Exercise 1 Guide

Installation:

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

rstudio.com

How to Install R Studio

In order to run R and R-studio on your system, you need to follow the following three steps in the same order.

1. **Install R**
2. **Install R-Studio**
3. **Install R-Packages**

1. Install R

Follow the steps below with respect to the operating system you are using

For Windows :

1. Download the binary setup file for R from the following link.(R for Windows)
2. Open the downloaded .exe file and Install R

For Mac :

1. Download the appropriate version of .pkg file form the following link. (R for Mac)
2. Open the downloaded .pkg file and Install R

For Linux :

1. For complete R System installation in Linux, follow the instructions on the following link (Link)
2. For Ubuntu with Apt-get installed, execute *sudo apt-get install r-base* in terminal.

2. Install R Studio

On the following link Download R Studio choose the appropriate installer file for your operating system, download it and then run it to install R-studio.

3. Install the packages

R requires a particular package/library to be installed in R-studio. You can follow the instructions below to do so

1. Run R studio

2. Click on the Packages tab in the bottom-right section and then click on install. The following dialog box will appear
3. In the Install Packages dialog, write the package name you want to install under the Packages field and then click install. This will install the package you searched for or give you a list of matching package based on your package text.

This completes the installation procedure for R Studio. If you want to continue with the Basic R tutorial click on the Basic Tutorial button in the left column.
R is the statistical language and not the IDE (Integrated Development Environment) – which is Rtstudio

Basic Data Analysis through R/R Studio

In this tutorial, I 'll design a basic data analysis program in R using R Studio by utilizing the features of R Studio to create some visual representation of that data. Following steps will be performed to achieve our goal.

1. Downloading/importing data in R
2. Transforming Data / Running queries on data
3. Basic data analysis using statistical averages
4. Plotting data distribution

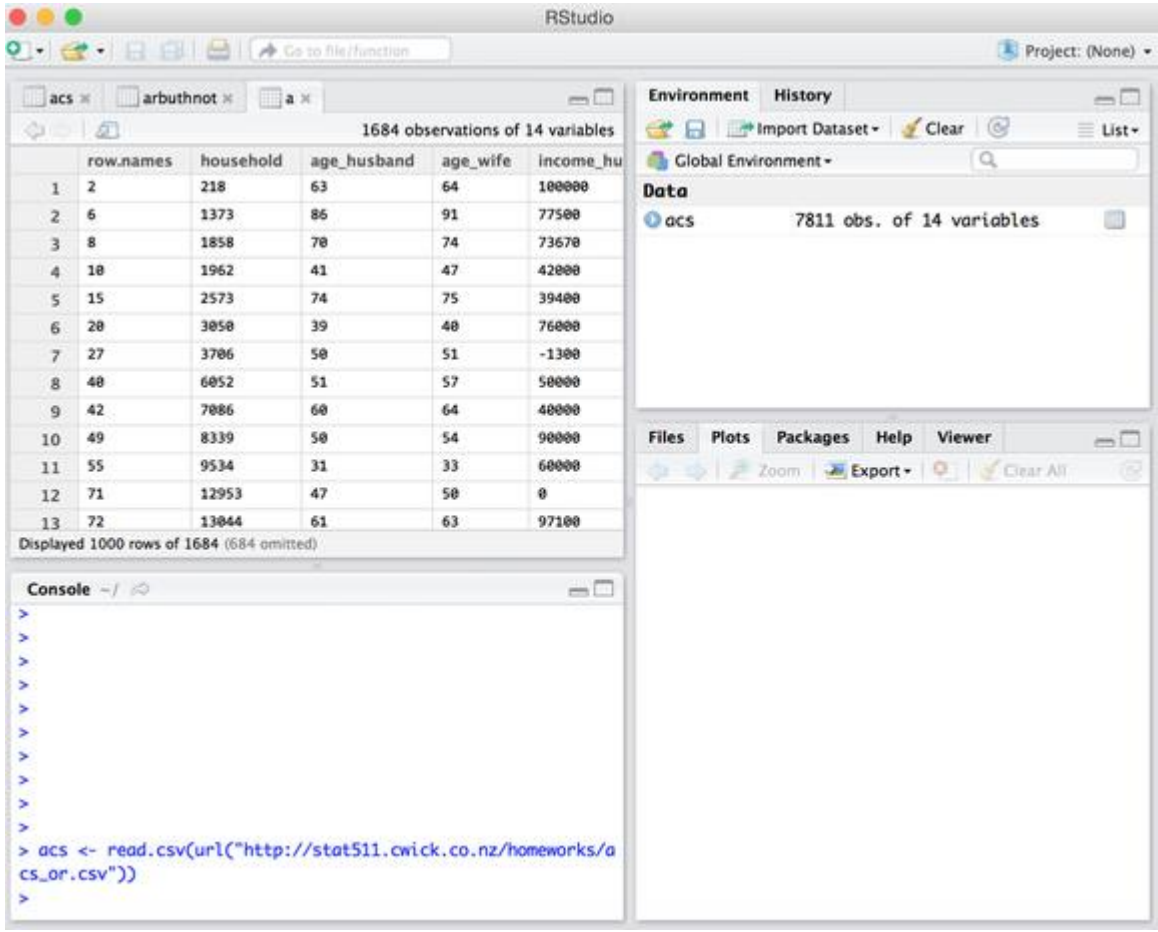
Let's go over the tutorial by performing one step at a time.

1. Importing Data in R Studio

For this tutorial we will use the sample census data set ACS . There are two ways to import this data in R. One way is to import the data programmatically by executing the following command in the console window of R Studio

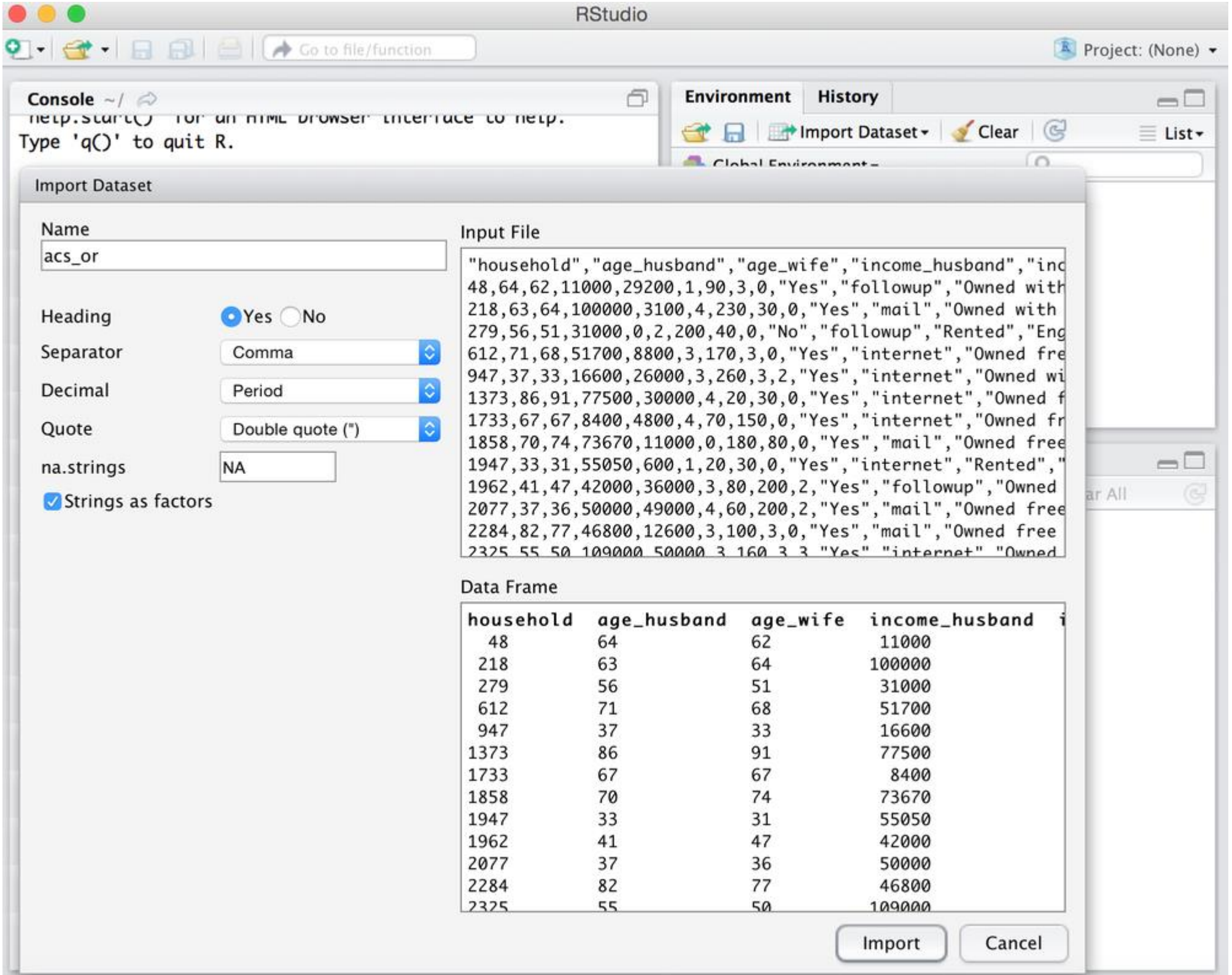
acs <- read.csv(url("http://stat511.cwick.co.nz/homeworks/acs_or.csv"))

Once this command is executed by pressing Enter, the dataset will be downloaded from the internet, read as a *csv* file and assigned to the variable name *acs.*



The second way to import the data set into R Studio is to first download it onto you local computer and use the *import dataset* feature of R Studio. To perform this follow the steps below

1. Click on the *import dataset* button in the top-right section under the environment tab. Select the file you want to import and then click open. The Import Dataset dialog will appear as shown below


RStudio
Go to file/function
Project: (None)
Console ~/
Type 'q()' to quit R.
Environment
History
Import Dataset
Clear
List
Import Dataset
Name
acs_or
Heading
Yes
No
Separator
Comma
Decimal
Period
Quote
Double quote (")
na.strings
NA
Strings as factors
Input File
"household","age_husband","age_wife","income_husband","inc
48,64,62,11000,29200,1,90,3,0,"Yes","followup","Owned with
218,63,64,100000,3100,4,230,30,0,"Yes","mail","Owned with
279,56,51,31000,0,2,200,40,0,"No","followup","Rented","Eng
612,71,68,51700,8800,3,170,3,0,"Yes","internet","Owned fre
947,37,33,16600,26000,3,260,3,2,"Yes","internet","Owned wi
1373,86,91,77500,30000,4,20,30,0,"Yes","internet","Owned f
1733,67,67,8400,4800,4,70,150,0,"Yes","internet","Owned fr
1858,70,74,73670,11000,0,180,80,0,"Yes","mail","Owned free
1947,33,31,55050,600,1,20,30,0,"Yes","internet","Rented","
1962,41,47,42000,36000,3,80,200,2,"Yes","followup","Owned
2077,37,36,50000,49000,4,60,200,2,"Yes","mail","Owned free
2284,82,77,46800,12600,3,100,3,0,"Yes","mail","Owned free
Data Frame
household age_husband age_wife income_husband i
48 64 62 11000
218 63 64 100000
279 56 51 31000
612 71 68 51700
947 37 33 16600
1373 86 91 77500
1733 67 67 8400
1858 70 74 73670
1947 33 31 55050
1962 41 47 42000
2077 37 36 50000
2284 82 77 46800
2325 55 50 109000
Import
Cancel

2. After setting up the preferences of separator, name and other parameters, click on the Import button. The dataset will be imported in R Studio and assigned to the variable name as set before.

Any dataset can be viewed by executing the following line:

```
View(acs)
```

where acs is the variable dataset is assigned to.

2. Transforming Data

Once you are done with importing the data in R Studio, you can use various transformation features of R to manipulate the data. Let's learn few of the basic data access techniques

To access a particular column, Ex. age_husband in our case.

```
acs$age_husband
```

To access data as a vector

```
acs[1,3]
```

To run some queries on data, you can use the *subset* function of R. Let's say I want those rows from the dataset in which the age_husband is greater than age_wife. For this we 'll run the following command in console

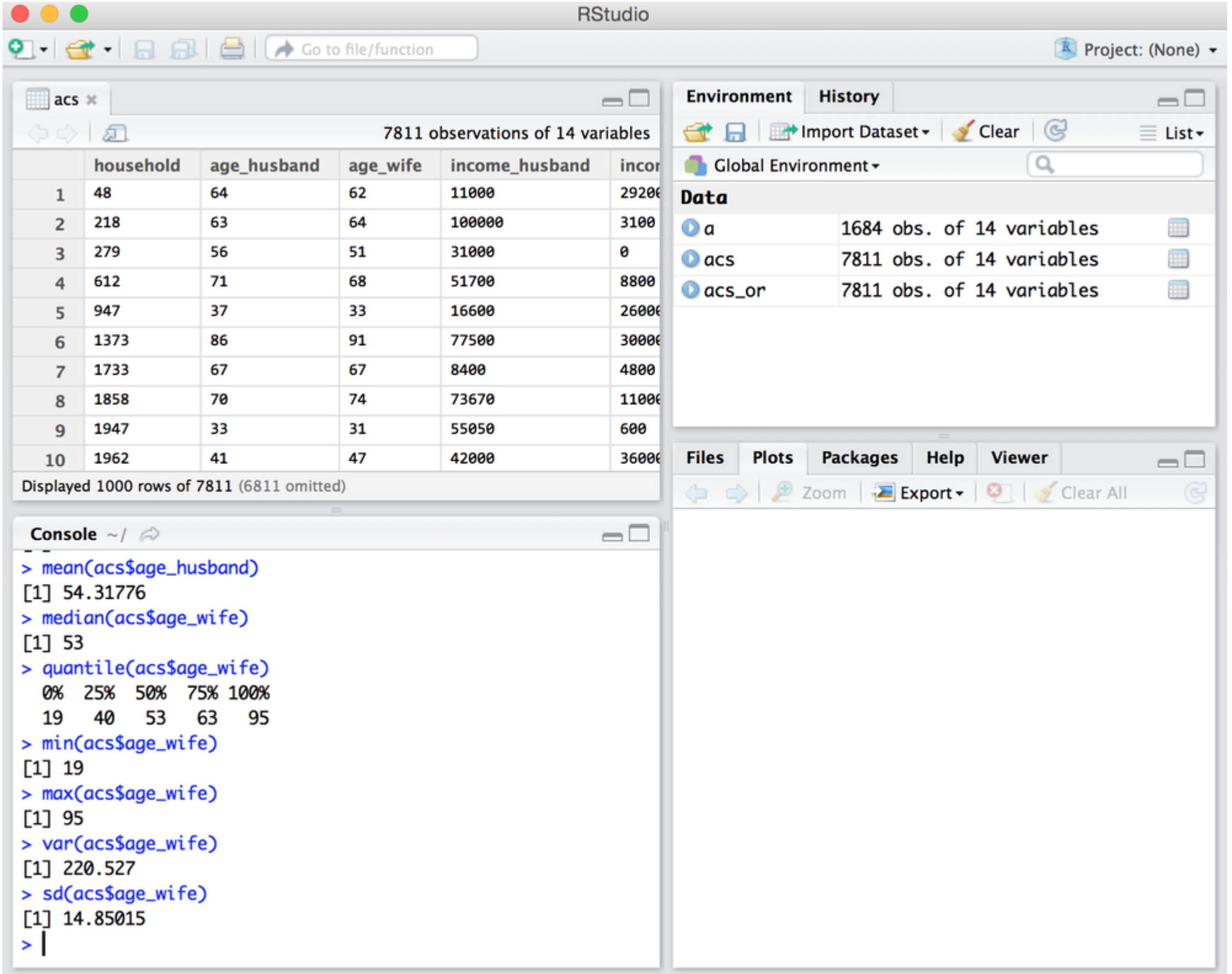
```
a <- subset(acs , age_husband > age_wife)
```

The first parameter to the subset function is the dataframe you want to apply that function to and the second parameter is the boolean condition that needs to be checked for each row to be included or not. So the above statement will return the set the rows in which the age_husband is greater than age_wife and assign those rows to a

Getting Statistical Averages from data

Following functions can be used to calculate the averages of the dataset

1. For mean of any column, run : `mean(acs$age_husband)`
2. Median, run : `median(acs$age_husband)`
3. Quantile , run : `quantile(acs$age_wife)`
4. Variance , run : `var(acs$age_wife)`
5. Standard Deviation , run : `sd(acs$age_wife)`


RStudio
Go to file/function
Project: (None)
acs
7811 observations of 14 variables
household age_husband age_wife income_husband incor
1 48 64 62 11000 29200
2 218 63 64 100000 3100
3 279 56 51 31000 0
4 612 71 68 51700 8800
5 947 37 33 16600 26000
6 1373 86 91 77500 30000
7 1733 67 67 8400 4800
8 1858 70 74 73670 11000
9 1947 33 31 55050 600
10 1962 41 47 42000 36000
Displayed 1000 rows of 7811 (6811 omitted)
Environment
History
Import Dataset
Clear
List
Global Environment
Data
a 1684 obs. of 14 variables
acs 7811 obs. of 14 variables
acs_or 7811 obs. of 14 variables
Files
Plots
Packages
Help
Viewer
Zoom
Export
Clear All
Console ~/
> mean(acs$age_husband)
[1] 54.31776
> median(acs$age_wife)
[1] 53
> quantile(acs$age_wife)
0% 25% 50% 75% 100%
19 40 53 63 95
> min(acs$age_wife)
[1] 19
> max(acs$age_wife)
[1] 95
> var(acs$age_wife)
[1] 220.527
> sd(acs$age_wife)
[1] 14.85015
>

You can also get the statistical summary of the dataset by just running on either a column or the complete dataset

```
summary(acs)
```

4. Plotting Data

A very liked feature of R studio is its built in data visualizer for R. Any data set imported in R can visualized using the plot and several other functions of R. For Example

Subset the first 100 rows of acs

```
s <- acs[1:100,]
```

To create a scatter plot of a data set, you can run the following command in console

```
plot(x = s$age_husband , y = s$age_wife, type = 'p')
```

Where s is the subset of the original dataset and type 'p' set the plot type as point. You can aslo choose line and other change type variable to 'L' etc.

RStudio

Go to file/function

Project: (None)

acs × s ×

100 observations of 14 variables

77	14676	78	77	48500	6000
78	14710	55	55	50550	3500
79	14866	37	31	30000	25000
80	15388	29	30	38700	38200
81	15532	59	52	49000	5200
82	15918	48	49	19000	47000
83	16282	29	23	14000	6000
84	16446	34	33	48000	0
85	16970	29	27	9000	28500
86	17056	41	36	35000	28000
87	17532	38	34	78000	0
88	17554	62	53	57420	50880
89	17799	74	71	66000	36500
90	17872	43	34	355000	0
91	17893	65	67	6700	15600
92	18230	24	23	27000	10000
93	18744	34	33	0	26000
94	19239	58	54	20100	0
95	19342	48	48	44000	26000
96	19509	60	59	40000	14000
97	19835	66	65	14200	3800

Console ~/

```
> View(s)
> View(s)
> plot(x = s$age_husband , y = s$age_wife, type = 'p')
>
```

Environment History

Import Dataset Clear List

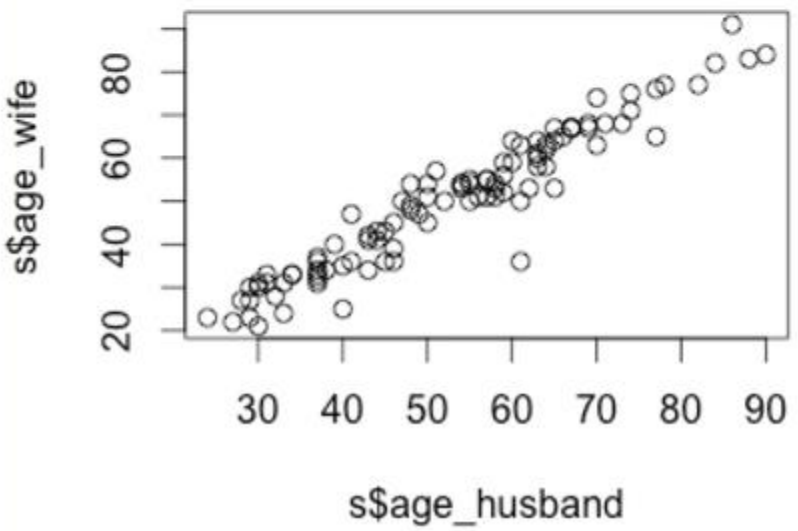
Global Environment

Data

a	1684 obs. of 14 variables
acs	7811 obs. of 14 variables
acs_or	7811 obs. of 14 variables
s	100 obs. of 14 variables

Files Plots Packages Help Viewer

Zoom Export Clear All



For data distribution plots, there are several features tools and packages available in R that you can use to draw any kind of distribution. For example

To draw a Histogram of a dataset, you can run the command

```
hist(acs$number_children)
```

RStudio

Go to file/function

Project: (None)

acs × s ×

7811 observations of 14 variables

ooms	electricity	gas	number_children	internet	mode	own
	90	3	0	Yes	followup	Owned
	230	30	0	Yes	mail	Owned
	200	40	0	No	followup	Rente
	170	3	0	Yes	internet	Owned
	260	3	2	Yes	internet	Owned
	20	30	0	Yes	internet	Owned
	70	150	0	Yes	internet	Owned
	180	80	0	Yes	mail	Owned
	20	30	0	Yes	internet	Rente
	80	200	2	Yes	followup	Owned
	60	200	2	Yes	mail	Owned
	100	3	0	Yes	mail	Owned
	160	3	3	Yes	internet	Owned
	60	50	0	Yes	internet	Owned
	70	130	0	No	mail	Owned
	160	3	0	Yes	mail	Owned
	280	140	2	Yes	followup	Owned
	60	60	2	Yes	internet	Owned
	110	3	0	Yes	mail	Owned
	110	3	2	Yes	internet	Owned

Displayed 1000 rows of 7811 (6811 omitted)

Console ~/

```
> View(s)
> plot(x = s$age_husband , y = s$age_wife, type = 'p')
> hist(acs$number_children)
>
```

Environment History

Import Dataset Clear List

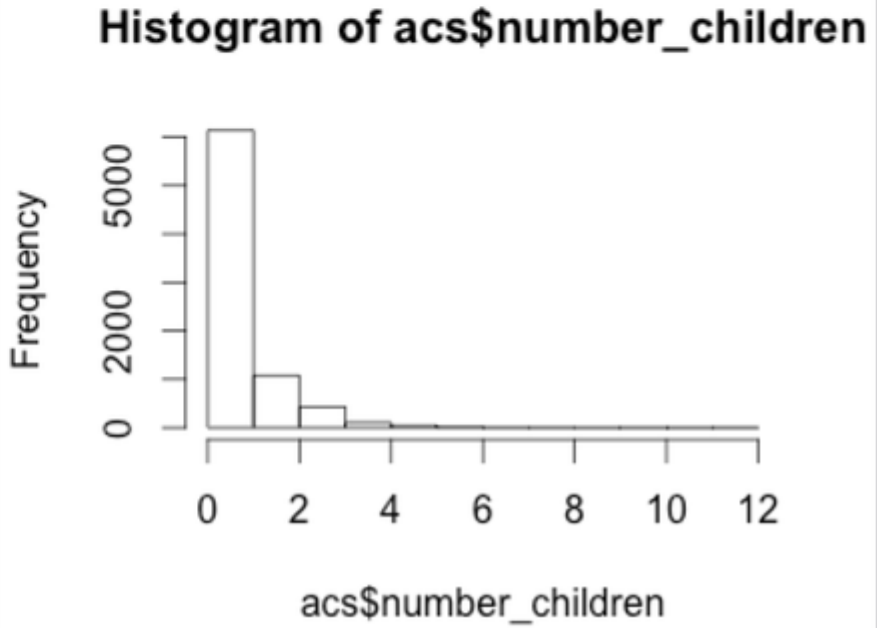
Global Environment

Data

a	1684 obs. of 14 variables
acs	7811 obs. of 14 variables
acs_or	7811 obs. of 14 variables
s	100 obs. of 14 variables

Files Plots Packages Help Viewer

Zoom Export Clear All



Similarly for Bar Plots, run the following set of commands

```
counts <- table(acs$bedrooms)
barplot(counts, main="Bedrooms Distribution",  xlab="Number of Bedrooms")
```


RStudio

Go to file/function

Project: (None)

acs × s ×

7811 observations of 14 variables

_wife	income_husband	income_wife	bedrooms	electricity	gas
	11000	29200	1	90	3
	100000	3100	4	230	30
	31000	0	2	200	40
	51700	8800	3	170	3
	16600	26000	3	260	3
	77500	30000	4	20	30
	8400	4800	4	70	150
	73670	11000	0	180	80
	55050	600	1	20	30
	42000	36000	3	80	200
	50000	49000	4	60	200
	46800	12600	3	100	3
	109000	50000	3	160	3
	95000	0	4	60	50
	39400	9700	2	70	130
	10000	122000	4	160	3
	110000	9300	4	280	140
	60000	33000	3	60	60
	22400	17660	3	110	3
	76000	10700	3	110	3

Displayed 1000 rows of 7811 (6811 omitted)

Console ~/

```
> counts <- table(acs$bedrooms)
> barplot(counts, main="Bedrooms Distribution",
+         xlab="Number of Bedrooms")
>
```

Environment History

Import Dataset Clear List

Global Environment

Data

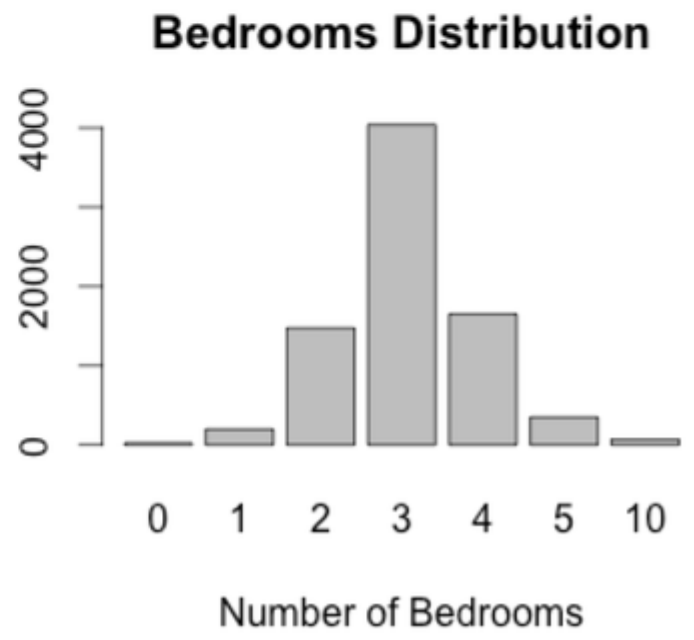
a	1684 obs. of 14 variables
acs	7811 obs. of 14 variables
acs_or	7811 obs. of 14 variables
s	100 obs. of 14 variables

Values

counts	'table' int [1:7(1d)] 24 198 147…

Files Plots Packages Help Viewer

Zoom Export Clear All



I hope this will give you a basic idea on how to do simple statistics in R.

Note

For any documentation or usage of the function in R Studio, just type the name of the function and then press *cntrl+space* to get the auto completion window.
You can use *?* before any function name to view the official documentation

'>' – indicates that this is where to type the input. This starts the beginning of the input and is not meant to be added. Enter commands after the '>'

From the "Console" (lower left quadrant of RStudio)
Example:
> 1+1
[1] 2

'[1]' - is the indicator of the index of the first line

Code Gotcha's

'+' can indicate missing code

- "+" after code generation means you are missing some syntax
  - Comma
  - Quote
  - Parenthesis
  - Press [ESC] to try again

Example
> 1+
+
To get back to the > you can press ESC

Case Sensitivity
R interprets variable names differently for example Z vs z

Quick commands

[ALT -] = <-

- “>” the prompt
- Keyboard Shortcut [ALT-] = “<-”
- Objects
    - Must start with a letter and can only contain letters, numbers, “_” and “.”
- Code completion using [Tab] for variables, functions, packages, etc

VARIABLE TYPES

- int – integer
- dbl – double or real number
- chr – character vector or string
- dttm – date-time
- lgl – logical – TRUE or FALSE
- fctr – factors R uses to represent categorical variables with fixed possible values
- date - dates

- The R console or standard out functions much like a calculator
- Assignment statements
    - Object_name <- value
    - You can mentally read this aloud as “object name gets (“<-”) value”
- Can use “=“ instead of “<-” but it gets confusing when you begin using math formulas

- **Comparisons & Boolean & NA**
- < - LESS THAN
- > - GREATER THAN
- <= - LESS THAN AND EQUAL TO
- <= - GREATER THAN AND EQUAL TO
- != - NOT EQUAL
- == - identifies equality
- = - identifies assignment and not equality
- & ”and”
- | “or”
- ! “not”
- NA “not available”
- is.na – determine if a value is missing or NA

- How do find those values that are not NA?

Example
1 != 1
[1] FALSE
^asking if 1 is not equal to 1. R identified it to be a FALSE Boolean statement

Evaluate a statement
> Average.Age = 41
> My.Age = 31
> (Average.Age >= 10)
[1] TRUE
(My.Age <= 32)

Average.Age >= 10 & My.Age >= 10
[1] TRUE

We can combine statements to get a Boolean answer back as well

ORDER OF OPERATIONS – PEMDAS () ^ * / + -

> (1+2) * 10
[1] 30

The answer is 30 because R follows the order of operations where it takes the 1+2 first which is in the parenthesis and then multiples the result times 10 to give an answer of 30

Example 2
> 10 / 2 * 6
[1] 30

Divide 10 by 2 first then multiple by 6

Functions

- **Mean – average - mean()**

- **Median – middle value – median()**
- **Mode – most common value – mode()**
- **Range – difference between the largest and the smallest value – range()**
- **Variance – numerical measure of how the data values are dispersed around the mean. The average of the squared distances from the mean – var()**
- **Standard deviation – a measure of how spread out the numbers are - sd()**

SORTING

??dplyr – ?? will get help on specified packages

- **Filter() - pick observations by their values**
- **Arrange()** – **reorder the rows**
- **Select()** – **pick variables by their names**
- **Mutate()** – **create new variables with functions of existing varables**
- **Summarize()** – **collapse many variables down to a single summary**
- **group_by()** – **changes the scope of each function from operating on the entire data set to operating on it group-by-group**

Examples of data transformation

```
1  library(quantmod)
2  getSymbols("GOOGL")
3  GOOGL.2016 <- GOOGL['2016']
4  GOOGL.2016.df <- as.data.frame(GOOGL.2016)
5
6  write.csv(GOOGL.2016.df, "googlestock.csv")
```

```r
subset(GOOGL.df, PX_LAST > 738)

median(GOOGL.df$PX_LAST)

centralvalue(GOOGL.df$date)

max(GOOGL.df$PX_LAST)

var(GOOGL.df$PX_LAST)

sd(GOOGL.df$PX_LAST)

describe(GOOGL.df)

summary(GOOGL.df)

range(GOOGL.df$PX_LAST)

#mathisfun.com/data/standard-deviation.html
```