Lab 7: Implementing Cross Validation (CV)

YOUR NAME HERE

Getting Started

1. Download the `.qmd` file from Moodle and any needed `.xlsx` or `.csv` data files. Save these in the *same folder/directory*.
2. Open the Quarto file in RStudio: `File > Open File... >`. If you're working on the MHC RStudio server, you need to upload the files first: go to the `Files` panel, then click `Upload`. Upload the `.qmd` file and any data files. You will need to upload each file one at a time.
3. Update the author and date in the YAML header of this file.
4. Click the `Render` button. If successful, you should have a new window pop up with a nice looking HTML document.

Ask for help if you encounter issues on any of the steps above. Once you've successfully made it through these steps, you can continue.

Goals

- Build a linear regression model of the height (in feet) of a black cherry tree given its diameter in inches (X)
- Use k-fold cross validation to evaluate strength of this predictive (linear regression) model

Context

- **Broader subject area: supervised learning**
  We want to build a model for some output RV Y given various predictor RVs $X_1, \ldots, X_p$.

- **Task: regression**
  Y is quantitative (takes numerical values)
- **Algorithm: linear regression model**
  We'll assume that the relationship between Y and X can be represented by

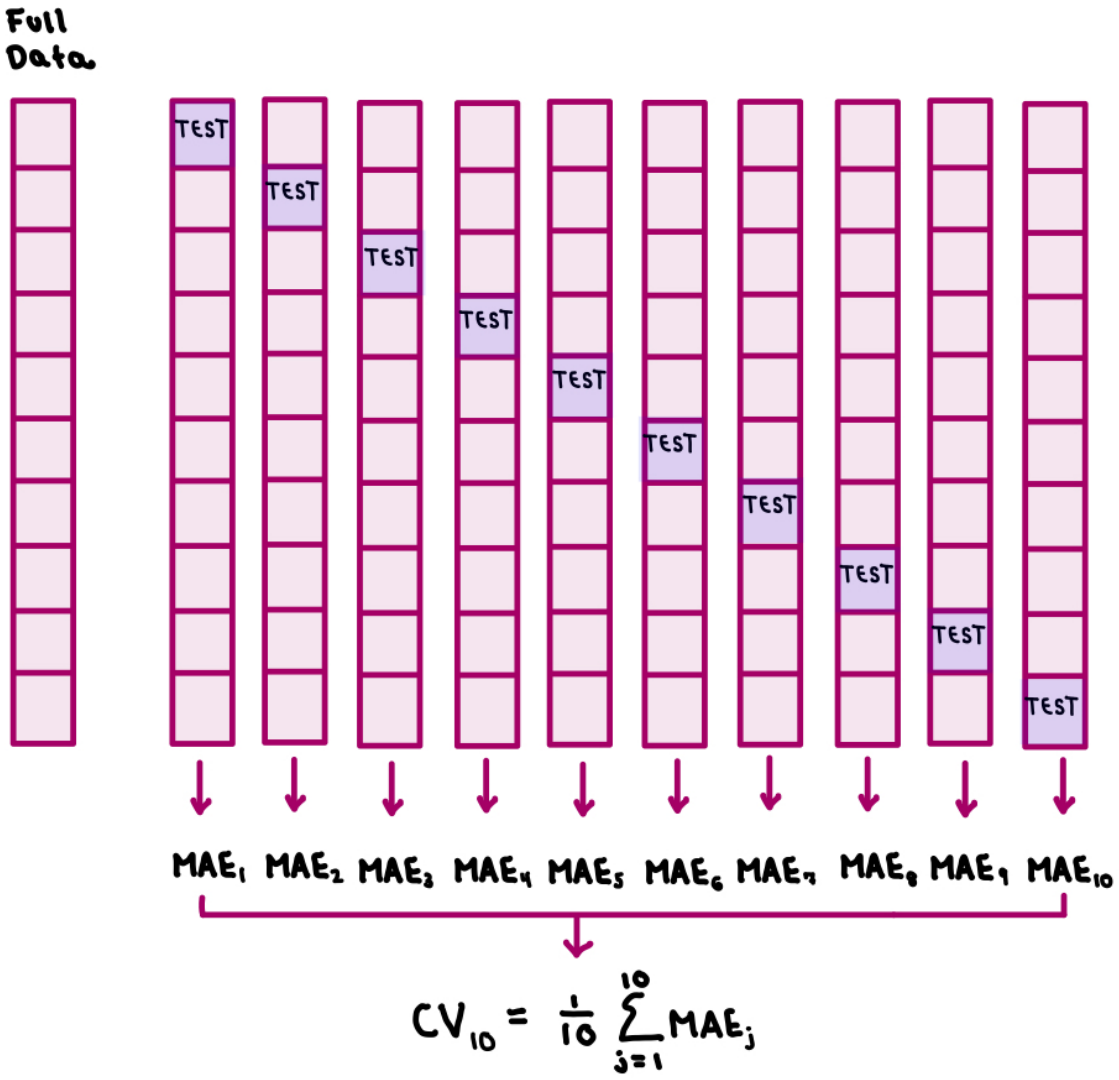
$$\mathbb{E}(Y \mid X_1, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Review: k-Fold Cross Validation

We can use **k-fold cross-validation** to estimate the typical error in our model predictions for *new* data:

- Divide the data into k folds (or groups) of approximately equal size.
- Repeat the following procedures for each fold $j = 1, 2, ..., k$:
  - Remove fold j from the data set.
  - Fit a model using the data in the other $k - 1$ folds (training).
  - Use this model to predict the responses for the n_j cases in fold j: $\hat{y}_1, ..., \hat{y}_{n_j}$.
  - Calculate the MAE for fold j (testing): $\text{MAE}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} |y_i - \hat{y}_i|$.
- Combine this information into one measure of model quality:

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{j=1}^{k} \text{MAE}_j$$



Vocabulary

A **tuning parameter** is *parameter* or quantity upon which an algorithm depends whose value is *selected* or *tuned* to "optimize" the algorithm.

For k-fold CV, the tuning parameter is k, where $2 \leq k \leq n$, where n is the number of observations.
Heuristic: k is usually picked to be something in the middle.

Set Up

Load Libraries

```
library(tidyverse)
library(tidymodels)
library(readxl)
# Load data
data(trees)
```

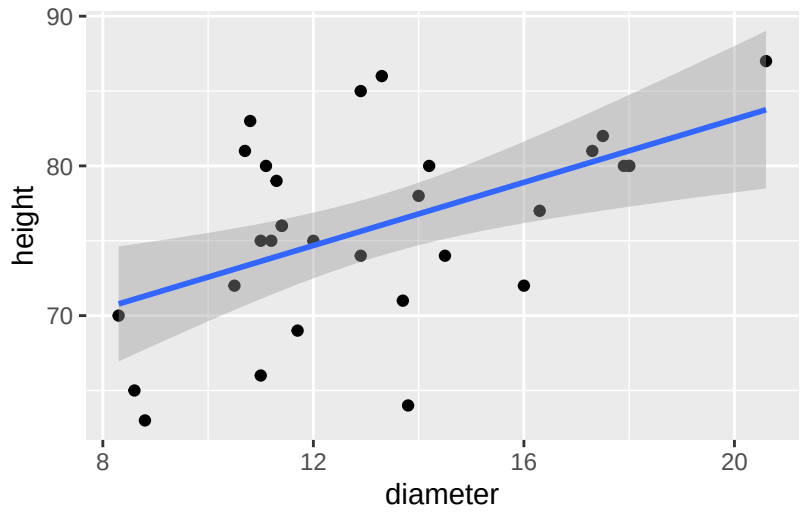
Rename Columns (Variables)

- Rename `Girth` to `diameter`
- Rename `Height` to `height`

```
# Rename columns
trees = trees %>%
  rename(diameter = Girth, height = Height) %>%
  # Only have height and diameter be the columns
  select(height, diameter)
```

Visualization (Scatter Plot)

```
# Create a scatter plot
ggplot(trees, aes(x = diameter, y = height)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Observations

- Height appears to be (weakly) positively correlated with diameter

```
# Step 1: Model specification
lm_spec <- linear_reg() %>%
  # Output Y is quantitative
  set_mode("regression") %>%
  # Want regression to be lienar
  set_engine("lm")
```

```
# Step 2: Model estimation
tree_model <- lm_spec %>%
  fit(height ~ diameter, data = trees)
```

10-Fold Cross Validation

Procedure

- Randomly split the data into 10 folds
- Built model 10 times, leaving 1 test fold out each time
- Evaluate each model on the test fold (using MAE/MSE and R-squared as error metrics)

```
# For reproducibility
set.seed(242)

tree_model_cv = lm_spec %>%
# fit_resamples() function is for fitting on folds
fit_resamples(
  # Specify the relationship
  height ~ diameter,
  # vfold_cv makes CV folds randomly from
  # trees data set
  resamples = vfold_cv(trees, v = 10),
  # Specify the error metrics
  # (MAE, square root MSE, R^2)
  metrics = metric_set(mae, rmse, rsq)
  )
```

```
tree_model_cv %>% collect_metrics()
```

```
# A tibble: 3 x 6
  .metric .estimator  mean     n std_err .config
  <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 mae     standard   4.38     10   0.639 Preprocessor1_Model1
2 rmse    standard   5.17     10   0.733 Preprocessor1_Model1
3 rsq     standard   0.341    10   0.121 Preprocessor1_Model1
```

Summarizing

```
# Get fold-by-fold results
# Get info for each test fold
tree_model_cv %>% unnest(.metrics) %>%
filter(.metric == "mae")
```

```
# A tibble: 10 x 7
   splits         id     .metric .estimator .estimate .config           .notes
   <list>         <chr>  <chr>   <chr>          <dbl> <chr>             <list>
 1 <split [27/4]> Fold01 mae     standard        4.23 Preprocessor1_Mo~ <tibble>
 2 <split [28/3]> Fold02 mae     standard        5.67 Preprocessor1_Mo~ <tibble>
 3 <split [28/3]> Fold03 mae     standard        7.92 Preprocessor1_Mo~ <tibble>
```

```
 4 <split [28/3]> Fold04 mae     standard        3.59 Preprocessor1_Mo~ <tibble>
 5 <split [28/3]> Fold05 mae     standard        6.11 Preprocessor1_Mo~ <tibble>
 6 <split [28/3]> Fold06 mae     standard        1.42 Preprocessor1_Mo~ <tibble>
 7 <split [28/3]> Fold07 mae     standard        1.99 Preprocessor1_Mo~ <tibble>
 8 <split [28/3]> Fold08 mae     standard        2.76 Preprocessor1_Mo~ <tibble>
 9 <split [28/3]> Fold09 mae     standard        5.70 Preprocessor1_Mo~ <tibble>
10 <split [28/3]> Fold10 mae     standard        4.37 Preprocessor1_Mo~ <tibble>
```

Observations

- Based on my random folds above, the prediction error (MAE) was best for fold 7 and worst for fold 3.

Exercises

```
# Load packages and data
health_data = read_xlsx("healthdata.xlsx")
```

```
Error: `path` does not exist: 'healthdata.xlsx'
```

EXERCISE 1: In-Sample Metrics

Use the `health_data` data to build two separate models of `height`:

```
# STEP 2: model estimation
model_1 <- lm_spec %>%
  fit(height ~ hip + weight + thigh + knee + ankle, data = health_data)
```

```
Error in is_list(data): object 'health_data' not found
```

```
model_2 <- lm_spec %>%
  fit(height ~ chest * age * weight + abdomen + hip + thigh + knee + ankle +
   ↪  biceps + forearm + wrist, data = health_data)
```

```
Error in is_list(data): object 'health_data' not found
```

Calculate the **in-sample** R-squared for both models:

```
# IN-SAMPLE R^2 for model_1 = ???
model_1 %>% glance()
```

```
Error in glance(.): object 'model_1' not found
```

```
# IN-SAMPLE R^2 for model_2 = ???
model_2 %>% glance()
```

```
Error in glance(.): object 'model_2' not found
```

> ANSWER. The R2 value for the first model is about 0.366, and for the second model, it's 0.526.

Calculate the **in-sample** MAE for both models:

```
# IN-SAMPLE MAE for model_1 = ???
model_1 %>%
  augment(new_data = health_data) %>%
  mae(truth = height, estimate = .pred)
```

```
Error in augment(., new_data = health_data): object 'model_1' not found
```

```
# IN-SAMPLE MAE for model_2 = ???
model_2 %>%
  augment(new_data = health_data) %>%
  mae(truth = height, estimate = .pred)
```

```
Error in augment(., new_data = health_data): object 'model_2' not found
```

> ANSWER. Based on the in-sample MAE (i.e., the MAE of the same data used to build/train the model), it appears that model 2 (whose MAE is about 3.366) is better than model 1 (whose MAE is about 3.481).

EXERCISE 2: In-Sample Model Comparison

Which model seems "better" by the in-sample metrics you calculated above? Any concerns about either of these models?

> Answered above! The concern is that we're using the same data that we built the model with to evaluate the model's error/performance, which means that the "better looking" model might be overfit (might be overly-specific to the data used to build it).

EXERCISE 3: 10-Fold CV

Complete the code to run 10-fold cross-validation for our two models.

`model_1: height ~ hip + weight + thigh + knee + ankle`
`model_2: height ~ chest * age * weight  + abdomen + hip + thigh + knee + ankle + biceps + forearm + wrist`

```
# 10-fold cross-validation for model_1
set.seed(244)
model_1_cv <- lm_spec %>%
  fit_resamples(
    height ~ height ~ hip + weight + thigh + knee + ankle,
    resamples = vfold_cv(health_data, v = 10),
    metrics = metric_set(mae, rsq)
  )
```

```
Error in nrow(data): object 'health_data' not found
```

```
# 10-fold cross-validation for model_2
set.seed(253)
model_2_cv <- lm_spec %>%
  fit_resamples(
    height ~ chest * age * weight + abdomen + hip + thigh + knee + ankle +
 ↪  biceps + forearm + wrist,
    resamples = vfold_cv(health_data, v = 10),
    metrics = metric_set(mae, rsq)
  )
```

```
Error in nrow(data): object 'health_data' not found
```

EXERCISE 4: Calculating the CV MAE

a. Use `collect_metrics()` to obtain the cross-validated MAE and R^2 for both models.

```
model_1_cv %>%
  collect_metrics()
```

```
Error in collect_metrics(.): object 'model_1_cv' not found
```

```
model_2_cv %>%
  collect_metrics()
```

```
Error in collect_metrics(.): object 'model_2_cv' not found
```

b. Interpret the cross-validated MAE *and* R^2 for `model_1`.

ANSWER. We expect our first model to produce predictions of height that are roughly off by 4.13 (the observed MAE) on average. For the first model, we expect it to explain roughly 0.28 (28%) of the variability (based on the R2 value) in the observed heights of patients in the data set.

EXERCISE 5: Fold-By-Fold Results

The command `collect_metrics()` gave the final CV MAE, or the average MAE across all 10 test folds. The command `unnest(.metrics)` provides the MAE from *each* test fold.

a. Obtain the fold-by-fold results for the `model_1` cross-validation procedure using `unnest(.metrics)`.

```
model_1_cv %>%
  unnest(.metrics) %>%
  filter(.metric == "mae")
```

```
Error in unnest(., .metrics): object 'model_1_cv' not found
```

b. Which fold had the worst average prediction error and what was it?

For me, fold 5 had the worst (highest) MAE (which was 10.9).

c. Recall that `collect_metrics()` reported a final CV MAE of 4.13 for `model_1`. Confirm this calculation by wrangling the fold-by-fold results from part a.

```r
# Code here
model_1_cv %>%
  unnest(.metrics) %>%
  filter(.metric == "mae") %>%
  summarize(mean(.estimate))
```

```
Error in unnest(., .metrics): object 'model_1_cv' not found
```

EXERCISE 6: Comparing Models

Fill in the table below to summarize the in-sample and 10-fold CV MAE for both models.

Model	IN-SAMPLE MAE	10-fold CV MAE
`model_1`	3.48	4.13
`model_2`	3.37	6.28

a. Based on the in-sample MAE alone, which model appears better?

   YOUR ANSWER HERE

b. Based on the CV MAE alone, which model appears better?

   YOUR ANSWER HERE

c. Based on all of these results, which model would you pick?

   YOUR ANSWER HERE

d. Do the in-sample and CV MAE suggest that `model_1` is overfit to our `health_data` sample data? What about `model_2`?

   For model 1, it looks like the MAE is roughly similar for when it's measured in-sample (3.48) versus when it's tested on "new" data (each test fold held out) (4.13). However, model 2 seems overfit because its predictions for new patient data (giving an MAE of 6.28) are much worse than its predictions for patients in our data sample (MAE of 3.37).

EXERCISE 7: LOOCV

a. Reconsider `model_1`. Instead of estimating its prediction accuracy using the 10-fold CV MAE, use the LOOCV MAE. THINK: How many people are in our `health_data` sample?

```
# CODE HERE

model_1_loocv <- lm_spec %>%
  fit_resamples(
    height ~ hip + weight + thigh + knee + ankle,
    resamples = vfold_cv(health_data, v = nrow(health_data)),
    metrics = metric_set(mae)
  )
```

```
Error in nrow(data): object 'health_data' not found
```

```
model_1_loocv %>% collect_metrics()
```

```
Error in collect_metrics(.): object 'model_1_loocv' not found
```

b. How does the LOOCV MAE compare to the 10-fold CV MAE of ____? NOTE: These are just two different *approaches* to estimating the same thing: the typical prediction error when applying our model to new data. Thus we should expect them to be similar.

ANSWER.

c. Explain why we technically don't *need* to `set.seed()` for the LOOCV algorithm.

ANSWER.