jamiaex3.R

jamia

2022-10-28

```
setwd("F:/INTERMATH/intermath 2021-2023/spain/DV/Ex3")
#ex-3.1
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

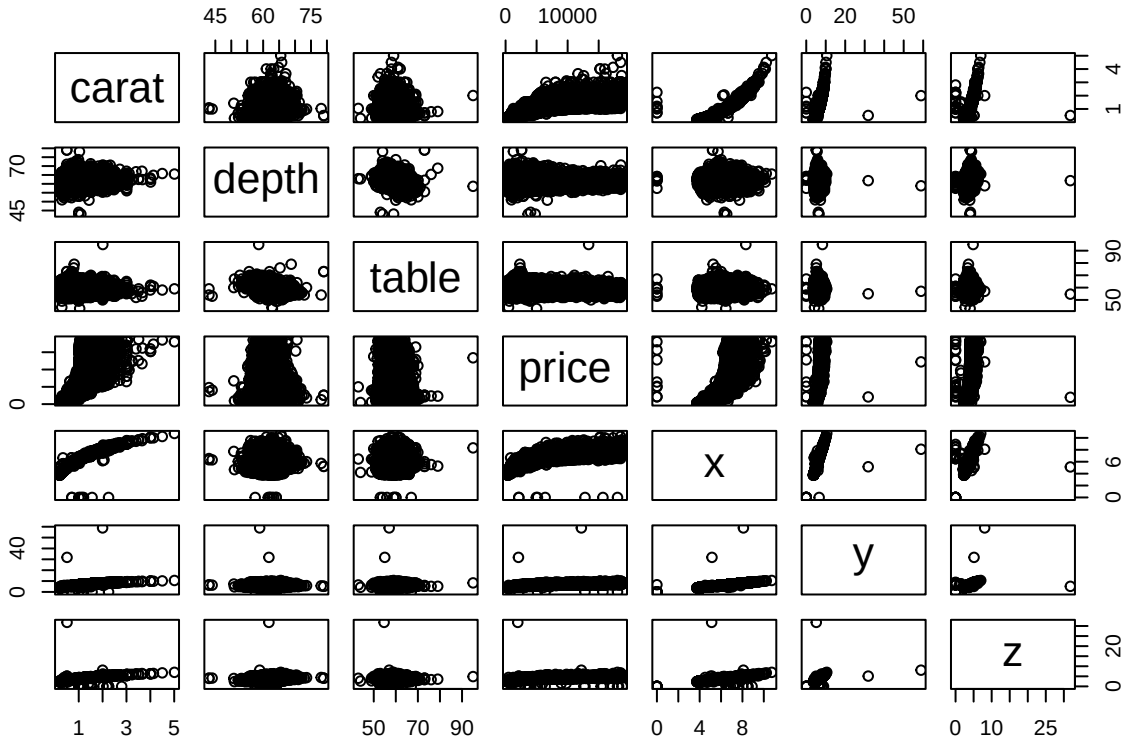
```
diamonds
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

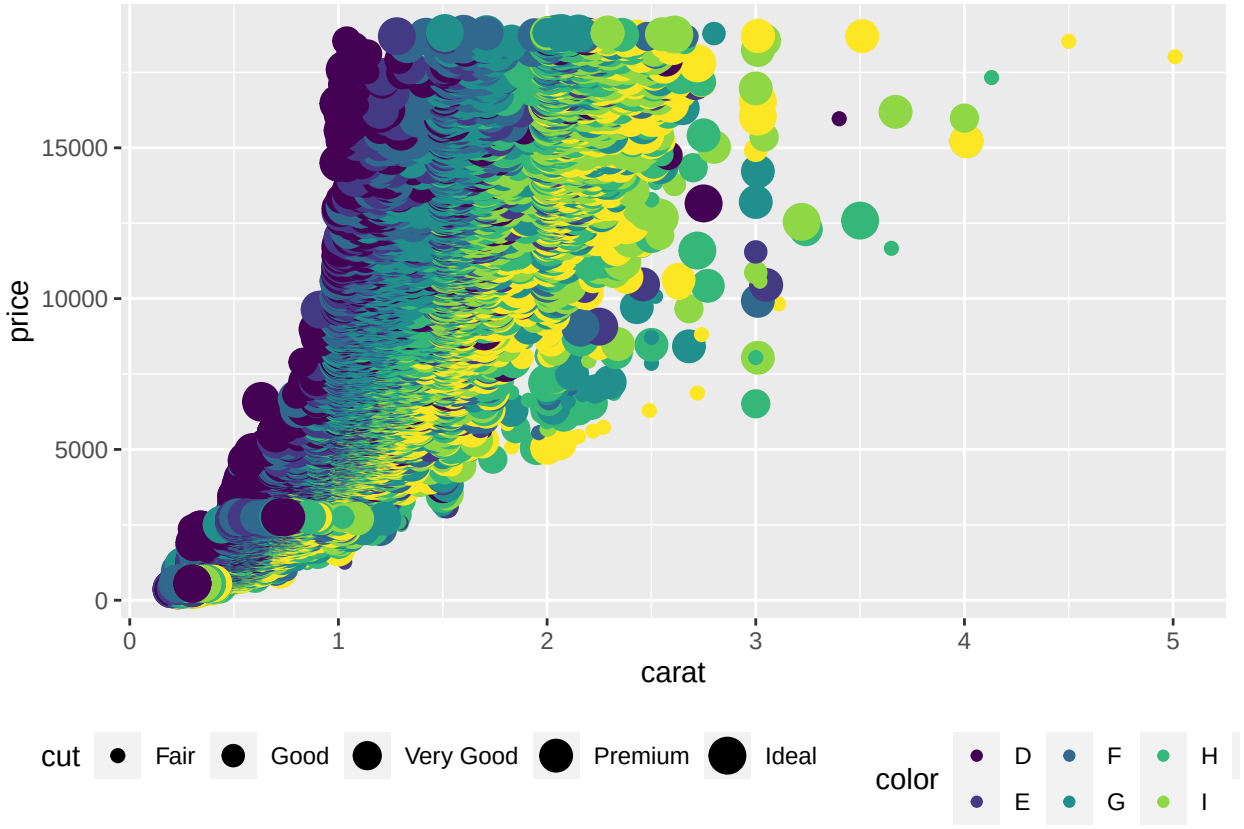
```
print(is.factor(diamonds$color))
```

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

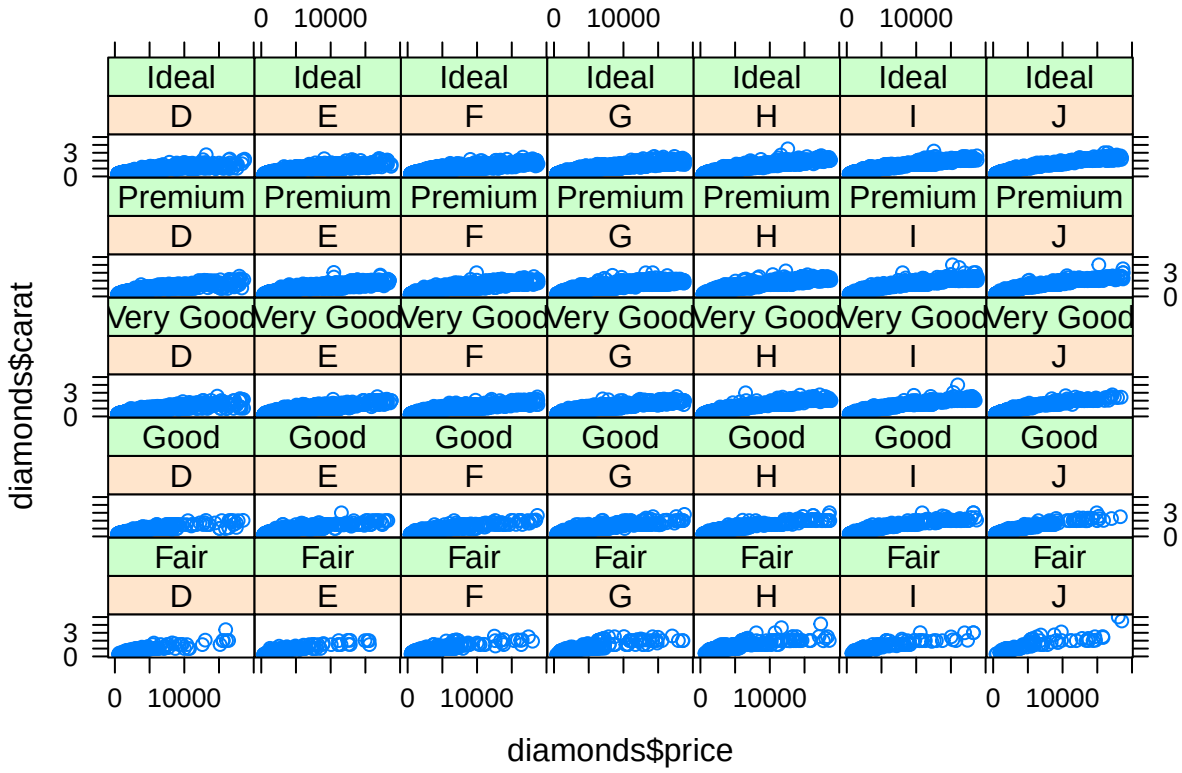
```
pairs(diamonds[,c(1,5,6,7,8,9,10)]) #pairs function gives a
```




```
#matrix of scatterplots to show the pairwise
#relationship between different variables in a dataset
#price and carat, carat and x,y,z etc are more correlated

library(ggplot2)
ggplot(data = diamonds,
       aes(x=carat, y=price, color=color, size=cut))+
  geom_point()+
  theme(legend.position="bottom")
```



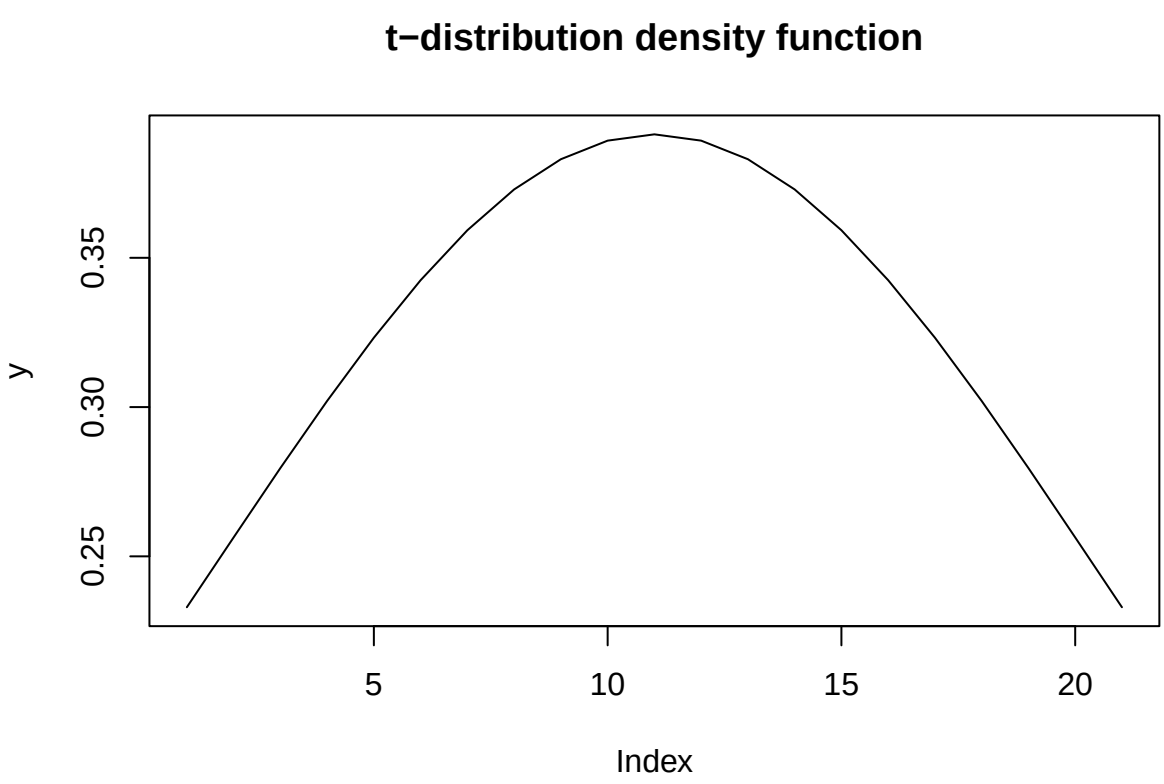

```
library(lattice)
xyplot(diamonds$carat~diamonds$price|diamonds$color*diamonds$cut)
```




```
#ex-3.2
#a)
set.seed (9)
sm<-seq(0, 2, length = 201)
sample (sm,100)
```

```
##   [1] 1.86 0.52 0.05 0.58 1.51 0.82 0.02 1.39 0.47 0.29 0.43 0.36 0.17 1.89 0.41
##  [16] 0.34 0.56 0.85 0.64 1.80 0.37 1.74 0.65 0.91 0.13 0.25 1.46 1.85 1.76 1.87
##  [31] 0.88 0.72 1.30 0.16 1.10 1.22 1.67 0.75 1.38 1.60 1.05 0.70 1.03 0.48 0.42
##  [46] 1.18 0.89 0.22 0.14 0.53 1.04 0.40 0.15 0.38 1.33 0.28 1.43 0.50 1.90 0.80
##  [61] 1.02 1.34 1.26 1.82 1.56 0.57 0.20 0.08 0.30 1.68 0.35 0.86 0.39 0.00 1.25
##  [76] 1.96 1.48 1.27 1.53 1.70 0.77 1.95 1.69 1.54 1.50 1.84 0.46 0.19 0.74 1.21
##  [91] 1.77 1.52 0.76 2.00 1.32 0.60 0.93 1.57 1.64 1.81
```

```
#b)
y<-dt(seq(-1, 1, length = 21),df=13)
plot(y, type = "l", main = "t-distribution density function")
```




```
#ex-3.3
#a)
x<- 1:20
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
#b)
w <- 1 + sqrt(x)/2
w
```

```
##  [1] 1.500000 1.707107 1.866025 2.000000 2.118034 2.224745 2.322876 2.414214
##  [9] 2.500000 2.581139 2.658312 2.732051 2.802776 2.870829 2.936492 3.000000
## [17] 3.061553 3.121320 3.179449 3.236068
```

```
#c)
set.seed(0)
dummy <- data.frame(x = x,
                    y = x + rnorm(x)*w)
df
```

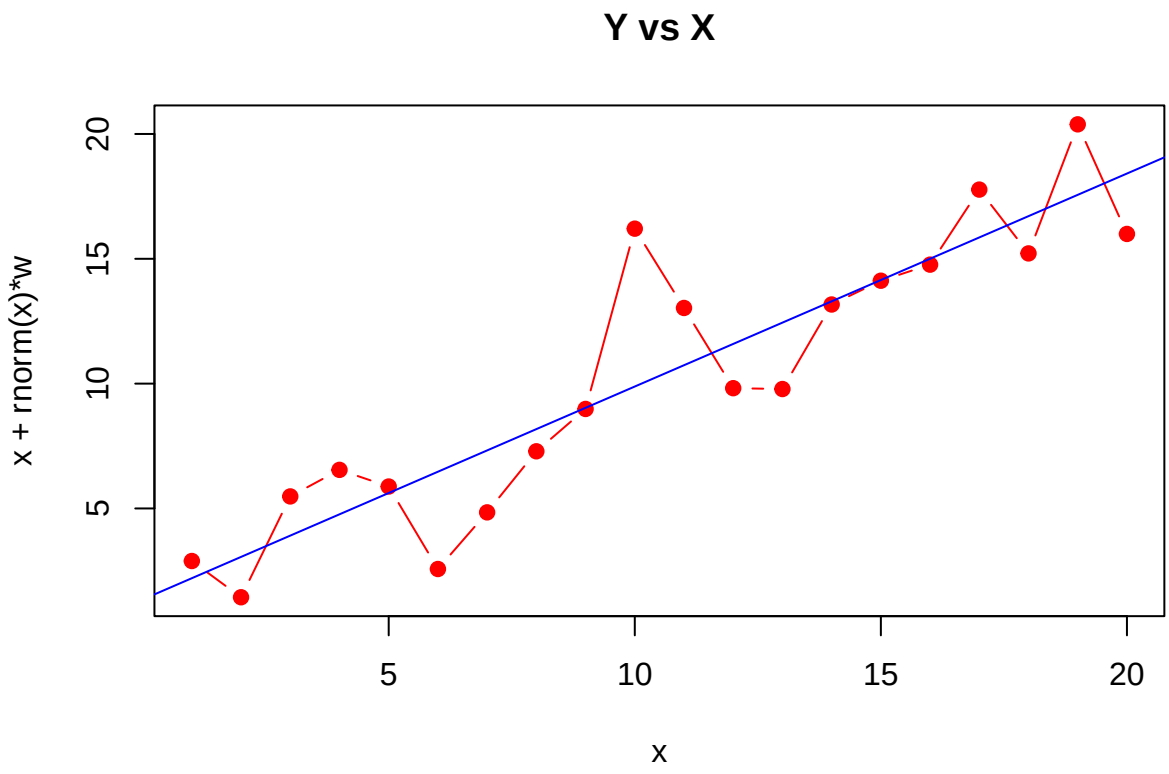
```
## function (x, df1, df2, ncp, log = FALSE)
## {
##     if (missing(ncp))
```

```
##         .Call(C_df, x, df1, df2, log)
##     else .Call(C_dnf, x, df1, df2, ncp, log)
## }
## <bytecode: 0x000001fbcea22d60>
## <environment: namespace:stats>
```

```
#d)
#To save a plot as jpeg image:
#need to call the function dev.off() after all the plotting,
#to save the file and return control to the screen
#This will save a jpeg image in the current directory
jpeg("histogram_boxplot.jpeg", quality = 75)
ly<- layout( matrix(c(1,2), ncol=2) )
hist(dummy$y, xlab="histogram of y",col=4,main="")
boxplot(dummy$y ,col=3, xlab="boxplot of y")
dev.off()
```

```
## pdf
##   2
```

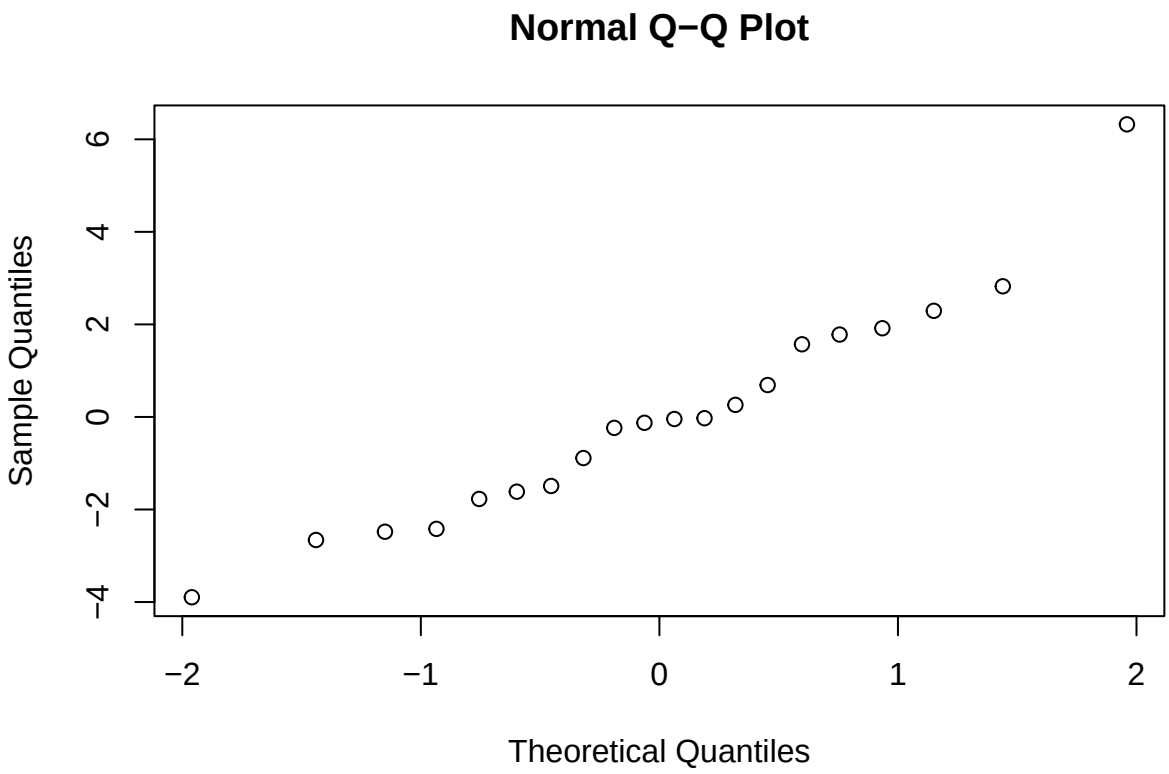
```
#e)
plot(dummy$x , dummy$y,type = "b", pch = 19,
     col = "red",
     ylab="x + rnorm(x)*w"  , xlab="x", main="Y vs X")

#f)
fm <- lm(y~x, data=dummy)
# adding a regression line
abline(fm,col='blue')
```




```
#g)
#residual of a regression
#The difference between an observed value of the response variable
#and the value of the response variable predicted from the regression line.
resids <- resid(fm)
resids
```

```
##           1           2           3           4           5           6
##  0.68944904 -1.61502966  1.57019264  1.78047999  0.26071390 -3.89663879
##           7           8           9          10          11          12
## -2.48072069 -0.88842512 -0.04445704  6.32357304  2.29356667 -1.77236924
##          13          14          15          16          17          18
## -2.65919257 -0.12669415 -0.02747435 -0.23649624  1.91709958 -1.49219948
##          19          20
##  2.82387303 -2.41925056
```

```
#The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess
#if a set of data plausibly
#came from some theoretical distribution such as a Normal or exponential
#A Q-Q plot is a scatterplot created by plotting two sets of quantiles against
#one another. If both sets of quantiles came from the same distribution,
#we should see the points forming a line that's roughly straight.
qqnorm(resids)
```




```
#In the normal Q-Q plot, x-axis plots the theoretical quantiles from the
#standard Normal distribution with mean 0 and standard deviation 1 and since
# the points are almost making a straight line,so we can say that the residuals
#are normally distributed
```