A. Suppose we want to estimate the value of the regression function y^* at some new point x^*, denoted $\hat{f}(x^*)$. Assume for the moment that $f(x)$ is linear, and that y and x have already had their means subtracted, in which case $y_i = \beta x_i + \epsilon_i$.

Return to your least-squares estimator for multiple regression. Show that for the one-predictor case, your prediction $\hat{y}^* = f(x^*) = \hat{\beta}x^*$ may be expressed as a *linear smoother* of the following form:

$$\hat{f}(x^*) = \sum_{i=1}^{n} w(x_i, x^*) y_i$$

for any x^*. Inspect the weighting function you derived. Briefly describe your understanding of how the resulting smoother behaves, compared with the smoother that arises from an alternate form of the weight function $w(x_i, x^*)$:

$$w_K(x_i, x^*) = \begin{cases} 1/K, & x_i \text{ one of the } K \text{ closest sample points to } x^*, \\ 0, & \text{otherwise.} \end{cases}$$

This is referred to as *K-nearest-neighbor smoothing*.

Recall the two following things we need to solve this problem:

$$\hat{\beta} = (X^T X)^{-1} X^T Y,$$
$$X^T Y = \sum_{i=1}^{n} x_i^T y_i,$$

where the first is a known result from multiple linear regression and the second is the expression of a product of matrices as the sum of outer products. With these two, we obtain the following:

$$\begin{aligned} \hat{\beta}x^* &= (X^T X)^{-1} X^T Y x^* \\ &= \left(\sum_{i=1}^{n} x_i^T x_i \right)^{-1} \sum_{i=1}^{n} x_i^T y_i x^* \\ &= \sum_{i=1}^{n} \frac{x_i^T x^*}{\sum_{i=1}^{n} x_i^T x_i} y_i \end{aligned}$$

From the above, we see that

$$w(x_i, x^*) = \frac{x_i^T x^*}{\sum_{i=1}^{n} x_i^T x_i},$$

which smooths the new x^* by scaling it with the ratio $\frac{x_i^T}{x_i^T x_i}$; this effectively takes the proximity of x^* to each x_i into account when summing over all x_i. In comparison, the K-nearest-neighbor smoothing simply scales x^* uniformly across all x_i that are "close" to x_i. We can do this by defining a neighborhood of x_i near x^*, then applying the uniform scale depending on how many x_i belong to this neighborhood.

B. A *kernel function* $K(x)$ is a smooth function satisfying

$$\int_{\mathbb{R}} K(x)dx = 1\,, \quad \int_{\mathbb{R}} xK(x)dx = 0\,, \quad \int_{\mathbb{R}} x^2K(x)dx > 0\,.$$

A very simple example is the uniform kernel,

$$K(x) = \frac{1}{2}I(x) \quad \textbf{where} \quad I(x) = \begin{cases} 1, & |x| \leq 1 \\ 0, & \textbf{otherwise}\,. \end{cases}$$

Another common example is the Gaussian kernel:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,.$$

Kernels are used as weighting functions for taking local averages. Specifically, define the weighting function

$$w(x_i, x^*) = \frac{1}{h} K\left(\frac{x_i - x^*}{h}\right)\,,$$

where h is the bandwidth. Using this weighting function in a linear smoother is called *kernel regression*. (The weighting function gives the unnormalized weights; you should normalize the weights so that they sum to 1.)

Write your own R function that will fit a kernel smoother for an arbitrary set of x-y pairs and arbitrary choice of (positive real) bandwidth h. You choose the kernel. Set up an R script that will simulate noisy data from some nonlinear function, $y = f(x) + \epsilon$; subtract the sample means from the simulated x and y; and use your function to fit the kernel smoother for some choice of h. Plot the estimated functions for a range of bandwidths wide enough to yield noticeable differences in the qualitative behavior of the prediction functions.

I chose to work with the Gaussian kernel since I have read papers that mention it. For the bandwidth, I used $h = 1, 2, 5, 10$ to yield noticeable differences in the

behavior of the estimated function. Thus, with these bandwidths, the Gaussian kernel, $f_1(x) = 4x^3 + x^2 - 12$, and $x^* \in [-10, 10]$, I obtain Figure 1. Notice that as h increases, our estimated function grows increasingly linear, which is the behavior we would expect to see from our linear smoothing. This behavior can also be seen in Figure 2, where I change the original function to $f_2(x) = x\sin(x)$.

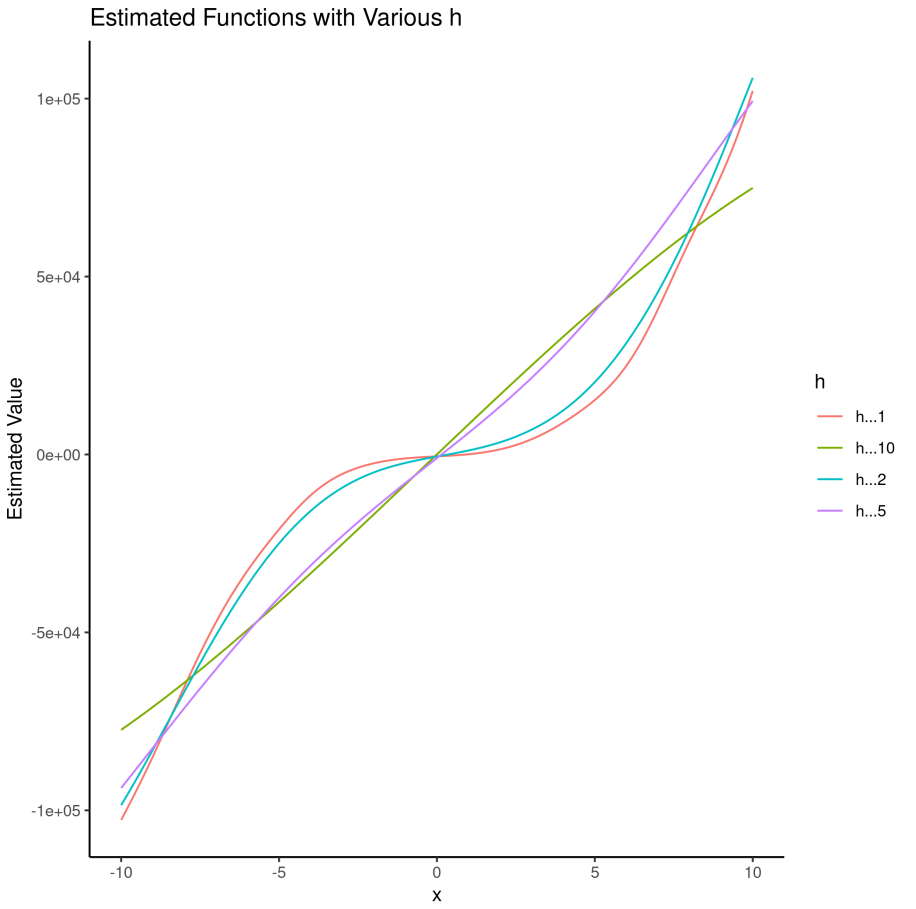

Figure 1: Estimated Functions Under Gaussian Kernel with Various Bandwidths and $f_1(x)$.

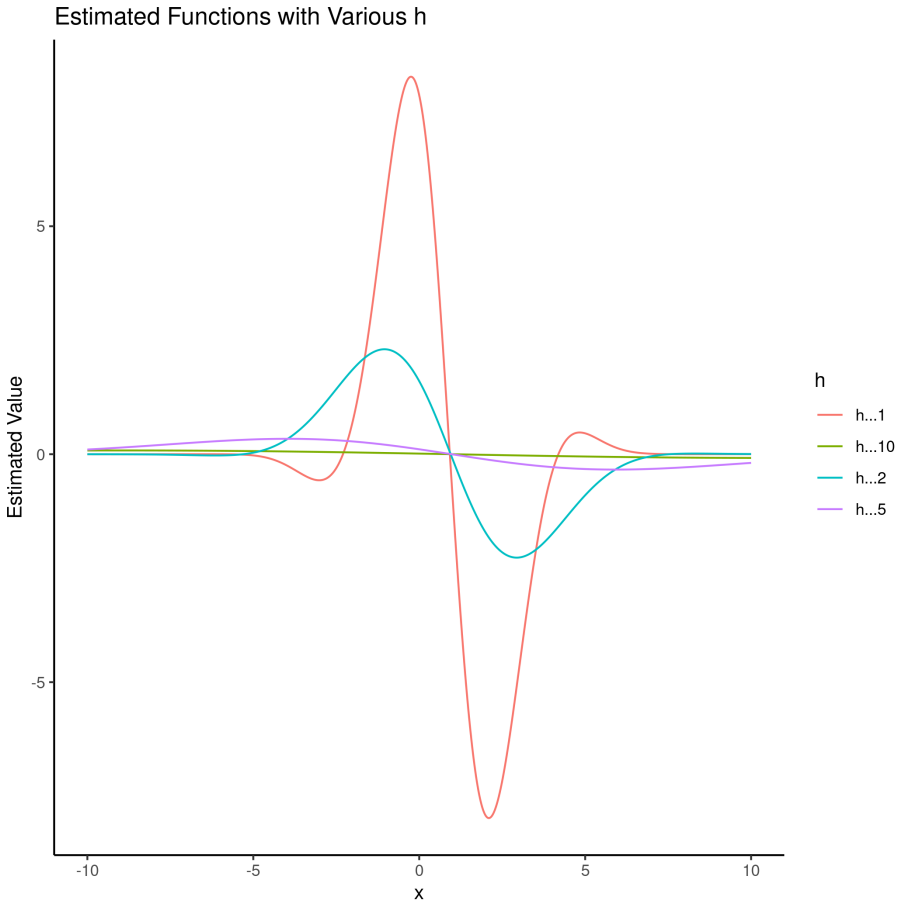

Figure 2: Estimated Functions Under Gaussian Kernel with Various Bandwidths and $f_2(x)$.

A. Presumably a good choice of h would be one that led to smaller predictive errors on fresh data. Write a function or script that will: (1) accept an old ("training") data set and a new ("testing") data set as inputs; (2) fit the kernel-regression estimator to the training data for specified choices of h; and (3) return the estimated functions and the realized prediction error on the testing data for each value of h. This should involve a fairly straightforward "wrapper" of the function you've already written.

Using $f_2(x)$ and $h = 1, 2, 5, 10$ from the previous section, I obtain Figure 3, where the black dots are data in the testing data set. Visually, we can see that the testing data best matches with the kernel-regression estimator when $h = 2$. This conclusion is *validated* when looking at the mean square prediction errors (MSPE) for $h = 1, 2, 5, 10$ which were $MSPE_h = 2871.7169, 504.5653, 777.8502,$ and 791.7469, respectively. Since we want to choose the estimated function with the lowest MSPE, our choice for bandwidth is surely $h = 2$.

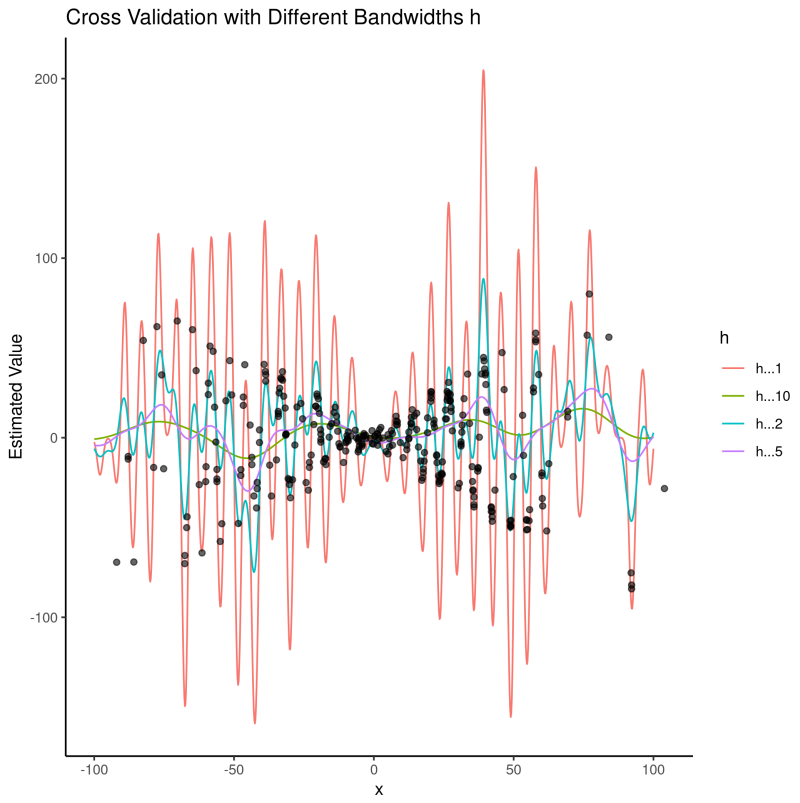

Figure 3: Cross Validation with Different Bandwidths h.

B. Imagine a conceptual two-by-two table for the unknown, true state of affairs. The rows of the table are "wiggly function" and "smooth function," and the columns are "highly noisy observations" and "not so noisy observations." Simulate one data set (say, 500 points) for each of the four cells of this table, where the x's take values in the unit interval. Then split each data set into training and testing subsets. You choose the functions. Apply your method to each case, using the testing data to select a bandwidth parameter. Choose the estimate that minimizes the average squared error in prediction, which estimates the mean-squared error:

$$L_n(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n^*}(y_i^* - \hat{y}_i^*)^2 \,,$$

where (y_i^*, x_i^*) are the points in the test set, and $\hat{y}_i^*$ is your predicted value arising from the model you fit using only the training data. Does your out-of-sample predictive validation method lead to reasonable choices of h for each case?

I used the following functions for this problem:

	High Noise	**Low Noise**
Wiggly Function	$f(x) = \cos(10x) + N(0, 0.8)$	$f(x) = \cos(10x) + N(0, 0.25)$
Smooth Function	$f(x) = x^{5/3} + N(0, 0.8)$	$f(x) = x^{5/3} + N(0, 0.25)$

Once again, I used the Gaussian kernel with bandwidths $h = 0.1, 0.2, 0.5, 1$. I split my 500 observations into 375 for training and 125 for testing. Note that, in the unit interval, the "high noise" significantly scales the observations, which is evident in the left column of Figure 4.

Under these four scenarios, the optimal h for the "wiggly" function and the "smooth" function under low noise was the lowest bandwidth $h = 0.1$. This makes sense since the wiggly function is best "matched" by a linear smoother that does the least amount of smoothing. Visually, we can see in the top row of Figure 4 that the less we smooth, the closer the testing data is to our estimated function. Meanwhile, when starting with a relatively smoother function such as in the bottom row, our optimal smoother may come in the form of a small value for h. In fact, for the "smooth" function with high noise, we find that $h = 0.2$ is the optimal bandwidth under seed *702*. Therefore, it seems that our out-of-sample predictive validation does lead to reasonable choices of h for each case since we have a decent amount of data.

	Wiggly/High	Wiggly/Low	Smooth/High	Smooth/Low
h=0.1	**0.638**	**0.139**	0.616	**0.066**
h=0.2	0.861	0.400	**0.614**	0.074
h=0.5	0.975	0.532	0.641	0.114
h=1	0.979	0.536	0.665	0.139

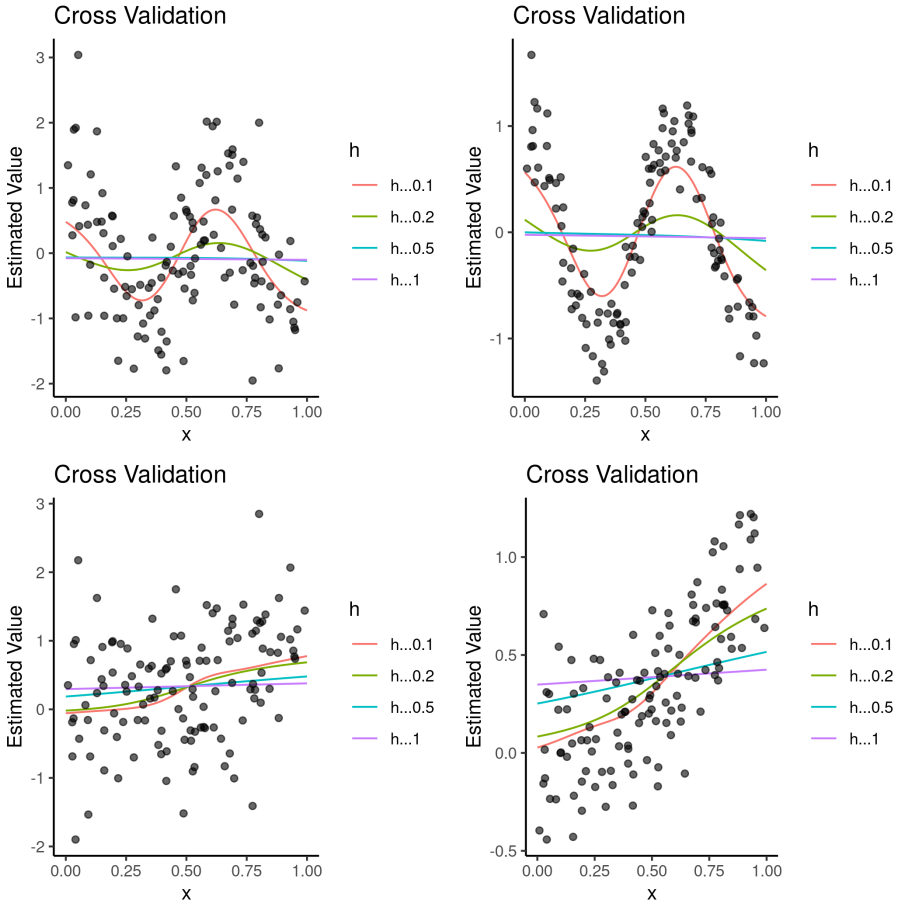

Figure 4: Comparing Optimal Bandwidths Across Different Scenarios in Out-of-Sample Cross Validation.

C. Use the leave-one-out lemma to revisit the examples you simulated in Part B, using leave-one-out cross validation to select h in each case. Because of the leave-one-out lemma, you won't need to actually refit the model N times!

Under linear smoothers, computation of LOOCV simplifies greatly to

$$\text{LOOCV} = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - H_{ii}} \right)^2$$

This is convenient, since we can easily obtain some metric to determine the optimal bandwidth without having to split our data set into two. Note that with the out-of-sample approach, we are only able to use 75% of the data to fit an estimated function, whereas with this method, we are able to use all of the data!

In Figure 5, we can see the same smoothed estimated functions from Figure 4 with all of the data points overlaid. Note that the estimated functions may differ slightly in shape because we were able to use all of the data to fit them. Using the LOOCV criterion, the optimal bandwidth for both the "wiggly" and "smooth" function under both high and low noise was $h = 0.1$, indicating that when we use all of the data to obtain our estimated function in a variety of conditions, it's most optimal not to smooth the data given a sufficient amount of data.

	Wiggly/High	**Wiggly/Low**	**Smooth/High**	**Smooth/Low**
h=0.1	**0.684**	**0.150**	**0.594**	**0.068**
h=0.2	0.988	0.426	0.601	0.074
h=0.5	1.139	0.564	0.644	0.114
h=1	1.143	0.569	0.672	0.139

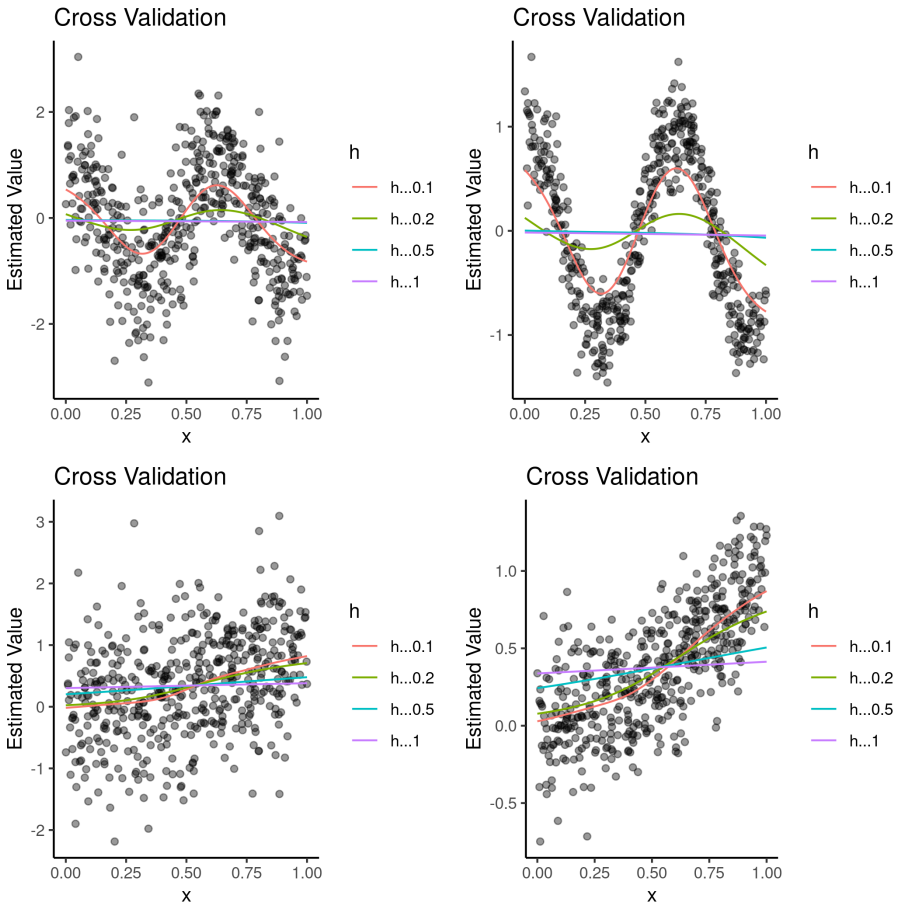

Figure 5: Comparing Optimal Bandwidths Across Different Scenarios in Leave-One-Out Cross Validation.

A. A natural generalization of locally constant regression is local polynomial regression. For points u in a neighborhood of the target point x, define the polynomial

$$g_x(u;a) = a_0 + \sum_{k=1}^{D} a_k(u-x)^k$$

for some vector of coefficients $a = (a_0, \ldots, a_D)$. As above, we will estimate the coefficients a in $g_x(u;a)$ at some target point x using weighted least squares:

$$\hat{a} = \arg\min_{R^{D+1}} \sum_{i=1}^{n} w_i \left\{ y_i - g_x(x_i;a) \right\}^2 ,$$

where $w_i \equiv w(x_i, x)$ are the kernel weights defined just above, normalized to sum to one. Derive a concise (matrix) form of the weight vector $\hat{a}$, and by extension, the local function estimate $\hat{f}(x)$ at the target value x. Life will be easier if you define the matrix R_x whose (i,j) entry is $(x_i - x)^{j-1}$, and remember that (weighted) polynomial regression is the same thing as (weighted) linear regression with a polynomial basis.

First, let's express $\sum_{i=1}^{n} w_i \left\{ y_i - g_x(x_i;a) \right\}^2$ in matrix form. This is not such an arduous task once we realize that the summation contains a quadratic. However, before we get to that, let's use the given hint:

$$\begin{aligned} R_x\mathbf{a} &= \begin{bmatrix} a_0 + a_1(x_1 - x) + \ldots + a_D(x_1 - x)^D \\ \vdots \\ a_0 + a_1(x_n - x) + \ldots + a_D(x_n - x)^D \end{bmatrix} \\ &= \begin{bmatrix} g_x(x_1|\mathbf{a}) \\ \vdots \\ g_x(x_n|\mathbf{a}) \end{bmatrix} \end{aligned}$$

Now we can write the summation in matrix notation:

$$\sum_{i=1}^{n} w_i \left\{ y_i - g_x(x_i;a) \right\}^2 = (\mathbf{y} - R_x\mathbf{a})^T \mathrm{diag}(\tilde{\mathbf{w}})(\mathbf{y} - R_x\mathbf{a})$$

Now, let's derive the matrix expression with respect to $\mathbf{a}$ to obtain $\hat{\mathbf{a}}$:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{a}} \left[(\mathbf{y} - R_x\mathbf{a})^T \mathrm{diag}(\tilde{\mathbf{w}})(\mathbf{y} - R_x\mathbf{a}) \right] &= -\left(\mathbf{y}^T \mathrm{diag}(\tilde{\mathbf{w}}) R_x \right)^T - R_x^T \mathrm{diag}(\tilde{\mathbf{w}})\mathbf{y} + 2\left(R_x^T \mathrm{diag}(\tilde{\mathbf{w}}) R_x \right)\mathbf{a} \\ &= -2R_x^T \mathrm{diag}(\tilde{\mathbf{w}})\mathbf{y} + 2\left(R_x^T \mathrm{diag}(\tilde{\mathbf{w}}) R_x \right)\mathbf{a} \stackrel{set}{=} 0 \\ R_x^T \mathrm{diag}(\tilde{\mathbf{w}}) R_x \mathbf{a} &= R_x^T \mathrm{diag}(\tilde{\mathbf{w}})\mathbf{y} \\ \hat{\mathbf{a}} &= \left(R_x^T \mathrm{diag}(\tilde{\mathbf{w}}) R_x \right)^{-1} R_x^T \mathrm{diag}(\tilde{\mathbf{w}})\mathbf{y} \end{aligned}$$

By extension, $\hat{f}(x) = \mathbf{e}^T\hat{\mathbf{a}}$, where $e^T = [1\ \mathbf{0}]$.

B. From this, conclude that for the special case of the local linear estimator ($D = 1$), we can write $\hat{f}(x)$ as a linear smoother of the form

$$\hat{f}(x) = \frac{\sum_{i=1}^n w_i(x) y_i}{\sum_{i=1}^n w_i(x)},$$

where the unnormalized weights are

$$\begin{aligned} w_i(x) &= K\left(\frac{x - x_i}{h}\right)\{s_2(x) - (x_i - x)s_1(x)\} \\ s_j(x) &= \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)(x_i - x)^j. \end{aligned}$$

With $D = 1$, note that $g_x(x_i|a) = a_0 + a_1(x_i - x)$. Further, we can write

$$\begin{aligned} \hat{f}(x) &= \mathbf{e}^T\hat{\mathbf{a}} \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix}(R_x^T \text{diag}(\tilde{\mathbf{w}})R_x)^{-1}R_x^T\text{diag}(\tilde{\mathbf{w}})\mathbf{y} \end{aligned}$$

With hopes to keep this computation as organized as possible, let's take a modular approach and obtain the following:

$$\begin{aligned} R_x^T\text{diag}(\tilde{\mathbf{w}}) &= \begin{bmatrix} 1 & \dots & 1 \\ x_1 - x & \dots & x_n - x \end{bmatrix}\begin{bmatrix} \tilde{w}_1 & \dots & 0 \\ \vdots & \tilde{w}_i & \vdots \\ 0 & \dots & \tilde{w}_n \end{bmatrix} \\ &= \begin{bmatrix} \tilde{w}_1 & \dots & \tilde{w}_n \\ \tilde{w}_1(x_1 - x) & \dots & \tilde{w}_n(x_n - x) \end{bmatrix}, \\ R_x^T\text{diag}(\tilde{\mathbf{w}})R_x &= \begin{bmatrix} \tilde{w}_1 & \dots & \tilde{w}_n \\ \tilde{w}_1(x_1 - x) & \dots & \tilde{w}_n(x_n - x) \end{bmatrix}\begin{bmatrix} 1 & x_1 - x \\ \vdots & \vdots \\ 1 & x_n - x \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^n \tilde{w}_i & \sum_{i=1}^n \tilde{w}_i(x_i - x) \\ \sum_{i=1}^n \tilde{w}_i(x_i - x) & \sum_{i=1}^n \tilde{w}_i(x_i - x)^2 \end{bmatrix}, \\ \left(R_x^T\text{diag}(\tilde{\mathbf{w}})R_x\right)^{-1} &= \frac{1}{\mathcal{D}}\begin{bmatrix} \sum_{i=1}^n \tilde{w}_i(x_i - x)^2 & -\sum_{i=1}^n \tilde{w}_i(x_i - x) \\ -\sum_{i=1}^n \tilde{w}_i(x_i - x) & \sum_{i=1}^n \tilde{w}_i \end{bmatrix}, \end{aligned}$$

where $\mathcal{D}$ is the determinant of the matrix:

$$\begin{aligned}\mathcal{D} &= \sum_{i=1}^{n} \tilde{w}_i (x_i - x)^2 - \left(\sum_{i=1}^{n} \tilde{w}_i (x_i - x) \right)^2 \\ &= \sum_{i=1}^{n} K(\cdot)(x_i - x)^2 - \left(\sum_{i=1}^{n} K(\cdot)(x_i - x) \right)^2 \\ &= s_2(x) - s_1^2(x),\end{aligned}$$

where I'm getting lazy so I'm using shorthand notation like $S_k^{-1} = 1/\sum_{i=1}^{n} K(\cdot)$ and $K(\cdot) = K((x_i - x)/h)$. Now, let's continue with the matrix multiplication:

$$\begin{aligned}&\begin{bmatrix}1 & 0\end{bmatrix} \frac{1}{\mathcal{D}} \begin{bmatrix} \sum_{i=1}^{n} \tilde{w}_i (x_i - x)^2 & -\sum_{i=1}^{n} \tilde{w}_i (x_i - x) \\ -\sum_{i=1}^{n} \tilde{w}_i (x_i - x) & \sum_{i=1}^{n} \tilde{w}_i \end{bmatrix} \\ &\quad = \begin{bmatrix} \frac{\sum_{i=1}^{n} K(\cdot)(x_i - x)^2}{\mathcal{D}} & \frac{-\sum_{i=1}^{n} K(\cdot)(x_i - x)}{\mathcal{D}} \end{bmatrix}, \\ \hat{f}(x) &= \begin{bmatrix} \frac{\sum_{i=1}^{n} K(\cdot)(x_i - x)^2}{\mathcal{D}} & \frac{-\sum_{i=1}^{n} K(\cdot)(x_i - x)}{\mathcal{D}} \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{n} K(\cdot) y_i \\ \sum_{i=1}^{n} K(\cdot)(x_i - x) y_i \end{bmatrix} \\ &\quad = \frac{s_2(x) \sum_{i=1}^{n} K(\cdot) y_i - s_1(x) \sum_{i=1}^{n} K(\cdot)(x_i - x) y_i}{s_2(x) - s_1^2(x)} \\ &\quad = \frac{\sum_{i=1}^{n} K(\cdot) \left[s_2(x) - (x_i - x) s_1(x) \right] y_i}{s_2(x) \sum_{i=1}^{n} K(\cdot) - s_1(x) \sum_{i=1}^{n} K(\cdot)(x_i - x)} \\ &\quad = \frac{\sum_{i=1}^{n} w_i(x) y_i}{\sum_{i=1}^{n} w_i(x)}\end{aligned}$$

C. Suppose that the residuals have constant variance σ^2 (that is, the spread of the residuals does not depend on x). Derive the mean and variance of the sampling distribution for the local polynomial estimate $\hat{f}(x)$ at some arbitrary point x. Note: the random variable $\hat{f}(x)$ is just a scalar quantity at x, not the whole function.

Note that we can write our local polynomial regression as

$$\begin{aligned}\hat{f}(x_i) &= e^T \hat{a} \\ &= e^T (X^T W X)^{-1} X^T W Y\end{aligned}$$

Now, we can take the expectation and variance of the above. Generally,

$$\begin{aligned}E[\hat{f}(x)] &= e^T (X^T W X)^{-1} X^T W f(x), \\ \text{var}[\hat{f}(x)] &= \sigma^2 e^T (X^T W X)^{-1} X^T W W^T X (X^T W X)^{-1} e\end{aligned}$$

In the case for (B) (i.e., $\hat{f}(x) = \tilde{w}^T y$), we can simplify the above expression and obtain:

$$E[\hat{f}(x)] = \tilde{w}^T f(x),$$
$$\text{var}[\hat{f}(x)] = \sigma^2 \tilde{w}^T \tilde{w},$$

where $\tilde{w} = \left[\frac{w_1(x)}{\sum_i w_i(x)}, ..., \frac{w_n(x)}{\sum_i w_i(x)}\right]^T$ comes from (B). Note that more smoothing implies more bias but less variance since $\tilde{w}^T \tilde{w} \leq 1$. This is another example of the bias-variance trade-off.

D. We don't know the residual variance, but we can estimate it. A basic fact is that if x is a random vector with mean μ and covariance matrix Σ, then for any symmetric matrix Q of appropriate dimension, the quadratic form $x^T Q x$ has expectation

$$E(x^T Q x) = \mathbf{tr}(Q\Sigma) + \mu^T Q \mu \,.$$

Write the vector of residuals as $r = y - \hat{y} = y - Hy$, where H is the smoothing matrix. Compute the expected value of the estimator

$$\hat{\sigma}^2 = \frac{\|r\|_2^2}{n - 2\mathbf{tr}(H) + \mathbf{tr}(H^T H)} \,,$$

and simplify things as much as possible. Roughly under what circumstances will this estimator be nearly unbiased for large n? Note: the quantity $2\mathbf{tr}(H) - \mathbf{tr}(H^T H)$ is often referred to as the "effective degrees of freedom" in such problems.

We can write $\hat{\sigma}^2$ as

$$\hat{\sigma}^2 = \frac{(y - Hy)^T (y - Hy)}{n - 2tr(H) + tr(H^T H)}$$

The denominator can be rewritten as

$$n - 2tr(H) + tr(H^T H) = tr\left[(I - H)^T (I - H)\right]$$

If we take the expectation of the numerator, we obtain

$$\begin{aligned} E\left[||r||_2^2\right] &= tr\left((I - H)^T (I - H)\sigma^2\right) + \mu^T (I - H)^T (I - H)\mu \\ &= tr\left((I - H)^T (I - H)\sigma^2\right) + ||f(x) - Hf(x)||_2^2, \end{aligned}$$

which is unbiased for σ^2 only when the second term is 0; this would only happen if the squared Euclidean norm of the bias was 0, implying bias is 0. Since this is unlikely, it's unlikely that $\hat{\sigma}^2$ is unbiased for σ^2.

E. Write code that fits the local linear estimator using a Gaussian kernel for a specified choice of bandwidth h. Then load the data in "utilities.csv" into R. This data set shows the monthly gas bill (in dollars) for a single-family home in Minnesota, along with the average temperature in that month (in degrees F), and the number of billing days in that month. Let y be the average daily gas bill in a given month (i.e. dollars divided by billing days), and let x be the average temperature. Fit y versus x using local linear regression and some choice of kernel. Choose a bandwidth by leave-one-out cross-validation.

Before fitting the local linear regression model, I tested 100 bandwidths between 0 and 15 to find the optimal bandwidth, which was $h = 6.873$; this was done using leave-one-out validation under a Gaussian kernel and the specified weighting scheme. The local linear estimates under these various bandwidths can be seen in Figure 6. Then, I obtained my estimates $\hat{\mathbf{y}}$ under this optimal bandwidth, which are reported in Figure 7, where the red dots are the actual observations and the black line are fitted values.

F. Inspect the residuals from the model you just fit. Does the assumption of constant variance (homoskedasticity) look reasonable? If not, do you have any suggestion for fixing it?

The residuals from the fitted model can be seen in Figure 8, which clearly depicts homoskedasticity. I do not believe the assumption of constant variance to be reasonable since variance is larger for smaller values of X. This may be remedied by admitting heterogeneous error in the model specification or with weighted local polynomial regression! The variance appears to be a function of X, so suppose we do the following to fix the heteroskedasticity:

$$\begin{aligned}
\log \text{var}(e_i) &= g(x_i), \\
\text{var}(e_i) &= E\left[(e_i - E(e_i))^2\right] \\
&= E[e_i^2], \\
\log E(e_i^2) &= g(x_i)
\end{aligned}$$

This should fix the heteroskedasticity since we are accounting for the values of X when determining the variance and, consequently, the residuals.

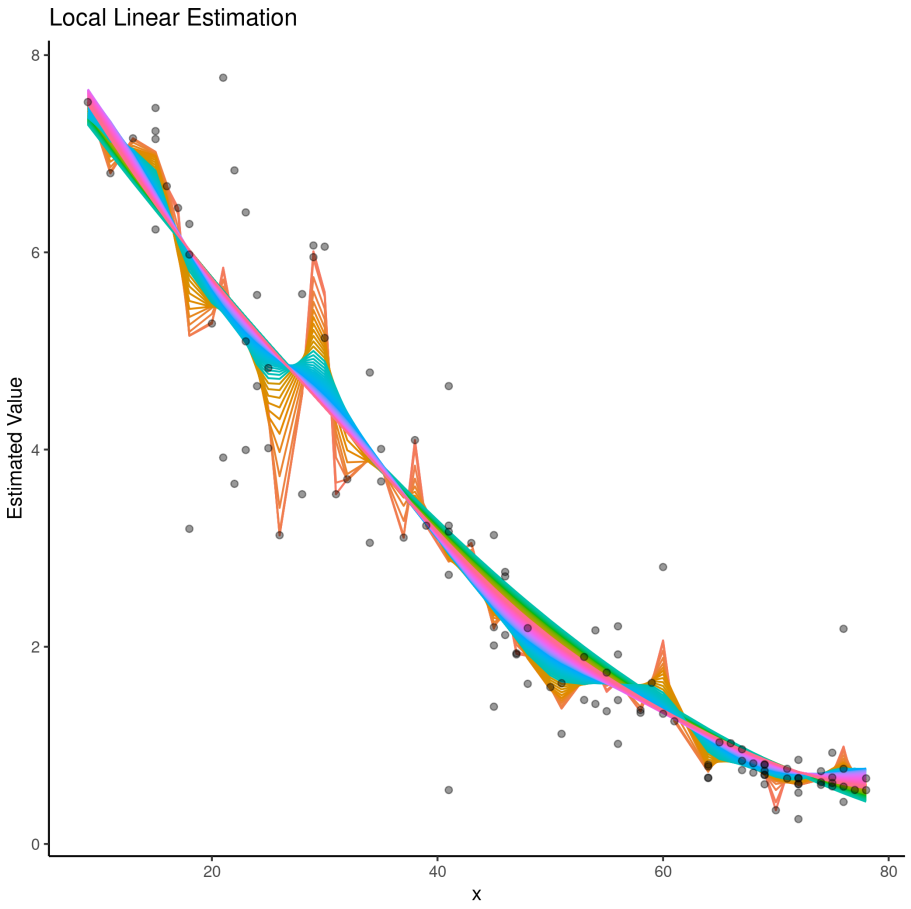

Figure 6: Local Linear Estimation at Various Bandwidths h.

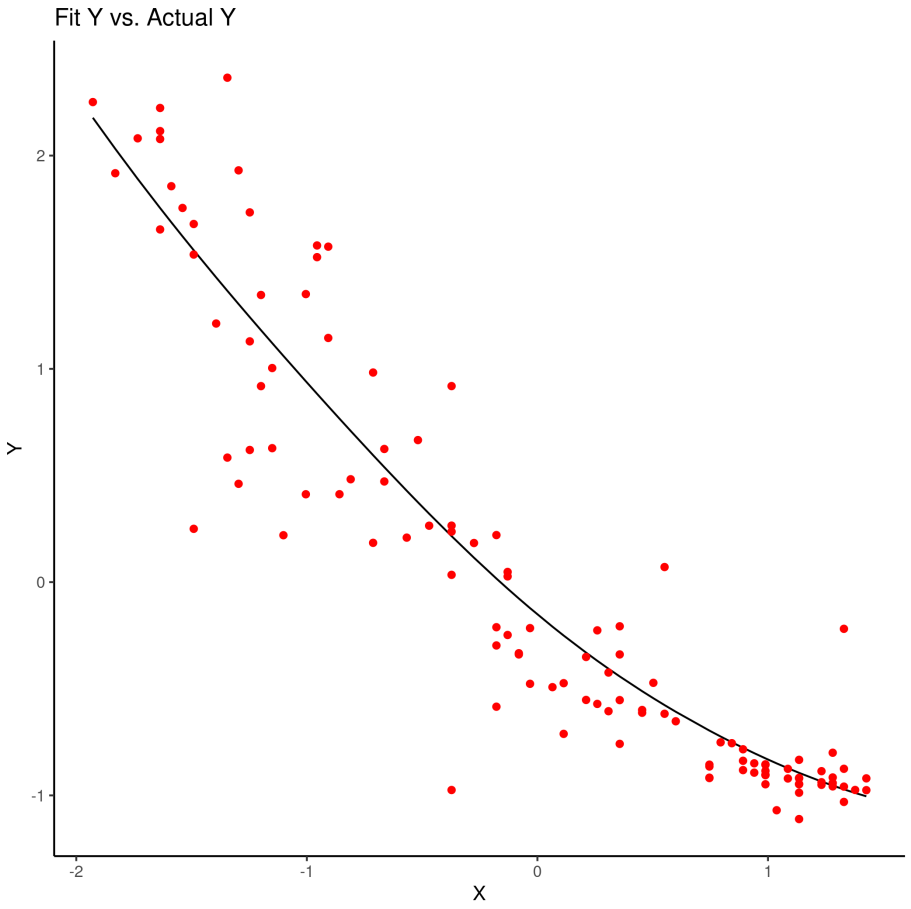

Figure 7: $\hat{y}$ vs. y Under Optimal Bandwidth.

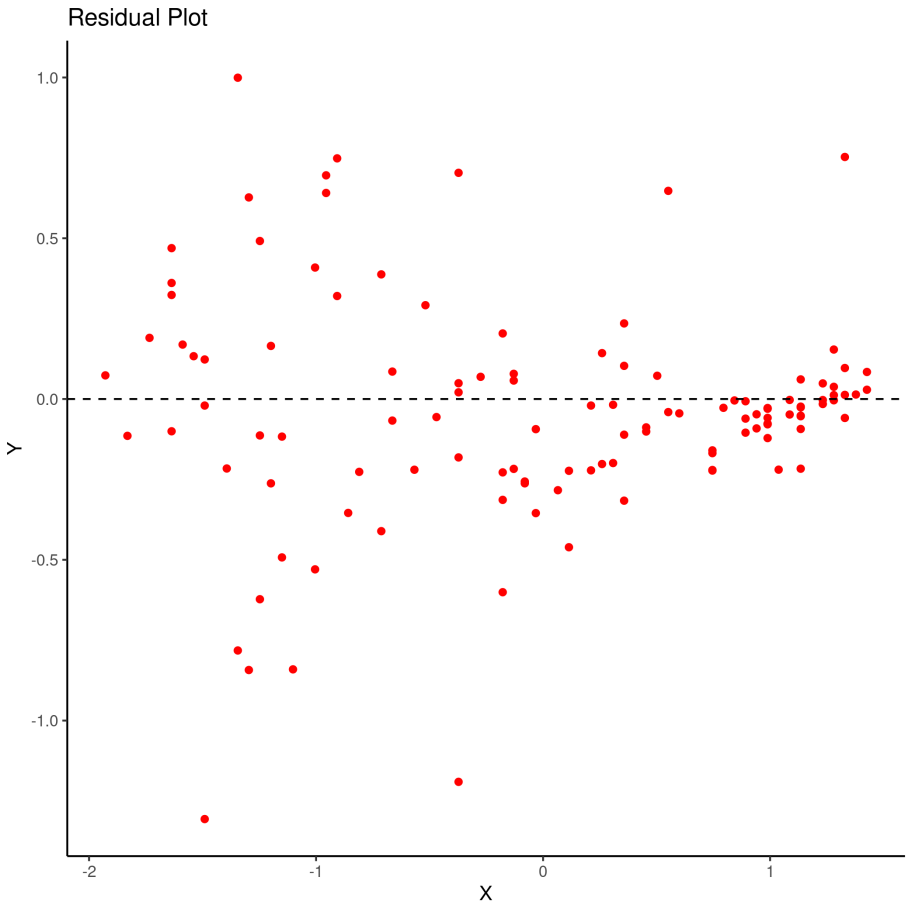

Figure 8: Residuals from Fit Model Under Optimal Bandwidth.

G. Put everything together to construct an approximate point-wise 95% confidence interval for the local linear model (using your chosen bandwidth) for the value of the function at each of the observed points x_i for the utilities data. Plot these confidence bands, along with the estimated function, on top of a scatter plot of the data.

The confidence bands (UT orange) overlaid on the estimated function (black line) and observations (red dots) can be found in Figure 9. It appears that, for the most part, our confidence bands slightly decrease in height as X increases. These bands were obtained from the following:

$$\hat{f}(x) \pm 1.96 \cdot \sqrt{\hat{\sigma}^2 ||h||_2^2},$$

where h is the row of the smoothing matrix and $\hat{\sigma}^2$ is obtained using (D).

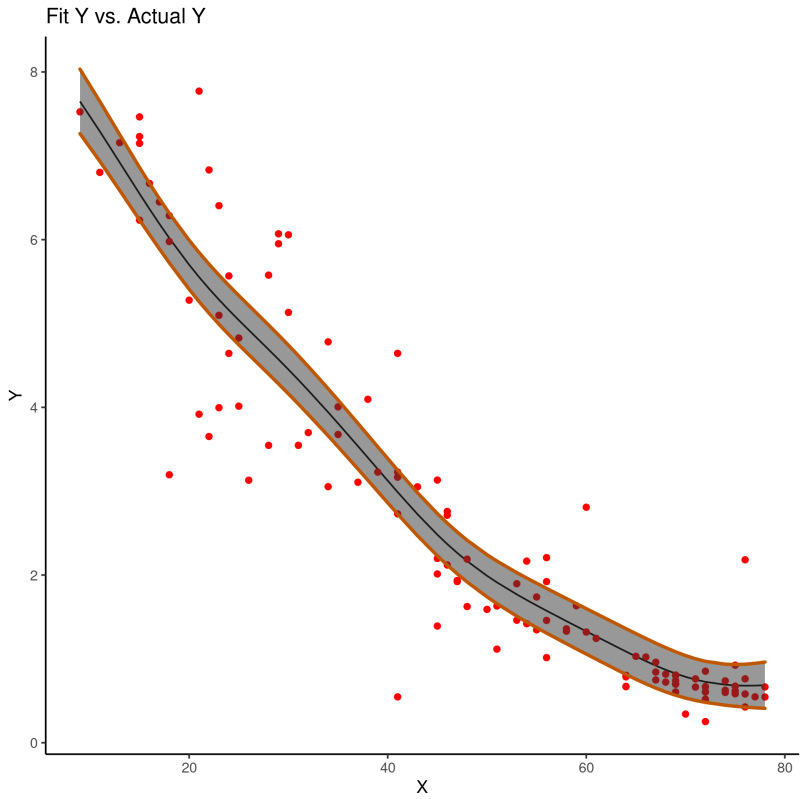

Figure 9: Point-wise 95% Confidence Intervals.

A. Read up on the Matern class of covariance functions. The Matern class has the *squared exponential* covariance function as a special case:

$$C_{SE}(x_1, x_2) = \tau_1^2 \exp\left\{-\frac{1}{2}\left(\frac{d(x_1, x_2)}{b}\right)^2\right\} + \tau_2^2 \delta(x_1, x_2)\,,$$

where $d(x_1, x_2) = \|x_1 - x_2\|_2$ is Euclidean distance (or just $|x - y|$ for scalars). The constants (b, τ_1^2, τ_2^2) are often called *hyperparameters*, and $\delta(a, b)$ is the Kronecker delta function that takes the value 1 if $a = b$, and 0 otherwise. But usually this covariance function generates functions that are "too smooth," and so we use other covariance functions in the Matern class as a default.

Let's start with the simple case where $\mathcal{X} = [0, 1]$, the unit interval. Write a function that simulates a mean-zero Gaussian process on $[0, 1]$ under the squared exponential covariance function. The function will accept as arguments: (1) finite set of points $x_1, \ldots, x_N$ on the unit interval; and (2) a triplet (b, τ_1^2, τ_2^2). It will return the value of the random process at each point: $f(x_1), \ldots, f(x_N)$.

Use your function to simulate (and plot) Gaussian processes across a range of values for b, τ_1^2, and τ_2^2. Try starting with a very small value of τ_2^2 (say, 10^{-6}) and playing around with the other two first. On the basis of your experiments, describe the role of these three hyperparameters in controlling the overall behavior of the random functions that result. What happens when you try $\tau_2^2 = 0$? Why? If you can fix this, do—remember our earlier discussion on different ways to simulate the MVN.

Now simulating a few functions with a different covariance function, the Matérn with parameter $5/2$:

$$C_{M52}(x_1, x_2) = \tau_1^2 \left\{1 + \frac{\sqrt{5}d}{b} + \frac{5d^2}{3b^2}\right\} \exp\left(\frac{-\sqrt{5}d}{b}\right) + \tau_2^2 \delta(x_1, x_2)\,,$$

where $d = \|x_1 - x_2\|_2$ is the distance between the two points x_1 and x_2. Comment on the differences between the functions generated from the two covariance kernels.

First, I test the two given covariance functions (the squared exponential and Matern $\frac{5}{2}$ covariance functions) across a grid of 100 different points in the parameter space for (b, τ_1^2). These points for b and τ_1^2 range from 0.0001 to 1 and 0

to 2, respectively. The resulting simulations for the squared exponential covariance function and the Matern $\frac{5}{2}$ covariance function can be found in Figures 10 and 11, respectively. Note how the squared exponential covariance function generates much smoother simulations at every point in the parameter space for (b, τ_1^2).

In the aforementioned simulations, we fixed $\tau_2^2 = 10^{-6}$. However, once $\tau_2^2 = 0$, we run into identifiability issues, because the nugget in the covariance functions (the Kronecker delta) coincides with the variance imposed by the Gaussian specification. Additionally, when $\tau_2^2 = 0$, our results are often numerically unstable. We can get around this by implementing a singular matrix specification of the normal distribution.

For the squared exponential covariance function, the b hyperparameter acts as our bandwidth; as b increases, our functions become smoother, which is evident in each of the discussed figures. In fact, in the top left subplot for each figure, you can see the lack of smoothness for each value τ_1^2. As τ_1^2 increases, the covariance increasingly scales. Note how when $\tau_1^2 \approx 0$, each subplot in Figure 10 depicts a horizontal line at 0. However, as τ_1^2 increases, the scale (amplitude) increases.

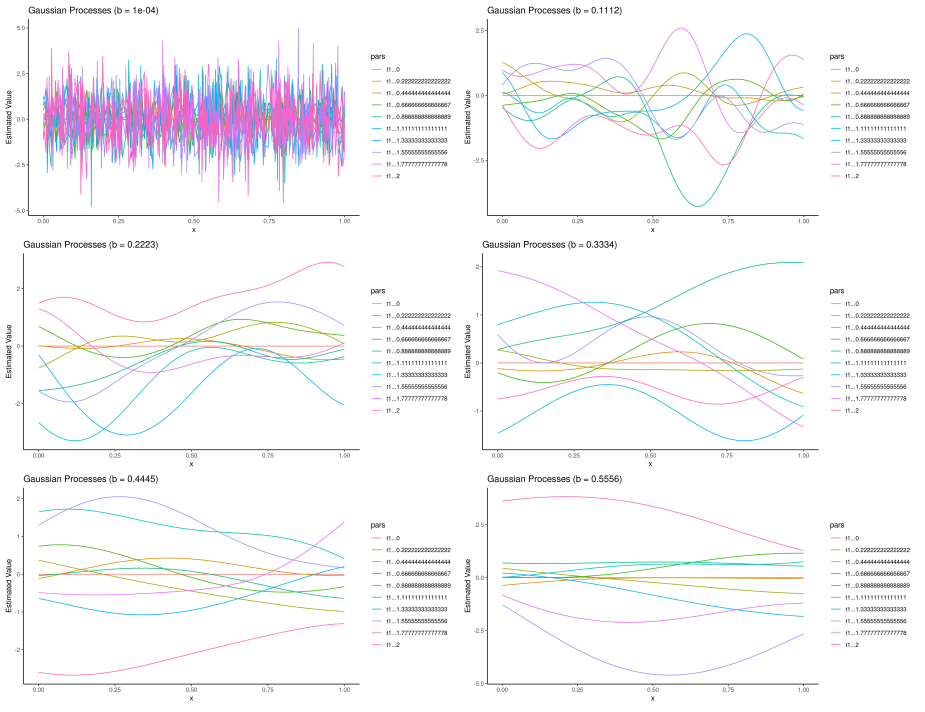

Figure 10: Gaussian Process Simulations with Squared Exponential Covariance Function.

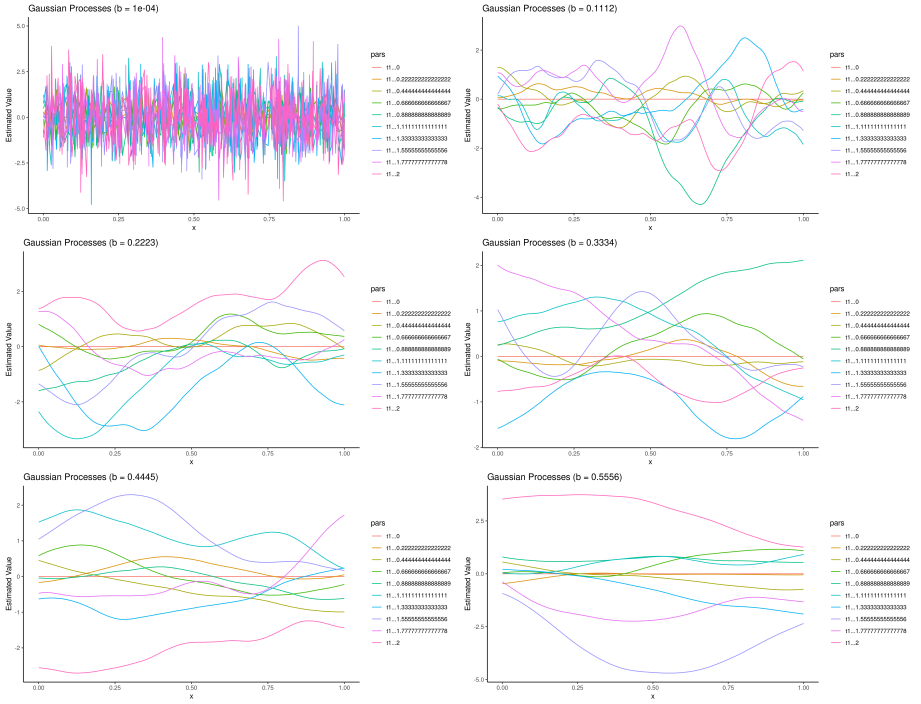

Figure 11: Gaussian Process Simulations with Matern $\frac{5}{2}$ Covariance Function.

B. Suppose you observe the value of a Gaussian process $f \sim \mathbf{GP}(m, C)$ at points $x_1, \ldots, x_N$. What is the conditional distribution of the value of the process at some new point $x^\star$? For the sake of notational ease simply write the value of the (i, j) element of the covariance matrix as $C_{i,j}$, rather than expanding it in terms of a specific covariance function.

Recall that we can obtain the conditional distribution of multivariate Gaussian distributions in the following way. Suppose that

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

Then, the posterior distribution of y_1 conditional on y_2 is given by

$$\begin{aligned} p(y_1|y_2) &\sim \mathcal{N}\left(\mu^*, \Sigma^*\right), \\ \mu^* &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_2 - \mu_2), \\ \Sigma^* &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned}$$

Here, we are given a *new* data point x^*, and we want to predict $f(x^*)$. The corresponding joint distribution of $\mathbf{f} = [f(x_1), ..., f(x_n)]$ and $f(x^*)$ is

$$\begin{bmatrix} f(x^*) \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x^*) \\ m(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} C^* & \tilde{C}^T \\ \tilde{C} & C \end{bmatrix} \right),$$

where

$$\begin{aligned} C &= C(\mathbf{x}, \mathbf{x}), \\ \tilde{C} &= C(\mathbf{x}, x^*), \\ C^* &= C(x^*, x^*) \end{aligned}$$

The corresponding conditional distribution of $f(x^*)$ given $\mathbf{f}$ and $\mathbf{x}$ is

$$f(x^*) \Big| \mathbf{f}, \mathbf{x}, x^* \sim \mathcal{N}\left(m(x^*) + \tilde{C}^T C^{-1}(\mathbf{y} - m(\mathbf{x})),\ C^* - \tilde{C}^T C^{-1} \tilde{C} \right)$$

C. Prove the following lemma.

Lemma 1 ***Suppose that the joint distribution of two vectors y and θ has the following properties: (1) the conditional distribution for y given θ is multivariate normal, $(y \mid \theta) \sim N(R\theta, \Sigma)$; and (2) the marginal distribution of θ is multivariate normal, $\theta \sim N(m, V)$. Assume that R, Σ, m, and V are all constants. Then, the joint distribution of y and θ is multivariate normal.***

We know that we can obtain the joint distribution for $(\boldsymbol{\theta}, \mathbf{y})$ with the following:

$$\begin{aligned} p(\boldsymbol{\theta}, \mathbf{y}) &= p(\mathbf{y}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \\ &\propto \exp\left\{ -\frac{1}{2} \left[\underbrace{(\mathbf{y} - R\boldsymbol{\theta})^T \Sigma^{-1} (\mathbf{y} - R\boldsymbol{\theta}) + (\boldsymbol{\theta} - \mathbf{m})^T V^{-1} (\boldsymbol{\theta} - \mathbf{m})}_{A} \right] \right\} \end{aligned}$$

Focusing on the A term, we can expand it to obtain the following:

$$A \propto \mathbf{y}^T \Sigma^{-1} \mathbf{y} - 2\mathbf{y}^T \Sigma^{-1} R\boldsymbol{\theta} + \boldsymbol{\theta}^T R^T \Sigma^{-1} R\boldsymbol{\theta} + \boldsymbol{\theta}^T V^{-1} \boldsymbol{\theta} - 2m^T V^{-1} \boldsymbol{\theta}$$

From here, we can complete the square to rewrite A in the following matrix form:

$$A \propto \begin{bmatrix} \boldsymbol{\theta} - \mathbf{m} \\ \mathbf{y} - R\mathbf{m} \end{bmatrix}^T \begin{bmatrix} V^{-1} + R^T \Sigma^{-1} R & -R^T \Sigma^{-1} \\ -\Sigma^{-1} R & \Sigma^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} - \mathbf{m} \\ \mathbf{y} - R\mathbf{m} \end{bmatrix}$$

We can multiply this out to verify the matrix form:

$$\begin{aligned} A &\propto (\theta - m)^T (V^{-1} + R^T \Sigma^{-1} R)(\theta - m) - (y - Rm)^T \Sigma^{-1} R(\theta - m) \\ &\quad - (\theta - m)^T R^T \Sigma^{-1} (y - Rm) + (y - Rm)^T \Sigma^{-1} (y - Rm) \\ &\propto \theta^T (V^{-1} + R^T \Sigma^{-1} R)\theta - 2m^T (V^{-1} + R^T \Sigma^{-1} R)\theta \\ &\quad - y^T \Sigma^{-1} R\theta + y^T \Sigma^{-1} Rm + m^T R^T \Sigma^{-1} R\theta \\ &\quad - \theta^T R^T \Sigma^{-1} y + \theta^T R^T \Sigma^{-1} Rm + m^T R^T \Sigma^{-1} y \\ &\quad + y^T \Sigma^{-1} y^T - 2m^T R^T \Sigma^{-1} y \\ &= \theta^T V^{-1} \theta + \theta^T R^T \Sigma^{-1} R\theta - 2m^T V^{-1} \theta - 2m^T R^T \Sigma^{-1} R\theta \\ &\quad - y^T \Sigma^{-1} R\theta + y^T \Sigma^{-1} Rm + m^T R^T \Sigma^{-1} R\theta \\ &\quad - \theta^T R^T \Sigma^{-1} y + \theta^T R^T \Sigma^{-1} Rm + m^T R^T \Sigma^{-1} y \\ &\quad + y^T \Sigma^{-1} y^T - 2m^T R^T \Sigma^{-1} y \end{aligned}$$

Then, the above green and red colored expressions cancel to obtain the original form of A. The blue expressions combine together nicely, as well. While we

found what A is proportional to, we can neglect all those additional summed terms that don't contain $\mathbf{y}$ or $\boldsymbol{\theta}$ since A is in an exponent. Thus, we have that

$$p(\boldsymbol{\theta}, \mathbf{y}) \propto \exp\left\{-\frac{1}{2}A\right\},$$

which indicates that $p(\boldsymbol{\theta}, \mathbf{y})$ is a multivariate normal distribution with

$$\begin{aligned} \text{mean} &= \begin{bmatrix} \mathbf{m} \\ R\mathbf{m} \end{bmatrix}, \\ \text{precision matrix} &= \begin{bmatrix} V^{-1} + R^T\Sigma^{-1}R & -R^T\Sigma^{-1} \\ -\Sigma^{-1}R & \Sigma^{-1} \end{bmatrix} \end{aligned}$$

By the given marginal distribution and conditional distribution, we knew the mean vector would take the above form. The trickier part is knowing the precision matrix. However, once we identify the quadratic terms in the original expression for A, we know the main diagonal elements. The only thing left to find is the off-diagonal elements in the precision matrix. We know that they are tranposes of each other. Additionally, with a keen eye, we may notice that the original expression for A contains the term $-2\mathbf{y}^T\Sigma^{-1}R\boldsymbol{\theta}$. This indicates that we need to have a Σ^{-1} and an R^T in the off-diagonal terms because we otherwise wouldn't be able to get a term with a single $\boldsymbol{\theta}$ and a single R^T. Completing the term specifications, we can note that they must be negative to cancel with the existing terms.

A. Suppose we observe data $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, for some unknown function f. Suppose that the prior distribution for the unknown function is a mean-zero Gaussian process: $f \sim \mathbf{GP}(0, C)$ for some covariance function C. Let $x_1, \ldots, x_N$ denote the previously observed x points. Derive the posterior distribution for the random vector $[f(x_1), \ldots, f(x_N)]^T$, given the corresponding outcomes $y_1, \ldots, y_N$, assuming that you know σ^2.

We can denote

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{f}, \sigma^2 I\right)$$

and obtain the full-conditional distribution for $\mathbf{f}$ just as we have in the past:

$$\begin{aligned}
[\mathbf{f}|\cdot] &\propto [\mathbf{y}|\mathbf{f}, \sigma^2] \cdot [\mathbf{f}] \\
&\propto \exp\left\{-\frac{1}{2}\left[(\mathbf{y}-\mathbf{f})^T(\sigma^2 I)^{-1}(\mathbf{y}-\mathbf{f}) + \mathbf{f}^T C^{-1}\mathbf{f}\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[-2\underbrace{\mathbf{y}^T(\sigma^2 I)^{-1}}_{b^T}\mathbf{f} + \mathbf{f}^T\left(\underbrace{(\sigma^2 I)^{-1} + C^{-1}}_{A}\right)\mathbf{f}\right]\right\}
\end{aligned}$$

Therefore, we can see that

$$[\mathbf{f}|\cdot] \equiv \mathcal{N}\left((I + \sigma^2 C^{-1})^{-1}\mathbf{y}, (\sigma^{-2} I + C^{-1})^{-1}\right)$$

B. As before, suppose we observe data $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, for $i = 1, \ldots, N$. Now we wish to predict the value of the function $f(x^\star)$ at some new point $x^\star$ where we haven't seen previous data. Suppose that f has a mean-zero Gaussian process prior, $f \sim GP(0, C)$. Show that the posterior mean $E\{f(x^\star) \mid y_1, \ldots, y_N\}$ is a linear smoother, and derive expressions both for the smoothing weights and the posterior variance of $f(x^\star)$.

Recall that we can obtain the joint distribution and, consequently, the conditional distribution of $f(x^*)$ given $\mathbf{y}, \mathbf{f}, \sigma^2$ using parts (c) and (b) in the previous section, respectively. Therefore, the expected posterior mean is

$$\begin{aligned} E\left[f(x^*)|\mathbf{y}\right] &= m(x^*) + \tilde{C}^T C^{-1}\left(\mathbf{y} - m(\mathbf{x})\right) \\ &= \tilde{C}^T C^{-1}\mathbf{y} \\ &= \sum_{i=1}^{n} \tilde{C}(x_i, x^*) C^{-1}(x_i, x_i) y_i \\ &= \sum_{i=1}^{n} w_i y_i, \end{aligned}$$

with smoothing weights $w_i = C(x^*, x_i) - C(x^*, \mathbf{x}_{-i})C(x_i, \mathbf{x}_{-i})$.

Additionally, we can write the posterior mean as a linear combination of the kernel function values, which may be interpretted as a correction to the prior mean consisting of a weighted combination of kernel functions (one for each training data point):

$$\begin{aligned} E\left[f(x^*)|\mathbf{y}\right] &= \sum_{i=1}^{n} \alpha_i C(x_i, x^*), \\ \alpha_i &= C^{-1}(\mathbf{x}, \mathbf{x}) y_i \end{aligned}$$

Recall from the aforementioned (b) that the posterior variance of $f(x^*)$ is

$$C^* - \tilde{C}^T C^{-1} \tilde{C}$$

where

$$\begin{aligned} C &= C(\mathbf{x}, \mathbf{x}), \\ \tilde{C} &= C(\mathbf{x}, x^*), \\ C^* &= C(x^*, x^*) \end{aligned}$$

C. Go back to the utilities data, and plot the point-wise posterior mean and 95% posterior confidence interval for the value of the function at each of the observed points x_i (again, superimposed on top of the scatter plot of the data itself). Choose τ_2^2 to be very small, say 10^{-6}, and choose (b, τ_1^2) that give a sensible-looking answer.

To obtain Figure 12, I set $\tau_2^2 = 10^{-6}$ and used a range of values for the other hyperparameters: $b \in \{3, 10, 15\}$ and $\tau_1^2 \in \{1, 5, 10\}$. Note that as b increases, the posterior means and intervals become more smooth; as τ_1^2 increases, we scale the information of weights more.

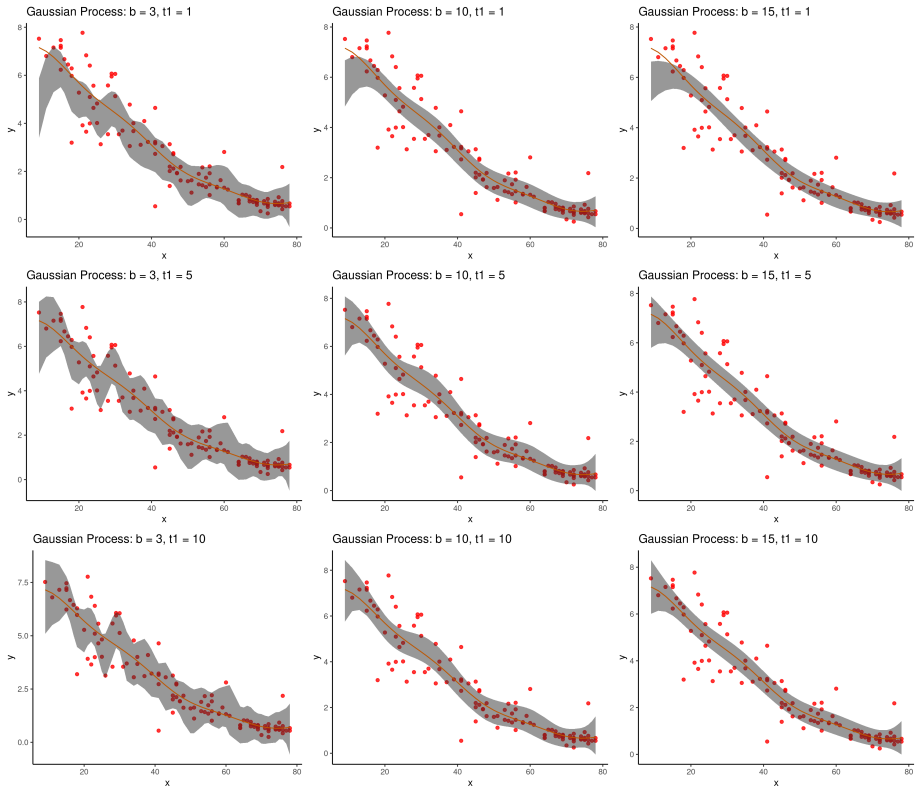

Figure 12: Gaussian Processes for Analyzing Utilities Under Various Hyperparameters.

D. Let $y_i = f(x_i) + \epsilon_i$, and suppose that f has a Gaussian-process prior under the Matern(5/2) covariance function C with scale τ_2^1, range b, and nugget τ_2^2. Derive an expression for the marginal distribution of $y = (y_1 \ldots, y_N)$ in terms of (τ_1^2, b, τ_2^2), integrating out the random function f. This is called a marginal likelihood.

Recall that, if $p(a|b) = \mathcal{N}(a|Ab, S)$ and $p(b) = \mathcal{N}(b|\mu, \Sigma)$, then

$$p(a) = \int p(a|b)p(b)db = \mathcal{N}(a|A\mu, A\Sigma A^T + S)$$

This is due to the convenient properties of multivariate normal distributions. Through applying this result, we see that

$$p(\mathbf{y}) = \mathcal{N}(0, C + \sigma^2 I)$$

E. Return to the utilities or ethanol data sets. Fix $\tau_2^2 = 0$, and evaluate the log of the marginal likelihood function $p(y \mid \tau_1^2, b)$ over a discrete 2-d grid of points. If you're getting errors in your code with $\tau_2^2 = 0$, use something very small instead. Use this plot to choose a set of values $(\hat{\tau_1^2}, \hat{b})$ for the hyperparameters. Then use these hyperparameters to compute the posterior mean for f, given y. Comment on any lingering concerns you have with your fitted model.

We can simply take the logarithm of the distribution in (D):

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \left[|2\pi(C + \sigma^2 I)|^{-1/2} \exp \left\{ -\frac{1}{2} \mathbf{y}^T (C + \sigma^2 I)^{-1} \mathbf{y} \right\} \right] \\
&= -\frac{1}{2} \log |2\pi(C + \sigma^2 I)| - \frac{1}{2} \mathbf{y}^T (C + \sigma^2 I)^{-1} \mathbf{y} \\
&= -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |C + \sigma^2 I| - \frac{1}{2} \mathbf{y}^T (C + \sigma^2 I)^{-1} \mathbf{y}
\end{aligned}$$

On a grid of 250,000 points for (τ_1^2, b) for $0.001 \leq b \leq 100$ and $0.001 \leq \tau_1^2 \leq 100$, the point that maximizes the log-marginal likelihood function is $(\hat{\tau}_1^2, \hat{b}) = (39.67996, 61.52308)$. The posterior mean for $\mathbf{f}|\mathbf{y}$ is depicted in Figure 13.

Although I have not worked with Gaussian processes extensively before, it seems that the optimal b and τ_1^2 are larger than we would want. Perhaps they are so large because the data is *roughly* linear; however, it seems like we may be over-smoothing.

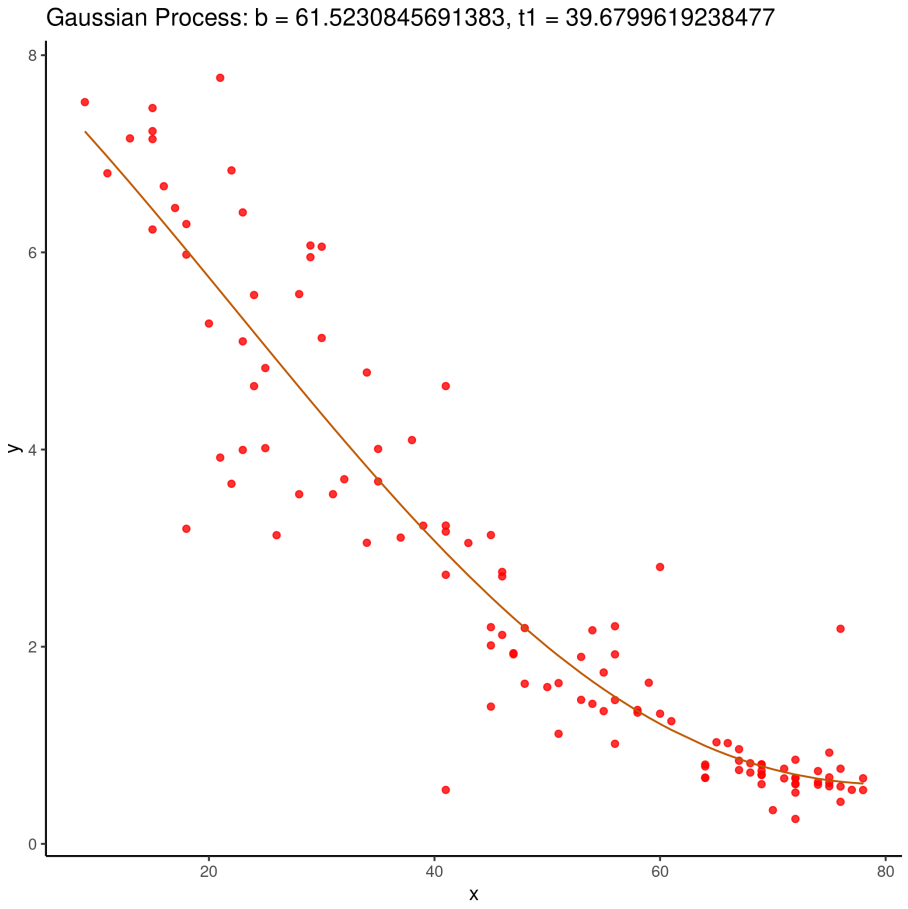

Figure 13: GP's Posterior Mean Under Optimal Hyperparameters.

F. In *weather.csv* you will find data on two variables from 147 weather stations in the American Pacific northwest: (1) pressure is the difference between the forecasted pressure and the actual pressure reading at that station in Pascals; (2) temperature is the difference between the forecasted temperature and the actual temperature reading at that station in Celsius. There are also latitude and longitude coordinates of each station. Fit a Gaussian process model for each of the temperature and pressure variables. Choose hyperparameters appropriately. Visualize your fitted functions (both the posterior mean and posterior standard deviation) on a regular grid using something like a contour plot or color image. Read up on the *image*, *filled.contour*, or *contourplot* functions in R. An important consideration: is Euclidean distance the appropriate measure to go into the covariance function? Or do we need separate length scales for the two dimensions, i.e.

$$d^2(x,z) = \frac{(x_1 - z_1)^2}{b_1^2} + \frac{(x_2 - z_2)^2}{b_2^2}.$$

Justify your reasoning for using Euclidean distance or this "nonisotropic" distance.

I, as well as several others,[1] believe that Euclidean distance is appropriate for one or two dimensions. Once our parameter space gets to higher dimensions, such as $d \geq 3$, Euclidean distance becomes less favorable. Therefore, I stand by our Euclidean distance for this application.

For pressure, Figures 14 and 15 depict the posterior predictive mean and standard deviation and the perspective of the posterior predictive mean at the optimal values $\hat{b}$ and $\hat{\tau}_1^2$, respectively. We obtained optimal values for the hyperparameters in the same way that did in (E), namely evaluating the log-marginal likelihood at a grid of points within the support of the data. The posterior predictive mean and standard deviation surfaces and perspective surface for temperature are depicted in Figures 16 and 17, respectively.

In the heat plots, white indicates highest values, then yellow, then red. For both temperature and pressure, we have very few observations in the lower left of the subplots. Therefore, the grid values in that area are too far from data to adequately learn. The posterior predictive standard deviation is incredibly

[1](1): Domingos, Pedro. "A few useful things to know about machine learning." Communications of the ACM 55.10 (2012): 78-87.
(2): Aggarwal, Charu C., Alexander Hinneburg, and Daniel A. Keim. "On the surprising behavior of distance metrics in high dimensional space." International conference on database theory. Springer, Berlin, Heidelberg, 2001.
(3): Mirkes, Evgeny M., Jeza Allohibi, and Alexander Gorban. "Fractional norms and quasinorms do not help to overcome the curse of dimensionality." Entropy 22.10 (2020): 1105.

high there, indicated by the bright white dot. Similarly, the posterior predictive mean is very low since it cannot borrow information from surrounding observations.

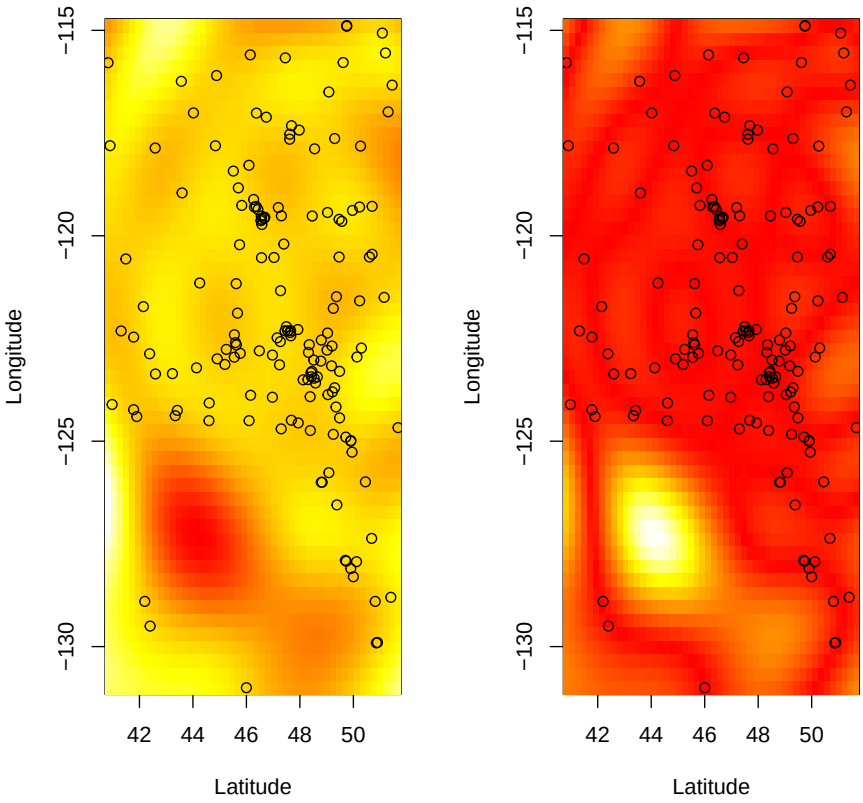

Figure 14: Posterior predictive mean (left) and standard deviation (right) surfaces for pressure. Data points are indicated by open circles.

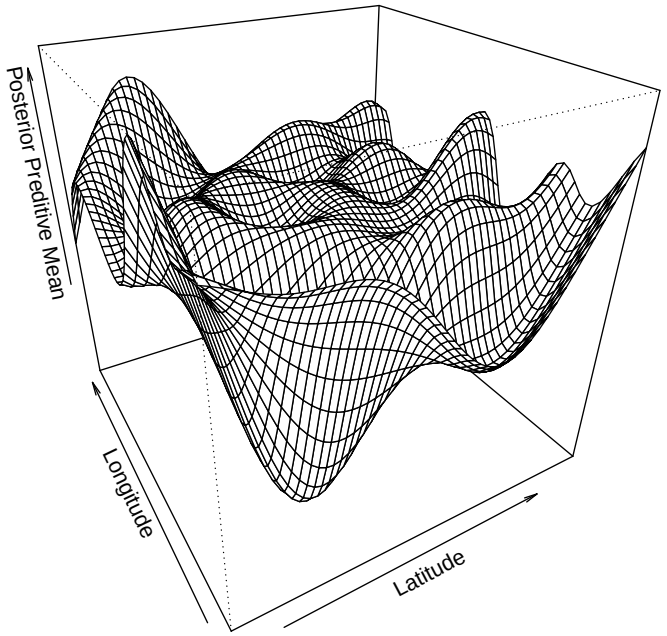

Figure 15: Perspective view on the posterior mean surface for pressure.

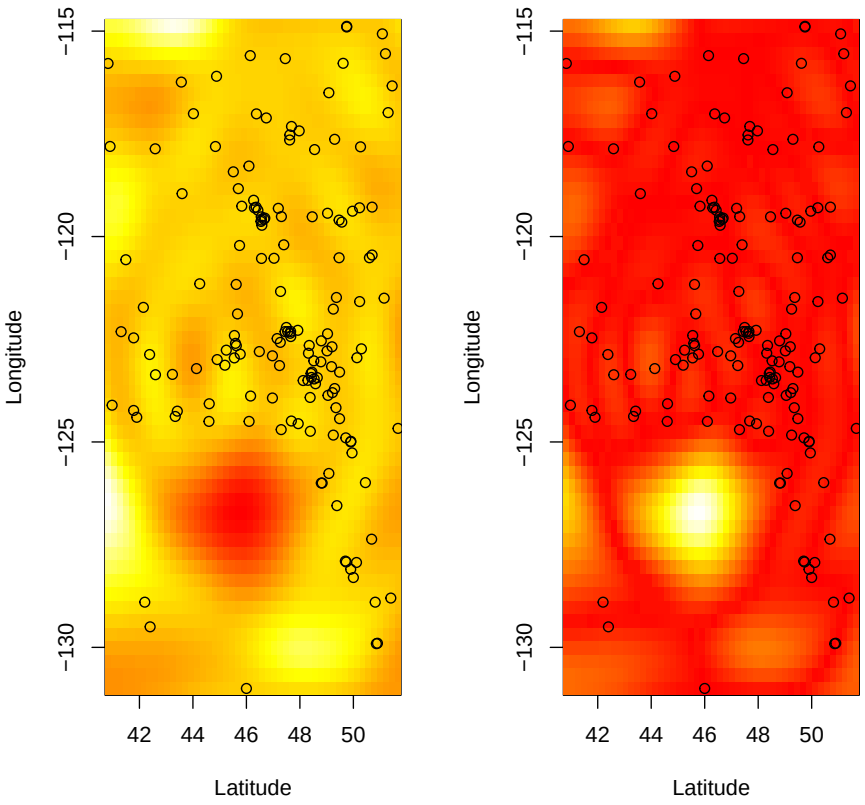

Figure 16: Posterior predictive mean (left) and standard deviation (right) surfaces for temperature. Data points are indicated by open circles.

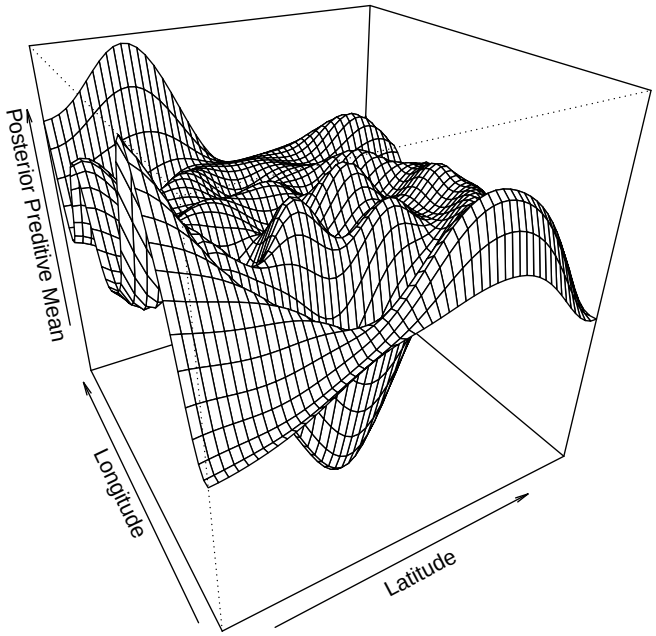

Figure 17: Perspective view on the posterior mean surface for temperature.