4.2 Cross Validation

1. Introduction

The test error can be easily calculated if a designated test set is available. Unfortunately, this is usually not the case.

On the contrary, *the training error can be easily calculated on the training dataset*; however, this can dramatically underestimate the test error.

Cross-Validation is a popular method for estimating the average test error Err.

Later we will see methods that make a *mathematical adjustment* to the training error rate, to estimate the test error rate. Here, we analyse methods based on **resampling**.

2. The validation set approach

Suppose we take the dataset at hand and randomly divide it into two parts

- Training set
- Validation set

The model is fit on the training set, and then used to predict the validation set.

The resulting validation set error rate provides an estimate of the test error rate.

Consider the following example using the **`cheddar`** dataset from the **`faraway`** package in **R**.

We fit multiple linear models including one or more of the following covariates **`Acetic`** (ace), **`H2S`** (h2s) and **`Lactic`** (lac) and calculate the corresponding MSEs by randomly dividing the dataset into two parts.

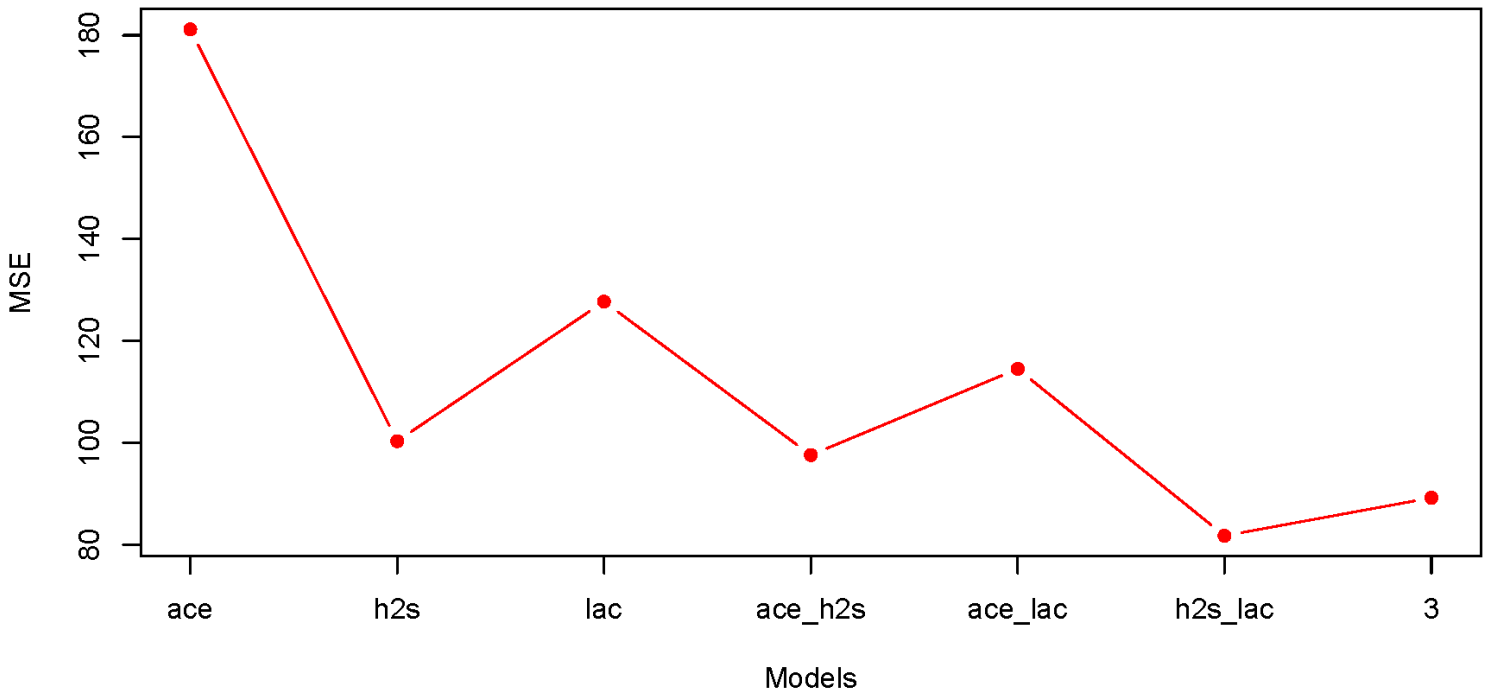

Figure 4.2.1: MSE values obtained for the **`cheddar`** dataset in order to estimate the test errors.

In the graph above, it is possible to obtain different values of MSE (here, the dataset is pretty small and the models' performance is stable, but this is not true in general).

The **validation set approach** is easy both to implement and conceptually, but it has two main drawbacks:

- The validation estimate of the test error rate can be very variable
- Only a part of the observations is used to fit the model, i.e. the sample size is much smaller than the actual sample size

3. Leave-one-out cross-validation

One of the approaches to estimate the minimum test MSE using the training data is **Cross-Validation (CV)**.

CV directly estimates out of sample predictive performance, and for many machine learning methods is the only method available for the performance comparison of different models. CV is a very general method and can be used with any type of predictive modelling.

Let's consider the **squared error loss**.

Leave-One-Out Cross-Validation (LOO-CV) is closely related to the validation set approach: it involved splitting the dataset into two parts:

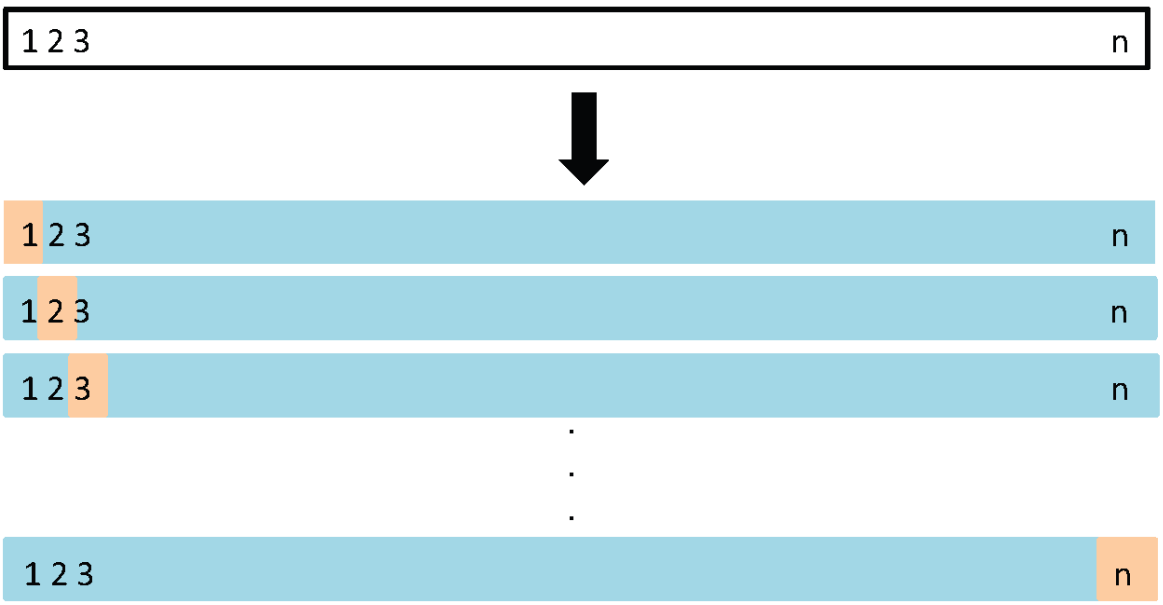

Figure 4.2.2: Schematic display of LOOCV.

Step 1:

- Split $\{(x_1, y_1)\}$ and $\{(x_2, y_2), \ldots, (x_N, y_N)\}$
- The statistical method is fit on the $N - 1$ observations of the training set
- The prediction $\hat{y}_1$ is compared on the single testing observation y_1
- Compute $\mathrm{MSE}_1 = (y_1 - \hat{y}_1)^2$

Step 2:

- Split $\{(x_2, y_2)\}$ and $\{(x_1, y_1), (x_3, y_3), \ldots, (x_N, y_N)\}$
- The statistical method is fit on the $N - 1$ observations of the training set

- The prediction $\hat{y}_2$ is compared on the single testing observation y_2
- Compute $\text{MSE}_2 = (y_2 - \hat{y}_2)^2$

Eventually, a vector $\text{MSE}_1, \text{MSE}_2, \ldots, \text{MSE}_N$ is produced and the **LOO-CV estimate of the test MSE** is

$$\text{CV}_{(N)} = \frac{1}{N} \sum_{i=1}^{N} \text{MSE}_i \tag{4.2.1}$$

The advantages of LOO-CV are

- It has less bias than the validation set approach because at each iteration $N - 1$ observations are used to fit the model
- The randomness due to how the dataset is split vanishes

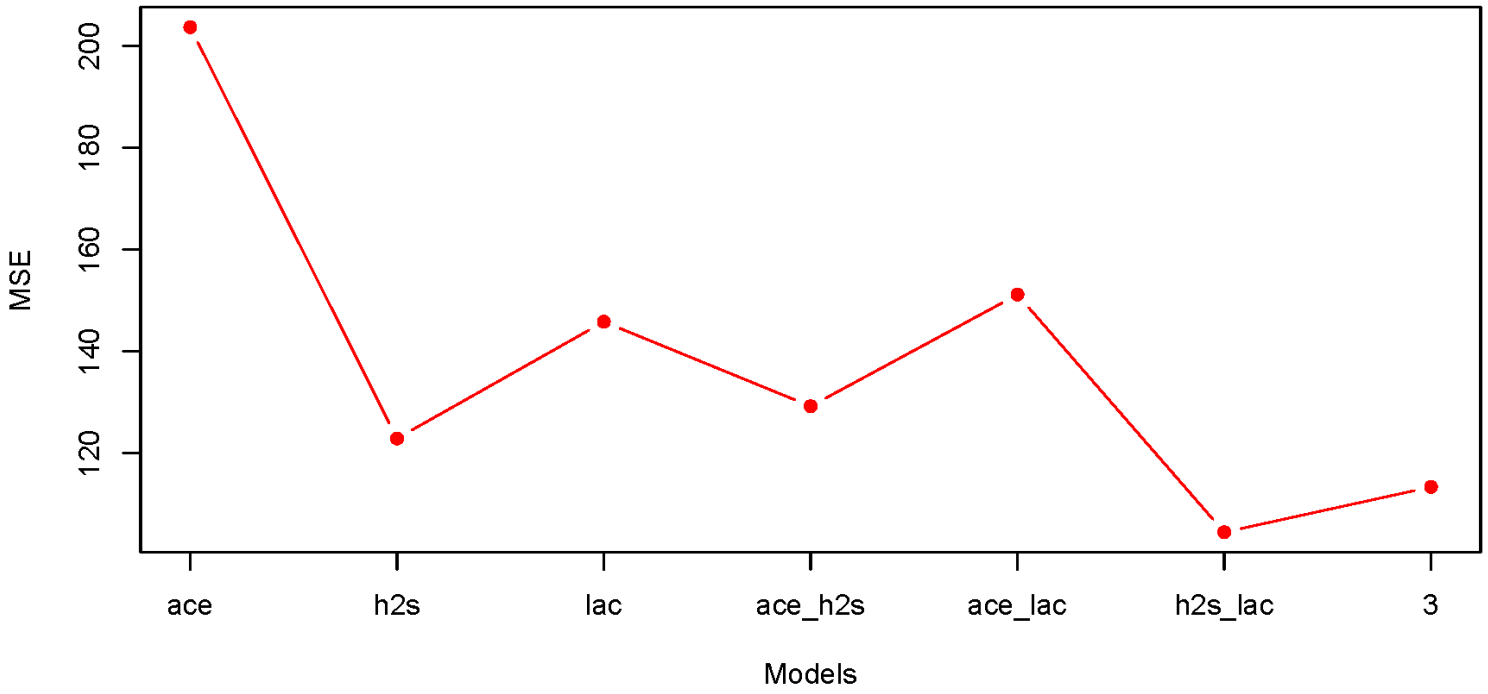

Figure 4.2.3: LOOCV error curves obtained for the **cheddar** dataset in order to estimate the test errors.

The R code to reproduce the above plot is as follows.

```
library(faraway)
data("cheddar")

N <- nrow(cheddar)
MSE_ace <- MSE_h2s <- MSE_lac <- MSE_3 <- vector(length=N)
MSE_ace_h2s <- MSE_ace_lac <- MSE_h2s_lac <- Mean_3 <- vector(length=N)

for(i in 1:N){
  train_set <- cheddar[-i,]
  vali_set <- cheddar[i,]
```

```
  res_lm_ace <- lm(taste ~ Acetic, data=train_set)
  res_lm_h2s <- lm(taste ~ H2S, data=train_set)
  res_lm_lac <- lm(taste ~ Lactic, data=train_set)
  res_lm2_ace_h2s <- lm(taste ~ Acetic + H2S, data=train_set)
  res_lm2_ace_lac <- lm(taste ~ Acetic + Lactic, data=train_set)
  res_lm2_h2s_lac <- lm(taste ~ H2S + Lactic, data=train_set)
  res_lm3_train <- lm(taste ~ Acetic + H2S + Lactic, data=train_set)

  pred_ace <- predict(res_lm_ace,vali_set)
  pred_h2s <- predict(res_lm_h2s,vali_set)
  pred_lac <- predict(res_lm_lac,vali_set)

  pred_ace_h2s <- predict(res_lm2_ace_h2s,vali_set)
  pred_ace_lac <- predict(res_lm2_ace_lac,vali_set)
  pred_h2s_lac <- predict(res_lm2_h2s_lac,vali_set)

  pred_lm3 <- predict(res_lm3_train,vali_set)

  MSE_ace[i] <- (vali_set[,1] - pred_ace)^2
  MSE_h2s[i] <- (vali_set[,1] - pred_h2s)^2
  MSE_lac[i] <- (vali_set[,1] - pred_lac)^2

  MSE_ace_h2s[i] <- (vali_set[,1] - pred_ace_h2s)^2
  MSE_ace_lac[i] <- (vali_set[,1] - pred_ace_lac)^2
  MSE_h2s_lac[i] <- (vali_set[,1] - pred_h2s_lac)^2

  MSE_3[i] <- (vali_set[,1] - pred_lm3)^2
}

models <- c("ace", "h2s", "lac", "ace_h2s", "ace_lac","h2s_lac", "3")
plot(c(mean(MSE_ace), mean(MSE_h2s), mean(MSE_lac),
       mean(MSE_ace_h2s), mean(MSE_ace_lac), mean(MSE_h2s_lac),
       mean(MSE_3)),
       xaxt = "n", xlab="Models", ylab="MSE", col="red", type="b", pch=16)
axis(1, at=1:7, labels=models)
```

LOO-CV has the drawback that it is potentially expensive to implement since the model has to be fitted N times.

For **least squares regression**, the solution has the form $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathrm{X}^\top \mathbf{y}$ and the predicted values are $\hat{\mathbf{y}} = \mathbf{X}^\top \hat{\boldsymbol{\beta}} = \mathbf{H}\mathbf{y}$ where $\mathbf{H}$ is the projection matrix.

Therefore, the CV score can be written

$$\mathrm{CV}_{(N)} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2 \tag{4.2.2}$$

where h_i is the leverage, i.e. the i-th element on the diagonal of $\mathbf{H}$. (A full proof here.)

This is the same as the ordinary MSE, except that each residual is divided by $1 - h_i$. **Recall:** the leverage represents the amount that an observation influences its own fit.

Practice: Write down some R code to reproduce Figure $4.2.2$ using $(4.2.2)$

4. m-fold cross-validation

In m-fold CV, the data is randomly split into m roughly equal parts:

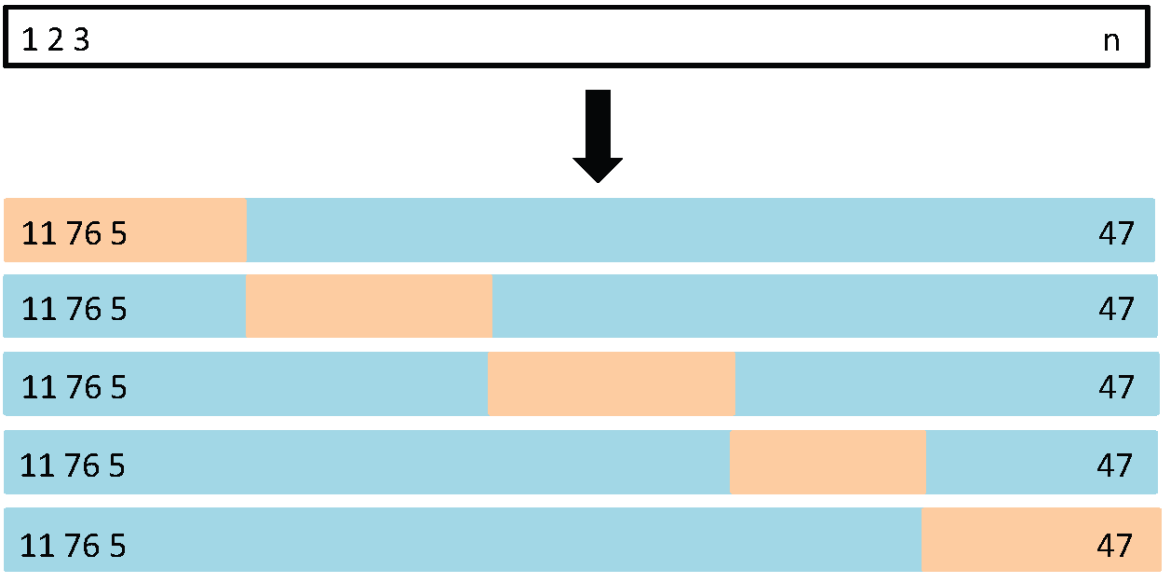

Figure 4.2.4: Schematic display of a 5-fold cross validation.

Consider the m equal sub-samples:

$$\{y_1^1,\ldots y_{n_1}^1\} \quad \ldots \quad \{y_1^m,\ldots,y_{n_m}^m\}$$

where $n_1,\ldots,n_m$ are approximate of the same size.

Write x_j^i for the vector of predictor values corresponding to the response y_j^i, and

$$y_{-i} = \{y_j^k : k \neq i, j = 1, ..., n_k\}$$

for the set of responses obtained by excluding from the full set of responses the ith of the m sub-samples.

Now let $\hat{y}_{-i}(x_j^k)$ be the prediction at x_j^k obtained by fitting to the responses y_{-i}.

For m**-fold cross-validation**, compute

$$\mathrm{CV}_{(m)} := \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i - \hat{y}_{-i}(x_j^i))^2.$$

A 5-step summary of the m-fold Cross-Validation is

1. Leave out the ith subsample

2. Fit the model on the remaining dataset
3. Predict the left-out observations
4. Sum up the squared prediction errors
5. Repeat this for every subsample
6. At the end, average the prediction errors.

The LOO-CV is a **special case** of m-fold CV, where $m = N$.

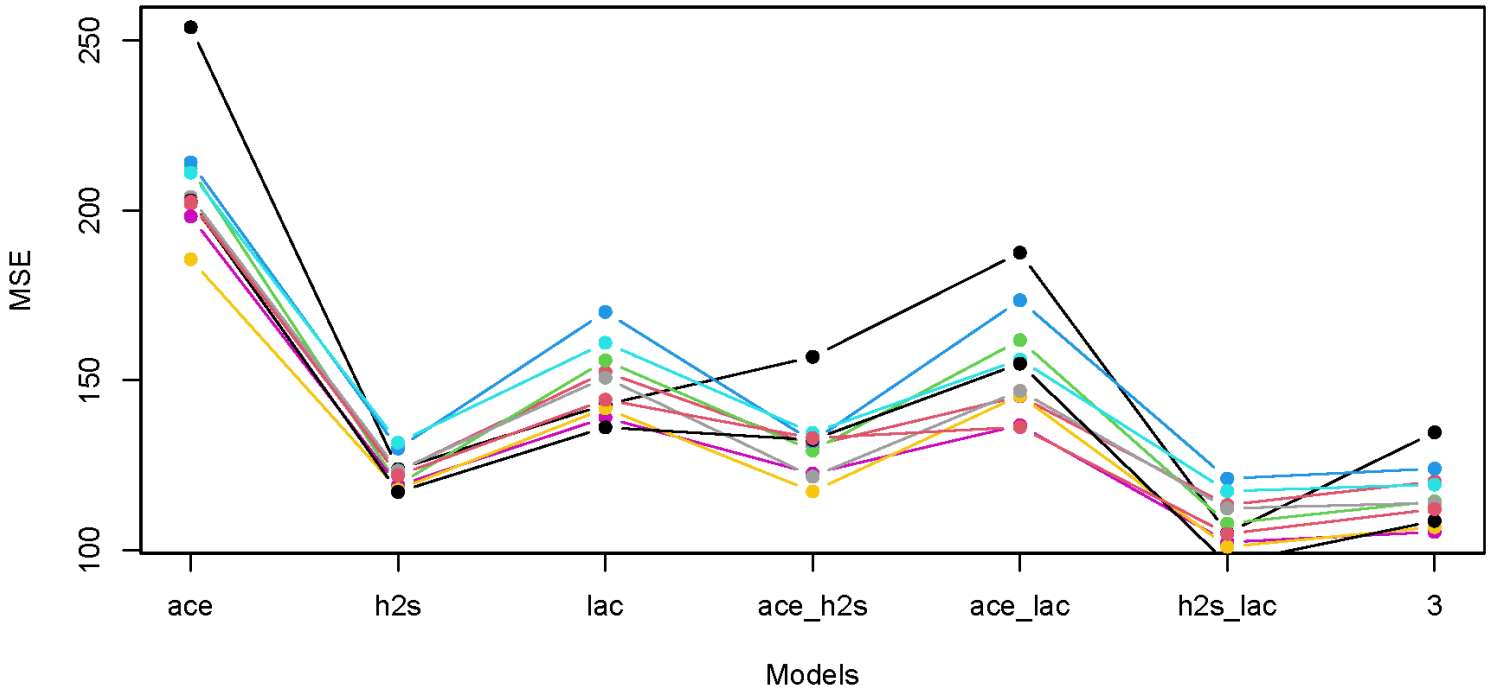

Figure 4.2.5: 10 repetitions of a 5-fold CV error curves obtained for the **cheddar** dataset in order to estimate the test errors.

The R code to reproduce the above plot is as follows.

```
library(faraway)
data("cheddar")

N <- nrow(cheddar)
m <- 5
nrep <- 10
MSE_mCV_ace <- MSE_mCV_h2s <- MSE_mCV_lac  <- matrix(nrow=nrep, ncol=m)
MSE_mCV_ace_h2s <- MSE_mCV_ace_lac <- matrix(nrow=nrep, ncol=m)
MSE_mCV_h2s_lac <- MSE_mCV_3 <- matrix(nrow=nrep, ncol=m)

for(i in 1:nrep){

  set.seed(122+i)
  shuffle_id <- sample(1:N, size=N, replace=F)
  cheddar_new <- cheddar[shuffle_id,]
  seq_id <- c(round(seq(0,N,by=N/m),0))

  for(j in 1:m){
    test_id <- (seq_id[j]+1):seq_id[j+1]
```

```
    train_set <- cheddar_new[-test_id,]
    vali_set <- cheddar_new[test_id,]

    res_lm_ace <- lm(taste ~ Acetic, data=train_set)
    res_lm_h2s <- lm(taste ~ H2S, data=train_set)
    res_lm_lac <- lm(taste ~ Lactic, data=train_set)
    res_lm2_ace_h2s <- lm(taste ~ Acetic + H2S, data=train_set)
    res_lm2_ace_lac <- lm(taste ~ Acetic + Lactic, data=train_set)
    res_lm2_h2s_lac <- lm(taste ~ H2S + Lactic, data=train_set)
    res_lm3_train <- lm(taste ~ Acetic + H2S + Lactic, data=train_set)

    pred_ace <- predict(res_lm_ace,vali_set)
    pred_h2s <- predict(res_lm_h2s,vali_set)
    pred_lac <- predict(res_lm_lac,vali_set)

    pred_ace_h2s <- predict(res_lm2_ace_h2s,vali_set)
    pred_ace_lac <- predict(res_lm2_ace_lac,vali_set)
    pred_h2s_lac <- predict(res_lm2_h2s_lac,vali_set)

    pred_lm3 <- predict(res_lm3_train,vali_set)

    MSE_mCV_ace[i,j] <- mean((vali_set[,1] - pred_ace)^2)
    MSE_mCV_h2s[i,j] <- mean((vali_set[,1] - pred_h2s)^2)
    MSE_mCV_lac[i,j] <- mean((vali_set[,1] - pred_lac)^2)

    MSE_mCV_ace_h2s[i,j] <- mean((vali_set[,1] - pred_ace_h2s)^2)
    MSE_mCV_ace_lac[i,j] <- mean((vali_set[,1] - pred_ace_lac)^2)
    MSE_mCV_h2s_lac[i,j] <- mean((vali_set[,1] - pred_h2s_lac)^2)

    MSE_mCV_3[i,j] <- mean((vali_set[,1] - pred_lm3)^2)

  }
  vals <- c(mean(MSE_mCV_ace[i,]), mean(MSE_mCV_h2s[i,]),
            mean(MSE_mCV_lac[i,]), mean(MSE_mCV_ace_h2s[i,]),
            mean(MSE_mCV_ace_lac[i,]), mean(MSE_mCV_h2s_lac[i,]),
            mean(MSE_mCV_3[i,]))
  if(i==1){
    models <- c("ace", "h2s", "lac", "ace_h2s", "ace_lac","h2s_lac", "3")
    plot(vals, xaxt="n", xlab="Models", ylab="MSE", col=i, type="b",
         pch=16)
    axis(1, at=1:7, labels=models)
  }else{
    points(vals, col=i, type="b", pch=16)
  }
}
```

Advantages:

- **Computational:** LOO-CV implies fitting the model N times, while m-fold CV implies fitting the model only m times
- Advantages for the bias-variance trade-off

LOO-CV gives **almost unbiased estimates** of the coefficients, since the model is fitted with $N - 1$

observations, while m-fold CV introduces **more bias**, because the model is fitted with $\frac{(m-1)N}{m}$ observations.

However, when we perform LOO-CV, the model is fitted N times on almost identical sets of observations, that means that each set is **highly correlated** with the others. For m-fold CV, the output of m fitted models are averaged and the sets are **less correlated**. The output of highly correlated variables can have higher variance than the mean of many variables less correlated.

To summarise, *m-fold CV tends to produce outputs which are less variable than LOO-CV*.

5. Activity in R: Cross-Validation

Explore the use of the $cv.glm()$ function from the **`boot`** library for estimation of the CV error in the 10-fold cross-validation using the Sydney maximum temperature (**`mos.df`**) dataset.

6. Activity

Consider the `Auto` dataset from the `ISLR` package in `R`. We want to use cross-validation in order to estimate the test error that results from predicting `mpg` using polynomial functions of `horsepower`. We consider polynomials of order up to 7.

1. Use the validation method 10 times, each time using a different equal (random) split of the observations into training and validation sets. Plot the resulting estimated test MSEs.
2. Compute and plot the LOOCV error curve (as function of the polynomial degree).
3. Run a 10-fold CV 10 times, each with different splits of the data. Plot the resulting CV error curves.
4. Reflect on 1. and 3. above and draw some conclusion about what you observe.

For reproducibility we set a seed that is equal to the repetition number ($\texttt{set.seed}(1), \ldots, \texttt{set.seed}(10)$).

7. Additional Activity

Question 1 *Submitted Mar 16th 2023 at 11:09:23 pm*

Choose the answer that best describes the aim of cross-validation.

- ○ split the data into training and testing datasets;
- ○ choose the best model using the best subset selection method;
- ◉ estimate the minimum test MSE.

Question 2 *Submitted Mar 16th 2023 at 11:09:30 pm*

Which of the following steps is not a valid m-fold cross-validation step?

- ○ randomly split data into m roughly equal parts;
- ○ leave out the ith subsample;
- ◉ fit model to the ith subsample and obtain predictions for the remaining subsamples;

Question 3 *Submitted Mar 16th 2023 at 11:09:52 pm*

Which of the following are true about cross-validation?

- ☑ Cross-validation can be used for model selection.
- ☑ Cross-validation estimates model's predictive performance.
- ☑ For m-fold cross-validation, choosing m between 5 and 10 shows a good trade-off between bias and variance of the generalisation error.