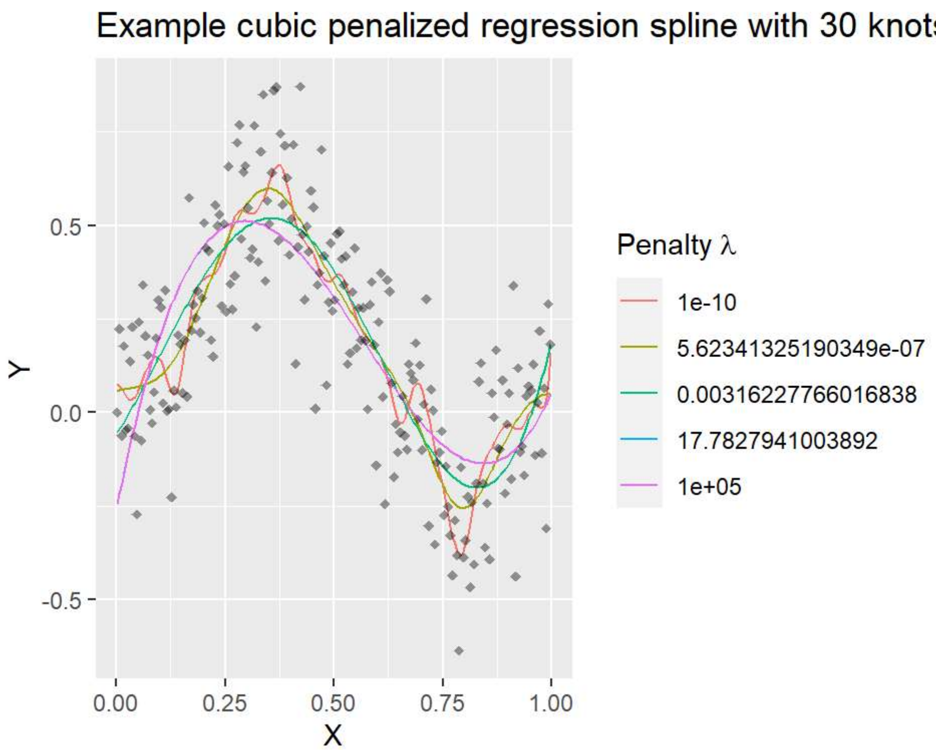
HW5

Sili Fan and Kwan Ho Lee

6/1/2021

Problem 1

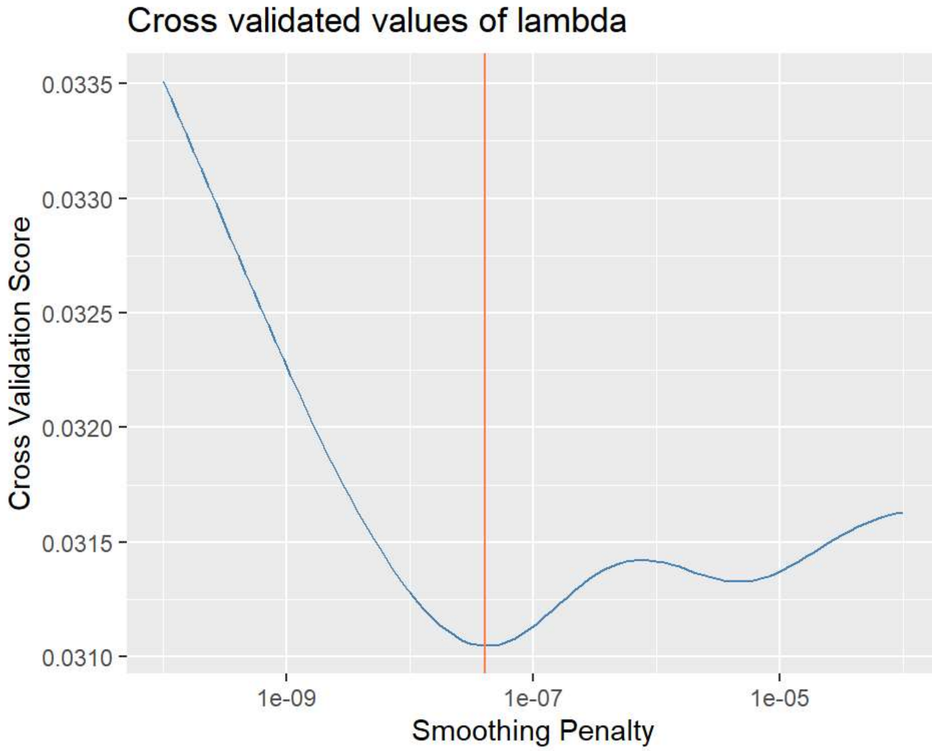


(a) Implement the cross-validation and generalized cross-validation methods for choosing the smoothing parameter λ

Cross validation (CV)

We interested in finding the value of λ that minimizes the following function for a particular sample $\{x_i, y_i\}$. We accomplish this with a simple grid search over an appropriate range of $\lambda > 0$.

$$CV(\lambda) = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{f}_\lambda(x_i)}{1-(S_\lambda)_{i,i}})^2$$

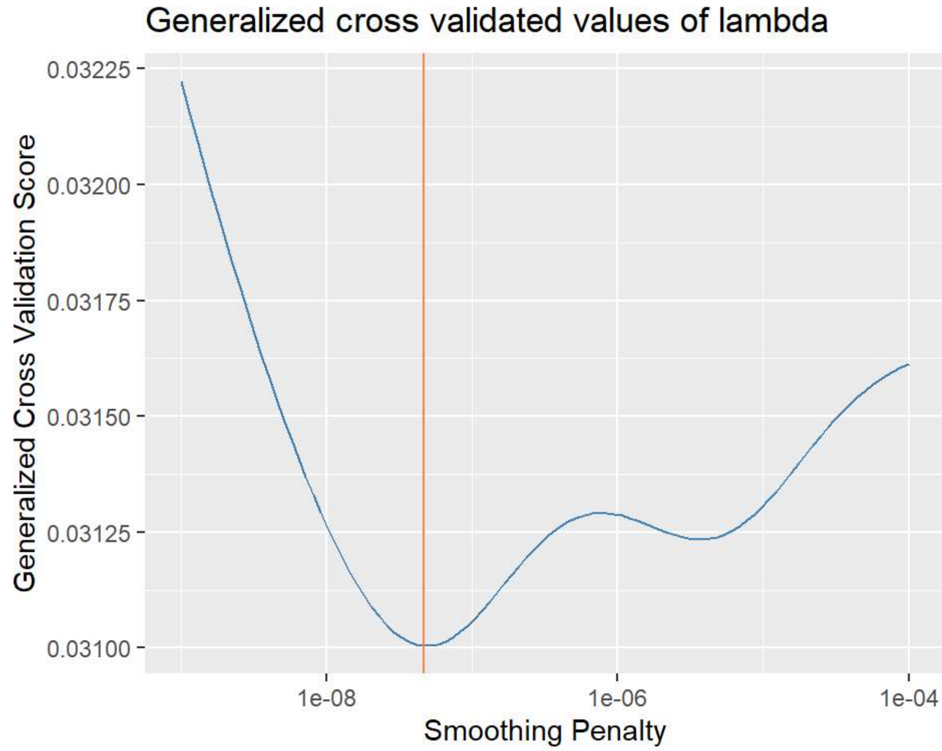


Note that the minimizing smoothing penalty is $\lambda_{CV} = 4.0370173 \times 10^{-8}$ with a cross validation score of $CV(\lambda_{CV}) = 0.0310471$.

Generalized cross validation (GCV)

We interested in finding the value of λ that minimizes the following function for a particular sample $\{x_i, y_i\}$. We accomplish this with a simple grid search over an appropriate range of $\lambda > 0$.

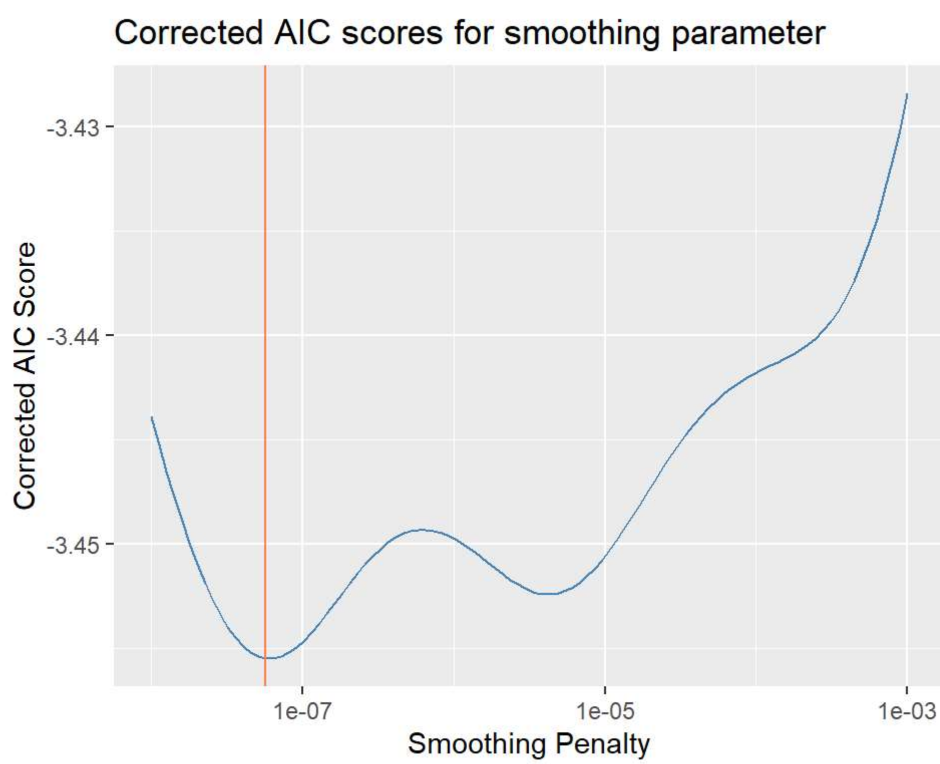
$$GCV(\lambda) = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - \frac{1}{N}\mathrm{tr}(S_\lambda))^2}$$



Note that the minimizing smoothing penalty is $\lambda_{GCV} = 4.6415888 \times 10^{-8}$ with a generalized cross validation score of $GCV(\lambda_{GCV}) = 0.0310048$.

(b)

$$AIC_C(\lambda) = \log(||y - \hat{f}_{\lambda}||_2^2) + \frac{2(\mathrm{tr}(H_\lambda) + 1)}{n - \mathrm{tr}(H_\lambda) - 2}$$



Note that the minimizing smoothing penalty is $\lambda_{AIC_C} = 5.7223677 \times 10^{-8}$ with a score of $AIC_C(\lambda_{AIC_C}) = -3.4554865$

(c)

Proof:

$E(||y - \hat{f}_\lambda||^2) = ||(I - H_\lambda)f||^2 + \sigma^2(tr(H_\lambda H_\lambda^\top) - 2tr(H_\lambda) + n)$ to get a sense for what this estimate will look like.

1. $E(||y - \hat{f}_\lambda||^2) = E(||(I - H_\lambda)y||^2)$ because $\hat{f}_\lambda = H_\lambda y$ and real matricies have both left and right distributivity.
2. $E(||(I - H_\lambda)y||^2) = E(||(I - H_\lambda)(f + e)||^2)$ because $y = f + e$.
3. $E(||(I - H_\lambda)(f + e)||^2) = E(||(I - H_\lambda)f||^2 + ||(I - H_\lambda)e)||^2)$ because $E(e) = 0$ by assumpion and the distributivity property noted in (1).
4. $E(||(I - H_\lambda)f||^2 + ||(I - H_\lambda)e||^2) = ||(I - H_\lambda)f||^2 + E(||(I - H_\lambda)e)||^2$ because the lefthand quantity is a constant. It remains to show that $E(||(I - H_\lambda)e||^2) = \sigma^2(tr(H_\lambda H_\lambda^\top) - 2tr(H_\lambda) + n)$
5. $E(||(I - H_\lambda)e||^2) = \sigma^2 tr(I(H_\lambda - I)(H_\lambda - I)^\top I^\top)$ because $Cov(e) = I\sigma^2$.
6. $tr(I(H_\lambda - I)(H_\lambda - I)^\top I^\top) = \sum_{i=1}^n (h_{ii} - 1)^2$.
7. $\sum_{i=1}^n (h_{\lambda,ii} - 1)^2 = tr(H_\lambda H_\lambda^\top) - 2tr(H_\lambda) + n$

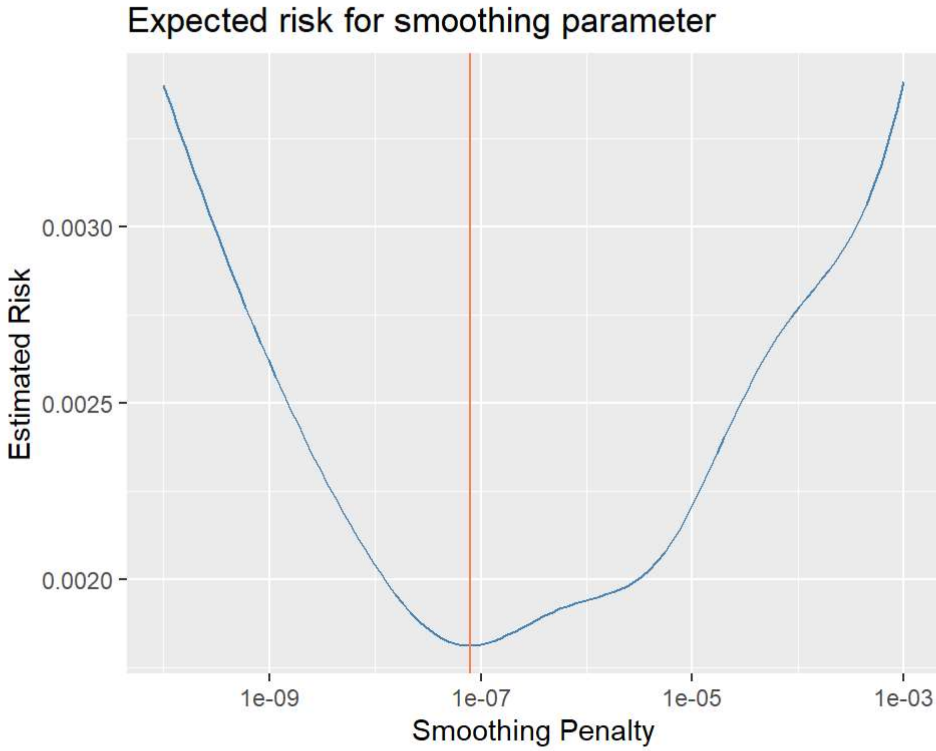
To complete the proof, we get the following steps, we develop an estimator for $risk(\lambda) = E(||f - \hat{f}_\lambda||^2)$:

1. $risk(\lambda) = E(||y - e - \hat{f}_\lambda||^2)$ by the model assumption $y = f + e$.
2. Expanding the L_2 norm in the expression above, using the linearity of expectation, and noting the fact that $E(e) = 0$; $E(||y - e - \hat{f}_\lambda|||^2) = E(y^\top y - y^\top \hat{f}_\lambda + e^\top e - \hat{f}_\lambda^\top y + \hat{f}_\lambda^\top \hat{f}_\lambda)$
3. Consolidating terms in (2), $risk(\lambda) = n\sigma^2 + E(||y - \hat{f}_\lambda||^2)$ and we use the method of moments to create an unbiased estimator when σ^2 is known.

y are the noisy observations, $\hat{f}_\lambda = H_\lambda y$ where H_λ depends only on X, and we can create an estimator for σ^2: $\hat{\sigma^2} \approx RSS(\hat{f}_\lambda) = \frac{1}{n}||y - \hat{f}_\lambda||^2$. Using this estimator for σ^2, the estimator for $E(||f - \hat{f}_\lambda||^2)$ is proportional to $||y - \hat{f}_\lambda||^2$.

An alternative estimator for the risk, using classical pilots, is given in section 2.2.1 of Lee (2003): use $\hat{f}_{\lambda_p} \approx f$ where λ_p is the CV parameter. This is the method that we employ in our algorithm.

The following algorithm exhibits this risk estimator minimization criterion for calculating the penalty parameter λ for a single experimental simulation parameterization.



Note that the minimizing smoothing penalty is $\lambda_{ER} = 7.924829 \times 10^{-8}$ with a score of $ER(\lambda_{ER}) = 0.0018119.$

(d) Conduct a simulation study to compare the above four methods for choosing λ. Use the experimental setting from the paper downloadable from Canvas.

In this homework, we simulate six different methods for selecting a penalty parameter λ under six different noise, design density, spatial variation, and variance heterogeneity states. The four methods for fitting the penalty parameter λ for a penalized cubic regression spline are minimization of the following scores: cross validation, generalized cross validation, corrected Akaike information criterion, and expected risk.

In this part, we illustrate the concepts required for the simulation in the context of a simple example: $y = f(x) + \epsilon.$

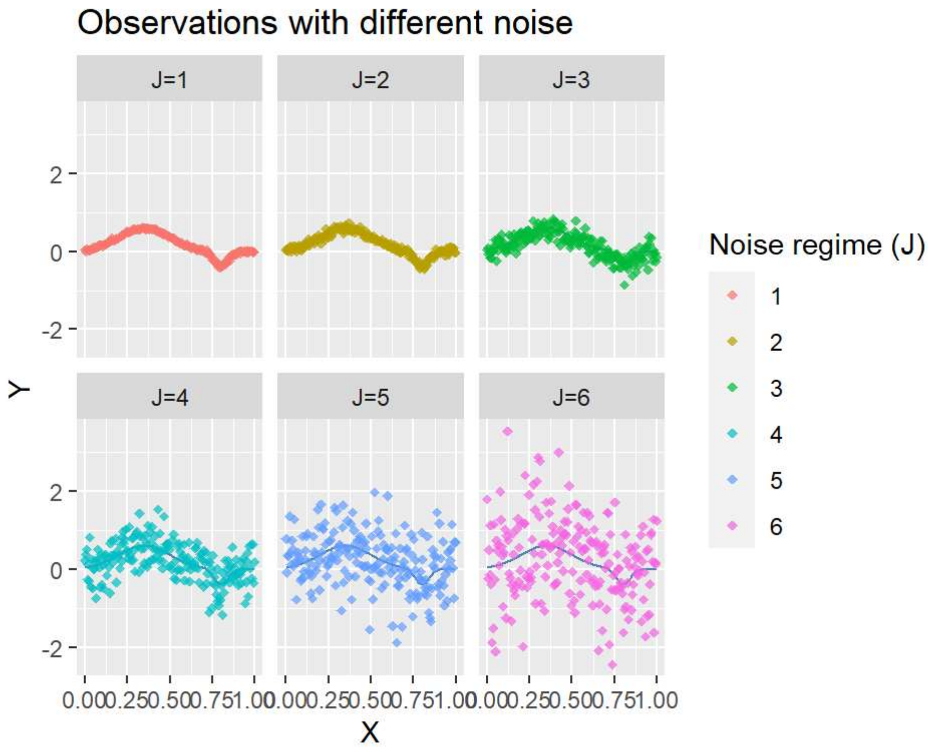
To fit a cubic penalized regression spline $\hat{f}_{\lambda}(x)$ which approximates $f(x)$, we minimize the following functional within the class $\mathbf{F}$ of cubic regression splines:

$$\hat{f}_{\lambda}(x) = \text{argmin}_{f \in \mathbf{F}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \sum_{j=1}^{k} \beta_{p_j}^2 = ||y - f(x)||_2^2 + \lambda \beta^\top D \beta$$

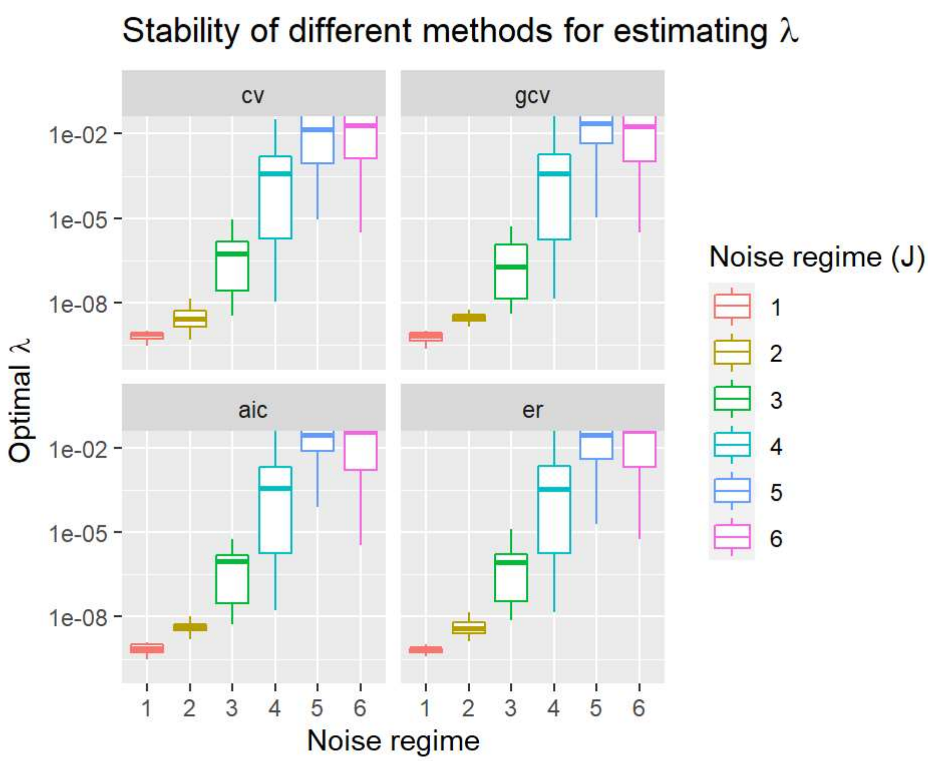
where D is the diagonal matrix with $p + 1$ 0s followed by K 1s.

Performance under different levels of noise

We evaluate the performance of λ estimators under the model $y_{ij} = f(x_i) + \sigma_j \epsilon_i$ where $\sigma_j = 0.02 + 0.04(j-1)^2$ and $\epsilon_i \sim \text{iid} N(0,1)$.



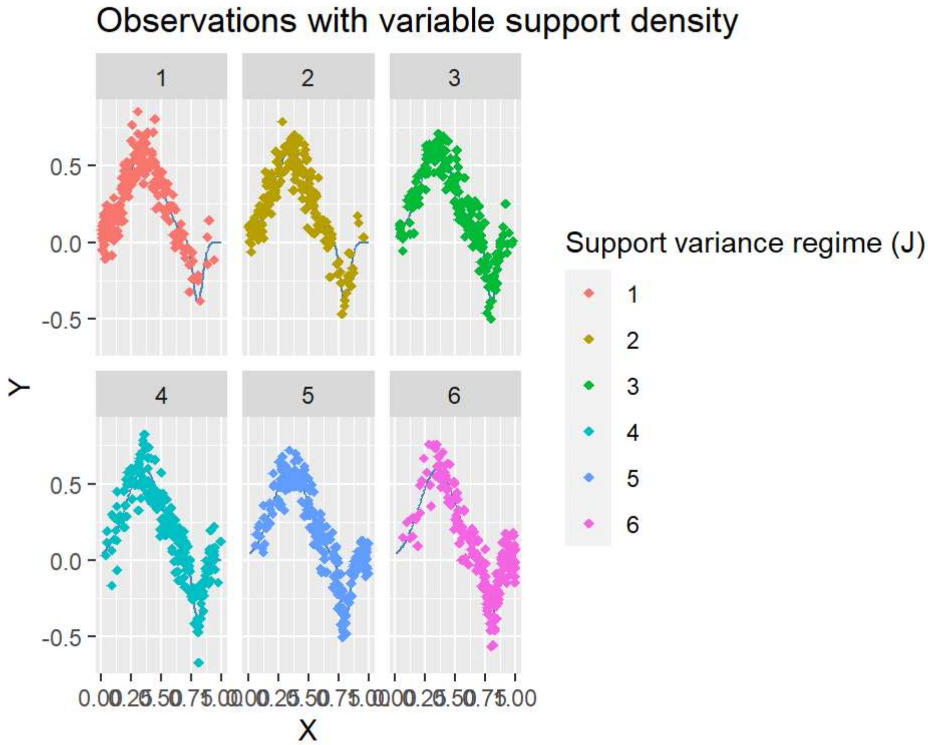
The above plot illustrates the effects of noise regimes $j \in \{1, \ldots, J = 6\}$.



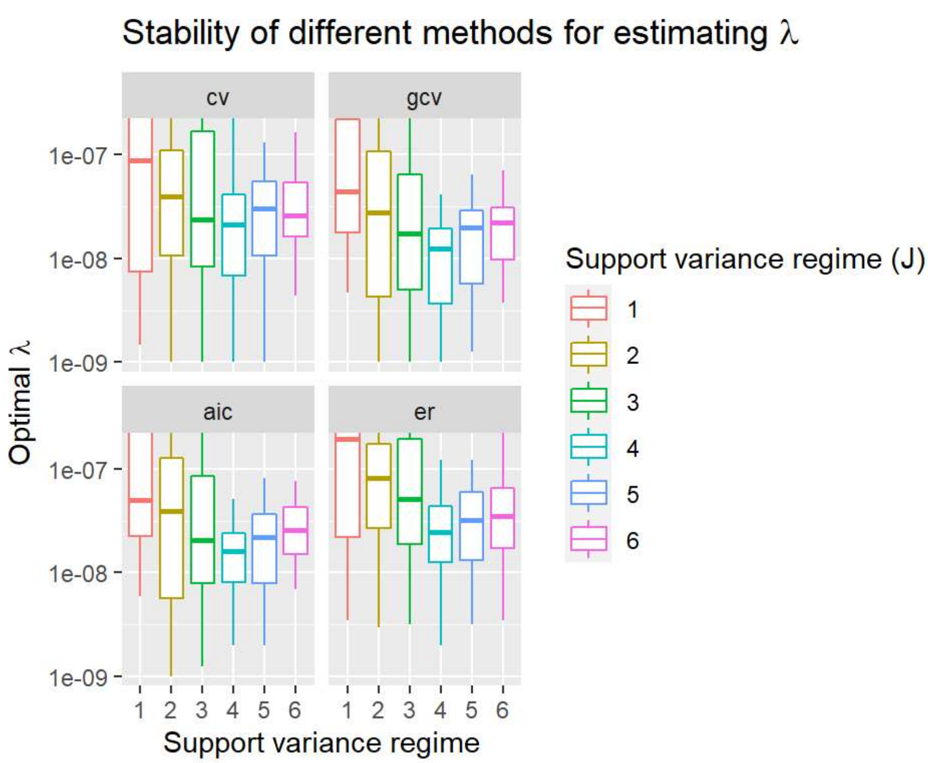
As the level of noise increases, the optimal penalty parameter increases and its stability decreases. The increase in λ means that we encourage the spline to be smoother - to fit less tightly to the noisy data - as the level of noise increases.

Performance under different design density

$$y_{ij} = f(X_{ij}) + \sigma\epsilon_i \quad \text{where} \quad \sigma = 0.1, X_{ij} = F^{-1}(X_i), F(X) = \text{Beta}(\frac{j+4}{5}, \frac{11-j}{5}), X_i \sim \text{Unif}(0,1)$$



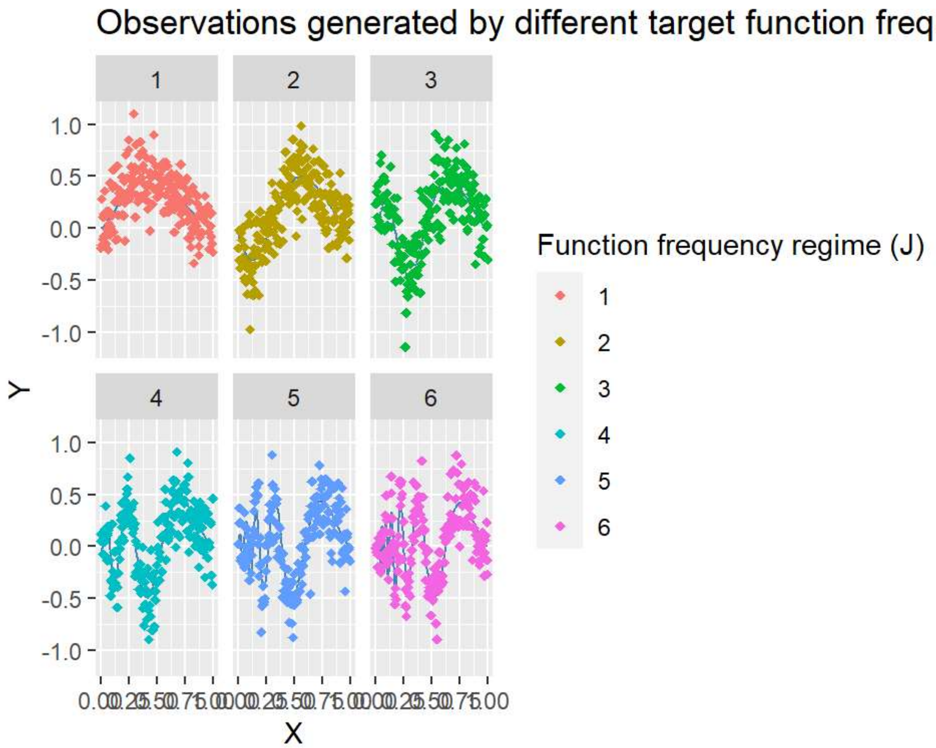
As illustrated above, the small values of j group points to the left-hand side of the support, while the large values of j group points to the right-hand side.



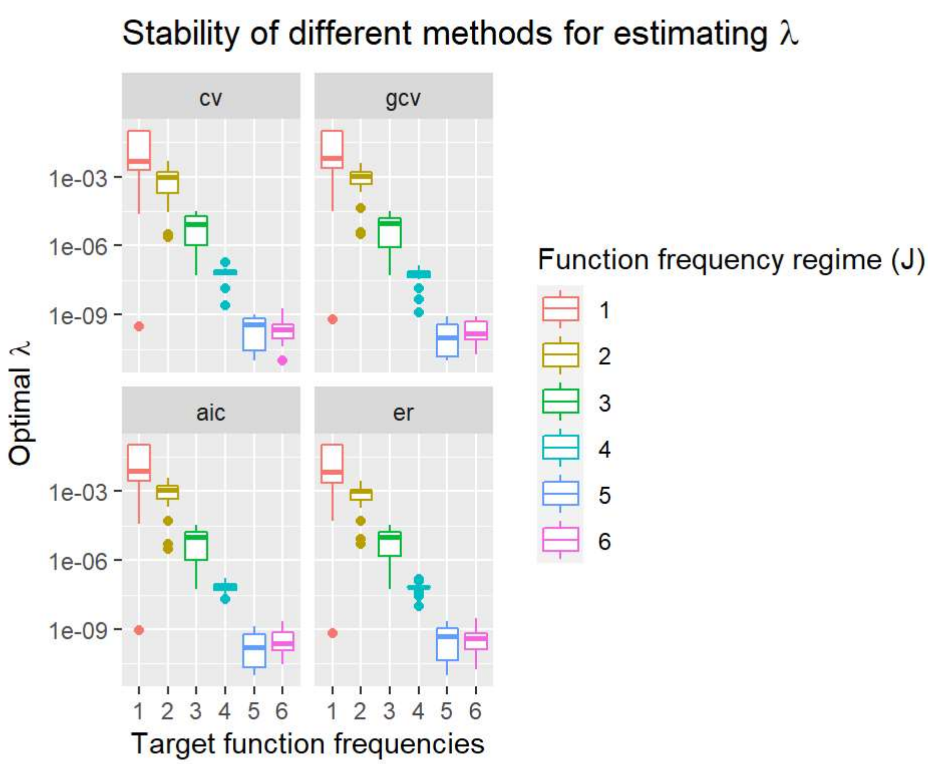
The stability of the penalty parameter estimations is much lower when the points are not homogeneously dense throughout the support. This is especially true considering that we fitting a spline with 30 equally spaced knots. A variable knot selection protocol might mitigate some of the instability of the penalty parameter estimates observed for $j \in \{1, 2, 5, 6\}$.

Performance under different frequency of target function

$$y_{ij} = f_j(x_i) + \sigma\epsilon_i \quad \text{where} \quad \sigma = 0.2, f_j(x) = \sqrt{x*(1-x)}\sin(\frac{2\pi(1+2^{(9-4j)/5})}{x+2^{(9-4j)/5}}), \epsilon_i \sim \text{iid}N(0,1)$$



As illustrated above, increasing j increases the frequency of the spatial distribution function $f(x)$.

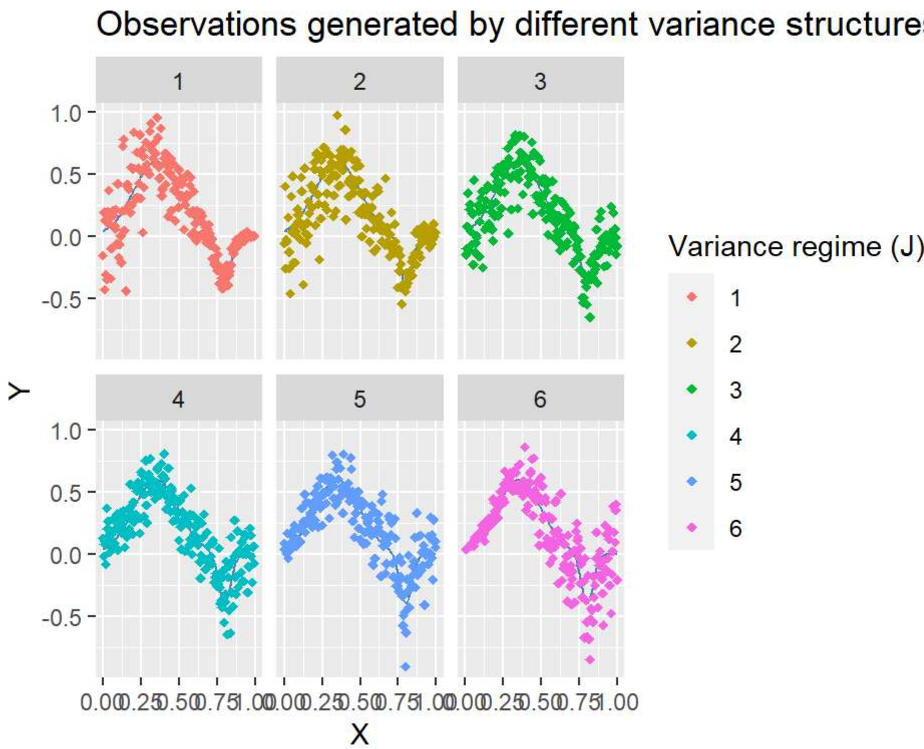


Increased spatial frequency stabilizes the estimates of the penalty parameter to a point. Furthermore, as the function becomes more erratic, the value of the penalty parameter λ decreases, forcing the spline to hug the data more closely.

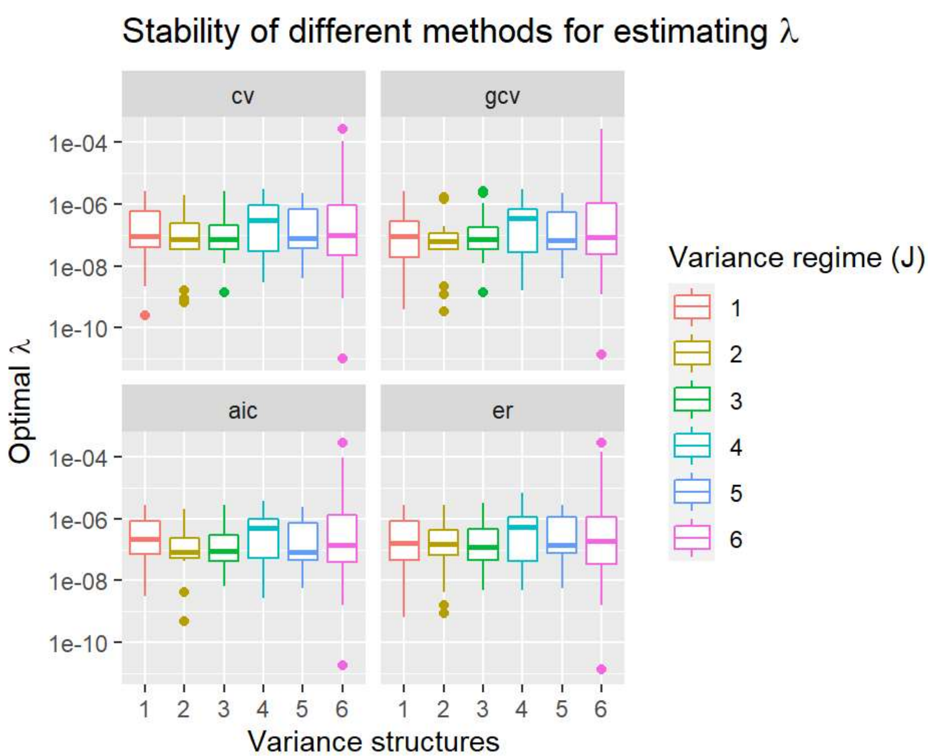
Performance under heterogeneous variances

$$y_{ij} = f(x_i) + \sqrt{v_j(x_i)}\epsilon_i \quad \text{where} \quad v_j(x) = (0.15(1 + 0.4(2j-7)(x-0.5)))^2$$

and $f(x)$ and ϵ_i are defined as in the above.



As illustrated above, this method for varying the variance spatially incurs a high variance for the left-hand side when j is small and a high variance for the right-hand side when j is large.



Spatial variance modulation destabilizes smoothing parameter estimates. Having 30 equi-spaced knots is a serious issue here and a more dynamic collocation procedure would ameliorate some of this instability in the λ estimates.

Appendix

The following code is used in Problem 1 part (a)

```
cv = function(l, k=k, X=X, Y=Y){


    n = dim(X)[1]
    D = diag(c(rep(0, dim(X)[2]-k), rep(1, k)))
    H = X %*% solve(t(X) %*% X + l*D) %*% t(X)
    fhys = H %*% Y
    hii = diag(H)
    r = mean(((Y-fhys)/(1-hii))^2)
    return(r)
}

cvv = Vectorize(cv, vectorize.args = c("l"))
lambdas = 10^(seq(-10, -4, length.out=100))
cvs = cvv(X=X, Y=Y, l=lambdas, k=k)

cv_df = data.frame(x=lambdas, y=cvs)
plt_cv = ggplot(data=cv_df) +
    geom_line(aes(x=x, y=y), color="steelblue") +
    labs(x="Smoothing Penalty", y="Cross Validation Score",
         title="Cross validated values of lambda") +
    scale_x_log10() +
    # geom_hline(yintercept = min(cvs), color="coral") +
    geom_vline(xintercept = lambdas[which(cvs == min(cvs))], color="coral")
```

```
gcv = function(X=X, Y=Y, k=k, l){


    n = dim(X)[1]
    D = diag(c(rep(0, dim(X)[2]-k), rep(1, k)))
    H = X %*% solve(t(X) %*% X + l*D) %*% t(X)

    fhys = H %*% Y
    tr = diag(H)

    numer = mean((Y-fhys)^2)
    denom = (1-mean(tr))^2
    r = numer/denom
    return(r)
}

gcvv = Vectorize(gcv, vectorize.args = c("l"))
lambdas = 10^(seq(-9, -4, length.out=100))
gcvs = gcvv(X=X, Y=Y, l=lambdas, k=k)

gcv_df = data.frame(x=lambdas, y=gcvs)
plt_gcv = ggplot(data=gcv_df) +
    geom_line(aes(x=x, y=y), color="steelblue") +
    labs(x="Smoothing Penalty", y="Generalized Cross Validation Score",
         title="Generalized cross validated values of lambda") +
    scale_x_log10() +
    geom_vline(xintercept = lambdas[which(gcvs == min(gcvs))], color="coral")
```

The following code is used in Problem 1 part (b)

```
aicc = function(X=X, Y=Y, k=k, l){


    n = dim(X)[1]
    D = diag(c(rep(0, dim(X)[2]-k), rep(1, k)))
    H = X %*% solve(t(X) %*% X + l*D) %*% t(X)

    fhys = H %*% Y
    tr = sum(diag(H))

    r1 = log(mean((Y-fhys)^2))
    r2n = 2*(tr+1)
    r2d = n - tr - 2
    r = r1 + r2n/r2d
    return(r)
}

aiccv = Vectorize(aicc, vectorize.args = c("l"))
lambdas = 10^(seq(-8, -3, length.out=100))
aiccs = aiccv(X=X, Y=Y, l=lambdas, k=k)

aicc_df = data.frame(x=lambdas, y=aiccs)
plt_aicc = ggplot(data=aicc_df) +
    geom_line(aes(x=x, y=y), color="steelblue") +
    labs(x="Smoothing Penalty", y="Corrected AIC Score",
         title="Corrected AIC scores for smoothing parameter") +
    scale_x_log10() +
    geom_vline(xintercept = lambdas[which(aiccs == min(aiccs))], color="coral")
```

The following code is used in Problem 1 part (c)

```
sighat = function(Y=Y, X=X, k=k, l){
    D = diag(c(rep(0, dim(X)[2]-k), rep(1, k)))
    H = X %*% solve(t(X) %*% X + l*D) %*% t(X)
    fhys = H %*% Y
    r = mean((Y-fhys)^2)
    return(r)
}

er = function(X=X, Y=Y, k=k, l, lp){


    n = dim(X)[1]
    D = diag(c(rep(0, dim(X)[2]-k), rep(1, k)))
    H = X %*% solve(t(X) %*% X + l*D) %*% t(X)
    Hcv = X %*% solve(t(X) %*% X + lp*D) %*% t(X)

    fcvhys = Hcv %*% Y
    fhys = H %*% Y
    sig = sighat(X=X, Y=Y, k=k, l=lp)

    I = diag(dim(H)[1])
    r = (1/n)*(sum(((H-I)%*%fcvhys)^2) + sig*sum(diag(H%*%t(H))))
    return(r)
}

erv = Vectorize(er, vectorize.args = c("l"))
lambdas = 10^(seq(-10, -3, length.out=100))
ers = erv(X=X, Y=Y, l=lambdas, k=k, lp=lcv)

er_df = data.frame(x=lambdas, y=ers)
plt_er = ggplot(data=er_df) +
    geom_line(aes(x=x, y=y), color="steelblue") +
    labs(x="Smoothing Penalty", y="Estimated Risk",
         title="Expected risk for smoothing parameter") +
    scale_x_log10() +
    geom_vline(xintercept = lambdas[which(ers == min(ers))], color="coral")
```

The following code is used in Problem 1 part (d)

```
## Parameterize
J = 6 # number of simulation parameterizations
M = 20 # number of simulations
L = 150 # number of lambdas to grid search
p = 3 # dimension of spline
k = 30 # number of knots
n = 200 # number of observations
a = 0
b = 1
knots = seq(a, b, length.out=k)
xs = ((1:n)-0.5)/n
fxs = f(xs)
lambdas = 10^seq(-10,0, length.out=L)


# Data storage objects

noise_raw_df = data.frame(x=xs, yt=fxs) # observations for plotting
noise_lam_df = data.frame( # best lambda for each simulation 1:M
  cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
  J=rep(NA, J*M))
noise_sco_df = data.frame( # score of the corresponding best lambda
  cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
  J=rep(NA, J*M))

D = diag(c(rep(0, p+1), rep(1, k)))
X1 = outer(xs, 0:p, "^")
X2 = outer(xs, knots, ">")*outer(xs, knots, "-")^p
X = cbind(X1, X2)

for(j in 1:J){

  for(i in 1:M){

    Y = fxs + rnorm(n, mean=0, sd=noise_sd(j))

    # calculate lambda using each method
    cvs = cvv(k=k, l=lambdas, X=X, Y=Y)
    gcvs = gcvv(k=k, l=lambdas, X=X, Y=Y)
    aiccs = aiccv(k=k, l=lambdas, X=X, Y=Y)

    bcv = min(cvs)
    cv_lam = lambdas[which(cvs == bcv)]

    ers = erv(k=k, l=lambdas, X=X, Y=Y, lp=cv_lam)

    bgcv = min(gcvs)
    gcv_lam = lambdas[which(gcvs == bgcv)]
    baicc = min(aiccs)
    aicc_lam = lambdas[which(aiccs == baicc)]
    ber = min(ers)
    er_lam = lambdas[which(ers == ber)]
```

```
    ix = (j-1)*M + i
    noise_lam_df$J[ix] = j
    noise_lam_df$cv[ix] = cv_lam
    noise_lam_df$gcv[ix] = gcv_lam
    noise_lam_df$aic[ix] = aicc_lam
    noise_lam_df$er[ix] = er_lam

    noise_sco_df$cv[ix] = bcv
    noise_sco_df$gcv[ix] = bgcv
    noise_sco_df$aic[ix] = baicc
    noise_sco_df$er[ix] = ber

  }


  colname = paste("J=", j, sep="")
  noise_raw_df[colname] = Y
}

## Plot results

plt_noise_obv = ggplot(data=melt(noise_raw_df, id=c("x", "yt"))) +
  geom_line(data=noise_raw_df, aes(x=x, y=yt), color="steelblue") +
  geom_point(aes(x=x, y=value, color=variable),
             alpha=0.7, size=1.5, shape=18) +
  facet_wrap(~variable) +
  scale_colour_discrete(
    labels=as.character(1:J),
    name="Noise regime (J)"
  ) +
  labs(x="X", y="Y", title="Observations with different noise")

noise_lam_df_m = melt(noise_lam_df, id=c("J"))
noise_lam_df_m$J = as.factor(noise_lam_df_m$J)

ylims = boxplot.stats(noise_lam_df_m$value)$stats[c(1, 5)]

plt_noise_lams = ggplot(data=noise_lam_df_m) +
  geom_boxplot(aes(x=J, y=value, color=J), outlier.shape=NA) +
  coord_cartesian(ylim = ylims*1.05) +
  facet_wrap(~variable) +
  scale_y_log10() +
  scale_colour_discrete(
    labels=as.character(1:J),
    name="Noise regime (J)"
  ) +
  labs(x="Noise regime", y=TeX("Optimal $\\lambda$"),
       title=TeX("Stability of different methods for estimating $\\lambda$"))
```

```
## Parameterize
lambdas = 10^seq(-9,-4, length.out=L)

# Data storage objects
xts = ((1:n)-0.05)/n
yts = f(xts)
dens_raw_df = data.frame(x=xts, y=yts, j=rep(0, n)) # observations for plotting
dens_lam_df = data.frame( # best lambda for each simulation 1:M
  cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
  J=rep(NA, J*M))
dens_sco_df = data.frame( # score of the corresponding best lambda
  cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
  J=rep(NA, J*M))

D = diag(c(rep(0, p+1), rep(1, k)))

for(j in 1:J){

  for(i in 1:M){


    xs = qbeta(runif(n), (j+4)/5, (11-j)/5)
    fxs = f(xs)
    X1 = outer(xs, 0:p, "^")
    X2 = outer(xs, knots, ">")*outer(xs, knots, "-")^p
    X = cbind(X1, X2)


    Y = fxs + rnorm(n, mean=0, sd=0.1)

    # calculate lambda using each method
    cvs = cvv(k=k, l=lambdas, X=X, Y=Y)
    gcvs = gcvv(k=k, l=lambdas, X=X, Y=Y)
    aiccs = aiccv(k=k, l=lambdas, X=X, Y=Y)

    bcv = min(cvs)
    cv_lam = lambdas[which(cvs == bcv)]

    ers = erv(k=k, l=lambdas, X=X, Y=Y, lp=cv_lam)

    bgcv = min(gcvs)
    gcv_lam = lambdas[which(gcvs == bgcv)]
    baicc = min(aiccs)
    aicc_lam = lambdas[which(aiccs == baicc)]
    ber = min(ers)
    er_lam = lambdas[which(ers == ber)]

    ix = (j-1)*M + i
    dens_lam_df$J[ix] = j
    dens_lam_df$cv[ix] = cv_lam
    dens_lam_df$gcv[ix] = gcv_lam
    dens_lam_df$aic[ix] = aicc_lam
    dens_lam_df$er[ix] = er_lam
```

```
    dens_sco_df$cv[ix] = bcv
    dens_sco_df$gcv[ix] = bgcv
    dens_sco_df$aic[ix] = baicc
    dens_sco_df$er[ix] = ber

  }


  tdf = data.frame(x=xs, y=Y, j=rep(j, n))
  dens_raw_df = rbind(dens_raw_df, tdf)
}

dens_raw_df$j = as.factor(dens_raw_df$j)
drdf0 = subset(dens_raw_df, j == 0, select=c("x","y"))
plt_dens_obv = ggplot(data=drdf0) +
  geom_line(aes(x=x,y=y), color="steelblue") +
  geom_point(data=subset(dens_raw_df, j != 0), aes(x=x, y=y, color=j),
             shape=18, size=1.5) +
  facet_wrap(~j) +
  scale_colour_discrete(
    labels=as.character(1:J),
    name="Support variance regime (J)"
  ) +
  labs(x="X", y="Y", title="Observations with variable support density")

dens_lam_df_m = melt(dens_lam_df, id=c("J"))
dens_lam_df_m$J = as.factor(dens_lam_df_m$J)
ylims = boxplot.stats(dens_lam_df_m$value)$stats[c(1, 5)]
plt_dens_lams = ggplot(data=dens_lam_df_m) +
  geom_boxplot(aes(x=J, y=value, color=J), outlier.shape=NA) +
  scale_y_log10() +
  coord_cartesian(ylim = ylims*1.05) +
  facet_wrap(~variable) +
  scale_colour_discrete(
    labels=as.character(1:J),
    name="Support variance regime (J)"
  ) +
  labs(x="Support variance regime", y=TeX("Optimal $\\lambda$"),
       title=TeX("Stability of different methods for estimating $\\lambda$"))
```

```
## Parameterize
xs = ((1:n)-0.5)/n
lambdas = 10^seq(-11,-1, length.out=L)


ff = function(x, j){
    r = sqrt(x*(1-x))*sin((2*pi*(1+2^((9-4*j)/5)))/(x+2^((9-4*j)/5)))
    return(r)
}

# Data storage objects

sp_raw_df = data.frame(x=NA, yt=NA, ys=NA, j=NA)
sp_lam_df = data.frame(
    cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
    J=rep(NA, J*M))

D = diag(c(rep(0, p+1), rep(1, k)))
X1 = outer(xs, 0:p, "^")
X2 = outer(xs, knots, ">")*outer(xs, knots, "-")^p
X = cbind(X1, X2)

for(j in 1:J){

    fxs = ff(xs, j)

    for(i in 1:M){


        Y = fxs + rnorm(n, mean=0, sd=0.2)

        # calculate lambda using each method
        cvs = cvv(k=k, l=lambdas, X=X, Y=Y)
        gcvs = gcvv(k=k, l=lambdas, X=X, Y=Y)
        aiccs = aiccv(k=k, l=lambdas, X=X, Y=Y)

        bcv = min(cvs)
        cv_lam = lambdas[which(cvs == bcv)]

        ers = erv(k=k, l=lambdas, X=X, Y=Y, lp=cv_lam)

        bgcv = min(gcvs)
        gcv_lam = lambdas[which(gcvs == bgcv)]
        baicc = min(aiccs)
        aicc_lam = lambdas[which(aiccs == baicc)]
        ber = min(ers)
        er_lam = lambdas[which(ers == ber)]

        ix = (j-1)*M + i
        sp_lam_df$J[ix] = j
        sp_lam_df$cv[ix] = cv_lam
        sp_lam_df$gcv[ix] = gcv_lam
        sp_lam_df$aic[ix] = aicc_lam
```

```
        sp_lam_df$er[ix] = er_lam

    }


    tdf = data.frame(x=xs, yt=fxs, ys=Y, j=rep(j, n))
    sp_raw_df = rbind(sp_raw_df, tdf)
}

sp_raw_df = subset(sp_raw_df, ! is.na(j))
sp_raw_df$j = as.factor(sp_raw_df$j)
plt_sp_obv = ggplot(data=sp_raw_df) +
    geom_line(aes(x=x,y=yt), color="steelblue") +
    geom_point(aes(x=x, y=ys, color=j), shape=18, size=1.5) +
    facet_wrap(~j) +
    scale_colour_discrete(
        labels=as.character(1:J),
        name="Function frequency regime (J)"
    ) +
    labs(x="X", y="Y",
        title="Observations generated by different target function frequencies")

sp_lam_df_m = melt(sp_lam_df, id=c("J"))
sp_lam_df_m$J = as.factor(sp_lam_df_m$J)
plt_sp_lams = ggplot(data=sp_lam_df_m) +
    geom_boxplot(aes(x=J, y=value, color=J)) +
    scale_y_log10() +
    facet_wrap(~variable) +
    scale_colour_discrete(
        labels=as.character(1:J),
        name="Function frequency regime (J)"
    ) +
    labs(x="Target function frequencies", y=TeX("Optimal $\\lambda$"),
         title=TeX("Stability of different methods for estimating $\\lambda$"))
```

```r
## Parameterize

xs = ((1:n)-0.5)/n
lambdas = 10^seq(-11,-1, length.out=L)


fv = function(x, j){
    r = (0.15*(1+0.4*(2*j-7)*(x-0.5)))^2
    return(sqrt(r))
}


# Data storage objects

v_raw_df = data.frame(x=NA, yt=NA, ys=NA, j=NA)
v_lam_df = data.frame(
    cv=rep(NA, J*M), gcv=rep(NA, J*M), aic=rep(NA, J*M), er=rep(NA, J*M),
    J=rep(NA, J*M))

D = diag(c(rep(0, p+1), rep(1, k)))
X1 = outer(xs, 0:p, "^")
X2 = outer(xs, knots, ">")*outer(xs, knots, "-")^p
X = cbind(X1, X2)
fxs = f(xs)

for(j in 1:J){

    for(i in 1:M){


        Y = fxs + fv(xs, j)*rnorm(n, mean=0, sd=1)

        # calculate lambda using each method
        cvs = cvv(k=k, l=lambdas, X=X, Y=Y)
        gcvs = gcvv(k=k, l=lambdas, X=X, Y=Y)
        aiccs = aiccv(k=k, l=lambdas, X=X, Y=Y)

        bcv = min(cvs)
        cv_lam = lambdas[which(cvs == bcv)]

        ers = erv(k=k, l=lambdas, X=X, Y=Y, lp=cv_lam)

        bgcv = min(gcvs)
        gcv_lam = lambdas[which(gcvs == bgcv)]
        baicc = min(aiccs)
        aicc_lam = lambdas[which(aiccs == baicc)]
        ber = min(ers)
        er_lam = lambdas[which(ers == ber)]

        ix = (j-1)*M + i
        sp_lam_df$J[ix] = j
        sp_lam_df$cv[ix] = cv_lam
        sp_lam_df$gcv[ix] = gcv_lam
```

```
        sp_lam_df$aic[ix] = aicc_lam
        sp_lam_df$er[ix] = er_lam

    }


    tdf = data.frame(x=xs, yt=fxs, ys=Y, j=rep(j, n))
    v_raw_df = rbind(v_raw_df, tdf)
}

v_raw_df = subset(v_raw_df, ! is.na(j))
v_raw_df$j = as.factor(v_raw_df$j)
plt_v_obv = ggplot(data=v_raw_df) +
    geom_line(aes(x=x,y=yt), color="steelblue") +
    geom_point(aes(x=x, y=ys, color=j), shape=18, size=1.5) +
    facet_wrap(~j) +
    scale_colour_discrete(
        labels=as.character(1:J),
        name="Variance regime (J)"
    ) +
    labs(x="X", y="Y",
        title="Observations generated by different variance structures")

v_lam_df_m = melt(sp_lam_df, id=c("J"))
v_lam_df_m$J = as.factor(v_lam_df_m$J)
plt_v_lams = ggplot(data=v_lam_df_m) +
    geom_boxplot(aes(x=J, y=value, color=J)) +
    scale_y_log10() +
    facet_wrap(~variable) +
    scale_colour_discrete(
        labels=as.character(1:J),
        name="Variance regime (J)"
    ) +
    labs(x="Variance structures", y=TeX("Optimal $\\lambda$"),
         title=TeX("Stability of different methods for estimating $\\lambda$"))
```