Feature Engineering With Sliding Windows and Lagged Inputs

Bryan Shalloway

Data Scientist, NetAPP

2021-01-19

@brshallo

bryanshalloway.com

"Time changes everything except something within us that is always surprised by change."

Thomas Hardy

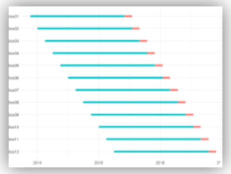
@brshallo

Agenda

- Splitting Data
- Building Features
- Putting it Together

For complete examples go:

Feature Engineering with Sliding Windows and Lagged Inputs 2020-10-12



The new rsample::sliding_*() functions bring the windowing approaches used in slider to the sampling procedures used in the tidymodels framework1. These functions make evaluation of models with ...

bryanshalloway.com

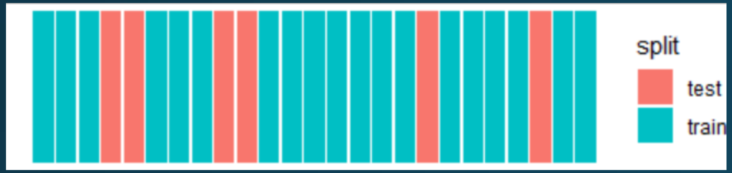
@brshallo

Data Splits

@brshallo

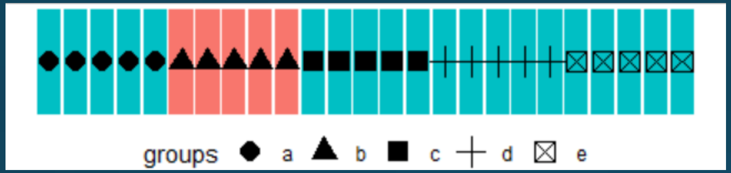
Types of Training – Testing Splits

Random



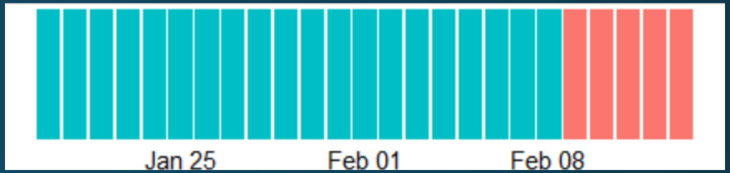
Most common
(or stratified by target)

Groups



e.g. customer transactions

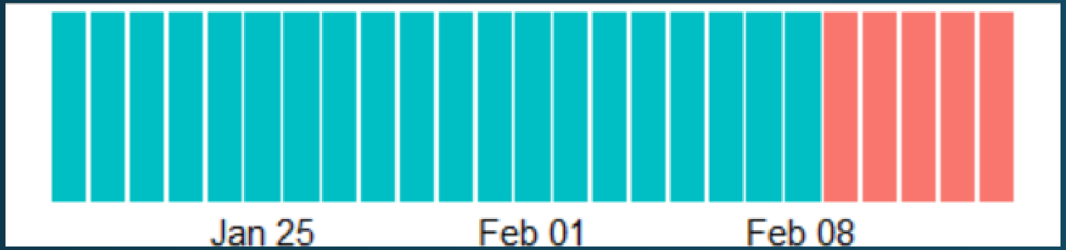
Time



e.g. forecasting

@brshallo

Time-based Split

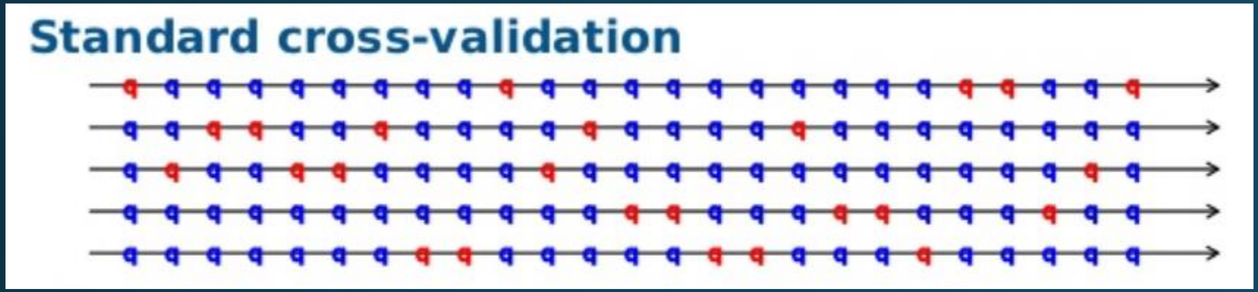


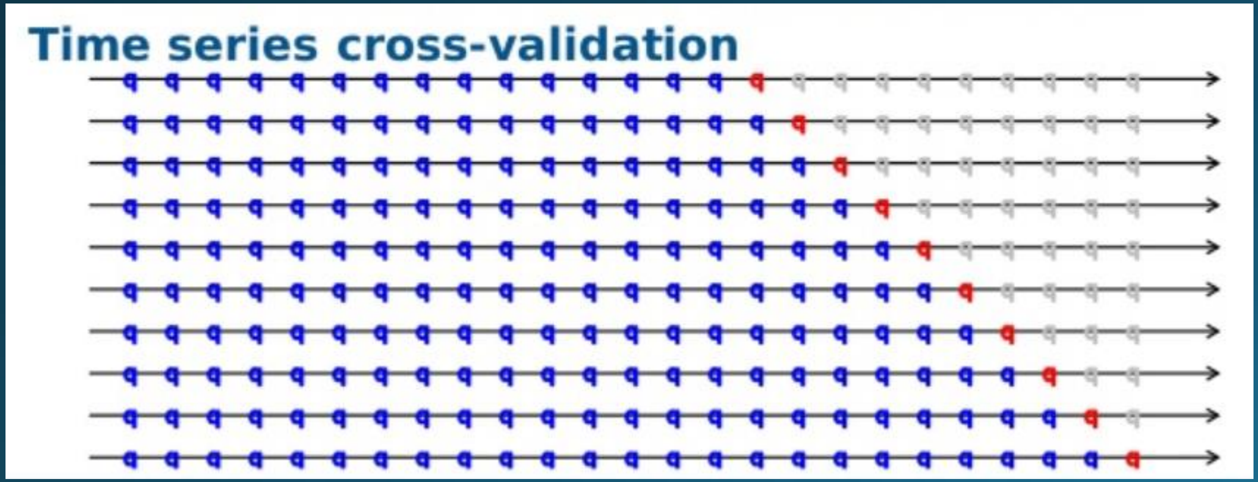
- Relationships in data often not independent across time
- Mirrors production

Drawbacks

- More thought and effort
- Model selection may not change
- Sacrifice data in some cross validation splits

Cross Validation





- Split data k (e.g. 5) times, each split has different obs in "test"
- Build models each → Measure performance each → Average performance across
- "Training" ; "Testing" AKA "Analysis" ; "Assessment"

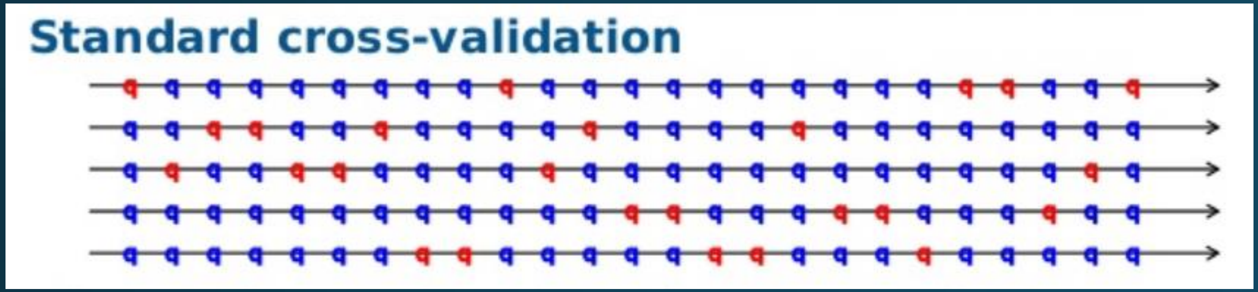
Time Series CV:

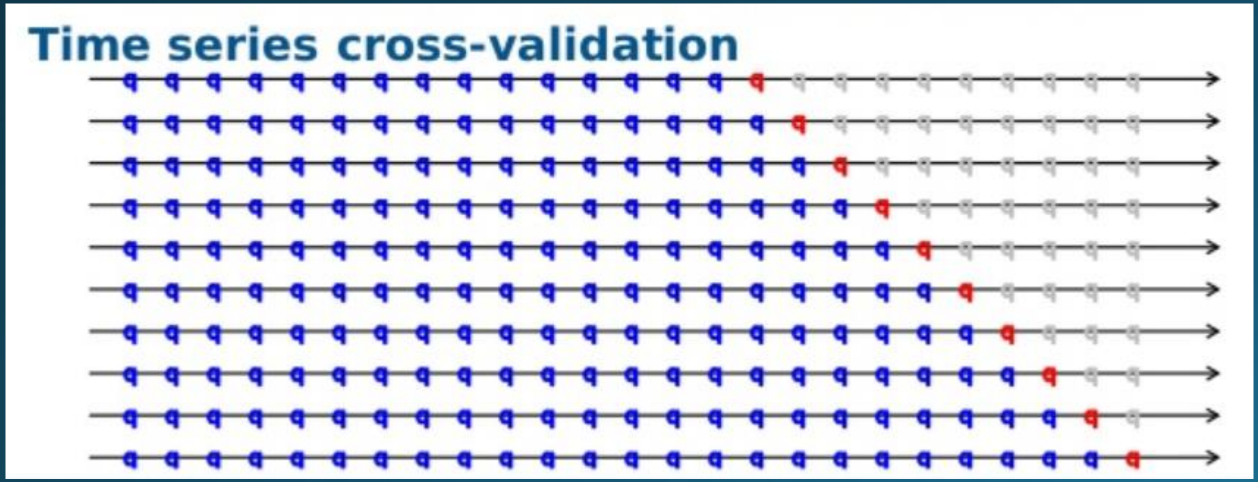
- SAME PROCESS, but splits are time based
- (Note data loss)

Image Source:
https://image.slidesharecdn.com/granada-140207061551-phpapp01/95/automatic-time-series-forecasting-71-638.jpg?cb=1392426574

@brshallo

Why / When it's used





- Allows for evaluation without touching "test"
- (Sometimes) used in place of separate test data
- May provide better estimate of performance
- Sense of variability

Downsides:

- Extra step, extra time (build many models)

Image Source: https://image.slidesharecdn.com/granada-140207061551-phpapp01/95/automatic-time-series-forecasting-71-638.jpg?cb=1392426574

@brshallo

Data: Wake County Food Inspections

HSISID	RESTAURANTOPENDATE	SCORE	INSPECTOR	date
<chr>	<date>	<dbl>	<chr>	<date>
04092017542	2017-03-01	94.5	Anne-Kathrin Bartoli	2017-04-07
04092017542	2017-03-01	92	Laura McNeill	2017-11-08
04092017542	2017-03-01	95	Laura McNeill	2018-03-23
04092017542	2017-03-01	93.5	Laura McNeill	2018-09-07
04092017542	2017-03-01	93	Joanne Rutkofske	2019-04-04
04092017542	2017-03-01	93.5	Naterra McQueen	2019-10-07
04092017542	2017-03-01	92.5	Naterra McQueen	2020-05-19
04092017542	2017-03-01	94	Nicole Bailey	2020-10-09
04092015321	2009-01-12	96	Jennifer Edwards	2013-07-31
04092015321	2009-01-12	96	Jennifer Edwards	2014-01-17

... with 25,537 more rows

- HSISID: restaurant ID
- SCORE: inspection score
- INSPECTOR: Name of person who performed inspection
- date: date of inspection

Target: SCORE

@brshallo

Examples

Setting-up splits

@brshallo

```r
library(tidyverse)
library(rsample)


inspections_restaurants <-
arrange(inspections_restaurants, date)

set.seed(1234)
initial_split <-
initial_time_split(inspections_restaurants,
                   prop = .8)

train <- training(initial_split)
test <- testing(initial_split)
```

```r
library(tidyverse)
library(rsample)



inspections_restaurants <-
arrange(inspections_restaurants, date)

set.seed(1234)
initial_split <-
initial_time_split(inspections_restaurants,
                   prop = .8)

train <- training(initial_split)
test <- testing(initial_split)
```

```
<Analysis/Assess/Total>
<17678/4420/22098>
```

@brshallo

```
library(tidyverse)
library(rsample)


inspections_restaurants <-
arrange(inspections_restaurants, date)

set.seed(1234)
initial_split <-
initial_time_split(inspections_restaurants,
                   prop = .8)

train <- training(initial_split)
test <- testing(initial_split)
```

@brshallo

rsample::sliding_window() → by position

rsample::sliding_index() → by index

rsample::sliding_period() → by index + split points by period

Examples

Setting-up splits, CV

@brshallo

```r
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

@brshallo

```
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

@brshallo

```r
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

```r
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

```r
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

```r
library(tidyverse)
library(rsample)


resamples <- sliding_period(
  data = train,
  index = date,
  period = "month",
  lookback = 36,
  assess_stop = 3,
  step = 3
)
```

```r
devtools::source_gist("https://gist.github.com/brshallo/7d180bde932628a151a4d935ffa586a5")
```

```r
resamples  %>%
  extract_dates_rset() %>%
  plot_dates_rset()
```

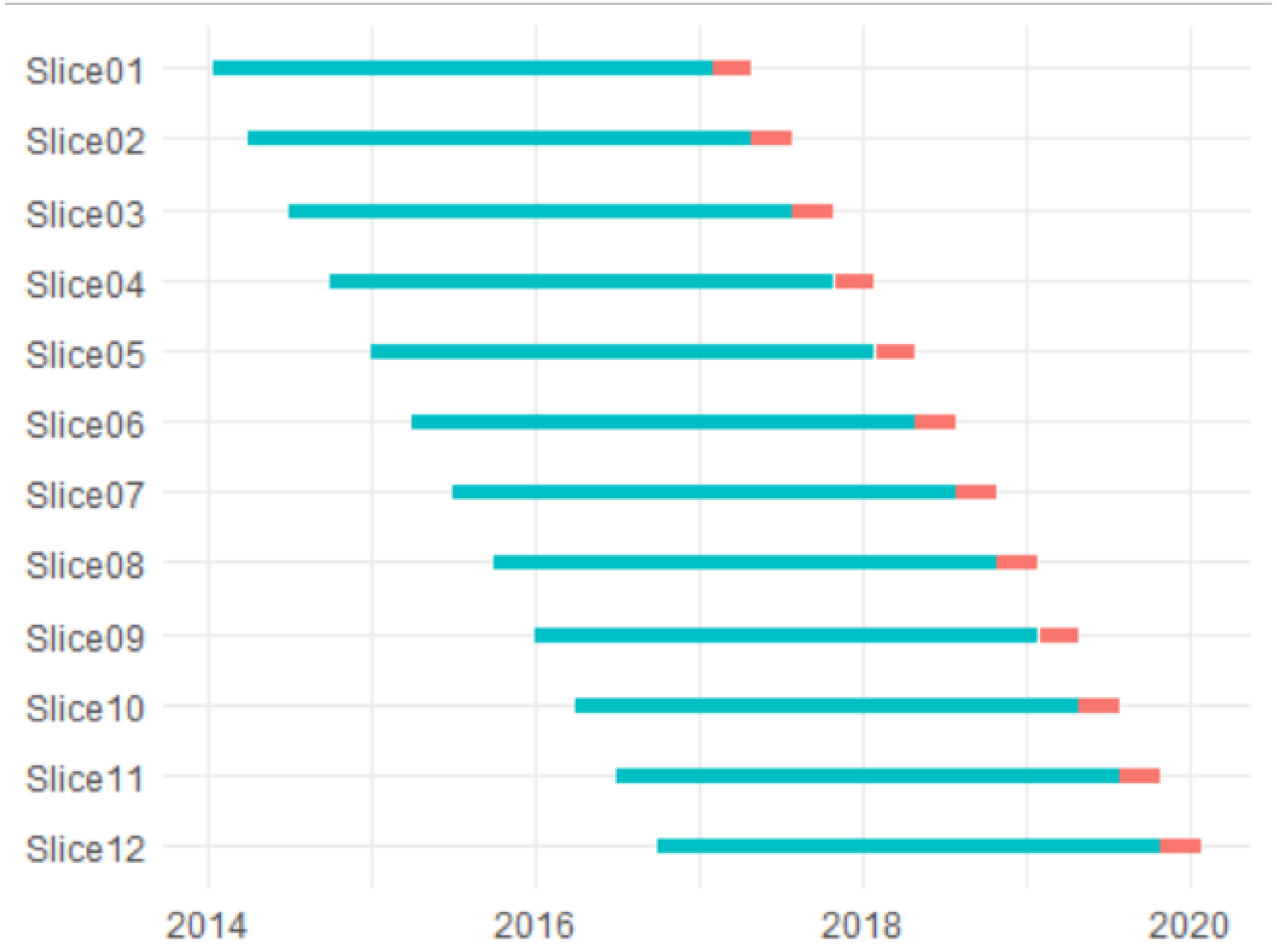
```
# Sliding period resampling
# A tibble: 12 x 2
   splits                id
   <list>                <chr>
 1 <split [6.4K/671]>    Slice01
 2 <split [6.7K/914]>    Slice02
 3 <split [7.2K/818]>    Slice03
 4 <split [7.5K/876]>    Slice04
 5 <split [7.9K/954]>    Slice05
 6 <split [8.3K/874]>    Slice06
 7 <split [8.6K/1K]>     Slice07
 8 <split [9K/999]>      Slice08
 9 <split [9.5K/1.1K]>   Slice09
10 <split [10K/942]>     Slice10
11 <split [10.4K/1K]>    Slice11
12 <split [10.8K/1.1K]>  Slice12
```

```r
devtools::source_gist("https://gist.github.com/brshallo/7d180bde932628a151a4d935ffa586a5")

resamples  %>%
  extract_dates_rset() %>%
  plot_dates_rset()
```

	splits	id	analysis_min	analysis_max	assessment_min	assessment_max
	<list>	<chr>	<date>	<date>	<date>	<date>
1	<split [6.4K/671]>	Slice01	2014-01-15	2017-01-31	2017-02-01	2017-04-28
2	<split [6.7K/914]>	Slice02	2014-04-01	2017-04-28	2017-05-01	2017-07-31
3	<split [7.2K/818]>	Slice03	2014-07-01	2017-07-31	2017-08-01	2017-10-31
4	<split [7.5K/876]>	Slice04	2014-10-01	2017-10-31	2017-11-01	2018-01-31
5	<split [7.9K/954]>	Slice05	2015-01-02	2018-01-31	2018-02-01	2018-04-30
6	<split [8.3K/874]>	Slice06	2015-04-01	2018-04-30	2018-05-01	2018-07-31
7	<split [8.6K/1K]>	Slice07	2015-07-01	2018-07-31	2018-08-01	2018-10-31
8	<split [9K/999]>	Slice08	2015-10-01	2018-10-31	2018-11-01	2019-01-31
9	<split [9.5K/1.1K]>	Slice09	2016-01-04	2019-01-31	2019-02-01	2019-04-30
10	<split [10K/942]>	Slice10	2016-04-01	2019-04-30	2019-05-01	2019-07-31
11	<split [10.4K/1K]>	Slice11	2016-07-01	2019-07-31	2019-08-01	2019-10-31
12	<split [10.8K/1.1K]>	Slice12	2016-10-03	2019-10-31	2019-11-01	2020-01-31

@brshallo

```
devtools::source_gist("https://gist.github.com/brshallo/7d180bde932628a151a4d935ffa586a5")

resamples  %>%
  extract_dates_rset() %>%
  plot_dates_rset()
```



@brshallo

Feature Engineering

@brshallo

Creating Features

- Careful of data leakage
- Two stages of feature engineering
    1. Time / lag-based features (prior to initial split)
    2. Other features and derivitive of raw data (after splitting, as part of a recipe)

1. Lag based features

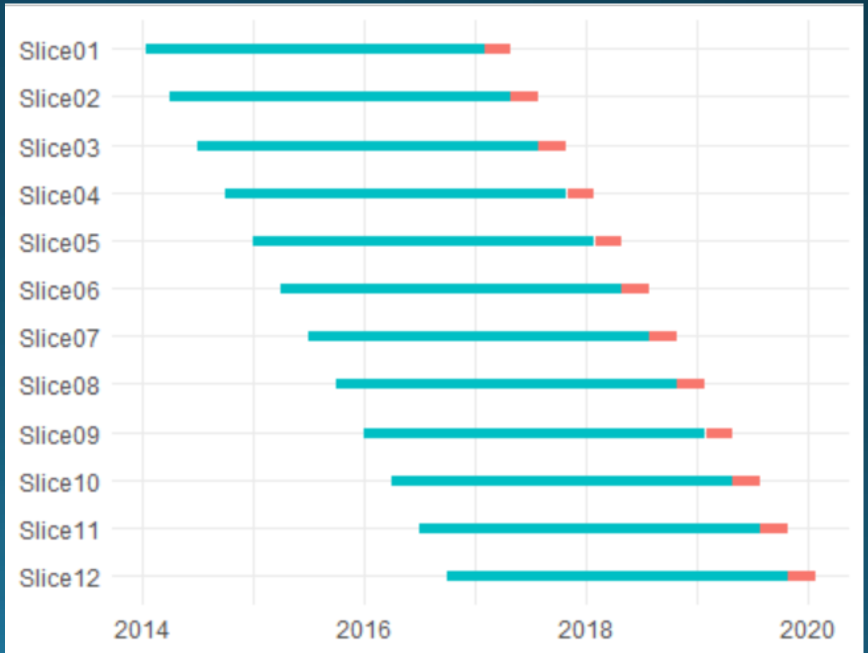
HSISID	RESTAURANTOPENDATE	SCORE	INSPECTOR	date
<chr>	<date>	<dbl>	<chr>	<date>
04092017542	2017-03-01	94.5	Anne-Kathrin Bartoli	2017-04-07
04092017542	2017-03-01	92	Laura McNeill	2017-11-08
04092017542	2017-03-01	95	Laura McNeill	2018-03-23
04092017542	2017-03-01	93.5	Laura McNeill	2018-09-07
04092017542	2017-03-01	93	Joanne Rutkofske	2019-04-04
04092017542	2017-03-01	93.5	Naterra McQueen	2019-10-07
04092017542	2017-03-01	92.5	Naterra McQueen	2020-05-19
04092017542	2017-03-01	94	Nicole Bailey	2020-10-09
04092015321	2009-01-12	96	Jennifer Edwards	2013-07-31
04092015321	2009-01-12	96	Jennifer Edwards	2014-01-17

... with 25,537 more rows

- Average SCORE across HSISID's for last year
- Average SCORE for particular HSISID over last 3 years

Consider which features may predict upcoming score?

- Most recent score
- How long ago the restaurant was opened
- How long since last inspection



@brshallo

slider::sliding_window() → by position

slider::sliding_index() → by index

slider::sliding_period() → by index + split points by period

Example

Creating lag-based features

@brshallo

```
library(tidyverse)
library(slider)



data_time_feats <- inspections_restaurants %>%
  group_by(HSISID) %>%
  mutate(
    SCORE_recent = slide_index_dbl(
      .x = SCORE,
      .i = date,
      .f = mean,
      na.rm = TRUE,
      .before = lubridate::days(365 * 3),
      .after = -lubridate::days(1),
      .complete = FALSE
    )
```

@brshallo

```r
library(tidyverse)
library(slider)


data_time_feats <- inspections_restaurants %>%
  group_by(HSISID) %>%
  mutate(
    SCORE_recent = slide_index_dbl(
      .x = SCORE,
      .i = date,
      .f = mean,
      na.rm = TRUE,
      .before = lubridate::days(365 * 3),
      .after = -lubridate::days(1),
      .complete = FALSE
    )
```

```
library(tidyverse)
library(slider)


data_time_feats <- inspections_restaurants %>%
  group_by(HSISID) %>%
  mutate(
    SCORE_recent = slide_index_dbl(
      .x = SCORE,
      .i = date,
      .f = mean,
      na.rm = TRUE,
      .before = lubridate::days(365 * 3),
      .after = -lubridate::days(1),
      .complete = FALSE
    )
```

@brshallo

New Data With Lag Based Features

HSISID	SCORE_yr_overall	SCORE_lag	SCORE_recent	days_since_open	days_since_last
04092015474	96.21098	98.5	98.50000	1574	362
04092014996	96.21098	97.5	97.25000	2274	145
04092011465	96.21098	95.0	95.00000	7239	167
04092011770	96.21098	98.0	96.16667	6644	114
04092014360	96.21098	97.0	98.00000	3089	258

@brshallo

Apply initial split, then CV splits (shown previously)

2. Other and derivative features

- Box-cox transform continuous predictors
- Inspector (collapse rare levels)
- Parts of date (e.g. month, day of week, etc.)
- …

Use recipes

- Identify feature parameters on training dataset
- Then Apply to test set
- (Works well with rest of tidymodels)

```r
library(tidyverse)
library(recipes)


rec <- recipe(SCORE ~ ., data = train) %>%
  step_BoxCox(days_since_open, days_since_last) %>%
  step_other(INSPECTOR, TYPE, threshold = 50) %>%
  step_novel(TYPE, INSPECTOR) %>%
  step_date(date, features = c("dow", "month"))
```

```r
library(tidyverse)
library(recipes)


rec <- recipe(SCORE ~ ., data = train) %>%
  step_BoxCox(days_since_open, days_since_last) %>%
  step_other(INSPECTOR, TYPE, threshold = 50) %>%
  step_novel(TYPE, INSPECTOR) %>%
  step_date(date, features = c("dow", "month"))
```

@brshallo

```
library(tidyverse)
library(recipes)


rec <- recipe(SCORE ~ ., data = train) %>%
  step_BoxCox(days_since_open, days_since_last) %>%
  step_other(INSPECTOR, TYPE, threshold = 50) %>%
  step_novel(TYPE, INSPECTOR) %>%
  step_date(date, features = c("dow", "month"))
```

@brshallo

Putting it together

```r
library(tidyverse)
library(tidymodels)


lm_mod <- parsnip::linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_workflow_rs <- workflows::workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(rec) %>%
  fit_resamples(resamples,
                control = control_resamples(save_pred = T))
```

```r
library(tidyverse)
library(tidymodels)


lm_mod <- parsnip::linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_workflow_rs <- workflows::workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(rec) %>%
  fit_resamples(resamples,
                control = control_resamples(save_pred = T))
```

```r
library(tidyverse)
library(tidymodels)


lm_mod <- parsnip::linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_workflow_rs <- workflows::workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(rec) %>%
  fit_resamples(resamples,
                control = control_resamples(save_pred = T))
```

@brshallo

```
library(tidyverse)
library(tidymodels)


lm_mod <- parsnip::linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_workflow_rs <- workflows::workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(rec) %>%
  fit_resamples(resamples,
                control = control_resamples(save_pred = T))
```

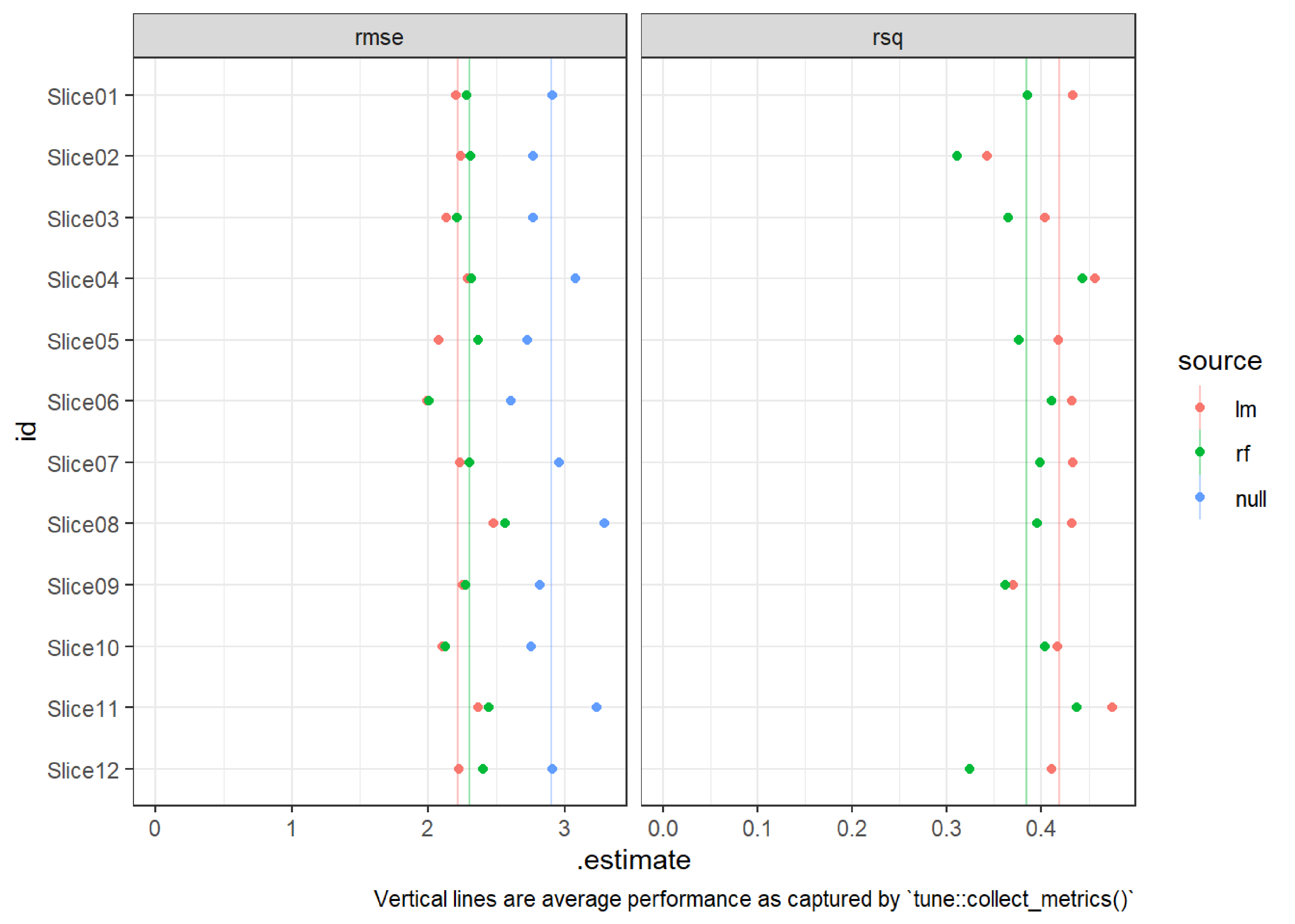
@brshallo

Do same for other models (Null and Random Forest)

Extract performance

Compare performance

@brshallo



paired t-test to provide indicator of if difference is real

@brshallo

Recap

- Be careful with how you set-up your data-splits, especially concerning time
- If appropriate, use cross-validation
- Build 'time-based features' in pre-processing
- Build 'other features' with a recipe
- Try multiple models
- Review performance across and within splits

- Use `tidyverse` and `tidymodels` suite of packages for intuitive, consistent style

Package Websites

- dplyr: https://dplyr.tidyverse.org/
- slider: https://davisvaughan.github.io/slider/
- rsample: https://rsample.tidymodels.org/
- recipes: https://recipes.tidymodels.org/
- parsnip: https://parsnip.tidymodels.org/
- workflows: https://workflows.tidymodels.org/

- yardstick: https://yardstick.tidymodels.org/
- ggplot2: https://ggplot2.tidyverse.org/

Other Resources

- https://www.bryanshalloway.com/2020/10/12/window-functions-for-resampling/
- tmwr.org
- tidymodels.org
- feat.engineering
- business-science (timetk, modeltime, etc.)
- tidyverts (fable, tsibble, etc.)