Project #3

Purpose, Process, Product

Various R features and finance topics will be reprised in this chapter. Specifically we will practice reading in data, exploring time series, estimating auto and cross correlations, and investigating volatility clustering in financial time series. We will summarize our experiences in debrief.

Assignment

This assignment will span Live Sessions 3 and 4 (two weeks). The project (3) is due before Live Session 5. Submit into **Coursework > Assignments and Grading > Project 3 > Submission** an `RMD` file with filename **lastname-firstname_Project3.Rmd**. If you have difficulties submitting a `.Rmd` file, then submit a `.txt` file.

1. Use headers (##), r-chunks for code, and text to build a report that addresses the two parts of this project.
2. List in the text the 'R' skills needed to complete this project.
3. Explain each of the functions (e.g., `ggplot()`) used to compute and visualize results.
4. Discuss how well did the results begin to answer the business questions posed at the beginning of each part of the project.

Part 1

In this set we will build and explore a data set using filters and `if` and `diff` statements. We will then answer some questions using plots and a pivot table report. We will then review a function to house our approach in case we would like 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. of 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.

```r
# Read in data
library(zoo)
library(xts)
library(ggplot2)
# Read and review a csv file from
# FRED
exrates <- na.omit(read.csv("data/exrates.csv",
    header = TRUE))
head(exrates)
```

```
##        DATE USD.EUR USD.GBP USD.CNY USD.JPY
## 1 1/28/2013  1.3459  1.5686  6.2240   90.73
```

```
## 2 1/29/2013  1.3484  1.5751  6.2259   90.65
## 3 1/30/2013  1.3564  1.5793  6.2204   91.05
## 4 1/31/2013  1.3584  1.5856  6.2186   91.28
## 5  2/1/2013  1.3692  1.5744  6.2265   92.54
## 6  2/4/2013  1.3527  1.5737  6.2326   92.57
```

```
tail(exrates)
```

```
##           DATE USD.EUR USD.GBP USD.CNY USD.JPY
## 1248 1/19/2018  1.2238  1.3857  6.3990  110.56
## 1249 1/22/2018  1.2230  1.3944  6.4035  111.15
## 1250 1/23/2018  1.2277  1.3968  6.4000  110.46
## 1251 1/24/2018  1.2390  1.4198  6.3650  109.15
## 1252 1/25/2018  1.2488  1.4264  6.3189  108.70
## 1253 1/26/2018  1.2422  1.4179  6.3199  108.38
```

```
str(exrates)
```

```
## 'data.frame':    1253 obs. of  5 variables:
##  $ DATE   : Factor w/ 1253 levels "1/10/2014","1/10/2017",..: 62 66 73 77 409 484 488 492 496 499 ..
##  $ USD.EUR: num  1.35 1.35 1.36 1.36 1.37 ...
##  $ USD.GBP: num  1.57 1.58 1.58 1.59 1.57 ...
##  $ USD.CNY: num  6.22 6.23 6.22 6.22 6.23 ...
##  $ USD.JPY: num  90.7 90.7 91 91.3 92.5 ...
```

```
summary(exrates)
```

```
##         DATE         USD.EUR         USD.GBP         USD.CNY
##  1/10/2014:   1   Min.   :1.038   Min.   :1.212   Min.   :6.040
##  1/10/2017:   1   1st Qu.:1.107   1st Qu.:1.324   1st Qu.:6.178
##  1/10/2018:   1   Median :1.158   Median :1.514   Median :6.261
##  1/11/2016:   1   Mean   :1.199   Mean   :1.474   Mean   :6.401
##  1/11/2017:   1   3rd Qu.:1.314   3rd Qu.:1.573   3rd Qu.:6.627
##  1/11/2018:   1   Max.   :1.393   Max.   :1.716   Max.   :6.958
##  (Other)  :1247
##     USD.JPY
##  Min.   : 90.65
##  1st Qu.:102.14
##  Median :109.88
##  Mean   :109.33
##  3rd Qu.:116.76
##  Max.   :125.58
##
```

Questions

1. What is the nature of exchange rates? We want to reflect the ups and downs of rate movements, known to managers as currency appreciation and depreciation. First, we calculate percentage changes as log returns. Our interest is in the ups and downs. To look at that we use `if` and `else` statements to define a new column called `direction`. We will build a data frame to house this analysis.

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

```
##      USD.EUR     USD.GBP     USD.CNY     USD.JPY
## 2  0.1855770  0.41352605  0.03052233 -0.08821260
## 3  0.5915427  0.26629486 -0.08837968  0.44028690
## 4  0.1473405  0.39811737 -0.02894123  0.25228994
## 5  0.7919091 -0.70886373  0.12695761  1.37092779
## 6 -1.2124033 -0.04447127  0.09792040  0.03241316
## 7  0.3100091 -0.54159233 -0.05456676  0.82836254
```

```
tail(exrates.r)
```

```
##          USD.EUR    USD.GBP     USD.CNY    USD.JPY
## 1248  0.00000000 -0.2306640 -0.28869056 -0.2890175
## 1249 -0.06539153  0.6258788  0.07029877  0.5322280
## 1250  0.38356435  0.1719691 -0.05467255 -0.6227176
## 1251  0.91621024  1.6332111 -0.54837584 -1.1930381
## 1252  0.78784876  0.4637771 -0.72690896 -0.4131289
## 1253 -0.52990891 -0.5976884  0.01582429 -0.2948224
```

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

```
##  num [1:1252, 1:4] 0.186 0.592 0.147 0.792 -1.212 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1252] "2" "3" "4" "5" ...
##   ..$ : chr [1:4] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY"
```

```
# Create size and direction
size <- na.omit(abs(exrates.r))  # size is indicator of volatility
head(size)
```

```
##     USD.EUR    USD.GBP    USD.CNY    USD.JPY
## 2 0.1855770 0.41352605 0.03052233 0.08821260
## 3 0.5915427 0.26629486 0.08837968 0.44028690
## 4 0.1473405 0.39811737 0.02894123 0.25228994
## 5 0.7919091 0.70886373 0.12695761 1.37092779
## 6 1.2124033 0.04447127 0.09792040 0.03241316
## 7 0.3100091 0.54159233 0.05456676 0.82836254
```

```
colnames(size) <- paste(colnames(size),
    ".size", sep = "")  # Teetor
direction <- ifelse(exrates.r > 0, 1,
    ifelse(exrates.r < 0, -1, 0))  # another indicator of volatility
colnames(direction) <- paste(colnames(direction),
    ".dir", sep = "")
head(direction)
```

```
##   USD.EUR.dir USD.GBP.dir USD.CNY.dir USD.JPY.dir
## 2           1           1           1          -1
## 3           1           1          -1           1
## 4           1           1          -1           1
## 5           1          -1           1           1
## 6          -1          -1           1           1
## 7           1          -1          -1           1
```

```
# 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)
# for dplyr pivoting we need a data
```

```r
# frame
exrates.df <- data.frame(dates = dates,
    returns = exrates.r, size = size,
    direction = direction)
str(exrates.df)  # notice the returns.* and direction.* prefixes
```

```
## 'data.frame':    1252 obs. of  13 variables:
##  $ dates              : Date, format: "2013-01-29" "2013-01-30" ...
##  $ returns.USD.EUR    : num  0.186 0.592 0.147 0.792 -1.212 ...
##  $ returns.USD.GBP    : num  0.4135 0.2663 0.3981 -0.7089 -0.0445 ...
##  $ returns.USD.CNY    : num  0.0305 -0.0884 -0.0289 0.127 0.0979 ...
##  $ returns.USD.JPY    : num  -0.0882 0.4403 0.2523 1.3709 0.0324 ...
##  $ size.USD.EUR.size  : num  0.186 0.592 0.147 0.792 1.212 ...
##  $ size.USD.GBP.size  : num  0.4135 0.2663 0.3981 0.7089 0.0445 ...
##  $ size.USD.CNY.size  : num  0.0305 0.0884 0.0289 0.127 0.0979 ...
##  $ size.USD.JPY.size  : num  0.0882 0.4403 0.2523 1.3709 0.0324 ...
##  $ 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 0 0 ...
##  $ direction.USD.JPY.dir: num  -1 1 1 1 1 1 1 -1 -1 1 ...
```

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

```
## An 'xts' object on 2013-01-29/2018-01-26 containing:
##   Data: num [1:1252, 1:12] 0.186 0.592 0.147 0.792 -1.212 ...
##  - 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
exrates.zr <- na.omit(as.zooreg(exrates.xts))
str(exrates.zr)
```

```
## 'zooreg' series from 2013-01-29 to 2018-01-26
##   Data: num [1:1252, 1:12] 0.186 0.592 0.147 0.792 -1.212 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:12] "USD.EUR" "USD.GBP" "USD.CNY" "USD.JPY" ...
##   Index:  Date[1:1252], format: "2013-01-29" "2013-01-30" "2013-01-31" "2013-02-01" "2013-02-04" ...
##   Frequency: 1
```

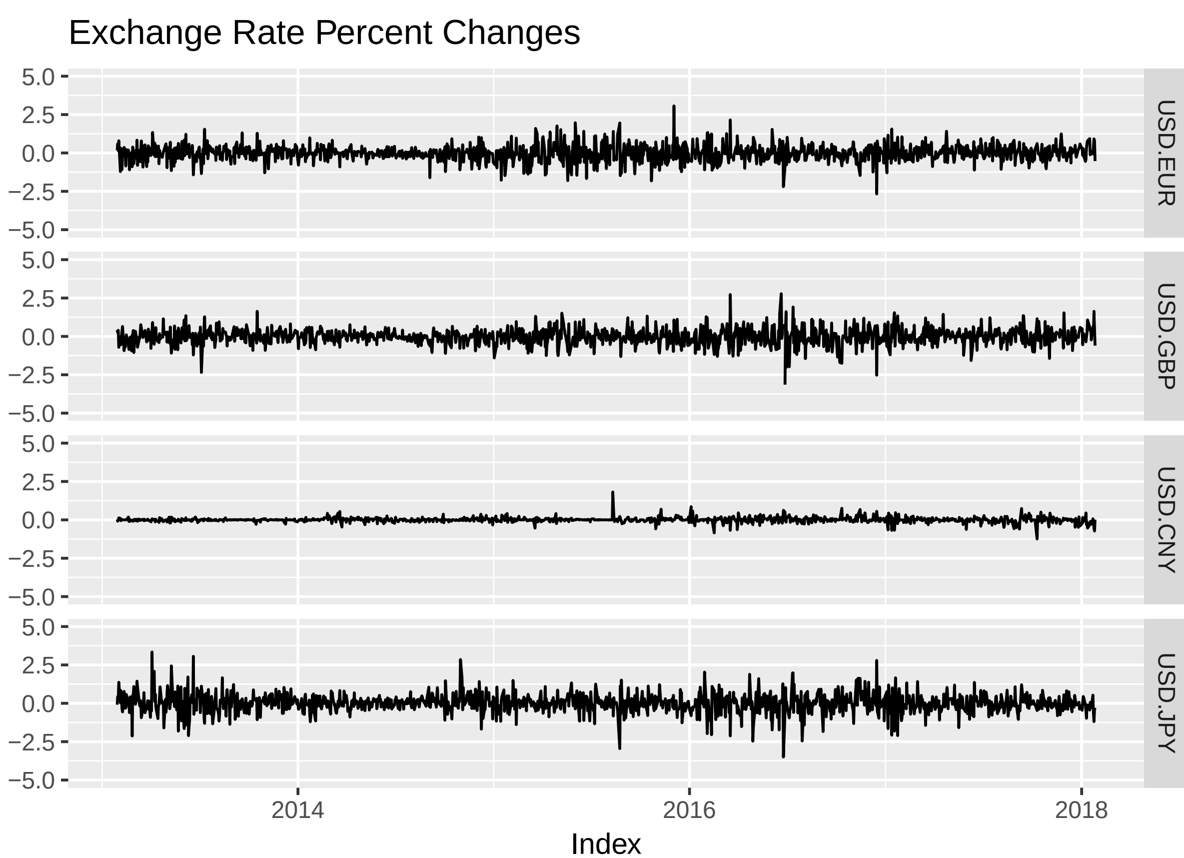
```r
head(exrates.xts)
```

```
##               USD.EUR     USD.GBP     USD.CNY     USD.JPY USD.EUR.size
## 2013-01-29  0.1855770  0.41352605  0.03052233 -0.08821260    0.1855770
## 2013-01-30  0.5915427  0.26629486 -0.08837968  0.44028690    0.5915427
## 2013-01-31  0.1473405  0.39811737 -0.02894123  0.25228994    0.1473405
## 2013-02-01  0.7919091 -0.70886373  0.12695761  1.37092779    0.7919091
## 2013-02-04 -1.2124033 -0.04447127  0.09792040  0.03241316    1.2124033
## 2013-02-05  0.3100091 -0.54159233 -0.05456676  0.82836254    0.3100091
```

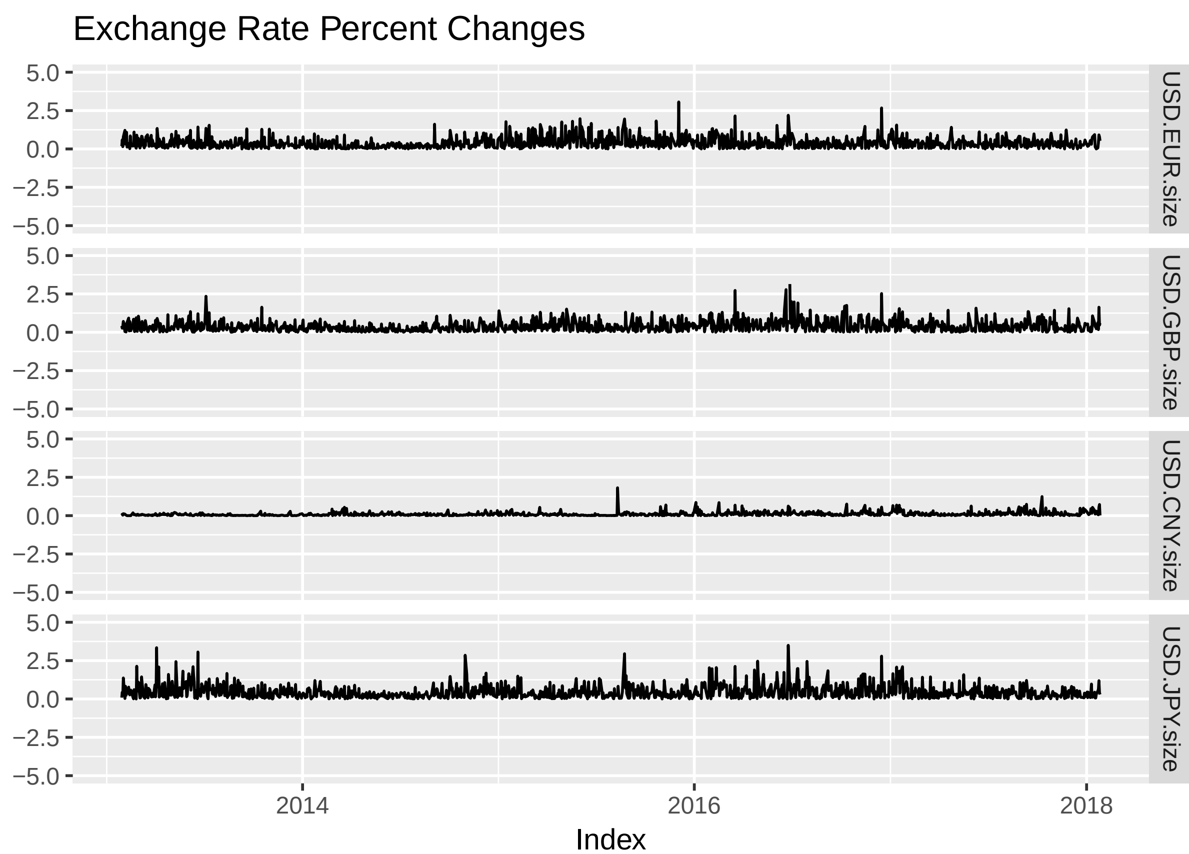
```
##            USD.GBP.size USD.CNY.size USD.JPY.size USD.EUR.dir USD.GBP.dir
## 2013-01-29   0.41352605   0.03052233   0.08821260           1           1
## 2013-01-30   0.26629486   0.08837968   0.44028690           1           1
## 2013-01-31   0.39811737   0.02894123   0.25228994           1           1
## 2013-02-01   0.70886373   0.12695761   1.37092779           1          -1
## 2013-02-04   0.04447127   0.09792040   0.03241316          -1          -1
## 2013-02-05   0.54159233   0.05456676   0.82836254           1          -1
##            USD.CNY.dir USD.JPY.dir
## 2013-01-29           1          -1
## 2013-01-30          -1           1
## 2013-01-31          -1           1
## 2013-02-01           1           1
## 2013-02-04           1           1
## 2013-02-05          -1           1
```

We can plot with the `ggplot2` package. In the `ggplot` statements we use `aes`, "aesthetics", to pick `x` (horizontal) and `y` (vertical) axes. Use `group =1` to ensure that all data is plotted. The added (`+`) `geom_line` is the geometrical method that builds the line plot.

```
library(ggplot2)
title.chg <- "Exchange Rate Percent Changes"
autoplot.zoo(exrates.xts[, 1:4]) + ggtitle(title.chg) +
    ylim(-5, 5)
```

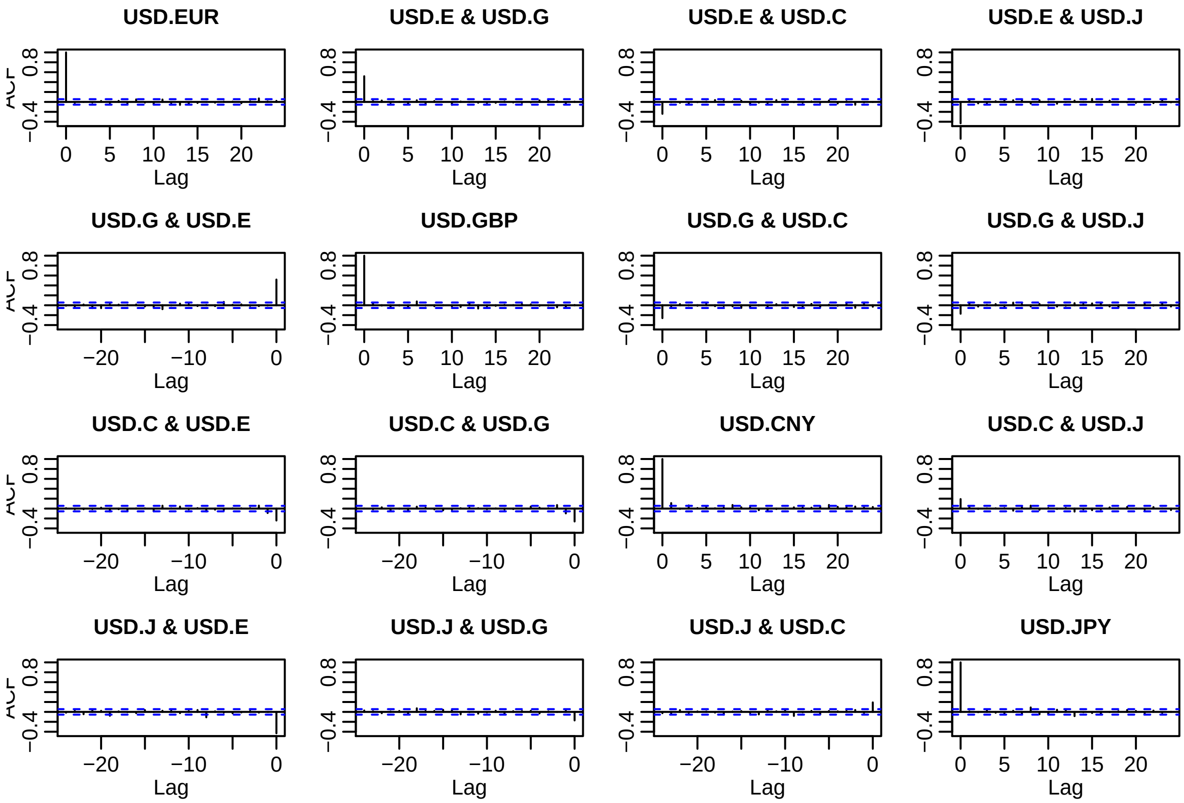



```
autoplot.zoo(exrates.xts[, 5:8]) + ggtitle(title.chg) +
    ylim(-5, 5)
```

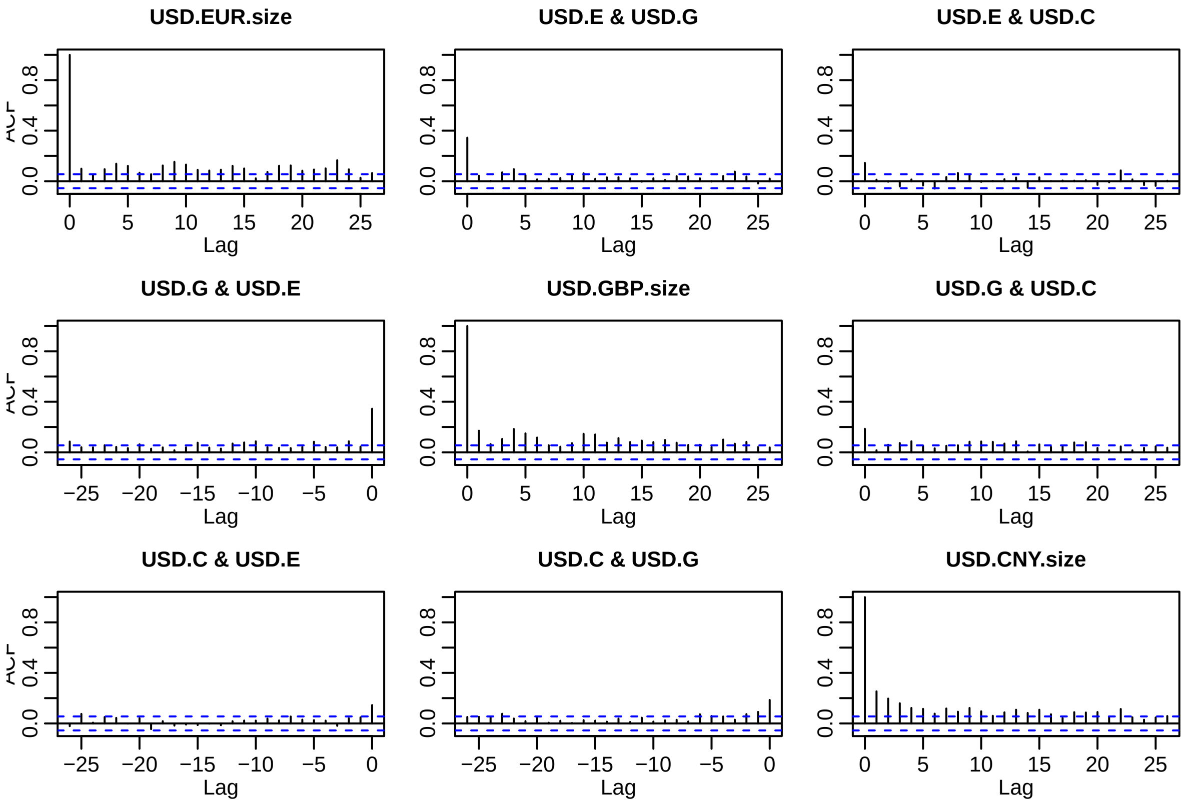


2. Let's dig deeper and compute mean, standard deviation, etc. Load the `data_moments()` function. Run the function using the `exrates` data and write a `knitr::kable()` report.

```
acf(coredata(exrates.xts[, 1:4]))  # returns
```




```
acf(coredata(exrates.xts[, 5:7]))  # sizes
```




```r
# Load the data_moments() function
# data_moments function INPUTS: r
# vector OUTPUTS: list of scalars
# (mean, sd, median, skewness,
# kurtosis)
data_moments <- function(data) {
    library(moments)
    library(matrixStats)
    mean.r <- colMeans(data)
    median.r <- colMedians(data)
    sd.r <- colSds(data)
    IQR.r <- colIQRs(data)
    skewness.r <- skewness(data)
    kurtosis.r <- kurtosis(data)
    result <- data.frame(mean = mean.r,
        median = median.r, std_dev = sd.r,
        IQR = IQR.r, skewness = skewness.r,
        kurtosis = kurtosis.r)
    return(result)
}
# Run data_moments()
answer <- data_moments(exrates.xts[,
    5:8])
# Build pretty table
answer <- round(answer, 4)
knitr::kable(answer)
```

	mean	median	std_dev	IQR	skewness	kurtosis
USD.EUR.size	0.4003	0.2935	0.3695	0.4313	1.7944	8.0424
USD.GBP.size	0.4008	0.2995	0.4266	0.4173	6.0881	93.5604
USD.CNY.size	0.1027	0.0601	0.1375	0.1154	3.9004	31.0222
USD.JPY.size	0.4533	0.3250	0.4455	0.4684	2.2201	10.4898

```r
mean(exrates.xts[, 4])
```

```
## [1] 0.01419772
```

Part 2

We will use the data from Set A to investigate the interactions of the distribution of exchange rates.

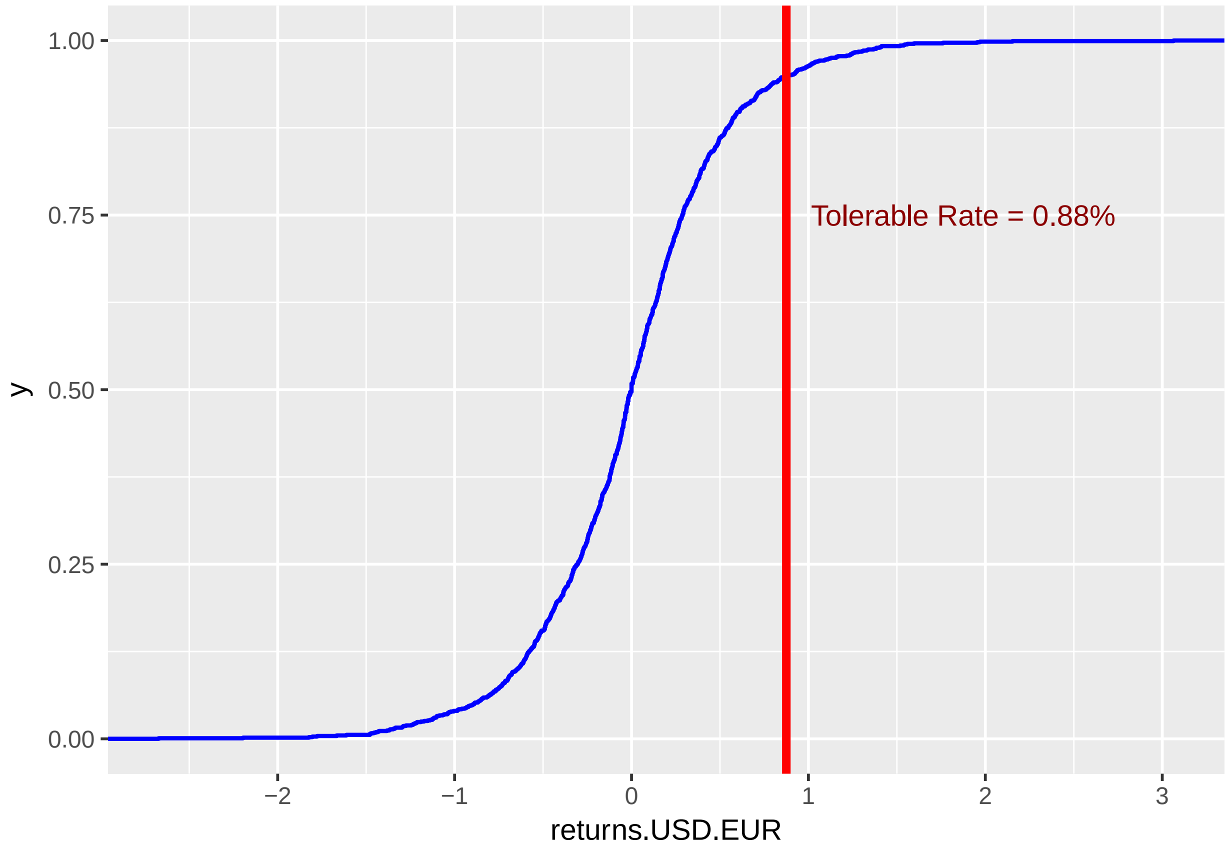
Problem

We want to characterize the distribution of up and down movements visually. Also we would like to repeat the analysis periodically for inclusion in management reports.

Questions

1. How can we show the shape of our exposure to euros, especially given our tolerance for risk? Suppose corporate policy set tolerance at 95%. Let's use the `exrates.df` data frame with `ggplot2` and the cumulative relative frequency function `stat_ecdf`.

```r
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")
```

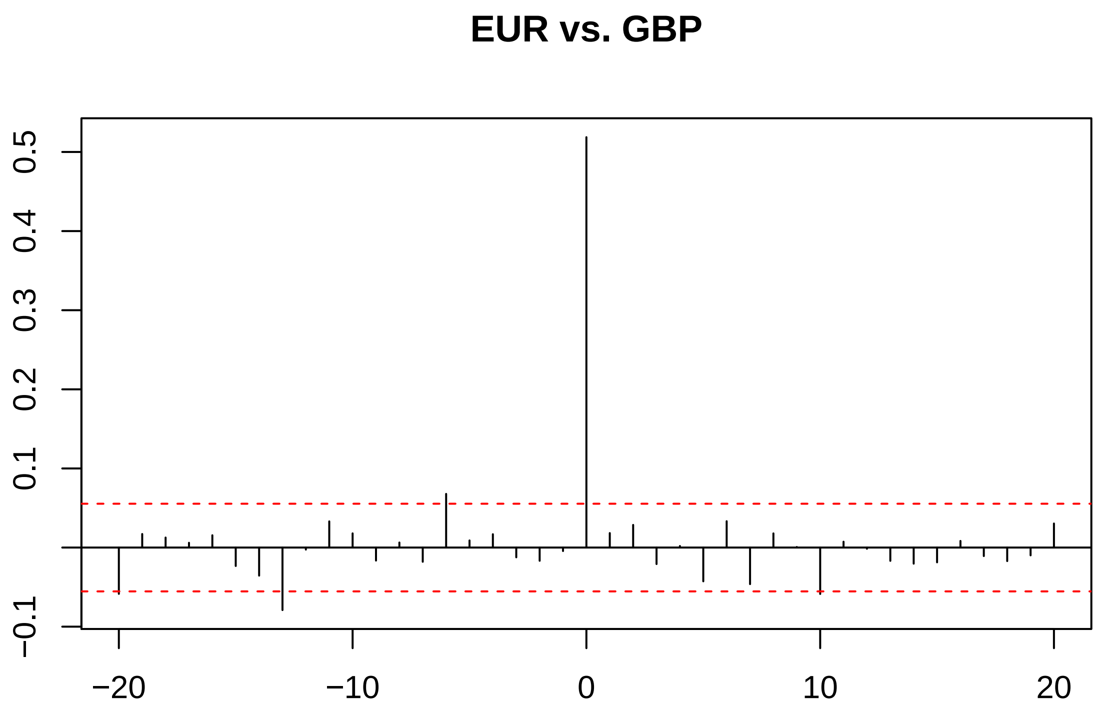


2. What is the history of correlations in the exchange rate markets? If this is a "history," then we have to manage the risk that conducting business in one country will definitely affect business in another. Further that bad things will be followed by more bad things more often than good things. We will create a rolling correlation function, `corr_rolling`, and embed this function into the `rollapply()` function (look this one up!).

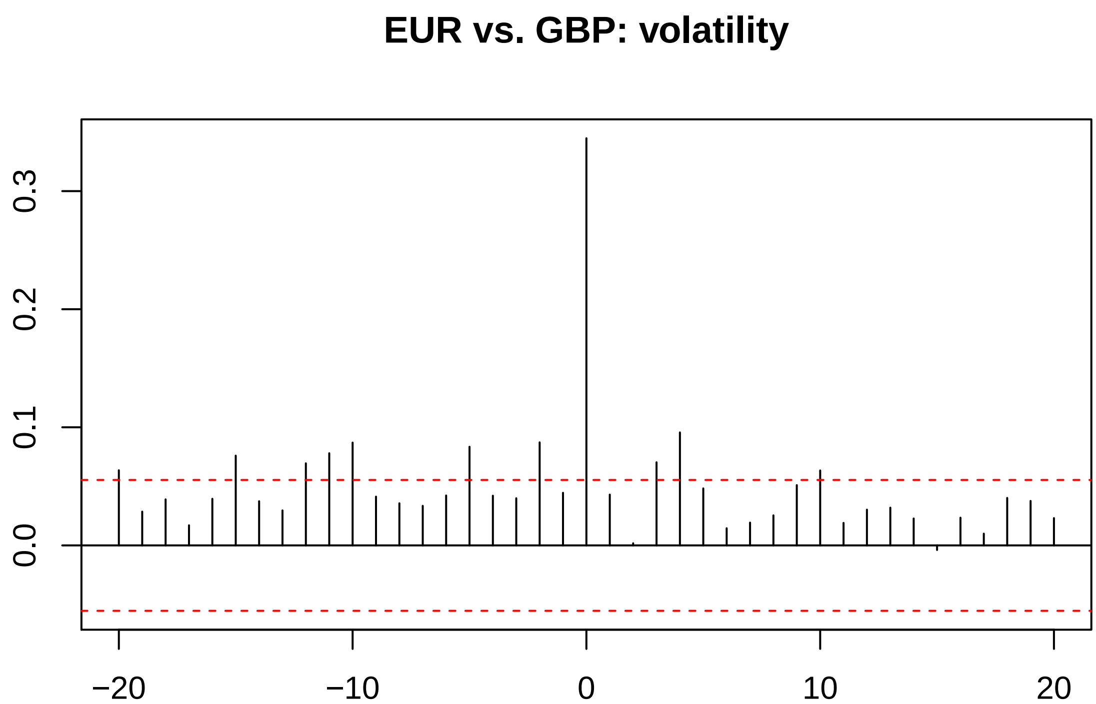
```
one <- ts(exrates.df$returns.USD.EUR)
two <- ts(exrates.df$returns.USD.GBP)
# or
one <- ts(exrates.zr[, 1])
two <- ts(exrates.zr[, 2])
ccf(one, two, main = "GBP vs. EUR", lag.max = 20,
    xlab = "", ylab = "", ci.col = "red")
```


```
# build function to repeat these
# routines
run_ccf <- function(one, two, main = "one vs. two",
    lag = 20, color = "red") {
    # one and two are equal length series
    # main is title lag is number of lags
    # in cross-correlation color is color
    # of dashed confidence interval
    # bounds
    stopifnot(length(one) == length(two))
    one <- ts(one)
    two <- ts(two)
    main <- main
    lag <- lag
    color <- color
    ccf(one, two, main = main, lag.max = lag,
        xlab = "", ylab = "", ci.col = color)
    # end run_ccf
}
one <- ts(exrates.df$returns.USD.EUR)
two <- ts(exrates.df$returns.USD.GBP)
# or
one <- exrates.zr[, 1]
two <- exrates.zr[, 2]
title <- "EUR vs. GBP"
run_ccf(one, two, main = title, lag = 20,
```

```
    color = "red")
```




```
# now for volatility (sizes)
one <- abs(exrates.zr[, 1])
two <- abs(exrates.zr[, 2])
title <- "EUR vs. GBP: volatility"
run_ccf(one, two, main = title, lag = 20,
    color = "red")
```




```
# We see some small raw correlations
# across time with raw returns. More
# revealing, we see volatility of
# correlation clustering using return
# sizes.
```

One more experiment, a rolling correlation using this 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]  # only returns
corr.returns <- rollapply(ALL.r, width = 20,
    corr_rolling, align = "right", by.column = FALSE)
head(corr.returns)
```

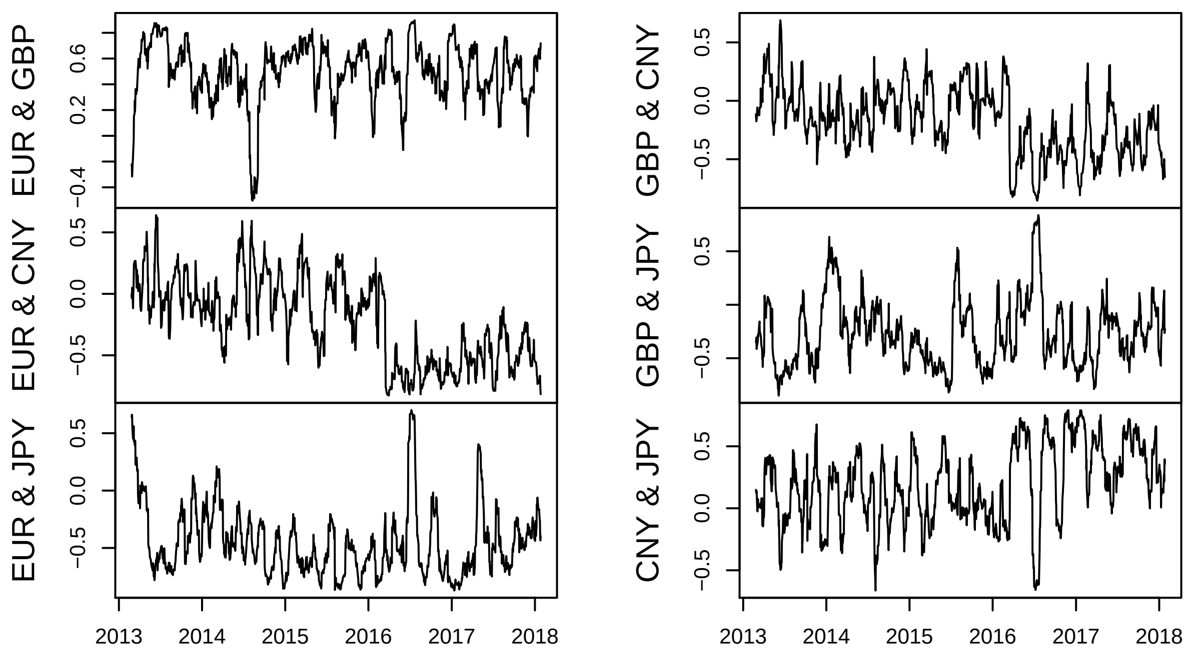
```
##
## 2013-02-26 -0.2213908 -0.0013282456 0.6389409 -0.11542729 -0.3494970
## 2013-02-27 -0.2391031 -0.0367435201 0.6614446 -0.15923778 -0.3253478
## 2013-02-28 -0.3166391  0.0494401197 0.6513980 -0.17599423 -0.3060324
## 2013-03-01 -0.2695033  0.0509298222 0.5484124 -0.16546653 -0.4133037
## 2013-03-04 -0.1584935 -0.1172357964 0.4510687 -0.05696293 -0.3642573
## 2013-03-05 -0.1097154 -0.0006885039 0.4900458 -0.11421984 -0.3600784
##
```

```
## 2013-02-26  0.14895485
## 2013-02-27  0.13192672
## 2013-02-28  0.12161559
## 2013-03-01  0.13905783
## 2013-03-04 -0.02337018
## 2013-03-05 -0.02525167
```

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

```
## 'zooreg' series from 2013-02-26 to 2018-01-26
##   Data: num [1:1233, 1:6] -0.221 -0.239 -0.317 -0.27 -0.158 ...
##   Index:  Date[1:1233], format: "2013-02-26" "2013-02-27" "2013-02-28" "2013-03-01" "2013-03-04" ...
##   Frequency: 1
```

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




```r
#' \t
```

4. How related are correlations and volatilities? Put another way, do we have to be concerned that inter-market transactions (e.g., customers and vendors transacting in more than one currency) can affect transactions in a single market? Let's take the `exrate` data to understand how dependent correlations and volatilities depend upon one another.

```r
library(matrixStats)
R.corr <- apply.monthly(as.xts(ALL.r),
    FUN = cor)
```

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

```
## An 'xts' object on 2013-01-31/2018-01-26 containing:
##   Data: num [1:61, 1:16] 1 1 1 1 1 1 1 1 1 1 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
head(ALL.r)
```

```
##               USD.EUR     USD.GBP     USD.CNY     USD.JPY
## 2013-01-29  0.1855770  0.41352605  0.03052233 -0.08821260
## 2013-01-30  0.5915427  0.26629486 -0.08837968  0.44028690
## 2013-01-31  0.1473405  0.39811737 -0.02894123  0.25228994
## 2013-02-01  0.7919091 -0.70886373  0.12695761  1.37092779
## 2013-02-04 -1.2124033 -0.04447127  0.09792040  0.03241316
## 2013-02-05  0.3100091 -0.54159233 -0.05456676  0.82836254
```

```r
tail(R.corr)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5] [,6]
## 2017-08-31    1 0.67155957 -0.4151418 -0.6369447 0.67155957    1
## 2017-09-29    1 0.50008090 -0.6520743 -0.4791779 0.50008090    1
## 2017-10-31    1 0.62842505 -0.4730468 -0.3895148 0.62842505    1
## 2017-11-30    1 0.01480578 -0.2490070 -0.3304712 0.01480578    1
## 2017-12-29    1 0.60885446 -0.5330930 -0.4200583 0.60885446    1
## 2018-01-26    1 0.71321310 -0.8077317 -0.4060424 0.71321310    1
##                   [,7]       [,8]       [,9]       [,10] [,11]     [,12]
## 2017-08-31 -0.46170916 -0.3759847 -0.4151418 -0.46170916     1 0.5395938
## 2017-09-29 -0.37205127 -0.2000270 -0.6520743 -0.37205127     1 0.6100217
## 2017-10-31 -0.51843547 -0.3730174 -0.4730468 -0.51843547     1 0.4670277
## 2017-11-30 -0.06637416  0.1234213 -0.2490070 -0.06637416     1 0.2592880
## 2017-12-29 -0.36008523 -0.4301835 -0.5330930 -0.36008523     1 0.2651777
## 2018-01-26 -0.64811958 -0.2306095 -0.8077317 -0.64811958     1 0.3655205
##                 [,13]      [,14]     [,15] [,16]
## 2017-08-31 -0.6369447 -0.3759847 0.5395938     1
## 2017-09-29 -0.4791779 -0.2000270 0.6100217     1
## 2017-10-31 -0.3895148 -0.3730174 0.4670277     1
## 2017-11-30 -0.3304712  0.1234213 0.2592880     1
## 2017-12-29 -0.4200583 -0.4301835 0.2651777     1
## 2018-01-26 -0.4060424 -0.2306095 0.3655205     1
```

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

```
##            [,1]       [,2]        [,3]       [,4]       [,5] [,6]
## 2013-01-31    1 -0.9850569 -0.82450861  0.7203934 -0.9850569    1
## 2013-02-28    1 -0.3756889  0.06319731  0.6505334 -0.3756889    1
## 2013-03-29    1  0.5958494  0.03932059 -0.1122205  0.5958494    1
##                   [,7]       [,8]        [,9]       [,10] [,11]
## 2013-01-31  0.90964362 -0.8290806 -0.82450861  0.90964362     1
## 2013-02-28 -0.15577112 -0.3383913  0.06319731 -0.15577112     1
## 2013-03-29  0.09110179 -0.5803680  0.03932059  0.09110179     1
##                  [,12]      [,13]      [,14]       [,15] [,16]
## 2013-01-31 -0.98642428  0.7203934 -0.8290806 -0.98642428     1
## 2013-02-28  0.12801906  0.6505334 -0.3383913  0.12801906     1
```

```
## 2013-03-29 -0.07042904 -0.1122205 -0.5803680 -0.07042904     1
```

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

```
##               USD.EUR    USD.GBP    USD.CNY   USD.JPY
## 2013-01-31 0.2461658 0.08092345 0.05945101 0.2678919
## 2013-02-28 0.5816039 0.47089762 0.07517850 0.7402470
## 2013-03-29 0.5396914 0.43001568 0.04259678 0.5783480
```

```r
# Form correlation matrix for one
# month
R.corr.1 <- matrix(R.corr[20, ], 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 -0.45594725  0.00100771 -0.59408796
## USD.GBP -0.45594725  1.00000000 -0.07241744  0.03686126
## USD.CNY  0.00100771 -0.07241744  1.00000000  0.22392476
## USD.JPY -0.59408796  0.03686126  0.22392476  1.00000000
```

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

```
##                  [,1]          [,2]        [,3]        [,4]        [,5]
## 2013-01-31 -0.9850569 -8.245086e-01  0.72039337  0.90964362 -0.82908064
## 2013-02-28 -0.3756889  6.319731e-02  0.65053343 -0.15577112 -0.33839131
## 2013-03-29  0.5958494  3.932059e-02 -0.11222051  0.09110179 -0.58036805
## 2013-04-30  0.6276803  3.686426e-01 -0.03163998  0.23406810 -0.01537205
## 2013-05-31  0.8092350  3.293765e-05 -0.63744877  0.12222145 -0.66640046
## 2013-06-28  0.8174040  1.255839e-02 -0.59736631  0.06057553 -0.66857681
##                   [,6]
## 2013-01-31 -0.98642428
## 2013-02-28  0.12801906
## 2013-03-29 -0.07042904
## 2013-04-30  0.40224431
## 2013-05-31 -0.00527013
## 2013-06-28 -0.13428342
```

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

```
##             EUR & GBP   EUR & CNY  EUR & JPY   GBP & CNY  GBP & JPY
## 2013-01-31 -0.9850569 -0.82450861  0.7203934  0.90964362 -0.8290806
## 2013-02-28 -0.3756889  0.06319731  0.6505334 -0.15577112 -0.3383913
## 2013-03-29  0.5958494  0.03932059 -0.1122205  0.09110179 -0.5803680
##              CNY & JPY
## 2013-01-31 -0.98642428
## 2013-02-28  0.12801906
## 2013-03-29 -0.07042904
```

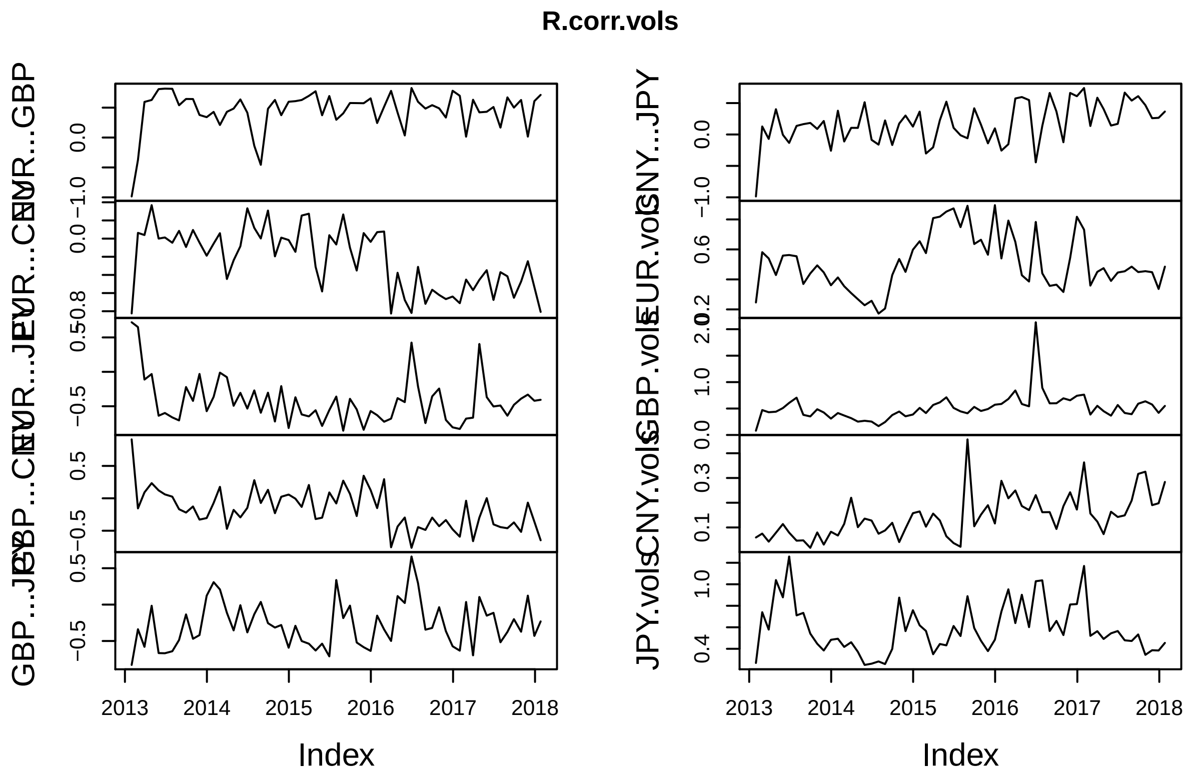
```
head(R.vols, 3)
```

```
##            EUR.vols   GBP.vols   CNY.vols  JPY.vols
## 2013-01-31 0.2461658 0.08092345 0.05945101 0.2678919
## 2013-02-28 0.5816039 0.47089762 0.07517850 0.7402470
## 2013-03-29 0.5396914 0.43001568 0.04259678 0.5783480
```

```
R.corr.vols <- merge(R.corr, R.vols)
head(R.corr.vols)
```

```
##              EUR...GBP     EUR...CNY   EUR...JPY   GBP...CNY   GBP...JPY
## 2013-01-31 -0.9850569 -8.245086e-01  0.72039337  0.90964362 -0.82908064
## 2013-02-28 -0.3756889  6.319731e-02  0.65053343 -0.15577112 -0.33839131
## 2013-03-29  0.5958494  3.932059e-02 -0.11222051  0.09110179 -0.58036805
## 2013-04-30  0.6276803  3.686426e-01 -0.03163998  0.23406810 -0.01537205
## 2013-05-31  0.8092350  3.293765e-05 -0.63744877  0.12222145 -0.66640046
## 2013-06-28  0.8174040  1.255839e-02 -0.59736631  0.06057553 -0.66857681
##              CNY...JPY  EUR.vols   GBP.vols   CNY.vols  JPY.vols
## 2013-01-31 -0.98642428 0.2461658 0.08092345 0.05945101 0.2678919
## 2013-02-28  0.12801906 0.5816039 0.47089762 0.07517850 0.7402470
## 2013-03-29 -0.07042904 0.5396914 0.43001568 0.04259678 0.5783480
## 2013-04-30  0.40224431 0.4290341 0.44029469 0.07857697 1.0391331
## 2013-05-31 -0.00527013 0.5587756 0.50856733 0.11377078 0.8785330
## 2013-06-28 -0.13428342 0.5626975 0.60663436 0.07906074 1.2583761
```

```
#'
plot.zoo(R.corr.vols)
```



```r
#' \t
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] 61
```

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

1. We set `taus` as the quantiles of interest.
2. We run the quantile regression using the `quantreg` package and a call to the `rq` function.
3. We can overlay the quantile regression results onto the standard 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.

```r
library(quantreg)
# hist(rho.fisher[, 1])
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) -0.76918  1.42345   -0.54037  0.59098
## EUR.vols     1.34494  1.75572    0.76603  0.44671
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.1
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.27473  0.49775   -0.55194  0.58307
## EUR.vols     0.78995  0.65026    1.21482  0.22927
```

```
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.15
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.27962  0.29209   -0.95731  0.34232
## EUR.vols     0.81834  0.49951    1.63829  0.10668
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.2
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.33536  0.30166   -1.11172  0.27077
## EUR.vols     1.14215  0.52619    2.17061  0.03400
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) -0.00416  0.30636   -0.01359  0.98920
## EUR.vols     0.88032  0.54904    1.60337  0.11419
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.3
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.06495 0.26311    0.24686 0.80587
## EUR.vols    0.80312 0.47814    1.67967 0.09831
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.35
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.05933 0.22244    0.26672 0.79061
## EUR.vols    0.89843 0.40588    2.21355 0.03074
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.4
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.14792 0.17135    0.86327 0.39149
## EUR.vols    0.80193 0.30835    2.60072 0.01174
```

```
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.45
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.22566 0.17474    1.29140 0.20160
## EUR.vols    0.72103 0.31557    2.28487 0.02593
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.25802 0.12461    2.07060 0.04278
## EUR.vols    0.68514 0.23355    2.93360 0.00477
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.55
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.25366 0.10841    2.33990 0.02269
## EUR.vols    0.73019 0.21547    3.38878 0.00126
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.6
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.31769 0.11856    2.67963 0.00954
## EUR.vols    0.68373 0.23773    2.87600 0.00560
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.65
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.33399 0.10615    3.14639 0.00259
## EUR.vols    0.72447 0.21507    3.36853 0.00134
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.7
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.26579 0.12874    2.06455 0.04337
## EUR.vols    0.93308 0.26125    3.57160 0.00071
```

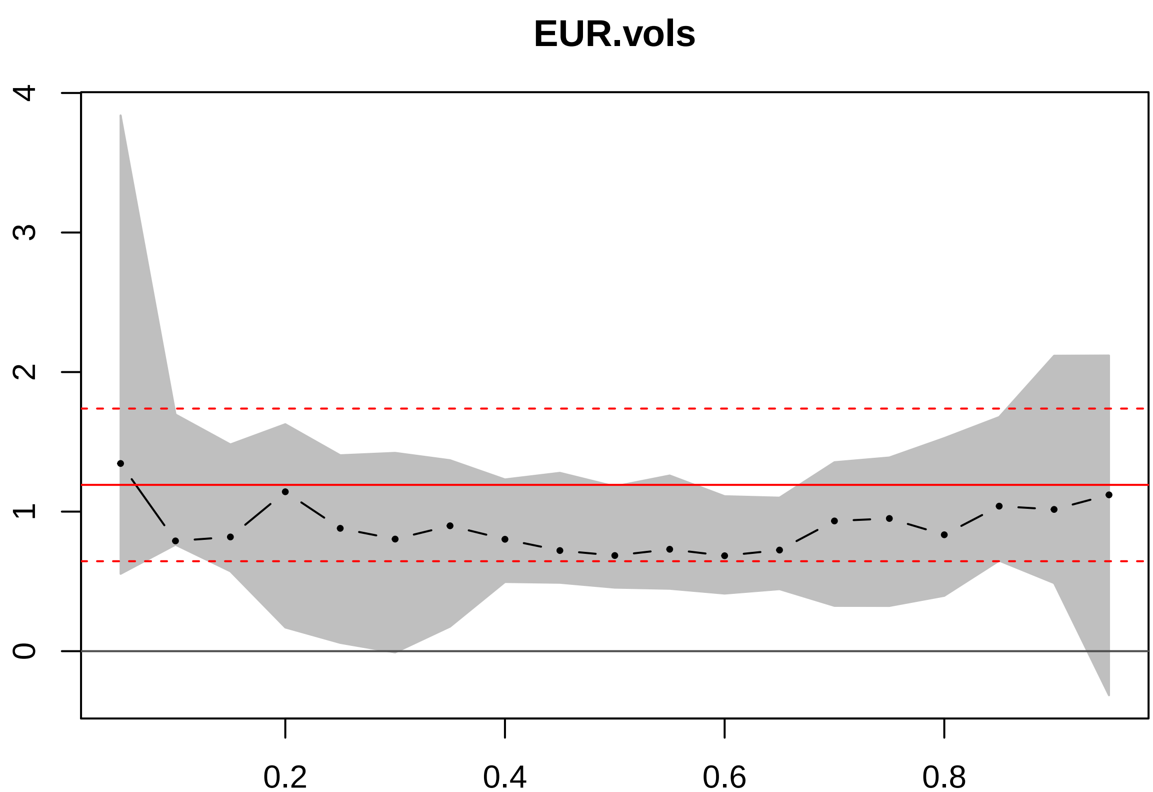
```
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.27334 0.13674    1.99901 0.05022
## EUR.vols    0.95076 0.29261    3.24921 0.00191
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.8
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.36878 0.16355    2.25491 0.02786
## EUR.vols    0.83408 0.34733    2.40142 0.01950
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.85
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.30936 0.16820    1.83926 0.07091
## EUR.vols    1.03920 0.35117    2.95926 0.00443
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.9
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.38600 0.19961    1.93382 0.05794
## EUR.vols    1.01559 0.42961    2.36397 0.02139
##
## Call: rq(formula = rho.fisher[, 1] ~ EUR.vols, tau = taus)
##
## tau: [1] 0.95
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 0.49892 0.25169    1.98230 0.05211
## EUR.vols    1.12011 0.49970    2.24159 0.02876
```

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

```
##
## Call:
## lm(formula = rho.fisher[, 1] ~ EUR.vols)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.65006 -0.12180  0.08383  0.25931  0.57049
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.08782    0.18036  -0.487 0.628131
## EUR.vols     1.19149    0.33257   3.583 0.000689 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.472 on 59 degrees of freedom
## Multiple R-squared:  0.1787, Adjusted R-squared:  0.1648
## F-statistic: 12.84 on 1 and 59 DF,  p-value: 0.0006895
```

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



Here we build the estimations and plot the upper and lower bounds.

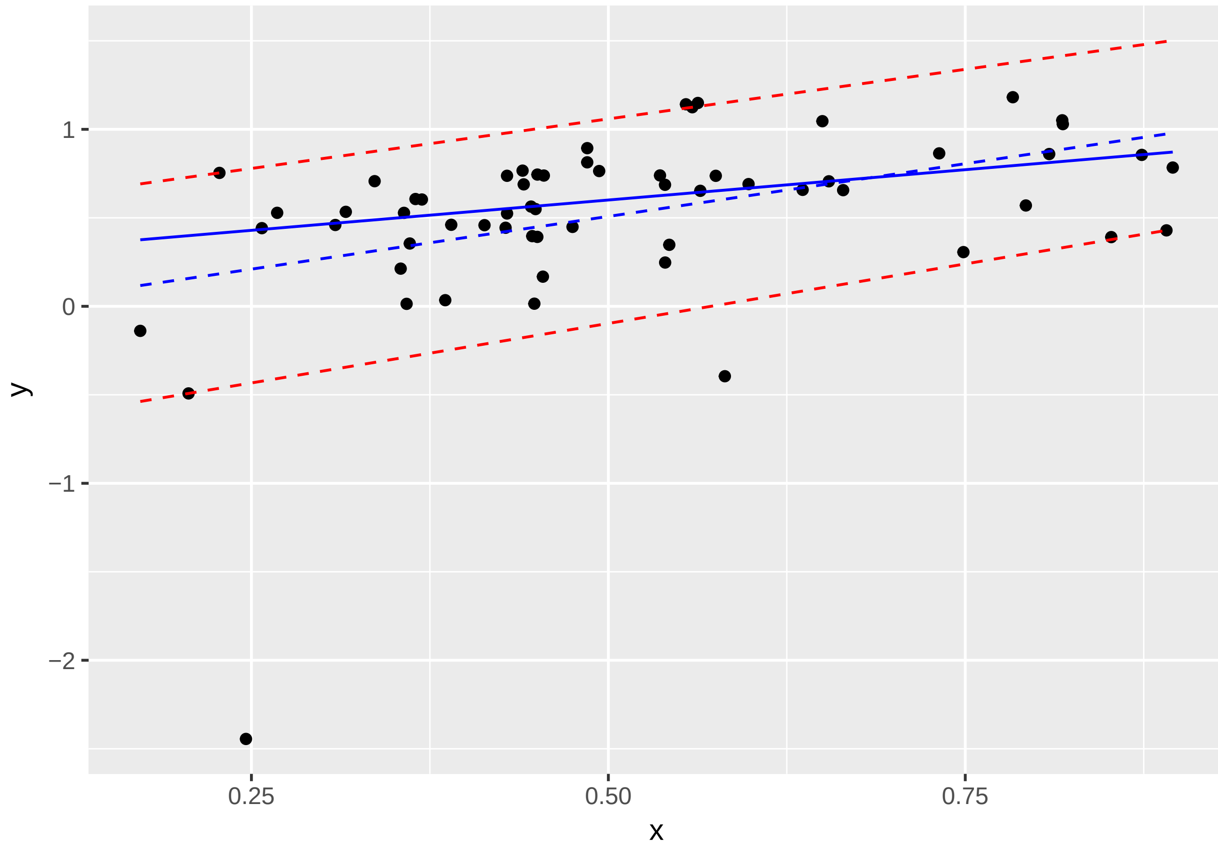
```r
taus1 <- c(0.05, 0.5, 0.95)  # fit the confidence interval (CI)
EUR.GBP.p <- predict(rq(rho.fisher[,
    1] ~ EUR.vols, tau = taus1))
EUR.GBP.lm.p <- predict(lm(rho.fisher[,
    1] ~ EUR.vols))
colnames(EUR.GBP.p) <- c(paste("tau",
    taus1[1] * 100, sep = ""), paste("tau",
    taus1[2] * 100, sep = ""), paste("tau",
    taus1[3] * 100, sep = ""))
head(EUR.GBP.p)
```

```
##              tau5     tau50     tau95
## [1,] -0.43810353 0.4266791 0.7746502
## [2,]  0.01303888 0.6565012 1.1503789
## [3,] -0.04333062 0.6277853 1.1034322
## [4,] -0.19215761 0.5519695 0.9794834
## [5,] -0.01766372 0.6408606 1.1248086
## [6,] -0.01238897 0.6435477 1.1292016
```

```
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)
```

```
##           x          y         y.5      y.50      y.95
## 1 0.2461658 -2.4445777 -0.43810353 0.4266791 0.7746502
## 2 0.5816039 -0.3950305  0.01303888 0.6565012 1.1503789
## 3 0.5396914  0.6866870 -0.04333062 0.6277853 1.1034322
## 4 0.4290341  0.7375791 -0.19215761 0.5519695 0.9794834
## 5 0.5587756  1.1248086 -0.01766372 0.6408606 1.1248086
## 6 0.5626975  1.1489441 -0.01238897 0.6435477 1.1292016
##   y.lm....EUR.GBP.lm.p
## 1            0.2054854
## 2            0.6051552
## 3            0.5552171
## 4            0.4233704
## 5            0.5779556
## 6            0.5826285
```

```
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")
```



Interpretations?