Sane's STATS Academy of Statistics

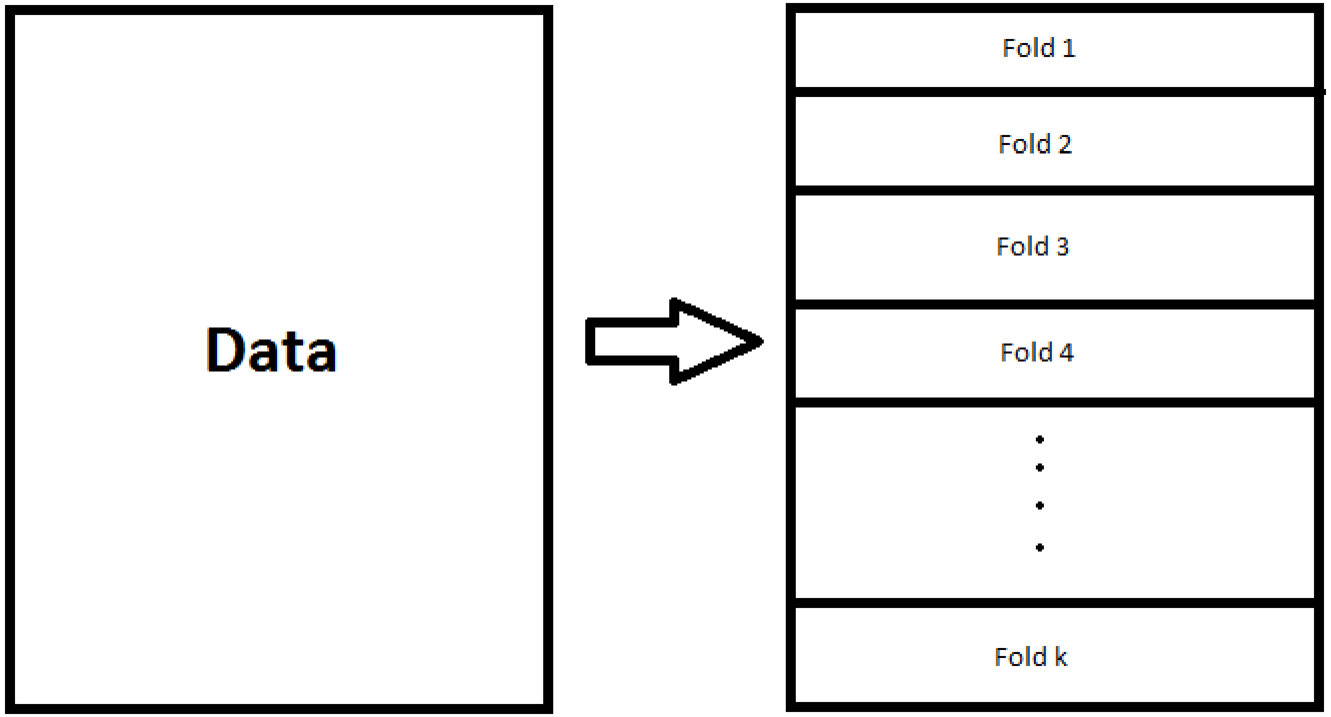
Cross-Validation

K-Folds

k-Folds

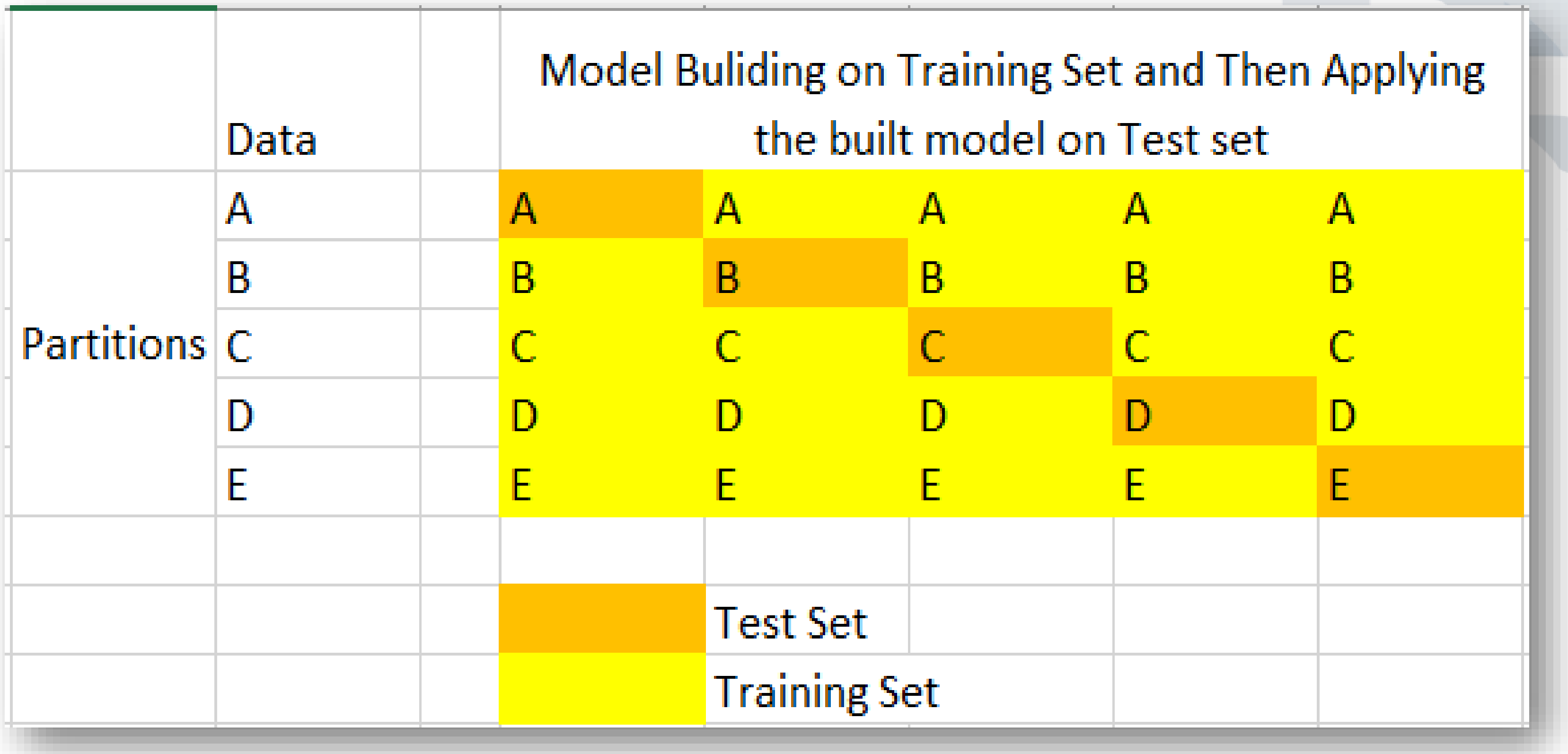
- In the cross-validation, so far we have partitioned the data into two parts or three parts
- In k-folds cross-validation, we divide the data into k equal (more or less) parts and then assuming each fold as test set average the error

K-Folds Cross-Validation



- The k-folds cross-validation process starts from dividing the data into k equal parts and that too randomly

K-Folds



	Data		Model Buliding on Training Set and Then Applying the built model on Test set				
Partitions	A		🟧 A	🟨 A	🟨 A	🟨 A	🟨 A
	B		🟨 B	🟧 B	🟨 B	🟨 B	🟨 B
	C		🟨 C	🟨 C	🟧 C	🟨 C	🟨 C
	D		🟨 D	🟨 D	🟨 D	🟧 D	🟨 D
	E		🟨 E	🟨 E	🟨 E	🟨 E	🟧 E
			🟧	Test Set			
			🟨	Training Set			

K-Folds Steps

1. Consider Fold 1 as a test set for the time being
2. With Fold 1 as test set, and remaining all other folds (data) as train set fit a model
3. Evaluate the model for the test set (Fold 1) and record the error in the evaluation
4. Consider Fold 2 as a test set
5. Repeat steps 2 and 3 for Fold 2
6. For each of the k folds, repeat 2 and 3
7. Aggregate the errors calculated in all the evaluations.

K-folds in R

- K-folds can be created using function *createFolds()* from package caret

Syntax : createFolds(y , k , …)

Where

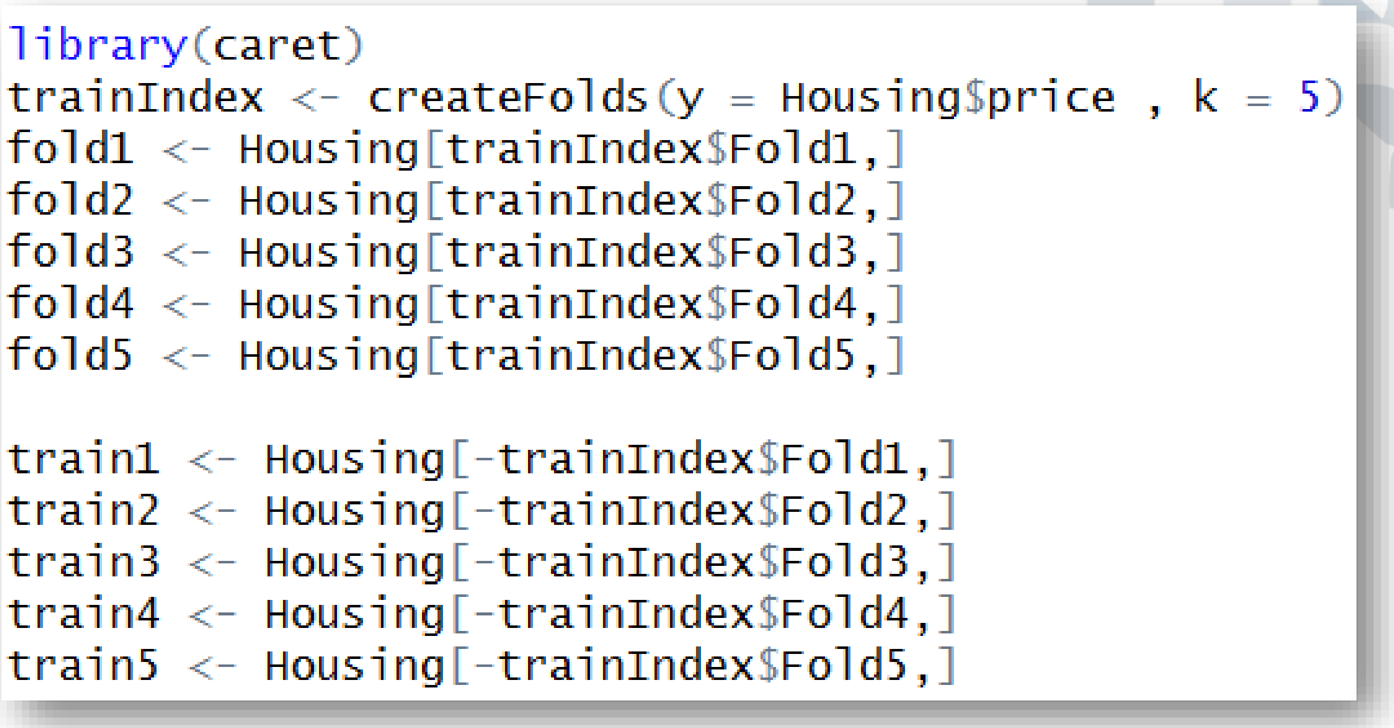
y : response variable

k : folds to be created

Illustration

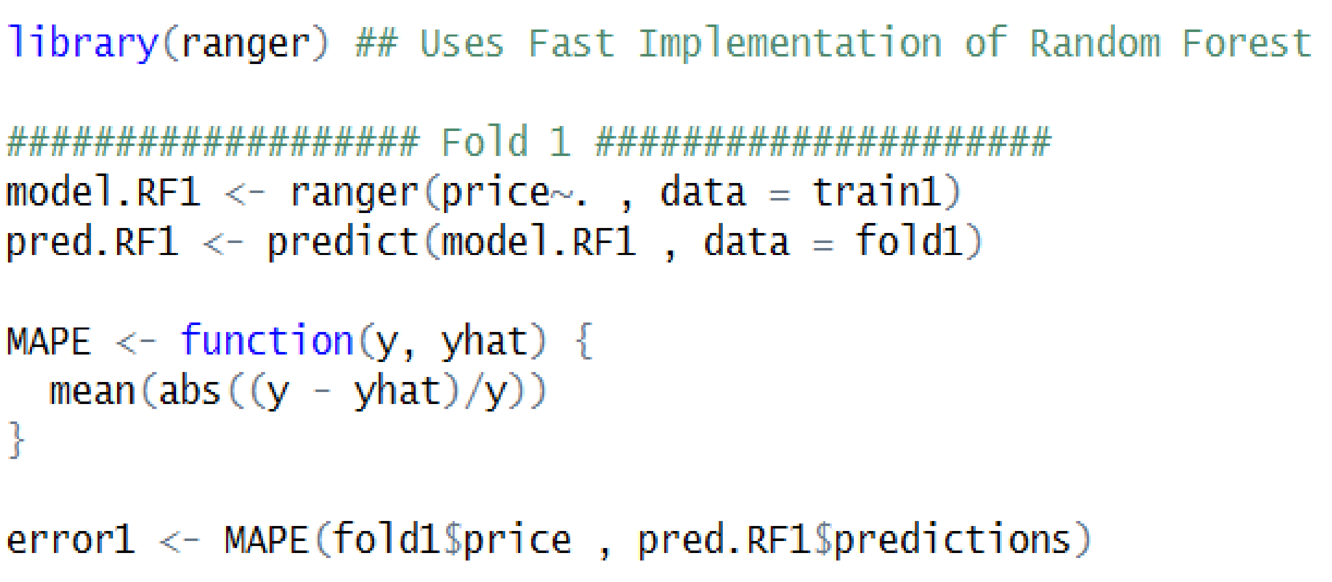
- **Description**
  - a cross-section from 1987
  - *number of observations* : 546
  - *country* : Canada
- A dataframe containing :
  - **price :** sale price of a house
  - **lotsize :** the lot size of a property in square feet
  - **bedrooms :** number of bedrooms
  - **bathrms :** number of full bathrooms
  - **stories :** number of stories excluding basement
  - **driveway :** does the house has a driveway ?
  - **recroom :** does the house has a recreational room ?
  - **fullbase :** does the house has a full finished basement ?
  - **gashw :** does the house uses gas for hot water heating ?
  - **airco :** does the house has central air conditioning ?
  - **garagepl :** number of garage places
  - **prefarea :** is the house located in the preferred neighbourhood of the city ?

R Program (Creating Folds)



```
library(caret)
trainIndex <- createFolds(y = Housing$price , k = 5)
fold1 <- Housing[trainIndex$Fold1,]
fold2 <- Housing[trainIndex$Fold2,]
fold3 <- Housing[trainIndex$Fold3,]
fold4 <- Housing[trainIndex$Fold4,]
fold5 <- Housing[trainIndex$Fold5,]

train1 <- Housing[-trainIndex$Fold1,]
train2 <- Housing[-trainIndex$Fold2,]
train3 <- Housing[-trainIndex$Fold3,]
train4 <- Housing[-trainIndex$Fold4,]
train5 <- Housing[-trainIndex$Fold5,]
```

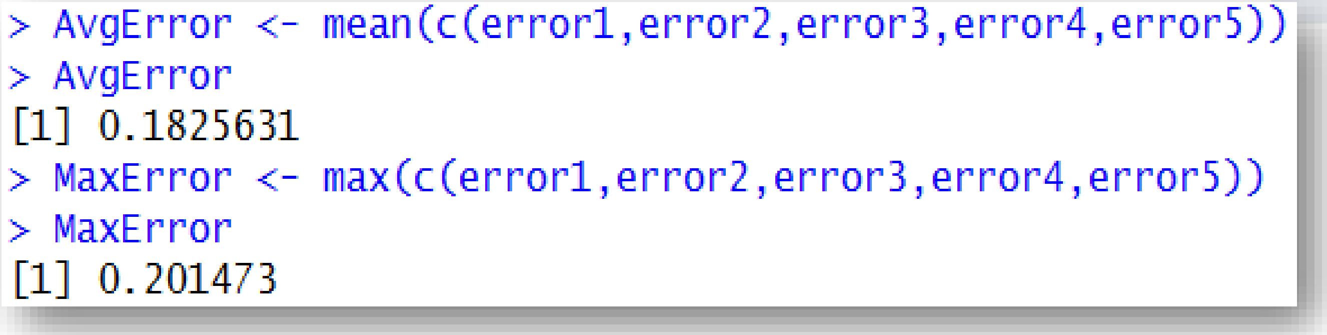
R Program (Evaluation)



```
library(ranger) ## Uses Fast Implementation of Random Forest

################### Fold 1 #####################
model.RF1 <- ranger(price~. , data = train1)
pred.RF1 <- predict(model.RF1 , data = fold1)

MAPE <- function(y, yhat) {
  mean(abs((y - yhat)/y))
}

error1 <- MAPE(fold1$price , pred.RF1$predictions)
```

- This model building and evaluation is repeated for all 5 folds

Aggregating the Error



```
> AvgError <- mean(c(error1,error2,error3,error4,error5))
> AvgError
[1] 0.1825631
> MaxError <- max(c(error1,error2,error3,error4,error5))
> MaxError
[1] 0.201473
```

- Errors on all the folds are averaged or maximum error can be considered.
- Hence, we get more promising estimate of the error.

Alternative – caret way

- In caret package, we have a train control function *trainControl(),* object of which needs to be passed to the *train()* function itself.

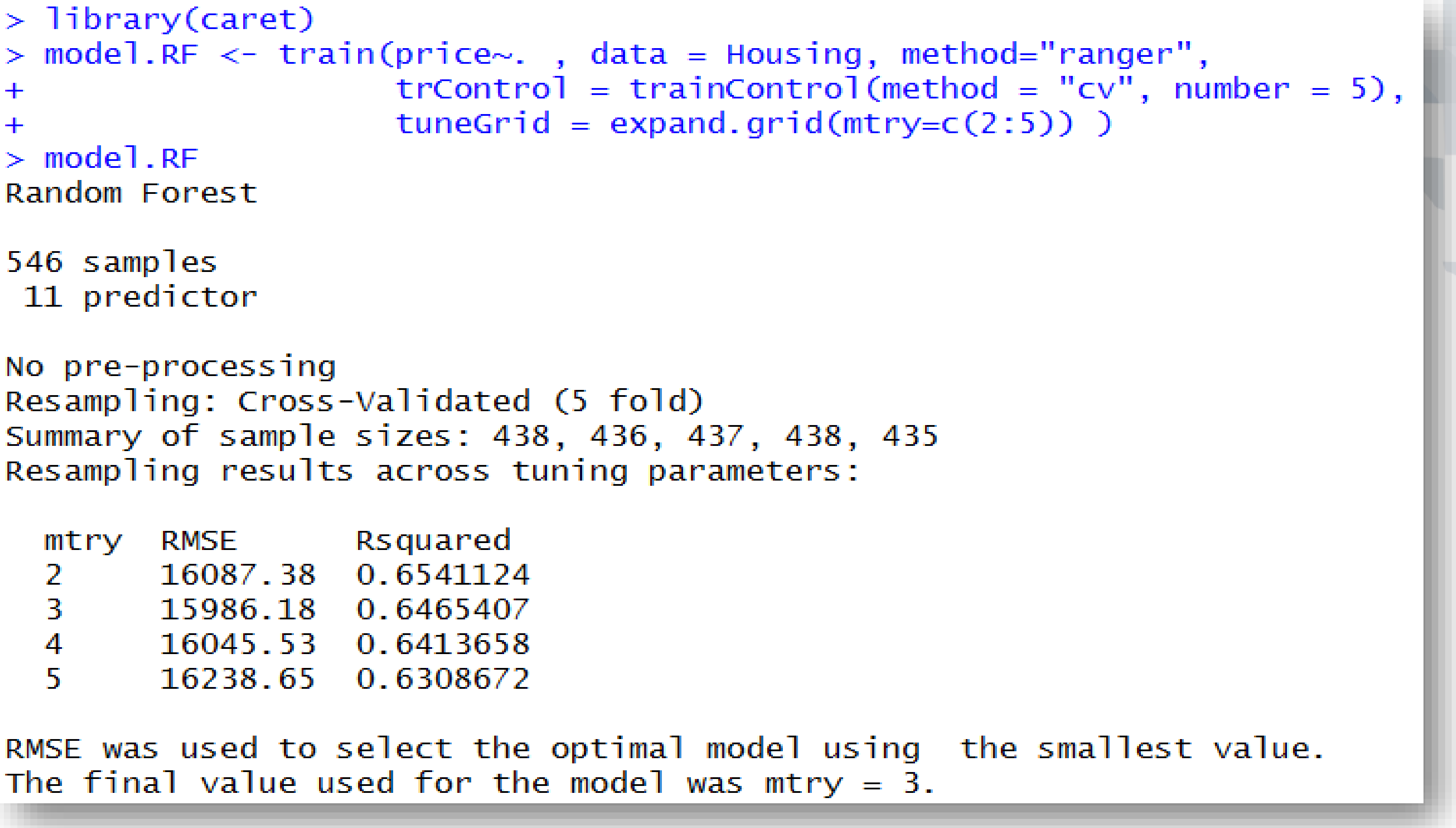
Syntax : trainControl(method, number, …)

Where

method : resampling method; “boot”, “cv”, “LOOCV” etc are its possible options

number : number of folds to be created

Using trainControl()



```
> library(caret)
> model.RF <- train(price~. , data = Housing, method="ranger",
+                   trControl = trainControl(method = "cv", number = 5),
+                   tuneGrid = expand.grid(mtry=c(2:5)) )
> model.RF
Random Forest

546 samples
 11 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 438, 436, 437, 438, 435
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared
  2     16087.38  0.6541124
  3     15986.18  0.6465407
  4     16045.53  0.6413658
  5     16238.65  0.6308672

RMSE was used to select the optimal model using  the smallest value.
The final value used for the model was mtry = 3.
```