Course 3 Step 1.18 - Your turn

Jiaying Wu

Make a start with the daily exchange rate

```
# library
library(tidyverse)
library(gridExtra)

# Read exchange rate data
ru <- read_csv("https://raw.githubusercontent.com/datascienceprogram/ids_course_data/master/rates_new.cs

# Arrange data by date
ru <- ru %>%
  arrange(date)

# Look at the data
head(ru, n = 10)
```

```
## # A tibble: 10 x 169
##    date         AED   AFN   ALL   AMD   ANG   AOA   ARS   AUD   AWG   AZN   BAM
##    <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2017-06-20  3.67  68.1  119.  481.  1.78  166.  16.1  1.32  1.80  1.7   1.76
##  2 2017-06-21  3.67  68.1  119.  480.  1.78  166.  16.2  1.32  1.79  1.7   1.75
##  3 2017-06-22  3.67  68.1  119.  481.  1.78  166.  16.1  1.33  1.79  1.7   1.75
##  4 2017-06-23  3.67  68.1  118.  479.  1.78  166.  16.2  1.32  1.80  1.7   1.75
##  5 2017-06-24  3.67  68.1  118.  479.  1.78  166.  16.2  1.32  1.80  1.7   1.75
##  6 2017-06-25  3.67  67.9  118.  479.  1.78  166.  16.1  1.32  1.80  1.7   1.75
##  7 2017-06-26  3.67  68.1  118.  480.  1.78  166.  16.3  1.32  1.79  1.7   1.75
##  8 2017-06-27  3.67  67.6  117.  477.  1.78  166.  16.4  1.32  1.79  1.7   1.74
##  9 2017-06-28  3.67  68.0  116.  479.  1.78  166.  16.4  1.31  1.79  1.7   1.72
## 10 2017-06-29  3.67  67.9  116   479.  1.77  166.  16.5  1.30  1.80  1.69  1.71
## # ... with 157 more variables: BBD <dbl>, BDT <dbl>, BGN <dbl>, BHD <dbl>,
## #   BIF <dbl>, BMD <dbl>, BND <dbl>, BOB <dbl>, BRL <dbl>, BSD <dbl>,
## #   BTC <dbl>, BTN <dbl>, BWP <dbl>, BYN <dbl>, BZD <dbl>, CAD <dbl>,
## #   CDF <dbl>, CHF <dbl>, CLF <dbl>, CLP <dbl>, CNH <dbl>, CNY <dbl>,
## #   COP <dbl>, CRC <dbl>, CUC <dbl>, CUP <dbl>, CVE <dbl>, CZK <dbl>,
## #   DJF <dbl>, DKK <dbl>, DOP <dbl>, DZD <dbl>, EGP <dbl>, ERN <dbl>,
## #   ETB <dbl>, EUR <dbl>, FJD <dbl>, FKP <dbl>, GBP <dbl>, GEL <dbl>,
## #   GGP <dbl>, GHS <dbl>, GIP <dbl>, GMD <dbl>, GNF <dbl>, GTQ <dbl>,
## #   GYD <dbl>, HKD <dbl>, HNL <dbl>, HRK <dbl>, HTG <dbl>, HUF <dbl>,
## #   IDR <dbl>, ILS <dbl>, IMP <dbl>, INR <dbl>, IQD <dbl>, IRR <dbl>,
## #   ISK <dbl>, JEP <dbl>, JMD <dbl>, JOD <dbl>, JPY <dbl>, KES <dbl>,
## #   KGS <dbl>, KHR <dbl>, KMF <dbl>, KPW <dbl>, KRW <dbl>, KWD <dbl>,
## #   KYD <dbl>, KZT <dbl>, LAK <dbl>, LBP <dbl>, LKR <dbl>, LRD <dbl>,
## #   LSL <dbl>, LYD <dbl>, MAD <dbl>, MDL <dbl>, MGA <dbl>, MKD <dbl>,
```

```
## #   MMK <dbl>, MNT <dbl>, MOP <dbl>, MRO <dbl>, MUR <dbl>, MVR <dbl>,
## #   MWK <dbl>, MXN <dbl>, MYR <dbl>, MZN <dbl>, NAD <dbl>, NGN <dbl>,
## #   NIO <dbl>, NOK <dbl>, NPR <dbl>, NZD <dbl>, OMR <dbl>, PAB <dbl>, ...
```

```
names(ru)
```

```
##   [1] "date" "AED"  "AFN"  "ALL"  "AMD"  "ANG"  "AOA"  "ARS"  "AUD"  "AWG"
##  [11] "AZN"  "BAM"  "BBD"  "BDT"  "BGN"  "BHD"  "BIF"  "BMD"  "BND"  "BOB"
##  [21] "BRL"  "BSD"  "BTC"  "BTN"  "BWP"  "BYN"  "BZD"  "CAD"  "CDF"  "CHF"
##  [31] "CLF"  "CLP"  "CNH"  "CNY"  "COP"  "CRC"  "CUC"  "CUP"  "CVE"  "CZK"
##  [41] "DJF"  "DKK"  "DOP"  "DZD"  "EGP"  "ERN"  "ETB"  "EUR"  "FJD"  "FKP"
##  [51] "GBP"  "GEL"  "GGP"  "GHS"  "GIP"  "GMD"  "GNF"  "GTQ"  "GYD"  "HKD"
##  [61] "HNL"  "HRK"  "HTG"  "HUF"  "IDR"  "ILS"  "IMP"  "INR"  "IQD"  "IRR"
##  [71] "ISK"  "JEP"  "JMD"  "JOD"  "JPY"  "KES"  "KGS"  "KHR"  "KMF"  "KPW"
##  [81] "KRW"  "KWD"  "KYD"  "KZT"  "LAK"  "LBP"  "LKR"  "LRD"  "LSL"  "LYD"
##  [91] "MAD"  "MDL"  "MGA"  "MKD"  "MMK"  "MNT"  "MOP"  "MRO"  "MUR"  "MVR"
## [101] "MWK"  "MXN"  "MYR"  "MZN"  "NAD"  "NGN"  "NIO"  "NOK"  "NPR"  "NZD"
## [111] "OMR"  "PAB"  "PEN"  "PGK"  "PHP"  "PKR"  "PLN"  "PYG"  "QAR"  "RON"
## [121] "RSD"  "RUB"  "RWF"  "SAR"  "SBD"  "SCR"  "SDG"  "SEK"  "SGD"  "SHP"
## [131] "SLL"  "SOS"  "SRD"  "SSP"  "STD"  "SVC"  "SYP"  "SZL"  "THB"  "TJS"
## [141] "TMT"  "TND"  "TOP"  "TRY"  "TTD"  "TWD"  "TZS"  "UAH"  "UGX"  "USD"
## [151] "UYU"  "UZS"  "VEF"  "VND"  "VUV"  "WST"  "XAF"  "XAG"  "XAU"  "XCD"
## [161] "XDR"  "XOF"  "XPD"  "XPF"  "XPT"  "YER"  "ZAR"  "ZMW"  "ZWL"
```

1.How can you tell that the currencies in the data are quoted against the USD?

```
ru %>%
  select(date, USD, AUD, EUR, CNY, JPY, GBP, CHF) %>%
  head(n = 10)
```

```
## # A tibble: 10 x 8
##    date         USD   AUD   EUR   CNY   JPY   GBP   CHF
##    <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2017-06-20     1  1.32 0.898  6.83  111. 0.792 0.975
##  2 2017-06-21     1  1.32 0.896  6.83  111. 0.789 0.973
##  3 2017-06-22     1  1.33 0.897  6.83  111. 0.789 0.972
##  4 2017-06-23     1  1.32 0.893  6.84  111. 0.786 0.969
##  5 2017-06-24     1  1.32 0.893  6.84  111. 0.786 0.969
##  6 2017-06-25     1  1.32 0.893  6.83  111. 0.785 0.969
##  7 2017-06-26     1  1.32 0.894  6.84  112. 0.786 0.972
##  8 2017-06-27     1  1.32 0.882  6.81  112. 0.780 0.961
##  9 2017-06-28     1  1.31 0.879  6.80  112. 0.773 0.960
## 10 2017-06-29     1  1.30 0.874  6.79  112. 0.769 0.956
```

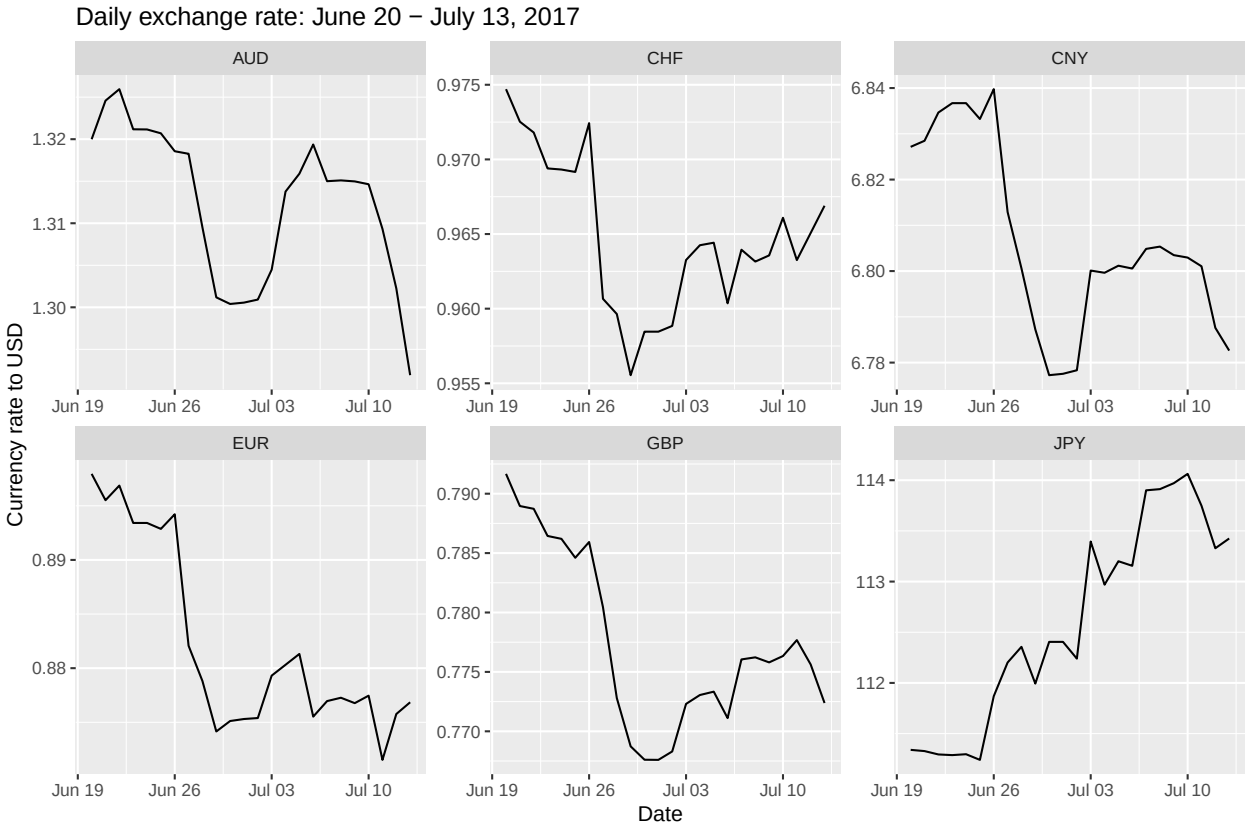
2.What is the first and last recorded exchange rate in this data set?

```
# First and last recording of exchange rates
ru %>%
  summarise(firstdate = first(date),
            lastdate = last(date))
```

```
## # A tibble: 1 x 2
##   firstdate  lastdate
##   <date>     <date>
## 1 2017-06-20 2017-07-13
```

3.Which exchange rate appreciated during this period?

```
# Line plot of various exchange rates
ru %>%
  select(date, AUD, EUR, CNY, JPY, GBP, CHF) %>%
  gather("currency", "rate", 2:7) %>%
  ggplot(aes(x = date, y = rate)) +
  geom_line() +
  labs(x = "Date",
       y = "Currency rate to USD",
       title = "Daily exchange rate: June 20 - July 13, 2017") +
  facet_wrap(~ currency, scales = "free")
```



The Japanese Yen.

4.When the Swiss franc was at its peak and trough, approximately how many Swiss francs could be exchanged for 1 United States dollar?

```
ru %>%
  select(date, USD, CHF) %>%
  filter(CHF == max(CHF))
```

```
## # A tibble: 1 x 3
##   date         USD   CHF
##   <date>     <dbl> <dbl>
## 1 2017-06-20     1 0.975
```

```
ru %>%
  select(date, USD, CHF) %>%
  filter(CHF == min(CHF))
```

```
## # A tibble: 1 x 3
##   date         USD   CHF
##   <date>     <dbl> <dbl>
## 1 2017-06-29     1 0.956
```

Peak: 0.975USD, trough: 0.955USD.

Pre-processing data

Create a data frame containing exchanges rate with a CV greater than 0.0027.

```
# Function of the CV
cv <- function(x) sd(x)/mean(x)

# Only analyse exchange rates that have moved substantially over this time
s <- ru %>%
  select(-date) %>% # dates dropped
  summarise_all(funs(cv)) %>% # compute CV of each exchange rate
  gather(curr, cv) %>%
  filter(cv > 0.0027)

# Exchange rates with CV greater than 0.0027
ru_sub <- ru %>% select(s$curr)
```

Remove non-exchange rates

```
# Remove non-exchange rates
ru_sub <- ru_sub %>%
  select(-ALL, -XAG, -XDR, -XPT)
```

5.Google these 'exchange rates': ALL, XAG, XDR, XPT. What are they?

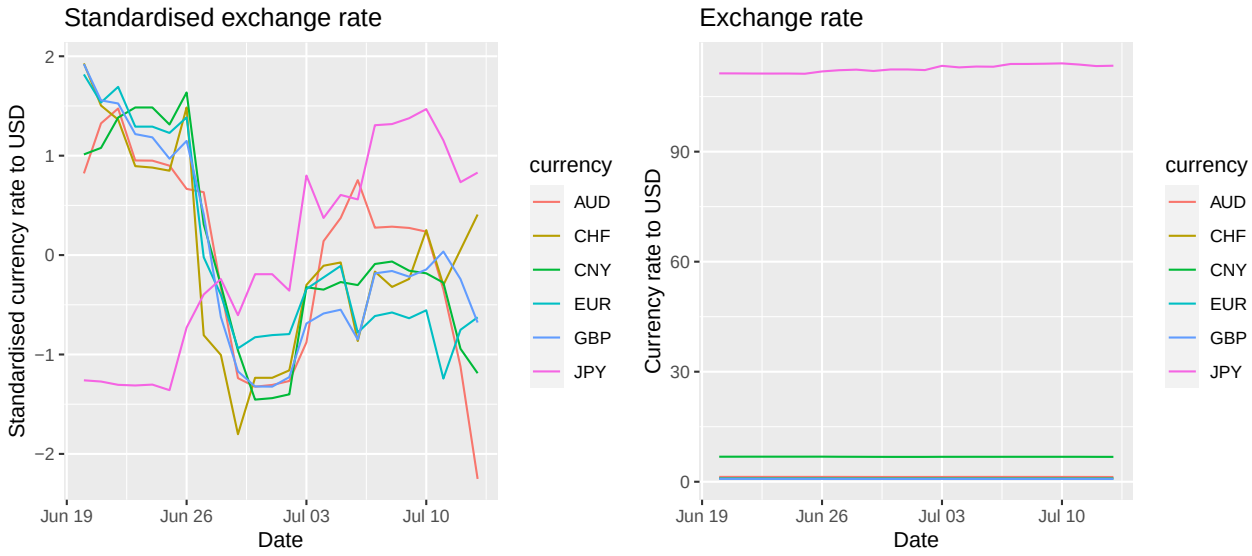
Albanian Lek, Silver, IMF, Platinum. Albanian lek is used inside the country, it cannot be exchanged to other currencies. Silver, IMF and Platinum are not country currencies.

Standardise the exchange rates

```
# Function to standardise exchange rates
scale01 <- function(x) (x-mean(x))/sd(x)

# Apply standardisation function to each exchange rate
ru_sub <-  ru_sub %>%
  mutate_all(funs(scale01))

# Join date variable from ru data frame into ru_sub
ru_sub_dt <- ru_sub %>%
  mutate(date = ru$date)
```

Generate line plots of some standardised and non-standardised exchange rates.

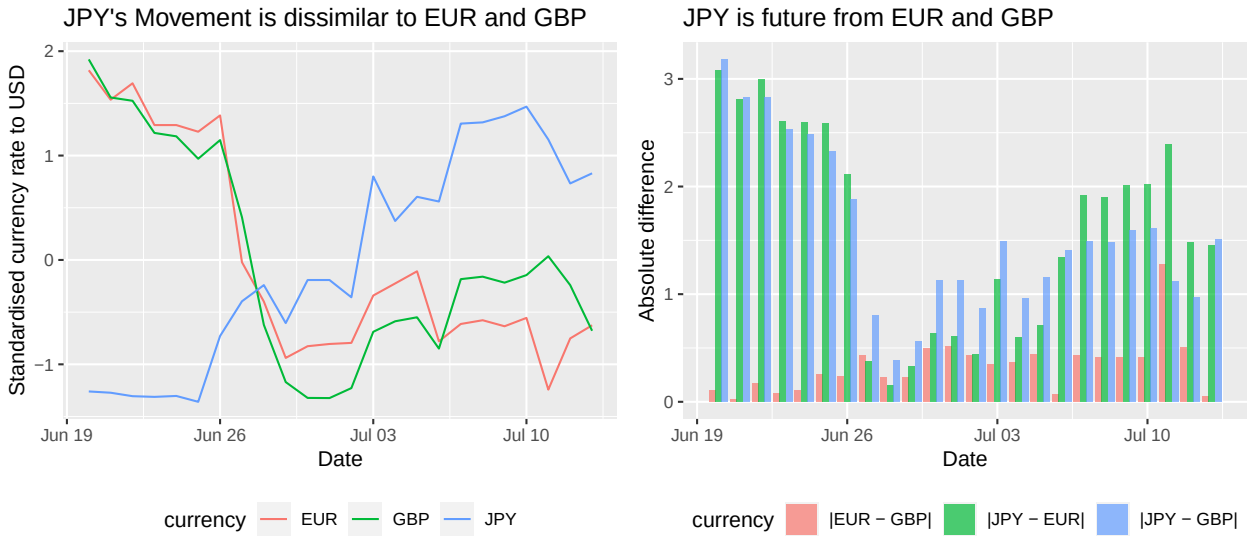
```
# Line plot: standardised exchange rates
line_plot_std_exch_rate <- ru_sub_dt %>%
  select(date, AUD, EUR, CNY, JPY, GBP, CHF) %>%
  gather("currency", "rate", 2:7) %>%
  ggplot(aes(x = date, y = rate, colour = currency)) +
  geom_line() +
  labs(x = "Date",
       y = "Standardised currency rate to USD",
       title = "Standardised exchange rate")

# Line plot: exchange rates
line_plot_exch_rate <- ru %>%
  select(date, AUD, EUR, CNY, JPY, GBP, CHF) %>%
  gather("currency", "rate", 2:7) %>%
  ggplot(aes(x = date, y = rate, colour = currency)) +
  geom_line() +
  labs(x = "Date",
       y = "Currency rate to USD",
       title = "Exchange rate")

# Compare standardised and non-standard exchange rate
grid.arrange(line_plot_std_exch_rate, line_plot_exch_rate, ncol = 2)
```



Distance between all pairs of exchange rates

Notice that some exchange rates move in a similar fashion, while others move in the opposite direction. If there is a large distance between standardised exchange rates, then this indicates that they do not move in a similar way. Exchange rates that move in a similar way should have little distance in their standardised figures.

```
# Line plot: JPY's Movement is dissimilar to EUR and GBP
line_plot_jpy <- ru_sub_dt %>%
  select(date, EUR, JPY, GBP) %>%
  gather("currency", "rate", 2:4) %>%
  ggplot(aes(x = date, y = rate, colour = currency)) +
  geom_line() +
  labs(x = "Date",
       y = "Standardised currency rate to USD",
       title = "JPY's Movement is dissimilar to EUR and GBP") +
  theme(legend.position = "bottom")

# bar plot: JPY is future from EUR and GBP
bar_plot_jpy <- ru_sub_dt %>%
  select(date, EUR, JPY, GBP) %>%
  mutate("|EUR - GBP|" = abs(EUR - GBP),
         "|JPY - EUR|" = abs(JPY - EUR),
         "|JPY - GBP|" = abs(JPY - GBP)) %>%
  gather("currency", "rate", 5:7) %>%
  ggplot(aes(x = date, y = rate, fill = currency)) +
  geom_col(position = "dodge", alpha = 0.7) +
  labs(x = "Date",
       y = "Absolute difference",
       title = "JPY is future from EUR and GBP") +
  theme(legend.position = "bottom")

grid.arrange(line_plot_jpy, bar_plot_jpy, ncol = 2)
```




```
# Transpose the data is that the dates are in the columns and exchange rates in rows

# t() function transposes the data, i.e., swap the rows with the columns
ru_sub_t <- t(ru_sub)

# Convert transposed data into a data frame
ru_sub_t <- data.frame(ru_sub_t)
```

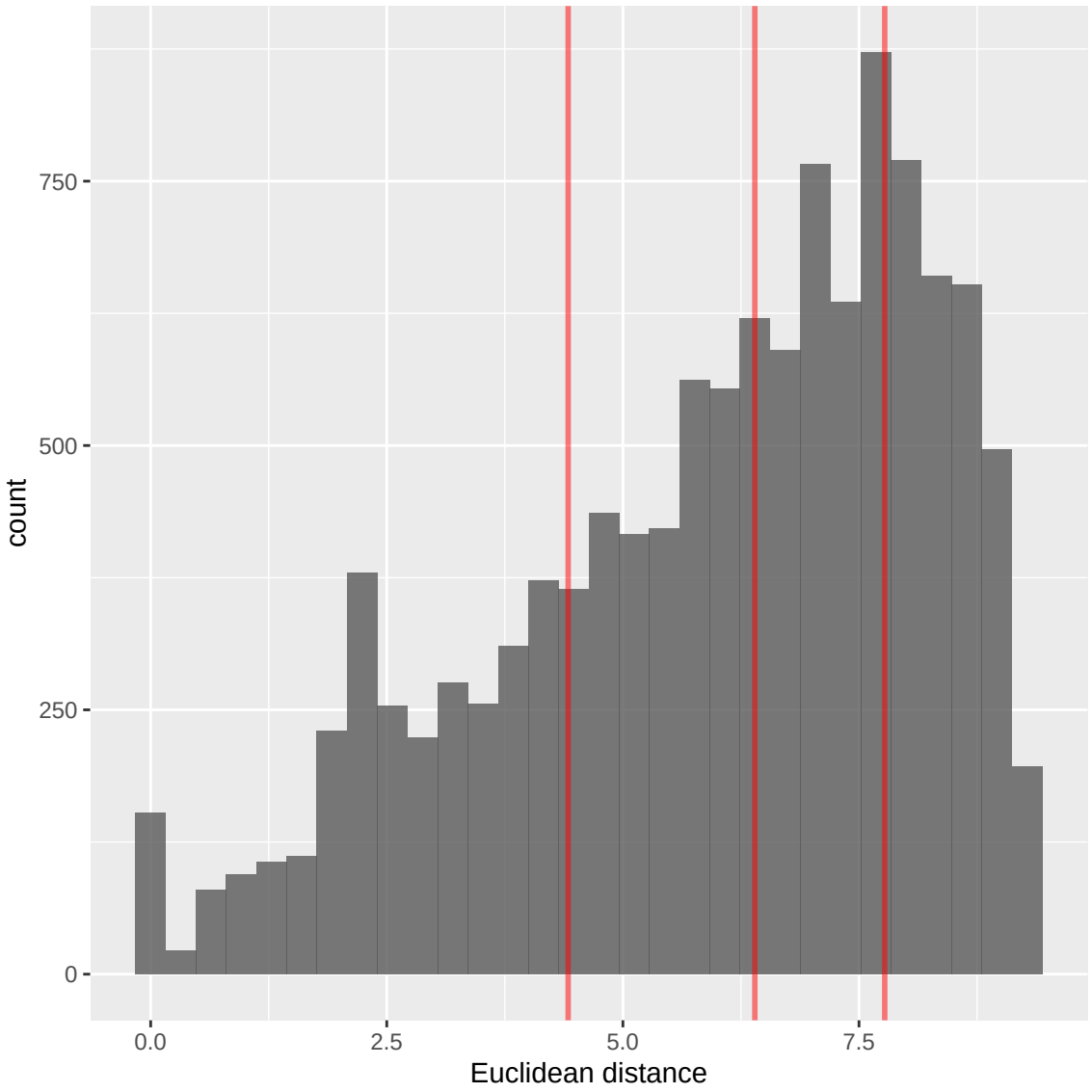
Generate the distance matrix of all pairs of standardised exchange rates.

```
# Distance matrix of standardised exchange rates
d <- as.matrix(dist(ru_sub_t, diag = TRUE, upper = TRUE))

# Give distance matrix col and row names
colnames(d) <- as.factor(colnames(ru_sub))
rownames(d) <- as.factor(colnames(ru_sub))

# Quartiles of the Euclidean distances in the distance matrix
quantile(d, probs = c(0, 0.25, 0.5, 0.75, 1))
```

```
##        0%       25%       50%       75%      100%
## 0.000000 4.420279 6.397129 7.772152 9.439114
```

It's worth noting that half of the Euclidean distances are above or below 6.39, with the top 25 percent of Euclidean distances greater than 7.8 and the bottom 25 percent under 4.35. This indicates that closely connected exchange rates have a Euclidean distance approximately less than 4.

```
d %>%
  as_tibble() %>%
  gather(key  = "Currency", value = "Euclidean distance") %>%
  ggplot(aes(x = `Euclidean distance`)) +
  geom_histogram(binwidth = 0.32, alpha = 0.8) +
  geom_vline(xintercept = quantile(d, probs = c(0.25, 0.5, 0.75)), color = "red", alpha = 0.5, size = 1
```



Make the network

Generate an edge list of exchange rates connected by similarity in movements, i.e., with a Euclidean distance less than 3.

```
# Create another data frame
d_zero <- d

# Filter for only standardised exchange rates w. Euclidean dist. less than 3
d_zero_tbl <- d_zero %>%
  as_tibble() %>%
  mutate(curr1 = rownames(d_zero)) %>% # create variable with each currency
  gather(curr2, dst, -curr1) %>% # edge list connecting each currency + attribute of euclidean dist.
  filter(dst < 3) %>% # filter the edge list to keep only closely connected exch. rates
  filter(curr1 != curr2) # remove exch. rates connected to themselves
```

Using the Kamada–Kawai layout algorithm to build a network diagram of closely related exchange rates.

It's worth noting that there are three clusters.

```
# Make network diagram
library(geomnet)
set.seed(10052016) # set a seed for reproducibility
ggplot(data = d_zero_tbl, aes(from_id = curr1, to_id = curr2)) +
  geom_net(layout.alg = "kamadakawai",
    size = 1, labelon = TRUE, vjust = -0.6, ecolour = "grey60",
    directed =FALSE, fontsize = 2, ealpha = 0.5) +
    theme_net() +
    theme(legend.position = "bottom")
```