Homework 2: Neo Lee

Introduction to Time Series, Fall 2023

Due Thursday September 21 at 5pm

The total number of points possible for this homework is 35. The number of points for each question is written below, and questions marked as "bonus" are optional. Submit the **knitted html file** from this Rmd to Gradescope.

If you collaborated with anybody for this homework, put their names here:

Simple regression

1. (2 pts) Derive the population least squares coefficients, which solve

$$\min_{\beta_1,\beta_0} \mathbb{E}\big[(y-\beta_0-\beta_1 x)^2\big],$$

by differentiating the criterion with respect to each β_j, setting equal to zero, and solving. Repeat the calculation but without intercept (without the β_0 coefficient in the model).

SOLUTION GOES HERE

With intercept:

Define

$$Q := \mathbb{E}\big[(y-\beta_0-\beta_1 x)^2\big].$$

Then we find β_0 by setting

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_0} = \mathbb{E}\big[2(\beta_0+\beta_1 x - y)\big] &= 0 \\
2\mathbb{E}\big[\beta_0\big] + 2\mathbb{E}\big[\beta_1 x\big] - 2\mathbb{E}\big[y\big] &= 0 \\
\beta_0 + \beta_1 \mu_x - \mu_y &= 0 \\
\underline{\beta_0} &= \mu_y - \beta_1 \mu_x.
\end{aligned}$$

Then, we find β_1 by setting

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_1} = \mathbb{E}\big[2x(\beta_0+\beta_1 x - y)\big] &= 0 \\
\beta_0 \mathbb{E}\big[y\big] + \beta_1 \mathbb{E}\big[x^2\big] - \mathbb{E}\big[xy\big] &= 0 \\
\beta_0 \mu_x + \beta_1 \left(Var(x) + \mu_x^2\right) - \left(Cov(x,y) + \mu_x \mu_y\right) &= 0 \\
(\mu_y - \mu_x \beta_1)\mu_x + \beta_1 Var(x) + \beta_1 \mu_x^2 - Cov(x,y) - \mu_x \mu_y &= 0 \\
\beta_1 Var(x) - Cov(x,y) &= 0 \\
\underline{\beta_1} &= \frac{Cov(x,y)}{Var(x)}.
\end{aligned}$$

Without intercept:

Define

$$Q := \mathbb{E}\left[(y-\beta_1 x)^2\right].$$

Then we have

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_1} &= \mathbb{E}\big[2x(\beta_1 x - y)\big] = 0 \\
\beta_1 \mathbb{E}\big[x^2\big] - \mathbb{E}\big[xy\big] &= 0 \\
\beta_1 \left(Var(x) + \mu_x^2\right) - \left(Cov(x,y) + \mu_x \mu_y\right) &= 0 \\
\underline{\beta_1} &= \frac{Cov(x,y) + \mu_x \mu_y}{Var(x) + \mu_x^2}.
\end{aligned}$$

2. (2 pts) As in Q1, now derive the sample least squares coefficients, which solve

$$\min_{\beta_1, \beta_0} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2.$$

Again, repeat the calculation but without intercept (no β_0 in the model).

SOLUTION GOES HERE

With intercept:

Define

$$Q := \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2.$$

To find β_0, we set

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_0} &= \sum 2(\beta_1 x_i + \beta_0 - y_i) = 0 \\
\sum \beta_1 x_i + n\beta_0 - \sum y_i &= 0 \\
\beta_1 \frac{1}{n} \sum x_i + \beta_0 - \frac{1}{n} \sum y_i &= 0 \\
\beta_0 &= \frac{1}{n} \sum y_i - \beta_1 \frac{1}{n} \sum x_i \\
\beta_0 &= \bar{y} - \beta_1 \bar{x}.
\end{aligned}$$

To find β_1, we set

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_0} &= \sum 2x(\beta_1 x_i + \beta_0 - y_i) = 0 \\
\beta_1 \sum x_i^2 + \beta_0 \sum x_i - \sum x_i y_i &= 0 \\
\beta_1 \sum x_i^2 + (\bar{y} - \beta_1 \bar{x}) \sum x_i - \sum x_i y_i &= 0 \\
\beta_1 \sum x_i^2 + \bar{y} \sum x_i - \beta_1 \bar{x} \sum x_i - \sum x_i y_i &= 0 \\
\beta_1 \left(\sum x_i^2 - \bar{x} \sum x_i\right) &= \sum x_i y_i - \bar{y} \sum x_i \\
\beta_1 &= \frac{\sum x_i y_i - \bar{y} \sum x_i}{\sum x_i^2 - \bar{x} \sum x_i}.
\end{aligned}$$

Now, we take apart the numerator and denominator of β_1:

$$\begin{aligned}
\sum x_i y_i - \bar{y}\sum x_i &= \sum (x_i y_i) - \bar{y}\cdot n\bar{x} \\
&= \sum (x_i y_i) - \bar{y}\cdot n\bar{x} - \bar{x}\cdot n\bar{y} + n\bar{x}\bar{y} \\
&= \sum (x_i y_i) - \bar{y}\sum x_i - \bar{x}\sum y_i + n\bar{x}\bar{y} \\
&= \sum (x_i y_i - x_i\bar{y} - \bar{x}y_i + \bar{x}\bar{y}) \\
&= \sum (x_i - \bar{x})(y_i - \bar{y}); \\
\sum x_i^2 - \bar{x}\sum x_i &= \sum x_i^2 - \bar{x}\cdot n\bar{x} \\
&= \sum x_i^2 - \bar{x}\cdot n\bar{x} - \bar{x}\cdot n\bar{x} + \bar{x}\cdot n\bar{x} \\
&= \sum x_i^2 - \bar{x}\sum x_i - \bar{x}\sum x_i + n\bar{x}^2 \\
&= \sum \left(x_i^2 - 2x_i\bar{x} + \bar{x}^2\right) \\
&= \sum (x_i - \bar{x})^2.
\end{aligned}$$

Hence,

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}.$$

Without intercept:

Define

$$Q := \sum_{i=1}^n (y_i - \beta_1 x_i)^2.$$

To find β_1, we set

$$\begin{aligned}
\frac{\partial Q}{\partial \beta_1} &= \sum 2x(\beta_1 x_i - y_i) = 0 \\
\beta_1 \sum x_i^2 - \sum x_i y_i &= 0 \\
\beta_1 &= \frac{\sum x_i y_i}{\sum x_i^2}.
\end{aligned}$$

3. (2 pts) Prove of disprove: in the model without intercept, is the regression coefficient of x on y the inverse of that from the regression of y on x? Answer the question for each of the the population and sample versions.

SOLUTION GOES HERE

Population version:

We are interested in knowing whether $\beta_{y|x} = \beta_{x|y}^{-1}$ in a model without intercept. We can check this by

multiplying $\beta_{y|x}$ and $\beta_{x|y}$ and check if the result is 1.

$$\begin{aligned}\beta_{y|x} \times \beta_{x|y} &= \frac{Cov(x,y) + \mu_x\mu_y}{Var(x) + \mu_x^2} \times \frac{Cov(x,y) + \mu_x\mu_y}{Var(y) + \mu_y^2} \\ &= \frac{\mathbb{E}\left[xy\right]^2}{\mathbb{E}\left[x^2\right]\mathbb{E}\left[y^2\right]} \\ &= \frac{\mathbb{E}\left[(xy)^2\right] - Var(xy)}{\mathbb{E}\left[x^2\right]\mathbb{E}\left[y^2\right]} \\ &= \frac{Cov(x,y) + \mathbb{E}\left[x^2\right]\mathbb{E}\left[y^2\right] - Var(xy)}{\mathbb{E}\left[x^2\right]\mathbb{E}\left[y^2\right]}.\end{aligned}$$

Hence, it is only true when $Cov(x,y) = Var(xy) = 0$, and is not true in general.

Sample version:

Similarly, we proceed by multiplying $\beta_{y|x}$ and $\beta_{x|y}$ and check if the result is 1.

$$\beta_{y|x} \times \beta_{x|y} = \frac{\sum x_i y_i}{\sum x_i^2} \times \frac{\sum x_i y_i}{\sum y_i^2}.$$

We can see easily that the product is not 1 in general. Arbitrarily, take $(1,2)$ for $i = 1$ and $(3,4)$ for $i = 2$, the product is 0.98.

4. (3 pts) Consider the following hypothetical. Let y be the height of a child and x be the height of their parent, and consider a regression of y on x, performed in in a large population. Suppose that we estimate the regression coefficients separately for male and female parents (two separate regressions) and we find that the slope coefficient from the former regression $\hat{\beta}_1^{\text{dad}}$ is smaller than that from the latter $\hat{\beta}_1^{\text{mom}}$. Suppose however that we find (in this same population) the sample correlation between a father's height and their child's height is *larger* than that between a mother's height and their child's height. What is a plausible explanation for what is happening here?

SOLUTION GOES HERE

Consider a model with intercept,

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \times \frac{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}},\end{aligned}$$

which is represented by the sample correlation times the ratio of response sample standard deviations over predictor sample standard deviations.

Therefore, one plausible explanation is that the sample standard deviation of father's height is significantly larger than that of mother's height. Hence, even the correlation between a father's height and their child's height is larger, the standard deviation of father's height may rescale $\hat{\beta}_1^{\text{dad}}$ to a smaller value than $\hat{\beta}_1^{\text{mom}}$.

Multiple regression

5. (2 pts) In class, we claimed that the multiple regression coefficients, with respect to responses y_i and feature vectors $x_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, can be written in two ways: the first is

$$\hat{\beta} = \left(\sum_{i=1}^n x_i x_i^T\right)^{-1} \sum_{i=1}^n x_i y_i.$$

The second is

$$\hat{\beta} = (X^T X)^{-1} X^T y,$$

where $X \in \mathbb{R}^{n \times p}$ is a feature matrix, with i^{th} row x_i, and $y \in \mathbb{R}^n$ is a response vector, with i^{th} component y_i. Prove that these two expressions are equivalent.

SOLUTION GOES HERE

We first show that

$$\left(\sum_{i=1}^{n} x_i x_i^T \right)^{-1} = \left(X^T X \right)^{-1},$$

then we show that

$$\sum_{i=1}^{n} x_i y_i = X^T y.$$

Notation: Denote $x_j^{(i)}$ as the j-th feature of the feature vector $\vec{x}$ at time step i. The subscript is the feature index, and the superscript is the time step index. We rewrite the equation as

$$\left(\sum_{i=1}^{n} x^{(i)} x^{(i)^T} \right)^{-1} = \left(X^T X \right)^{-1}$$

to align with our notation.

We first check $\sum_{i=1}^{n} x^{(i)} x^{(i)^T}$. For each time step i, $x^{(i)} x^{(i)^T}$ will produce a $p \times p$ matrix, denote $\mathcal{P}^{(i)}$, where

$$\mathcal{P}_{k,l}^{(i)} = x_k^{(i)} \cdot x_l^{(i)}.$$

Hence, the summation $\sum_{i=1}^{n} x^{(i)} x^{(i)^T} = \mathcal{P}$ is a $p \times p$ matrix of sum of all $\mathcal{P}^{(i)}$, for which

$$\mathcal{P}_{k,l} = \sum_{i=1}^{n} x_k^{(i)} \cdot x_l^{(i)}.$$

Now, we check $X^T X$, denote $\mathcal{P}^*$. X^T is a $p \times n$ matrix, and X is a $n \times p$ matrix, so indeed their product will produce a $p \times p$ matrix. Notice the k-th row of X^T represents the k-th feature vector of all n time steps, which means

$$X_{k,\cdot}^T = \left[x_k^{(1)}, x_k^{(2)}, \cdots, x_k^{(n)} \right].$$

The l-th column of X represents the l-th feature of all n time steps, which means

$$X_{\cdot,l} = \begin{bmatrix} x_l^{(1)} \\ x_l^{(2)} \\ \vdots \\ x_l^{(n)} \end{bmatrix}.$$

Hence, the k, l-th entry of $X^T X$ is

$$\begin{aligned} \mathcal{P}_{k,l}^* &= X_{k,\cdot}^T X_{\cdot,l} \\ &= x_k^{(1)} x_l^{(1)} + x_k^{(2)} x_l^{(2)} + \cdots + x_k^{(n)} x_l^{(n)} \\ &= \sum_{i=1}^{n} x_k^{(i)} \cdot x_l^{(i)}, \end{aligned}$$

and we can see that $\mathcal{P}^* = \mathcal{P} \Rightarrow \mathcal{P}^{*^{-1}} = \mathcal{P}^{-1}$, which means $\left(\sum_{i=1}^{n} x^{(i)} x^{(i)^T} \right)^{-1} = \left(X^T X \right)^{-1}$.

Next, we show

$$\sum_{i=1}^{n} x_i y_i = X^T y.$$

Again, we rewrite the equation to match our notation:

$$\sum_{i=1}^{n} x^{(i)} y^{(i)} = X^T y.$$

For each i,

$$x^{(i)} y^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix} y^{(i)} = \begin{bmatrix} x_1^{(i)} y^{(i)} \\ x_2^{(i)} y^{(i)} \\ \vdots \\ x_p^{(i)} y^{(i)} \end{bmatrix}$$

Hence, $\sum_{i=1}^{n} x^{(i)} y^{(i)}$ is a $p \times 1$ vector, denote $\vec{v}$, and the k-th entry

$$\vec{v}_k = \sum_{i=1}^{n} x_k^{(i)} y^{(i)}.$$

Now we check $X^T y$. X^T is a $p \times n$ matrix, and y is a $n \times 1$ vector, so indeed their product will produce a $p \times 1$ vector, denote $\vec{v}^*$. Notice the k-th row of X^T represents the k-th feature vector of all n time steps, which means

$$X_{k,\cdot}^T = \left[x_k^{(1)}, x_k^{(2)}, \cdots, x_k^{(n)} \right].$$

y is a vector of all n time steps, which means

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}.$$

Indeed, the k-th entry of $\vec{v}^*$ is

$$\begin{aligned} \vec{v}_k^* &= X_{k,\cdot}^T y \\ &= x_k^{(1)} y^{(1)} + x_k^{(2)} y^{(2)} + \cdots + x_k^{(n)} y^{(n)} \\ &= \sum_{i=1}^{n} x_k^{(i)} y^{(i)}. \end{aligned}$$

Therefore, $\vec{v}^* = \vec{v}$, which means $\sum_{i=1}^{n} x^{(i)} y^{(i)} = X^T y$.

6. (Bonus) Derive the population and sample multiple regression coefficients by solving the corresponding least squares problem (differentiating the criterion with respect to each β_j, setting equal to zero, and solving). For the sample least squares coefficient, deriving either representation in Q5 will be fine.

SOLUTION GOES HERE

Population:

Define

$$Q := \mathbb{E}\left[(y - \beta_0 - x^T \beta)^2 \right].$$

Note: y is a random variable, x is a random vector, β_0 is a scalar, and β is a vector.

To find β_0, we set

$$\begin{aligned}\frac{\partial Q}{\partial \beta_0} &= 2\mathbb{E}\left[(\beta_0 + x^T\beta - y)\right]0 \\ \beta_0 + \mu_x^T\beta - \mu_y &= 0 \\ \underline{\beta_0} &= \mu_y - \mu_x^T\beta.\end{aligned}$$

To find β, we set

$$\begin{aligned}\frac{\partial Q}{\partial \beta} &= 0 \\ 2\mathbb{E}\left[(\beta_0 + x^T\beta - y)x^T\right] &= 0 \qquad (1) \qquad (1)\\ 2\mathbb{E}\left[x(\beta_0 + x^T\beta - y)^T\right] &= 0 \qquad (2) \qquad (2)\\ 2\mathbb{E}\left[x(\beta_0 + x^T\beta - y)\right] &= 0 \qquad (3) \qquad (3)\\ \mathbb{E}\left[x\right]\beta_0 + \mathbb{E}\left[xx^T\right]\beta - \mathbb{E}[xy] &= 0 \\ \mathbb{E}[x]\left(\mathbb{E}[y] - \mathbb{E}\left[x^T\right]\beta\right) + \mathbb{E}\left[xx^T\right]\beta - \mathbb{E}[xy] &= 0 \\ \mathbb{E}[x]\mathbb{E}[y] - \mathbb{E}[x]\mathbb{E}\left[x^T\right]\beta + \mathbb{E}\left[xx^T\right]\beta - \mathbb{E}[xy] &= 0 \\ \left(\mathbb{E}[x]\mathbb{E}\left[x^T\right] - \mathbb{E}\left[xx^T\right]\right)\beta &= \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y] \\ \mathrm{Cov}(x)\beta &= \mathrm{Cov}(x,y) \\ \underline{\beta} &= \mathrm{Cov}(x)^{-1}\mathrm{Cov}(x,y).\end{aligned}$$

Note: from (1) to (2), we take transpose of everything inside the expectation because we want to keep the the expectation as a column vector to follow the tradition. The tranpose only change the representation and does not affect the actual meaning. From (2) to (3), we use the fact that $(\beta_0 + x^T\beta - y)$ is a scalar, and the transpose of a scalar is itself.

Sample:

Define

$$Q := (y - X\beta)^T(y - X\beta).$$

To fine β, we set

$$\begin{aligned}\frac{\partial Q}{\partial \beta} &= 2(y - X\beta)^T(-X) = 0 \qquad (4) \qquad (4)\\ y^TX - \beta^TX^TX &= 0 \qquad (5) \qquad (5)\\ \beta^TX^TX &= y^TX \\ \beta^T &= y^TXX^{-1}\left(X^T\right)^{-1} \\ \beta^T &= y^TX\left(X^TX\right)^{-1} \\ \beta &= \left(\left(X^TX\right)^{-1}\right)^T X^Ty \qquad (6) \qquad (6)\\ \underline{\beta} &= \left(X^TX\right)^{-1}X^Ty.\end{aligned}$$

Note: in (4), we used the chain rule in matrix form. In (5), we used the fact that a transpose of sum is the sum of transposes, then distributed the transpose to each term. In (6), we used the fact the transpose of inverse is the inverse of transpose.

Marginal-multiple connection

7. (1 pts) Consider the simple linear regression of a generic response y_i on a constant predictor $x_i = 1$, $i = 1, \ldots, n$, without intercept. Give the exact form of the sample regression coefficient.

SOLUTION GOES HERE

Reusing the formula we derived from question 2, we have

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}.$$

Taking $x_i = 1$ for all i,

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n y_i}{\sum_{i=1}^n 1} = \frac{\sum_{i=1}^n y_i}{n} = \bar{y}.$$

8. (3 pts) Recall the connection between multiple and marginal regression coefficients, as covered in lecture: the j^{th} multiple regression coefficient can be written in general as

$$\hat{\beta}_j = \frac{(\hat{x}_j^{-j})^T \hat{y}^{-j}}{(\hat{x}_j^{-j})^T \hat{x}_j^{-j}},$$

which we interpret as the simple linear regression coefficient of $\hat{y}^{-j}$ on $\hat{x}^{-j}$. These are y and x_j, respectively, after we regress out the contributions of all other features. (See the lecture notes for the precise details.)

Now note that we can treat a simple linear regression with an intercept term as a multiple regression with two features, with the first feature just equal to the constant 1. Using the above formula, and the answer from Q7, re-derive the expression for the slope in the simple linear model with intercept:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}.$$

SOLUTION GOES HERE

Let's first find $\hat{x}_j^{-j}$ and $\hat{y}^{-j}$.

Notation: Denote x_0 as the constant feature vector, and x_1 as the actual meaningful feature vector. We will take $j = 1$ for the rest of this problem. Any superscript $^{-j}$ will mean dropping x_1, and this is not to be confused with $^{-1}$, the inverse.

$\hat{x}_1^{-j}$ is the residual of regressing x_1 on X^{-j}. Note: X^{-j} is essentially x_0. From question 7, we know that

$$\hat{\beta}_{x_1|x_0} = \bar{x}_1.$$

Hence, the residual of regressing x_1 on X^{-j} is

$$\begin{aligned}
\hat{x}_1^{-j} &= x_1 - x_0 \hat{\beta}_{x_1|x_0} \\
&= x_1 - x_0 \bar{x}_1 \\
&= \begin{bmatrix} x_1^{(1)} - \bar{x}_1 \\ x_1^{(2)} - \bar{x}_1 \\ \vdots \\ x_1^{(n)} - \bar{x}_1 \end{bmatrix}
\end{aligned}$$

$\hat{y}^{-j}$ is the residual of regressing y on X^{-j}. Note: X^{-j} is essentially x_0. From question 7, we know that

$$\hat{\beta}_{y|x_0} = \bar{y}.$$

Hence, the residual of regressing y on X^{-j} is

$$\begin{aligned}\hat{y}^{-j} &= y - x_0\hat{\beta}_{y|x_0} \\ &= y - x_0\bar{y} \\ &= \begin{bmatrix} y^{(1)} - \bar{y} \\ y^{(2)} - \bar{y} \\ \vdots \\ y^{(n)} - \bar{y} \end{bmatrix}\end{aligned}$$

Now, we put everything together.

$$\begin{aligned}\hat{\beta}_{1,(y|x_1)} &= \frac{\left(\hat{x}_j^{-j}\right)^T \hat{y}^{-j}}{\left(\hat{x}_j^{-j}\right)^T \hat{x}_j^{-j}} \\ &= \frac{\sum_{i=1}^n \left(x_1^{(i)} - \bar{x}_1\right)\left(y^{(i)} - \bar{y}\right)}{\sum_{i=1}^n \left(x_1^{(i)} - \bar{x}_1\right)^2}.\end{aligned}$$

After switching back to the standard notation, for which $x_1 = x$, the above equation is equivalently to

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}.$$

Covariance calculations

9. (3 pts) Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be random vectors, and let $A \in \mathbb{R}^{k\times n}$ and $B \in \mathbb{R}^{\ell\times m}$ be fixed matrices. Prove that

$$\mathrm{Cov}(Ax, By) = A\mathrm{Cov}(x, y)B^T. \qquad (1) \tag{7}$$

Prove as a consequence that

$$\mathrm{Cov}(Ax) = A\mathrm{Cov}(x)A^T. \qquad (2)$$

Hint: you may use the rule for covariances of linear combinations (as reviewed in the lecture from week 2, "Measures of dependence and stationarity").

SOLUTION GOES HERE

Note that the covariance matrix between two random vectors x, y can be represented as

$$\mathrm{Cov}(x, y) = \mathbb{E}\left[xy^T\right] - \mathbb{E}\left[x\right]\mathbb{E}\left[y\right]^T.$$

Indeed, the right hand side of the equation is a $\dim x \times \dim y$ matrix, in which the (i, j)-th entry is $\mathbb{E}[x_i y_j] - \mathbb{E}[x_i]\mathbb{E}[y_j] = \mathrm{Cov}(x_i, y_j)$.

Now we see that the left hand side of (1)

$$\begin{aligned}\mathrm{Cov}(Ax, By) &= \mathbb{E}\left[Ax(By)^T\right] - \mathbb{E}[Ax]\mathbb{E}[By]^T \\ &= \mathbb{E}\left[Axy^TB^T\right] - \mathbb{E}[Ax]\left(B\mathbb{E}[y]\right)^T \\ &= A\mathbb{E}\left[xy^T\right]B^T - A\mathbb{E}[x]\mathbb{E}[y]^TB^T.\end{aligned}$$

Note: we used the fact that linearity of expectaion holds for matrix form, which allows $\mathbb{E}[Ax] = A\mathbb{E}[x]$ and $\mathbb{E}[xB] = \mathbb{E}[x]B$

The right hand side of (1)

$$\begin{aligned} A\text{Cov}(x,y)B^T &= A\left(\mathbb{E}\left[xy^T\right] - \mathbb{E}[x]\mathbb{E}[y]^T\right)B^T \\ &= A\mathbb{E}\left[xy^T\right]B^T - A\mathbb{E}[x]\mathbb{E}[y]^T B^T \qquad (\textit{distributive property}) \\ &= \text{Cov}(Ax, By). \end{aligned}$$

From (1), we can derive (2)

$$\begin{aligned} \text{Cov}(Ax) &= \text{Cov}(Ax, Ax) \\ &= A\text{Cov}(x,x)A^T \\ &= A\text{Cov}(x)A^T. \end{aligned}$$

10. (2 pts) Suppose that $y = X\beta + \epsilon$, with X and β fixed, and where ϵ is a vector with white noise entries, with variance σ^2. Use the rule in Q9 to prove that for the sample least squares coefficients, namely, $\hat{\beta} = (X^TX)^{-1}X^Ty$, it holds that

$$\text{Cov}(\hat{\beta}) = \sigma^2(X^TX)^{-1}.$$

SOLUTION GOES HERE

$$\begin{aligned} \text{Cov}(\hat{\beta}) &= \text{Cov}\left(\left(X^TX\right)^{-1}X^Ty\right) \\ &= \left(\left(X^TX\right)^{-1}X^T\right)\text{Cov}(y)\left(\left(X^TX\right)^{-1}X^T\right)^T \\ &= \left(\left(X^TX\right)^{-1}X^T\right)\sigma^2 I\left(X\left(X^TX\right)^{-1}\right) \\ &= \sigma^2\left(\left(X^TX\right)^{-1}X^TX\left(X^TX\right)^{-1}\right) \\ &= \sigma^2\left(X^TX\right)^{-1}. \end{aligned}$$

11. (4 pts) An equivalent way to state the Gauss-Markov theorem is as follows. Under the model from Q10, if $\tilde{\beta}$ is any other unbiased linear estimator of β (where linearity means that $\tilde{\beta} = My$ for a fixed matrix M) then

$$\text{Cov}(\hat{\beta}) \lesssim \text{Cov}(\tilde{\beta})$$

where $\lesssim$ means less than or equal to in the *PSD (positive semidefinite) ordering*. Precisely, $A \lesssim B$ if and only if $B - A$ is a PSD matrix, which recall, means $z^T(B-A)z \geq 0$ for all vectors z. Prove that this is indeed equivalent to the statement of the Gauss-Markov theorem given in lecture.

SOLUTION GOES HERE

According to the Gauss Markov theorem,

$$\begin{aligned} \text{MSE}(a^T\hat{\beta}) &\leq \text{MSE}(a^T\tilde{\beta}) = \text{MSE}(c^Ty) \\ \text{Bias}(a^T\hat{\beta})^2 + \text{Var}(a^T\hat{\beta}) &\leq \text{Bias}(a^T\tilde{\beta})^2 + \text{Var}(a^T\tilde{\beta}) \\ \text{Var}(a^T\hat{\beta}) &\leq \text{Var}(a^T\tilde{\beta}) \\ \text{Cov}(a^T\hat{\beta}) &\leq \text{Cov}(a^T\tilde{\beta}) \\ a\text{Cov}(\hat{\beta})a^T &\leq a\text{Cov}(\tilde{\beta})a^T \\ 0 &\leq a\text{Cov}(\tilde{\beta})a^T - a\text{Cov}(\hat{\beta})a^T \\ 0 &\leq a\left(\text{Cov}(\tilde{\beta}) - \text{Cov}(\hat{\beta})\right)a^T \\ \text{Cov}(\hat{\beta}) &\lesssim \text{Cov}(\tilde{\beta}). \end{aligned}$$

Note that the derivation is true whether reading from top to bottom or bottom to top. Hence, the Gauss Markov theorem is indeed equivalent to the statement that $\text{Cov}(\hat{\beta}) \lesssim \text{Cov}(\tilde{\beta})$.

Cross-validation

12. (3 pts) Recall the R code from lecture that performs time series cross-validation to evaluate the mean absolute error (MAE) of predictions from the linear regression of cardiovascular mortality on 4-week lagged particulate levels. Adapt this to evaluate the MAE of predictions from the regression of cardiovascular mortality on 4-week lagged particulate levels and 4-week lagged temperature (2 features). Fit each regression model using a trailing window of 200 time points (not all past). Plot the predictions, and print the MAE on the plot, following the code from lecture.

    Additionally (all drawn on the same figure), plot the fitted values on the training set. By the training set here, we mean what is also called the "burn-in set" in the lecture notes, and indexed by times 1 through `t0` in the code. The fitted values should come from the initial regression model that is fit to the burn-in set. Print the MAE from the fitted values the training set somewhere on the plot (and label this as "Training MAE" to clearly differentiate it).

```
# CODE GOES HERE

library(tidyverse)
```

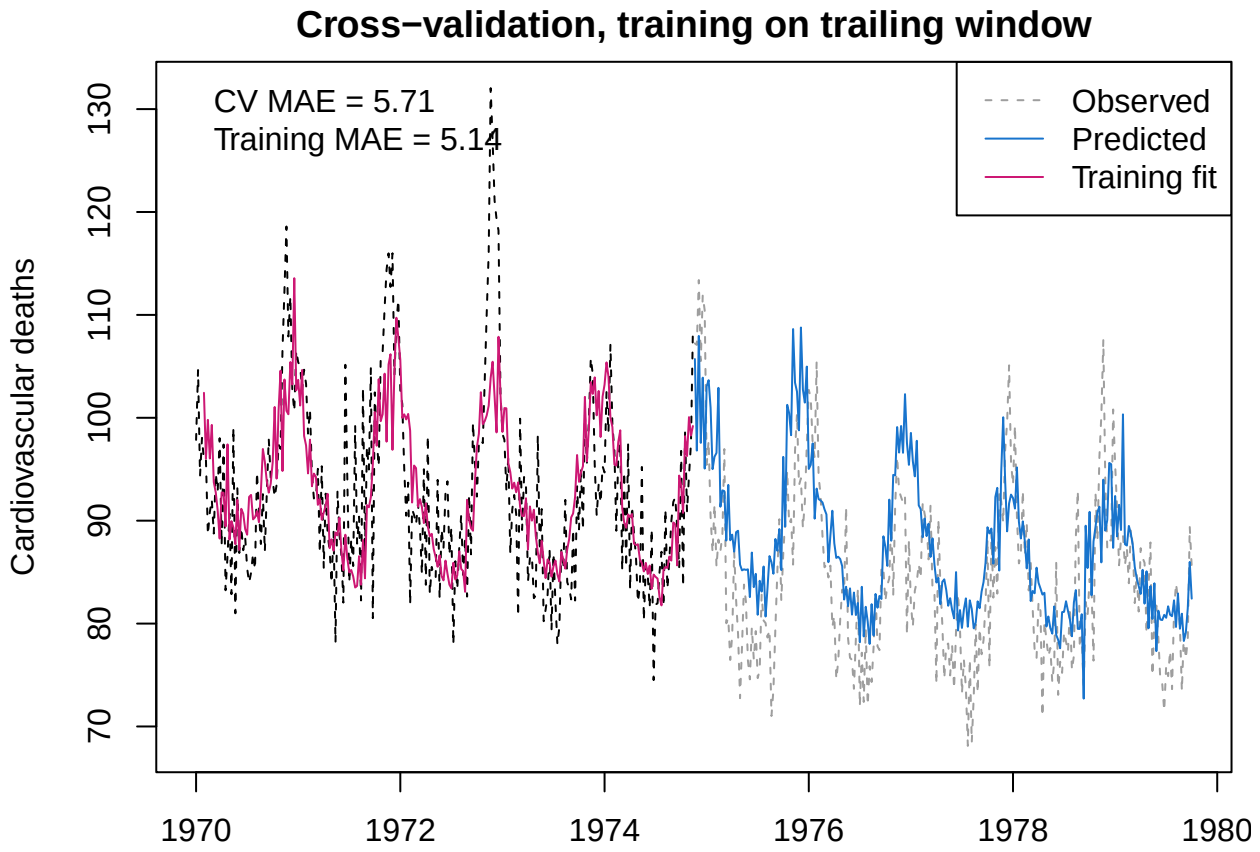
```
## -- Attaching core tidyverse packages ------------------------------------------------------------
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(astsa)
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v tsibble     1.1.3     v fable       0.3.3
## v tsibbledata 0.4.1     v fabletools  0.3.3
## v feasts      0.3.1
## -- Conflicts ------------------------------------------------------------------------------------
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
library(epidatasets)

k = 4 # lag
n = length(cmort)
first_half = 1:n <= floor(n/2)
second_half = 1:n > floor(n / 2)
cmort_vec = as.numeric(cmort)
tempr_vec = as.numeric(tempr)
part_vec = as.numeric(part)

lagged_part_vec = dplyr::lag(part_vec, k)
```

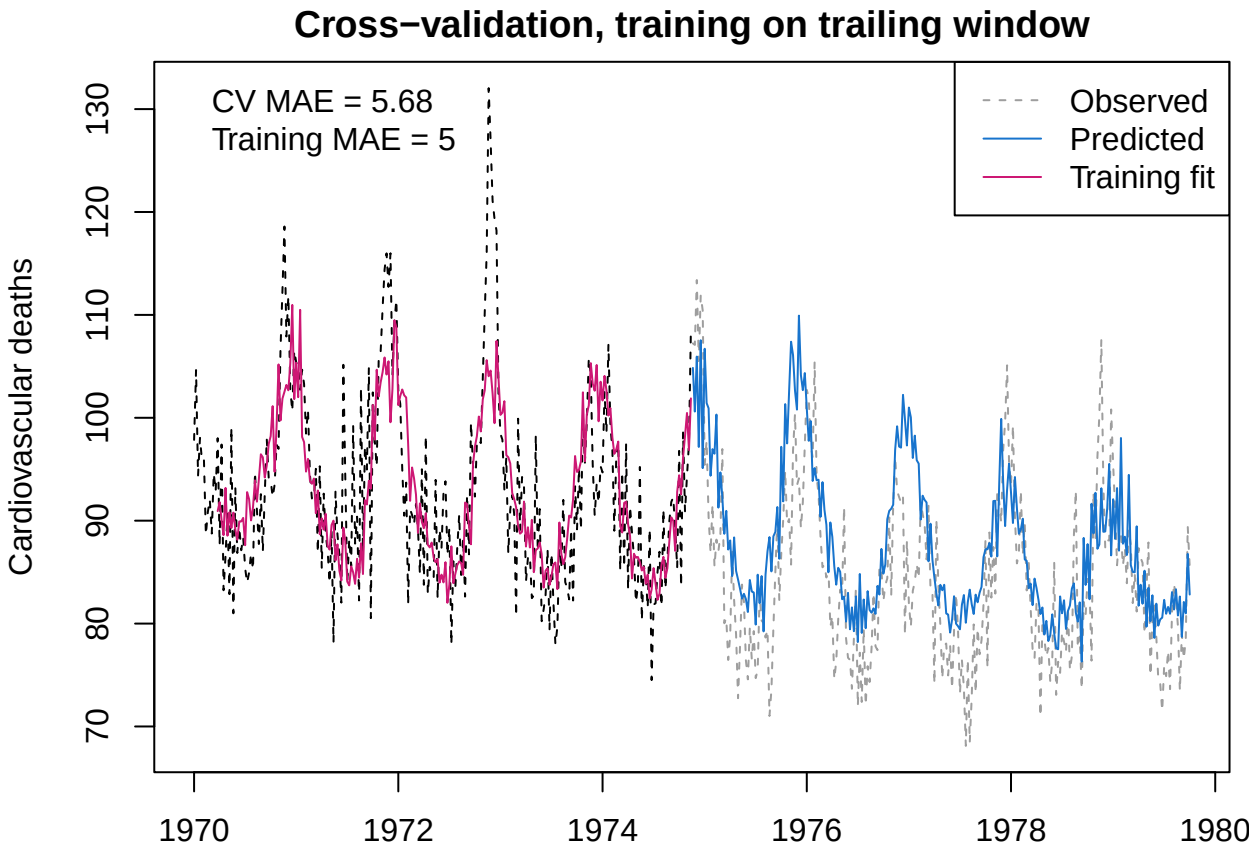
```r
lagged_tempr_vec = dplyr::lag(tempr_vec, k)

t0 = floor(n/2)
yhat_trailing = rep(NA, length = n-t0)
w = 200 # This is our trailing window length

for (t in (t0+1):n) {
  reg_trailing = lm(
    cmort_vec ~ lagged_tempr_vec + lagged_part_vec,
    subset = (1:n) <= t - k & (1:n) > t - w - k
  )
  yhat_trailing[t - t0] = predict(
    reg_trailing,
    newdata = data.frame(
      lagged_tempr_vec = lagged_tempr_vec[t],
      lagged_part_vec = lagged_part_vec[t]
    )
  )
}

mae_trailing = mean(abs(cmort[second_half] - yhat_trailing))

reg_first_half = lm(
  cmort_vec ~ lagged_tempr_vec + lagged_part_vec,
  subset = first_half
)

training_fitted = predict(reg_first_half, newdata = data.frame(
  lagged_tempr_vec = lagged_tempr_vec[first_half],
  lagged_part_vec = lagged_part_vec[first_half]
))

mae_training = mean(abs(cmort[first_half] - training_fitted), na.rm = TRUE)

par(mar = c(2, 4, 2, 0.5))
plot(time(cmort)[first_half], cmort[first_half], type = "l",
     xlim = range(time(cmort)), ylim = range(cmort),
     xlab = "", ylab = "Cardiovascular deaths",
     main = "Cross-validation, training on trailing window", lty = 2)
lines(time(cmort)[second_half], cmort[second_half], col = 8, lty = 2)
lines(time(cmort)[second_half], yhat_trailing, col = 4)
lines(time(cmort)[first_half], training_fitted, col = 6, lty = 1)
legend("topright", legend = c("Observed", "Predicted", "Training fit"),
       lty = c(2, 1, 1), col = c(8, 4, 6))
legend("topleft", legend = c(paste("CV MAE =", round(mae_trailing, 2)),
                             paste("Training MAE =", round(mae_training, 2))),
       bty = "n")
```



13. (2 pts) Repeat the same exercise as in Q12 but now with multiple lags per variable: use lags 4, 8, 12 for each of particulate level and temperature (thus 6 features in total). Did the training MAE go down? Did the cross-validated MAE go down? Discuss. Hint: you may find it useful to know that `lm()` can take a predictor matrix, as in `lm(y ~ x)` where `x` is a matrix; in this problem, you can form the predictor matrix by calling `cbind()` on the lagged feature vectors.

```
# CODE GOES HERE

# Define lag sizes and initialize vectors
lags = c(4, 8, 12)
n = length(cmort)
n_lags = length(lags)
first_half = 1:n <= floor(n/2)
second_half = 1:n > floor(n/2)

cmort_vec = as.numeric(cmort)
tempr_vec = as.numeric(tempr)
part_vec = as.numeric(part)

# Create matrix for all lags
mat = dplyr::lag(part_vec, lags[1])
mat = cbind(mat, dplyr::lag(tempr_vec, lags[1]))

for (i in 2:n_lags) {
  mat = cbind(mat, dplyr::lag(part_vec, lags[i]))
  mat = cbind(mat, dplyr::lag(tempr_vec, lags[i]))
}
```

```r
# Adjust the prediction part
t0 = floor(n/2)
yhat_trailing = rep(NA, n - t0)
w = 200 # Trailing window length

for (t in (t0+1):n) {
  reg_trailing = lm(cmort_vec ~ .,
                    data.frame(mat),
                    subset = (1:n) <= t - 4 & (1:n) > t - w - 4)

  yhat_trailing[t - t0] = predict(
    reg_trailing,
    newdata = data.frame(t(mat[t, ]))
  )
}

mae_trailing = mean(abs(cmort[second_half] - yhat_trailing), na.rm = TRUE)

# Now, for the initial training set
reg_first_half = lm(cmort_vec ~ ., data.frame(mat), subset = first_half)

training_fitted = predict(reg_first_half, newdata = data.frame(mat[first_half, ]))

mae_training = mean(abs(cmort[first_half] - training_fitted), na.rm = TRUE)

# Plot
par(mar = c(2, 4, 2, 0.5))
plot(time(cmort)[first_half], cmort[first_half], type = "l",
     xlim = range(time(cmort)), ylim = range(cmort),
     xlab = "", ylab = "Cardiovascular deaths",
     main = "Cross-validation, training on trailing window", lty = 2)
lines(time(cmort)[second_half], cmort[second_half], col = 8, lty = 2)
lines(time(cmort)[second_half], yhat_trailing, col = 4)
lines(time(cmort)[first_half], training_fitted, col = 6, lty = 1)
legend("topright", legend = c("Observed", "Predicted", "Training fit"),
       lty = c(2, 1, 1), col = c(8, 4, 6))
legend("topleft", legend = c(paste("CV MAE =", round(mae_trailing, 2)),
                             paste("Training MAE =", round(mae_training, 2))),
       bty = "n")
```

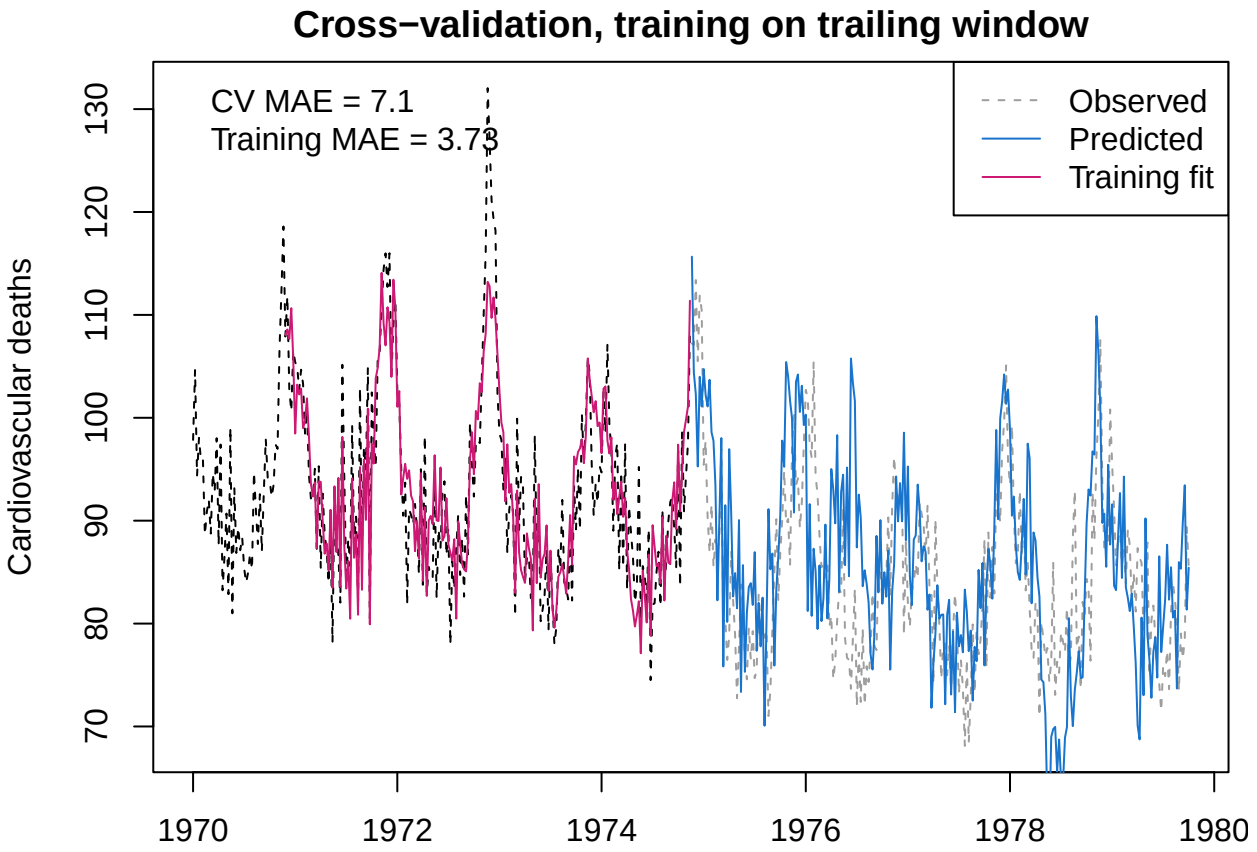


Did the training MAE go down? Did the cross-validated MAE go down? Yes, the training MAE did go down as we added more features, and hence be able to fit the training data better. The cross-validated MAE also went down, but not as much as the training MAE. This is because the cross-validated MAE is measured on the test set, which is not used to train the model.

14. (2 pts) Repeat once more the same exercise as in the last question but but now with many lags per variable: use lags 4, 5, . . . , through 50 for each of particulate level and temperature (thus 47 x 2 = 94 features in total). Did the training MAE go down? Did the cross-validated MAE go down? Are you surprised? Discuss.

```
# CODE GOES HERE

lags = seq(4:50)
n = length(cmort)
n_lags = length(lags)
first_half = 1:n <= floor(n/2)
second_half = 1:n > floor(n/2)

cmort_vec = as.numeric(cmort)
tempr_vec = as.numeric(tempr)
part_vec = as.numeric(part)

# Create matrix for all lags
mat = dplyr::lag(part_vec, lags[1])
mat = cbind(mat, dplyr::lag(tempr_vec, lags[1]))

for (i in 2:n_lags) {
  mat = cbind(mat, dplyr::lag(part_vec, lags[i]))
```

```r
  mat = cbind(mat, dplyr::lag(tempr_vec, lags[i]))
}

# Adjust the prediction part
t0 = floor(n/2)
yhat_trailing = rep(NA, n - t0)
w = 200 # Trailing window length

for (t in (t0+1):n) {
  reg_trailing = lm(cmort_vec ~ .,
                    data.frame(mat),
                    subset = (1:n) <= t - 4 & (1:n) > t - w - 4)

  yhat_trailing[t - t0] = predict(
    reg_trailing,
    newdata = data.frame(t(mat[t, ]))
  )
}

mae_trailing = mean(abs(cmort[second_half] - yhat_trailing), na.rm = TRUE)

# Now, for the initial training set
reg_first_half = lm(cmort_vec ~ ., data.frame(mat), subset = first_half)

training_fitted = predict(reg_first_half, newdata = data.frame(mat[first_half, ]))

mae_training = mean(abs(cmort[first_half] - training_fitted), na.rm = TRUE)

# Plot
par(mar = c(2, 4, 2, 0.5))
plot(time(cmort)[first_half], cmort[first_half], type = "l",
     xlim = range(time(cmort)), ylim = range(cmort),
     xlab = "", ylab = "Cardiovascular deaths",
     main = "Cross-validation, training on trailing window", lty = 2)
lines(time(cmort)[second_half], cmort[second_half], col = 8, lty = 2)
lines(time(cmort)[second_half], yhat_trailing, col = 4)
lines(time(cmort)[first_half], training_fitted, col = 6, lty = 1)
legend("topright", legend = c("Observed", "Predicted", "Training fit"),
       lty = c(2, 1, 1), col = c(8, 4, 6))
legend("topleft", legend = c(paste("CV MAE =", round(mae_trailing, 2)),
                             paste("Training MAE =", round(mae_training, 2))),
       bty = "n")
```



Did the training MAE go down? Did the cross-validated MAE go down? Are you surprised?? Yes the training MAE did go down, which I am not surprised because more features are able to pick up more details in the training data and fit more closely to the training data. The cross-validated MAE did not go down but instead rised significantly. I was surprised in the beginning, but then I realized that the cross-validated MAE is measured on the test set, which is not used to train the model. Hence, the model is overfitting to the training data and does not represent the true relationship between the features and the response. Therefore, when we use the model to predict on the test set, the model does not perform well and the cross-validated MAE is high.

More features, the merrier?

15. (2 pts) Let y_i be an arbitrary response, and $x_i \in^p$ be an arbitrary feature vector, for $i = 1, \ldots, n$. Let

$$\tilde{x}_i = (x_{i1}, \ldots, x_{ip}, \tilde{x}_{i,p+1}), \quad i = 1, \ldots, n$$

be the result of appending one more feature. Let $\hat{y}_i$ denote the fitted values from the regression of y_i on x_i, and let $\tilde{y}_i$ denote the fitted values from the regression of y_i on $\tilde{x}_i$. Prove that

$$\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \leq \sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

In other words, *the training MSE will never get worse as we add features* to a given sample regression problem.

SOLUTION GOES HERE

Assume for the sake of contradiction that the training MSE

$$\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 > \sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

However, notice we can take $\tilde{\beta}$ to be $\hat{\beta}$ and appending 0's to the end of $\tilde{\beta}$ to make it a $p+1$-dimensional vector. Then, $\tilde{y} = X\tilde{\beta} = X\hat{\beta} = \hat{y}$. Hence,

$$\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

which contradicts our assumption. Therefore, the training MSE will always be less than or equal to the original MSE if we add more features to it.

16. (2 pts) How many linearly independent features do we need (how large should p be) in order to achieve a perfect training accuracy, i.e., training MSE of zero? Why?

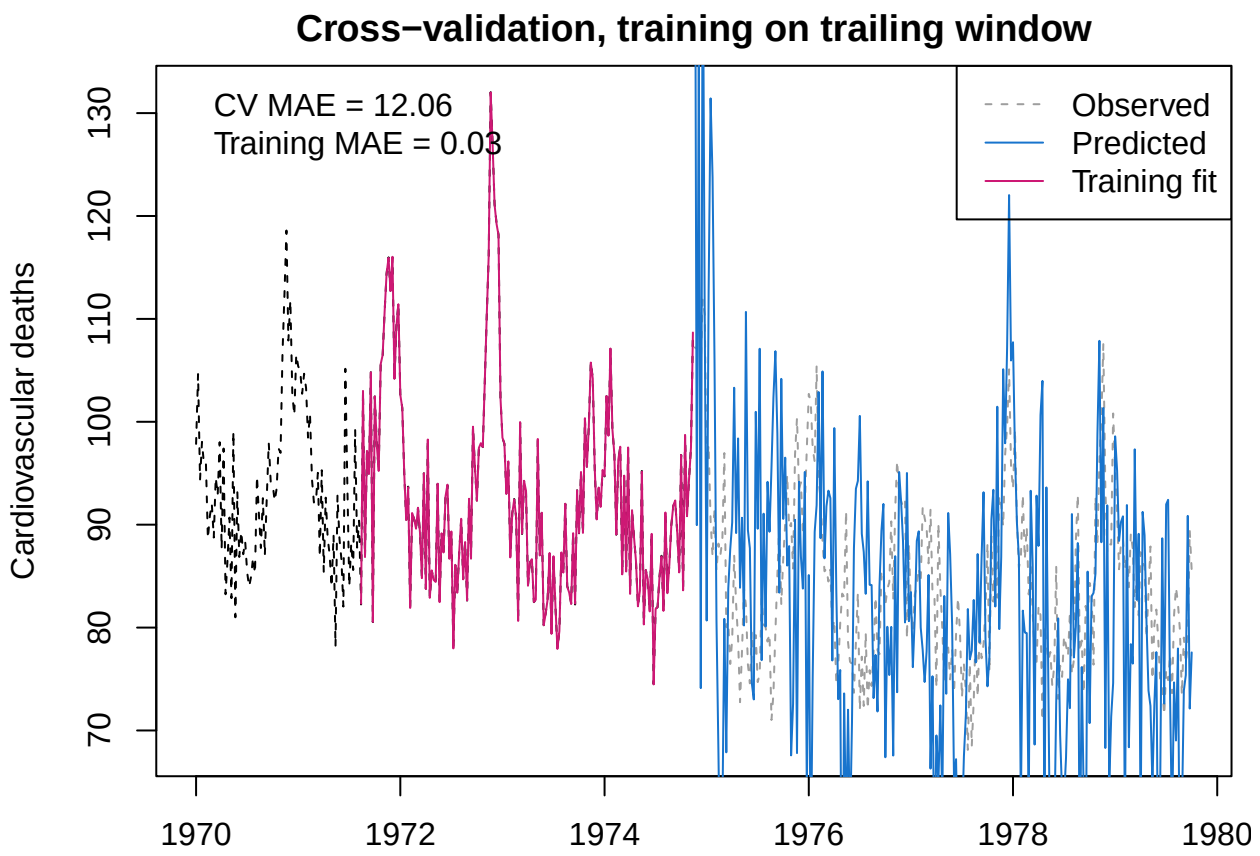
SOLUTION GOES HERE

There need to be n linearly independent features, $p = n$. For MSE to be zero, the ℓ_2 norm $||y - \hat{y}||_2 = 0$, which means $y = \hat{y}$. Note that $\hat{y} = X\hat{\beta}$, a linear combination of the column vectors of X. In order to make sure there exists $\hat{y}$ such that $y = \hat{y}$ for all $y \in \mathbb{R}^n$, the column space of X needs to span the entire space of $\mathbb{R}^n$, which means the column space of X needs to have dimension at least $n = \dim y$. Then, notice there can never be more than n linearly independent features, because the dimension of the column space of X is at most n.

17. (Bonus) Implement an example in R in order to verify your answer to Q16 empirically. Extra bonus points if you do it on the cardiovascular mortality data, using enough lagged features. You should be able to plot the fitted values from the training set and see that they match the observations perfectly (and the CV predictions should be super wild looking).

```
# CODE GOES HERE
n = length(cmort)
first_half = 1:n <= floor(n/2)
second_half = 1:n > floor(n/2)
lags = 1:floor(n/6)
n_lags = length(lags)

cmort_vec = as.numeric(cmort)
tempr_vec = as.numeric(tempr)
part_vec = as.numeric(part)

# Create matrix for all lags
mat = dplyr::lag(part_vec, lags[1])
mat = cbind(mat, dplyr::lag(tempr_vec, lags[1]))

for (i in 2:n_lags) {
  mat = cbind(mat, dplyr::lag(part_vec, lags[i]))
  mat = cbind(mat, dplyr::lag(tempr_vec, lags[i]))
}

# Adjust the prediction part
t0 = floor(n/2)
yhat_trailing = rep(NA, n - t0)
w = 200 # Trailing window length

for (t in (t0+1):n) {
  reg_trailing = lm(cmort_vec ~ .,
                    data.frame(mat),
                    subset = (1:n) <= t - 1 & (1:n) > t - w - 1)

  yhat_trailing[t - t0] = predict(
    reg_trailing,
```

```
    newdata = data.frame(t(mat[t, ]))
  )
}

mae_trailing = mean(abs(cmort[second_half] - yhat_trailing), na.rm = TRUE)

# Now, for the initial training set
reg_first_half = lm(cmort_vec ~ ., data.frame(mat), subset = first_half)

training_fitted = predict(reg_first_half, newdata = data.frame(mat[first_half, ]))

mae_training = mean(abs(cmort[first_half] - training_fitted), na.rm = TRUE)

# Plot
par(mar = c(2, 4, 2, 0.5))
plot(time(cmort)[first_half], cmort[first_half], type = "l",
     xlim = range(time(cmort)), ylim = range(cmort),
     xlab = "", ylab = "Cardiovascular deaths",
     main = "Cross-validation, training on trailing window", lty = 2)
lines(time(cmort)[second_half], cmort[second_half], col = 8, lty = 2)
lines(time(cmort)[second_half], yhat_trailing, col = 4)
lines(time(cmort)[first_half], training_fitted, col = 6, lty = 1)
legend("topright", legend = c("Observed", "Predicted", "Training fit"),
       lty = c(2, 1, 1), col = c(8, 4, 6))
legend("topleft", legend = c(paste("CV MAE =", round(mae_trailing, 2)),
                             paste("Training MAE =", round(mae_training, 2))),
       bty = "n")
```


Cross−validation, training on trailing window
CV MAE = 12.06
Training MAE = 0.03
Observed
Predicted
Training fit
Cardiovascular deaths
70
80
90
100
110
120
130
1970
1972
1974
1976
1978
1980