Mikhail Gaerlan (914437675)
21 April 2017
Assignment 4 STA 243 Lee

1

The black line represents the true function, the blue line represents the unpenalized regression spline, and the red line represents the penalized regression spline. The `nlm` command in R was used to minimize the following expressions.

1.a

1.a.1 Cross Validation

1.a.2 Generalized Cross Validation

1.b Corrected Akaike Information Criterion

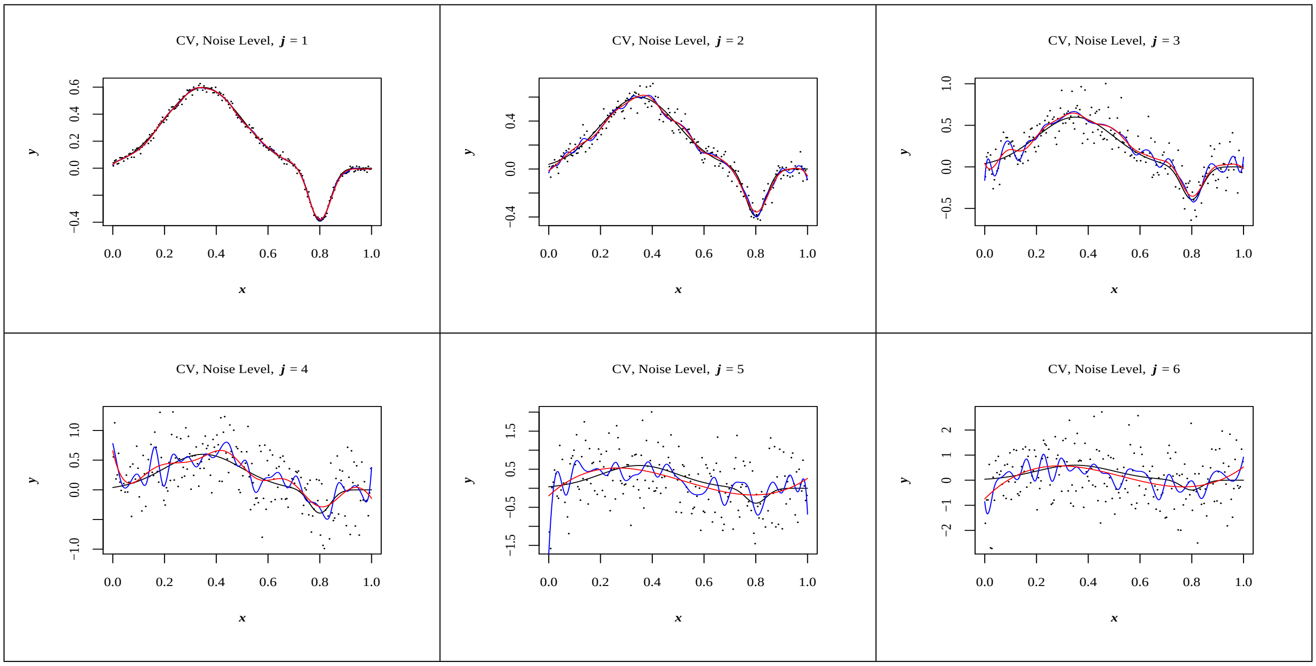
1.c Risk Minimization

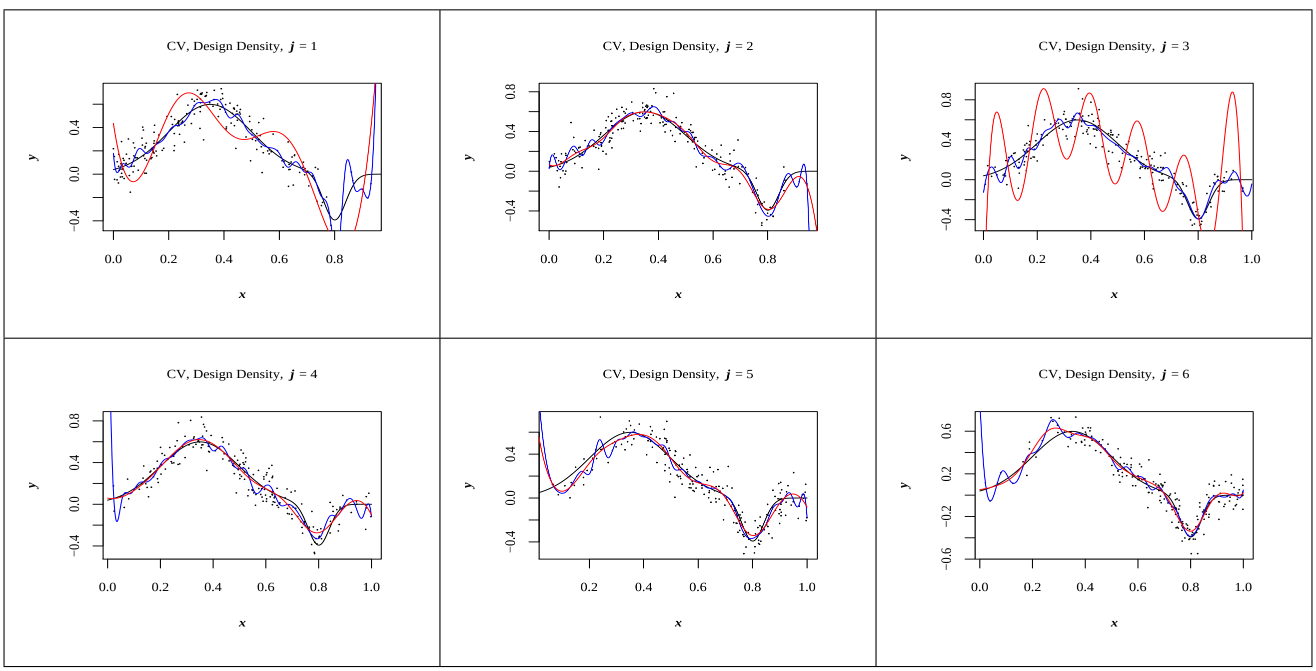
$$\begin{aligned}
\mathbb{E}\|\boldsymbol{y}-\hat{\mathbf{f}}_\lambda\|^2 &= \mathbb{E}\|\boldsymbol{y}-\boldsymbol{H}_\lambda\boldsymbol{y}\|^2 \\
&= \mathbb{E}\|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{y}\|^2 \\
&= \mathbb{E}\|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\left(\boldsymbol{f}+\boldsymbol{e}\right)\|^2 \\
&= \mathbb{E}\|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}+\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{e}\|^2 \\
&= \mathbb{E}\|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{e}\|^2+2\mathbb{E}\left[\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\cdot\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{e}\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}\left[\left(\boldsymbol{f}-\boldsymbol{H}_\lambda\boldsymbol{f}\right)\cdot\left(\boldsymbol{e}-\boldsymbol{H}_\lambda\boldsymbol{e}\right)\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}\left[\boldsymbol{f}\cdot\left(\boldsymbol{e}-\boldsymbol{H}_\lambda\boldsymbol{e}\right)-\boldsymbol{H}_\lambda\boldsymbol{f}\cdot\left(\boldsymbol{e}-\boldsymbol{H}_\lambda\boldsymbol{e}\right)\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}\left[\boldsymbol{f}\cdot\boldsymbol{e}-\boldsymbol{f}\cdot\boldsymbol{H}_\lambda\boldsymbol{e}-\boldsymbol{H}_\lambda\boldsymbol{f}\cdot\boldsymbol{e}+\boldsymbol{H}_\lambda\boldsymbol{f}\cdot\boldsymbol{H}_\lambda\boldsymbol{e}\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}\left[\boldsymbol{f}\cdot\boldsymbol{e}-\boldsymbol{H}_\lambda^T\boldsymbol{f}\cdot\boldsymbol{e}-\boldsymbol{H}_\lambda\boldsymbol{f}\cdot\boldsymbol{e}+\boldsymbol{H}_\lambda^T\boldsymbol{H}_\lambda\boldsymbol{f}\cdot\boldsymbol{e}\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathbb{E}\left[\left(\boldsymbol{I}-\boldsymbol{H}_\lambda^T-\boldsymbol{H}_\lambda+\boldsymbol{H}_\lambda^T\boldsymbol{H}_\lambda\right)\boldsymbol{f}\cdot\boldsymbol{e}\right] \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\mathrm{Cov}\left(\boldsymbol{e}\right)\left(\boldsymbol{I}-\boldsymbol{H}_\lambda-\boldsymbol{H}_\lambda^T-\boldsymbol{H}_\lambda^T\boldsymbol{H}_\lambda\right) \\
&= \|\left(\boldsymbol{I}-\boldsymbol{H}_\lambda\right)\boldsymbol{f}\|^2+\sigma^2\left\{\mathrm{tr}\left(\boldsymbol{H}_\lambda\boldsymbol{H}_\lambda^T\right)-2\mathrm{tr}\left(\boldsymbol{H}_\lambda\right)+n\right\}
\end{aligned}$$

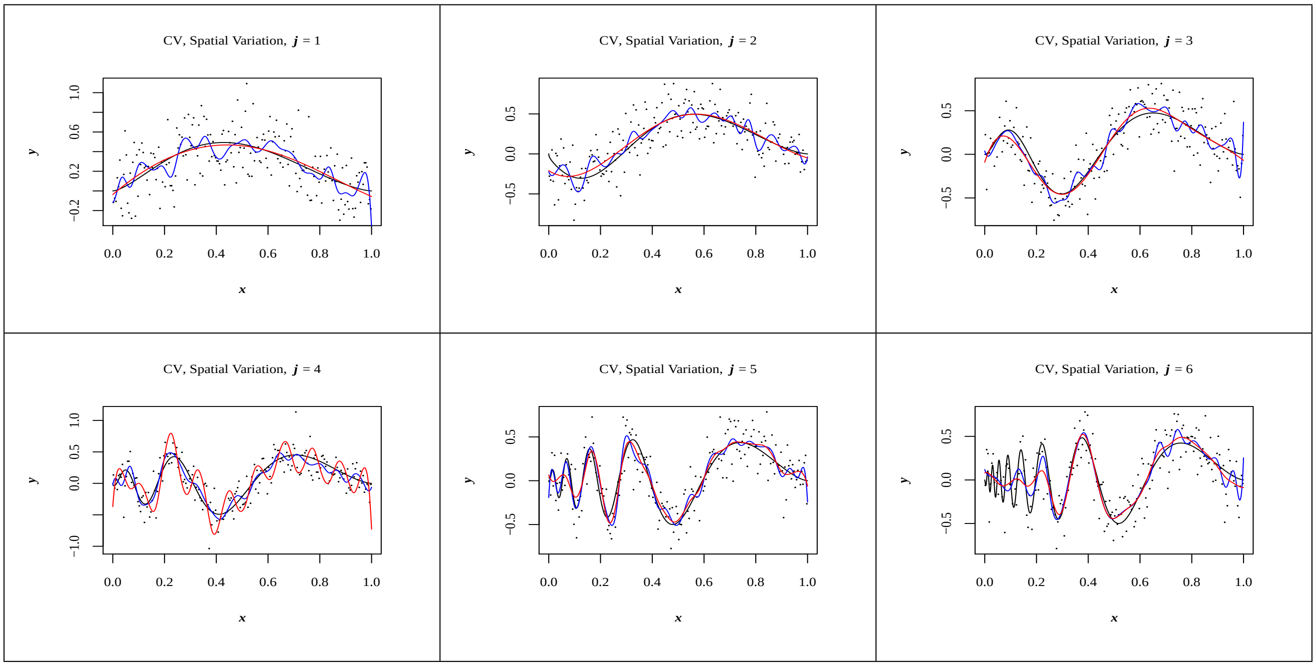
```
require("LaplacesDemon")
require("extrafont")
par(family = 'Times New Roman')
setwd("/Users/mikhailgaerlan/Box Sync/Education/UC Davis/2016-2017 Spring/STA 243 Computational
Statistics/Assignments/Assignment 5")
rm(list=ls())
set.seed(0518)

# Parameter Values
p = 3
n = 200
k = 30
export = T

phi = function(u){
  return(exp(-u^2/2)/sqrt(2*pi))
}

f = function(x){
  return(1.5*phi((x-0.35)/0.15)-phi((x-0.8)/0.04))
}

modelf = function(x,knots,betas){
  sums = 0
  for(i in 1:(p+1)){
    sums = sums + betas[i]*x^(i-1)
  }
  for(i in 1:k){
    sums = sums + betas[p+1+i]*(max((x-knots[i]),0))^p
  }
  return(sums)
}

designmatrix = function(x,p,k){
  n = length(x)
  X = array(0,c(n,p+1+k))
  for(i in 1:(p+1)){
    for(j in 1:n){
      X[j,i] = x[j]^(i-1)
    }
  }
  for(i in 1:k){
    for(j in 1:n){
      X[j,p+1+i] = (max(c((x[j]-knots[i]),0)))^p
    }
  }
  return(X)
}

for(o in 1:4){
  if (o == 1){
    print("Cross Validation")
    method = "CV"
  } else if(o == 2){
    print("Generalized Cross Validation")
    method = "GCV"
  } else if(o == 3){
    print("AICc")
    method = "AICc"
  } else if(o == 4){
    print("Risk")
    method = "Risk"
  }
  for(l in 1:4){
    if (l == 1){
      print("Noise Level")
      functions = "Noise Level"
    } else if(l == 2){
      print("Design Density")
      functions = "Design Density"
    } else if(l == 3){
      print("Spatial Variation")
      functions = "Spatial Variation"
    } else if(l == 4){
      print("Variance Function")
      functions = "Variance Function"
    }
    for(m in 1:6){
      if (export) file_name =
paste("Graphs/",toString(o),"/",toString(l),"/assignment5_a_",toString(o),"_",toString(l),"_",toString(m),".pdf",sep=""
```

```
    if (export) pdf(file = file_name,width = 4,height=3.5,family="Times New Roman")
    # Data generation
    xtest = seq(0,1,0.001)
    if(l==1){
      # Noise Level
      f = function(x){
        return(1.5*phi((x-0.35)/0.15)-phi((x-0.8)/0.04))
      }
      x = (1:200-0.5)/n
      knots = min(x) + 1:k*(max(x)-min(x))/(k+1)
      sigma = 0.02 + 0.04*(m-1)^2
      epsilon = rnorm(n,0,1)
      truef = f(x)
      y = truef + sigma*epsilon
    }else if(l==2){
      # Design Density
      f = function(x){
        return(1.5*phi((x-0.35)/0.15)-phi((x-0.8)/0.04))
      }
      x = qbeta(runif(n,0,1),(m+4)/5,(11-m)/5)
      knots = min(x) + 1:k*(max(x)-min(x))/(k+1)
      sigma = 0.1
      epsilon = rnorm(n,0,1)
      truef = f(x)
      y = truef + sigma*epsilon
      data = cbind(x,y,truef)
      sorteddata = data[order(data[,1]),]
      x = sorteddata[,1]
      y = sorteddata[,2]
      truef = sorteddata[,3]
    }else if(l==3){
      # Spatial Variation
      f = function(x){
        return(sqrt(x*(1-x))*sin((2*pi*(1+2^((9-4*m)/5)))/(x+2^((9-4*m)/5))))
      }
      x = (1:200-0.5)/n
      knots = min(x) + 1:k*(max(x)-min(x))/(k+1)
      sigma = 0.2
      epsilon = rnorm(n,0,1)
      truef = f(x)
      y = truef + sigma*epsilon
    }else if(l==4){
      # Variance Function
      f = function(x){
        return(1.5*phi((x-0.35)/0.15)-phi((x-0.8)/0.04))
      }
      x = (1:200-0.5)/n
      knots = min(x) + 1:k*(max(x)-min(x))/(k+1)
      sigma = sqrt((0.15*(1+0.4*(2*m-7)*(x-0.5)))^2)
      epsilon = rnorm(n,0,1)
      truef = f(x)
      y = truef + sigma*epsilon
    }
    plot_name = bquote(paste(.(method),", ",.(functions),",  ",italic(j)," = ",.(m)))

plot(x,y,pch=20,main=plot_name,cex=0.15,cex.main=0.8,cex.lab=0.8,cex.axis=0.8,xlab=expression(italic(x)),ylab=expressio

    lines(xtest,f(xtest),col="black")

    # Spline Regression
    X = designmatrix(x,p,k)
  betas = solve(t(X) %*% X,tol=1e-50) %*% t(X) %*% y
    fhat = X %*% betas
    ytest = 1:(length(xtest))
    for(nn in 1:(length(xtest))){
      ytest[nn] = modelf(xtest[nn],knots,betas)
    }
    lines(xtest,ytest,col="blue")

    # Penalized Spline Regression
    if(o==1){
      # Cross Validation
      minimizer = function(lambda,x,y){
        X = designmatrix(x,p,k)

        D = diag(c(0*(1:(p+1)),0*(1:k)+1))
        hlambda = X %*% (solve((t(X) %*% X)+lambda*D,tol=1e-50) %*% t(X))
        fhatlambda = hlambda %*% y
        hii = diag(hlambda)

        return(sum(((y-fhatlambda)/(1-hii))^2))
```

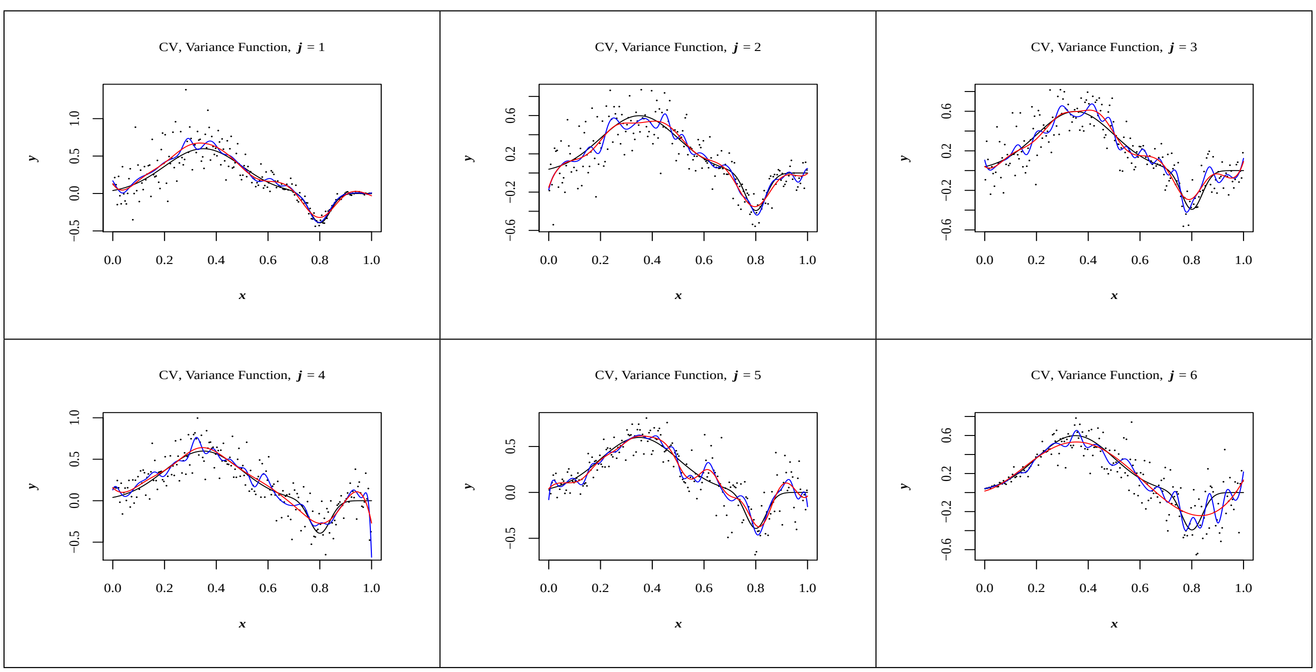
```
      }
    } else if(o == 2){
      # Generalized Cross Validation
      minimizer = function(lambda,x,y){
        X = designmatrix(x,p,k)

        n = length(x)
        D = diag(c(0*(1:(p+1)),0*(1:k)+1))
        hlambda = X %*% (solve(t(X) %*% X+lambda*D,tol=1e-50) %*% t(X))
        fhatlambda = hlambda %*% y
        hii = sum(diag(hlambda))/n

        return(sum(((y-fhatlambda)/(1-hii))^2))
      }
    } else if(o == 3){
      # AICc
      minimizer = function(lambda,x,y){
        X = designmatrix(x,p,k)

        D = diag(c(0*(1:(p+1)),0*(1:k)+1))
        hlambda = X %*% (solve(t(X) %*% X + lambda*D,tol=1e-50) %*% t(X))
        fhatlambda = hlambda %*% y

        n = length(x)
        tr = sum(diag(hlambda))
        norm = sum((y-fhatlambda)^2)
        return(log(norm)+2*(tr+1)/(n-tr-2))
      }
    }else if(o == 4){
      # Risk
      minimizer = function(lambda,x,y){
        X = designmatrix(x,p,k)

        D = diag(c(0*(1:(p+1)),0*(1:k)+1))
        hlambda = X %*% (solve(t(X) %*% X + lambda*D,tol=1e-50) %*% t(X))
        fhatlambda = hlambda %*% y

        n = length(x)
        norm = sum((fhatlambda-(hlambda %*% fhatlambda))^2)
        sigma2 = sum((y-fhatlambda)^2)/(n-1)

        return((norm+sigma2*(sum(diag(hlambda %*% t(hlambda)))-2*sum(diag(hlambda))+n)))
      }
    }

    lambda = 1e-7
    minlambda = nlm(minimizer,c(lambda),x = x, y = y,steptol = 1e-20)
    lambda = minlambda$estimate
    D = diag(c(0*(1:(p+1)),0*(1:k)+1))
    betalambda = (solve(t(X) %*% X + lambda*D,tol=1e-50) %*% t(X)) %*% y
    hlambda = X %*% betalambda
    fhatlambda = hlambda %*% y
    ytest = 1:(length(xtest))
    for(nn in 1:(length(xtest))){
      ytest[nn] = modelf(xtest[nn],knots,betalambda)
    }
    print(paste("j =",m,", lambda =",lambda, ", min =",minlambda$minimum))
    lines(xtest,ytest,col="red")
    if (export) dev.off()
  }
 }
}
```


CV, Noise Level, j = 1
CV, Noise Level, j = 2
CV, Noise Level, j = 3
CV, Noise Level, j = 4
CV, Noise Level, j = 5
CV, Noise Level, j = 6
x
y

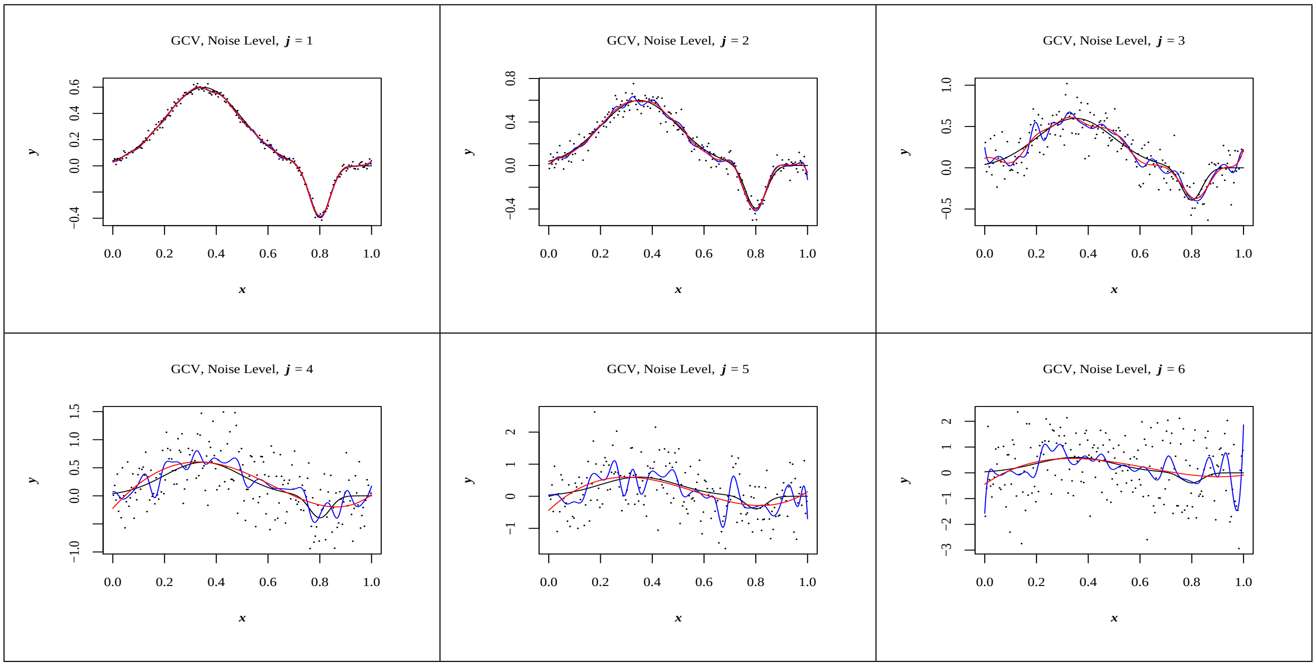

CV, Design Density, j = 1
CV, Design Density, j = 2
CV, Design Density, j = 3
CV, Design Density, j = 4
CV, Design Density, j = 5
CV, Design Density, j = 6
x
y

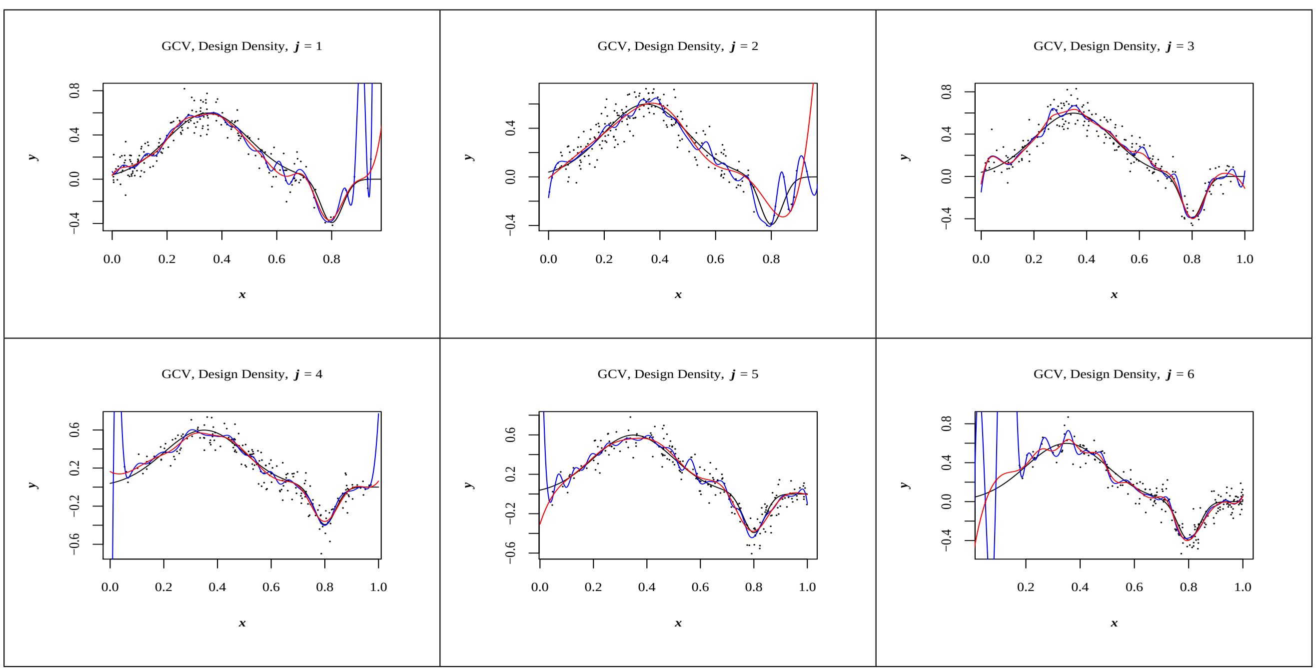


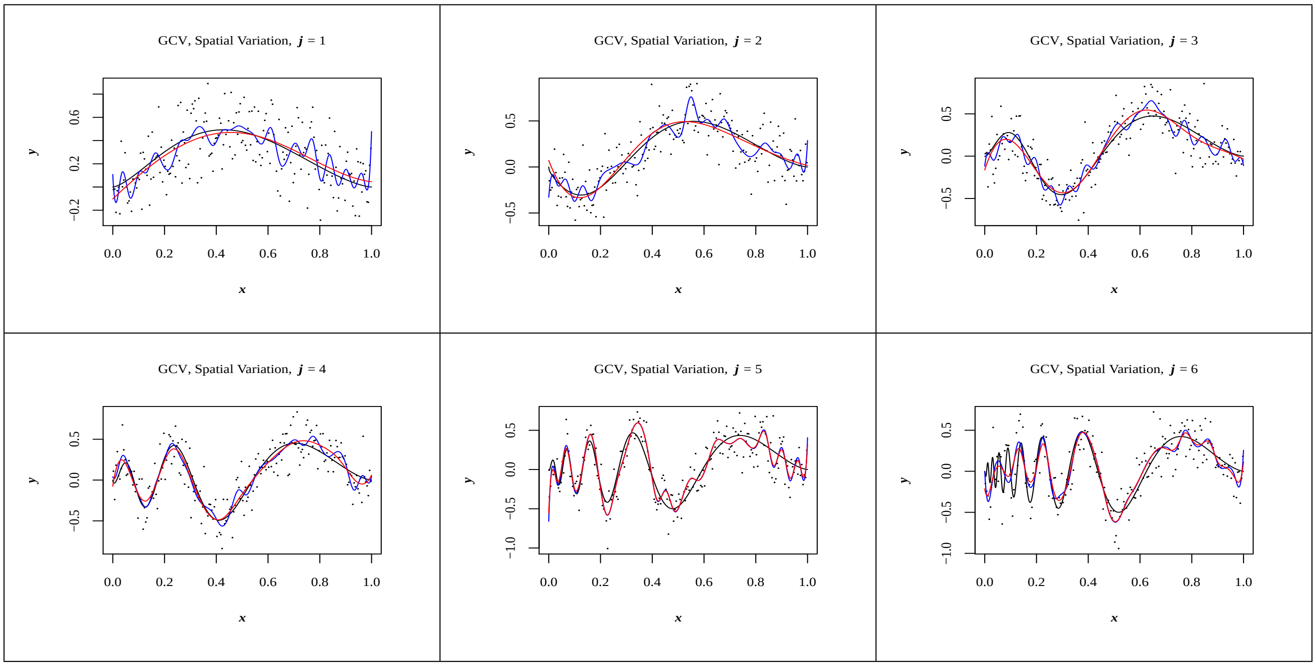
CV, Spatial Variation, j = 1
CV, Spatial Variation, j = 2
CV, Spatial Variation, j = 3
CV, Spatial Variation, j = 4
CV, Spatial Variation, j = 5
CV, Spatial Variation, j = 6
x
y

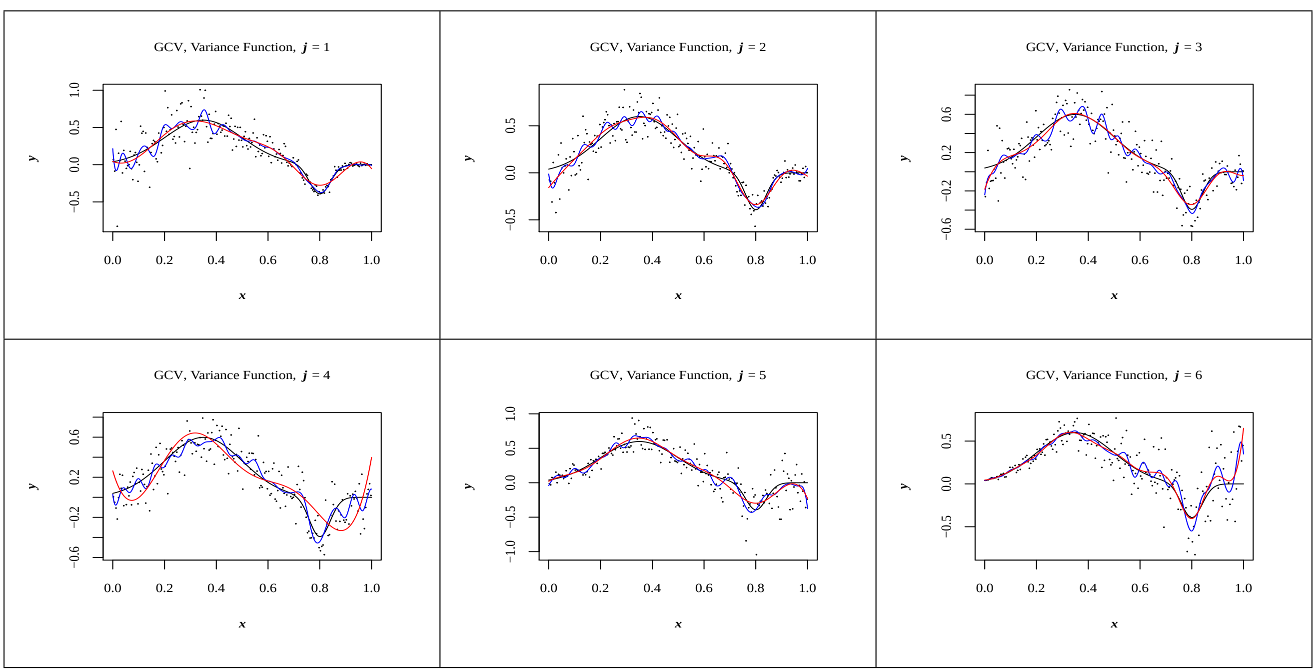


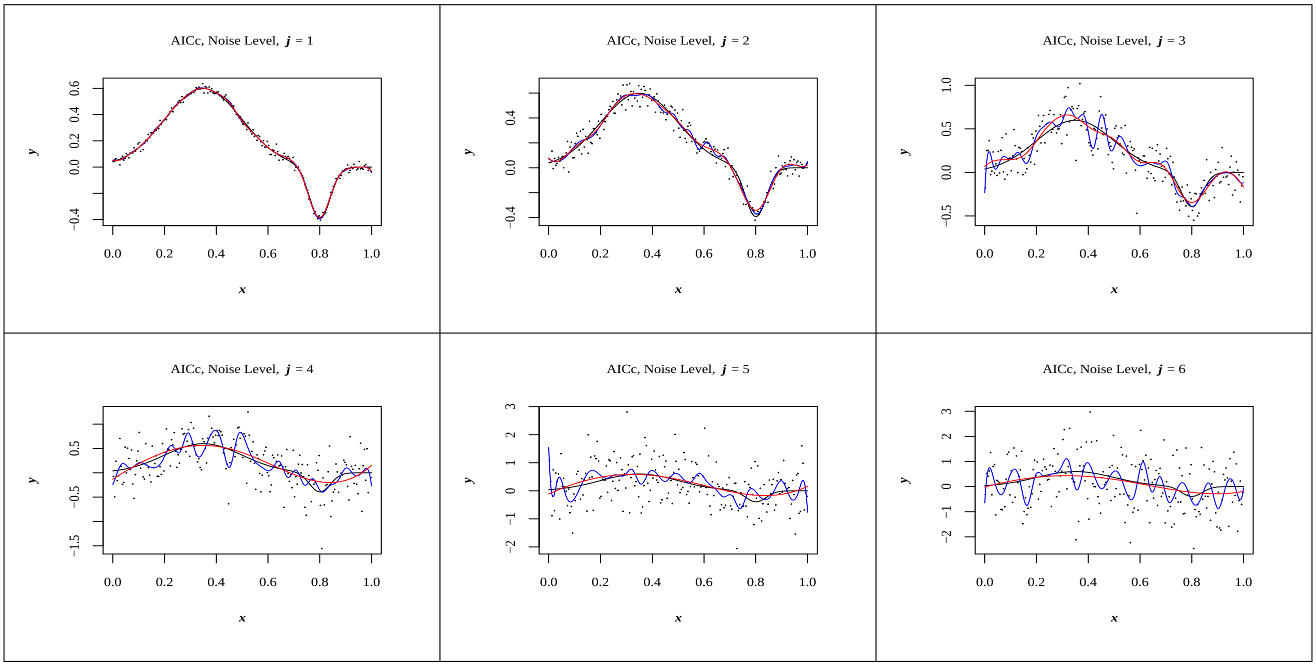
CV, Variance Function, j = 1
CV, Variance Function, j = 2
CV, Variance Function, j = 3
CV, Variance Function, j = 4
CV, Variance Function, j = 5
CV, Variance Function, j = 6
x
y

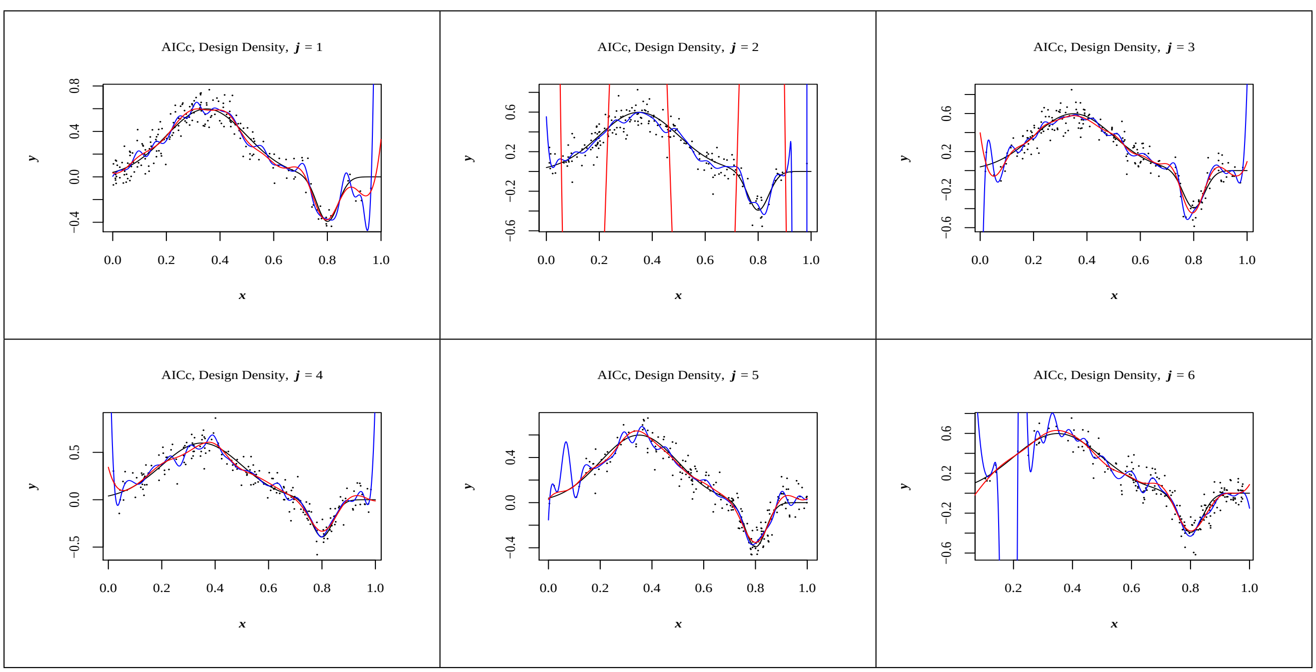

GCV, Noise Level, j = 1
GCV, Noise Level, j = 2
GCV, Noise Level, j = 3
GCV, Noise Level, j = 4
GCV, Noise Level, j = 5
GCV, Noise Level, j = 6
x
y

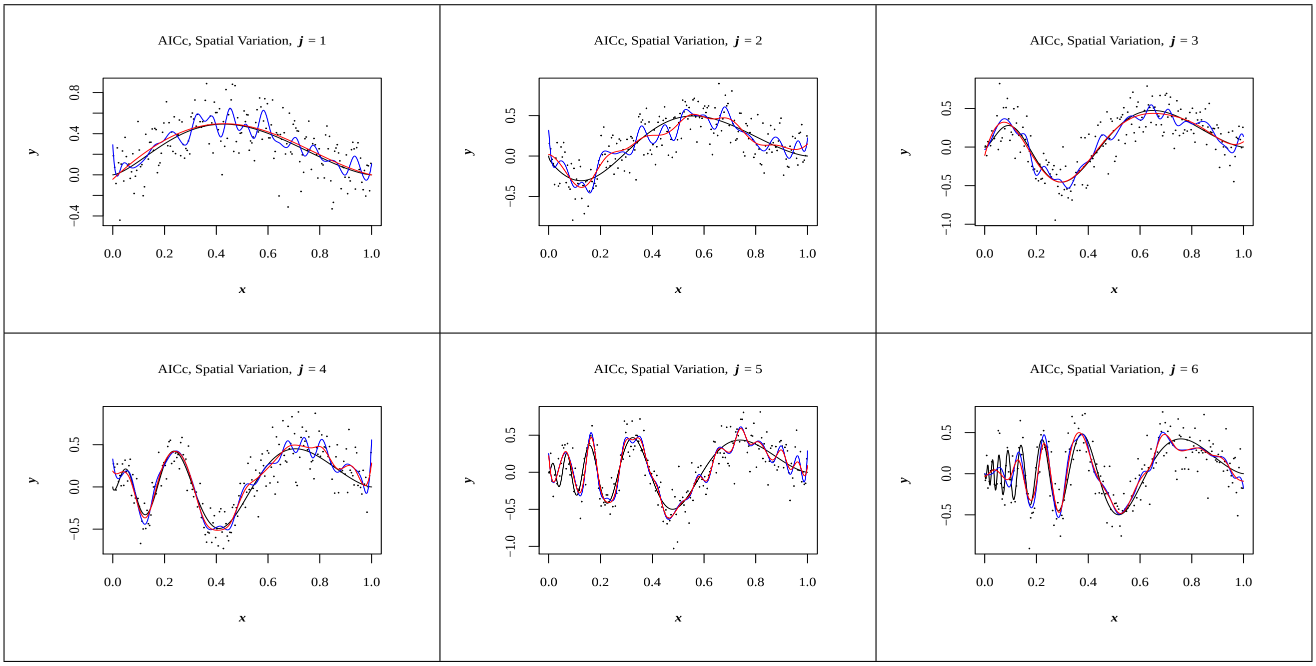

GCV, Design Density, j = 1
GCV, Design Density, j = 2
GCV, Design Density, j = 3
GCV, Design Density, j = 4
GCV, Design Density, j = 5
GCV, Design Density, j = 6
x
y

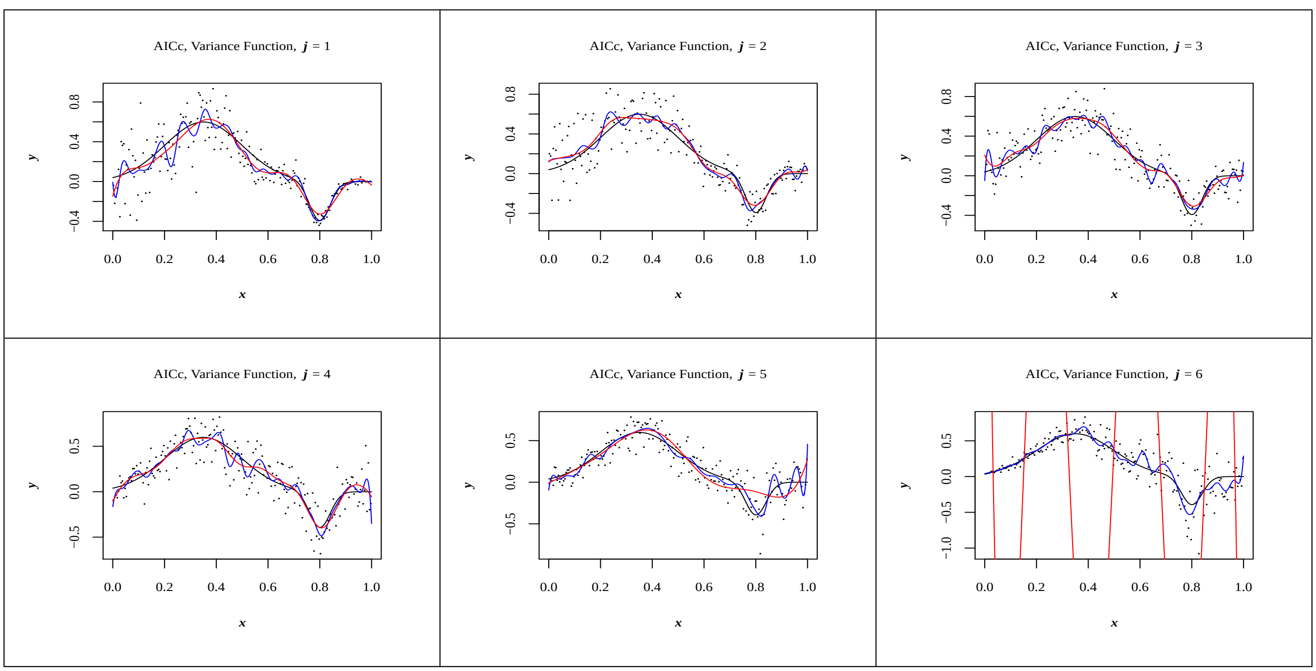

GCV, Spatial Variation, j = 1
GCV, Spatial Variation, j = 2
GCV, Spatial Variation, j = 3
GCV, Spatial Variation, j = 4
GCV, Spatial Variation, j = 5
GCV, Spatial Variation, j = 6

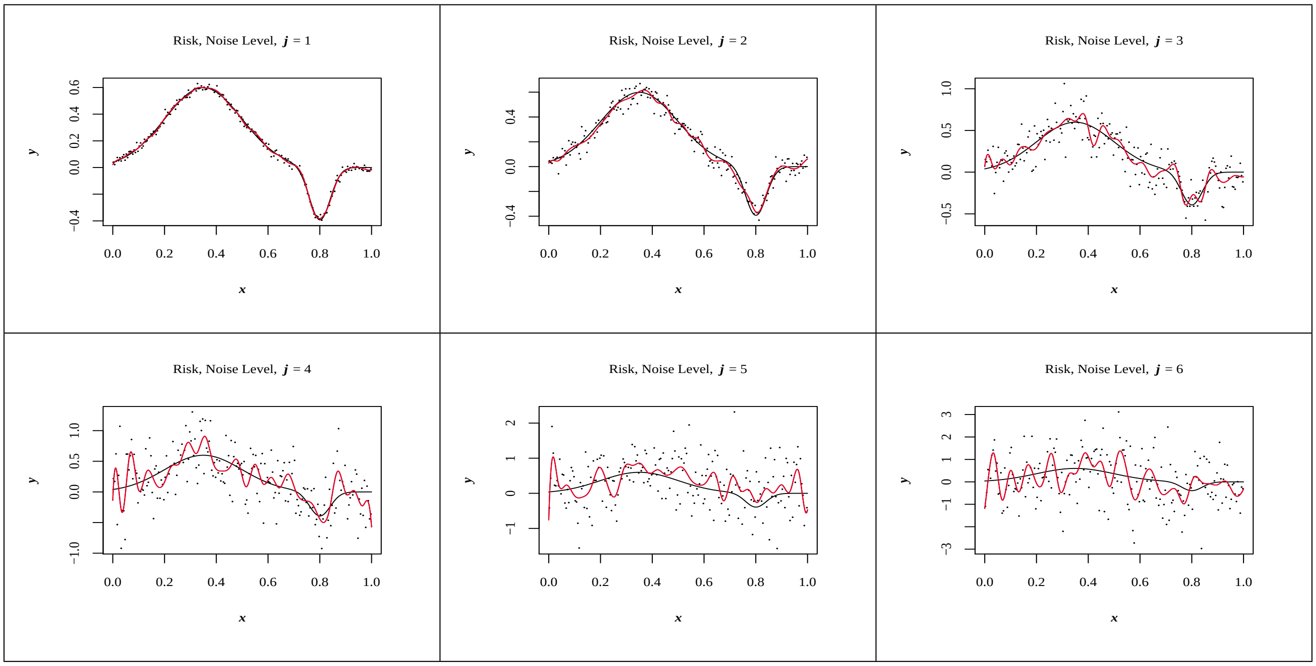

GCV, Variance Function, j = 1
GCV, Variance Function, j = 2
GCV, Variance Function, j = 3
GCV, Variance Function, j = 4
GCV, Variance Function, j = 5
GCV, Variance Function, j = 6

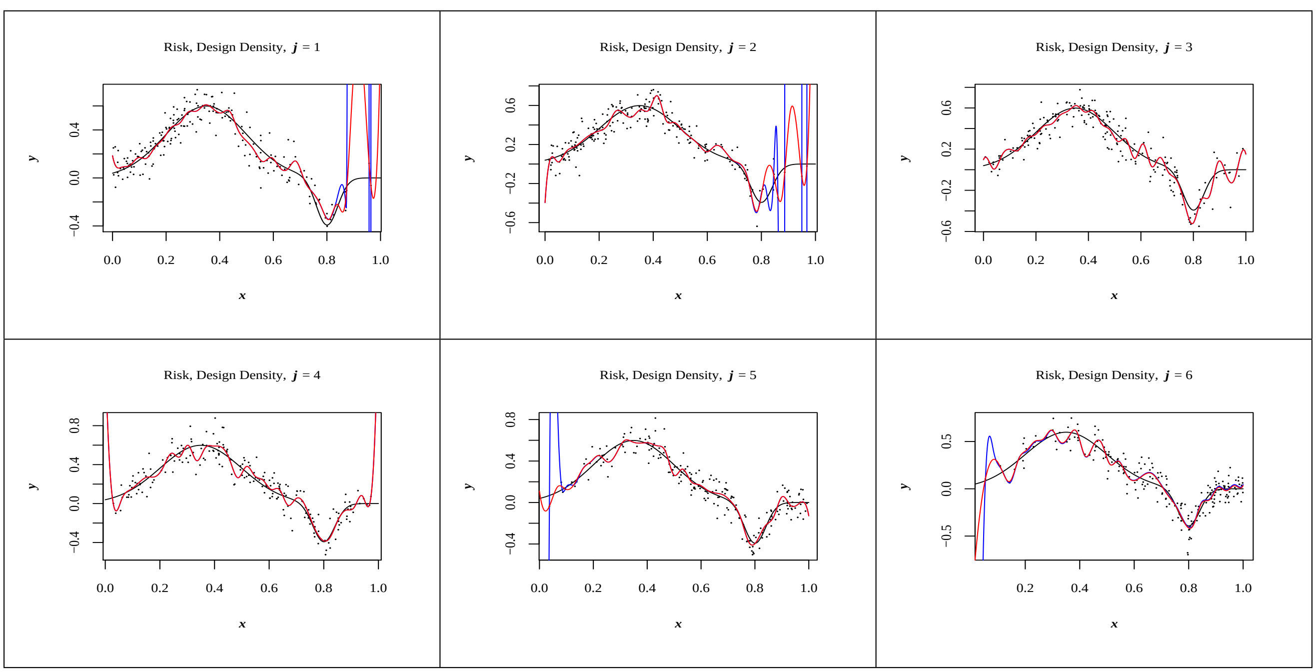

AICc, Noise Level, j = 1
AICc, Noise Level, j = 2
AICc, Noise Level, j = 3
AICc, Noise Level, j = 4
AICc, Noise Level, j = 5
AICc, Noise Level, j = 6
x
y

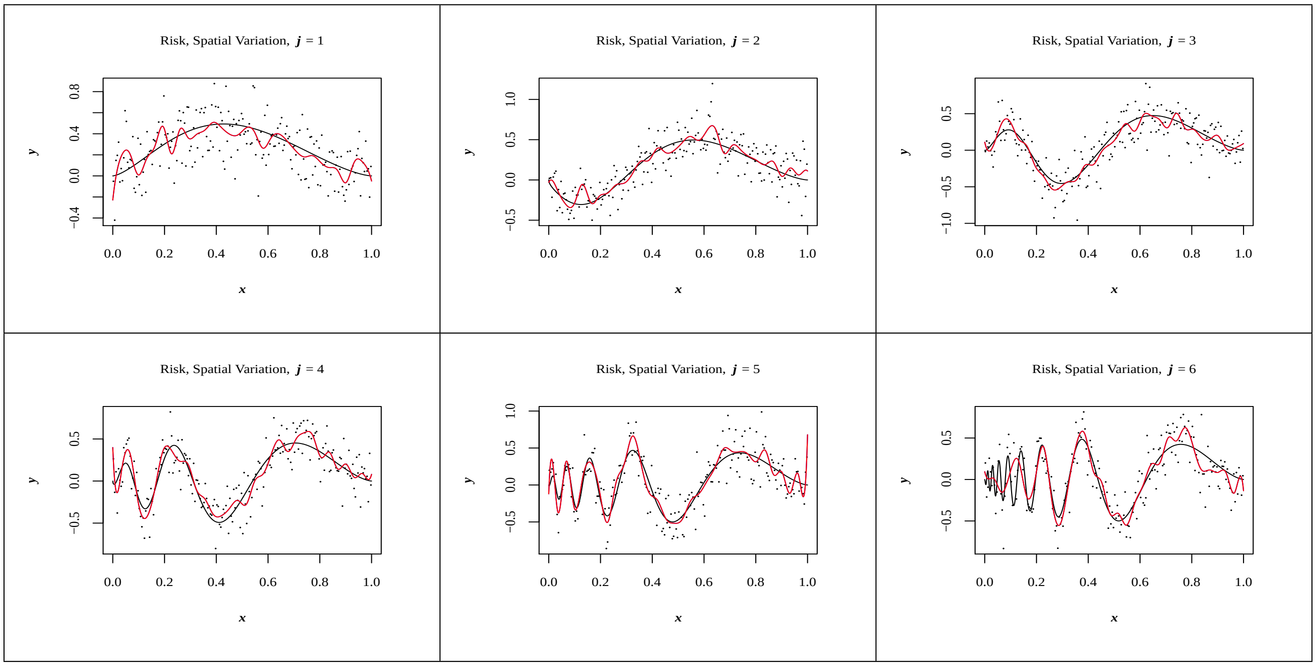

AICc, Design Density, j = 1
AICc, Design Density, j = 2
AICc, Design Density, j = 3
AICc, Design Density, j = 4
AICc, Design Density, j = 5
AICc, Design Density, j = 6
x
y

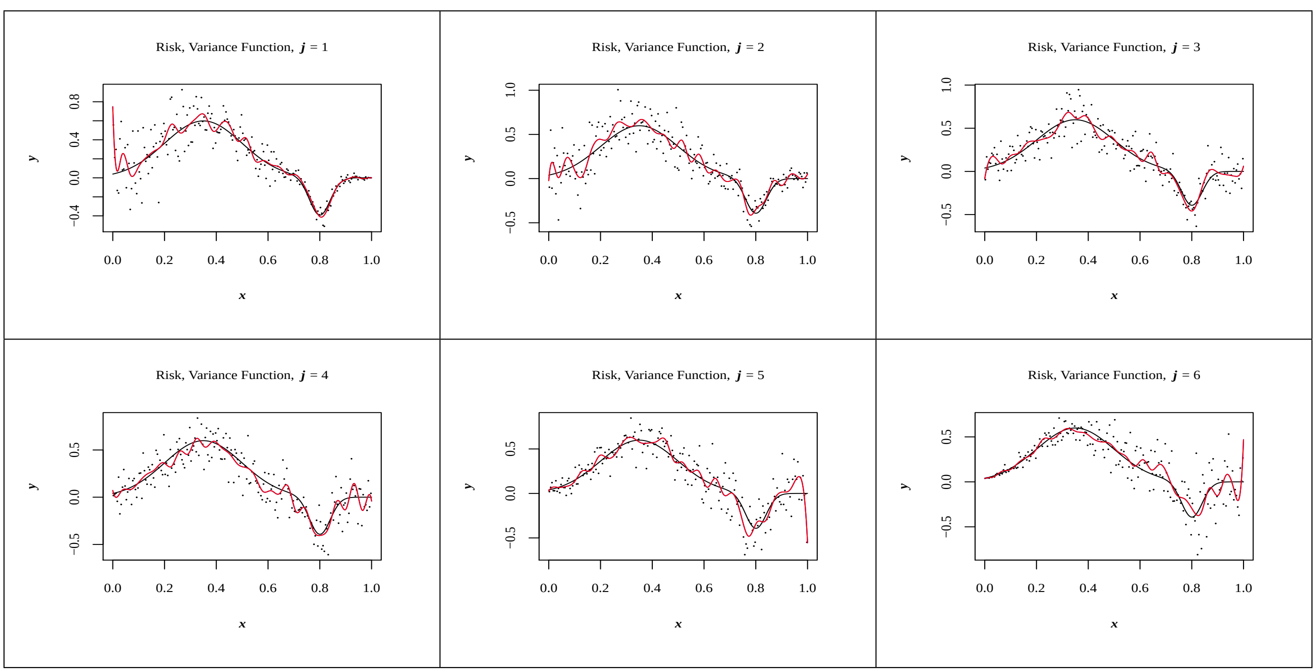

AICc, Spatial Variation, j = 1
AICc, Spatial Variation, j = 2
AICc, Spatial Variation, j = 3
AICc, Spatial Variation, j = 4
AICc, Spatial Variation, j = 5
AICc, Spatial Variation, j = 6
x
y


AICc, Variance Function, j = 1
AICc, Variance Function, j = 2
AICc, Variance Function, j = 3
AICc, Variance Function, j = 4
AICc, Variance Function, j = 5
AICc, Variance Function, j = 6
x
y


Risk, Noise Level, j = 1
Risk, Noise Level, j = 2
Risk, Noise Level, j = 3
Risk, Noise Level, j = 4
Risk, Noise Level, j = 5
Risk, Noise Level, j = 6
x
y


Risk, Design Density, j = 1
Risk, Design Density, j = 2
Risk, Design Density, j = 3
Risk, Design Density, j = 4
Risk, Design Density, j = 5
Risk, Design Density, j = 6
x
y


Risk, Spatial Variation, j = 1
Risk, Spatial Variation, j = 2
Risk, Spatial Variation, j = 3
Risk, Spatial Variation, j = 4
Risk, Spatial Variation, j = 5
Risk, Spatial Variation, j = 6
x
y


Risk, Variance Function, j = 1
Risk, Variance Function, j = 2
Risk, Variance Function, j = 3
Risk, Variance Function, j = 4
Risk, Variance Function, j = 5
Risk, Variance Function, j = 6
x
y