BIOS 635: Cross-Validation

Kevin Donovan

1/28/2021

Review

- Homework 4 due on 2/26 at 11PM through GitHub Classroom
- Article Evaluation 1 assigned, due on 3/2 through GitHub Classroom
- Last lecture: nonlinear modeling using splines

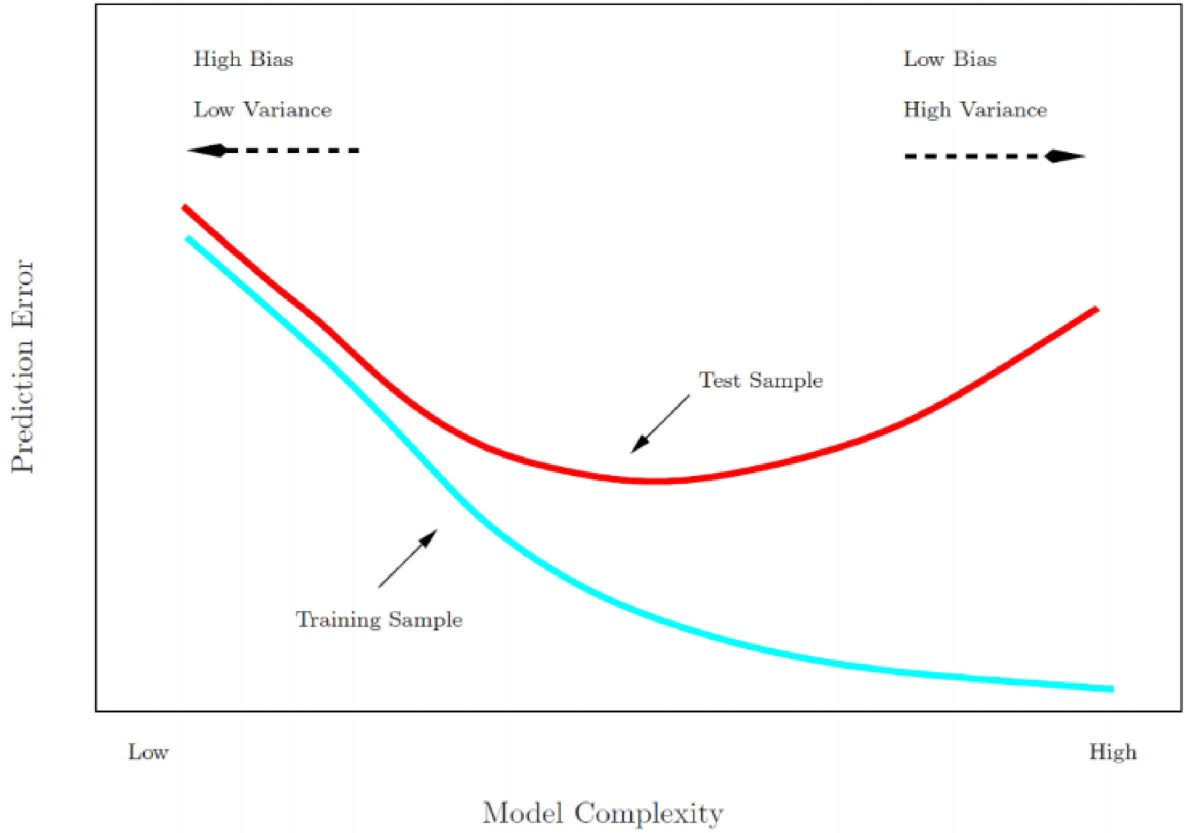
Prediction Error

- Recall prediction error can be calculated a variety of subsets of your data

  1. **training error**: average error when predicting outcome on data used to create algorithm
  2. **testing error**: average error when predicting outcome on data from that used in training

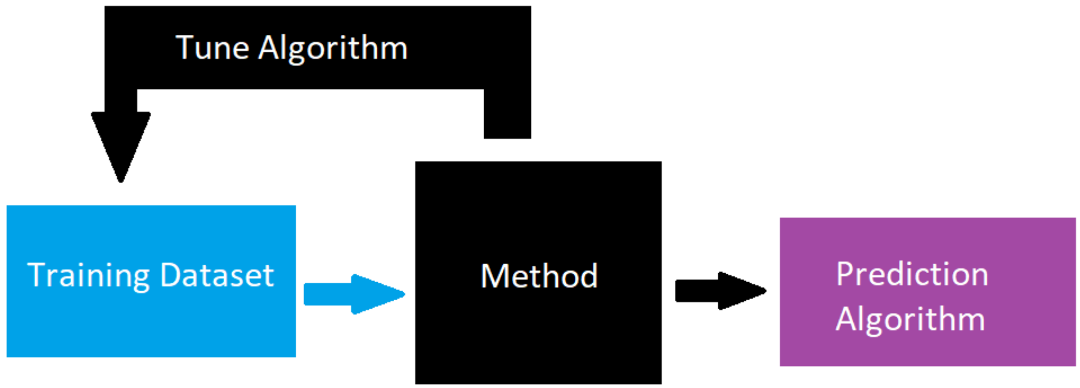
- Training error poor measure of algorithm's performance on general sample from population

  - ***Biased downward***
  - *Need separate and **independent** datasets for testing and training*

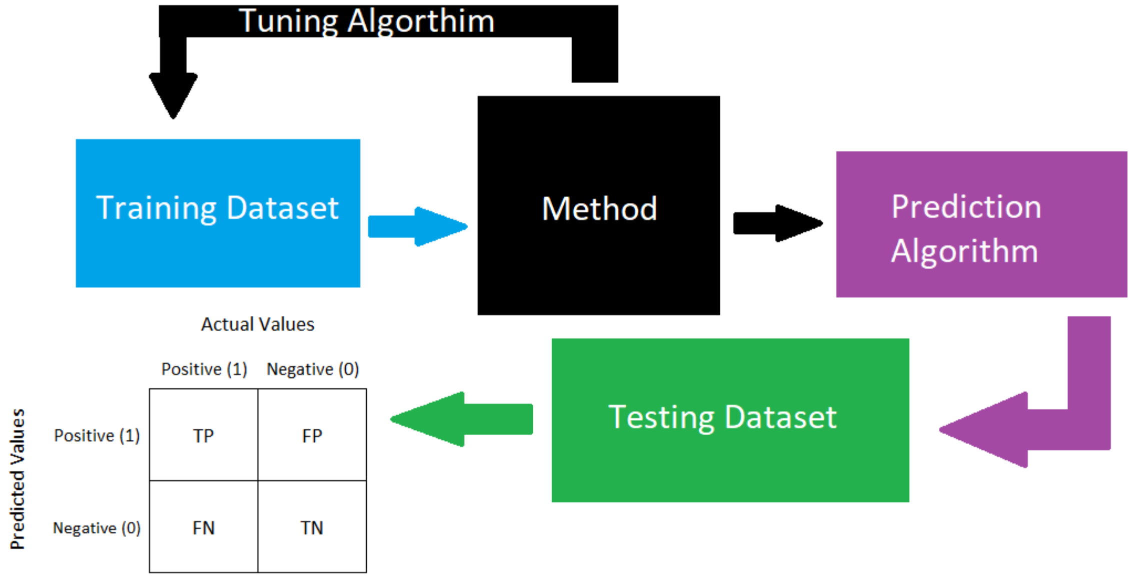

High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Training Sample
Low
High
Model Complexity

Training and Testing

1. Training and Tuning



2. Training, Tuning, and Testing


Tuning Algorthim
Training Dataset
Method
Prediction Algorithm
Actual Values
Positive (1)
Negative (0)
Predicted Values
Positive (1)
Negative (0)
TP
FP
FN
TN
Testing Dataset

Testing error

- Various ways have been developed to estimate this testing error:

1. "Correct" training set error to be more generalizable

- **Idea**: $Error = f(MSE) + \lambda * ModelComplexity$ where $\lambda > 0$
- Ex. *Mallow's Cp, AIC, BIC*

2. Use large, independent and separate test set

- Often not available, though best option

3. Generate test set using **hold out**

- Randomly split available data into 2 partitions
- Use one partition for training, other for testing
- Testing = predict outcome on test set, compute prediction error (ex. MSE or misclassification rate)

Hold out

A random splitting into two halves: left part is training set, right part is validation set

```
wage_data <- Wage # contained in ISLR package
# Holdout 40% for tesing
tt_indicies <- createDataPartition(y=wage_data$wage, p=0.6, list = FALSE)
wage_data_train <- wage_data[tt_indicies,]
wage_data_test <- wage_data[-tt_indicies,]

# Look at datasets
paged_table(wage_data_train)
```

	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
231655	2006	18	1. Never Married	1. White	1. < HS Grad

	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
450601	2009	44	2. Married	4. Other	3. Some College
81404	2004	52	2. Married	1. White	2. HS Grad
305706	2007	34	2. Married	1. White	2. HS Grad
8690	2005	35	1. Never Married	1. White	2. HS Grad
153561	2003	39	2. Married	1. White	4. College Grad
449654	2009	54	2. Married	1. White	2. HS Grad
447660	2009	51	2. Married	1. White	3. Some College
160191	2003	37	1. Never Married	3. Asian	4. College Grad
230312	2006	50	2. Married	1. White	5. Advanced Degree

1-10 of 1,802 rows | 1-9 of 12 columns Previous 1 2 3 4 5 6 ... 18 Next

```
paged_table(wage_data_test)
```

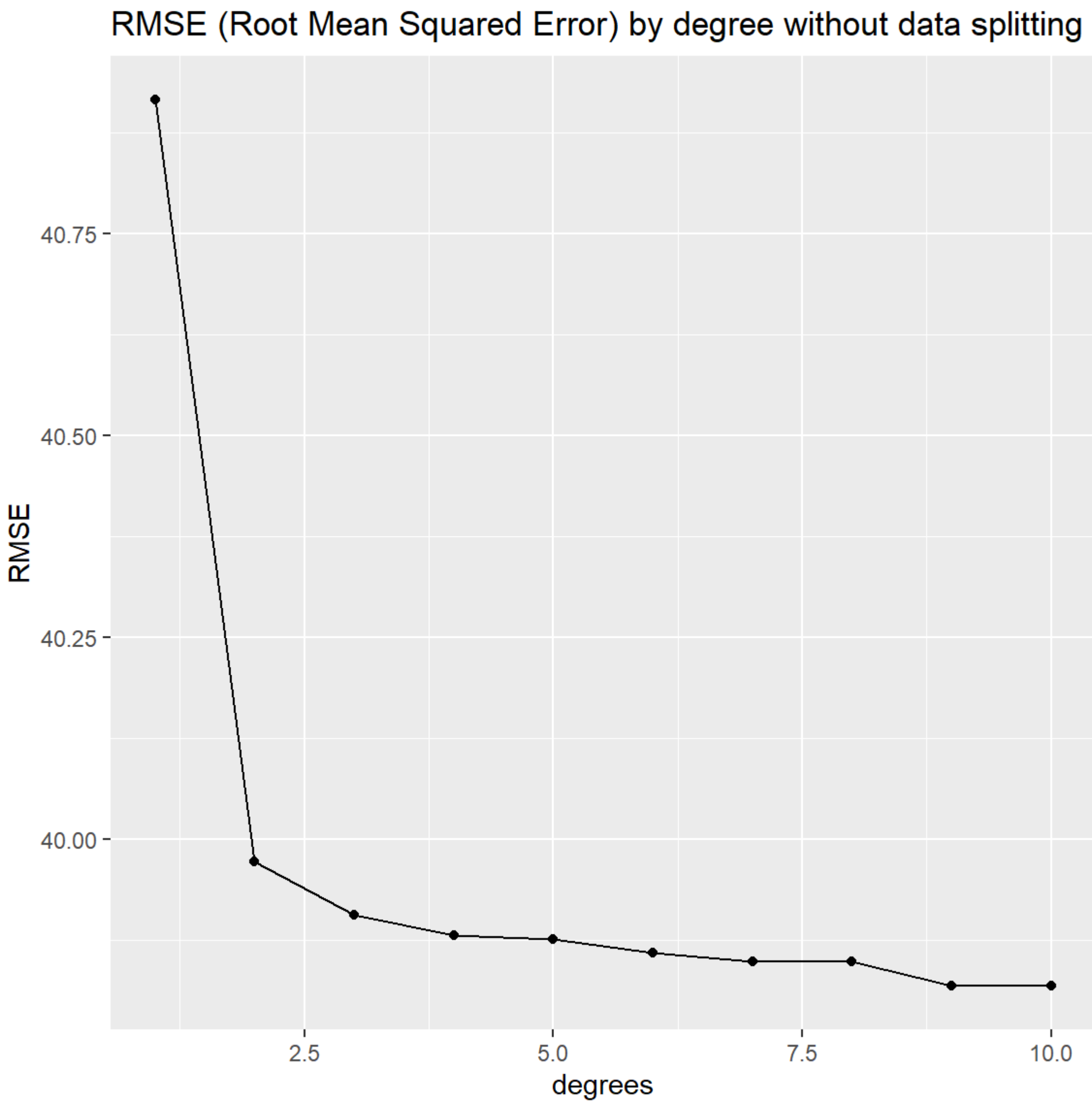
	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
86582	2004	24	1. Never Married	1. White	4. College Grad
161300	2003	45	2. Married	1. White	3. Some College
155159	2003	43	2. Married	3. Asian	4. College Grad

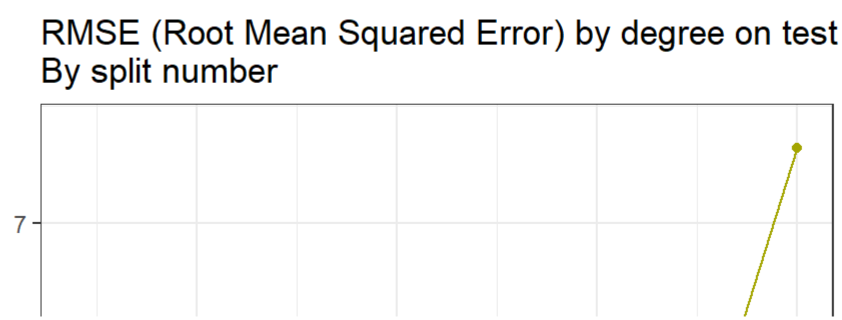
	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
11443	2005	50	4. Divorced	1. White	2. HS Grad
376662	2008	54	2. Married	1. White	4. College Grad
377954	2008	30	1. Never Married	3. Asian	3. Some College
228963	2006	41	1. Never Married	2. Black	3. Some College
302778	2007	45	4. Divorced	1. White	3. Some College
153682	2003	37	1. Never Married	1. White	3. Some College
11141	2005	40	4. Divorced	1. White	2. HS Grad

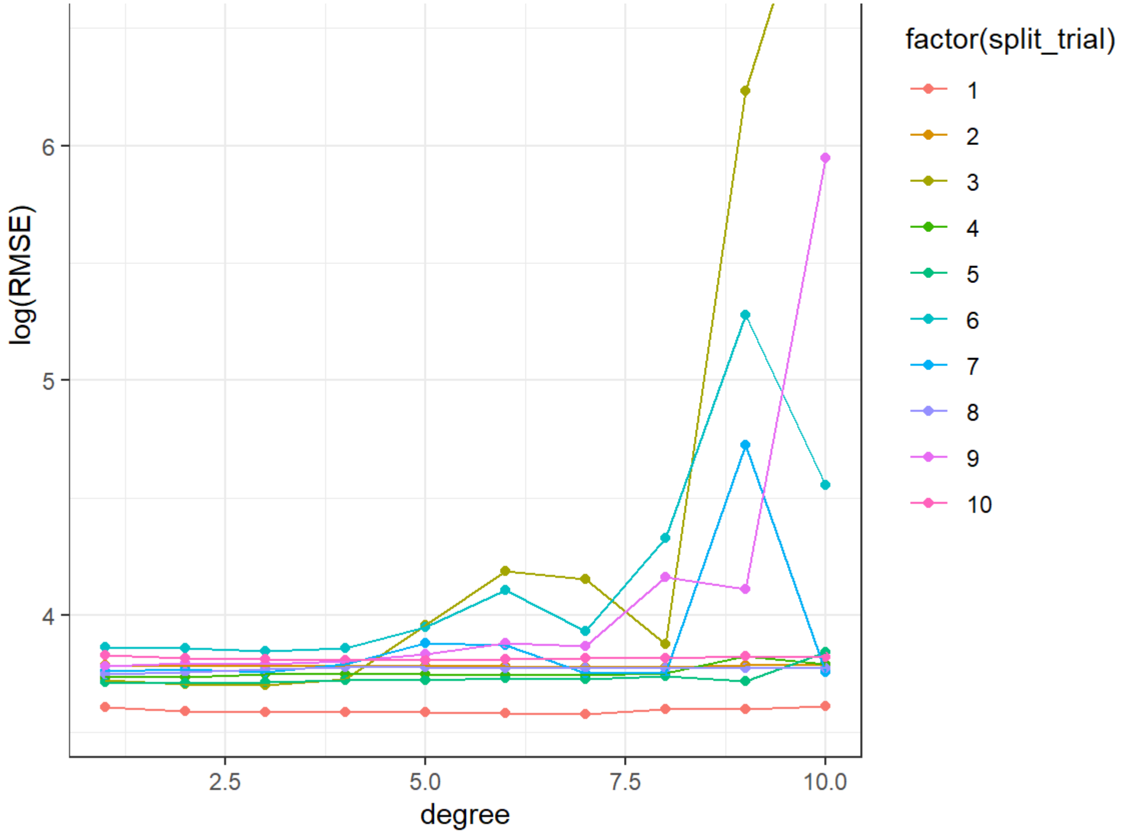
1-10 of 1,198 rows | 1-9 of 12 columns Previous 1 2 3 4 5 6 ... 120 Next

Training and testing

- Consider fitting nonlinear polynomial to wage data
- Using training error vs. testing error to choose spline order


RMSE (Root Mean Squared Error) by degree without data splitting
RMSE
40.75
40.50
40.25
40.00
2.5
5.0
7.5
10.0
degrees


RMSE (Root Mean Squared Error) by degree on test set
By split number
7



factor(split_trial)
1
2
3
4
5
6
7
8
9
10
log(RMSE)
6
5
4
2.5
5.0
7.5
10.0
degree

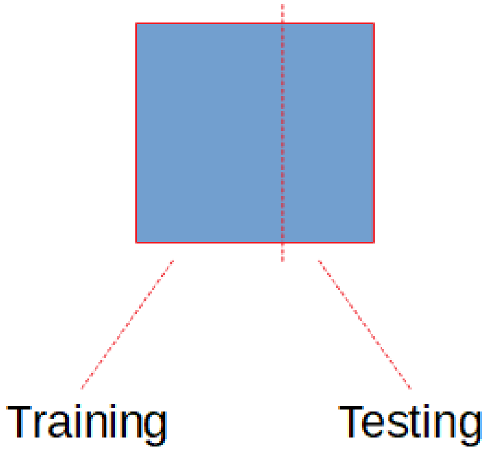
Drawbacks of holdout

- Test set error can be highly dependent on split
  - *Thus* ***highly variable***
  - *Especially for small dataset or* ***small group sizes***
- Only subset of data used to train algorithm
  - *May result in poorer algorithm*
  - $\rightarrow$ *may* ***overestimate*** *test error*
- Can we **aggregate results over multiple test sets?**

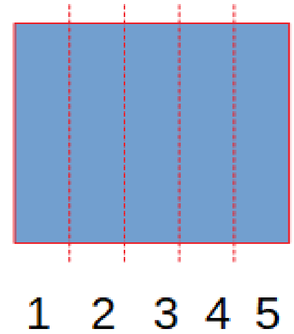
K-fold cross validation

- **Widely used** approach for estimating test error
- **Idea**: Still use entire data for training but evaluate **average** performance by aggregating over multiple test sets
    - *Test sets still must be **independent***
    - *Example: 5-fold CV*

Single Partition



Cross Validation



1) Training : 1-4, Testing : 5
2) Training : 1-3+4, Testing : 4
3) Training : 1-2+4-5, Testing : 3
4) Training : 1+3-5, Testing : 2
5) Training : 2-5, Testing : 1

K-fold cross validation

- Denote K folds by $C_1, C_2, \ldots, C_K$, each with n_k observations
- For a given fold l:

    1. *Train algorithm on data in other folds: $\{C_k\}$ s.t. $k \neq l$*
    2. *Test by computing predicted values for data in C_l* ***only***
    3. *Repeat for each fold $l = 1, \ldots, K$, average error (ex. MSE_l)*

- K fold CV error rate

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ where y_i is outcome and $\hat{y}_i$ is predicted outcome from training on C_k **only**

- $K = n$ yields $n - fold$ or *leave-one out cross-validation*

- $CV_(K)$ is accurate measure of generalized error rate for algorithm trained on whole sample

K-fold CV in R

```
# 5 fold CV partitions
cv_folds <- createFolds(y=wage_data_subset$wage, k=5)
# Can see whose in fold 1
cv_folds$Fold1
```

```
##  [1]   2  10  12  19  20  26  33  39  40  42  49  51  63  75  81  87  88  96  98
## [20]  99 103 104 108 109 116 128 137 140 142 143 145 151 156 159 163 166 176 185
## [39] 186 192 194 199 219 223 225 228 241 248 251 266 270 274 276 278 282 284 285
## [58] 288 292 297 301 314 319 322 323 326 328 333 335 343 345 353 355 357 365 371
## [77] 374 375 384 391 395
```

```
# Look at dataset for fold 1
wage_data_fold_1 <- wage_data_subset[cv_folds$Fold1,]
paged_table(wage_data_fold_1)
```

	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
306688	2007	22	1. Never Married	1. White	3. Some College
85840	2004	50	2. Married	1. White	2. HS Grad
303306	2007	26	2. Married	1. White	2. HS Grad
81295	2004	33	2. Married	1. White	2. HS Grad
10047	2005	47	2. Married	1. White	5. Advanced Degree

	year	age	maritl	race	education
	<int>	<int>	<fctr>	<fctr>	<fctr>
302409	2007	45	2. Married	1. White	4. College Grad
380151	2008	30	2. Married	3. Asian	5. Advanced Degree
307821	2007	44	2. Married	1. White	3. Some College
81780	2004	24	1. Never Married	1. White	3. Some College
377037	2008	44	2. Married	1. White	1. < HS Grad

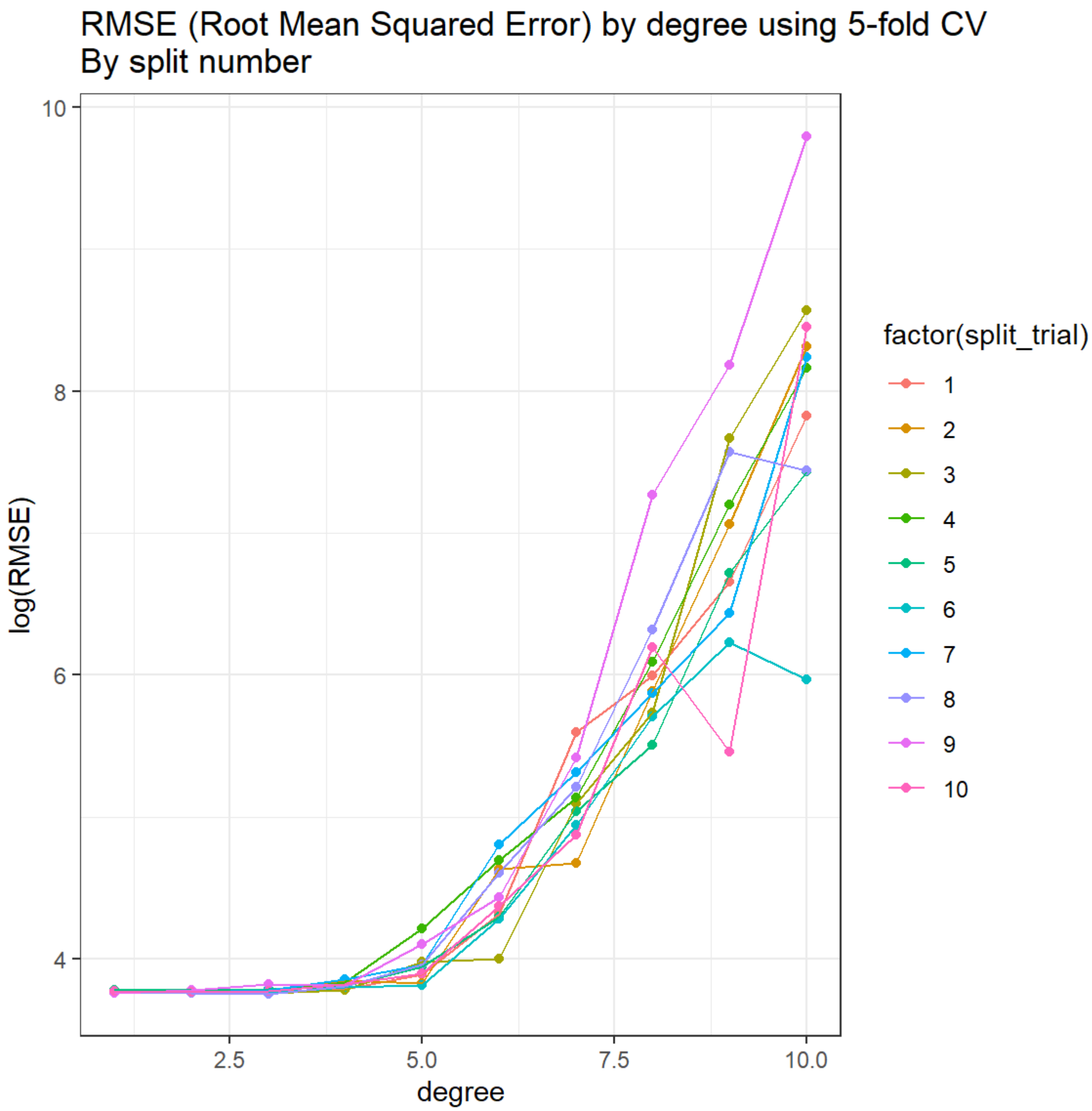
1-10 of 81 rows | 1-9 of 12 columns Previous 1 2 3 4 5 6 ... 9 Next

K-fold CV analysis

- Let's look back at the nonlinear fitting example from before. Instead of using a holdout testing method, we use 5-fold CV

```
# Fit model for each degree considered, compute RMSE (on training in this ex.)
poly_reg_fit <- list()
predict_wages <- list()
residuals_wages <- list()
rmse_poly_reg <- list()
mae_poly_reg <- list()
error_rates_degrees <- list()

counter <- 1
trials <- 10 # Look at 10 different 60:40 splits

for(j in 1:trials){
  set.seed(j) # Set seed to get different splits

  tt_indicies <- createFolds(y=wage_data_subset$wage, k=5)

    for(i in 1:length(degrees)){
      for(f in 1:length(tt_indicies)){
        wage_data_train <- wage_data_subset[tt_indicies[[f]],]
        wage_data_test <- wage_data_subset[-tt_indicies[[f]],]

        poly_reg_fit[[f]] <- lm(wage~poly(age, degrees[i]),
                                 data=wage_data_train)
```

```
        predict_wages[[f]] <- predict(poly_reg_fit[[f]], newdata = wage_data_test)
        residuals_wages[[f]] <- wage_data_test$wage-predict_wages[[f]]
        rmse_poly_reg[[f]] <- sqrt(mean(residuals_wages[[f]]^2))
        mae_poly_reg[[f]] <- mean(abs(residuals_wages[[f]]))
      }

      # Save in data frame
      error_rates_degrees[[counter]] <-
        data.frame("RMSE"=mean(unlist(rmse_poly_reg)),
                   "MAE"=mean(unlist(mae_poly_reg)),
                   "degree"=degrees[i],
                   "split_trial"=j)
      counter <- counter+1
    }
}

  # Bind all degree-specific results together into single data frame/table
  error_rates_degrees_df <- do.call("rbind", error_rates_degrees)

  # Plot results as function of degree
  ggplot(data=error_rates_degrees_df,
         mapping=aes(x=degree, y=log(RMSE), color=factor(split_trial)))+
    geom_point()+
    geom_line()+
    labs(title="RMSE (Root Mean Squared Error) by degree using 5-fold CV\nBy split number")+
    theme_bw()
```



RMSE (Root Mean Squared Error) by degree using 5-fold CV
By split number
log(RMSE)
10
8
6
4
2.5
5.0
7.5
10.0
degree
factor(split_trial)
1
2
3
4
5
6
7
8
9
10