UBS - Code Assessment

Exercise 2 - Outlier detection and statistics

In [4]:
```
library(data.table)
library(stringr)
library(ggplot2)
```

2.A) Load the data of the csv file into a data format of your choice

I) Data Gathering

In [10]:
```
raw_data <- read.csv('/Users/alvarosanchezfernandez/Documents/Programming/Raw Data/da
raw_data
```

Description	X199501	X199502	X199503	X199504	X199505	X199506	X199507	X199508	X1
GDP Level	NA	NA	179700.0	NA	NA	177070.000	NA	NA	1.74
LIBOR 1M %	3.8125	3.625	3.5	3.4375	3.25	3.125	2.625	2.875	2.46
Equities Index 1 (index points)	3017.3000	3041.200	3143.1	3220.4000	3340.60	3323.700	3449.900	3509.400	3.52
Equities Index 2 (index points)	NA	NA	NA	NA	NA	NA	NA	100.000	8.30
Equities Index 3 (index points)	NA	NA	NA	NA	NA	NA	NA	NA	NA

II) Data Cleaning

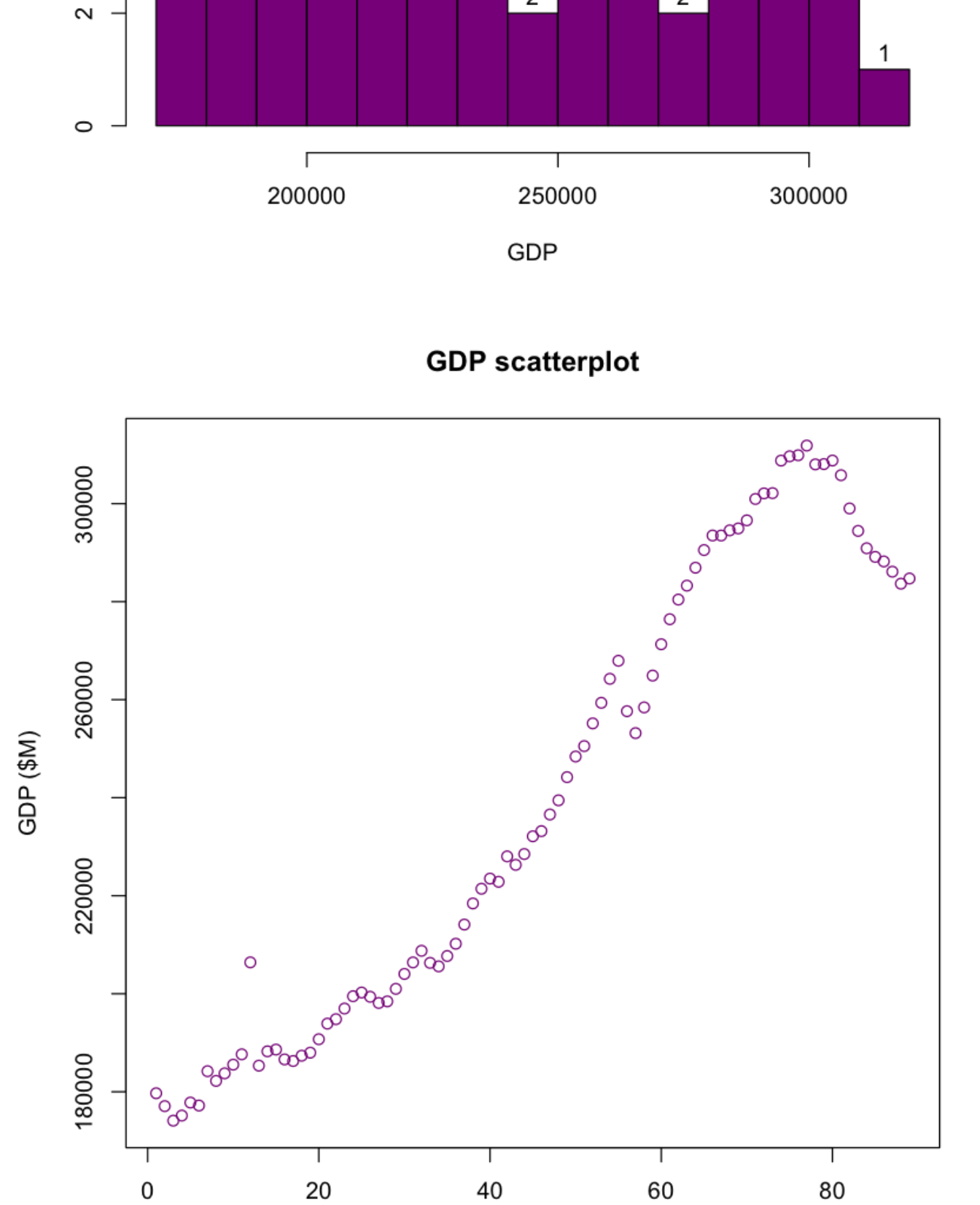
In [11]:
```
# Create Dataframe Organised Vertically
df <- transpose(raw_data)

# Set Row Names
rownames(df) <- colnames(raw_data)

# Set Column Names
colnames(df) <- df[1,]
df <- df[-1, ]
df

# Eliminate NA ?
```

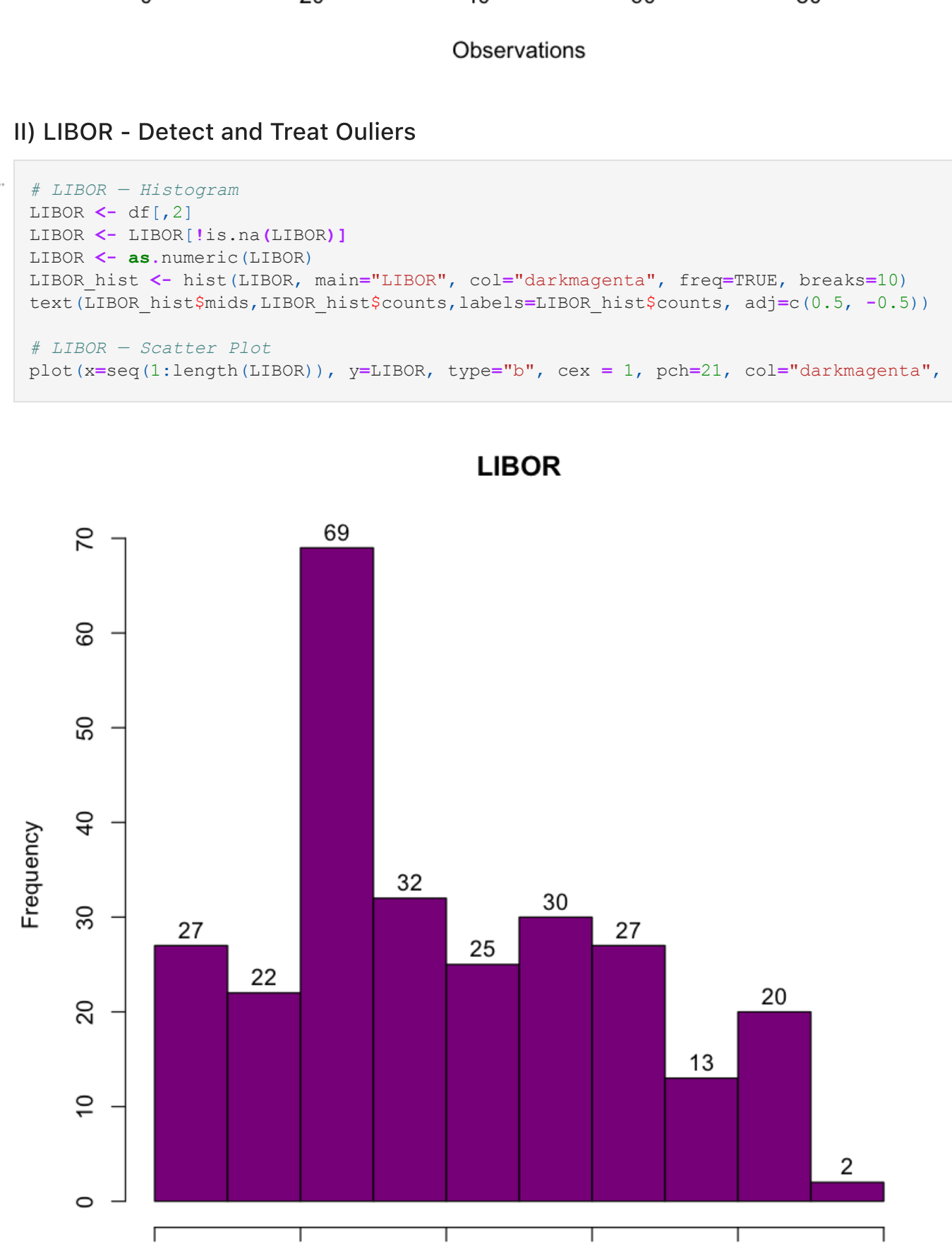
	GDP Level	LIBOR 1M %	Equities Index 1 (index points)	Equities Index 2 (index points)	Equities Index 3 (index points)
X199501	NA	3.8125	3017.3	NA	NA
X199502	NA	3.625	3041.2	NA	NA
X199503	179700	3.5	3143.1	NA	NA
X199504	NA	3.4375	3220.4	NA	NA
X199505	NA	3.25	3340.6	NA	NA
X199506	177070	3.125	3323.7	NA	NA
X199507	NA	2.625	3449.9	NA	NA
X199508	NA	2.875	3509.4	100	NA
X199509	174104	2.46875	3520.2	83	NA
X199510	NA	2	3518.7	72.7	NA
X199511	NA	2.3125	3680.4	70.95	NA
X199512	175146	1.8125	3689.3	82.92	NA
X199601	NA	1.76563	3752.8	81.19	NA
X199602	NA	1.57813	3752.7	70.23	NA
X199603	177800	1.76563	3718.4	77.56	NA
X199604	NA	1.875	3806	104.28	NA
X199605	NA	2.70313	3739.2	149.15	NA
X199606	177186	2.5	3725.6	198.63	NA
X199607	NA	2.5	3734.4	158.05	NA
X199608	NA	2.11719	3684.4	185.52	NA
X199609	184197	1.39063	3992.2	165.57	NA
X199610	NA	1.67969	3948.5	178.64	NA
X199611	NA	2.125	4038.5	189.73	NA
X199612	182216	1.8125	4118.5	200.5	NA
X199701	NA	1.71094	4257.8	276.79	NA
X199702	NA	1.75	4307.1	322.31	NA
X199703	183758	1.875	4248.1	298.52	NA
X199704	NA	1.73438	4445	321.19	NA
X199705	NA	0.98828	4562.8	355.46	NA
X199706	185535	1.375	4728.3	423.87	NA
⋮	⋮	⋮	⋮	⋮	⋮
X201410	NA	0.002	6487.97	1078.33	99.91
X201411	NA	-0.003	6656.37	958.33	107.93
X201412	308793	-0.038	6566.09	790.71	92
X201501	NA	-0.904	6782.55	745.82	61.46
X201502	NA	-0.873	6940.64	903.56	70.49
X201503	305796	-0.851	6809.5	910.41	900.9
X201504	NA	-0.815	6985.95	1029.31	870.03
X201505	NA	-0.821	6953.58	953.88	967.37
X201506	299015	-0.832	6608.59	930.66	1000.56
X201507	NA	-0.787	6688.62	834.89	903.4
X201508	NA	-0.781	6058.54	805.06	872.03
X201509	294431	-0.792	6072.47	781.65	844.9
X201510	NA	-0.774	6361.8	853.77	781.56
X201511	NA	-0.848	6395.65	839.14	718.64
X201512	290892	-0.798	6242.32	757.04	730.56
X201601	NA	-0.789	6060.1	730.39	638.54
X201602	NA	-0.8332	6152.88	786.36	621.93
X201603	289127	-0.7826	6146.05	861.96	579.86
X201604	NA	-0.779	6241.89	951.11	551.22
X201605	NA	-0.7746	6191.93	890.52	510.23
X201606	288202	-0.8286	6577.83	933.32	539.47
X201607	NA	-0.7976	6693.95	922.53	566.47
X201608	NA	-0.8108	6745.97	944.96	560.73
X201609	286118	-0.802	6983.52	1004.69	561.01
X201610	NA	-0.7854	6917.14	996.63	604.5
X201611	NA	-0.8178	6752.93	1047.43	636.28
X201612	283665	-0.799	7142.83	1152.33	649.48
X201701	NA	-0.7884	7107.65	1167.53	712.27
X201702	NA	-0.8008	7382.9	1109.39	734.84
X201703	284722.938	-0.7792	7282.69	1122.98	664.46

2.B) Detect potential outliers for each time series based on a simple methodology

I) GDP - Detect and Treat Ouliers

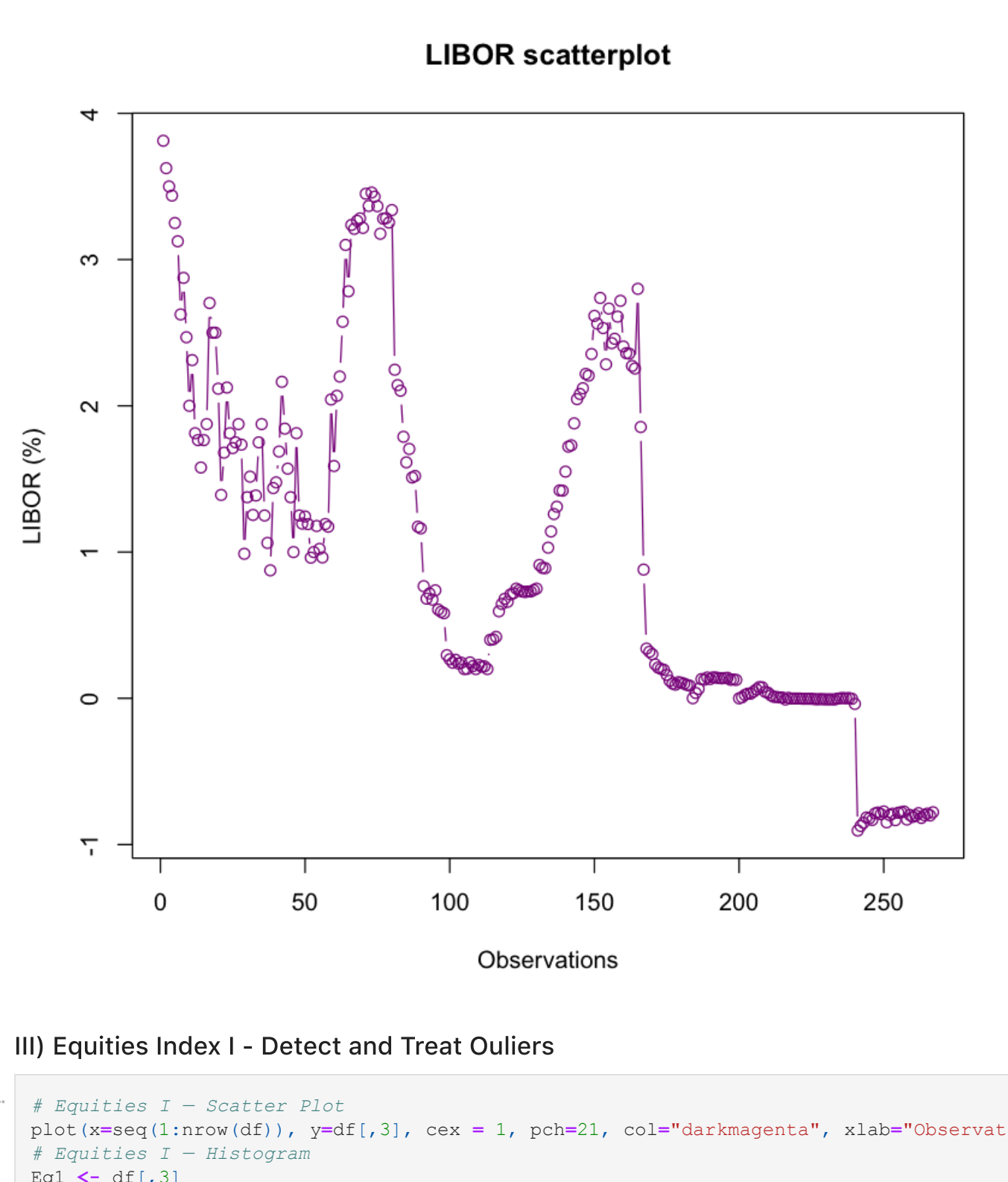
In [123]:
```
# Detect Outliers Visually

# GDP - Histogram
GDP <- df[,1]
GDP <- GDP[!is.na(GDP)]
GDP <- as.numeric(GDP)
GDP_hist <- hist(GDP, main="GDP Histogram", col="darkmagenta", freq=TRUE, breaks=10)
text(GDP_hist$mids,GDP_hist$counts,labels=GDP_hist$counts, adj=c(0.5, -0.5))

# GDP - Scatter Plot
plot(x=seq(1:length(GDP)), y=GDP, type="p",cex = 1, pch=21, col="darkmagenta", xlab="
```

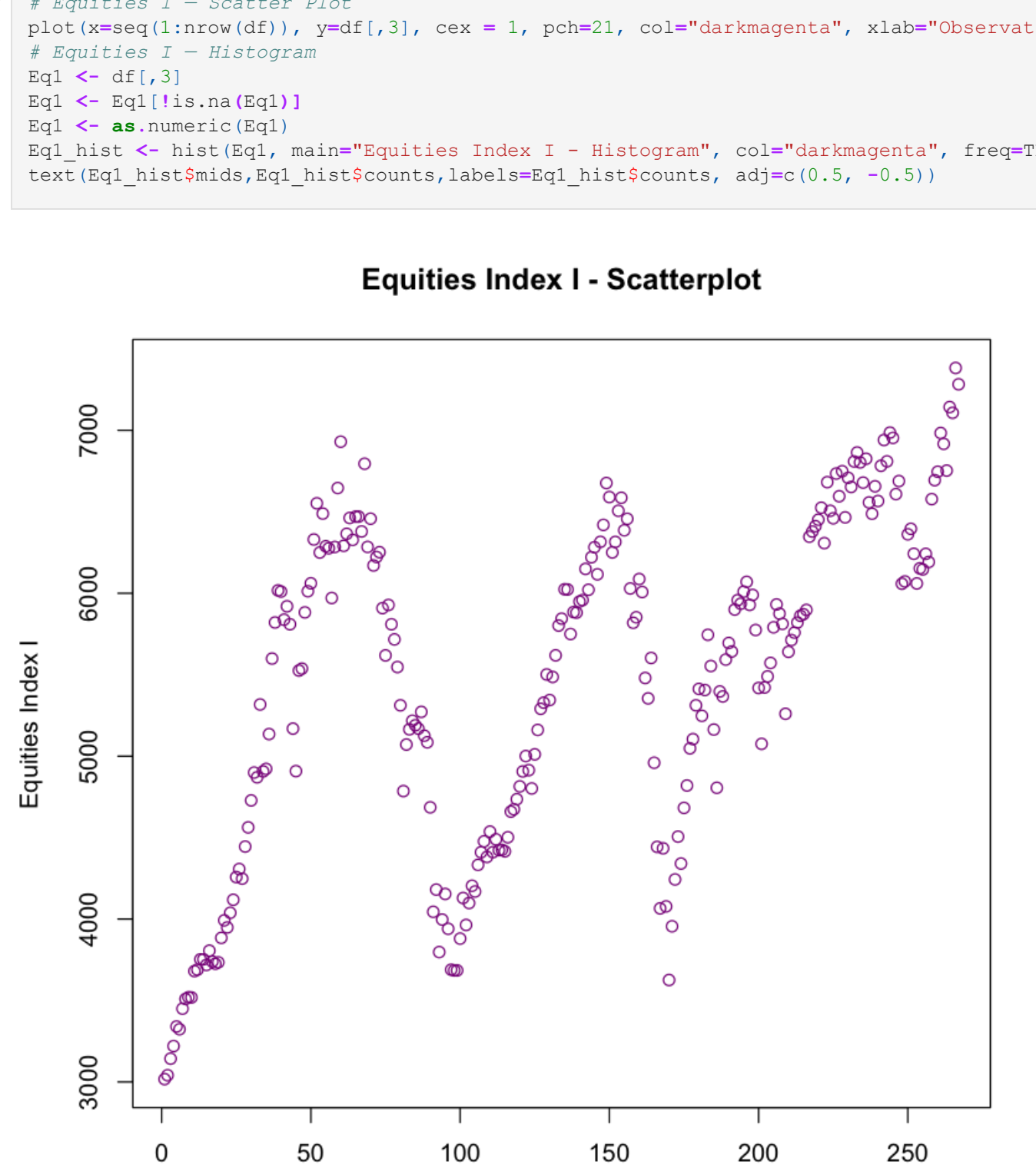




II) LIBOR - Detect and Treat Ouliers

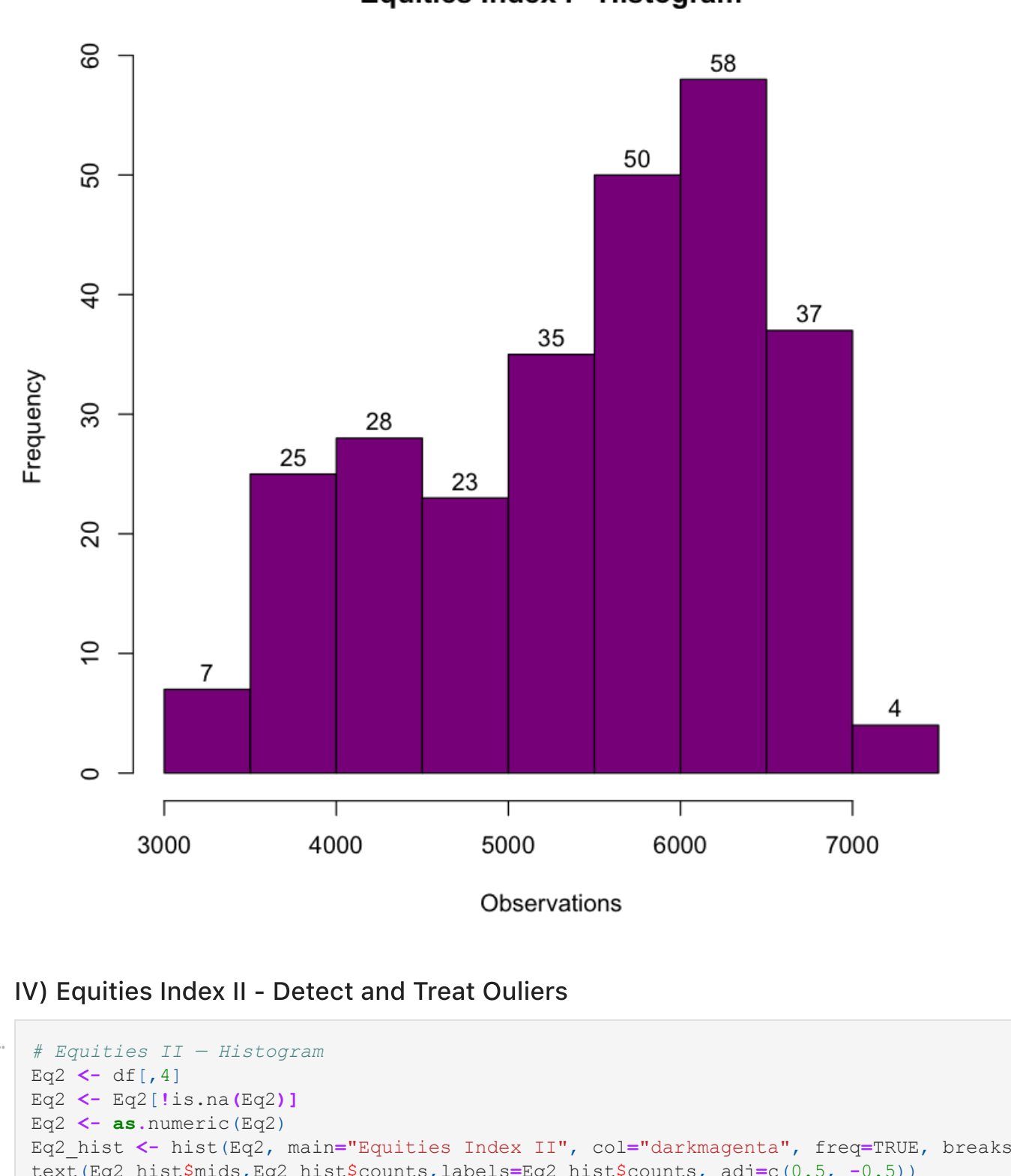
In [127]:
```
# LIBOR - Histogram
LIBOR <- df[,2]
LIBOR <- LIBOR[!is.na(LIBOR)]
LIBOR <- as.numeric(LIBOR)
LIBOR_hist <- hist(LIBOR, main="LIBOR", col="darkmagenta", freq=TRUE, breaks=10)
text(LIBOR_hist$mids,LIBOR_hist$counts,labels=LIBOR_hist$counts, adj=c(0.5, -0.5))

# LIBOR - Scatter Plot
plot(x=seq(1:length(LIBOR)), y=LIBOR, type="b", cex = 1, pch=21, col="darkmagenta", x
```

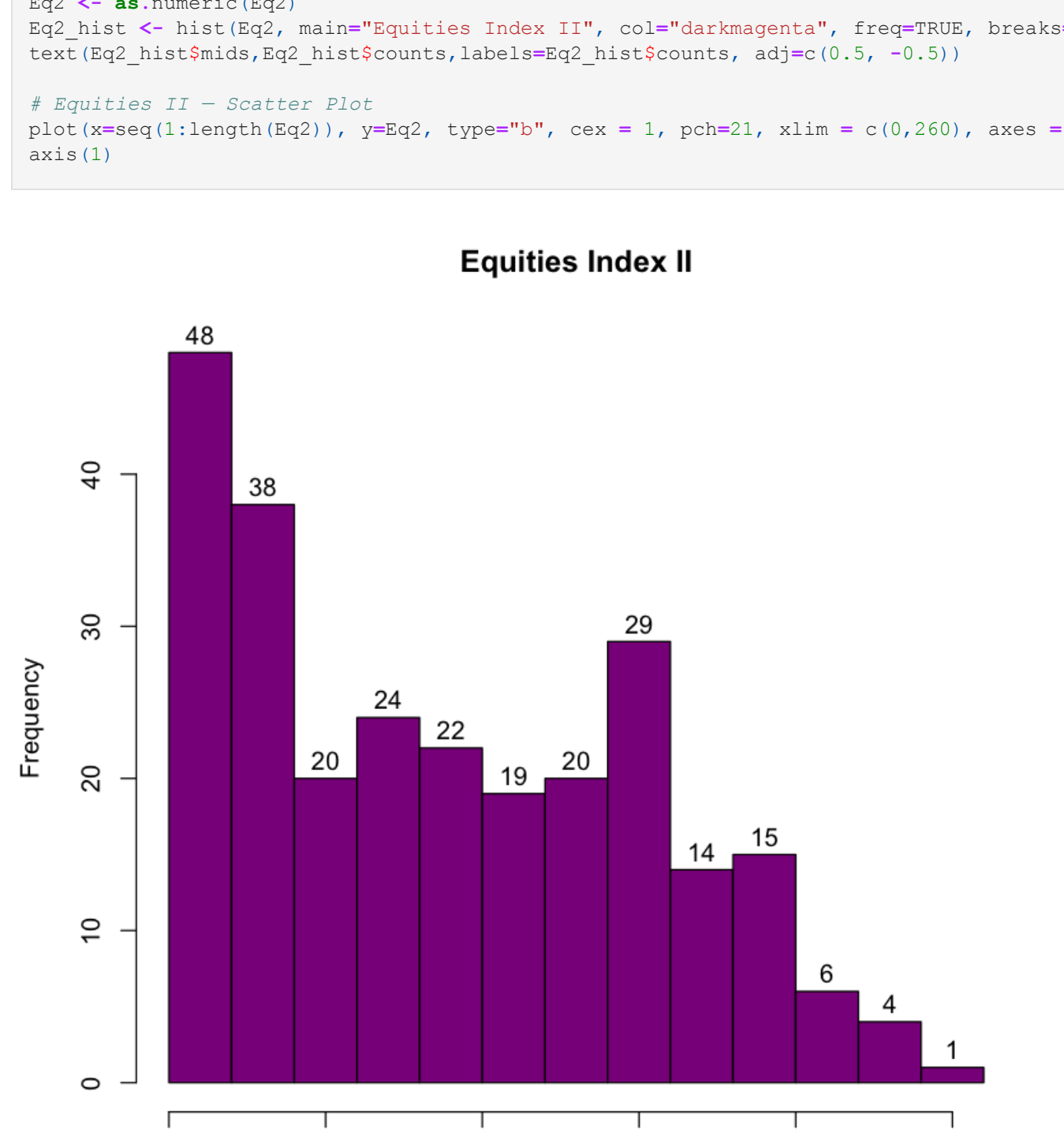




III) Equities Index I - Detect and Treat Ouliers

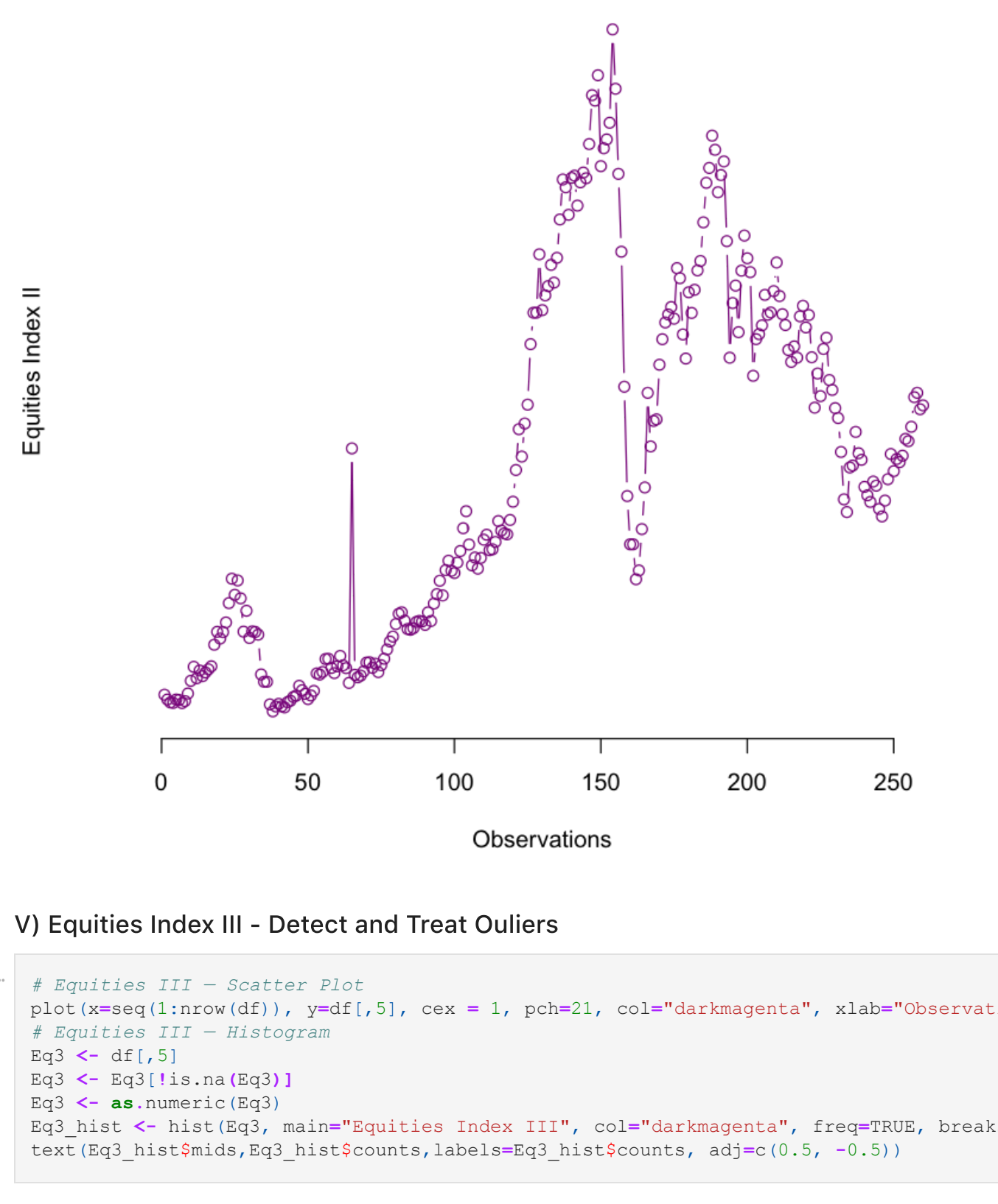
In [231]:
```
# Equities I - Scatter Plot
plot(x=seq(1:nrow(df)), y=df[,3], cex = 1, pch=21, col="darkmagenta", xlab="Observati
# Equities I - Histogram
Eq1 <- df[,3]
Eq1 <- Eq1[!is.na(Eq1)]
Eq1 <- as.numeric(Eq1)
Eq1_hist <- hist(Eq1, main="Equities Index I - Histogram", col="darkmagenta", freq=TRU
text(Eq1_hist$mids,Eq1_hist$counts,labels=Eq1_hist$counts, adj=c(0.5, -0.5))
```

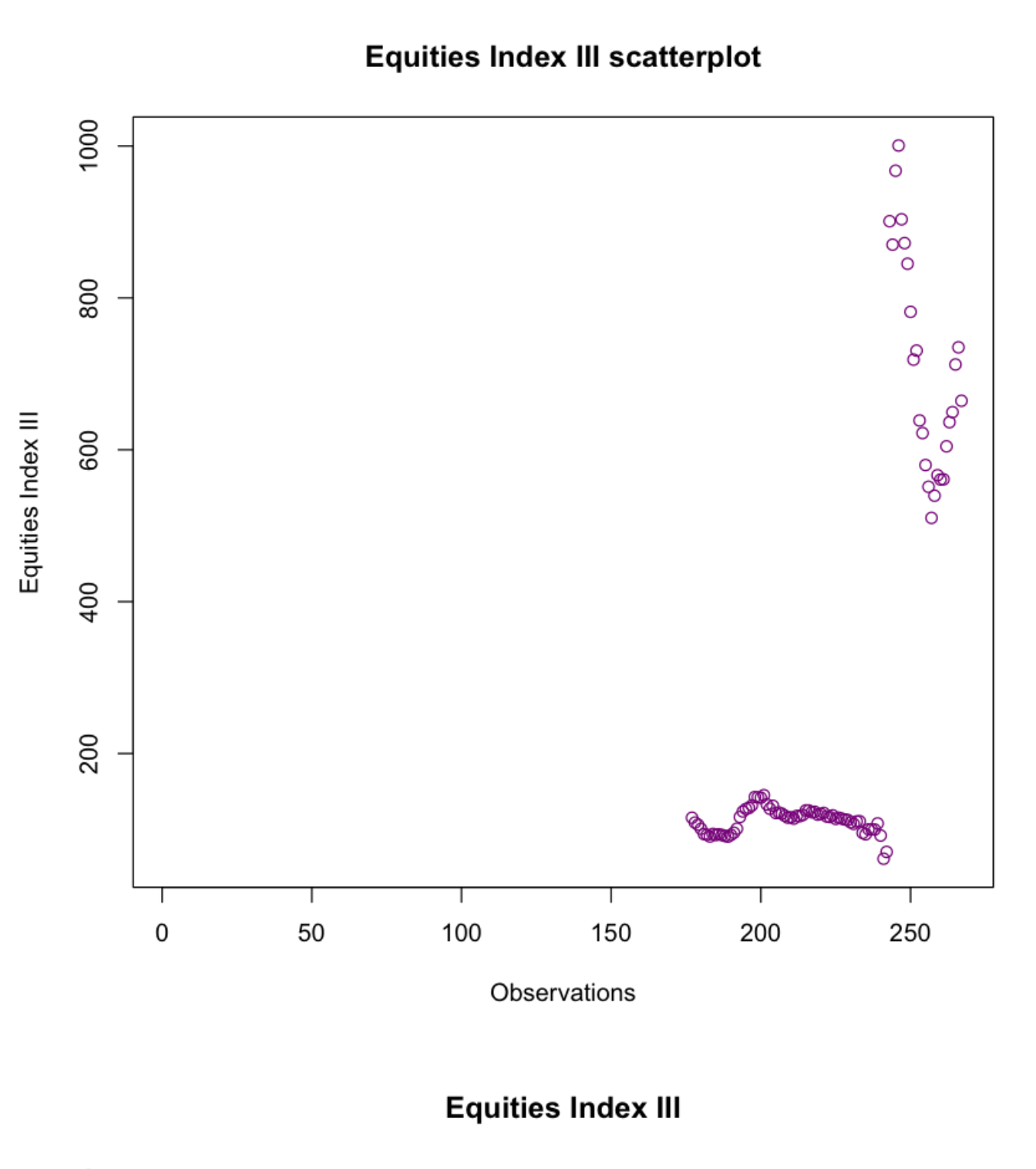




IV) Equities Index II - Detect and Treat Ouliers

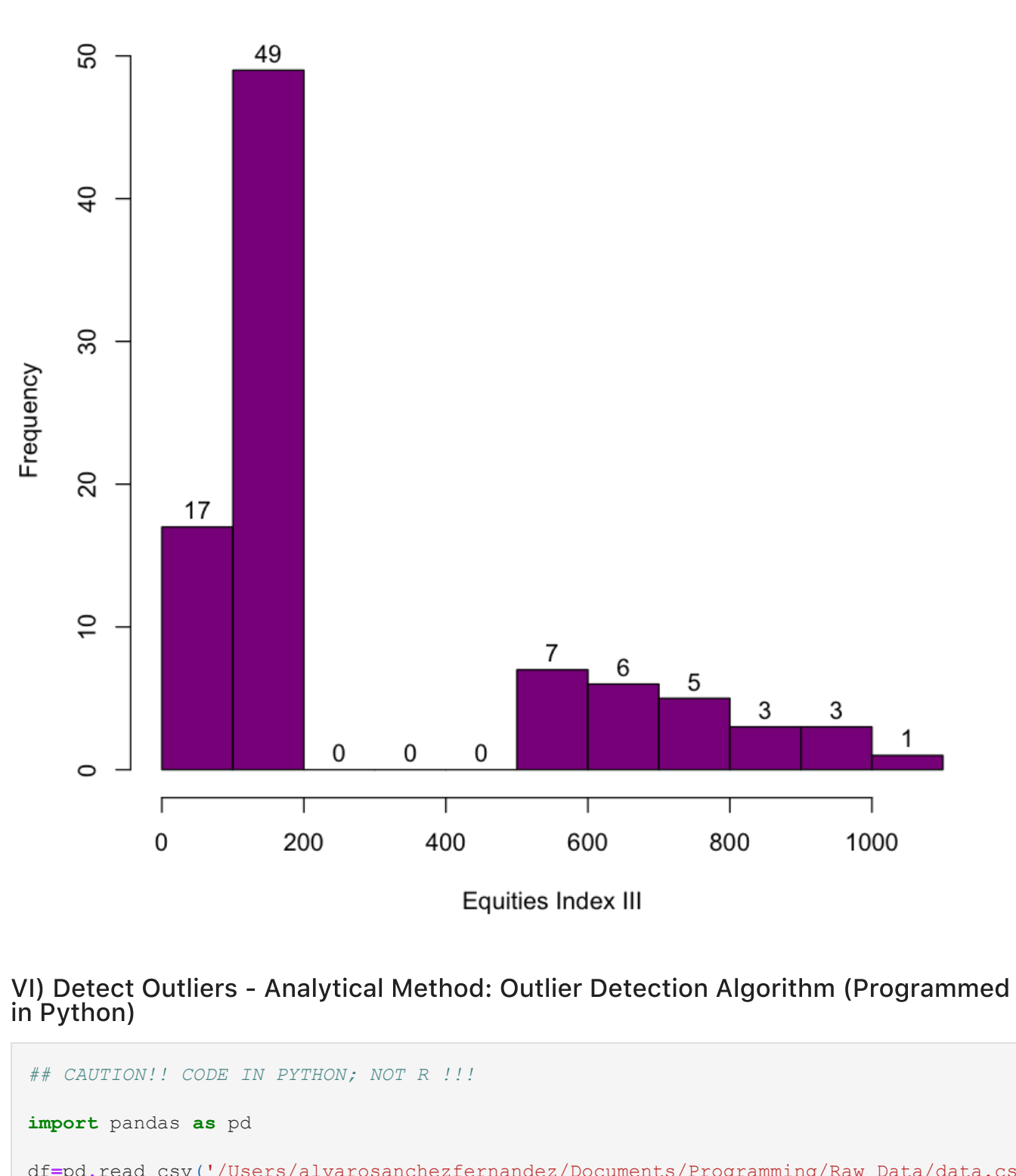
In [140]:
```
# Equities II - Histogram
Eq2 <- df[,4]
Eq2 <- Eq2[!is.na(Eq2)]
Eq2 <- as.numeric(Eq2)
Eq2_hist <- hist(Eq2, main="Equities Index II", col="darkmagenta", freq=TRUE, breaks=
text(Eq2_hist$mids,Eq2_hist$counts,labels=Eq2_hist$counts, adj=c(0.5, -0.5))

# Equities II - Scatter Plot
plot(x=seq(1:length(Eq2)), y=Eq2, type="p", cex = 1, pch=21, xlim = c(0,260), xaxs =
axis(1)
```





V) Equities Index III - Detect and Treat Ouliers

In [233]:
```
# Equities III - Scatter Plot
plot(x=seq(1:nrow(df)), y=df[,5], cex = 1, pch=21, col="darkmagenta", xlab="Observatio
# Equities III - Histogram
Eq3 <- df[,5]
Eq3 <- Eq3[!is.na(Eq3)]
Eq3 <- as.numeric(Eq3)
Eq3_hist <- hist(Eq3, main="Equities Index III", col="darkmagenta", freq=TRUE, breaks
text(Eq3_hist$mids,Eq3_hist$counts,labels=Eq3_hist$counts, adj=c(0.5, -0.5))
```



VI) Detect Outliers - Analytical Method: Outlier Detection Algorithm (Programmed in Python)

In [2]:
```
## CAUTION!! CODE IN PYTHON; NOT R !!!

import pandas as pd

df=pd.read_csv('/Users/alvarosanchezfernandez/Documents/Programming/Raw Data/data.csv
df_list = df.values.tolist()

GDP=df_list[0]
GDP=GDP[1:len(GDP)]
GDP = [x for x in GDP if pd.isnull(x) == False]
GDP_vect=[]
GDP_dx=(max(GDP)-min(GDP))/(len(GDP))
for i in range(len(GDP)):
    GDP_vect.append([GDP[i],i*GDP_dx])
#print (GDP_vect)

LIBOR=df_list[1]
LIBOR=LIBOR[1:len(LIBOR)]
LIBOR_vect=[]
LIBOR_dx=(max(LIBOR)-min(LIBOR))/(len(LIBOR))
for i in range(len(LIBOR)):
    LIBOR_vect.append([LIBOR[i],i*LIBOR_dx])
#print (LIBOR_vect)

Eq1=df_list[2]
Eq1=Eq1[1:len(Eq1)]
Eq1_vect=[]
Eq1_dx=(max(Eq1)-min(Eq1))/(len(Eq1))
for i in range(len(Eq1)):
    Eq1_vect.append([Eq1[i],i*Eq1_dx])
#print (Eq1_vect)

Eq2=df_list[3]
Eq2=Eq2[1:len(Eq2)]
Eq2 = [x for x in Eq2 if pd.isnull(x) == False]
Eq2_vect=[]
Eq2_dx=(max(Eq2)-min(Eq2))/(len(Eq2))
for i in range(len(Eq2)):
    Eq2_vect.append([Eq2[i],i*Eq2_dx])
#print (Eq2_vect)

Eq3=df_list[4]
Eq3=Eq3[1:len(Eq3)]
Eq3 = [x for x in Eq3 if pd.isnull(x) == False]
Eq3_vect=[]
Eq3_dx=(max(Eq3)-min(Eq3))/(len(Eq3))
for i in range(len(Eq3)):
    Eq3_vect.append([Eq3[i],i*Eq3_dx])
#print (Eq3_vect)
```

In [5]:
```
## CAUTION!! CODE IN PYTHON; NOT R !!!

from sklearn.metrics.pairwise import euclidean_distances
from sklearn.cluster import DBSCAN
import pandas as pd
import numpy as np

def DBSCAN_alg (eps, M, dataset_vector):
    print("Outlier Detection Algorithm:")

    mDistance = euclidean_distances(dataset_vector)
    clustering = density_clustering_DBSCAN(mDistance,eps,M)

    return clustering


def density_clustering_DBSCAN(mDistance, eps, M):
    labels = np.zeros(mDistance.shape[0])
    #print(labels)
    C = 0

    for i in np.arange(mDistance.shape[0]):

        if labels[i] != 0:
            continue

        vecindario = find_vecindario(mDistance, i, eps)
        #print('Vecindario:', vecindario)  # Uncomment to see output

        if len(vecindario)<M:
            # If it is NOT a nuclear point
            labels[i] = -1
        else:
            # If it is a nuclear point, a new C is assigned and we try to expand it
            C += 1
            labels[i] = C
            expand_cluster(mDistance, labels, vecindario, C, eps, M)
            #print(labels)

    return labels

def expand_cluster(mDistance, labels, vecindario, C, eps, M):
    it = 0
    while it < vecindario.size:
        j = vecindario[it]

        if labels[j] == -1:
            # If element has been labelled as no-nuclear
            labels[j] = C
        elif labels[j] == 0:
            # If element has not been labelled yet
            labels[j] = C
            # Calculation of vecindario
            vecindario_j = find_vecindario(mDistance, j, eps)
            if vecindario_j.size >= M:
                # If "j" is a nuclear point, it is mingledd with its neighbours
                # with the neighbourhood that we are analyzing
                vecindario = np.append(vecindario, vecindario_j)
        it += 1

def find_vecindario(mDistance, i, eps):
    vecinos = np.where(mDistance[i,:] < eps)[0]

    vecinos = vecinos[vecinos != i]

    return vecinos


# IMPORTANT! Dataset must have constant volatility
def outlier_detection (dataset, name):
    # I) Configuration of x-axis scale: x=range(y)
    #    between consecutive elements of the dataset
    dx =(max(dataset)-min(dataset))/(len(dataset))

    # II) Calculation of difference (delta)
    #    between consecutive elements of the dataset
    dataset_delta=[]
    for i in range(0,len(dataset)-1):
        dataset_delta.append(abs(dataset[i]-dataset[i+1]))

    # III) Calculation of the mean and standard deviation
    #     of the dataset differences
    dataset_delta_mean=np.mean(dataset_delta)
    dataset_delta_std=np.std(dataset_delta)**(1/2)

    # IV) Calculation of Chebyshev k: nº of std that
    #      contain at least 95% of observations
    k=(1/0.05)**(1/2)

    # V) Calculation of the epsilon: min distance
    #     between 2 points to be considered same cluster:
    #        Eps = 2*(dx^2 + 95% CI of daily variability)^(1/2)
    eps=2*(dx**2+(dataset_delta_mean+dataset_delta_std*k)**2)**(1/2)

    # VI) Determination of M: min 3 points within Eps. distance
    M=3

    # VII) Clustering process
    # Creation of the Dataset vector: [y,i*dx]
    dataset_vect=[]
    for i in range(len(dataset)):
        dataset_vect.append([dataset[i],i*dx])
    #clustering = DBSCAN (eps=eps, min_samples=M).fit(dataset_vect)
    clustering = DBSCAN_alg (eps,M,dataset_vect)

    # VIII) Printing Results
    print (name+" Outliers:")
    print (" - Delta mean:",dataset_delta_mean)
    print (" - Delta std:",dataset_delta_std)
    print (" - k: ",k)
    print (" - Epsilon: ",eps)
    print (" - M: ",M)
    #print (clustering.labels_)
    print(clustering)
    print ()

    return clustering

cluster_GDP   = outlier_detection (GDP, "GDP")
cluster_LIBOR = outlier_detection (LIBOR, "LIBOR")
cluster_Eq1   = outlier_detection (Eq1, "Equities I")
cluster_Eq2   = outlier_detection (Eq2, "Equities II")
cluster_Eq3   = outlier_detection (Eq3, "Equities III")
```

```
Outlier Detection Algorithm:
GDP Outliers:
 - Delta mean: 3080.262931818182
 - Delta std: 56.41750591131117
 - k:  4.47213595499958
 - Epsilon:  7348.799001038649
 - M:  3
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.
  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  3.  3.  3.  3.  3.  3.  3.]

Outlier Detection Algorithm:
LIBOR Outliers:
 - Delta mean: 0.12706601503759402
 - Delta std: 0.44290452184477486
 - k:  4.47213595499958
 - Epsilon:  4.215738544315322
 - M:  3
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1.]

Outlier Detection Algorithm:
Equities I Outliers:
 - Delta mean: 170.56627819548874
 - Delta std: 11.987532911705578
 - k:  4.47213595499958
 - Epsilon:  449.54327669403915
 - M:  3
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1.]

Outlier Detection Algorithm:
Equities II Outliers:
 - Delta mean: 77.20996138996139
```

```
 - Delta std: 10.03284466137934
 - k:  4.47213595499958
 - Epsilon:  244.8601224491558
 - M:  3
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.]

Outlier Detection Algorithm:
Equities III Outliers:
 - Delta mean: 23.07144444444444
 - Delta std: 9.398752236818474
 - k:  4.47213595499958
 - Epsilon:  131.83354900643172
 - M:  3
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
```

VII) Data-frame - Outliers Treatment

In [12]:

```
# Approach 1: Eliminate them if they are human error
# Approach 2: Eliminate abnormal (non-statistical significant) outliers
df_clean <- df[-36,]; # Clean Dataset From Outlier of GDP list
df_clean <- df[-72,]; # Clean Dataset From Outliers of Equity II list

# Approach 3: Winsorize abnormal outliers (we set them equal to the x% percentil) (in
# Approach 4: Do nothing (they hold statistical significance)
```

2.C) Provide a measure of correlation for the different variables at hand

In [13]:

```
# Change Dataset to numeric values
df_clean_num <- as.data.frame(apply(df_clean, 2, as.numeric))
sapply(df_clean_num, class)

# Calculate Correlation Matrix
#cor(df[sapply(df, function(x) !is.factor(x))]) # Function useful to calculate correla
cor(df_clean_num, use="pairwise.complete.obs")
```

GDP Level 'numeric'
LIBOR 1M % 'numeric'
Equities Index 1 (index points) 'numeric'
Equities Index 2 (index points) 'numeric'
Equities Index 3 (index points) 'numeric'

	GDP Level	LIBOR 1M %	Equities Index 1 (index points)	Equities Index 2 (index points)	Equities Index 3 (index points)
GDP Level	1.00000000	-0.6771155	0.5599311	0.7350430	-0.06735564
LIBOR 1M %	-0.67711550	1.0000000	-0.2646416	-0.2584438	-0.88807372
Equities Index 1 (index points)	0.55993115	-0.2646416	1.0000000	0.4201413	0.45426347
Equities Index 2 (index points)	0.73504303	-0.2584438	0.4201413	1.0000000	-0.65223651
Equities Index 3 (index points)	-0.06735564	-0.8880737	0.4542635	-0.6522365	1.00000000

2.D) Build a simple model that explains Equity 2 by one or more other variables provided. Provide an assessment of the out-of-sample performance for your model for the out-of-sample period 2015/01 to 2017/03

I) Model Generation

In [37]:

```
# Eliminate dataset values from 2015 to 2017 (out-of-sample period)
df_regression <- df_clean_num[-c(240:nrow(df_clean_num)), ]
#df_regression

# Creating variable names
GDP <- df_regression[,1]
LIBOR <- df_regression[,2]
EQ1 <- df_regression[,3]
EQ2 <- df_regression[,4]
EQ3 <- df_regression[,5]

# Benchmark Regression (Eq.II = b0 + b1*GDP)
lm_case1 = lm(EQ2~GDP)
anova(lm_case1)
summary (lm_case1)

# Check whether adding Eq.1 improves regression results
lm_case2 = lm(EQ2~GDP+EQ1)
anova(lm_case2)
summary (lm_case2)

# Check whether adding Eq.3 improves regression results
lm_case3 = lm(EQ2~GDP+EQ3)
anova(lm_case3)
summary (lm_case3)

# Check whether eliminating any parameter improves regression results
lm_case4 = lm(EQ2~GDP+EQ1+EQ3)
anova(lm_case4)
summary (lm_case4)

# Check whether adding Libor improves regression results
lm_case5 = lm(EQ2~GDP+LIBOR)
anova(lm_case5)
summary (lm_case5)

# Check whether adding Eq.1 improves regression results
lm_case6 = lm(EQ2~GDP+LIBOR+EQ1)
anova(lm_case6)
summary (lm_case6)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	21051082	21051082.4	122.9958	1.791297e-17
Residuals	75	12836465	171152.9	NA	NA

```
Call:
lm(formula = EQ2 ~ GDP)

Residuals:
   Min     1Q Median     3Q    Max 
-960.9 -231.1 -129.0  111.5 1170.0 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.882e+03  2.545e+02  -7.396 1.67e-10 ***
GDP          1.177e-02  1.061e-03  11.090  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 413.7 on 75 degrees of freedom
  (162 observations deleted due to missingness)
Multiple R-squared:  0.6212,	Adjusted R-squared:  0.6162 
F-statistic:   123 on 1 and 75 DF,  p-value: < 2.2e-16
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
EQ1	1	20561054	20561054.2	57.50279	8.274701e-13
Residuals	230	82240224	357566.2	NA	NA

```
Call:
lm(formula = EQ2 ~ EQ1)

Residuals:
     Min       1Q   Median       3Q      Max 
-1202.86  -287.50     7.06   493.57  1371.77 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -853.84758  232.38366  -3.674 0.000297 ***
EQ1            0.32206    0.04247   7.583 8.27e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 598 on 230 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared:    0.2,	Adjusted R-squared:  0.1965 
F-statistic:  57.5 on 1 and 230 DF,  p-value: 8.275e-13
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	21051082.4	21051082.4	125.555198	1.325578e-17
EQ1	1	429331.1	429331.1	2.560664	1.138135e-01
Residuals	74	12407133.4	167664.0	NA	NA

```
Call:
lm(formula = EQ2 ~ GDP + EQ1)

Residuals:
     Min       1Q   Median       3Q      Max 
-1003.86  -276.76   -57.19   138.31  1090.27 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.163e+03  3.068e+02  -7.048 7.99e-10 ***
GDP          1.090e-02  1.182e-03   9.219 6.42e-14 ***
EQ1          9.011e-02  5.631e-02   1.600    0.114    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 409.5 on 74 degrees of freedom
  (162 observations deleted due to missingness)
Multiple R-squared:  0.6339,	Adjusted R-squared:  0.624 
F-statistic: 64.06 on 2 and 74 DF,  p-value: < 2.2e-16
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	125271.2	125271.2	2.040670	0.16937586
EQ3	1	249208.1	249208.1	4.059604	0.05829403
Residuals	19	1166358.7	61387.3	NA	NA

```
Call:
lm(formula = EQ2 ~ GDP + EQ3)

Residuals:
    Min      1Q  Median      3Q     Max 
-420.17 -119.61   22.15  136.31  502.35 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)  
(Intercept)  2.659e+03  1.207e+03   2.204   0.0401 *
GDP         -6.738e-03  4.036e-03  -1.670   0.1114  
EQ3          6.875e+00  3.412e+00   2.015   0.0583 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 247.8 on 19 degrees of freedom
  (217 observations deleted due to missingness)
Multiple R-squared:  0.243,	Adjusted R-squared:  0.1634 
F-statistic:  3.05 on 2 and 19 DF,  p-value: 0.07099
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	125271.22	125271.22	2.4660101	0.133744785
EQ1	1	49261.12	49261.12	0.9697233	0.337801736
EQ3	1	451920.99	451920.99	8.8962309	0.007980661
Residuals	18	914384.76	50799.15	NA	NA

```
Call:
lm(formula = EQ2 ~ GDP + EQ1 + EQ3)

Residuals:
    Min      1Q  Median      3Q     Max 
-478.81 -123.81   25.84  133.55  366.13 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept) 3967.64760 1245.07832   3.187  0.00511 **
GDP           -0.01996    0.00698  -2.860  0.01042 * 
EQ1            0.36997    0.16612   2.227  0.03894 * 
EQ3           10.37954    3.47996   2.983  0.00798 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 225.4 on 18 degrees of freedom
  (217 observations deleted due to missingness)
Multiple R-squared:  0.4066,	Adjusted R-squared:  0.3077 
F-statistic: 4.111 on 3 and 18 DF,  p-value: 0.02192
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	21051082	21051082.4	147.4788	2.726565e-19
LIBOR	1	2273729	2273728.7	15.9292	1.532950e-04
Residuals	74	10562736	142739.7	NA	NA

```
Call:
lm(formula = EQ2 ~ GDP + LIBOR)

Residuals:
    Min      1Q  Median      3Q     Max 
-928.19 -213.49  -55.64  241.73  836.58 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.774e+03  3.223e+02  -8.605 9.28e-13 ***
GDP          1.457e-02  1.197e-03  12.173  < 2e-16 ***
LIBOR        2.095e+02  5.248e+01   3.991 0.000153 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 377.8 on 74 degrees of freedom
  (162 observations deleted due to missingness)
Multiple R-squared:  0.6883,	Adjusted R-squared:  0.6799 
F-statistic:  81.7 on 2 and 74 DF,  p-value: < 2.2e-16
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GDP	1	2.105108e+07	2.105108e+07	1.454861e+02	4.748399e-19
LIBOR	1	2.273729e+06	2.273729e+06	1.571396e+01	1.700250e-04
EQ1	1	1.608490e+01	1.608490e+01	1.111643e-04	9.916165e-01
Residuals	73	1.056272e+07	1.446948e+05	NA	NA

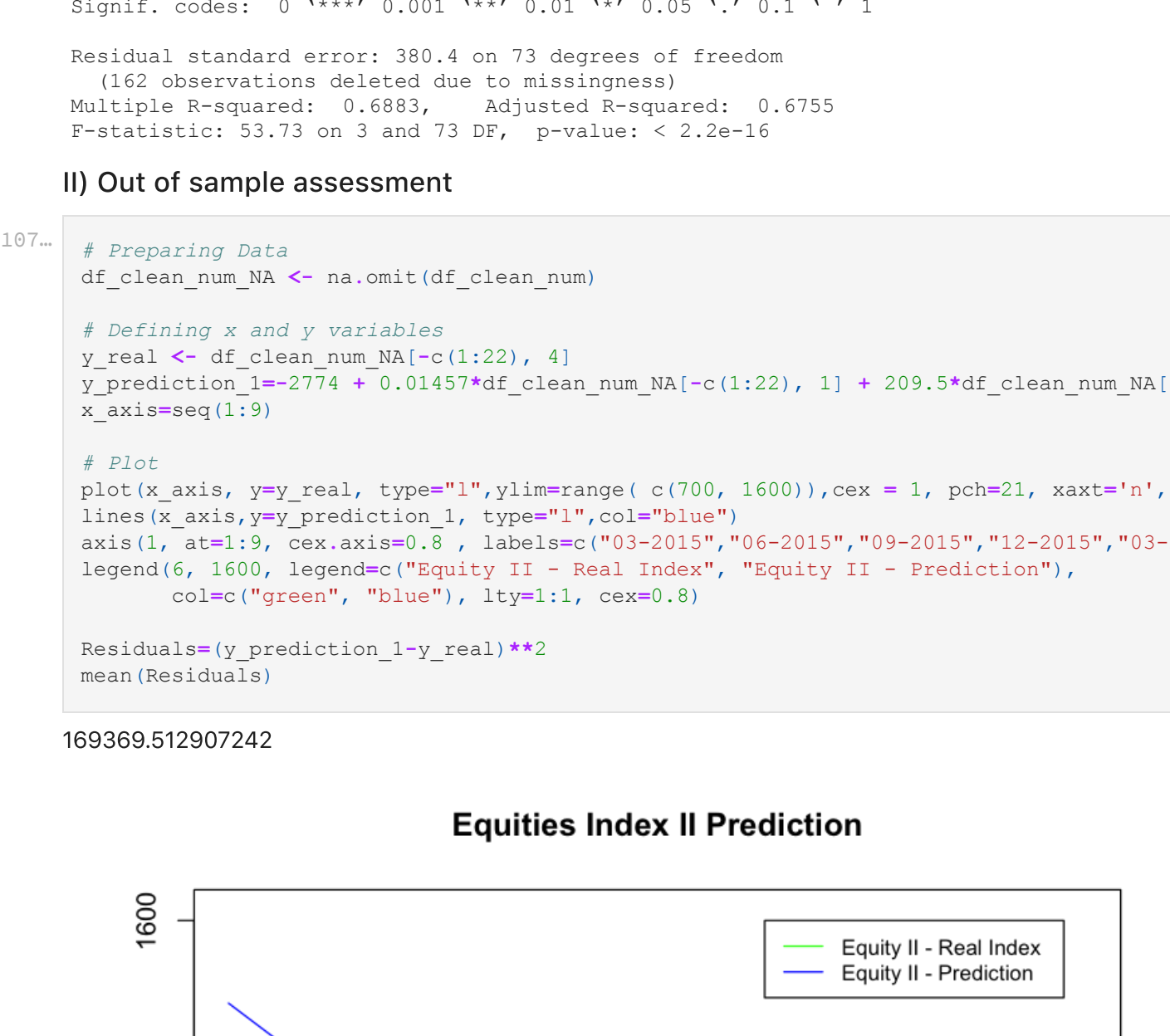
```
Call:
lm(formula = EQ2 ~ GDP + LIBOR + EQ1)

Residuals:
    Min      1Q  Median      3Q     Max 
-927.85 -213.94  -56.25  241.67  836.69 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.773e+03  3.324e+02  -8.343 3.19e-12 ***
GDP          1.458e-02  1.507e-03   9.677 1.01e-14 ***
LIBOR        2.097e+02  5.875e+01   3.570 0.000635 ***
EQ1         -6.132e-04  5.815e-02  -0.011 0.991616    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 380.4 on 73 degrees of freedom
  (162 observations deleted due to missingness)
Multiple R-squared:  0.6883,	Adjusted R-squared:  0.6755 
F-statistic: 53.73 on 3 and 73 DF,  p-value: < 2.2e-16
```

II) Out of sample assessment

In [107…

```
# Preparing Data
df_clean_num_NA <- na.omit(df_clean_num)

# Defining x and y variables
y_real <- df_clean_num_NA[-c(1:22), 4]
y_prediction_1=-2774 + 0.01457*df_clean_num_NA[-c(1:22), 1] + 209.5*df_clean_num_NA[-c
x_axis=seq(1:9)

# Plot
plot(x_axis, y=y_real, type="l",ylim=range( c(700, 1600)),cex = 1, pch=21, xaxt='n', c
lines(x_axis,y=y_prediction_1, type="l",col="blue")
axis(1, at=1:9, cex.axis=0.8 , labels=c("03-2015","06-2015","09-2015","12-2015","03-20
legend(6, 1600, legend=c("Equity II - Real Index", "Equity II - Prediction"),
       col=c("green", "blue"), lty=1:1, cex=0.8)

Residuals=(y_prediction_1-y_real)**2
mean(Residuals)
```

169369.512907242



In [ ]: