STAT-S675

Homework 4

John Koo

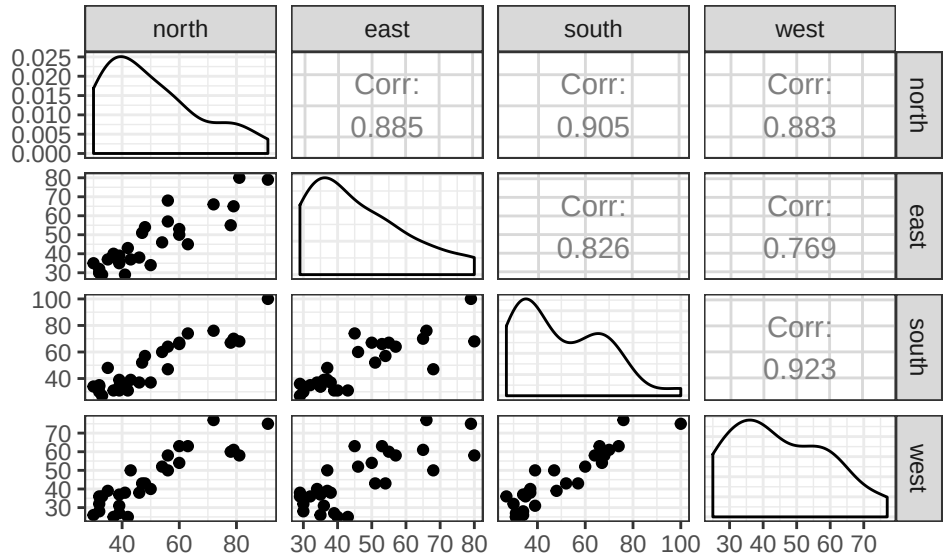
September 20, 2017

```
import::from(readr, read_table)
library(ggplot2)
import::from(GGally, ggpairs)
theme_set(theme_bw())
import::from(magrittr, `%>%`)
import::from(ggrepel, geom_label_repel)
```

Problem 1

```
cork.df <- read_table('http://pages.iu.edu/~mtrosset/Courses/675/mkb141.dat',
                      col_names = c('north', 'east', 'south', 'west'))

ggpairs(cork.df)
```

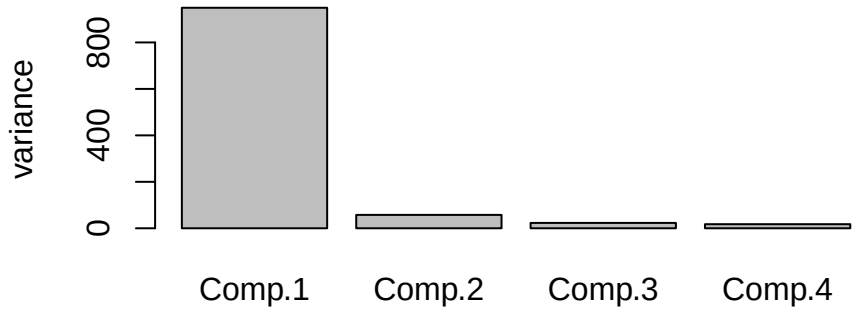


The first thing we can notice is that all of these columns are highly correlated with each other. So we expect the first principal component to capture almost all of the variability.

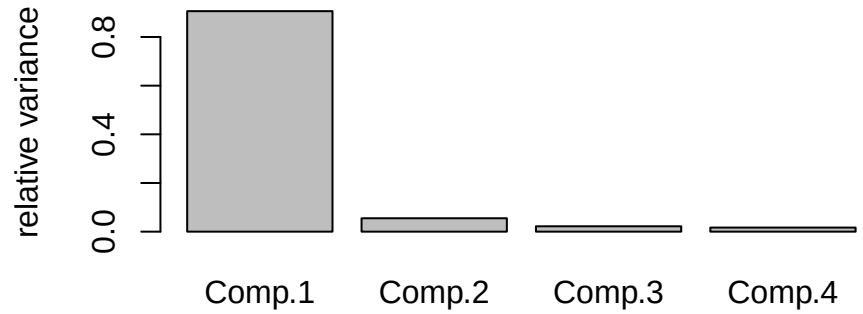
```
cork.pca <- princomp(cork.df)
```

Here we want to use the covariance matrix (which it does by default) since all of the measurements are comparable to each other (i.e., same units, same type of measurement).

```
barplot(cork.pca$sdev ** 2,
        ylab = 'variance')
```




```
barplot(cork.pca$sdev ** 2 / sum(cork.pca$sdev ** 2),
        ylab = 'relative variance')
```




```
cork.pca$sdev[1] ** 2 / sum(cork.pca$sdev ** 2)
```

```
   Comp.1
0.9061975
```

The first principal component captures >90% of the variance.

```
cork.pca$loadings
```

```
Loadings:
      Comp.1 Comp.2 Comp.3 Comp.4
north -0.526  0.225  0.678 -0.462
east  -0.429  0.752 -0.320  0.385
south -0.579 -0.379 -0.599 -0.403
west  -0.452 -0.490  0.282  0.690

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
```

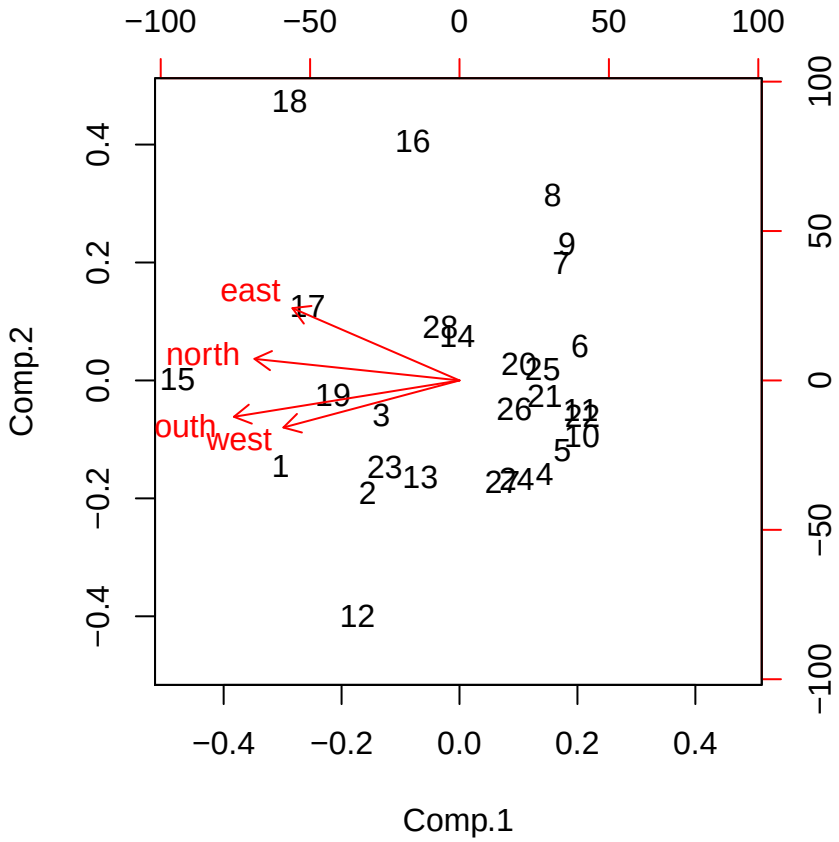
Here we can see the actual components. One thing to note is that the signs are arbitrary as long as they're consistent (e.g., all positive is equivalent to all negative). An interpretation of this might be something like:

- The first component has the same sign and roughly similar magnitudes for all of north, east, south, and west, so we can interpret it as the overall magnitude of all of the variables. This is consistent with our observation that all of the columns are highly correlated with each other.
- The second component has positive values for north and east while having negative values for south and west. This can be interpreted as measuring how different north and east are from south and west.
- The third and fourth have similar interpretations as the second. The third measures how different north and west are from east and south, while the fourth measures how different north and south are from

east and west.

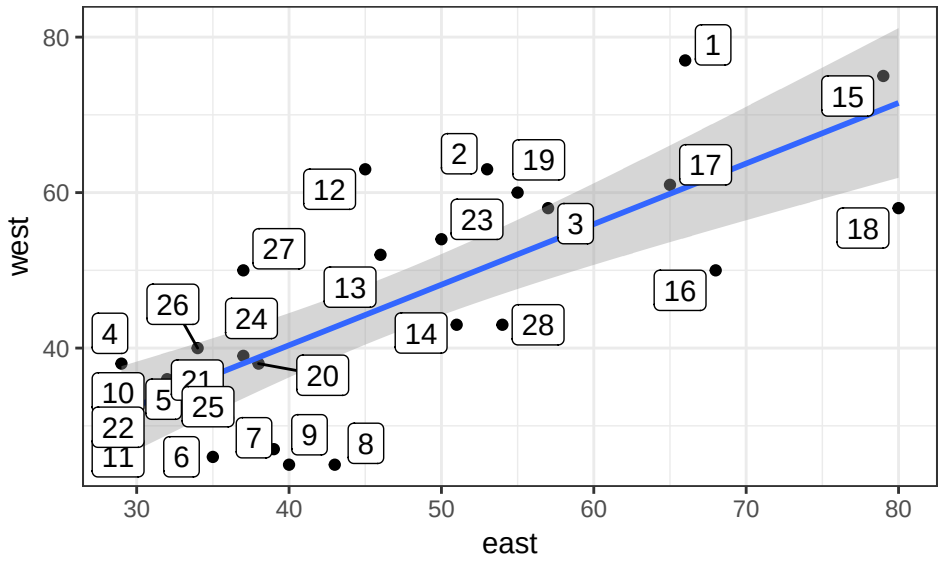
- One quick observation is that the second principal component measures north vs south and east vs west, which agrees somewhat with our intuition. The sun is more to the north or to the south depending on the season and hemisphere, so it's reasonable to hypothesize that there is some natural difference there. Similarly, the sun rises in the east and sets in the west, and it's reasonable to hypothesize that there would be some difference there as well.

```
biplot(cork.pca)
```



From the biplot, we can see that the angles between each pair of columns is very close, implying strong correlation. The two columns with the largest angle between them are east and west. This is consistent with our observation from the matrix scatterplot showing that these two columns are the least correlated.

```
cork.df %>%
  dplyr::mutate(row = rownames(.)) %>%
  ggplot() +
  geom_point(aes(x = east, y = west)) +
  stat_smooth(aes(x = east, y = west), method = 'lm') +
  geom_label_repel(aes(x = east, y = west, label = row))
```



From eyeballing the biplot, we can see how far off from a linear fit of east vs west each point is. For example, the 8th point in the biplot has a very small yet positive "east" component while it has a very negative "west" component, and in the scatterplot of west vs east the 8th point is very far from the linear fit line. On the other hand, the 15th point has relatively similar values for east and west in the biplot and in the scatterplot it lies very close to the linear fit line.

Problem 2

Let $q = 30000$. The following R commands generate a random $q \times 10$ matrix:

```
set.seed(675)
q <- 30000
l <- 10
Q <- matrix(rnorm(q*l),ncol=l)/sqrt(q)
```

Each column of `Q` is a vector in $\mathbb{R}^q$.

Investigate how nearly orthonormal these vectors turn out to be.

Two vectors are orthonormal iff they are unit vectors (i.e., length 1) and they are orthogonal (i.e., their cross product is 0).

Vector length

We can compute the length of each of these vectors:

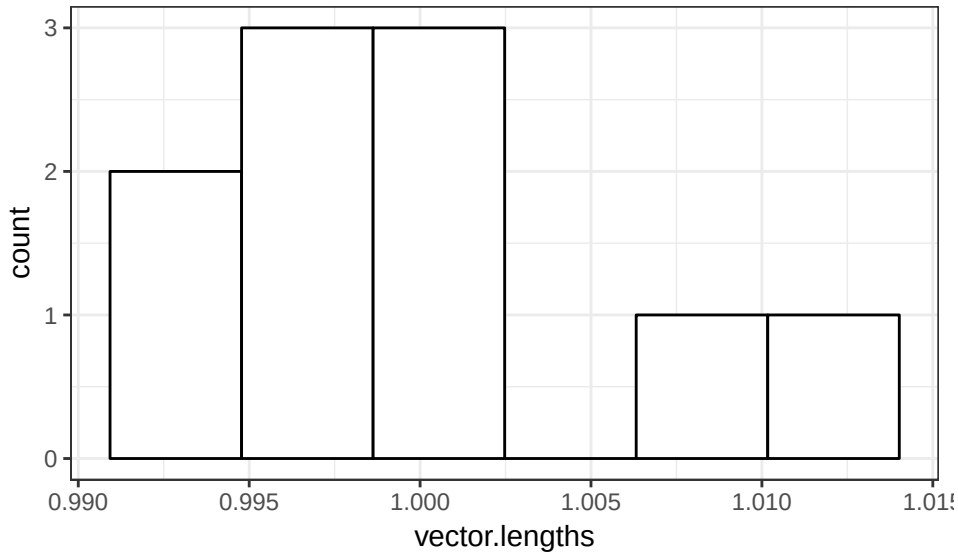
```
vector.lengths <- sqrt(colSums(Q ** 2))
sort(vector.lengths)
```

```
 [1] 0.9917474 0.9943254 0.9954685 0.9978324 0.9984334 0.9999294 1.0003004
 [8] 1.0015163 1.0077176 1.0109888
```

```
summary(vector.lengths)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9917  0.9961  0.9992  0.9998  1.0012  1.0110
```

```
ggplot() +
  geom_histogram(aes(x = vector.lengths),
                 bins = 6, colour = 'black', fill = 'white')
```



Here we can see that all of the lengths are very close to 1. This is expected since we know that for each vector, the elements are normally distributed with $\mu = 0$ and $\sigma = \frac{1}{\sqrt{q}}$, so $E[|x_i|_2] = E[\sqrt{\sum_j x_{ij}^2}] = \sqrt{q\sigma^2} = 1$ (using some very loose, hand-wavy logic). So on average, the vectors will be of length 1.

Orthogonality

Note that $[Q^TQ]_{ij} = \sum_k Q_{ki}Q_{kj} =< Q_{\cdot,i}, Q_{\cdot,j} >$

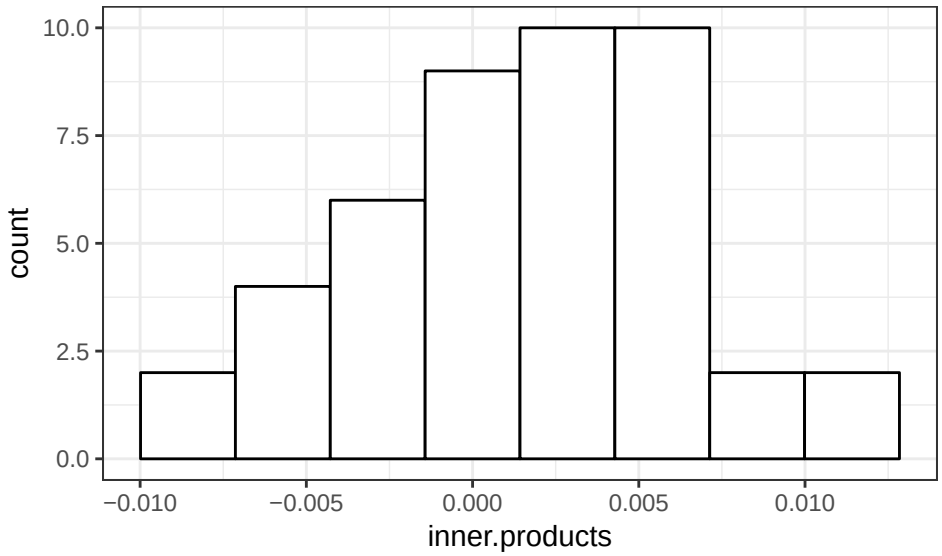
So we can just compute Q^TQ and look at either the upper or lower triangular terms.

```
QTQ <- t(Q) %*% Q

inner.products <- QTQ[lower.tri(QTQ)]
summary(inner.products)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.009122 -0.001516  0.001848  0.001478  0.005461  0.010852
```

```
ggplot() +
  geom_histogram(aes(x = inner.products),
                 bins = 8, colour = 'black', fill = 'white')
```



Here we can see that the inner products are all close to 0. Using similar hand-wavy logic as before, we can say something like:

$$E[< Q_{\cdot,i}, Q_{\cdot,j} >] = E[\sum_k Q_{ki}Q_{kj}] \sim q \times cov(Q_{\cdot,i}, Q_{\cdot,j}) = 0$$

since the columns of Q are independent.

Problem 3

Download the 201×2 data matrix `X.spiral` and scan it into R. Use the R Package Installer to download and install the package `pracma`. Let `q = 30000`. The following R commands construct an isometrically equivalent configuration of $n = 201$ points in $\mathbb{R}^q$:

```
X.spiral <- read.table('http://pages.iu.edu/~mtrosset/Courses/675/X.spiral') %>%
  as.matrix()
library(pracma)
q <- 30000
A <- matrix(rnorm(q*2),ncol=2)
A.QR <- gramSchmidt(A)
Y <- X.spiral %*% t(A.QR$Q)
```

Part a

Vectors containing the $n(n-1)/2$ pairwise Euclidean distances of each configuration can be computed using the `dist` function. Verify that the configurations `Y` and `X.spiral` are isometrically equivalent by showing that they have the same pairwise distances, e.g., by showing that the *root mean squared error* (RMSE) vanishes:

```
dX <- dist(X.spiral)
dY <- dist(Y)
m <- length(dX)
rmse <- sqrt(sum((dY-dX)^2)/m)
```

```
# compare to:
.Machine$double.eps
```

```
[1] 2.220446e-16
```

The RMSE here is 6.881×10^{-15}.

Part b

Investigate the extent of distortion introduced by the following random projection technique, which replaces the $n \times q$ data matrix `Y` with the $n \times 10$ data matrix `Y10`:

```
Q10 <- matrix(rnorm(q*10),ncol=10)/sqrt(q)
Y10 <- Y %*% Q10
```

Find the distance matrix of `Y10` and compute its RMSE to the distance matrix of `Y`:

```
dY10 <- dist(Y10)

sqrt(sum((dY - dY10) ** 2) / m)
```

```
[1] 2.830612
```

Here the RMSE is quite large, especially when compared to the typical spread of the values of the distance matrix of Y:

```
sqrt(sum((dY - dY10) ** 2) / m) / sd(dY)
```

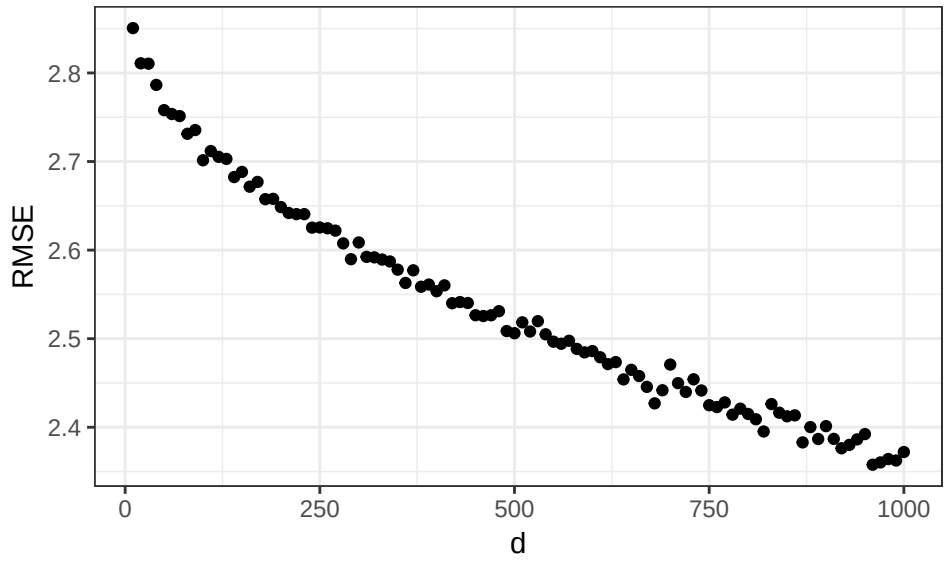
```
[1] 2.282281
```

So the RMSE is twice the standard deviation of the distances in the original Y.

We can also see how this changes as the number of random projections changes:

```
d <- seq(10, 1000, 10)
rmse.vector <- sapply(d, function(x) {
  Qd <- matrix(rnorm(q * x), ncol = x) / sqrt(q)
  Yd <- Y %*% Qd
  dYd <- dist(Yd)
  sqrt(sum((dY - dYd) ** 2) / m)
})

ggplot() +
  geom_point(aes(x = d, y = rmse.vector)) +
  labs(y = 'RMSE')
```



And as expected, the RMSE decreases as we increase the number of projections.