Midterm 1

Sebastian Griego

October 17, 2024

Problem 1: Use R to read the NOAAGlobalT.csv data file, and select the four grid boxes that cover the following four locations: San Francisco (USA), Chicago (USA), London (UK), and Tokyo (Japan). Generate a 4×6 space-time data matrix W for the June mean surface air temperature anomaly data of the four selected grid boxes and the six years from 1991 to 1996. Please output your matrix in data.frame like the one shown below, but with the correct city names for rows and years for columns. The row and column names are not part of the data matrix W, but can help describe the matrix.

```
# 1981 1982 1983 1984 1985 1986 1987
#San Diego 0.9260 -0.7491 1.2609 1.1079 -0.8644 0.9502 -0.3180
#New York -3.1299 -3.0661 1.5059 -1.7508 -1.5179 0.3121 0.2546
#Paris -0.5337 1.1086 1.7997 0.5446 -5.3362 0.1774 -5.6329
#Tokyo -1.1684 -0.3927 0.1208 -1.0764 -1.1135 -1.2139 -0.1822
```

Hint: You can download the NOAA Global Surface Temperature (NOAAGlobalT.csv) dataset from this midterm site on Canvas. Use the homework solution as a reference.

Solution:

R code:

```
setwd("C:/Users/sebas/OneDrive/Desktop/Homework/LinAlg")

data <- read.csv("NOAAGlobalT.csv", header = TRUE)

#I will find gridboxed for San Francisco, Chicago, London, and Tokyo
#SF (37.7749 N, 122.4194 W)
#Chicago (41.8781 N, 87.6298 W)
#London (51.5072 N, 0.1276 W)
```

```
#Tokyo (35.6764 N, 139.6500 E)
# The dataset uses longitude 0 to 360
# For San Francisco, 122.4194 W corresponds to 237.5806 (= 360 - 122.4194)
# For Chicago, 87.6298 W corresponds to 272.3702 (= 360 - 87.6298)
# For London, 0.1276 W corresponds to 359.8724 (= 360 - 0.1276)
# For Tokyo, 139.6500 E stays because it is east

#Find the rows
row_sf = which(data$LAT > 35 & data$LAT < 40 & data$LON > 235 & data$LON < 24
row_ch = which(data$LAT > 40 & data$LAT < 45 & data$LON > 270 & data$LON < 27
row_l = which(data$LAT > 50 & data$LAT < 55 & data$LON > 355 & data$LON < 360
row_t = which(data$LAT > 35 & data$LAT < 40 & data$LON > 135 & data$LON < 140

#Get the data and combine it
data4 = data[c(row_sf, row_ch, row_l, row_t), ]
data4[, 1:8]

#I want to find the data from 1991 to 1996
#data4[,4] corresponds to January 1880
#4 + 12 * (1991-1880)= 1336

#check
data4[1336:1341]
#this gives 1991.1 - 1996.6

#get 6 years of data 1336 + 72 = 1408
data6years = data4[,1336:1408]

#check
data6years[, 66:68]
#works

JuneOnly = data6years[ ,seq(6, 66, by = 12)]

#check
JuneOnly
#      X1991.6 X1992.6 X1993.6 X1994.6 X1995.6 X1996.6
# 1848 -1.4826  0.7795  0.9494  0.2273 -0.2399  0.6615
# 1927  2.0285 -1.4971 -1.0721  0.4447  1.3542 -0.4503
# 2088 -1.2829  1.2994  0.4849 -0.0177  0.3695  0.0444
# 1828  1.0840 -0.1687 -0.4509  0.2126 -0.4990 -0.1933

rownames(JuneOnly) = c("San Francisco", "Chicago", "London", "Tokyo")
colnames(JuneOnly) = seq(1991, 1996, by = 1)
```

```
W = JuneOnly
W
#                 1991     1992     1993     1994     1995     1996
# San Francisco -1.4826   0.7795   0.9494   0.2273  -0.2399   0.6615
# Chicago        2.0285  -1.4971  -1.0721   0.4447   1.3542  -0.4503
# London        -1.2829   1.2994   0.4849  -0.0177   0.3695   0.0444
# Tokyo          1.0840  -0.1687  -0.4509   0.2126  -0.4990  -0.1933
```

Problem 2:

a) Use the R command `image()` or another R command to visualize (i.e., to plot) the 4×6 space-time data matrix W from the previous problem. Please include your code and figure in the PDF file. Search for a proper color scheme, e.g., `col=topo.colors(64)`, for your `image()` command so that your grid boxes are clearly shown. Transpose your matrix if necessary to show space locations as rows and time stamps as columns in your figure.

b) Use R to make an SVD analysis of the 4×6 matrix W.

c) Plot EOF1, i.e., the first column of the U matrix from the SVD analysis in Part (b), on a world map, as you did in your homework, but now you have four points instead of two points. You may use the dot size to represent the data values, and use red for positive and blue for negative. Or use your own way to illustrate EOF1 values. See a sample EOF1 visualization shown in Fig1.

Solution:

a) R code:

```
# Transpose the matrix
W_t = t(W)

# Plot the matrix
image(W_t, col = topo.colors(64),
      xlab = "Time (Years)",
      ylab = "Locations",
      main = "Space-Time Data Matrix",
      axes = FALSE)

# Add axis labels
axis(1, at = seq(0, 1, length = 6),
```

```
        labels = rownames(W_t))
axis(2, at = seq(0, 1, length= 4),
        labels = colnames(W_t), las = 2)
```

Here is the plot. I use VScode, so it looks different than it would in RStudio.

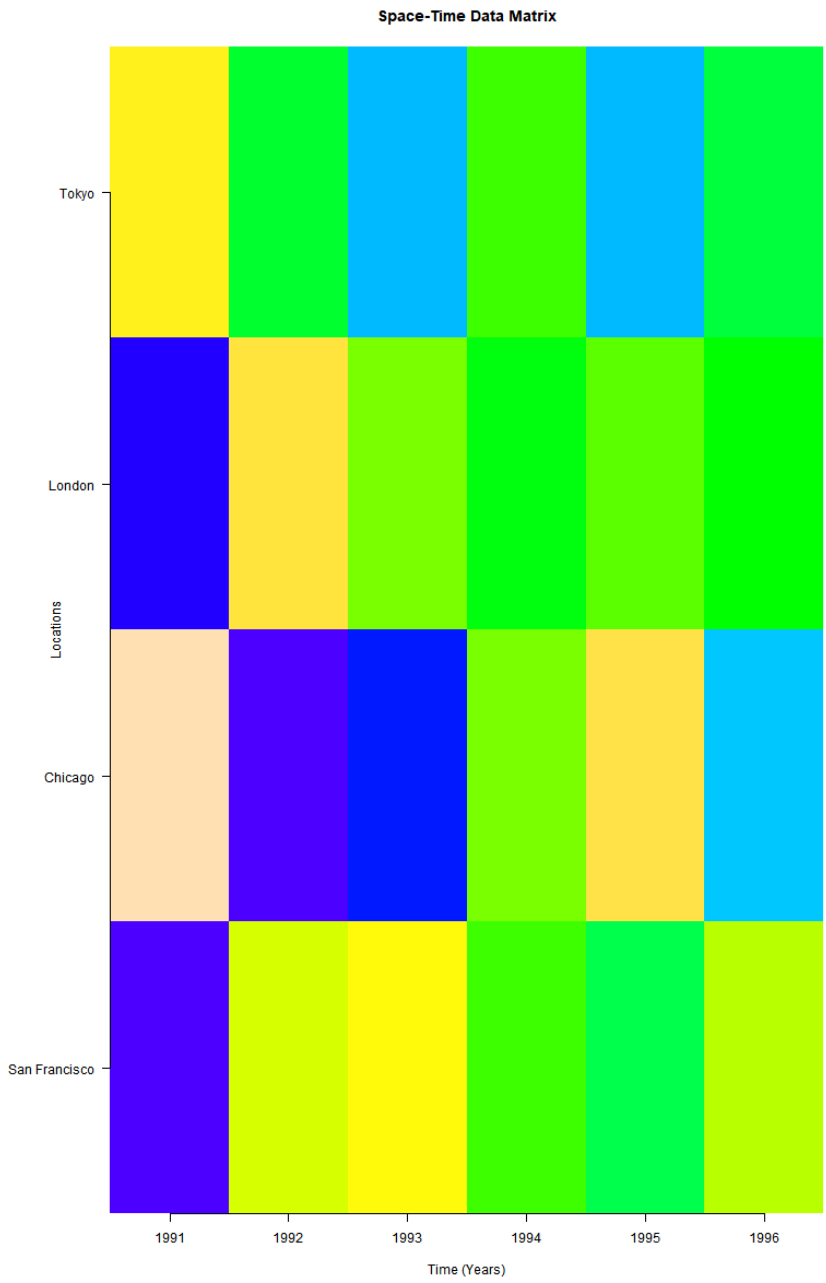


Figure 1: Space-Time Data Matrix

b) R code:

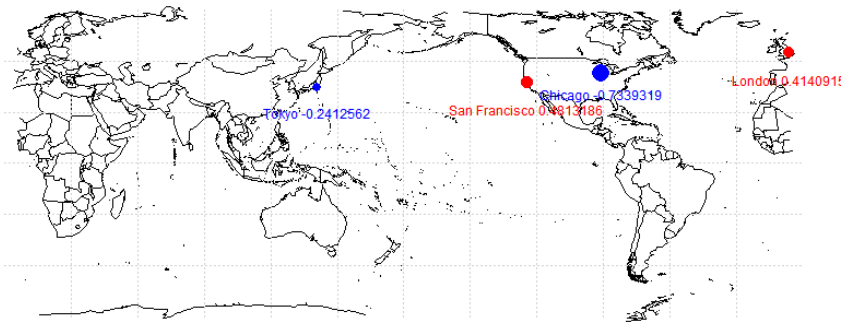
```
svdW = svd(W)

U = svdW$u
D = diag(svdW$d)
V = svdW$v

U
D
V

#> U
#             [,1]       [,2]       [,3]      [,4]
#[1,]  0.4813186 -0.1546374  0.5825904 0.6364024
#[2,] -0.7339319 -0.5973775  0.1198705 0.3001918
#[3,]  0.4140915 -0.5527022 -0.6975263 0.1910643
#[4,] -0.2412562  0.5601318 -0.3995954 0.6843766
#> D
#          [,1]     [,2]      [,3]      [,4]
#[1,] 4.108776 0.000000 0.0000000 0.0000000
#[2,] 0.000000 1.311814 0.0000000 0.0000000
#[3,] 0.000000 0.000000 0.7559272 0.0000000
#[4,] 0.000000 0.000000 0.0000000 0.5158252
#> V
#             [,1]       [,2]       [,3]       [,4]
#[1,] -0.72896255  0.25440333 -0.21020047  0.31436362
#[2,]  0.49959597 -0.02963954 -0.74647769  0.34793405
#[3,]  0.37806583 -0.02053120  0.35260708  0.12877717
#[4,] -0.06707504 -0.13106736  0.14964596  0.81474541
#[5,] -0.20345850 -0.95714823 -0.04732287 -0.03307095
#[6,]  0.17375061  0.02583673  0.49962155  0.31405427
```

c) R code:

```
eof1 = U[,1]
eof1
library(maps)
library(mapdata)

par(mar = c(0,0,0,0))
map(database = "world2Hires", ylim = c(-70, 70), mar = c(0,0,0,0))
grid(nx = 12, ny = 6)
points(237.5806, 37.7749, pch = 16, col = "red", cex = 2)
text(237.5806, 25, labels = "San-Francisco-0.4813186", col = "red")
points(272.3702, 41.8781, pch = 16, col = "blue", cex = 3)
```

```
text(272.3702, 32, labels = "Chicago -0.7339319", col = "blue")
points(359.8724, 51.5072, pch = 16, col = "red", cex = 1.9)
text(359.8724, 39, labels = "London 0.4140915", col = "red")
points(139.6500, 35.6764, pch = 16, col = "blue", cex = 1.5)
text(139.6500, 23, labels = "Tokyo -0.2412562", col = "blue")
```

Plot:



Problem 3:

a) The data file EarthTemperatureData.csv contains the monthly and annual global average temperature anomalies from 1850 to 2015. Extract the monthly data from this data matrix and convert the monthly data into a sequence, i.e., a vector, according to time order: Jan 1850, Feb 1850, Mar 1850, ... , Dec 1850, Jan 1851, Feb 1881, ..., Nov 2015, Dec 2015.

b) Plot the temperature data sequence from Jan 1901 to Dec 2000. The horizontal axis should be marked as Year from 1901 to 2000. The vertical axis is for the temperature anomaly data. See the solution for the sample midterm on Canvas for the figure style.

Solution:

a) R code:

```
setwd("C:/Users/sebas/OneDrive/Desktop/Homework/LinAlg")
earth_temp <- read.csv('EarthTemperatureData.csv', header = TRUE)

#Verify data
dim(earth_temp)
earth_temp[1:3, 1:6]

# Extract monthly data
monthly_temp = earth_temp[, 2:13]

# Verify data
monthly_temp[1:2, 1:4]
monthly_temp[1,]

# Transpose the matrix and convert to a vector
vector_temp = c(t(monthly_temp))

# Compare with original data
vector_temp[1:24]
earth_temp[1:2,]

#> vector_temp[1:24]
# [1] -0.702 -0.284 -0.732 -0.570 -0.325 -0.213 -0.128 -0.233 -0.444 -0.4
# [11] -0.190 -0.268 -0.303 -0.362 -0.485 -0.445 -0.302 -0.189 -0.215 -0
# [21] -0.108 -0.063 -0.030 -0.067
#> earth_temp[1:2,]
```

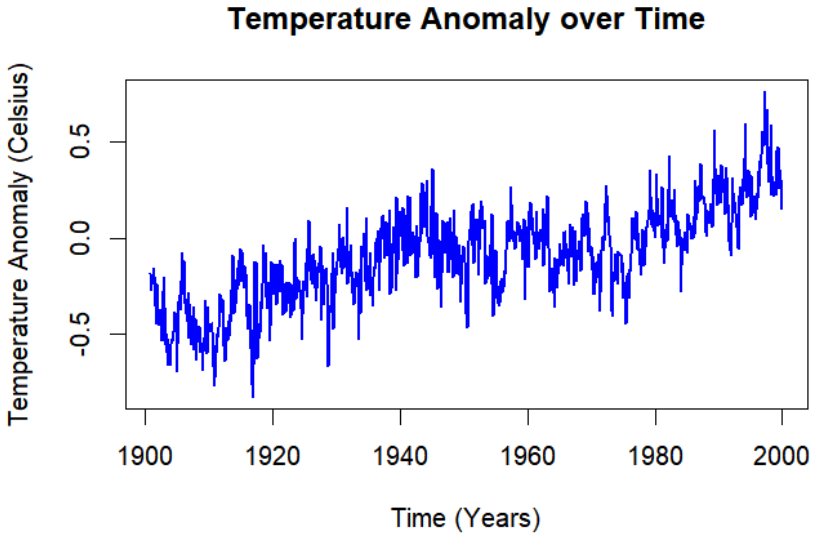
```
# YEAR    JAN    FEB    MAR    APR    MAY    JUN    JUL    AUG
SEP    OCT
# 1 1850 -0.702 -0.284 -0.732 -0.570 -0.325 -0.213 -0.128 -0.233 -0.444 -
# 2 1851 -0.303 -0.362 -0.485 -0.445 -0.302 -0.189 -0.215 -0.153 -0.108 -
#    NOV    DEC ANNUAL
# 1 -0.19 -0.268 -0.375
# 2 -0.03 -0.067 -0.223

# Check the length of the vector
length(vector_temp)
#1992
```

b) R code:

```
#The data is from 1850 to 2015, but we want 1901 to 2000.
#1901 corresponds to the 52nd year, so we need to remove the first 51 yea
#2000 corresponds to the 150th year, so we need to remove the last 15 yea

#Remove the first 612 data points and the last 180 data points
new_vector_temp = vector_temp[613:1812]

length(new_vector_temp)
#1200

#Verify
new_vector_temp[1:5]
earth_temp[52, 1:5 ]

#> new_vector_temp[1:5]
#[1] -0.182 -0.270 -0.246 -0.193 -0.197
#> earth_temp[52, 1:5 ]
#   YEAR    JAN   FEB    MAR    APR
#52 1901 -0.182 -0.27 -0.246 -0.193

new_vector_temp[1189:1200]
earth_temp[151, ]

#> new_vector_temp[1189:1200]
#[1] 0.227 0.455 0.382 0.479 0.280 0.275 0.262 0.358 0.307 0.222 0.162 0.
#> earth_temp[151, ]
#    YEAR   JAN   FEB   MAR   APR  MAY   JUN   JUL   AUG   SEP
OCT   NOV   DEC
#151 2000 0.227 0.455 0.382 0.479 0.28 0.275 0.262 0.358 0.307 0.222 0.16
#    ANNUAL
#151  0.295
```

```
#We have the correct data for the years 1901 to 2000.

time = seq(1901, 2000, len = length(new_vector_temp))

plot(time, new_vector_temp, type = "l", col = "blue", lwd = 2,
     xlab = "Time (Years)",
     ylab = "Temperature Anomaly (Celsius)",
     main = "Temperature Anomaly over Time")
```

Plot:



Problem 4:

a) State Theorem 1.1 on Page 23 of the textbook in your way.

b) Prove the theorem. You may follow the proof in the textbook by filling in all the omitted details. Or you can generate your own proof that may be very different.

c) Explain the theorem using layman's language. Use as few formulas as you can. You can draw a diagram if you think it is helpful. It is limited to 100-300 words. A figure may be counted as 50 words

Solution:

a) My statement of the theorem:

Consider a data matrix A that represents anomalies, where each row corresponds to a different location in space (N total locations) and each column corresponds to a different point in time (Y time points). C is the covariance matrix that is found by multiplying $\frac{1}{Y}AA^T$. The eigenvectors of this covariance matrix will match the spatial patterns obtained from performing SVD on A. Furthermore, if λ_k is an eigenvalue of C and d_k is an eigenvalue of A (The diagonal values of D from the SVD of A), then $\lambda_k = \frac{d_k^2}{Y}$, assuming that $Y \leq N$.

b) New Proof of Theorem 1.1

Start by expressing the anomaly data matrix A using SVD:

$$A = UDV^T$$

- U is an $N \times N$ orthogonal matrix whose columns are the left singular vectors (spatial modes).

- D is an $N \times Y$ diagonal matrix containing the singular values.

- V^T is a $Y \times Y$ orthogonal matrix whose rows are the right singular vectors.

Compute the Covariance Matrix C The covariance matrix C is defined as

$$C = \frac{1}{Y}AA^T$$

Substitute the SVD of A into this expression:

$$\begin{aligned}
C &= \frac{1}{Y}(UDV^T)(UDV^T)^T \\
&= \frac{1}{Y}(UDV^T)(VDU^T) \quad \text{Since } D^T = D \\
&= \frac{1}{Y}UD(V^TV)DU^T \\
&= \frac{1}{Y}UD(I)DU^T \\
C &= \frac{1}{Y}UD^2U^T \\
CU &= \frac{1}{Y}UD^2U^TU \\
CU &= \frac{1}{Y}UD^2
\end{aligned}$$

D is diagonal, it is commutative.

This gives:

$$CU = \frac{D^2}{Y}U$$

Here: - U contains the eigenvectors of C. - The diagonal matrix $\frac{D^2}{Y}$ contains the eigenvalues $\lambda_k = \frac{d_k^2}{Y}$.

If you break it down into its components, you get the definition of an eigenvalue and eigenvector with this relation.

The eigenvectors of the covariance matrix C are the columns of U, which are the spatial modes from the SVD of A. The corresponding eigenvalues λ_k of C satisfy $\lambda_k = \frac{d_k^2}{Y}$, establishing the relationship between the eigenvalues of C and the singular values of A.

Thus, Theorem 1.1 is proven.

c) Explantion of theorem:

Suppose A is a space-time anomaly data matrix. You can use the given formula to find the covariance matrix of A. If you do SVD on A, the columns of U are the EOFs. These EOFs are equivalent to the eigenvectors of the covariance matrix. The eigenvalues of the covariance matrix are equal to the squares of the singular values of A divided by the number of time points.

This is useful because it allows us to understand the spatial patterns in the data and the temporal patterns in the data without calculating the covariance matrix directly. You just need to do SVD on A.

Problem 5:

a) Show that the three row-vectors of the following matrix A are not linearly independent.

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

b) Columns 2, 3, and 4 of the above matrix A form a square matrix, denoted by H. Use R to compute the determinant of this matrix H. Does this matrix H have an inverse?

c) Use R to compute the inverse matrix of the following matrix B

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 0 \\ 7 & 1 & 9 & 0 \\ 0 & 1 & 0 & 8 \end{pmatrix}$$

d) Use R to compute the determinant of the matrix B.

e) Use R to find all four unit eigenvectors of B and their corresponding eigenvalues of B.

f) Are the first two unit eigenvectors from Part (e)'s results orthogonal? You must use R to do some computing to substantiate your answer.

Solution:

a) Looking at the rows of A, I see

$$2r_1 - 8r_3 = r_2$$

Therefore, the rows are not linearly independent because row 2 is a linear combination of row 1 and row 3.

b) R code:

```
H = matrix(c(2,3,4,4,6,0,0,0,1), nrow = 3, byrow = TRUE)
H

det(H)
#[1] 0

#The determinant of H is 0, so it is not invertible.
```

c) R code:

```
B = matrix(c(1, 2, 3, 4, 4, 5, 6, 0, 7, 1, 9, 0, 0, 1, 0, 8),
      nrow = 4,
      byrow = TRUE)
B

solve(B)
#> solve(B)
#              [,1]        [,2]        [,3]        [,4]
#[1,] -0.86666667  0.23333333  0.13333333  0.43333333
#[2,] -0.13333333  0.26666667 -0.13333333  0.06666667
#[3,]  0.68888889 -0.21111111  0.02222222 -0.34444444
#[4,]  0.01666667 -0.03333333  0.01666667  0.11666667
```

d) R code:

```
det(B)
#[1] -360
```

e) R code:

```
eigenB = eigen(B)
eigenB

evectorsB = eigenB$vectors
evectorsB

u1 = evectorsB[,1]
u1
#[1] -0.3147967 -0.6507493 -0.6795365 -0.1251344

norm(u1, type = "2")
#[1] 1

u2 = evectorsB[,2]
u2
#[1]  0.1563868 -0.5525792 -0.3132630  0.7563503

norm(u2, type = "2")
#[1] 1

u3 = evectorsB[,3]
u3
#[1] -0.07853434 -0.93494531  0.27385363  0.21145639

norm(u3, type = "2")
#[1] 1
```

```
u4 = evectorsB[,4]
u4
#[1]  0.818962684  0.027010169 -0.573202985 -0.002985091

norm(u4, type = "2")
#[1] 1

#All of the eigenvectors are normalized to 1, so eigenB gives unit eigenv

eigenB
#eigen() decomposition
#$values
#[1] 13.200401  7.269414  3.578543 -1.048358

#$vectors
#          [,1]       [,2]       [,3]       [,4]
#[1,] -0.3147967  0.1563868 -0.07853434  0.818962684
#[2,] -0.6507493 -0.5525792 -0.93494531  0.027010169
#[3,] -0.6795365 -0.3132630  0.27385363 -0.573202985
#[4,] -0.1251344  0.7563503  0.21145639 -0.002985091

#These are the eigenvalues their corresponding eigenvectors of B
```

f) R code:

```
library(geometry)
dot(u1, u2)
#[1] 0.4285886

#The dot product of u1 and u2 is not 0, so they are not orthogonal.
```