04_AutoML_example2

June 27, 2019

1 AutoML and Ensemble Learning with H2O.ai

- Introduction to H2O ai
- getting started
- AutoML workflow
- Performance and Prediction
- AutoML variable importance
- Available algorithms

```
In [1]: library( readr )
        library( dplyr )
        library( GGally )
```

```
Attaching package: dplyr

The following objects are masked from package:stats:

    filter, lag

The following objects are masked from package:base:

    intersect, setdiff, setequal, union

Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2

Attaching package: GGally

The following object is masked from package:dplyr:
```

```
    nasa
```

2 Unequal variance data example

```
In [2]: unequal_var_data <- read_csv( "unequal_variance_data.csv" ) %>%
            mutate( Y = factor( Y ) )
```

```
Parsed with column specification:
cols(
  `0` = col_double(),
  `1` = col_double(),
  `2` = col_double(),
  `3` = col_double(),
  `4` = col_double(),
  `5` = col_double(),
  `6` = col_double(),
  `7` = col_double(),
  `8` = col_double(),
  `9` = col_double(),
  Y = col_double()
)
```

```
In [3]: library( skimr )
```

```
Attaching package: skimr

The following object is masked from package:stats:

    filter
```

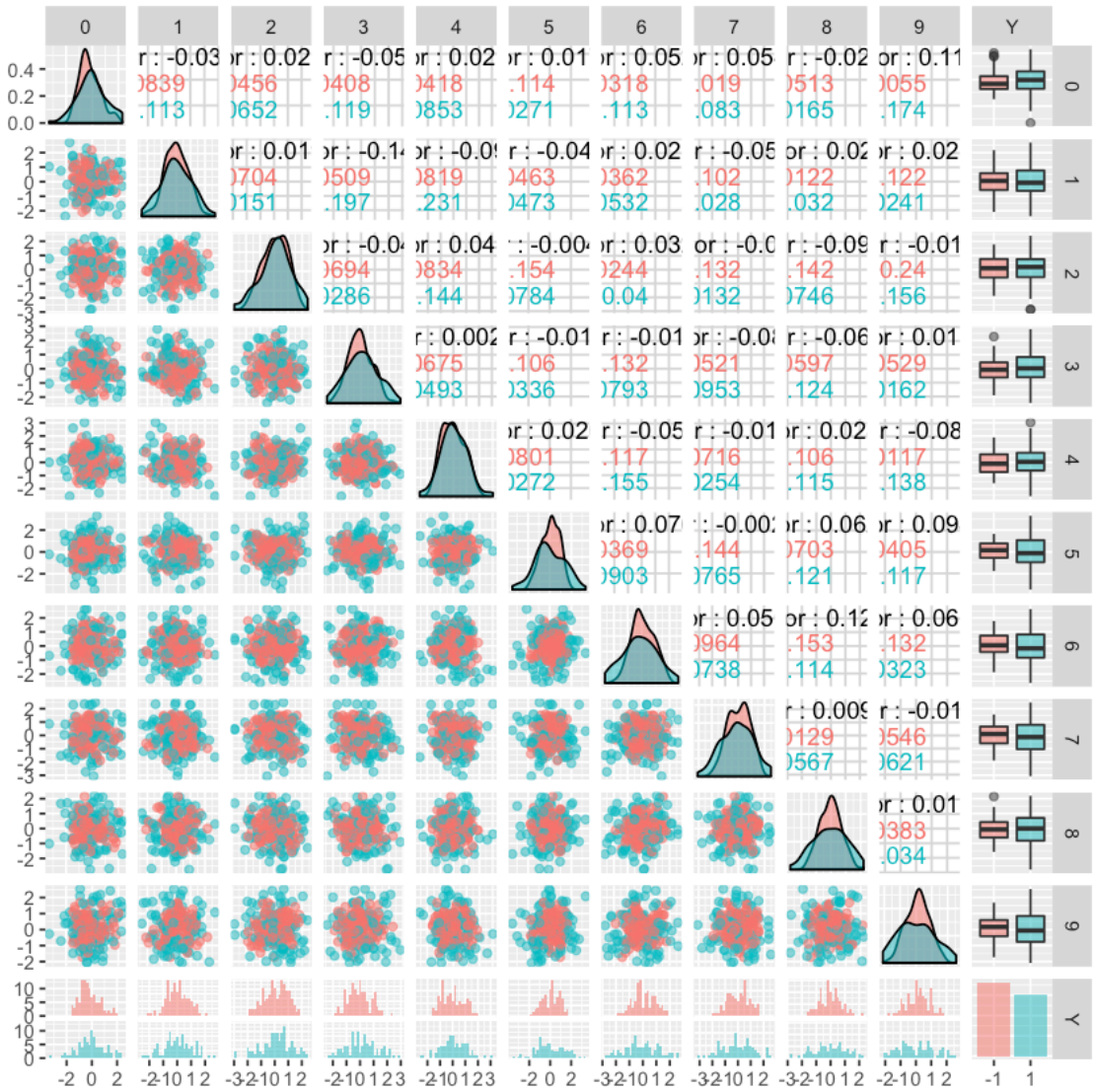
```
In [5]: skim_to_wide( unequal_var_data )
```

A tibble: 11 Œ 16

type <chr>	variable <chr>	missing <chr>	complete <chr>	n <chr>	n_unique <chr>	top_counts <chr>	orde <chr
factor	Y	0	200	200	2	-1: 109, 1: 91, NA: 0	FALS
numeric	0	0	200	200	NA	NA	NA
numeric	1	0	200	200	NA	NA	NA
numeric	2	0	200	200	NA	NA	NA
numeric	3	0	200	200	NA	NA	NA
numeric	4	0	200	200	NA	NA	NA
numeric	5	0	200	200	NA	NA	NA
numeric	6	0	200	200	NA	NA	NA
numeric	7	0	200	200	NA	NA	NA
numeric	8	0	200	200	NA	NA	NA
numeric	9	0	200	200	NA	NA	NA

```
In [6]: ggpairs( unequal_var_data, aes( alpha=0.1, color=Y ) )
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [7]: library(h2o)
```

```
----------------------------------------------------------------------

Your next step is to start H2O:
    > h2o.init()

For H2O package documentation, ask for help:
    > ??h2o

After starting H2O, you can use the Web UI at http://localhost:54321
For more information visit http://docs.h2o.ai
```

```
----------------------------------------------------------------------


Attaching package: h2o

The following objects are masked from package:stats:

    cor, sd, var

The following objects are masked from package:base:

    &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
    colnames<-, ifelse, is.character, is.factor, is.numeric, log,
    log10, log1p, log2, round, signif, trunc
```

```
In [8]: h2o.init()
```

```
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         4 minutes 4 seconds
    H2O cluster timezone:       America/New_York
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.24.0.5
    H2O cluster version age:    8 days
    H2O cluster name:           H2O_started_from_R_colettace_yhm690
    H2O cluster total nodes:    1
    H2O cluster total memory:   3.42 GB
    H2O cluster total cores:    8
    H2O cluster allowed cores:  8
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.6.0 (2019-04-26)
```

```
In [9]: unequal_var_data = as.h2o( unequal_var_data )
```

```
  |======================================================================| 100%
```

```
In [10]: class( unequal_var_data )
```

'H2OFrame'

```
In [11]: summary( unequal_var_data )
```

```
Warning message in summary.H2OFrame(unequal_var_data):
Approximated quantiles computed! If you are interested in exact quantiles, please pass the `exa
```

```
 0                 1                 2                 3
 Min.   :-3.3010   Min.   :-2.38189   Min.   :-2.83698   Min.   :-2.41305
 1st Qu.:-0.7189   1st Qu.:-0.61613   1st Qu.:-0.56368   1st Qu.:-0.60635
 Median :-0.1952   Median :-0.01907   Median : 0.14354   Median :-0.05626
 Mean   :-0.1203   Mean   : 0.01376   Mean   : 0.09163   Mean   : 0.01124
 3rd Qu.: 0.4310   3rd Qu.: 0.63388   3rd Qu.: 0.74487   3rd Qu.: 0.64244
 Max.   : 2.3152   Max.   : 2.69944   Max.   : 2.39199   Max.   : 2.80109
 4                   5                   6                   7
 Min.   :-2.577600   Min.   :-3.44532   Min.   :-2.632282   Min.   :-3.069000
 1st Qu.:-0.720913   1st Qu.:-0.70444   1st Qu.:-0.623167   1st Qu.:-0.697988
 Median :-0.030965   Median : 0.05045   Median :-0.021879   Median : 0.015672
 Mean   : 0.008994   Mean   : 0.05048   Mean   : 0.005324   Mean   :-0.008708
 3rd Qu.: 0.699650   3rd Qu.: 0.89821   3rd Qu.: 0.755715   3rd Qu.: 0.689145
 Max.   : 3.031727   Max.   : 3.30979   Max.   : 2.630626   Max.   : 2.473987
 8                  9                   Y
 Min.   :-2.74964   Min.   :-2.08380    -1:109
 1st Qu.:-0.66207   1st Qu.:-0.60256    1 : 91
 Median :-0.01489   Median : 0.13286
 Mean   :-0.03836   Mean   : 0.08194
 3rd Qu.: 0.58403   3rd Qu.: 0.73321
 Max.   : 2.20013   Max.   : 2.53424
```

2.1 Split into train val test

- H2O you give your "desired" train/val/test ratios and it gives you back approximately the proportions you want
- E.g., Say we want 500 samples for training, 100 samples for validation and 400 samples for test data:

```
In [12]: unequal_splits <- h2o.splitFrame( unequal_var_data, ratios=c(0.5) )
```

```
In [13]: unequal_train <- unequal_splits[[1]]
         unequal_test <- unequal_splits[[2]]
```

```
In [14]: dim( unequal_train )
```

1. 109 2. 11

```
In [15]: dim( unequal_test )
```

1. 91 2. 11

```
In [16]: library(tictoc)
```

```
In [17]: tic()
         aml_results <- h2o.automl(
             # x is omitted since we want to use all the columns except "Y" as predictors
             y = 'Y',
             training_frame = unequal_train,
             leaderboard_frame = unequal_test,
             max_runtime_secs = 120,
             exclude_algos = 'GBM',
         )
         toc()
```

```
  |======================================================================| 100%
112.521 sec elapsed
```

2.2 AutoML results

- Printing the results object shows you info from the winning "leader" model, and well as the "leaderboard" of how well the various models performed

```
In [18]: dim( aml_results@leaderboard )
```

1. 26 2. 6

```
In [19]: as.data.frame( aml_results@leaderboard  )
```

A df[,6]: 26 Œ 6

model_id <chr>	auc <dbl>	logloss <dbl>	mea <dbl
StackedEnsemble_BestOfFamily_AutoML_20190627_121345	0.8590909	0.5101965	0.21
DeepLearning_grid_1_AutoML_20190627_121345_model_5	0.8348485	0.5894617	0.18
StackedEnsemble_AllModels_AutoML_20190627_121345	0.8242424	0.5238543	0.21
XGBoost_grid_1_AutoML_20190627_121345_model_1	0.8111111	0.5449321	0.24
XRT_1_AutoML_20190627_121345	0.8000000	0.5499661	0.23
DRF_1_AutoML_20190627_121345	0.7914141	0.5408793	0.26
DeepLearning_grid_1_AutoML_20190627_121345_model_1	0.7777778	0.5968758	0.26
XGBoost_grid_1_AutoML_20190627_121345_model_6	0.7575758	0.6706766	0.25
XGBoost_grid_1_AutoML_20190627_121345_model_2	0.7388889	0.6208542	0.26
XGBoost_3_AutoML_20190627_121345	0.7368687	0.6340328	0.27
XGBoost_grid_1_AutoML_20190627_121345_model_5	0.6964646	0.6524089	0.40
XGBoost_1_AutoML_20190627_121345	0.6272727	0.6680748	0.34
XGBoost_grid_1_AutoML_20190627_121345_model_4	0.6151515	0.6750542	0.39
DeepLearning_grid_1_AutoML_20190627_121345_model_3	0.5762626	0.7792168	0.50
DeepLearning_1_AutoML_20190627_121345	0.5757576	0.7314827	0.49
DeepLearning_grid_1_AutoML_20190627_121345_model_6	0.5560606	1.0172884	0.48
XGBoost_grid_1_AutoML_20190627_121345_model_3	0.5358586	0.6993592	0.46
DeepLearning_grid_1_AutoML_20190627_121345_model_8	0.5323232	0.8493258	0.50
DeepLearning_grid_1_AutoML_20190627_121345_model_7	0.5323232	3.1488140	0.48
XGBoost_grid_1_AutoML_20190627_121345_model_7	0.5252525	0.6971028	0.45
GLM_grid_1_AutoML_20190627_121345_model_1	0.5227273	0.6947986	0.50
XGBoost_2_AutoML_20190627_121345	0.5000000	0.6926350	0.50
DeepLearning_grid_1_AutoML_20190627_121345_model_4	0.4969697	0.9189918	0.50
DeepLearning_grid_1_AutoML_20190627_121345_model_9	0.4964646	1.3594697	0.50
DeepLearning_grid_1_AutoML_20190627_121345_model_10	0.4924242	0.9334537	0.50
DeepLearning_grid_1_AutoML_20190627_121345_model_2	0.4328283	1.1307177	0.50

```
In [ ]: h2o.performance( aml_results@leader )
```

```
In [20]: getParms( aml_results@leader )
         # or a synonym:
         # aml_results@leader@parameters
```

$model_id 'StackedEnsemble_BestOfFamily_AutoML_20190627_121345'

$training_frame 'automl_training_RTMP_sid_971b_3'

$base_models 1. **$'__meta' $schema_version** 3
$schema_name 'ModelKeyV3'
$schema_type 'Key<Model>'
$name 'DeepLearning_grid_1_AutoML_20190627_121345_model_5'
$type 'Key<Model>'
$URL '/3/Models/DeepLearning_grid_1_AutoML_20190627_121345_model_5'

2. **$'__meta' $schema_version** 3
$schema_name 'ModelKeyV3'
$schema_type 'Key<Model>'

$name 'XGBoost_grid_1_AutoML_20190627_121345_model_1'
$type 'Key<Model>'
$URL '/3/Models/XGBoost_grid_1_AutoML_20190627_121345_model_1'

3. **$'__meta' $schema_version** 3
   $schema_name 'ModelKeyV3'
   $schema_type 'Key<Model>'
   $name 'XRT_1_AutoML_20190627_121345'
   $type 'Key<Model>'
   $URL '/3/Models/XRT_1_AutoML_20190627_121345'
4. **$'__meta' $schema_version** 3
   $schema_name 'ModelKeyV3'
   $schema_type 'Key<Model>'
   $name 'DRF_1_AutoML_20190627_121345'
   $type 'Key<Model>'
   $URL '/3/Models/DRF_1_AutoML_20190627_121345'
5. **$'__meta' $schema_version** 3
   $schema_name 'ModelKeyV3'
   $schema_type 'Key<Model>'
   $name 'GLM_grid_1_AutoML_20190627_121345_model_1'
   $type 'Key<Model>'
   $URL '/3/Models/GLM_grid_1_AutoML_20190627_121345_model_1'

$metalearner_nfolds 5

$seed '-9152382088731368410'

$keep_levelone_frame TRUE

$x 1. '0' 2. '1' 3. '2' 4. '3' 5. '4' 6. '5' 7. '6' 8. '7' 9. '8' 10. '9'

$y 'Y'

```
In [27]: Y_pred <- as.data.frame( predict( aml_results@leader, unequal_test ) )
```

```
  |======================================================================| 100%
```

```
In [28]: Y_pred
```

predict <fct>	p.1 <dbl>	p1 <dbl>
1	0.5470479	0.4529521
1	0.3189163	0.6810837
1	0.2299066	0.7700934
1	0.1745561	0.8254439
-1	0.7220257	0.2779743
1	0.1884686	0.8115314
1	0.2530447	0.7469553
1	0.2374462	0.7625538
1	0.1561083	0.8438917
1	0.2198686	0.7801314
1	0.5988607	0.4011393
-1	0.7200160	0.2799840
1	0.3962069	0.6037931
1	0.5526435	0.4473565
1	0.2031153	0.7968847
-1	0.8262724	0.1737276
-1	0.8054580	0.1945420
1	0.2117329	0.7882671
1	0.2456118	0.7543882
1	0.4984435	0.5015565
-1	0.7057647	0.2942353
-1	0.6082703	0.3917297
-1	0.6091654	0.3908346
1	0.2030139	0.7969861
-1	0.8229378	0.1770622
1	0.3133801	0.6866199
1	0.3613497	0.6386503
1	0.1925685	0.8074315
-1	0.7873900	0.2126100
-1	0.6114910	0.3885090
1	0.3183629	0.6816371
-1	0.8238490	0.1761510
-1	0.7082807	0.2917193
-1	0.8685050	0.1314950
1	0.1696189	0.8303811
1	0.3841467	0.6158533
-1	0.7693309	0.2306691
1	0.1802563	0.8197437
1	0.4352397	0.5647603
1	0.1924819	0.8075181
1	0.4461558	0.5538442
-1	0.7112040	0.2887960
1	0.4330186	0.5669814
1	0.4773919	0.5226081
1	0.4814221	0.5185779
-1	0.8827244	0.1172756
1	0.2335976	0.7664024
1	0.2912440	0.7087560
-1	0.7905401	0.2094599
1	0.4995243	0.5004757

A df[,3]: 91 Œ 3

2.3 Visualize mistakes

- HC = High-variance correct
- HI = High-variance incorrect
- LC = Low-variance correct
- LI = Low-variance incorrect

```
In [62]: test_data <- as.data.frame( unequal_test )
```

```
In [63]: dim( test_data )
```

1. 91 2. 11

```
In [64]: dim( Y_pred )
```

1. 91 2. 3

```
In [65]: library( tibble )
```

```
In [66]: prob1 <- Y_pred$p1
```

```
In [71]: aug <- test_data %>%
             cbind( prob1 ) %>%
             mutate( Y=if_else( as.character( Y ) == '-1', 0, 1 ) ) %>%
             mutate( prob1_round=round( prob1 ) )
```

```
In [71]: aug <- test_data %>%
             cbind( prob1 ) %>%
             mutate( Y=if_else( as.character( Y ) == '-1', 0, 1 ) ) %>%
             mutate( prob1_round=round( prob1 ) )
```

```
In [89]: aug$class <- 'LC'
```

```
In [90]: aug[ (aug$Y == 1) & (aug$prob1_round == 1), 'class' ] <- 'HC'
         aug[ (aug$Y == 1) & (aug$prob1_round == 0), 'class' ] <- 'HI'
         aug[ (aug$Y == 0) & (aug$prob1_round == 1), 'class' ] <- 'LI'
```

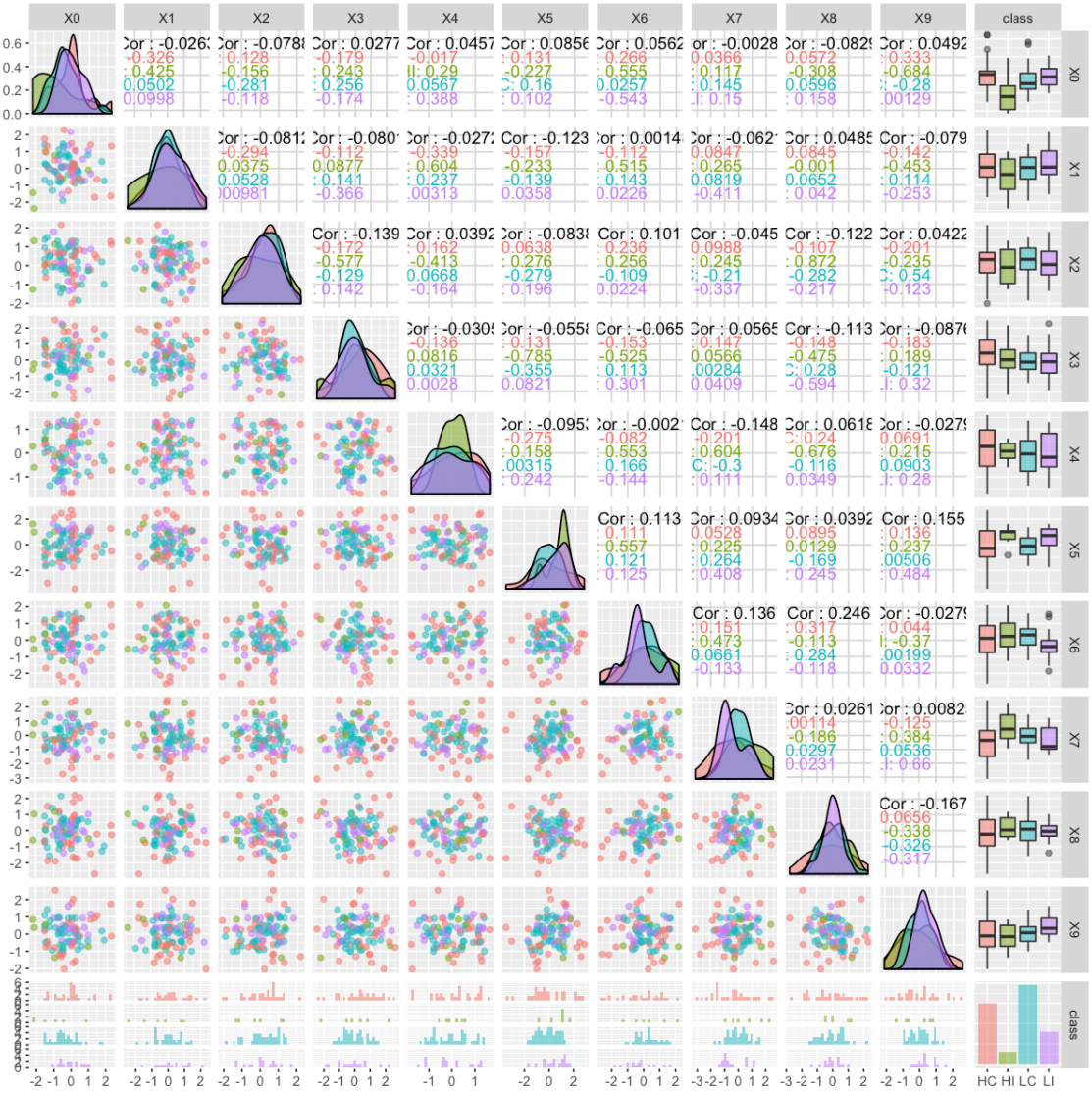
```
In [91]: aug$class <- factor( aug$class )
```

```
In [92]: options( repr.plot.width=10, repr.plot.height=10 )
```

```
In [93]: aug %>%
             select( -Y, -prob1, -prob1_round ) %>%
             ggpairs( aes( color=class, alpha=0.1))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In [ ]: