Homework #2: Time Series Analysis

Cameron Bracken

CVEN6833 Fall, 2009

Problem 1

Problem 1 (i) and (ii)

The code to read in the Lees Ferry data is shown in Figure 1. The function to calulate the *ACF* and *PACF* is shown in Figure 2 and Figure 2 respectively. Figure 4 and Figure 5 show the *ACF* and *PACF* calulated from first principals.

```
        #read support functions
source('lib.R')
require(pear)

nsims <- 100
nyears <- 95

        #read data (ac-ft)
x <- as.matrix(read.table('data/Leesferry-mon-data.txt')[,-1])
        # creat timeseries and convert to cms
x <- x*0.000469050
x.mean <- apply(x,2,mean)
x.sd <- apply(x,2,sd)
x.skew <- apply(x,2,skew)

        #remove mean
x.raw <- array(t(x))
x.scale <- t((t(x) - x.mean) / x.sd)
x.may.scale <- x.scale[,5]
x.may <- x[,5]
x.ts <- ts(array(t(x.scale)),start=c(1906,1),frequency=12)
x.ts.raw <- ts(x.raw,start=c(1906,1),frequency=12)


x.lag1 <- peacf(x.ts.raw,plot=FALSE,lag.max=1)$acf

save(x,x.raw,x.ts,x.ts.raw,x.may,x.may.scale,
        x.mean,x.sd,x.skew,x.lag1,nyears,nsims,file='output/1.Rdata')
```

Figure 1: Reading the data and calculating the statistics.

```
my.acf <- function(x, lag.max=2*frequency(x), plot=TRUE){

        acf <- numeric(lag.max)
        for(i in 1:lag.max)
                acf[i] <- mylag(x,i,docor=T)
        ci <- 2/sqrt(length(x))
        if(plot){
                plot(acf,type='h',xlab='Lag',ylab='ACF')
                abline(h=c(ci,-ci),col='blue',lty=2)
                abline(h=0)
        }
        invisible(acf)
}
```

Figure 2: Function to calculate the *ACF*

```
my.pacf <- function(x, lag.max=2*frequency(x), plot=TRUE){
        pacf <- numeric(lag.max)
        acf <- my.acf(x, lag.max = lag.max, plot=FALSE)
        if(lag.max==1) return(acf)
        pacf[1] <- acf[1]
                # pacf for each lag
        for(q in 2:lag.max){
                r <- acf[1:q]
                        #get the top and bottom matricies for pacf
                top <- matrix(NA,q,q)
                top[1:(q-1),1:(q-1)] <- ac.mat(x,q-1)
                top[,q] <- r
                top[q,1:(q-1)] <- rev(r[1:(q-1)])
                bot <- ac.mat(x,q)
                pacf[q] <- det(top)/det(bot)
        }
        ci <- 2/sqrt(length(x))
        if(plot){
                plot(pacf,type='h',xlab='Lag',ylab='PACF')
                abline(h=c(ci,-ci),col='blue',lty=2)
                abline(h=0)
        }
        invisible(pacf)
}
```

Figure 3: Function to calculate the *PACF*

```
> my.acf(x.ts)
```

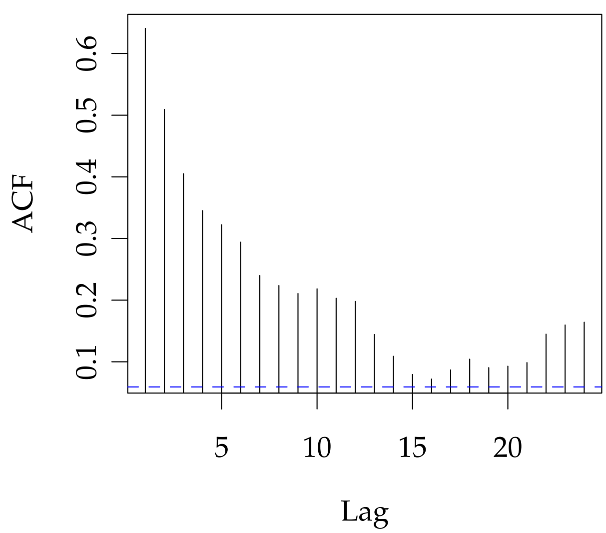

Figure 4: acf

```
> my.pacf(x.ts)
```

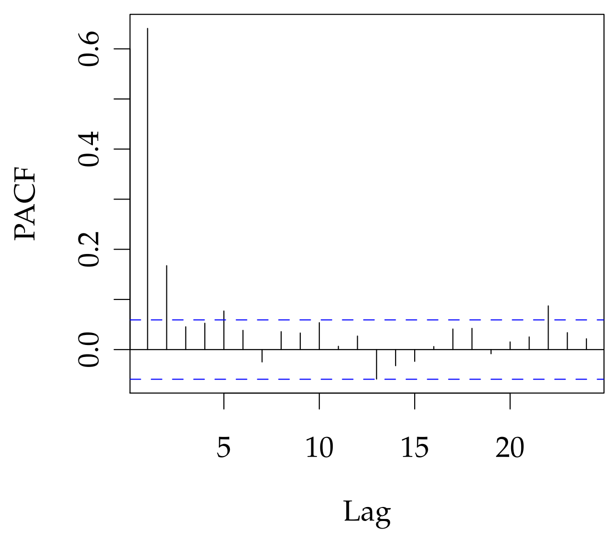

Figure 5: pacf

Problem 1 (iii), (vi) and (v)

The function `solve.yw()` was created to solve for the AR parameters of any order. The results for the AR fitting are shown in Table 1. Of the three models, the AIC value for the order 3 model is the lowest so that model is the best.

```
solve.yw <- function(x,p){
                #autocorrelation
        r <- my.acf(x, lag.max = p, plot=FALSE)
        M <- ac.mat(x,p)
                #get the AR parameters
        phi <- solve(M) %*% cbind(r)
        colnames(phi) <- ''
        sigsq <- var(x)*(1-sum(r * phi))
        n <- length(x)
                # Goodness of fit parameters
        aic <- n*log(sigsq)+2*p
        my.gcv <- n*sigsq/(n-(p+1))^2

        return(list(phi=phi,sigsq=sigsq,aic=aic,gcv=my.gcv))
}
ac.mat <- function(x,p){
                # Calculate the coefficient matrix for AR model
        rho <- c(1,my.acf(x, lag.max = p - 1, plot=FALSE))
        n <- p
        M <- matrix(NA,n,n)
        for(i in 1:n){
                if(i==1){
                        v <- 1:n
                }else if(i==n){
                        v <- rev(1:n)
                }else{
                        v <- c(i:1,2:(n-i+1))
                }
                M[i,] <- rho[v]
        }
        return(M)
}
```

```
> ar1 <- lapply(solve.yw(x.ts,1),round,3)
> ar2 <- lapply(solve.yw(x.ts,2),round,3)
> ar3 <- lapply(solve.yw(x.ts,3),round,3)
> ar_ols <- cbind(mylag(x.ts,1),mylag(x.ts,2),mylag(x.ts,3))
> olsfit <- lm(x.ts~ar_ols)
> olsfit$coefficients <- round(olsfit$coefficients,3)
```

Table 1: AR parameters for model orders 1,2 and 3.

Model	Method	ϕ_1	ϕ_2	ϕ_3	σ^2	AIC
AR(1)	Yule-Walker	0.641	–	–	0.584	-611.733
AR(2)	Yule-Walker	0.534	0.167	–	0.567	-642.141
AR(3)	Yule-Walker	0.526	0.143	0.045	0.566	-642.504
AR(1)	Least Squares	0.526	0.144	0.045	0.753	2591.428

Problem 2

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

arma.mod <- list()
for(i in 0:3)
        for(j in 0:3)
                arma.mod[[paste('ARMA(',i,',',j,')',sep='')]] <- arima(x.ts,c(i,0,j))

aics <- sapply(arma.mod,AIC)
gcvs <- sapply(arma.mod,gcv.arma)

best.arma.aic <- names(which(aics == min(aics)))
best.arma.gcv <- names(which(gcvs == min(gcvs)))

save(best.arma.aic,best.arma.gcv,arma.mod,aics,gcvs,file='output/2.Rdata')
```

Figure 6

It turned out that the same model ($ARMA(3,1)$) was selected as the best using both AIC and GCV. Looking at the plots though, all the models after $ARMA(1,1)$ are virtually the same.

```
> layout(rbind(c(1, 2)))
> s <- seq(1, length(gcvs), length.out = length(gcvs))
> smallnames <- function(v) paste("{\\small", names(v), "}")
> plot(s, gcvs, xlab = "", ylab = "GCV", pch = 20, xaxt = "n")
> text(s, gcvs, smallnames(gcvs), pos = 3, offset = 2, srt = 90)
> plot(s, aics, xlab = "", ylab = "AIC", pch = 20, xaxt = "n")
> text(s, aics, smallnames(aics), pos = 3, offset = 2, srt = 90)
```

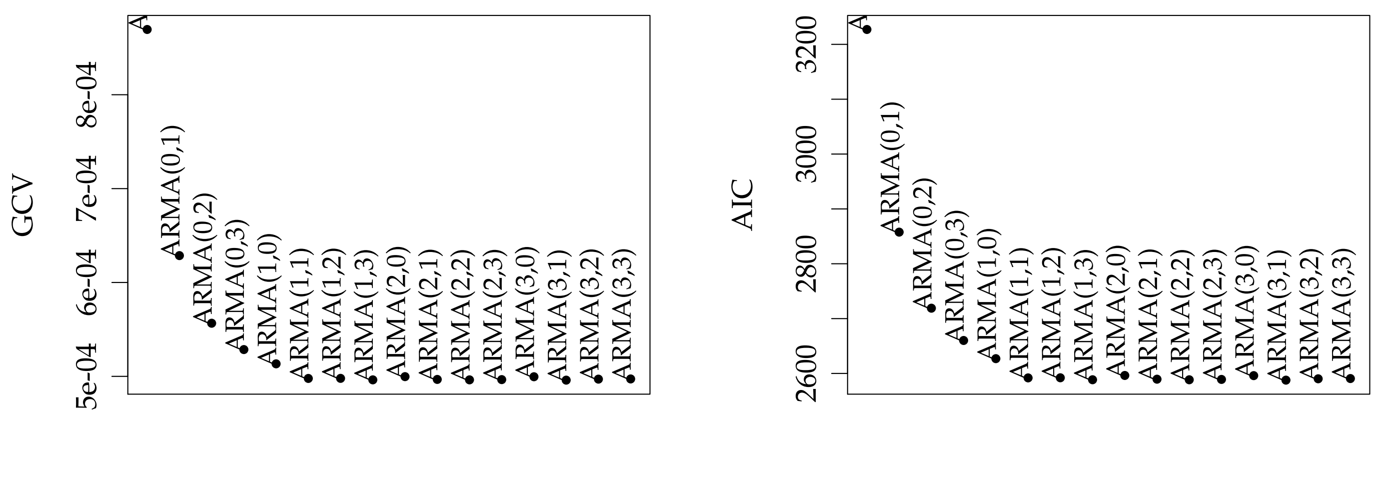

Figure 7: AIC and GCV as a function of p and q. Using both criteria, the $ARMA(3,1)$ model was selected as the best.

Problem 3

The code to simulate from the best arma model is shown below. The monthly mean and standatd deviation were of course captured. The monthly lag 1 correlation and the skew were not at all captured consistantly because this model is not designed to capture these statistics. The lag 1 correlation is a particularly good example of the deficiency of an *ARMA* model, as the simulated values are simply constant, not reflecting the variability in the historical data. The sequential peak analysis for this model did not seem terribly bad.

```
# Get the data
source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
if(file.exists('output/2.Rdata')) load('output/2.Rdata') else source('2.R')
require(pear)

# collect the simulations
sim <- array(NA,c(nyears,12,nsims))
b <- arma.mod[[best.arma.aic]]
p <- b$arma[1]
q <- b$arma[2]

sim.ts <- arima.sim(n=nyears*12*nsims, list(ar=b$coef[1:p],
                ma=b$coef[(p+1):(p+q)]), sd=sqrt(b$sigma2))
sim.ts <- ts(sim.ts,start=c(2001,1),frequency=12)

for(i in 1:nsims){
        sim[,,i] <-
        matrix(sim.ts[((i-1)*12*nyears+1):(i*12*nyears)],nyears,12,byrow=T)
}

for(i in 1:12){
        sim[,i,] <- sim[,i,] * x.sd[i] + x.mean[i]
        mon <- seq(i,length(sim.ts),by=12)
        sim.ts[mon] <- sim.ts[mon] * x.sd[i] + x.mean[i]
}

#Get the statistics
sim.agg <- ts.annual.mean(sim.ts)
stats <- sim.stats(sim,sim.agg,nsims)

x.ts.ann <- ts.annual.mean(x.ts.raw)

sp <- seqpeak.multi(sim,x.ts.raw,nsims)

save(stats,x.ts.ann,sim,sp,file='output/3.Rdata')
```

Figure 8: Code to simulte from the best *ARMA* model.

```
> layout(rbind(c(1,2),c(3,4)))
> boxplot(stats$mean,main='Mean',las=3)
>         lines(x.mean,col='red')
>         points(13,mean(x.ts.ann),col='red')
> boxplot(stats$sd,main='Standard Deviation',las=3)
>         lines(x.sd,col='red')
>         points(13,sd(x.ts.ann),col='red')
> boxplot(stats$lag1,main='Lag1 Correlation',las=3)
>         lines(x.lag1,col='red')
>         points(13,mylag(x.ts.ann,1,docor=TRUE),col='red')
> boxplot(stats$skew,main='Skew',
+         ylim=range(range(stats$skew),range(x.skew)),las=3)
>         lines(x.skew,col='red')
>         points(13,skew(x.ts.ann),col='red')
```

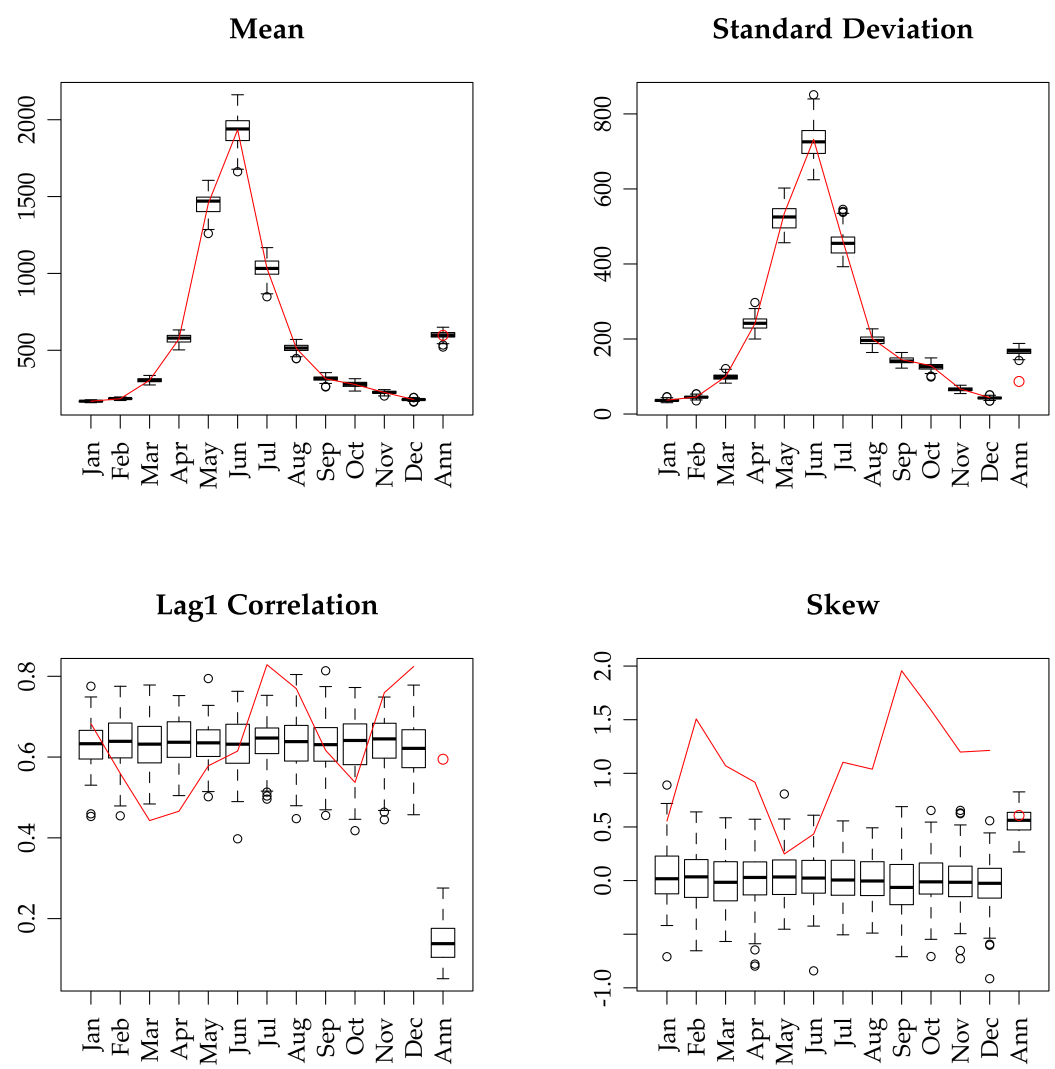

Figure 9: *ARMA* statistics. The furthest box on the right is the annual statistic.

```
seqpeak.multi <- function(sim,hist,nsims){

        days <- c(31,28,31,30,31,30,31,30,31,31,30,31)
        x.ts.s <- hist * days
        y <- c((0:9)/10,0.95)
        s.sim <- rep(0,length(y))
        sim.sy <- matrix(NA,nsims,length(y))
        for(i in 1:nsims){
                simf <- as.vector(t(sim[,,i]))* days
                for (j in 1:length(y))
                        s.sim[j] <- seqpeak.r(simf,y[j]*mean(x.ts.s))$s
                sim.sy[i,] <- s.sim
        }
        sim.sy <- as.data.frame(sim.sy)
        names(sim.sy) <- round(mean(x.ts.s)*y)
        s <- rep(0,length(y))
        for (j in 1:length(y))
                s[j] <- seqpeak.r(x.ts.s,y[j]*mean(x.ts.s))$s

        return(list(s=s,y=y,sim.sy=sim.sy))
}
```

Figure 10: Code to perform stochastic sequential peak analysis. Used in all subsequent problems.

```
> boxplot(sp$sim.sy, cex = 0.3, ylab = "Storage", xlab = "Yield")
> lines(1:length(sp$y), sp$s, col = "red", lwd = 2)
```

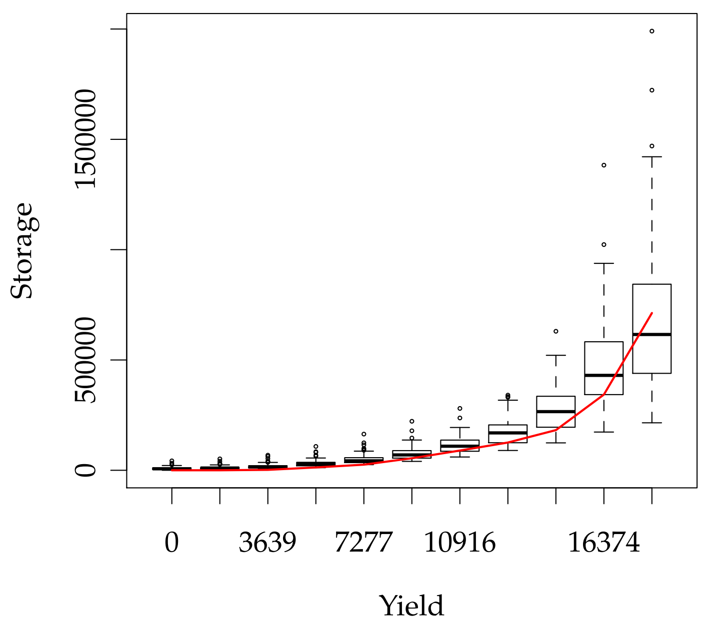

Figure 11: *ARMA* best model sequential peak analysis.

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

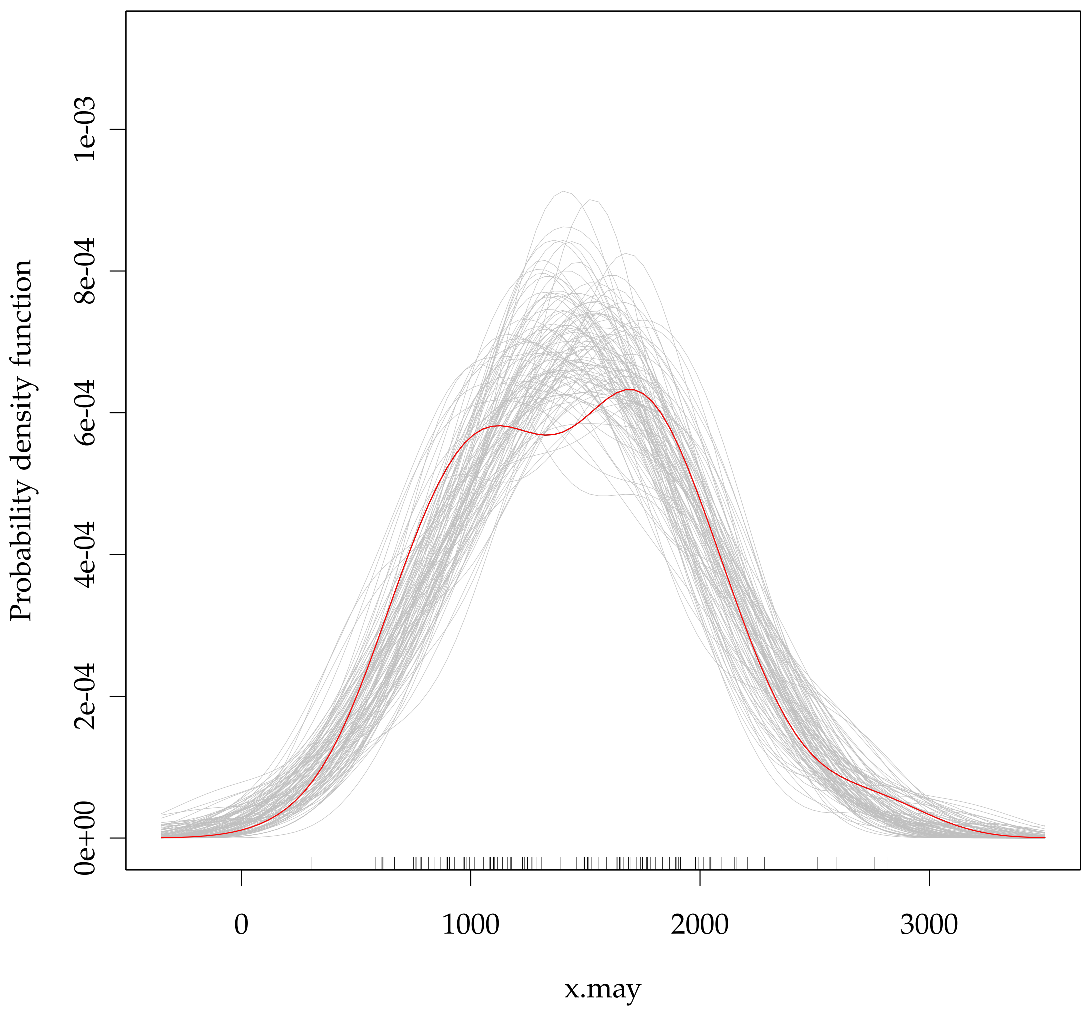

Figure 12: May pdf.

Problem 4

I used the pear package in R to calculate the $PAR(1)$ coefficients. The function to do simulations is also shown below. This model was able to effectively capture the seasonal mean, standard deviation, and lag 1 correlation (though the annual lag 1 was not captured). The skew was not captured (as seen in the may pdf) because I did not normalize each month. It is easy to see why seasonal parametric models quickly become cumbersome, as even a lag 1 seasonal model is unweildy.

```
# Get the data
source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
require(pear)

win <- frequency(x.ts.raw)

peacf <- peacf(x.ts.raw, lag.max=1, plot=FALSE)
pepacf <- pepacf(x.ts.raw, lag.max=1, plot=FALSE)
pearp <- pear(x.ts.raw, m = 1)
sim.seas.ts <- ar.sim.seas(pearp, peacf, (nsims+1)*nyears*win)
sim.seas.agg <- ts.annual.mean(sim.seas.ts)
sim.seas <- array(NA,c(nyears,12,nsims))
for(i in 1:nsims)
        sim.seas[,,i] <-
        matrix(sim.seas.ts[((i-1)*12*nyears+1):(i*12*nyears)],nyears,12,byrow=T)

stats.seas <- sim.stats(sim.seas,sim.seas.agg,nsims=nsims)

sp.seas <- seqpeak.multi(sim.seas,x.ts.raw,nsims)

save(stats.seas,sim.seas,sim.seas.agg,sp.seas,file='output/4.Rdata')
```

Figure 13: Code to fit and simulate from the $PAR(1)$ model.

```r
ar.sim.seas <- function(model, peacf, n, n.start = NA){

    # simulate from a peariodic AR model
    nobs <- length(model$residuals)
    p <- model$model.orders

    x.means <- peacf$means
    freq <- frequency(model$residuals)

        #find the starting period of the prediction
    end <- end(model$residuals)
    end <-  if(end[2] == freq)
                c(end[1] + 1, 1)
            else
                c(end[1], end[2] + 1)

    if(is.na(n.start))
        n.start <- 10*freq

    pred <- ts(rep(NA, n), start = end, frequency = freq)
    start.per <- end[2]
    per <- if(start.per == 1)
                rep(start.per:freq, length.out = n)
           else if(start.per == frequency(model$residuals))
                rep(c(freq, 1:(freq-1)), length.out = n)
           else
                rep(c(start.per:freq,1:(start.per-1)), length.out = n)

    per <- ts(per, start = end, frequency = freq)
    len <- 10
    modvec <- rep(1:freq,len)

    for(t in 1:n){
        which.per <- modvec[((len-1)*freq + per[t] - p[per[t]]):((len-1)*freq - 1 + per[t])]
        m <- x.means[which.per]

        lastp <- if(t <= p[per[t]])
                    c(m[(length(m) - p[per[t]] + t):length(m)], pred[1:(t - 1)])[1:p[per[t]]]
                 else
                    pred[(t - p[per[t]]):(t - 1)]

        e <- rnorm(1,sd = sqrt(model$resvar[per[t]]))

        these.phis <- model$phi[per[t],1:p[per[t]]]

        pred[t] <- sum(these.phis * (lastp - m)) + x.means[per[t]] + e

    }
    pred <- pred[(n.start + 1):n]
    pred <- ts(pred, start = end, frequency = freq)

    return(pred)
}
```

Figure 14: Code to simulate from a *PAR* model.

```
> layout(rbind(c(1,2),c(3,4)))
> boxplot(stats.seas$mean,main='Mean',las=3)
>         lines(x.mean,col='red')
>         points(13,mean(x.ts.ann),col='red')
> boxplot(stats.seas$sd,main='Standard Deviation',las=3)
>         lines(x.sd,col='red')
>         points(13,sd(x.ts.ann),col='red')
> boxplot(stats.seas$lag1,main='Lag1 Correlation',las=3)
>         lines(x.lag1,col='red')
>         points(13,mylag(x.ts.ann,1,docor=TRUE),col='red')
> boxplot(stats.seas$skew,main='Skew',
+         ylim=range(range(stats.seas$skew),range(x.skew)),las=3)
>         lines(x.skew,col='red')
>         points(13,skew(x.ts.ann),col='red')
```

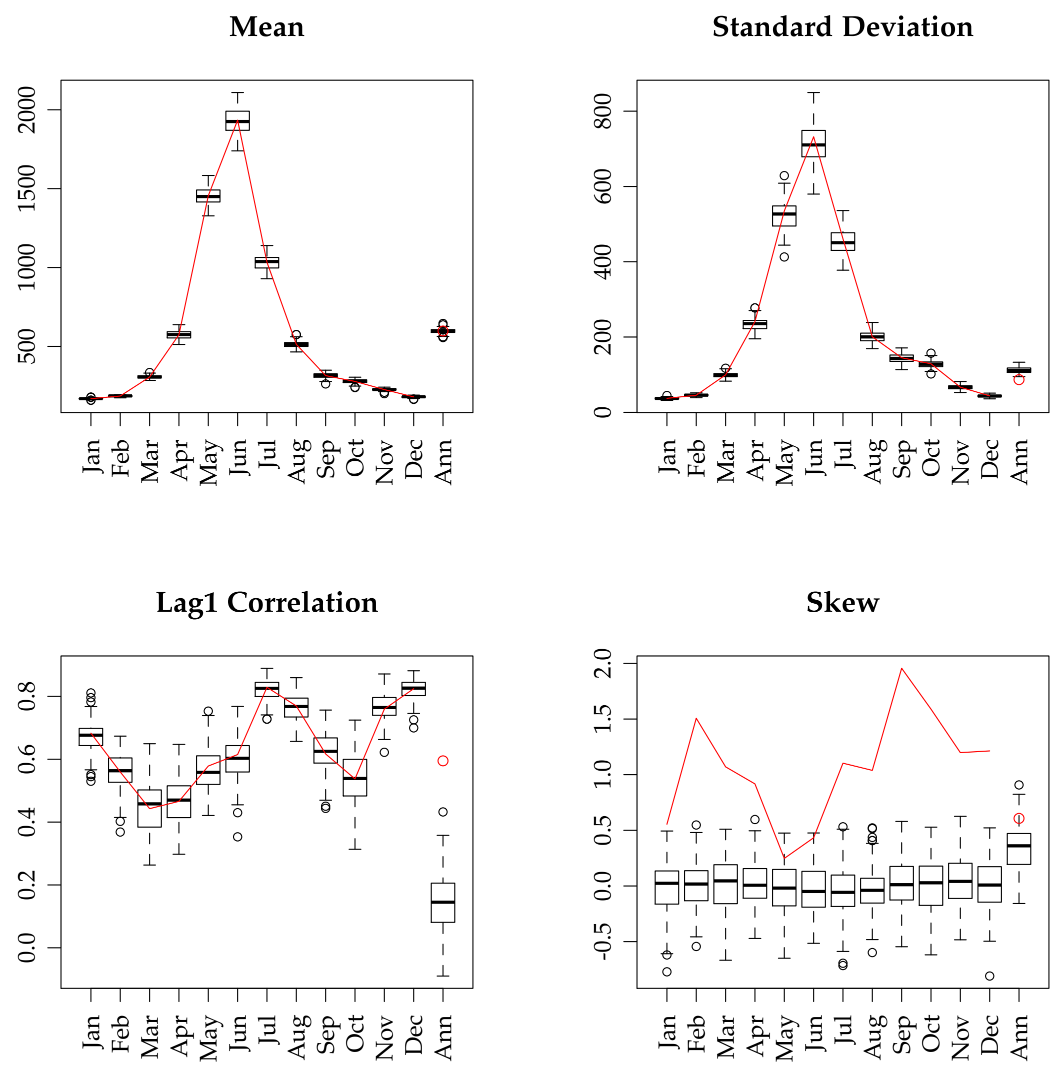

Figure 15: *PAR* statistics. The furthest box on the right is the annual statistic.

```
> boxplot(sp.seas$sim.sy, cex = 0.3, ylab = "Storage", xlab = "Yield")
> lines(1:length(sp.seas$y), sp.seas$s, col = "red", lwd = 2)
```

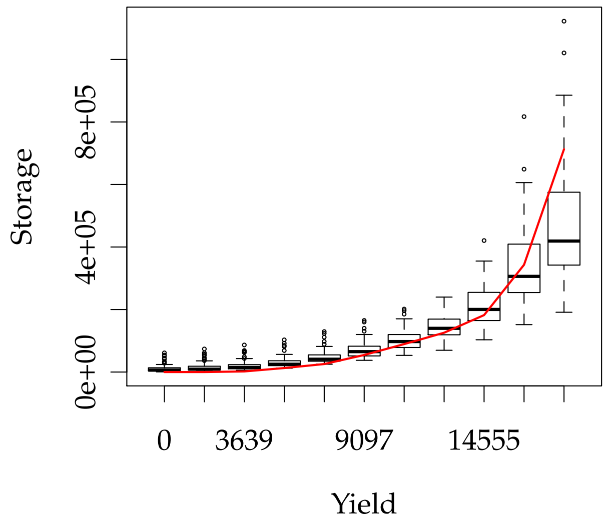

Figure 16: *PAR* best model sequential peak analysis.

```
> for(i in 1:nsims){
+         boxpt[i,] <- sm.density(sim.seas[,5,i],
+                 eval.points=pt,display='none')$estimate
+         lines(pt,boxpt[i,],col='gray',lwd=.5)
+ }
> sm.density(x.may,add=T,col='red',lwd=2)
```

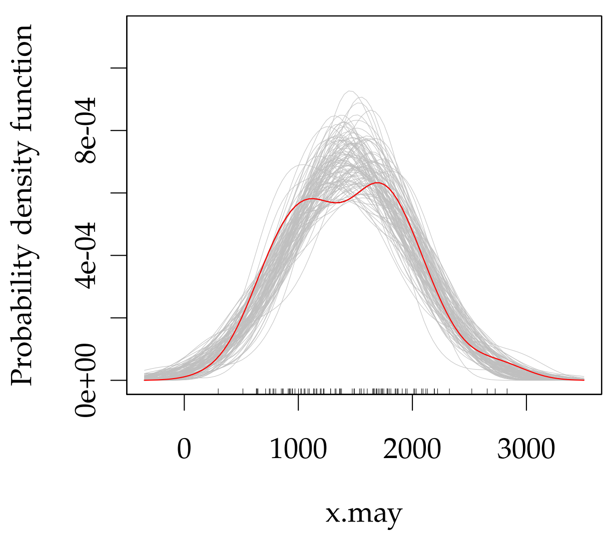

Figure 17: May pdf.

Problem 5

For this problem I implemented the modifed K-NN method. The code is show below to as well as the function to generate simulaitons. This method was able to effectivel capture monthly lag 1 correlation, skew, and the bimodality of the PDF. Only annual lag 1 correlation and skew were not captured well. Surprisingly, the modified K-NN preformed worse than the *PAR* at the high end of the sequent peak analysis, though the difference may not be significant.

```
# Get the data
source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
require(locfit)
require(pear)

st <- mean(x[,1])
k <- round(sqrt(length(x[,1])))

w <- numeric(k)
for(i in 1:k)
        w[i] <- (1/i)/(sum(1/(1:i)))
w <- cumsum(w/sum(w))

sim.mknn.v <- mknn(x,nsims=nsims)

sim.mknn.ts <- ts(sim.mknn.v,start=c(2001,1),frequency=12)
sim.mknn.agg <- ts.annual.mean(sim.mknn.ts)
sim.mknn <- array(NA,c(nyears,12,nsims))
for(i in 1:nsims)
        sim.mknn[,,i] <-
        matrix(sim.mknn.ts[((i-1)*12*nyears+1):(i*12*nyears)],nyears,12,byrow=T)

stats.mknn <- sim.stats(sim.mknn, sim.mknn.agg, nsims=nsims,nyears=nyears)

sp.mknn <- seqpeak.multi(sim.mknn,x.ts.raw,nsims)


save(sim.mknn,sim.mknn.agg,stats.mknn,sp.mknn,file='output/5.Rdata')
```

Figure 18: Code to simulte from the best *ARMA* model.

```
mknn <- function(x,nsims,nyears=nrow(x)){

                #fit locfit models for modified knn
        models <- list()
        for( i in 1:12 ){

                models[[i]] <- list()
                if(i == 12){
                        this.x <- x[,i]
                        this.y <- x[,1]
                }else{
                        this.x <- x[,i]
                        this.y <- x[,i+1]
                }
                best <- best.par(this.x,this.y,f=gcvplot)
                models[[i]]$lf <- locfit(this.y~this.x, deg=best$p, alpha=best$a, kern='bisq')
                models[[i]]$pred <- predict(models[[i]]$lf,cbind(x[,i]))
                models[[i]]$res <- this.y - models[[i]]$pred
        }

        st <- mean(x[,1])
        k <- round(sqrt(length(x[,1])))

        w <- numeric(k)
        for(i in 1:k)
                w[i] <- (1/i)/(sum(1/(1:i)))
        w <- cumsum(w/sum(w))

        sim.mknn.v <- numeric(nsims*12*nyears)
        for(i in 1:(nsims*12*nyears-1)){

                        #Loop through and simulate
                val <- if(i==1) st else sim.mknn.v[i]
                m <- ifelse((i %% 12) == 0,12,i %% 12)
                mp1 <- (i %% 12) + 1
                this.pred <- predict(models[[m]]$lf,cbind(val))
                this.dist <- as.matrix(dist(c(val,x[,m])))[,1]
                neighbors <- order(this.dist)[2:(k+1)] - 1

                        #resample the residuals
                r <- runif(1)
                this.neighbor <- which(order(c(r,w))==1)
                this.res <- models[[m]]$res[neighbors][this.neighbor]

                sim.mknn.v[i+1] <- this.pred + this.res
        }
        return(sim.mknn.v)
}
```

Figure 19: Code to simulte from the Modified K-NN. The `best.par()` function to calculate the best alpha and degree is shown in the appendix.

```
> layout(rbind(c(1,2),c(3,4)))
> boxplot(stats.mknn$mean,main='Mean',las=3)
>         lines(x.mean,col='red')
>         points(13,mean(x.ts.ann),col='red')
> boxplot(stats.mknn$sd,main='Standard Deviation',las=3)
>         lines(x.sd,col='red')
>         points(13,sd(x.ts.ann),col='red')
> boxplot(stats.mknn$lag1,main='Lag1 Correlation',las=3)
>         lines(x.lag1,col='red')
>         points(13,mylag(x.ts.ann,1,docor=TRUE),col='red')
> boxplot(stats.mknn$skew,main='Skew',
+         ylim=range(range(stats.mknn$skew),range(x.skew)),las=3)
>         lines(x.skew,col='red')
>         points(13,skew(x.ts.ann),col='red')
```

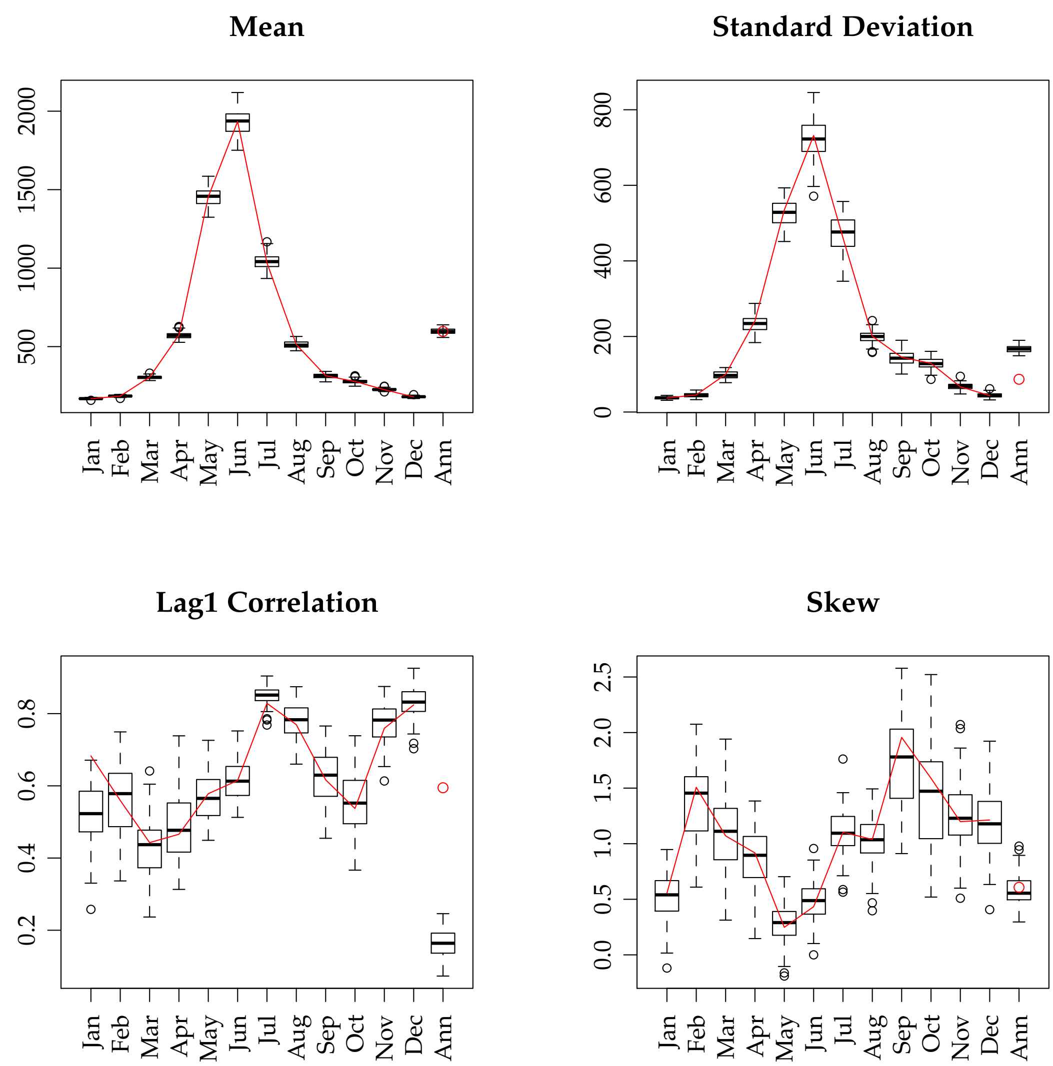

Figure 20: Modified K-NN statistics. The furthest box on the right is the annual statistic.

```
> boxplot(sp.mknn$sim.sy, cex = 0.3, ylab = "Storage", xlab = "Yield")
> lines(1:length(sp.mknn$y), sp.mknn$s, col = "red", lwd = 2)
```

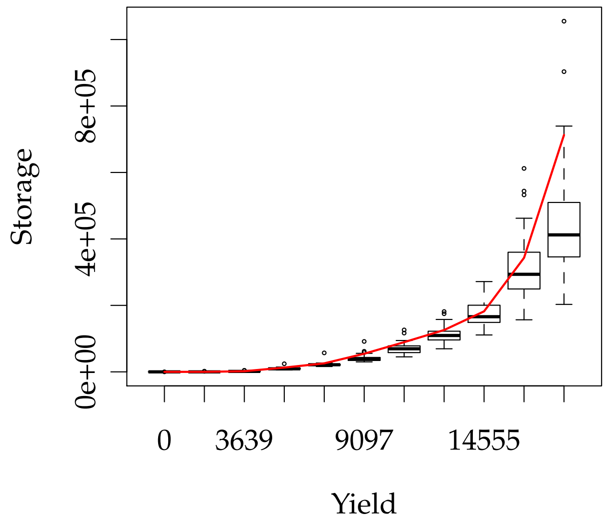

Figure 21: Modified K-NN sequential peak analysis.

```
> for(i in 1:nsims){
+         boxpt[i,] <- sm.density(sim.mknn[,5,i],
+                 eval.points=pt,display='none')$estimate
+         lines(pt,boxpt[i,],col='gray',lwd=.5)
+ }
> sm.density(x.may,add=T,col='red',lwd=2)
```

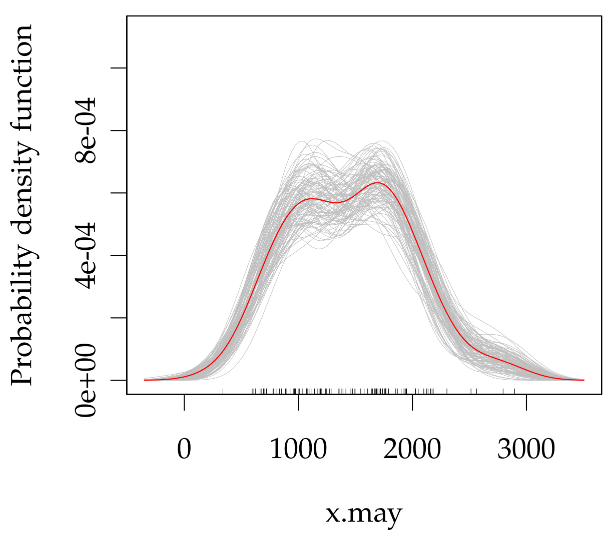

Figure 22: Modified K-NN May pdf.

Problem 6

The code which performas the interannual Modified K-NN is shown below as well as the function that returns the simulations. I believe there may be an error in the implemenation because I saw a degraded ability to capture monthly statistics. Especially skew and lag 1 correlation ware captured less well than the Modified K-NN. The May pdf was also not captured as well. On the other hand, the sequent peak analysis did show an improvement over the regular modified K-NN. This indicates that possibly the simulations are having the desired effect but simply at the cost of other stastics.

```
# Get the data
source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
require(locfit)
require(pear)

        #Interannual Modified KNN
sim.imknn.v <- imknn(x,nsims=nsims)

sim.imknn.ts <- ts(sim.imknn.v,start=c(2001,1),frequency=12)
sim.imknn.agg <- ts.annual.mean(sim.imknn.ts)
sim.imknn <- array(NA,c(nyears,12,nsims))
for(i in 1:nsims)
        sim.imknn[,,i] <-
        matrix(sim.imknn.ts[((i-1)*12*nyears+1):(i*12*nyears)],nyears,12,byrow=T)

stats.imknn <- sim.stats(sim.imknn, sim.imknn.agg, nsims=nsims,nyears=nyears)

sp.imknn <- seqpeak.multi(sim.imknn,x.ts.raw,nsims)


save(sim.imknn,sim.imknn.agg,stats.imknn,sp.imknn,file='output/6.Rdata')
```

Figure 23: Code to simulte from Interannual Modified K-NN.

```
imknn <- function(x,nsims,nyears=nrow(x)){

                #fit locfit models for interannual modified knn
        x.ann <- apply(x,1,sum)
        models <- list()
        for( i in 1:12 ){

                        #get the sum of the last 12 mponths of flow for every point
                last.12 <- last.12.sum(x)
                models[[i]] <- list()
                if(i == 12) this.y <- x[,1][-1] else this.y <- x[,i+1][-1]
                this.x <- cbind(x[,i][-1],last.12[,i])

                best <- best.par(this.x,this.y,f=gcvplot)
                models[[i]]$lf <- locfit(this.y~this.x, deg=best$p, alpha=best$a, kern='bisq')
                models[[i]]$pred <- predict(models[[i]]$lf,cbind(x[,i][-1],last.12[,i]))
                models[[i]]$res <- this.y - models[[i]]$pred
        }
        st <- mean(x[,1])
        k <- round(sqrt(length(x[,1])))
        w <- numeric(k)
        for(i in 1:k)
                w[i] <- (1/i)/(sum(1/(1:i)))
        w <- cumsum(w/sum(w))

        sim.imknn.v <- numeric(nsims*12*nyears)
        for(i in 1:(nsims*12*nyears-1)){

                        #Loop through and simulate
                val <- if(i==1) st else sim.imknn.v[i]
                last.ann <- if(i <= 12) mean(x.ann) else sum(sim.imknn.v[(i-12):(i-1)])

                m <- ifelse((i %% 12) == 0,12,i %% 12)
                mp1 <- (i %% 12) + 1
                newp <- c(val,last.ann)
                this.pred <- predict(models[[m]]$lf,rbind(newp))
                dat <- cbind(x[,m],x.ann)[1:(nyears-1),]
                this.dist <- as.matrix(dist(rbind(newp,dat)))[,1]
                neighbors <- order(this.dist)[2:(k+1)] - 1

                #resample the residuals
                r <- runif(1)
                this.neighbor <- which(order(c(r,w))==1)
                this.res <- models[[m]]$res[neighbors][this.neighbor]

                sim.imknn.v[i+1] <- this.pred + this.res
        }
        return(sim.imknn.v)
}
```

Figure 24: Code to simulte from the Modified K-NN. The `best.par()` function to calculate the best alpha and degree is shown in the appendix.

```
> layout(rbind(c(1,2),c(3,4)))
> boxplot(stats.imknn$mean,main='Mean',las=3)
>         lines(x.mean,col='red')
>         points(13,mean(x.ts.ann),col='red')
> boxplot(stats.imknn$sd,main='Standard Deviation',las=3)
>         lines(x.sd,col='red')
>         points(13,sd(x.ts.ann),col='red')
> boxplot(stats.imknn$lag1,main='Lag1 Correlation',las=3)
>         lines(x.lag1,col='red')
>         points(13,mylag(x.ts.ann,1,docor=TRUE),col='red')
> boxplot(stats.imknn$skew,main='Skew',
+         ylim=range(range(stats.imknn$skew),range(x.skew)),las=3)
>         lines(x.skew,col='red')
>         points(13,skew(x.ts.ann),col='red')
```

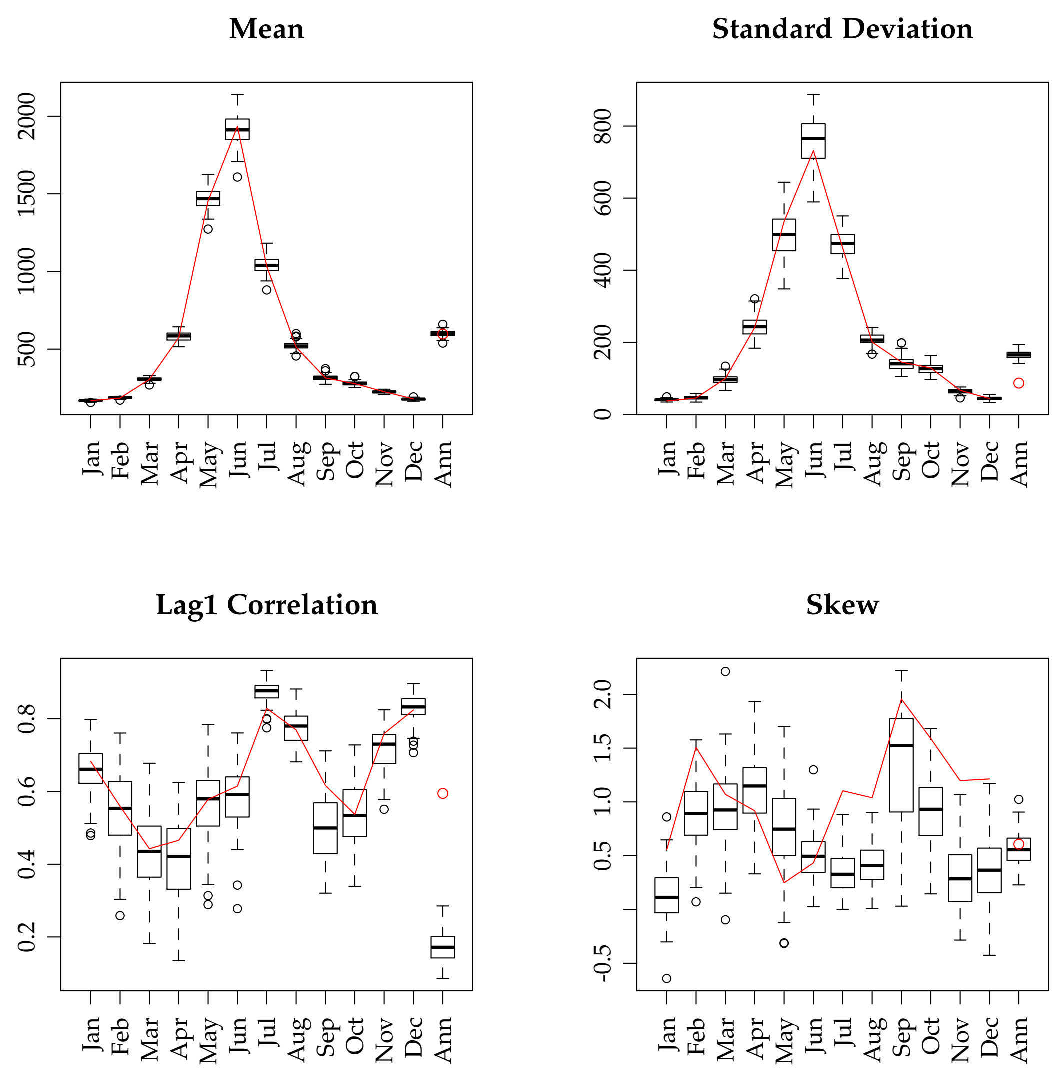

Figure 25: Modified K-NN statistics. The furthest box on the right is the annual statistic.

```
> boxplot(sp.imknn$sim.sy, cex = 0.3, ylab = "Storage", xlab = "Yield")
> lines(1:length(sp.imknn$y), sp.imknn$s, col = "red", lwd = 2)
```

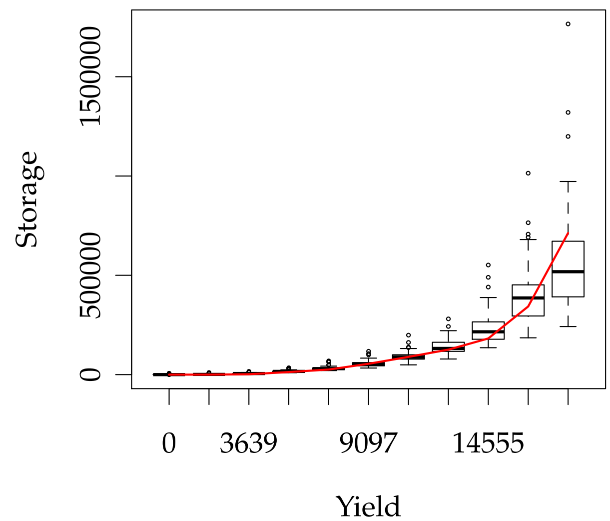

Figure 26: Interannual Modified K-NN sequential peak analysis.

```
> for(i in 1:nsims){
+         boxpt[i,] <- sm.density(sim.imknn[,5,i],
+                 eval.points=pt,display='none')$estimate
+         lines(pt,boxpt[i,],col='gray',lwd=.5)
+ }
> sm.density(x.may,add=T,col='red',lwd=2)
```

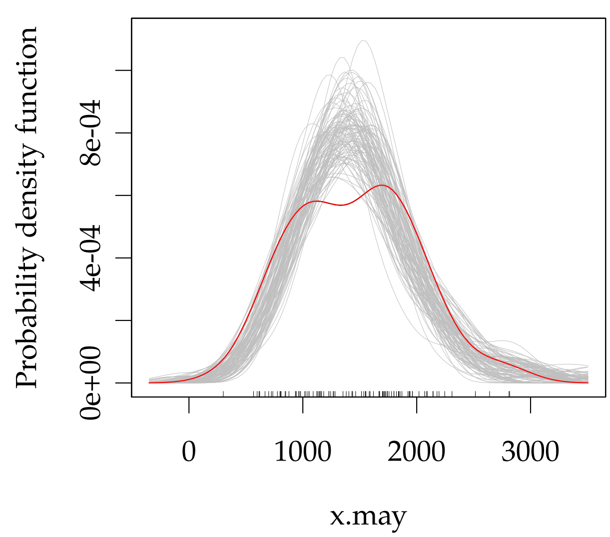

Figure 27: Interannual Modified K-NN May pdf. I believe there may be an error in my implementation since adding the interannual information should not have degraded the models ability to represent the monthly pdf.

Problem 7

The code to calculate the best lag based on GCV and MI are shown below. Based on the calcualtions, the best lag is 1 after all, with a best K of 38.

```
# Get the data
source('lib.R')
if(file.exists('output/1.Rdata')) load('output/1.Rdata') else source('1.R')
require(locfit)
options(warn=-1)

y.may <- x[,5]
x.lag <- list()
gcvs <- ks <- numeric(1)
for(i in 1:4){
        x.lag[[i]] <- cbind(x[,(5-i):4])
        bp <- best.par(x.lag[[i]],y.may,f=gcvplot,maxk=1000)
        gcvs[i] <- bp$gcv
        ks[i] <- bp$a * nrow(x.lag[[i]])
}
#tab <- rbind(gcvs,ks,mi)
#print(xtable(tab))
mi <- aminf(as.vector(t(x)),4)
```

Figure 28: Code to calculate best lag and K.

Table 2: GCV, K and MI for each lag up to 4. Lag 1 appears to be the best with both GCV and MI criteria, corresponding to a K of 38.

Lag	1	2	3	4
GCV	172051.29	178349.56	175870.69	198287.50
MI	0.15	0.30	0.22	0.13
K	38.00	85.50	71.25	95.00

Appendix - Auxilary functions

```
sim.stats <-
function(sim,sim.agg,nsims,nyears=nrow(sim[,,1]),start=c(1906,1)){


        stats <- list()
        stats$mean <- stats$sd <- stats$skew <- stats$lag1 <- matrix(NA,nsims,13)

        # seasonal simulated stats
        for(i in 1:12){

                this.mon <- sim[,i,]
                stats$mean[,i] <- apply(this.mon,2,mean)
                stats$sd[,i]   <- apply(this.mon,2,sd)
                stats$skew[,i] <- apply(this.mon,2,skew)

        }
        #lag 1 correlation
        for(i in 1:nsims){
                this.sim <- sim[,,i]
                this.sim.ts <- ts(array(t(this.sim)),start=start,frequency=12)
                stats$lag1[i,1:12] <- peacf(this.sim.ts,plot=FALSE,lag.max=1)$acf
        }

        # aggregate seasonal statistics
        stats$mean[,13] <- wapply(sim.agg,mean,nyears)
        stats$sd[,13] <-   wapply(sim.agg,sd,nyears)
        stats$skew[,13] <- wapply(sim.agg,skew,nyears)
        stats$lag1[,13] <- wapply(sim.agg,mylag,nyears,lag=1,docor=T)

        #setup for plotting
        stats <- lapply(stats,as.data.frame)
        mon <- c('Jan','Feb','Mar','Apr','May','Jun',
                'Jul','Aug','Sep','Oct','Nov','Dec','Ann')
        for(i in 1:length(stats)) names(stats[[i]]) <- mon

        return(stats)
}
```

Figure 29: Function to calulate simulation statistics.

```
        #returns the best alpha and degree
best.par <-
function(x,y, a = seq(0.2,1.0,by=0.05), n = length(a), f=c(gcvplot,aicplot),...){

                # get the gcv values for all combinations of deg and alpha
        d1 <- f(y~x, deg=1, alpha=a, kern='bisq', scale=T,...)
        d2 <- f(y~x, deg=2, alpha=a, kern='bisq', scale=T,...)

        gcvs <- c(d1$values,d2$values)
        best <- order(gcvs)[1]
                #get the best alpha and degree
        bestd <- c(rep(1,n),rep(2,n))[best]
        bestalpha <- c(a,a)[best]

        return(list(p=bestd,a=bestalpha,gcv=gcvs[best]))
}
```

Figure 30: Function to calulate the best degree and α for a locfit model.

```
skew <- function(x){
        n <- length(x)
        nfact <- n/((n - 1) * (n - 2))
        xm <- mean(x)
        xv <- sqrt(var(x))
        skw <- sum((x - xm)^3)
        skew <- (nfact * skw)/xv^3
        return(skew)
}
```

Figure 31: Skew function.

```
seqpeak.r <- function(q,r=0.7*mean(q)){
        #   r is used as input so that for simulations that may have different mean
        #  the same release may be used.  r may be a vector to impose a release pattern
        # net change in each time period.  Also this takes care of vector wrap around
        rmq <- r-q
        # if r and q are of different lengths
        n <- length(q)
        k <- rep(0,n)
        k[1] <- max(rmq[1],0)     #  first time k[0] is 0
        for (j in 2:n){
                k[j] <- max(k[j-1]+rmq[j],0)
        }
        s=max(k)
        return(list(s=s,k=k))
}
```

Figure 32: Single sequential peak calulation.

```
mylag <-
function(x,lag,docor=FALSE){

    if(lag>length(x)) warning("Lag is larger than input vector length, returning NA's")

    if(lag<0)  lagn = c(rep(NA,abs(lag)),x[-(length(x):(length(x)+lag+1))])
    if(lag==0) lagn = x
    if(lag>0)  lagn = c(x[-(1:lag)],rep(NA,lag))

    remove = !is.na(lagn) & !is.na(x)
    if(docor){
                return(cor(x[remove],lagn[remove]))
    }else{
                return(lagn)
        }
}
```

Figure 33: Returns lagged version of timeseries or optionally the lag-n correlation.

```
gcv.arma <-
function(m){

        n <- length(m$residuals)
        n*m$sigma2/(n-length(m$coef))^2

}
```

Figure 34: calculates the gcv of an $ARMA$ model.

```
ts.annual.mean <-
function(x){

    options(warn=-1)
    syear <- start(x)[1]
        sper <- start(x)[2]
        eyear <- end(x)[1]
        eper <- end(x)[2]
    nst <- numeric(nyears)

        #discard incomplete years at beginning or end of series
        if(sper != 1) {
                syear <- syear + 1
                sper <- 1
        }
        if(eper != frequency(x)){
                eyear <- eyear - 1
                eper <- frequency(x)
        }
        nyears <- length(unique(floor(time(x))))
        mat <- matrix(window(x,syear,c(eyear,eper)),nyears,frequency(x))

    nst <- ts(apply(mat,1,mean), start = c(syear,1), frequency = 1)
    return(nst)

}
```

Figure 35: Returns the annual mean of a timeseries object.

```
wapply <-
function(x, fun, win.len = length(x), ...){

    r <- (length(x) %% win.len)
    if(r > 0) x <- x[1:(length(x)-r)]
    stack <- matrix(x,nrow = win.len)
    return(apply(stack,2,fun,...))

}
```

Figure 36: Apply a funciton to successive windows of a time series.

```
last.12.sum <- function(x){
        v <- as.vector(t(x))
        n <- length(v)
        mat <- matrix(NA,nrow(x)-1,ncol(x))
        i <- j <- 1
        for(t in (ncol(x)+1):n){
                if(j > ncol(mat)){
                        j <- 1
                        i <- i + 1
                }
                mat[i,j] <- sum(v[(t-12):(t-1)])
                j <- j + 1
        }
        return(mat)
}
```

Figure 37: Calculates the sum of flow for every point in a matrix.