SYR-MBA FIN 654 Financial Analytics Practice Set #3

Khan and Khalifa

February 05, 2017

Practice Sets for Time series, Auto and Cross correlations, and Volatility clustering

Various `R` features and finance topics are in these practice sets. Specifically there is focus on reading in data, exploring time series, estimating auto and cross correlations, and investigating volatility clustering in financial time series. Summary of experiences are in the debrief.

Set A

Build and explore a data set using filters and `if` and `diff` statements. Answer some questions using plots and a pivot table report. Review a function to house the approach to run some of the same analysis on other data sets.

Problem

Marketing and accounts receivables managers at our company continue to note we have a significant exposure to exchange rates. Our customer base is located in the United Kingdom, across the European Union, and in Japan. The exposure hits the gross revenue line of our financials. Cash flow is further affected by the ebb and flow of accounts receivable components of working capital for producing several products. When exchange rates are volatile, so is earnings, and more importantly, our cash flow. Our company has also missed earnings forecasts for five straight quarters. To get a handle on exchange rate exposures we download this data set and review some basic aspects of the exchange rates.

Read in data

```
# Read and review a csv file from FRED
exrates1 <- read.csv("data/exrates.csv", header = TRUE)
exrates1 <- na.omit(exrates1)  ## to clean up any missing data
exrates <- exrates1[order(as.Date(exrates1$DATE, format = "%m/%d/%Y")),
    ]
head(exrates, n = 5)
```

```
##        DATE  USD.EUR  USD.GBP USD.CNY USD.JPY
## 1 1/29/2012 0.763678 0.638932 6.29509 77.1840
## 2  2/5/2012 0.760684 0.633509 6.29429 76.3930
## 3 2/12/2012 0.757491 0.632759 6.29232 77.2049
## 4 2/19/2012 0.760889 0.634166 6.29644 78.7109
## 5 2/26/2012 0.750301 0.632641 6.29710 80.3373
```

```
tail(exrates, n = 5)
```

```
##           DATE  USD.EUR  USD.GBP USD.CNY USD.JPY
## 256 12/18/2016 0.950478 0.796572 6.93042 116.796
## 257 12/25/2016 0.958288 0.810481 6.94908 117.469
## 258   1/1/2017 0.954067 0.813594 6.94929 117.100
```

```
## 259   1/8/2017 0.951493 0.812388 6.92820 116.968
## 260  1/15/2017 0.943352 0.820854 6.91781 115.287
```

```r
str(exrates, strict.width = "wrap", width = 80)
```

```
## 'data.frame':    260 obs. of  5 variables:
## $ DATE : Factor w/ 260 levels "1/1/2017","1/10/2016",..: 15 103 89 94 100 126
##    109 114 120 130 ...
## $ USD.EUR: num 0.764 0.761 0.757 0.761 0.75 ...
## $ USD.GBP: num 0.639 0.634 0.633 0.634 0.633 ...
## $ USD.CNY: num 6.3 6.29 6.29 6.3 6.3 ...
## $ USD.JPY: num 77.2 76.4 77.2 78.7 80.3 ...
```

Questions

Q1. What is the nature of exchange rates? Reflect on the ups and downs of rate movements, known to managers as currency appreciation and depreciation. First, calculate percentage changes as log returns. Then use `if` and `else` statements to define a new column called `direction`. Then build a data frame to house this analysis.

```r
# Compute log differences percent using as.matrix to force
# numeric type
exrates.r <- diff(log(as.matrix(exrates[, -1]))) * 100
head(exrates.r, n = 3)
```

```
##      USD.EUR    USD.GBP     USD.CNY   USD.JPY
## 2 -0.3928206 -0.8523826 -0.01270912 -1.030111
## 3 -0.4206372 -0.1184583 -0.03130311  1.057186
## 4  0.4475830  0.2221127  0.06545522  1.931872
```

```r
tail(exrates.r, n = 3)
```

```
##       USD.EUR    USD.GBP      USD.CNY    USD.JPY
## 258 -0.441446  0.3833571  0.003021937 -0.3146198
## 259 -0.270157 -0.1483412 -0.303945688 -0.1127877
## 260 -0.859284  1.0367203 -0.150079365 -1.4475722
```

```r
str(exrates.r)
```

```
##  num [1:259, 1:4] -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:259] "2" "3" "4" "5" ...
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
```

```r
# Create size and direction - each is an indicator of volatility
size <- na.omit(abs(exrates.r))
colnames(size) <- paste(colnames(size), ".size", sep = "")
direction <- ifelse(exrates.r > 0, 1, ifelse(exrates.r < 0, -1, 0))
colnames(direction) <- paste(colnames(direction), ".dir", sep = "")
# Convert into a time series object:
#' 1. Split into date and rates
dates <- as.Date(exrates$DATE[-1], "%m/%d/%Y")
values <- cbind(exrates.r, size, direction)
#' 2. Construct a dataframe with dates, rates and direction
exrates.df <- data.frame(dates = dates, returns = exrates.r, direction = direction)
str(exrates.df)  # notice the returns.* and direction.* prefixes
```

```
## 'data.frame':    259 obs. of  9 variables:
##  $ dates                : Date, format: "2012-02-05" "2012-02-12" ...
##  $ returns.USD.EUR      : num  -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  $ returns.USD.GBP      : num  -0.852 -0.118 0.222 -0.241 -0.455 ...
##  $ returns.USD.CNY      : num  -0.0127 -0.0313 0.0655 0.0105 0.0259 ...
##  $ returns.USD.JPY      : num  -1.03 1.06 1.93 2.05 1.02 ...
##  $ direction.USD.EUR.dir: num  -1 -1 1 -1 1 1 1 -1 -1 1 ...
##  $ direction.USD.GBP.dir: num  -1 -1 1 -1 -1 1 1 -1 -1 1 ...
##  $ direction.USD.CNY.dir: num  -1 -1 1 1 1 1 1 -1 -1 -1 ...
##  $ direction.USD.JPY.dir: num  -1 1 1 1 1 1 1 -1 -1 -1 ...
```

```r
#' 3. Make an xts object with row names equal to the dates
require(xts)
exrates.xts <- na.omit(as.xts(values, dates))  #order.by=as.Date(dates, '%d/%m/%Y')))
head(exrates.xts, n = 3)
```

```
##                 USD.EUR    USD.GBP     USD.CNY   USD.JPY USD.EUR.size
## 2012-02-05 -0.3928206 -0.8523826 -0.01270912 -1.030111    0.3928206
## 2012-02-12 -0.4206372 -0.1184583 -0.03130311  1.057186    0.4206372
## 2012-02-19  0.4475830  0.2221127  0.06545522  1.931872    0.4475830
##            USD.GBP.size USD.CNY.size USD.JPY.size USD.EUR.dir USD.GBP.dir
## 2012-02-05    0.8523826   0.01270912     1.030111          -1          -1
## 2012-02-12    0.1184583   0.03130311     1.057186          -1          -1
## 2012-02-19    0.2221127   0.06545522     1.931872           1           1
##            USD.CNY.dir USD.JPY.dir
## 2012-02-05          -1          -1
## 2012-02-12          -1           1
## 2012-02-19           1           1
```

```r
str(exrates.xts)
```

```
## An 'xts' object on 2012-02-05/2017-01-15 containing:
##   Data: num [1:259, 1:12] -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:12] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
#' 4. Make a zoo object (essentially, a 'zoo' series with a 'frequency' attribute)
require(zoo)
exrates.zr <- as.zooreg(exrates.xts)
head(exrates.zr, n = 3)
```

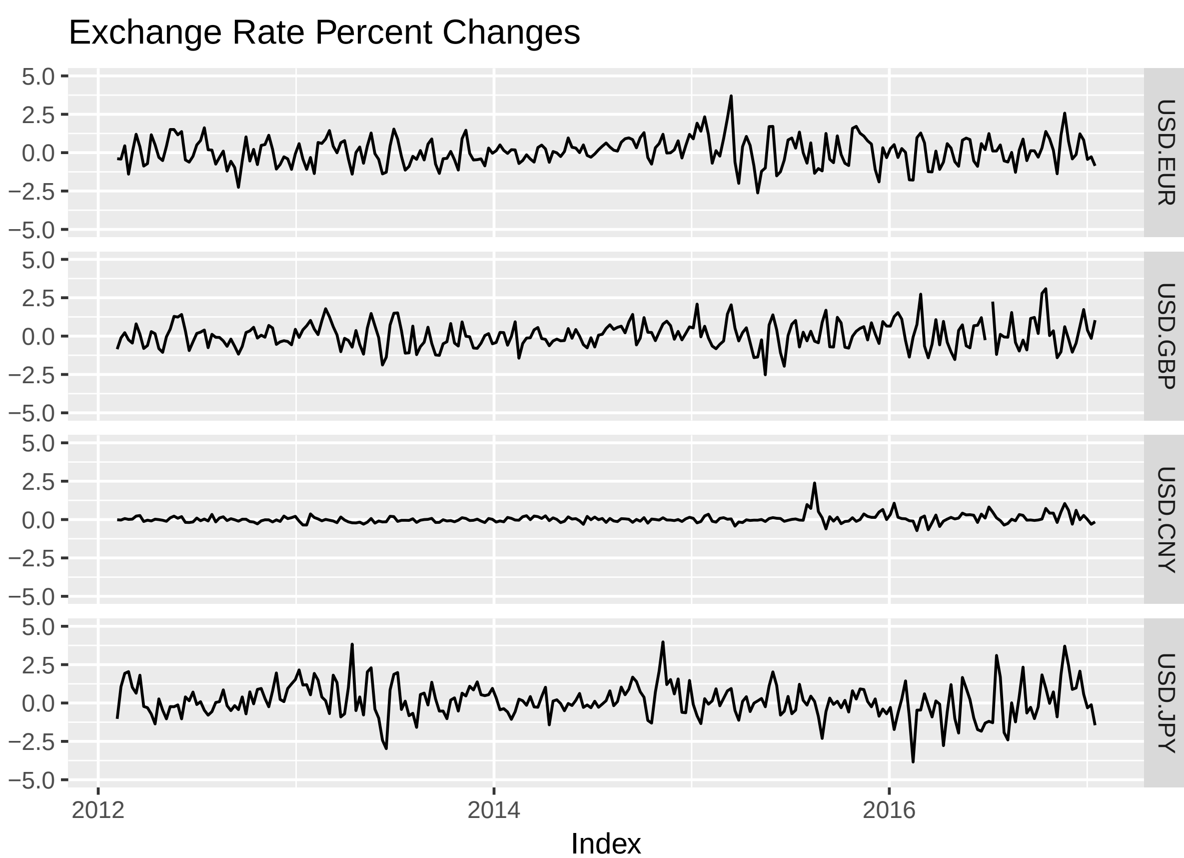
```
##                 USD.EUR    USD.GBP     USD.CNY   USD.JPY USD.EUR.size
## 2012-02-05 -0.3928206 -0.8523826 -0.01270912 -1.030111    0.3928206
## 2012-02-12 -0.4206372 -0.1184583 -0.03130311  1.057186    0.4206372
## 2012-02-19  0.4475830  0.2221127  0.06545522  1.931872    0.4475830
##            USD.GBP.size USD.CNY.size USD.JPY.size USD.EUR.dir USD.GBP.dir
## 2012-02-05    0.8523826   0.01270912     1.030111          -1          -1
## 2012-02-12    0.1184583   0.03130311     1.057186          -1          -1
## 2012-02-19    0.2221127   0.06545522     1.931872           1           1
##            USD.CNY.dir USD.JPY.dir
## 2012-02-05          -1          -1
## 2012-02-12          -1           1
```

```
## 2012-02-19           1           1
```

```r
str(exrates.zr)
```

```
## 'zooreg' series from 2012-02-05 to 2017-01-15
##   Data: num [1:259, 1:12] -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:12] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY" ...
##   Index:  Date[1:259], format: "2012-02-05" "2012-02-12" "2012-02-19" "2012-02-26" ...
##   Frequency: 0.142857142857143
```

```r
#' 5. Plot with the 'zoo' and `ggplot2` packages.
require(zoo)
require(ggplot2)
#' Use autoplot.zoo() function for easy plotting of data.
title.chg <- "Exchange Rate Percent Changes"
autoplot.zoo(exrates.xts[, 1:4]) + ggtitle(title.chg) + ylim(-5,
    5)
```



Q2. Dig deeper. Show autocovariance or autocorrelations. Compute mean, standard deviation, etc. Load the `data_moments()` function. Run the function using the `exrates.xts` data and write a `knitr::kable()` report.

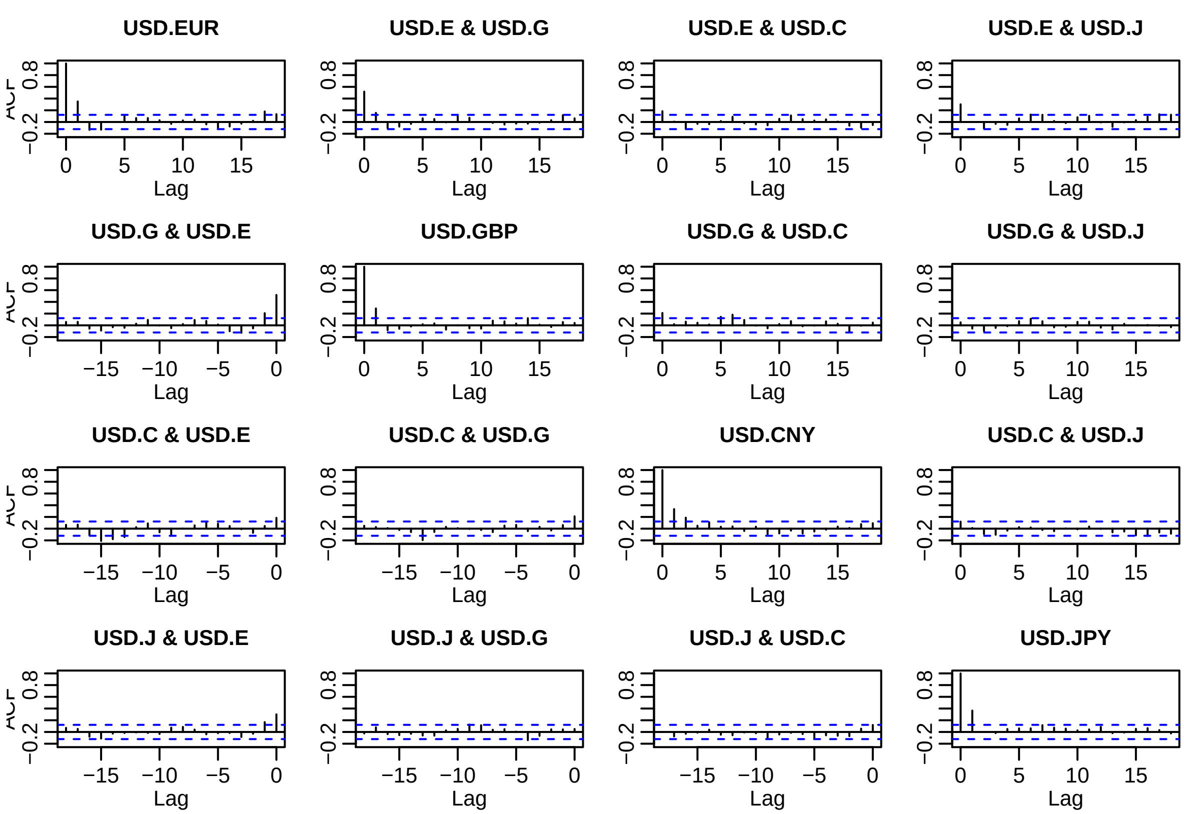
```
# Extract coredata
exrates.cd <- coredata(exrates.xts[, 1:4])
head(exrates.cd, n = 3)
```

```
##         USD.EUR    USD.GBP     USD.CNY   USD.JPY
## [1,] -0.3928206 -0.8523826 -0.01270912 -1.030111
## [2,] -0.4206372 -0.1184583 -0.03130311  1.057186
## [3,]  0.4475830  0.2221127  0.06545522  1.931872
```

```
str(exrates.cd)
```

```
##  num [1:259, 1:4] -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
```

```
# Compute estimates of the autocorrelation for the rates
acf(exrates.cd)
```




```
# Extract coredata
exrates.cd <- coredata(exrates.xts[, 5:8])
head(exrates.cd, n = 3)
```

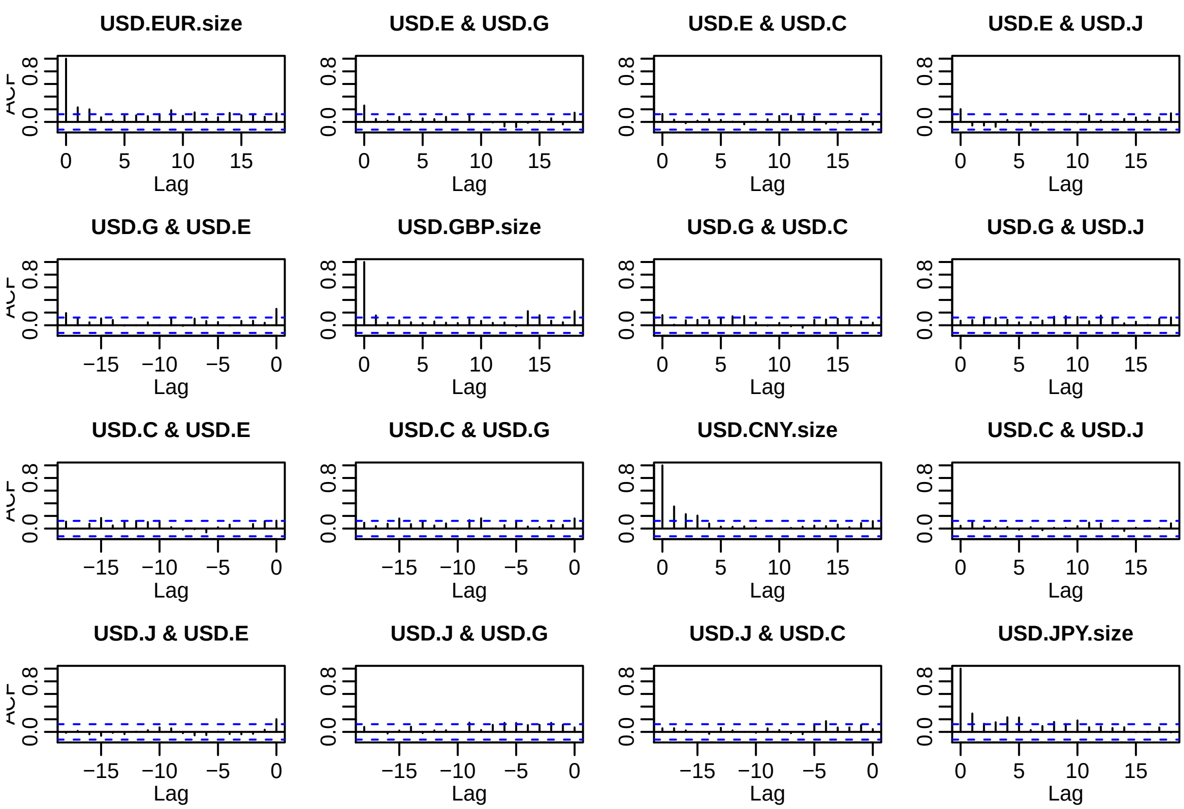
```
##      USD.EUR.size USD.GBP.size USD.CNY.size USD.JPY.size
## [1,]    0.3928206    0.8523826   0.01270912     1.030111
```

```
## [2,]     0.4206372     0.1184583   0.03130311     1.057186
## [3,]     0.4475830     0.2221127   0.06545522     1.931872
```

```
str(exrates.cd)
```

```
##  num [1:259, 1:4] 0.3928 0.4206 0.4476 1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "USD.EUR.size" "USD.GBP.size" "USD.CNY.size" "USD.JPY.size"
```

```
# Compute estimates of the autocorrelation for the magnitude of
# the rates
acf(exrates.cd)
```




```
# Define the data_moments() function
#' data_moments function
#' INPUTS: r vector
#' OUTPUTS: list of scalars (mean, sd, median, skewness, kurtosis)
data_moments <- function(data) {
    require(moments)
    mean.r <- mean(data)
    sd.r <- sd(data)
    median.r <- median(data)
    skewness.r <- skewness(data)
    kurtosis.r <- kurtosis(data)
    result <- data.frame(mean = mean.r, std_dev = sd.r, median = median.r,
        skewness = skewness.r, kurtosis = kurtosis.r)
    return(result)
```

```
}
# Run data_moments()
#' --- original code ---
answer <- data_moments(exrates.xts)
answer <- round(answer, 4)
#' --- corrected code ---
answer <- data_moments(exrates.xts[, 1])
answer <- round(answer, 4)
for (i in 2:12) {
    answer.k <- data_moments(exrates.xts[, i])
    answer <- rbind(answer, round(answer.k, 4))
}
# Build pretty table
knitr::kable(answer)
```

	mean	std_dev	median	skewness	kurtosis
USD.EUR	0.0816	0.9022	0.1043	0.1690	3.5901
USD.GBP	0.0967	0.9473	-0.0130	1.6627	12.9297
USD.CNY	0.0364	0.2785	-0.0005	2.9623	23.2779
USD.JPY	0.1549	1.1011	0.1131	0.1906	4.3916
USD.EUR.size	0.7185	0.5499	0.5895	1.3773	6.3808
USD.GBP.size	0.6884	0.6565	0.5601	4.0555	34.3779
USD.CNY.size	0.1700	0.2233	0.1118	4.9157	41.4959
USD.JPY.size	0.8310	0.7371	0.6358	1.6373	6.3185
USD.EUR.dir	0.0811	0.9986	1.0000	-0.1627	1.0265
USD.GBP.dir	-0.0116	1.0019	-1.0000	0.0232	1.0005
USD.CNY.dir	-0.0116	1.0019	-1.0000	0.0232	1.0005
USD.JPY.dir	0.0734	0.9992	1.0000	-0.1471	1.0216

Set B

Use the data from Set A to investigate the interactions of the distribution of exchange rates.

Problem

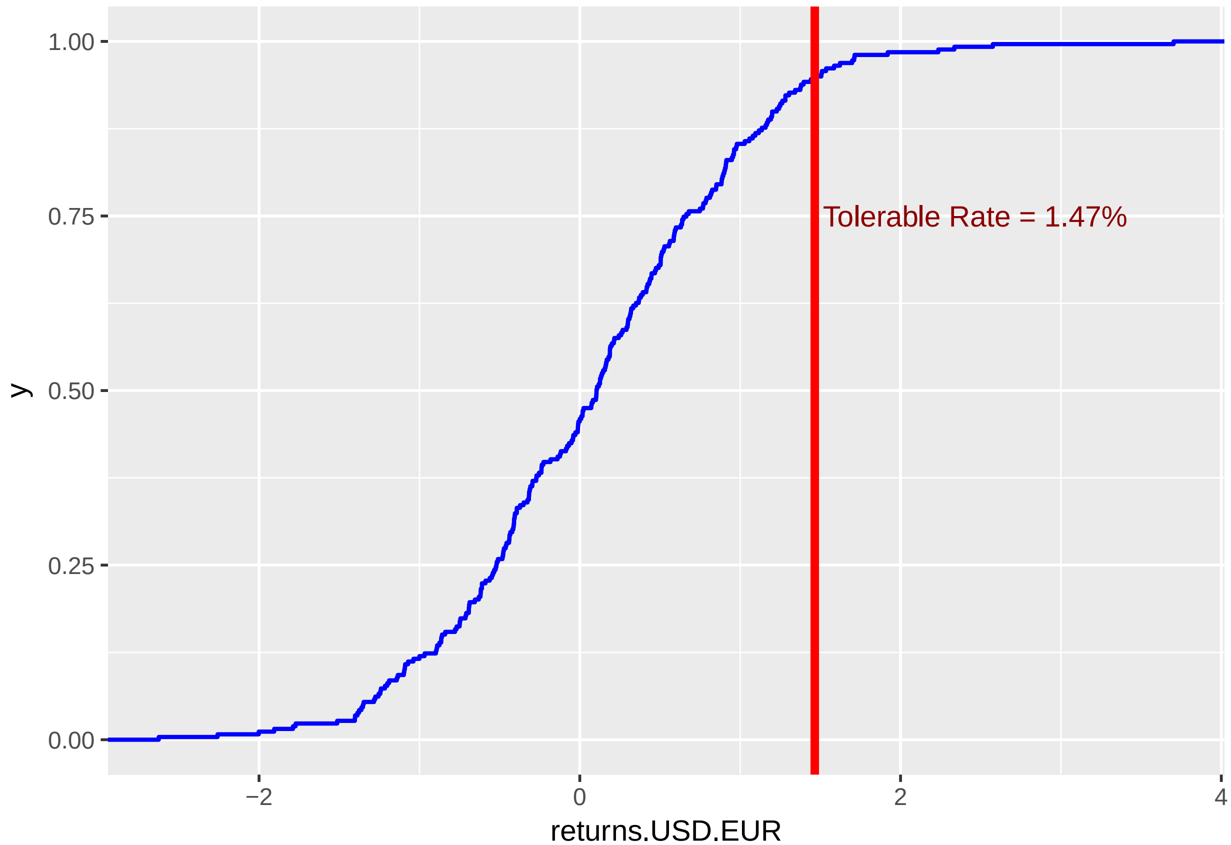
Characterize the distribution of up and down movements visually. Further, set up for a periodic repeat analysis for inclusion in management reports.

Questions

Q1. Show the shape of exposure to euros, especially given the tolerance for risk. Assume corporate policy set tolerance at 95%. Use the `exrates.df` data frame with `ggplot2` and the empirical cumulative relative frequency function `stat_ecdf`.

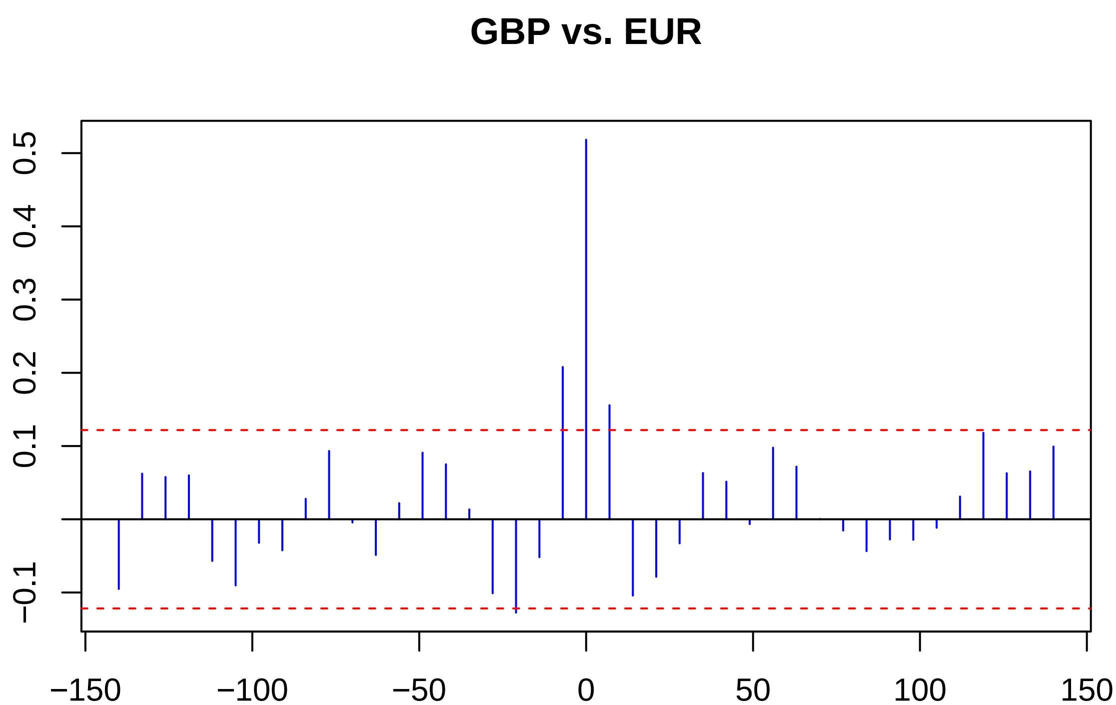
```
exrates.tol.pct <- 0.95
exrates.tol <- quantile(exrates.df$returns.USD.EUR, exrates.tol.pct)
exrates.tol.label <- paste("Tolerable Rate = ", round(exrates.tol,
    2), "%", sep = "")
ggplot(exrates.df, aes(returns.USD.EUR, fill = direction.USD.EUR.dir)) +
    stat_ecdf(colour = "blue", size = 0.75) + geom_vline(xintercept = exrates.tol,
```

```
    colour = "red", size = 1.5) + annotate("text", x = exrates.tol +
    1, y = 0.75, label = exrates.tol.label, colour = "darkred")
```



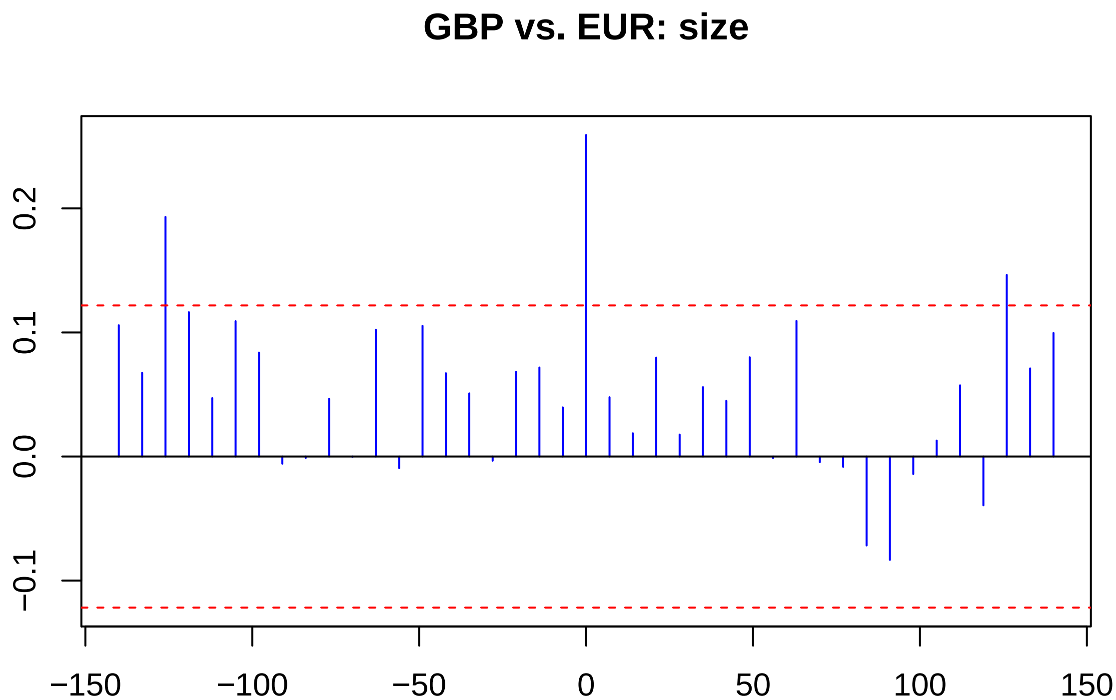
Q2. What is the history of correlations in the exchange rate markets? If this is a “history,” then consider the risk that conducting business in one country will definitely affect business in another. Further, bad things may be followed by more bad things more often than good things.

```
lag1 <- 20
ccf(exrates.zr[, 1], exrates.zr[, 2], main = "GBP vs. EUR", lag.max = lag1,
    ylab = "", xlab = "", col = "blue", ci.col = "red")
```



Applying cross-correlation function on the rates, we find that there is a correlation beween exchange rates for GBP and EUR.

```r
lag1 <- 20
ccf(abs(exrates.zr[, 1]), abs(exrates.zr[, 2]), main = "GBP vs. EUR: size",
    lag.max = lag1, ylab = "", xlab = "", col = "blue", ci.col = "red")
```



Applying cross-correlation function on the absolute sizes, we note the volatility of correlation is also related.

Q3. One more experiment: a rolling correlation. Create a rolling correlation function, `corr.rolling`, and embed this function into the `rollapply()` function.

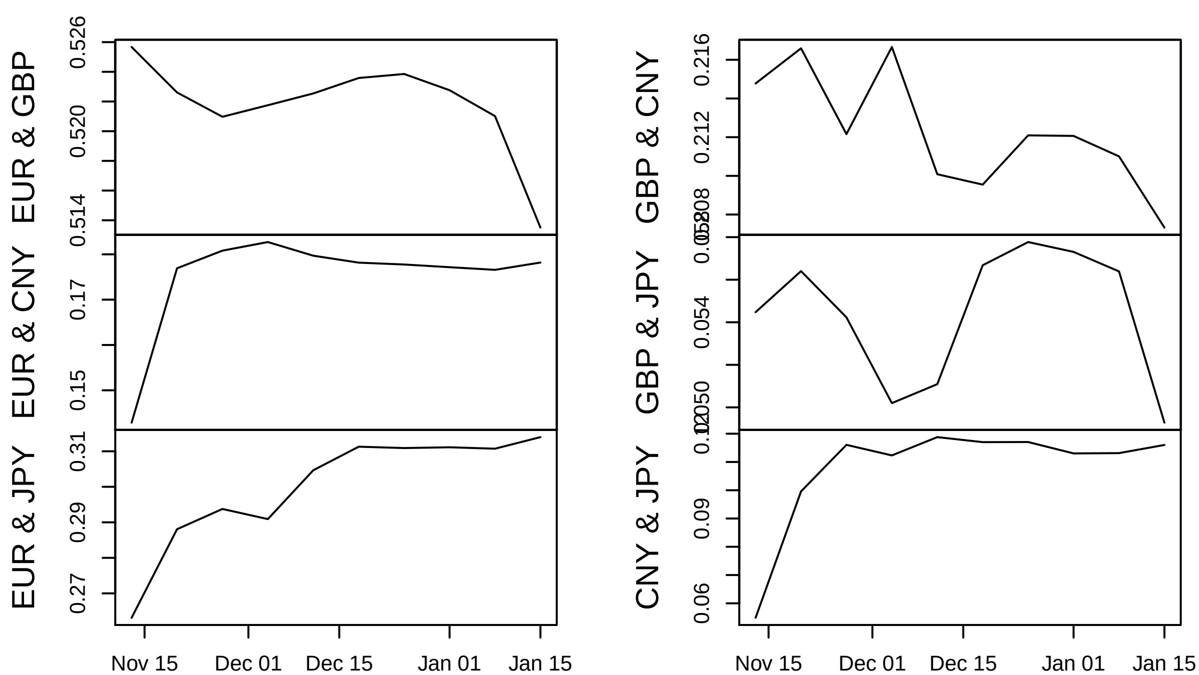
```
corr.rolling <- function(x) {
    dim <- ncol(x)
    corr.r <- cor(x)[lower.tri(diag(dim), diag = FALSE)]
    return(corr.r)
}
ALL.r <- exrates.zr[, 1:4]
corr.returns <- rollapply(ALL.r, width = 250, corr.rolling, align = "right",
    by.column = FALSE)
head(corr.returns, n = 2)
```

```
##
## 2016-11-13 0.5256724 0.1428576 0.2631368 0.2147757 0.05447472 0.05489404
## 2016-11-20 0.5226087 0.1769266 0.2880611 0.2165933 0.05640353 0.09959976
```

```
str(corr.returns)
```

```
## 'zooreg' series from 2016-11-13 to 2017-01-15
##   Data: num [1:10, 1:6] 0.526 0.523 0.521 0.522 0.523 ...
##   Index:  Date[1:10], format: "2016-11-13" "2016-11-20" "2016-11-27" "2016-12-04" ...
##   Frequency: 0.142857142857143
```

```
colnames(corr.returns) <- c("EUR & GBP", "EUR & CNY", "EUR & JPY",
    "GBP & CNY", "GBP & JPY", "CNY & JPY")
plot(corr.returns, xlab = "", main = "")
```



Q4. How related are correlations and volatilities? Put another way, are there concerns that inter-market transactions (e.g., customers and vendors transacting in more than one currency) can affect transactions in a single market? Take the `exrate` data to understand how dependent correlations and volatilities depend upon one another.

```r
require(matrixStats)
head(ALL.r, n = 3)
```

```
##              USD.EUR    USD.GBP     USD.CNY   USD.JPY
## 2012-02-05 -0.3928206 -0.8523826 -0.01270912 -1.030111
## 2012-02-12 -0.4206372 -0.1184583 -0.03130311  1.057186
## 2012-02-19  0.4475830  0.2221127  0.06545522  1.931872
```

```r
str(ALL.r)
```

```
## 'zooreg' series from 2012-02-05 to 2017-01-15
##   Data: num [1:259, 1:4] -0.3928 -0.4206 0.4476 -1.4013 0.0231 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
##   Index:  Date[1:259], format: "2012-02-05" "2012-02-12" "2012-02-19" "2012-02-26" ...
##   Frequency: 0.142857142857143
```

```r
R.corr <- apply.monthly(as.xts(ALL.r), FUN = cor)
head(R.corr, n = 3)
```

```
##            [,1]      [,2]       [,3]        [,4]      [,5] [,6]       [,7]
## 2012-02-26    1 0.3779270  0.4926042 -0.08431313 0.3779270    1  0.6104888
## 2012-03-25    1 0.9718639  0.8787135  0.49720832 0.9718639    1  0.8796893
## 2012-04-29    1 0.9059433 -0.2923581 -0.57795431 0.9059433    1 -0.1252051
##                  [,8]       [,9]      [,10] [,11]      [,12]       [,13]
## 2012-02-26  0.8769294  0.4926042  0.6104888     1  0.5351171 -0.08431313
## 2012-03-25  0.4101965  0.8787135  0.8796893     1  0.7796609  0.49720832
## 2012-04-29 -0.7075547 -0.2923581 -0.1252051     1 -0.1424010 -0.57795431
##                 [,14]      [,15] [,16]
## 2012-02-26  0.8769294  0.5351171     1
## 2012-03-25  0.4101965  0.7796609     1
## 2012-04-29 -0.7075547 -0.1424010     1
```

```r
str(R.corr)
```

```
## An 'xts' object on 2012-02-26/2017-01-15 containing:
##   Data: num [1:60, 1:16] 1 1 1 1 1 1 1 1 1 1 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
R.vols <- apply.monthly(ALL.r, FUN = colSds)  # from MatrixStats\t
head(R.vols, n = 3)
```

```
##              USD.EUR   USD.GBP    USD.CNY   USD.JPY
## 2012-02-26 0.7559750 0.4483734 0.04195577 1.4242570
## 2012-03-25 0.8595039 0.7034406 0.18111483 0.8461026
## 2012-04-29 0.7891844 0.5999862 0.04331953 0.5956700
```

```r
str(R.vols)
```

```
## 'zoo' series from 2012-02-26 to 2017-01-15
##   Data: num [1:60, 1:4] 0.756 0.86 0.789 0.514 0.923 ...
```

```
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
##   Index:  Date[1:60], format: "2012-02-26" "2012-03-25" "2012-04-29" "2012-05-27" ...
```

```r
#'
#' Form correlation matrix for one month
#'
R.corr.1 <- matrix(R.corr[1, ], nrow = 4, ncol = 4, byrow = FALSE)
rownames(R.corr.1) <- colnames(ALL.r[, 1:4])
colnames(R.corr.1) <- rownames(R.corr.1)
head(R.corr.1)
```

```
##             USD.EUR   USD.GBP   USD.CNY     USD.JPY
## USD.EUR  1.00000000 0.3779270 0.4926042 -0.08431313
## USD.GBP  0.37792703 1.0000000 0.6104888  0.87692936
## USD.CNY  0.49260422 0.6104888 1.0000000  0.53511710
## USD.JPY -0.08431313 0.8769294 0.5351171  1.00000000
```

```r
str(R.corr.1)
```

```
##  num [1:4, 1:4] 1 0.3779 0.4926 -0.0843 0.3779 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
```

```r
#
R.corr <- R.corr[, c(2, 3, 4, 7, 8, 12)]
head(R.corr, n = 3)
```

```
##                 [,1]       [,2]        [,3]       [,4]       [,5]
## 2012-02-26 0.3779270  0.4926042 -0.08431313  0.6104888  0.8769294
## 2012-03-25 0.9718639  0.8787135  0.49720832  0.8796893  0.4101965
## 2012-04-29 0.9059433 -0.2923581 -0.57795431 -0.1252051 -0.7075547
##                  [,6]
## 2012-02-26  0.5351171
## 2012-03-25  0.7796609
## 2012-04-29 -0.1424010
```

```r
colnames(R.corr) <- colnames(corr.returns)
colnames(R.vols) <- c("EUR.vols", "GBP.vols", "CNY.vols", "JPY.vols")
head(R.corr, n = 3)
```

```
##            EUR & GBP  EUR & CNY   EUR & JPY  GBP & CNY  GBP & JPY
## 2012-02-26 0.3779270  0.4926042 -0.08431313  0.6104888  0.8769294
## 2012-03-25 0.9718639  0.8787135  0.49720832  0.8796893  0.4101965
## 2012-04-29 0.9059433 -0.2923581 -0.57795431 -0.1252051 -0.7075547
##             CNY & JPY
## 2012-02-26  0.5351171
## 2012-03-25  0.7796609
## 2012-04-29 -0.1424010
```

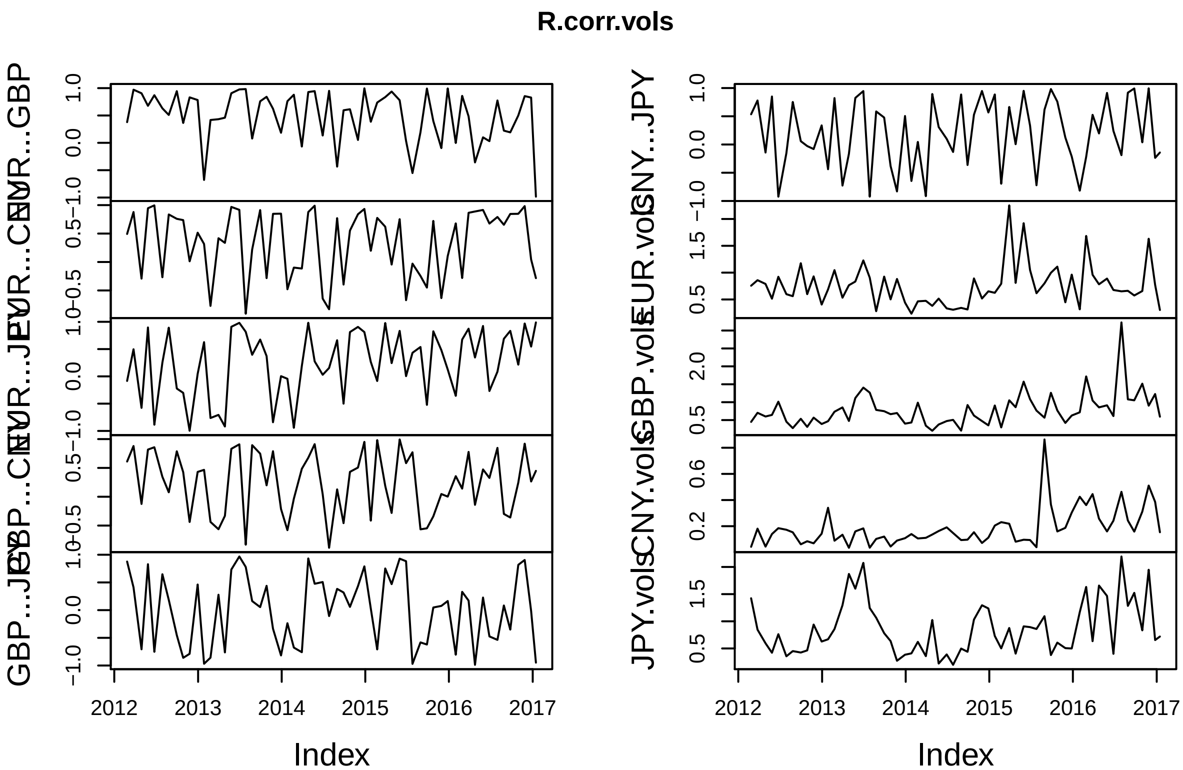
```r
head(R.vols, n = 3)
```

```
##             EUR.vols  GBP.vols   CNY.vols  JPY.vols
## 2012-02-26 0.7559750 0.4483734 0.04195577 1.4242570
## 2012-03-25 0.8595039 0.7034406 0.18111483 0.8461026
## 2012-04-29 0.7891844 0.5999862 0.04331953 0.5956700
```

```r
R.corr.vols <- merge(R.corr, R.vols)
head(R.corr.vols, n = 3)
```

```
##              EUR...GBP  EUR...CNY   EUR...JPY  GBP...CNY  GBP...JPY
## 2012-02-26 0.3779270  0.4926042 -0.08431313  0.6104888  0.8769294
## 2012-03-25 0.9718639  0.8787135  0.49720832  0.8796893  0.4101965
## 2012-04-29 0.9059433 -0.2923581 -0.57795431 -0.1252051 -0.7075547
##              CNY...JPY  EUR.vols  GBP.vols   CNY.vols  JPY.vols
## 2012-02-26  0.5351171 0.7559750 0.4483734 0.04195577 1.4242570
## 2012-03-25  0.7796609 0.8595039 0.7034406 0.18111483 0.8461026
## 2012-04-29 -0.1424010 0.7891844 0.5999862 0.04331953 0.5956700
```

```r
plot.zoo(R.corr.vols)
```




```r
EUR.vols <- as.numeric(R.corr.vols[, "EUR.vols"])
GBP.vols <- as.numeric(R.vols[, "GBP.vols"])
CNY.vols <- as.numeric(R.vols[, "CNY.vols"])
length(EUR.vols)
```

```
## [1] 60
```

```r
#'
#' Smooth data volatility
#'
fisher <- function(r) {
    0.5 * log((1 + r)/(1 - r))
}
```

```r
rho.fisher <- matrix(fisher(as.numeric(R.corr.vols[, 1:6])), nrow = length(EUR.vols),
    ncol = 6, byrow = FALSE)
#'
#' ***
#' Here is the quantile regression part of the package.
#'
#' ## Notice
#' 1. We set `taus` as the quantiles of interest.
#' 2. We run the quantile regression using the `quantreg` package and a call to rq().
#' 3. We can overlay the quantile regression results onto the linear model regression.
#' 4. We can sensitize our analysis with the range of upper and lower bounds on the
#'    parameter estimates of the relationship between correlation and volatility.
#'
require(quantreg)
taus <- seq(0.05, 0.95, 0.05)
fit.rq.EUR.GBP <- rq(rho.fisher[, 1] ~ EUR.vols, tau = taus)
fit.lm.EUR.GBP <- lm(rho.fisher[, 1] ~ EUR.vols)
#'
summary(fit.rq.EUR.GBP, se = "boot")
```

```
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.05
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -1.30970  0.61560   -2.12751  0.03764
## EUR.vols     1.11618  0.46153    2.41844  0.01875
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.1
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.76669  0.49992   -1.53362  0.13056
## EUR.vols     0.87530  0.38011    2.30278  0.02490
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.15
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.41944  0.28171   -1.48893  0.14192
## EUR.vols     0.72127  0.25723    2.80394  0.00686
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.2
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
```

```
## (Intercept) -0.34183  0.19676   -1.73732  0.08764
## EUR.vols     0.68684  0.18842    3.64519  0.00057
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.20006  0.16916   -1.18264  0.24178
## EUR.vols     0.62395  0.15646    3.98803  0.00019
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.3
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.16748  0.15908   -1.05278  0.29681
## EUR.vols     0.62911  0.18171    3.46218  0.00101
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.35
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.05569  0.16638   -0.33472  0.73905
## EUR.vols     0.57094  0.19583    2.91546  0.00504
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.4
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.08166 0.19869    0.41101 0.68258
## EUR.vols    0.49946 0.23386    2.13575 0.03693
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.45
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.11248  0.25197   -0.44641  0.65696
## EUR.vols     0.83120  0.28836    2.88255  0.00552
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
```

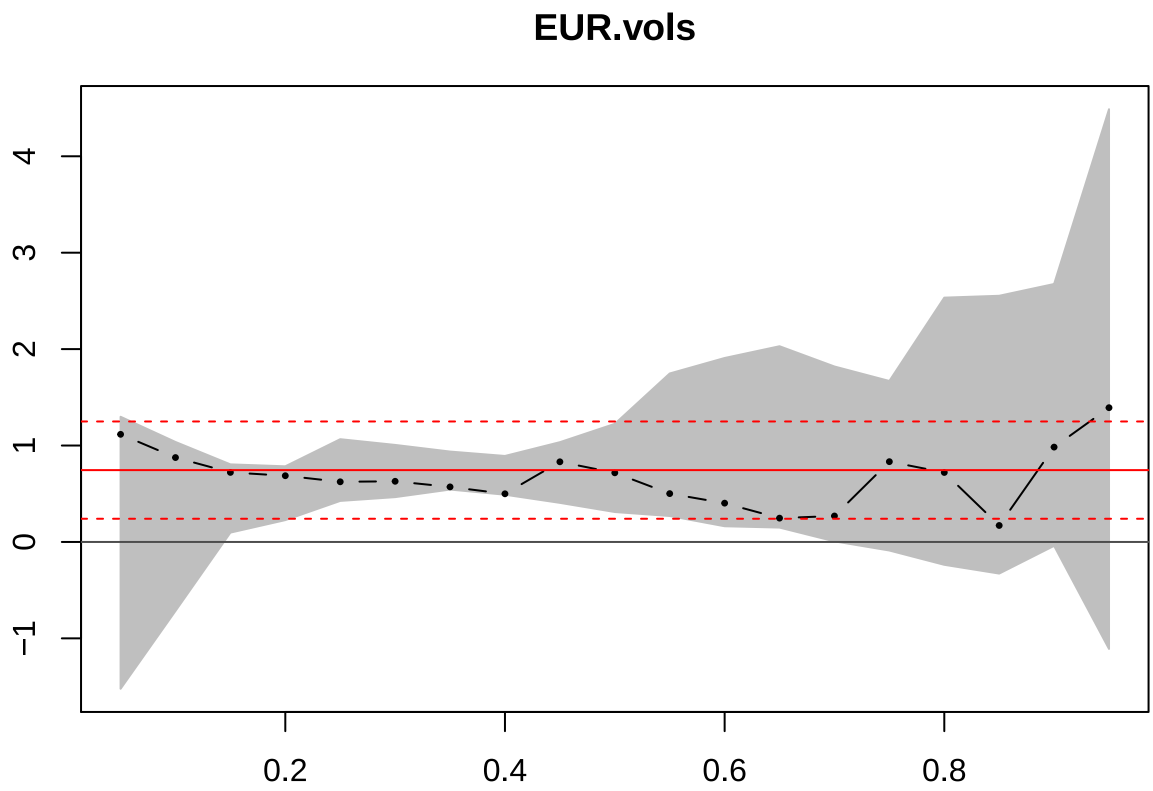
```
## (Intercept) 0.07989 0.31742    0.25169 0.80217
## EUR.vols    0.71696 0.35210    2.03626 0.04630
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.55
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.44312 0.38722    1.14435 0.25718
## EUR.vols    0.50126 0.43685    1.14745 0.25591
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.6
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.61565 0.43733    1.40775 0.16454
## EUR.vols    0.40224 0.51911    0.77487 0.44156
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.65
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.87137 0.45774    1.90364 0.06192
## EUR.vols    0.24695 0.52675    0.46882 0.64096
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.7
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.94211 0.41710    2.25870 0.02768
## EUR.vols    0.26943 0.51783    0.52030 0.60484
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.80393 0.44950    1.78851 0.07892
## EUR.vols    0.83269 0.60563    1.37493 0.17444
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.8
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
```

```
## (Intercept) 0.93575 0.51007    1.83454 0.07170
## EUR.vols    0.72058 0.65346    1.10271 0.27471
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.85
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 1.58199 0.49135    3.21969 0.00210
## EUR.vols    0.17092 0.69278    0.24672 0.80600
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.9
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 1.27929 0.54908    2.32986 0.02332
## EUR.vols    0.98377 0.77498    1.26941 0.20937
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.95
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 1.41097 0.95816    1.47258 0.14627
## EUR.vols    1.39234 1.02729    1.35535 0.18056
```

```
summary(fit.lm.EUR.GBP, se = "boot")
```

```
##
## Call:
## lm(formula = rho.fisher[, 1] ~ EUR.vols)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8826 -0.5731 -0.1813  0.5917  2.4763
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2396     0.2613   0.917   0.3630
## EUR.vols      0.7450     0.3067   2.429   0.0183 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9209 on 58 degrees of freedom
## Multiple R-squared:  0.09233,    Adjusted R-squared:  0.07668
## F-statistic:   5.9 on 1 and 58 DF,  p-value: 0.01826
```

```
#'
plot(summary(fit.rq.EUR.GBP), parm = "EUR.vols")
```




```
#' ***
#' Here we build the estimations and plot the upper and lower bounds.
#'
taus1 <- c(0.05, 0.5, 0.95)  # fit the confidence interval (CI)
EUR.GBP.p <- predict(rq(rho.fisher[, 1] ~ EUR.vols, tau = taus1))
EUR.GBP.lm.p <- predict(lm(rho.fisher[, 1] ~ EUR.vols))
colnames(EUR.GBP.p) <- c(paste("tau", taus1[1] * 100, sep = ""),
    paste("tau", taus1[2] * 100, sep = ""), paste("tau", taus1[3] *
        100, sep = ""))
head(EUR.GBP.p, n = 3)
```

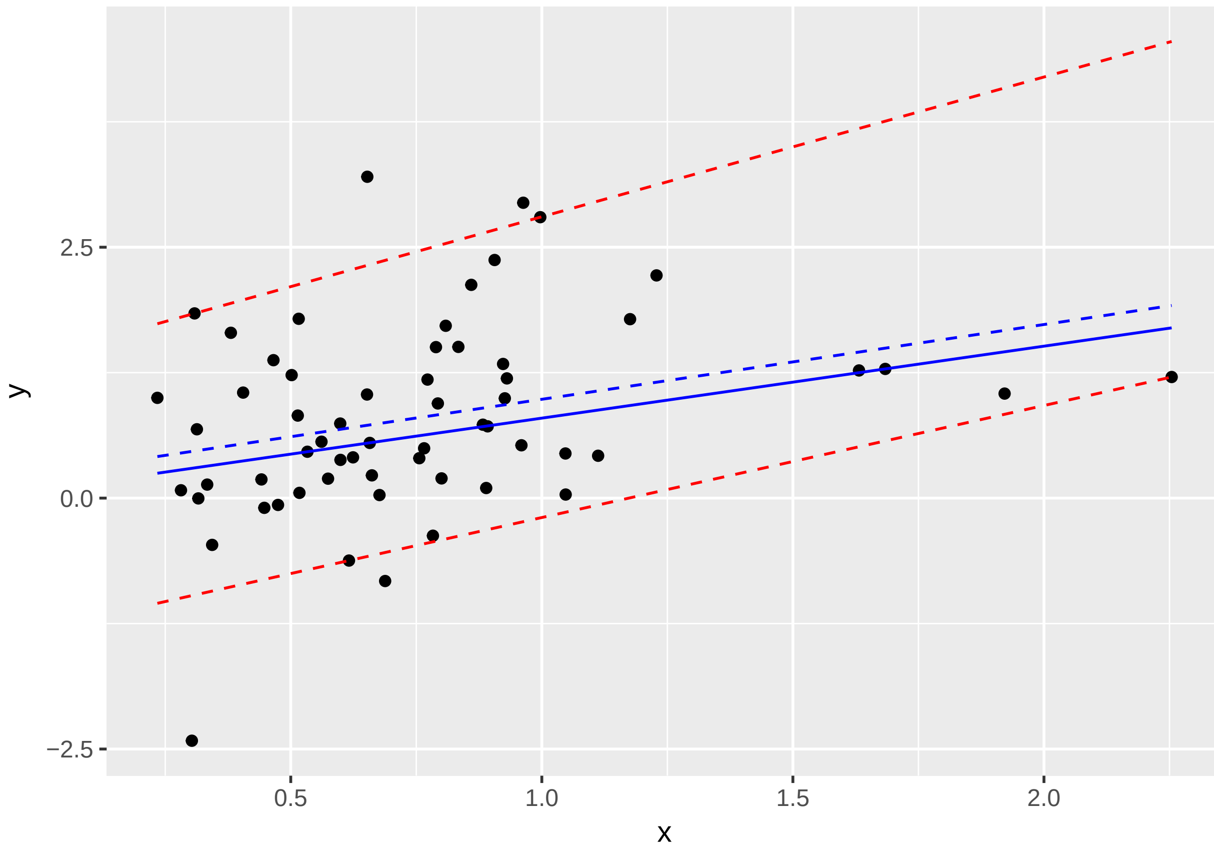
```
##             tau5     tau50    tau95
## [1,] -0.4658975 0.6218969 2.463547
## [2,] -0.3503409 0.6961234 2.607695
## [3,] -0.4288300 0.6457068 2.509786
```

```
EUR.GBP.CI <- data.frame(x = EUR.vols, y = rho.fisher[, 1], y.5 = EUR.GBP.p[,
    1], y.50 = EUR.GBP.p[, 2], y.95 = EUR.GBP.p[, 3], y.lm <- EUR.GBP.lm.p)
head(EUR.GBP.CI, n = 3)
```

```
##           x         y        y.5      y.50     y.95 y.lm....EUR.GBP.lm.p
## 1 0.7559750 0.3976391 -0.4658975 0.6218969 2.463547            0.8027790
## 2 0.8595039 2.1248414 -0.3503409 0.6961234 2.607695            0.8799036
## 3 0.7891844 1.5044171 -0.4288300 0.6457068 2.509786            0.8275186
```

```
ggplot(EUR.GBP.CI, aes(x, y)) + geom_point() + geom_line(aes(y = y.5),
    colour = "red", linetype = "dashed") + geom_line(aes(y = y.95),
    colour = "red", linetype = "dashed") + geom_line(aes(y = y.50),
```

```
    colour = "blue") + geom_line(aes(y = y.lm), colour = "blue",
    linetype = "dashed")
```



Practice Set Debrief

List the R skills needed to complete these practice sets.

Plotting of graphs continues to be a skill that builds upon previous basic graphing skills. Here plot and ggplot are used with plot.zoo and ggplot.zoo to build plots for time series, correlatons, cumulative distribution and summary of the fit. Functions are again used to capture repetitive tasks with different inputs. The ability to delete the first row or column to align data is a handy skill at times, as is the abilty to "bind" rows or columns to form larger matrices. Further, names and labels may be formed using "paste" to concatenate strings and other data. Additionally, merge() provides vlookup style functionality. Lastly, the use of apply(), apply.monthly() and rollapply() are ways to pass data and functions as parameters to generate new tables.

What are the packages used to compute and graph results? Explain each of them.

Set A uses zoo, xts and ggplot2 packages. Package zoo contains methods for totally ordered indexed observations. It is particularly aimed at irregular time series of numeric vectors/matrices and factors. The package provides methods to extend standard generics, keeping consistency with ts and base R and independence of any particular index/date/time class. For example, coredata() is used for extracting the core data contained in a (more complex) object and replacing it. Package xts specifies the extensible time series class and methods, extending and behaving like zoo. The package has methods to coerce data objects of arbitrary classes to class xts and back, without losing any attributes of the original format. Package ggplot2 is a system for creating elegant graphics. You provide the data and the variables that need to be mapped to aesthetics, and what kind of charts to plot (bar, line, pie, etc.) and ggplot2 does the rest. You can output several charts with different styles to be superimposed on the same graph.

In addition, Set B uses moments, matrixStats and quantreg packages. Package moments provides many of the descriptive statistics functions (beyond the summary() function) to understand a data set: mean, median, mode, quartile, standard deviation, skewness and kurtosis. Package matrixStats supplements matrix manipulations with several other functions for rows and columns - in particluar, colSds() provides the standard deviation estimates for each column in a matrix. Further, while the fisher() function is used as a smoothing routine to help stabilize the volitility of a variate, the ccf() from stats package computes the cross-correlation or cross-covariance of two univariate series, and the lm() from stats package is used to fit linear models. Package quantreg has a number of estimation and inference methods for models of conditional quantiles - linear and nonlinear parametric and non-parametric models. Specifically, rq() from quantreg package gives the quantile regression fit, which is used in the generic function predict() to obtain a suitable fitting prediction model.

How well do the results begin to answer the business questions posed at the beginning of each practice set?

The overall objective here is to analyze any significant exposure to exchange rates. Because the customer base is located in the United Kingdom, across the European Union, and in Japan, the analytics is based on available exchange rates with respect to USD. We read in the data, examine it in the raw form, and then proceed to build stylized facts of the market. First, we use the notion of continuous compounding rates to build a table of changes in the exchange rates using ln(), or log() in R. We also compute the size or magnitude (absolute value) of the rates and the direction (up/down/same) of changes between successive points. By constructing a dataframe and extending that to time series (xts) and ordered indexed observations (zoo), we are able to graph the changes in the exchange rates over time. The USD.CNY rate appears relatively stable over time, whereas the other three - USD.EUR, USD.GBP and USD.JPY - show volatility. Digging deeper, we look at the autocorrelation between the different rates and between the different sizes (absolute value of rates), and compute several statistics including skewness and kurtosis. The autocorrelation function shows spikes for relationships in EUR with the other three and for GBP with CNY. That means persistence and correlations for those specific markets. Also, only USD.EUR.dir and USD.JPY.dir skew to the left. Further, USB.GBP and USB.CNY are thick tailed (with respect to the rate changes and the sizes of the rate changes).

Then, we begin to examine the exposure to euros, given a corporate policy and tolerance at 95%. A plot of USD.EUR based on cumulative relative frequency distribution shows a tolerance rate of 1.47% beyond which there could be exposure to the exchange rate for euros. Moving on to examine the cross correlation between USD.GBP and USD.EUR rates, we show that correlation does exist and the volatility of correlation is high with respect to JPY rates. From the plot of corr.returns, it appears that the relationship among EUR, CNY and JPY levels off, but is varying with respect to GBP. In the merge of the tables of correlations and volatilities, we again note high correlations between EUR and GBP, and negative correlations between Europe (GBP & EUR) rates and Asian (CNY & JPY) rates. The volatility for JPY is particularly high in Feb 2012 (for example). Using the fisher function to stabilize the volatility, we then run a quantile regression (5% thru 95% in steps of 5%) on the monthly rolled data. This gives us the upper and lower bounds on correlation and volatility, and a plot of the summary of the fit. Next, we build the estimations and plot the upper and lower bounds along with the linear regression. This reveals a definite pattern with a distance of about 1 between upper and lower bounds. Surprisingly, as volatility rises, the amount of dispersion of correlation grows very little, and the linear regression line approaches the lower bound.