



STOR 320 Modeling III

Lecture 17

Yao Li

Department of Statistics and Operations Research

UNC Chapel Hill

Introduction

- Instructions
 - Download Tutorial 13 Zip
 - Unzip Folder
 - Required Packages
 - `library(modelr)`
 - `library(tidyverse)`
 - `library(purrr)`
 - `library(broom)`
 - Open .Rmd File and Knit

Discussion

- Problems With Current Approach
 - Same Model For All Locations
 - Not All Locations Used in Train
 - Not All Locations Used in Test
 - Residuals Indicate that Model Can Be Improved
 - Not Helpful for Forecasting
 - Ambiguous Results: No Clear Winner

Part 1: Cross Validation by Location

- Previously
 - Split Data in Train and Test
 - Train (28 Rivers)
 - Test (3 Rivers)
 - Purpose
 - Estimate Out-of-Sample Error
 - Pick Best Model Based on This Estimate
 - Combat Overfitting
 - Robustification
 - Goal: Find the Simplest Model that Adequately Predicts

Part 1: Cross Validation by Location

- Current Issues
 - Decision on Final Model Heavily Influenced by the Test Data
 - Loss of Data in Model Fitting
 - Not Appropriate in Small Datasets
- Cross Validation Idea
 - Split Data Into Many Groups
 - Each Group Acts as a Test Set
 - All Data is Used in Both Model Fitting and Model Testing
 - Help: Chapter 5 (ISLR)

Part 1: Cross Validation by Location

- Tidyverse Concepts
 - Chapter 20 (R4DS)
 - List-Columns
 - Columns in Data Frames or Tibbles Can Be Lists
 - What this Means
 - Column of Tables
 - Column of Models
 - Column of Functions
 - Functions
 - `nest()`: Converts Rows of a Data Frame into a List
 - `unnest()`: What do You Think It Does?

Part 1: Cross Validation by Location

- Run Chunk 1
 - Observe the Output
 - Column of Tibbles
- Run Chunk 2
 - Imagine We Wanted to Split
 - Test: Data For Location 103
 - Train: All Remaining Data
 - Use of `filter()` and `unnest()`
 - First Glimpse -> 365 x 8
 - Second Glimpse -> 10,972 x 8

Part 1: Cross Validation by Location

- Chunk 3
 - Run Each Line
 - What is Happening?
 - Use View() on DATA2 and Scan Through the Data
 - What do You Notice?
- Chunk 4
 - Create a Loop that Repeats this Process for Each Location
 - Each Location Is a Test Set
 - Predictions Saved are All Out-of-Sample
 - Run Chunk 4 to Test Your Code

Part 1: Cross Validation by Location

- Chunk 4 (Continued)

```
DATA2=DATA
DATA2$linpred=NA

for(k in unique(DATA2$L)){
  TEST = NEST.DATA %>% filter(L==k) %>%
unnest()
  TRAIN = NEST.DATA %>% filter(L!=k) %>%
unnest()

  linmod=lm(W~A, data=TRAIN)
  linmodpred=predict(linmod,newdata=TEST)

  DATA2$linpred[which(DATA2$L==k)]=linmodpred
}
```

Part 1: Cross Validation by Location

- Chunk 5
 - In Our Data, We Have:
 - Actual Water Temperatures
 - Out-of-Sample Predicted Water Temperatures
 - Create `RMSE.func()` With Two Arguments
 - `actual`= vector of actual water temperatures
 - `predict`=vector of predicted water temperatures
 - Use This Function on the Two Columns in `DATA2` for RMSE
 - `actual=W`
 - `predict=linpred`

Part 1: Cross Validation by Location

- Chunk 5 (Continued)

```
RMSE.func = function(actual,predict){  
  mse=mean((actual-predict)^2,na.rm=T)  
  rmse=sqrt(mse)  
  return(rmse)  
}
```

```
RMSE.func(actual=DATA2$W,  
           predict=DATA2$linpred)
```



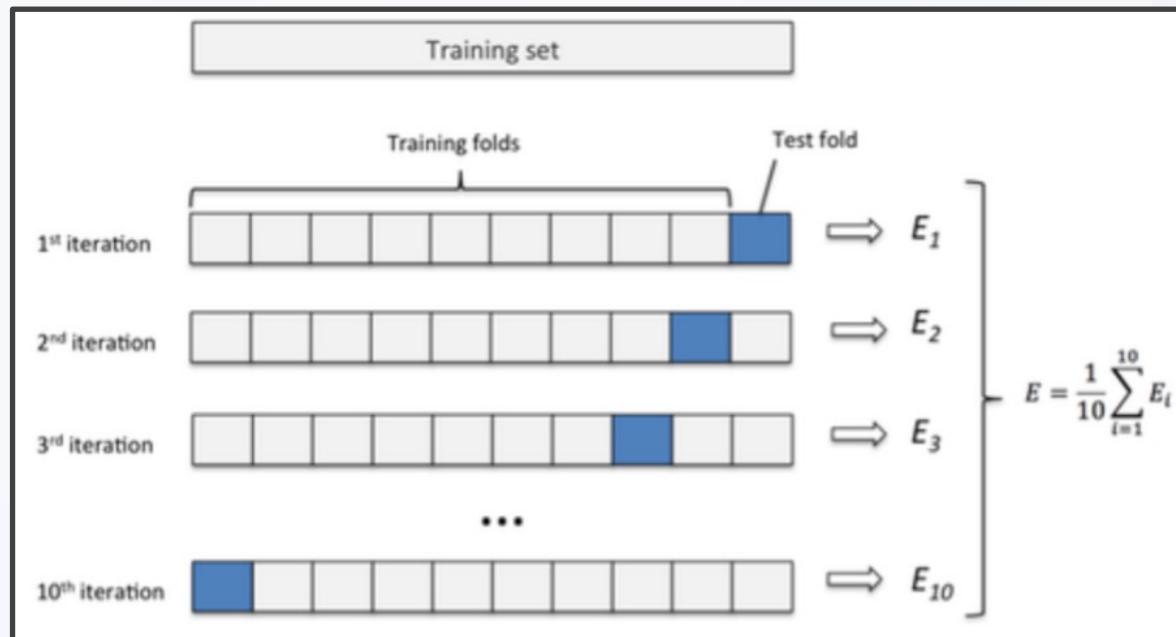
```
RMSE.func(actual=DATA2$W,predict=DATA2$linpred)  
[1] 3.147084
```

Intermission

- Current
 - Using the Natural Grouping of Data for 31-Fold Cross Validation
 - Only Fit One Linear Model
 - Should Use Cross-Validation for Multiple Different Models and Compare Cross-Validated RMSE
- Next
 - Randomly Assign Observations to K -Folds
 - CV Function: `crossv_kfold(K)`

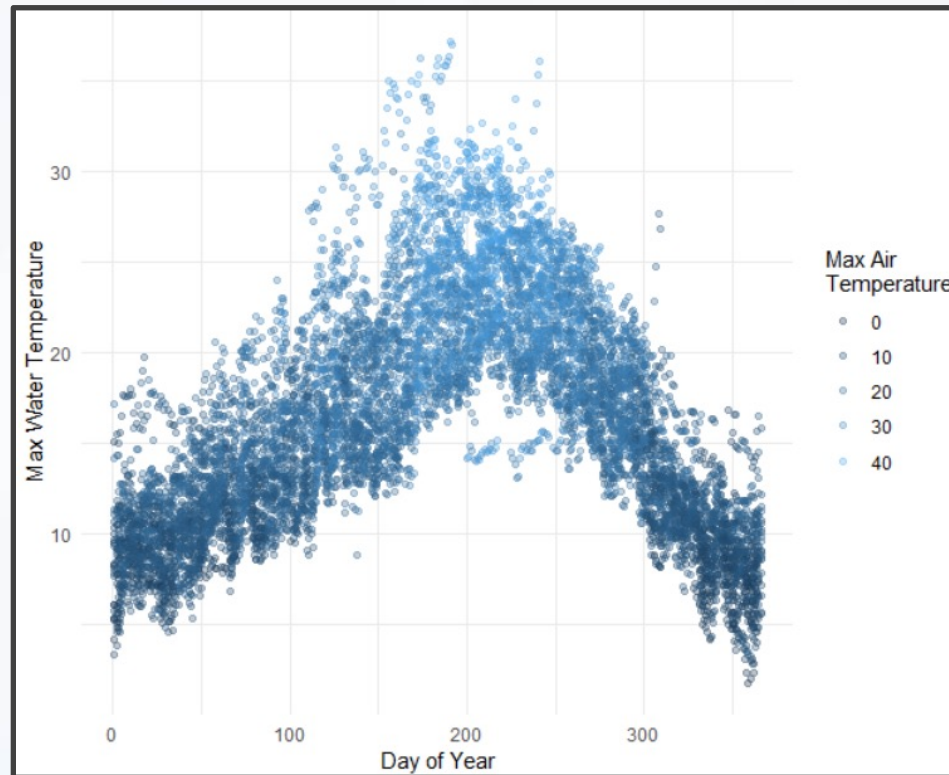
Part 2: K-Fold CV

- Overview ($K=10$)
 - Randomly Split Observations Into K Groups
 - Each Fold Acts as a Test Set
 - If Each Fold Contains Approximately the Same # of Observations,



Part 2: K-Fold CV

- Run Chunk 1
 - Variables (Julian Day)
 - Clear Non-Linear Relationship



Part 2: K-Fold CV

- General Polynomial Model

$$W = a + \sum_{i=1}^I b_i A^i + \sum_{j=1}^J c_j D^j + \varepsilon$$

- Perform K-Fold CV to Estimate Out-of-Sample RMSE for Choices of $I=4$ and $J=3$
- Ultimate Goal is To Select Best I and J

Part 2: K-Fold CV

- Run Chunk 2
 - Fit Model with $I=4$ and $J=3$
 - Functions from broom Package
 - tidy()
 - glance()
 - Used to Preview Models

```
tidy(polymodel)
A tibble: 8 x 5
  term                estimate std.error statistic  p.value
<chr>                <dbl>    <dbl>    <dbl>    <dbl>
(Intercept)          16.2      0.0273   595.      0.
poly(A, 4)1          328.       4.36     75.3     0.
poly(A, 4)2           49.0       2.80     17.5    1.62e-67
poly(A, 4)3            2.85       2.78      1.02    3.06e- 1
poly(A, 4)4           -3.62       2.72     -1.33    1.84e- 1
poly(JULIAN_DAY, 3)1  46.0       2.78     16.6    8.85e-61
poly(JULIAN_DAY, 3)2 -226.       4.31    -52.5     0.
poly(JULIAN_DAY, 3)3 -59.3       2.89    -20.5    8.66e-92
```

```
glance(polymodel)
A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic p.value  df logLik  AIC  BIC
<dbl>      <dbl> <dbl>    <dbl>    <dbl> <int> <dbl> <dbl> <dbl>
0.797      0.797  2.71     525.      0      8 -23804. 47626. 47691.
```


Part 2: K-Fold CV

- Run Chunk 3
 - Divide Data into 10 Folds
 - Use `crossv_kfold()` Function
 - Variables are Lists of Train and Test Sets
 - For Each Row, We Want to Fit on Train and Predict on Test

```
head(DATA3)
A tibble: 6 x 3
  train          test          .id
  <list>        <list>        <chr>
1 <S3: resample> <S3: resample> 01
2 <S3: resample> <S3: resample> 02
3 <S3: resample> <S3: resample> 03
4 <S3: resample> <S3: resample> 04
5 <S3: resample> <S3: resample> 05
6 <S3: resample> <S3: resample> 06
```

Part 2: K-Fold CV

- Run Chunk 4
 - Create Function to Fit Models
 - Apply Function to All Train Sets Using `purrr::map()` Function

```
DATA4=DATA3 %>%  
  mutate(tr.model=map(train,train.model.func,i=i,j=j))  
head(DATA4)  
A tibble: 6 x 4  
  train          test          .id  tr.model  
  <list>        <list>        <chr> <list>  
  <S3: resample> <S3: resample> 01    <S3: 1m>  
  <S3: resample> <S3: resample> 02    <S3: 1m>  
  <S3: resample> <S3: resample> 03    <S3: 1m>  
  <S3: resample> <S3: resample> 04    <S3: 1m>  
  <S3: resample> <S3: resample> 05    <S3: 1m>  
  <S3: resample> <S3: resample> 06    <S3: 1m>
```

- Functions from `purrr` Package
 - `map()` – Loop Over Train
 - `map2()` – Loop Over Fitted Models and Test

Part 2: K-Fold CV

- Run Chunk 5
 - purrr::map2() Iterates Function Over Two Arguments
 - For Every Test Set and Trained Model, We Use augment() to Get Predictions

```
DATA4.PREDICT = DATA4 %>%  
  mutate(predict=map2(test,tr.model,~augment(.y,newdata=.x))) %>%  
  select(predict) %>%  
  unnest()  
head(DATA4.PREDICT)  
A tibble: 6 x 10  
  JULIAN_DAY YEAR L W A TIME MONTH DAY .fitted .se.fit  
    <int> <int> <int> <dbl> <dbl> <int> <int> <int> <dbl> <dbl>  
1 9 2003 103 9.8 5.1 9 1 9 7.27 0.138  
2 12 2003 103 9.9 6.2 12 1 12 7.67 0.119  
3 25 2003 103 9.8 14 25 1 25 10.4 0.0744  
4 30 2003 103 9.5 9 30 1 30 9.14 0.0803  
5 47 2003 103 12.5 11.4 47 2 16 10.5 0.0621  
6 50 2003 103 10.7 14 50 2 19 11.5 0.0548
```

- Next, Compare Actual With Fitted Using RMSE.func()

```
RMSE.func(actual=DATA4.PREDICT$W,predict=DATA4.PREDICT$.fitted)  
[1] 2.709727
```

Look Ahead

- What We Have Done
 - Specify I and J
 - Use 10-Fold Cross Validation to Estimate Out-of-Sample RMSE
- How We Should Use This
 - Choose Max I and Max J
(Example: 10)
 - Initiate 10 x 10 Matrix of NA
 - Loop Through All i and j to Capture Out-of-Sample RMSE
 - Create a Tile Plot that Visualizes the RMSE for Each Combination of i and j
 - Choose Best i and j