Lecture 003

Resampling

Edward Rubin

21 January 2020

Admin

Admin

Class today

Review

- Regression and loss
- Classification
- KNN
- The bias-variance tradeoff

Resampling methods

- Cross validation 🚸
- The bootstrap 👢

Admin

Upcoming

Readings

Today: *ISL* Ch. 5 (changed—sorry)

Next: *ISL* Ch. 3–4

Problem set due today (Tuesday) before midnight (PST)

Review

Review

Regression and loss

For **regression settings**, the loss is our prediction's distance from truth, *i.e.*,

$$\text{error}_i = y_i - \hat{y}_i \qquad \text{loss}_i = \left| y_i - \hat{y}_i \right| = \left| \text{error}_i \right|$$

Depending upon our ultimate goal, we choose **loss/objective functions**.

$$\text{L1 loss} = \sum_i \left| y_i - \hat{y}_i \right| \qquad \text{MAE} = \frac{1}{n} \sum_i \left| y_i - \hat{y}_i \right|$$

$$\text{L2 loss} = \sum_i \left(y_i - \hat{y}_i \right)^2 \qquad \text{MSE} = \frac{1}{n} \sum_i \left(y_i - \hat{y}_i \right)^2$$

Whatever we're using, we care about **test performance** (*e.g.*, test MSE), rather than training performance.

Classification

For **classification problems**, we often use the **test error rate**.

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i \neq \hat{y}_i)$$

The **Bayes classifier**

1. predicts class j when $\mathbf{Pr}\big(y_0 = j|\mathbf{X} = \mathbf{x}_0\big)$ exceeds all other classes.
2. produces the **Bayes decision boundary**—the decision boundary with the lowest test error rate.
3. is unknown: we must predict $\mathbf{Pr}\big(y_0 = j|\mathbf{X} = \mathbf{x}_0\big)$.

KNN

K-nearest neighbors (KNN) is a non-parametric method for estimating

$$\Pr\left(y_0 = j | \mathbf{X} = \mathbf{x}_0\right)$$

that makes a prediction using the most-common class among an observation's "nearest" K neighbors.

- **Low values of K** (*e.g.*, 1) are exteremly flexible but tend to overfit (increase variance).
- **Large values of K** (*e.g.*, N) are very inflexible—essentially making the same prediction for each observation.

The *optimal* value of K will trade off between overfitting and accuracy.

The bias-variance tradeoff

Finding the optimal level of flexibility highlights the **bias**-**variance** **tradeoff**.

Bias The error that comes from inaccurately estimating f.

- More flexible models are better equipped to recover complex relationships (f), reducing bias. (Real life is seldom linear.)
- Simpler (less flexible) models typically increase bias.

Variance The amount $\hat{f}$ would change with a different **training sample**

- If new **training sets** drastically change $\hat{f}$, then we have a lot of uncertainty about f (and, in general, $\hat{f} \not\approx f$).
- More flexible models generally add variance to f.

Review

The bias-variance tradeoff

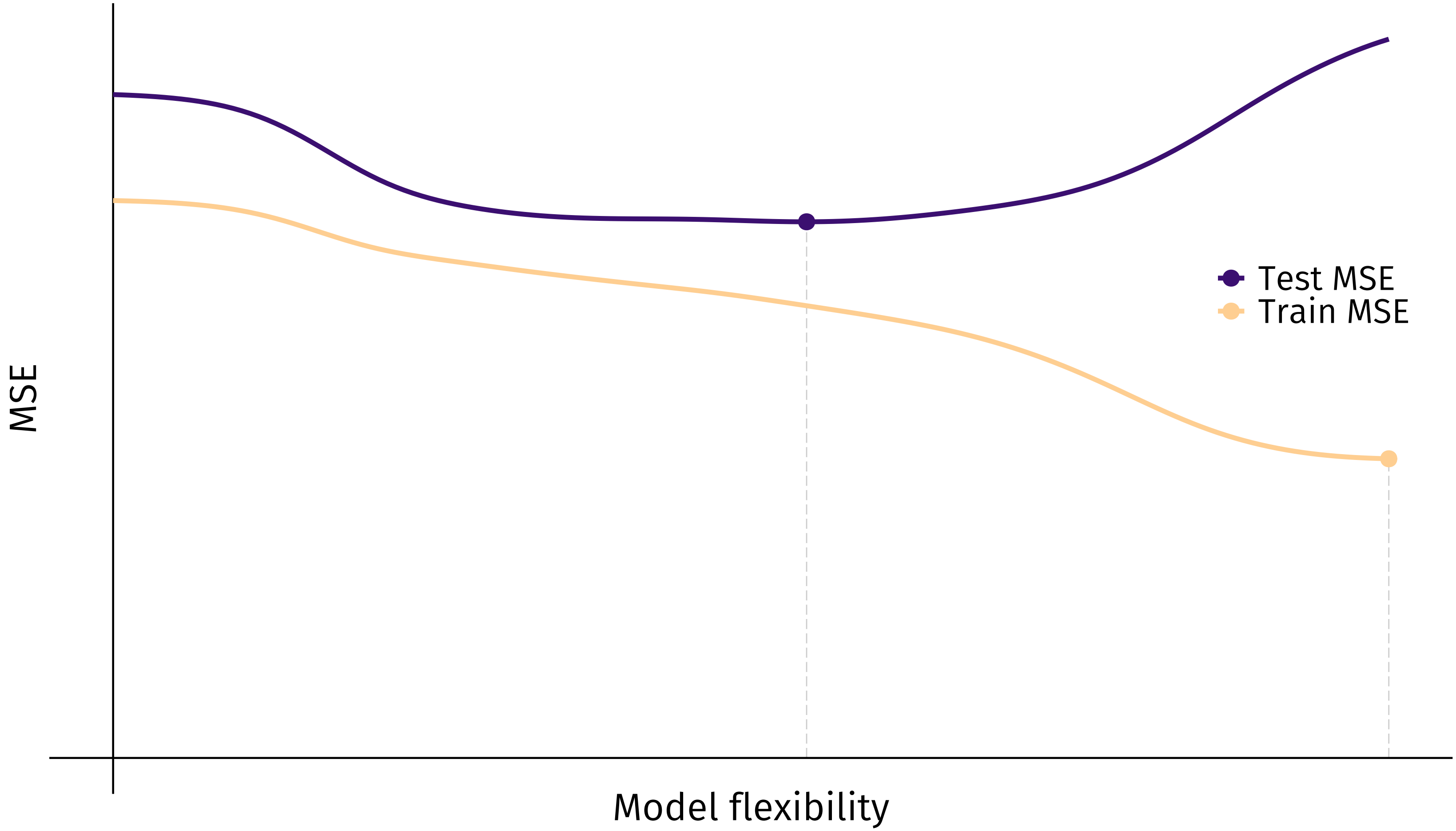
The expected value[†] of the **test MSE** can be written

$$E\left[\left(\mathbf{y_0} - \hat{f}\left(\mathbf{X}_0\right)\right)^2\right] = \underbrace{\mathrm{Var}\left(\hat{f}\left(\mathbf{X}_0\right)\right)}_{\text{Variance}} + \underbrace{\left[\mathrm{Bias}\left(\hat{f}\left(\mathbf{X}_0\right)\right)\right]^2}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irr. error}}$$

The tradeoff in terms of model flexibility

- Increasing flexibility *from total inflexibility* generally **reduces bias more** than it increases variance (reducing test MSE).
- At some point, the marginal benefits of flexibility **equal** marginal costs.
- Past this point (optimal flexibility), we **increase variance more** than we reduce bias (increasing test MSE).

U-shaped test MSE with respect to model flexibility (KNN here). Increases in variance eventually overcome reductions in (squared) bias.



Resampling methods

Resampling methods

Intro

Resampling methods help understand uncertainty in statistical modeling.

- *Ex.* *Linear regression:* How precise is your $\hat{\beta}_1$?
- *Ex.* *With KNN:* Which K minimizes (out-of-sample) test MSE?

The process behind the magic of resampling methods:

1. **Repeatedly draw samples** from the **training data**.
2. **Fit your model**(s) on each random sample.
3. **Compare** model performance (or estimates) **across samples**.
4. Infer the **variability/uncertainty in your model** from (3).

Warning$_1$ Resampling methods can be computationally intensive.
Warning$_2$ Certain methods don't work in certain settings.

Resampling methods

Today

Let's distinguish between two important **modeling tasks:**

- **Model selection** Choosing and tuning a model
- **Model assessment** Evaluating a model's accuracy

We're going to focus on two common **resampling methods:**

1. **Cross validation** used to estimate test error, evaluating performance or selecting a model's flexibility
2. **Bootstrap** used to assess accuracy—parameter estimates or methods

Hold out

Recall: We want to find the model that **minimizes out-of-sample test error**.

If we have a large test dataset, we can use it (once).

Q_1 What if we don't have a test set?

Q_2 What if we need to select and train a model?

Q_3 How can we avoid overfitting our training[†] data during model selection?

$A_{1,2,3}$ **Hold-out methods** (*e.g.*, cross validation) use training data to estimate test performance—**holding out** a mini "test" sample of the training data that we use to estimate the test error.

† Also relevant for *testing* data.

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

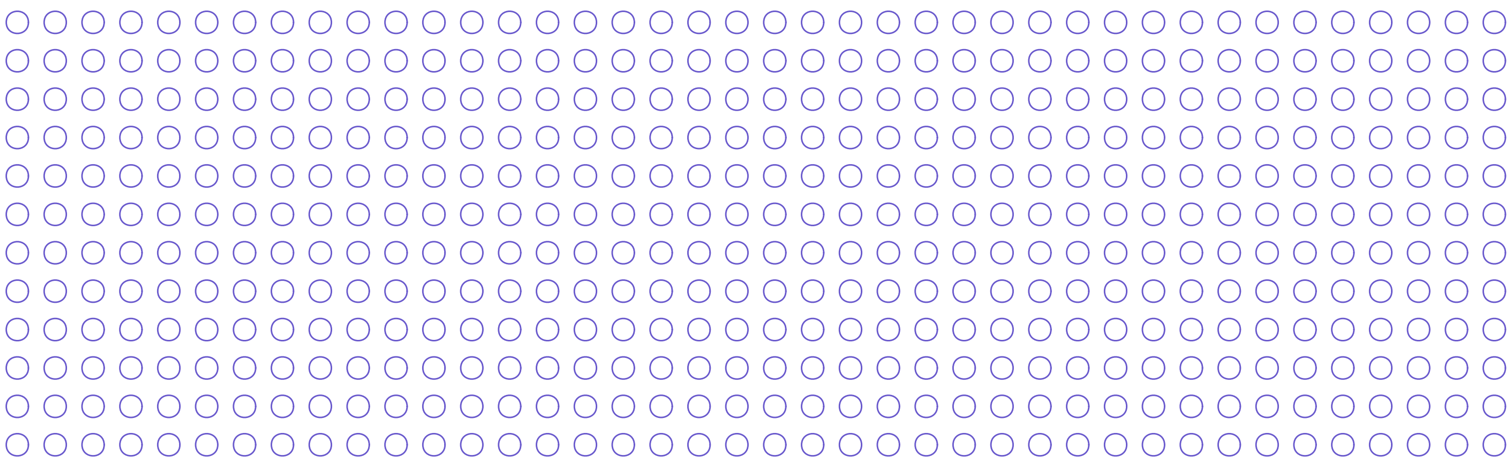
- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.



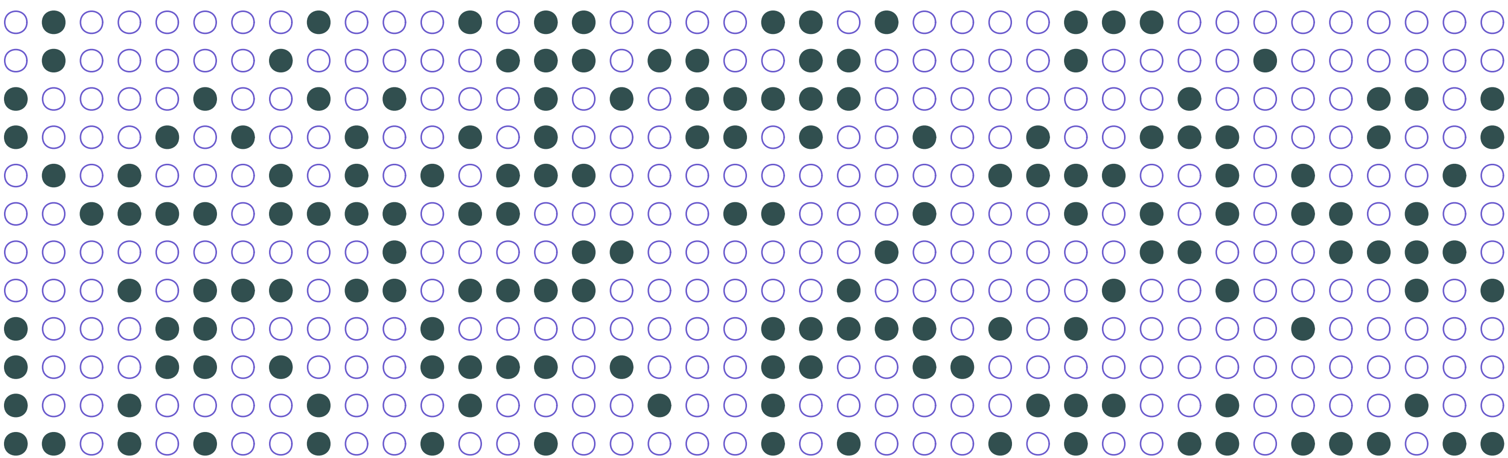
Initial training set

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.



Validation (sub)set **Training set:** Model training

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.

Hold-out methods

Option 1: The *validation set* approach

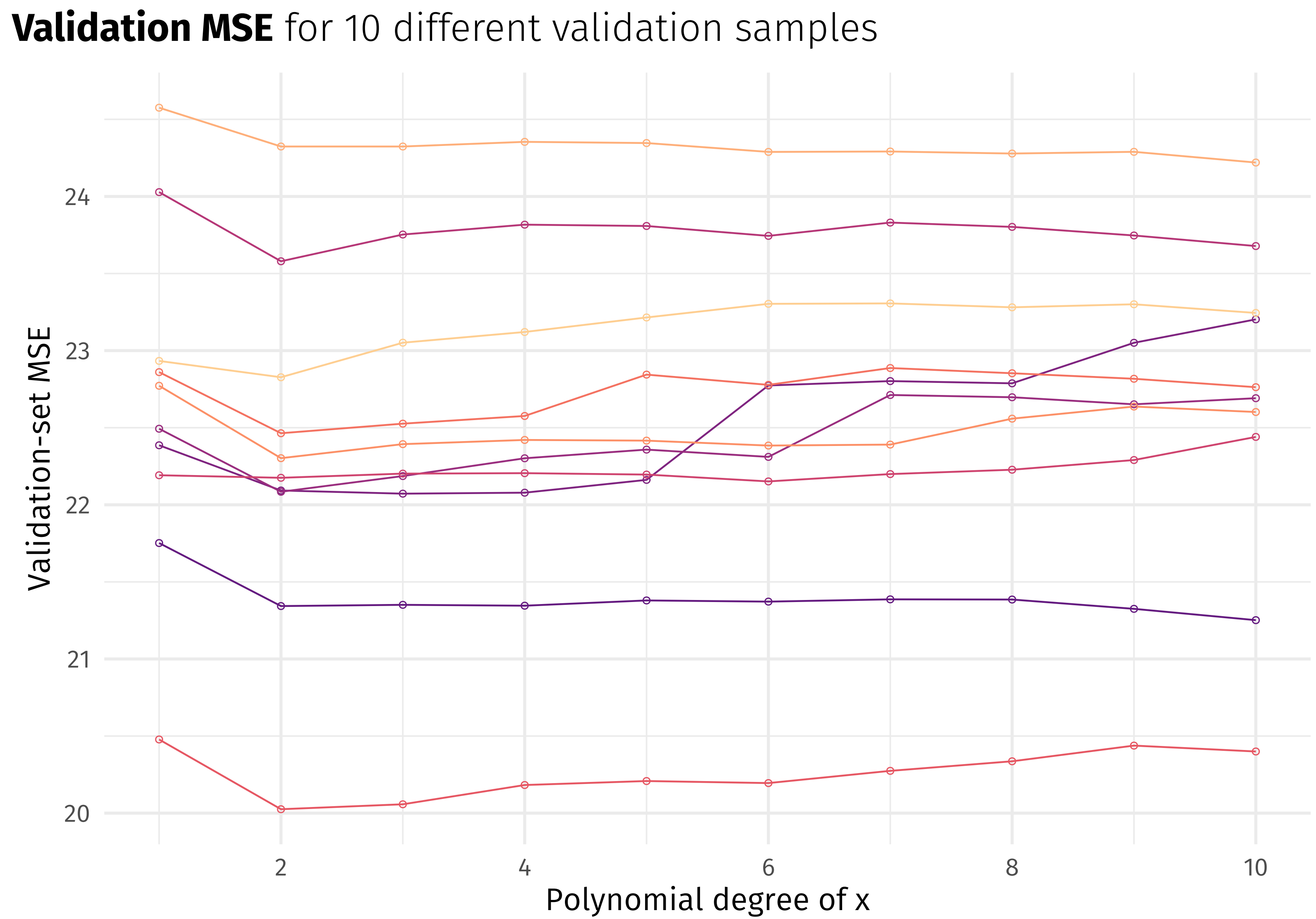
Example We could use the validation-set approach to help select the degree of a polynomial for a linear-regression model (Kaggle).

The goal of the validation set is to ***estimate* out-of-sample (test) error.**

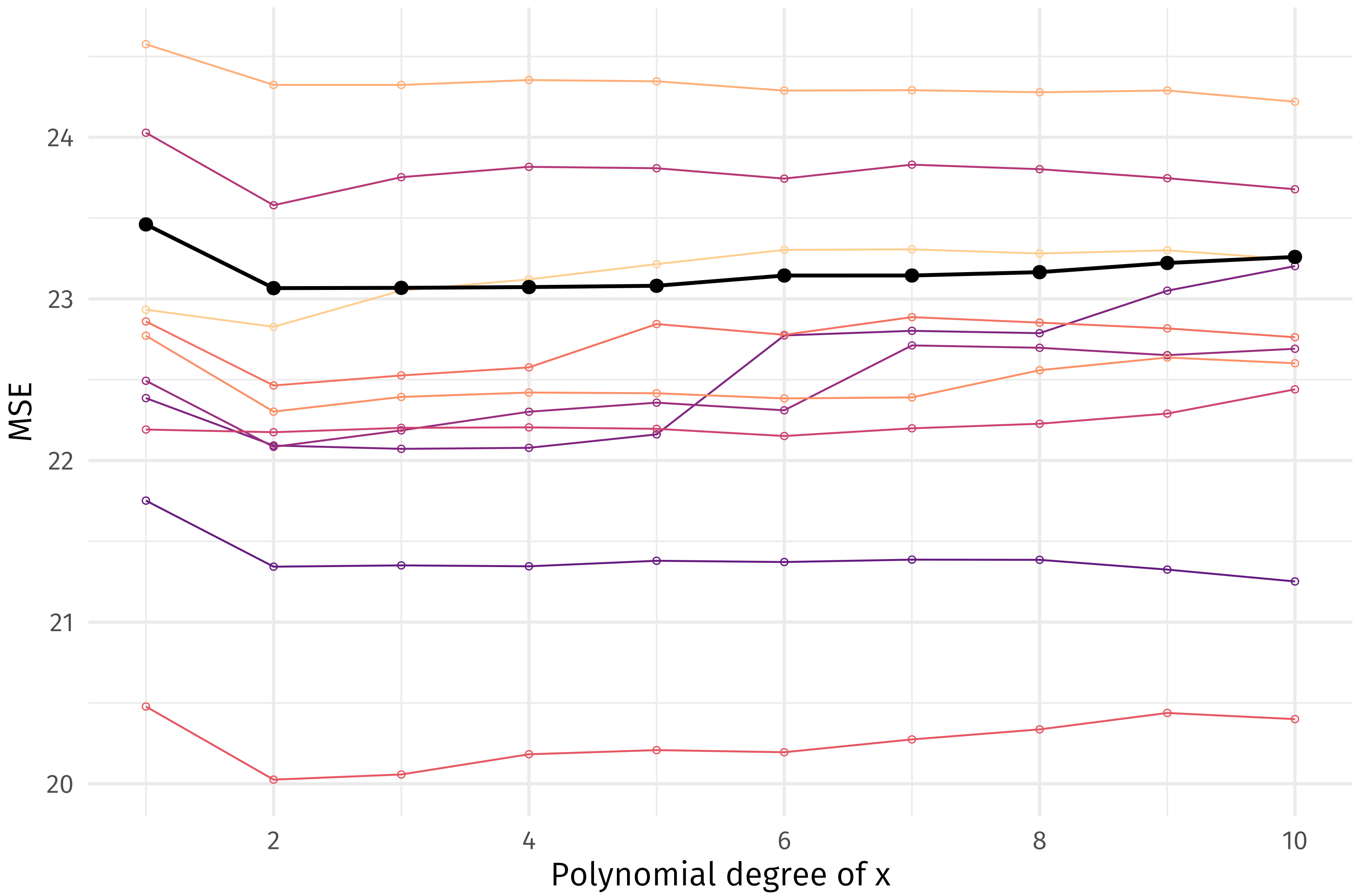
Q So what?

- Estimates come with **uncertainty**—varying from sample to sample.
- Variability (standard errors) is larger with **smaller samples**.

Problem This estimated error is often based upon a fairly small sample (<30% of our training data). So its variance can be large.


Validation MSE for 10 different validation samples
Validation-set MSE
20
21
22
23
24
2
4
6
8
10
Polynomial degree of x

True test MSE compared to validation-set estimates



Hold-out methods

Option 1: The *validation set* approach

Put differently: The validation-set approach has ($\geq$) two major drawbacks:

1. **High variability** Which observations are included in the validation set can greatly affect the validation MSE.

2. **Inefficiency in training our model** We're essentially throwing away the validation data when training the model—"wasting" observations.

(2) $\Longrightarrow$ validation MSE may overestimate test MSE.

Even if the validation-set approach provides an unbiased estimator for test error, it is likely a pretty noisy estimator.

Hold-out methods

Option 2: Leave-one-out cross validation

Cross validation solves the validation-set method's main problems.

- Use more (= all) of the data for training (lower variability; less bias).
- Still maintains separation between training and validation subsets.

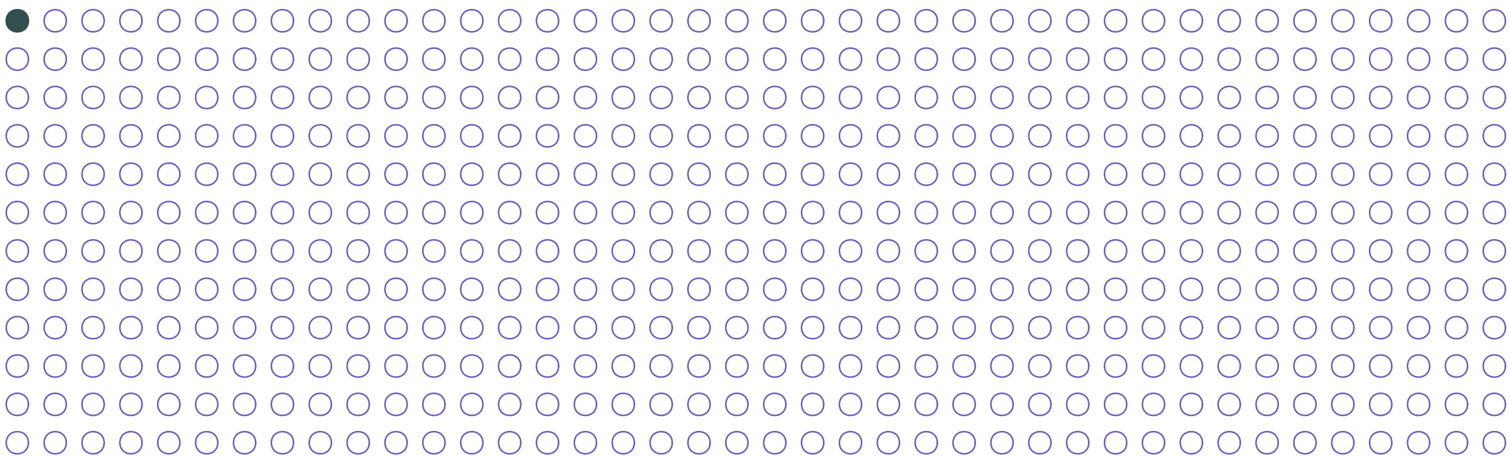
Leave-one-out cross validation (LOOCV) is perhaps the cross-validation method most similar to the validation-set approach.

- Your validation set is exactly one observation.
- *New* You repeat the validation exercise for every observation.
- *New* Estimate MSE as the mean across all observations.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

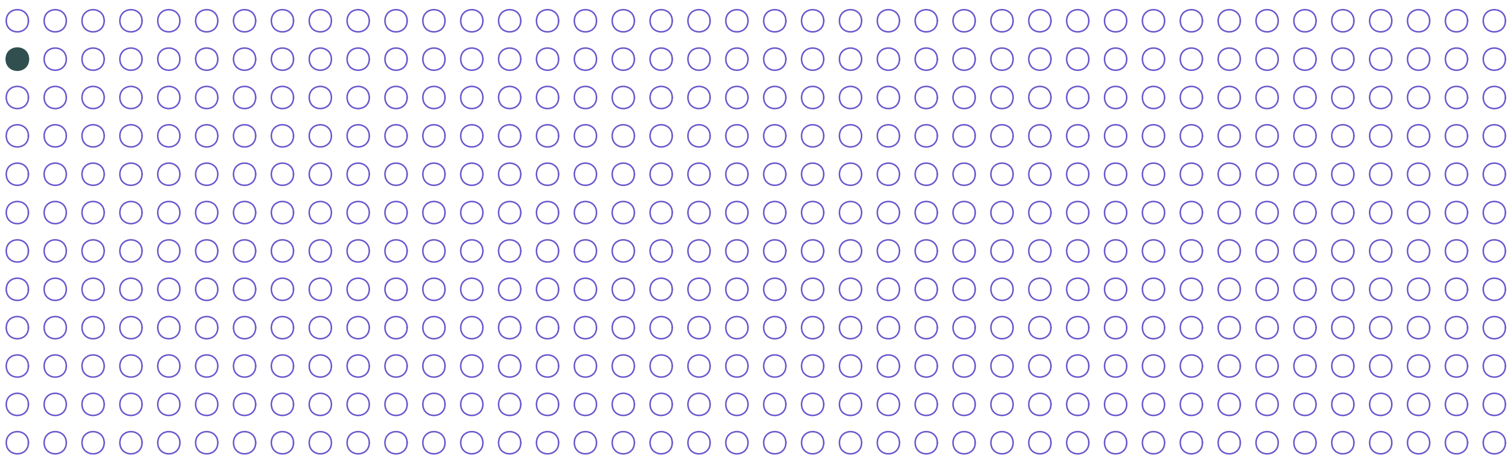


Observation 1's turn for validation produces MSE_1.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

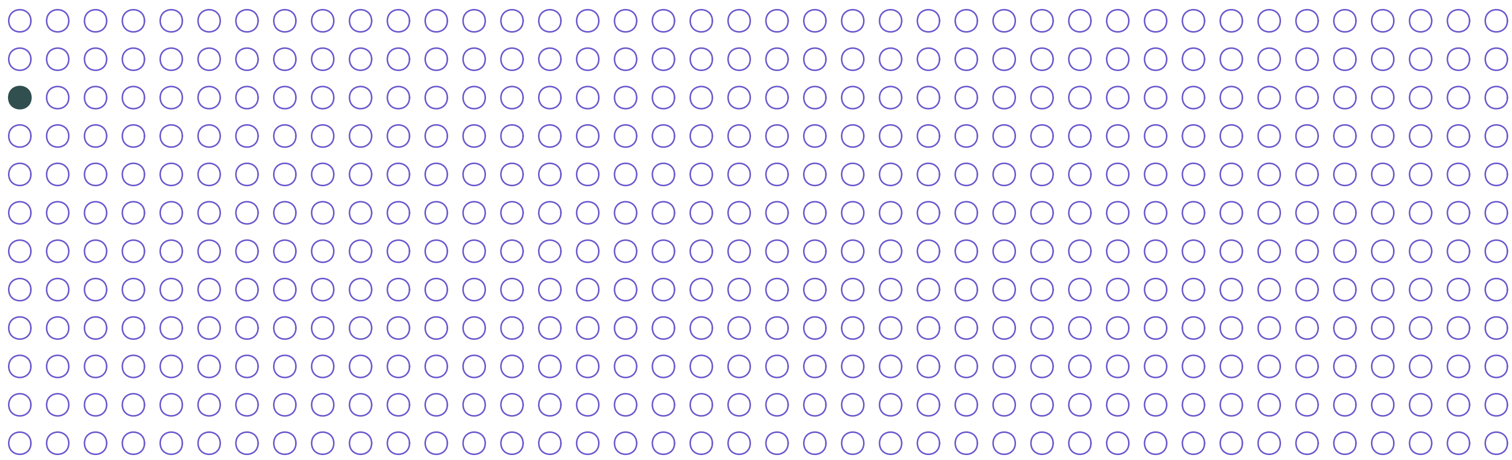


Observation 2's turn for validation produces MSE_2.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

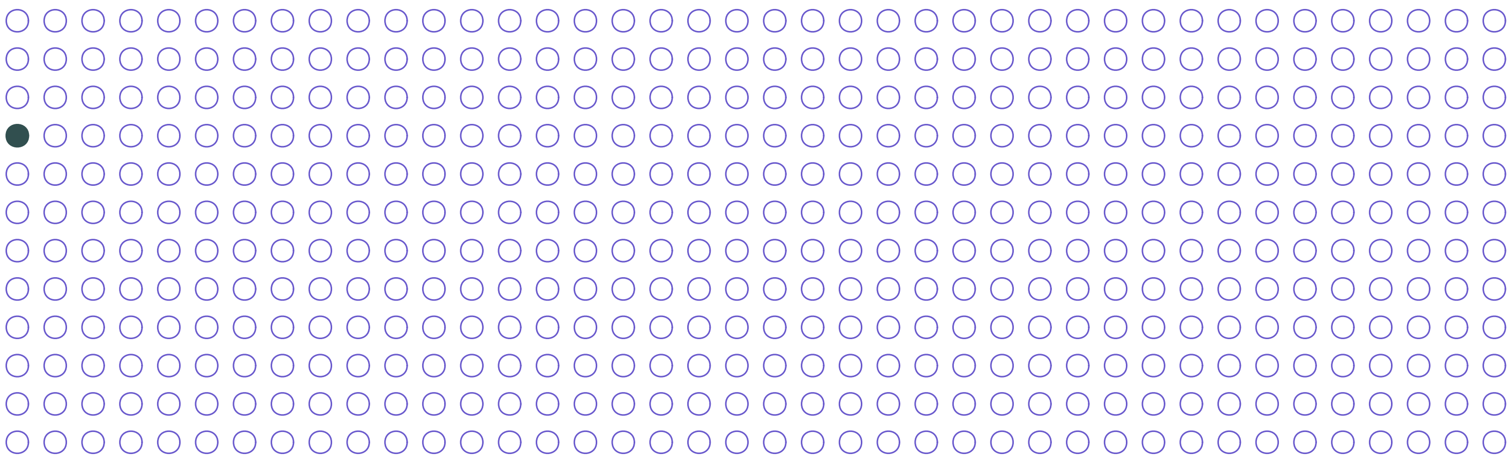


Observation 3's turn for validation produces MSE_3.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

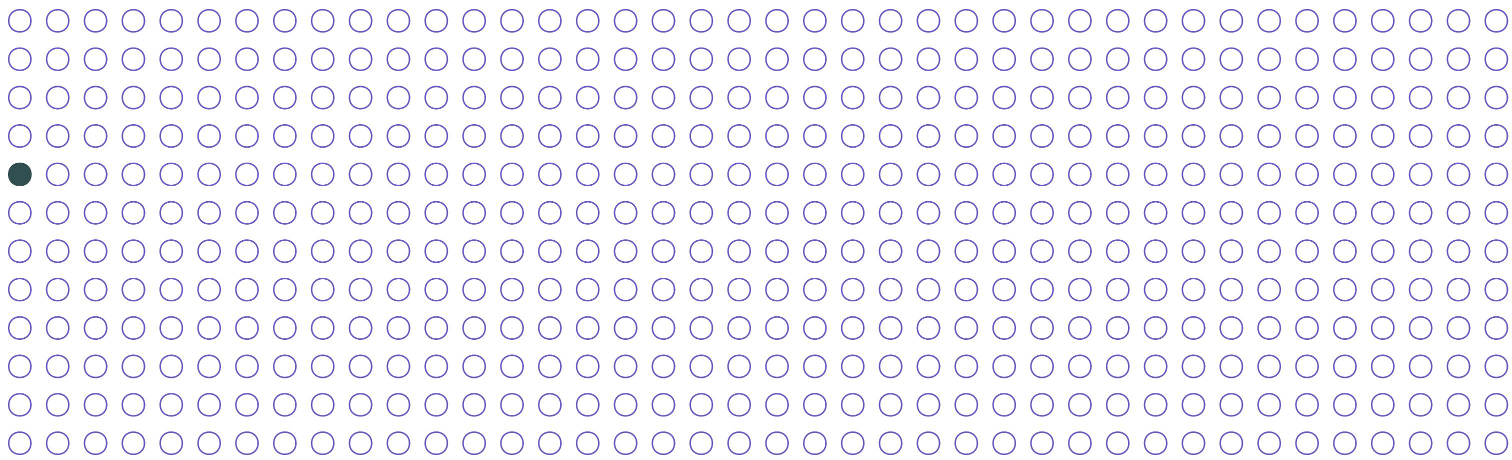


Observation 4's turn for validation produces MSE_4.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

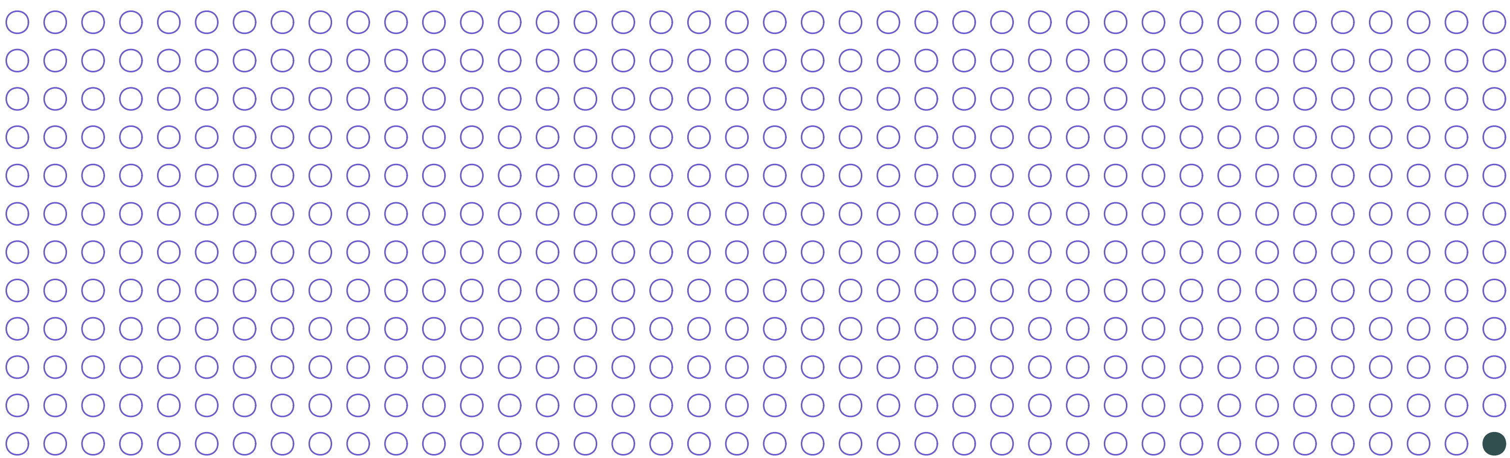


Observation 5's turn for validation produces MSE_5.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.



Observation n's turn for validation produces MSE_n.

Hold-out methods

Option 2: Leave-one-out cross validation

Because **LOOCV uses n-1 observations** to train the model,[†] MSE_i (validation MSE from observation i) is approximately unbiased for test MSE.

Problem MSE_i is a terribly noisy estimator for test MSE (albeit ≈unbiased).

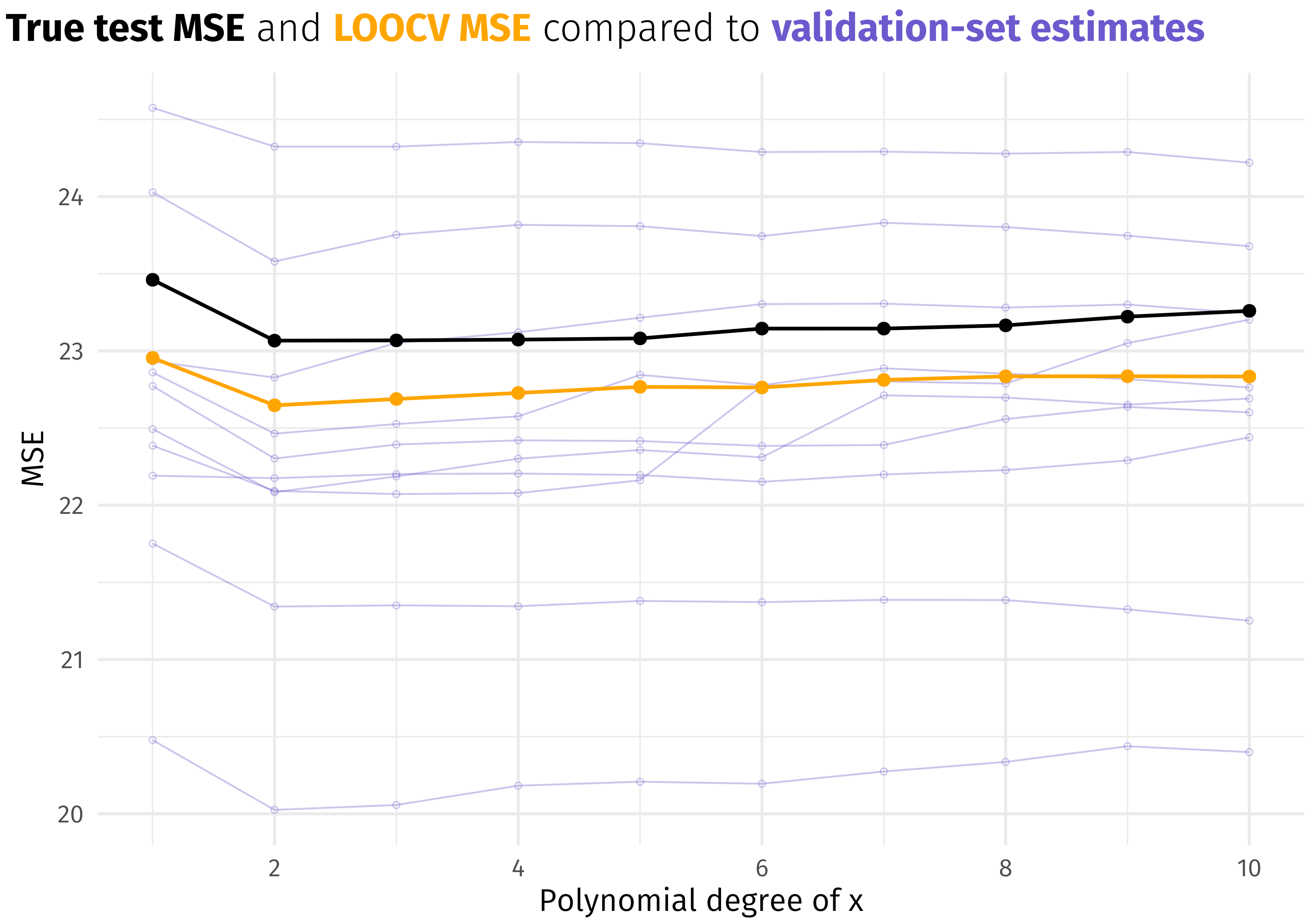
Solution Take the mean!

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i$$

1. LOOCV **reduces bias** by using n-1 (almost all) observations for training.

   1. LOOCV **resolves variance**: it makes all possible comparison (no dependence upon which validation-test split you make).

† And because often n-1 ≈ n.


True test MSE and LOOCV MSE compared to validation-set estimates
MSE
24
23
22
21
20
2
4
6
8
10
Polynomial degree of x

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

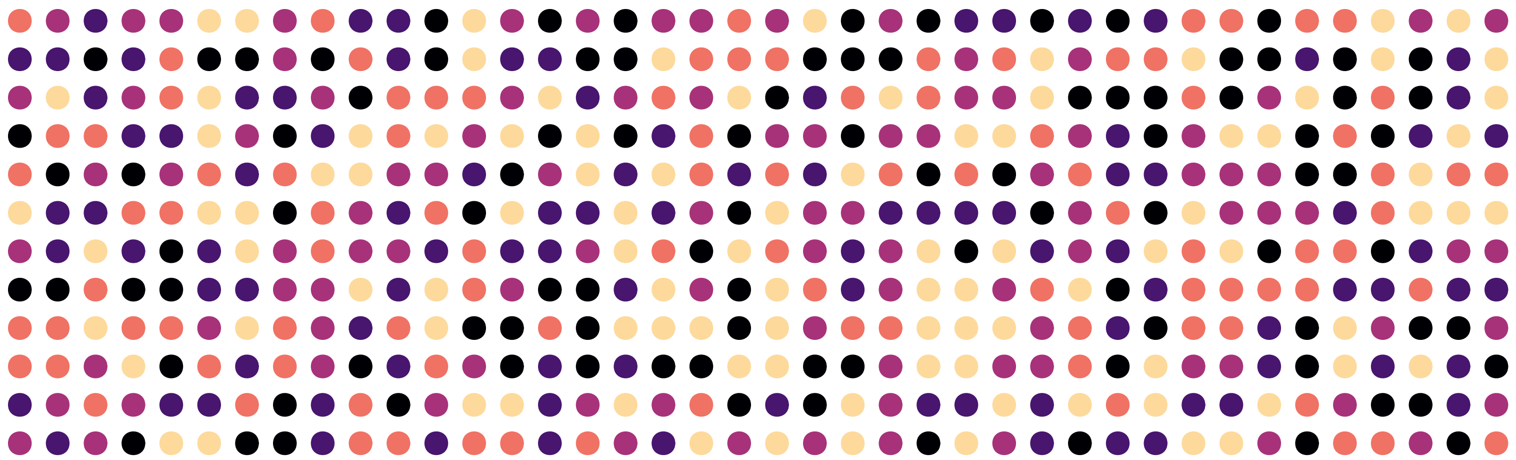
Benefits?1. **Less computationally demanding** (fit model $k =$ 5 or 10 times; not n).2. **Greater accuracy** (in general) due to bias-variance tradeoff! - Somewhat higher bias, relative to LOOCV: $n-1$ vs. $(k-1)/k$. - Lower variance due to high-degree of correlation in LOOCV MSE_i.🤯

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i$$



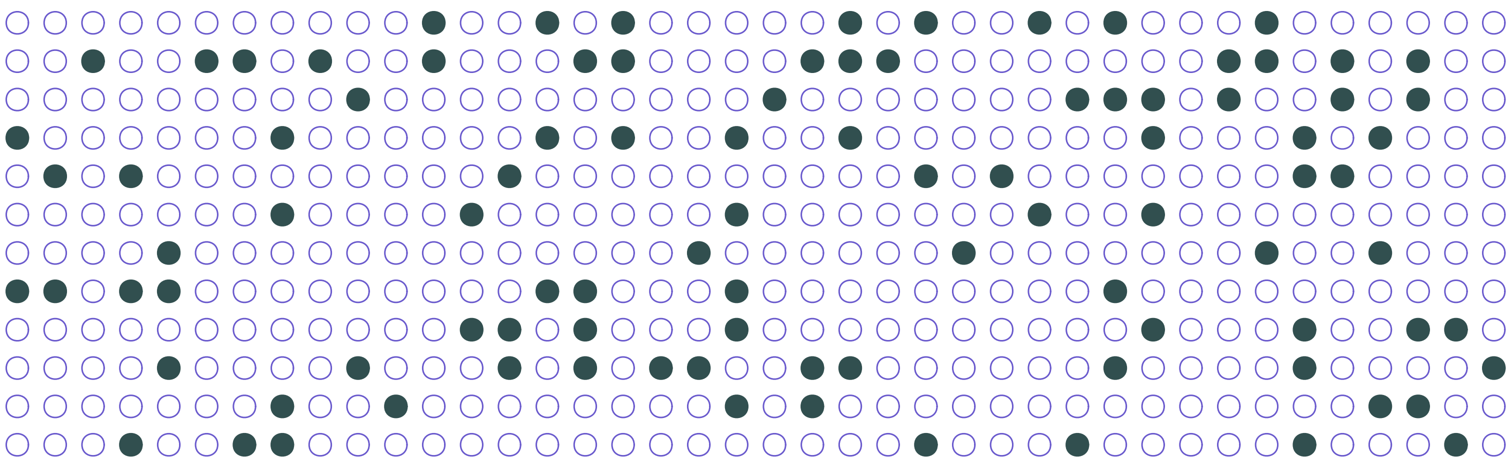
Our $k = 5$ folds.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



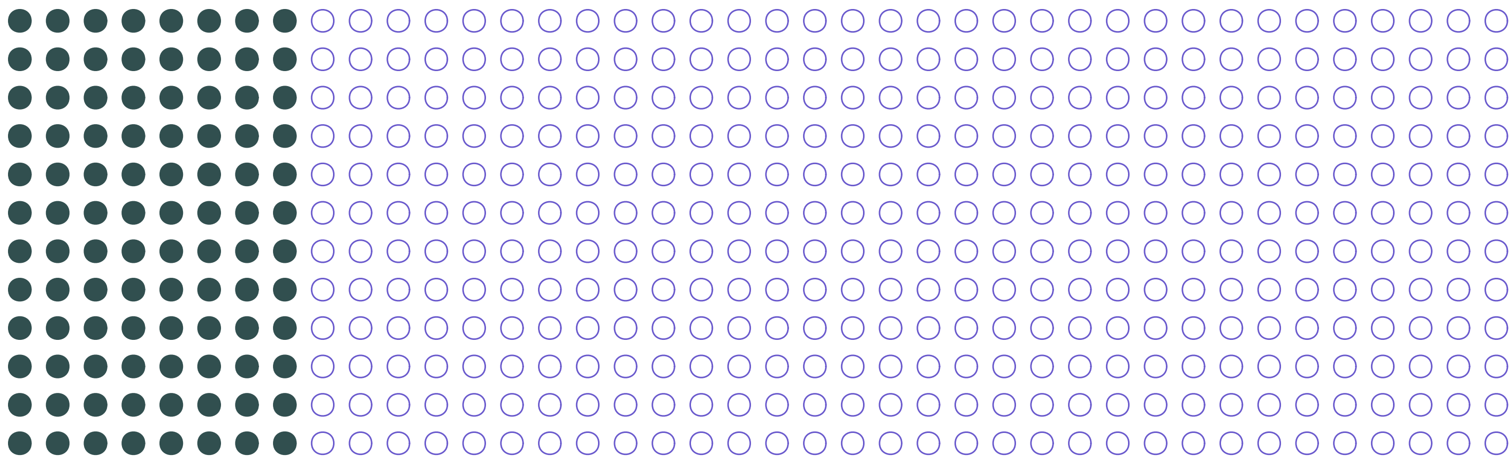
Each fold takes a turn at **validation**. The other $k - 1$ folds **train**.

Hold-out methods

Option 3: k-fold cross validation

With $\boldsymbol{k}$-fold cross validation, we estimate test MSE as

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i$$



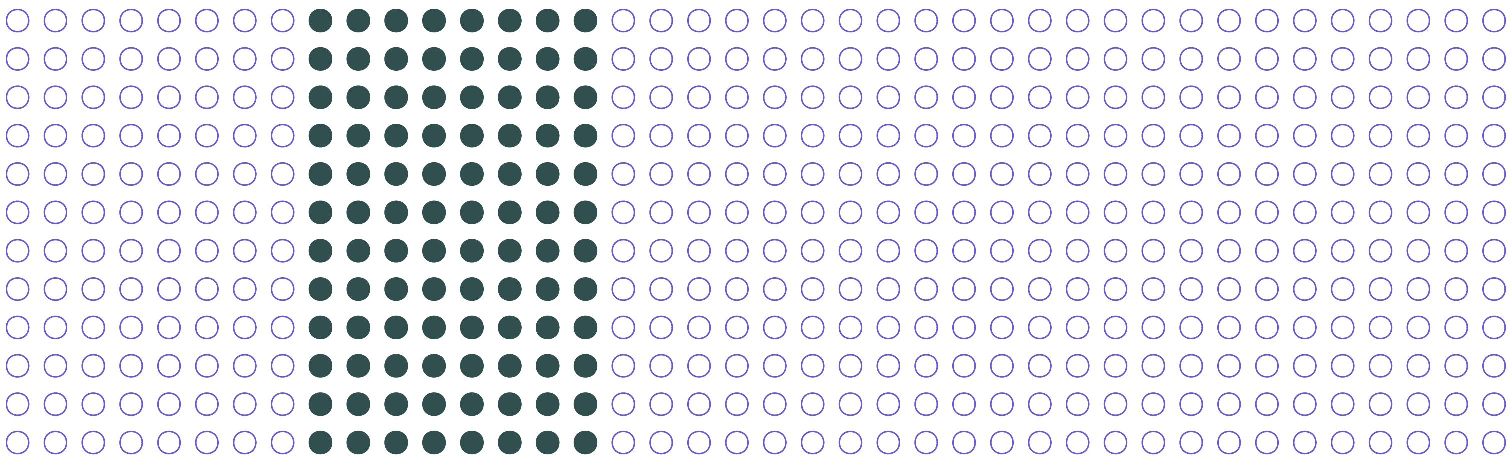
For $\boldsymbol{k} = 5$, fold number 1 as the **validation set** produces $\text{MSE}_{\text{k=1}}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



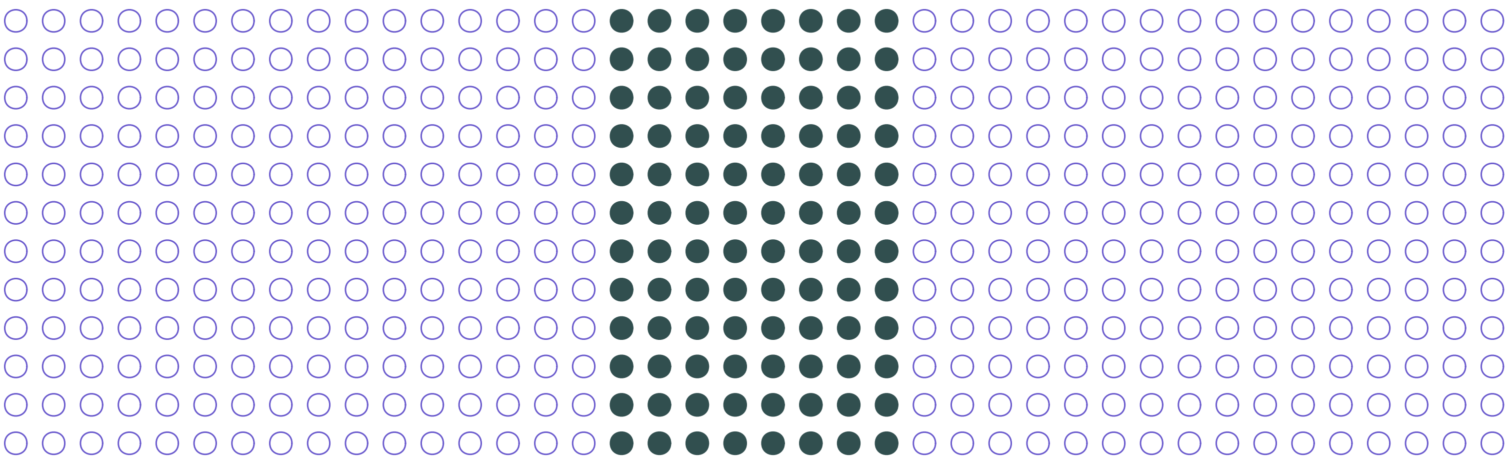
For $k = 5$, fold number 2 as the **validation set** produces $\text{MSE}_{k=2}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$



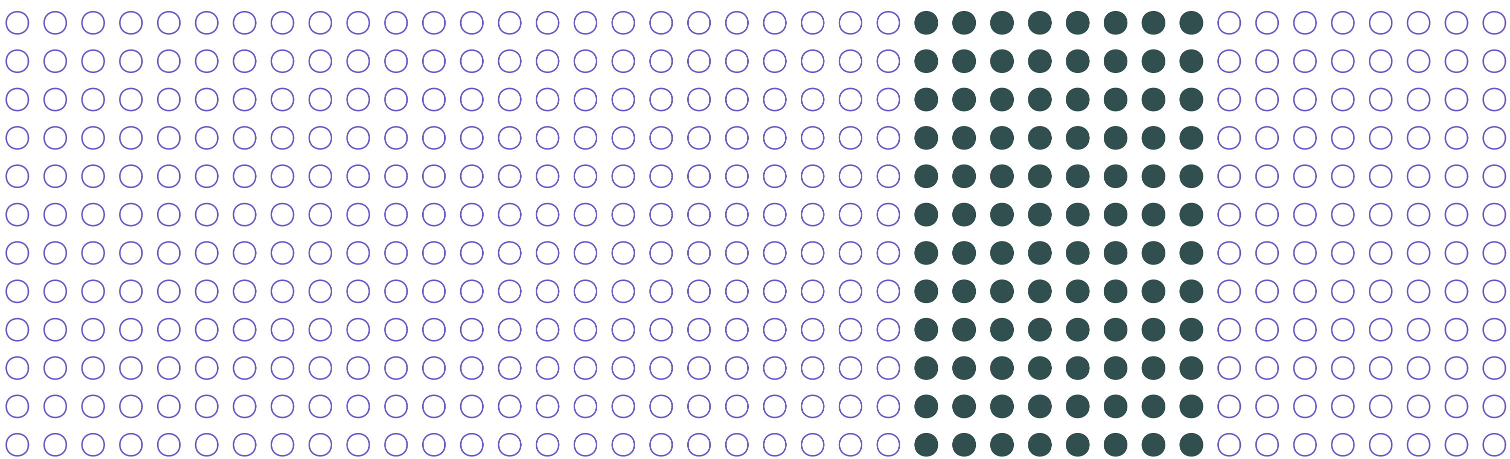
For $k = 5$, fold number 3 as the **validation set** produces $\mathrm{MSE}_{k=3}$.

Hold-out methods

Option 3: k-fold cross validation

With $\boldsymbol{k}$-fold cross validation, we estimate test MSE as

$$\mathrm{CV}_{(k)} = \frac{1}{\boldsymbol{k}} \sum_{i=1}^{k} \mathrm{MSE}_i$$



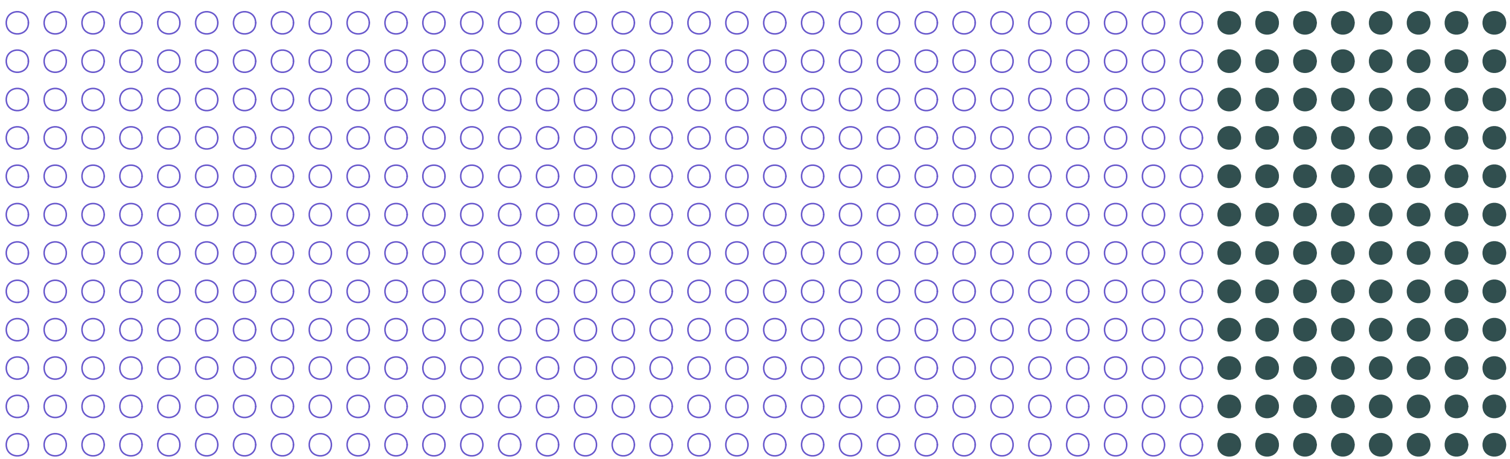
For $\boldsymbol{k} = 5$, fold number 4 as the **validation set** produces $\mathrm{MSE}_{k=4}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



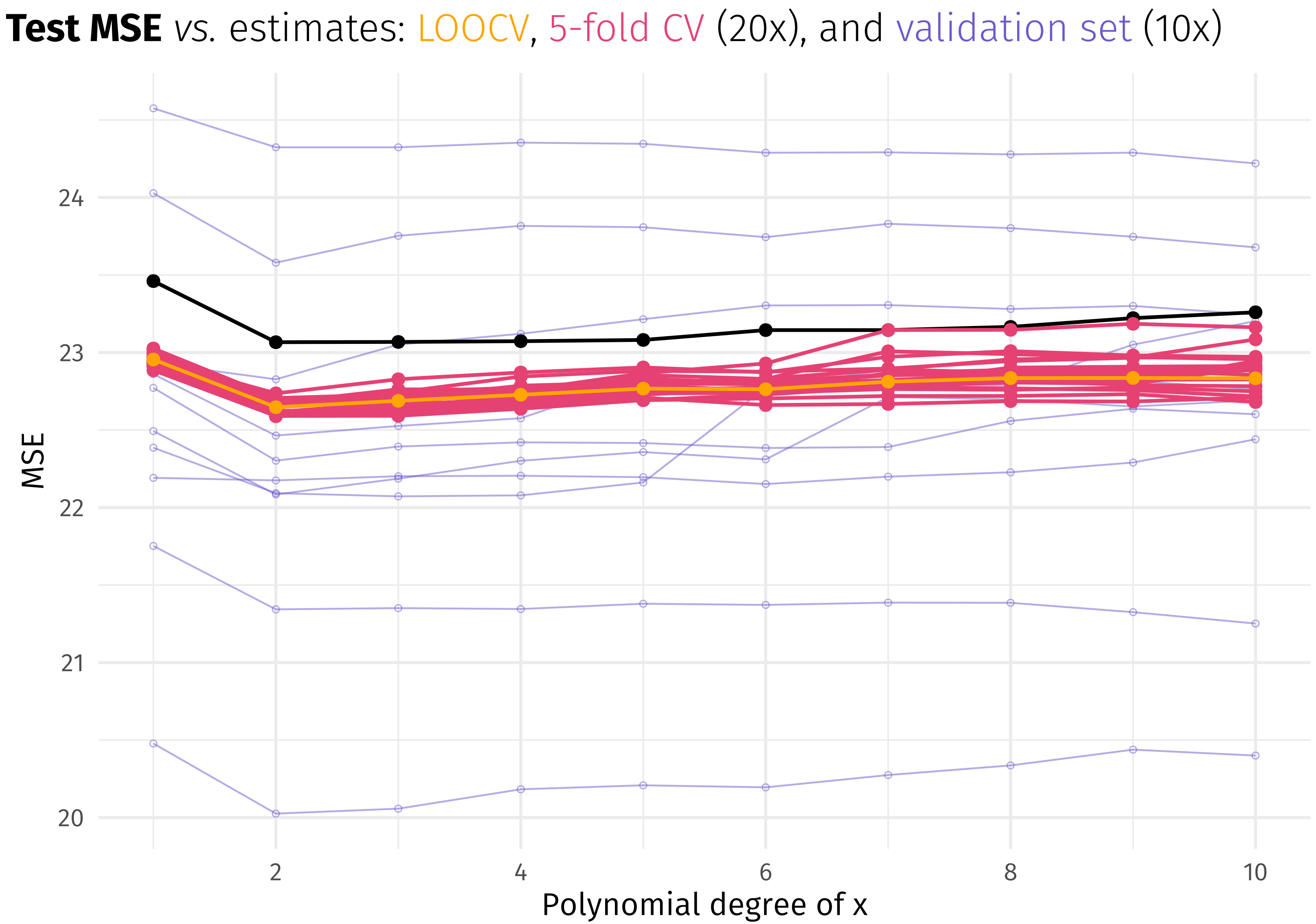
For $k = 5$, fold number 5 as the **validation set** produces $\text{MSE}_{k=5}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$



Test MSE vs. estimates: LOOCV, 5-fold CV (20x), and validation set (10x)
24
23
22
21
20
MSE
2
4
6
8
10
Polynomial degree of x

Note: Each of these methods extends to classification settings, *e.g.*, LOOCV

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i \neq \hat{y}_i)$$

Hold-out methods

Caveat

So far, we've treated each observation as separate/independent from each other observation.

The methods that we've defined so far actually need this independence.

The bootstrap

The bootstrap

Intro

The **bootstrap** is a resampling method often used to quantify the uncertainty (variability) underlying an estimator or learning method.

Hold-out methods

- randomly divide the sample into training and validation subsets
- train and validate ("test") model on each subset/division

Bootstrapping

- randomly samples **with replacement** from the original sample
- estimates model on each of the *bootstrap samples*

The bootstrap

Intro

Estimating a estimate's standard error involves assumptions and theory.[†]

There are times this derivation is difficult or even impossible, *e.g.*,

$$\mathbf{Var}\left(\frac{\hat{\beta}_1}{1-\hat{\beta}_2}\right)$$

The bootstrap can help in these situations.

Rather than deriving an estimator's variance, we use bootstrapped samles to build a distribution and then learn about the estimator's variance.

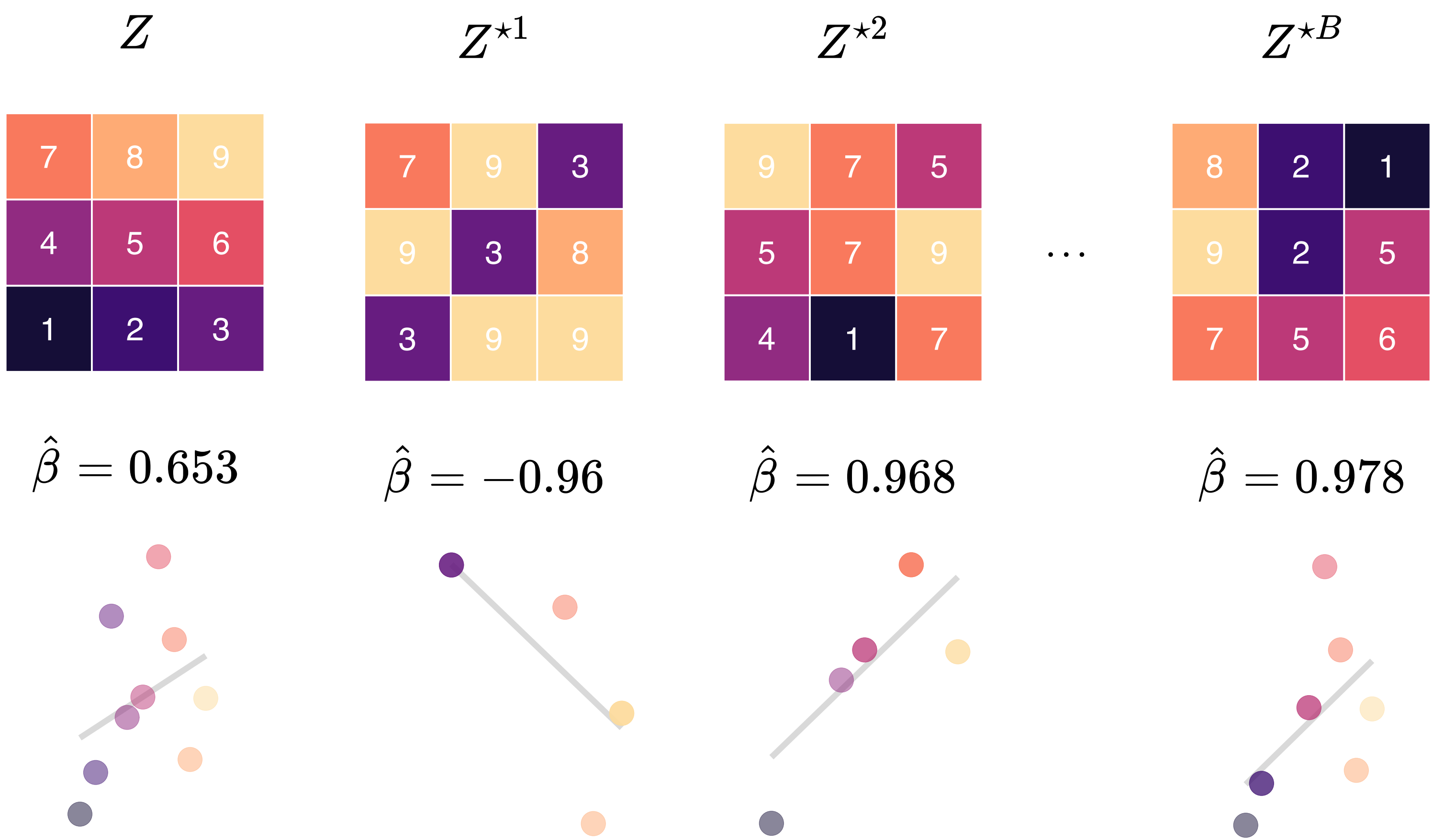
† Recall the standard-error estimator for OLS.

Intuition

Idea: Bootstrapping builds a distribution for the estimate using the variability embedded in the training sample.

The bootstrap

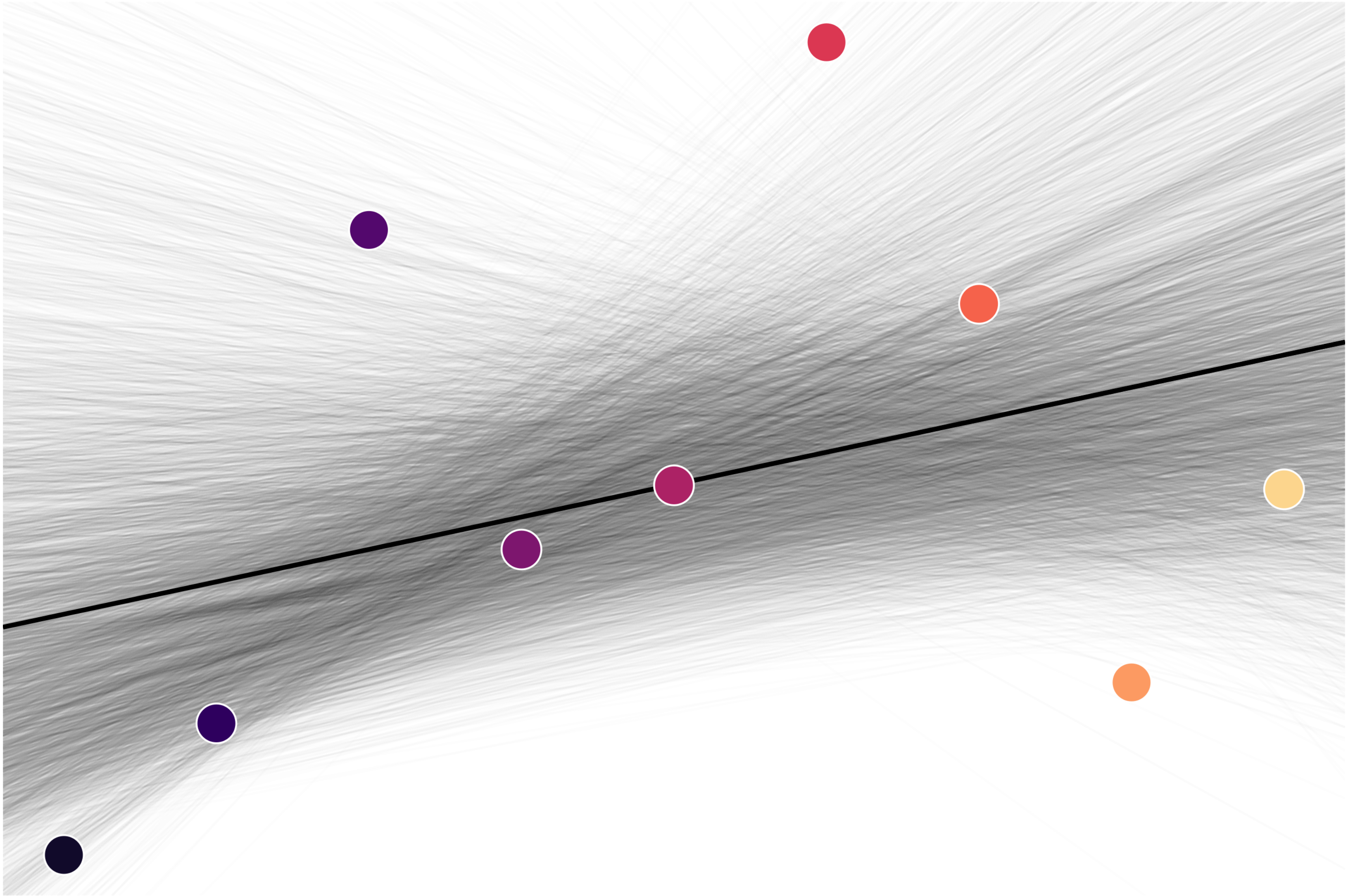
Graphically



The bootstrap

Running this bootstrap 10,000 times

```r
plan(multiprocess, workers = 10)
# Set a seed
set.seed(123)
# Run the simulation 1e4 times
boot_df <- future_map_dfr(
  # Repeat sample size 100 for 1e4 times
  rep(n, 1e4),
  # Our function
  function(n) {
    # Estimates via bootstrap
    est <- lm(y ~ x, data = z[sample(1:n, n, replace = T), ])
    # Return a tibble
    data.frame(int = est$coefficients[1], coef = est$coefficients[2])
  },
  # Let furrr know we want to set a seed
  .options = future_options(seed = T)
)
```



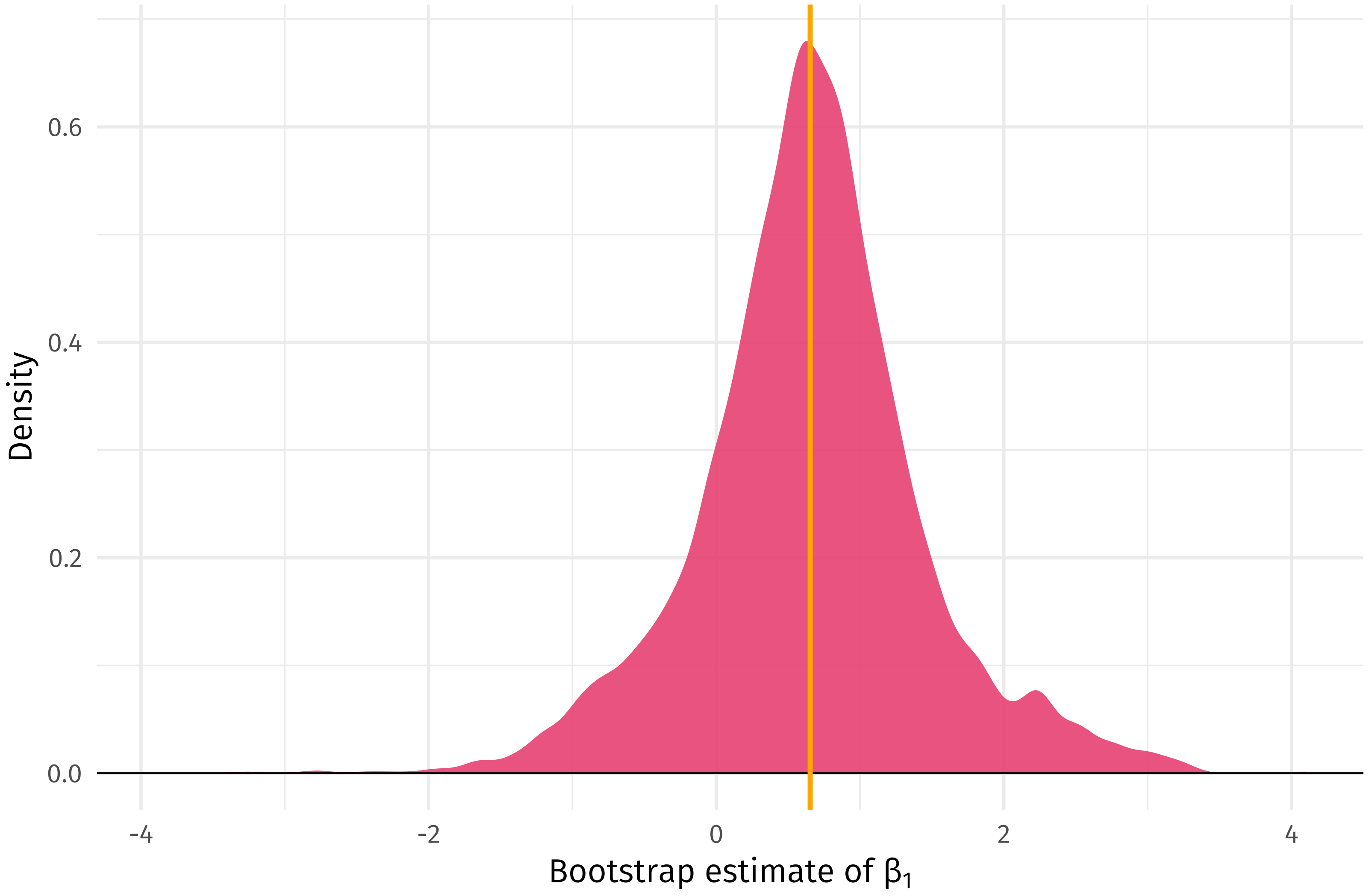
Comparison: Standard-error estimates

The **bootstrapped standard error** of $\hat{\alpha}$ is the standard deviation of the $\hat{\alpha}^{\star b}$

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\left(\hat{\alpha}^{\star b} - \frac{1}{B}\sum_{\ell=1}^{B}\hat{\alpha}^{\star \ell}\right)^2}$$

This 10,000-sample bootstrap estimates $\mathbf{S.E.}\left(\hat{\beta}_1\right) \approx 0.786$.

If we go the old-fashioned OLS route, we estimate 0.673.



Density
0.6
0.4
0.2
0.0
-4
-2
0
2
4
Bootstrap estimate of β1

Resampling

Review

Previous resampling methods

- Split data into **subsets**: n_v validation and n_t training $(n_v + n_t = n)$.
- Repeat estimation on each subset.
- Estimate the true test error (to help tune flexibility).

Bootstrap

- Randomly samples from training data **with replacement** to generate B "samples", each of size n.
- Repeat estimation on each subset.
- Estimate the variance estimate using variability across B samples.

Sources

These notes draw upon

- An Introduction to Statistical Learning (*ISL*)
  James, Witten, Hastie, and Tibshirani
- Python Data Science Handbook
  Jake VanderPlas

Table of contents

Admin

- Today
- Upcoming

Review

- Regression and loss
- Classification
- KNN
- The bias-variance tradeoff

Examples

- Validation-set simulation
- LOOCV MSE
- k-fold CV

Resampling

- Intro
- Hold-out methods
  - Validation sets
  - LOOCV
  - k-fold cross validation
- The bootstrap
  - Intro
  - Graphically
  - Example

Other

- Sources/references