Regression Analysis HW2

Tuorui Peng, tuoruipeng2028@u.northwestern.edu

Notation: I use abbr SMW for Sherman-Morrison-Woodbury Formula, which we proved in the first homework.

$$(A+UCV')^{-1}=A^{-1}-A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$$

Question 1.1

$\hat{\beta}_{n+1}$ is obtained by

$$\begin{aligned}
0 =&\frac{\partial}{\partial\beta}(Y-X\beta)'(Y-X\beta)+(y_{n+1}-x_{n+1}'\beta)^2\\
=&-2X'(Y-X\beta)+2(y_{n+1}-x_{n+1}'\beta)(-x_{n+1})\\
\Rightarrow&(X'X+x_{n+1}x_{n+1}')\beta=X'Y+x_{n+1}y_{n+1}\\
\Rightarrow&\beta=(X'X+x_{n+1}x_{n+1}')^{-1}(X'Y+x_{n+1}y_{n+1})\\
\underset{\text{SMW}}{=}&(I-\frac{(X'X)^{-1}x_{n+1}x_{n+1}'}{1+x_{n+1}'(X'X)^{-1}x_{n+1}})(X'X)^{-1}(X'Y+x_{n+1}y_{n+1})\\
=&(I-\frac{(X'X)^{-1}x_{n+1}x_{n+1}'}{1+x_{n+1}'(X'X)^{-1}x_{n+1}})(\hat{\beta}_n-(X'X)^{-1}x_{n+1}y_{n+1}')
\end{aligned}$$

In which the computation cost is estimated as:

- $(X'X)^{-1}x_{n+1}$: $O(d^2)$
- $(X'X)^{-1}x_{n+1}x_{n+1}'$: $O(d)$
- $x_{n+1}'(X'X)^{-1}x_{n+1}$: $O(d)$
- $(I-\frac{(X'X)^{-1}x_{n+1}x_{n+1}'}{1+x_{n+1}'(X'X)^{-1}x_{n+1}})(\hat{\beta}_n-(X'X)^{-1}x_{n+1}y_{n+1}')$: $O(2d^2)$

In total: $\sim O(3d^2+2d)$

Question 1.2

Here I solve the sub-problem (b) directly, in which we just need to check the positive semi-definiteness of matrix $C_\rho \in \mathbb{R}^{n\times n}$, which is equivalent to check the sign of its determinants of $C_\rho \in \mathbb{R}^{m\times m}$, $\forall m \leq n$:

$$
\begin{aligned}
\det \underset{m\times m}{C_\rho} &= \det \begin{bmatrix} 1 & -\rho & \cdots & -\rho \\ -\rho & 1 & \cdots & -\rho \\ \vdots & \vdots & \ddots & \vdots \\ -\rho & -\rho & \cdots & 1 \end{bmatrix}_{m\times m} \\
&= \det \begin{bmatrix} 1 & -\rho & \cdots & -\rho \\ -1-\rho & 1+\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1-\rho & 0 & \cdots & 1+\rho \end{bmatrix}_{m\times m} \\
&= \det \begin{bmatrix} 1-(m-1)\rho & -\rho & \cdots & -\rho \\ 0 & 1+\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1+\rho \end{bmatrix}_{m\times m} \\
&= (1-(m-1)\rho)\prod_{i=2}^{m}(1+\rho) = (1-(m-1)\rho)(1+\rho)^{m-1}
\end{aligned}
$$

We have positive semi-definiteness of $\underset{n\times n}{C_\rho}$ iff $1-(m-1)\rho \geq 0$, $\forall m \leq n$ iff $\rho \leq \dfrac{1}{n-1}$.

Question 1.3

```r
wine <- read.csv("winequality-red.csv", sep = ";")
pairs(wine[, c("quality", "density", "alcohol", "pH", "volatile.acidity")])
```

We have a strong positive relation between `quality` and `alcohol`, and a strong negative relation between `density` and `alcohol`.

Question 1.4

Question 1.5

```r
sfo <- read.csv("simplified-sfo-weather.csv", sep = ",")
```

(a)

Verify directly we have

$$\begin{aligned}
cov(Y-\hat{Y}, Y_{\text{new}}-\hat{Y}_{\text{new}}) =& cov(X\beta+\varepsilon-X(X'X)^{-1}X(X\beta+\varepsilon), Z\beta+\varepsilon_{\text{new}}-Z(X'X)^{-1}X'(X\beta+\varepsilon)) \\
=& cov((I-X(X'X)^{-1}X')\varepsilon, \varepsilon_{\text{new}}-Z(X'X)^{-1}X'\varepsilon) \\
=& cov((I-X(X'X)^{-1}X')\varepsilon, -Z(X'X)^{-1}X'\varepsilon) \\
=& -(I-X(X'X)^{-1}X')\sigma^2 IX(X'X)^{-1}Z' = 0
\end{aligned}$$

(b)

We have

$$\begin{aligned}
Y_{\text{new}}-\hat{Y}_{\text{new}} =& Z\beta+\varepsilon_{\text{new}}-Z(X'X)^{-1}X'(X\beta+\varepsilon) \\
=& \varepsilon_{\text{new}}-Z(X'X)^{-1}X'\varepsilon \\
\sim& N(0, \sigma^2(I-Z(X'X)^{-1}Z'))
\end{aligned}$$

So the matrix M s.t. $M(Y_{\text{new}}-\hat{Y}_{\text{new}}) \sim N(0,\sigma^2 I)$ can be chosen as

$$\begin{aligned}
M =& (I-Z(X'X)^{-1}Z')^{-1/2} \\
\underset{\text{SMW}}{=}& (I-Z(X'X+Z'Z)^{-1}Z')^{1/2}
\end{aligned}$$

(c)

Since we have proven the normality and the independence between $Y-\hat{Y}$ and $Y_{\text{new}}-\hat{Y}_{\text{new}}$ (which is equivalent to covariance being zero for normal variables), we have

$$A = \frac{\|M(Y_{\text{new}}-\hat{Y}_{\text{new}})\|_2^2/n}{\|Y-\hat{Y}\|_2^2/(m-d)} \sim F_{n,m-d}$$

(d)

```r
library("tidyverse")
mat_inverse_sqrt <- function(mat) {
    a <- eigen(mat)
    idx <- which(a$value > 1e-8)
    return(a$vector[, idx] %*% diag(1 / sqrt(a$value[idx])) %*% t(a$vector[, idx]))
}

f_stat <- function(X, Z, Y, Y_new) {
    X <- as.matrix(X)
    Z <- as.matrix(Z)
    m <- nrow(X)
    n <- nrow(Z)
    d <- ncol(X)
    beta <- solve(t(X) %*% X) %*% t(X) %*% Y
    Y_hat <- X %*% beta
    Y_new_hat <- Z %*% beta
    eps <- Y - Y_hat
    eps_new <- Y_new - Y_new_hat
    M <- mat_inverse_sqrt(diag(n) - Z %*% solve(t(X) %*% X) %*% t(Z))
    A <- (sum((M %*% eps_new)^2) / n) / (sum(eps^2) / (m - d))
    return(A)
}
time_to_X <- function(date_seq) {
    df <- data.frame(interc = rep(1, length(date_seq)), sincomp = sin(2 * pi * date_seq / 365.25), co
    return(df)
}

years <- c(1966:2020)
p_values <- c()
for (year in years) {
    X <- time_to_X(sfo$day[sfo$year < year])
    Z <- time_to_X(sfo$day[sfo$year == year])
    Y <- sfo$precip[sfo$year < year]
    Y_new <- sfo$precip[sfo$year == year]
    dof1 <- nrow(Z)
    dof2 <- nrow(X) - ncol(X)
    p_values <- c(p_values, 1 - pf(f_stat(X, Z, Y, Y_new), dof1, dof2))
}
names(p_values) <- years
p_values

# plot of p-value v.s. year
ggplot(data.frame(years = years, p_values = (p_values)), aes(x = years, y = p_values)) +
    geom_line() +
    geom_point() +
    geom_hline(yintercept = 0.05, linetype = "dashed") +
    labs(x = "year", y = "p-value") +
    ggtitle("p-value of F-test for precipitation pattern change")

# plot of log(p-value) v.s. year
ggplot(data.frame(years = years, p_values = log(p_values)), aes(x = years, y = p_values)) +
    geom_line() +
    geom_point() +
    labs(x = "year", y = "log(p-value)") +
    ggtitle("log(p-value) of F-test for precipitation pattern change")
```

In the above we plotted log(p-value) v.s. year plot and p-value v.s. year plot. Seems in most years we don't reject the null hypothesis that there's no change in the precipitation pattern. But there are some years we observe significant low p-value, suggesting a rejection to null hypothesis.

(e)

I would say that 'changing over time' should be some kind of smooth, structural change, in which sense we should observe a long-range low p-value in the p-value v.s. year plot. But in the above plot we don't observe such a long-range low p-value. Actually we can see that in most of the years the p-value is nearly one, suggesting no change in the precipitation pattern. So I would say these 'outlier' p-values might just due to some occassional, short-term incidents happening in those years, instead of a long-term 'change in the precipitation pattern'.

Question 2.1

(a)

Using any test function $g \in \mathcal{L}^2(\mathbb{R}^d)$ with $t \in \mathbb{R}$ to denote $f^*(x) = \mathbb{E}[Y|X = x] = (f - tg)(x)$, i.e. $f(x) = \mathbb{E}[Y|X = x] + tg(x)$

$$\begin{aligned}
\mathbb{E}[(Y - f(X))^2] =& \mathbb{E}[(Y - (f^* + tg)(X))^2] \\
=& \mathbb{E}[(Y - f^*(X))^2] - 2t\mathbb{E}[(Y - f^*(X))(g(X))] + t^2\mathbb{E}[(g(X))^2] \\
=& \mathbb{E}[(Y - f^*(X))^2] - 2t\mathbb{E}_X\mathbb{E}_Y[(Y - f^*(X))g(X)|X] + t^2\mathbb{E}[(g(X))^2] \\
=& \mathbb{E}[(Y - f^*(X))^2] + t^2\mathbb{E}[(g(X))^2] \\
\geq& \mathbb{E}[(Y - f^*(X))^2], \quad \forall t \in \mathbb{R}, g \in \mathcal{L}^2(\mathbb{R}^d)
\end{aligned}$$

which proves that $f^*(x) = \mathbb{E}[Y|X = x]$ is the minimizer of $\mathbb{E}[(Y - f(X))^2]$.

(b)

Minimization of the expected squared loss yields

$$\begin{aligned}
0 = \frac{\partial}{\partial b}L(b) =& \frac{\partial}{\partial b}\mathbb{E}[(Y - X'b)^2] = \mathbb{E}[\frac{\partial}{\partial b}(Y - X'b)^2] \\
=& \mathbb{E}[-2(Y - X'b)X] = \mathbb{E}[2XX'b - 2XY]
\end{aligned}$$

So here again we have the solution as $\hat{\beta} = \mathbb{E}[XX']^{-1} mathbbE[XY]$. To verify its optimality, we denote any elements in linear space as $b = \hat{\beta} + ta$, $t \in \mathbb{R}$, $a \in \mathbb{R}^n$, and follow similar steps as in (a):

$$\begin{aligned} L(b) = L(\hat{\beta} + ta) =& \mathbb{E}[(Y - X'(\hat{\beta} + ta))^2] \\ =& \mathbb{E}[(Y - X'\hat{\beta})^2] - 2t\mathbb{E}_X\mathbb{E}_Y[(Y - \hat{\beta}'X)(X'a)|X] + t^2\mathbb{E}[(X'a)^2] \\ =& \mathbb{E}[(Y - X'\hat{\beta})^2] - 2t\mathbb{E}_X[\mathbb{E}\left[Y|X\right]\left(I - X'\mathbb{E}\left[X'X\right]X\right)X'a] + t^2\mathbb{E}[(X'a)^2] \\ =& \mathbb{E}[(Y - X'\hat{\beta})^2] + t^2\mathbb{E}[(X'a)^2] \\ \geq& \mathbb{E}[(Y - X'\hat{\beta})^2], \quad \forall t \in \mathbb{R},\, a \in \mathbb{R}^n \end{aligned}$$

which proves that $\hat{\beta} = \mathbb{E}[XX']^{-1} mathbbE[XY]$ is the minimizer of $\mathbb{E}[(Y - X'b)^2]$.

(c)

Here we use $\{\mathbf{X}, \mathbf{Y}\}$ to represent the sample. From OLS estimator we know that $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$. We can see that on population level

$$\beta^* = \mathbb{E}\left[(\mathbf{X}'\mathbf{X})^{-1}\right]\mathbb{E}\left[\mathbf{X}'\mathbf{Y}\right] = \mathbb{E}[\mathbf{X}\mathbf{X}']^{-1}\mathbb{E}[\mathbf{X}\mathbf{Y}]$$

while the estimator

$$\mathbb{E}\left[\hat{\beta}\right] = \mathbb{E}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}\right] \neq \mathbb{E}[\mathbf{X}\mathbf{X}']^{-1}\mathbb{E}[\mathbf{X}\mathbf{Y}], \quad \text{generally speaking.}$$

Question 2.2

Using the QR decomposition of X, say $X = QR$, we can express

$$Y = X\beta + \epsilon = QR\beta + \epsilon$$

Multiply both sides by Q', and we make some now notation

$$Z = Q'Y = Q'QR\beta + Q'\epsilon := R\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2 I)$$

And since Q as a unitary matrix, we can solve the problem for Z, R here. With R being upper triangular, and WLOG take $j = d$ we obtain:

$$\begin{aligned} \hat{\beta}_d =& \frac{z_d}{R_{dd}} \\ var(\hat{\beta}_d) =& \sigma^2(R'R)^{-1}_{dd} \underset{\text{cramer's rule}}{=} \sigma^2\frac{\det(R'R)^*_{dd}}{\det(R'R)} = \frac{\sigma^2}{R^2_{dd}} \\ v\hat{a}r(\hat{\beta}_d) =& \frac{\hat{\sigma}^2}{R^2_{dd}} \\ \Rightarrow& \begin{cases} Hz = R\hat{\beta} \\ H_d z = R_{\wedge d}\hat{\beta}_{\wedge d} \end{cases} \Rightarrow (H - H_d)z = R(\hat{\beta} - \hat{\beta}_{\wedge d}) = R_{dd}\frac{z_d}{R_{dd}} = z_d \end{aligned}$$

with the above results we obtain that

$$t_j^2 = \frac{(\hat{\beta}_d)^2}{\hat{se}(\hat{\beta}_j)} = \frac{(z_d/R_{dd})^2}{\hat{\sigma}^2/R_{dd}^2} = \frac{\|(H-H_d)z\|_2^2}{\text{MSE}} = F_j, \ \forall j \in [d]$$

Question 2.3

(a)

Note that $(\frac{1}{n}\mathbf{1}\mathbf{1}')^2 = \frac{1}{n}\mathbf{1}\mathbf{1}'$ and that $I' = I$, $(\mathbf{1}\mathbf{1}')' = \mathbf{1}\mathbf{1}'$. Verify directly:

$$(I - \frac{1}{n}\mathbf{1}\mathbf{1}')Z \sim N(0, (I - \frac{1}{n}\mathbf{1}\mathbf{1}')'C_\rho(I - \frac{1}{n}\mathbf{1}\mathbf{1}')) = N(0, (1-\rho)(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}'))$$

(b)

Using the result from the previous question, we have

$$Y_i - \bar{Y}_i\mathbb{1} \sim N(0, (1-\rho)(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}')), \quad i = 1, 2$$

degree of freedom of squared value:

$$\text{dof}\,\frac{(Y_i - \bar{Y}_i\mathbb{1})'(Y_i - \bar{Y}_i\mathbb{1})}{1-\rho} = \text{rank}(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}') = n-1$$

then we have

$$\begin{aligned}\frac{1}{1-\rho}S_n^2 =& \frac{1}{1-\rho}\frac{\sum_{i=1}^n (Y_{1i} - \bar{Y}_1)^2 + \sum_{i=1}^n (Y_{2i} - \bar{Y}_2)^2}{2(n-1)} \\ \sim& \frac{\sigma^2}{2(n-1)}\chi^2_{2n-2}\end{aligned}$$

(c)

Note that

$$\bar{Y}_i = \frac{1}{n}\mathbf{1}'Y_i \sim N(\mu + \alpha_i, \frac{\sigma^2(1-\rho)}{n} + \sigma^2\rho)$$

then we have

$$\bar{Y}_1 - \bar{Y}_2 \sim N(0, 2\sigma^2(\frac{1-\rho}{n} + \rho))$$

(d)

Since both $\bar{Y}_1 - \bar{Y}_2$ and $Y_{ij} - \bar{Y}_i$ $\forall i \in \{1, 2\}, 1 \le j \le n$ are normal, and S_n^2 is function of $Y_{ij} - \bar{Y}_i$, it suffices to show that

$$cov(\bar{Y}_1 - \bar{Y}_2, Y_{ij} - \bar{Y}_i) = 0, \quad \forall i \in \{1, 2\}, 1 \le j \le n$$

We can obtain that

$$\begin{aligned}cov(\bar{Y}_1-\bar{Y}_2, Y_{ij}-\bar{Y}_i) =& cov(\varepsilon_1'\frac{1}{n}\mathbf{1}-\varepsilon_2'\frac{1}{n}\mathbf{1}, \varepsilon_{1j}-\varepsilon_1'\frac{1}{n}\mathbf{1})\\ =& cov(\frac{1}{n}\mathbf{1}'\varepsilon_i, \varepsilon_{1j}-\frac{1}{n}\varepsilon_i'\mathbf{1})\\ =& \frac{1}{n}(1+(n-1)\rho)-\frac{1}{n^2}(n(1-\rho)+n^2\rho)=0\end{aligned}$$

thus we have the independence relation $\bar{Y}_1-\bar{Y}_2 \perp\!\!\!\perp S_n^2$

(e)

Using the distribution of $\bar{Y}_1-\bar{Y}_2$, we have

$$\frac{(\bar{Y}_1-\bar{Y}_2)^2}{2\sigma^2(\frac{1-\rho}{n}+\rho)} \sim \chi_1^2$$

Then

$$\frac{\frac{n}{2(1-\rho)+2\rho n}(\bar{Y}_1-\bar{Y}_2)^2}{\frac{1}{1-\rho}S_n^2} \sim F_{1,2n-2}$$

(f)

For $n \to \infty$, notice that

$$\hat{F} = \frac{1-\rho+\rho n}{1-\rho}\hat{F}_\rho \to \infty$$

which causes frequent rejection of H_0.

Question 2.4

(a)

A direct result by left multiply $\Sigma^{-1/2}$:

$$\Sigma^{-1/2}y = \Sigma^{-1/2}X\beta + \Sigma^{-1/2}\varepsilon = \Sigma^{-1/2}X\beta + \xi$$

in which $\xi = \Sigma^{-1/2}\varepsilon \sim N(0, I)$.

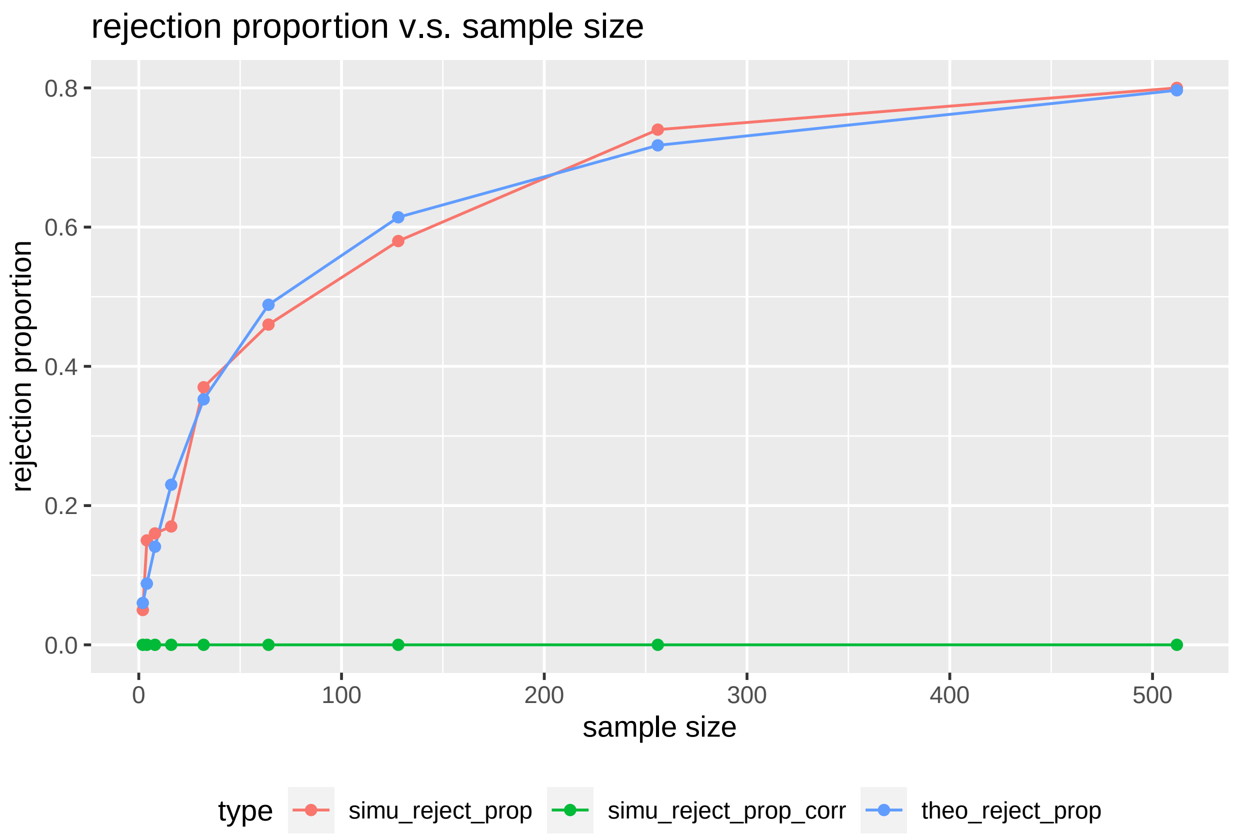
(b)(c)

Simulation with correction to correlation matrix:

```r
library('mvtnorm')
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
set.seed(42)

N <- 1e2
nsamples <- 2^(1:9)
rho <- 0.1

f.stat.mat <- matrix(0, nrow = N, ncol = length(nsamples))
f.stat.mat.corr <- matrix(0, nrow = N, ncol = length(nsamples))

for(nsample in nsamples){
    for(n in 1:N){
        Sigma <- matrix(rho, nsample, nsample) + diag(1 - rho, nsample)
        X_1 <- rmvnorm(1, rep(0, nsample), Sigma) %>% t()
        X_2 <- rmvnorm(1, rep(0, nsample), Sigma) %>% t()
        f.stat <- (t.test(X_1, X_2, var.equal = TRUE)$statistic)^2
        hatsigma2 <- (apply(as.matrix(X_1), 1, function(x) (x-X_2)^2) %>% sum()) / (2*nsample^2)
        Sn2 <- (var(X_1)+var(X_2))/2
        hatrho <- max(1 - Sn2/hatsigma2, 0)
        f.stat.mat[n, which(nsamples == nsample)] <- f.stat
        f.stat.mat.corr[n, which(nsamples == nsample)] <- f.stat * (1 - hatrho) / (1 - hatrho + hatrh
    }
}

# calculate rejection proportion
reject_prop <- apply(f.stat.mat, 1, function(x)x > qf(0.95, 1, 2 * nsamples - 2)) %>% t() %>% apply(2
reject_prop.corr <- apply(f.stat.mat.corr, 1, function(x)x > qf(0.95, 1, 2 * nsamples - 2)) %>% t() %

# theoretical rejection proportion
theo_reject_prop <- 1 - pf(qf(0.95, 1, 2 * nsamples - 2) * (1 - rho) / (1 - rho + rho*nsamples), 1, 2
# plotting of rejection proportion v.s. sample size
plot_dat <- data.frame(nsamples = nsamples, simu_reject_prop = reject_prop, theo_reject_prop = theo_r
ggplot(plot_dat, aes(x = nsamples, y = rejection_prop, color = type)) + geom_line() + geom_point() +
```



Question 2.5

Question 2.6

(a)

A main problem of the figure is that it actually only test the $m = 5$ methods on $p = 1$ dataset (problem), and show the training process. Actually if we want to illustrate the 'generalization ability' of the methods, we should test the methods on multiple datasets.

(b)

First definitely runtime $R \geq 0$, so our normality assumption might not be that precise, and might cause confusion in understanding the quantative relations between α, β and runtime.

(c)

The transformation

$$Y_{ijk} = \phi(R_{ijk}) = \mu + \alpha_i + \beta_j + \varepsilon_{ijk}$$

should satisfy

$$\phi(2r) = 1 + \phi(r) \Rightarrow \phi(\,\cdot\,) = \log_2(\,\cdot\,)$$

(d)

The minimizer

$$\begin{aligned}(\hat{\mu}, \hat{\alpha}_{i=1}^m, \hat{\beta}_{j=1}^p) &= \underset{(\mu, \alpha_{i=1}^m, \beta_{j=1}^p)}{\arg\min} \sum_{i,j,k=1}^{m,p,n} \left(Y_{ijk} - \mu - \alpha_i - \beta_j\right)^2 \\ &:= \underset{(\mu, \alpha_{i=1}^m, \beta_{j=1}^p)}{\arg\min} \mathcal{L}(\mu, \alpha_{i=1}^m, \beta_{j=1}^p), \quad w.r.t. \sum_{i=1}^m \alpha_i = \sum_{j=1}^p \beta_j = 0\end{aligned}$$

satisties

$$\begin{cases} 0 = \frac{\partial \mathcal{L}}{\partial \mu} = -\sum_{i,j,k=1}^{m,p,n} 2(Y_{ijk} - \mu - \alpha_i - \beta_j) & \Rightarrow \hat{\mu} = \dfrac{\sum_{i,j,k=1}^{m,p,n} Y_{ijk}}{mnp} = \bar{Y}_{\cdots} \\ 0 = \frac{\partial \mathcal{L}}{\partial \alpha_i} = -\sum_{j,k=1}^{p,n} 2(Y_{ijk} - \mu - \alpha_i - \beta_j) & \Rightarrow \hat{\alpha}_i = \dfrac{\sum_{j,k=1}^{p,n} Y_{ijk}}{pn} - \bar{Y}_{\cdots} = \bar{Y}_{i\cdot\cdot} - \bar{Y}_{\cdots}, \quad \forall i \in [m] \\ 0 = \frac{\partial \mathcal{L}}{\partial \beta_j} = -\sum_{i,k=1}^{m,n} 2(Y_{ijk} - \mu - \alpha_i - \beta_j) & \Rightarrow \hat{\beta}_j = \dfrac{\sum_{i,k=1}^{m,n} Y_{ijk}}{mn} - \bar{Y}_{\cdots} = \bar{Y}_{\cdot j\cdot} - \bar{Y}_{\cdots}, \quad \forall j \in [p] \end{cases}$$

(e)

We have

$$\hat{\alpha}_1 - \hat{\alpha}_i = \bar{Y}_{1\cdot\cdot} - \bar{Y}_{i\cdot\cdot} \sim N(\alpha_1 - \alpha_i, \frac{2\sigma^2}{np}), \quad \forall i \in \{2, \cdots, m\}$$

and each $\hat{\alpha}_1 - \hat{\alpha}_i$ and $\hat{\alpha}_1 - \hat{\alpha}_j$ are independent if $i \neq j$. For each individual test

$$H_{0i} : \alpha_1 \geq \alpha_i$$

we can use the one-sided t-test

$$T_i = \frac{\hat{\alpha}_1 - \hat{\alpha}_i}{\sqrt{\dfrac{2\hat{\sigma}^2}{np}}} \sim t_{N-m-p+1}, \qquad \hat{\sigma}^2 = S^2 = \frac{1}{N-m-p+1} \sum_{i,j,k=1}^{m,p,n} (Y_{ijk} - \bar{Y}_{ij\cdot})^2$$

reject H_{0i} if $T_i \leq t_{N-m-p+1,-a}$.

Note: here I think there's a mistake in the HW material, in which $\hat{\sigma}^2$ should have degree of freedom $N-m-p+1$ if we are using model without interaction term.

(f)

If $H_{0i}: \alpha_1 \geq \alpha_i$ holds, then

$$\begin{aligned}\mathbb{P}\left(R_{1jk} \leq R_{ijk}\right) =&\mathbb{P}\left(Y_{1jk} \leq Y_{ijk}\right)\\ =&\mathbb{P}\left(\mu+\alpha_1+\beta_j+\varepsilon_{ijk} \leq \mu+\alpha_i+\beta_j+\varepsilon_{ijk}\right)\\ =&\mathbb{P}\left(\alpha_1+\varepsilon_{1jk} \leq \alpha_i+\varepsilon_{ijk}\right)\\ =&\mathbb{P}\left(2N(0,\sigma^2) \leq \alpha_i-\alpha_1\right) \leq \frac{1}{2}, \quad \forall i \in \{2,\cdots,m\},\, j\in[p],\, k\in[n]\end{aligned}$$

(g)

Consider for given $i \geq 2$, for each $j \in [p]$, we denote

$$B_{jk} = \mathbf{1}_{R_{1jk} \leq R_{ijk}} \sim \text{Bernoulli}(\rho_j), \quad k \in [n]$$

In this way null hypotheis is $H'_{0i}: \mathbb{P}\left(R_{1jk} \leq R_{ijk}\right) = \mathbb{E}\left[B_{jk}\right] = \rho_j \leq \frac{1}{2}$, $\forall j \in [p]$, $k \in [n]$.

On the other hand, notice that for given t,

$$\mathbb{P}_{\rho_{j=1}^p}\left(\sum_{j=1}^p\sum_{k=1}^n B_{jk} \geq t\right)$$

is monotone increasing w.r.t. ρ_j $\forall j \in [p]$, and note that if null H'_{0i} holds we have

$$\sum_{j=1}^p\sum_{k=1}^n B_{jk} \sim \text{Binom}(\frac{1}{2}, np)$$

then we can obtain p-value as

$$\hat{p} = \mathbb{P}_\rho\left(\sum_{j=1}^p\sum_{k=1}^n B_{jk} \geq t\right) \leq \mathbb{P}_{\{1/2\}^p}\left(\sum_{j=1}^p\sum_{k=1}^n B \geq t\right) = \mathbb{P}\left(\text{Binom}(np, 1/2) \geq t\right) := \tilde{p}$$

since here $\hat{p} \leq \tilde{p}$, then we can examine threshold on $\tilde{p}? \leq a$, if so then we can definitely reject H'_{0i}.

(h)

```r
runtimes <- read.csv("runtimes.csv", sep = ",")

## data processing
runtimes$y <- log2(runtimes$runtime)
runtimes$alg.name <- as.factor(runtimes$alg.name)
runtimes$prob.ind <- as.factor(runtimes$prob.ind)
N <- nrow(runtimes)
m <- length(unique(runtimes$alg.name))
```

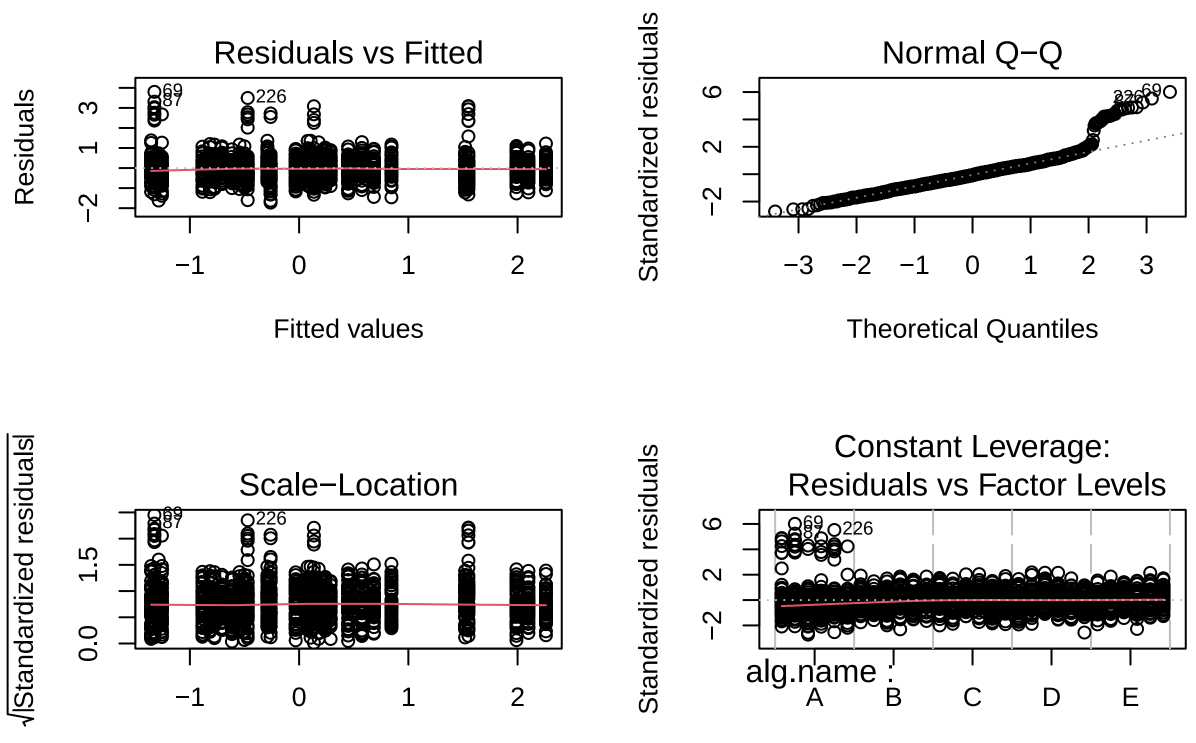
```
p <- length(unique(runtimes$prob.ind))
n <- N / (m * p)

## Simulation for $H_{0i}: \alpha _1 \leq \alpha _i, \forall i\geq 2$
aovfit <- aov(y ~ alg.name + prob.ind, data = runtimes)
S2 <- (aovfit$residuals)^2 %>% sum() / (N - m - p + 1)
p_values <- c()
for(alg in c('alg.nameB', 'alg.nameC', 'alg.nameD', 'alg.nameE')){
    alpha_diff <- -aovfit$coefficients[alg]
    t_stat <- alpha_diff / sqrt(2 * S2 / (n * p))
    p_values <- c(p_values, pt(t_stat, N - m - p + 1))
}
p_values_H <- p_values

## Simulation for $H_{0i}': \rho _j \leq 1/2, \forall j\in[p]$
p_values <- c()
for(alg in c('B', 'C', 'D', 'E')){
    ## first construct $B_{jk}$ matrix
    diff_1i <- runtimes$y[runtimes$alg.name == 'A'] - runtimes$y[runtimes$alg.name == alg]
    B_mat <- matrix(diff_1i<=0, nrow = n, ncol = p)
    ## then calculate p-value
    p_values <- c(p_values, pbinom(sum(B_mat), n * p, 1/2, lower.tail = FALSE))
}
p_values_H_prime <- p_values

print(data.frame(p_values_H, p_values_H_prime, row.names = c('B', 'C', 'D', 'E')))
```

```
##     p_values_H p_values_H_prime
## B 1.354876e-40     1.376150e-35
## C 2.682117e-17     2.051051e-22
## D 7.179311e-01     1.209099e-02
## E 1.228562e-25     4.414478e-30
```

Here we choose threshold $a = 0.05$. If we are using tests H_{0i}, we will accept H_{0D}, for which p-value is 0.72; while if we are using tests H'_{0i}, we will reject all H'_{0i}.

Observing from the following diagonisis figures, we see that actually the model does **not** fit the normality assumption quite well, which might cause the problem that the p-values are not that precise if we use H_{0i}. So I hereby choose to use tests H'_{0i} and believe that algorithm A is the best.

```
par(mfrow = c(2, 2))
plot(aovfit)
```



Question 3.1

(a)

We can simply define $Z = XV_rV_r' \in \mathbb{R}^{n\times r}$, $r \in [d]$. Then we can see that b is just the OLS estimator of Y over Z, so the dof is simply

$$\text{dof}(\hat{f}_{\text{pcr}}) = tr(Z(Z'Z)^{-1}Z') + 1 = r + 1$$

(b)

We know that ridge regression estimator is

$$\hat{\beta}_\lambda = (X'X + \lambda I)^{-1}X'Y$$

so

$$\begin{aligned}
\text{dof}(\hat{f}_{\text{ridge}}) =& \frac{1}{\sigma^2}\sum_{i=1}^{n} cov(\hat{Y}_i, Y_i)\\
=& \frac{1}{\sigma^2}\mathbb{E}\left[(Y-\mathbb{E}[Y])'\hat{Y}\right]\\
=& \frac{1}{\sigma^2}\mathbb{E}\left[(Y-\mathbb{E}[Y])'X(X'X+\lambda I)^{-1}X'Y\right]\\
=& \frac{1}{\sigma^2}\mathbb{E}\left[\varepsilon' X(X'X+\lambda I)^{-1}X'\varepsilon'\right]\\
=& tr(X(X'X+\lambda I)^{-1}X')
\end{aligned}$$

Question 3.2

(a)

The Kernel is just mapping the original data points to a new spaces, where we expect the 'structure' of the data is better captured.

(b)

Plug in the kernel expression $h(\,\cdot\,) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$, we have

$$\begin{aligned}
\hat{\alpha} =& \arg\min_{\alpha} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{n}\alpha_j k(x_i, x_j))^2 + \lambda\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(x_i, x_j)\\
=& \arg\min_{\alpha}(Y-G\alpha)'(Y-G\alpha) + \lambda\alpha' G\alpha\\
:=& \arg\min_{\alpha}\mathcal{L}(\alpha)
\end{aligned}$$

the minimizer satisties

$$\begin{aligned}
0 = \frac{\partial\mathcal{L}}{\partial\alpha} =& -2G'Y + 2G'G\alpha + 2\lambda G\alpha\\
\Rightarrow \hat{\alpha} =& (G+\lambda I)^{-1}Y
\end{aligned}$$

Yes, because the regularizer λI will help keep the matrix $G + \lambda I$ invertible.

(c)

With model $y = f(x) + \varepsilon$, $\varepsilon \sim (0, \sigma^2)$, we have

$$\begin{aligned}
\text{dof}(\hat{f}_{\text{KRR}}) =& \frac{1}{\sigma^2}\sum_{i=1}^{n} cov(\hat{Y}_i, Y_i)\\
=& \frac{1}{\sigma^2} tr(cov(G(G+\lambda I)^{-1}(f(X)+\varepsilon), f(X)+\varepsilon))\\
=& \frac{1}{\sigma^2} tr(G(G+\lambda I)^{-1}\sigma^2) = tr(G(G+\lambda I)^{-1})
\end{aligned}$$

(d)

For RBF Kernel $k(\cdot,\cdot) = \exp(-\|\cdot - \cdot\|_2^2/2\tau^2)$, we have

- **(i)** for $\tau^2 \to \infty$, we have $k(x_i, x_j) \to 1$, $\forall i, j \in [n]$, so

$$\begin{aligned}\lim_{\tau^2\to\infty} \mathrm{dof}(\hat{f}_{\mathrm{KRR}}) = \lim_{\tau^2\to\infty} tr(G(G+\lambda I)^{-1}) &= tr(\mathbb{1}\mathbb{1}'(\mathbb{1}\mathbb{1}'+\lambda I)^{-1}) \\ &\overset{\mathrm{SMW}}{=} tr((\frac{1}{\lambda}(I - \frac{\mathbb{1}\mathbb{1}'}{n+\lambda})\mathbb{1}\mathbb{1}')) = \frac{1}{\lambda}(1-\frac{n}{n+\lambda})tr(\mathbb{1}\mathbb{1}') \\ &= \frac{n}{n+\lambda}\end{aligned}$$

- **(ii)** for $\tau^2 \to 0$, we have $k(x_i, x_j) \to \delta_{ij}$, $\forall i, j \in [n]$, so

$$\lim_{\tau^2\to 0} \mathrm{dof}(\hat{f}_{\mathrm{KRR}}) = \lim_{\tau^2\to 0} tr(I(G+\lambda I)^{-1}) = tr((I+\lambda I)^{-1}) = \frac{n}{1+\lambda}$$

  if $\lambda \approx 0$ and $\tau \approx 0$ we have $\mathrm{dof}(\hat{f}_{\mathrm{KRR}}) \approx 1$, Here the 1 degree of freedom is just the intercept or 'mean value' term, so the model is just a constant model, not capturing any structure of the data.

  And in this case, the estimator is

$$\hat{y}_i = \sum_{j=1}^{n} \delta_{ij}\alpha_j = \alpha_i = \frac{y_i}{1+\lambda} \to y_i$$

  which is just a 'local' estimate, each $\hat{y}_i$ is almost its observed value y_i.

(e)

We have

$$\begin{aligned}\mathbf{f} - \mathbb{E}\left[\hat{Y}\right] &= \mathbf{f} - \mathbb{E}\left[G(G+\lambda I)^{-1}(\mathbf{f}+\varepsilon)\right] \\ &= (I - G(G+\lambda I)^{-1})\mathbf{f} \\ &\overset{\mathrm{SMW}}{=} (I - G(G^{-1} - G^{-1}(\frac{1}{\lambda}I + G^{-1})^{-1}G^{-1}))\mathbf{f} \\ &= \lambda(I+\lambda G^{-1})^{-1}G^{-1}\mathbf{f} \\ &= \lambda G^{-1}\mathbf{f} + O(\lambda^2)\end{aligned}$$

(f)

We have

$$\begin{aligned}
\mathbb{E}\left[\text{RSS}\right] =& \mathbb{E}\left[\|\hat{Y}-Y\|_2^2\right] = \mathbb{E}\left[(\mathbf{f}+\varepsilon)'(I-H_\lambda)'(I-H_\lambda)(\mathbf{f}+\varepsilon)\right] \\
=& \mathbf{f}(I-H_\lambda)'(\mathbf{f}-\mathbb{E}\left[\hat{Y}\right]) + \mathbb{E}\left[\varepsilon'(I-H_\lambda)'(I-H_\lambda)\varepsilon\right] \\
=& \mathbf{f}(I-H_\lambda)'\lambda G^{-1}\mathbf{f} + \sigma^2 tr((I-H_\lambda)'(I-H_\lambda)) + O(\lambda^2) \\
=& \lambda\mathbf{f}'G^{-1}(I-(I+\lambda G^{-1})^{-1})\mathbf{f} + \sigma^2 tr((I-H_\lambda)'(I-H_\lambda)) + O(\lambda^2) \\
=& \lambda\mathbf{f}'G^{-1}(\lambda G^{-1}+O(\lambda^2))\mathbf{f} + \sigma^2 tr((I-H_\lambda)^2) + O(\lambda^2) \\
=& \sigma^2 tr((I-H_\lambda)) \\
=& \sigma^2(n - 2tr(H_\lambda) + tr(H_\lambda^2)) + O(\lambda^2) \\
=& \sigma^2(n - 2\cdot\text{dof}(\hat{f}) + tr(H_\lambda^2)) + O(\lambda^2)
\end{aligned}$$

In this way, if we use $\hat{\sigma}^2 = \dfrac{1}{n-2\cdot\text{dof}(\hat{f})+tr(H_\lambda^2)}\|\hat{Y}-Y\|_2^2$, then we have

$$\mathbb{E}\left[\hat{\sigma}^2\right] = \frac{1}{n-2\cdot\text{dof}(\hat{f})+tr(H_\lambda^2)}\mathbb{E}\left[\|\hat{Y}-Y\|_2^2\right] = \sigma^2 + \frac{O(\lambda^2)}{n-2\cdot\text{dof}(\hat{f})+tr(H_\lambda^2)}$$

Question 3.3

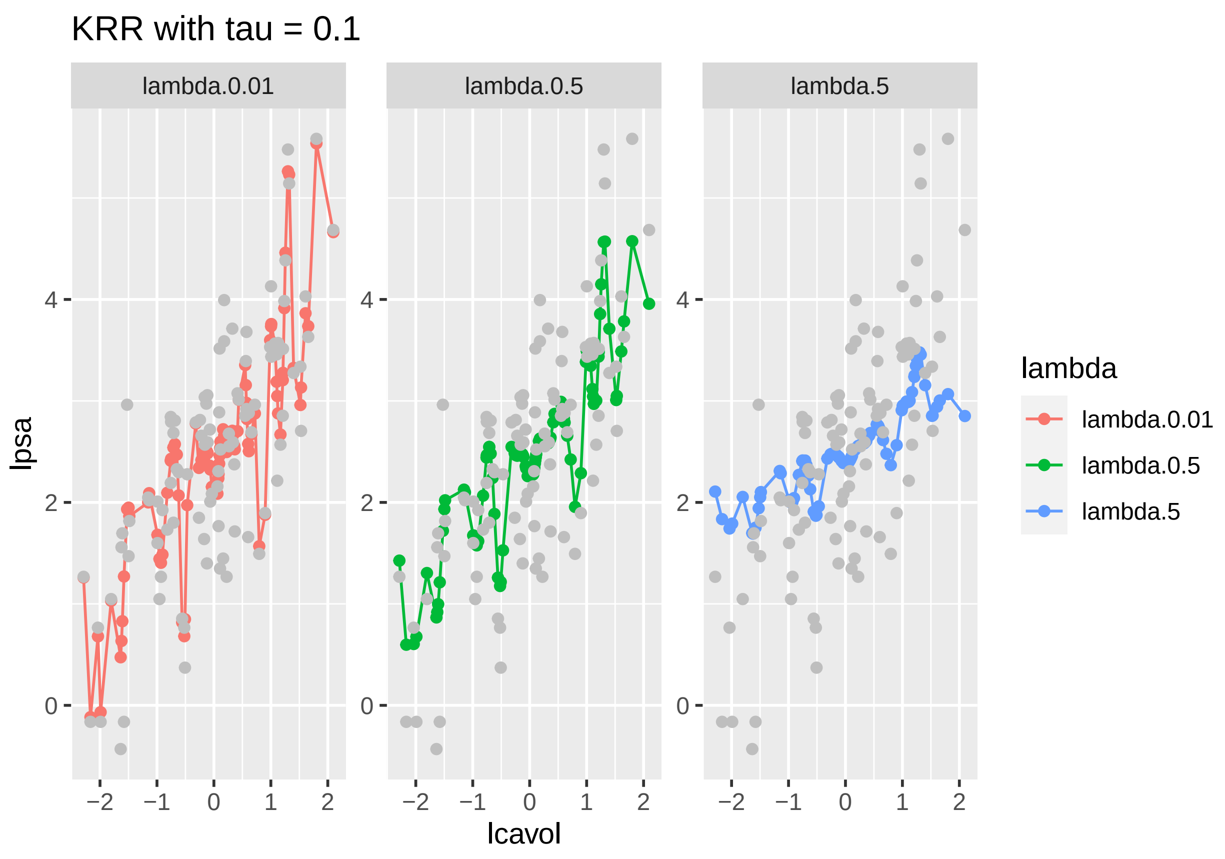
(a)(b)

```r
library('tidyverse')
lprostate <- read.csv("lprostate.dat", sep = "\t")

predictKRR <- function(X,Z,alpha,tau,offset){
    ## X: n by d matrix, training data
    ## Z: m by d matrix, data to be predicted
    ## alpha: n by 1 vector, KRR estimator
    ## tau: scalar in RBF function exp(-||x-y||^2/2tau^2)
    ## offset: scalar, intercept
    dist_xzfull <- dist(rbind(X, Z), method = 'euclidean')
    dist_xz <- as.matrix(dist_xzfull[ (nrow(X)+1):nrow(dist_xzfull),1:nrow(X)])
    K <- exp(-dist_xz^2 / (2 * tau^2))
    return(K %*% alpha + offset)
}
fitKRR <- function(X,y,lambda,tau){
    ## X: n by d matrix, training data
    ## y: n by 1 vector, training response
    ## lambda: scalar, regularization parameter
    ## tau: scalar in RBF function exp(-||x-y||^2/2tau^2)
    n <- nrow(X)
    centered_y <- y - mean(y)
    K <- exp(-as.matrix(dist(X, method = 'euclidean'))^2 / (2 * tau^2))
    alpha <- solve(K + lambda * diag(n)) %*% centered_y
    yMean <- K %*% alpha + mean(y)
    return( list(alpha = alpha, yMean = yMean) )
}
```

(c)

From the plot we can see that the model with $\tau = 0.1$ seems to be too ‘local’, we observe that the model with $\lambda = 0.01$ could fit the overall trend but has significant overfitting, while the model with $\lambda = 5$ is too ‘smooth’ and cannot capture the trend.

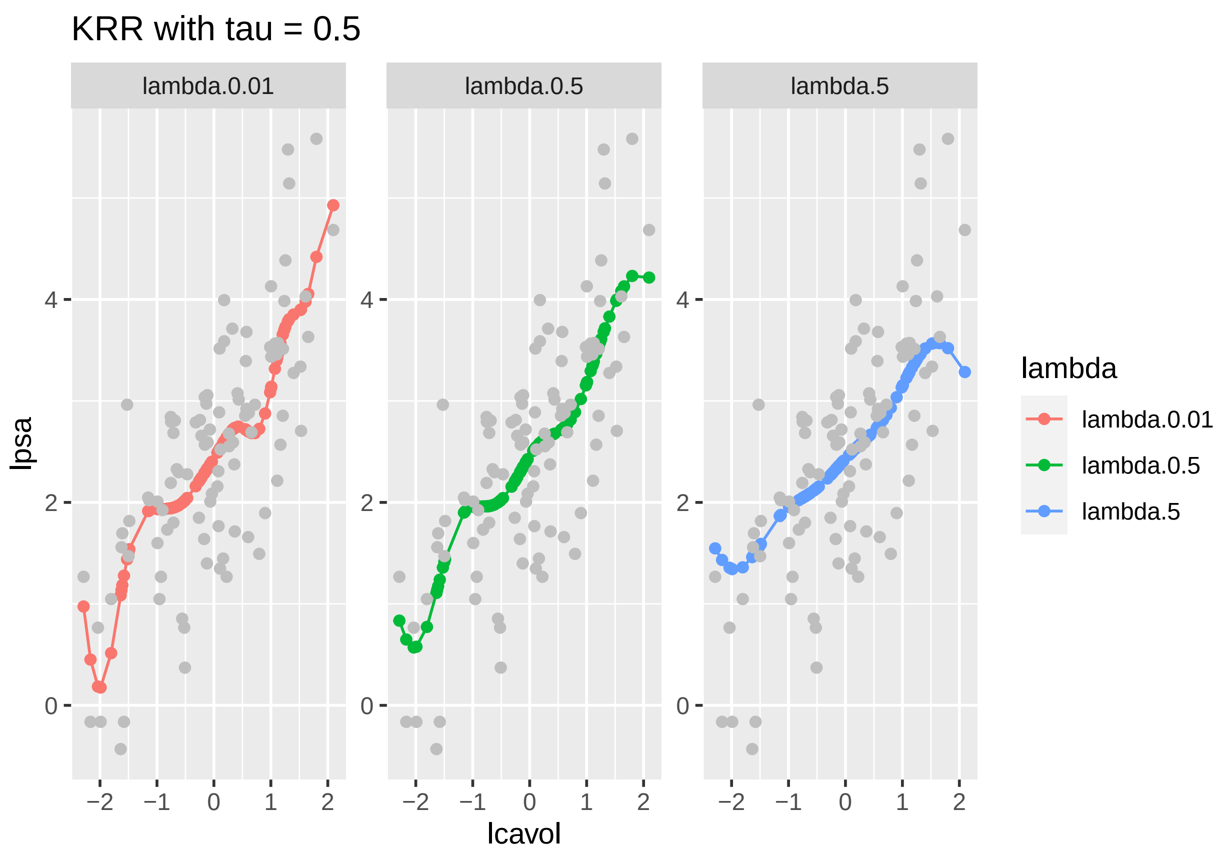
```
# fit KRR on lprostate data with tau = 0.1
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.1
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.1 <- data.frame(x = X, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.5 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.1[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.1 %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, color
```



(d)

Plot for $\tau = 0.5$ seems to be quite good, models with $\lambda = 0.01$ and $\lambda = 0.5$ could fit quite well, being smooth enough and capturing the trend.

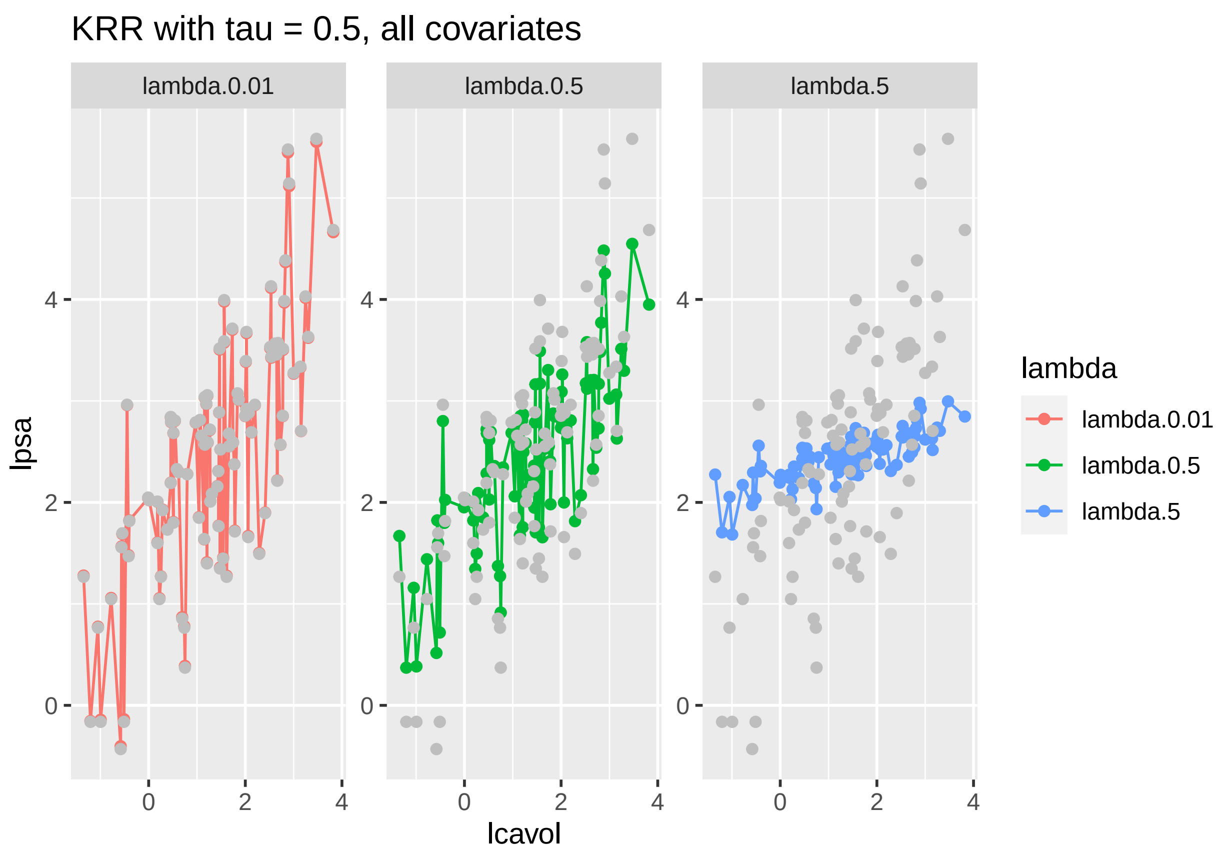
```r
# fit KRR on lprostate data with tau = 0.5
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.5 <- data.frame(x = X, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.5 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.5 %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, color
```



(e)

Plot for $\tau = 0.5$ with all covariates involve. This time the prediction seems to be less satisfactory, the model with $\lambda = 0.01$ is too 'local' and the model with $\lambda = 5$ is too close to mean, perhaps $\lambda = 0.5$ model is better, because we can't see the plotting projection on other dimensions.

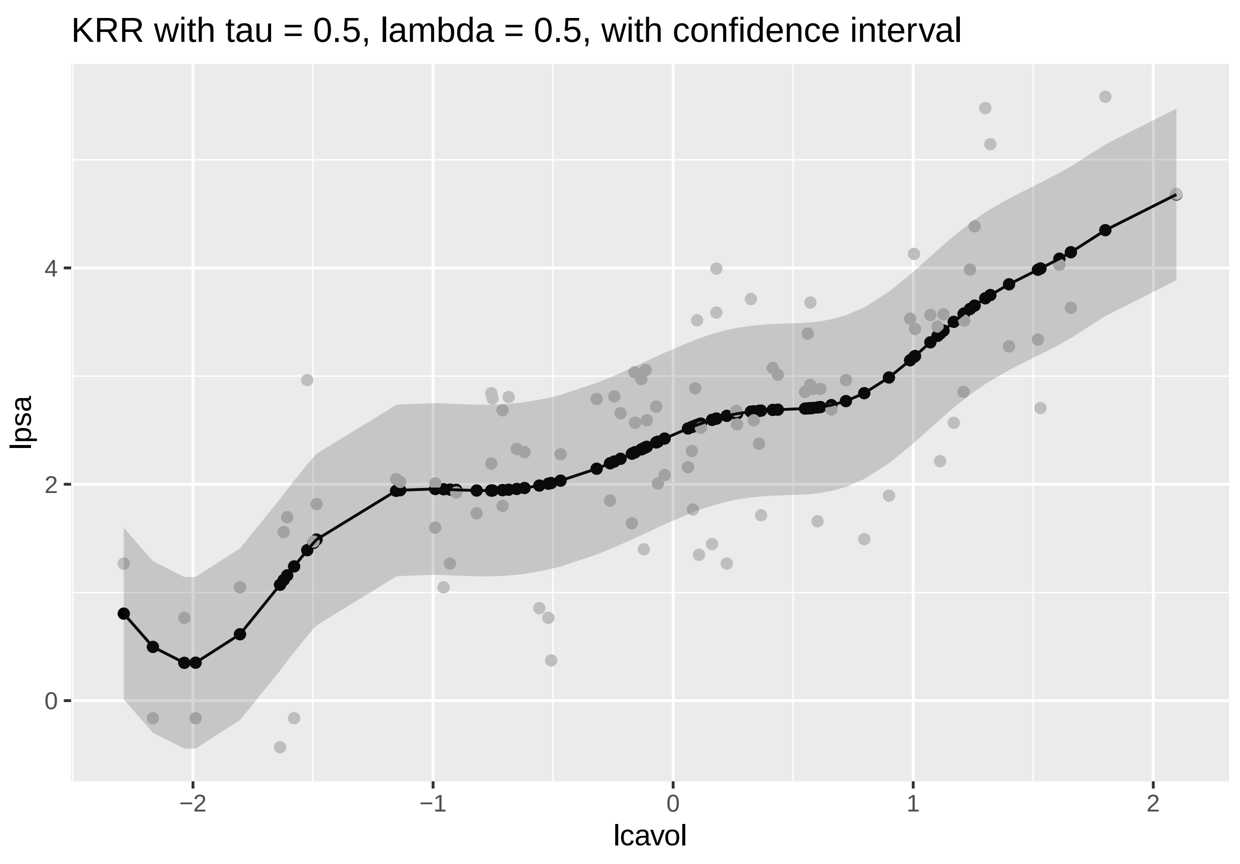
```r
# fit KRR on lprostate data with tau = 0.5
X <- lprostate[, c('lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason', 'pgg45')] %>% scale()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.5_all <- data.frame(x = lprostate$lcavol, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5_all[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.5_all %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, co
```



(f)

```
# fit KRR on lprostate data with tau = 0.5, lambda = 0.1 to get variance estimator
X <- lprostate[, c('lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason', 'pgg45')] %>% scale()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- 0.1
KRR_tau_0.5_all_getvar <- data.frame(x = lprostate$lcavol, y = y, lambda.0.1 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5_all_getvar[, paste0('lambda.', lambda)] <- fit$yMean
}
## get variance estimator
yhat <- KRR_tau_0.5_all_getvar$lambda.0.1
n <- length(yhat)
K <- exp(-as.matrix(dist(X, method = 'euclidean'))^2 / (2 * tau^2))
H <- K %*% solve(K + lambda * diag(n))
sigma2_hat <- 1 / (n - 2 * sum(diag(H)) + sum(diag( t(H) %*% H ))) * (sum((y - yhat)^2) )
```

```
# fit KRR on lprostate data with tau = 0.5, lambda = 0.1
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- 0.1
KRR_tau_0.5band <- data.frame(x = X, y = y, lambda.0.1 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5band[, paste0('lambda.', lambda)] <- fit$yMean
}
KRR_tau_0.5band$upper <- KRR_tau_0.5band$lambda.0.1 + sqrt(sigma2_hat)
KRR_tau_0.5band$lower <- KRR_tau_0.5band$lambda.0.1 - sqrt(sigma2_hat)
# plot showing y and yhat against x, with confidence interval using variance estimator

KRR_tau_0.5band %>% ggplot(aes(x = x, y = lambda.0.1)) + geom_point() + geom_point(aes(x = x, y = y),
```



Question 3.4

(a)

We have

$$\hat{\beta}_\lambda := \arg\min_b \|Xb - y\|_2^2 + b'V\Lambda V'b := \arg\min_b \mathcal{L}(b)$$

the minimizer satisfy

$$\begin{aligned} 0 = \frac{\partial \mathcal{L}}{\partial b} =& 2X'(Xb - y) + 2V\Lambda V'b \\ \Rightarrow \hat{\beta}_\lambda =& (X'X + V\Lambda V')^{-1}X'y = (V\Gamma^2 V' + V\Lambda V')^{-1}V\Gamma U'y = V(\Gamma^2 + \Lambda)^{-1}\Gamma U'y = V\Gamma(\Gamma^2 + \Lambda)^{-1}U'y \end{aligned}$$

similarly, we have

$$H_\lambda = X(X'X + V\Lambda V')^{-1}X' = U\Gamma(\Gamma^2 + \Lambda)^{-1}\Gamma U' = U\Gamma^2(\Gamma^2 + \Lambda)^{-1}U'$$

(b)

Note that with model $Y_i = f(x_i) + \varepsilon_i$, $\varepsilon_i \sim (0, \sigma^2)$ we have

$$\mathbb{E}\left[(\hat{Y}_\lambda - f(x)) \cdot (f(x) - Y)\right] = cov(\hat{Y}_\lambda, Y)$$

thus

$$\begin{aligned} \frac{1}{n}\mathbb{E}[\text{RSS}] =& \frac{1}{n}\mathbb{E}\left[\|\hat{Y}_\lambda - Y\|_2^2\right] \\ =& \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[(\hat{Y}_{\lambda,i} - Y_i)^2\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i) + f(x_i) - Y_i)^2\right] \\ =& \frac{1}{n}\left[\mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i))^2\right] + \mathbb{E}\left[(f(x_i) - Y_i)^2\right] - 2\mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i)) \cdot (f(x_i) - Y_i)\right]\right] \\ =& R_{\text{in}}(\hat{\beta}_\lambda) + \sigma^2 - \frac{2}{n}\sum_{i=1}^n cov(\hat{Y}_{\lambda,i}, Y_i) \end{aligned}$$

in which note that

$$\begin{aligned} cov(\hat{Y}_\lambda, Y) =& \mathbb{E}\left[(\hat{Y}_\lambda - f(x)) \cdot (f(x) - Y)\right] \\ =& \mathbb{E}\left[\hat{Y}_\lambda \cdot (f(x) - Y)\right] \\ =& \mathbb{E}\left[(H_\lambda Y) \cdot (f(x) - Y)\right] \\ =& \mathbb{E}\left[\varepsilon' H_\lambda \varepsilon\right] \\ =& \sigma^2 tr(H_\lambda) \end{aligned}$$

which gives

$$\frac{1}{n}\mathbb{E}\left[\|\hat{Y}_\lambda - Y\|_2^2\right] + \frac{2\sigma^2}{n}tr(H_\lambda) = R_{\text{in}}(\hat{\beta}_\lambda) + \sigma^2$$

(c)

We have

$$\begin{aligned}
r(\lambda) =& \frac{1}{n}\|\hat{Y}_\lambda - Y\|_2^2 + \frac{2\hat{\sigma}^2}{n} tr(H_\lambda) \\
=& \frac{1}{n} Y'(I-H_\lambda)'(I-H_\lambda)Y + \frac{2\hat{\sigma}^2}{n} tr(H_\lambda) \\
=& \frac{1}{n} Y'(I - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')'(I - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')Y + \frac{2\hat{\sigma}^2}{n} tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U') \\
=& \frac{1}{n} Y'(UU' + U_\perp U_\perp' - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')^2 Y + \frac{2\hat{\sigma}^2}{n} tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U') \\
=& \frac{1}{n} Y'U(I - \Gamma^2(\Gamma^2+\Lambda)^{-1})U'Y + \frac{1}{n} Y'U_\perp U_\perp' Y + \frac{2\hat{\sigma}^2}{n} tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U') \\
=& \frac{1}{n}\sum_{j=1}^{d}\left[\frac{\lambda_j^2}{(\gamma_j^2+\lambda_j)^2}(u_j'Y)^2 + 2\hat{\sigma}^2\frac{\gamma_j^2}{\gamma_j^2+\lambda_j}\right] + \frac{1}{n}\|U_\perp' Y\|_2^2
\end{aligned}$$

and take derivative $\frac{\partial}{\partial \lambda_j}$ to obtain

$$\begin{aligned}
\frac{n}{2}\frac{\partial}{\partial\lambda_j} r(\lambda) =& \frac{1}{2}\frac{\partial}{\partial\lambda_j}\sum_{j=1}^{d}\left[\frac{\lambda_j^2}{(\gamma_j^2+\lambda_j)^2}(u_j'Y)^2 + 2\hat{\sigma}^2\frac{\gamma_j^2}{\gamma_j^2+\lambda_j}\right] \\
=& \frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot\frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2}\cdot(u_j'Y)^2 - \hat{\sigma}^2\frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2} \\
=& \frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2}\cdot\left[\frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot(u_j'Y)^2 - \hat{\sigma}^2\right]
\end{aligned}$$

(d)

First for $\lambda \geq 0$, we have

$$\frac{\partial}{\partial\lambda_j} r(\lambda) \propto \frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot(u_j'Y)^2 - \hat{\sigma}^2 < (u_j'Y)^2 - \hat{\sigma}^2 < 0, \quad \text{iff } \hat{\sigma}^2 \geq (u_j'Y)^2$$

and for $\hat{\sigma}^2 < (u_j'Y)^2$, we have

$$0 = \frac{\partial}{\partial\lambda_j} r(\lambda) \propto \frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot(u_j'Y)^2 - \hat{\sigma}^2 \Rightarrow \lambda_j^* = \frac{\hat{\sigma}^2\gamma_j^2}{(u_j'Y)^2 - \hat{\sigma}^2}$$

to summarize, we have

$$\lambda^* = \begin{cases} +\infty, & \text{if } \hat{\sigma}^2 \geq (u_j'Y)^2 \\ \dfrac{\hat{\sigma}^2\gamma_j^2}{(u_j'Y)^2 - \hat{\sigma}^2}, & \text{if } \hat{\sigma}^2 < (u_j'Y)^2 \end{cases}$$

Intuitively, if $\hat{\sigma}^2$ is too large, then any penalization will not be enough to reduce in-sample risk, because the signal of the fitted model is dominated by noise. On the other hand, if $\hat{\sigma}^2$ is small, then we can use a proper penalization to reduce in-sample risk, i.e. increase signal-to-noise ratio.

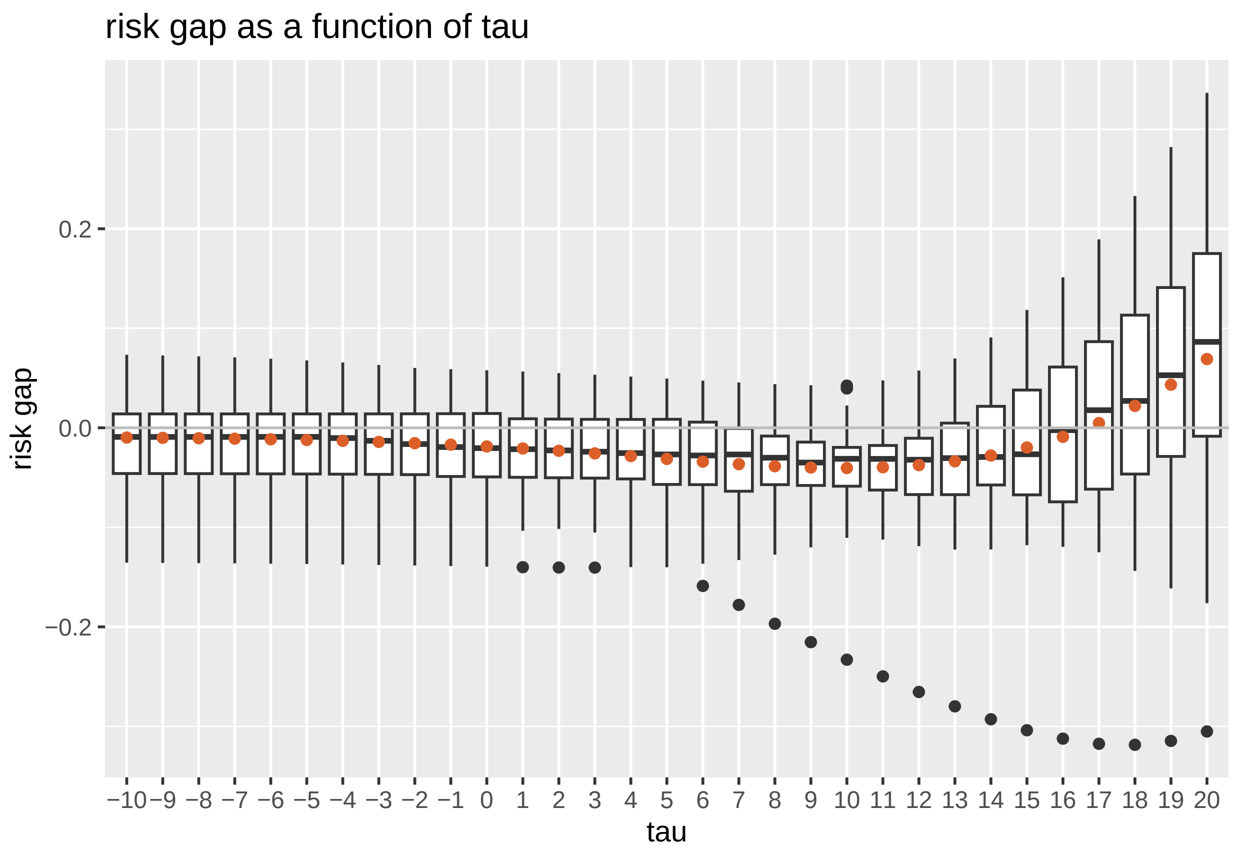
(e)

Simulation on `lprostate.dat`

```r
## ridge-prostatde-dataprep.r
library(tidyverse);
prostate.data = tibble(read.csv("lprostate.dat", sep = "\t", header=TRUE));

construct.train.and.test <- function(dataset, response = "", split.prop = .6) {
    if (all(names(dataset) != response)) {
        stop("Dataset does not contain the given response")
    }
    total.n = nrow(dataset)
    permutation = sample(1:total.n, total.n)
    train.size = as.integer(split.prop * total.n)
    train.data = dataset[permutation[1:train.size], ]
    test.data = dataset[permutation[(train.size+1):total.n], ]

    X.train = train.data[, !names(dataset) %in% c(response)]
    y.train = train.data[, response]
    X.test = test.data[, !names(dataset) %in% c(response)]
    y.test = test.data[, response]

    y.train = scale(y.train, scale = F)
    X.train = scale(X.train)
    y.test = y.test - attr(y.train, "scaled:center")
    X.test = sapply(1:dim(X.test)[2],
                    function(i) {
                        ((X.test[,i] - attr(X.train, "scaled:center")[i])
                            / attr(X.train, "scaled:scale")[i])
                    })

    train.data[, !names(dataset) %in% c(response)] = X.train
    train.data[, response] = y.train
    test.data[, !names(dataset) %in% c(response)] = X.test
    test.data[, response] = y.test

    return(list(train = tibble(train.data),
                test = tibble(test.data)))
}

## Remove columns and save our target name
```

```r
prostate.data = prostate.data[! names(prostate.data) == "row.names"];

## Simulation
tau <- 10^((-10:20)/10)
N <- 25

risk.gap.df <- matrix(NA, nrow = length(tau), ncol = N)
for(iter in 1:N){
    # data prep
    set.seed(iter)
    train.test <- construct.train.and.test(prostate.data, response = "lpsa", split.prop = .6)
    X.train <- train.test$train %>% select(-lpsa) %>% as.matrix()
    y.train <- train.test$train %>% select(lpsa) %>% as.matrix()
    X.test <- train.test$test %>% select(-lpsa) %>% as.matrix()
    y.test <- train.test$test %>% select(lpsa) %>% as.matrix()

    # first compoute \hat{\sigma}^2 using OLS
    beta.hat.OLS <- solve(t(X.train) %*% X.train) %*% t(X.train) %*% y.train
    y.hat.OLS <- X.train %*% beta.hat.OLS
    sigma2.hat.OLS <- 1 / (nrow(X.train) - ncol(X.train)) * sum((y.train - y.hat.OLS)^2)

    # SVD of X.train
    svd.X.train <- svd(X.train)
    U <- svd.X.train$u
    V <- svd.X.train$v
    Gamma_diag <- svd.X.train$d

    # then compute the optimal lambda^*
    lambda <- ifelse(sigma2.hat.OLS >= (t(U) %*% y.train)^2, Inf, sigma2.hat.OLS * Gamma_diag^2 / ((t

    # compute the optimal ridge estimator \hat{\beta}_\lambda^* and compute held-out risk on test set
    beta.hat.ridge.optimal <- V %*% diag(Gamma_diag / (Gamma_diag^2 + lambda)) %*% t(U) %*% y.train
    held.out.risk.optimal <- 1 / nrow(X.test) * sum((y.test - X.test %*% beta.hat.ridge.optimal)^2)
    # compute the ridge estimator \hat{\beta}_\lambda^* for each \tau
    beta.hat.ridge <- matrix(NA, nrow = ncol(X.train), ncol = length(tau))
    held.out.risk <- rep(NA, length(tau))
    for(i in 1:length(tau)){
        beta.hat.ridge[,i] <- solve(t(X.train) %*% X.train + tau[i] * diag(ncol(X.train))) %*% t(X.tr
        held.out.risk[i] <- 1 / nrow(X.test) * sum((y.test - X.test %*% beta.hat.ridge[,i])^2)
    }
    # compute the risk gap
    risk.gap <- held.out.risk - held.out.risk.optimal
    risk.gap.df[, iter] <- risk.gap
}
risk.gap.df <- data.frame(t(risk.gap.df))
names(risk.gap.df) <- -10:20

# make a plot of the risk gap as a function of \tau, sort the \tau in increasing order

plotdf <- risk.gap.df %>% gather(key = "tau", value = "risk.gap") %>% mutate(tau = as.numeric(tau)) %
# boxplot for distribution and line-point plot for mean-value
plotdf %>% ggplot(aes(x = as.factor(tau), y = risk.gap)) + geom_boxplot() + geom_point(stat = "summar
```

```
## No summary function supplied, defaulting to `mean_se()`
```



We can see that that:

- for large τ value, the risk gap $\hat{r}_\tau - \hat{r}_*$ is significantly increasing and being larger than 0. Which means that in that case the 'optimal' ridge estimator $\hat{\beta}_{\lambda^*}$ out perform the regular ridge regression method (in the sense to minimize expected risk).
- But for small τ, such improvement is not significant, and the risk gap could even be negative, which means that the 'optimal' ridge estimator $\hat{\beta}_{\lambda^*}$ is slightly worse than the regular ridge regression method.

Question 4.1

(a)

Here note that H and H_k just differ by a rank-1 matrix

$$H_k = H - \frac{1}{n}\ell''(\hat{\varepsilon}_i)x_k x_k'$$

so using SMW formula we have

$$\begin{aligned}
H_k^{-1} =& (H - \frac{1}{n}\ell''(\hat{\varepsilon}_k)x_k x_k')^{-1} \\
\overset{\text{SMW}}{=}& H^{-1} + \frac{\frac{1}{n}\ell''(\hat{\varepsilon}_k)H^{-1}x_k x_k' H^{-1}}{1 - \frac{1}{n}\ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k} \\
=& (I + \frac{\ell''(\hat{\varepsilon}_k)H^{-1}x_k x_k'}{n - \ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k})H^{-1} \\
\Rightarrow H_k^{-1}g_k =& (I + \frac{\ell''(\hat{\varepsilon}_k)H^{-1}x_k x_k'}{n - \ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k})H^{-1}g_k
\end{aligned}$$

(b)

Using the above expreesion we have

$$\hat{\Delta}_k = -H_k^{-1}g_k = -(I + \frac{\ell''(\hat{\varepsilon}_k)H^{-1}x_k x_k'}{n - \ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k})H^{-1}g_k$$

then

$$\begin{aligned}
\hat{y}_{-k} =& x_k'(\hat{\beta} + \hat{\Delta}_k) = x_k'\hat{\beta} - x_k' H_k^{-1}g_k \\
\hat{\varepsilon}_{-k} =& y_k - \hat{y}_{-k} = y_k - x_k'\hat{\beta} + x_k' H_k^{-1}g_k \\
=& \hat{\varepsilon}_k + x_k' H_k^{-1}g_k
\end{aligned}$$

So to compute $\{\hat{y}_{-k}\}_{k=1}^n$, we can, for each k: first compute $x_k' H^{-1} x_k' := \xi_k$, which takes time $O(d^2)$, then compute

$$\begin{aligned}
x_k' H_k^{-1}g_k =& x_k'(I + \frac{\ell''(\hat{\varepsilon}_k)H^{-1}x_k x_k'}{n - \ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k})H^{-1}\frac{1}{n}\ell'(\hat{\varepsilon}_k)x_k \\
=& \frac{1}{n}\ell'(\hat{\varepsilon}_k)[x_k' H^{-1}x_k + \frac{\ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k x_k' H^{-1}x_k}{n - \ell''(\hat{\varepsilon}_k)x_k' H^{-1}x_k}] \\
=& \frac{1}{n}\ell'(\hat{\varepsilon}_k)[\xi_k + \frac{\ell''(\hat{\varepsilon}_k)\xi_k^2}{n - \ell''(\hat{\varepsilon}_k)\xi_k}]
\end{aligned}$$

which takes computation time $O(1)$ (all scalar computation involved). And computation of $\hat{y}_k = x_k'\hat{\beta}$ costs $O(n)$. So the total computation time is $O(nd^2 + n) = O(nd^2)$.

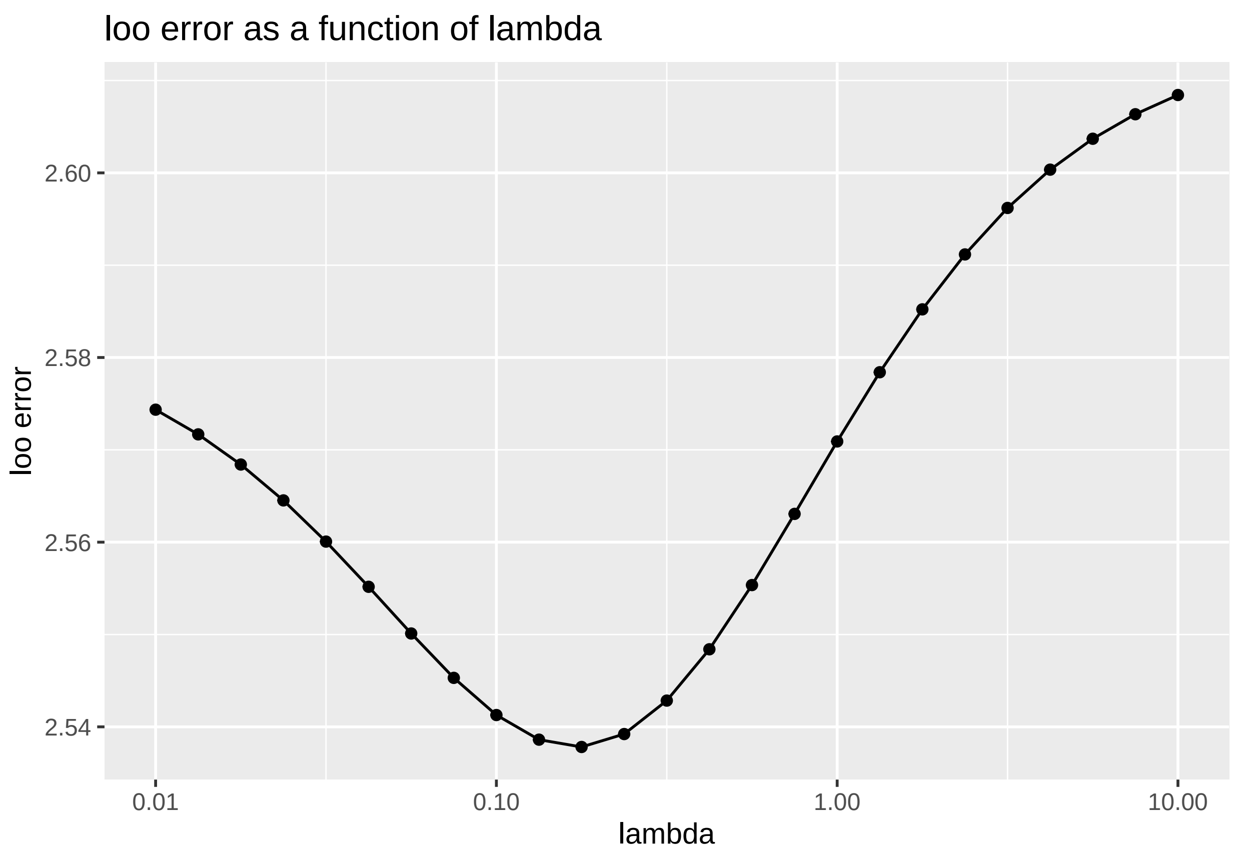
(c)

```
library(CVXR);
```

```
##
## 载入程辑包: 'CVXR'

## The following object is masked from 'package:dplyr':
##
##     id

## The following object is masked from 'package:purrr':
##
##     is_vector

## The following object is masked from 'package:stats':
##
##     power
```

```
library(tidyverse);
## generate.data(n, d, num.outliers)
##
## Generates training data for the robust regression problem. The
## number of outliers num.outliers defaults to 10.
generate.data <- function(nn = 100, dd = 25, num.outliers = 10) {
    X.train = matrix(rnorm(nn * dd, 0, 1), nn, dd);
    X.test = matrix(rnorm(nn * dd, 0, 1), nn, dd);

    beta.star = rnorm(dd, 0, 1);
    beta.star = beta.star / sqrt(sum(beta.star^2));  # Makes X * beta ~ N(0, 1)

    train.noise = rnorm(nn, 0, 1);
    train.outliers = sample.int(nn, num.outliers);
    test.noise = rnorm(nn, 0, 1);
    test.outliers = sample.int(nn, num.outliers);

    ## Generate outlier measurements

    y.train = X.train %*% beta.star + train.noise;
    signs = sign(rnorm(num.outliers)) # Symmetric random outliers
    y.train[train.outliers] = 5 * signs * rnorm(num.outliers)^4
    y.test = X.test %*% beta.star + test.noise;
    signs = sign(rnorm(num.outliers)) # Symmetric noise
    y.test[test.outliers] = 5 * signs * rnorm(num.outliers)^4
    return(list("X.train" = X.train, "y.train" = y.train,
                "X.test" = X.test, "y.test" = y.test))
}

## Function to fit the best model to this data using the log(1 + exp)
## loss. To use this function, simply call
##
## minimize.robust.ridge(X.train, y.train, lambda),
##
## which will return a vector minimizing
##
##  (1/n) sum_{i = 1}^n l(y - x_i' * b) + (lambda/2) * ||b||^2
##
## where ||.|| denotes the l2 norm.

minimize.robust.ridge <- function(X, y, lambda) {
    nn = dim(X)[1]
    dd = dim(X)[2]
    beta <- Variable(dd)
    obj <- ((1/nn) * sum(logistic(X %*% beta - y))
        + (1/nn) * sum(logistic(y - X %*% beta))
        + (lambda/2) * sum_squares(beta))
    problem <- Problem(Minimize(obj))
    result <- solve(problem)
    return(result$getValue(beta))
}
# we use loss function ell(x)=log(1+exp(x))+log(1+exp(-x))
# first derivative
ell <- function(x, threshold = 10){ # for numerical stability put some threshold
    return(ifelse(x > threshold, x+2*log(1+exp(-x)), ifelse(x < -threshold, -x+2*log(1+exp(x)), log(1
}
ellprime <- function(x){
    return((exp(x)-1)/(exp(x)+1))
}
# second derivative
ellprimeprime <- function(x){
    return(2*exp(x)/(exp(x)+1)^2)
}

set.seed(2)
dat <- generate.data(100, 25, 10)
lambdas <- exp(seq(log(0.01), log(10), length.out = 25))
```

```
n <- nrow(dat$X.train)
d <- ncol(dat$X.train)
loo.errors <- rep(NA, length(lambdas))
for(lambda in lambdas){
    beta.hat <- minimize.robust.ridge(dat$X.train, dat$y.train, lambda)
    # compute residuals
    eps.hat <- dat$y.train - dat$X.train %*% beta.hat
    # as described, compute $H=1/n * \sum \ell''(\hat{\varepsilon }_i)x_ix_i' + \lambda I$
    H <- 1 / n * t(dat$X.train) %*% diag(as.vector(ellprimeprime(eps.hat))) %*% dat$X.train  + lambda
    Hinverse <- solve(H)
    # compute \xi
    xi <- rep(NA, n)
    for(k in 1:n){
        xi[k] <- t(dat$X.train[k,]) %*% Hinverse %*% dat$X.train[k,]
    }
    xHg <- 1/n*ellprime(eps.hat) * (xi + ellprimeprime(eps.hat) * xi^2 / (n - ellprimeprime(eps.hat)
    # compute \{\hat{\varepsilon}_{-k}\}
    loo.residuals <- eps.hat + xHg
    loo.error <- 1/n * sum(ell(loo.residuals))
    loo.errors[which(lambda == lambdas)] <- loo.error
}

# plotting, lambda on a log scale
plotdf <- data.frame(lambda = lambdas, loo.error = loo.errors)
plotdf %>% ggplot(aes(x = (lambda), y = loo.error)) + geom_point() + geom_line() + scale_x_log10() +
```



(d)

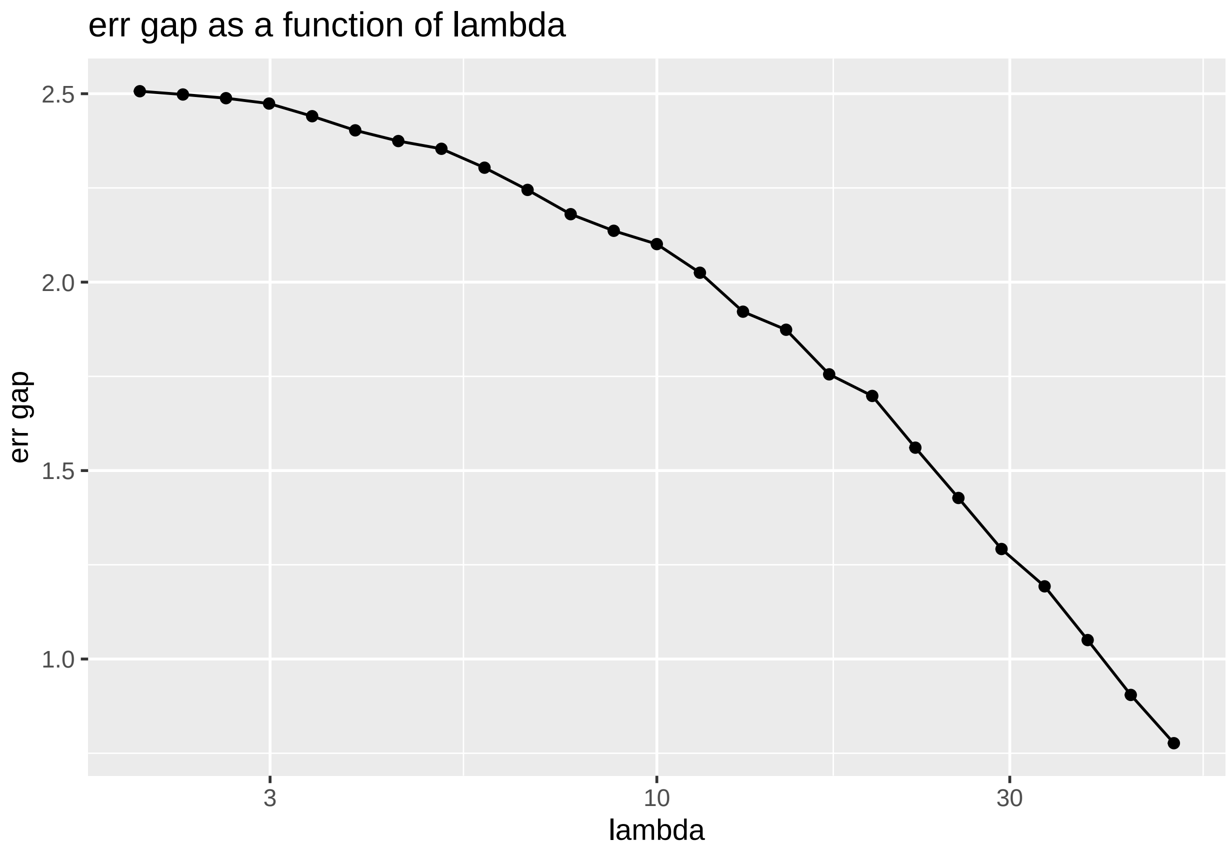
Using the result above, we choose $\lambda \approx 0.178$ for now, to do the following steps.

```
library(glmnet)
```

```
## 载入需要的程辑包：Matrix

##
## 载入程辑包：'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```
# use the best lambda to fir the robust ridge estimator
best_lambda.robust <- lambdas[which.min(loo.errors)]
beta.hat.robust <- minimize.robust.ridge(dat$X.train, dat$y.train, best_lambda.robust)
err_median_robust <- median(abs(dat$X.test %*% beta.hat.robust - dat$y.test))

# get OLS ridge estimator with l2 loss
```

```r
lambdas.ols <- exp(seq(log(2), log(50), length.out = 25))
err_gap <- vector(length = length(lambdas.ols))
for(lambda in lambdas.ols){
    beta.hat.OLS <- solve(t(dat$X.train) %*% dat$X.train + lambda * diag(d)) %*% t(dat$X.train) %*% d
    err_median_OLS <- median(abs(dat$X.test %*% beta.hat.OLS - dat$y.test))
    err_gap[which(lambda == lambdas.ols)] <- err_median_OLS - err_median_robust
}

# plotting, lambda on a log scale
plotdf <- data.frame(lambda = lambdas.ols, err_gap = err_gap)
plotdf %>% ggplot(aes(x = (lambda), y = err_gap)) + geom_point() + geom_line() + scale_x_log10() + la
```



The absolute error gap is monotonely decreasing, which probably means that the influence of outlier is significant to OLS ridge estimator. Larger λ in OLS ridge can somehow fit this problem, but, we notice that the gap always has $\mathrm{ERR}_{\mathrm{test}}^{\mathrm{ls}(\lambda)} - \mathrm{ERR}_{\mathrm{test}} > 0$, which means that the robust ridge estimator always outperforms the OLS ridge estimator (in the sense minimizing expected risk, calculated as the median absolute prediction error on test set).

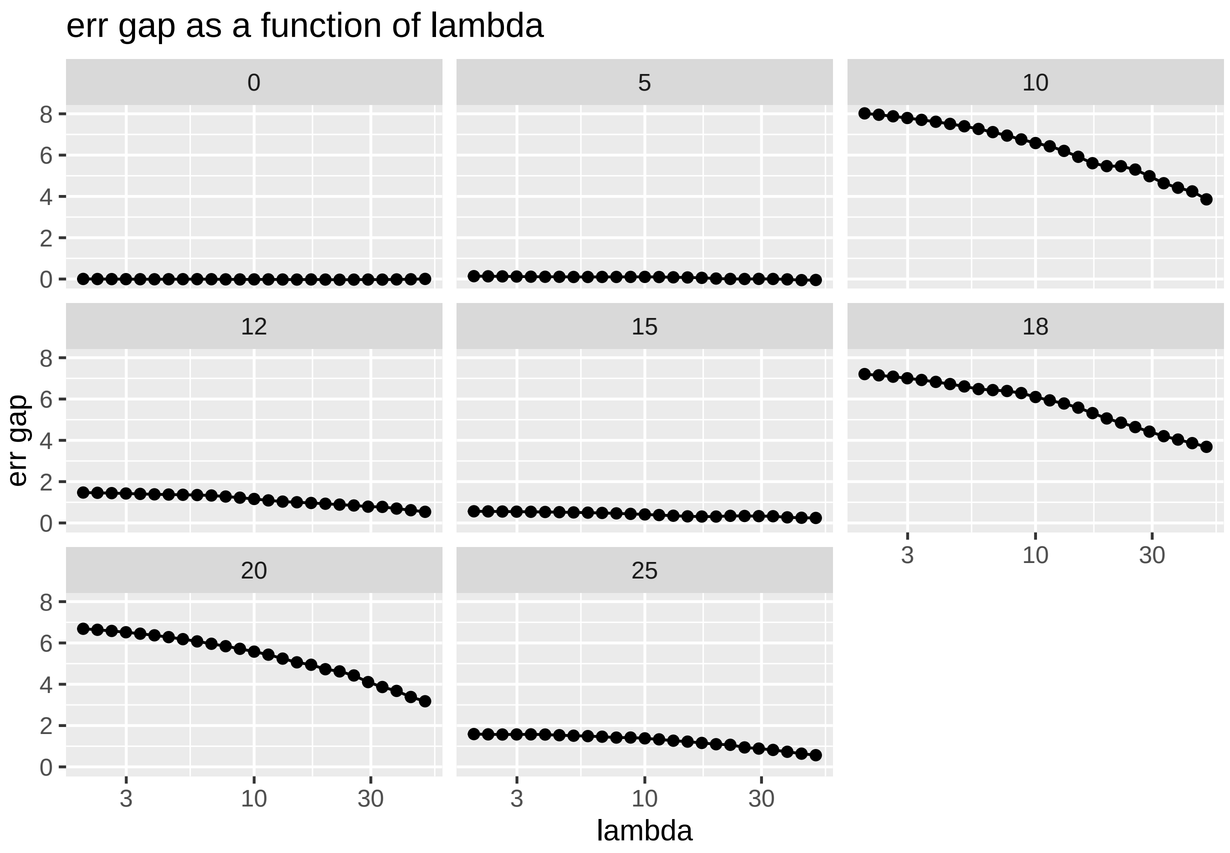
(e)

repeat for different number of outliers

```
num_outliers <- c(0,5,10,12,15,18,20,25)
lambdas <- exp(seq(log(0.01), log(10), length.out = 25))
lambdas.ols <- exp(seq(log(2), log(50), length.out = 25))

err_gap_mat <- matrix(NA, nrow = length(lambdas.ols), ncol = length(num_outliers))
for(num_outlier in num_outliers){
    set.seed(num_outlier)
    dat <- generate.data(100, 25, num_outlier)
    n <- nrow(dat$X.train)
    d <- ncol(dat$X.train)
    loo.errors <- rep(NA, length(lambdas))
    for(lambda in lambdas){
        beta.hat <- minimize.robust.ridge(dat$X.train, dat$y.train, lambda)
        # compute residuals
        eps.hat <- dat$y.train - dat$X.train %*% beta.hat
        # as described, compute $H=1/n * \sum \ell''(\hat{\varepsilon }_i)x_ix_i' + \lambda I$
        H <- 1 / n * t(dat$X.train) %*% diag(as.vector(ellprimeprime(eps.hat))) %*% dat$X.train  + la
        Hinverse <- solve(H)
        # compute \xi
        xi <- rep(NA, n)
        for(k in 1:n){
            xi[k] <- t(dat$X.train[k,]) %*% Hinverse %*% dat$X.train[k,]
        }
        xHg <- 1/n*ellprime(eps.hat) * (xi + ellprimeprime(eps.hat) * xi^2 / (n - ellprimeprime(eps.h
        # compute \{\hat{\varepsilon}_{-k}\}
        loo.residuals <- eps.hat + xHg
        loo.error <- 1/n * sum(ell(loo.residuals))
        loo.errors[which(lambda == lambdas)] <- loo.error
    }

    best_lambda.robust <- lambdas[which.min(loo.errors)]
    beta.hat.robust <- minimize.robust.ridge(dat$X.train, dat$y.train, best_lambda.robust)
    err_median_robust <- median(abs(dat$X.test %*% beta.hat.robust - dat$y.test))
    print(cat('err_median_robust for num_outlier = ', num_outlier, ' is ', err_median_robust, sep = '

    # get OLS ridge estimator with l2 loss

    err_gap <- rep(NA, length(lambdas.ols))
    for(lambda in lambdas.ols){
        beta.hat.OLS <- solve(t(dat$X.train) %*% dat$X.train + lambda * diag(d)) %*% t(dat$X.train) %
        err_median_OLS <- median(abs(dat$X.test %*% beta.hat.OLS - dat$y.test))
        err_gap[which(lambda == lambdas.ols)] <- err_median_OLS - err_median_robust
    }

    err_gap_mat[, which(num_outlier == num_outliers)] <- err_gap
}
```

```
## err_median_robust for num_outlier = 0 is 0.819753NULL
## err_median_robust for num_outlier = 5 is 0.8252419NULL
## err_median_robust for num_outlier = 10 is 0.8777035NULL
## err_median_robust for num_outlier = 12 is 1.045174NULL
## err_median_robust for num_outlier = 15 is 0.8204534NULL
## err_median_robust for num_outlier = 18 is 1.048308NULL
## err_median_robust for num_outlier = 20 is 0.9295824NULL
## err_median_robust for num_outlier = 25 is 1.050889NULL
```

```
## plotting, lambda on a log scale, facet with different number of outliers
plotdf <- data.frame(lambda = lambdas.ols, err_gap = err_gap_mat)
names(plotdf) <- c("lambda", num_outliers)
plotdf %>% gather(key = "num_outliers", value = "err_gap", -lambda) %>% mutate(num_outliers = as.nume
```



We can see similar result for different numbers of outlier $\in \{0, 5, 10, 12, 15, 18, 20, 25\}$. Here I tried many times and observe strange result:

- For small number of outlier, two models (using ℓ_2 loss v.s. robust loss) are almost the same. Because intuitively the effect of outlier is not quite severe, so robust fit and OLS fit are similar.
- For number of outlier ≈ 10 we have a significant outperfom of robust regression over OLS regression. Maybe in this case the robust can really fix the problem of outlier
- For number of outlier $12-15$ we have just slight outperform of robust regression over OLS regression. Maybe in this case the robust can fix the problem of outlier, but not quite well.
- Finally for really large number of outlier, the robust regression is again outperforming OLS regression. Maybe because intuitively the effect of outlier is quite severe, so robust fit and OLS fit are both bad, but just OLS fit is influenced so much.

(Well anyway I can't explain this, perhaps I got something wrong)

Question 4.2

(a)

Using the expression of $L_n(b) := \frac{1}{n}\sum_{i=1}^n \ell(y_i - x_i'b)$, we have

$$\nabla^2 L_n(\hat\beta+\Delta) - \nabla^2 L_n(\hat\beta) = \frac{1}{n}\sum_{i=1}^n x_i x_i' \left(\ell''(\hat\varepsilon_i - x_i'\Delta) - \ell''(\hat\varepsilon)\right)$$

thus $\forall \xi \in \mathbb{R}^d$ we have

$$\begin{aligned}\xi'\left[\nabla^2 L_n(\hat\beta+\Delta) - \nabla^2 L_n(\hat\beta) + D_x\|\Delta\|_2\frac{1}{n}X'X\right]\xi =& \xi'\left[\frac{1}{n}\sum_{i=1}^n x_i x_i'\left(\ell''(\hat\varepsilon_i - x_i'\Delta) - \ell''(\hat\varepsilon) + D_x\|\Delta\|_2\right)\right]\xi \\ =& \frac{1}{n}\sum_{i=1}^n (\xi' x_i)^2\left(\ell''(\hat\varepsilon_i - x_i'\Delta) - \ell''(\hat\varepsilon) + D_x\|\Delta\|_2\right) \\ \geq& \frac{1}{n}\sum_{i=1}^n (\xi' x_i)^2\left(D_x\|\Delta\|_2 + \min \ell''(\hat\varepsilon_i - x_i'\Delta) - \ell''(\hat\varepsilon)\right) \\ =& 0\end{aligned}$$

which proves the positive semi-definiteness of $\nabla^2 L_n(\hat\beta+\Delta) - \nabla^2 L_n(\hat\beta) + D_x\|\Delta\|_2\frac{1}{n}X'X$, and gives $\nabla^2 L_n(\hat\beta+\Delta) \succeq \nabla^2 L_n(\hat\beta) - D_x\|\Delta\|_2\frac{1}{n}X'X$.

Further if $H := \nabla^2 L_n(\hat\beta) \succeq 2\lambda I$ then we have

$$\begin{aligned}\nabla^2 L_n(\hat\beta+\Delta) \succeq& \nabla^2 L_n(\hat\beta) - D_x\|\Delta\|_2\frac{1}{n}X'X \\ \succeq& 2\lambda I - D_x\|\Delta\|_2\frac{1}{n}X'X\end{aligned}$$

then we have, if $\|\Delta\|_2 \leq \lambda/(2D_x\left\|n^{-1}X'X\right\|_{\text{op}})$, then noticing that I has the same eigenvector as $X'X$, we have

$$\begin{aligned}\lambda_i(2\lambda I - D_x\|\Delta\|_2\frac{1}{n}X'X) =& 2\lambda - \lambda_i(D_x\|\Delta\|_2\frac{1}{n}X'X) \\ \geq& 2\lambda - \lambda_i(D_x\|\Delta\|_2\frac{1}{n}\|X'X\|_{\text{op}}) \\ \geq& \frac{3}{2}\lambda, \quad \forall i\end{aligned}$$

which means we have

$$\lambda_{\min}(\nabla^2 L_n(\hat{\beta}+\Delta)) \succeq \frac{3}{2}\lambda$$

(b)

Consider $\nabla^2 L_{-k}(b)$, we have

$$\begin{aligned}\nabla^2 L_{-k}(b) =& \nabla^2 \frac{1}{n}\sum_{i\neq k} x_i x_i' \ell''(y_i - x_i' b) \\ =& \nabla^2 L_n(b) - \frac{1}{n}x_k x_k' \ell''(y_k - x_k' b) \\ \Rightarrow \nabla^2 L_{-k}(\hat{\beta}+\Delta) =& \nabla^2 L_n(\hat{\beta}+\Delta) - \frac{1}{n}x_k x_k' \ell''(\hat{\varepsilon}_k - x_k'\Delta)\end{aligned}$$

Then utilizing the result from previous question, when $\|\Delta\|_2 \leq \lambda/(2D_x\|\|n^{-1}X'X\|\|_{\text{op}})$, and when $n \geq 2D_x^2/\lambda$, we have

$$\begin{aligned}\lambda_i(\nabla^2 L_{-k}(\hat{\beta}+\Delta)) \geq& \frac{3}{2}\lambda - \frac{1}{n}\lambda_i(x_k x_k' \ell''(\hat{\varepsilon}_k - x_k'\Delta)) \\ \overset{(i)}{\geq}& \frac{3}{2}\lambda - \frac{1}{n}D_x^2 \\ \geq& \lambda, \quad \forall i\end{aligned}$$

in which (i) uses $\sup|\ell''(t)| \leq 1$ and $\|x_k\|_2 \leq D_x$. Thus we prove

$$\lambda_i(\nabla^2 L_{-k}(\hat{\beta}+\Delta)) \succeq \lambda I$$

(c)

Here definitely our LOO loss function $L_{-k}(b)$ is convex, and we've proven that $\nabla^2 L_{-k}(\hat{\beta}+\Delta) \succeq \lambda I$, thus we can use the lemma introduced to get: $\forall \|\Delta\| \leq c \equiv \lambda/(2D_x\|\|n^{-1}X'X\|\|_{\text{op}})$, we have

$$\begin{aligned}L_{-k}(\hat{\beta}+\Delta) \geq& L_{-k}(\hat{\beta}) + \nabla L_{-k}(\hat{\beta})'\Delta + \frac{\lambda}{2}\min\{c, \|\Delta\|_2\}\|\Delta_2\|_2 \\ =& L_{-k}(\hat{\beta}) + \frac{1}{n}\ell'(\hat{\varepsilon}_k)x_k'\Delta + \frac{\lambda}{2}\min\{c, \|\Delta\|_2\}\|\Delta\|_2 \\ \geq& L_{-k}(\hat{\beta}) - \frac{1}{n}D_x\|\Delta\|_2 + \frac{\lambda}{2}\|\Delta\|_2^2\end{aligned}$$

which is a quadratic function of $\|\Delta\|_2$, we can see that the minimum is achieved at $\|\Delta\|_2 = \dfrac{D_x}{n\lambda}$. Consider the minimizer of $L_{-k}(\hat{\beta}+\Delta)$, we can see that the minimizer Δ_k must have $|\|\Delta_k\|_2 - \dfrac{D_x}{n\lambda}| \leq \dfrac{D_x}{\lambda n}$, otherwise we have $L_{-k}(\widehat{\beta}+\Delta_k) \geq L_{-k}(\widehat{\beta}) - \frac{1}{n}D_x\|\Delta_k\|_2 + \frac{\lambda}{2}\|\Delta_k\|_2^2 \geq L_{-k}(\widehat{\beta})$, which contradicts the fact that Δ_k is the minimizer. Thus we have

$$|\|\Delta_k\|_2 - \frac{D_x}{n\lambda}| \leq \frac{D_x}{\lambda n} \Rightarrow \|\Delta_k\|_2 \leq \frac{2D_x}{\lambda n}$$

in which we notice that $n \geq \frac{4D_x^2 \|n^{-1}X'X\|_{\text{op}}}{\lambda^2}$ automatically guarentees that

$$\|\Delta_k\|_2 \leq \frac{2D_x}{\lambda n} \leq \frac{\lambda}{2D_x \left\|\!\left|n^{-1}X'X\right|\!\right\|_{\text{op}}}$$

which means our $\widehat{\beta}+\Delta_k$ is still in the neighborhood of $\widehat{\beta}$, i.e. $\|\Delta\|_2 \leq c$ such that the quadric bound lemma still holds.

(d)

(I did't find a complete proof of the problem, but I think the following is a good try)

If we express $\Delta_k := (H_k+E)^{-1}(-g_k)$ where E is some 'noise matrix', then we have

$$\begin{aligned}\|\Delta_k - \hat{\Delta}_k\|_2 =& \|((H_k+E)^{-1} - H_k^{-1})(-g_k)\| \\ \leq& \left\|\!\left|(H_k+E)^{-1} - H_k^{-1}\right|\!\right\|_{\text{op}} \|g_k\|_2 \\ \leq& \frac{\left\|\!\left|E\right|\!\right\|_{\text{op}}}{\lambda_{\min}(H_k) - \left\|\!\left|E\right|\!\right\|_{\text{op}}} \frac{D_x}{n} \\ \leq& \frac{\left\|\!\left|E\right|\!\right\|_{\text{op}}}{\lambda - \left\|\!\left|E\right|\!\right\|_{\text{op}}} \frac{D_x}{\lambda}\end{aligned}$$

In this way, **IF** we can prove that $\left\|\!\left|E\right|\!\right\|_{\text{op}} \leq \frac{2D_x^4}{\lambda n}$, then we can prove

$$\begin{aligned}\|\Delta_k - \hat{\Delta}_k\|_2 \leq& \frac{\left\|\!\left|E\right|\!\right\|_{\text{op}}}{\lambda - \left\|\!\left|E\right|\!\right\|_{\text{op}}} \frac{D_x}{\lambda} \\ \leq& \frac{2D_x^4/\lambda n}{\lambda - 2D_x^4/\lambda n} \frac{D_x}{\lambda} \\ =& \frac{2D_x^5}{\lambda^2 - 2D_x^4} \frac{1}{n^2} = O(\frac{1}{n^2})\end{aligned}$$

Question 4.3

(a)

- "$\Rightarrow$": if $Z_{n+1} \leq Z_{(k,n)}$, then we have

  $$\{Z_{n+1}, \{Z_{(\kappa,n)}\}_{\kappa=1}^{k-1}\} = \{Z_{(\kappa,n+1)}\}_{\kappa=1}^{k}$$

  which gives

  $$Z_{n+1} \leq \max\{Z_{(\kappa,n+1)}\}_{\kappa=1}^{k} = Z_{(k,n+1)}$$

- "$\Leftarrow$": if $Z_{n+1} \le Z_{(k,n+1)}$, note that in this case $Z_{n+1} \in \{Z_{(\kappa,n+1)}\}_{\kappa=1}^{n+1}$, we have

$$\left\{Z_{(\kappa,n+1)}\right\}_{\kappa=1}^{k} = \left\{Z_{(\kappa,n)}\right\}_{\kappa=1}^{k-1} \uplus \left\{Z_{n+1}\right\}$$

thus get

$$Z_{n+1} \le Z_{(k,n+1)} \le Z_{(k,n)}$$

(b)

Note that for empirical quantile we have

$$\mathbb{P}\left(Z_n \le \hat{Q}_n(\alpha)\right) \ge \alpha$$

in which $\hat{Q}_n(\alpha) = Z_{(\lceil n\alpha \rceil, n)}$. So now it suffices to show that

$$\begin{aligned}
&\mathbb{P}\left(Z_{n+1} \le \hat{Q}_n((1+\frac{1}{n})\alpha)\right) \ge \alpha \\
\Leftrightarrow &\mathbb{P}\left(Z_{n+1} \le Z_{(\lceil n(1+\frac{1}{n})\alpha \rceil, n)}\right) \ge \alpha \\
\Leftrightarrow &\mathbb{P}\left(Z_{n+1} \le Z_{(\lceil (n+1)\alpha \rceil, n)}\right) \ge \alpha \\
\overset{(i)}{\Leftarrow} &\mathbb{P}\left(Z_{n+1} \le Z_{(\lceil (n+1)\alpha \rceil, n+1)}\right) \ge \alpha \\
\Leftrightarrow &\mathbb{P}\left(Z_{n+1} \le \hat{Q}_{n+1}(\alpha)\right) \ge \alpha
\end{aligned}$$

in which (i) uses the fact that $Z_{(k,n)} \ge Z_{(k,n+1)}$, $\forall k$, so

$$\mathbb{P}\left(Z_{n+1} \le Z_{(\lceil (n+1)\alpha \rceil, n)}\right) \ge \mathbb{P}\left(Z_{n+1} \le Z_{(\lceil (n+1)\alpha \rceil, n+1)}\right) \ge \alpha$$

thus proves the result.

(c)

Replace $\alpha \mapsto (1-\alpha)$ in result off previous question, we have

$$\mathbb{P}\left(Z_{n+1} > \hat{\tau}_n\right) = \mathbb{P}\left(Z_{n+1} > \hat{Q}_n((1+\frac{1}{n})(1-\alpha))\right) < \alpha$$

(d)

Using definition $\hat{C}_{\hat{\tau}_n}(x) = [\hat{f}(x) - \hat{\tau}_n, \hat{f}(x)\hat{\tau}_n]$, we have

$$\begin{aligned}
\mathbb{P}\left(Y_{n+1} \in \hat{C}_{\hat{\tau}_n}(X_{n+1})\right) =& \mathbb{P}\left(\left|Y_{n+1} - \hat{f}(X_{n+1})\right| \le \hat{\tau}_n\right) \\
=& \mathbb{P}\left(Z_{n+1} \le \hat{Q}_n((1+\frac{1}{n})(1-\alpha))\right) \ge 1-\alpha
\end{aligned}$$

(e)

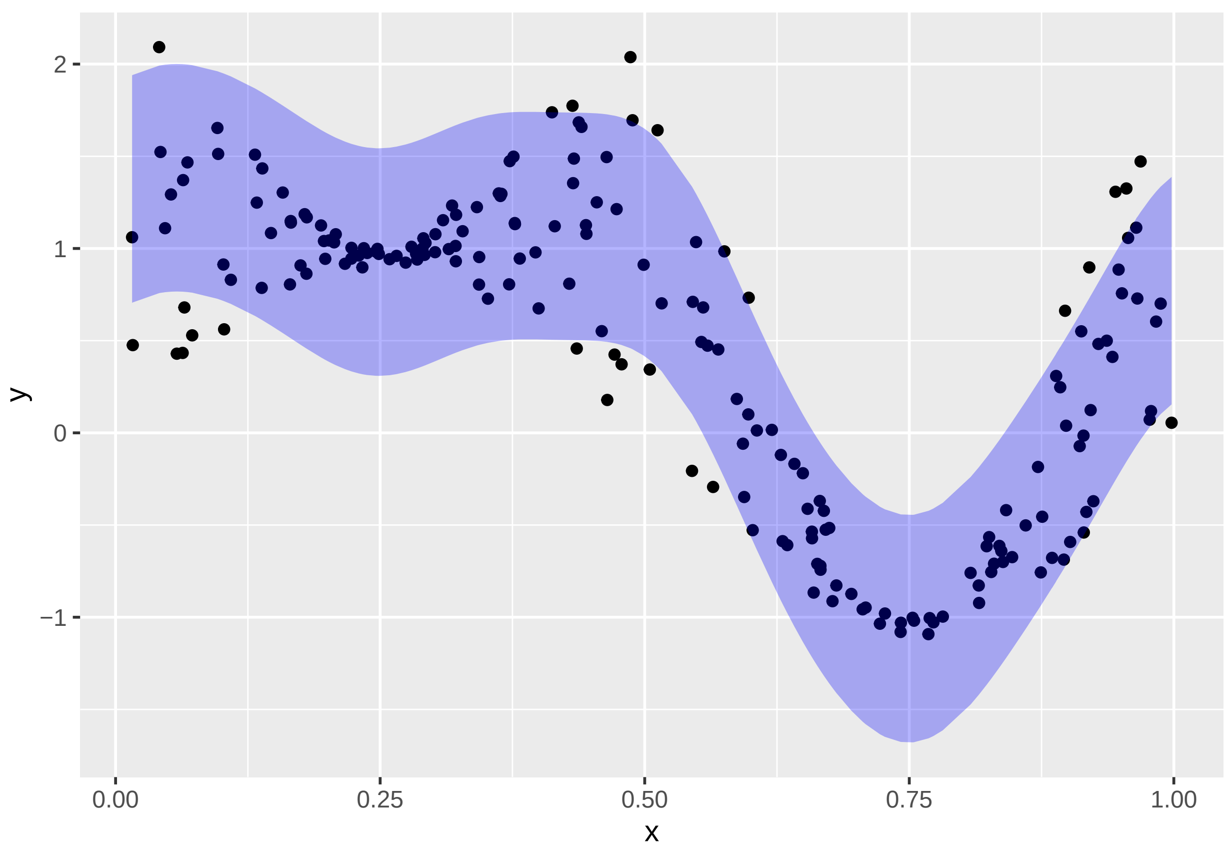
With the definition mentioned, we have

$$\begin{aligned}\mathbb{P}\left(Y_{n+1} \in \hat{C}_\delta(X_{n+1})\right) =& \mathbb{P}\left(\max\{\hat{q}_{\delta_{\text{low}}} - Y_{n+1}, Y_{n+1} - \hat{q}_{\delta_{\text{high}}}\} \le \hat{\tau}_n\right) \\ =& \mathbb{P}\left(Z_{n+1} \le \hat{Q}_n(\frac{1+n}{n}(1-\alpha))\right) \ge 1-\alpha\end{aligned}$$

(f)

```r
## function provided in lecture file
n.sample = 200

X.train = sort(runif(n.sample, min=0, max = 1))
X.valid = sort(runif(n.sample, min=0, max = 1))
X.test = sort(runif(n.sample, min=0, max = 1))

y.train = sin(2 * pi * X.train) - .1 +
    (1.1 + cos(4 * pi * X.train)) * runif(n.sample, min = 0, max = 1)

y.valid = sin(2 * pi * X.valid) - .1 +
    (1.1 + cos(4 * pi * X.valid)) * runif(n.sample, min = 0, max = 1)

y.test = sin(2 * pi * X.test) - .1 +
    (1.1 + cos(4 * pi * X.test)) * runif(n.sample, min = 0, max = 1)

library(CVXR)
library(tidyverse)

predictKRR <- function(X, Z, beta, tau = .1, offset = 0) {
    if ((is.matrix(X) && ncol(X) != 1) ||
        (is.matrix(Z) && ncol(Z) != 1)) {
        stop("Data should be 1-dimensional")
    }
    nn = length(X)
    mm = length(Z)
    ip.matrix = X %*% t(Z)  # n-by-m matrix with entries x_i' * z_j
    squared.Xs = replicate(mm, X * X)  # n-by-m matrix whose ith
                                        # row is all x_i' * x_i
    squared.Zs = replicate(nn, Z * Z)  # m-by-n matrix whose ith row
                                        # is all z_i' * z_i
    dist.squared.matrix = squared.Xs - 2 * ip.matrix + t(squared.Zs)
    kernel.matrix = t(exp(-dist.squared.matrix / (2 * tau^2)))
    predictions = offset + kernel.matrix %*% beta;
    return(predictions)
}
fitQuantileKernel <- function(X, y, quantile.level = .5,
                              lambda = .01, tau = .1) {
    nn = length(X)
    if (is.matrix(X) && ncol(X) != 1) {
        stop("Data X should be 1-dimensional")
    }
    ## Construct the Gram matrix
    ip.matrix = X %*% t(X)  # n-by-n matrix with entries x_i' * x_j
    squared.Xs = (X * X) %*% t(rep(1, nn))  # turn it into an n-by-n matrix
    dist.squared = squared.Xs + t(squared.Xs) - 2 * ip.matrix
    G = exp(-dist.squared / (2 * tau^2))
    ## We formulate the problem as solving
    ##
    ##  minimize    sum_i l_alpha(z_i) + (lambda/2) * beta' * G * beta
    ##  subject to  z = y - G * beta
    ##
    ## but we don't actually introduce the new variable z...
    beta = Variable(nn)
    obj = ((1/nn) * sum(quantile.level * pos(y - G %*% beta) +
                        (1 - quantile.level) * neg(y - G %*% beta)) +
           (lambda/2) * quad_form(beta, G))
    problem = Problem(Minimize(obj))
    result = solve(problem)
    return(result$getValue(beta))
}
fitKernelRidge <- function(X, y, lambda = .01, tau = .1) {
    nn = length(X)
    if (is.matrix(X) && ncol(X) != 1) {
        stop("Data X should be 1-dimensional")
    }
    ## Construct the Gram matrix
    ip.matrix = X %*% t(X)  # n-by-n matrix with entries x_i' * x_j
    squared.Xs = (X * X) %*% t(rep(1, nn))  # turn it into an n-by-n matrix
    dist.squared = squared.Xs + t(squared.Xs) - 2 * ip.matrix
    G = exp(-dist.squared / (2 * tau^2))
    diag(G) = 1 + lambda;
    beta = solve(G, y)
    return(beta)
}
plotUpperAndLower <- function(X, y, y.low, y.high, filename) {
    dat = tibble(x = X, y = y,
                 low = y.low, high = y.high)
    p = ggplot(dat, aes(x = x)) +
        geom_point(aes(y = y)) +
        geom_ribbon(aes(ymin = low, ymax = high),
                    fill = "blue", alpha = .3)
    ggsave(filename, p, device="pdf")
    return(p)
}

delta.low <- 0.1
delta.high <- 0.9
tau <- 0.1
lambda <- 0.01
alpha <- 0.1
n <- length(X.valid)

## Fit squared loss KRR
beta.krr <- fitKernelRidge(X.train, y.train, lambda = lambda, tau = tau)
## Fit quantile KRR
beta.qkrr.low <- fitQuantileKernel(X.train, y.train, quantile.level = delta.low, lambda = lambda, tau
beta.qkrr.high <- fitQuantileKernel(X.train, y.train, quantile.level = delta.high, lambda = lambda, t

# For ridge regression estimate, find its conformal tau
yhat.krr.val <- predictKRR(X.train, X.valid, beta.krr, tau = tau)
tau.hat.krr <- quantile(abs(yhat.krr.val - y.valid), (1+1/n)*(1 - alpha))
# For quantile kernel ridge regression estimate, find its conformal tau
yhat.qkrr.low.val <- predictKRR(X.train, X.valid, beta.qkrr.low, tau = tau)
yhat.qkrr.high.val <- predictKRR(X.train, X.valid, beta.qkrr.high, tau = tau)
Z.qkrr.val <- pmax(yhat.qkrr.low.val - y.valid, y.valid - yhat.qkrr.high.val)
tau.hat.qkrr <- quantile(Z.qkrr.val, (1+1/n)*(1 - alpha))

# Plot conformal band on test set
yhat.krr.test <- predictKRR(X.train, X.test, beta.krr, tau = tau)
yhat.qkrr.low.test <- predictKRR(X.train, X.test, beta.qkrr.low, tau = tau)
yhat.qkrr.high.test <- predictKRR(X.train, X.test, beta.qkrr.high, tau = tau)

plot.krr <- plotUpperAndLower(X.test, y.test, yhat.krr.test - tau.hat.krr, yhat.krr.test + tau.hat.kr
## Saving 6.5 x 4.5 in image
```

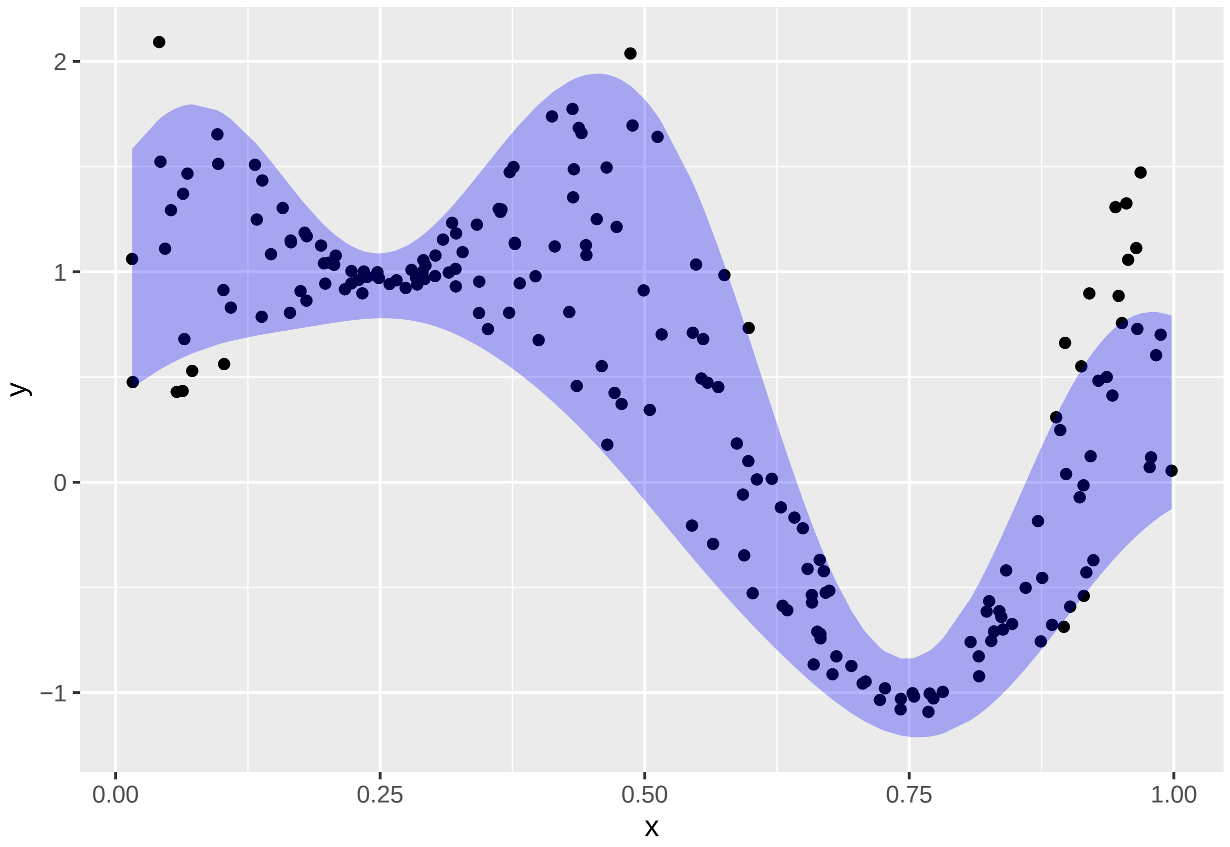
```
plot.krr
```



```
plot.qkrr <- plotUpperAndLower(X.test, y.test, yhat.qkrr.low.test - tau.hat.qkrr, yhat.qkrr.high.test
```

```
## Saving 6.5 x 4.5 in image
```

```
plot.qkrr
```



```
## Compute covering rate
covering.rate.krr <- mean(abs(y.test - yhat.krr.test) <= tau.hat.krr)
covering.rate.qkrr <- mean( pmax(yhat.qkrr.low.test - y.test, y.test - yhat.qkrr.high.test) <= tau.ha

cat("Covering rate for KRR is ", covering.rate.krr, "\n", sep = '', labels = '.')
```

```
## Covering rate for KRR is 0.875
```

```
cat("Covering rate for quantile KRR is ", covering.rate.qkrr, "\n", sep = '', labels = '.')
```

```
## Covering rate for quantile KRR is 0.91
```