Bias-variance Tradeoff

Data Splitting and Resampling

Joe Nese

Week 2, Class 1

Agenda

- Understanding the bias-variance tradeoff
- Data splitting and why it matters
- Introduce resampling methods

Bias Variance Trade-off

Bias Variance Trade-off

- The goal of machine learning is to predict results based on new (unseen) data
- We use existing data to teach the machine how to predict results for new (unseen) data
- We want the best predictions, and we generally do this by minimizing prediction error
- Two types of prediction error:
  - Bias
  - Variance

Bias

- The difference between the (average) prediction of our model and the true value we are trying to predict
  - This is equivalent to statistical bias
  - Does not have to be average
- Gives us an idea how well a model fits the underlying structure of the data
- A model with low bias is providing predictions close to “truth”
- A model with high bias systematically ignores relevant details in the data

Variance

- The variability of a model prediction for a given data point
  - A measure of the variability of predictions if we <u>repeat the model multiple times with small differences in the data</u>
  - The more the model fits to small differences in data, the higher the variance
- Highly flexible models are more prone to higher variance
- Highly flexible models are more prone to overfitting to the (training) data
  - Results in very good performance measures, BUT
  - Poor generalizability to new (unseen) data

Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

outcome

predicted
outcome

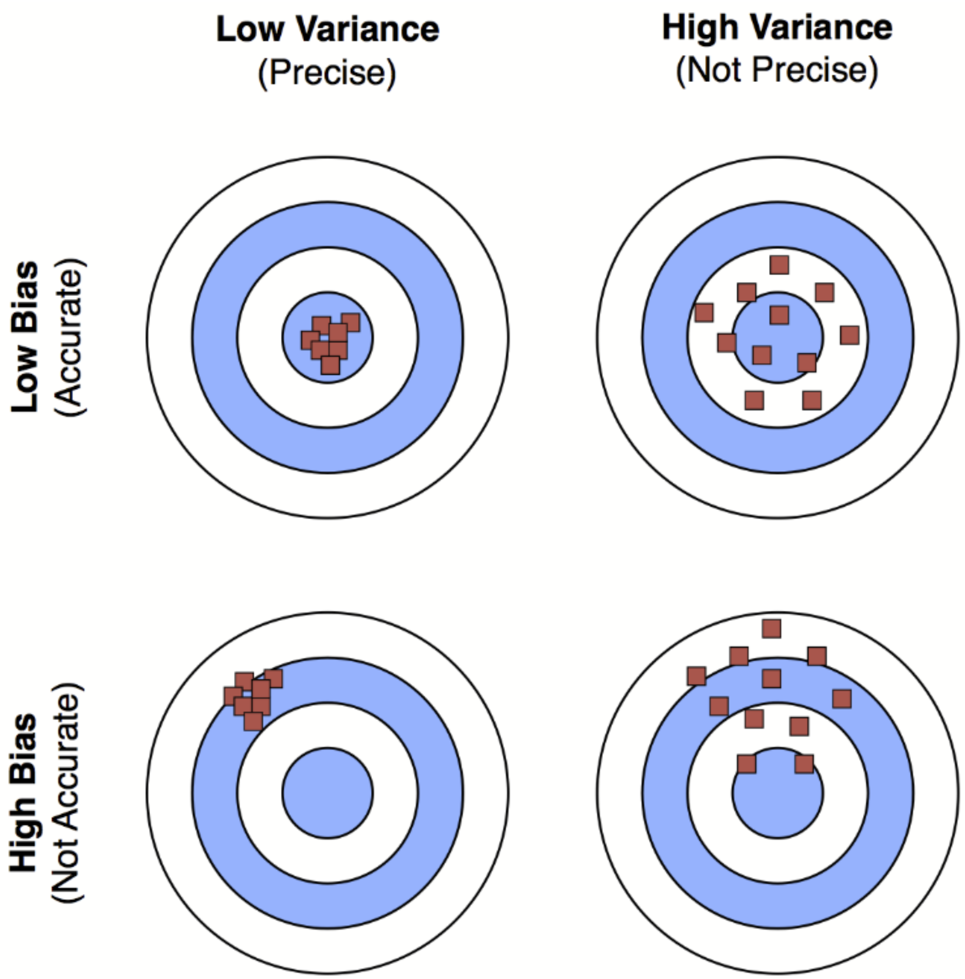
Mean Square Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

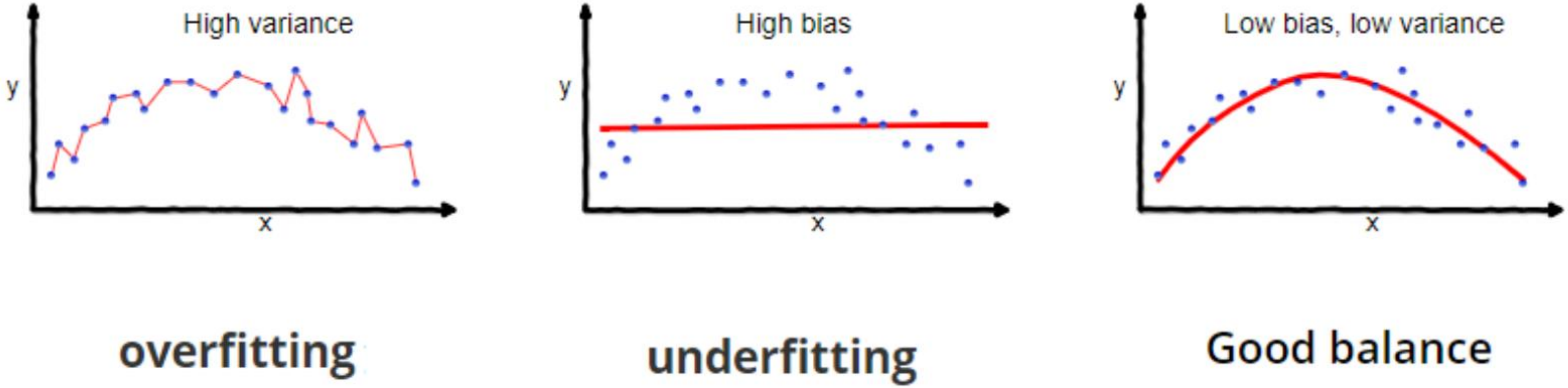
$$Estimated\ MSE = \sigma^2 + (Model\ Bias)^2 + Model\ Variance$$

variance of residuals
this is the "noise" in the data
(unaffected by modeling)

One way to look at it



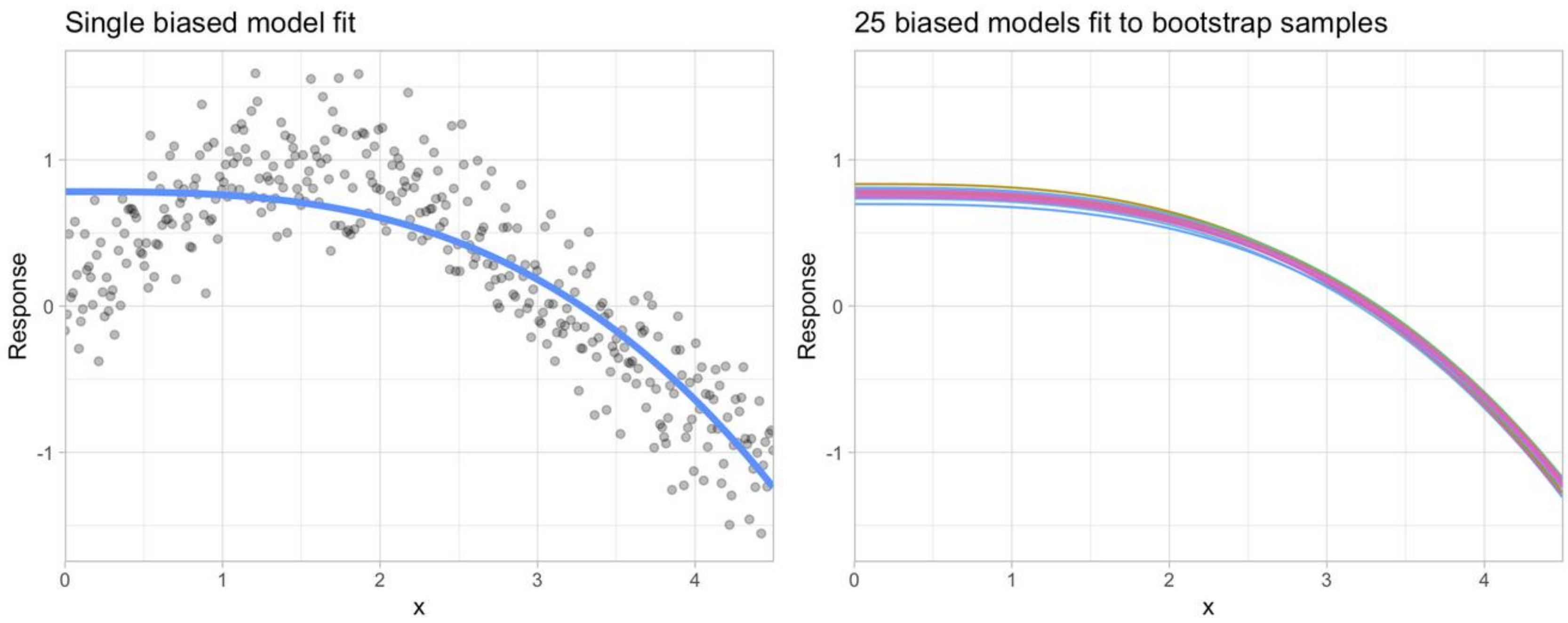
General idea



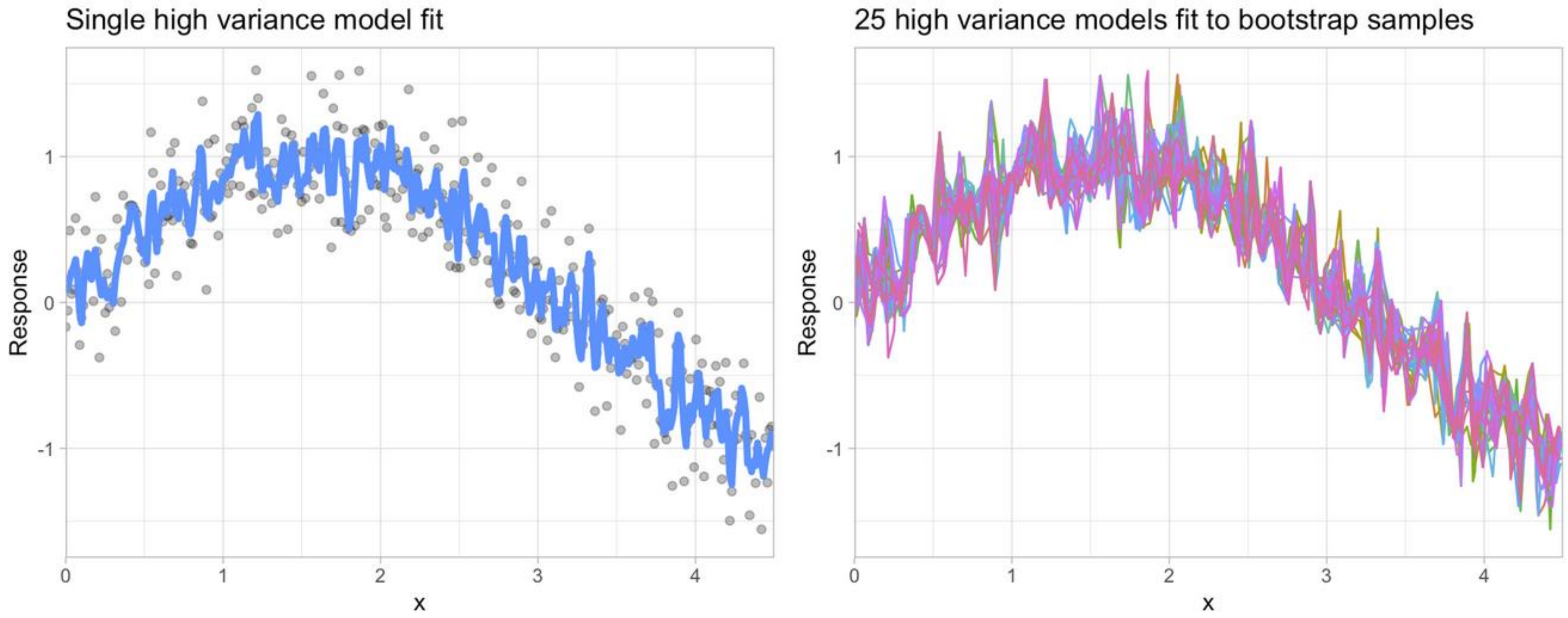
…but really, you're not getting at variance without multiple model fits

https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Bias



Variance

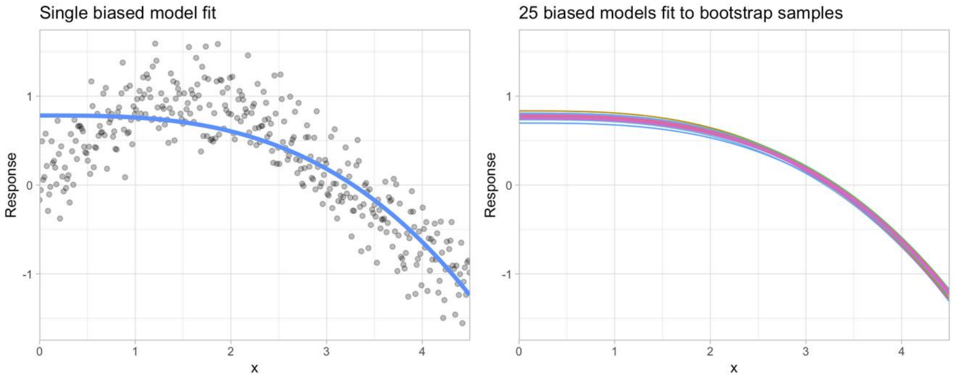


Figures from [HOML]

Properties of selected models

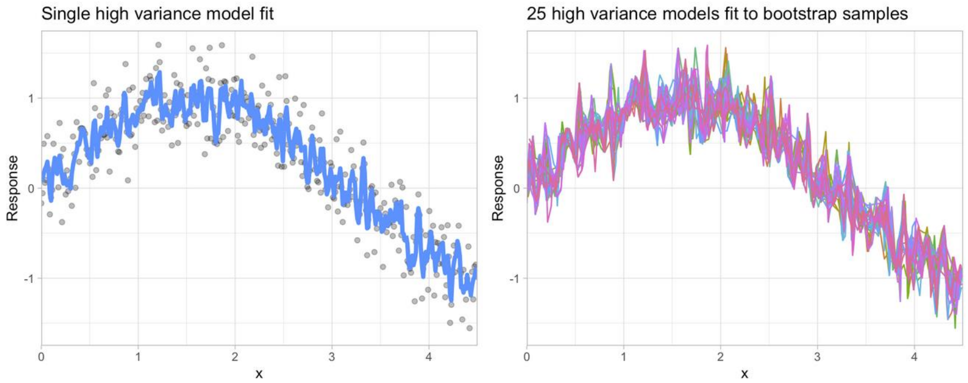
High Bias – Low Variance

- Linear models
    - Linear regression
    - Logistic regression
    - Partial least squares (PLS)



High Variance – Low Bias

- *k*-nearest neighbor
- decision trees
- gradient boosting machines
- neural networks



Figures from [HOML]

Properties of selected models

- Regularization methods can help reduce overfitting
- Helps reduce variance while maintaining bias
- Regularization generally helps controls the model from excessively fluctuating to (over) fit to the data
- Reduces the amount that an individual variable impacts the predictions for a given sample
  - penalties in linear regression
  - dropout in a neural net, trees to randomly drop nodes from the model during fitting
  - sample variables in random forests

Bias Variance Trade-off

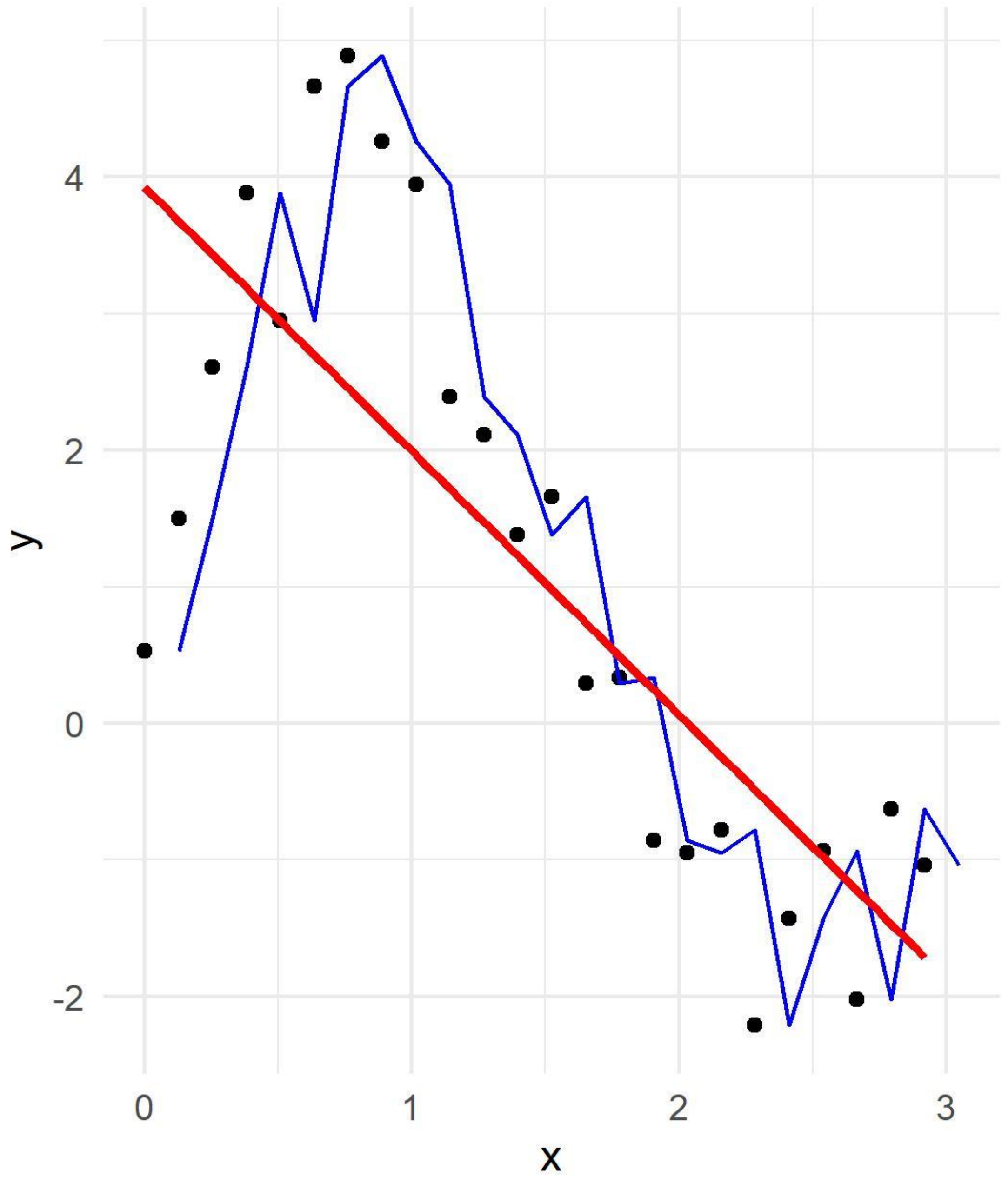
- There is a tradeoff between a model's ability to minimize bias AND variance
- Understanding how different sources of error lead to bias and variance helps us improve the data fitting process resulting in more accurate models
- **Bias** – difference between the (average) prediction of our model and the true value we are trying to predict
- **Variance** – variability of a model prediction for a given data point
  - Implies fitting a model multiple times to different data

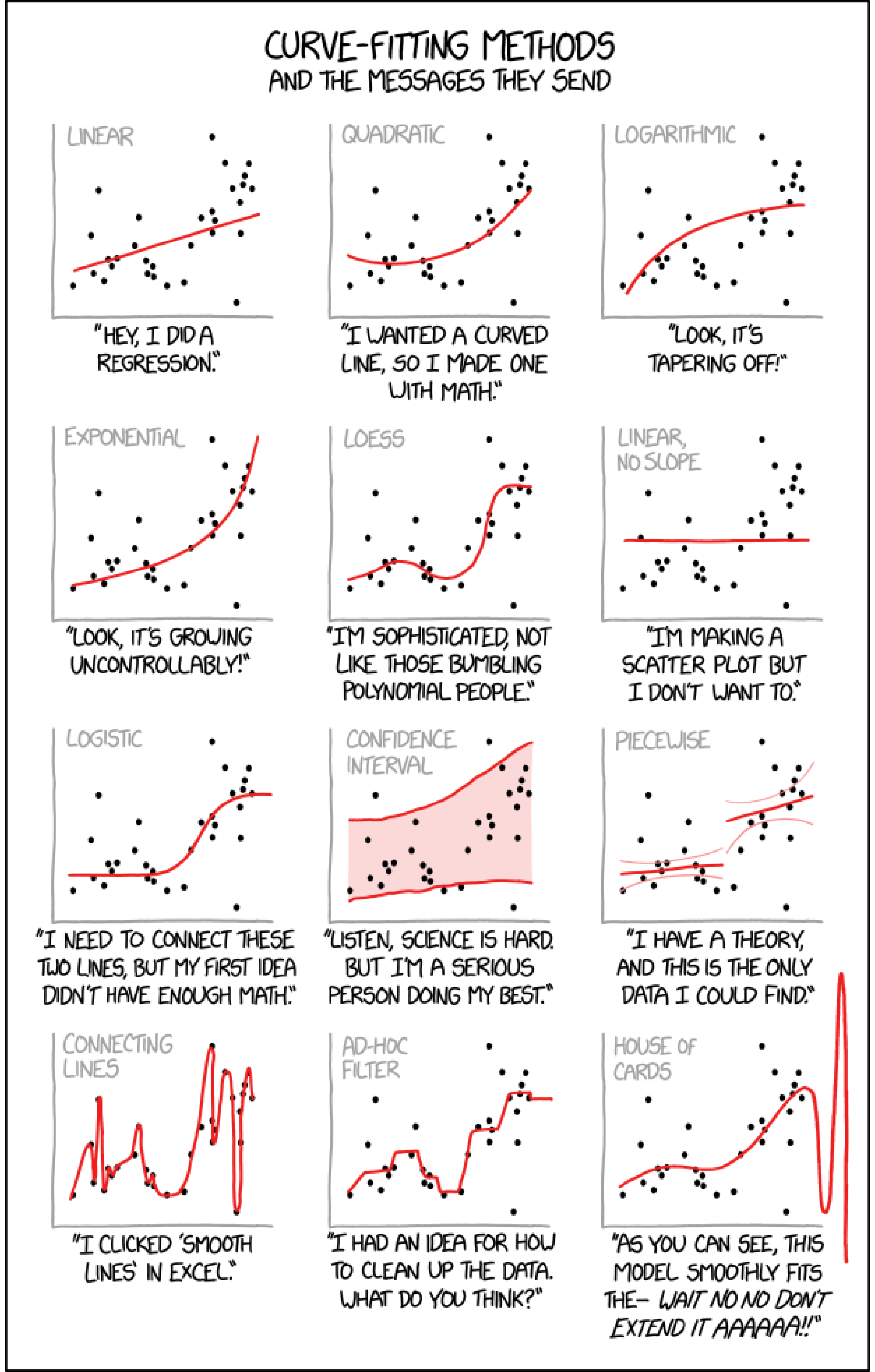
“Simpler” model

- low variance because the model would not change with new data from the same population
- under-fit
- high bias (distance from data)

More “complex” model

- high variance because small changes to the data would change the model fit
- over-fit




CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND
LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE– WAIT NO NO DON'T EXTEND IT AAAAAA!!"

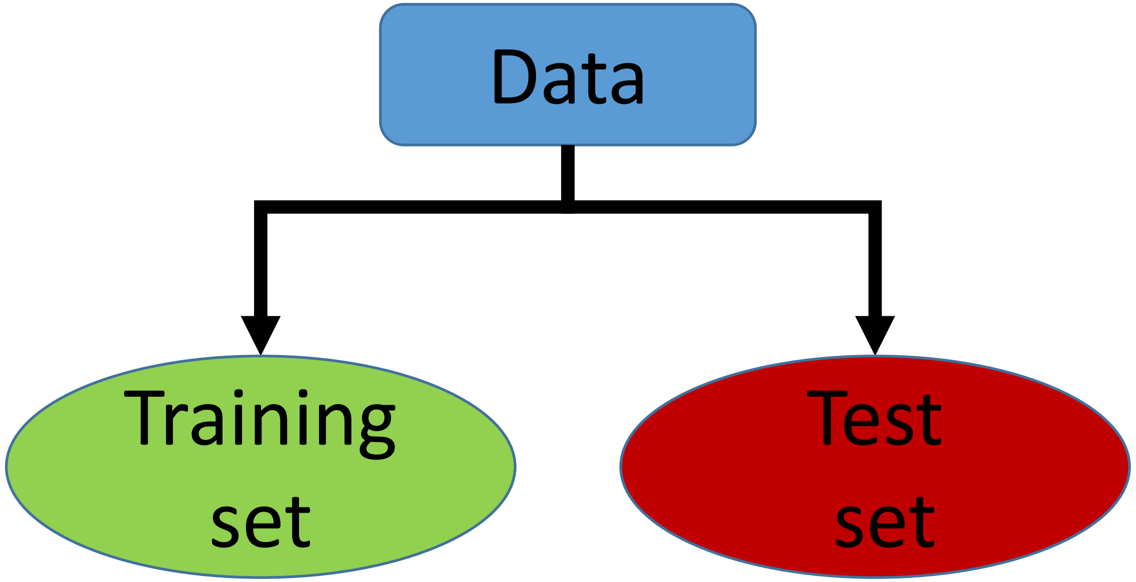
Data Splitting

Data Splitting

- The goal of machine learning is to predict results based on new (unseen) data
- “The best way to measure a model's performance at predicting new data is to predict new data.” – paraphrasing/quoting multiple experts
- The simplest way to do this is to split our data into two parts:
  - **Training set**
  - **Test set**
- We then fit a model to the training data and predict the results of the test set

Data splitting

- We can do anything we want to the training set
  - train our algorithms, tune hyperparameters, compare models, and all of the other activities required to choose a final model
- We do **nothing** with the test set until we have finalized our model using from the training set
  - **Data leakage** is using ANY part of the test set in our training set
    - Using the test set during our modeling process
    - Pre-processing or feature engineering the full data (training and test sets together)
    - Time series design, when the outcome of one series is used in the prediction of the next


Data
Training set
Test set


Data
Training
Test

{rsample}

```
math_split <- initial_split(math)

math_train <- training(math_split)
math_test  <- testing(math_split)
```

- These three functions are meant to be used in conjunction
- A good rule is to make these the first lines of your ML project code
    - some differ

`initial_split()` help documentation

```
initial_split(data, prop = 3/4, strata = NULL, breaks = 4, ...)

initial_time_split(data, prop = 3/4, ...)

training(x)

testing(x)
```

Arguments

data A data frame.

prop The proportion of data to be retained for modeling/analysis.

strata A variable that is used to conduct stratified sampling to create the resamples. This could be a single character value or a variable name that corresponds to a variable that exists in the data frame.

breaks A single number giving the number of bins desired to stratify a numeric stratification variable.

... Not currently used.

x An `rsplit` object produced by `initial_split`

Let's take a look at `initial_split()`

follow along if you can

```
math <- read_csv(here::here("data", "edld-654-spring-2020", "train.csv"))

set.seed(210)
(math_split <- initial_split(math))
<142070/47356/189426>

math_split %>% training() %>% nrow() / nrow(math)
[1] 0.7500026

names(math_split)
[1] "data"    "in_id"   "out_id" "id"

class(math_split)
[1] "rsplit"    "mc_split"
```

Additional arguments

```
initial_split(data, prop = 3/4, strata = NULL, breaks = 4, ...)

initial_time_split(data, prop = 3/4, ...)

training(x)

testing(x)
```

Arguments

data A data frame.

prop The proportion of data to be retained for modeling/analysis.

strata A variable that is used to conduct stratified sampling to create character value or a variable name that corresponds to a varia

breaks A single number giving the number of bins desired to stratify a

... Not currently used.

x An `rsplit` object produced by `initial_split`

The default is simple random assignment, with:

- 75% to the training set, and
- 25% to the test set

A general guideline is somewhere between 60%/40% & 80%/20%.

- Spending too much in training (e.g., > 80%) may mean poor predictive performance. It may fit the training data very well, but is not generalizable (overfitting).
- Spending too much in testing (e.g., > 40%) may mean poor assessment of model parameters (underfitting).

- If you have a lot of data, you may see little predictive benefit of using the entire data, but an increase in computational time.
- If you have more features/predictors than rows, you may need a larger sample size to identify consistent signals in the features.

```
split_data <- initial_split(ames, prop = .70)
```

Let's take a look at `prop`

```
set.seed(210)
(math_split70 <- initial_split(math, prop = .70))
<132599/56827/189426>

math_split70 %>% training() %>% nrow() / nrow(math)
[1] 0.7000042
```

Additional arguments

```
initial_split(data, prop = 3/4, strata = NULL,

initial_time_split(data, prop = 3/4, ...)

training(x)

testing(x)
```

Arguments

data A data frame.

prop The proportion of data to be retaine

strata A variable that is used to conduct st character value or a variable name t

breaks A single number giving the number

... Not currently used.

x An `rsplit` object produced by `initial_split`

As opposed to simple random assignment, you can use **stratified sampling** to ensure the training and test sets have similar outcome (*Y*) distributions/proportions (equal to that of the full data set). Helps ensure a balanced representation of the response distribution in both the training and test sets.

Especially useful if:

- the continuous outcome is not normally distributed (skewed)
  - stratified sampling will segment outcome into quantiles and randomly sample from each
- the categorical outcome has substantial unbalanced classes (e.g., 6% HS dropout, 94% graduate)

```
split_data <- initial_split(ames, strata = Sales_Price)
```

Let's take a look at `strata`

Not a great example because we're stratifying by a predictor and not the outcome but…

```
math_split %>%
  training() %>%
  janitor::tabyl(ethnic_cd)
 ethnic_cd     n    percent
         A  5885 0.04142324
         B  3148 0.02215809
         H 34537 0.24309847
         I  1848 0.01300767
         M  8930 0.06285634
         P  1077 0.00758077
         W 86645 0.60987541
```

```
math_split %>%
  testing() %>%
  janitor::tabyl(ethnic_cd)
 ethnic_cd     n     percent
         A  1810 0.038221134
         B  1002 0.021158882
         H 11345 0.239568376
         I   594 0.012543289
         M  2965 0.062610862
         P   353 0.007454177
         W 29287 0.618443281
```

```
math_split_strat <- initial_split(math, strata = ethnic_cd)
```

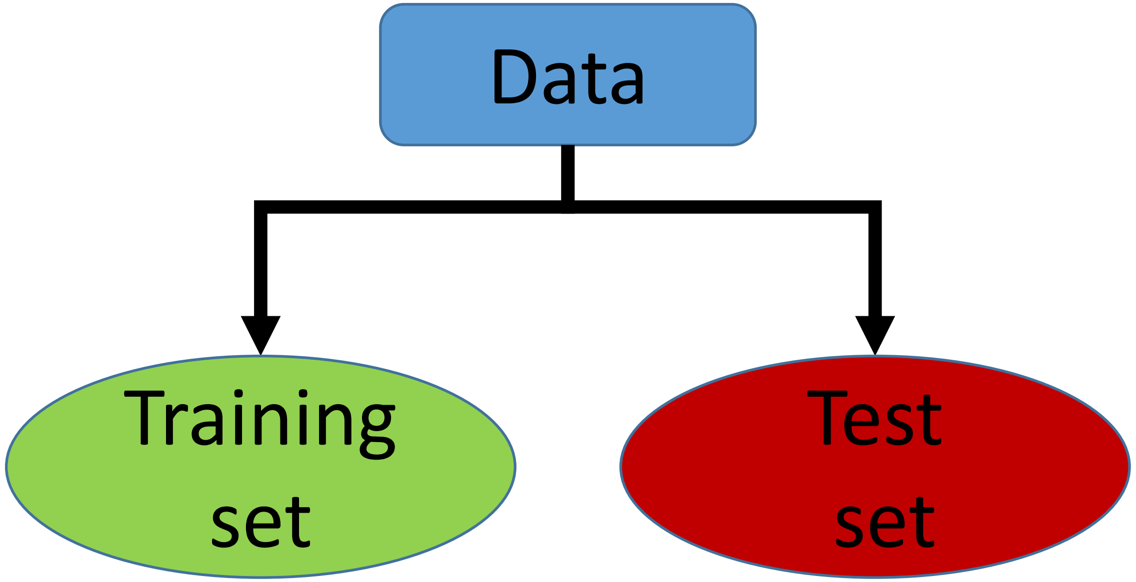
```
math_split_strat %>%
  training() %>%
  janitor::tabyl(ethnic_cd)
 ethnic_cd     n     percent
         A  5718 0.040247765
         B  3114 0.021918772
         H 34465 0.242591680
         I  1841 0.012958401
         M  8910 0.062715563
         P  1067 0.007510382
         W 86955 0.612057436
```

```
math_split_strat %>%
  testing() %>%
  janitor::tabyl(ethnic_cd)
 ethnic_cd     n     percent
         A  1977 0.041747614
         B  1036 0.021876848
         H 11417 0.241088774
         I   601 0.012691106
         M  2985 0.063033195
         P   363 0.007665343
         W 28977 0.611897120
```

Resampling

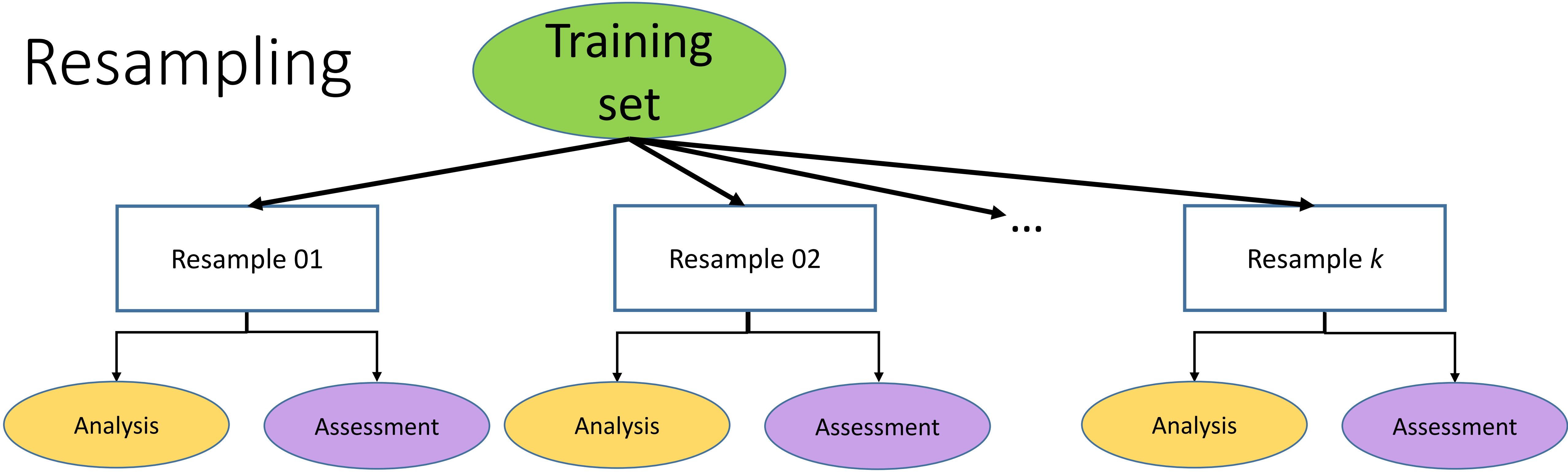
We split – now what?

- Again, we NEVER use the test set until we have a “final model”
- And “the best way to measure a model's performance at predicting new data is to predict new data”
- So how do we measure model performance during the training phase? What new data do we predict?
- Just re-predicting the training set is not ideal
  - biases results - may well predict training set but won’t generalize to new data
  - no measure of variance if we only have one measure of performance (based on predicting the training set)
- We **resample** training set


Data
Training set
Test set

Resampling

Resampling



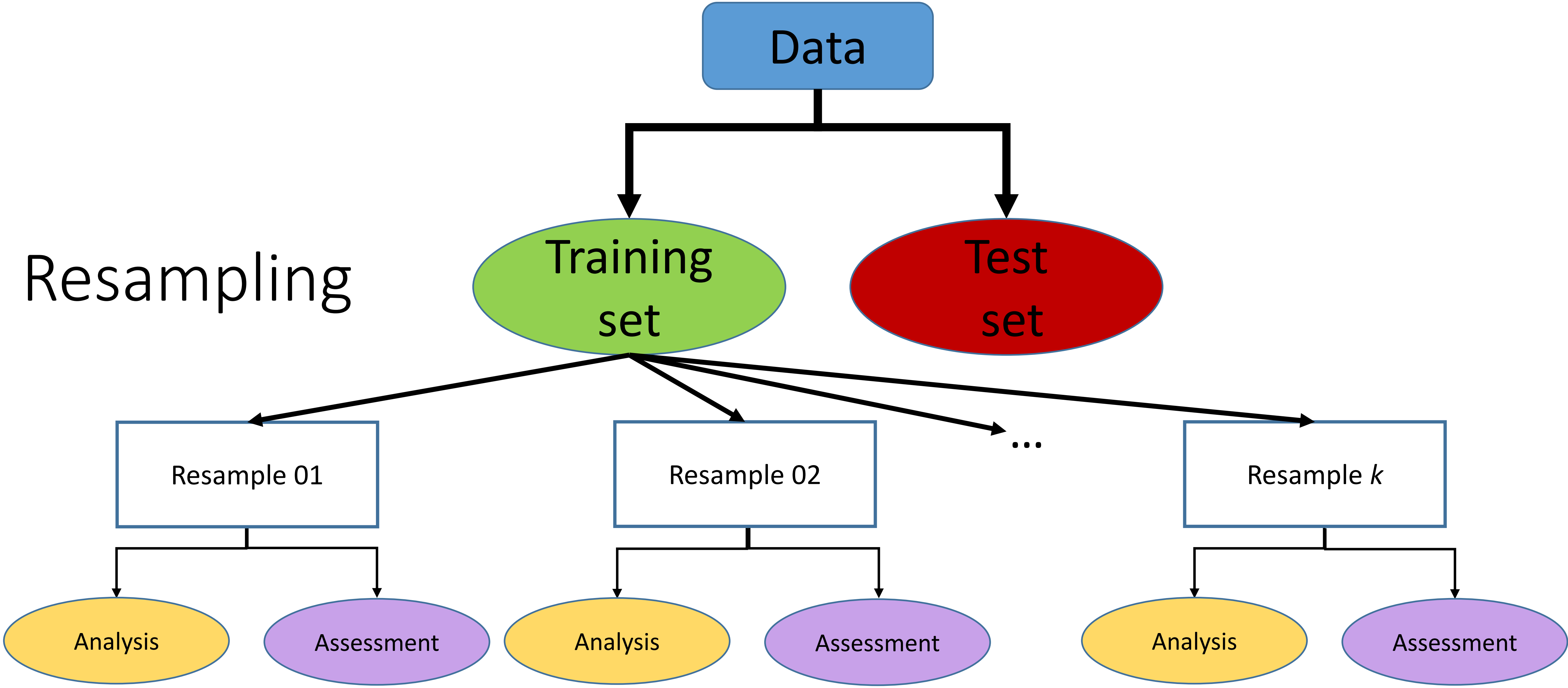
Training:Test::Analysis:Assessment

-OR-

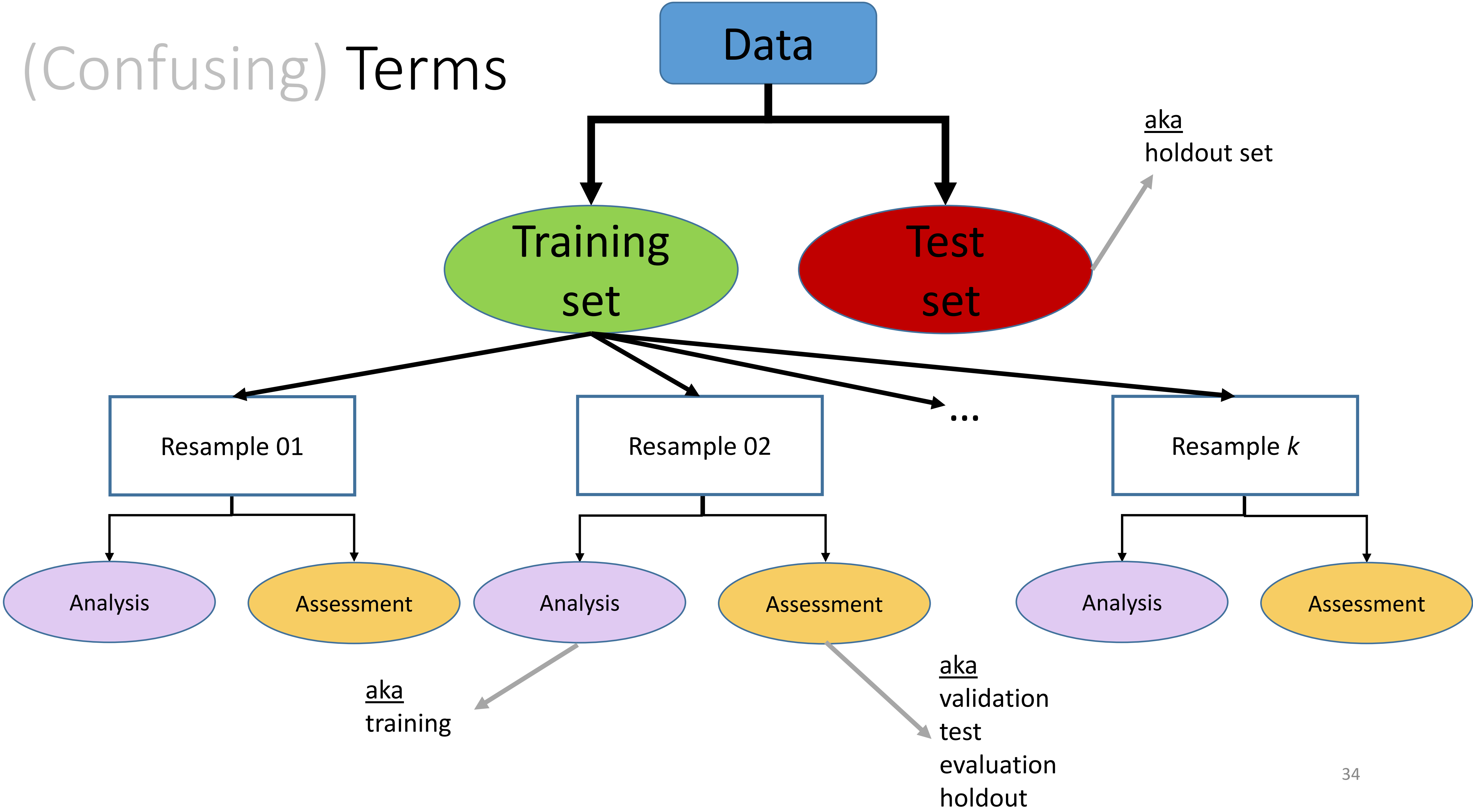
Analysis functions like the Training set

Assessment functions like the Test set

Resampling


Data
Training set
Test set
Resample 01
Resample 02
...
Resample *k*
Analysis
Assessment
Analysis
Assessment
Analysis
Assessment

(Confusing) Terms



Common Resampling Methods

- *k*-fold cross-validation
  - Probably the most common resampling method for model evaluation and model selection in applied ML
- Monte Carlo cross-validation
- Bootstrapping
- Others (most not discussed here)
  - Leave one out cross validation (LOOCV)
  - Rolling origin forecasting – for time series data
  - 632 and 632+ methods
  - Maximum dissimilarity sampling

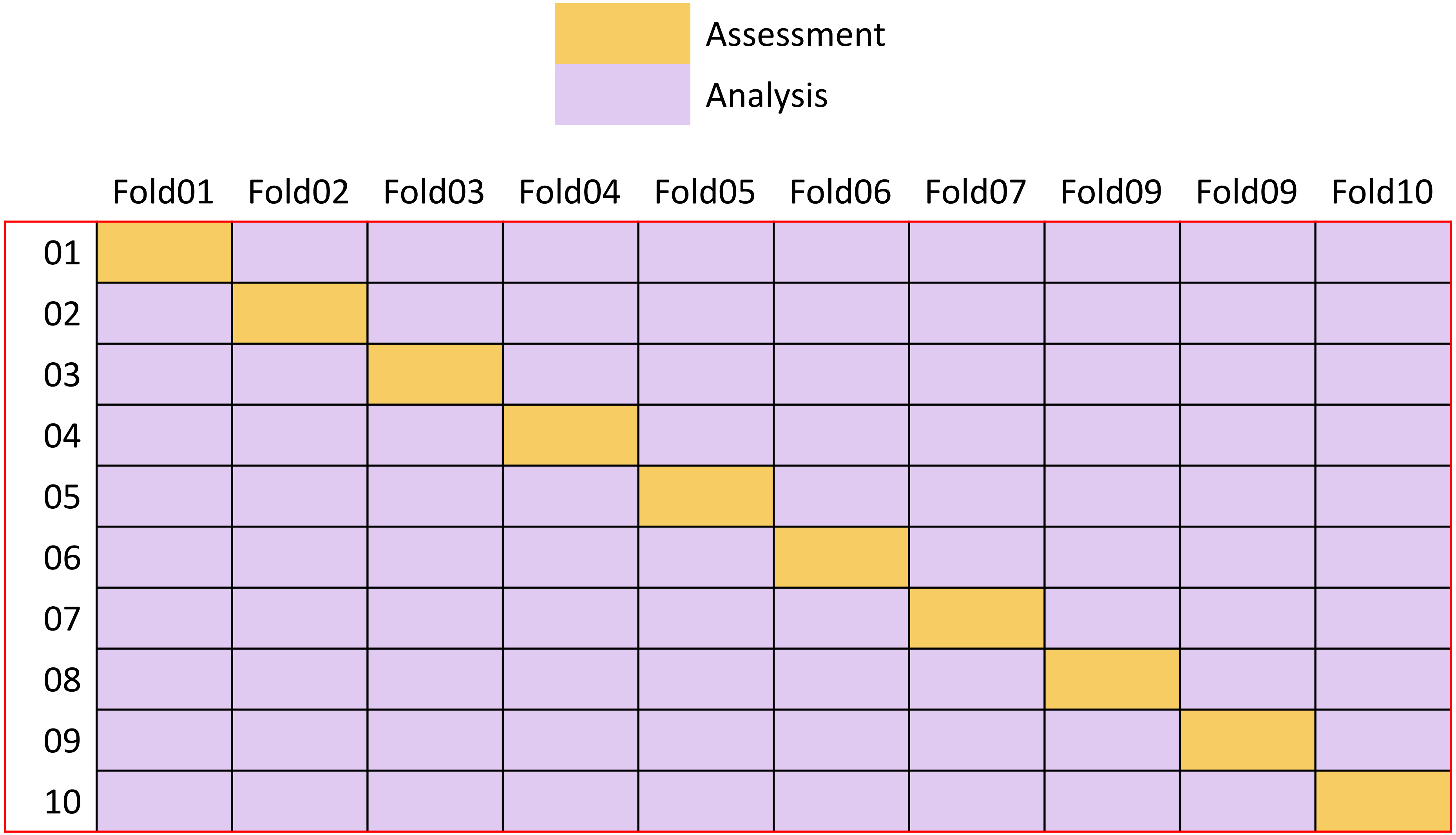
k-fold cross-validation (*k*-fold CV)

- We randomly split the training data into *k* distinct samples ("folds") of (approximately) equal size

10-fold CV

- $k = 10$
- Within each fold, a random 10% of training data are sampled for the assessment set
  - The 10% assessment sample is completely different for each fold
  - Each observation (row) serves in **one and only one assessment sample**
- The remaining 90% of the training data serve as the analysis set in the fold

10-fold CV



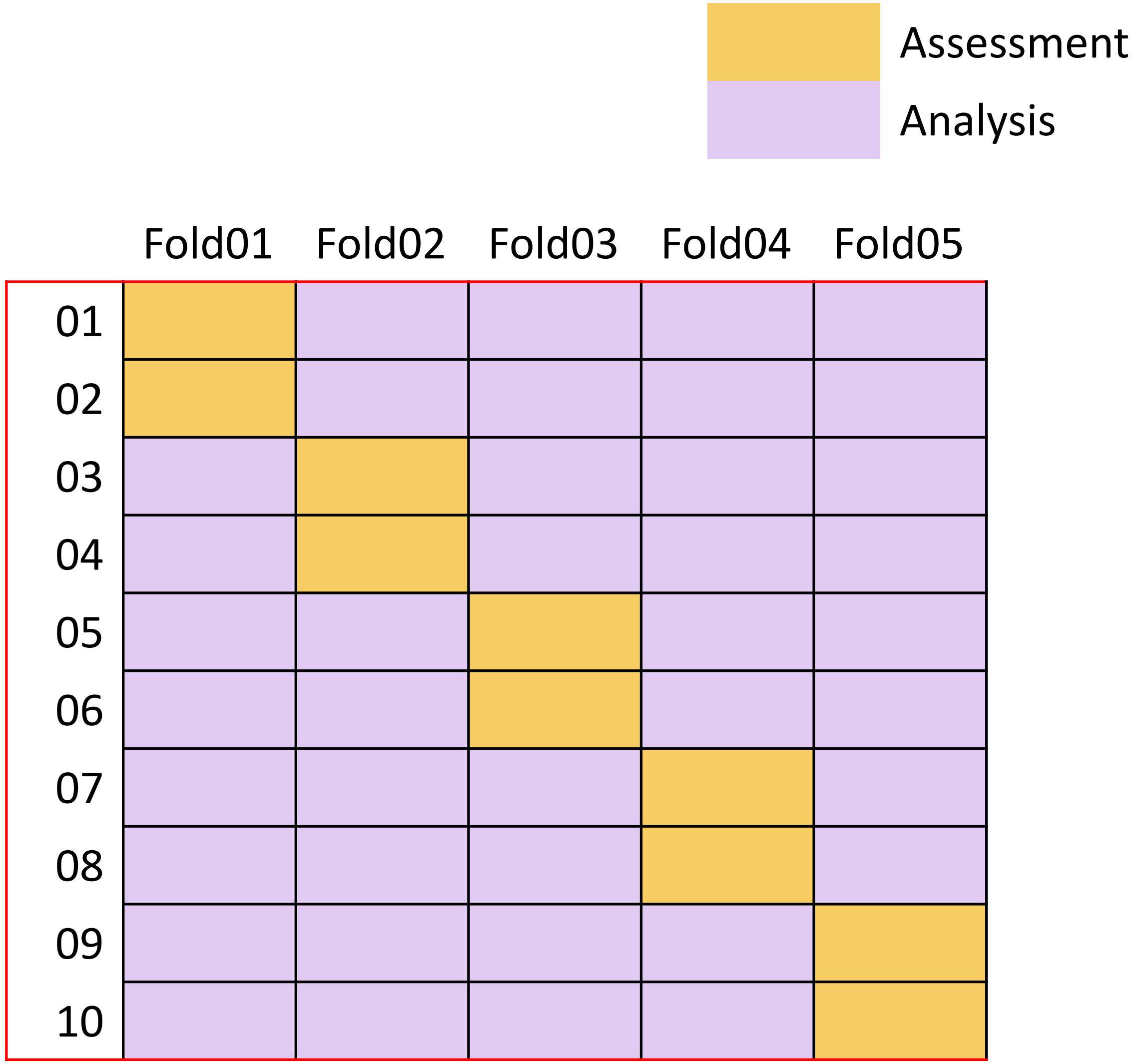
Assessment
Analysis
Fold01
Fold02
Fold03
Fold04
Fold05
Fold06
Fold07
Fold09
Fold09
Fold10
01
02
03
04
05
06
07
08
09
10

k-fold CV

5-fold CV

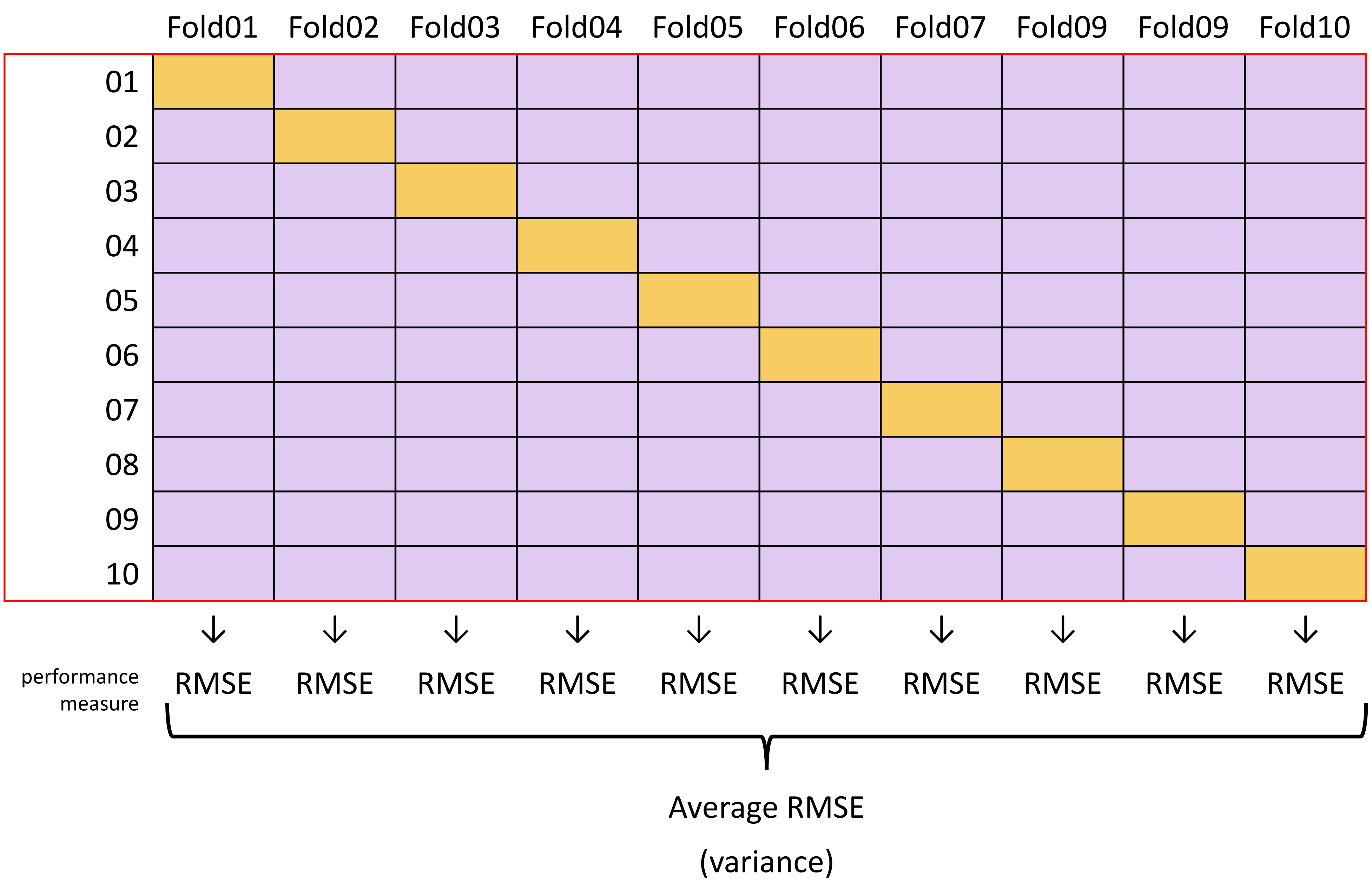
- $k = 5$
- Within each fold, a random 20% of training data are sampled for the assessment set
  - The 20% assessment sample is completely different for each fold
  - Each observation (row) serves in **one and only one assessment sample**
- The remaining 80% of the training data serve as the analysis set in the fold

5-fold CV


Assessment
Analysis
Fold01
Fold02
Fold03
Fold04
Fold05
01
02
03
04
05
06
07
08
09
10

Results

- `Fold01`
  - We fit our model on the `Fold01` analysis set (leaving out the assessment set)
  - We apply our resulting model parameters to predict the assessment set
  - We get our performance measures (objective functions)
- We repeat this process until we've predicted all *k* assessment sets
- The final performance is the *average* performance measure across the *k* folds


Fold01
Fold02
Fold03
Fold04
Fold05
Fold06
Fold07
Fold09
Fold09
Fold10
01
02
03
04
05
06
07
08
09
10
performance measure
RMSE
RMSE
RMSE
RMSE
RMSE
RMSE
RMSE
RMSE
RMSE
RMSE
Average RMSE
(variance)

k-fold CV suggestions

- Larger values of *k*:
  - produce less bias (because the difference between a fold and the training set decreases)
  - more computationally intensive
- 10 folds is a good rule-of-thumb
  - Leave-one-out is the most extreme resampling technique
    - Use *n* - 1 to predict each row
  - 10-fold CV performed comparably to LOOCV (Molarino, 2005)

k-fold CV suggestions

- Has more variability compared to other resampling methods (bootstrapping)
  - **Repeating** *k*-fold CV can improve the accuracy of the estimates while maintaining small bias (Molarino, 2005; Kim, 2009)
  - Helps reduce variability between folds; gives a more complete estimate of the overall between-fold variability (i.e., the variance distribution)
    - 10-fold CV repeated 5 times = 50 models/performance measures
    - Particularly useful for smaller data sets
    - For large training sets, variance and bias issues are less of a concern
  - Repeated CV is not equivalent to increasing the number of folds (e.g., 50-fold CV)

vfold_cv()

```
vfold_cv(data, v = 10, repeats = 1, strata = NULL, breaks = 4, ...)
```

`data` = your training set

`v` = number of folds (default = 10)

`repeats` = number of repeats (default = 1)

`strata` = variable to conduct stratified sampling to create the folds

`breaks` = the number of bins desired to stratify a numeric stratification variable

vfold_cv()

```
set.seed(210)
(cv_splits <- vfold_cv(math_train))
```

```
#  10-fold cross-validation
# A tibble: 10 x 2
   splits                    id
   <named list>              <chr>
 1 <split [127.9K/14.2K]> Fold01
 2 <split [127.9K/14.2K]> Fold02
 3 <split [127.9K/14.2K]> Fold03
 4 <split [127.9K/14.2K]> Fold04
 5 <split [127.9K/14.2K]> Fold05
 6 <split [127.9K/14.2K]> Fold06
 7 <split [127.9K/14.2K]> Fold07
 8 <split [127.9K/14.2K]> Fold08
 9 <split [127.9K/14.2K]> Fold09
10 <split [127.9K/14.2K]> Fold10
```

```
cv_splits$splits[[1]]
<127863/14207/142070>

cv_splits$splits[[1]] %>%
  assessment()

# A tibble: 14,207 x 40
      id gndr  ethnic_cd attnd_dist_inst~ attnd_schl_inst~ enrl_grd calc_admn_cd tst_bnch tst_dt 
migrant_ed_fg
   <dbl> <chr> <chr>                <dbl>            <dbl>    <dbl> <lgl>        <chr>    <chr>  <chr>
 1    36 F     W                     2048              422        8 NA           3B       5/16/~ N
 2    52 F     W                     1944              161        8 NA           3B       5/23/~ N
 3    61 M     H                     1901             1322        8 NA           3B       5/16/~ N
 4    69 M     H                     2183              934        8 NA           3B       5/21/~ N
 5    70 M     H                     2053             1773        7 NA           G7       5/3/2~ N
 6    72 M     W                     2057              480        7 NA           G7       5/15/~ N
 7    80 M     H                     1974              235        7 NA           G7       5/15/~ N
 8   115 M     W                     2041              380        8 NA           3B       5/18/~ N
 9   221 F     W                     2183             1312        7 NA           G7       4/18/~ N
10   230 M     B                     2180              847        8 NA           3B       4/24/~ N
# ... with 14,197 more rows, and 30 more variables: ind_ed_fg <chr>, sp_ed_fg <chr>, tag_ed_fg <chr>,
#   econ_dsvntg <chr>, ayp_lep <chr>, stay_in_dist <chr>, stay_in_schl <chr>, dist_sped <chr>,
#   trgt_assist_fg <chr>, ayp_dist_partic <chr>, ayp_schl_partic <chr>, ayp_dist_prfrm <chr>,
#   ayp_schl_prfrm <chr>, rc_dist_partic <chr>, rc_schl_partic <chr>, rc_dist_prfrm <chr>, rc_schl_prfrm 
<chr>,
#   partic_dist_inst_id <dbl>, partic_schl_inst_id <dbl>, lang_cd <chr>, tst_atmpt_fg <chr>,
#   grp_rpt_dist_partic <chr>, grp_rpt_schl_partic <chr>, grp_rpt_dist_prfrm <chr>, grp_rpt_schl_prfrm 
<chr>,
#   score <dbl>, classification <dbl>, ncessch <dbl>, lat <dbl>, lon <dbl>
```

vfold_cv()

```
cv_splits$splits[[1]]
<127863/14207/142070>
```

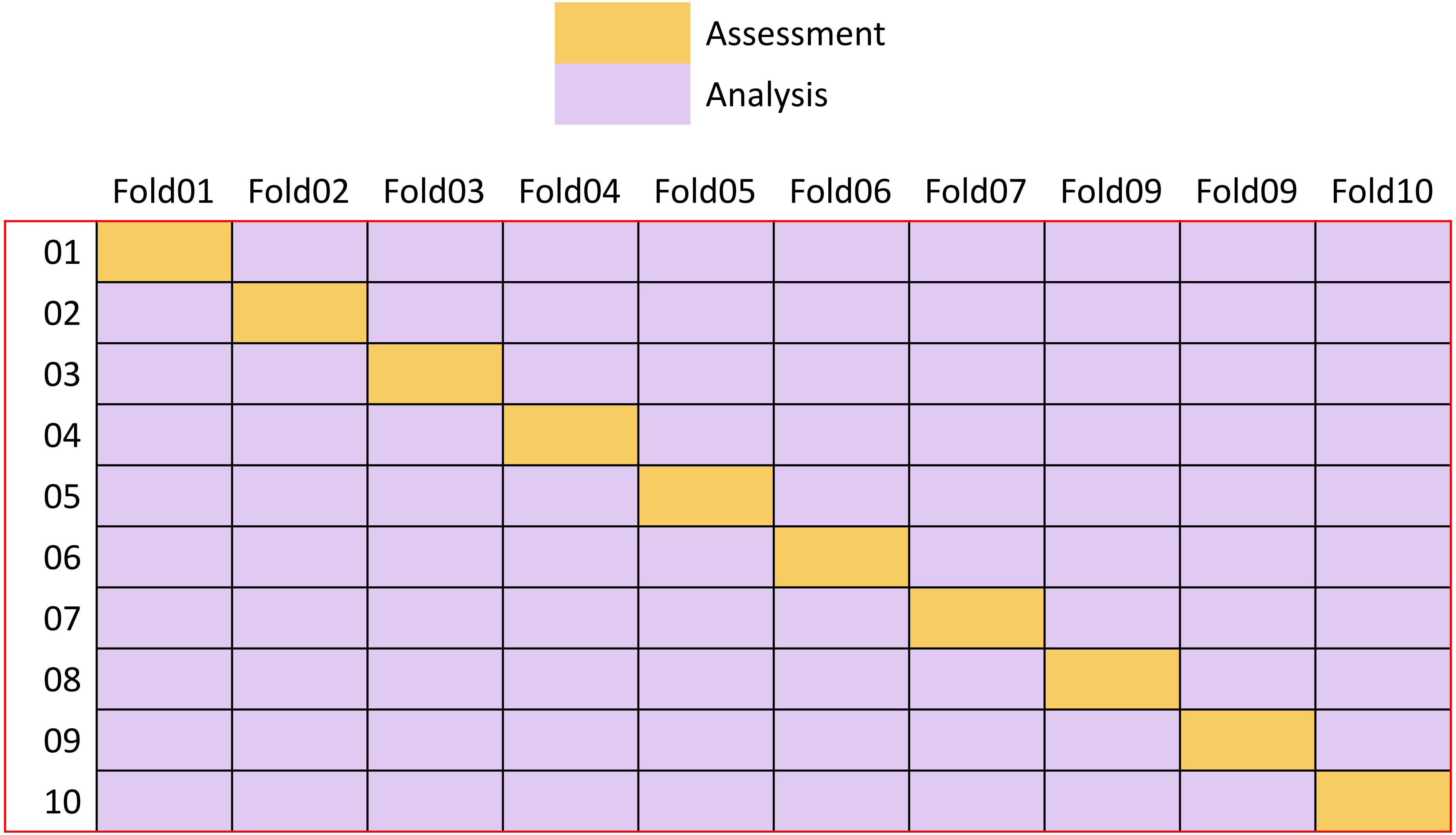
analysis assessment total

```
cv_splits$splits[[1]] %>%
  analysis() %>%
  nrow()
[1] 127863

cv_splits$splits[[1]] %>%
  assessment() %>%
  nrow()
[1] 14207
```

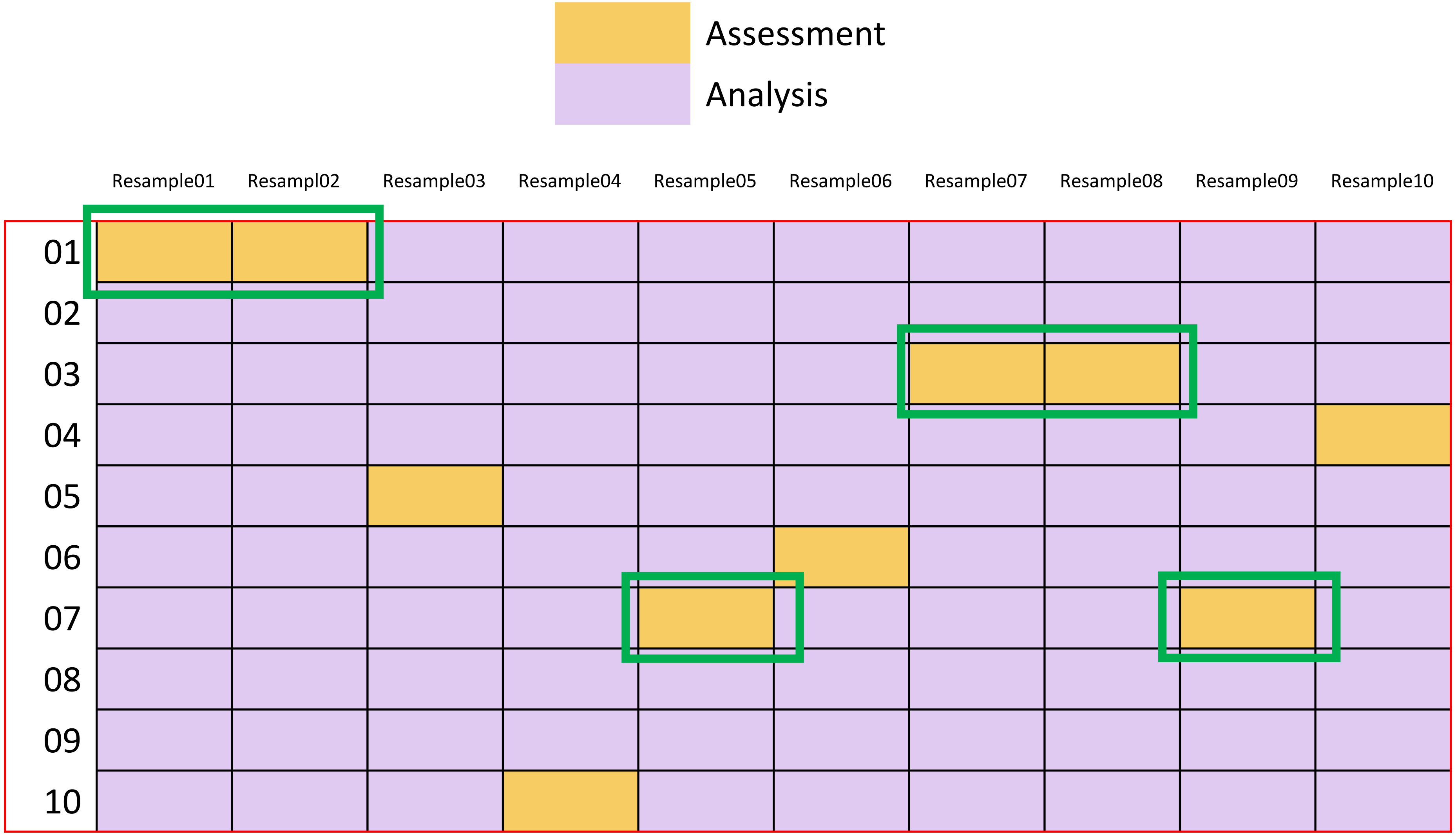
Monte Carlo Cross-Validation

- For each split, a random sample (without replacement) is taken with a specified proportion going into the analysis set and the rest going to the assessment set
- The splitting procedure is conducted a specified number times
  - The number of splits must be large enough have adequate precision
- Like *k*-fold CV, a model is created on the analysis set and the assessment set is used to evaluate the model, and the average of the results across resamples are used to estimate future performance
- As opposed to *k*-fold CV, MC CV produces resamples that are likely to contain overlap

10-fold CV


Assessment
Analysis
Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold09 Fold09 Fold10
01
02
03
04
05
06
07
08
09
10

Monte Carlo CV (10 times)



`mc_cv()`

```
mc_cv(data, prop = 3/4, times = 25, strata = NULL, breaks = 4, ...)
```

`data` = your training set

`prop` = proportion going to the analysis set (default = .75)

`times` = number of times to repeat the sample (default = 25)

`strata` = variable to conduct stratified sampling to create the folds

`breaks` = the number of bins desired to stratify a numeric stratification variable (default = 4)

mc_cv()

```r
(mc_splits <- mc_cv(math_train))
```

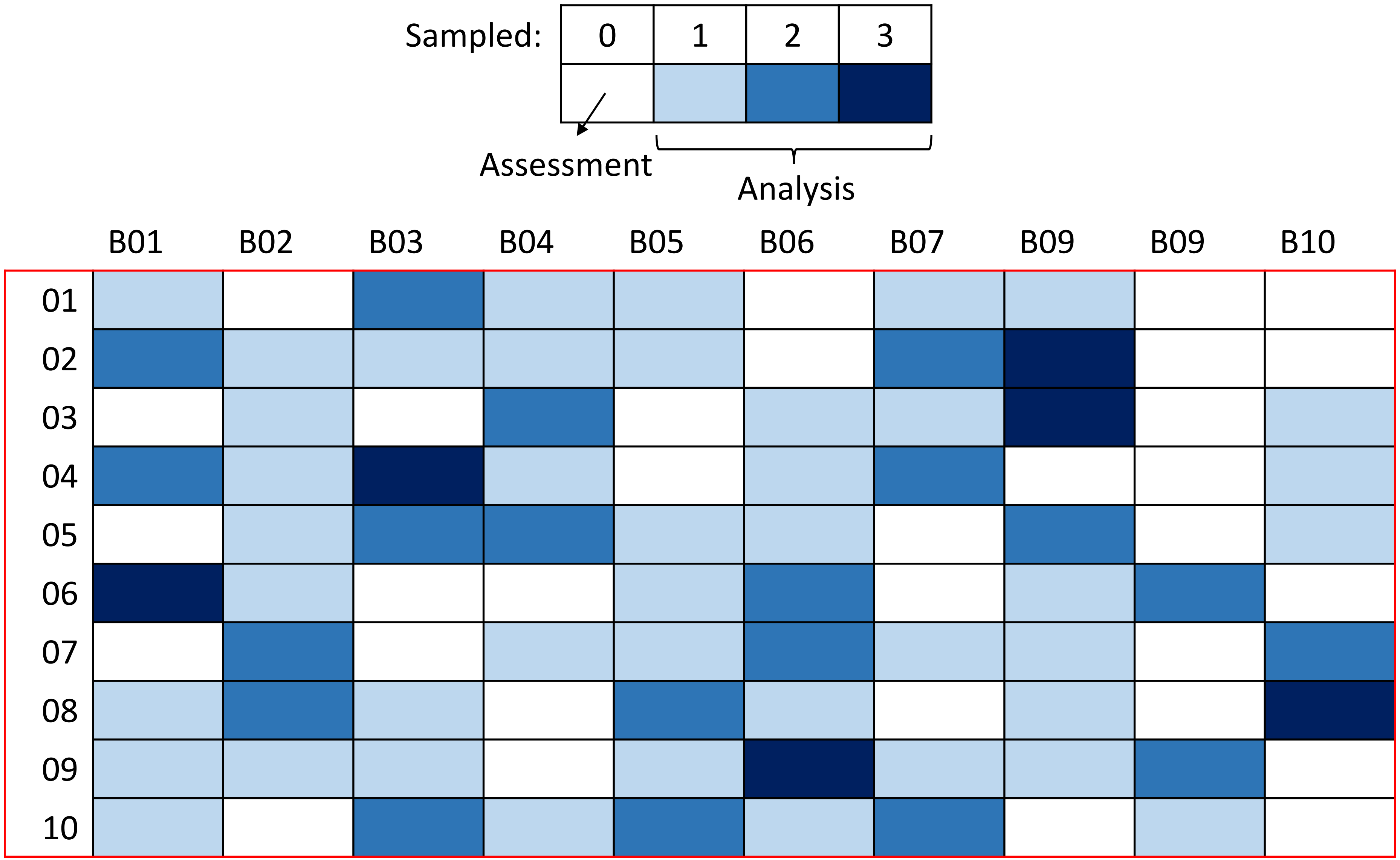
```
# # Monte Carlo cross-validation (0.75/0.25) with 25 resamples
# A tibble: 25 x 2
   splits                      id
   <list>                      <chr>
 1 <split [106.6K/35.5K]> Resample01
 2 <split [106.6K/35.5K]> Resample02
 3 <split [106.6K/35.5K]> Resample03
 4 <split [106.6K/35.5K]> Resample04
 5 <split [106.6K/35.5K]> Resample05
 6 <split [106.6K/35.5K]> Resample06
 7 <split [106.6K/35.5K]> Resample07
 8 <split [106.6K/35.5K]> Resample08
 9 <split [106.6K/35.5K]> Resample09
10 <split [106.6K/35.5K]> Resample10
# ... with 15 more rows
```

mc_cv()

```
nrow(math_train)
[1] 142070

mc_splits$splits[[1]]
<106553/35517/142070>

mc_splits$splits[[12]]
<106553/35517/142070>

mc_splits$splits[[25]]
<106553/35517/142070>

analysis(mc_splits$splits[[1]]) %>% nrow() / nrow(mc_splits$splits[[1]]$data)
[1] 0.7500035
```

bootstrapping

- A bootstrap sample is a simple random sample that is the same size as the training set where the data are sampled with replacement
  - So after a row is selected for inclusion in the subset, it’s still available for further selection
- Each bootstrap sample is likely to contain duplicate values
  - Analysis set
    - On average, 63.21% of the original sample ends up in a bootstrap sample
  - Assessment set
    - Those rows not selected in a bootstrap sample are considered **out-of-bag** (OOB)


Sampled:
0
1
2
3
Assessment
Analysis
B01
B02
B03
B04
B05
B06
B07
B09
B09
B10
01
02
03
04
05
06
07
08
09
10

Results

- `B01`
  - We fit our model on the `B01` analysis set (leaving out the assessment set)
  - We apply our resulting model parameters to predict the assessment set
  - We get our performance measures (loss functions)
- We repeat this process until we've predicted all *B* assessment sets
- The final performance is the *average* performance measure across the *B* sets

Bootstrap notes

- Bootstrap tends to have less variability in the error measure compared to *k*-fold CV
- But because of replacement, bootstrap has more bias (similar to $k = 2$)
  - This is problematic when the training set is small, and less so as the sample increases ($n \geq 1{,}000$)

bootstraps()

```
bootstraps(data, times = 25, strata = NULL, breaks = 4,
  apparent = FALSE, ...)
```

`data` = your training set

`times` = number of bootstrap samples (default = 25)

`strata` = variable to conduct stratified sampling to create the folds

`breaks` = the number of bins desired to stratify a numeric stratification variable

`apparent` = enables the option of an additional resample where the analysis and assessment data sets are the same as the original data set. This can be required for some types of analysis of the bootstrap results.

bootstraps()

> (boot_splits <- bootstraps(math_train))

```
# Bootstrap sampling
# A tibble: 25 x 2
   splits                     id
   <list>                     <chr>
 1 <split [142.1K/52.1K]> Bootstrap01
 2 <split [142.1K/52.2K]> Bootstrap02
 3 <split [142.1K/52.2K]> Bootstrap03
 4 <split [142.1K/52.4K]> Bootstrap04
 5 <split [142.1K/52.3K]> Bootstrap05
 6 <split [142.1K/52.2K]> Bootstrap06
 7 <split [142.1K/52.2K]> Bootstrap07
 8 <split [142.1K/52.5K]> Bootstrap08
 9 <split [142.1K/52.3K]> Bootstrap09
10 <split [142.1K/52.4K]> Bootstrap10
# ... with 15 more rows
```

bootstraps()

```
nrow(math_train)
[1] 142070
```

```
boot_splits$splits[[1]]
<142070/52088/142070>
```

```
boot_splits$splits[[12]]
<142070/52447/142070>
```

```
boot_splits$splits[[25]]
<142070/52149/142070>
```

Leave-one-out (LOO) cross-validation

- Uses one data point in the original set as the assessment data and all other data points as the analysis set
- A LOO resampling set has as many resamples as rows in the original data set

loo_cv()

```
loo_cv(data, ...)
```

```
> (loo_splits <- loo_cv(sample_n(math_train, 10000)))
```

```
# Leave-one-out cross-validation
# A tibble: 10,000 x 2
   splits              id
   <named list>        <chr>
 1 <split [10K/1]> Resample1
 2 <split [10K/1]> Resample2
 3 <split [10K/1]> Resample3
 4 <split [10K/1]> Resample4
 5 <split [10K/1]> Resample5
 6 <split [10K/1]> Resample6
 7 <split [10K/1]> Resample7
 8 <split [10K/1]> Resample8
 9 <split [10K/1]> Resample9
10 <split [10K/1]> Resample10
# ... with 9,990 more rows
```

```
> loo_splits$splits[[1]]
<9999/1/10000>

> loo_splits$splits[[12]]
<9999/1/10000>

> loo_splits$splits[[101]]
<9999/1/10000>
```

Quick summary

- High variance models are more prone to overfitting, and resampling is critical to reduce this risk
- Many models that are capable of achieving good generalization performance have lots of *hyperparameters* that control the level of model complexity (i.e., the tradeoff between bias and variance)
- We'll be talking more about this in the coming weeks

Next time

- Lab 1
- Readings

Lab 1