

# Homework Assignment 9: Patterns of Exchange

36-402, Advanced Data Analysis, Spring 2011

## SOLUTIONS

1. (a) ANSWER:

```
# Load fx.csv as "fx"
fx = read.csv("http://www.stat.cmu.edu/~cshalizi/402/hw/09/fx.csv",
  header = T, row.names = 1)
# verify that the matrix is now 3779x22:
dim(fx); colnames(fx)
```

As for getting the names of the currencies from their three-letter codes, there are several possible sources. One is Wikipedia under “Currency Codes” (which redirects to “ISO 4217”, since the International Standards Organization actually has an official standard for this). Here’s another:

```
# see http://www.currencysystem.com/codes/
# Copy-paste entire list into a text file, here called codes.txt.
# Clean up the data a bit in the text file, and insert semicolons, or whatever
# Then read in the file:
codes = read.table("hw9/codes.txt", sep = ";")
# Make a matrix that will give each currency and code:
translation = as.data.frame(matrix(rep(NA, 22*2), ncol = 2))
colnames(translation) = c("currency", "code")
translation$code = colnames(fx)
for(ii in 1:22){
  match.index = which(codes[2] == colnames(fx)[ii])
  translation$currency[ii] = as.character(codes[match.index,1])}
```

```
> translation
      currency code
1      US dollar  USD
2    Japanese yen  JPY
3         EU euro   EUR
4   British pound  GBP
5 Australian dollar AUD
6   Brazilian real  BRL
7   Canadian dollar CAD
```

8	Chinese yuan renminbi	CNY
9	Danish krone	DKK
10	Hong Kong SAR dollar	HKD
11	Indian rupee	INR
12	Malaysian ringgit	MYR
13	Mexican peso	MXN
14	Norwegian krone	NOK
15	New Zealand dollar	NZD
16	South Korean won	KRW
17	Swedish krona	SEK
18	Singapore dollar	SGD
19	Sri Lanka rupee	LKR
20	South African rand	ZAR
21	Taiwan New dollar	TWD
22	Thai baht	THB

Making this table is not strictly necessary for the problem, but simplifies some later coding.

- (b) ANSWER: Rates are given as the number of units of a currency that exchanges with 1 Swiss franc. So 0.94 USD means that 94 US cents was worth 1 Swiss franc. So yen per dollar is (JPY/Swiss) over (USD/Swiss):

```
> fx["2003-02-28","JPY"]/fx["2003-02-28","USD"]
[1] 118.22
```

- (c) ANSWER:

```
# histograms and time series:
par(mfrow = c(5,5), mar = c(1,1,3,1))
for(cur.code in translation$code){ #cur.code = "USD"
  hist(fx[cur.code][[1]], xlab = "", ylab = "", main = "")
  mtext(cur.code)
}

par(mfrow = c(5,5), mar = c(1,1,3,1))
for(cur.code in translation$code){ #cur.code = "USD"
  plot(fx[cur.code][[1]], xlab = "", ylab = "", main = "",
       cex = 0.2, xaxt = "n")
  mtext(cur.code)
}
```

See Figure 1 for distributions of exchange rates for each currency.

See Figure 2 for time series plots of exchange rates for each currency.

The plots show a wide variety of ranges and distributions for the exchange rates. There is no particular reason to think that they *should* resemble each other. The fact that there are usually  $\approx 100$

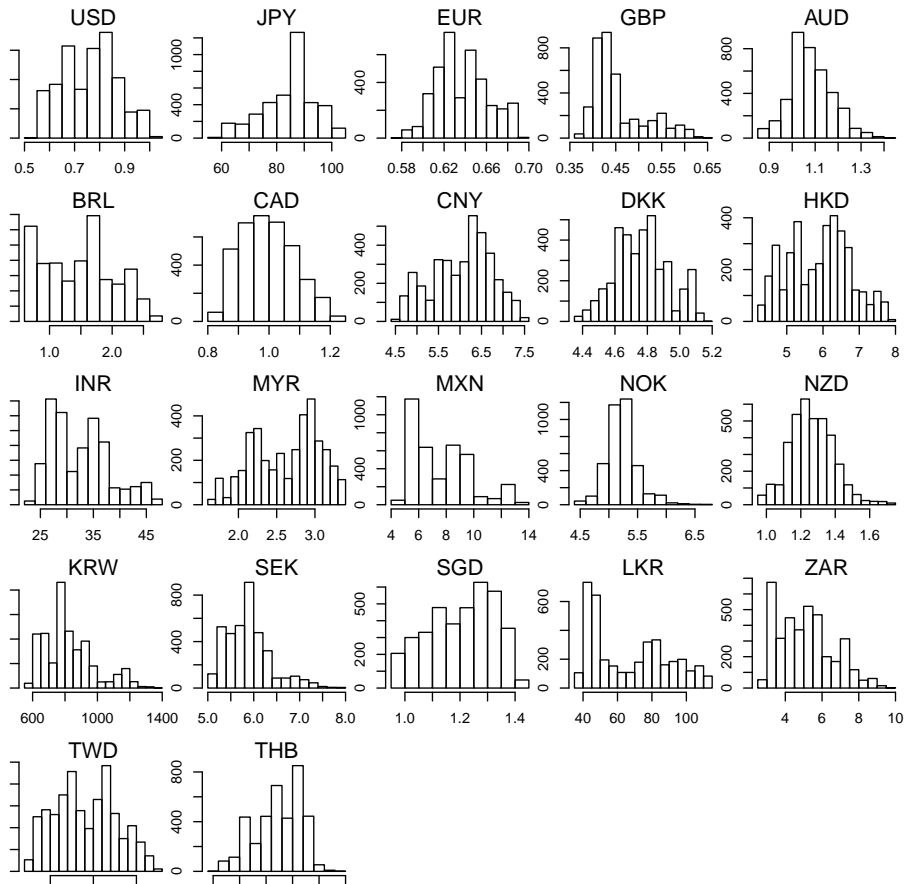


Figure 1: Histograms of exchange rates against the Swiss franc.

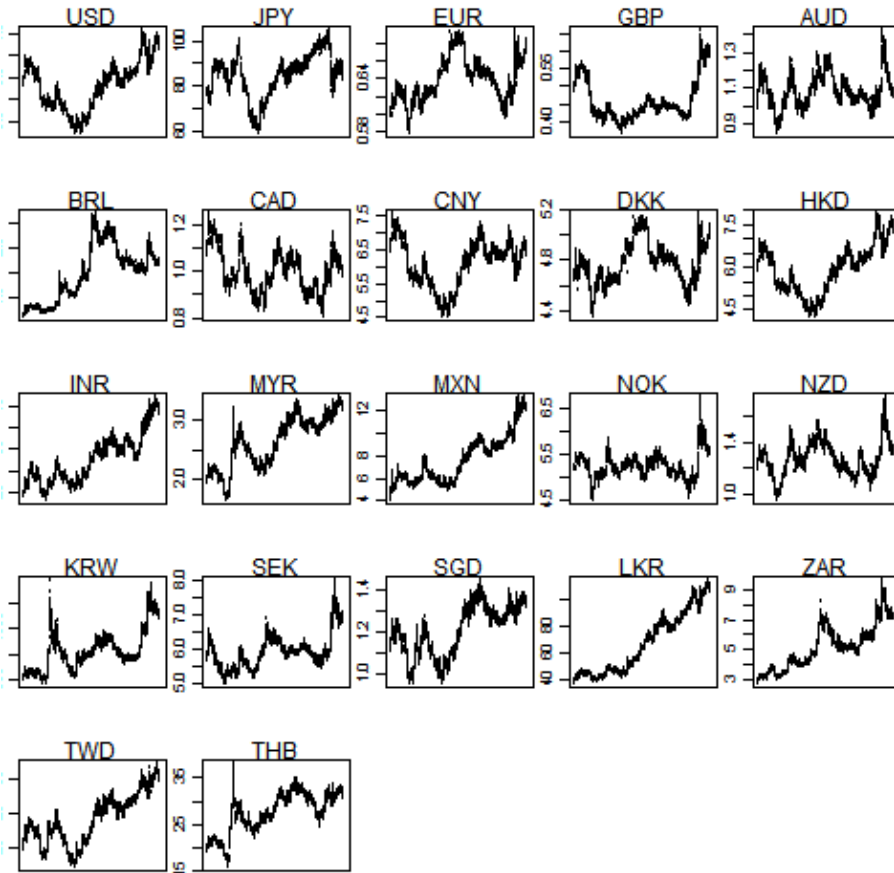


Figure 2: Time series of exchange rates against the Swiss franc.

yen to the Swiss franc but  $\approx 1$  Canadian dollars mostly reflects the historical accident that the yen is simply a smaller unit of currency for most purposes; if Canada switched to quoting prices in pennies the difference would go away. There is also no reason to expect the distributions to be similar, since there are many different economic forces which affect exchange rates, and these vary from country to country.

2. (a) ANSWER: Note that  $\log\left(\frac{r_t}{r_{t-1}}\right) = \log(r_t) - \log(r_{t-1})$ . Now use the hint:

```
fx.diff.log = as.data.frame(apply(log(fx), 2, diff))
```

Even though `fx` is a data frame, `apply` pushes its output down to a mere array, so we re-cast this back up to a data frame.

- (b) ANSWER: See Figure 3 for distributions of exchange rates for each currency. Now the distributions for each currency look rather similar, with some variability in the range of the distributions and extremity of outliers. The seeming normality of each distribution should not be too surprising; precisely because there are many highly idiosyncratic and unpredictable forces which move them, it's plausible that the central limit theorem should be at work<sup>1</sup>

3. (a) ANSWER: To make the table readable, we multiply the covariances by 10000:

```
covariances = round(10000*cov(fx.diff.log), 2)
```

	USD	JPY	EUR	GBP	AUD	BRL	CAD	CNY	DKK	HKD	INR	MYR	MXN	NOK	NZD	KRW	SEK	SGD	LKR	ZAR	TWD	THB
USD	0.49	0.26	0.08	0.25	0.30	0.46	0.39	0.48	0.09	0.48	0.46	0.46	0.54	0.11	0.28	0.45	0.13	0.39	0.48	0.31	0.45	0.43
JPY	0.26	0.57	0.02	0.10	0.11	0.17	0.17	0.26	0.02	0.26	0.24	0.28	0.24	0.02	0.12	0.23	0.03	0.28	0.27	0.10	0.26	0.31
EUR	0.08	0.02	0.08	0.09	0.13	0.14	0.11	0.08	0.08	0.08	0.09	0.09	0.13	0.09	0.12	0.10	0.10	0.08	0.08	0.13	0.08	0.08
GBP	0.25	0.10	0.09	0.36	0.29	0.34	0.27	0.25	0.10	0.25	0.27	0.26	0.33	0.14	0.27	0.30	0.15	0.23	0.25	0.28	0.25	0.24
AUD	0.30	0.11	0.13	0.29	0.79	0.55	0.45	0.30	0.14	0.30	0.35	0.37	0.52	0.23	0.64	0.43	0.26	0.34	0.29	0.51	0.32	0.35
BRL	0.46	0.17	0.14	0.34	0.55	1.51	0.52	0.46	0.15	0.46	0.50	0.48	0.80	0.25	0.48	0.57	0.27	0.43	0.46	0.60	0.46	0.45
CAD	0.39	0.17	0.11	0.27	0.45	0.52	0.58	0.39	0.12	0.39	0.42	0.41	0.55	0.19	0.40	0.44	0.22	0.36	0.39	0.42	0.39	0.38
CNY	0.48	0.26	0.08	0.25	0.30	0.46	0.39	0.49	0.09	0.48	0.46	0.46	0.53	0.12	0.28	0.45	0.13	0.39	0.48	0.31	0.45	0.43
DKK	0.09	0.02	0.08	0.10	0.14	0.15	0.12	0.09	0.11	0.09	0.10	0.10	0.14	0.10	0.13	0.11	0.11	0.09	0.09	0.14	0.09	0.09
HKD	0.48	0.26	0.08	0.25	0.30	0.46	0.39	0.48	0.09	0.48	0.46	0.46	0.53	0.11	0.28	0.45	0.13	0.39	0.48	0.31	0.45	0.43
INR	0.46	0.24	0.09	0.27	0.35	0.50	0.42	0.46	0.10	0.46	0.60	0.46	0.55	0.15	0.32	0.47	0.16	0.40	0.46	0.35	0.45	0.44
MYR	0.46	0.28	0.09	0.26	0.37	0.48	0.41	0.46	0.10	0.46	0.46	0.76	0.55	0.14	0.35	0.51	0.16	0.48	0.46	0.36	0.47	0.58
MXN	0.54	0.24	0.13	0.33	0.52	0.80	0.55	0.53	0.14	0.53	0.55	0.55	1.37	0.22	0.46	0.60	0.25	0.47	0.53	0.55	0.52	0.52
NOK	0.11	0.02	0.09	0.14	0.23	0.25	0.19	0.12	0.10	0.11	0.15	0.14	0.22	0.30	0.21	0.17	0.20	0.13	0.12	0.23	0.13	0.13
NZD	0.28	0.12	0.12	0.27	0.64	0.48	0.40	0.28	0.13	0.28	0.32	0.35	0.46	0.21	0.80	0.38	0.23	0.33	0.27	0.45	0.29	0.34
KRW	0.45	0.23	0.10	0.30	0.43	0.57	0.44	0.45	0.11	0.45	0.47	0.51	0.60	0.17	0.38	1.39	0.20	0.43	0.44	0.44	0.51	0.47
SEK	0.13	0.03	0.10	0.15	0.26	0.27	0.22	0.13	0.11	0.13	0.16	0.16	0.25	0.20	0.23	0.20	0.33	0.15	0.13	0.26	0.14	0.14
SGD	0.39	0.28	0.08	0.23	0.34	0.43	0.36	0.39	0.09	0.39	0.40	0.48	0.47	0.13	0.33	0.43	0.15	0.45	0.39	0.33	0.40	0.47
LKR	0.48	0.27	0.08	0.25	0.29	0.46	0.39	0.48	0.09	0.48	0.46	0.46	0.53	0.12	0.27	0.44	0.13	0.39	0.63	0.30	0.45	0.43
ZAR	0.31	0.10	0.13	0.28	0.51	0.60	0.42	0.31	0.14	0.31	0.35	0.36	0.55	0.23	0.45	0.44	0.26	0.33	0.30	1.20	0.32	0.34
TWD	0.45	0.26	0.08	0.25	0.32	0.46	0.39	0.45	0.09	0.45	0.45	0.47	0.52	0.13	0.29	0.51	0.14	0.40	0.45	0.32	0.54	0.44
THB	0.43	0.31	0.08	0.24	0.35	0.45	0.38	0.43	0.09	0.43	0.44	0.58	0.52	0.13	0.34	0.47	0.14	0.47	0.43	0.34	0.44	0.96

- (b) ANSWER:

<sup>1</sup>This point is actually more subtle than it seems. Most financial time series, these included, have tails which are too heavy to be Gaussian. These can still arise from situations *like* that of the central limit theorem, but need either dependence in the increments over time, or a randomly-fluctuating variance for the increments, or a constant but infinite variance. (That last is unlikely.) Relying on Gaussian distributions for financial time series was one of the things which brought Long Term Capital Management down in the late 1990s, and the global economy very nearly with it.

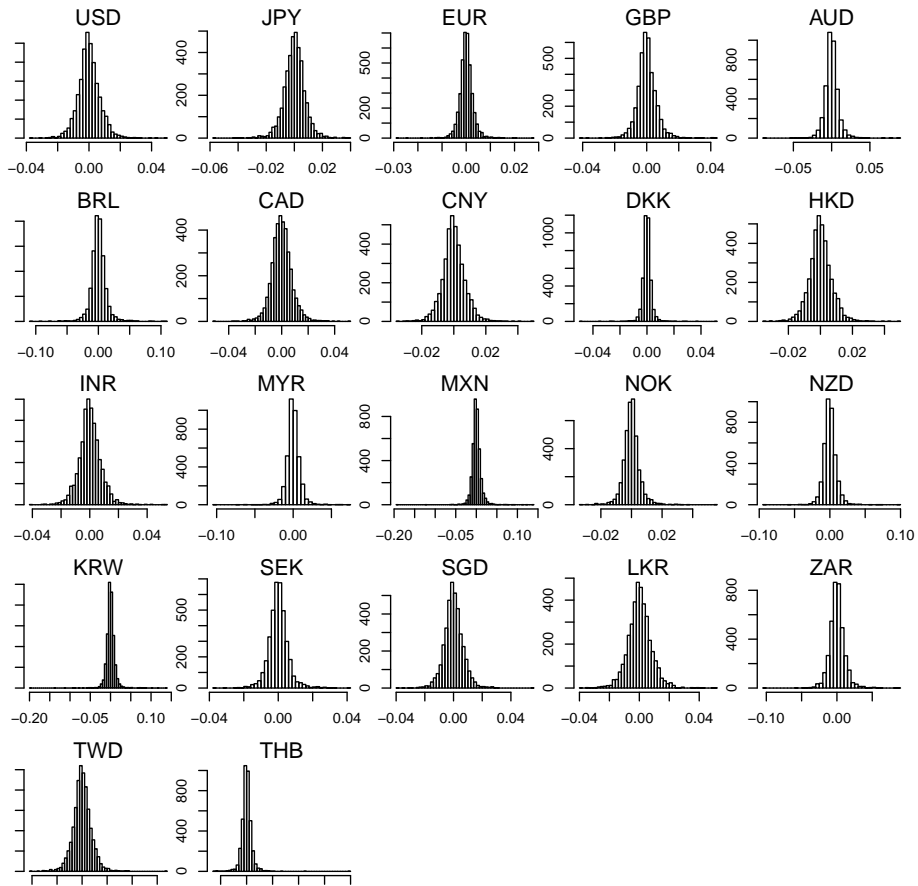


Figure 3: Histograms of logarithmic rates of change for the 22 currencies.

```

image(covariances, xaxt = "n", yaxt = "n")
par(cex = 0.5)
axis(1, at = seq(0, 1, by = 1/21), labels = colnames(fx.diff.log), lwd = 0.3)
axis(2, at = seq(0, 1, by = 1/21), labels = colnames(fx.diff.log), lwd = 0.3)

```

See Figure 4 for a visualization of the covariances.

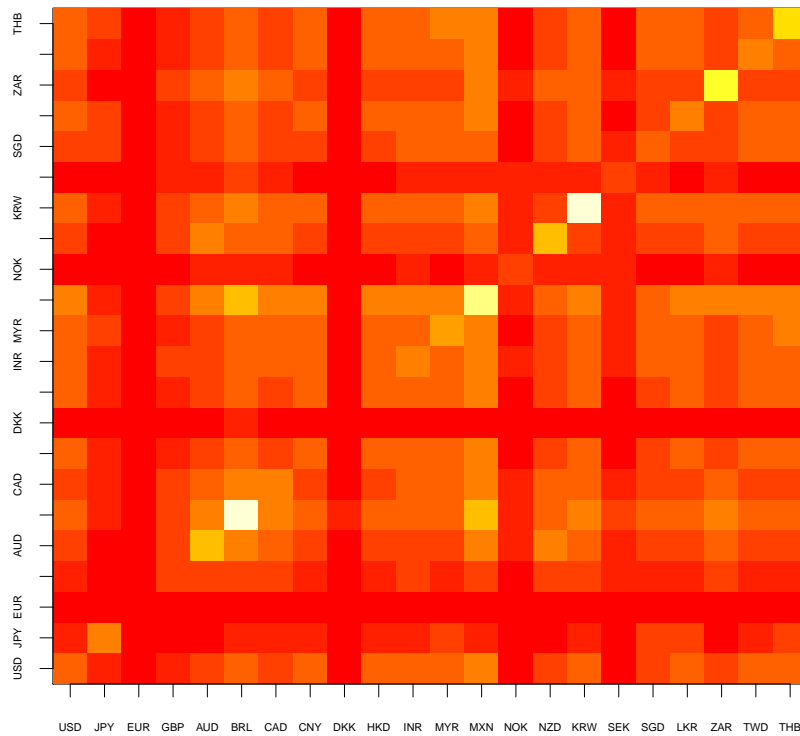


Figure 4: Heatmap showing covariance between log changes for different currencies. Dark red indicates the lowest levels of covariance, brighter and more yellow to white indicate higher covariance.

- (c) ANSWER: As expected, some of the diagonal entries have high covariance, reflecting the correlation between each variable and itself. Currencies strongly correlated with the USD include BRL, CNY, HKD, INR, MYR, MXN, and KRW. There could also be a strong correlation between MXN and BRL.

4. (a) ANSWER:

```
fx.pca = prcomp(t(fx.diff.log), scale=FALSE)
```

Notice that the principal components are computed *after* centering. For instance, the daily logarithmic change for all currencies on 28 February 2003 was

```
> mean(fx.diff.log["2003-02-28",])
[1] 0.004122745
```

so we subtract this from all the time series on that date, before making the covariance matrix, finding the eigenvectors, projecting the data on to the eigenvectors, etc. (Or rather, R does this for us.) We do *not* scale the variables, because we have already transformed them so that they are comparable to each other — the fact that some days have more variance in exchange rates than others is fine, in fact it's interesting and economically meaningful.

Because there are fewer cases (22 currencies) than features (3778 days), we have only 22 principal components.

- (b) ANSWER: Add up the variances associated with each eigenvector, and divide by the total variance;

```
cumulative.pct = cumsum(fx.pca$sdev^2)/sum(fx.pca$sdev^2)
```

The `cumsum` function calculates the cumulative sums of the vector you give it as an argument. That is, `cumsum(x)` returns a vector as long as `x`, whose entries are `x[1]`, `x[1]+x[2]`, and so forth. If you don't know about `cumsum`, you can achieve the same effect with a `for` loop.

Plot the cumulative variances:

```
plot(1:22, cumulative.pct, type = "l", xlab = "Number of components",
     ylab = "Fraction of variance retained by principal components")
points(1:22, cumulative.pct)
abline(h = c(0.5,0.9), lty = 2, col = 2)
```

Figure 5 shows that 50% of the variance is kept with 3 components, and 90% is kept when including 13 components.

- (c) ANSWER:

```
plot(fx.pca$x[,1:2], type="n",
     main = "Projections of countries onto the first two principal components")
text(fx.pca$x[,1:2], labels = colnames(fx), color=1:22)
```

See Figure 6.

- (d) ANSWER: There are several clusters. The Scandinavian countries (NOK, SEK, DKK) are all close to each other, high on both components, and very close to the Euro (EUR). New Zealand and Australia are close to each other and not too far from northern Europe. Apart

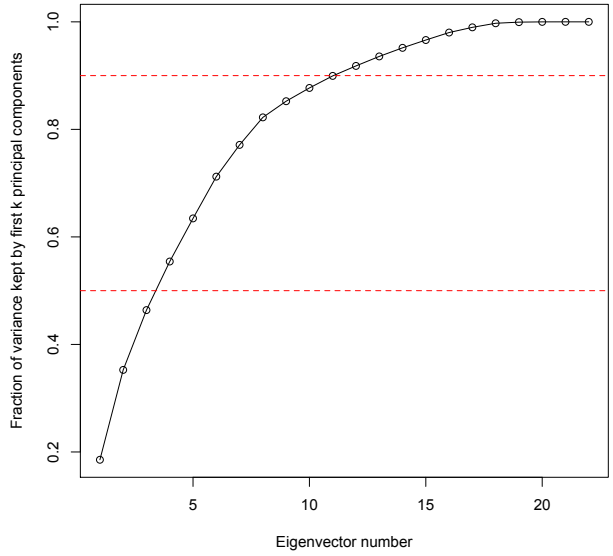


Figure 5: Fraction of variance retained by the first  $q$  principal components.

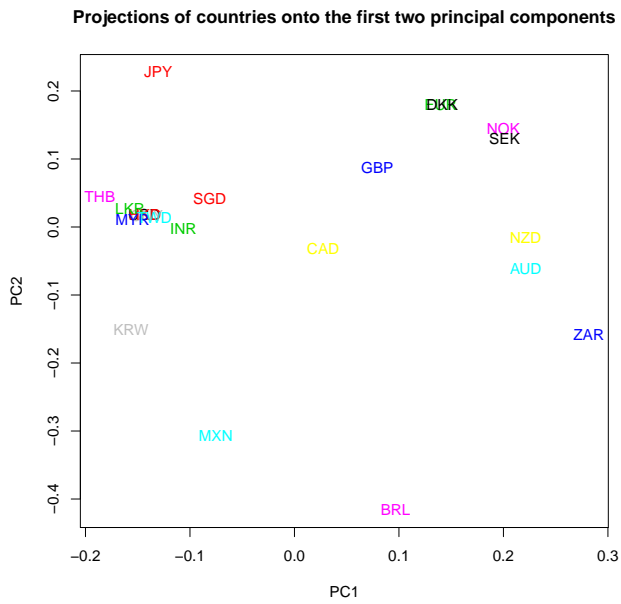


Figure 6: Projection of the currencies on to the first two principal components.

from Australia and New Zealand, countries on the Pacific tend to have negative values on the first principal component, and there is a very tight cluster of currencies with negative values on the first principal component and near-zero values on the second which includes the United States, China, Malaysia, Thailand, Taiwan, and Sri Lanka. This group consists basically of the US and the developing Asian economies. India and Singapore are nearby but slightly apart, and Korea and Japan, while similar in their score on the first principal component, are much further away in the second component. (Japan, Korea and Singapore are, of course, developed countries.) Mexico does not have quite so extreme a value on the first principal component, and is even further away on the second.

A reasonable interpretation for the first principal component is that it reflects how much a country is bound up in trade across the Pacific, especially with the United States. The second component is not quite so transparent from the plot.

In general, the covariance matrix does not match well with this projection plot. However, this should not be terribly surprising, since countries will tend to be close to each other here if they have time series which are *close* to each other, while the covariance plot looks at whether they can be *predicted* from each other.

5. The principal components here are *time series*, of logarithmic changes in exchange rates. We can think of these as the basic templates out of which the individual currencies' histories are composed.

- (a) ANSWER: The following code does the plotting, putting a tick mark on the axis every 100 days, with the accompanying dates (turned on their side and slightly shrunk so we can read them).

```
plot(1:nrow(fx.diff.log), fx.pca$rotation[,1], type = "l",
     xaxt = "n", xlab = "Date", ylab = "PC1")
axis.dates = seq(from=1,to=length(fx.diff.log),by=100)
axis(1, at=axis.dates,labels=rownames(fx.diff.log)[axis.dates],
     las=2,cex.axis=0.4)
```

See Figure 7.

- (b) ANSWER: There are spikes of volatility around 1997 (the Asian financial crisis), and of course late 2008 (the collapse of the world economy into the Great Recession).
- (c) ANSWER: See Figure 8.

```
plot(1:nrow(fx.diff.log), fx.pca$rotation[,2], type = "l",
     xaxt = "n", xlab = "Date", ylab = "PC2")
axis(1, at=axis.dates,labels=rownames(fx.diff.log)[axis.dates],
     las=2,cex.axis=0.4)
```

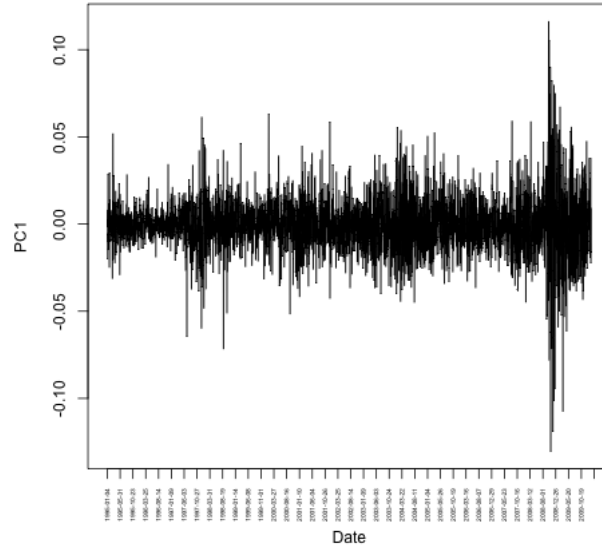


Figure 7: First principal component.

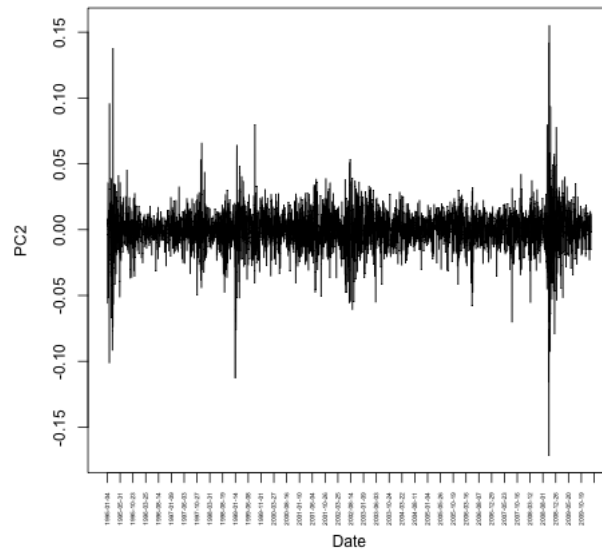


Figure 8: Second principal component.

While there is enhanced activity around 1997 and of course 2008 again, the most notable thing here is the huge activity in early 1995. This is the “peso crisis” which began in December 1994 with the Mexican peso, but went on to effect multiple other currencies in Central and South America, including that of Brazil. This suggests an interpretation for the second principal component, namely that negative values correspond to countries which got caught up in that crisis.

6. *General remarks about PCA and geometry:* A direction in space is represented by a unit-length vector  $\vec{w}$ ; this marks out a line through the origin. The length of a vector  $\vec{X}$ 's projection on to tht line is  $\vec{X} \cdot \vec{w}$ . The projection itself is a new vector,  $(\vec{X} \cdot \vec{w})\vec{w}$ . The projection of a vector  $\vec{X}$  on to a plane spanned by  $\vec{w}_1$  and  $\vec{w}_2$  is

$$(\vec{X} \cdot \vec{w}_1)\vec{w}_1 + (\vec{X} \cdot \vec{w}_2)\vec{w}_2 \quad (1)$$

assuming that  $\vec{w}_1$  and  $\vec{w}_2$  are orthogonal to each other and both have unit length. This pattern continues for projecting on to  $q$  orthogonal vectors of unit length:

$$\sum_{j=1}^q (\vec{X} \cdot \vec{w}_j)\vec{w}_j \quad (2)$$

In the case of PCA, the orthogonal length-one vectors are the eigenvectors of the covariance matrix, i.e., the principal components themselves. In the output of `prcomp`, these are stored in the `rot` attribute. The inner products in the formula above, *lengths* of each data vector's projections on to the components, are stored in the `x` attribute.

The one subtlety is that `prcomp` centered all of the data vectors before finding the principal components and before taking the inner products. That is, it really deals with  $\vec{X} - \vec{\mu}$ , where  $\vec{\mu}$  is the vector of feature means. Geometrically, this means that the principal components mark out lines through the center of the data, not through the origin. We need to add those means back in to make the projections comparable to the original data vectors, i.e., we need to work with

$$\vec{\mu} + \sum_{j=1}^q ((\vec{X} - \vec{\mu}) \cdot \vec{w}_j)\vec{w}_j \quad (3)$$

Once again, the necessary inner products are already computed for us by `prcomp`, and stored in the `x` component of what it returns.

- (a) ANSWER: We plot the original USD series:

```
plot(1:nrow(fx.diff.log), fx.diff.log["USD"], type = "l",
     xaxt = "n", xlab = "Date", ylab = "log change",
     main = "USD log changes")
axis(1, at=axis.dates, labels=rownames(fx.diff.log)[axis.dates],
     las=2, cex.axis=0.4)
```

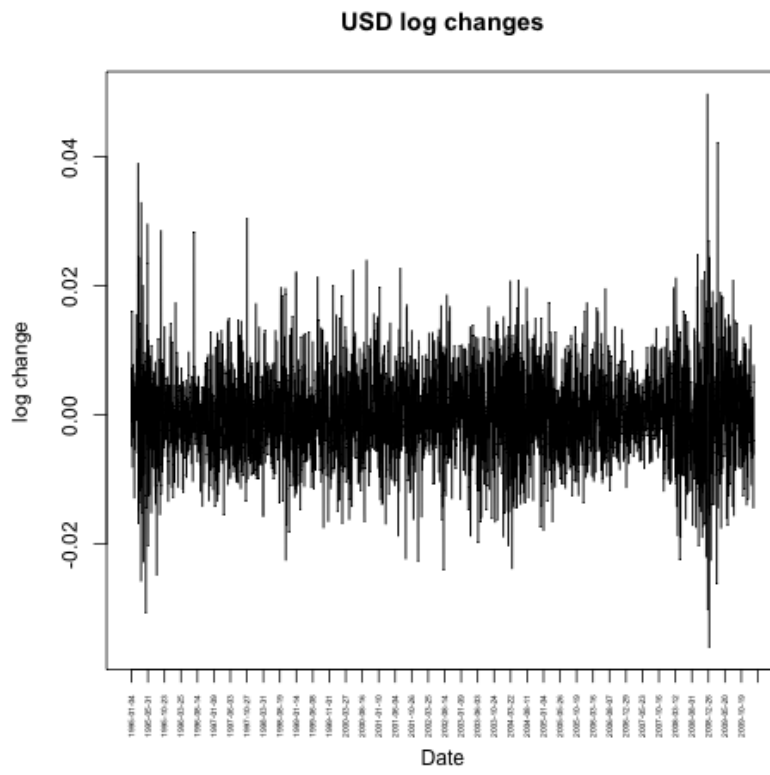


Figure 9:

See Figure 9.

- (b) ANSWER: We write a function to implement Eq. 3 above.

```
get.reconstruction = function(v,q, pca=fx.pca,center=pca$center) {  
  Wq = pca$rotation[,1:q] # This is a p*q matrix  
  inner.products = as.matrix(pca$x[v,1:q],nrow=q) # This is a q*1 matrix  
  centered.projection = Wq %*% inner.products # Multiplies (p*q) by (q*1)  
  return(center+centered.projection)  
}
```

Here  $v$  should be the number of the name of the data vector we want to reconstruct. A function which could apply to an arbitrary vector would be:

```
reconstruct.vector = function(x,q,pca=fx.pca,center=pca$center) {  
  x.0 <- as.matrix(x - center,nrow=length(x)) # A p*1 matrix  
  Wq = pca$rotation[,1:q] # This is a p*q matrix  
  inner.products = t(t(x.0) %*% Wq) # Gives a q*1 matrix  
  centered.projection = Wq %*% inner.products # Multiplies (p*q) by (q*1)  
  return(center+centered.projection)  
}
```

See Figure 10 for the result of plotting `get.reconstruction("USD",q=3)`.

(You can verify that this is the same as plotting `reconstruct.vector(fx.diff.log[, "USD"],q=3)`.)

The over-all pattern is extremely similar to the original time series, but somewhat less volatile (which makes since the first three components only retain a bit less than half the variance).

Notice that to simplify the visual comparison of the reconstructed time series for the US dollar against the reality, the figure also plots the original data first, in grey, and then plots the reconstruction on top of that.

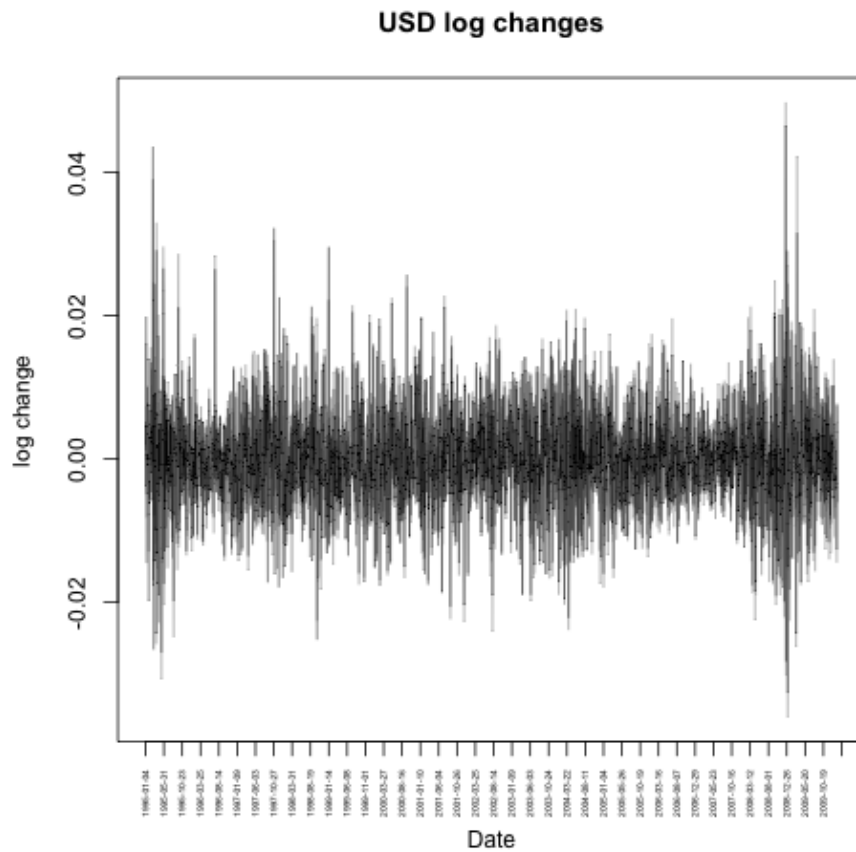
- (c) ANSWER: Rather than calculating MSEs by hand 22 times, write a function

```
mse.USD.q = function(q) {  
  mse = mean((fx.diff.log[, "USD"] - get.reconstruction("USD",q))^2)  
  return(mse)  
}
```

and apply it:

```
mse.USD = sapply(1:22,mse.USD.q)
```

and then plot it (Figure 11). To give a sense of scale, if we approximated the USD time series by the global mean vector  $\mu$ , i.e., set  $q = 0$ , we'd get an MSE of  $9.96 \times 10^{-6}$ , more than double the MSE with just  $q = 1$ .

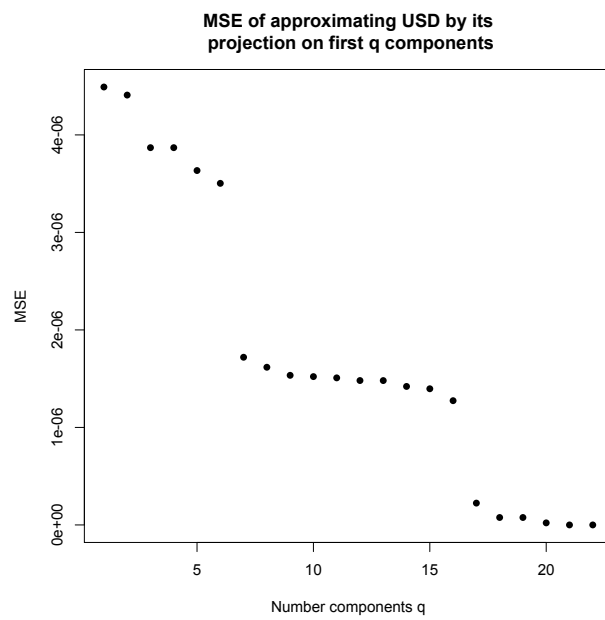


```

plot(1:nrow(fx.diff.log), fx.diff.log[,"USD"], type = "l",
     xaxt = "n", xlab = "Date", ylab = "log change",
     main = "USD log changes", col="grey")
axis(1, at=axis.dates, labels=rownames(fx.diff.log)[axis.dates],
     las=2, cex.axis=0.4)
lines(1:nrow(fx.diff.log), get.reconstruction("USD", 3), lwd=0.5)

```

Figure 10: Actual time series for USD (grey, background) and the reconstruction from its projection on to the first three principal components (black).



```
plot(1:22, mse.USD, pch = 16,  
     ylab = "MSE", xlab = "Number components q",  
     main = "MSE of approximating USD by its\n projection on first q components")
```

Figure 11: