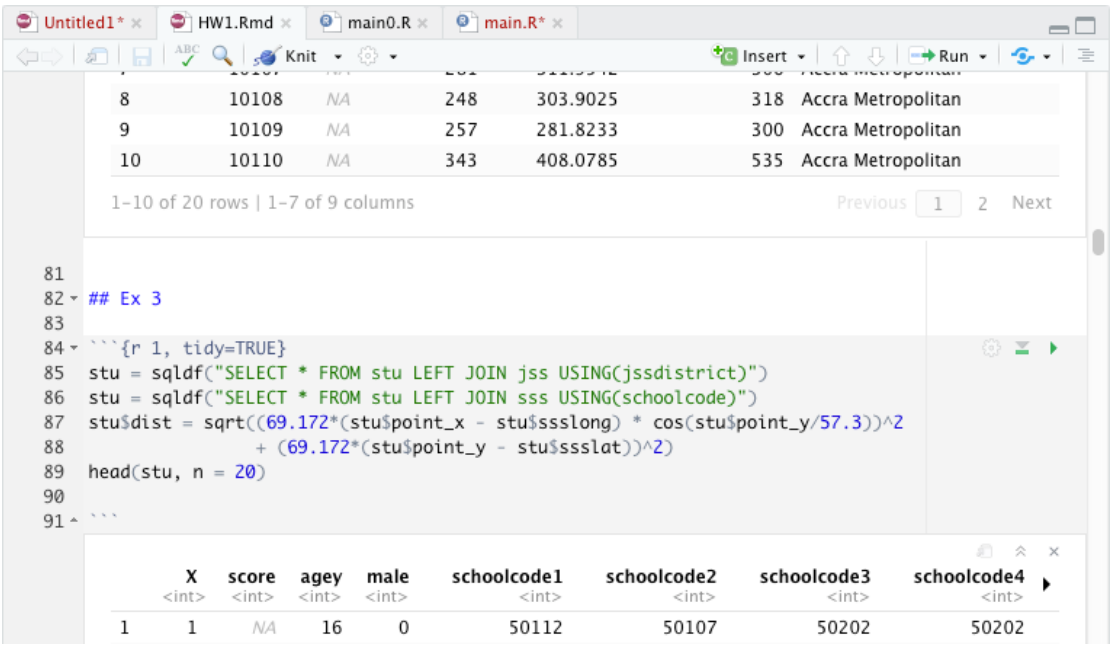
prog = as.matrix(stu[,11:16])
prog = matrix(prog, ncol=1)
prog = unique(prog)
prog = prog[prog!='']
length(prog) # Total of 32 programs.
```
```


Untitled1*
HW1.Rmd
main0.R
main.R*
Knit
Insert
Run
40
41 prognum = matrix(data=c(prog, 1:32),ncol=2)
42 for (i in 1:6){
43 stu[,i+10] = mapvalues(stu[,i+10], from=prognum[,1], to=prognum[,2], warn_missing = FALSE)
44 stu[,i+10][stu[,i+10] == ''] = NA
45 }
46 for (i in 1:6){
47 stu[,i+18] = stu[,i+4]*100+as.numeric(levels(stu[,i+10]))[stu[,i+10]]
48 }
49 choice = as.matrix(stu[,19:24])
50 choice = matrix(choice, ncol=1)
51 choice = unique(choice)
52 choice = choice[!is.na(choice)]
53 length(choice) # Total of 2,773 choices.
54
55 summary(stu$score)["NA's"] # Total of 179,887 missing test scores.
56
57 cc = apply(stu[,5:10], 1, function(x) length(x[!is.na(x)]) - length(unique(x[!is.na(x)])))
58 length(cc[cc != 0])
59 rm(cc) # Total of 120,071 students apply to the same school.
60
61 sum(apply(stu[,11:16],1,function(x) sign(sum(is.na(x)))))
62 # Total of 20,988 students apply to less than 6 choices.

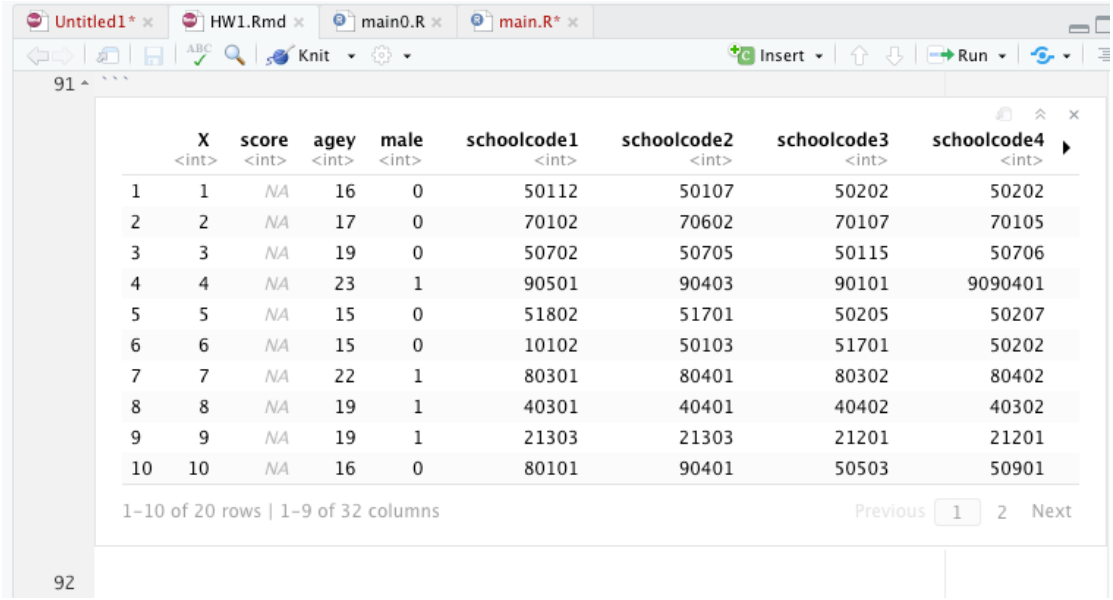

Untitled1*
HW1.Rmd
main0.R
main.R*
Knit
Insert
Run
61 sum(apply(stu[,11:16],1,function(x) sign(sum(is.na(x)))))
62 # Total of 20,988 students apply to less than 6 choices.
63 ```
[1] 340823
[1] 139
[1] 898
[1] 32
[1] 0
NA's
179887
[1] 120071
[1] 20988
64
65 ## Ex 2
66
67 ```{r 1, tidy=TRUE}
68 stu$schoolcode = NA
69 stu$prog = NA
70 for (i in 1:6){
71 stu$schoolcode[which(stu$rankplace == i)] = stu[,i+4][which(stu$rankplace == i)]
72 stu$prog[which(stu$rankplace == i)] = stu[,i+18][which(stu$rankplace == i)]%%100



	schoolcode <int>	prog <dbl>	score.min <dbl>	score.mean <dbl>	score.length <dbl>	sssdistrict <chr>
1	10101	*NA*	284	320.2312	398	Accra Metropolitan
2	10102	*NA*	343	394.1492	248	Accra Metropolitan
3	10103	*NA*	316	353.8330	443	Accra Metropolitan
4	10104	*NA*	245	296.9182	220	Accra Metropolitan
5	10105	*NA*	260	351.2139	346	Accra Metropolitan
6	10106	*NA*	293	340.1013	395	Accra Metropolitan
7	10107	*NA*	281	311.9542	306	Accra Metropolitan
8	10108	*NA*	248	303.9025	318	Accra Metropolitan

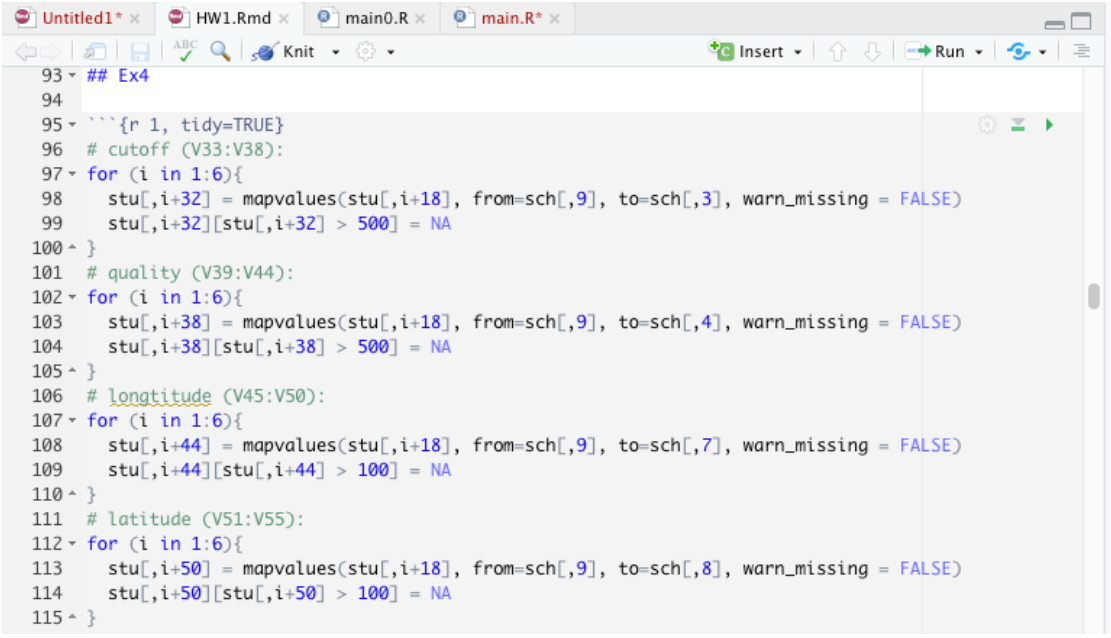


	X <int>	score <int>	agey <int>	male <int>	schoolcode1 <int>	schoolcode2 <int>	schoolcode3 <int>	schoolcode4 <int>
1	1	*NA*	16	0	50112	50107	50202	50202



	X <int>	score <int>	agey <int>	male <int>	schoolcode1 <int>	schoolcode2 <int>	schoolcode3 <int>	schoolcode4 <int>
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

1–10 of 20 rows | 1–9 of 32 columns Previous 1 2 Next




```
## Ex4

```{r 1, tidy=TRUE}
# cutoff (V33:V38):
for (i in 1:6){
  stu[,i+32] = mapvalues(stu[,i+18], from=sch[,9], to=sch[,3], warn_missing = FALSE)
  stu[,i+32][stu[,i+32] > 500] = NA
}
# quality (V39:V44):
for (i in 1:6){
  stu[,i+38] = mapvalues(stu[,i+18], from=sch[,9], to=sch[,4], warn_missing = FALSE)
  stu[,i+38][stu[,i+38] > 500] = NA
}
# longtitude (V45:V50):
for (i in 1:6){
  stu[,i+44] = mapvalues(stu[,i+18], from=sch[,9], to=sch[,7], warn_missing = FALSE)
  stu[,i+44][stu[,i+44] > 100] = NA
}
# latitude (V51:V55):
for (i in 1:6){
  stu[,i+50] = mapvalues(stu[,i+18], from=sch[,9], to=sch[,8], warn_missing = FALSE)
  stu[,i+50][stu[,i+50] > 100] = NA
}
```
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
  stu[,i+50][stu[,i+50] > 100] = NA
}
# distance (V56:V61):
for (i in 1:6){
  stu[,i+56] = sqrt((69.172*(stu$point_x - stu[,i+44]) * cos(stu[,i+50]/57.3))^2
                    + (69.172*(stu$point_y - stu[,i+50]))^2)
}

# create tables to report Exercise 4:

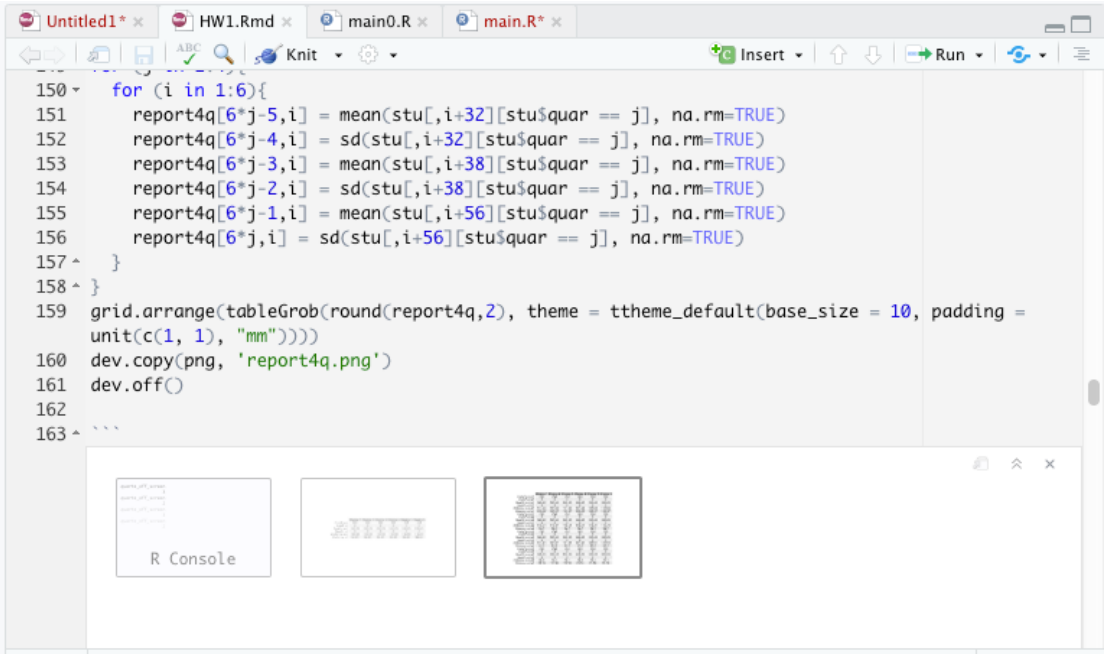
report4 = matrix(rep(NA, 36), ncol=6, byrow=TRUE)
colnames(report4) = c('Choice 1', 'Choice 2', 'Choice 3', 'Choice 4', 'Choice 5', 'Choice 6')
rownames(report4) = c('Cutoff avg', 'Cutoff sd', 'Quality avg', 'Quality sd', 'Distance avg',
'Distance sd')
for (i in 1:6){
  report4[1,i] = mean(stu[,i+32], na.rm=TRUE)
  report4[2,i] = sd(stu[,i+32], na.rm=TRUE)
  report4[3,i] = mean(stu[,i+38], na.rm=TRUE)
  report4[4,i] = sd(stu[,i+38], na.rm=TRUE)
  report4[5,i] = mean(stu[,i+56], na.rm=TRUE)
  report4[6,i] = sd(stu[,i+56], na.rm=TRUE)
}
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
grid.arrange(tableGrob(round(report4,2), theme = ttheme_default(base_size = 10, padding =
unit(c(1, 1), "mm"))))
setwd("~/Desktop/613")
dev.copy(png, 'report4.png')
dev.off()

# students' quartile of score:
stu$quar = cut(stu$score, breaks = quantile(stu$score, probs = seq(0, 1, .25), na.rm = TRUE),
include.lowest = TRUE, labels = 1:4)
report4q = matrix(rep(NA, 36*4), ncol=6, byrow=TRUE)
colnames(report4q) = colnames(report4)
rownames(report4q) = c('Cutoff avg q1', 'Cutoff sd q1', 'Quality avg q1', 'Quality sd q1',
'Distance avg q1', 'Distance sd q1',
                       'Cutoff avg q2', 'Cutoff sd q2', 'Quality avg q2', 'Quality sd q2',
'Distance avg q2', 'Distance sd q2',
                       'Cutoff avg q3', 'Cutoff sd q3', 'Quality avg q3', 'Quality sd q3',
'Distance avg q3', 'Distance sd q3',
                       'Cutoff avg q4', 'Cutoff sd q4', 'Quality avg q4', 'Quality sd q4',
'Distance avg q4', 'Distance sd q4'
                       )
for (j in 1:4){
  for (i in 1:6){
    report4q[6*j-5,i] = mean(stu[,i+32][stu$quar == j], na.rm=TRUE)
```



	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Choice 6
Cutoff avg	284	284	284	284	284	284
Cutoff sd	0	0	0	0	0	0
Quality avg	320.23	320.23	320.23	320.23	320.23	320.23
Quality sd	0	0	0	0	0	0
Distance avg	116.99	116.99	116.99	116.99	116.99	116.99
Distance sd	96.83	96.83	96.83	96.83	96.83	96.83

	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Choice 6
Cutoff avg q1	284	284	284	284	284	284
Cutoff sd q1	0	0	0	0	0	0
Quality avg q1	320.23	320.23	320.23	320.23	320.23	320.23
Quality sd q1	0	0	0	0	0	0
Distance avg q1	131.63	131.63	131.63	131.63	131.63	131.63
Distance sd q1	105.37	105.37	105.37	105.37	105.37	105.37
Cutoff avg q2	284	284	284	284	284	284
Cutoff sd q2	0	0	0	0	0	0
Quality avg q2	320.23	320.23	320.23	320.23	320.23	320.23
Quality sd q2	0	0	0	0	0	0
Distance avg q2	116.41	116.41	116.41	116.41	116.41	116.41
Distance sd q2	92.7	92.7	92.7	92.7	92.7	92.7
Cutoff avg q3	284	284	284	284	284	284
Cutoff sd q3	0	0	0	0	0	0
Quality avg q3	320.23	320.23	320.23	320.23	320.23	320.23
Quality sd q3	0	0	0	0	0	0
Distance avg q3	101.91	101.91	101.91	101.91	101.91	101.91
Distance sd q3	84.4	84.4	84.4	84.4	84.4	84.4
Cutoff avg q4	284	284	284	284	284	284
Cutoff sd q4	0	0	0	0	0	0
Quality avg q4	320.23	320.23	320.23	320.23	320.23	320.23
Quality sd q4	0	0	0	0	0	0
Distance avg q4	76.1	76.1	76.1	76.1	76.1	76.1
Distance sd q4	74.18	74.18	74.18	74.18	74.18	74.18

````
164
165 ## Ex 5
166
167 ```{r 1, tidy=TRUE}
168 set.seed(1)
169 X1 = runif(10000, 1, 3)
170 X2 = rgamma(10000, 3, scale=2)
171 X3 = rbinom(10000, 1, .3)
172 eps = rnorm(10000, 2, 1)
173 b = c(.5, 1.2, -.9, .1)
174 X = matrix(c(rep(1, 10000), X1, X2, X3), ncol=4)
175 Y = (X %*% as.matrix(b)) + eps
176 ydum = as.numeric(Y > mean(Y))
177
178
179 ```
180
````

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
181 ## Ex 6
182
183 ```{r 1, tidy=TRUE}
184 cor(X1, Y)
185 # The correlation is significantly different from 1.2.
186
187 coefYX = solve(t(X)%*%X)%*%t(X)%*%Y
188 residYX = Y - X%*%coefYX
189 varYX = (t(residYX)%*%residYX)/(length(Y)-length(b))
190 vcovYX = c(varYX) * solve(t(X)%*%X)
191 seYX = sqrt(diag(vcovYX))
192
193 # likelihood function for the probit model:
194 probitl = function(beta, y, X){
195   p = pnorm(X%*%beta)
196   l = sum(y*log(p)) + sum((1-y)*log(1-p))
197   return(l)
198 }
199 probitlb = function(par){
200   -probitl(par, ydum, X)
201 }
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
probitlb = function(par){
  -probitl(par, ydum, X)
}
# gradient function for the probit model:
dprobitl = function(beta, y, X){
  p = pnorm(X%*%beta)
  f = dnorm(X%*%beta)
  n = length(y)
  k = length(beta)
  d = t(matrix(rep(f/p, k), nrow=n)*X)%*%y -
    t(matrix(rep(f/(1-p), k), nrow=n)*X)%*%(1-y)
  return(d)
}
dprobitlb = function(par){
  -dprobitl(par, ydum, X)
}
# steepest ascent optimization algorithm:
par = rep(0,4)
for (i in 1:10000){
  par = par - .00001 * c(dprobitlb(par))
}

```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```

```

```
          [,1]
[1,] 0.2162589
```

```
## Ex 7

```{r 1, tidy=TRUE}
# probit model:
optimprobit = optim(par=rep(0,4), fn=probitlb, hessian=T)
# logit model:
logitl = function(beta, y, X){
  p = 1/(1 + exp(-X%*%beta))
  l = sum(y*log(p)) + sum((1-y)*log(1-p))
  return(l)
}
logitlb = function(par){
  -logitl(par, ydum, X)
}
optimlogit = optim(par=rep(0,4), fn=logitlb, hessian=T)
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
optimlogit = optim(par=rep(0,4), fn=logitlb, hessian=T)
# linear probability model:
lpms = function(par){
  s = (ydum - X%*%par)^2
  return(sum(s))
}
optimlpm = optim(par=rep(0,4), fn=lpms, hessian=T)
# The probit model returns parameters closest to the actual values,
# and the other two models have approximately the same accuracy.
tprobit = optimprobit$par/sqrt(diag(solve(optimprobit$hessian)))
tlogit = optimlogit$par/sqrt(diag(solve(optimlogit$hessian)))
tlpm = optimlpm$par/sqrt(diag(solve(optimlpm$hessian)))
# All the estimates from all the three models are statistically
# significant, except the last parameter in the linear probability
# model under 95% confidence level.

```
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
## Ex 8

```{r 1, tidy=TRUE}
# probit model: F'(x) = (-exp(-x^2/2))/sqrt(2*pi)
# logit model: F'(x) = exp(-x)/(1 + exp(-x))^2
meoptimprobit = dnorm(X%*%optimprobit$par)%*%optimprobit$par
meoptimlogit = (exp(-X%*%optimlogit$par)/(1 + exp(-X%*%optimlogit$par))^2)%*%optimlogit$par
# probit standard error:
vcovprobit = solve(optimprobit$hessian)
jacprobit = array(0,dim=c(4,4,10000))
for (k in 1:10000){
  jacprobit[,,k] = diag(4)
  f = dnorm(X[k,]%*%optimprobit$par)
  for (i in 1:4){
    for (j in 1:4){
      jacprobit[i,j,k] = f*(jacprobit[i,j,k]-
                              (X[k,]%*%optimprobit$par)*optimprobit$par[j]*X[k,i])
    }
  }
}
jbprobit = rowMeans(jacprobit, dims=2)
deltaprobit = sqrt(diag(t(jbprobit)%*%vcovprobit%*%jbprobit))
```

Untitled1* | HW1.Rmd | main0.R | main.R*

Knit | Insert | Run

```
}
jbprobit = rowMeans(jacprobit, dims=2)
deltaprobit = sqrt(diag(t(jbprobit)%*%vcovprobit%*%jbprobit))
# logit standard error:
vcovlogit = solve(optimlogit$hessian)
jaclogit = array(0,dim=c(4,4,10000))
for (k in 1:10000){
  jaclogit[,,k] = diag(4)
  f = 1/(1 + exp(-X[k,]%*%optimlogit$par))
  for (i in 1:4){
    for (j in 1:4){
      jaclogit[i,j,k] = f * (1-f) * (
        jaclogit[i,j,k] + (1-2*f)*(optimlogit$par[j]*X[k,i])
      )
    }
  }
}
jblogit = rowMeans(jaclogit, dims=2)
deltalogit = sqrt(diag(t(jblogit)%*%vcovlogit%*%jblogit))


```
```

179:4 | Ex 5 | R Markdown