Homework 2 - SDS

Andrea Conti - Davide Mascolo - Mario Napoli

27 dicembre 2021

1. Introductory

Let's define the remaining v values: $v(AB) = 2, \quad v(AC) = 0, \quad v(BC) = 2$. Now we can build the table as follows:

Permutations	A	B	C
ABC	0	2	2
ACB	0	4	0
BAC	2	0	2
BCA	2	0	2
CAB	0	4	0
CBA	2	2	0

From this table we can compute Shapley values: $\psi(A) = \frac{6}{6} = 1, \quad \psi(B) = \frac{12}{6} = 2, \quad \psi(C) = \frac{6}{6} = 1$. We can normalize these values in a way that they sum up to 1, in other words we can divide them by 4 but the interpretation would remain the same: A save 1 hour of work (or 25% of the total savings if we normalize), B 2 hours (or 50% of the total) and C another hour (the remaining 25%).

2. Probabilistic

```
## Utils
library(tseries)
library(igraph)
library(knitr)
```

```
## Load CharFun
load('char_fun.RData')
```

```
options(scipen = 999)
```

We separate coalitions in which it is present *j* from those in which *j* is not present.

```
## Split lists
split_list <- function(j){
  # Input: a specific player
  # Output: a list with all the coalitions in which j is not present
  no_j <- vector(mode = "list", length = 2048-1)
  k <- 1
  for (i in 1:12){
      for (c in 1:length(char_fun[[i]])){
        name <- names(char_fun[[i]][c])
        value <- char_fun[[i]][[c]]
        if (!(j %in% as.integer(strsplit(name, '-')[[1]]))){
          no_j[[k]] <- c(name, value)
          k = k + 1
        }
      }
  }
  return(no_j)
}
```

Compute the differences between the values of the coalitions without the player *j* and the coalitions with the player *j*.

```
compute_diff <- function(j){
  # Input: a specific player
  # Output: list in which are stored the difference: C U {j} - C
  no_j <- split_list(j)
  diff_list <-c()
  for (i in 1:length((no_j))){
    coal <- no_j[[i]][1]
    val_coal <- as.integer(no_j[[i]][2])
    # Just elaboration to obtain the number (players) in coalition C U {j}
    coal2 <- paste(sort(c(as.numeric(unlist(strsplit(coal, split = '-'))), j)), collapse = '-')
    tmp <- strsplit(coal2, split = '-')[[1]]
    val_coal2 <- char_fun[[length(tmp)]][coal2][[1]]
    diff_list <- c(diff_list, val_coal2 - val_coal)
  }
  return(diff_list)
}
```

```
## Simulation
simul <- function(j){
  # Input: a specific player
  # Output: our estimated Shapley values plus more things useful for later steps:
  #         - max and min of the difference list
  #         - M, our computed simulation size
  alpha <- 0.05
  epsilon <- 0.1

  diff_list <- compute_diff(j)

  M <- ceiling(((max(diff_list) - min(diff_list))^2*log(2/alpha))/(2*(epsilon^2)))
  tot <- 0

  for (i in 1:M){
    n <-sample(diff_list, 1, replace = TRUE)
    tot = tot + n
  }

  return(c(tot / M, max(diff_list), min(diff_list), M))
}
```

We run the simulation for the three selected players and compute the respective confidence intervals.

```
## Set players
players <- c(2, 3, 5)

df <- data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(df) <- c('LowerBound', 'Value', 'UpperBound')
rownames(df) <- c(players)
i <- 1
for (p in players){
  shap_ret = simul(p)
  shapley_val = shap_ret[1]
  mx <- shap_ret[2]
  mn <- shap_ret[3]
  M <- shap_ret[4]
  eps_alpha <- sqrt(((mx - mn)^2 * log(2/0.05)) / (2*M))
  df$LowerBound[i] <- shapley_val - eps_alpha
  df$Value[i] <- shapley_val
  df$UpperBound[i] <- shapley_val + eps_alpha
  i = i + 1
}
kable(df)
```

	LowerBound	Value	UpperBound
2	6.697495	6.797485	6.897475
3	5.818037	5.918022	6.018007
5	2.874096	2.974096	3.074096

We set $\alpha = 0.05$ and $\epsilon = 0.1$. Then, starting from the following formula $\epsilon_\alpha = \sqrt{\frac{(b-a)^2}{2M}\log(\frac{2}{\alpha})}$ we reversed it in order to compute M, that is the size of the simulation that allows us to reach a $1-\alpha$ confidence interval. After that, we computed the confidence intervals of the Shapley values for each player. Regardless of the value of M, we can note that ϵ_α is almost the same for the three players that we have chosen. This implies that the confidence intervals for the three different players about the same width due to our "α-level".

3. Statistical

List of S&P 500

- AMZN - Customer Discretionary
- MSFT - Informational Technlogy
- BLK - Financials
- CAH - Healt-Care
- VZ - Communication Services
- CVX - Energy
- ECL - Materials
- EIX - Utilitis
- DAL - Industrials
- SYY - Customer Staples

```
## Load Data Function
load_data <- function(code){
  # Input: a specific code of stock (e.g. Amazon -> AMZN)
  # Output: the historical series of that title

  data <- get.hist.quote(instrument = code,
                         start = '2020-01-01',
                         end = '2021-12-22',
                         quote = c('Open', 'Close'),
                         provider = 'yahoo',
                         drop = TRUE)
  return(data)
}
```

Load data

```
options("getSymbols.warning4.0"=FALSE)
## AMZN
amz_data <- load_data('AMZN')
## MSFT
msft_data <- load_data('MSFT')
## BR
blk_data  <- load_data('BR')
## CAH
cah_data  <- load_data('CAH')
## VZ
vz_data   <- load_data('VZ')
## CVX
cvx_data  <- load_data('CVX')
## ECL
ecl_data  <- load_data('ECL')
## EIX
eix_data  <- load_data('EIX')
## DAL
dal_data  <- load_data('DAL')
## SYY
syy_data  <- load_data('SYY')
```

Build portfolio

```
port <- data.frame(matrix(0, nrow(amz_data-1), 10))
names(port) <- c('AMZN', 'MSFT', 'BLK', 'CAH', 'VZ',
                 'CVX', 'ECL', 'EIX', 'DAL', 'SYY')

## Populate Port
for (i in (2:nrow(amz_data))){
  port$AMZN[i-1]<-log(amz_data$Close[[i]]/
                        amz_data$Close[[i-1]])
  port$MSFT[i-1]<-log(msft_data$Close[[i]]/
                      msft_data$Close[[i-1]])
  port$BLK[i-1]<-log(blk_data$Close[[i]]/
                       blk_data$Close[[i-1]])
  port$CAH[i-1]<-log(cah_data$Close[[i]]/
                       cah_data$Close[[i-1]])
  port$VZ[i-1]<-log(vz_data$Close[[i]]/
                       vz_data$Close[[i-1]])
  port$CVX[i-1]<-log(cvx_data$Close[[i]]/
                        cvx_data$Close[[i-1]])
  port$ECL[i-1]<-log(ecl_data$Close[[i]]/
                        ecl_data$Close[[i-1]])
  port$EIX[i-1]<-log(eix_data$Close[[i]]/
                        eix_data$Close[[i-1]])
  port$DAL[i-1]<-log(dal_data$Close[[i]]/
                        dal_data$Close[[i-1]])
  port$SYY[i-1]<-log(syy_data$Close[[i]]/
                        syy_data$Close[[i-1]])
}

## View
kable(head(port))
```

AMZN	MSFT	BLK	CAH	VZ	CVX	ECL	EIX	DAL	SYY
-0.0122133	-0.0125299	0.0008075	-0.0146810	-0.0107040	-0.0034648	-0.0047951	0.0053634	-0.0167382	-0.0036685
0.0147759	0.0025815	0.0090004	0.0082720	-0.0021547	-0.0033939	-0.0085405	0.0046695	-0.0069133	-0.0013049
0.0020894	-0.0091596	-0.0035262	0.0004018	-0.0111789	-0.0128517	-0.0012794	0.0096696	-0.0008675	-0.0081049
-0.0078393	0.0158029	0.0064021	0.0054081	0.0018440	-0.0114886	0.0003733	-0.0015832	0.0212956	0.0115416
0.0047878	0.0124156	0.0216216	0.0047828	-0.0145077	-0.0016155	0.0132424	0.0052673	0.0018674	0.0011823
-0.0094552	-0.0046378	-0.0039893	0.0065392	0.0023762	-0.0091473	-0.0058580	-0.0084411	0.0047378	0.0034208

```
## Estimate Correlation Matrix
kable(cor(port), allign = 'ccccccccc')
```

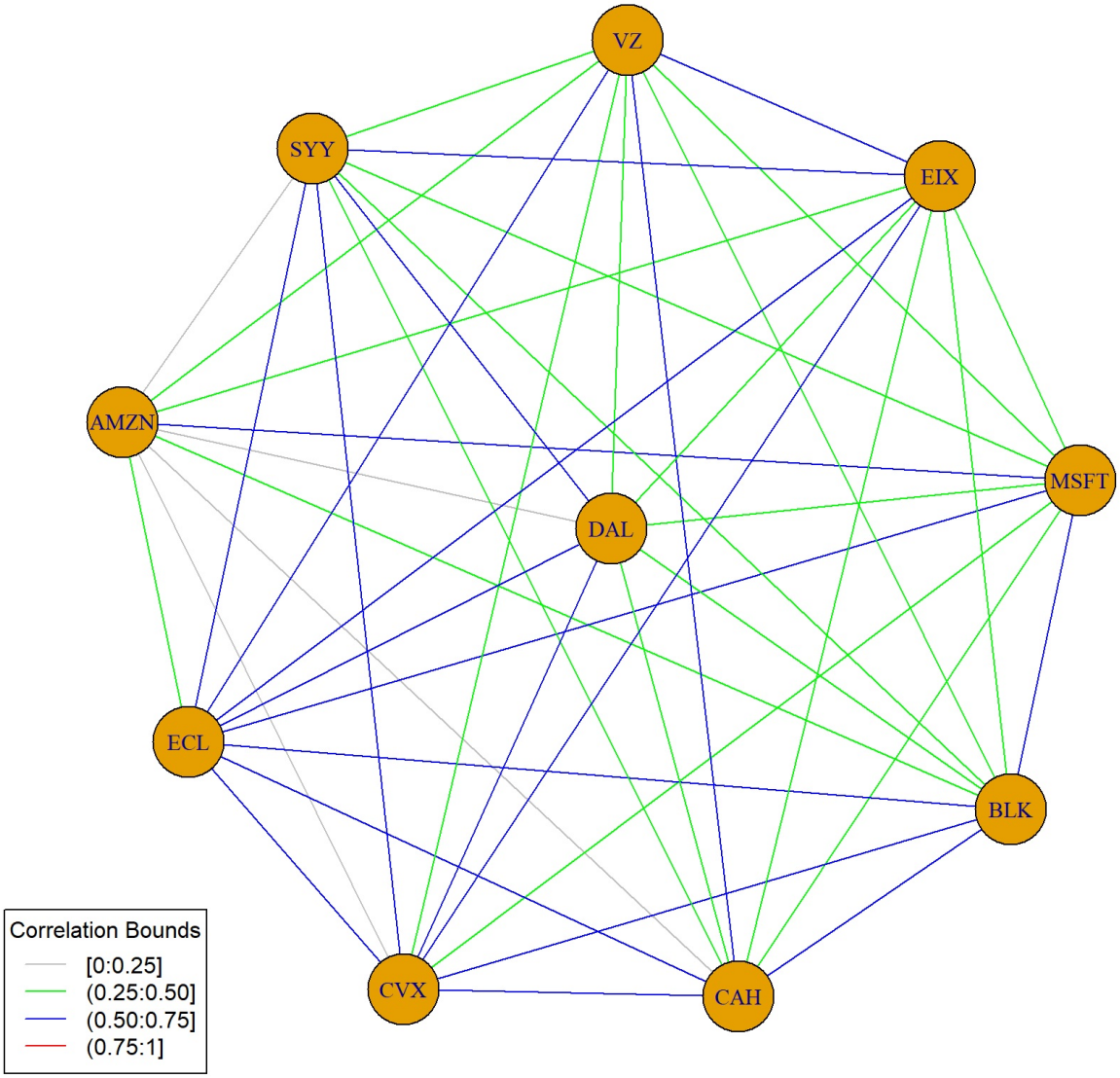
	AMZN	MSFT	BLK	CAH	VZ	CVX	ECL	EIX	DAL	SYY
AMZN	1.0000000	0.6995065	0.3449303	0.2237681	0.2642836	0.1856723	0.2743749	0.2616399	0.1183332	0.1944775
MSFT	0.6995065	1.0000000	0.5919804	0.4622477	0.4560432	0.4746435	0.5551238	0.4626765	0.2781587	0.4046860
BLK	0.3449303	0.5919804	1.0000000	0.5182275	0.4573839	0.5462833	0.6335207	0.4960012	0.3671207	0.3592832
CAH	0.2237681	0.4622477	0.5182275	1.0000000	0.5297572	0.5221106	0.5218033	0.4743590	0.3996110	0.4234221
VZ	0.2642836	0.4560432	0.4573839	0.5297572	1.0000000	0.4614280	0.5200294	0.5591718	0.2793635	0.4353010
CVX	0.1856723	0.4746435	0.5462833	0.5221106	0.4614280	1.0000000	0.6577619	0.5771347	0.6149195	0.6406492
ECL	0.2743749	0.5551238	0.6335207	0.5218033	0.5200294	0.6577619	1.0000000	0.6071047	0.5329082	0.5774348
EIX	0.2616399	0.4626765	0.4960012	0.4743590	0.5591718	0.5771347	0.6071047	1.0000000	0.4313016	0.5845748
DAL	0.1183332	0.2781587	0.3671207	0.3996110	0.2793635	0.6149195	0.5329082	0.4313016	1.0000000	0.6648280
SYY	0.1944775	0.4046860	0.3592832	0.4234221	0.4353010	0.6406492	0.5774348	0.5845748	0.6648280	1.0000000

```
## List of Companies
companies <- c(names(port))

## Init Graph (Undirected)
net <- make_empty_graph(10, F)

## Vertex names
V(net)$name <- companies

## Colors
colors <- c('gray', 'green', 'blue', 'red')
```

We place an edge only if 0 doesn't belong in the confidence interval and we choose the color of this edge based on the correlation value.

```
for (i in 1:(length(companies)-1)){
  for (j in (i+1):length(companies)){
    cor_value <- cor.test(port[,i],
                          port[,j])
    conf_int <- cor_value$conf.int
    est_cor  <- cor_value$estimate
    if (!((conf_int[1]<= 0) & (conf_int[2]>=0))){
      if (est_cor <= 0.25)
        col <- colors[1]
      else if (est_cor <= 0.50)
        col <- colors[2]
      else if (est_cor <= 0.75)
        col <- colors[3]
      else
        col <- colors[4]
      net <- add_edges(net, c(companies[i],
                              companies[j]),
                       color = col)
    }
  }
}
```

```
## Plot Graph
plot(net)
legend('bottomleft',
       legend = c('[0:0.25]', '(0.25:0.50]',
                  '(0.50:0.75]', '(0.75:1]'),
       title = 'Correlation Bounds',
       lty = 1,
       col = colors)
```



We choose to pick one company from each *GICS Sector* in order to diversificate our portfolio, this is also an attempt to reduce correlations (e.g. we imagine that Facebook and Google are more co-related than Google and an energy company) between stocks. We also choose a *95% Normal* confidence interval setting $\alpha = 0.05$. With this setting no correlation contains 0 within the range and no correlation exceeds the value of 0.75 and this, in our graph, results in a complete graph, despite this we noticed that more than a few stocks are weakly correlated with others and this is the reason why we choose to place edges of different colors.

```
## Estimate Shapley Values
sv <- rep(0, ncol(port))
sv_hat <- function(X, omega = 0.7){
  for (i in 1:ncol(port)){
    sv[i] <- mean(X[,i]) + sum(cov(X)[,i]) * omega
  }
  return(sv)
}

## Check
shapley <- sv_hat(port)
```

```
## Bootstrap
B <- 1000
sv_boot <- rep(NA, B)

## Cycle
for (i in 1:B){
  idx       <- sample(1:nrow(port), replace = T)
  port_boot <- port[idx, ]       ## Bootrap sample
  sv_boot[i]<- sv_hat(port_boot) ## Bootstrap Shapley Value
}

## Check
c(head(sv_boot / B, 10))
```

```
 [1] 0.000002419221 0.000002770367 0.000001139065 0.000002462952 0.000001547588
 [6] 0.000003860828 0.000002420715 0.000000723538 0.000003174543 0.000001756395
```

```
## Bootstrap Standard Error
se_boot <- sqrt(var(sv_boot))

## Confidence Level
alpha   <- 0.05

## Bootstrap / Normal CI / 95%
z <- qnorm(1 - alpha / 2)

## Check
df <- data.frame(shapley - z * se_boot,
                 shapley,
                 shapley + z * se_boot)
## Rename columns
colnames(df) <- c('LowerBound', 'Value', 'UpperBound')
rownames(df) <- companies

## View
kable(df, allign = 'ccc')
```

	LowerBound	Value	UpperBound
AMZN	0.0005201	0.0023099	0.0040998
MSFT	0.0015640	0.0033539	0.0051437
BLK	0.0005336	0.0023235	0.0041133
CAH	0.0002538	0.0020436	0.0038335
VZ	-0.0010659	0.0007239	0.0025138
CVX	0.0011818	0.0029716	0.0047615
ECL	0.0009460	0.0027359	0.0045257
EIX	0.0000942	0.0018841	0.0036740
DAL	0.0010460	0.0028359	0.0046258
SYY	0.0012553	0.0030452	0.0048351

Setting $\omega = 0.7$, $\alpha = 0.05$ and based on a *Normal Confidence Interval*, we estimate the *Shapley Values* for each stock with the respective CI, based on Bootstrap sampling. We have chosen this value of ω because we think that *expected return* of a stock is more important than the *volatility* that refers to a degree of fluctuation of the asset returns because we assume that when someone invests his money he is interesting in the expected revenue despite the possible fluctuations that the *title* may have.

We discussed about this homework with Cruoglio Antonella and Iovino Giuliana.