Session1-5 for Econometrics

sn0wfree

11/12/2016

Contents

1 Session1-5

1.1 Basic Objects

1.1.1 list enviornment

```
ls()# to list the current environment
```

```
## character(0)
```

1.2 Vector and Matrix

1.2.1 Vector

to create Vector based on c() function c() function is the function to create tuple or called Vector

```
# c():create a tuple, also can be seen a Vector if it is numberic tuple. because the tuple is the list
nv=c(6,5,4,3,2,1)
cv=c("a string of characters","also a string of characters")
nv[3]
```

```
## [1] 4
```

```
cv[1]
```

```
## [1] "a string of characters"
```

```r
vec=vector(mode = "list",length = 2)# in the mode, there are three options:"list", "experession" and "a
length(nv) #show the number of elements
```

```
## [1] 6
```

```r
min(nv) #minimum value of nv
```

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

```r
which.min(nv) #the index of the minimum value of nv
```

```
## [1] 6
```

```r
max(nv) #maximum value of nv
```

```
## [1] 6
```

```r
which.max(nv) #the index of the maximum value of nv sort(nv,decreasing=TRUE) #list nv's elements of nv
```

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

```r
nv[2:5] #print 2-5 elements of nv
```

```
## [1] 5 4 3 2
```

the code for matrix is matrix():it look like a dataframe

```r
m1=matrix(nv,nrow=3,ncol=2)
m1
```

```
##      [,1] [,2]
## [1,]    6    3
## [2,]    5    2
## [3,]    4    1
```

```r
class(nv)#class() function is to show the class of target element, like the type function in Python
```

```
## [1] "numeric"
```

```r
class(m1)
```

```
## [1] "matrix"
```

```r
#show the dimension
attributes(nv)
```

```
## NULL
```

```r
attributes(m1)# show the dimension of element:(row,colmn)
```

```
## $dim
## [1] 3 2
```

```r
dim(m1)#another function can show the dimension as well:(row,colmn)
```

```
## [1] 3 2
```

```r
m1[3,2] #print the (3, 2) element of m1
```

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

```r
m1[,2] #print the 2nd column of m1
```

```
## [1] 3 2 1
```

```r
2*m1 #scalar multiplication
```

```
##      [,1] [,2]
## [1,]   12    6
## [2,]   10    4
## [3,]    8    2
```

```r
diag(2) #identity matrix of dimension 2
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```r
rep(1,2) #replicate 1 twice,this function can create the identity matrix
```

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

```r
cbind(rep(1,6),nv) #put two vectors vertically together to make a matrix
```

```
##        nv
## [1,] 1  6
## [2,] 1  5
## [3,] 1  4
## [4,] 1  3
## [5,] 1  2
## [6,] 1  1
```

```r
m3=m1%*%t(m1) #t() is to transpose a matrix and %*% is for matrix multiplication

qr(m3)#A=QR,use QR Decomposition: an orthogonal matrix Q and an upper triangular matrix R. QR decomposi
```

```
## $qr
##             [,1]        [,2]        [,3]
## [1,] -63.6396103 -51.1945310 -38.7494516
## [2,]   0.5656854  -0.3464102  -0.6928203
## [3,]   0.4242641   0.9517782   0.0000000
##
## $rank
## [1] 2
##
## $qraux
## [1] 1.707107 1.306787 0.000000
##
## $pivot
## [1] 1 2 3
##
## attr(,"class")
## [1] "qr"
```

```
qr(m3)$rank #rank of m3
```

```
## [1] 2
```

```
diag(m3)#extract the diagonal of a matrix(m3)
```

```
## [1] 45 29 17
```

```
sum(diag(m3)) #trace of m3: the trace of an n-by-n square matrix A is defined to be the sum of the elem
```

```
## [1] 91
```

```
diag(m3)<-1#replace the diagonal of matrix with 1
solve(m3) #inverse of m3
```

```
##             [,1]        [,2]        [,3]
## [1,] -0.01199702  0.01385991  0.01900149
## [2,]  0.01385991 -0.01808246  0.02359662
## [3,]  0.01900149  0.02359662 -0.03216592
```

1.3 Infinite Values and Missing Data

the computer software exist several limit on the precision format and 32-bit/64-bit limit

```
typeof(1.9)# another kind of class funtion with more detail in the numeric part, it categories the type
```

```
## [1] "double"
```

```
as.integer(1.9)# this function transfer the 1.9(double numeric) into the integer number
```

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

```r
is.integer(1.9)# this function judege the type of this number(a),return the pool value:(FASLE or TRUE)
```

```
## [1] FALSE
```

```r
as.integer(2^31 - 1) #transfer (2^31 - 1) into integer number ,larger than this number  can not  be cal
```

```
## [1] 2147483647
```

```r
2^1024 #this is also too larger than the upper limits which is based on the 32-bit limit of software
```

```
## [1] Inf
```

1.4 Probability Related Functions

```r
runif(n=10,min = 1,max = 10)# generate 10 random numbers uniformly from 1 to 10
```

```
##  [1] 4.037550 5.823609 6.375857 6.889476 8.248140 2.389865 1.203745
##  [8] 6.520246 3.269259 5.277858
```

```r
rnorm(n=10,mean=0,sd=1)# generate 10 normally distributed random numbers with mean=0 and standard devia
```

```
##  [1]  0.4269484 -0.7474745  1.5034915  0.5583608  2.2922638 -1.0136342
##  [7]  0.2130808  1.3395750 -1.5583255  1.0500899
```

```r
rt(n=10,df=2) # generate 10 random numbers from student t-distribution with df=2 degress of freedom
```

```
##  [1] -1.0503787 10.5623084  0.3140259  0.2112197 -1.8549355  0.4716793
##  [7]  1.0865071  0.5964313 -3.0527752 -1.7637586
```

```r
dnorm(0.005) # return the density function value of standard normal distribution evaluated at 0.005
```

```
## [1] 0.3989373
```

```r
pnorm(0.5) # return the cumulative distribution function of standard normal distribution eveluated at 0
```

```
## [1] 0.6914625
```

```r
qnorm(p=0.25) # return the 0.25th quantile of standard normal distribution(0<=p<=1)
```

```
## [1] -0.6744898
```

1.5 Other Useful Objetcs

1.5.1 Factor object

organize the data point into different categories.- tag as character variable retag the data point with factor tag,

```r
gender<-factor(c("male","male","male","female","female"))# create two level categories: male and female
levels(gender)# show the categories.
```

```
## [1] "female" "male"
```

1.5.2 Time Series Object

use ts() function to create a special vector with a time index.

`ts(data,start=c(start-year,start-mounth),frequency)` will create a series of random normally dirtributed numbers (n=100) and order them by mounths, and start mouths is april 1987 (12 mouths as a circle; mounthly times series). if change the freq as 4, so it will create quarterly times series. other freq will not be defined as times series,but based on year.

```r
myts1=ts(data=rnorm(100),start=c(1987,4),freq = 12)#create a series of random normally dirtributed numb
tsp(myts1) # return the start, end time(in numeric format) and frequency
```

```
## [1] 1987.25 1995.50   12.00
```

1.5.3 List Object

use list objects(`list()`) to contain the data with different types/classes

```r
lobj<-list(name="Jack",stid=123,score=100)# create a list with different attributes,it likes the dictio
lobj$name# will print the value of name in list of lobj
```

```
## [1] "Jack"
```

```r
lobj[2]# will print the 2rd element,include the name and value
```

```
## $stid
## [1] 123
```

```r
lobj[[2]] # will print the value of 2rd element, if there not exist the name, will use the order instea
```

```
## [1] 123
```

```r
lobj2 =vector(mode="list",length = 2)# in the mode, there are three options:"list", "experession" and "
lobj2[[1]]<-lobj # put the lobj into the 2rd element of lobj2
lobj2[[1]]<-list(name="Jack",stid=123,score=100)# equivalently lobj2[[1]]<-lobj
lobj2[[2]]<-list(name="Mike",stid=245,score=60)
lobj2[[1]]$stid # the find student's ID number
```

```
## [1] 123
```

```r
lobj2[[2]][[3]] # the second student's score
```

```
## [1] 60
```

1.5.4 Dataframe Object

the dataframe is the most important function for data mining and data analysis, and it can help us to contain exteran data file into R. and almost all the data will be handle in the dataframe use `setwd()` fucntion to set up the working directory, that can help us more easy to import the data and #### set up working directory and import data

```
setwd("/Users/sn0wfree/Dropbox/PhD(1st)/sn0wfree.github.io/BST169_Econometrics/computing/session1-5")#s
london<-read.csv('london.csv')# import data from london.csv
class(london)
```

```
## [1] "data.frame"
```

```
typeof(london)
```

```
## [1] "list"
```

1.5.4.1 structure, head and colname

now about look the structure of data, head of data and the label or colmn's name

```
str(london)# show the structure of data, include the types, observation and variables
```

```
## 'data.frame':    1519 obs. of  11 variables:
##  $ X...  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ wfood : num  0.427 0.374 0.194 0.444 0.333 ...
##  $ wfuel : num  0.1342 0.1686 0.4056 0.1258 0.0824 ...
##  $ wcloth: num  0 0.0091 0.0012 0.0539 0.0399 ...
##  $ walc  : num  0.0106 0.0825 0.0513 0.0397 0.1571 ...
##  $ wtrans: num  0.1458 0.1215 0.2063 0.0652 0.2403 ...
##  $ wother: num  0.282 0.244 0.141 0.272 0.147 ...
##  $ totexp: int  50 90 180 80 90 70 140 50 100 90 ...
##  $ income: int  130 150 230 100 100 70 190 100 260 110 ...
##  $ age   : int  25 39 47 33 31 24 46 25 30 41 ...
##  $ nk    : int  2 2 2 2 1 1 1 1 1 1 ...
```

```
head(london)# show first 6 row
```

```
##   X...  wfood  wfuel wcloth   walc wtrans wother totexp income age nk
## 1    1 0.4272 0.1342 0.0000 0.0106 0.1458 0.2822     50    130  25  2
## 2    2 0.3739 0.1686 0.0091 0.0825 0.1215 0.2444     90    150  39  2
## 3    3 0.1941 0.4056 0.0012 0.0513 0.2063 0.1415    180    230  47  2
## 4    4 0.4438 0.1258 0.0539 0.0397 0.0652 0.2716     80    100  33  2
## 5    5 0.3331 0.0824 0.0399 0.1571 0.2403 0.1473     90    100  31  1
## 6    6 0.3752 0.0481 0.1170 0.0210 0.0955 0.3431     70     70  24  1
```

```
colnames(london)# show the label or colmn's name, also called the variables' name
```

```
##  [1] "X..."   "wfood"  "wfuel"  "wcloth" "walc"   "wtrans" "wother"
##  [8] "totexp" "income" "age"    "nk"
```

```
london$wcloth# show the wcloth variable in the london dataset.
```

```
##    [1] 0.0000 0.0091 0.0012 0.0539 0.0399 0.1170 0.0453 0.1131 0.1671
##   [10]     NA 0.2547 0.1617 0.0018 0.0728 0.2010 0.0501 0.0722 0.0727
##   [19] 0.1511 0.0454 0.0931 0.0246 0.0506 0.1451 0.0054 0.0180 0.2388
##   [28] 0.0392 0.0788 0.3875 0.0077 0.0085 0.1310 0.1428 0.0458 0.2943
##   [37] 0.0662 0.1530 0.2702 0.1808 0.1046 0.0033 0.0058 0.0419 0.0662
##   [46] 0.0542 0.1605 0.1109 0.0000 0.0796 0.1156 0.1846 0.2889 0.1610
##   [55] 0.0928 0.2845 0.1902 0.1102 0.0517 0.1495 0.1060 0.1012 0.0254
##   [64] 0.2637 0.0155 0.0049 0.1491 0.0834 0.0098 0.1980 0.0147 0.1582
##   [73] 0.0596 0.0419 0.1914 0.0200 0.0927 0.0217 0.2294 0.0017 0.2278
##   [82] 0.1393 0.0144 0.0451 0.0000 0.0168 0.0306 0.0038 0.2158 0.1370
##   [91] 0.0717 0.0834 0.0446 0.0000 0.1680 0.0000 0.0856 0.0674 0.3036
##  [100] 0.0985 0.1172 0.0030 0.0184 0.0456 0.0693 0.1423 0.0213 0.0363
##  [109] 0.1034 0.0755 0.0000 0.1856 0.0049 0.2395 0.2728 0.0613 0.0863
##  [118] 0.1169 0.3933 0.0453 0.2002 0.1613 0.0616 0.0830 0.0623 0.0054
##  [127] 0.1744 0.1345 0.0493 0.0579 0.0545 0.2106 0.1114 0.1848 0.1940
##  [136] 0.0842 0.1344 0.1353 0.1261 0.0825 0.1731 0.1140 0.0985 0.0447
##  [145] 0.0167 0.1440 0.2382 0.0126 0.1297 0.3082 0.0000 0.0452 0.0004
##  [154] 0.0134 0.1747 0.0319 0.1465 0.0317 0.0110 0.0000 0.0981 0.0872
##  [163] 0.1310 0.1583 0.0371 0.1875 0.0645 0.1037 0.0639 0.1647 0.0952
##  [172] 0.1739 0.0714 0.0717 0.2046 0.0499 0.0260 0.0970 0.0599 0.0806
##  [181] 0.0861 0.1077 0.1823 0.1086 0.0883 0.0510 0.1519 0.0398 0.1463
##  [190] 0.0000 0.0155 0.0780 0.1368 0.2587 0.0030 0.1253 0.3148 0.0033
##  [199] 0.0252 0.0509 0.0621 0.0109 0.0205 0.0932 0.0738 0.1237 0.0000
##  [208] 0.0245 0.0136 0.0517 0.0173 0.2086 0.0991 0.0264 0.1863 0.0129
##  [217] 0.0247 0.0751 0.1545 0.0447 0.0570 0.0233 0.2344 0.2543 0.0054
##  [226] 0.2590 0.0888 0.0076 0.0000 0.1498 0.1533 0.0369 0.1174 0.0935
##  [235] 0.0128 0.0000 0.1243 0.0673 0.2385 0.0000 0.0090 0.0612 0.0395
##  [244] 0.1426 0.1055 0.1012 0.2756 0.0550 0.0542 0.1584 0.1196 0.1759
##  [253] 0.0000 0.0483 0.1092 0.0317 0.0253 0.1200 0.0497 0.0030 0.2278
##  [262] 0.1715 0.0787 0.0000 0.0805 0.1438 0.0992 0.1458 0.0451 0.0791
##  [271] 0.0769 0.0000 0.0861 0.0032 0.1735 0.0907 0.0400 0.1325 0.0332
##  [280] 0.0839 0.1406 0.2199 0.1409 0.0365 0.0499 0.2540 0.0054 0.0459
##  [289] 0.2309 0.0039 0.0558 0.0306 0.0241 0.0074 0.0247 0.0255 0.0739
##  [298] 0.0278 0.1066 0.0036 0.2119 0.1293 0.1089 0.0316 0.0332 0.0735
##  [307] 0.1710 0.0957 0.2256 0.1929 0.0310 0.0058 0.0990 0.0215 0.0670
##  [316] 0.0185 0.0080 0.1803 0.0890 0.0624 0.0039 0.0931 0.2183 0.0143
##  [325] 0.0345 0.0671 0.3472 0.1578 0.2345 0.1626 0.0958 0.2317 0.0051
##  [334] 0.0097 0.1050 0.0076 0.0000 0.0503 0.1922 0.1527 0.0946 0.3358
##  [343] 0.0410 0.0501 0.3022 0.2068 0.2739 0.1492 0.1270 0.1206 0.0407
##  [352] 0.0987 0.0980 0.2278 0.4057 0.0283 0.2279 0.0022 0.0021 0.1774
##  [361] 0.0897 0.1733 0.0281 0.0785 0.0264 0.0816 0.1732 0.2212 0.1285
##  [370] 0.0000 0.1788 0.3340 0.0401 0.0196 0.1217 0.1516 0.0105 0.0556
##  [379] 0.1451 0.0482 0.0395 0.0693 0.0069 0.0856 0.1761 0.1004 0.2709
##  [388] 0.1397 0.0804 0.1061 0.2733 0.2905 0.2035 0.1251 0.1241 0.0981
##  [397] 0.0085 0.1777 0.1640 0.0000 0.1037 0.0275 0.0308 0.0495 0.0879
##  [406] 0.0436 0.2345 0.2118 0.1671 0.0430 0.0000 0.0094 0.0899 0.1994
##  [415] 0.0878 0.0100 0.0035 0.0320 0.1326 0.0131 0.0744 0.0705 0.1424
##  [424] 0.2661 0.1269 0.0039 0.1201 0.0108 0.0389 0.0892 0.0330 0.0295
##  [433] 0.0517 0.0498 0.0542 0.0690 0.3070 0.1726 0.1367 0.1939 0.1939
##  [442] 0.0145 0.3252 0.0241 0.0550 0.0287 0.0651 0.0393 0.1118 0.0328
##  [451] 0.1370 0.2926 0.1177 0.0234 0.0660 0.2948 0.1543 0.3283 0.0000
```

```
##  [460] 0.0544 0.0075 0.0000 0.3010 0.0116 0.0114 0.4293 0.3883 0.0553
##  [469] 0.0217 0.0527 0.1686 0.0272 0.2200 0.1388 0.1021 0.2824 0.0189
##  [478] 0.1566 0.1158 0.0946 0.1186 0.0605 0.0102 0.0345 0.0731 0.2002
##  [487] 0.0853 0.2052 0.1642 0.0418 0.0051 0.0617 0.0052 0.0117 0.1409
##  [496] 0.0355 0.2354 0.0120 0.0202 0.0727 0.0219 0.0069 0.1284 0.0040
##  [505] 0.0017 0.3018 0.0645 0.0681 0.1285 0.0886 0.2214 0.0338 0.1649
##  [514] 0.1013 0.0494 0.2073 0.0993 0.4169 0.4340 0.0000 0.1194 0.1113
##  [523] 0.1014 0.0583 0.1235 0.3414 0.0052 0.1332 0.0092 0.0144 0.1371
##  [532] 0.0157 0.0631 0.1298 0.0482 0.1672 0.2472 0.0000 0.0857 0.1746
##  [541] 0.0881 0.0314 0.0472 0.0593 0.1313 0.0042 0.0000 0.2277 0.0021
##  [550] 0.0023 0.0696 0.3152 0.1185 0.1557 0.0531 0.2338 0.1215 0.0536
##  [559] 0.2163 0.1253 0.1033 0.0277 0.1105 0.0321 0.3703 0.0693 0.1924
##  [568] 0.0048 0.2136 0.0000 0.0134 0.3270 0.0079 0.0906 0.0137 0.0000
##  [577] 0.1324 0.2193 0.1358 0.0574 0.1613 0.0128 0.1385 0.0171 0.3181
##  [586] 0.0593 0.0000 0.1879 0.2261 0.0339 0.0659 0.0881 0.0297 0.0463
##  [595] 0.2694 0.1568 0.0395 0.0937 0.1088 0.0413 0.0539 0.0221 0.1333
##  [604] 0.1506 0.1911 0.0028 0.0798 0.0461 0.0135 0.0102 0.0457 0.0054
##  [613] 0.0990 0.0133 0.0881 0.1171 0.0183 0.1291 0.0412 0.0200 0.0028
##  [622] 0.0314 0.2163 0.1608 0.0722 0.0584 0.0582 0.0956 0.0233 0.1609
##  [631] 0.0942 0.2269 0.1650 0.1910 0.0494 0.0880 0.0000 0.1344 0.1025
##  [640] 0.0000 0.0362 0.0148 0.2505 0.0114 0.0888 0.0974 0.0769 0.2404
##  [649] 0.1119 0.2758 0.2433 0.1310 0.1184 0.0000 0.1461 0.0666 0.1734
##  [658] 0.0098 0.0755 0.0972 0.0000 0.4282 0.0721 0.0658 0.1143 0.0860
##  [667] 0.2098 0.1157 0.1302 0.0855 0.0451 0.2350 0.1038 0.2097 0.0000
##  [676] 0.0842 0.0858 0.2600 0.0867 0.0296 0.0572 0.0112 0.0524 0.0676
##  [685] 0.1697 0.1498 0.1566 0.0677 0.0090 0.0142 0.0596 0.1096 0.3897
##  [694] 0.4039 0.2443 0.0616 0.0602 0.1741 0.2704 0.1163 0.0387 0.1991
##  [703] 0.0239 0.0213 0.3842 0.1591 0.0322 0.0579 0.1565 0.0561 0.2043
##  [712] 0.0713 0.0043 0.1927 0.1074 0.0854 0.0861 0.0000 0.0000 0.0691
##  [721] 0.1442 0.1295 0.0000 0.0086 0.1283 0.0231 0.0172 0.0267 0.0623
##  [730] 0.1982 0.0851 0.1302 0.0353 0.1034 0.1111 0.0678 0.0197 0.0337
##  [739] 0.0048 0.1665 0.0416 0.0656 0.1603 0.0698 0.0200 0.0907 0.0935
##  [748] 0.0947 0.0988 0.1147 0.3009 0.1472 0.3911 0.1041 0.0277 0.1816
##  [757] 0.1376 0.0396 0.0288 0.0742 0.1270 0.0000 0.0619 0.0832 0.1752
##  [766] 0.0072 0.0000 0.2805 0.1672 0.0081 0.1357 0.0348 0.1628 0.3721
##  [775] 0.0625 0.1381 0.1706 0.0000 0.0403 0.1397 0.0985 0.3093 0.2034
##  [784] 0.0689 0.1126 0.2445 0.0195 0.0237 0.0440 0.2673 0.0083 0.0570
##  [793] 0.0000 0.0546 0.0110 0.2702 0.3519 0.1101 0.0079 0.1070 0.0944
##  [802] 0.0324 0.1894 0.2239 0.1534 0.0995 0.0888 0.1290 0.1075 0.1035
##  [811] 0.0878 0.5035 0.0812 0.0444 0.0000 0.0123 0.2948 0.0691 0.0572
##  [820] 0.0210 0.0625 0.1818 0.1027 0.1743 0.0275 0.1977 0.0677 0.0420
##  [829] 0.3106 0.0000 0.1211 0.1125 0.1100 0.0036 0.0623 0.4381 0.0248
##  [838] 0.0477 0.0668 0.0000 0.0536 0.0210 0.0396 0.0493 0.3620 0.2331
##  [847] 0.1566 0.0090 0.1053 0.1557 0.0791 0.1404 0.2086 0.0410 0.1234
##  [856] 0.1163 0.2170 0.1714 0.1611 0.0781 0.0783 0.0000 0.2178 0.1690
##  [865] 0.0141 0.1072 0.0671 0.1516 0.3622 0.2153 0.0038 0.0845 0.0040
##  [874] 0.0000 0.0573 0.1859 0.1162 0.0000 0.0381 0.0642 0.2645 0.1988
##  [883] 0.2198 0.3612 0.0240 0.0978 0.0996 0.1653 0.0117 0.0164 0.1303
##  [892] 0.0094 0.0520 0.0251 0.2327 0.0824 0.2497 0.0000 0.1905 0.1797
##  [901] 0.0646 0.1574 0.0267 0.1614 0.0080 0.0991 0.0853 0.1320 0.0019
##  [910] 0.0503 0.0386 0.0062 0.1685 0.2202 0.0019 0.0408 0.0306 0.1546
##  [919] 0.0285 0.1199 0.0000 0.1388 0.0000 0.1261 0.3058 0.2840 0.0649
##  [928] 0.1306 0.0000 0.1833 0.0193 0.1555 0.0243 0.0461 0.1784 0.1752
##  [937] 0.1707 0.2214 0.0259 0.1913 0.0759 0.2610 0.2882 0.1820 0.1289
```

```
##  [946] 0.0749 0.0550 0.0574 0.0502 0.1183 0.1753 0.1099 0.0592 0.1557
##  [955] 0.1664 0.1760 0.1892 0.1523 0.0799 0.0212 0.0955 0.0728 0.0822
##  [964] 0.0204 0.1180 0.0078 0.0552 0.2315 0.2306 0.4280 0.0345 0.0517
##  [973] 0.1254 0.0164 0.3321 0.0764 0.0604 0.1145 0.1076 0.2357 0.0000
##  [982] 0.0743 0.1091 0.2973 0.0464 0.1207 0.0378 0.0692 0.0000 0.0070
##  [991] 0.0884 0.2602 0.1670 0.0789 0.2640 0.0740 0.1178 0.1485 0.1105
## [1000] 0.0495 0.1431 0.0192 0.0281 0.0449 0.0057 0.1692 0.0000 0.0266
## [1009] 0.0538 0.1888 0.0079 0.3188 0.7602 0.0937 0.0261 0.0119 0.0501
## [1018] 0.0032 0.0588 0.0416 0.1294 0.1868 0.1612 0.2364 0.0000 0.0517
## [1027] 0.3128 0.3284 0.1852 0.0982 0.2266 0.1059 0.0804 0.1844 0.2679
## [1036] 0.1386 0.0411 0.0120 0.1118 0.1950 0.0661 0.0354 0.0145 0.2622
## [1045] 0.1661 0.0735 0.0220 0.1125 0.1985 0.0000 0.0563 0.1319 0.0000
## [1054] 0.0335 0.0673 0.2093 0.0285 0.0484 0.0225 0.1932 0.0411 0.1723
## [1063] 0.3913 0.1613 0.0259 0.1851 0.0470 0.1259 0.0453 0.0088 0.3873
## [1072] 0.2734 0.1921 0.0595 0.1781 0.2069 0.1684 0.0764 0.0054 0.0148
## [1081] 0.0788 0.0802 0.1065 0.0508 0.0848 0.1533 0.2415 0.2660 0.0099
## [1090] 0.1241 0.4789 0.0808 0.0945 0.0361 0.0705 0.0501 0.1160 0.0431
## [1099] 0.1342 0.1683 0.0730 0.0174 0.1268 0.0000 0.0112 0.1245 0.1679
## [1108] 0.0743 0.0067 0.3573 0.0000 0.1575 0.1123 0.0620 0.0439 0.2444
## [1117] 0.2151 0.1262 0.2311 0.2090 0.0549 0.0108 0.2635 0.0412 0.0921
## [1126] 0.1630 0.1324 0.1125 0.1047 0.0485 0.0063 0.0073 0.0000 0.0953
## [1135] 0.0488 0.1418 0.0000 0.0244 0.1109 0.1532 0.0600 0.0440 0.0810
## [1144] 0.0000 0.1512 0.0204 0.0691 0.2310 0.0143 0.0000 0.1356 0.0708
## [1153] 0.2209 0.3643 0.0405 0.2610 0.3136 0.0322 0.0000 0.2243 0.1164
## [1162] 0.3520 0.0763 0.1290 0.0044 0.1303 0.1323 0.0000 0.1454 0.3624
## [1171] 0.1720 0.1033 0.0711 0.0000 0.0000 0.3622 0.0663 0.1479 0.0806
## [1180] 0.0000 0.1547 0.0876 0.1518 0.1438 0.0466 0.0342 0.6155 0.1900
## [1189] 0.0106 0.2177 0.1740 0.0484 0.1871 0.1122 0.0677 0.0653 0.0801
## [1198] 0.0779 0.0184 0.1719 0.0000 0.1491 0.3208 0.0460 0.0000 0.3157
## [1207] 0.0000 0.0344 0.0057 0.0000 0.0362 0.0763 0.1267 0.0695 0.0000
## [1216] 0.1473 0.1923 0.2847 0.0745 0.1615 0.1738 0.2082 0.1155 0.1863
## [1225] 0.0170 0.0000 0.1450 0.1360 0.1345 0.0678 0.1197 0.1736 0.2642
## [1234] 0.0531 0.3034 0.1731 0.1145 0.0122 0.1586 0.1803 0.0738 0.1100
## [1243] 0.0613 0.4454 0.0354 0.1184 0.2160 0.0029 0.1445 0.1153 0.0000
## [1252] 0.1706 0.1050 0.3499 0.0112 0.0578 0.0512 0.0721 0.3145 0.0612
## [1261] 0.2210 0.0000 0.0358 0.2305 0.1592 0.0786 0.0755 0.1184 0.2754
## [1270] 0.0748 0.1196 0.0800 0.1648 0.0200 0.0000 0.0766 0.0363 0.2508
## [1279] 0.3349 0.1539 0.0879 0.3341 0.0062 0.0241 0.1050 0.0577 0.0807
## [1288] 0.0000 0.2620 0.1258 0.0916 0.0625 0.0987 0.2052 0.0555 0.0000
## [1297] 0.0051 0.1943 0.1136 0.1246 0.0719 0.1495 0.3718 0.1912 0.0000
## [1306] 0.1827 0.1989 0.3034 0.0738 0.0250 0.0337 0.0250 0.1438 0.0221
## [1315] 0.0012 0.0022 0.0000 0.1171 0.0685 0.0000 0.0100 0.1035 0.0000
## [1324] 0.3053 0.1827 0.1154 0.1969 0.0946 0.1306 0.1006 0.0752 0.0682
## [1333] 0.0062 0.0000 0.0046 0.0526 0.0378 0.1534 0.0438 0.1780 0.2310
## [1342] 0.0169 0.0658 0.1863 0.0856 0.1431 0.0271 0.0498 0.0682 0.1253
## [1351] 0.1556 0.1132 0.1179 0.3404 0.0000 0.0659 0.0572 0.1118 0.1492
## [1360] 0.0164 0.0936 0.0000 0.2092 0.0265 0.1368 0.0247 0.0000 0.1978
## [1369] 0.1244 0.1198 0.0350 0.0000 0.0767 0.1265 0.0527 0.2224 0.0045
## [1378] 0.0768 0.0035 0.1941 0.1878 0.0837 0.2087 0.1523 0.0026 0.2892
## [1387] 0.0038 0.0713 0.0094 0.0656 0.1082 0.1262 0.1319 0.1906 0.2457
## [1396] 0.0000 0.1707 0.1759 0.1350 0.0862 0.0419 0.1765 0.0925 0.4824
## [1405] 0.0034 0.1255 0.0044 0.1952 0.0538 0.0910 0.2430 0.0455 0.1410
## [1414] 0.1804 0.0714 0.1858 0.0289 0.1689 0.1712 0.0000 0.0576 0.2003
## [1423] 0.0396 0.1115 0.0167 0.1045 0.0341 0.0000 0.1231 0.0506 0.1288
```

```
## [1432] 0.3873 0.2335 0.0719 0.2019 0.3601 0.1015 0.0655 0.1856 0.0000
## [1441] 0.0107 0.2918 0.0990 0.0972 0.1368 0.0315 0.1410 0.3747 0.0250
## [1450] 0.0671 0.2153 0.0206 0.0196 0.0320 0.0745 0.1465 0.1965 0.1284
## [1459] 0.1329 0.0000 0.0764 0.0028 0.0105 0.2763 0.0000 0.1138 0.0109
## [1468] 0.0247 0.0334 0.1162 0.0292 0.0782 0.1205 0.0122 0.1349 0.0258
## [1477] 0.0201 0.2029 0.0811 0.1291 0.0492 0.0248 0.2538 0.0347 0.0000
## [1486] 0.1072 0.1481 0.0233 0.1730 0.3352 0.1821 0.0310 0.1103 0.1106
## [1495] 0.1131 0.1292 0.2737 0.0000 0.0346 0.1544 0.0674 0.0430 0.0119
## [1504] 0.0726 0.0114 0.3681 0.1715 0.0315 0.0089 0.0402 0.1740 0.0899
## [1513] 0.2391 0.0000 0.0054 0.0176 0.2566 0.0787 0.1279
```

```
london[,1]=as.character(london[,1])# reset the 1st column as.character
```

1.5.5 Some Statistical Functions

Some useful Statistical Functions

```
sum(london$income)#sum of the income of all obervations in london dataset
```

```
## [1] 206960
```

```
mean(london$income)
```

```
## [1] 136.2475
```

```
var(london$income)
```

```
## [1] 3728.466
```

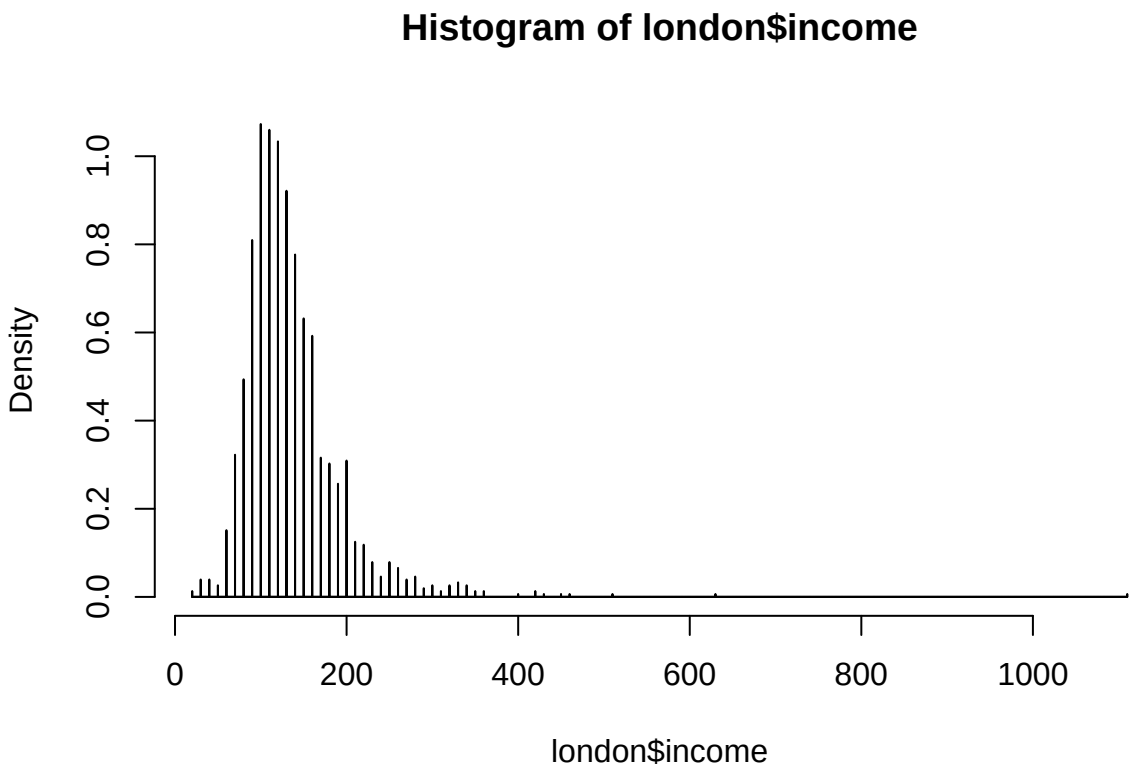
```
sd(london$income)
```

```
## [1] 61.06116
```

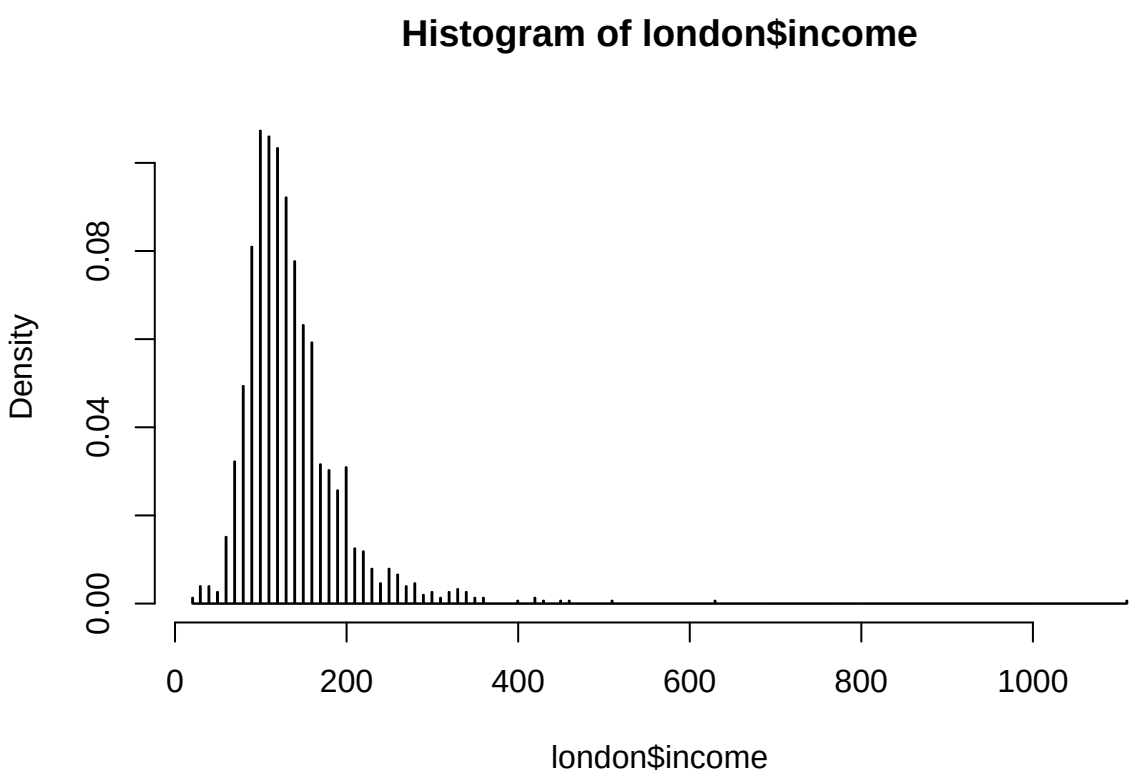
```
quantile(london$income)
```

```
##   0%  25%  50%  75% 100%
##   20  100  120  160 1110
```

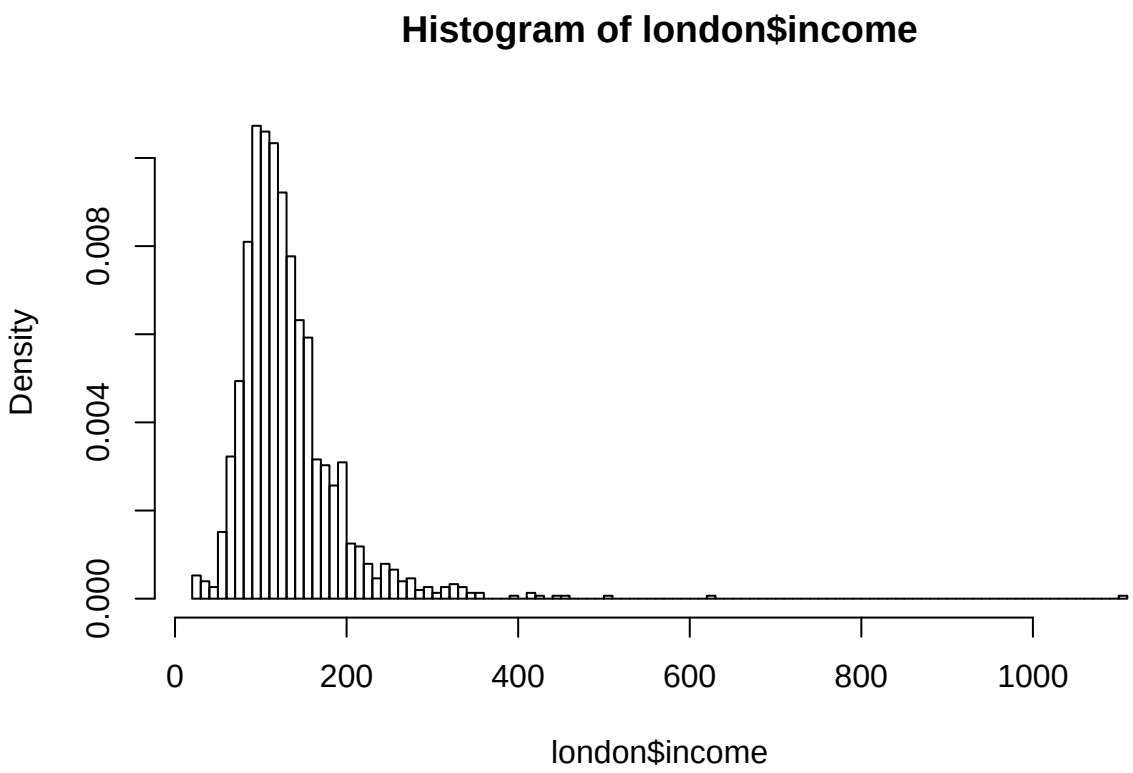
```
hist(london$income,breaks=10000,freq = FALSE)# draw a histogram based on london$income, the breaks is t
```



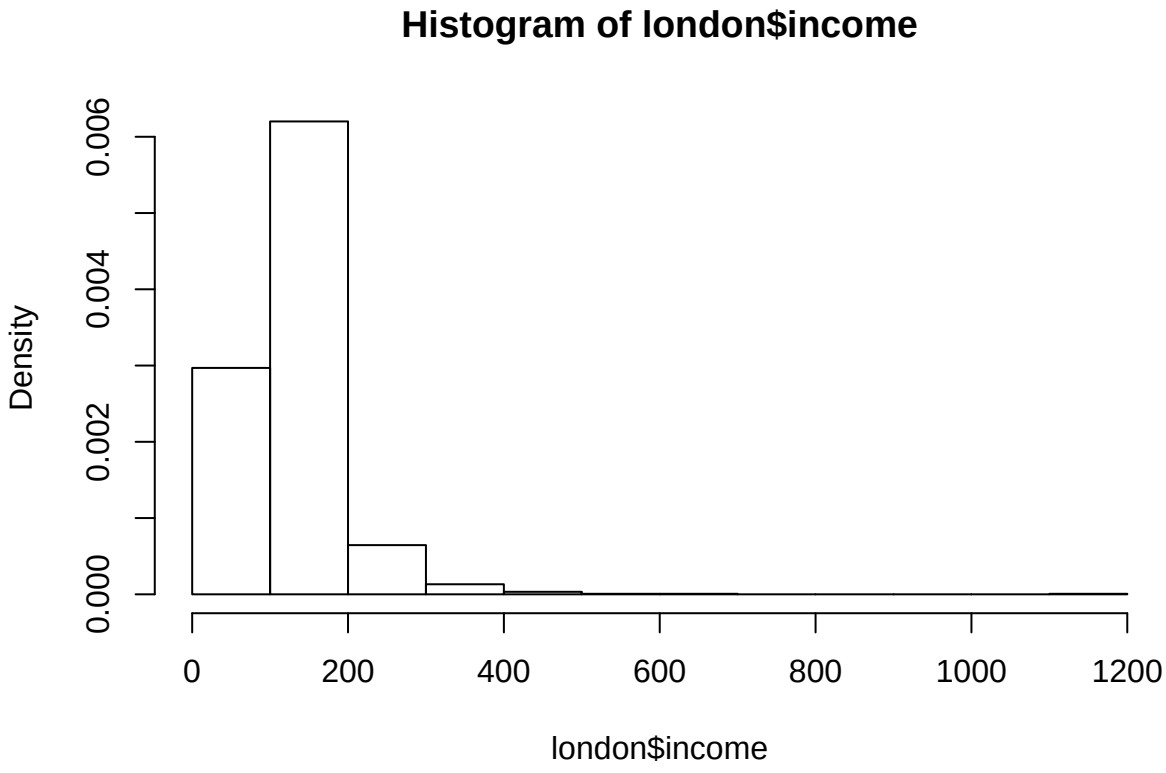

```
hist(london$income,breaks=1000,freq = FALSE)
```



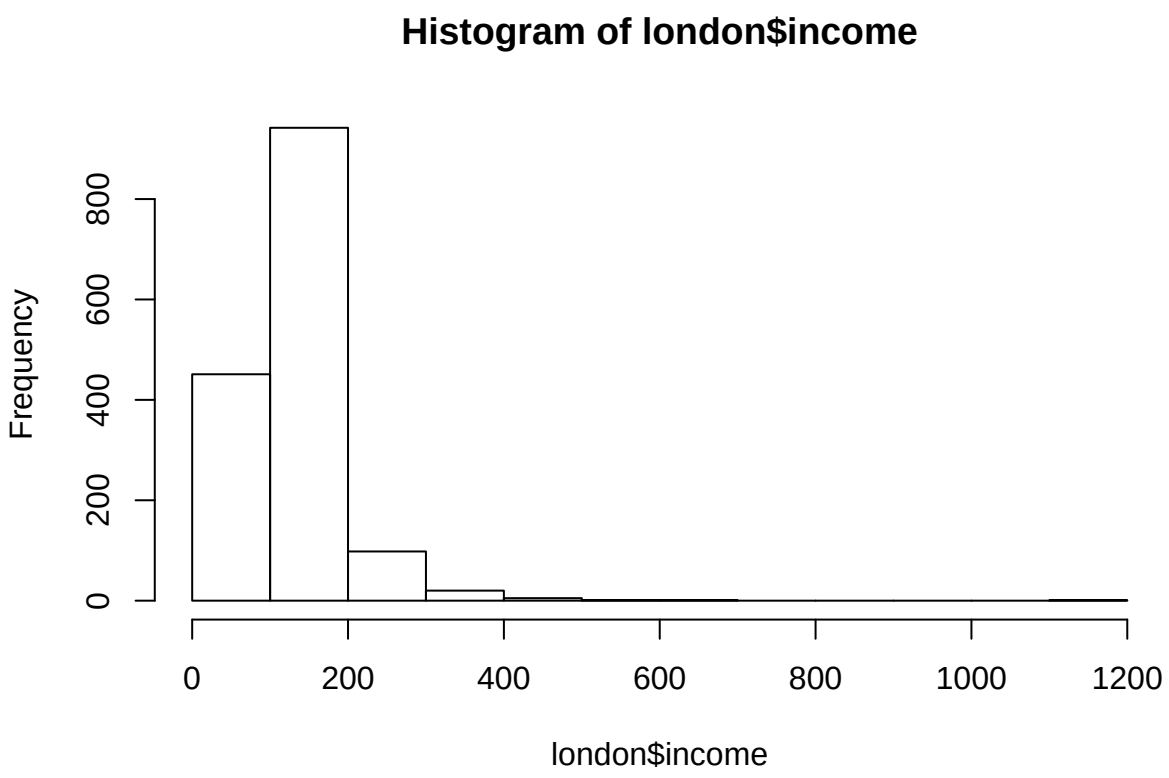
```
hist(london$income,breaks=100,freq = FALSE)
```



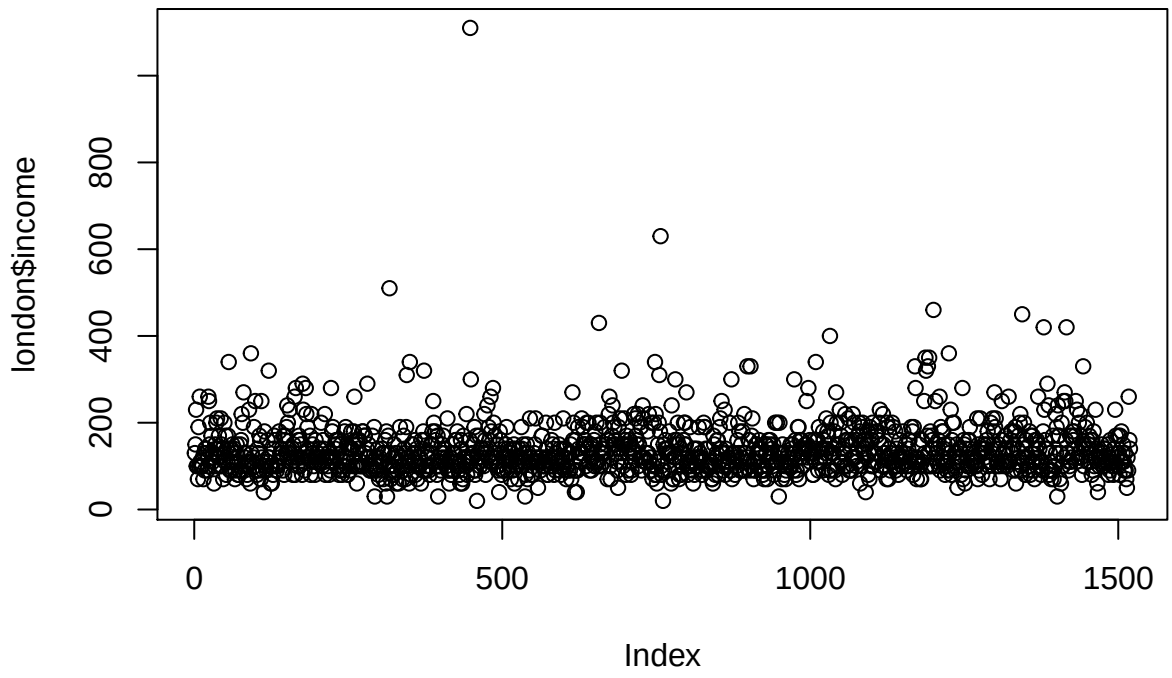

```
hist(london$income,breaks=10,freq = FALSE)
```



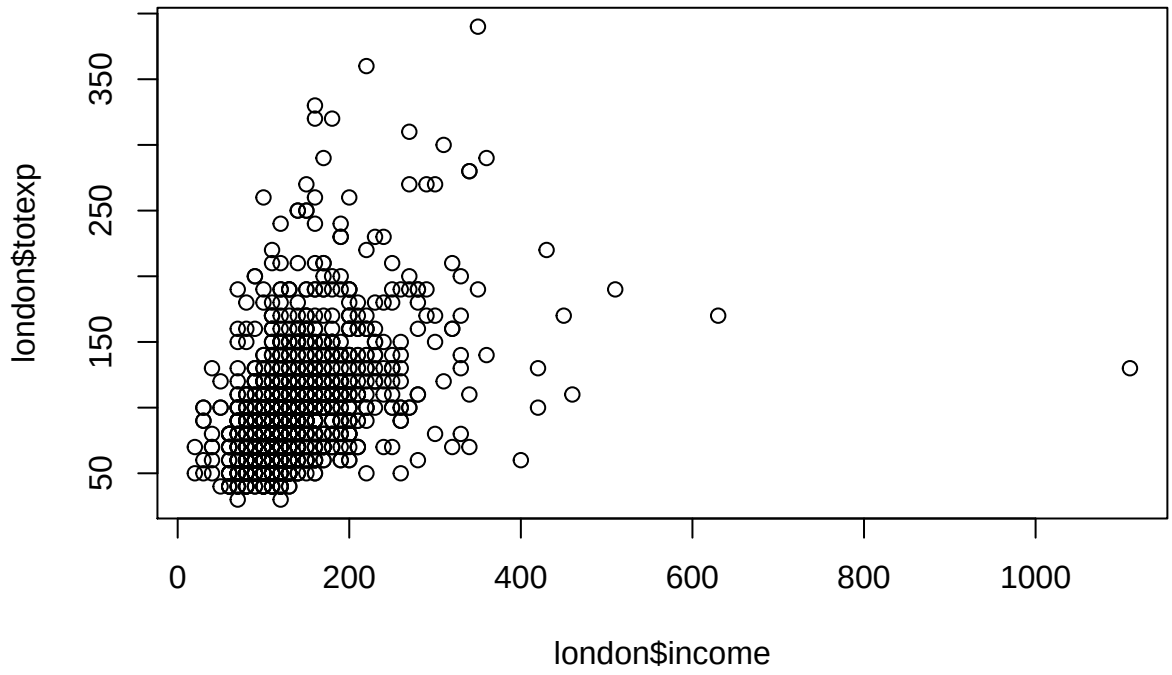
```
hist(london$income,breaks=10,freq = TRUE)
```



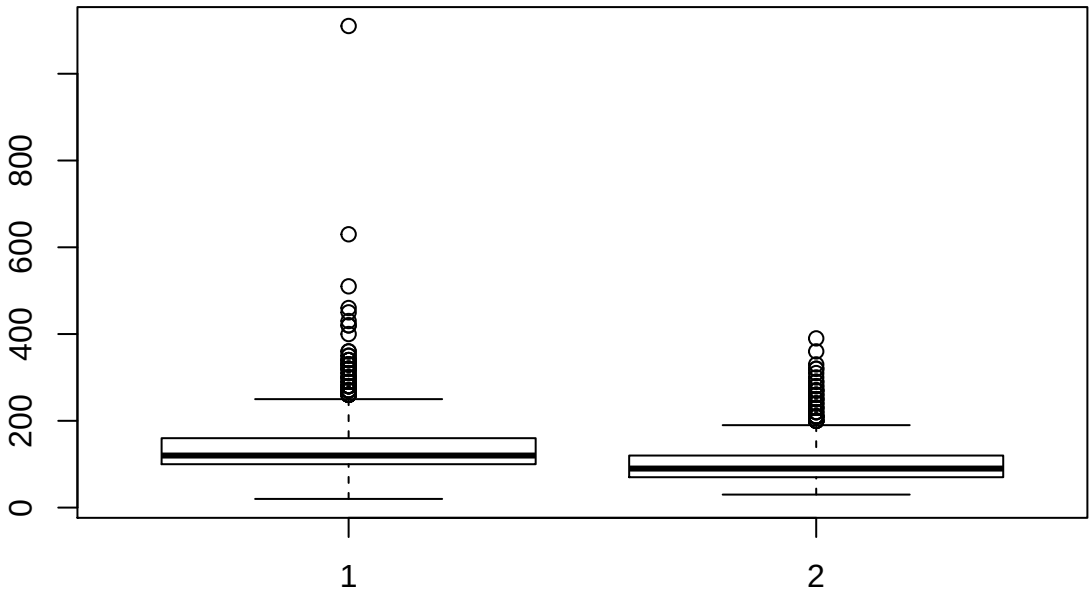

```
plot(london$income)
```



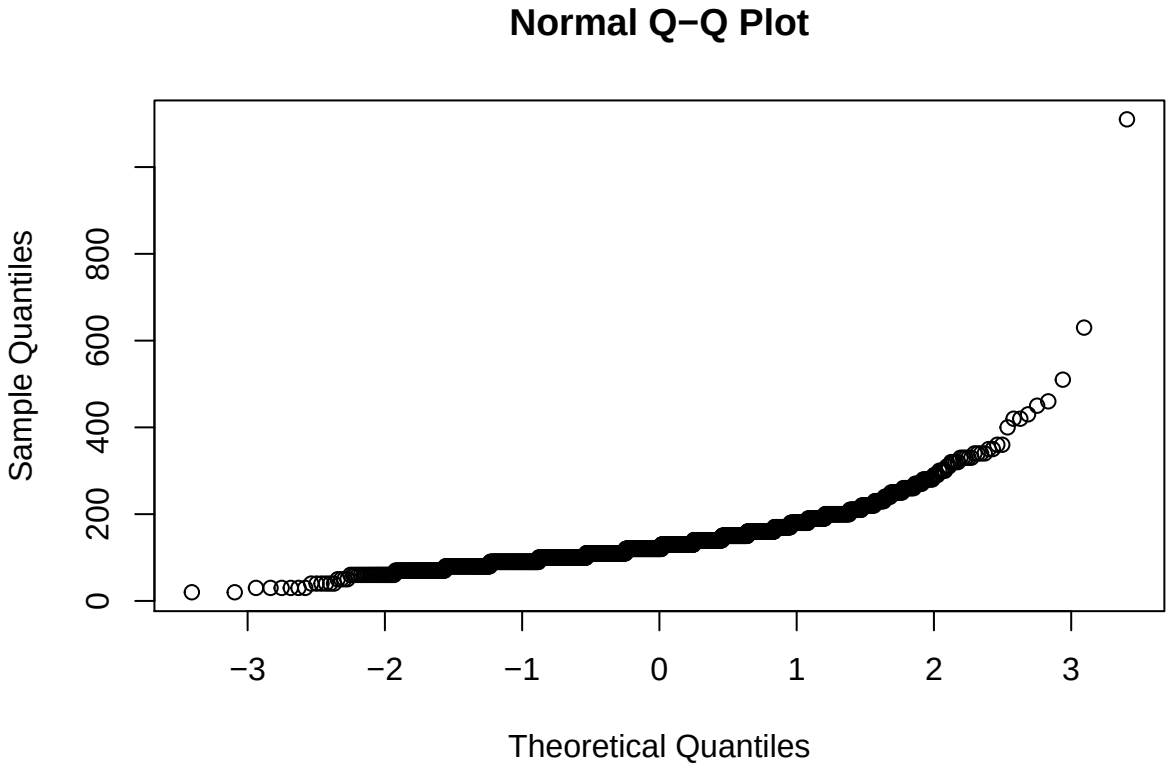

```
plot(london$income,london$totexp)
```




```
boxplot(london$income,london$totexp)#draw a boxplot
```




```
qqnorm(london$income)#plot the qq-plot
```




```
which(london$income>140)#return the indices for observations with income higher than 140.
```

```
##   [1]    2    3    7    9   23   24   25   26   28   34   36   37   40   43
##  [15]   44   49   51   55   56   62   69   75   76   77   79   80   89   92
##  [29]   97   99  106  107  109  114  119  121  130  131  138  145  147  148
##  [43]  150  151  152  155  161  163  165  169  170  172  173  176  177  181
##  [57]  182  184  188  190  192  197  206  209  212  222  224  225  228  230
##  [71]  233  234  238  244  246  248  255  258  259  260  262  265  268  273
##  [85]  274  276  277  281  283  289  292  310  312  315  317  325  334  340
##  [99]  344  345  350  353  355  358  369  372  373  374  384  386  387  388
## [113]  389  391  392  395  402  404  411  419  424  425  427  433  437  442
## [127]  447  448  449  453  455  458  463  471  472  476  478  479  481  482
## [141]  485  486  488  492  496  497  506  507  519  531  534  540  545  553
## [155]  554  564  565  571  580  582  585  592  599  611  614  615  616  619
## [169]  623  629  631  632  633  634  635  637  640  643  644  649  651  653
## [183]  657  658  659  666  668  669  670  672  673  674  678  679  680  681
## [197]  682  683  684  689  691  694  695  697  698  699  700  702  706  707
## [211]  708  714  716  717  718  719  720  722  723  725  728  729  736  738
## [225]  742  746  748  749  753  754  755  756  757  762  765  771  775  780
## [239]  781  783  784  785  786  794  795  796  797  798  799  805  819  827
## [253]  828  829  843  844  849  850  853  854  856  861  869  872  882  883
## [267]  885  888  890  893  898  900  901  903  905  906  912  913  917  927
## [281]  928  932  934  935  940  943  946  950  955  968  971  974  976  980
## [295]  981  983  986  992  994  995  997  999 1006 1009 1011 1012 1014 1021
## [309] 1022 1026 1028 1029 1032 1033 1039 1042 1043 1045 1046 1048 1049 1050
## [323] 1057 1058 1061 1062 1064 1066 1069 1071 1074 1077 1079 1080 1083 1084
## [337] 1085 1086 1089 1092 1093 1097 1098 1100 1101 1107 1108 1110 1111 1112
## [351] 1113 1115 1118 1119 1125 1126 1133 1135 1138 1145 1151 1155 1160 1161
## [365] 1164 1166 1168 1170 1171 1176 1181 1185 1187 1188 1190 1191 1193 1195
```

```
## [379] 1197 1198 1200 1203 1204 1208 1210 1212 1214 1216 1218 1224 1225 1228
## [393] 1230 1231 1233 1238 1247 1249 1250 1256 1258 1259 1271 1273 1276 1282
## [407] 1286 1287 1288 1293 1296 1297 1298 1299 1300 1301 1306 1311 1312 1317
## [421] 1322 1325 1326 1327 1331 1332 1333 1338 1340 1341 1344 1346 1347 1350
## [435] 1356 1358 1361 1365 1368 1369 1370 1378 1379 1380 1381 1385 1387 1394
## [449] 1398 1400 1402 1403 1404 1408 1411 1413 1414 1415 1416 1422 1424 1426
## [463] 1427 1428 1430 1431 1433 1436 1438 1442 1443 1445 1446 1449 1450 1452
## [477] 1453 1456 1458 1460 1463 1473 1483 1487 1495 1496 1500 1503 1505 1508
## [491] 1517 1518
```

```
cov(london$income,london$totexp)# show the sample covariance between income and total expenditure.
```

```
## [1] 1183.511
```

```
cor(london[,2:11])#sample correlation matrix for the 2-11 variables(columns) in london
```

```
##              wfood       wfuel wcloth walc       wtrans       wother
## wfood   1.00000000  0.10156032     NA   NA -0.333761601 -0.353580774
## wfuel   0.10156032  1.00000000     NA   NA -0.160910408 -0.132764775
## wcloth          NA          NA      1   NA           NA           NA
## walc            NA          NA     NA    1           NA           NA
## wtrans -0.33376160 -0.16091041     NA   NA  1.000000000 -0.295970628
## wother -0.35358077 -0.13276478     NA   NA -0.295970628  1.000000000
## totexp -0.47874726 -0.31937891     NA   NA  0.148271628  0.157614011
## income -0.23466674 -0.02870070     NA   NA  0.008442807  0.153370720
## age     0.02140373 -0.03951842     NA   NA  0.026928761  0.026064288
## nk      0.10178633 -0.02723013     NA   NA -0.043608816 -0.005305898
##             totexp       income          age           nk
## wfood  -0.47874726 -0.234666736  0.021403728  0.101786327
## wfuel  -0.31937891 -0.028700703 -0.039518424 -0.027230131
## wcloth          NA           NA           NA           NA
## walc            NA           NA           NA           NA
## wtrans  0.14827163  0.008442807  0.026928761 -0.043608816
## wother  0.15761401  0.153370720  0.026064288 -0.005305898
## totexp  1.00000000  0.448740348  0.189450718  0.071415282
## income  0.44874035  1.000000000  0.218494045  0.025439105
## age     0.18945072  0.218494045  1.000000000  0.008092095
## nk      0.07141528  0.025439105  0.008092095  1.000000000
```

```
summary(london[,2:11])#summary some statistics for the 2-11 variables(columns) in london
```

```
##      wfood            wfuel             wcloth             walc
##  Min.   :0.0571   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.2817   1st Qu.:0.05530   1st Qu.:0.03220   1st Qu.:0.01252
##  Median :0.3540   Median :0.08000   Median :0.08575   Median :0.04220
##  Mean   :0.3565   Mean   :0.09101   Mean   :0.10729   Mean   :0.06057
##  3rd Qu.:0.4258   3rd Qu.:0.11365   3rd Qu.:0.15827   3rd Qu.:0.08983
##  Max.   :0.7890   Max.   :0.48030   Max.   :0.76020   Max.   :0.42810
##                                     NA's   :1         NA's   :1
##      wtrans            wother           totexp          income
##  Min.   :0.00000   Min.   :0.0361   Min.   : 30.0   Min.   :  20.0
##  1st Qu.:0.05695   1st Qu.:0.1785   1st Qu.: 70.0   1st Qu.: 100.0
```

```
##  Median :0.11530   Median :0.2397   Median : 90.0   Median : 120.0
##  Mean   :0.13235   Mean   :0.2523   Mean   : 98.7   Mean   : 136.2
##  3rd Qu.:0.17940   3rd Qu.:0.3100   3rd Qu.:120.0   3rd Qu.: 160.0
##  Max.   :0.76380   Max.   :0.8066   Max.   :390.0   Max.   :1110.0
##
##       age              nk
##  Min.   :19.00   Min.   :1.000
##  1st Qu.:30.00   1st Qu.:1.000
##  Median :35.00   Median :2.000
##  Mean   :35.78   Mean   :1.609
##  3rd Qu.:40.00   3rd Qu.:2.000
##  Max.   :60.00   Max.   :2.000
##
```

```
colSums(london[,2:11])# return the column sums of the 2-11 variables in
```

```
##       wfood       wfuel      wcloth        walc      wtrans      wother
##    541.4616    138.2480          NA          NA    201.0404    383.3165
##      totexp      income         age          nk
## 149920.0000 206960.0000  54348.0000   2444.0000
```

```
#rowSums(london[2:11,])# return the row sums of the 2-11 rows in london
```

Question 1: Can you find out which variables have missing obervation(s)? in what positions?

1st: use manual way to seek the missing data

```
#colSums(london)# 2 columns:wcloth and walc
#rowSums(london)# 2 rows: 10 and 15

london$wcloth[10]#NA
```

```
## [1] NA
```

```
london$walc[15]#NA
```

```
## [1] NA
```

```
is.na(london$wcloth[1:20])
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

2rd: use auto way to seek the missing data

```
setwd("/Users/sn0wfree/Dropbox/PhD(1st)/sn0wfree.github.io/BST169_Econometrics/computing/session1-5")#s
london<-read.csv('london.csv')# import data from london.csv

compare_missing_data_function<-function(symbol){
  b<-c()
  for (i in 1:length(symbol)){
```

```
    if(is.na(symbol[i])==TRUE){
      b=append(b,i)
    }
  }
  return(b)
}

seek_missing_data<- function(data){
  missing_data_location<-c()
  missing_col_number<-compare_missing_data_function(colSums(data))
  missing_row_number<-compare_missing_data_function(rowSums(data))
  for(i in missing_col_number){
    for(j in missing_row_number){
      if (is.na(data[j,i])==TRUE){
        missing_data_location<-append(missing_data_location,c(names(data)[i],j))}
      }
  }
  return(missing_data_location)
}

seek_missing_data(london)
```

```
## [1] "wcloth" "10"     "walc"   "15"
```

1.6 Least Squares Regression

1.6.1 OLS

```
lmobj<-lm(wfood~log(totexp)+log(income),data=london)
summary(lmobj)#summary of the fitted model
```

```
##
## Call:
## lm(formula = wfood ~ log(totexp) + log(income), data = london)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33339 -0.06201 -0.00229  0.06075  0.32267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.964691   0.033414  28.871   <2e-16 ***
## log(totexp) -0.133104   0.006924 -19.224   <2e-16 ***
## log(income) -0.001564   0.007142  -0.219    0.827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09139 on 1516 degrees of freedom
## Multiple R-squared:  0.2452, Adjusted R-squared:  0.2442
## F-statistic: 246.2 on 2 and 1516 DF,  p-value: < 2.2e-16
```

```r
typeof(lmobj)# is a list
```

```
## [1] "list"
```

```r
class(lmobj)# is a "lm"
```

```
## [1] "lm"
```

`typeof(lmobj)` and `class(lmobj)` have different return value

```r
coef(lmobj)#estimated model parameters (estimated_beta of OLS)
```

```
##  (Intercept)  log(totexp)  log(income)
##  0.964691151 -0.133104065 -0.001564379
```

```r
head(resid(lmobj)) #estimated residuals M_{x,y}
```

```
##            1            2            3            4            5
## -0.009170321  0.015990337 -0.070880272  0.069578635 -0.025443964
##            6
## -0.017352911
```

```r
head(fitted(lmobj)) #fitted values of the model (P_{x,y})
```

```
##         1         2         3         4         5         6
## 0.4363703 0.3579097 0.2649803 0.3742214 0.3585440 0.3925529
```

```r
deviance(lmobj) #the residual sum of squares (SSR or y'M_{x,y})
```

```
## [1] 12.66322
```

```r
confint(lmobj) #confidence interval
```

```
##                    2.5 %      97.5 %
## (Intercept)  0.89914931  1.03023299
## log(totexp) -0.14668549 -0.11952264
## log(income) -0.01557339  0.01244463
```

```r
logLik(lmobj) #value of the log likelihood function (assume normal er- ror)
```

```
## 'log Lik.' 1480.439 (df=4)
```

```r
AIC(lmobj,k=2) #information criterion, k = 2 for AIC and k = log(N) for BIC (assume normal error, to be
```

```
## [1] -2952.879
```

```r
vcov(lmobj)# estimated-beta OLS's estimated variance-covariance matrix (\espison^{hat}_{2}(X'X)_{-1})
```

```
##                (Intercept)   log(totexp)   log(income)
## (Intercept)   1.116471e-03 -9.885454e-05 -1.373587e-04
## log(totexp)  -9.885454e-05  4.794031e-05 -2.427202e-05
## log(income)  -1.373587e-04 -2.427202e-05  5.100646e-05
```

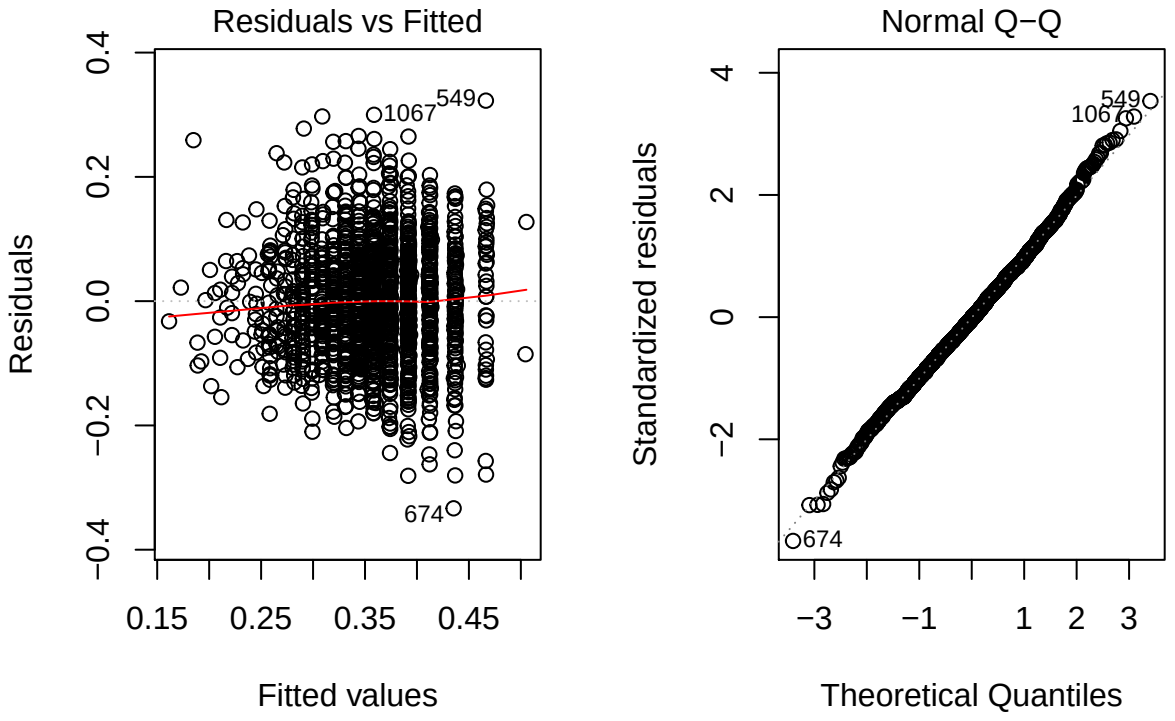
```r
anova(lmobj) #returns an anova table; ANOVA analysis
```

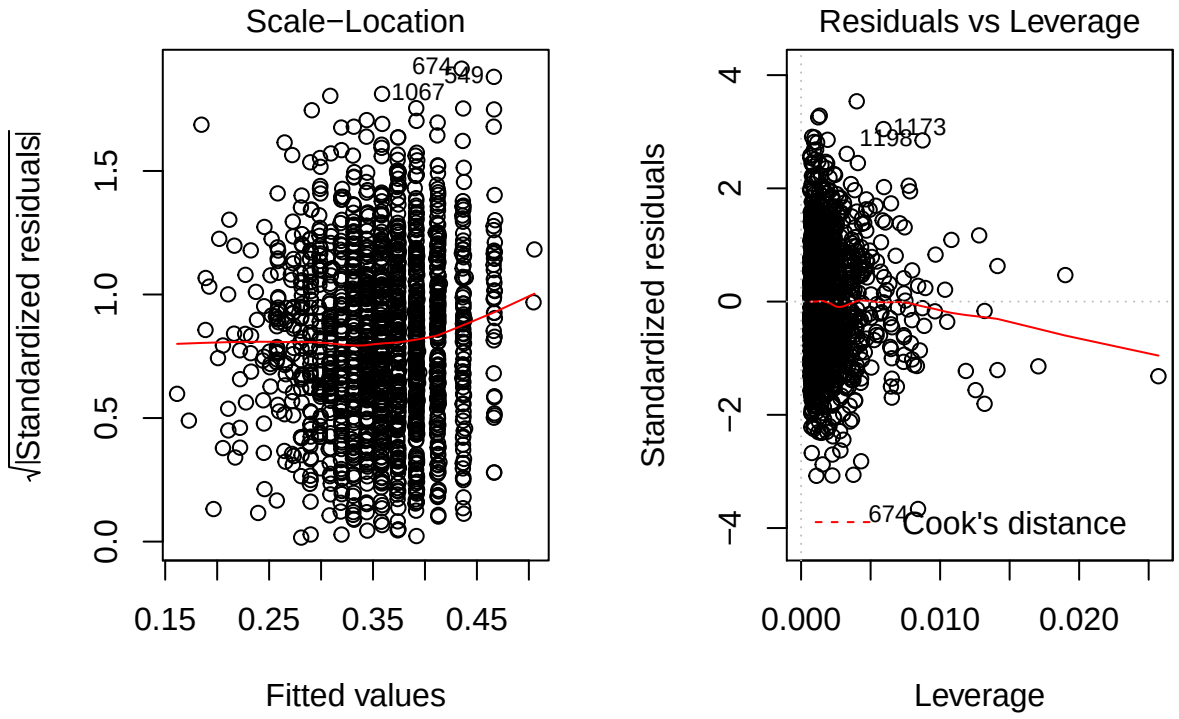
```
## Analysis of Variance Table
##
## Response: wfood
##               Df  Sum Sq Mean Sq F value Pr(>F)
## log(totexp)    1  4.1123  4.1123 492.315 <2e-16 ***
## log(income)    1  0.0004  0.0004   0.048 0.8266
## Residuals   1516 12.6632  0.0084
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
predict(lmobj,data.frame(totexp=1,income=1)) #returns predictions
```

```
##         1
## 0.9646912
```

```r
par(mfrow=c(1,2))#par graphes by 2*2
plot(lmobj) #create diagnostic plots
```






```
par(mfrow=c(1,1))#restore the default one
```

Question 2: Can you use matrix algebra in R to reproduce the outputs of the following functions:

1. `coef(lmobj)`
2. `resid(lmobj)`
3. `fitted(lmobj)`
4. `deviance(lmboj)`
5. `vcov(lmboj)`

Equation:

1). the normal form :

$$wfood = \beta_0 + \beta_1 * totexp + \beta_2 * income + \epsilon$$

2). the matrix form:

$$wfood = x * \beta + \epsilon$$

here,

$$x = |\beta_0 \quad totexp \quad income|$$

because the matrix form can be easy solved out thus, i show the solution of this OLS methods:

the OLS mainly aims to solve the β, based on the minimum of Sum Squared Residuals(SSR):

in normal form

$$SSR = \sum_{i=1}^{N} \epsilon^2$$

in matrix form

$$SSR = (wfood - x\beta)' * (wfood - x\beta)$$

and minimize the SSR

$$\min SSR = \min (wfood - x\beta)' * (wfood - x\beta)$$

for minimum, i make the 1st different should as zero. Thus,

$$\frac{\mathrm{d}(wfood - x\beta)' * (wfood - x\beta)}{\mathrm{d}\beta} = 0$$

$$\frac{\mathrm{d}(wfood - x\beta)' * (wfood - x\beta)}{\mathrm{d}\beta} = \frac{\mathrm{d}(wfood' - \beta' x') * (wfood - x\beta)}{\mathrm{d}\beta} = 0$$

$$\frac{\mathrm{d}(wfood - x\beta)' * (wfood - x\beta)}{\mathrm{d}\beta} = \frac{\mathrm{d}(wfood' * wfood - 2\beta' x' y + \beta' x' x\beta)}{\mathrm{d}\beta} = 0$$

here, the $wfood' * wfood$ and $\beta' x' y$ are non-beta martix or single-beta matrix, after differentiate on β, there will left 0 and the coefficent of single-beta:$x'y$. Thus,

$$\frac{\mathrm{d}(wfood - x\beta)' * (wfood - x\beta)}{\mathrm{d}\beta} = 0 - 2x'y + 2x'x\beta = 0$$

and

$$\beta = (x'x)^{-1} x'y$$

that is the solution of beta from OLS methods

```
x<-cbind(rep(1,length(london$wfood)),log(london$totexp),log(london$income))# create the martix x in the
coef_of_lmboj<-solve(t(x)%*%x)%*%t(x)%*%london$wfood#coef(lmobj)
coef(lmobj)
```

```
##  (Intercept)  log(totexp)  log(income)
##  0.964691151 -0.133104065 -0.001564379
```

```
coef_of_lmboj
```

```
##              [,1]
## [1,]  0.964691151
## [2,] -0.133104065
## [3,] -0.001564379
```

```
fitted_of_lmobj<-x%*%coef_of_lmboj#fitted(lmobj)
head(fitted(lmobj))
```

```
##         1         2         3         4         5         6
## 0.4363703 0.3579097 0.2649803 0.3742214 0.3585440 0.3925529
```

```r
head(fitted_of_lmobj)
```

```
##           [,1]
## [1,] 0.4363703
## [2,] 0.3579097
## [3,] 0.2649803
## [4,] 0.3742214
## [5,] 0.3585440
## [6,] 0.3925529
```

```r
resid_of_lmobj<-london$wfood-x%*%coef_of_lmboj#resid(lmobj)
head(resid(lmobj))
```

```
##            1            2            3            4            5
## -0.009170321  0.015990337 -0.070880272  0.069578635 -0.025443964
##            6
## -0.017352911
```

```r
head(resid_of_lmobj)
```

```
##              [,1]
## [1,] -0.009170321
## [2,]  0.015990337
## [3,] -0.070880272
## [4,]  0.069578635
## [5,] -0.025443964
## [6,] -0.017352911
```

```r
deviance_of_lmobj<-t(resid_of_lmobj)%*%resid_of_lmobj#just calulate the SSR,equal to deviance(lmobj)
deviance(lmobj)
```

```
## [1] 12.66322
```

```r
deviance_of_lmobj
```

```
##          [,1]
## [1,] 12.66322
```

```r
vcov(lmobj)
```

```
##               (Intercept)   log(totexp)   log(income)
## (Intercept)  1.116471e-03 -9.885454e-05 -1.373587e-04
## log(totexp) -9.885454e-05  4.794031e-05 -2.427202e-05
## log(income) -1.373587e-04 -2.427202e-05  5.100646e-05
```