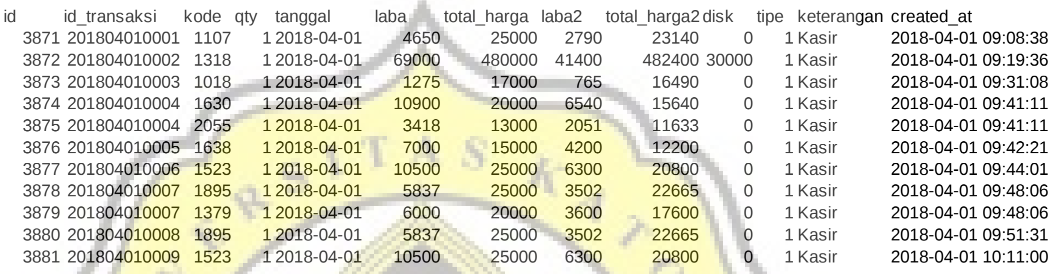
CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

CSV data used consists of sales data for pet shops, as follows, sample sales of pet shops:



id	id_transaksi	kode	qty	tanggal	laba	total_harga	laba2	total_harga2	disk	tipe	keterangan	created_at
3871	201804010001	1107	1	2018-04-01	4650	25000	2790	23140	0	1	Kasir	2018-04-01 09:08:38
3872	201804010002	1318	1	2018-04-01	69000	480000	41400	482400	30000	1	Kasir	2018-04-01 09:19:36
3873	201804010003	1018	1	2018-04-01	1275	17000	765	16490	0	1	Kasir	2018-04-01 09:31:08
3874	201804010004	1630	1	2018-04-01	10900	20000	6540	15640	0	1	Kasir	2018-04-01 09:41:11
3875	201804010004	2055	1	2018-04-01	3418	13000	2051	11633	0	1	Kasir	2018-04-01 09:41:11
3876	201804010005	1638	1	2018-04-01	7000	15000	4200	12200	0	1	Kasir	2018-04-01 09:42:21
3877	201804010006	1523	1	2018-04-01	10500	25000	6300	20800	0	1	Kasir	2018-04-01 09:44:01
3878	201804010007	1895	1	2018-04-01	5837	25000	3502	22665	0	1	Kasir	2018-04-01 09:48:06
3879	201804010007	1379	1	2018-04-01	6000	20000	3600	17600	0	1	Kasir	2018-04-01 09:48:06
3880	201804010008	1895	1	2018-04-01	5837	25000	3502	22665	0	1	Kasir	2018-04-01 09:51:31
3881	201804010009	1523	1	2018-04-01	10500	25000	6300	20800	0	1	Kasir	2018-04-01 10:11:00

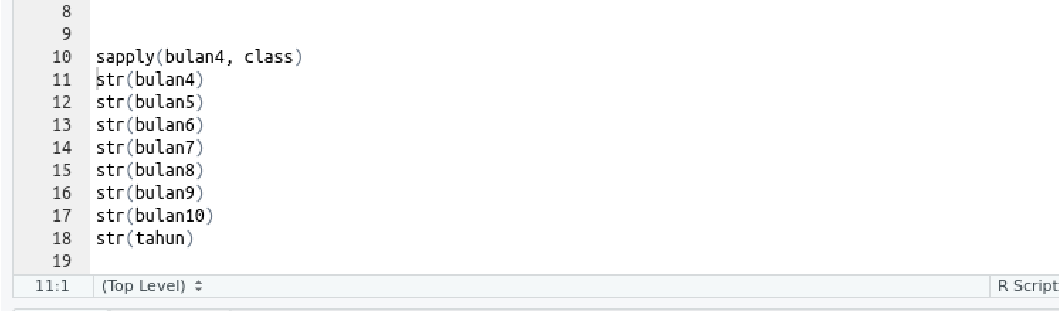
Illustration 5.1:Examples of CSV sales data

Figure above is an example of CSV data in sales that will be processed by R programming calculations. Only the code, profit and date are processed in the data, because to determine sales it is necessary to know what the sales profit is in one month.

How to read CSV data on the R programming language can be done as follows:

```
bulan4 <- read.csv(file='/home/samuel/bulan4.csv', header = TRUE)
```

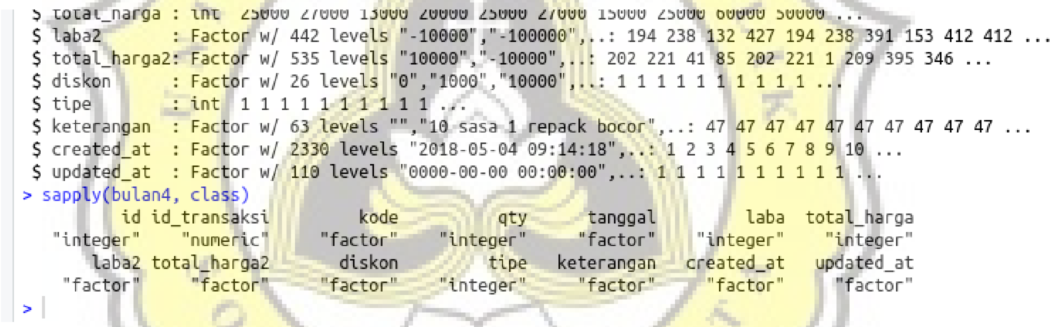
In month 4 identification of data can be run globally. Read the desired file according to the library file: `file='/home/samuel/bulan4.csv'`. And `header = TRUE` will make the first line a header.



```
sapply(bulan4, class)
str(bulan4)
str(bulan5)
str(bulan6)
str(bulan7)
str(bulan8)
str(bulan9)
str(bulan10)
str(tahun)
```

Illustration 5.2: Syntax to see the type of CSV data type

The picture above is to find out the type of data type in CSV that has been inputted with the program: sapply(bulan4,class) and `str(bulan4)` the process of converting data into a string data type.

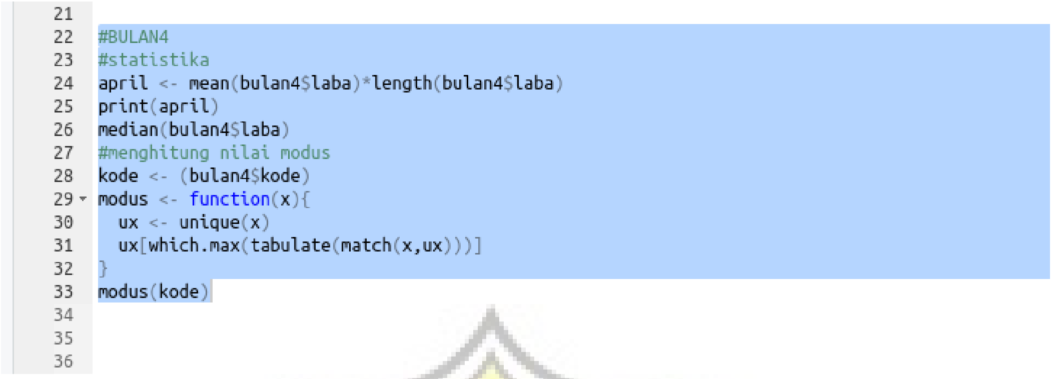


```
 $ total_harga : int  25000 27000 13000 20000 25000 27000 15000 25000 60000 50000 ...
 $ laba2       : Factor w/ 442 levels "-10000","-100000",..: 194 238 132 427 194 238 391 153 412 412 ...
 $ total_harga2: Factor w/ 535 levels "10000","-10000",..: 202 221 41 85 202 221 1 209 395 346 ...
 $ diskon      : Factor w/ 26 levels "0","1000","10000",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ tipe        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ keterangan  : Factor w/ 63 levels "","10 sasa 1 repack bocor",..: 47 47 47 47 47 47 47 47 47 47 ...
 $ created_at  : Factor w/ 2330 levels "2018-05-04 09:14:18",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ updated_at  : Factor w/ 110 levels "0000-00-00 00:00:00",..: 1 1 1 1 1 1 1 1 1 1 ...
> sapply(bulan4, class)
         id id_transaksi         kode          qty      tanggal         laba  total_harga 
  "integer"    "numeric"     "factor"    "integer"     "factor"    "integer"    "integer" 
      laba2 total_harga2       diskon         tipe   keterangan   created_at   updated_at 
   "factor"     "factor"     "factor"    "integer"     "factor"     "factor"     "factor" 
> 
```

Illustration 5.3: Output results on illustration 5.2

The output results can find out the data type that the code has a data type “*factor*”, date has a data type “*factor*” and profit has a data type “*integer*”.

Processing data with R can be seen in the picture below



```
#BULAN4
#statistika
april <- mean(bulan4$laba)*length(bulan4$laba)
print(april)
median(bulan4$laba)
#menghitung nilai modus
kode <- (bulan4$kode)
modus <- function(x){
  ux <- unique(x)
  ux[which.max(tabulate(match(x,ux)))]
}
modus(kode)
```

Illustration 5.4: Statistical syntax

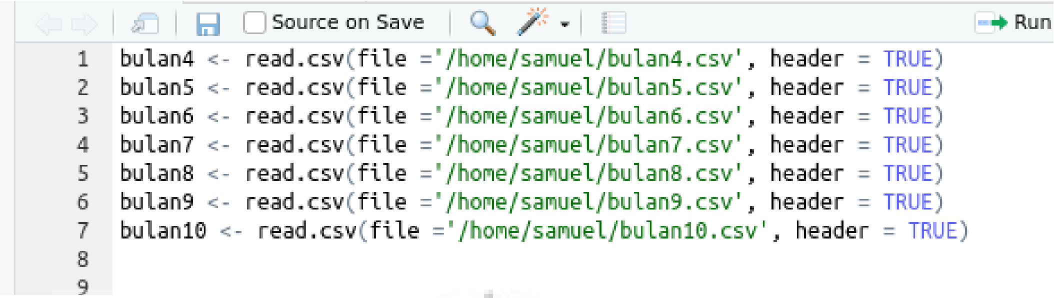
In the picture above the process of looking for a syntax calculation on csv data that has been input to R. Mean, the median in the R language has been provided with the default R function that is *`mean(), median().`* In calculated mode use *function()* that *ux* is unique data from (x) and then *ux* processed looking for values that often appear continuously.

How to display data into a graphical form can use an existing R package package that is `plot(x, y, ...)` “x” is the coordinate of point x, "y" is the coordinate of point y and "..." the argument already in the program R is made as desired.

Read the output data in the form of CSV on processes that have been processed in R by way `write.csv(x, “file”) the default package R with "x" is an object in the process that has data type and` “file” desired output file name.

5.2 Testing

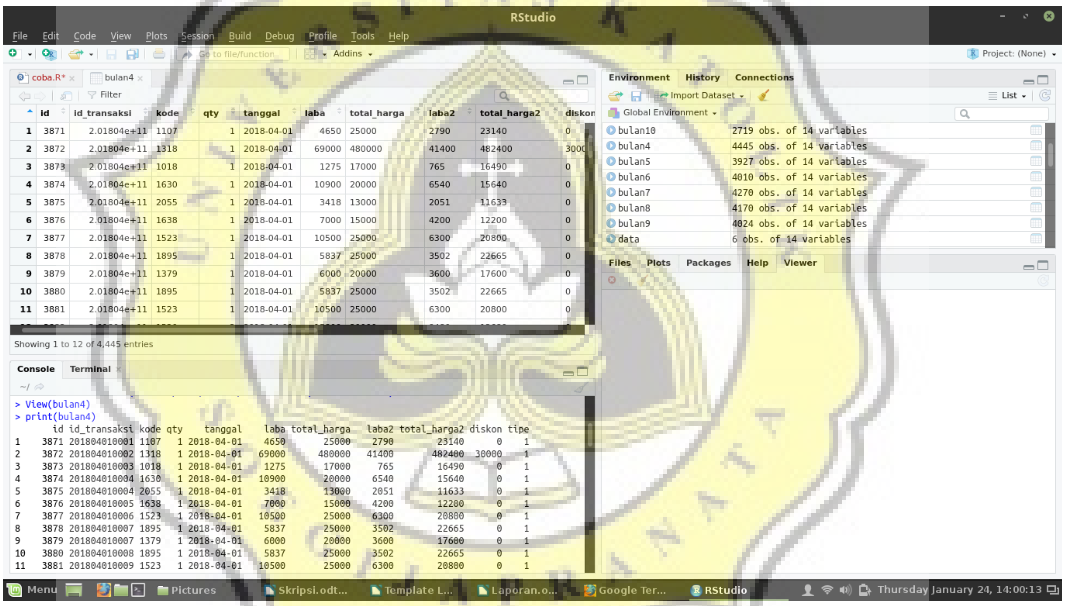
Discussion of sales data with R programming for example has a CSV file as much as monthly sales. So each CSV file is a one-month sale. For example, input CSV data from files 4 to 10 months as follows:



```
bulan4 <- read.csv(file ='/home/samuel/bulan4.csv', header = TRUE)
bulan5 <- read.csv(file ='/home/samuel/bulan5.csv', header = TRUE)
bulan6 <- read.csv(file ='/home/samuel/bulan6.csv', header = TRUE)
bulan7 <- read.csv(file ='/home/samuel/bulan7.csv', header = TRUE)
bulan8 <- read.csv(file ='/home/samuel/bulan8.csv', header = TRUE)
bulan9 <- read.csv(file ='/home/samuel/bulan9.csv', header = TRUE)
bulan10 <- read.csv(file ='/home/samuel/bulan10.csv', header = TRUE)
```

Illustration 5.5: Input CSV data to program R

Output from the raed CSV program



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
coba.R*
bulan4
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
Console Terminal
~/
Menu Pictures Skripsi.odt... Template L... Laporan.o... Google Ter... RStudio
Thursday January 24, 14:00:13

	id	id_transaksi	kode	qty	tanggal	laba	total_harga	laba2	total_harga2	diskon
1	3871	2.01804e+11	1107	1	2018-04-01	4650	25000	2790	23140	0
2	3872	2.01804e+11	1318	1	2018-04-01	69000	480000	41400	482400	3000
3	3873	2.01804e+11	1018	1	2018-04-01	1275	17000	765	16490	0
4	3874	2.01804e+11	1630	1	2018-04-01	10900	20000	6540	15640	0
5	3875	2.01804e+11	2055	1	2018-04-01	3418	13000	2051	11633	0
6	3876	2.01804e+11	1638	1	2018-04-01	7000	15000	4200	12200	0
7	3877	2.01804e+11	1523	1	2018-04-01	10500	25000	6300	20800	0
8	3878	2.01804e+11	1895	1	2018-04-01	5837	25000	3502	22665	0
9	3879	2.01804e+11	1379	1	2018-04-01	6000	20000	3600	17600	0
10	3880	2.01804e+11	1895	1	2018-04-01	5837	25000	3502	22665	0
11	3881	2.01804e+11	1523	1	2018-04-01	10500	25000	6300	20800	0

Showing 1 to 12 of 4,445 entries

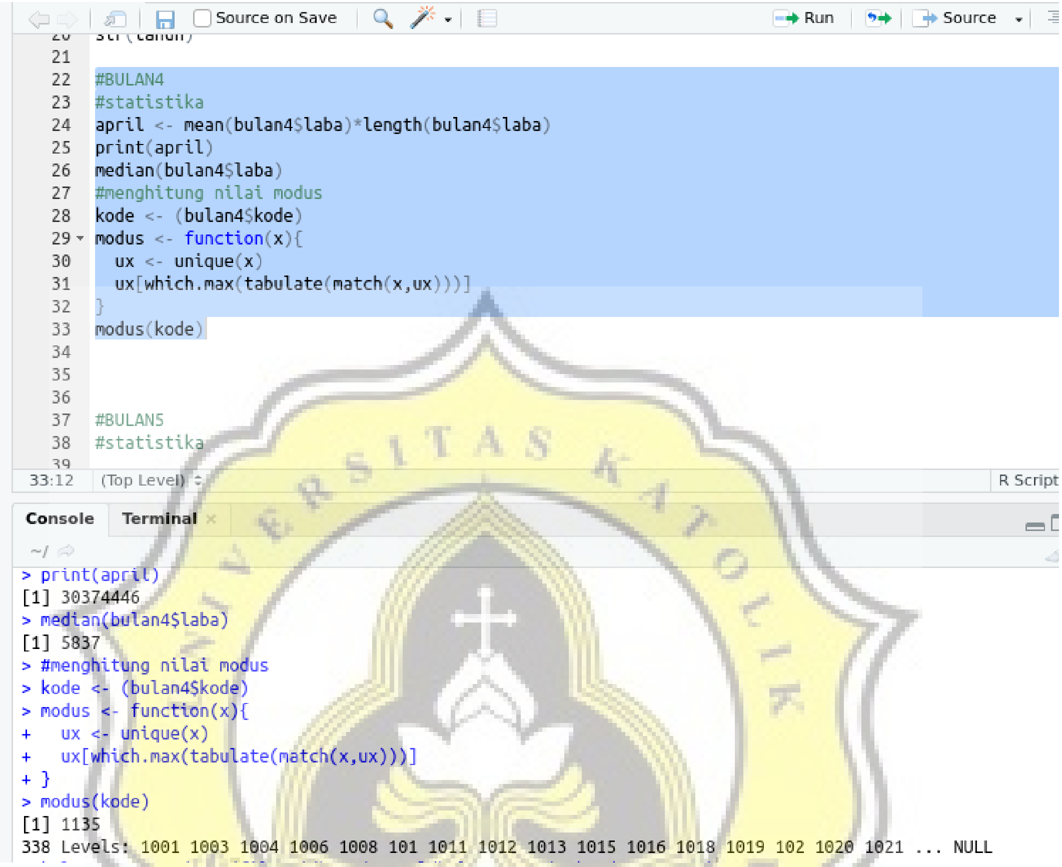
bulan10	2719 obs. of 14 variables
bulan4	4445 obs. of 14 variables
bulan5	3927 obs. of 14 variables
bulan6	4010 obs. of 14 variables
bulan7	4270 obs. of 14 variables
bulan8	4170 obs. of 14 variables
bulan9	4024 obs. of 14 variables
data	6 obs. of 14 variables

```
> View(bulan4)
> print(bulan4)
     id id_transaksi kode qty    tanggal  laba total_harga  laba2 total_harga2 diskon tipe
1  3871 201804010001 1107   1 2018-04-01  4650       25000   2790        23140      0    1
2  3872 201804010002 1318   1 2018-04-01 69000      480000  41400       482400  30000    1
3  3873 201804010003 1018   1 2018-04-01  1275       17000    765        16490      0    1
4  3874 201804010004 1630   1 2018-04-01 10900       20000   6540        15640      0    1
5  3875 201804010004 2055   1 2018-04-01  3418       13000   2051        11633      0    1
6  3876 201804010005 1638   1 2018-04-01  7000       15000   4200        12200      0    1
7  3877 201804010006 1523   1 2018-04-01 10500       25000   6300        20800      0    1
8  3878 201804010007 1895   1 2018-04-01  5837       25000   3502        22665      0    1
9  3879 201804010007 1379   1 2018-04-01  6000       20000   3600        17600      0    1
10 3880 201804010008 1895   1 2018-04-01  5837       25000   3502        22665      0    1
11 3881 201804010009 1523   1 2018-04-01 10500       25000   6300        20800      0    1
```

Illustration 5.6: Output read CSV

From the picture above shows the input results from CSV

After inputting the data file, the next stage processes and calculates from the data that has been inputted in the following way:



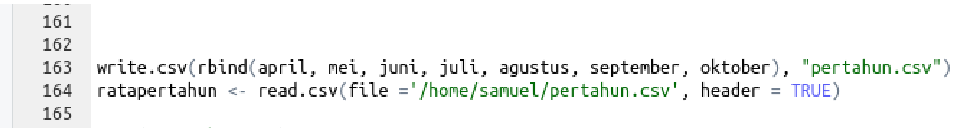
```
str(tahun)

#BULAN4
#statistika
april <- mean(bulan4$laba)*length(bulan4$laba)
print(april)
median(bulan4$laba)
#menghitung nilai modus
kode <- (bulan4$kode)
modus <- function(x){
  ux <- unique(x)
  ux[which.max(tabulate(match(x,ux)))]
}
modus(kode)



#BULAN5
#statistika
```

```
> print(april)
[1] 30374446
> median(bulan4$laba)
[1] 5837
> #menghitung nilai modus
> kode <- (bulan4$kode)
> modus <- function(x){
+   ux <- unique(x)
+   ux[which.max(tabulate(match(x,ux)))]
+ }
> modus(kode)
[1] 1135
338 Levels: 1001 1003 1004 1006 1008 101 1011 1012 1013 1015 1016 1018 1019 102 1020 1021 ... NULL
```

Illustration 5.7:The statistical process of data and process results

With the result that the average in month 4 is 30,374,446 in one month, has a middle value of profit of 5,837 and the mode of the item code that often appears in transactions is code 1135. From month 4 process can be made until month 10 according to data which exists. So that you can know the average, median, and mode value.

In the process of counting each month with different files that have weaknesses or disadvantages, they cannot see income in one year. From the shortcomings there is a solution that can help see income in one year with a graph. The process is as follows:

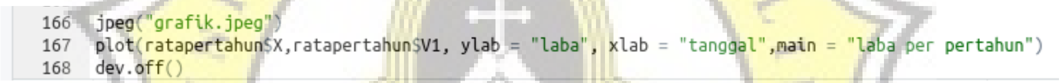


```
write.csv(rbind(april, mei, juni, juli, agustus, september, oktober), "pertahun.csv")
ratapertahun <- read.csv(file ='/home/samuel/pertahun.csv', header = TRUE)
```

Illustration 5.8: Output data to CSV then input

The picture above makes the program output to CSV with syntax `write.csv(rbind(april,mei,juni,juli,agustus,september,oktober),”pertahun.csv”).` Read the R program with CSV output. After making the output data to CSV directly input the data again from the file stored with the file name “pertahun.csv”.

After the process of input file output has finished the next step is to make / convert to the graph, in the following way:



```
jpeg("grafik.jpeg")
plot(ratapertahun$X,ratapertahun$V1, ylab = "laba", xlab = "tanggal",main = "laba per pertahun")
dev.off()
```

Illustration 5.9: Make a graph

In R programming a package graph has been provided with `plot(ratapertahun$X, ratapertahun$V1, ylab=”laba”, xlab=”tanggal”, main=”laba per tahun”). Explanation` ratapertahun$X CSV data file whose input reads the file "ratapertahun$X" with the X header occurs in "ratapertahun$ V1". In ylab and xlab for titles with an x, y axis. And main=”laba per tahun” to make a title. So that the output graph is as follows:

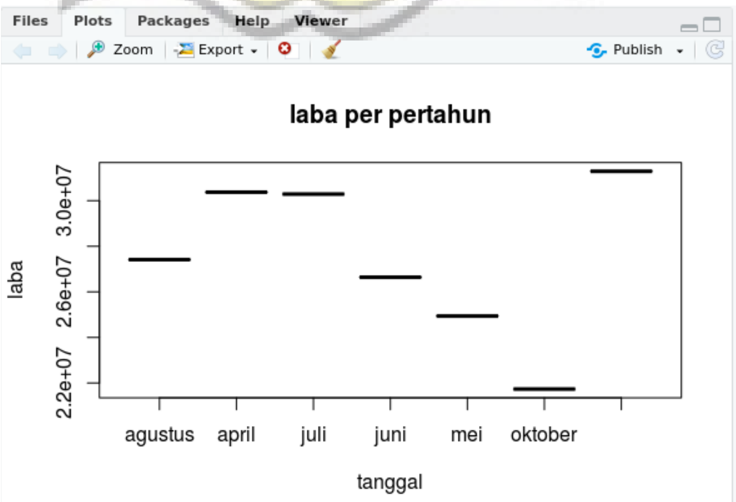


Illustration 5.10: Annual profit graph image

How to save graphics to image format files with `jpeg("grafik.jpeg")` dan `dev.off()`.

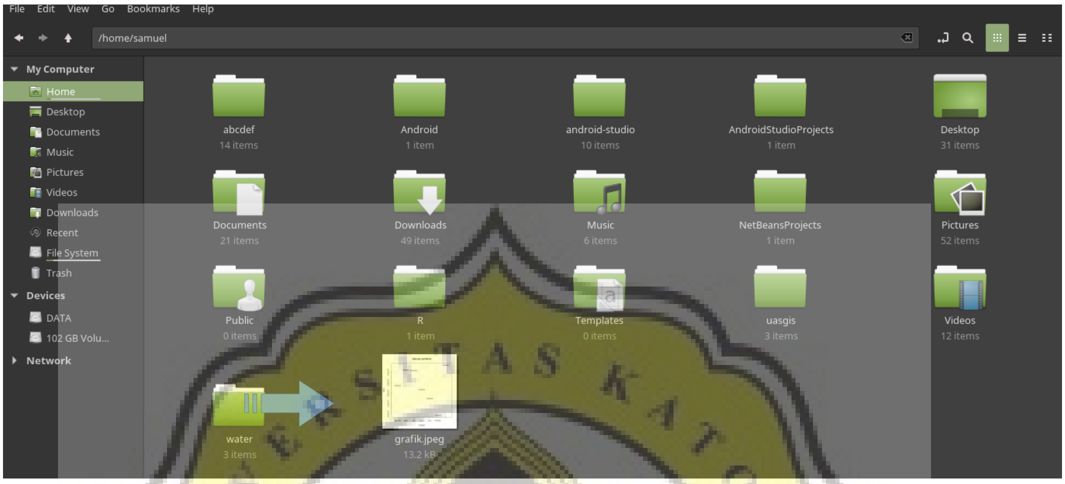


Illustration 5.11: The results save the graph to JPEG

The output of the Rstudio graph which is directly stored on the destination document computer. The picture above explains that rstudio graphics are saved in jpeg format, such as arrows.