Jin Kweon (3032235207) HW2

Jin Kweon

9/18/2017

Name: Jin Kweon

Lab: 11am - 1pm (section 102) (course number: 20980)

Monthly temperatures (in Celsius) were collected for the main European capitals and other major cities. In addition to the monthly temperatures, the average annual temperature and t he thermal amplitude (difference between the maximum monthly average and the minimum monthly average of a city) were recorded for each city. The data set also includes two quantitative positioning variables (latitude and longitude) as well as one categorical variable: Area (with four categories north, south, east, and west).

Thus, the capital will be regarded as active individuals (the first 23 rows) while the other cities will be regarded as supplementary individuals (i.e. individuals which are not involved in the computation of the components).

Research Question. The main research question is: Can we summarize monthly precipitation with only a small number of components?

```r
temp <- read.csv("/Users/yjkweon24/Desktop/Cal/2017 Fall/Stat 154/data/temperature.csv",
                 stringsAsFactors = F, header = T)

dim(temp)
```

```
## [1] 35 18
```

```r
class(temp)
```

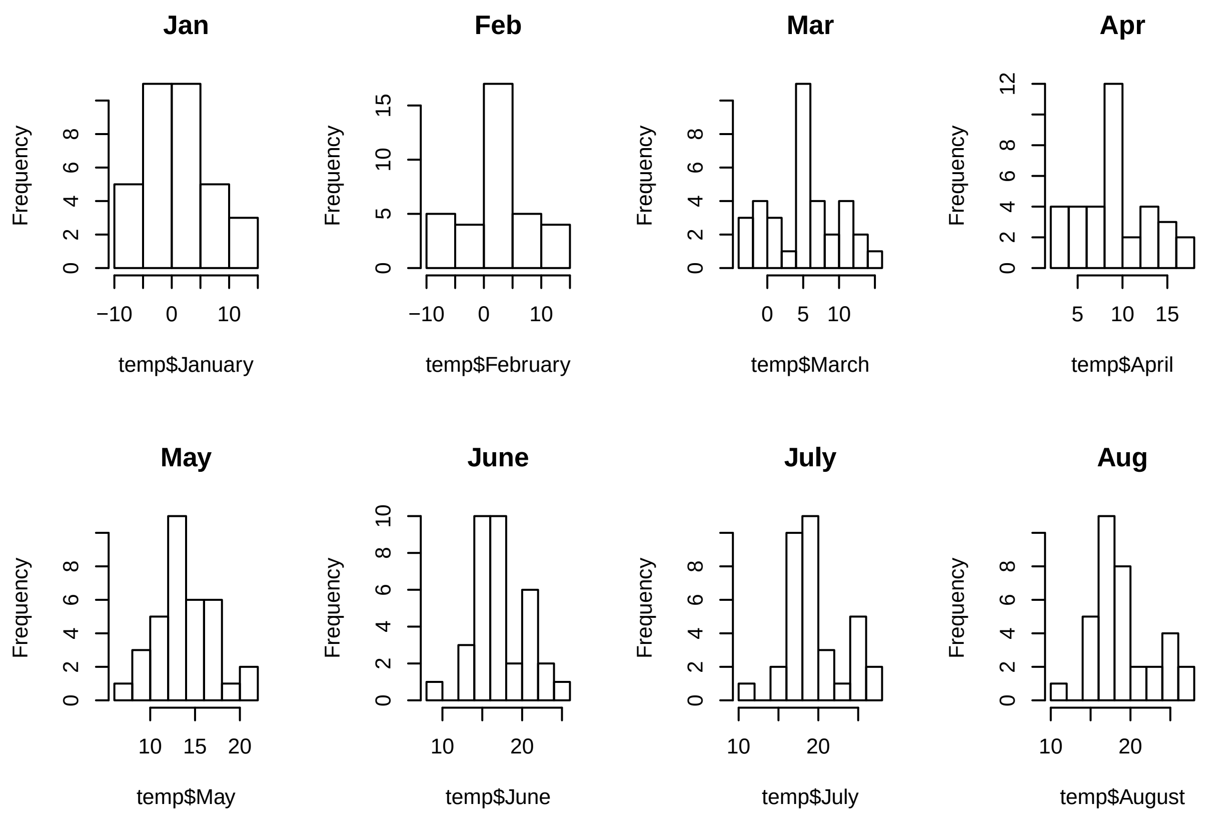
```
## [1] "data.frame"
```

```r
temp <- na.omit(temp)
```

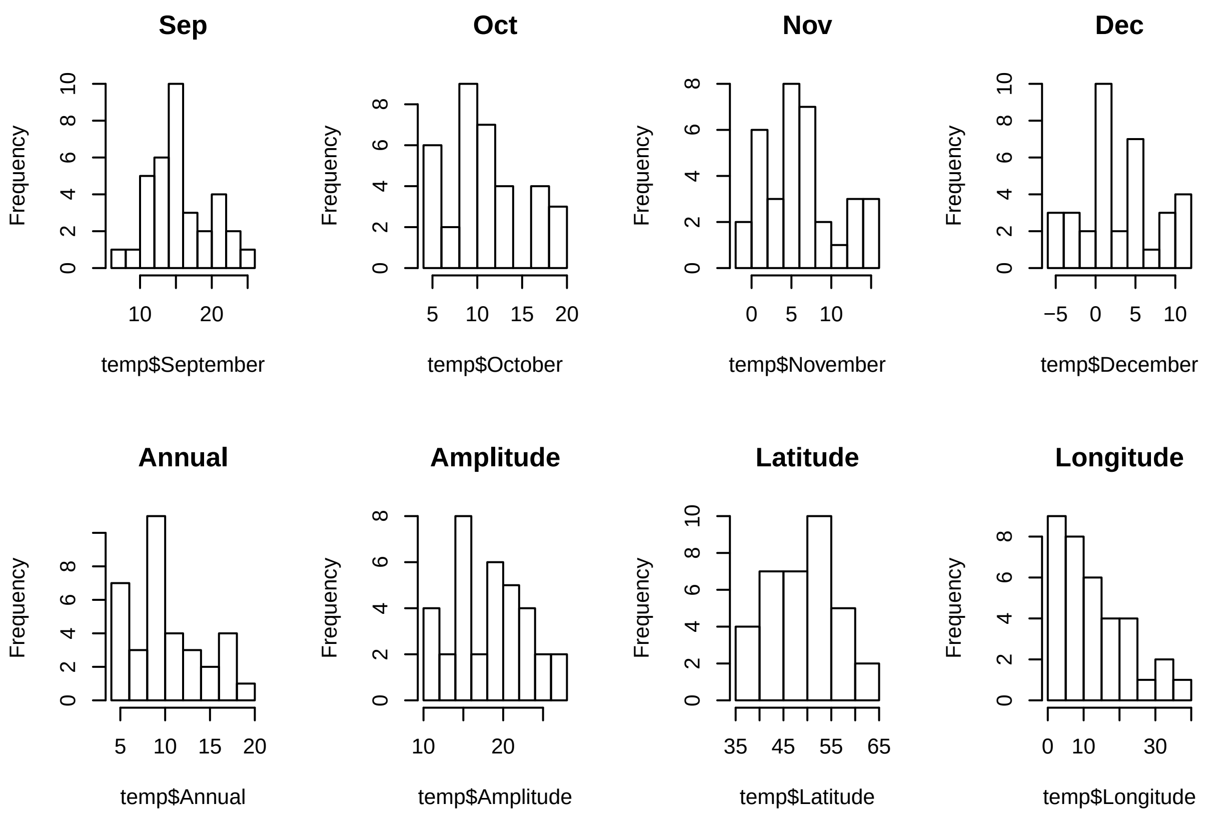
```r
summary(temp)
```

```
##       X                 January          February           March
##  Length:35          Min.   :-9.300   Min.   :-7.900   Min.   :-3.700
##  Class :character   1st Qu.:-1.550   1st Qu.:-0.150   1st Qu.: 1.600
##  Mode  :character   Median : 0.200   Median : 1.900   Median : 5.400
##                     Mean   : 1.346   Mean   : 2.217   Mean   : 5.229
##                     3rd Qu.: 4.900   3rd Qu.: 5.800   3rd Qu.: 8.500
##                     Max.   :10.700   Max.   :11.800   Max.   :14.100
##      April             May             June            July
##  Min.   : 2.900   Min.   : 6.50   Min.   : 9.30   Min.   :11.10
##  1st Qu.: 7.250   1st Qu.:12.15   1st Qu.:15.40   1st Qu.:17.30
##  Median : 8.900   Median :13.80   Median :16.90   Median :18.90
##  Mean   : 9.283   Mean   :13.91   Mean   :17.41   Mean   :19.62
##  3rd Qu.:12.050   3rd Qu.:16.35   3rd Qu.:19.80   3rd Qu.:21.75
##  Max.   :16.900   Max.   :20.90   Max.   :24.50   Max.   :27.40  
##      August        September        October         November     
##  Min.   :10.60   Min.   : 7.90   Min.   : 4.50   Min.   :-1.100  
##  1st Qu.:16.65   1st Qu.:13.00   1st Qu.: 8.65   1st Qu.: 3.200  
##  Median :18.30   Median :14.80   Median :10.20   Median : 5.100  
##  Mean   :18.98   Mean   :15.63   Mean   :11.00   Mean   : 6.066  
##  3rd Qu.:21.60   3rd Qu.:18.25   3rd Qu.:13.30   3rd Qu.: 7.900  
##  Max.   :27.20   Max.   :24.30   Max.   :19.40   Max.   :14.900  
##     December         Annual        Amplitude        Latitude    
##  Min.   :-6.00   Min.   : 4.50   Min.   :10.20   Min.   :37.20  
##  1st Qu.: 0.25   1st Qu.: 7.75   1st Qu.:14.90   1st Qu.:43.90  
##  Median : 1.70   Median : 9.70   Median :18.50   Median :50.00  
##  Mean   : 2.88   Mean   :10.27   Mean   :18.32   Mean   :49.04  
##  3rd Qu.: 5.40   3rd Qu.:12.65   3rd Qu.:21.45   3rd Qu.:53.35  
##  Max.   :12.00   Max.   :18.20   Max.   :27.60   Max.   :64.10  
##    Longitude         Area          
##  Min.   : 0.00   Length:35         
##  1st Qu.: 5.05   Class :character  
##  Median :10.50   Mode  :character  
##  Mean   :13.01                     
##  3rd Qu.:19.30                     
##  Max.   :37.60                     
```

```r
par(mfrow = c(2,4))

#histogram
hist(temp$January, main = "Jan")
hist(temp$February, main = "Feb")
hist(temp$March, main = "Mar")
hist(temp$April, main = "Apr")
hist(temp$May, main = "May")
hist(temp$June, main = "June")
hist(temp$July, main = "July")
hist(temp$August, main = "Aug")
```



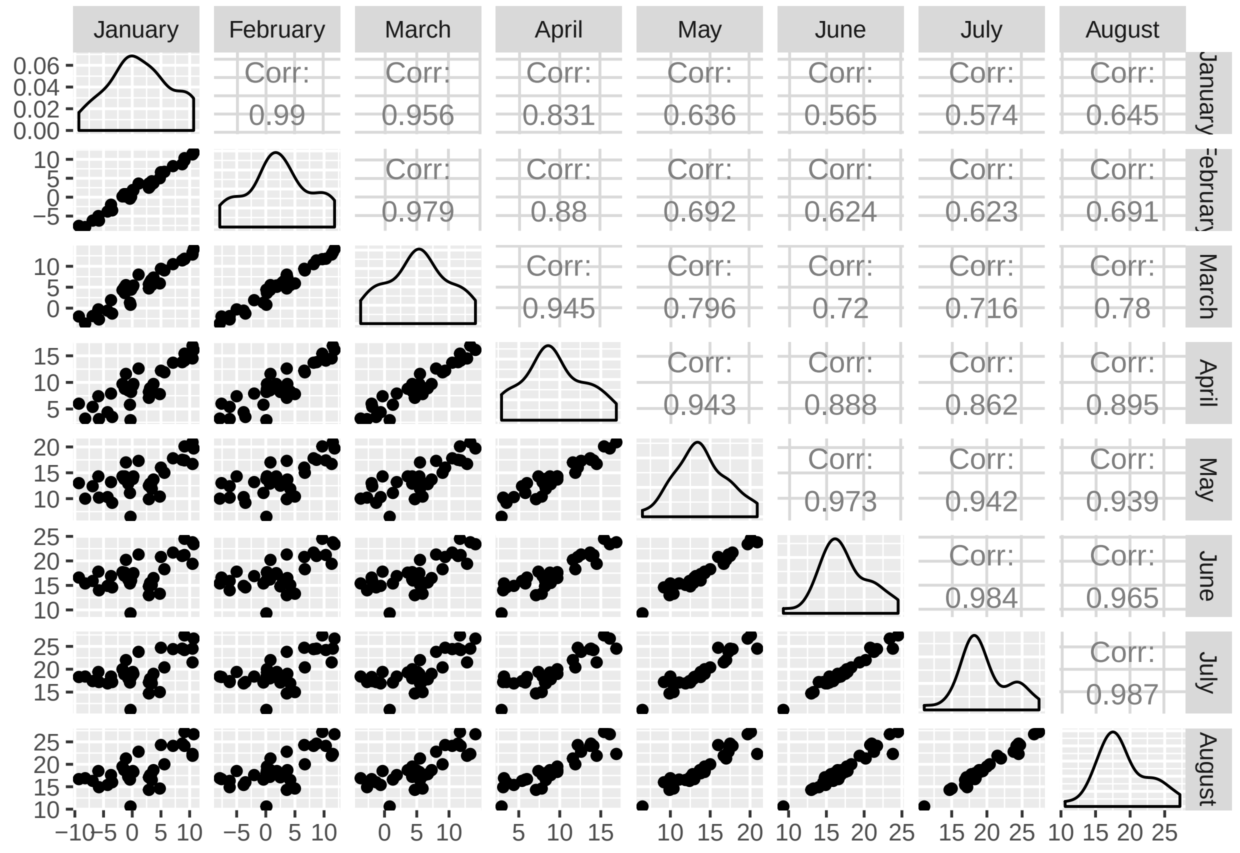
```r
hist(temp$September, main = "Sep")
hist(temp$October, main = "Oct")
hist(temp$November, main = "Nov")
hist(temp$December, main = "Dec")
hist(temp$Annual, main = "Annual")
hist(temp$Amplitude, main = "Amplitude")
hist(temp$Latitude, main = "Latitude")
hist(temp$Longitude, main = "Longitude")
```



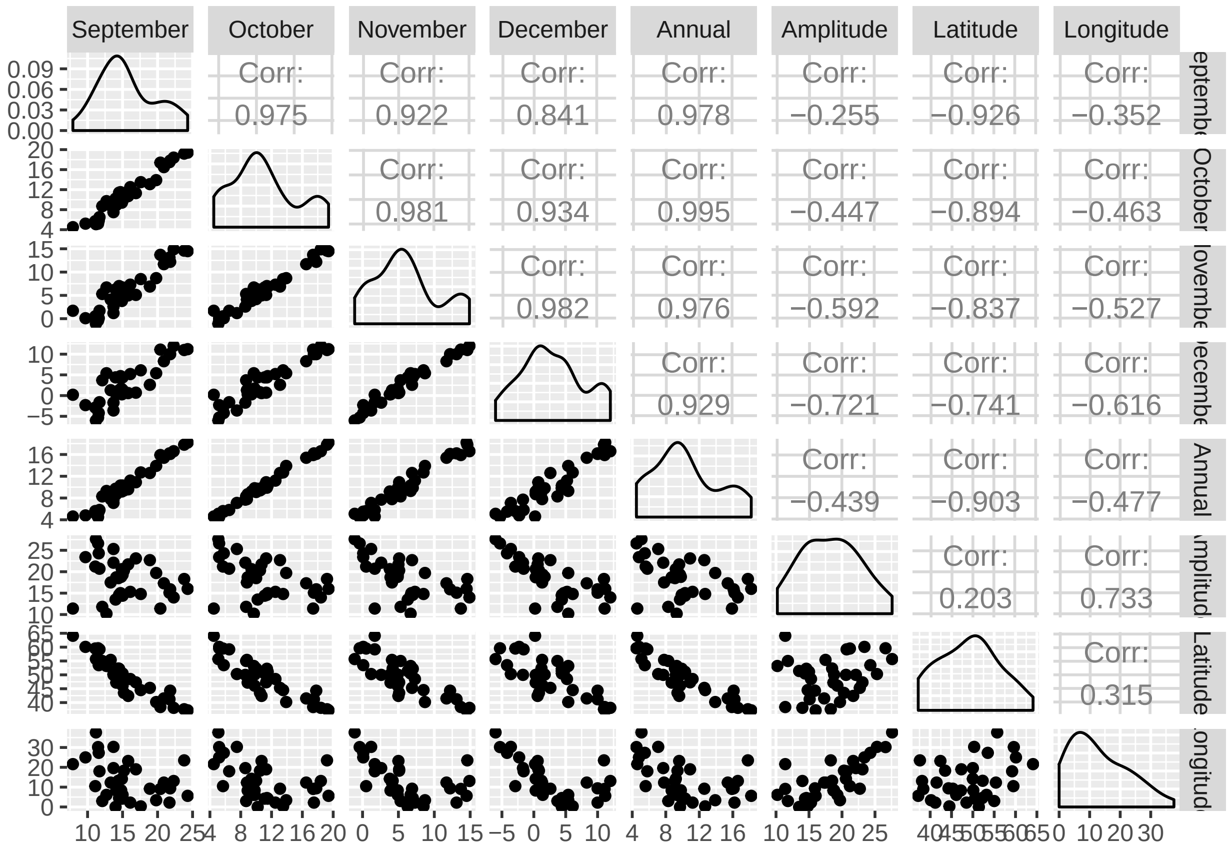
```r
par(mfrow = c(1,1))
#correlation matrix
#rcorr(as.matrix(temp[,2:17]))
cor(temp[,2:17])
```

```
##              January   February      March     April         May
## January    1.0000000  0.9900015  0.9558438 0.8312999  0.63578025
## February   0.9900015  1.0000000  0.9792534 0.8804197  0.69237412
## March      0.9558438  0.9792534  1.0000000 0.9454295  0.79591065
## April      0.8312999  0.8804197  0.9454295 1.0000000  0.94340233
## May        0.6357803  0.6923741  0.7959107 0.9434023  1.00000000
## June       0.5653482  0.6238858  0.7204128 0.8883443  0.97348737
## July       0.5739173  0.6230933  0.7164001 0.8623990  0.94159722
## August     0.6449861  0.6911152  0.7795227 0.8952261  0.93878858
## September  0.8140647  0.8501741  0.9102336 0.9683572  0.94024617
## October    0.9119085  0.9296643  0.9642011 0.9620058  0.87667078
## November   0.9670438  0.9729540  0.9733819 0.9222615  0.78982392
## December   0.9941371  0.9826778  0.9565235 0.8510111  0.67672343
## Annual     0.9114518  0.9362291  0.9712308 0.9715578  0.88435174
## Amplitude -0.7630887 -0.7133580 -0.5979189 -0.3316471 -0.03068248
## Latitude  -0.7190498 -0.7835144 -0.8629710 -0.9480485 -0.91291397
## Longitude -0.6692735 -0.6543726 -0.6107641 -0.4002547 -0.15629135
##                 June       July      August September    October
## January   0.56534823 0.57391734 0.644986084 0.8140647  0.9119085
## February  0.62388583 0.62309329 0.691115222 0.8501741  0.9296643
## March     0.72041282 0.71640008 0.779522722 0.9102336  0.9642011
## April     0.88834432 0.86239896 0.895226114 0.9683572  0.9620058
## May       0.97348737 0.94159722 0.938788583 0.9402462  0.8766708
## June       1.00000000  0.98371056  0.964775206  0.9280953  0.8334642
## July       0.98371056  1.00000000  0.986771890  0.9319571  0.8377886
## August     0.96477521  0.98677189  1.000000000  0.9611721  0.8852003
## September  0.92809533  0.93195707  0.961172118  1.0000000  0.9746050
## October    0.83346416  0.83778863  0.885200326  0.9746050  1.0000000
## November   0.73676842  0.73856449  0.792693502  0.9221664  0.9806903
## December   0.60870176  0.61705361  0.680595684  0.8407749  0.9342659
## Annual     0.84427416  0.84846656  0.894804345  0.9781066  0.9953382
## Amplitude  0.08719624  0.09087881 -0.006177653 -0.2553916 -0.4471522
## Latitude  -0.87416792 -0.85211733 -0.885375962 -0.9264719 -0.8938376
## Longitude -0.08358145 -0.10522045 -0.186829886 -0.3520903 -0.4631182
##             November   December     Annual    Amplitude   Latitude
## January    0.9670438  0.9941371  0.9114518 -0.763088732 -0.7190498
## February   0.9729540  0.9826778  0.9362291 -0.713357998 -0.7835144
## March      0.9733819  0.9565235  0.9712308 -0.597918907 -0.8629710
## April      0.9222615  0.8510111  0.9715578 -0.331647130 -0.9480485
## May        0.7898239  0.6767234  0.8843517 -0.030682481 -0.9129140
## June       0.7367684  0.6087018  0.8442742  0.087196239 -0.8741679
## July       0.7385645  0.6170536  0.8484666  0.090878813 -0.8521173
## August     0.7926935  0.6805957  0.8948043 -0.006177653 -0.8853760
## September  0.9221664  0.8407749  0.9781066 -0.255391576 -0.9264719
## October    0.9806903  0.9342659  0.9953382 -0.447152158 -0.8938376
## November   1.0000000  0.9815333  0.9758787 -0.592480031 -0.8368799
## December   0.9815333  1.0000000  0.9285954 -0.720842361 -0.7413846
## Annual     0.9758787  0.9285954  1.0000000 -0.439046750 -0.9027853
## Amplitude -0.5924800 -0.7208424 -0.4390467  1.000000000  0.2031530
## Latitude  -0.8368799 -0.7413846 -0.9027853  0.203152955  1.0000000
## Longitude -0.5266451 -0.6156694 -0.4769927  0.732863148  0.3154657
##             Longitude
## January   -0.66927347
## February  -0.65437264
## March     -0.61076414
## April     -0.40025470
## May       -0.15629135
## June      -0.08358145
## July      -0.10522045
## August    -0.18682989
## September -0.35209033
## October   -0.46311819
## November  -0.52664511
## December  -0.61566936
## Annual    -0.47699270
## Amplitude  0.73286315
## Latitude   0.31546570
## Longitude  1.00000000
```

```r
#Scatterplot matrix
ggpairs(temp[,2:9])
```



```
ggpairs(temp[,10:17])
```



As it can be easily caught, the most of the histogram shows the the mode to be placed in the middle. And, another interesting thing is that most of the plots show the decent correlation. Especially, it is interesting latitude has high correlation with other months, and actually it makes sense as latitude gets higher, it gets colder. However, longitude has smaller effects on temperatures compared to latitude.

Part 1

Here, I extract the active and supplementary variables and individuals, just in case.

```
#active individuals and variables, with the citi names.
active <- temp[1:23, 1:13]
#supplementary individuals and variables, with the citi names
supplementary <- temp[24:35, c(1, 14:18)]
```

I did not change the column and row names for prcomp, since they are just for double checking my answers.

a. Obtain the loadings and store them in a matrix, include row and column names. Display the first four loadings (10 pts).

```
#a - first way (eigen decomposition)
scale_active <- scale(active[,-1], T, T)
R <- cor(scale_active)
v1 <- eigen(R)$vectors
colnames(v1) <- paste0("PC", 1:12)
rownames(v1) <- colnames(active[,-1])
v1[,1:4]
```

```
##                  PC1         PC2           PC3          PC4
## January   -0.2671050 -0.39091041  0.1907187341 -0.059731884
## February  -0.2803688 -0.33534791 -0.0097552190 -0.427798846
## March     -0.2996355 -0.21137095 -0.3399569587 -0.397667051
## April     -0.3087780  0.07324821 -0.5579573828 -0.127078736
## May       -0.2757927  0.33680390 -0.4392770157  0.392591602
## June      -0.2642082  0.40118372  0.1394431457 -0.000489339
## July      -0.2676478  0.37421361  0.4325313064 -0.222824851
## August    -0.2882824  0.29568869  0.2462557102 -0.226852869
## September -0.3124996  0.11221817  0.0636774480 -0.026537477
## October   -0.3144017 -0.06235990 -0.0001874864  0.366581807
## November  -0.3019515 -0.21291689  0.1244515912  0.356372148
## December  -0.2768287 -0.34787886  0.2386777766  0.349002937
```

```
#a - second way (eigen decomposition 2)
R <- (1/ (nrow(scale_active) - 1)) * t(scale_active) %*% scale_active
v2 <- eigen(R)$vectors
colnames(v2) <- paste0("PC", 1:12)
rownames(v2) <- colnames(active[,-1])
#v2[,1:4]

#a - third way (svd decomposition)
svd <- svd(scale_active)
v3 <- svd$v
colnames(v3) <- paste0("PC", 1:12)
rownames(v3) <- colnames(active[,-1])
#v3[,1:4]

#check with prcomp
prcomp(active[,-1], scale = T)$rotation[,1:4]
```

```
##                  PC1         PC2           PC3          PC4
## January   -0.2671050 -0.39091041  0.1907187341  0.059731884
## February  -0.2803688 -0.33534791 -0.0097552190  0.427798846
## March     -0.2996355 -0.21137095 -0.3399569587  0.397667051
## April     -0.3087780  0.07324821 -0.5579573828  0.127078736
## May       -0.2757927  0.33680390 -0.4392770157 -0.392591602
## June      -0.2642082  0.40118372  0.1394431457  0.000489339
## July      -0.2676478  0.37421361  0.4325313064  0.222824851
## August    -0.2882824  0.29568869  0.2462557102  0.226852869
## September -0.3124996  0.11221817  0.0636774480  0.026537477
## October   -0.3144017 -0.06235990 -0.0001874864 -0.366581807
## November  -0.3019515 -0.21291689  0.1244515912 -0.356372148
## December  -0.2768287 -0.34787886  0.2386777766 -0.349002937
```

b. Obtain the principal components and store them in a matrix, include row and column names. Display the first four PCs (10 pts).

```r
#b - first way (eigen decompsition)
z1 <- scale_active %*% v1
colnames(z1) <- paste0("PC", 1:12)
rownames(z1) <- active[,1]
z1[,1:4]
```

```
##                    PC1          PC2         PC3         PC4
## Amsterdam  -0.22195025 -1.341234829 -0.10209889  0.27657677
## Athens     -7.43360390  0.909925426  0.54908835  0.28025851
## Berlin      0.28153099  0.016092403 -0.28422057  0.05437108
## Brussels   -0.61729994 -1.151341565 -0.14870076 -0.01669466
## Budapest   -1.63136395  1.675051425 -0.48801530 -0.10996512
## Copenhagen  1.43025066 -0.481240562  0.43068897  0.17283180
## Dublin      0.49413580 -2.614731574 -0.17458563 -0.02925371
## Elsinki     3.94757646  0.451883416  0.58015037  0.23907168
## Kiev        1.67458427  1.963469194 -0.16691889  0.11032784
## Krakow      1.23099109  0.855756199 -0.26794138 -0.03573418
## Lisbon     -5.47621202 -1.520180219 -0.26440940  0.13422375
## London     -0.05637309 -1.539174219 -0.08281278 -0.05087152
## Madrid     -3.97473636  0.682329696  0.45164881 -0.64836153
## Minsk       3.16672621  1.360708200 -0.07068160  0.17931195
## Moscow      3.38650106  2.134053560 -0.29467958  0.00526448
## Oslo        3.23331905  0.303237840  0.28881834 -0.18641912
## Paris      -1.38850720 -0.877868695 -0.10790241  0.07732927
## Prague      0.10660691  0.682697725 -0.23723947 -0.09816888
## Reykjavik   4.60066569 -2.892196405 -0.05662577 -0.19107214
## Rome       -5.26370105  0.287243017  0.18510843  0.01231239
## Sarajevo   -0.15985914  0.312466849 -0.35657228 -0.07199691
## Sofia      -0.40862719  0.777598162 -0.23556939 -0.04675281
## Stockholm   3.07934588  0.005454959  0.85347084 -0.05658895
```

```r
#b - second way (svd decomposition)
u <- svd$u
d <- diag(svd$d)
z2 <- u %*% d
colnames(z2) <- paste0("PC", 1:12)
rownames(z2) <- active[,1]
#z2[,1:4]
```

```
#b - check with procomp
z3 <- scale_active %*% prcomp(active[,-1], scale = T)$rotation
colnames(z3) <- paste0("PC", 1:12)
rownames(z3) <- active[,1]
#z3[,1:4]

#b - doublecheck with prcomp
prcomp(active[,-1], scale = T)$x[,1:4]
```

```
##             PC1          PC2         PC3         PC4
## 1  -0.22195025 -1.341234829 -0.10209889 -0.27657677
## 2  -7.43360390  0.909925426  0.54908835 -0.28025851
## 3   0.28153099  0.016092403 -0.28422057 -0.05437108
## 4  -0.61729994 -1.151341565 -0.14870076  0.01669466
## 5  -1.63136395  1.675051425 -0.48801530  0.10996512
## 6   1.43025066 -0.481240562  0.43068897 -0.17283180
## 7   0.49413580 -2.614731574 -0.17458563  0.02925371
## 8   3.94757646  0.451883416  0.58015037 -0.23907168
## 9   1.67458427  1.963469194 -0.16691889 -0.11032784
## 10  1.23099109  0.855756199 -0.26794138  0.03573418
## 11 -5.47621202 -1.520180219 -0.26440940 -0.13422375
## 12 -0.05637309 -1.539174219 -0.08281278  0.05087152
## 13 -3.97473636  0.682329696  0.45164881  0.64836153
## 14  3.16672621  1.360708200 -0.07068160 -0.17931195
## 15  3.38650106  2.134053560 -0.29467958 -0.00526448
## 16  3.23331905  0.303237840  0.28881834  0.18641912
## 17 -1.38850720 -0.877868695 -0.10790241 -0.07732927
## 18  0.10660691  0.682697725 -0.23723947  0.09816888
## 19  4.60066569 -2.892196405 -0.05662577  0.19107214
## 20 -5.26370105  0.287243017  0.18510843 -0.01231239
## 21 -0.15985914  0.312466849 -0.35657228  0.07199691
## 22 -0.40862719  0.777598162 -0.23556939  0.04675281
## 23  3.07934588  0.005454959  0.85347084  0.05658895
```

c. Obtain the eigenvalues and store them in a vector. Display the entire vector, and compute their sum. (10 pts)

```
#c - first way (eigen decomposition)
eigen1 <- eigen(R)$values
eigen1 <- as.vector(eigen1)
eigen1
```

```
##  [1] 9.9477504204 1.8476485015 0.1262558038 0.0382934463 0.0167094089
##  [6] 0.0128330357 0.0058302931 0.0020318929 0.0010234516 0.0009527707
## [11] 0.0005367834 0.0001341917
```

```
sum(eigen1) # and it makes sense, as we have 12 variables.
```

```
## [1] 12
```

```r
#c - second way (svd decomposition)
x <- scale(u %*% d %*% t(v3), T, T)
#eigen(cor(x))$values
#sum(eigen(cor(x))$values)

#c - check with prcomp
(prcomp(active[,-1], scale = T)$sdev)^2
```

```
##  [1] 9.9477504204 1.8476485015 0.1262558038 0.0382934463 0.0167094089
##  [6] 0.0128330357 0.0058302931 0.0020318929 0.0010234516 0.0009527707
## [11] 0.0005367834 0.0001341917
```

```r
#sum((prcomp(active[,-1], scale = T)$sdev)^2)

#c - check with prcomp
#as.vector(apply(prcomp(active[,-1], scale = T)$x, 2, sd)^2)
#sum(as.vector(apply(prcomp(active[,-1], scale = T)$x, 2, sd)^2))
```

Z = XV and dim(Z) = n by p, dim(V) = p by p, dim(X) = n by p.

scale(data, T, T) -> standaradized matrix, X.

pca_prcomp$rotation -> loadings/eigen-vectors, V

pca_prcomp$x == scale(quant, T, T) %*% pca_prcomp$rotation -> principal components (matrix) / scores, Z

pca_prcomp$center -> colMeans

pca_prcomp$scale -> apply(data, 2, sd)

pca_prcomp$sdev -> apply(pca_prcomp$x, 2, sd) or sqrt(eigen(cor(scale(quant, T, T)))$values)

Sum of eigenvalues means sum of variations of PCA, as each variance is equal to the corresponding eigen value.

Part 2

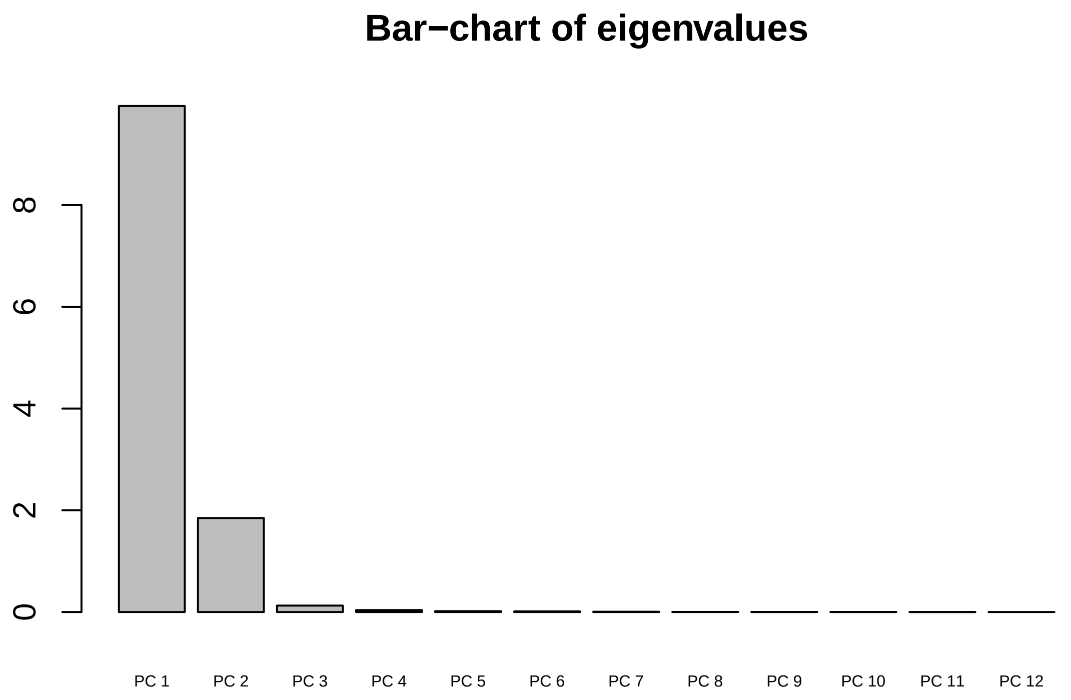
a. Make a summary table of the eigenvalues: eigenvalue in the first column (each eigenvalue

represents the variance captured by each component); percentage of variance in the second column; and cumulative percentage in the third column. Comment on the table. (10 pts)

```
#a
eigen_data <- matrix(0, nrow = round(sum(eigen1),0), ncol = 3)
colnames(eigen_data) <- c("eigenvalue", "percentage", "cumulative.percentage")
rownames(eigen_data) <- paste0("comp", 1:sum(eigen1))

eigen_data[,1] <- eigen1
percentage <- apply(as.matrix(eigen1), 2, sum(eigen1), FUN = "/") * 100
eigen_data[,2] <- percentage

cum_fun <- function(x){ #x should be n * 1 column matrix
  for (i in 2:nrow(x)){
    x[i,] <- x[i-1,] + x[i,]
  }
  return(x)
}
cumulative <- cum_fun(percentage) #or use cumsum!!!
eigen_data[,3] <- cumulative

print(eigen_data)
```

```
##          eigenvalue   percentage cumulative.percentage
## comp1  9.9477504204 82.897920170              82.89792
## comp2  1.8476485015 15.397070846              98.29499
## comp3  0.1262558038  1.052131698              99.34712
## comp4  0.0382934463  0.319112052              99.66623
## comp5  0.0167094089  0.139245074              99.80548
## comp6  0.0128330357  0.106941964              99.91242
## comp7  0.0058302931  0.048585776              99.96101
## comp8  0.0020318929  0.016932441              99.97794
## comp9  0.0010234516  0.008528764              99.98647
## comp10 0.0009527707  0.007939756              99.99441
## comp11 0.0005367834  0.004473195              99.99888
## comp12 0.0001341917  0.001118264             100.00000
```

```
barplot(eigen_data[,1], main = "Bar-chart of eigenvalues", names = c(paste("PC", 1:12)),
        cex.names = 0.5)
```



```r
paste("Variation in first PC:", eigen_data[1,2], "%")
```

```
## [1] "Variation in first PC: 82.897920169599 %"
```

```r
paste("Variation in second PC:", eigen_data[2,2], "%")
```

```
## [1] "Variation in second PC: 15.3970708461502 %"
```

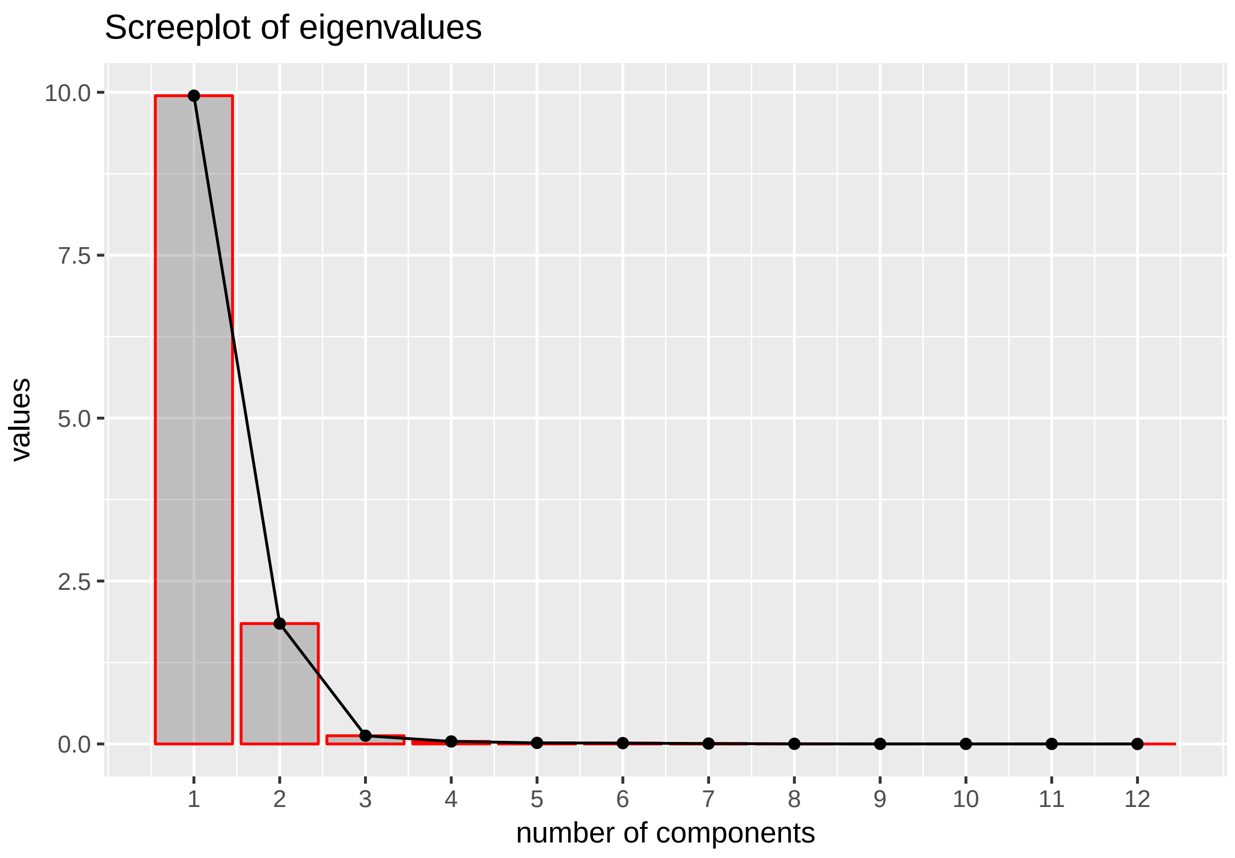
Comment on part a:

As it can be easily seen in the table, there are two/three first big eigen values on the top. And, the first two/three PC have the most variations. So, I will only retain first two PCs.

b. Create a scree-plot (with axis labels) of the eigenvalues. What do you see? How do you

read/interpret this chart? (10 pts)

```r
#b
graph <- ggplot(as.data.frame(eigen_data[,1]), aes(x = 1:12, y = as.numeric(eigen_data[,1])))
graph <- graph + geom_bar(stat = "identity", alpha = 0.3, color = "red") + geom_point() +
  geom_line() +
  labs(title = "Screeplot of eigenvalues", x = "number of components", y = "values") +
  scale_x_continuous(breaks=seq(1,12,1))
graph
```



Comment on part b:

I will say the elbow is on the second/third one. So, I will retain two/three PCs for my further investigations (for most of the other parts of the homeowrk, the problems only ask me to investigate PC1 and PC2, so it might be enough for me to retain only two). However, this is not 100% accurate, and other people might have differnt opinions, and that is why I personally do not like screeplot method compared to other methods.

c. If you had to choose a number of dimensions (i.e. a number of PCs), how many would you choose and why? (10 pts)

Comment on part c:

Choose the the number of PCs/components to retain. There are four big different ways: screeplot with elbow, predetermined amount of variation, kaiser rule, jollife rule,

- Retain just enough components to explain some specified, large percentage of the total variation of the original variables. For example, how many PCs would you retain to capture 85% of the total variation? —> predetermined amount of variation.

Ans: Retain only the first one.

- Exclude those PCs whose eigenvalues are less than the average: $\sum_{h=1}^{p} \frac{\lambda_h}{p}$. Using this criterion, how many PCs would you retain?

Ans: The average will be 1, always. And, I can retain 2 PCs.

- When the PCs are extracted from the correlation matrix, like in this case, the average eigenvalue is one; components with eigenvalues less than one are therefore excluded. This rule is known as Kaiser's rule. Using this criterion, how many PCs would you retain? —> Kaiser's rule

Ans: Retain first two.

- Jollife (1972) proposed a variation of the Kaiser's rule: exclude PCs whose eigenvalues are less than 0.7. Under this criterion: how many PCs would you retain? —> Jollife's rule

Ans: Retain first three.

- Cattel (1965) suggests plotting the eigenvalues with the so-called scree-plot or scree diagram—like the bar-chart of eigenvalues. Cattel suggests looking for an "elbow" in the diagram, this would correspond to a point where "large" eigenvalues cease and "small" eigenvalues begin. With this rule, how many PCs would you retain? —-> Scree plot

Ans: Retain first three.

Conclusion:

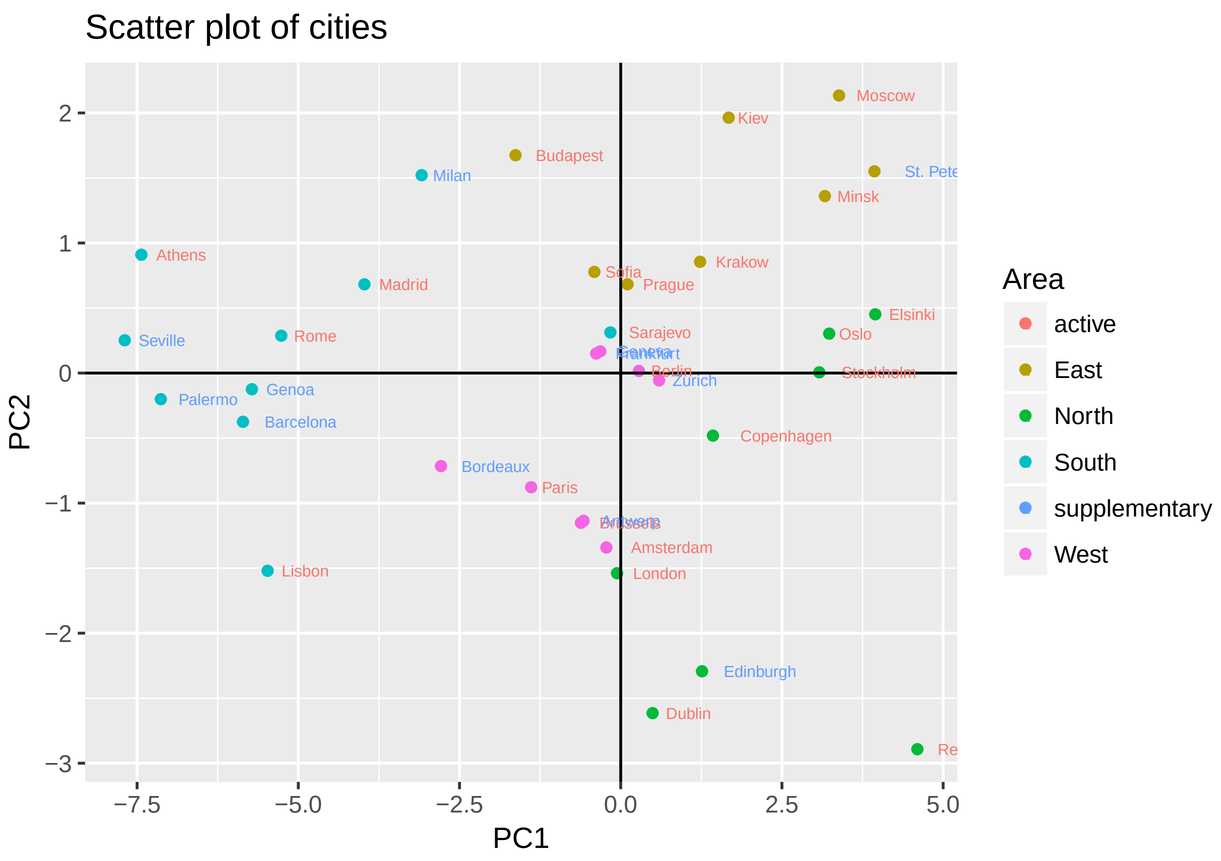
For consistency, I will just keep two PCs for my future investigations.

Part 3

a. Create a scatter plot of the cities on the 1st and 2nd PCs (10 pts).

–> The key point here is to get supplementary scaled by active mean and standard deviation. Also, loadings should come from active, not from supplementary. It is because we want to use one city from one country to make it equal PC effect, and also, we want to compare how supplementary works on the active PC plots of individuals. That is the point of this example.

```r
options(scipen=999)
#a

#supplementary individuals, with the city names.
supplementary <- temp[24:35, 1:13]

#scale with active mean and sd.
scale_suppl <- scale(supplementary[,-1], colMeans(active[,-1]), apply(active[,-1], 2, sd))


pc1_suppl <- scale_suppl %*% v1[,1]
rownames(pc1_suppl) <- supplementary[,1]
pc2_suppl <- scale_suppl %*% v1[,2]
rownames(pc2_suppl) <- supplementary[,1]

pc1_comb <- rbind(z1[,1, drop = F], pc1_suppl)
pc2_comb <- rbind(z1[,2, drop = F], pc2_suppl)

#PCA matrix/scores for active and supplementary combined.
pca_combined <- cbind(pc1_comb, pc2_comb)
#add this so it will help me draw ggplots.
ac_sup_diff <- c(rep("active", nrow(active)), rep("supplementary", nrow(supplementary)))
city <- temp[,ncol(temp), drop = F]
pca_combined <- cbind(pca_combined, ac_sup_diff, city)

graph2 <- ggplot(as.data.frame(pca_combined), aes(x = PC1, y = PC2, colour = Area))
graph2 <- graph2 + geom_point() + labs(title = "Scatter plot of cities") +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +
  geom_text(aes(color = factor(ac_sup_diff), label = rownames(pca_combined), hjust = -0.3), size = 2)
graph2
```



Comment on part a:

Most of the supplementary variables cities (the ones that have color of blue on the writings) are on the negative sides on PC1. And, it is because most of the supplementary cities are on the low latitude, and that totally makes sense. From the color of areas on the graph, PC1 might stand for something similiar to latitude. Also, PC2 might stand for longitude, so for example, Moscow is in the eastl on the other hand, Paris is on the west side of the Europe. Furthermore, the graph displays the cities from the east on the positive side of PC2 and PC1, the cities from the north on the positive side of PC1 but negative side of PC2, the cities from the south on the negative side of PC1 and equally spreaded on PC2, and the cities from the west compacted near the origin of PC1 and slightly negative side of PC2.

b. Compute the quality of individuals representation, that is, the squared cosines given by:

$cos^2(i,\ k) = \frac{z_{ik}^2}{d^2(x_i,\ g)}$, where z_{ik} is the square value of the i-th individual on PC k, x_i represents the row-vector of the i-th individual, and g is the centroid (i.e. average individual)

And, $\sum_{k=1}^{r} cos^2(i,\ k) = 1$ should be satisfied.

```
options(scipen=999)
#b
quality <- function(pca, standardized){
  for (i in 1: nrow(z1)){
    pca[i,] <- pca[i,]^2 / sum((standardized[i,])^2)
  }
  return(pca)
}

quality_mat <- quality(z1, scale_active)
quality_mat[,1:4]
```

```
##                    PC1            PC2         PC3            PC4
## Amsterdam  0.02474831 0.903740763090 0.005236924 0.038429580157
## Athens     0.97830645 0.014658437773 0.005337778 0.001390572682
## Berlin     0.32789958 0.001071347207 0.334194626 0.012229939815
## Brussels   0.21639751 0.752780143771 0.012557007 0.000158275914
## Budapest   0.46337591 0.488526383866 0.041466618 0.002105433981
## Copenhagen 0.80588015 0.091236916682 0.073075813 0.011767751283
## Dublin     0.03411320 0.955177510969 0.004258404 0.000119561598
## Elsinki    0.95654320 0.012534190565 0.020659730 0.003508324581
## Kiev       0.41732984 0.573737981505 0.004146449 0.001811488721
## Krakow     0.64509341 0.311754616560 0.030562745 0.000543601236
## Lisbon     0.92554429 0.071322548310 0.002157697 0.000556026444
## London     0.00131785 0.982421961196 0.002843919 0.001073178220
## Madrid     0.93424771 0.027531757431 0.012062772 0.024858783384
## Minsk      0.84071389 0.155223400617 0.000418832 0.002695539247
## Moscow     0.71081284 0.282269181645 0.005382114 0.000001717765
## Oslo       0.97838231 0.008605542521 0.007806583 0.003252313194
## Paris      0.69481859 0.277737348752 0.004196019 0.002155078014
## Prague     0.01987080 0.814894959580 0.098405324 0.016849709692
## Reykjavik  0.71527677 0.282675606254 0.000108358 0.001233751020
## Rome       0.99549987 0.002964542880 0.001231151 0.000005446829
## Sarajevo   0.07819664 0.298759028151 0.389052585 0.015861376589
## Sofia      0.18627345 0.674538726845 0.061906201 0.002438438522
## Stockholm  0.92423563 0.000002900338 0.070997512 0.000312125390
```

```
rowSums(quality_mat) #Prove the sum of every row is equal to 1.
```

```
##  Amsterdam     Athens     Berlin   Brussels   Budapest Copenhagen
##          1          1          1          1          1          1
##     Dublin    Elsinki       Kiev     Krakow     Lisbon     London
##          1          1          1          1          1          1
##     Madrid      Minsk     Moscow       Oslo      Paris     Prague
##          1          1          1          1          1          1
##  Reykjavik       Rome   Sarajevo      Sofia  Stockholm
##          1          1          1          1          1
```

```
paste("Best represented city on the first PC:", names(which.max(quality_mat[,1])))
```

```
## [1] "Best represented city on the first PC: Rome"
```

```
paste("Best represented city on the second PC:", names(which.max(quality_mat[,2])))
```

```
## [1] "Best represented city on the second PC: London"
```

```
paste("Worst represented city on the first PC:", names(which.min(quality_mat[,1])))
```

```
## [1] "Worst represented city on the first PC: London"
```

```
paste("Worst represented city on the second PC:", names(which.min(quality_mat[,2])))
```

```
## [1] "Worst represented city on the second PC: Stockholm"
```

Supplementary cities should not contribute to PCA, so I would not bother calculating them. Also, since my X is standardized before in part a, the g will be the origin. One thing I should be aware is that when they said the distance, it does not mean to use "dist" function embedded in R. "dist" function in R get the distance for each element in vector; however, we try to get distance of each element to zero. Also, be aware that every element of the same individual from PCA matrix is divided by the same constant.

c. Compute the contributions of the individuals to each extracted PC, using the formula:

$ctr(i,\ k) = \frac{m_i\ z_{i,k}^2}{\lambda_k} \times 100$, where m_i is the mass or weight of individual i, in our case: $\frac{1}{n-1}$, z_{ik} is the value of k-th PC for individual i, and λ_k is the eigenvalue associated to k-th PC.

And, sum of the contributions of all observations is equal to 100.

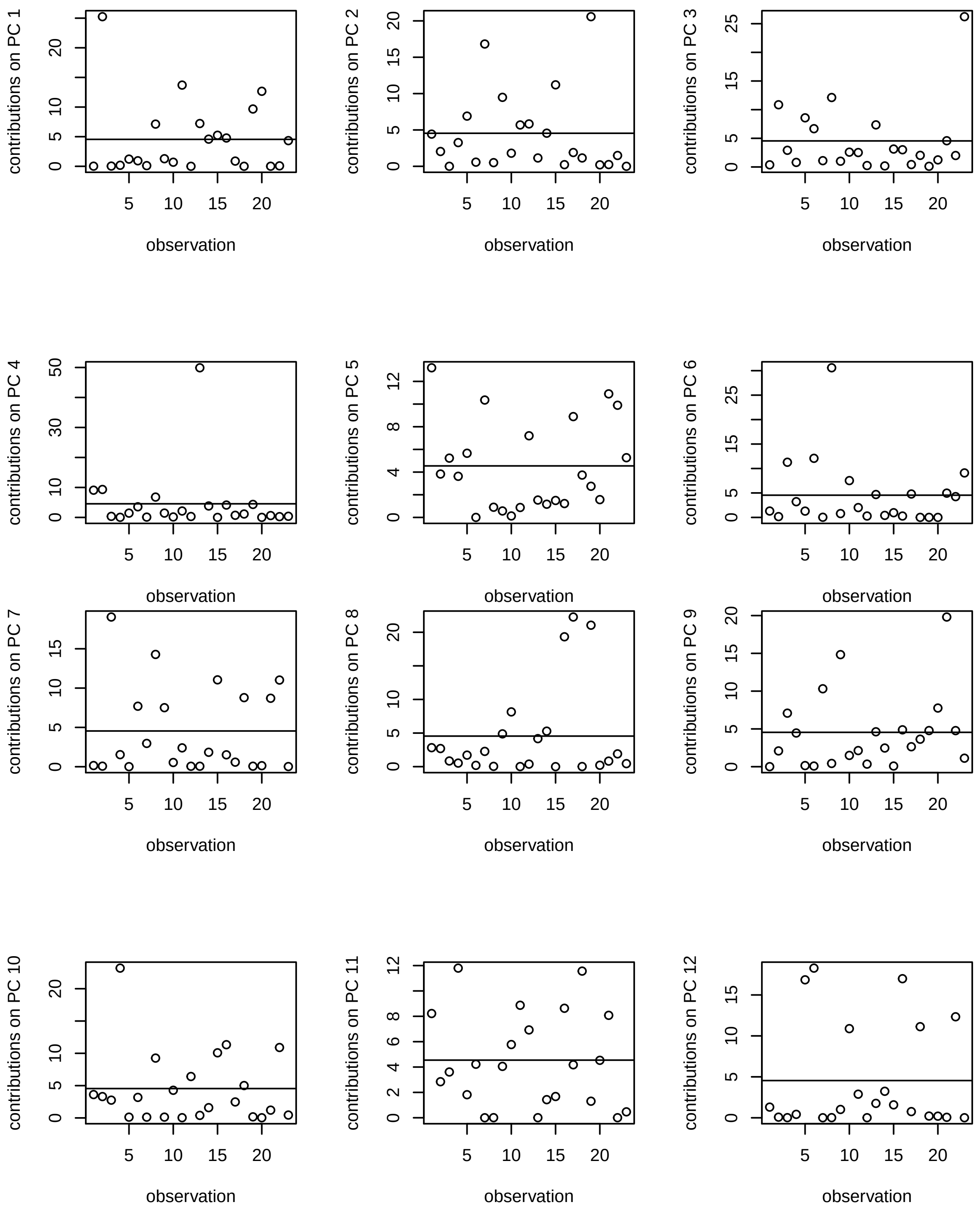
```
options(scipen=999)

contribution <- function(pca, eigen){
  for (i in 1:ncol(pca)){
    pca[,i] <- (pca[,i]^2 / (eigen)[i]) * 100 / (nrow(pca) - 1)
  }
  return(pca)
}
contribution_mat <- contribution(z1, eigen1)

colSums(contribution_mat) #all columns are summed to 100.
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11 PC12
##  100  100  100  100  100  100  100  100  100  100  100  100
```

```
contribution_mat[,1:4]
```

```
##                     PC1            PC2        PC3          PC4
## Amsterdam   0.022509389  4.42555365455  0.3752909  9.079967206
## Athens     25.249412071  2.03689933650 10.8545150  9.323317492
## Berlin      0.036216364  0.00063708848  2.9082851  0.350904333
## Brussels    0.174118494  3.26111663909  0.7960720  0.033083230
## Budapest    1.216057639  6.90262459048  8.5741849  1.435366347
## Copenhagen  0.934709699  0.56974748329  6.6781085  3.545685387
## Dublin      0.111569397 16.81946539237  1.0973444  0.101581521
## Elsinki     7.120549993  0.50235504783 12.1173352  6.784364061
## Kiev        1.281346111  9.48431890320  1.0030831  1.444851066
## Krakow      0.692408290  1.80159875395  2.5846726  0.151572536
## Lisbon     13.702914482  5.68523104811  2.5169800  2.138511623
## London      0.001452098  5.82818764416  0.2468998  0.307186563
## Madrid      7.218867870  1.14537186325  7.3439161 49.898483504
## Minsk       4.582193987  4.55499567214  0.1798617  3.816553334
## Moscow      5.240284554 11.20388377916  3.1262670  0.003289757
## Oslo        4.776937527  0.22621674680  3.0031395  4.125093151
## Paris       0.880944827  1.89590726081  0.4191682  0.709807693
## Prague      0.005193057  1.14660775875  2.0262818  1.143932947
## Reykjavik   9.671498999 20.57849117466  0.1154394  4.333588025
## Rome       12.660033938  0.20298171879  1.2336114  0.017994418
## Sarajevo    0.011676896  0.24019604986  4.5774239  0.615291054
## Sofia       0.076296912  1.48753918875  1.9978537  0.259458701
## Stockholm   4.332807405  0.00007320502 26.2242659  0.380116049
```

```r
#The line is showing the average under uniform contribution.
par(mfrow = c(2,3))
for (i in 1:12){
  plot(contribution_mat[,i], xlab = "observation", ylab = paste("contributions on PC", i))
  abline(h = 100/(nrow(z1) - 1), lty = 1)
}
```



```r
#Print out the 90 percentile of the contribution for each PC
for (i in 1:12){
  paste("PC", i,":",round(quantile(contribution_mat[,i], 0.9), 7))
}
print("So, any of points given above numbers for each PC can be regarded as outliers")
```

```
## [1] "So, any of points given above numbers for each PC can be regarded as outliers"
```

```r
print("and it is better to take them out, as they have influences.")
```

```
## [1] "and it is better to take them out, as they have influences."
```

```r
print("Influential cities are below:")
```

```
## [1] "Influential cities are below:"
```

```r
for (i in 1:ncol(contribution_mat)){
  for (j in 1:nrow(contribution_mat)){
    if(round(quantile(contribution_mat[,i], 0.90), 7) < contribution_mat[j,i]){
      print(paste("Influential cities for PC", i," are", rownames(contribution_mat)[j]))
    }
  }
}
```

```
## [1] "Influential cities for PC 1  are Athens"
## [1] "Influential cities for PC 1  are Lisbon"
## [1] "Influential cities for PC 1  are Rome"
## [1] "Influential cities for PC 2  are Dublin"
## [1] "Influential cities for PC 2  are Moscow"
## [1] "Influential cities for PC 2  are Reykjavik"
## [1] "Influential cities for PC 3  are Athens"
## [1] "Influential cities for PC 3  are Elsinki"
## [1] "Influential cities for PC 3  are Stockholm"
## [1] "Influential cities for PC 4  are Amsterdam"
## [1] "Influential cities for PC 4  are Athens"
## [1] "Influential cities for PC 4  are Madrid"
## [1] "Influential cities for PC 5  are Amsterdam"
## [1] "Influential cities for PC 5  are Dublin"
## [1] "Influential cities for PC 5  are Sarajevo"
## [1] "Influential cities for PC 6  are Berlin"
## [1] "Influential cities for PC 6  are Copenhagen"
## [1] "Influential cities for PC 6  are Elsinki"
## [1] "Influential cities for PC 7  are Berlin"
## [1] "Influential cities for PC 7  are Elsinki"
## [1] "Influential cities for PC 7  are Moscow"
## [1] "Influential cities for PC 8  are Oslo"
## [1] "Influential cities for PC 8  are Paris"
## [1] "Influential cities for PC 8  are Reykjavik"
## [1] "Influential cities for PC 9  are Dublin"
## [1] "Influential cities for PC 9  are Kiev"
## [1] "Influential cities for PC 9  are Sarajevo"
## [1] "Influential cities for PC 10  are Brussels"
## [1] "Influential cities for PC 10  are Oslo"
## [1] "Influential cities for PC 10  are Sofia"
## [1] "Influential cities for PC 11  are Brussels"
## [1] "Influential cities for PC 11  are Lisbon"
## [1] "Influential cities for PC 11  are Prague"
## [1] "Influential cities for PC 12  are Budapest"
## [1] "Influential cities for PC 12  are Copenhagen"
## [1] "Influential cities for PC 12  are Oslo"
```

Part 4

a. Calculate the correlation of all quantitative variables (active and supplementary) with the principal components. Store the correlations in a matrix or data frame, include row and column names. Display the first four columns. (10 pts)

I include two different ways to calculate.

```r
var_comb <- temp[1:23, 1:17]
var_comb_scale <- scale(var_comb[,-1], T, T) #X

#First - Multiply square root of eigenvalues of R with eigenvectors/loadings column wise
#(or, I can say that multiply standard deviation of PCA matrix with eigenvectors/loadings column wise)
#This only works for active individual and variable case since loading&e-values are all from original o

#sd <- sqrt(eigen1)
#cor_mat <- round(sweep(v1, 2, sd, FUN = "*"), 6)
#rownames(cor_mat) <- names(var_comb[-1])
#colnames(cor_mat) <- paste0("PC", 1:12)

#Second - get correlation matrix with data set and PCs.
cor_mat <- function(data, pc){
  new_mat <- matrix(0, nrow = ncol(data), ncol = ncol(pc))
  for (i in 1:ncol(data)){
    for (j in 1:ncol(pc)){
      new_mat[i,j] <- cor(data[,i], pc[,j])
    }
  }
  rownames(new_mat) <- names(data)
  colnames(new_mat) <- paste0("PC", 1:12)
  return(round((new_mat), 6))
}
cor_mat(var_comb[,-1], z1)
```

```
##                 PC1       PC2       PC3       PC4       PC5       PC6
## January   -0.842451 -0.531358  0.067767 -0.011689 -0.031620 -0.018270
## February  -0.884285 -0.455833 -0.003466 -0.083715 -0.043625  0.028869
## March     -0.945052 -0.287313 -0.120795 -0.077818  0.043317 -0.036942
## April     -0.973888  0.099565 -0.198256 -0.024868 -0.016577  0.015504
## May       -0.869852  0.457812 -0.156086  0.076825 -0.031288 -0.041168
## June      -0.833314  0.545322  0.049548 -0.000096 -0.058509  0.041561
## July      -0.844163  0.508662  0.153689 -0.043604 -0.003554 -0.046822
## August    -0.909244  0.401924  0.087501 -0.044392  0.036429 -0.006378
## September -0.985625  0.152536  0.022626 -0.005193  0.043630  0.043745
## October   -0.991625 -0.084765 -0.000067  0.071735  0.059634  0.016596
## November  -0.952357 -0.289414  0.044221  0.069737  0.002706  0.029829
## December  -0.873119 -0.472866  0.084808  0.068295 -0.021124 -0.036425
## Annual    -0.997548 -0.068453  0.004567  0.000004 -0.002369 -0.009787
## Amplitude  0.314076  0.944414  0.039188 -0.005742  0.034458 -0.022407
## Latitude   0.909911 -0.215437  0.181985  0.059290 -0.127730 -0.119352
## Longitude  0.364458  0.644973 -0.036434  0.247323 -0.183461  0.108200
##                 PC7       PC8       PC9      PC10      PC11      PC12
## January    0.039876 -0.003800 -0.011373  0.004609  0.010870  0.000440
## February  -0.001421  0.010381  0.015217 -0.012597 -0.000611  0.000125
## March     -0.003018 -0.012813  0.007609  0.013919 -0.003736 -0.000812
## April     -0.028314  0.000277 -0.018782 -0.005451  0.002117  0.001413
## May        0.024576  0.011649  0.009046 -0.001854  0.000773 -0.001542
## June       0.003707 -0.021274  0.003789  0.010395 -0.003568  0.002138
## July      -0.028439 -0.004531  0.000137 -0.006107  0.006679 -0.003823
## August     0.012204  0.022619 -0.003376  0.002991 -0.003773  0.006454
## September  0.027681  0.000111 -0.006723 -0.004548 -0.006312 -0.006857
## October   -0.001209 -0.016520  0.007726 -0.010609  0.007498  0.003892
## November  -0.030762  0.018116  0.003054  0.015169  0.004613 -0.002233
## December  -0.011743 -0.004887 -0.005484 -0.005324 -0.014502  0.000979
## Annual     0.006193  0.000707 -0.001353  0.000845 -0.002210 -0.002223
## Amplitude -0.073794  0.004096  0.012238 -0.014048  0.005151 -0.004307
## Latitude   0.123397 -0.067131  0.065610  0.088979  0.057159  0.075763
## Longitude -0.281277  0.362721  0.041406  0.055039  0.066834 -0.010892
```

```r
#Third - get correlation matrix with standaridized matrix with PCA matrix
#(Realize that although you standardize the matrix and do correlation, you get the same correlation.)
x <- round(cor(var_comb_scale, z1), 6)


#check
identical(x, cor_mat(var_comb[,-1], z1))
```

```
## [1] TRUE
```

```r
x[,1:4]
```

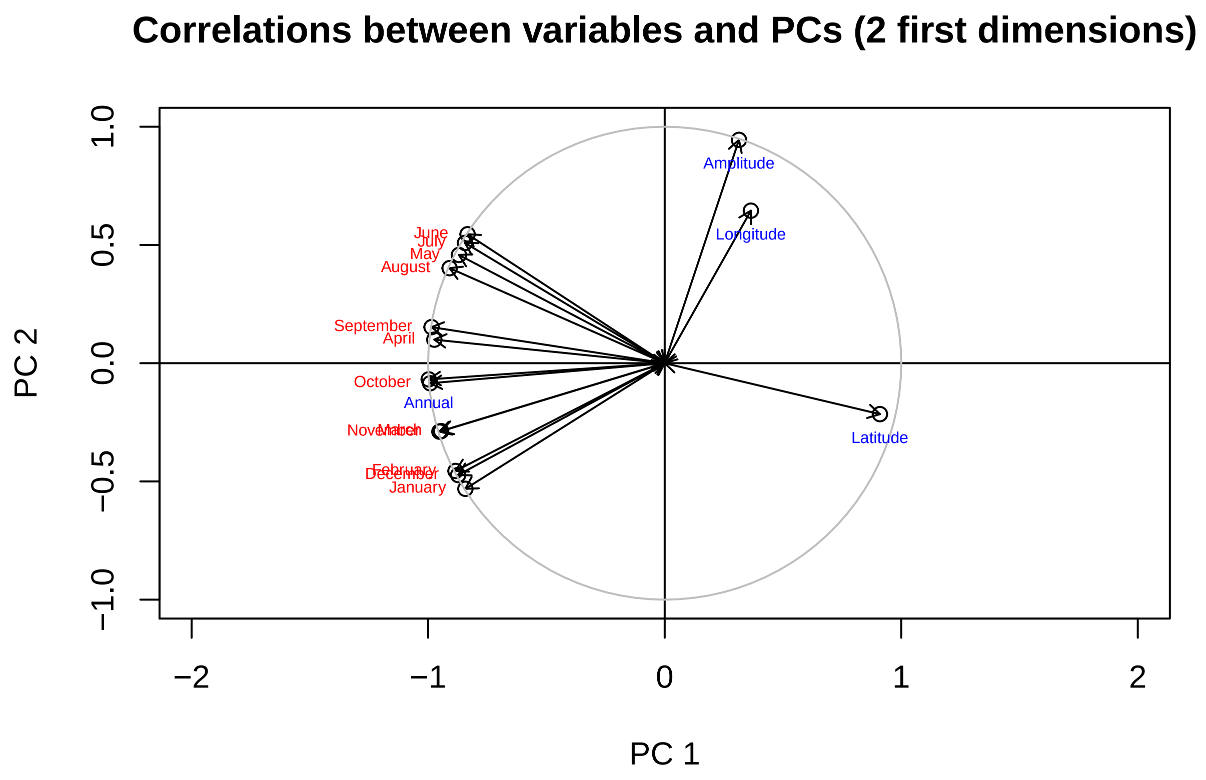
```
##                 PC1       PC2       PC3       PC4
## January   -0.842451 -0.531358  0.067767 -0.011689
## February  -0.884285 -0.455833 -0.003466 -0.083715
## March     -0.945052 -0.287313 -0.120795 -0.077818
## April     -0.973888  0.099565 -0.198256 -0.024868
## May       -0.869852  0.457812 -0.156086  0.076825
## June      -0.833314  0.545322  0.049548 -0.000096
## July      -0.844163  0.508662  0.153689 -0.043604
## August    -0.909244  0.401924  0.087501 -0.044392
## September -0.985625  0.152536  0.022626 -0.005193
## October   -0.991625 -0.084765 -0.000067  0.071735
## November  -0.952357 -0.289414  0.044221  0.069737
## December  -0.873119 -0.472866  0.084808  0.068295
## Annual    -0.997548 -0.068453  0.004567  0.000004
## Amplitude  0.314076  0.944414  0.039188 -0.005742
## Latitude   0.909911 -0.215437  0.181985  0.059290
## Longitude  0.364458  0.644973 -0.036434  0.247323
```

b. Make a Circle of Correlations plot between the PCs and all the quantitative variables

(10 pts).

I will make a circle of correlation plot for PC1 and PC2 only.

```
plot(x[,1], x[,2], type = "p", main = "Correlations between variables and PCs (2 first dimensions)",
     xlab = "PC 1", ylab = "PC 2", xlim = c(-1.0, 1.0), ylim = c(-1.0, 1.0), asp = 1)

abline(h = 0, v = 0, lty = 1)
arrows(0, 0, x[,1], x[,2], length = 0.07, angle = 30, code = 3)
draw.circle(0, 0, 1, border = "gray", lty = 1, lwd = 1)

text(x[1:12,1], x[1:12,2], labels = as.vector(rownames(x))[1:12], pos = 2,
     col = "red", cex = 0.5)
text(x[13:16,1], x[13:16,2], labels = as.vector(rownames(x))[13:16], pos = 1,
     col = "blue", cex = 0.5)
```



c. Based on the above parts (a) and (b), how are the active and supplementary variables related to the components? (10 pts)

Definitely, in PC1, the variable "annual" is highly correlated with other months' temperatures. And, it makes sense, since annual variable is the average of sum of temperatures. These varaibles do not seem closely related to other three supplementary variables on PC1. So, I assume that PC1 might stand for average European temeperatures. Then, that makes sense why latitude is other side of correlation circle. (If latitude goes up, the temperatures in Europe go down, or vice versa.) On PC2, the months for winter~Spring are related with latitude, and summer~Fall's are related with amplitude and longitude. So, I can interpret PC2 as precipitation level. And, this totally makes sense. During the winter~early Spring periods, the amounts of snow differ based on latitude; on the other hand, During Summer periods, western Europe rains more often than the countries on the eastern sides.

Part 5

Our research question is "can we summarize monthly precipitations with only a small number of components?"

First of all, it is clear that we only needed to retain two or three PCs for our analysis, and these were done by examining the eigen values. Second, by looking at the correlation of a circle or correlation tables (or loadings), in PC1, I conclude that all the months and annual variables are related, but also latitude is realted to PC1 (but the other direction). For PC2, the months for winter~Spring are related with latitude, and summer~Fall's are related with amplitude and longitude. (Please refer to Part 4-c for more detailed explanations.) For PC3, there is not much of association between variables and PC. Furthermore, as I have mentioned in 3-a), supplementary cities tend to stay on the negative sides of PC1, on the other hand, active cities are staying on the positive side.

So, my answer for the research question is "YES." As you can see, even just by using PC1 and PC2, how the temepratures from each month are related with other variables. Also, as we conducted the test to see the relationship between individuals and PC and variables and PC, we can summarize monthly precipitations. I added all the details explanation how I can summarize monthly precipitations (snow and rain) in Part 4-c.

Quick recap!!! ==> First, I need to know how to decompose matrix with both eigen and SVD (either refer to my codes above for part A or slide 8 and 9 from the class - for example, 1. SVD ==> when X is standardized X = UDt(V), then UD = Z = Principal Component matrix and V = loading, and we can get Z = XV as t(V)V = I as V is orthonormal 2. Eigen ==> 1/(n-1) t(X) X = R = P D t(P) where P is orthonormal, then Z = XP - refer slide 9), and get loadings, standardized X, and principal components matrix. Also, know the relationship between eigenvalues and standard deviation/variances. Then, I need to know what methods to use to decide dimensions of PC, by computing the proportion of variation captured by each PC, and we solved it by looking at how each eigenvalue performs compared to sum of eigenvalues (remember that always! sum of eigenvalues/total inertia equals the number of variables when X is standardized! That is why each eigen value is equal to variance of each column in Z = Principal component.) The key points for 3a and 4a are we use the SAME eigen value, PCA matrix, and loadings/eigen vectors for all the times. It is because we are projecting supplementary individuals and variables onto the PCA matrix derived from active variables only! That way, projecting supplementary ones helps us interpret all the things, since they can be compared with active ones!!! So, learn how to project individuals (active and supplementary, or might not include supplementary depends on the context) on retained PCs. And, learn that we check quality of representation with cosines and the sum of each row should add up 1. Another way to check is to use contribution, and sum of all inviduals of it should give 100. For both, contribution and quality of representation, I know that supplementary invidiuals should have NA or 0 value, as they do not contribute anything, when building PCs. Understand that! After, that I need to learn relatinoship between varaibles and PCs. Also, we only check the correlation of variables and PC (or, check how big loadings are —– R^2 = squared correlation is easier to compute - refer slide 8), and we do not check the correlation between individuals and PC. If I think logically, correlation between variables and PCs make sense, but why do we need to find a correlation of each individual and PC? If some supplementary variables are categofical (like when they are not numerical variables, but dummy), we find mean of PC (lab 3). And, then to show it visually, we make a circle of correlation plot. For individual, we instead find contribution or quality of representation. (But what if we want to project supplementary individuals on PCs? Then, we need to center and scale supplementary indivials with the ones from ACTIVE!!! Then, multiply i-th supplementary individual with k-th loading to find projection of supplementary i onto k-th pc [refer slide 8]) But, why do we need PCA method (for unsupervised quantitative data/descriptive continuous variables)? The purpose of this study of PCA is to find out more important data and reduce the dimensions by taking out less important variables. So, we make new PC and rotate in terms of those new axis/PC. PC1 is the axis that spans the most variation, pc2 the second variation, ... Points that are highly expressed in some variable and not expressed in others will have a lot of variation and influence on PCs. So, when we projects cloud of our original data into new PCs, we want to minimize distorting distances/variance between individuals, and this concept is the same as our efforts of trying projected inertia as close as possible to original inertia. And, to do that, we maximize the projected inertia, yet smaller than original inertia. And, this inertia can be understood as PC for our class. So, we can definitely see that each PC is a linear combination of original column/variable. I personally found that it is better to understand how we get eigenvalue of 1/(n-1) t(X) X = R and the corresponding eigen vector (this eigen vectors compose loadings. Cuz, the eigen vectors of R is loadings.)