Homework #3

Cameron Bracken

CVEN6833 Fall, 2009

Problem 1

```
 source('lib.R')

sst <- as.matrix(read.table('data/kaplan-sst-wy-1925-2003-revised.txt'))
sst <- t(matrix(sst[,3],nrow=length(unique(sst[,1])),byrow=T))
sst.lat <- as.matrix(read.table('data/kaplan-sst-wy-1925-2003-II.txt'))
sst.lon <- sst.lat[,2]; sst.lat <- sst.lat[,1]
sst.lon[sst.lon < 0] <- sst.lon[sst.lon < 0] + 360

pdsi <- as.matrix(read.table('data/pdsi-wy-1925-2003.txt'))
pdsi <- t(matrix(pdsi[,3],nrow=length(unique(pdsi[,1])),byrow=T))
pdsi.lat <- as.matrix(read.table('data/pdsi-wy-1925-2003-II.txt'))
pdsi.lon <- pdsi.lat[,2]; pdsi.lat <- pdsi.lat[,1]
pdsi.lon[pdsi.lon < 0] <- pdsi.lon[pdsi.lon < 0] + 360

pacific <- sst.lat > -20 & sst.lon >= 120 & sst.lon <= 280
atlantic <- sst.lat > -20 & sst.lat < 70 & sst.lon >= 250 & sst.lon <= 360
states <- pdsi.lat > 15 & pdsi.lat < 60 & pdsi.lon >= 230 & pdsi.lon <= 295

lon.pac <- sst.lon[pacific]
lat.pac <- sst.lat[pacific]
sst.pac <- sst[,pacific]
lon.atl <- sst.lon[atlantic]
lat.atl <- sst.lat[atlantic]
sst.atl <- sst[,atlantic]
lon.usa <- pdsi.lon[states]
lat.usa <- pdsi.lat[states]
pdsi.usa <- pdsi[,states]

glo <- my.pca(sst)
pac <- my.pca(sst.pac)
atl <- my.pca(sst.atl)
usa <- my.pca(pdsi.usa)

save(lat, lon, lat.pac, lon.pac, lat.atl, lon.atl, lat.usa, lon.usa, pac, atl,
        glo, usa, sst, sst.lat, sst.lon, sst.pac, sst.atl, pdsi.usa,
        file='output/1.Rdata')
```

Figure 1: Reading the data and calculating the statistics.

```r
my.pca <- function(x){
        s <- svd(var(x))

        #do an Eigen decomposition..
        eigv <- s$u
        eig <- s$d

        #Principal Components...
        pcs <- t(t(eigv) %*% t(x))

        #Eigen Values.. - fraction variance
        eigf <- (eig/sum(eig))

        return(list(eigv=eigv,eigf=eigf,eig=eig,pc=pcs))

}
```

Figure 2: PCA function.

Problem 1 (i)

```
> layout(rbind(c(1,2),c(3,4)))
> plot(pac$eigf[1:12], type="b", xlab="Modes",
+         ylab="Frac. Var. explained",main="Eigen Variance Spectrum")
> for(i in 1:3){
+         surf <- Tps(cbind(lon.pac-360,lat.pac),pac$eigv[,i])
+         surface(surf,xlab="Longitude", ylab="Latitude",main=sprintf("Eigen Vector %d",i))
+         map('world',add=TRUE,fill=TRUE)
+ }
```

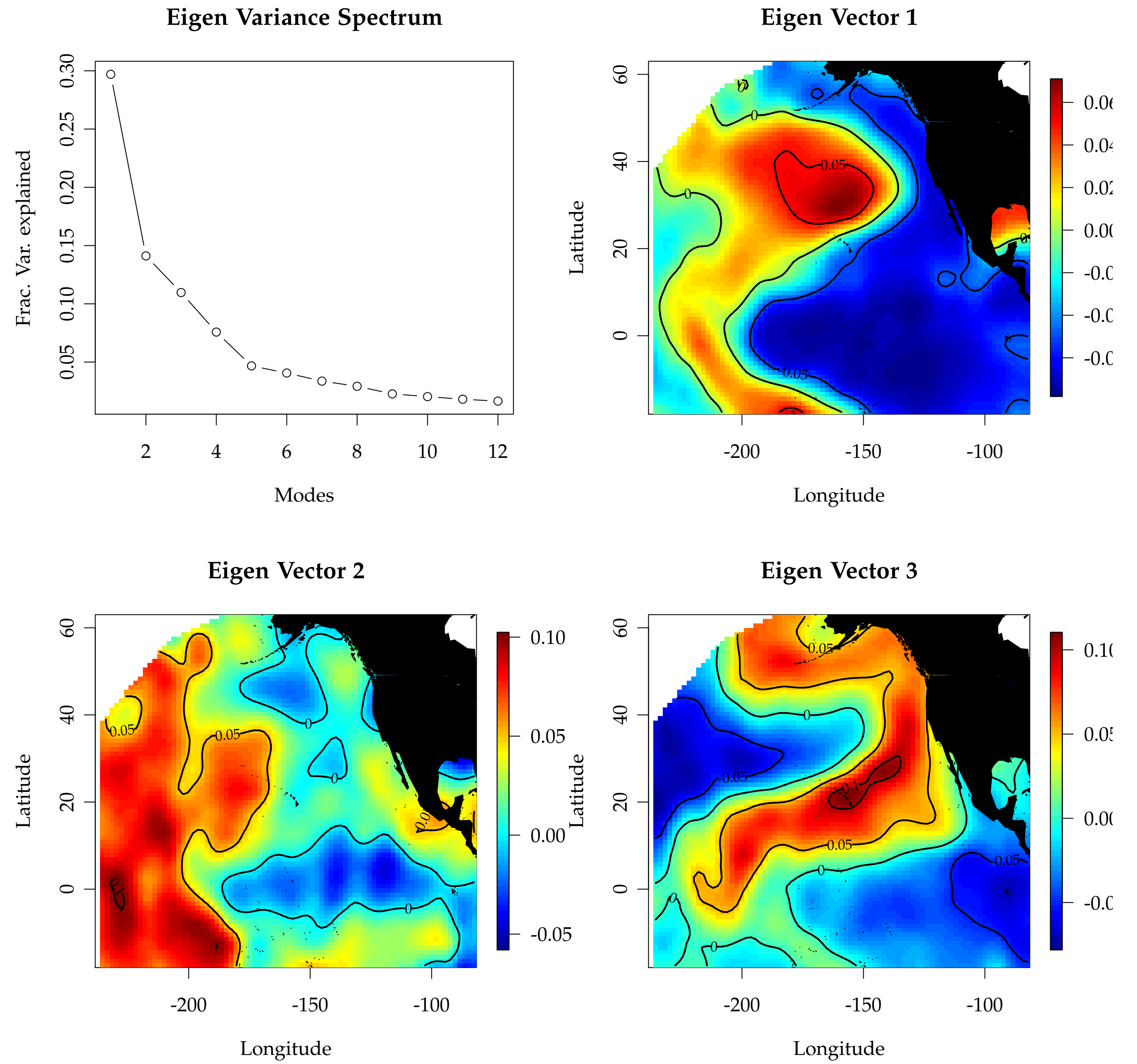

Figure 3: Pacific SST

```
> layout(cbind(c(1,2,3)))
> for(i in 1:3){
+         plot(ts(pac$pc[,i],start=1925,frequency=1),
+                 main=sprintf("PC %d",i),ylab='')
+ }
```

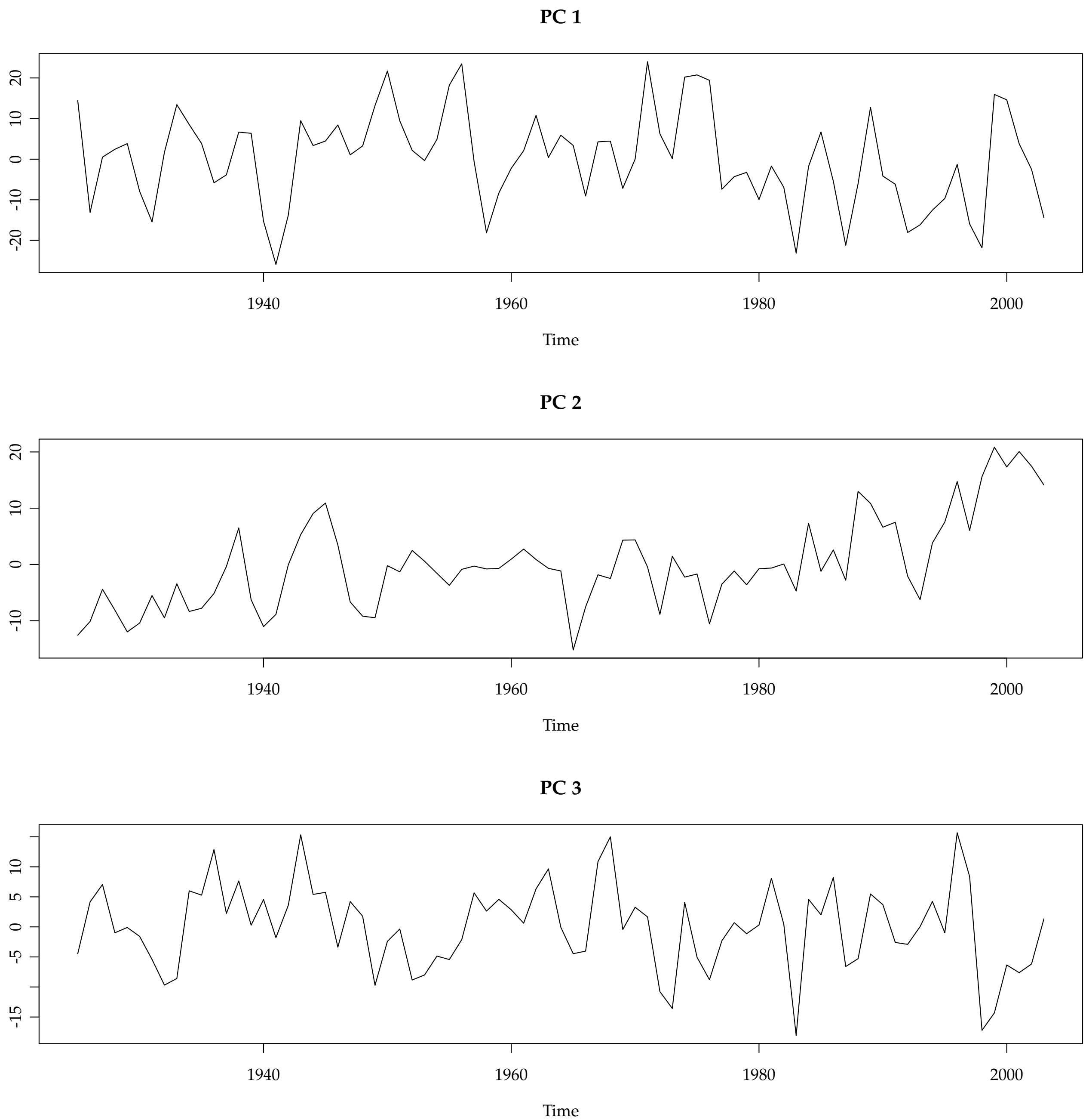

Figure 4: PC's of Pacific SST

Problem 1 (ii)

```
> layout(rbind(c(1,2),c(3,4)))
> plot(atl$eigf[1:12], type="b", xlab="Modes",
+         ylab="Frac. Var. explained",main="Eigen Variance Spectrum")
> for(i in 1:3){
+         surf <- Tps(cbind(lon.atl-360,lat.atl),atl$eigv[,i])
+         surface(surf,xlab="Longitude", ylab="Latitude",main=sprintf("Eigen Vector %d",i))
+         map('world',add=TRUE,fill=T)
+ }
```

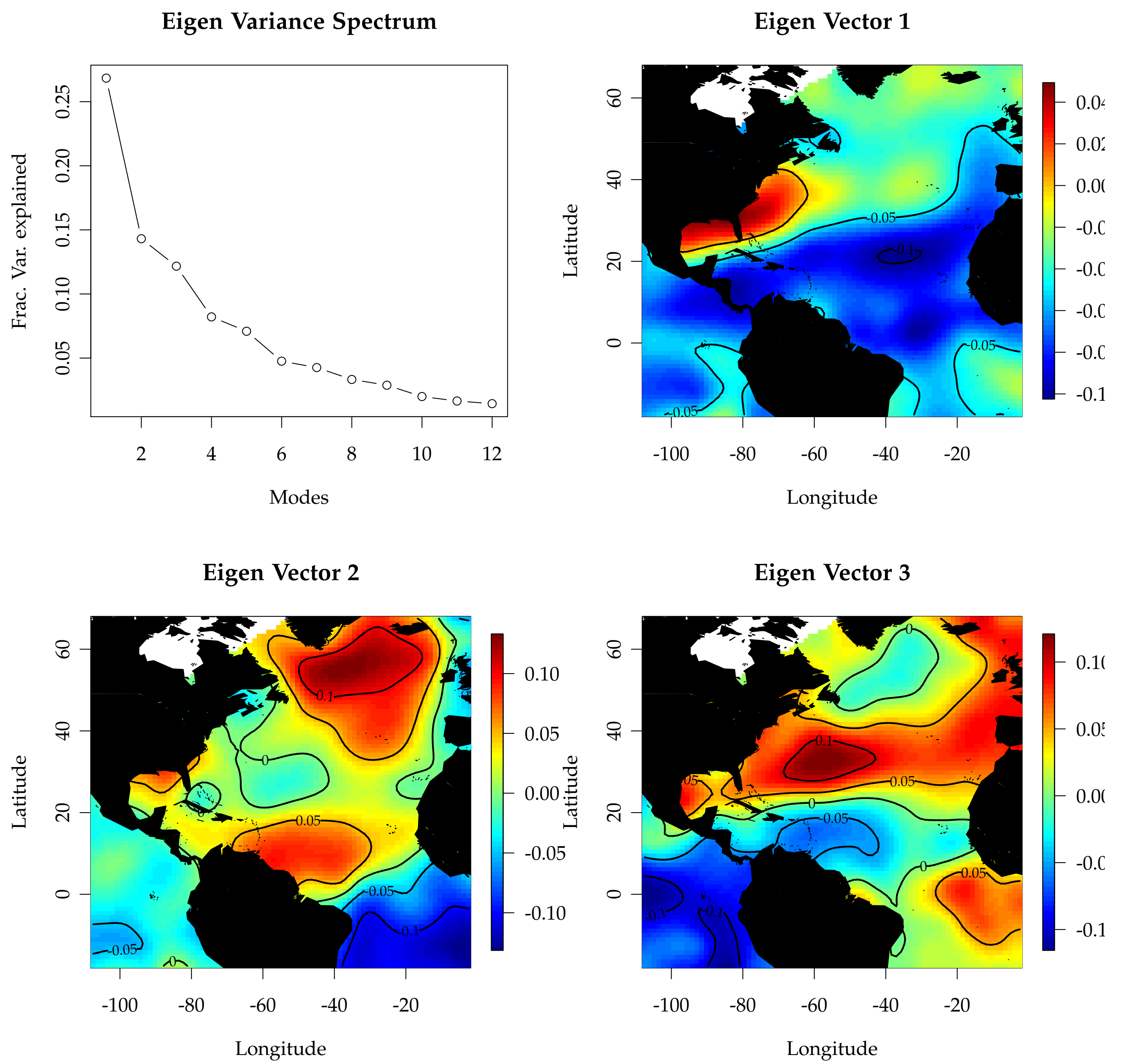

Figure 5: Atlantic SST

```
> layout(cbind(c(1,2,3)))
> for(i in 1:3){
+         plot(ts(atl$pc[,i],start=1925,frequency=1),
+                 main=sprintf("PC %d",i),ylab='')
+ }
```

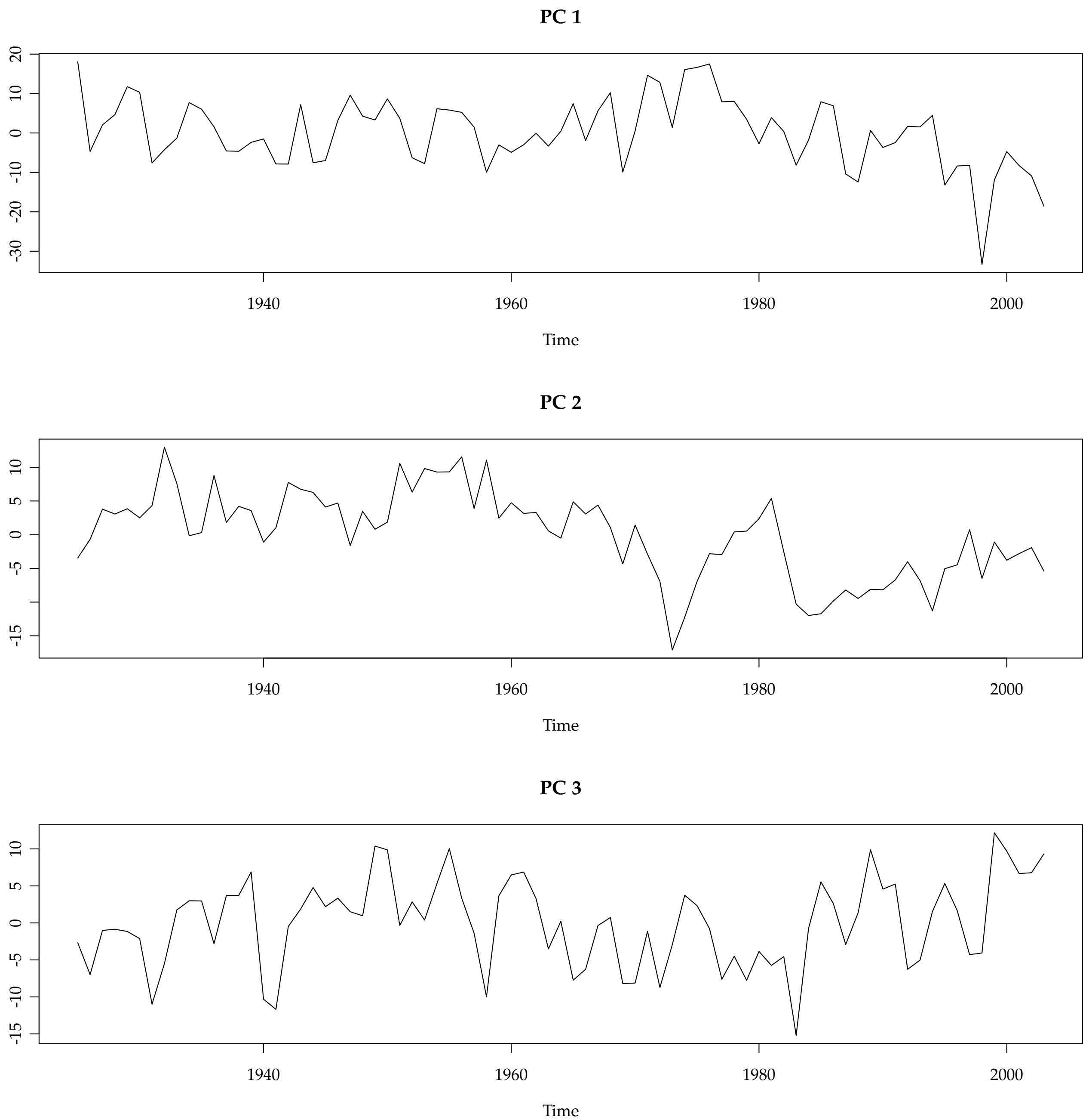

Figure 6: PC's of Atlantic SST

Problem 1 (iii)

```
> layout(rbind(c(1,2),c(3,4)))
> plot(usa$eigf[1:12], type="b", xlab="Modes",
+         ylab="Frac. Var. explained",main="Eigen Variance Spectrum")
> for(i in 1:3){
+         surf <- Tps(cbind(lon.usa-360,lat.usa),usa$eigv[,i])
+         surface(surf,xlab="Longitude", ylab="Latitude",main=sprintf("Eigen Vector %d",i))
+         map('world',add=TRUE)
+ }
```

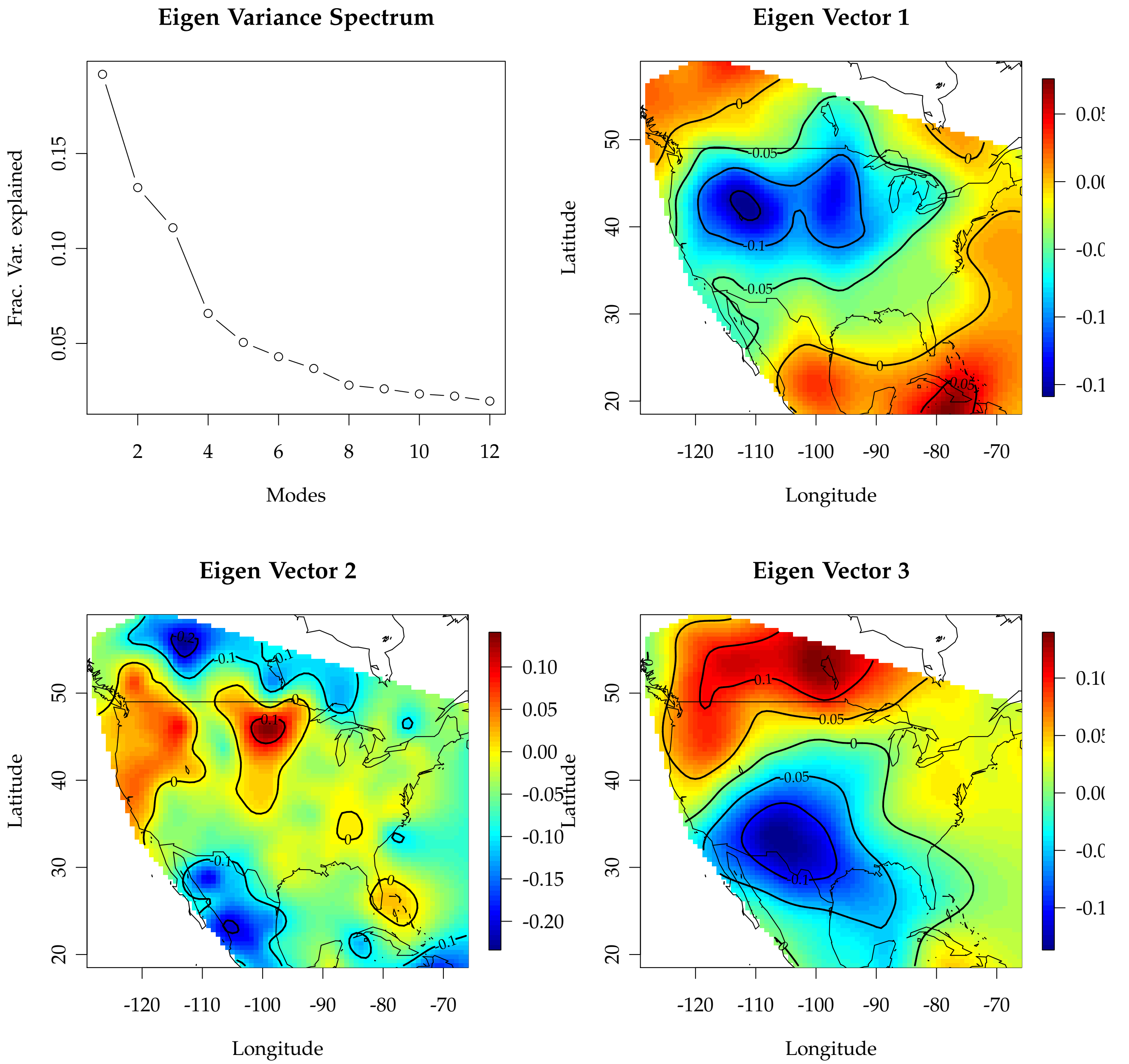

Figure 7: PDSI over the United States

```
> layout(cbind(c(1,2,3)))
> for(i in 1:3){
+         plot(ts(usa$pc[,i],start=1925,frequency=1),
+                 main=sprintf("PC%d",i),ylab='')
+ }
```

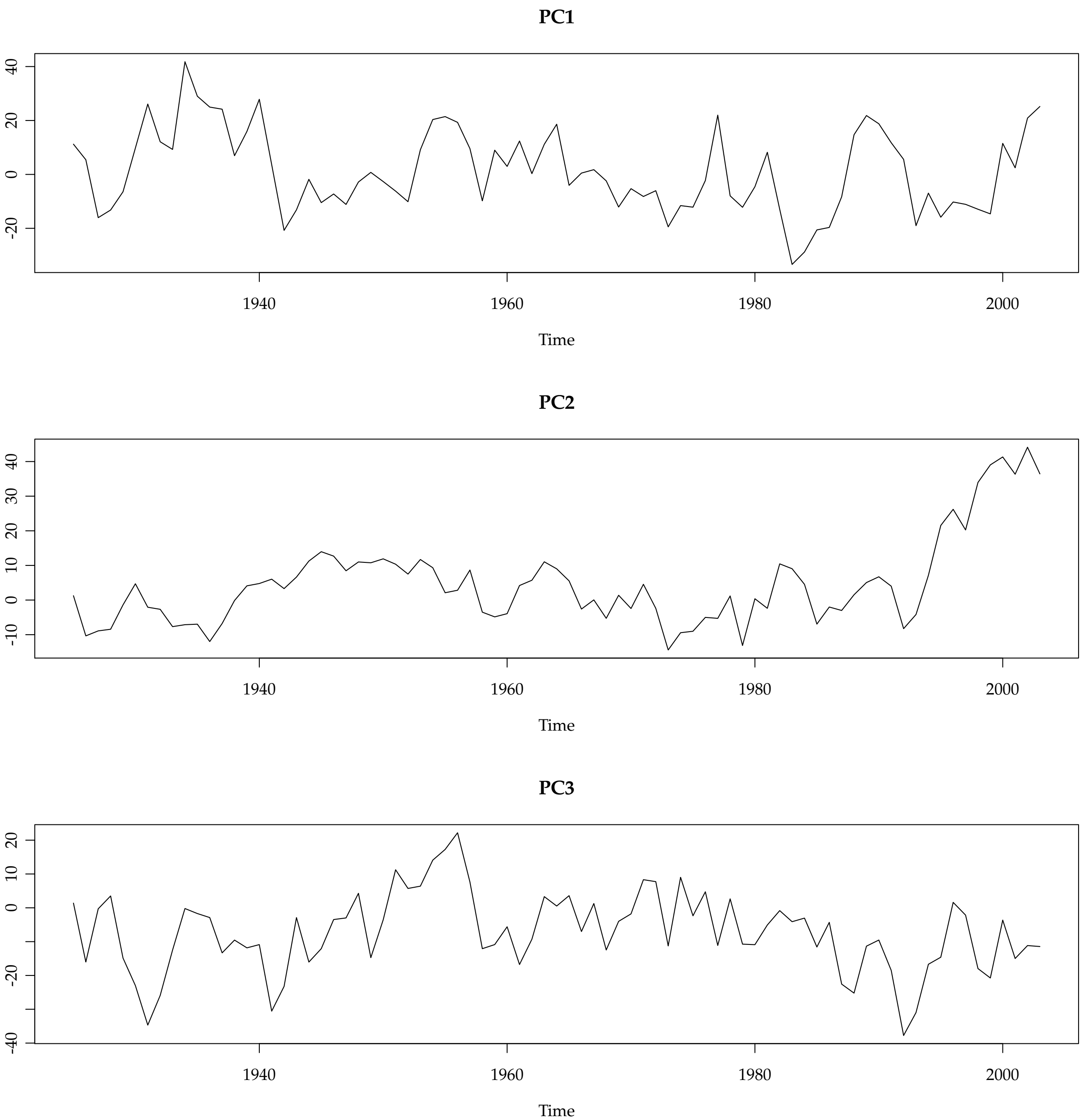

Figure 8: PC's of US PDSI

Problem 2

Below is the code to calculate the correlation maps and local poynomial fits for SST and PDSI.

```
 source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
library(locfit)

#usa - x
#pac - y
lfpc <- list()
for(i in 1:2)
        for(j in 1:2){
                b <- best.par(pac$pc[,i],usa$pc[,j])
                lfpc[[paste(i,j,sep='')]] <- locfit(usa$pc[,j]~pac$pc[,i],alpha=b$a,deg=b$p)
        }

#correlation maps
cor.pdsi.sst.pac1 <- as.vector(cor(pac$pc[,1],pdsi.usa))
cor.pdsi.sst.pac2 <- as.vector(cor(pac$pc[,2],pdsi.usa))
cor.pdsi.sst.atl1 <- as.vector(cor(atl$pc[,1],pdsi.usa))
cor.pdsi.sst.atl2 <- as.vector(cor(atl$pc[,2],pdsi.usa))

cor.map <- list()
cor.map[[1]] <- Tps(cbind(lon.usa-360,lat.usa),cor.pdsi.sst.pac1)
cor.map[[2]] <- Tps(cbind(lon.usa-360,lat.usa),cor.pdsi.sst.pac2)
cor.map[[3]] <- Tps(cbind(lon.usa-360,lat.usa),cor.pdsi.sst.atl1)
cor.map[[4]] <- Tps(cbind(lon.usa-360,lat.usa),cor.pdsi.sst.atl2)

save(lfpc,cor.map,file='output/2.Rdata')
```

Figure 9: Code for problem 2.

```
> layout(rbind(c(1,2),c(3,4)))
> for(i in 1:2)
+         for(j in 1:2){
+                 plot(pac$pc[,i],usa$pc[,j],
+                         xlab=sprintf('sst pc %d',i),
+                         ylab=sprintf('pdsi pc %d',j))
+                 lines(lfpc[[paste(i,j,sep='')]])
+
+         }
```

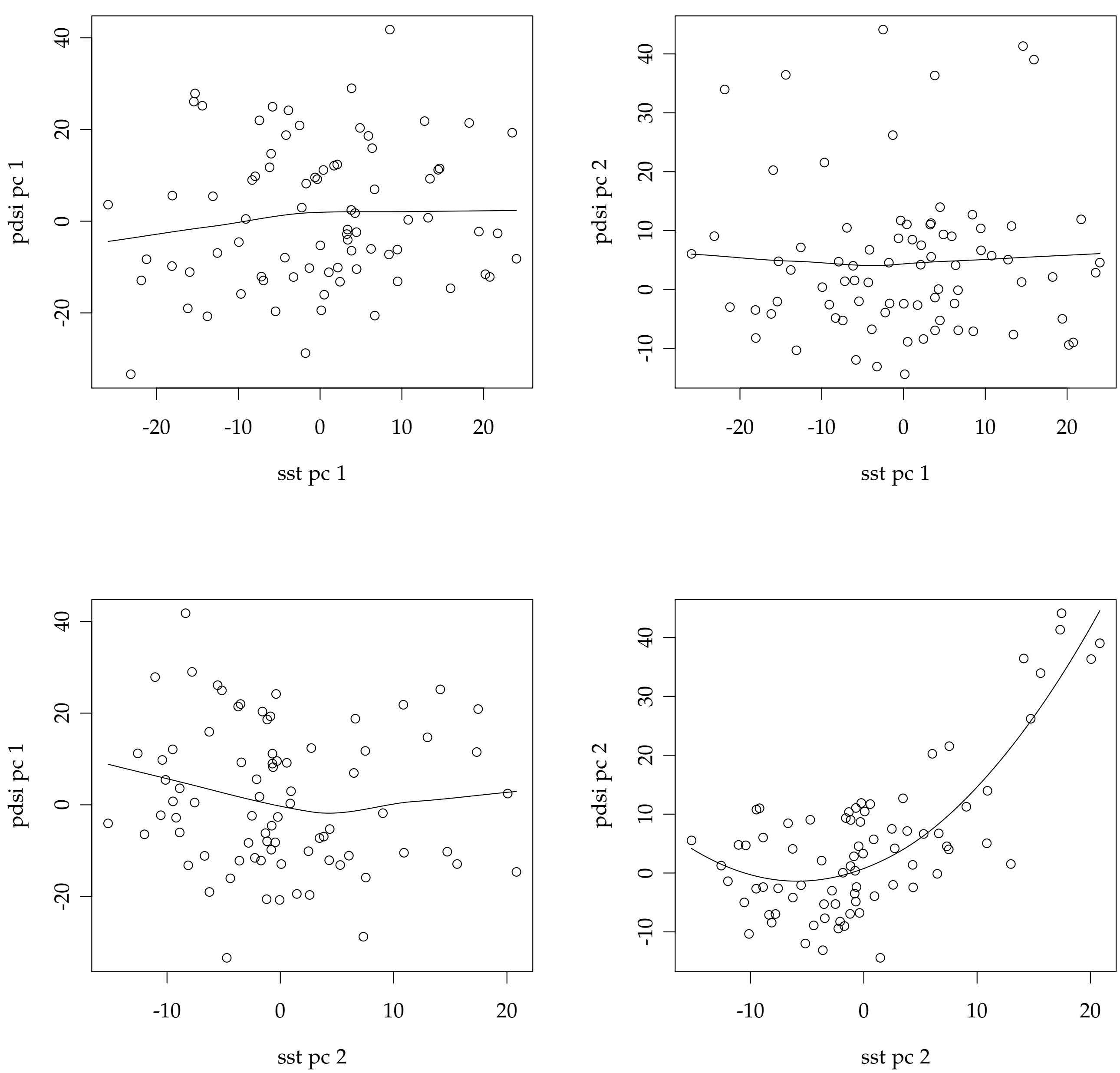

Figure 10: States PC

```
> layout(rbind(c(1,2),c(3,4)))
> for(i in 1:4){
+         surface(cor.map[[i]], xlab="Longitude", ylab="Latitude",
+                 main=paste("PDSI correlated with ",
+                         ifelse(i>2,"Atlantic","Pacific")," SST PC",
+                         ifelse(i%%2==0,2,1)))
+         map('world',add=TRUE)
+ }
```

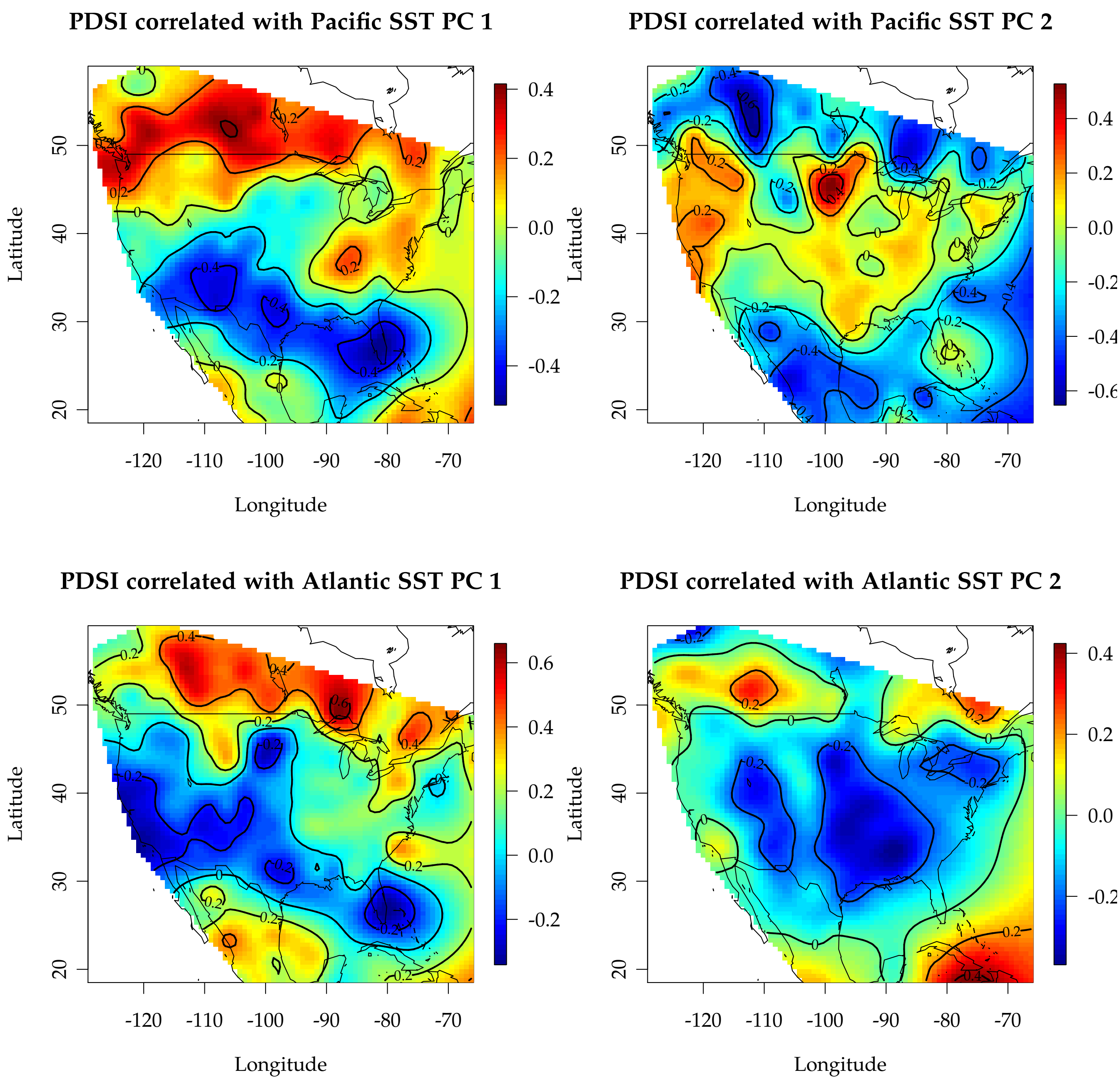

Figure 11: SST principal components correlated with PDSI

Problem 3

Below is the code which performs SVD and calcualtes the heterogeneous correlation maps.

```
 source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')

C <- var(pdsi.usa,sst)
S <- svd(C)

#Time coefficients
pdsi.tc <- pdsi.usa %*% S$u
sst.tc <- sst %*% S$v

cor.map.sst <- cor.map.pdsi <- list()
for(i in 1:3){
        cor.map.sst[[i]] <-
                Tps(cbind(sst.lon-180,sst.lat),as.vector(cor(sst.tc[,1],sst)))
        cor.map.pdsi[[i]] <-
                Tps(cbind(lon.usa-360,lat.usa),as.vector(cor(pdsi.tc[,1],pdsi.usa)))
}

save(sst.tc, pdsi.tc, cor.map.sst, cor.map.pdsi, file='output/3.Rdata')
```
Figure 12: Code for problem 3.

```
> layout(rbind(c(1,2),c(3,4)))
> plot(ts(sst.tc[,1],start=1925,frequency=1),ylab="SST TC 1")
> plot(ts(pdsi.tc[,1],start=1925,frequency=1),ylab="PDSI TC 1")
> surface(cor.map.sst[[1]], xlab="Longitude", ylab="Latitude",
+                  main='Het. Correlation map of SST (mode 1)')
> map('world',add=TRUE,fill=TRUE)
> surface(cor.map.pdsi[[1]], xlab="Longitude", ylab="Latitude",
+                  main='Het. Correlation map of PDSI (mode 1)')
> map('world',add=TRUE)
```

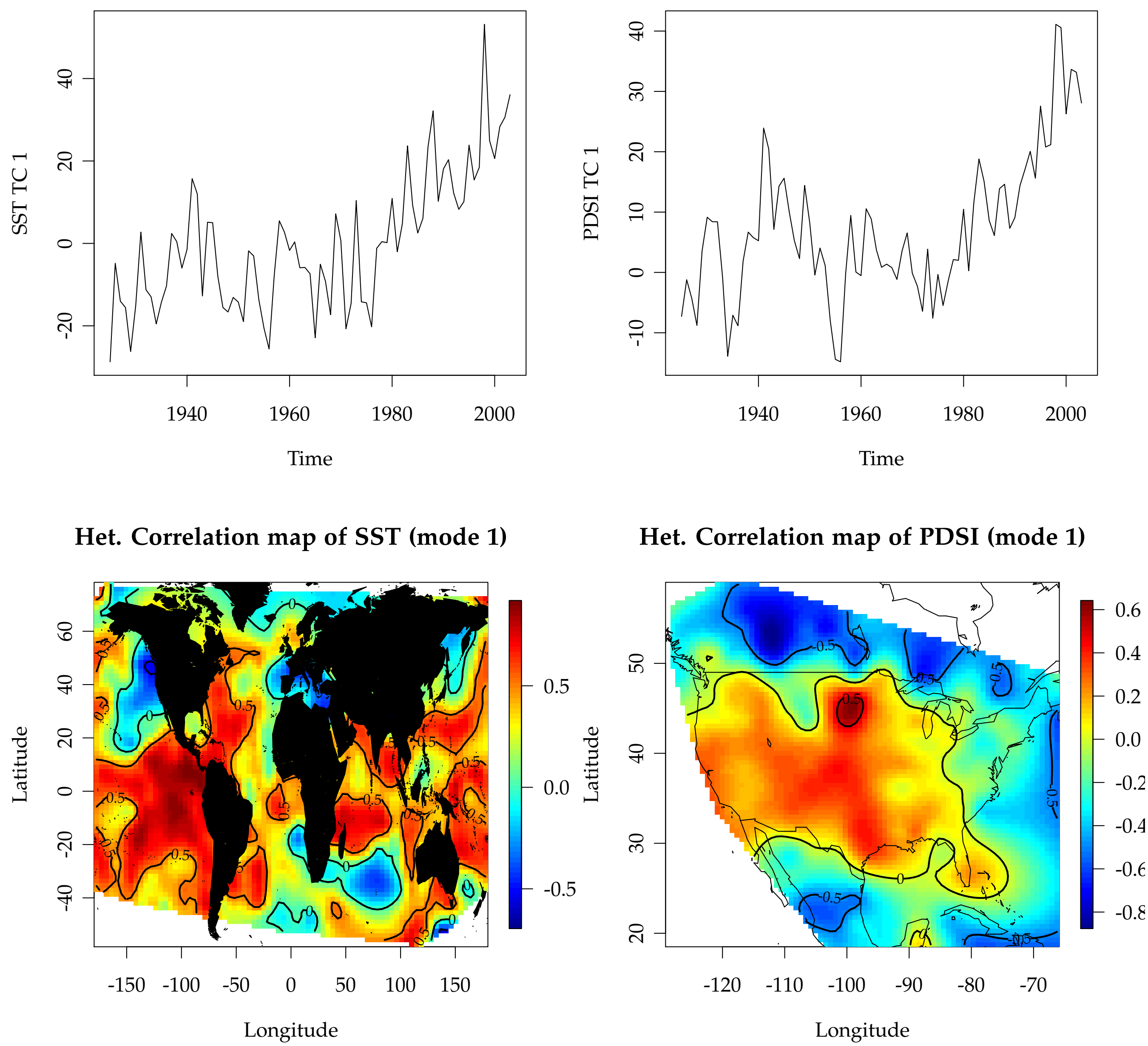

Figure 13: Heterogeneous correlation maps and time coefficients for mode 1

```
> layout(rbind(c(1,2),c(3,4)))
> plot(ts(sst.tc[,2],start=1925,frequency=1),ylab="SST TC 2")
> plot(ts(pdsi.tc[,2],start=1925,frequency=1),ylab="PDSI TC 2")
> surface(cor.map.sst[[2]], xlab="Longitude", ylab="Latitude",
+                 main='Het. Correlation map of SST (mode 2)')
> map('world',add=TRUE,fill=TRUE)
> surface(cor.map.pdsi[[2]], xlab="Longitude", ylab="Latitude",
+                 main='Het. Correlation map of PDSI (mode 2)')
> map('world',add=TRUE)
```

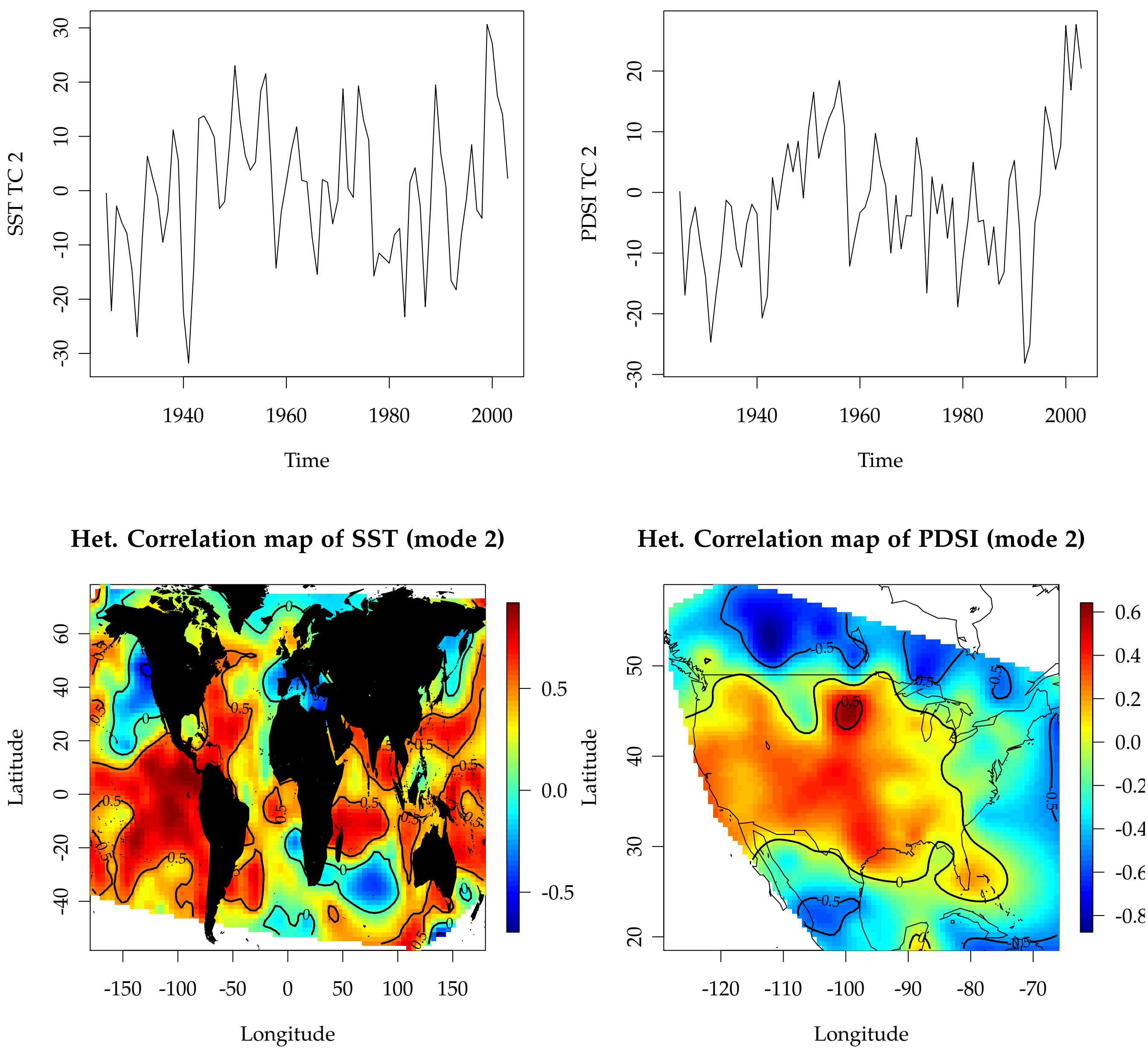

Figure 14: Heterogeneous correlation maps and time coefficients for mode 2

```
> layout(rbind(c(1,2),c(3,4)))
> plot(ts(sst.tc[,3],start=1925,frequency=1),ylab="SST TC 3")
> plot(ts(pdsi.tc[,3],start=1925,frequency=1),ylab="PDSI TC 3")
> surface(cor.map.sst[[3]], xlab="Longitude", ylab="Latitude",
+                   main='Het. Correlation map of SST (mode 3)')
> map('world',add=TRUE,fill=TRUE)
> surface(cor.map.pdsi[[3]], xlab="Longitude", ylab="Latitude",
+                   main='Het. Correlation map of PDSI (mode 3)')
> map('world',add=TRUE)
```

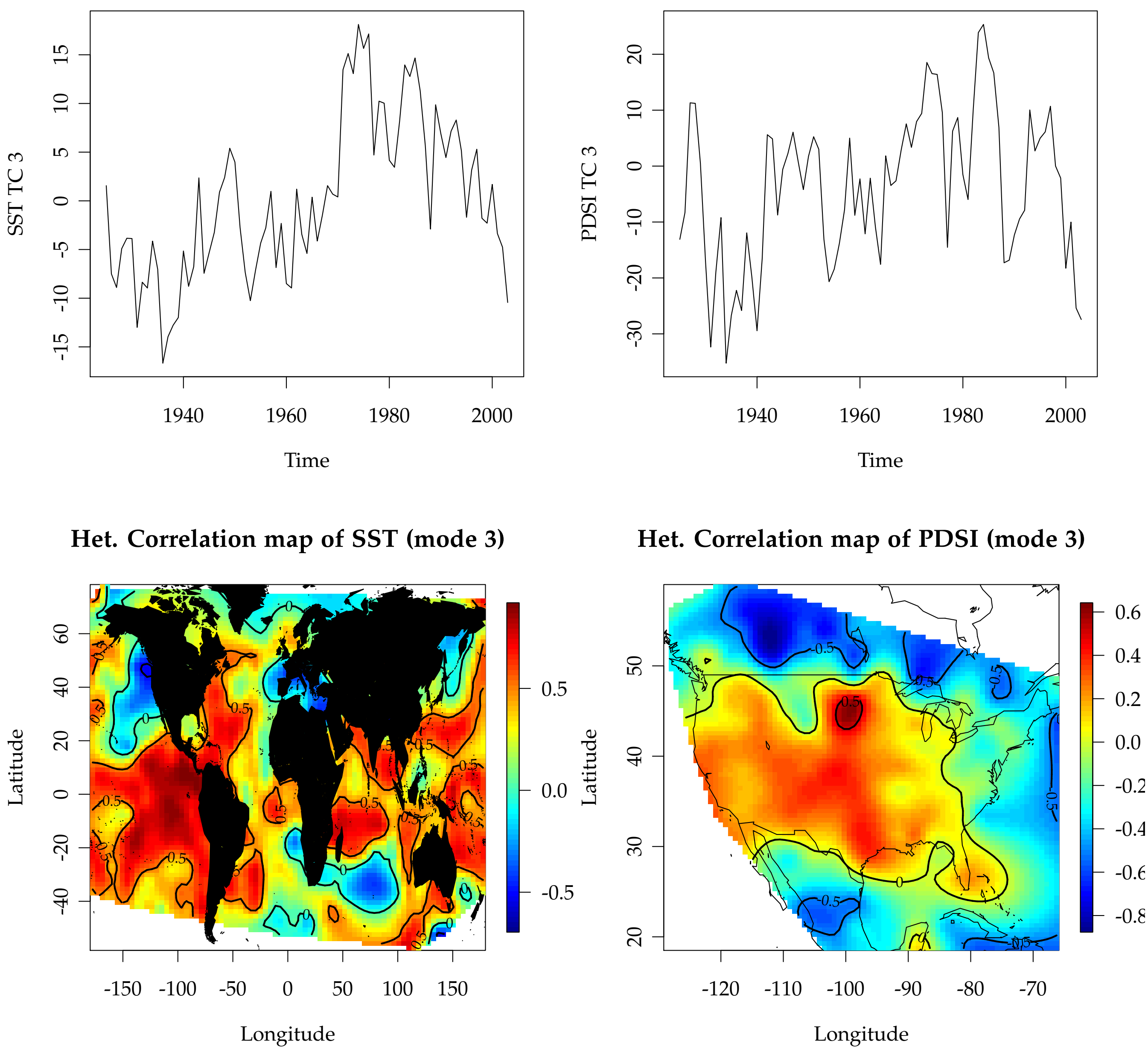

Figure 15: Heterogeneous correlation maps and time coefficients for mode 3

Problem 4

```
 source('lib.R')
flow <- as.matrix(read.table("data/AMJJFLOW-49-99.txt"))/(10^3)
swe <- as.matrix(read.table("data/March1SWE-1949-99.txt"))
nsim <- 250

#Singular value decomposition
C <- var(flow,swe)  #note that the variable with fewer columns should be first
S <- svd(C)

#Time coefficients
flow.tc <- flow %*% S$u
swe.tc <- swe %*% S$v

#heterogeneous Correlation maps
cor.map.swe <- cor.map.flow <- list()
for(i in 1:3){
        cor.map.swe[[i]] <- as.vector(cor(swe.tc[,1],swe))
        cor.map.flow[[i]] <- as.vector(cor(flow.tc[,1],flow))
}

#Local polynomial fit of TC
lftc <- list()
for(i in 1:2)
        for(j in 1:2){
                b <- best.par(swe.tc[,i],flow.tc[,j])
                lftc[[paste(i,j,sep='')]] <-
                        locfit(flow.tc[,j]~swe.tc[,i],alpha=b$a,deg=b$p)
        }

#Forecasting
ypred.svd <- fc.svd(flow,swe,nsim)

#get the rpss and median correlation stats
stats.svd <- fc.stats(flow,ypred.svd,nsim)
stats.svd$rpss <-

save(swe.tc, flow.tc, cor.map.swe, cor.map.flow, lftc, stats.svd, ypred.svd,
        flow, file='output/4.Rdata')
```

Figure 16: SVD code.

```r
fc.svd <- function(flow,swe,nsim){
        n <- nrow(flow)
        n.sites <- ncol(flow)
        nsim <- 100
        ypred <- array(NA,c(n.sites,nsim,n))
        yy <- numeric(n.sites)

        for(i in 1:n){
                #drop a point..
                swe.cv <- swe[-i,]
                flow.cv <- flow[-i,]

                #perform SVD
                C <- var(flow.cv,swe.cv)
                S <- svd(C)

                #Get the tc's
                flow.cv.tc <- flow.cv %*% S$u
                swe.cv.tc  <- swe.cv %*% S$v
                cv.model <- lsfit(swe.cv.tc[,1],flow.cv.tc[,1])
                stderr <- sum((cv.model$resid)^2) / (nrow(flow.cv) - 2)

                #prediction error..
                Sxx <- sum((swe.cv.tc[,1] - mean(swe.cv.tc[,1]))^2)

                #the TCs of the dropped point..
                xp <- swe[i,] %*% S$v

                #generate an ensemble..
                for(isim in 1:nsim){
                        this.stderr <- sqrt(stderr *
                                (1 + (1/(n-1)) +
                                        ((xp[1] - mean(swe.cv.tc[,1]))^2)/Sxx))

                        yy[1] <- cv.model$coef[1] +
                                cv.model$coef[2]*xp[1] + rnorm(1,0,this.stderr)

                        #grab random TCs for the remaining TCs
                        for(site in 2:n.sites){

                                r.tc <- round(runif(1,1,(n-1)))
                                yy[site] <- flow.cv.tc[r.tc,site]
                        }
                        ypred[1:n.sites,isim,i]=yy %*% t(S$u)
                }
        }
        ypred[ypred < 0] <- 0
        return(ypred)
}
```

Figure 17: SVD forecast code.

```
fc.stats <- function(flow,ypred,nsim){

       n.sites <- ncol(flow)
       site.rpss <- site.mc <- numeric(n.sites)
       n <- nrow(flow)

       ##Calculate rpss
       for(i in 1:n.sites){

               thresh <- quantile(flow[,i], c(0.33, 0.66))

               # climatological forecast..
               climo=cumsum(c(1/3, 1/3, 1/3))

               rpss = numeric(n)
               for(jj in 1:n){

                       # forecast categorical probabilities
                       fcastprob=1:3
                       yypred=ypred[i,,jj]
                       fcastprob[1] <- length(yypred[yypred <= thresh[1]]) / nsim
                       fcastprob[2] <-
                               length(yypred[yypred > thresh[1] &
                                       yypred < thresh[2]]) / nsim
                       fcastprob[3] <- length(yypred[yypred >= thresh[2]]) / nsim

                       fcastprob=cumsum(fcastprob)

                       # actual..
                       actual=rep(0,3)
                       if(flow[jj,i] <= thresh[1])actual[1]=1
                       if(flow[jj,i] > thresh[1] & flow[jj,i] < thresh[2])actual[2]=1
                       if(flow[jj,i] >= thresh[2])actual[3]=1


                       rpsclimo = sum((climo-actual)^2)
                       rpsfcast = sum((fcastprob - actual)^2)
                       rpss[jj] = 1 - (rpsfcast/rpsclimo)

               }
               #print out the median RPSS
               site.rpss[i] <- median(rpss)
               site.mc[i] <- cor(flow[,i],apply(ypred[i,,],2,median))
       }
       return(list(rpss=site.rpss,mc=site.mc))
}
```

Figure 18: SVD forecast stats code.

```
> layout(rbind(c(1,2),c(3,4)))
> plot(ts(swe.tc[,1],start=1925,frequency=1),ylab="SWE TC 1")
> plot(ts(flow.tc[,1],start=1925,frequency=1),ylab="Flow TC 1")
> plot(cor.map.swe[[1]], xlab="Site", ylab="Correlation",
+                main='Het. Correlation map of SWE (mode 1)',type='l')
> plot(cor.map.flow[[1]], xlab="Site", ylab="Correlation",
+                main='Het. Correlation map of Flow (mode 1)',type='l')
```

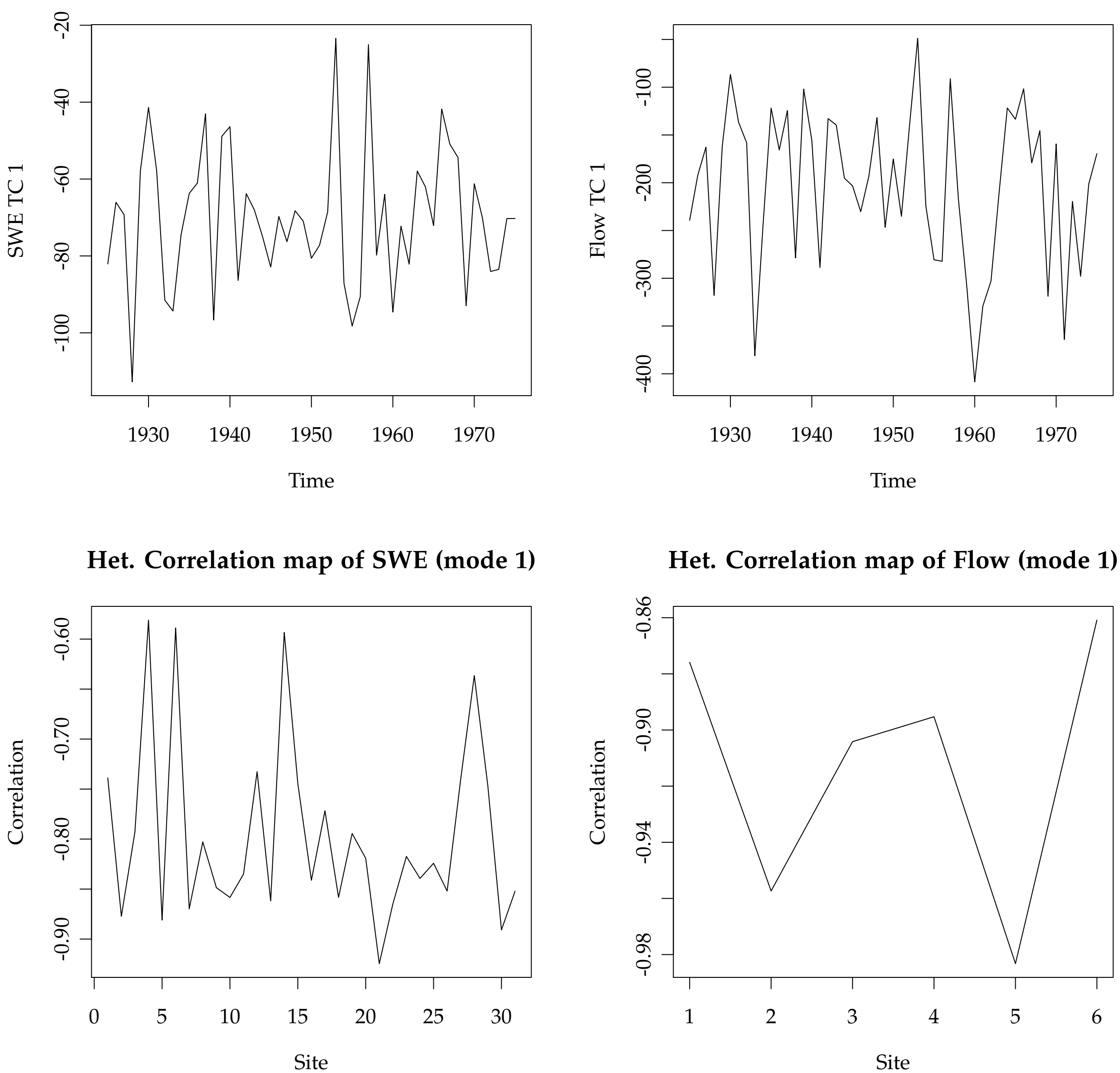

Figure 19: Heterogeneous correlation maps and time coefficients for mode 1

```
> layout(rbind(c(1,2),c(3,4)))
> plot(ts(swe.tc[,2],start=1925,frequency=1),ylab="SWE TC 2")
> plot(ts(flow.tc[,2],start=1925,frequency=1),ylab="Flow TC 2")
> plot(cor.map.swe[[2]], xlab="Site", ylab="Correlation",
+                  main='Het. Correlation map of SWE (mode 2)',type='l')
> plot(cor.map.flow[[2]], xlab="Site", ylab="Correlation",
+                  main='Het. Correlation map of Flow (mode 2)',type='l')
```

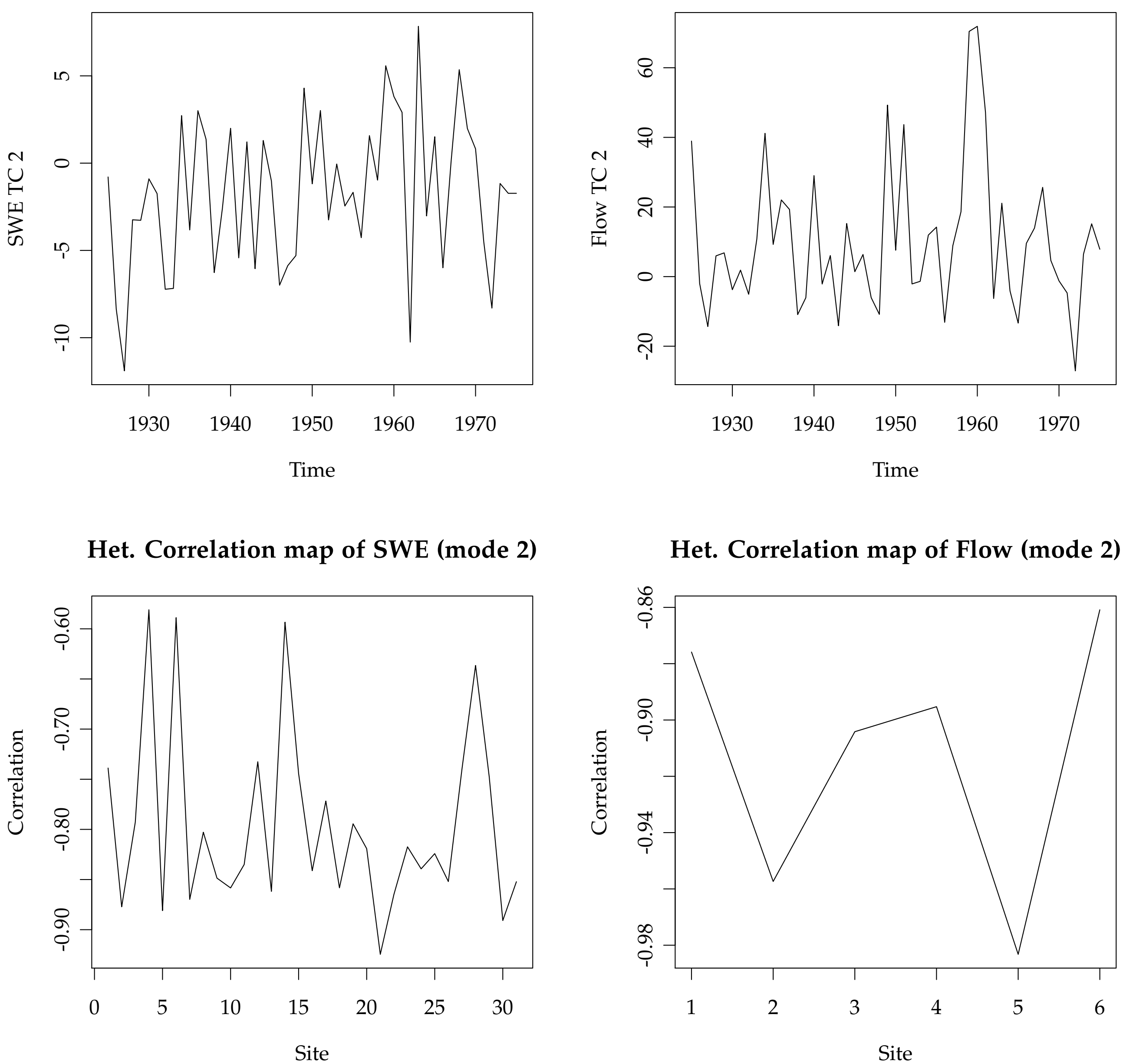

Figure 20: Heterogeneous correlation maps and time coefficients for mode 2

```
> layout(rbind(c(1,2),c(3,4)))
> for(i in 1:2)
+         for(j in 1:2){
+                 plot(swe.tc[,i],flow.tc[,j],
+                         xlab=sprintf('SWE TC %d',i),
+                         ylab=sprintf('Flow TC %d',j))
+                 lines(lftc[[paste(i,j,sep='')]])
+
+         }
```

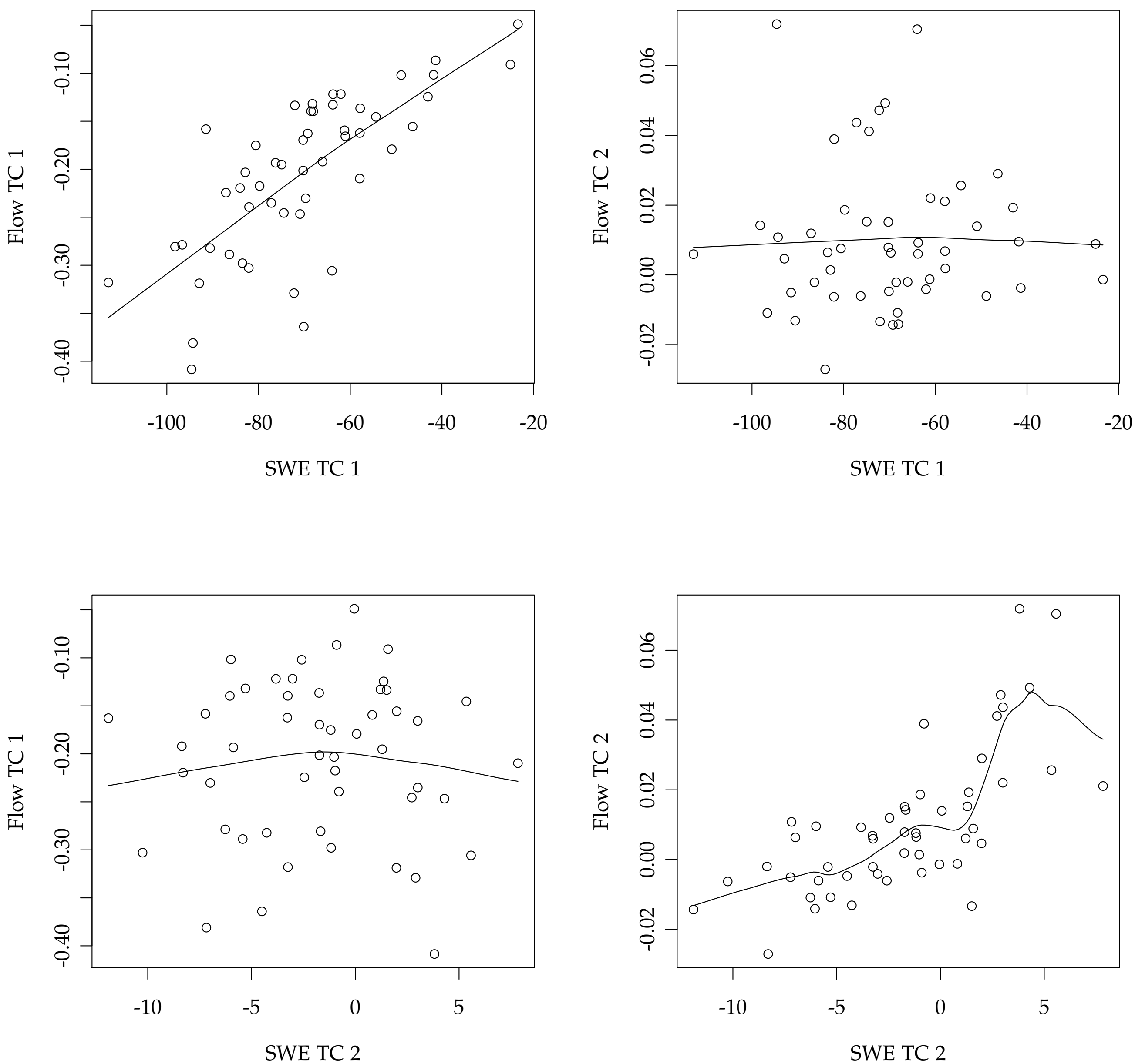

Figure 21: States PC

```
> layout(cbind(c(1:3)))
> for(i in 1:3){
+         this.site <- ypred.svd[i,,]
+         this.site <- as.data.frame(this.site)
+         names(this.site) <- 1949:1999
+         boxplot(this.site,xlab=paste('Site',i))
+         lines(flow[,i],col='blue')
+         mtext(paste('RPSS = ',round(stats.svd$rpss[i],3),
+         'MC = ',round(stats.svd$rpss[i],3)),line=1)
+ }
```

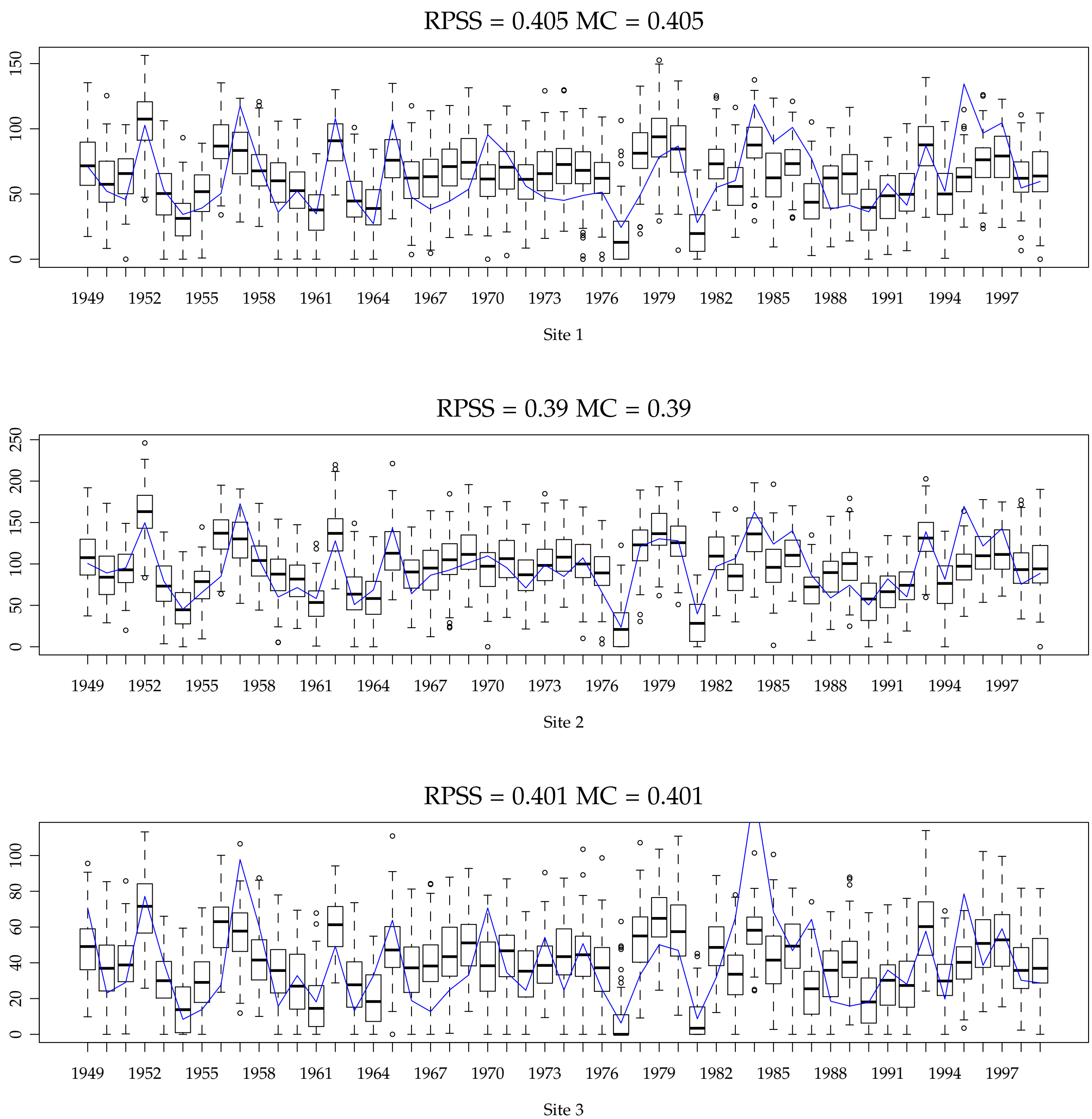

Figure 22: Svd predictions for first three sites

```
> layout(cbind(c(1:3)))
> for(i in 4:6){
+         this.site <- ypred.svd[i,,]
+         this.site <- as.data.frame(this.site)
+         names(this.site) <- 1949:1999
+         boxplot(this.site,xlab=paste('Site',i))
+         lines(flow[,i],col='blue')
+         mtext(paste('RPSS = ',round(stats.svd$rpss[i],3),
+                 'MC = ',round(stats.svd$rpss[i],3)),line=1)
+ }
```

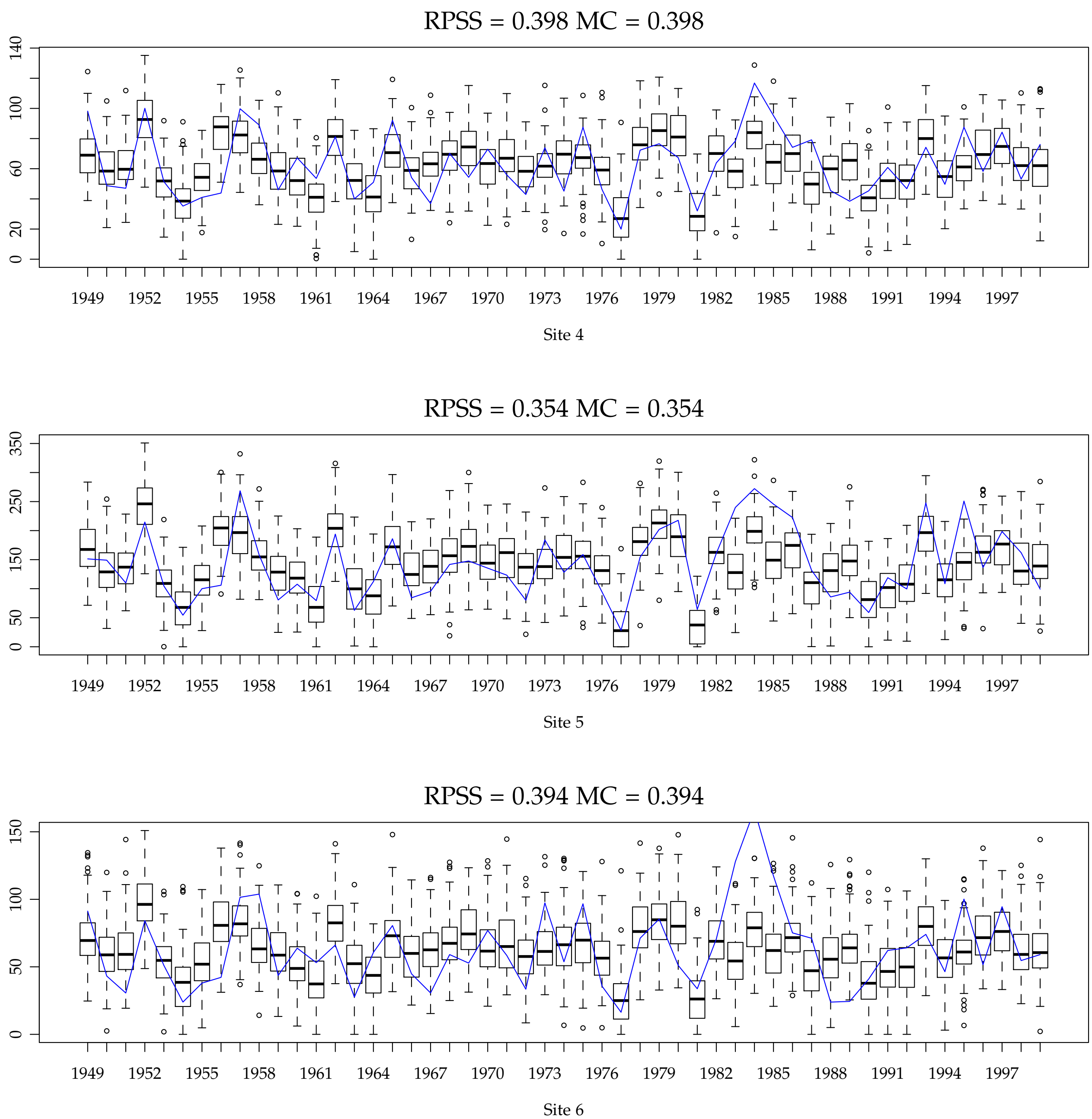

Figure 23: Svd predictions for second three sites

Problem 5

I did not significantly modeify the CCA code from the class page so it is not shown here, but it was used to generate forecasts.

```
> layout(cbind(c(1:6)))
> for(i in 1:6){
+         par(mar=c(.9,2,1.5,1))
+         plot(1949:1999, ypred.cca[,i], xlab=paste('Site',i),
+                 ylim=range(c(ypred.cca[,i],flow[,i])),main=paste('Site',i),
+                 type='l')
+         lines(1949:1999,flow[,i],col='blue')
+ }
```

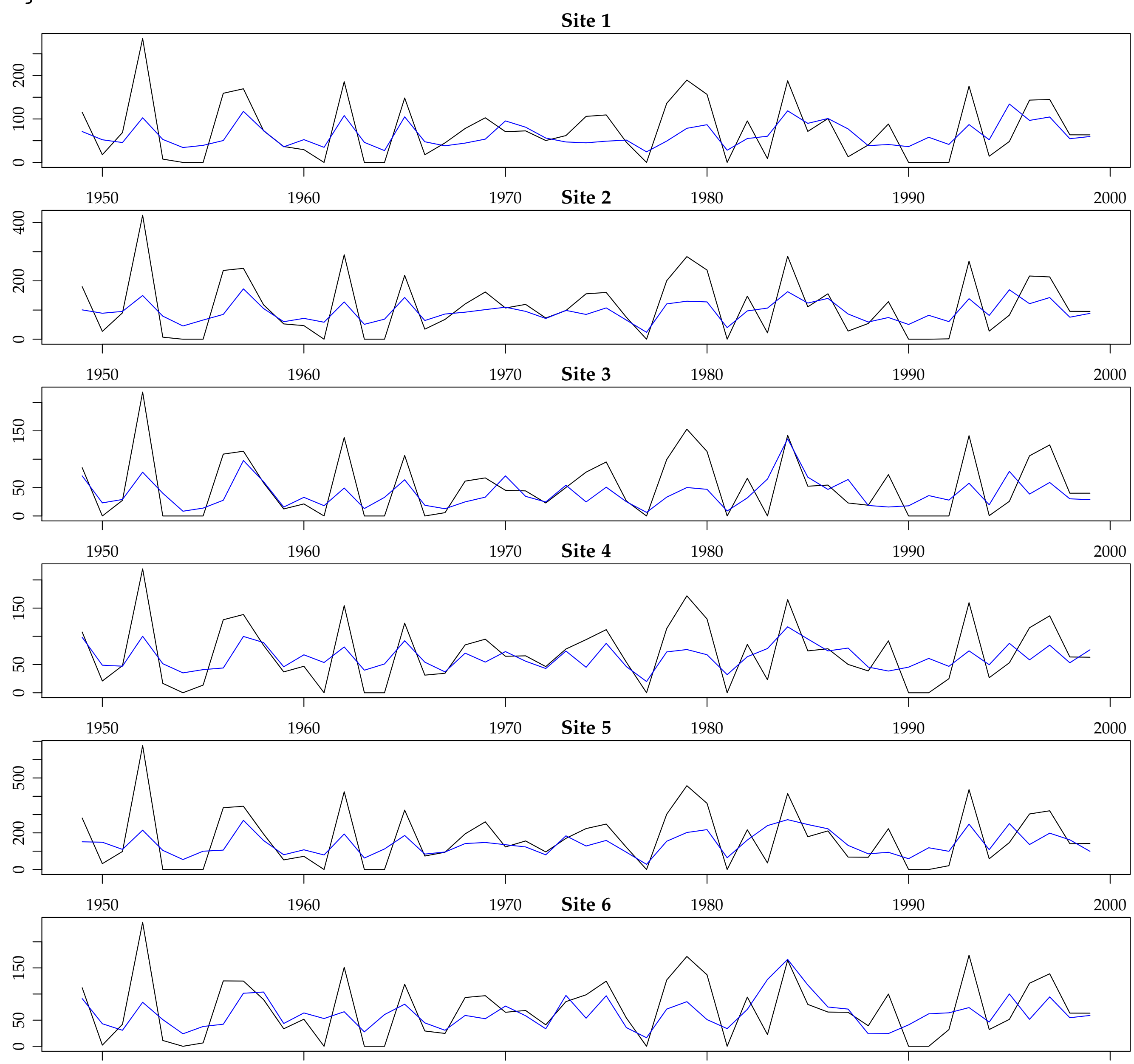

Figure 24: CCA predictions for all sites, historical flow is in blue. Units are thousand cubic meters

Problem 6

Below is the code that does the Markov KNN simulation.

```
 source('lib.R')
nyears <- 95
nsim <- nyears * 100
        #read data (ac-ft) and convert to cms
x <- as.matrix(read.table('data/Leesferry-mon-data.txt')[,-1])*0.000469050
x.raw <- array(t(x))
x.ann <- x.bin <- ts.annual.mean(ts(x.raw,start=c(1906,1),frequency=12))
n <- length(x.ann)
        #historical stats
x.mean <- mean(x.ann); x.sd <- sd(x.ann)
x.skew <- skew(x.ann); x.lag1 <- mylag(x.ann,1,docor=T)

        #binary series
x.bin[(x.ann > median(x.ann))] <- 1
x.bin[!(x.ann > median(x.ann))] <- 0

x.wd <- x.ww <- x.dw <- x.dd <- numeric()
p.wd <- p.ww <- p.dw <- p.dd <- n.wd <- n.ww <- n.dw <- n.dd <- 0
for(i in 2:length(x.bin)){
        if(x.bin[i-1] == 0 && x.bin[i] == 0){n.dd <- n.dd + 1; x.dd[n.dd] <- x.ann[i]}
        if(x.bin[i-1] == 0 && x.bin[i] == 1){n.dw <- n.dw + 1; x.dw[n.dw] <- x.ann[i]}
        if(x.bin[i-1] == 1 && x.bin[i] == 0){n.wd <- n.wd + 1; x.wd[n.wd] <- x.ann[i]}
        if(x.bin[i-1] == 1 && x.bin[i] == 1){n.ww <- n.ww + 1; x.ww[n.ww] <- x.ann[i]}
}
p.wd <- n.wd / (n.wd+n.ww); p.ww <- n.ww / (n.wd+n.ww)
p.dw <- n.dw / (n.dw+n.dd); p.dd <- n.dd / (n.dw+n.dd)

#Simulate
x.sim.state <- x.sim <- numeric(nsim)
x.sim.state[1] <- 1
x.sim[1] <- quantile(x.ann,.75)
for(i in 2:nsim){
        r <- runif(1)
        if(x.sim.state[i-1] == 1){
                x.sim.state[i] <- if(r < p.ww) 1 else 0
                pool <- if(r < p.ww) x.ww else x.wd
        }else{
                 x.sim.state[i] <- if(r < p.dd) 0 else 1
                pool <- if(r < p.dd) x.dd else x.dw
        }
        x.sim[i] <- sample(pool,1)
}
#calculate stats
x.sim <- matrix(x.sim,ncol=nyears)
x.sim.stats <- annual.stats(x.sim)

save(x.sim.stats, x.sim, x.ann, x.mean, x.sd, x.skew, x.lag1, nsim, nyears,
        file='output/6.Rdata')
```

Figure 25: Markov KNN resampling code.

```
> layout(rbind(1:4))
> boxplot(x.sim.stats[,1],ylim=range(c(x.sim.stats[,1],x.mean)),xlab='Mean')
> points(x.mean,col='red')
> boxplot(x.sim.stats[,2],ylim=range(c(x.sim.stats[,2],x.sd)),xlab='SD')
> points(x.sd,col='red')
> boxplot(x.sim.stats[,3],ylim=range(c(x.sim.stats[,3],x.skew)),xlab='Skew')
> points(x.skew,col='red')
> boxplot(x.sim.stats[,4],ylim=range(c(x.sim.stats[,4],x.lag1)),xlab='Lag1 Cor')
> points(x.lag1,col='red')
```

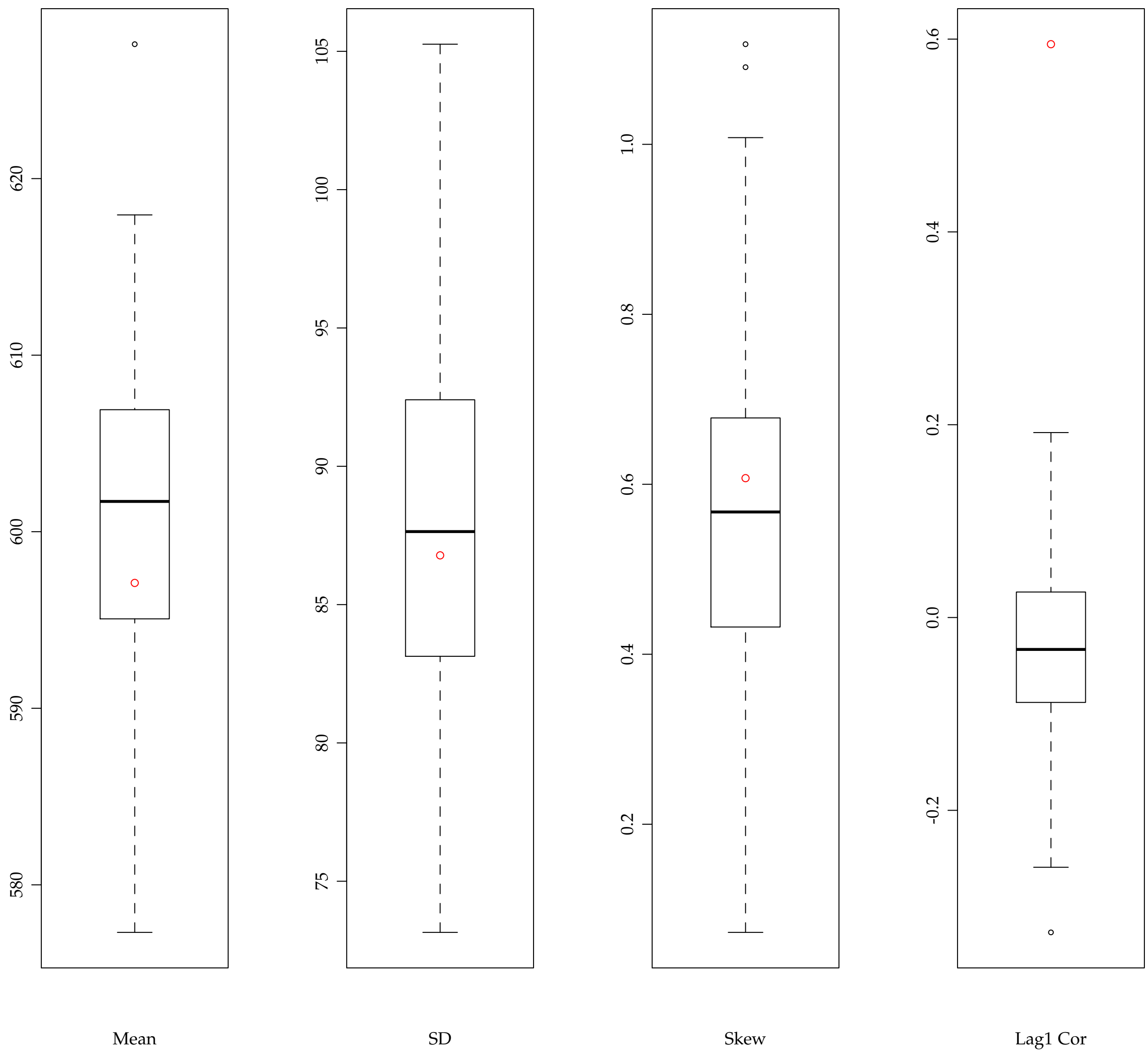

Figure 26: Annual simulates statistics. All statistics are well captured except for Lag 1 autocorrelation. I believe this is to to the way values re sampled from each 'bin.' The only way yo get enough sampling variation is to select from all of the vaues in a bin.

One interesting result I noticed was that the historical pdf was only reroduced when I used a uniform weight function and selected from all the neighbors in a bin. If I used a $1/k$ weight function and set k to $\sqrt{n}$, the simulated values did not have very much variability at all.

```
> n.pt <- 100
> out <- sm.density(x.ann)
> pt <- seq(min(out$eval.points),max(out$eval.points),length.out=n.pt)
> boxpt <- matrix(NA,nyears,n.pt)
> for(i in 1:nyears){
+         boxpt[i,] <- sm.density(x.sim[i,],
+                 eval.points=pt,display='none')$estimate
+         lines(pt,boxpt[i,],col='gray',lwd=.5)
+ }
> sm.density(x.ann,add=T,col='red',lwd=2)
```

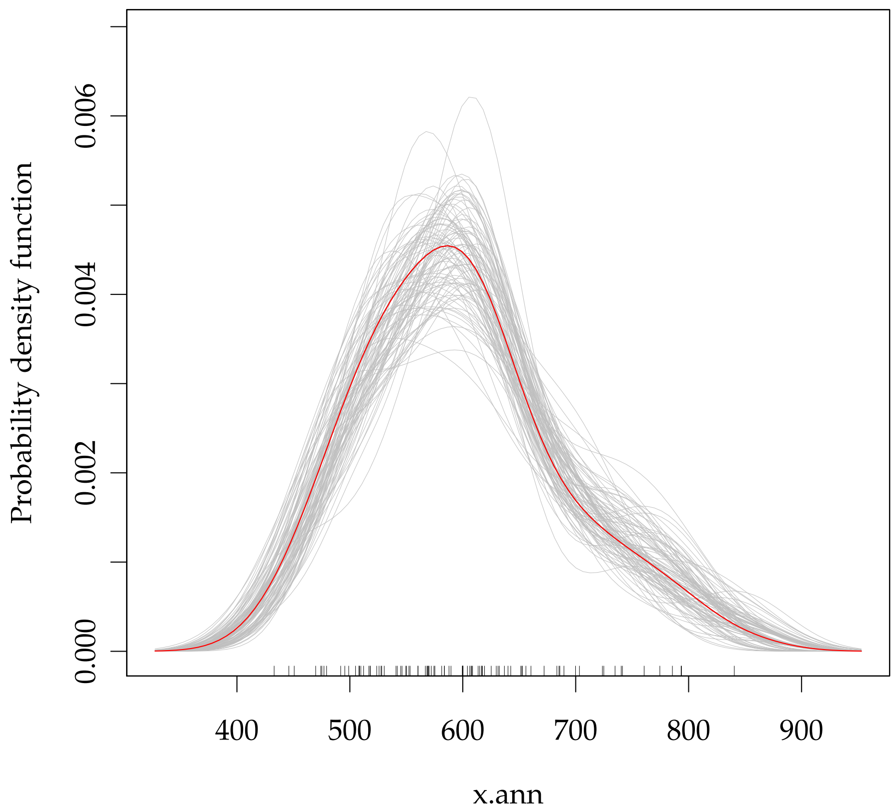

Figure 27: Annual Historical pdf and simulated pdf