# Modeling Tail Dependence

Domenica Melone

## Table of contents



## Libraries and Data Loading

```
library(FRAPO)
library(readxl)
library(dplyr)
library(corrr)
library(ggplot2)
library(tidyr)
library(patchwork)
```

We start by uploading the excel file containing all the data.

```r
rm(list=ls())
data_table = readxl::read_xlsx("COMPLETE_TABLE.xlsx")
data_table %>% head ()
```

```
# A tibble: 6 x 7
  COUNTRY       YEAR  LIFEXP     GDP INFLATION   STOCK TEN_Y
  <chr>         <chr> <chr>    <dbl>     <dbl>   <dbl> <dbl>
1 UNITED STATES 1933  60.90   NA        NA      NA      3.31
2 UNITED STATES 1934  60.19    0.103     0.0150 -0.0617  3.11
3 UNITED STATES 1935  60.91    0.0853    0.0294  0.347   2.78
4 UNITED STATES 1936  60.35    0.122     0.0144  0.246   2.65
5 UNITED STATES 1937  61.03    0.0500    0.0282 -0.488   2.69
6 UNITED STATES 1938  62.39   -0.0351   -0.0282  0.226   2.55
```

## Data Cleaning

```r
data_table = data_table %>%
  mutate(
    LIFEXP = as.numeric(LIFEXP),
    YEAR = as.numeric(YEAR),
    COUNTRY = as.factor(COUNTRY))
# coverting the data type of
# the columns LIFEXP and YEAR to numeric

# converting the data type of
# the column COUNTRY to factor (categorical variable)
```

We group the dataset by country so that we can perform future operations separately.

```r
data_table = data_table %>% group_by(COUNTRY)
```

We check for NA values and expect to find one for each country in the three variables GDP, inflation, and stock, since we lose the first observation when computing the logarithmic change.

```r
data_table %>% is.na() %>% colSums()
```

```
  COUNTRY      YEAR    LIFEXP       GDP INFLATION     STOCK     TEN_Y
        0         0         0         6         6         6         0
```

## Summary Statistics

We can now calculate some summary statistics for the dataset and the relevant variables.

```
data_table %>%
  summarise(observations = n(),
            'first year' = min(YEAR),
            'last year' = max(YEAR))
```

```
# A tibble: 6 x 4
  COUNTRY        observations `first year` `last year`
  <fct>                 <int>        <dbl>       <dbl>
1 AUSTRALIA               100         1921        2020
2 FRANCE                  142         1880        2021
3 JAPAN                    76         1947        2022
4 SWEDEN                  123         1901        2023
5 UNITED KINGDOM          100         1922        2021
6 UNITED STATES            89         1933        2021
```

```
data_table %>%
  summarize(
    min = min(LIFEXP),
    max = max(LIFEXP),
    median = median(LIFEXP),
    mean = mean(LIFEXP),
    SD = sd(LIFEXP)) %>%
  mutate(across(
    min:SD,
    ~ round(., digits = 1))) %>%
  print()
```

```
# A tibble: 6 x 6
  COUNTRY          min   max median  mean    SD
  <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AUSTRALIA       61    83.7   71.3  72.8   6.4
2 FRANCE          34.8  82.7   66.2  63    14.1
3 JAPAN           51.7  84.7   77.7  75.5   7.8
4 SWEDEN          49.7  83.2   73.5  71.1   8.8
5 UNITED KINGDOM  57    81.4   72.1  71.4   7
6 UNITED STATES   60.2  79     73.2  72.2   5.3
```

```
data_table %>%
  summarize(
```

```r
    min = min(GDP, na.rm = TRUE),
    max = max(GDP, na.rm = TRUE),
    median = median(GDP, na.rm = TRUE),
    mean = mean(GDP, na.rm = TRUE),
    SD = sd(GDP, na.rm = TRUE)) %>%
  mutate(across(
    min:SD,
    ~ round(.*100, digits = 1))) %>%
  print()
```

```
# A tibble: 6 x 6
  COUNTRY          min   max median  mean    SD
  <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AUSTRALIA      -11.4  13.7    3.5   3.3   3.3
2 FRANCE         -26.7  28.9    1.9   1.7   6.3
3 JAPAN           -5.7  12.8    3.3   4.4   4.3
4 SWEDEN         -10.7  19.7    2.8   2.8   4
5 UNITED KINGDOM -10.9  10.7    2.6   2.2   3.2
6 UNITED STATES  -11.7  17      3.1   3.6   4.1
```

```r
data_table %>%
  summarize(
    min = min(STOCK, na.rm = TRUE),
    max = max(STOCK, na.rm = TRUE),
    median = median(STOCK, na.rm = TRUE),
    mean = mean(STOCK, na.rm = TRUE),
    SD = sd(STOCK, na.rm = TRUE)) %>%
  mutate(across(
    min:SD,
    ~ round(.*100, digits = 1))) %>%
  print()
```

```
# A tibble: 6 x 6
  COUNTRY          min   max median  mean    SD
  <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AUSTRALIA      -56.2  46.8    7.9   5.7  17.4
2 FRANCE         -56.4 104.     4     5.7  21.2
3 JAPAN          -54.1  78.1    7.8   8.4  24.5
4 SWEDEN         -61.7  61.7    6.5   6.1  21.5
5 UNITED KINGDOM -80.6  86      7.5   5.1  19.5
6 UNITED STATES  -48.8  37.2   11     7    17.1
```

```
data_table %>%
  summarize(
    min = min(TEN_Y),
    max = max(TEN_Y),
    median = median(TEN_Y),
    mean = mean(TEN_Y),
    SD = sd(TEN_Y)) %>%
  mutate(across(
    min:SD,
    ~ round(., digits = 1))) %>%
  print()
```

```
# A tibble: 6 x 6
  COUNTRY          min   max median  mean    SD
  <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AUSTRALIA        0.9  15.4    5.3   6.2   3.2
2 FRANCE          -0.1  16.3    4.3   5.1   2.9
3 JAPAN           -0.1  12.2    5.5   5     3.3
4 SWEDEN           0    13.5    4.3   5.2   3.1
5 UNITED KINGDOM   0.7  15.2    4.7   5.9   3.2
6 UNITED STATES    0.9  13.9    4.1   4.9   2.9
```

```
data_table %>%
  summarize(
    min = min(INFLATION, na.rm = TRUE),
    max = max(INFLATION, na.rm = TRUE),
    median = median(INFLATION, na.rm = TRUE),
    mean = mean(INFLATION, na.rm = TRUE),
    SD = sd(INFLATION, na.rm = TRUE)) %>%
  mutate(across(
    min:SD,
    ~ round(.*100, digits = 1))) %>%
  print()
```

```
# A tibble: 6 x 6
  COUNTRY          min   max median  mean    SD
  <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AUSTRALIA       -9.8  22.3    2.8   3.9   4.5
2 FRANCE         -27.2  55.4    2.1   5.6  10.8
3 JAPAN           -1.7  49.7    1.7   3.5   6.6
4 SWEDEN         -29    41.6    2.2   3.5   6.8
5 UNITED KINGDOM  -8.2  22.3    2.8   3.7   4.6
6 UNITED STATES   -2.8  16.7    2.8   3.5   3.2
```

For visualization, we can examine the boxplots. It is worthwhile to investigate the boxplot of life expectancy both for the original variable measured in years and for its logarithmic change, which represents our mortality index in the analysis.

```
pdf('lifexp.pdf', width = 20, height = 10)
# to save the boxplots as pdf
# and import them in latex

# first boxplot for life expectancy

bp1 = data_table %>%
  ggplot(aes(x = COUNTRY, y = LIFEXP)) +
  geom_boxplot() +
  theme_minimal() +
  labs(x = " ", y = "Life Expectancy")

# ggplot() initializes a ggplot object

# aes sets
# the variable 'country' on the x-axis
# the variable 'life expectancy' on the y-axis

# + geom_boxplot() adds the boxplot

# the other commands are for the look of the plot
# including theme and labels

# second boxplot for
# the logarithmic change of life expectancy

bp2 = data_table %>%
  mutate(log_diff_LIFEXP = c(NA, diff(log(LIFEXP)))) %>%
  ggplot(aes(x = COUNTRY, y = log_diff_LIFEXP)) +
  geom_boxplot() +
  theme_minimal() +
  labs(x = " ", y = "Logarithmic Change of Life Expectancy")

# combining the two boxplots side by side
# using the library patchwork
combined_plot = bp1 + bp2 + plot_layout(ncol = 2)

print(combined_plot)

dev.off()
# for the pdf export
```

For the economic variables, we examine the boxplot of the logarithmic changes for all variables, except the 10-year government yield, which will not be transformed in the analysis. For the 10-year government yield, we will use the boxplot of the original variable.

```
pdf('eco_var.pdf', width = 20, height = 10)

# transforming the data for easier plotting
data_long = data_table %>%
  pivot_longer(cols = c(STOCK, TEN_Y, INFLATION, GDP),
               names_to = "variable",
               values_to = "value") %>%
  # pivot_longer()
  # from the tidyverse package
  # "lengthens" data
  # by increasing the number of rows
  # and decreasing the number of columns
  mutate(variable = recode(variable,
                           STOCK = "Stock Index",
                           TEN_Y = "10 Years Government Yield",
                           INFLATION = "Consumer Price Index",
                           GDP = "Gross Domestic Product"))

  # boxplots
ggplot(data_long, aes(x = COUNTRY, y = value)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales = "free_y") +
  # this allows to
  # create separate plots for each variable
  # and arranges them in a grid layout

  # "free_y" allows each plot to have its own y-axis scale
  # needed here because the variables have different ranges

  theme_minimal() +
  labs(x = " ",
       y = " ")

dev.off()
```

## Ten Worst Years and Average Performances

We take the logarithmic change, defined as below, for the variable Life Expectancy.

$$x_i = ln(p_i) - ln(p_{i-1})$$

We furthermore exclude the two World Wars years, except for Sweden which stayed neutral.

```
data_table = data_table %>%
  mutate(LIFEXP = c(NA, diff(log(LIFEXP)))) %>%
  filter(!(YEAR %in% c(1914:1918, 1939:1945) & cur_group()$COUNTRY != "SWEDEN"))

data_table %>% head()
```

```
# A tibble: 6 x 7
# Groups:   COUNTRY [1]
  COUNTRY        YEAR   LIFEXP     GDP INFLATION   STOCK TEN_Y
  <fct>         <dbl>    <dbl>   <dbl>     <dbl>   <dbl> <dbl>
1 UNITED STATES  1933 NA        NA        NA       NA       3.31
2 UNITED STATES  1934 -0.0117    0.103     0.0150 -0.0617  3.11
3 UNITED STATES  1935  0.0119    0.0853    0.0294  0.347   2.78
4 UNITED STATES  1936 -0.00924   0.122     0.0144  0.246   2.65
5 UNITED STATES  1937  0.0112    0.0500    0.0282 -0.488   2.69
6 UNITED STATES  1938  0.0220   -0.0351   -0.0282  0.226   2.55
```

We can now look for the 10 worst years of the mortality index (the logarithmic change of Life Expectancy).

```
ten_worst_years_df = data_table %>%
  arrange(LIFEXP) %>%
  slice(1:10) %>%
  arrange(YEAR, .by_group = T)

ten_worst_years_df %>%
  select(1:3) %>%
  mutate(LIFEXP = round(LIFEXP*100, 2)) %>%
  print(n = Inf)
```

```
# A tibble: 60 x 3
# Groups:   COUNTRY [6]
   COUNTRY         YEAR LIFEXP
   <fct>          <dbl>  <dbl>
 1 AUSTRALIA       1923  -1.85
 2 AUSTRALIA       1926  -0.46
 3 AUSTRALIA       1934  -0.98
 4 AUSTRALIA       1946  -0.67
 5 AUSTRALIA       1951  -0.44
 6 AUSTRALIA       1959  -0.58
 7 AUSTRALIA       1962  -0.31
```

| | | | |
|---|---|---|---|
| 8 | AUSTRALIA | 1964 | -0.55 |
| 9 | AUSTRALIA | 1968 | -0.52 |
| 10 | AUSTRALIA | 1970 | -0.59 |
| 11 | FRANCE | 1884 | -1.89 |
| 12 | FRANCE | 1886 | -1.77 |
| 13 | FRANCE | 1890 | -4.74 |
| 14 | FRANCE | 1898 | -3.98 |
| 15 | FRANCE | 1899 | -1.84 |
| 16 | FRANCE | 1906 | -1.35 |
| 17 | FRANCE | 1911 | -6.31 |
| 18 | FRANCE | 1925 | -1.53 |
| 19 | FRANCE | 1929 | -2.13 |
| 20 | FRANCE | 1949 | -1.35 |
| 21 | JAPAN | 1956 | -0.23 |
| 22 | JAPAN | 1957 | -0.18 |
| 23 | JAPAN | 1980 | -0.04 |
| 24 | JAPAN | 1988 | -0.11 |
| 25 | JAPAN | 1995 | -0.19 |
| 26 | JAPAN | 2005 | -0.13 |
| 27 | JAPAN | 2010 | -0.06 |
| 28 | JAPAN | 2011 | -0.3 |
| 29 | JAPAN | 2021 | -0.15 |
| 30 | JAPAN | 2022 | -0.57 |
| 31 | SWEDEN | 1905 | -1.55 |
| 32 | SWEDEN | 1908 | -0.95 |
| 33 | SWEDEN | 1910 | -1.03 |
| 34 | SWEDEN | 1914 | -0.75 |
| 35 | SWEDEN | 1915 | -1.82 |
| 36 | SWEDEN | 1918 | -16.8 |
| 37 | SWEDEN | 1924 | -1.55 |
| 38 | SWEDEN | 1927 | -1.9 |
| 39 | SWEDEN | 1944 | -1.44 |
| 40 | SWEDEN | 2020 | -0.75 |
| 41 | UNITED KINGDOM | 1924 | -2.1 |
| 42 | UNITED KINGDOM | 1927 | -1.03 |
| 43 | UNITED KINGDOM | 1929 | -3.9 |
| 44 | UNITED KINGDOM | 1931 | -1.28 |
| 45 | UNITED KINGDOM | 1936 | -0.34 |
| 46 | UNITED KINGDOM | 1949 | -0.38 |
| 47 | UNITED KINGDOM | 1951 | -0.57 |
| 48 | UNITED KINGDOM | 1968 | -0.54 |
| 49 | UNITED KINGDOM | 1972 | -0.35 |
| 50 | UNITED KINGDOM | 2020 | -1.3 |
| 51 | UNITED STATES | 1934 | -1.17 |
| 52 | UNITED STATES | 1936 | -0.92 |
| 53 | UNITED STATES | 1957 | -0.33 |

```
54 UNITED STATES   1962  -0.19
55 UNITED STATES   1963  -0.24
56 UNITED STATES   1968  -0.44
57 UNITED STATES   1993  -0.28
58 UNITED STATES   2015  -0.2
59 UNITED STATES   2020  -2.46
60 UNITED STATES   2021  -0.81
```

We now compute the average over the entire sample ('full sample'), over the ten worst years ('tail'), over the subsequent years of the ten worst years ('tail +1 year'). For the stock index and the 10 years government yield, we also compute the reduction R, as in the formula below, to grasp a better idea of how much the average over the tail sample ($\bar{x}_t$) is smaller (or bigger!) with respect to the average over the full sample ($\bar{x}_s$).

$$R = \frac{\bar{x}_s - \bar{x}_t}{\bar{x}_s}$$

```
# 'full sample' in the perfomances table
full_sample = data_table %>%
  summarise(
    across(GDP:TEN_Y,
           \(x) mean(x, na.rm = TRUE)))

full_sample %>%
  mutate(
    across(GDP:STOCK, ~ round(.*100, 2)),
    TEN_Y = round(TEN_Y, 2)) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY          GDP INFLATION STOCK TEN_Y
  <fct>          <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA       3.33      3.97  5.91  6.37
2 FRANCE          2.3       4.27  5.07  5.18
3 JAPAN           4.38      3.51  8.39  4.96
4 SWEDEN          2.79      3.51  6.11  5.2
5 UNITED KINGDOM  2.15      3.64  5.02  6.07
6 UNITED STATES   3.06      3.44  7.26  5.12
```

```
# 'tail' in the perfomances table
extreme = ten_worst_years_df %>%
  summarise(
    across(GDP:TEN_Y,
           \(x) mean(x, na.rm = TRUE)))
```

```
extreme %>%
  mutate(
    across(GDP:STOCK, ~ round(.*100, 2)),
    TEN_Y = round(TEN_Y, 2)) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY          GDP INFLATION STOCK TEN_Y
  <fct>          <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA       4.24      4.73  7.47  4.71
2 FRANCE          2.81      2.25 -0.4   3.8
3 JAPAN           3.68      1.62  6.9   3.91
4 SWEDEN          1.36      5.52  8.93  3.64
5 UNITED KINGDOM  1.85      2.36  1.78  4.53
6 UNITED STATES   4.58      2.5   6     3.33
```

```
# 'tail + 1 year' in the perfomances table for the GDP
extremeplus1 = data_table %>%
  arrange(LIFEXP, .by_group = T) %>%
  filter(YEAR %in% (YEAR[1:10] + 1), .preserve = T) %>%
  summarise('tail + 1 year' = mean(GDP))

extremeplus1 %>%
  mutate(`tail + 1 year` = round(`tail + 1 year` * 100, 2)) %>%
  print()
```

```
# A tibble: 6 x 2
  COUNTRY        `tail + 1 year`
  <fct>                    <dbl>
1 AUSTRALIA                 4.47
2 FRANCE                    2.59
3 JAPAN                     3.09
4 SWEDEN                    7.43
5 UNITED KINGDOM            2.88
6 UNITED STATES             4.42
```

```
# 'reduction' in the perfomances table
# for stock and 10y gov. yield
# results are in percentage
reduction = extreme %>%
  select(COUNTRY, STOCK, TEN_Y) %>%
  mutate(
    across(STOCK:TEN_Y,
```

```
          ~ ((full_sample[[cur_column()]] - .) / full_sample[[cur_column()]])))

reduction %>%
  mutate(
    across(c(STOCK, TEN_Y), ~ round(. * 100, 1))) %>%
  print()
```

```
# A tibble: 6 x 3
  COUNTRY        STOCK TEN_Y
  <fct>          <dbl> <dbl>
1 AUSTRALIA      -26.5  26.1
2 FRANCE         108    26.6
3 JAPAN           17.8  21.2
4 SWEDEN         -46.2  30
5 UNITED KINGDOM  64.5  25.3
6 UNITED STATES   17.4  34.8
```

Given

$$p\text{-value} = \mathbb{P}(X \leq x),$$

we compute the p-value for the tail averages using bootstrap with replacement and using the definition of empirical C.D.F.

Definition: Let $X_1, X_2, \dots, X_n$ be i.i.d. rv's, then the *empirical c.d.f.* is:

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(X_i \leq x).$$

```
# bootstrapping with replacement 10.000 sets of ten years of data

# number of botstrapped samples
num_bootstraps = 10000

# empty tibbles to store bootrapped samples' average and correlation
average_datatable = tibble()
corr_datatable = tibble()
kendall_datatable = tibble()
spearman_datatable = tibble()

set.seed(999)
for (i in 1:num_bootstraps) {
  bs_datatable = data_table %>%
    slice_sample(n = 10, replace = TRUE)
```

```r
  # 10 random years table

  avg_dt = bs_datatable %>%
    summarise(across(GDP:TEN_Y, \(x) mean(x, na.rm = TRUE)))
  # average over 10 random years for the 4 variables
  average_datatable = bind_rows(average_datatable, avg_dt)

  cr_dt = bs_datatable %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs")))
  # correlation between the mortality index and the 4 variables
  # over 10 random years
  corr_datatable = bind_rows(corr_datatable, cr_dt)

  kendall_dt = bs_datatable %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "kendall")))
  # kendall tau between the mortality index and the 4 variables
  # over 10 random years
  kendall_datatable = bind_rows(kendall_datatable, kendall_dt)

  spearman_dt = bs_datatable %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "spearman")))
  # spearman rho between the mortality index and the 4 variables
  # over 10 random years
  spearman_datatable = bind_rows(spearman_datatable, spearman_dt)

}
```

```r
countries = unique(average_datatable$COUNTRY)
# retrieve countries
variables = names(average_datatable)[-1]
# retrieve macroeconomic/financial indicators

#ecdf_percountry = list()
# empty list to store ecdfs
# it is a list of lists (one list for each Country)

pvalues_extreme = tibble()
```

```
# a tibble to store the p-values
# of the 'tail' values
# in the perfomances table

pvalues_extremeplus1 = tibble()
# a tibble to store the p-values
# of the 'tail + 1 year' values
# for the GDP
# in the perfomances table

for (c in countries) {
    # iterating across countries

    #ecdf_pervariable = list()
    # empty list to store ecdfs
    # one ecdf per each macro./fin. indicator

    for (v in variables) {
      # iterating across macro./fin. indicators
        variable_values = average_datatable %>%
            filter(COUNTRY == c) %>%
            pull(v)
            # retrieving the values to estimate the ecdf
            # values are country & variable specific!

        ecdf_fun = ecdf(variable_values)
        # computing the ecdf

        #ecdf_pervariable[[v]] = ecdf_fun
        # storing the ecdf in the Country (e.g. Australia) list

        pv_extr = ecdf_fun(extreme %>% filter(COUNTRY == c) %>% pull(v))
        # using the newly estimated ecdf
        # to compute the pvalues of the 'tail' table

        new_row = tibble(
            COUNTRY = c,
            VARIABLE = v,
            P_VALUE = pv_extr)
        pvalues_extreme = bind_rows(pvalues_extreme, new_row)
        # saving the p-values in the first tibble

        if (v == 'GDP') {pv_extr1 = ecdf_fun(extremeplus1 %>%
                                   filter(COUNTRY == c) %>%
                                   pull('tail + 1 year'))
```

```
                        # only for the GDP
                        # we compute the p-values
                        # of the 'tail + 1 year' averages
                        new_row = tibble(
                            COUNTRY = c,
                            P_VALUE = pv_extr1)
                        pvalues_extremeplus1 = bind_rows(pvalues_extremeplus1,
                                                         new_row)
                        # saving the p-values in the second tibble
        }
    }
    #ecdf_percountry[[c]] = ecdf_pervariable
    # storing the Country (e.g. Australia) list in the list of lists
}

pvalues_extreme %>%
  mutate(P_VALUE = P_VALUE * 100) %>%
  print(n = Inf)
```

```
# A tibble: 24 x 3
   COUNTRY        VARIABLE  P_VALUE
   <chr>          <chr>       <dbl>
 1 AUSTRALIA      GDP         86.3
 2 AUSTRALIA      INFLATION   71.9
 3 AUSTRALIA      STOCK       59.0
 4 AUSTRALIA      TEN_Y        3.82
 5 FRANCE         GDP         67.8
 6 FRANCE         INFLATION   26.0
 7 FRANCE         STOCK       18.3
 8 FRANCE         TEN_Y        5.95
 9 JAPAN          GDP         30.3
10 JAPAN          INFLATION   12.8
11 JAPAN          STOCK       41.5
12 JAPAN          TEN_Y       15.6
13 SWEDEN         GDP         12.3
14 SWEDEN         INFLATION   83.7
15 SWEDEN         STOCK       65.1
16 SWEDEN         TEN_Y        4.42
17 UNITED KINGDOM GDP         34.8
18 UNITED KINGDOM INFLATION   19.3
19 UNITED KINGDOM STOCK       29.9
20 UNITED KINGDOM TEN_Y        5.78
21 UNITED STATES  GDP         92.7
22 UNITED STATES  INFLATION   16.7
23 UNITED STATES  STOCK       38.8
```

```
24 UNITED STATES   TEN_Y        1.4
```

```
pvalues_extremeplus1 %>%
  mutate(P_VALUE = P_VALUE * 100) %>%
  print()
```

```
# A tibble: 6 x 2
  COUNTRY        P_VALUE
  <chr>            <dbl>
1 AUSTRALIA         91.7
2 FRANCE            62.1
3 JAPAN             16.8
4 SWEDEN           100.
5 UNITED KINGDOM    78.2
6 UNITED STATES     90.3
```

## Measures of Dependence

### Pearson's rho

```
# 'full sample' in Pearson's rho table
corr_fullsample = data_table %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs")))
corr_fullsample %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY          GDP INFLATION STOCK TEN_Y
  <fct>          <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA      -0.13     -0.15  0     0.11
2 FRANCE          0.31      0.32  0.06 -0.04
3 JAPAN           0.43      0.71  0.39  0.14
4 SWEDEN          0.35     -0.41 -0.03 -0.01
5 UNITED KINGDOM -0.05     -0.14 -0.04 -0.05
6 UNITED STATES  -0.22      0.11 -0.01  0.09
```

```
# 'tail' in Pearson's rho table
corr_extreme = ten_worst_years_df %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs")))

corr_extreme %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY          GDP INFLATION STOCK TEN_Y
  <fct>          <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA       0.11      0.16 -0.15 -0.13
2 FRANCE         -0.58      0.03 -0.45  0.44
3 JAPAN           0.36     -0.11  0.35  0.35
4 SWEDEN          0.45     -0.93  0.48 -0.29
5 UNITED KINGDOM  0.09      0.45  0.29  0.15
6 UNITED STATES   0.08      0.1  -0.34  0.63
```

```
pvalues_corr = tibble()
# a tibble to store the tail correlations p-values

for (c in countries) {
    # iterating across countries
    for (v in variables) {
      # iterating across macro./fin. indicators
        variable_values = corr_datatable %>%
            filter(COUNTRY == c) %>%
            pull(v)
            # retrieving the values to estimate the ecdf
            # values are country & variable specific!

        ecdf_fun = ecdf(variable_values)
        # computing the ecdf

        pv_corr = ecdf_fun(corr_extreme %>% filter(COUNTRY == c) %>% pull(v))
        # using the newly estimated ecdf to compute the pvalues

        new_row = tibble(
            COUNTRY = c,
            VARIABLE = v,
            P_VALUE = pv_corr)
        pvalues_corr = bind_rows(pvalues_corr, new_row)
```

```
        # saving the p-values in the tibble
    }
}

pvalues_corr %>%
  mutate(P_VALUE = P_VALUE * 100) %>%
  print(n = Inf)
```

```
# A tibble: 24 x 3
   COUNTRY        VARIABLE  P_VALUE
   <chr>          <chr>       <dbl>
 1 AUSTRALIA      GDP         78.5
 2 AUSTRALIA      INFLATION   67.7
 3 AUSTRALIA      STOCK       28.5
 4 AUSTRALIA      TEN_Y       20.1
 5 FRANCE         GDP         10.5
 6 FRANCE         INFLATION   50.0
 7 FRANCE         STOCK        7.33
 8 FRANCE         TEN_Y       97.1
 9 JAPAN          GDP         32.2
10 JAPAN          INFLATION   22.5
11 JAPAN          STOCK       48.9
12 JAPAN          TEN_Y       54.0
13 SWEDEN         GDP         68.4
14 SWEDEN         INFLATION    3.72
15 SWEDEN         STOCK       92.8
16 SWEDEN         TEN_Y       13.1
17 UNITED KINGDOM GDP         67.4
18 UNITED KINGDOM INFLATION   90.1
19 UNITED KINGDOM STOCK       80.6
20 UNITED KINGDOM TEN_Y       74.3
21 UNITED STATES  GDP         70.4
22 UNITED STATES  INFLATION   40.0
23 UNITED STATES  STOCK       21.3
24 UNITED STATES  TEN_Y       95.2
```

### Kendall's tau

```
kendall_fullsample = data_table %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "kendall")))
```

```r
kendall_fullsample %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY           GDP INFLATION STOCK TEN_Y
  <fct>           <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA       -0.17      0.03  0.08  0.12
2 FRANCE           0.06     -0.01 -0.07  0
3 JAPAN            0.31      0.18  0.13  0.27
4 SWEDEN           0.07     -0.06 -0.04 -0.03
5 UNITED KINGDOM -0.08     -0.07  0.03 -0.01
6 UNITED STATES  -0.12      0.15  0.02  0.07
```

```r
# 'full sample' in Kendall's tau table
```

```r
kendall_extreme = ten_worst_years_df %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "kendall")))

kendall_extreme %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY           GDP INFLATION STOCK TEN_Y
  <fct>           <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA        0.07     -0.16 -0.29 -0.02
2 FRANCE          -0.38     -0.26 -0.29  0.29
3 JAPAN            0.2      -0.02  0.29  0.24
4 SWEDEN          -0.16     -0.33 -0.2  -0.47
5 UNITED KINGDOM  0.29      0.29  0.29 -0.02
6 UNITED STATES  -0.2      -0.2  -0.33  0.33
```

```r
# 'tail' in Kendall's tau table
```

```r
pvalues_kendall = tibble()
# a tibble to store the tail kendall's tau p-values

for (c in countries) {
    # iterating across countries
```

```r
    for (v in variables) {
      # iterating across macro./fin. indicators
        variable_values = kendall_datatable %>%
            filter(COUNTRY == c) %>%
            pull(v)
            # retrieving the values to estimate the ecdf
            # values are country & variable specific!

        ecdf_fun = ecdf(variable_values)
        # computing the ecdf

        pv_kendall = ecdf_fun(kendall_extreme %>% filter(COUNTRY == c) %>% pull(v))
        # using the newly estimated ecdf to compute the pvalues

        new_row = tibble(
            COUNTRY = c,
            VARIABLE = v,
            P_VALUE = pv_kendall)
        pvalues_kendall = bind_rows(pvalues_kendall, new_row)
        # saving the p-values in the tibble
    }
}

pvalues_kendall %>%
  mutate(P_VALUE = P_VALUE * 100) %>%
  print(n = Inf)
```

```
# A tibble: 24 x 3
   COUNTRY        VARIABLE  P_VALUE
   <chr>          <chr>       <dbl>
 1 AUSTRALIA      GDP         81.6
 2 AUSTRALIA      INFLATION   26
 3 AUSTRALIA      STOCK        9.1
 4 AUSTRALIA      TEN_Y       30.0
 5 FRANCE         GDP          7.98
 6 FRANCE         INFLATION   19.4
 7 FRANCE         STOCK       18.6
 8 FRANCE         TEN_Y       90.2
 9 JAPAN          GDP         32.2
10 JAPAN          INFLATION   22.9
11 JAPAN          STOCK       71.3
12 JAPAN          TEN_Y       45.5
13 SWEDEN         GDP         23.5
14 SWEDEN         INFLATION   16.6
15 SWEDEN         STOCK       30.3
```

```
16 SWEDEN         TEN_Y        2.82
17 UNITED KINGDOM GDP         91.8
18 UNITED KINGDOM INFLATION   90.4
19 UNITED KINGDOM STOCK       86.3
20 UNITED KINGDOM TEN_Y       50.4
21 UNITED STATES  GDP         40.8
22 UNITED STATES  INFLATION   12.3
23 UNITED STATES  STOCK       10.7
24 UNITED STATES  TEN_Y       83.0
```

### Spearman's rho

```
spearman_fullsample = data_table %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "spearman")))
spearman_fullsample %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY          GDP INFLATION STOCK TEN_Y
  <fct>          <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA      -0.23      0.04  0.12  0.18
2 FRANCE          0.08     -0.01 -0.1   0.01
3 JAPAN           0.44      0.26  0.2   0.39
4 SWEDEN          0.1      -0.1  -0.07 -0.04
5 UNITED KINGDOM -0.1      -0.11  0.05 -0.02
6 UNITED STATES  -0.15      0.22  0.04  0.08
```

```
# 'full sample' in Spearman's rho table
```

```
spearman_extreme = ten_worst_years_df %>%
    summarise(across(GDP:TEN_Y,
                     ~ cor(LIFEXP, .,
                           use = "pairwise.complete.obs",
                           method = "spearman")))

spearman_extreme %>%
  mutate(across(GDP:TEN_Y, ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY           GDP INFLATION STOCK TEN_Y
  <fct>           <dbl>     <dbl> <dbl> <dbl>
1 AUSTRALIA        0.18     -0.18 -0.39 -0.03
2 FRANCE          -0.54     -0.33 -0.48  0.39
3 JAPAN            0.32      0.04  0.44  0.32
4 SWEDEN          -0.16     -0.37 -0.25 -0.7
5 UNITED KINGDOM  0.36      0.59  0.44 -0.02
6 UNITED STATES  -0.22     -0.22 -0.39  0.49
```

```
# 'tail' in Spearman's rho table
```

```
pvalues_spearman = tibble()
# a tibble to store the tail spearman's rho p-values

for (c in countries) {
    # iterating across countries
    for (v in variables) {
      # iterating across macro./fin. indicators
        variable_values = spearman_datatable %>%
            filter(COUNTRY == c) %>%
            pull(v)
            # retrieving the values to estimate the ecdf
            # values are country & variable specific!

        ecdf_fun = ecdf(variable_values)
        # computing the ecdf

        pv_spearman = ecdf_fun(spearman_extreme %>% filter(COUNTRY == c) %>% pull(v))
        # using the newly estimated ecdf to compute the pvalues

        new_row = tibble(
            COUNTRY = c,
            VARIABLE = v,
            P_VALUE = pv_spearman)
        pvalues_spearman = bind_rows(pvalues_spearman, new_row)
        # saving the p-values in the tibble
    }
}

pvalues_spearman %>%
  mutate(P_VALUE = P_VALUE * 100) %>%
  print(n = Inf)
```

```
# A tibble: 24 x 3
```

```
   COUNTRY        VARIABLE  P_VALUE
   <chr>          <chr>       <dbl>
 1 AUSTRALIA      GDP         85.8
 2 AUSTRALIA      INFLATION   28.4
 3 AUSTRALIA      STOCK        8.48
 4 AUSTRALIA      TEN_Y       27.9
 5 FRANCE         GDP          5.65
 6 FRANCE         INFLATION   20.3
 7 FRANCE         STOCK       12.1
 8 FRANCE         TEN_Y       88.3
 9 JAPAN          GDP         35.4
10 JAPAN          INFLATION   27.8
11 JAPAN          STOCK       75.0
12 JAPAN          TEN_Y       41.9
13 SWEDEN         GDP         25.3
14 SWEDEN         INFLATION   21.9
15 SWEDEN         STOCK       31.5
16 SWEDEN         TEN_Y        1.35
17 UNITED KINGDOM GDP         89.2
18 UNITED KINGDOM INFLATION   97.1
19 UNITED KINGDOM STOCK       88.3
20 UNITED KINGDOM TEN_Y       49.9
21 UNITED STATES  GDP         43.5
22 UNITED STATES  INFLATION   13.5
23 UNITED STATES  STOCK       12.9
24 UNITED STATES  TEN_Y       86.1
```

## Coefficients of tail dependence

We are going to use the function tdc() that returns the pairwise tail dependence coefficients between n series. The TDCs are estimated non-parametrically using the empirical tail copula.

The first argument of the function is the matrix with the four economic indicators and the mortality indicator. We retain only the mortality indicator column which contains all coefficients of interest.

The threshold value k in the function is the upper/lower bound for the order statistics to be considered. By default is used

$$k = \sqrt{\text{number of rows of the matrix}}$$

### Upper tail dependence

```r
upper = tibble()

for (c in countries) {
  upper_matrix = data_table %>%
    ungroup() %>%
    # necessary because

    # data_table %>% filter(COUNTRY == 'FRANCE')
    # is equal to
    # data_table %>% ungroup() %>% filter(COUNTRY == 'FRANCE')

    # but

    # data_table %>% filter(COUNTRY == c)
    # %>% select(-YEAR, -COUNTRY) %>% tdc(method = "EmpTC", lower = FALSE)

    # adds back 'missing grouping variables: `COUNTRY`'
    # and tdc() computes the tail coeff. for country too (not meaningfull)

    # moreover, tdc() uses
    # k, the upper/lower bound for the order statistics,
    # equal to sqrt(nrow(x))
    # hence, having an addtional column 'COUNTRY' would
    # result in biased tail coeff.
    # so it is necessary to remove both YEAR and COUNTRY

    filter(COUNTRY == c) %>%
    select(-YEAR, -COUNTRY) %>%
    tdc(method = "EmpTC", lower = FALSE)

  row = tibble(
    COUNTRY = c,
    LEXP_GDP = upper_matrix["LIFEXP", "GDP"],
    LEXP_INFL = upper_matrix["LIFEXP", "INFLATION"],
    LEXP_STOCK = upper_matrix["LIFEXP", "STOCK"],
    LEXP_TENY = upper_matrix["LIFEXP", "TEN_Y"]
    )
  upper = bind_rows(upper, row)
}

upper %>%
  mutate(across(LEXP_GDP:LEXP_TENY,
                ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY        LEXP_GDP LEXP_INFL LEXP_STOCK LEXP_TENY
  <chr>             <dbl>     <dbl>      <dbl>     <dbl>
1 AUSTRALIA          0.11      0.22       0.33      0.22
2 FRANCE             0.55      0.36       0.18      0
3 JAPAN              0.5       0.38       0.5       0.12
4 SWEDEN             0.36      0.09       0.09      0
5 UNITED KINGDOM     0.22      0.11       0.11      0
6 UNITED STATES      0.22      0.44       0.44      0
```

#### Lower tail dependence

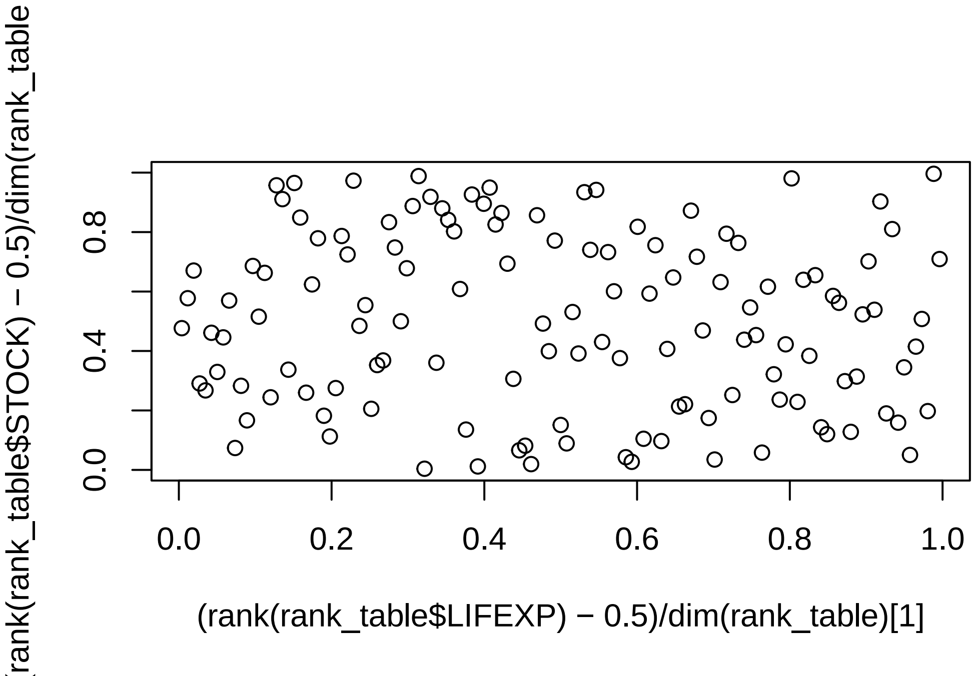
```r
lower = tibble()

for (c in countries) {
  lower_matrix = data_table %>%
    ungroup() %>%
    # necessary because

    # data_table %>% filter(COUNTRY == 'FRANCE')
    # is equal to
    # data_table %>% ungroup() %>% filter(COUNTRY == 'FRANCE')

    # but

    # data_table %>% filter(COUNTRY == c)
    # %>% select(-YEAR, -COUNTRY) %>% tdc(method = "EmpTC", lower = FALSE)

    # adds back 'missing grouping variables: `COUNTRY`'
    # and tdc() computes the tail coeff. for country too (not meaningfull)

    # moreover, tdc() uses
    # k, the upper/lower bound for the order statistics,
    # equal to sqrt(nrow(x))
    # hence, having an addtional column 'COUNTRY' would
    # result in biased tail coeff.
    # so it is necessary to remove both YEAR and COUNTRY

    filter(COUNTRY == c) %>%
    select(-YEAR, -COUNTRY) %>%
    tdc(method = "EmpTC", lower = TRUE)

  row = tibble(
    COUNTRY = c,
```

```
    LEXP_GDP = lower_matrix["LIFEXP", "GDP"],
    LEXP_INFL = lower_matrix["LIFEXP", "INFLATION"],
    LEXP_STOCK = lower_matrix["LIFEXP", "STOCK"],
    LEXP_TENY = lower_matrix["LIFEXP", "TEN_Y"]
  )
  lower <- bind_rows(lower, row)
}

lower %>%
  mutate(across(LEXP_GDP:LEXP_TENY,
                ~ round(., digits = 2))) %>%
  print()
```

```
# A tibble: 6 x 5
  COUNTRY        LEXP_GDP LEXP_INFL LEXP_STOCK LEXP_TENY
  <chr>             <dbl>     <dbl>      <dbl>     <dbl>
1 AUSTRALIA          0.11      0          0.11      0
2 FRANCE             0.09      0          0.09      0
3 JAPAN              0.12      0.12       0.12      0.25
4 SWEDEN             0.27      0.09       0.27      0.09
5 UNITED KINGDOM     0.22      0.33       0.11      0.11
6 UNITED STATES      0.11      0.11       0.11      0.33
```

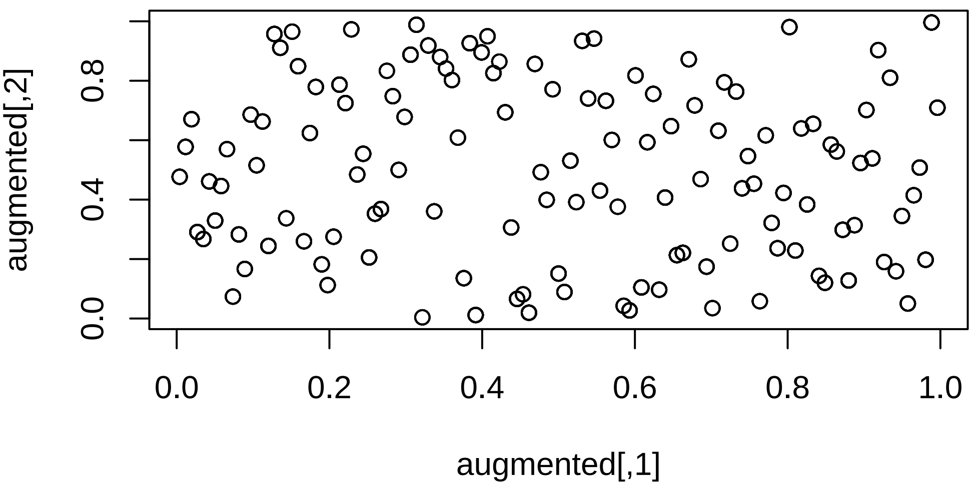
## Copulas

```
rank_table = data_table %>% filter(COUNTRY == 'FRANCE') %>% select('LIFEXP', 'STOCK') %>%

plot((rank(rank_table$LIFEXP)-0.5)/dim(rank_table)[1],(rank(rank_table$STOCK)-0.5)/dim(ran
```




```
library(copula)
library(copulaSim)
```

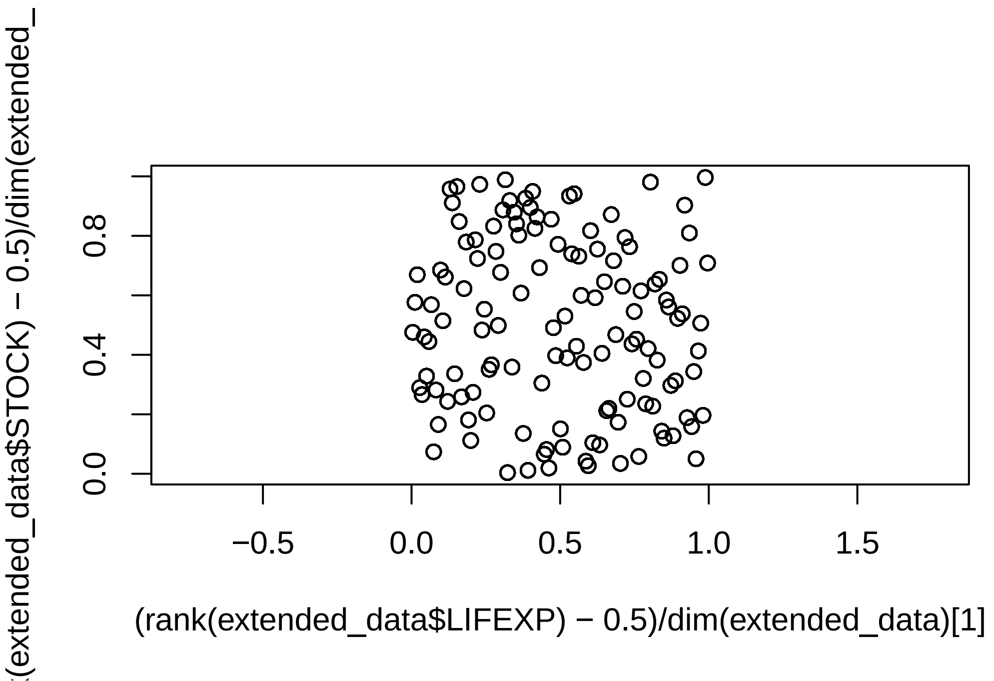
```
cop_data = cbind((rank(rank_table$LIFEXP)-0.5)/dim(rank_table)[1],(rank(rank_table$STOCK)-

fitted_cop = empCopula(cop_data)
augmented = rCopula(1000, fitted_cop)
plot(augmented)
```



## Block Bootstrap

Fot France and Lifexp & Stock

```
library('monotonicity')

rank_table = data_table %>% filter(COUNTRY == 'FRANCE') %>% select('LIFEXP', 'STOCK') %>%

set.seed(999)
statBOO_indices = statBootstrap(T = dim(rank_table)[1], bootstrapRep = 1000, block_length

all_indices <- as.vector(statBOO_indices)

bootstrap_data <- rank_table[all_indices, ]

extended_data <- rbind(rank_table, bootstrap_data)


plot((rank(extended_data$LIFEXP)-0.5)/dim(extended_data)[1],(rank(extended_data$STOCK)-0.5
```




```
png("comparison_plot.png", width = 800, height = 400)

par(mfrow = c(1,2))

plot((rank(extended_data$LIFEXP)-0.5)/dim(extended_data)[1],(rank(extended_data$STOCK)-0.5
```

```
     main = "Extended Data")

plot((rank(rank_table$LIFEXP)-0.5)/dim(rank_table)[1],(rank(rank_table$STOCK)-0.5)/dim(ran
     main = "Original Data")

dev.off()
```

```
pdf
  2
```

example: RANK REPEATS -> no change in the graph

```
ex = c(0.1, 0.55, 2, 0.87)

rank(ex)
```

```
[1] 1 2 4 3
```

```
ex_extended = c(0.1, 0.55, 2, 0.87, 0.55, 2, 0.87, 0.1)
rank(ex_extended)
```

```
[1] 1.5 3.5 7.5 5.5 3.5 7.5 5.5 1.5
```

should we plot each bootstrapped dataset individually and build a distribution of the ranked scatter plots (average and sd as for the statistics distribution BS is usually used for)?

No because we are going to end up with the only difference of having:

- fewer observations (BS with repl. is going to repeat (extract multiple times) some of the obs which will end up having the same rank and hence figure once (overlapping) on the rank scatterplot)
- the highest rank (once scaled by the number of obs) won't be 1
- the other obs will be the same (we will still get 1/129, 2/129, … )

```
set.seed(999)
fewBS_indices = statBootstrap(T = dim(rank_table)[1], bootstrapRep = 7, block_length = 2)

png("individualBS.png", width = 800, height = 800)  # Specify file name and dimensions

par(mfrow = c(4,2))

plot((rank(rank_table$LIFEXP)-0.5)/dim(rank_table)[1],(rank(rank_table$STOCK)-0.5)/dim(ran
```

```
for(i in 1:(ncol(fewBS_indices))){

bs_data <- rank_table[fewBS_indices[,i], ]

plot((rank(bs_data$LIFEXP)-0.5)/dim(bs_data)[1],(rank(bs_data$STOCK)-0.5)/dim(bs_data)[1],

}
dev.off()
```

```
pdf
  2
```