Project 4 economic modeling

```
#these library used in codes
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(rcompanion)
```

```
## Warning: package 'rcompanion' was built under R version 4.3.2
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.2
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(e1071)
library (moments)
```

```
##
## Attaching package: 'moments'
```

```
## The following objects are masked from 'package:e1071':
##
##     kurtosis, moment, skewness
```

```
library (faraway)
```

```
##
## Attaching package: 'faraway'
```

```
## The following object is masked from 'package:plyr':
##
##     ozone
```

```
library(dplyr)
library(tidyr)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following objects are masked from 'package:faraway':
##
##     logit, vif
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(xts)
```

```
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################

##
## Attaching package: 'xts'

## The following objects are masked from 'package:data.table':
##
##     first, last

## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```r
Bitcoin<- read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/BITCOIN.csv",sep=",")%
mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
Etherium<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/ETHERIUM.csv",sep=",")
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
Litecoin<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/litecoin-LTC.csv",sep=
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
Chainlink<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/CHAINLINK-LINK.csv",s
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
Ripple<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/ripple-XRP.csv",sep=",")
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
Greyscale_etherium_trust<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/GREYS
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%
```

```
  arrange(Date)
silver <- read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/SILVER.csv",sep=",")%
  mutate(Date = as.Date(as.character(Date), format = "%B %d,%Y")) %>%

  arrange(Date)
Nvidea <- read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/NViDiA(etherium).csv"
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
AMD<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/AMD.csv",sep=",")%>%
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
marathon_digital_holdings <-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/Mara
  mutate(Date = as.Date(as.character(Date), format = "%d-%m-%Y")) %>%

  arrange(Date)
```

The R programme reads several CSV files from specified file paths that contain historical data for various financial assets (Bitcoin, Ethereum, Litecoin, Chainlink, Ripple, Grayscale Ethereum Trust, Silver, Nvidia, AMD, Marathon Digital Holdings). The "Date" column is converted to the Date format and the dataframes are arranged in ascending order based on the date using the %>% pipe operator and the mutate and arrange functions. The code prepares datasets for additional analysis or merging based on the shared "Date" column by ensuring consistency in date formats across datasets. For each asset, a dataframe with daily price and volume information is displayed.

There are different reasons for different crypto and stocks, why I Selected the data, they are as follows: • Bitcoin: Bitcoin is one of the oldest successful cryptocurrencies ever made that inspired the formation of many other cryptocurrencies. So it has a very high chance to correlate with many cryptocurrencies. Bitcoin was introduced in 2009 so it also has powerful historical data.

• Ethereum: Ethereum is a decentralized blockchain with smart contract functionality. It was created in 2013. It is fascinating because it caused a huge shortage of graphic cards during the lockdown due to the Ethereum mining trend in that time period. It also has good historical data suitable for this particular project

• Litecoin: Litecoin is basically "silver to bitcoin's gold." As it contains the slightly modified codebase of Bitcoin. It will be a good opportunity to analyze Bitcoin and Litecoin in one project

• Chainlink & Ripple: Chainlink is a decentralized oracle network just like Ethereum while Ripple includes a digital payment protocol. Chainlink and Ripple may have different risk-return profiles compared to other cryptocurrencies, providing diversification benefits

• Greyscale Ethereum Trust: Greyscale Ethereum Trust holds a significant amount of Ethereum and they have a positive correlation. Here we can do the factor analyses and check whether other factors will affect both the same or not.

• SILVER: Precious metal silver has been utilized as money, a store of value, and an industrial resource. Commodity exchanges, the most popular of which is the COMEX (Commodity Exchange, Inc.), are where silver is actively traded We wanted to use factor analysis on a traditional asset that was unrelated, so we took silver.

• NVIDIA & AMD: We took both graphic card manufacturing companies mostly because they were directly related to the crypto trend that started in lockdown as mentioned above. Also, they are companies with substitute products which makes their correlation negative.

• Marathon Digital Holdings: In the context of the cryptocurrency market, Marathon Digital Holdings

(MARA) and Bitcoin (BTC) are connected, especially in the domain of Bitcoin mining. Here we took it to better understand the factor analysis and check how good factors of Bitcoin affect Marathon Digital Holdings

```
#we removed everything except close
choose=c("Volume","Open","High","Low","Adj.Close")
Bitcoin<-Bitcoin[, !(names(Bitcoin) %in% choose)]
Etherium<-Etherium[, !(names(Etherium) %in% choose)]
Litecoin<-Litecoin[, !(names(Litecoin) %in% choose)]
Chainlink<-Chainlink[, !(names(Chainlink) %in% choose)]
Ripple<-Ripple[, !(names(Ripple) %in% choose)]
Greyscale_etherium_trust<-Greyscale_etherium_trust[, !(names(Greyscale_etherium_trust) %in% choose)]
silver<-silver[, !(names(silver) %in% choose)]
Nvidea<-Nvidea[, !(names(Nvidea) %in% choose)]
AMD<-AMD[, !(names(AMD) %in% choose)]
marathon_digital_holdings<-marathon_digital_holdings[, !(names(marathon_digital_holdings) %in% choose)]
```

we take close columns along with date in this project and removed all other columns

```
#merging every dataframes for wich we will find correlation together
merged_df <- merge(Bitcoin, Etherium, by="Date")
merged_df<-merge(merged_df,Litecoin)
merged_df<-merge(merged_df,Chainlink)
merged_df<-merge(merged_df,Ripple)
merged_df<-merge(merged_df,Greyscale_etherium_trust)
merged_df<-merge(merged_df,silver)
merged_df<-merge(merged_df,Nvidea)
merged_df<-merge(merged_df,AMD)
merged_df<-merge(merged_df,marathon_digital_holdings)
Assets=list("Date","Bitcoin","Etherium","Litecoin","Chainlink","Ripple","Greyscale_etherium_trust","sil
colnames(merged_df)<-Assets
```

Using the "Date" column as the common identifier, the R code combines several dataframes that each represent daily data for various financial assets (Bitcoin, Ethereum, Litecoin, Chainlink, Ripple, Grayscale Ethereum Trust, Silver, Nvidia, AMD, Marathon Digital Holdings). There are columns in the combined dataframe, called "merged_df," for the date and the daily price or values of each asset. For clarity, the code also appends a list of asset names to the merged dataframe's column names. The relationships and correlations between the daily returns of various financial assets over time can be easily analysed with this consolidated dataframe.

```
# doing log return for volatility
Total<- list(Bitcoin,Etherium,Litecoin,Ripple,Chainlink,Greyscale_etherium_trust,silver,Nvidea,AMD,mara
for (i in 1:10) {
  log_returns <- diff(log(as.matrix(Total[[i]][,2])))
  volatility<- sd(log_returns)
  cat(paste(names(Total[[i]])))
  print(volatility)

}
```

```
## Date BTC[1] 0.1722016
## Date Etherium[1] 0.2146005
## Date Litcoin[1] 0.1829385
## Date Ripple[1] 0.2222076
```

```
## Date CHAINLINK[1] 0.210923
## Date GREYSCALE[1] 0.2970558
## Date Silver[1] 0.09655193
## Date NVIDIA[1] 0.1840043
## Date AMD[1] 0.1766942
## Date MDH[1] 0.2869552
```

```
library(broom)
library(ggplot2)
log_returns <- diff(log(as.matrix(merged_df[, 2:11])))
correlation_matrix <- cor(log_returns)
volatility<- sd(log_returns)
print(correlation_matrix)
```

```
##                             Bitcoin  Etherium  Litecoin Chainlink    Ripple
## Bitcoin                   1.0000000 0.9022646 0.6736151 0.7601794 0.6679179
## Etherium                  0.9022646 1.0000000 0.6736937 0.6732464 0.6241737
## Litecoin                  0.6736151 0.6736937 1.0000000 0.5985285 0.5161590
## Chainlink                 0.7601794 0.6732464 0.5985285 1.0000000 0.7565682
## Ripple                    0.6679179 0.6241737 0.5161590 0.7565682 1.0000000
## Greyscale_etherium_trust  0.8603892 0.9283886 0.5049238 0.7274451 0.6256698
## silver                    0.1892979 0.1428963 0.4199199 0.2082319 0.3806788
## Nvidea                    0.5840878 0.6959891 0.6476927 0.3505705 0.4175276
## AMD                       0.5110361 0.6051272 0.4368167 0.1372609 0.3143438
## marathon_digital_holdings 0.8725976 0.9145883 0.5752384 0.7340612 0.6970190
##                           Greyscale_etherium_trust     silver    Nvidea
## Bitcoin                                 0.86038921 0.18929789 0.5840878
## Etherium                                0.92838859 0.14289633 0.6959891
## Litecoin                                0.50492378 0.41991989 0.6476927
## Chainlink                               0.72744510 0.20823194 0.3505705
## Ripple                                  0.62566976 0.38067878 0.4175276
## Greyscale_etherium_trust                1.00000000 0.04778099 0.6068189
## silver                                  0.04778099 1.00000000 0.1190701
## Nvidea                                  0.60681892 0.11907009 1.0000000
## AMD                                     0.44365045 0.01878094 0.7606954
## marathon_digital_holdings               0.95115879 0.01695061 0.6385725
##                                  AMD marathon_digital_holdings
## Bitcoin                   0.51103610                 0.87259760
## Etherium                  0.60512721                 0.91458835
## Litecoin                  0.43681672                 0.57523843
## Chainlink                 0.13726088                 0.73406123
## Ripple                    0.31434381                 0.69701903
## Greyscale_etherium_trust  0.44365045                 0.95115879
## silver                    0.01878094                 0.01695061
## Nvidea                    0.76069538                 0.63857246
## AMD                       1.00000000                 0.46822512
## marathon_digital_holdings 0.46822512                 1.00000000
```

```
print(volatility)
```

```
## [1] 0.2499068
```

This r code computes and displays the volatility of daily logarithmic returns for ten financial assets, including cryptocurrencies (Bitcoin, Ethereum, Litecoin, Ripple, Chainlink, Grayscale Ethereum Trust, Silver),

semiconductor stocks (Nvidia, AMD), and a cryptocurrency mining firm (Marathon Digital Holdings). Iterating through the list of assets, it calculates the logarithmic returns and prints the volatility for each. The code also computes the correlation matrix of returns for all assets and displays the overall volatility. For additional statistical analysis and visualization, the 'broom' and 'ggplot2' libraries are loaded. Results of observations(Volitility): Greyscale Ethereum Trust and Marathon Digital Holdings appear to be more volatile (0.2970558 & 0.2869552 respectively) than cryptocurrencies. The volatility of silver (0.09655193) is relatively low compared to cryptocurrencies (BTC: 0.1722016 ,Etherium:0.2146005, Litcoin: 0.1829385, Ripple: 0.2222076, CHAINLINK:0.210923).This means that silver is more safer in investing compared to cryptocurrencies and if you want more risk and more return invest in Greyscale_etherium_trust and marathon_digital_holdings as it invests in cryptocurrencies along with risk of company managent.

Correlation Matrix: The correlation matrix shows the correlation coefficients pairwise between the logarithmic returns of different assets. Notable observations: There is a strong positive correlation between Bitcoin and Ethereum (0.9022646), Ripple and Chainlink. The correlation of Greyscale Ethereum Trust and silver is quite high between themselves and with other cryptocurrencies,suggesting that its performance aligns closely with the broader cryptocurrency market. Some negative correlations, such as the one between Nvidia and AMD(-0.76069538) as they are competetors in the graphic card market and also used for crypto currency market.Due to its involvement in cryptocurrency mining, Marathon Digital Holdings has particularly strong correlations with Bitcoin(0.87259760), Ethereum(0.91458835), and other cryptocurrencies(0.57523843,0.73406123 & 0.69701903).

```r
#factor model
factor_model <- prcomp(log_returns, scale. = TRUE)
print(factor_model)
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.4956856 1.1496467 1.0604571 0.6672406 0.5969437 0.4913348 0.3697526
##  [8] 0.2935195 0.2004216 0.1389074
##
## Rotation (n x k) = (10 x 10):
##                                PC1         PC2         PC3         PC4
## Bitcoin                  0.37283224  0.02355185  0.10621900 -0.08044028
## Etherium                 0.38121041 -0.11156892  0.05643229 -0.04834286
## Litecoin                 0.30481328  0.14994512 -0.35697551 -0.66274204
## Chainlink                0.31582935  0.33864651  0.30036870 -0.16019479
## Ripple                   0.30753257  0.33488834  0.05576469  0.58908982
## Greyscale_etherium_trust 0.36322989 -0.07783527  0.27836740  0.05193552
## silver                   0.09561109  0.60463471 -0.56999747  0.19373131
## Nvidea                   0.30079704 -0.34130434 -0.36179800 -0.06667389
## AMD                      0.24109502 -0.49306205 -0.40646728  0.36478657
## marathon_digital_holdings 0.37249973 -0.08026326  0.25743227  0.03914554
##                                PC5         PC6         PC7         PC8
## Bitcoin                  -0.2183731 -0.45983004 -0.07781974  0.75181415
## Etherium                 -0.3313857 -0.08191372  0.16237662 -0.34517973
## Litecoin                  0.2655176 -0.13384371  0.38112401 -0.15773057
## Chainlink                 0.3618092 -0.09518924 -0.66430301 -0.22174957
## Ripple                    0.4816868  0.05355357  0.36984534  0.06587089
## Greyscale_etherium_trust -0.3775621  0.24370387 -0.10207489 -0.25359213
## silver                   -0.4468713  0.09446850 -0.15835763 -0.02962849
## Nvidea                    0.2018540  0.64082262 -0.26544361  0.31871639
## AMD                       0.1118128 -0.47484751 -0.17717098 -0.26357416
## marathon_digital_holdings -0.1165110  0.21806813  0.32694351  0.03158130
##                                PC9        PC10
## Bitcoin                  -0.07628828 -0.08035861
```

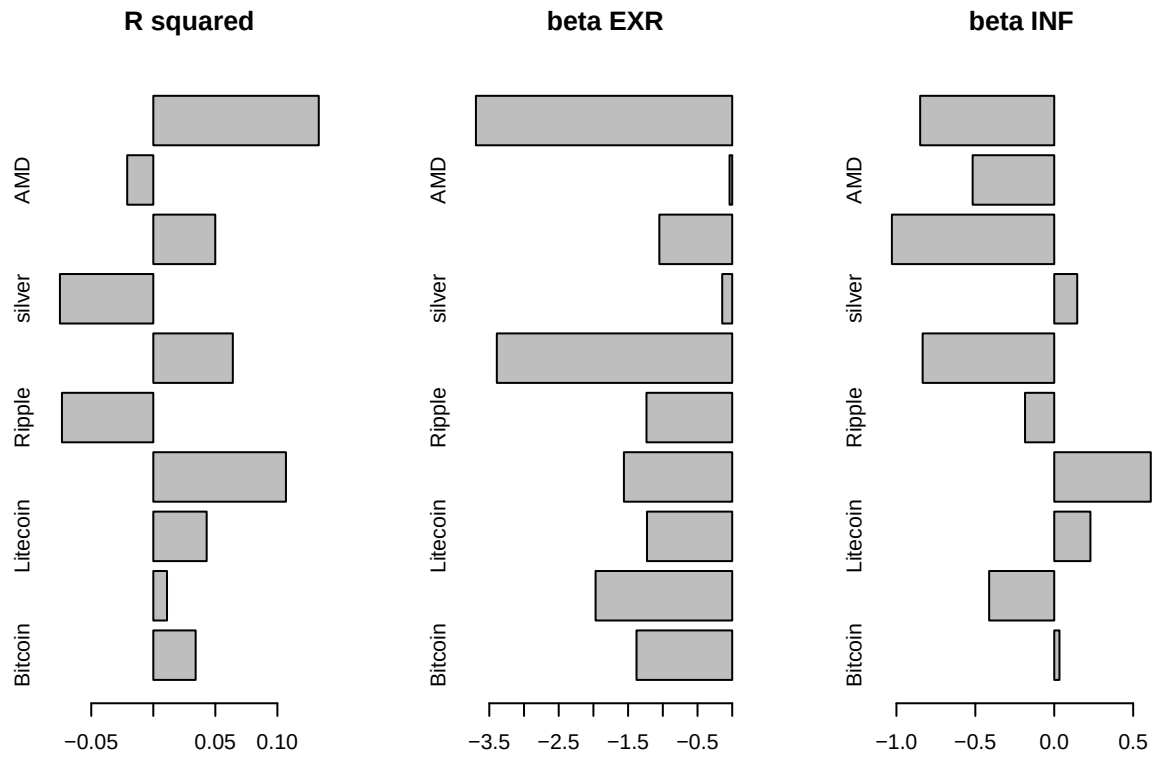
```
## Etherium                  -0.62253987  0.43256436
## Litecoin                   0.08273348 -0.22924617
## Chainlink                  0.07060458  0.18510376
## Ripple                    -0.21566094 -0.14357106
## Greyscale_etherium_trust   0.04774042 -0.70913182
## silver                     0.14416990  0.08653447
## Nvidea                    -0.17128333  0.07008718
## AMD                        0.24473426 -0.04171444
## marathon_digital_holdings  0.66029371  0.42660551
```

Principal Component Analysis (PCA) is carried out by the R code on the matrix of log returns (log_returns) by means of the prcomp function. In order to represent a factor model, it computes the principal components and scales the data. The principal components' contributions to variance are detailed in the result (factor_model). The first few principal components (PC1, PC2, etc.) often explain most of the variability in the data.

PC1 appears to have positive loadings for the majority of assets, reflecting a general market trend.Positive loadings are high positive for PC1, Bitcoin, and Greyscale Ethereum Trust, while low positive loadings are low for Silver. This suggests that PC1 captures a pattern that distinguishes Bitcoin and Greyscale Ethereum Trust from Silver. In PC2, Ethereum and Litecoin have opposite signs, indicating a possible negative correlation in their returns in the context of this factor model. PC3 appears to support cryptocurrency except lit coin and the directly related assets of bitcoin and etherium PC2, PC4, and so on capture further patterns of variability in the data. It's worth noting that PC10 has a nearly zero standard deviation and loadings close to nil for all assets. This shows that PC10 does not add much to explaining the variability in the dataset and may be regarded as inconsequential.

```
#Example : Factor analysis of crypto and traditional assets taking currency exchange and inflation
EXchange_rate<-read.csv("C:/Users/Siddharth Sharma/Desktop/Semester1/GROUP project 2 codes/EURUSD=X (1)
  mutate(Date = as.Date(as.character(Date), format = "%Y-%m-%d")) %>%

  arrange(Date)
EXchange_rate<-EXchange_rate[, !(names(EXchange_rate) %in% choose)]

EXR = as.matrix(EXchange_rate$Close)
exr=as.data.frame(diff(log(EXR)))
names(exr)[1]="EXR"
INFLATION<-read.csv("C:/Users/Siddharth Sharma/Desktop/PROJECT 4 Economic modeling/Inflation.csv",sep="
  mutate(Date = as.Date(as.character(Date), format = "%B %d,%Y")) %>%

  arrange(Date)
INF=as.matrix(INFLATION$INFLATION[0:25])
inf = as.data.frame(diff(log(INF)))
names(inf)[1]="INF"
VART<-cbind(exr,inf)
arFit = ar.ols(cbind(exr, inf), order.max = 1)
res = arFit$resid[(4:22),] # residuals of the AR (5) modle (the interpretations of the residuals is tha
lmfit = lm(log_returns~res[,1]+res[,2]) #fit a regression Y is a set of teh stock returns and X1 and X2
slmfit = summary(lmfit)
rsq = rep(0,10)
for (i in 1:10){rsq[i]= slmfit[[i]][[9]]} # we substruct the values of R2 from each of 9 models
beta_CPI = lmfit$coef[2,] # extract all CPI betas (b1)
beta_IP = lmfit$coef[3,] # extract all IP betas (b2)

par(mfrow=c(1,3)) # building three graphs in a row
```

```r
barplot(rsq,horiz=T,names=names(beta_CPI),main="R squared") #Creates a bar plot with vertical or horizo
barplot(beta_CPI,hori=T,main="beta EXR") #Creates a bar plot with vertical or horizontal bars.
barplot(beta_IP,hori=T,main="beta INF") #Creates a bar plot with vertical or horizontal bars.
```



The provided R code performs a factor analysis on cryptocurrency and traditional assets, taking currency exchange rates and inflation into account. Here's a more in-depth breakdown:

Importing and Preprocessing Data: The data for exchange rates is read from a CSV file, and the "Date" column is converted to Date format. extra columns are removed, and the exchange rate log returns (EXR) are computed. To obtain the log returns of inflation (INF), inflation data is read and similar preprocessing is applied. Vector Autoregression (VAR): The code creates a matrix (VART) by combining the log-returns of the exchange rate and inflation.The ar.ols function is used to fit a Vector Autoregression (VAR) model to the data. The AR model's residuals represent unexpected shocks. Linear Regression Model: The log returns on financial assets are used as the dependent variable, and the residuals from the VAR model for the exchange rate and inflation are used as the independent variables.The regression summary is used to extract the R-squared values and coefficients (betas). Visualisation: The code generates three bar plots for display. The R-squared values for each model are shown in the first plot. The beta coefficients for the exchange rate (EXR) and inflation (INP) are shown in the second and third plots, respectively.

After analysing the graphs we got to know that Exchange rate supports all of the crypto and Traditional assets but is majorly supports Marathon Digital Holdings and Greyscale Ethereum Trust and support silver and AMD in a negligable amount.Also Inflation and silver(precious metal) has positive relation which can be due to silver bieng believed as a hedge against inflation.Bitcoin and litecoin have almost the same basecode so both will have a same type of correlation but the correlation itself differs.Although Greyscale Ethereum Trust should have a negative relation with inflation because of etherium and inflation relation but it has a positive relation,this can be due to its other income sources

```r
library (psych) # the library for factor analysis
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:car':
##
##     logit

## The following object is masked from 'package:faraway':
##
##     logit

## The following object is masked from 'package:rcompanion':
##
##     phi

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library (GPArotation) # to estimate oblimin rotation
```

```
##
## Attaching package: 'GPArotation'

## The following objects are masked from 'package:psych':
##
##     equamax, varimin
```

```r
crypto = merged_df[,2:11]
describe(crypto) # general description of the data
```

```
##                           vars  n     mean      sd   median  trimmed     mad
## Bitcoin                      1 20 29092.98 8795.61 28478.48 28377.20 8540.09
## Etherium                     2 20  1958.32  674.54  1822.02  1871.13  290.97
## Litecoin                     3 20    84.66   26.28    89.11    82.24   30.52
## Chainlink                    4 20     9.25    4.31     7.59     8.55    1.53
## Ripple                       5 20     0.53    0.16     0.50     0.51    0.16
## Greyscale_etherium_trust     6 20    12.98    7.22    10.66    11.78    3.10
## silver                       7 20    22.35    2.20    22.60    22.52    2.52
## Nvidea                       8 20   268.21  113.58   258.86   259.19  136.77
## AMD                          9 20    97.57   21.75    98.26    97.55   23.46
## marathon_digital_holdings   10 20   322.40  105.04   296.98   319.45   93.38
##                               min      max    range  skew kurtosis      se
## Bitcoin                   16547.50 46306.45 29758.95  0.58    -0.79 1966.76
## Etherium                   1067.30  3682.63  2615.33  1.16     0.44  150.83
## Litecoin                     53.40   146.51    93.12  0.55    -0.63    5.88
## Chainlink                     5.57    19.59    14.02  1.28     0.02    0.96
## Ripple                        0.33     0.83     0.50  0.60    -0.81    0.03
```

```
## Greyscale_etherium_trust      4.76    32.36    27.60  1.36     0.73    1.61
## silver                       17.76    25.12     7.36 -0.51    -1.00    0.49
## Nvidea                      121.39   493.55   372.16  0.52    -0.98   25.40
## AMD                          60.06   143.90    83.84 -0.03    -0.69    4.86
## marathon_digital_holdings   141.57   544.49   402.92  0.31    -0.70   23.49
```

The R code makes use of the 'psych' library for factor analysis and the 'GPArotation' library for oblimin rotation. Columns 2 to 11 of the'merged_df' dataframe representing cryptocurrency and traditional asset returns are chosen. The 'describe' function provides a general statistical description of the selected data, allowing for preliminary insights into its properties.

The table shows the mean, standard deviation (sd), median, trimmed mean, median absolute deviation (mad), minimum, maximum, range, skewness, kurtosis, and standard error for each asset. Bitcoin: Skewness: 0.58 (right-skewed distribution) The mean(29092.98) is influenced by extreme values, most likely outliers, and thus exceeds the median(28478.48). The trimmed mean(28377.20) is a more robust measure of central tendency because it excludes some outlier values. The high standard deviation indicates a wide dispersion of returns, implying that Bitcoin's performance is highly volatile.

Etherium: Skewness: 1.16 (right-skewed distribution) The mean(1958.32) is higher than the median(1822.02), indicating that outliers are present and pulling the mean upwards. By reducing(1871.13) the impact of outliers, the trimmed mean provides a more robust measure. Etherium's lower standard deviation suggests lower volatility than Bitcoin, implying less variability in returns.

Litecoin: Skewness: 0.55 (right-skewed distribution) The mean(84.66) is lower than the median(89.11), indicating that outliers may be present and pulling the mean down. The trimmed mean(82.24) estimates central tendency more accurately. The low standard deviation of Litecoin indicates relatively stable returns with less variability.

Chainlink: Skewness: 1.28 (right-skewed distribution) The mean(9.25) is higher than the median(7.59), indicating that outliers may be influencing the mean. Outliers have less of an impact on the trimmed mean(8.55), making it a more reliable measure. Chainlink has low volatility and a low standard deviation, implying relatively stable returns.

Ripple: Skewness: 0.60 (right-skewed distribution) The mean(0.53)is close to the median(0.50), indicating that the distribution is relatively symmetric, with fewer outliers influencing the mean. Ripple has an extremely low standard deviation, indicating low volatility and consistent returns.

Greyscale Ethereum Trust: Skewness: 1.36 (right-skewed distribution) The mean(12.98) exceeds the median(10.66), indicating the presence of outliers. Extreme values have less of an impact on the trimmed mean(11.78). Greyscale Ethereum Trust's standard deviation suggests moderate volatility with some return variability.

Silver: Skewness: -0.51 (left-skewed distribution) The mean (22.35) is close to the median(22.60), indicating that the distribution is symmetric with few outliers. Silver has low volatility and a low standard deviation, implying relatively stable returns.

Nvidia: Skewness: 0.52 (right-skewed distribution) The mean(268.21) exceeds the median(258.86), indicating the presence of outliers. The trimmed mean(259.19) is a more reliable metric. Nvidia is more volatile than some other assets, with a relatively large standard deviation.

AMD: Skewness: -0.03 (Slightly left-skewed distribution) The mean(97.57) is close to the median(98.26), indicating that the distribution is relatively symmetric with few outliers. AMD's standard deviation is moderate, indicating moderate volatility and some variability in returns.

Marathon Digital Holdings: Skewness: 0.31 (right-skewed distribution) The mean(322.40) is higher than the median(296.98), indicating the possibility of outliers. Extreme values have less of an impact on the trimmed mean(319.45). Marathon Digital Holdings exhibits higher volatility, as evidenced by a higher standard deviation, implying greater variability in returns.

```r
#KMO
KMO(crypto)
```

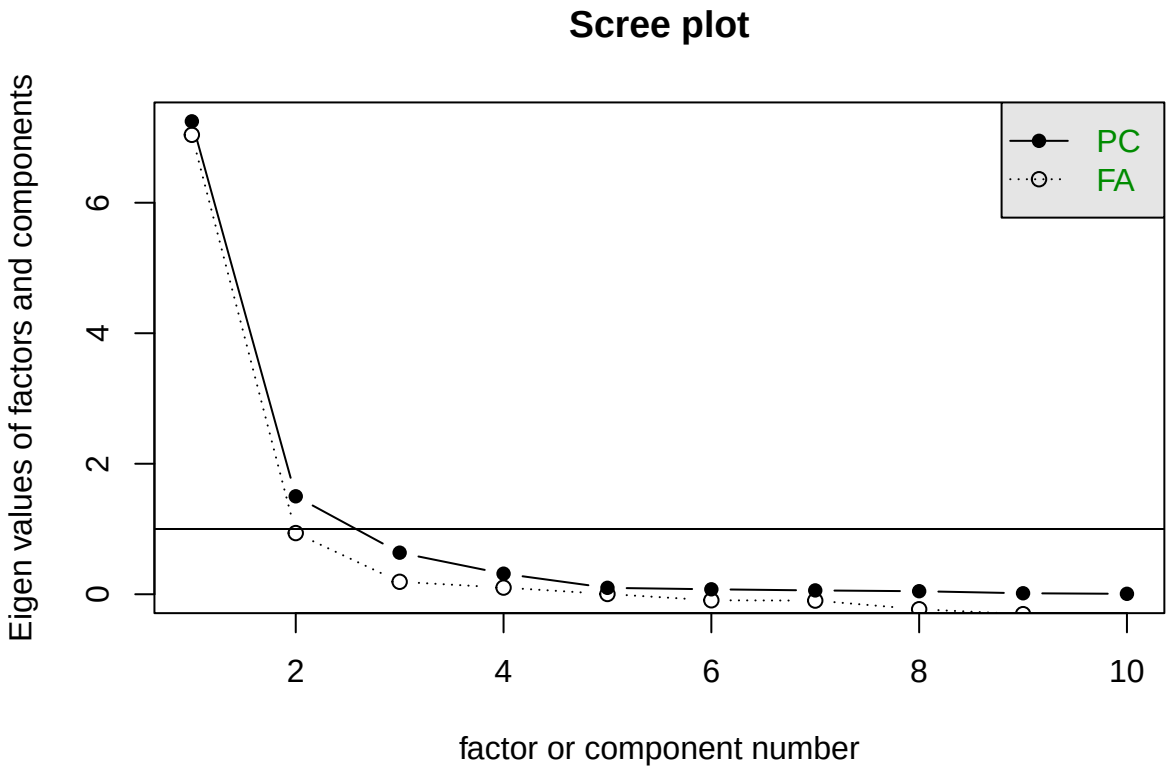
```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = crypto)
## Overall MSA =  0.75
## MSA for each item =
##                   Bitcoin                  Etherium                   Litecoin
##                      0.82                      0.78                      0.74
##                 Chainlink                    Ripple  Greyscale_etherium_trust
##                      0.75                      0.81                      0.75
##                    silver                   Nvidea                       AMD
##                      0.59                      0.49                      0.75
## marathon_digital_holdings
##                      0.76
```

The code "KMO(crypto)" computes the Kaiser-Meyer-Olkin (KMO) sampling adequacy measure for the specified dataset "crypto." The KMO statistic evaluates the data's suitability for factor analysis by calculating the proportion of variance in the variables that could be explained by underlying factors. A higher KMO value, typically between 0 and 1, indicates greater suitability for factor analysis. A KMO closer to one indicates that the variables in the dataset are suitable for capturing common factors, thereby supporting the validity of performing factor analysis on the data.

According to the result, silver and Nvidia are not that suitable for factor analysis compared to bitcoin and Ripple which are best suited for factor analysis but all are passable values

```r
##Determining the Number of Factors to Extract
# scree plot
scree(crypto)
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## The estimated weights for the factor scores are probably incorrect.  Try a
## different factor score estimation method.
```

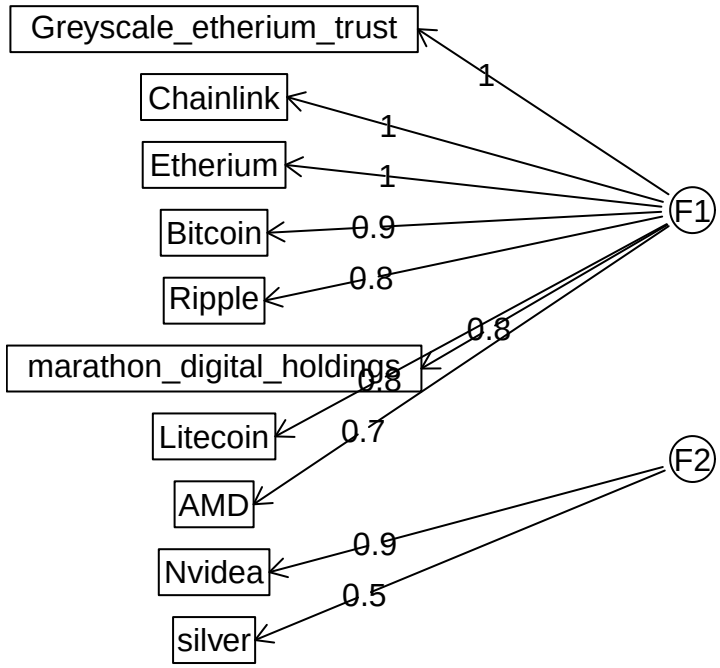


scree plot show the number of factors to be taken in the estimation of factior model which is the number of factors above 1 or values forming an elbow shape looking at the scree plot we get to know that the elbow lies on 2nd factor so we take number of factors as 2

```
# estimation factor model
factor.model <- fa(crypto, nfactors = 2, fm="ols", max.iter = 100, rotate = "oblimin")
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## The estimated weights for the factor scores are probably incorrect.  Try a
## different factor score estimation method.
```

```
# we estimate a factor model with 3 factors, estimated by means of OLS, 100 is the number of iterations
# rotate - we apply oblimin rotation, allowing factors to correlate.
# make it visual
fa.diagram(factor.model) #
```




```
# Communality
factor.model$communality
```

```
##                   Bitcoin                  Etherium                  Litecoin
##                 0.9451498                 0.9859651                 0.7735222
##                 Chainlink                    Ripple  Greyscale_etherium_trust
##                 0.9469694                 0.8656159                 0.9597199
##                    silver                    Nvidea                       AMD
##                 0.4628338                 0.8354565                 0.7603232
## marathon_digital_holdings
##                 0.8970763
```

```
#Eeigenvalues
factor.model$e.values
```

```
##  [1] 7.247327660 1.500868186 0.636886977 0.314787504 0.098163397 0.075592929
##  [7] 0.058549590 0.046463575 0.014924424 0.006435758
```

The code computes and retrieves eigenvalues and communalities from a factor model. The communalities are provided by factor.model$communality, which represents the proportion of each variable's variance explained by the factors. The
indicate the amount of total variance explained by each factor in the model.

In a factor model, the output represents the communalities and eigenvalues for each variable. Communalities, which range from 0 to 1, represent the proportion of a variable's variance that can be explained by the factors. Notably, Etherium (0.99) and Greyscale Ethereum Trust (0.96) have high communalities, indicating that

the underlying factors explain a significant portion of their variances. Silver (0.46), on the other hand, has a lower communality, indicating that its variance is less well explained by the factors. The eigenvalues indicate how much of the total variance is explained by each factor. The first factor in this case has a relatively large eigenvalue of 7.25, indicating that it explains a significant portion of the overall variance, while subsequent factors contribute progressively less. This data assists in determining the significance of each variable in the factor analysis

```
#Percentage of Variance Accounted For
100*factor.model$e.values/length(factor.model$e.values)
```

```
##  [1] 72.47327660 15.00868186  6.36886977  3.14787504  0.98163397  0.75592929
##  [7]  0.58549590  0.46463575  0.14924424  0.06435758
```

```
print(factor.model$loadings, cutoff=0, digits=3)
```

```
##
## Loadings:
##                            [,1]   [,2]
## Bitcoin                     0.921  0.143
## Etherium                    0.999 -0.023
## Litecoin                    0.769  0.257
## Chainlink                   1.017 -0.264
## Ripple                      0.831  0.242
## Greyscale_etherium_trust    1.022 -0.233
## silver                      0.301  0.529
## Nvidea                     -0.007  0.916
## AMD                         0.651  0.421
## marathon_digital_holdings   0.784  0.350
##
##                  [,1]  [,2]
## SS loadings     6.335 1.688
## Proportion Var  0.634 0.169
## Cumulative Var  0.634 0.802
```

```
print(factor.model$Structure, cutoff=0, digits=3)
```

```
##
## Loadings:
##                            [,1]  [,2]
## Bitcoin                    0.962 0.412
## Etherium                   0.993 0.269
## Litecoin                   0.844 0.482
## Chainlink                  0.940 0.032
## Ripple                     0.901 0.484
## Greyscale_etherium_trust   0.954 0.065
## silver                     0.455 0.616
## Nvidea                     0.260 0.914
## AMD                        0.774 0.610
## marathon_digital_holdings 0.886 0.578
##
##                  [,1]  [,2]
## SS loadings     6.888 2.636
```

```
## Proportion Var 0.689 0.264
## Cumulative Var 0.689 0.952
```

The code computes the percentage of variance that each factor in the factor model accounts for. By dividing each eigenvalue by the total number of factors, the expression "100*factor.model$e.values/length(factor.model$e.values)" computes the percentage. The loadings and structure matrix of the factor model are printed in the following lines. Loadings represent the magnitude and direction of the relationship between variables and factors, whereas the structure matrix represents the correlation between variables and factors. The "cutoff" parameter removes loadings or correlations that are less than a certain threshold (which is set to 0), and "digits" controls the number of decimal places displayed in the output.

In a factor model, the output shows the percentage of variance accounted for by each factor as well as the loadings for variables on those factors. The first factor explains the majority of the variance (72.47%), with subsequent factors contributing less. In the first factor, for example, Etherium and Greyscale Ethereum Trust have high loadings, indicating strong positive associations. The sum of squared loadings (SS loadings) quantifies how much variance each factor captures, with the first factor explaining 63.4% of the total variance. According to the cumulative variance, the first two factors account for 80.2% of the variance. These findings aid in interpreting the significance of each factor and the variables that contribute to to them in explaining the overall dataset variability.

Also the analysis can be interpreted in the following way :

All of the assets accept Nvidia and Silver belong to factor 1 and Nvidia and Silver belong to factor 2 which was already assumed as they were not perfectly fitting in the KMO