MAST30034 Assignment 1

Ethan Spencer 997028

25/08/2021

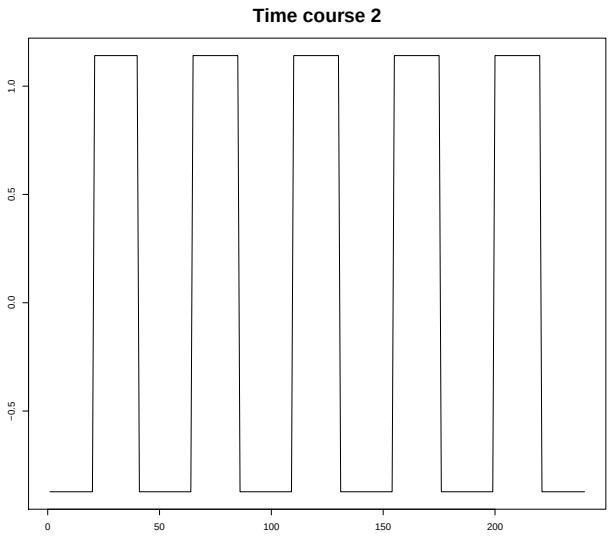
Question 1

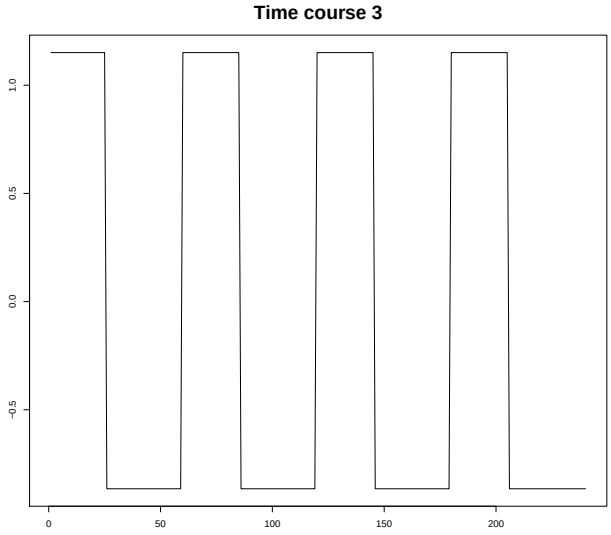
Part 1

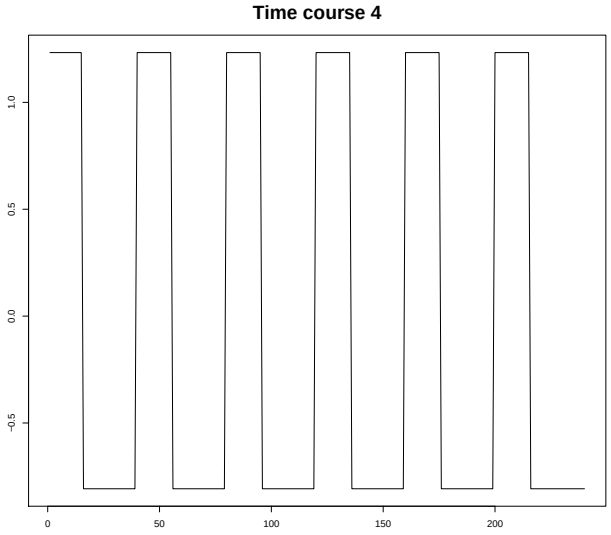
It is better to standardise because each time course would have a different variance if they were normalised.

```
# Vectors from spec
AV <- c(0, 20, 0, 0, 0, 0)
IV <- c(30, 45, 60, 40, 40, 40)
duration_ones <- c(15, 20, 25, 15, 20, 25)

# Matrix to be populated with temporal sources
TC <- matrix(rep(0, 240*6), ncol=6)

par(mfrow = c(3,2))

# Create each temporal source based on the three vectors and plot it
for (i in (1:6)){
  tc <- rep(0, 240)

  arrival <- AV[i]
  interval <- IV[i]
  duration <- duration_ones[i]

  for (j in 1:(240-arrival)){
    if (j%%interval <= duration){
      tc[j+arrival] <- 1
    }
  }

  tc[221:240] <- rep(0, 20)

  TC[, i] <- scale(tc)

  plot(TC[, i], type="l", main = paste("Time course", i),
                          xlab = "", ylab = "",
                          cex.main = 2)
}
```

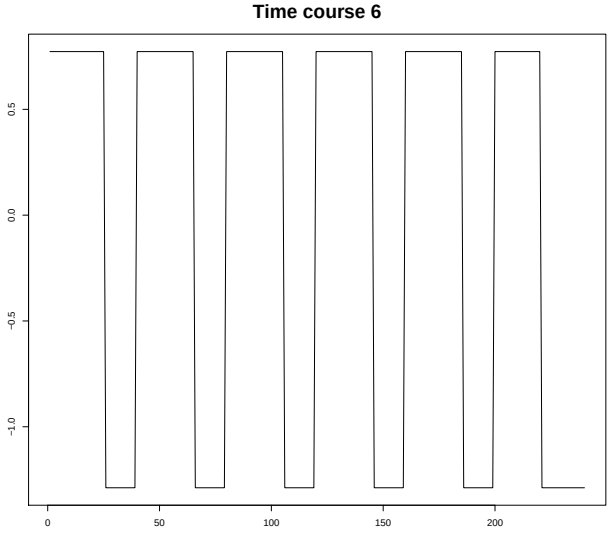

Time course 1
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200


Time course 2
1.0
0.5
0.0
−0.5
0
50
100
150
200


Time course 3
1.0
0.5
0.0
−0.5
0
50
100
150
200


Time course 4
1.0
0.5
0.0
−0.5
0
50
100
150
200

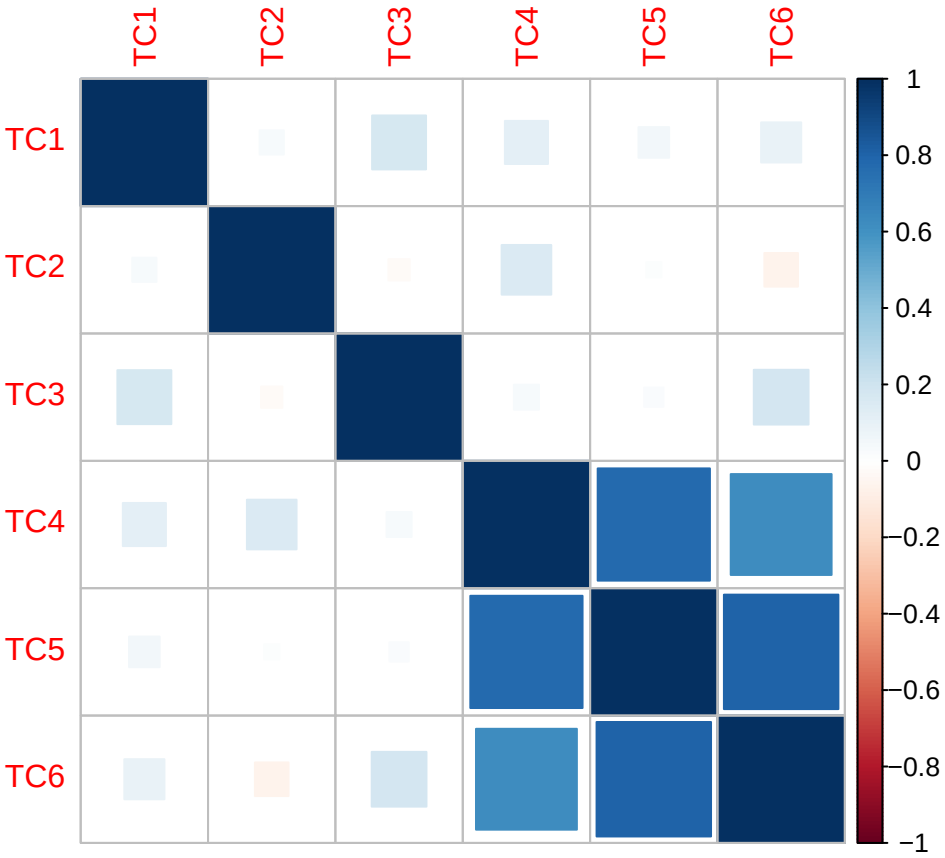

Time course 5
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200


Time course 6
0.5
0.0
−0.5
−1.0
0
50
100
150
200

Part 2

From the plot of the correlation matrix we can see that time courses four, five and six are highly correlated. In particular, four with five, and five with six. It makes sense that these time courses would be correlated as they have the same period and zero onset arrival, thus the nth series of ones in each of these series begins at the same time

```
library(corrplot)

# Convert TC to a dataframe
TCdf <- as.data.frame(TC)
colnames(TCdf) <- c("TC1", "TC2", "TC3", "TC4", "TC5", "TC6")

# Calculate and plot the correlation martrix
corrplot(cor(TCdf), method = "square")
```



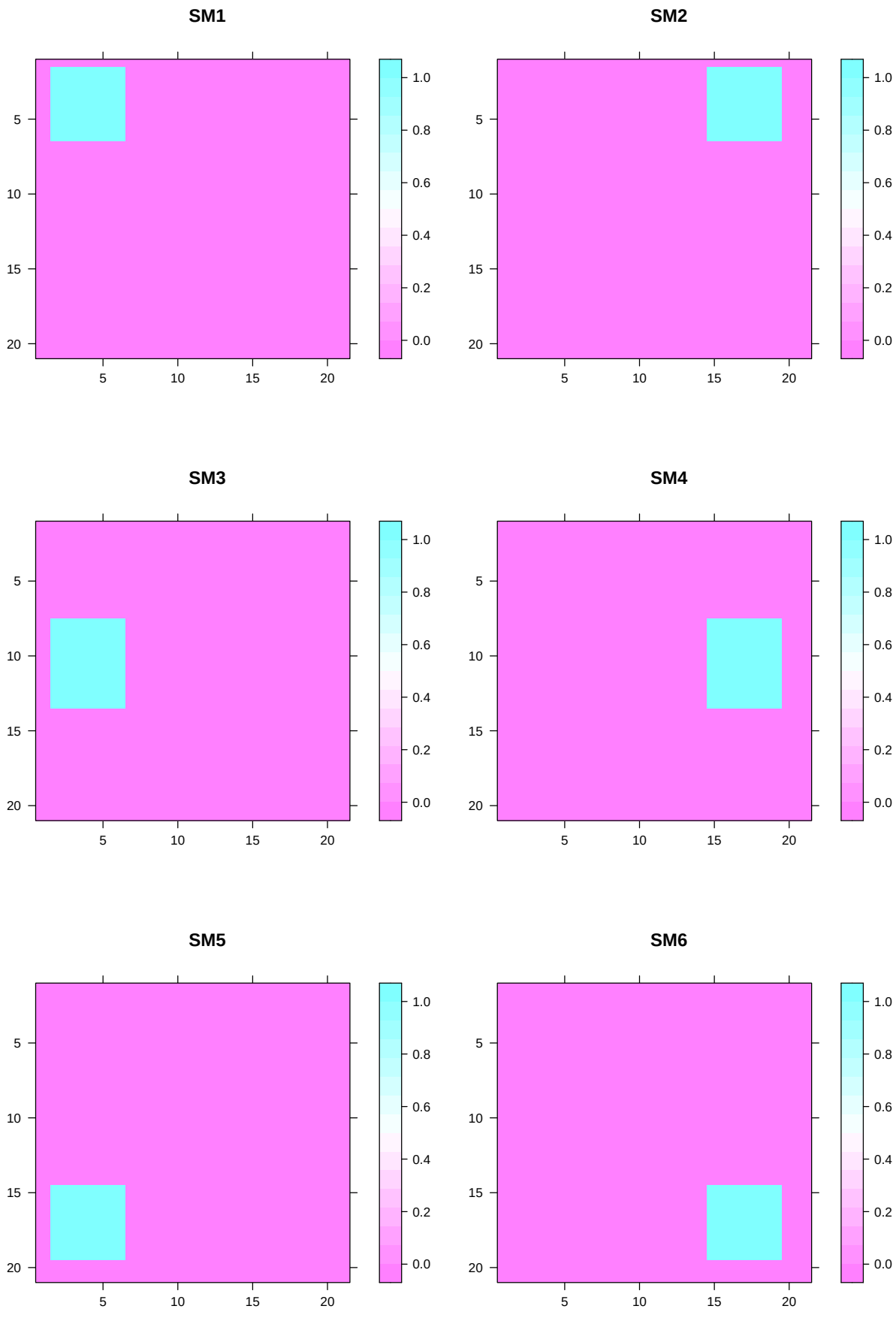
Part 3

```
library(lattice)
library(gridExtra)

# Initialise array with zeroes
tmpSM <- array(0, dim = c(6, 21, 21))

# Add ones to the array per the spec
tmpSM[1,2:6,2:6] <- 1
tmpSM[2,15:19,2:6] <- 1
tmpSM[3,2:6,8:13] <- 1
tmpSM[4,15:19,8:13] <- 1
tmpSM[5,2:6,15:19] <- 1
tmpSM[6,15:19,15:19] <- 1

# Plot each SM source
par
```

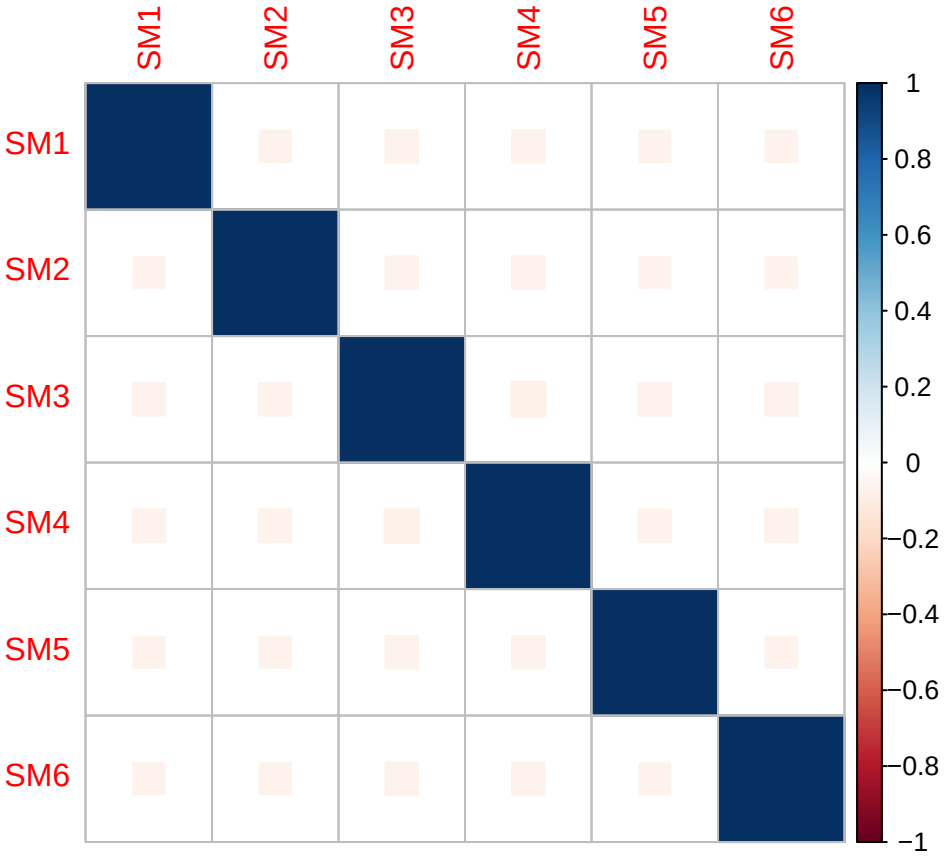
```
## function (..., no.readonly = FALSE)
## {
##     .Pars.readonly <- c("cin", "cra", "csi", "cxy", "din", "page")
##     single <- FALSE
##     args <- list(...)
##     if (!length(args))
##         args <- as.list(if (no.readonly)
##             .Pars[-match(.Pars.readonly, .Pars)]
##         else .Pars)
##     else {
##         if (all(unlist(lapply(args, is.character))))
##             args <- as.list(unlist(args))
##         if (length(args) == 1) {
##             if (is.list(args[[1L]]) | is.null(args[[1L]]))
##                 args <- args[[1L]]
##             else if (is.null(names(args)))
##                 single <- TRUE
##         }
##     }
##     value <- .External2(C_par, args)
##     if (single)
##         value <- value[[1L]]
##     if (!is.null(names(args)))
##         invisible(value)
##     else value
## }
## <bytecode: 0x7fe044029388>
## <environment: namespace:graphics>
```

```
p1 <- levelplot(tmpSM[1,,], xlab="", ylab="", main="SM1", ylim=c(21, 1))
p2 <- levelplot(tmpSM[2,,], xlab="", ylab="", main="SM2", ylim=c(21, 1))
p3 <- levelplot(tmpSM[3,,], xlab="", ylab="", main="SM3", ylim=c(21, 1))
p4 <- levelplot(tmpSM[4,,], xlab="", ylab="", main="SM4", ylim=c(21, 1))
p5 <- levelplot(tmpSM[5,,], xlab="", ylab="", main="SM5", ylim=c(21, 1))
p6 <- levelplot(tmpSM[6,,], xlab="", ylab="", main="SM6", ylim=c(21, 1))
```

```r
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=2)
```


SM1
SM2
SM3
SM4
SM5
SM6
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0

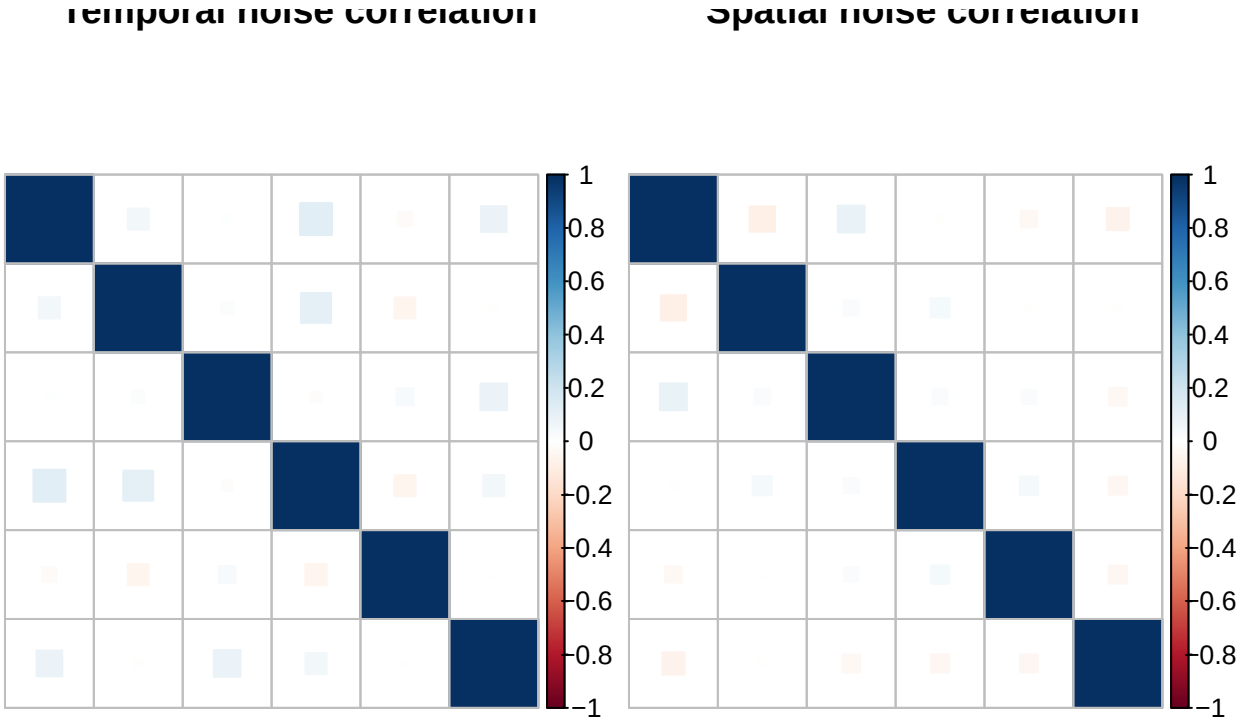
```r
library(reticulate)
# Reshape the array into a 6 by 441 matrix
SM <- array_reshape(tmpSM, dim = c(6, 441))


# Convert SM to a dataframe
SMdf <- as.data.frame(t(SM))
colnames(SMdf) <- c("SM1", "SM2", "SM3", "SM4", "SM5", "SM6")

# Calculate and plot the correlation matrix
corrplot(cor(SMdf), method = "square")
```



The correlation matrix shows that the spatial maps are uncorrelated. It is not necessary to standardise the spatial maps as any difference in pixel values would be corrected when the synthetic dataset is standardised.

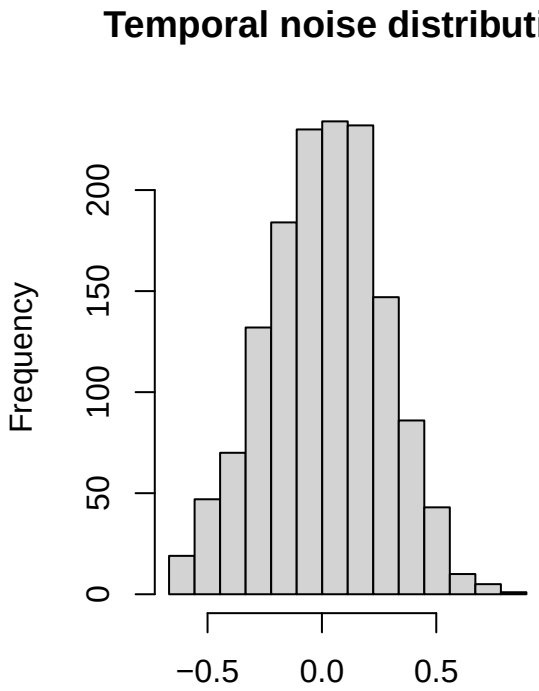
Part 4

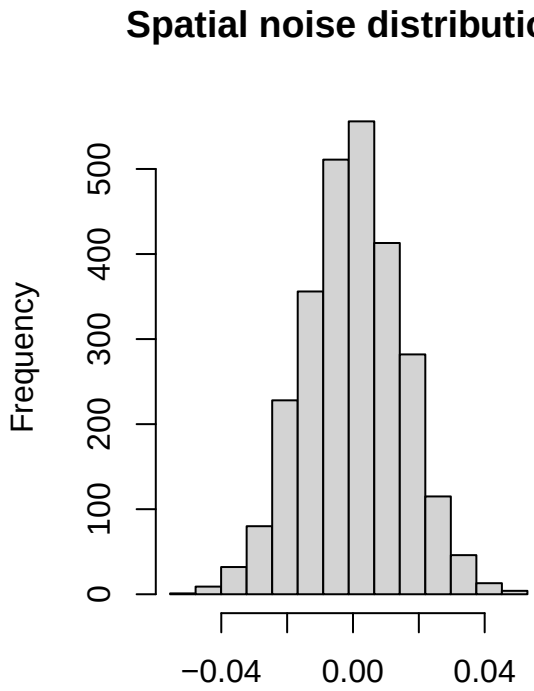
```
# Generate noise matrices
noise_t <- matrix(rnorm(240*6, sd=0.25), 240, 6)
noise_s <- matrix(rnorm(6*21*21, sd=0.015), 6, 21*21)

# Convert noise to a dataframe
ntdf <- as.data.frame(noise_t)
nsdf <- as.data.frame(t(noise_s))

par(mfrow = c(1,2))

# Calculate and plot the correlation martrices
corrplot(cor(ntdf), method = "square", tl.pos = "n",
         title = "Temporal noise correlation")

corrplot(cor(nsdf), method = "square", tl.pos = "n",
         title = "Spatial noise correlation")
```



Both noise matrices are uncorrelated across sources.

```
# Plot histogram of both noise sources
par(mfrow = c(1, 2))
hist(noise_t, breaks = seq(min(noise_t), max(noise_t), length.out = 15),
     main = "Temporal noise distribution", xlab = NA)
hist(noise_s, breaks = seq(min(noise_s), max(noise_s), length.out = 15),
     main = "Spatial noise distribution", xlab = NA)
```






```
print(paste("Temporal noise: mean =", round(mean(noise_t), 3),
            "standard deviation = ", round(sd(noise_t), 3)))
```

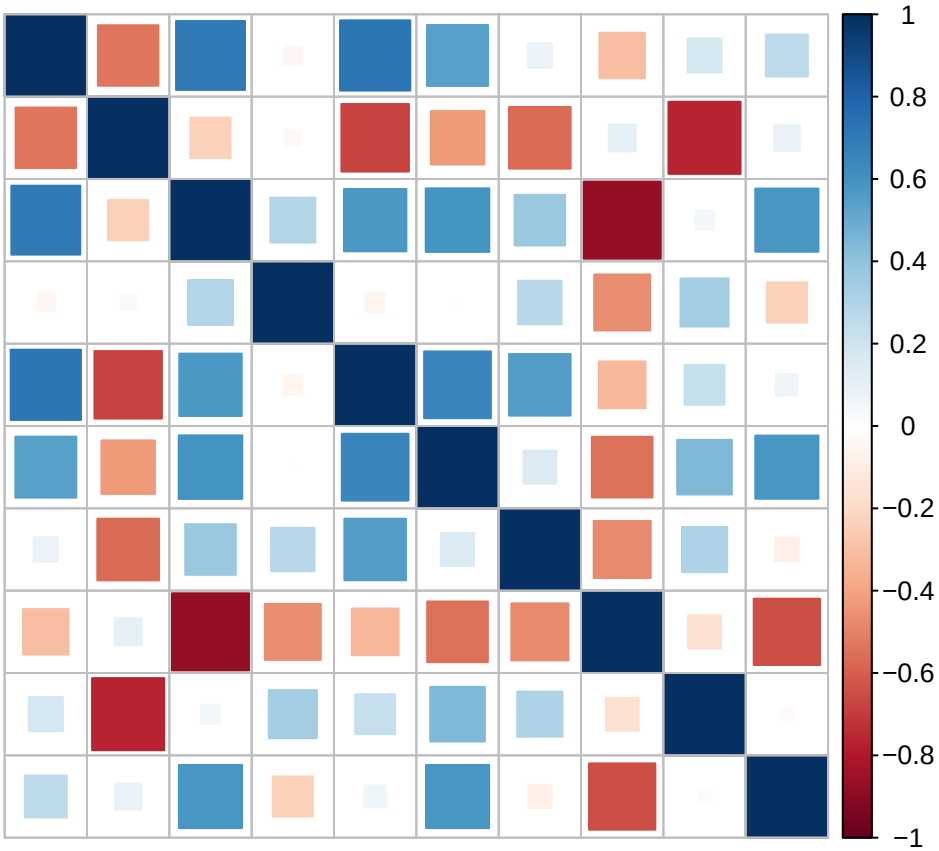
```
## [1] "Temporal noise: mean = 0.013 standard deviation =  0.255"
```

```
print(paste("Spatial noise: mean =", round(mean(noise_s), 3),
            "standard deviation = ", round(sd(noise_s), 3)))
```

```
## [1] "Spatial noise: mean = 0 standard deviation =  0.015"
```

The mean and standard deviations of the noise sources fulfill the required criteria and the histograms show an approximately normal distribution.

```
# Calculate product of noise sources
noise_prod <- noise_t%*%noise_s

# Calculate and plot correlation matrix of a subset of the correlation variables
npdf <- as.data.frame(noise_prod[,1:10])
corrplot(cor(npdf), method = "square", tl.pos = "n")
```

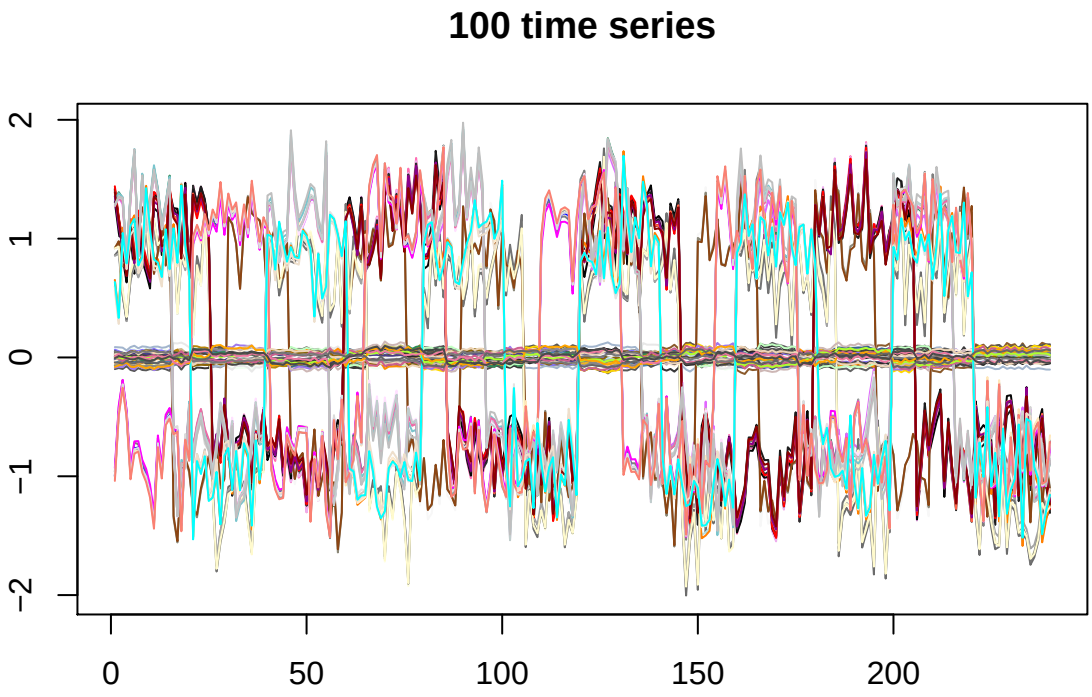


This correlation matrix shows that the $\mathbf{\Gamma}_t\mathbf{\Gamma}_s$ is correlated across variables.

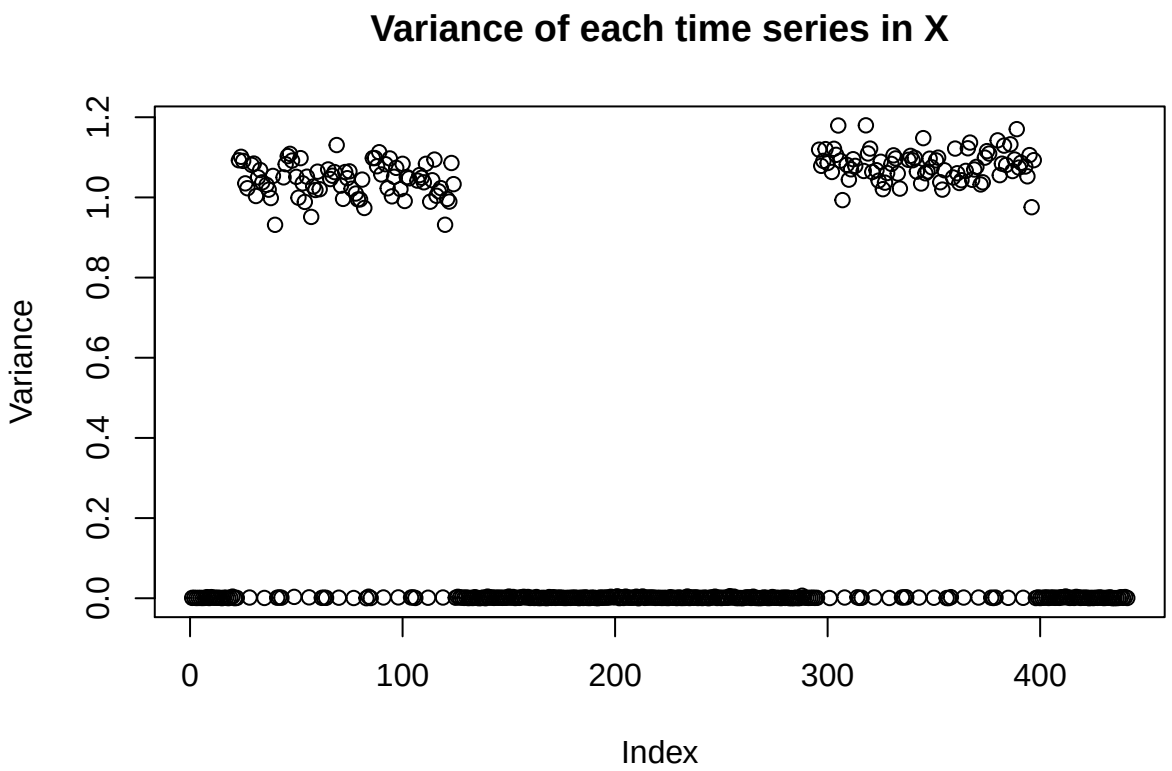
Part 5

The products $\boldsymbol{TC} \times \boldsymbol{\Gamma}_s$ and $\boldsymbol{\Gamma}_t \times \boldsymbol{SM}$ exist and in the model they are captured by the error term. i.e. $\boldsymbol{E} = \boldsymbol{TC} \times \boldsymbol{\Gamma}_s + \boldsymbol{\Gamma}_t \times \boldsymbol{SM} + \boldsymbol{\Gamma}_t\boldsymbol{\Gamma}_s$

```
# Generate synthetic dataset
X <- (TC + noise_t)%*%(SM + noise_s)

# Select random subset of 100 variables
subset <- X[, sample(1:441, 100)]

# Plot the time-series in the subset
plot(c(1, 240), c(min(subset), max(subset)), type="n",
     xlab="", ylab="", main="100 time series")

x <- 1:240
colours <- sample(colors(), 100)

for (i in 1:100){
  lines(x, subset[, i], col = colours[i])
}
```




```
# plot variances of time series
variances <- rep(0, 441)
for (i in 1:441){
  variances[i] <- var(X[, i])
}

plot(variances, main = "Variance of each time series in X", xlab = "Index", ylab = "Variance")
```



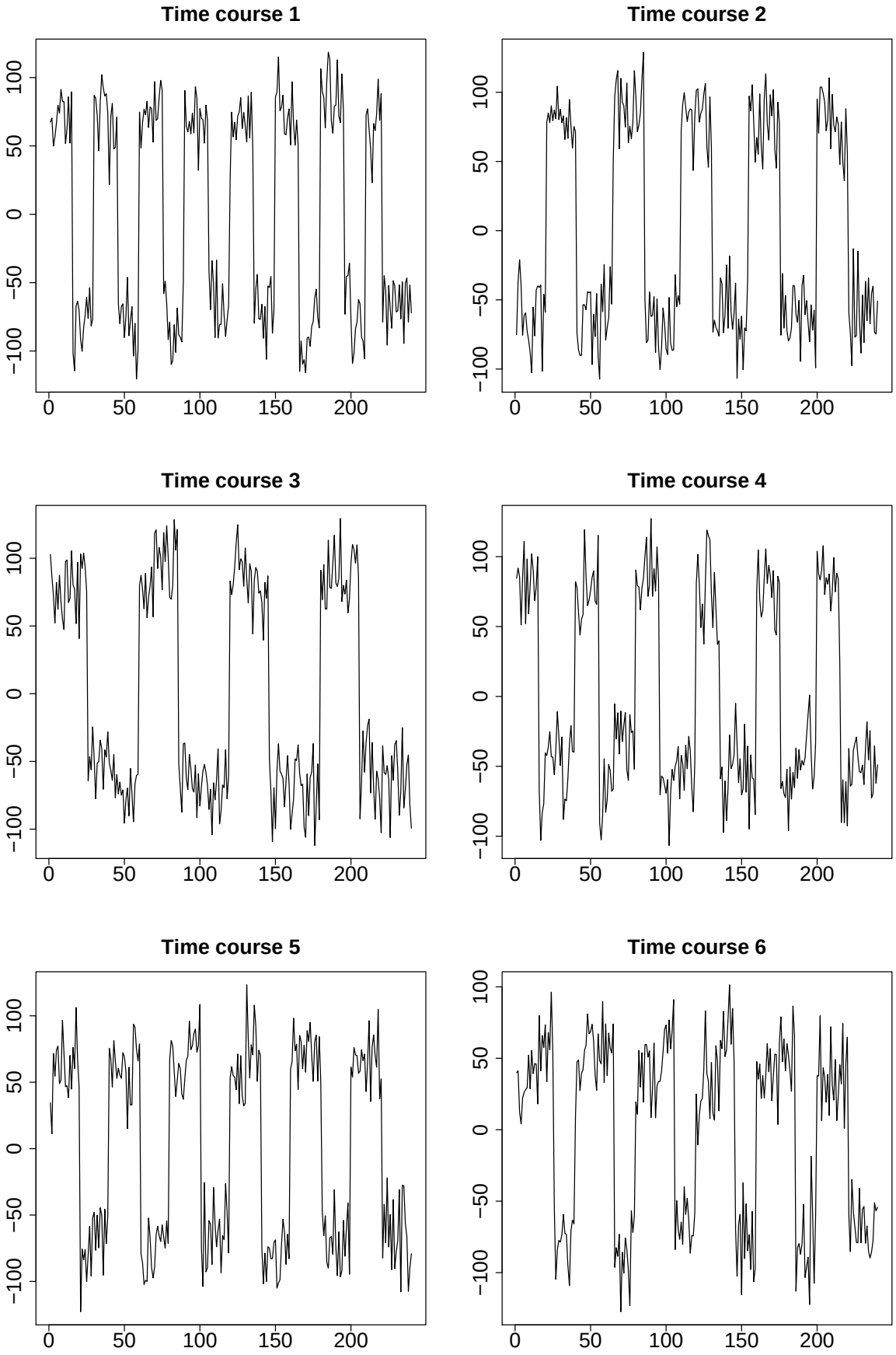
This scatter plot shows that the variance of most of the time series in $\boldsymbol{X}$ are close to 0. This is to be expected as most of the elements of $\boldsymbol{SM}$ are 0. Additionally, there are two distinct clusters of time-series with non-zero variance. This occurs because all the non-zero elements in the original spatial maps are between columns 2 and 6, and columns 15 and 19.

```
# Standardize columns of X
for (i in 1:441){
  X[, i] <- scale(X[, i])
}
```

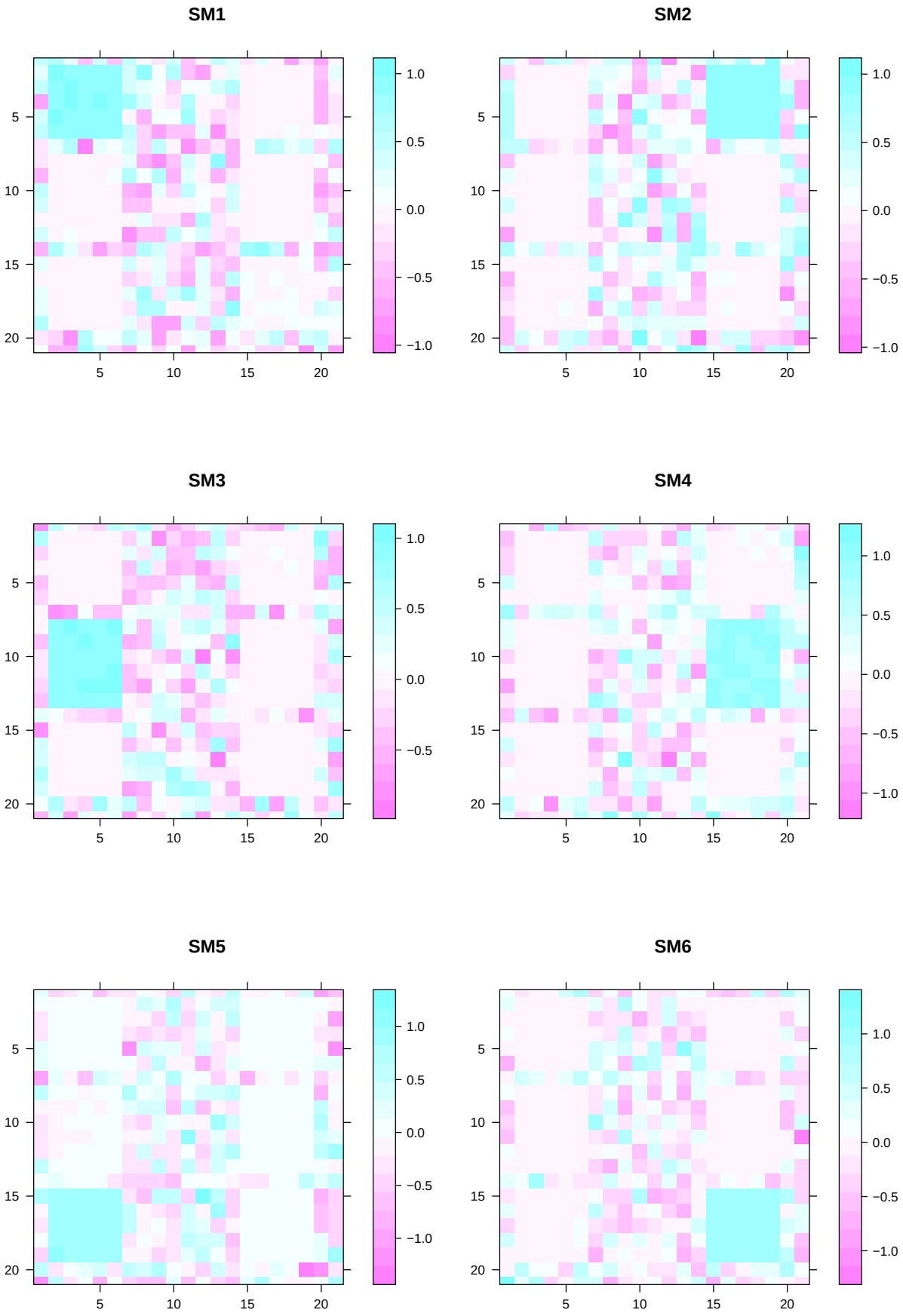
Question 2

Part 1

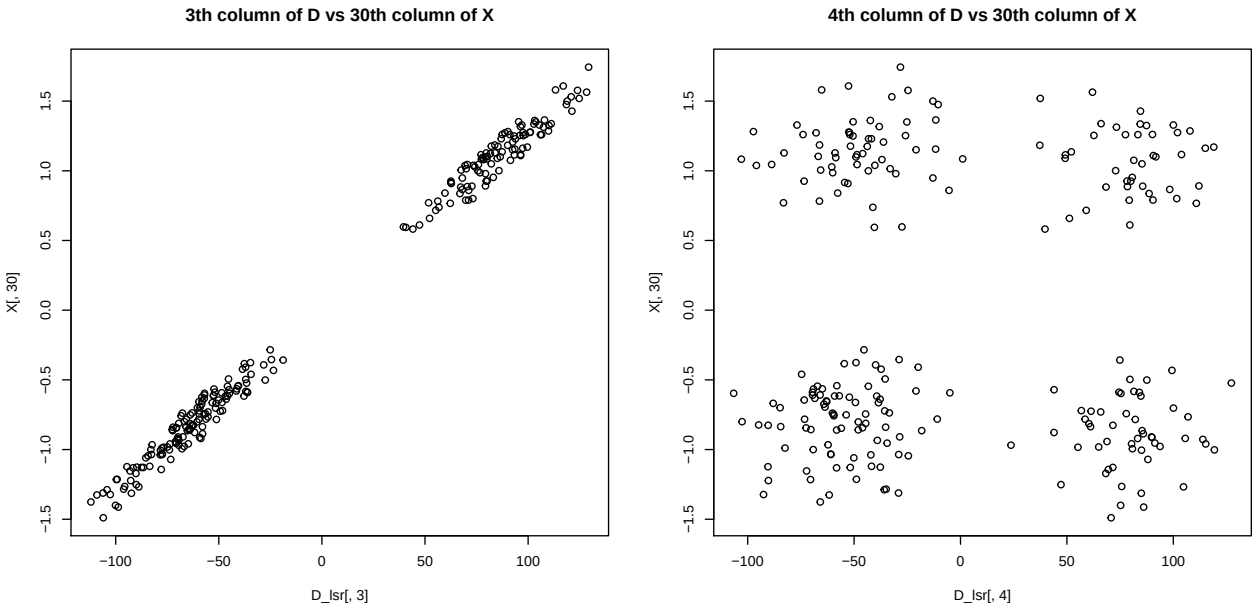
```
# Estimate A and D using least squares
A_lsr <- solve(t(TC)%*%TC)%*%t(TC)%*%X
D_lsr <- X%*%t(A_lsr)

# dim(A_lsr) <- c(6, 21, 21)
A_lsr_reshaped <- array_reshape(A_lsr, dim = c(6, 21, 21))
par(mfrow = c(3,2))

# Plot each retrieved time course
for (i in (1:6)){
  plot(D_lsr[, i], type="l", main = paste("Time course", i),
                             xlab = "", ylab = "",
                             cex.main = 2, cex.axis = 2)
}
```


Time course 1
Time course 2
Time course 3
Time course 4
Time course 5
Time course 6
100
50
0
−50
−100
0
50
100
150
200

```
# Plot each retrieved spatial map
p1 <- levelplot(A_lsr_reshaped[1,,], xlab="", ylab="", main="SM1", ylim=c(21, 1))
p2 <- levelplot(A_lsr_reshaped[2,,], xlab="", ylab="", main="SM2", ylim=c(21, 1))
p3 <- levelplot(A_lsr_reshaped[3,,], xlab="", ylab="", main="SM3", ylim=c(21, 1))
p4 <- levelplot(A_lsr_reshaped[4,,], xlab="", ylab="", main="SM4", ylim=c(21, 1))
p5 <- levelplot(A_lsr_reshaped[5,,], xlab="", ylab="", main="SM5", ylim=c(21, 1))
p6 <- levelplot(A_lsr_reshaped[6,,], xlab="", ylab="", main="SM6", ylim=c(21, 1))

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=2)
```


SM1
SM2
SM3
SM4
SM5
SM6
5
10
15
20
1.0
0.5
0.0
−0.5
−1.0

```r
# Scatter plot of 3rd column of D and 30th column of X
plot(D_lsr[, 3], X[, 30], main="3th column of D vs 30th column of X")

# Scatter plot of 3rd column of D and 30th column of X
plot(D_lsr[, 4], X[, 30], main="4th column of D vs 30th column of X")
```



Because the shape map array was reshaped in a column-wise fashion, the 30th column of $\boldsymbol{X}$ corresponds to the signal at position (9, 2). This explains why there is a strong linear correlation between column 30 $\boldsymbol{X}$ and column 3 of $\boldsymbol{D}_{lsr}$, but no correlation with column 4 of $\boldsymbol{D}_{lsr}$. The third spatial map is the only one with a one in this position, hence, the 3rd time course is the only signal contributing to the series in this position.

Part 2

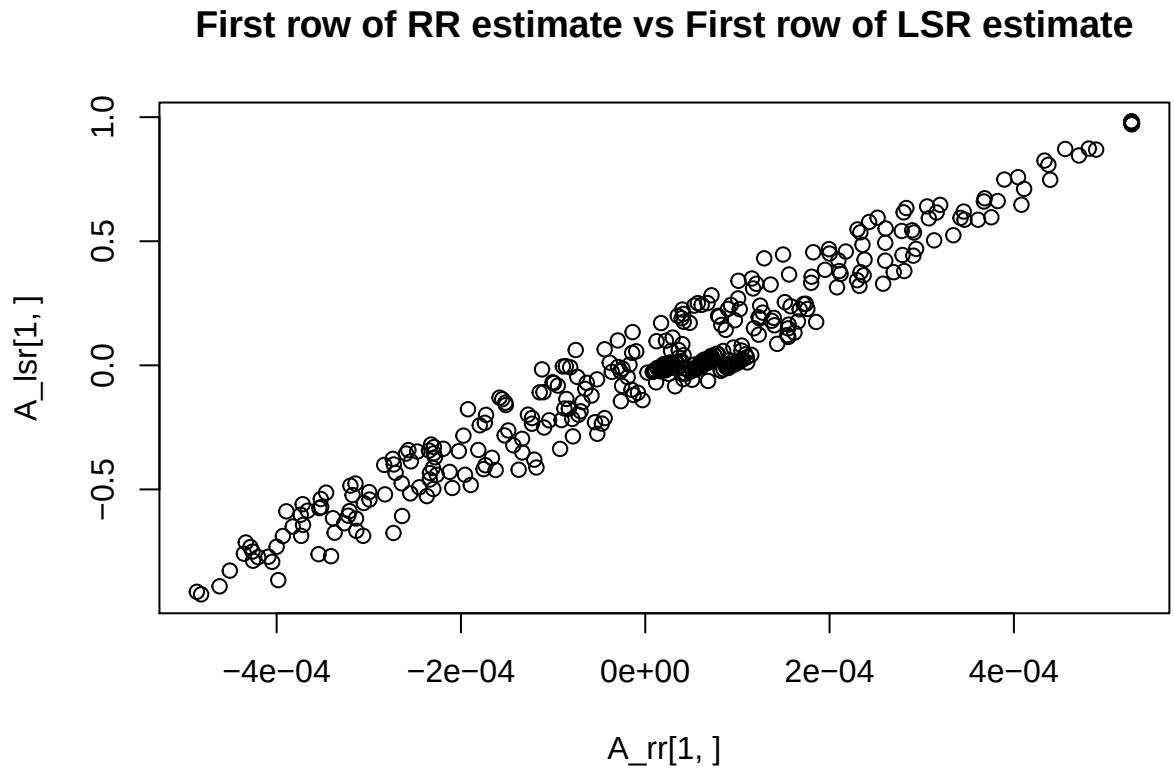
```
library(matrixStats)

# Calculate ridge regression estimates
lambda <- 441*0.11
A_rr <- solve(t(TC)%*%TC + lambda*diag(6))%*%t(TC)%*%X
D_rr <- X%*%t(A_rr)

# Compute correlation vectors
c_tlsr <- rowMaxs(abs(cor(TC, D_lsr)))
c_trr <- rowMaxs(abs(cor(TC, D_rr)))

# Calculate sum of correlation vectors
sum(c_trr)
```

```
## [1] 5.806246
```

```
sum(c_tlsr)
```

```
## [1] 5.742089
```

```
# Calculate ridge regression estimates for lambda equals 1000 and plot first
# of A_rr vs first row of A_lsr
lambda <- 441*1000

A_rr <- solve(t(TC)%*%TC + lambda*diag(6))%*%t(TC)%*%X
D_rr <- X%*%t(A_rr)

plot(A_rr[1, ], A_lsr[1, ],
     main="First row of RR estimate vs First row of LSR estimate")
```

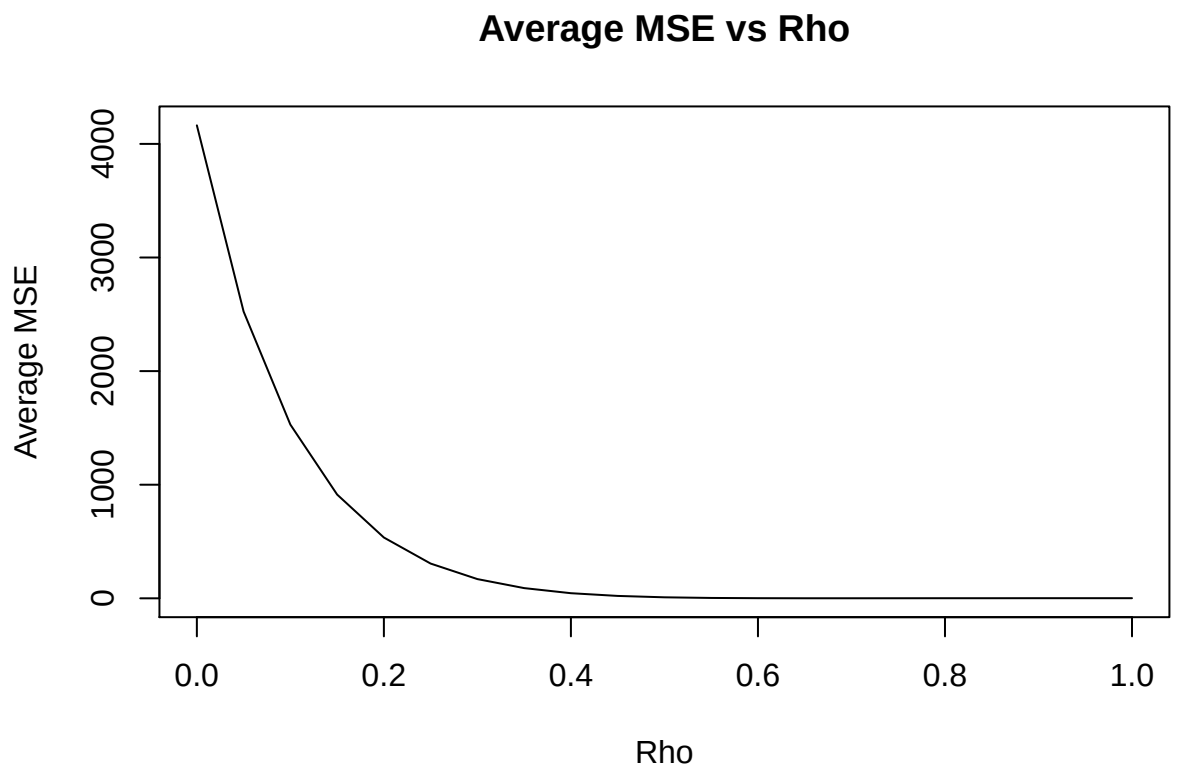


This plot shows that with $\lambda = 1000$, the values of $\boldsymbol{a}_{RR}^1$ shrink to zero.

Part 3

```
nsrcs <- 6
N <- 240
V <- 441
x1 <- 21
x2 <- 21


MSE <- rep(0, 21)

for (i in 1:10){
  # Generate noise matrices
  noise_t <- matrix(rnorm(240*6, sd=0.25), 240, 6)
  noise_s <- matrix(rnorm(6*21*21, sd=0.015), 6, 21*21)

  # Generate synthetic dataset
  X <- (TC + noise_t)%*%(SM + noise_s)

  # Standardize columns of X
  for (i in 1:441){
    X[, i] <- scale(X[, i])
  }

  for (j in 1:21){
    rho <- (j-1)*0.05


    # Calculate LR estimate (Code from spec)
    step <- 1/(norm(TC %*% t(TC)) * 1.1)
    thr <- rho*N*step
    Ao <- matrix(0, nsrcs, 1)
    A <- matrix(0, nsrcs, 1)
    A_lr <- matrix(0, nsrcs, x1*x2)

    for (k in 1:(x1*x2)) {
      A <- Ao+step*(t(TC) %*% (X[,k]-(TC%*%Ao)))
      A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))

      for (i in 1:10) {
        Ao <- A
        A <- Ao+step * (t(TC)%*%(X[,k]-(TC%*%Ao)))
        A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))
      }
      A_lr[,k] <- A
    }

    # Estimate D
    D_lr <- X%*%t(A_lr)

    # Calculate MSE

    MSE[j] <- MSE[j] + sum((X - D_lr%*%A_lr)^2)/(N*V)
  }
```

```
}

# Calculate average MSE for each rho
MSE <- MSE/10

# Plot average MSE vs rho
plot(seq(0, 1, 0.05), MSE, type = 'l',
     main = "Average MSE vs Rho", xlab = "Rho", ylab = "Average MSE")
```




```
# Calculate rho that minimizes MSE
(rho <- (which.min(MSE)-1)*0.05)
```

```
## [1] 0.7
```

The value of ρ that minimised average MSE is 0.7, for values of ρ greater than this, MSE began to increase again. Give that this value was found to minimise MSE over several trials it is appropriate to select as it is unlikely to be overfit to noise.

Part 4

```
# Calculate LR estimate (Code from spec)
step <- 1/(norm(TC %*% t(TC)) * 1.1)
thr <- rho*N*step
Ao <- matrix(0, nsrcs, 1)
A <- matrix(0, nsrcs, 1)
A_lr <- matrix(0, nsrcs, x1*x2)

for (k in 1:(x1*x2)) {
  A <- Ao+step*(t(TC) %*% (X[,k]-(TC%*%Ao)))
  A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))

  for (i in 1:10) {
    Ao <- A
    A <- Ao+step * (t(TC)%*%(X[,k]-(TC%*%Ao)))
    A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))
  }
  A_lr[,k] <- A
}

# Estimate D_lr
D_lr <- X%*%t(A_lr)


# Compute correlation vectors
c_tlr <- rowMaxs(abs(cor(TC, D_lr)))
c_trr <- rowMaxs(abs(cor(TC, D_rr)))
c_slr <- rowMaxs(abs(cor(t(SM), t(A_lr))))
c_srr <- rowMaxs(abs(cor(t(SM), t(A_rr))))

# Calculate sums of correlation vectors
sum(c_tlr)
```

```
## [1] 5.829982
```

```
sum(c_trr)
```

```
## [1] 5.644746
```

```
sum(c_slr)
```

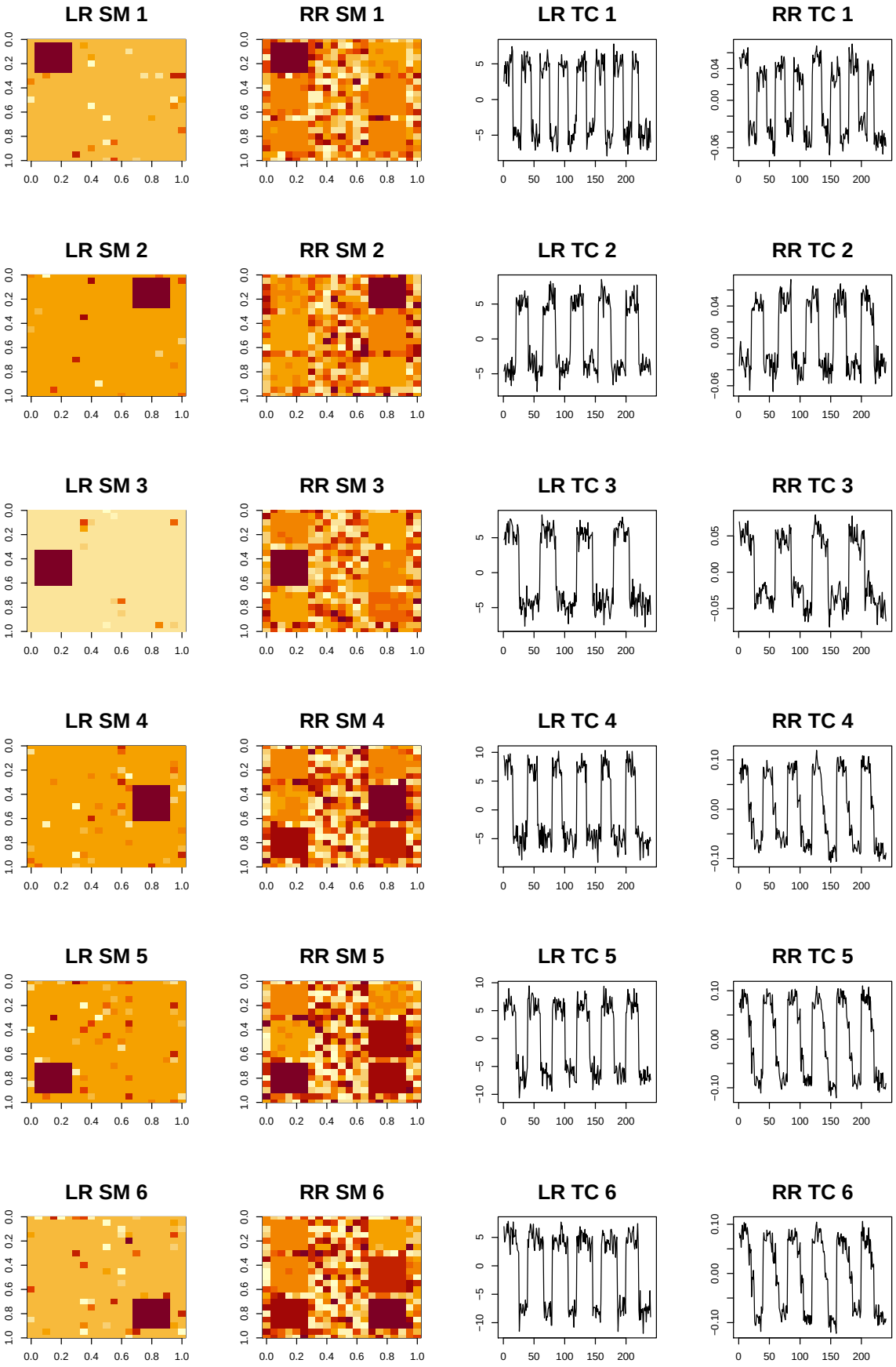
```
## [1] 5.422784
```

```
sum(c_srr)
```

```
## [1] 2.855429
```

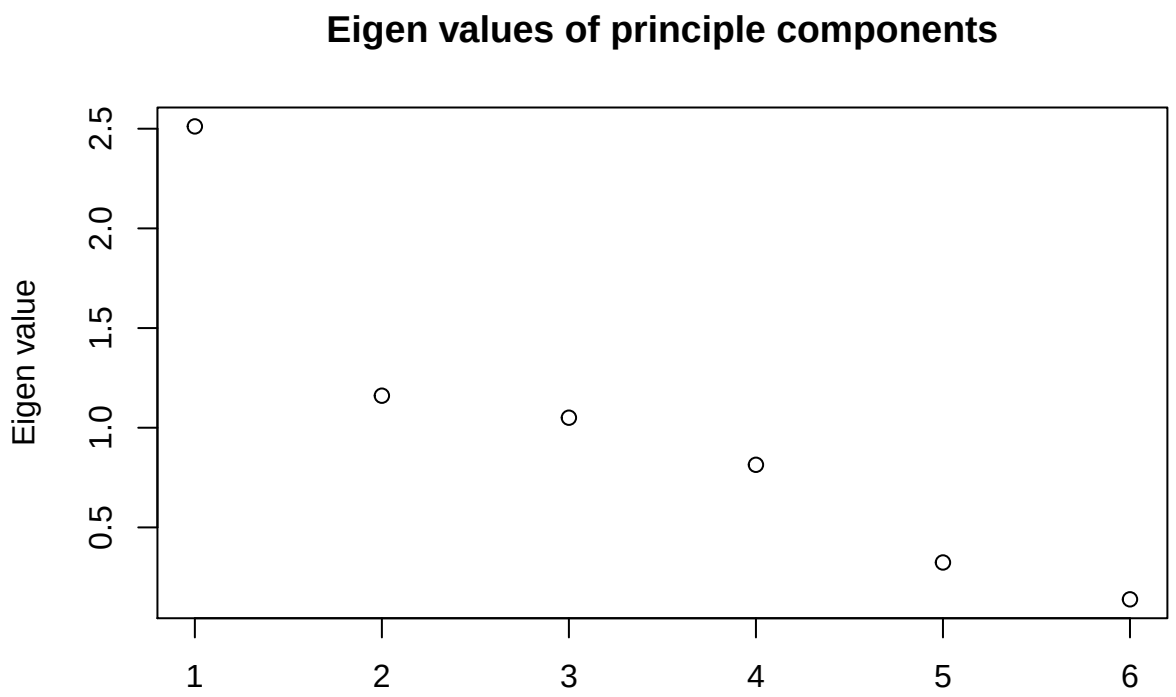
The retrieved spatial maps for Lasso regression have far fewer false positives than the spatial maps from Ridge regression, this is a result of the tendency of Lasso regression to set insignificant parameters to 0.

```
par(mfrow = c(6, 4))

# Plot retrieved spatial maps and time courses from ridge regression and lasso regression
for (i in 1:6){
  image(array_reshape(A_lr, dim = c(6, 21, 21))[i,,], main = paste("LR SM", i),
        cex.main = 2, ylim = c(1, 0))
  image(array_reshape(A_rr, dim = c(6, 21, 21))[i,,], main = paste("RR SM", i),
        cex.main = 2, , ylim = c(1, 0))

  plot(D_lr[, i], type="l", main = paste("LR TC", i), cex.main = 2,
                            xlab = "", ylab = "")
  plot(D_rr[, i], type="l", main = paste("RR TC", i), cex.main = 2,
                            xlab = "", ylab = "")

}
```


LR SM 1
RR SM 1
LR TC 1
RR TC 1
LR SM 2
RR SM 2
LR TC 2
RR TC 2
LR SM 3
RR SM 3
LR TC 3
RR TC 3
LR SM 4
RR SM 4
LR TC 4
RR TC 4
LR SM 5
RR SM 5
LR TC 5
RR TC 5
LR SM 6
RR SM 6
LR TC 6
RR TC 6

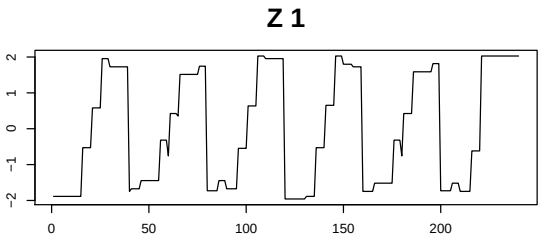
Part 5

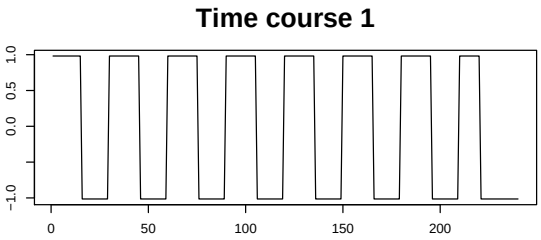
```
# Calculate principle components
pc <- prcomp(TC, retx = TRUE)

# Extract eigen values
e <- pc$sdev^2

# Plot eigen values
plot(e, main = 'Eigen values of principle components',
     xlab = "", ylab = "Eigen value")
```

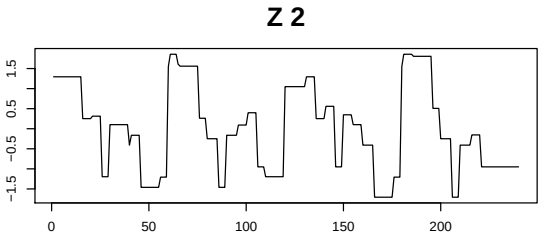


The sixth principle component by definition captures the least variance and thus has the lowest eigen value. The transformed regressors are a linear combination of the original time courses, and this process destroys their shape.

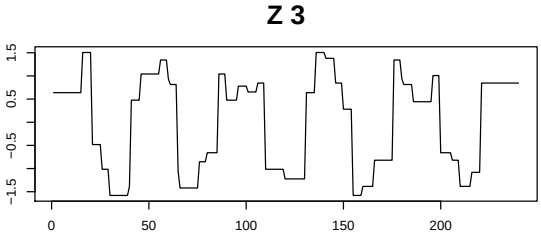
```
# Extract transformed regressors
Z <- pc$x

# Plot PC regressors and original time sources
par(mfrow = c(6, 2))
for (i in 1:6){
  plot(Z[,i], type = "l", main = paste("Z", i), cex.main = 2,
       xlab = "", ylab = "")
  plot(TC[, i], type="l", main = paste("Time course", i),
                          xlab = "", ylab = "",
                          cex.main = 2)


}
```


Z 1
2
1
0
−1
−2
0
50
100
150
200

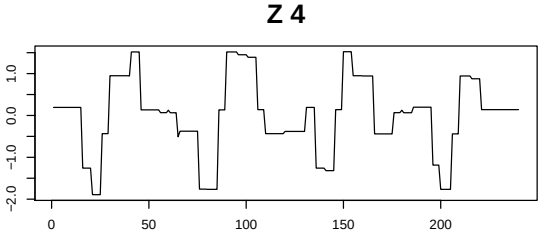

Time course 1
1.0
0.5
0.0
−1.0
0
50
100
150
200


Z 2
1.5
0.5
−0.5
−1.5
0
50
100
150
200

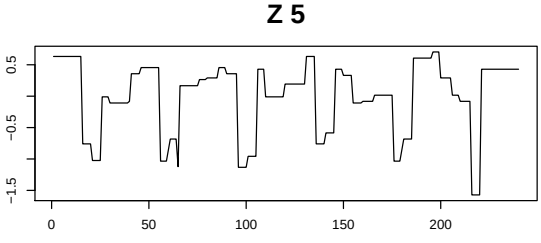

Time course 2
1.0
0.5
0.0
−0.5
0
50
100
150
200


Z 3
1.5
0.5
−0.5
−1.5
0
50
100
150
200

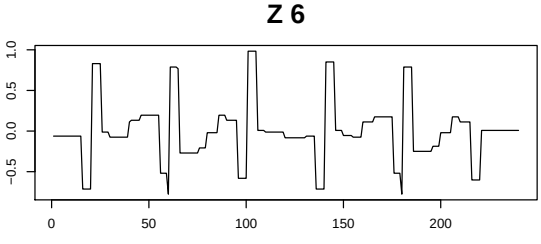

Time course 3
1.0
0.5
0.0
−0.5
0
50
100
150
200


Z 4
1.0
0.0
−1.0
−2.0
0
50
100
150
200


Time course 4
1.0
0.5
0.0
−0.5
0
50
100
150
200


Z 5
0.5
−0.5
−1.5
0
50
100
150
200


Time course 5
1.0
0.5
0.0
−1.0
0
50
100
150
200

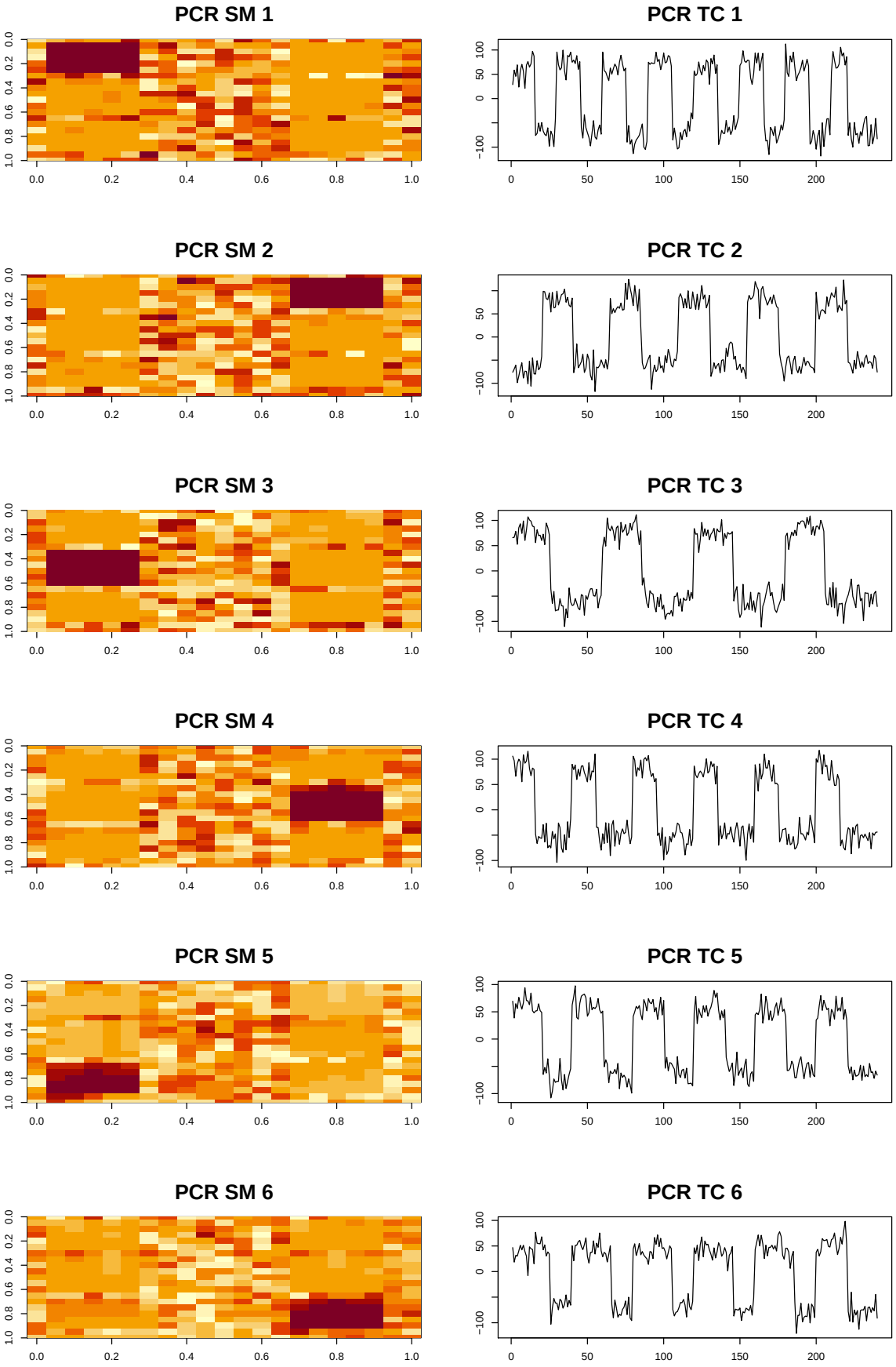

Z 6
1.0
0.5
0.0
−0.5
0
50
100
150
200


Time course 6
0.5
0.0
−1.0
0
50
100
150
200

```
# Apply lasso regression with Z
rho <- 0.001

step <- 1/(norm(Z %*% t(Z)) * 1.1)
thr <- rho*N*step
Ao <- matrix(0, nsrcs, 1)
A <- matrix(0, nsrcs, 1)
A_pcr <- matrix(0, nsrcs, x1*x2)

for (k in 1:(x1*x2)) {
  A <- Ao+step*(t(Z) %*% (X[,k]-(Z%*%Ao)))
  A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))

  for (i in 1:10) {
    Ao <- A
    A <- Ao+step * (t(Z)%*%(X[,k]-(Z%*%Ao)))
    A <- (1/(1+thr)) * (sign(A)*pmax(replicate(nsrcs, 0), abs(A)-thr))
  }
  A_pcr[,k] <- A
}

# Apply inverse principle component transformation
A_pcr <- t(t(A_pcr)%*%solve(pc$rotation))

# Estimate D_pcr
D_pcr <- X%*%t(A_pcr)
```

The results from principle component regression are inferior to the results from standard lasso regression, however from the following plots it is not obvious to tell that they are inferior to ridge or least squares regression. Additionally, it is only obvious that they are inferior in extracting the source spatial map. A possible explanation for this reduced performance is that in the principle component space the regressors are not spatially independent. It is also interesting to note that the performance deteriorates for each spatial map source (spatial maps 1 and 2 are retrieved more accurately than spatial maps 5 and 6).

```
# Plot Results of PCR
par(mfrow = c(6, 2))
for (i in 1:6){
  image(array_reshape(A_pcr, dim = c(6, 21, 21))[i,,], main = paste("PCR SM", i),
        cex.main = 2, ylim = c(1, 0))
  plot(D_pcr[, i], type="l", main = paste("PCR TC", i), cex.main = 2,
                           xlab = "", ylab = "")
}
```


PCR SM 1
PCR TC 1
PCR SM 2
PCR TC 2
PCR SM 3
PCR TC 3
PCR SM 4
PCR TC 4
PCR SM 5
PCR TC 5
PCR SM 6
PCR TC 6