Lecture `004`

Regression strikes back

Edward Rubin

Admin

Admin

Today

In-class

- A roadmap (where are we going?)
- Linear regression and model selection

Admin

Upcoming

Readings

- *Today*
  - *ISL* Ch. 3 and 6.1
- *Next*
  - *ISL* Ch. 6 and 4

Problem set Next problem set very soon!

Roadmap

Where are we?

We've essentially covered the central topics in statistical learning[†]

- Prediction and inference
- Supervised *vs.* unsupervised methods
- Regression and classification problems
- The dangers of overfitting
- The bias-variance tradeoff
- Model assessment
- Holdouts, validation sets, and cross validation[††]
- Model training and tuning
- Simulation

† Plus a few of the "basic" methods: OLS regression and KNN.

†† And the bootstrap!

Roadmap

Where are we going?

Next, we will cover many common machine-learning algorithms, *e.g.*,

- Decision trees
- Random forests and ensemble techniques
- SVM
- Neural nets
- Clustering

Roadmap

Where are we going?

Next, we will cover many common machine-learning algorithms, *e.g.*,

- Decision trees
- Random forests and ensemble techniques
- SVM
- Neural nets
- Clustering

But first, we return to good old **linear regression**—in a new light...

- Linear regression
- Variable/model selection and LASSO/Ridge regression
- *Plus:* Logistic regression and discriminant analysis

Roadmap

Why return to regression?

Motivation 1

We have new tools. It might help to first apply them in a **familiar** setting.

Roadmap

Why return to regression?

Motivation 1

We have new tools. It might help to first apply them in a **familiar** setting.

Motivation 2

We have new tools. Maybe linear regression will be (even) **better now?**

E.g., did (cross) validation help you beat your old model?

Roadmap

Why return to regression?

Motivation 1

We have new tools. It might help to first apply them in a **familiar** setting.

Motivation 2

We have new tools. Maybe linear regression will be (even) **better now?**
E.g., did (cross) validation help you beat your old model?

Motivation 3

> many fancy statistical learning approaches can be seen as **generalizations or extensions of linear regression**.

Source: ISL, p. 59; emphasis added

Linear regression

Regression regression

Recall Linear regression "fits" coefficients $\beta_0, \ldots, \beta_p$ for a model

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_p x_{p,i} + \varepsilon_i$$

and is often applied in two distinct settings with fairly distinct goals:

1. **Causal inference** estimates and interprets the coefficients.

2. **Prediction** focuses on accurately estimating outcomes.

Regardless of the goal, the way we "fit" (estimate) the model is the same.

Linear regression

Fitting the regression line

As is the case with many statistical learning methods, regression focuses on minimizing some measure of loss/error.

$$e_i = y_i - \hat{y}_i$$

Linear regression

Fitting the regression line

As is the case with many statistical learning methods, regression focuses on minimizing some measure of loss/error.

$$e_i = y_i - \hat{y}_i$$

Linear regression uses the L_2 loss function—also called *residual sum of squares* (RSS) or *sum of squared errors* (SSE)

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2 = \sum_{i=1}^{n} e_i^2$$

Specifically: OLS chooses the $\hat{\beta}_j$ that **minimize RSS**.

Performance

There's a large variety of ways to assess the fit† of linear-regression models.

Residual standard error (**RSE**)

$$\text{RSE} = \sqrt{\frac{1}{n-p-1}\text{RSS}} = \sqrt{\frac{1}{n-p-1}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

R-squared (**R^2**)

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \quad \text{where} \quad \text{TSS} = \sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2$$

† or predictive performance

Linear regression

Performance and overfit

As we've seen throughout the course, we need to be careful **not to overfit**.

Performance and overfit

As we've seen throughout the course, we need to be careful **not to overfit**.

R^2 provides no protection against overfitting—and actually encourages it.

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Add a new variable: RSS $\downarrow$ and TSS is unchanged. Thus, R^2 increases.

Linear regression

Performance and overfit

As we've seen throughout the course, we need to be careful **not to overfit**.

R^2 provides no protection against overfitting—and actually encourages it.

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Add a new variable: RSS $\downarrow$ and TSS is unchanged. Thus, R^2 increases.

RSE *slightly* penalizes additional variables:

$$\text{RSE} = \sqrt{\frac{1}{n-p-1}\text{RSS}}$$

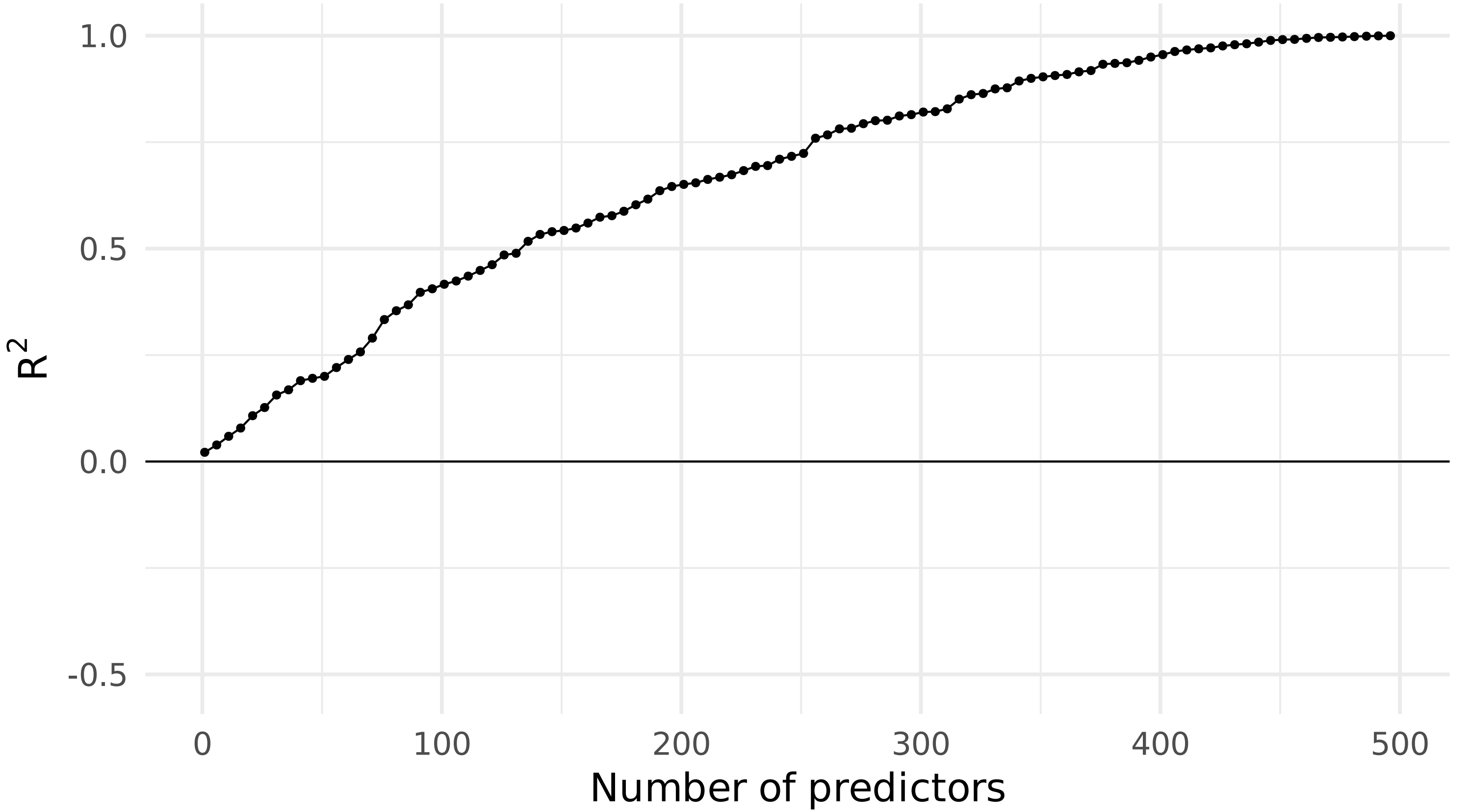
Add a new variable: RSS $\downarrow$ but p increases. Thus, RSE's change is uncertain.

Example

Let's see how **R^2** and **RSE** perform with 500 very weak predictors.

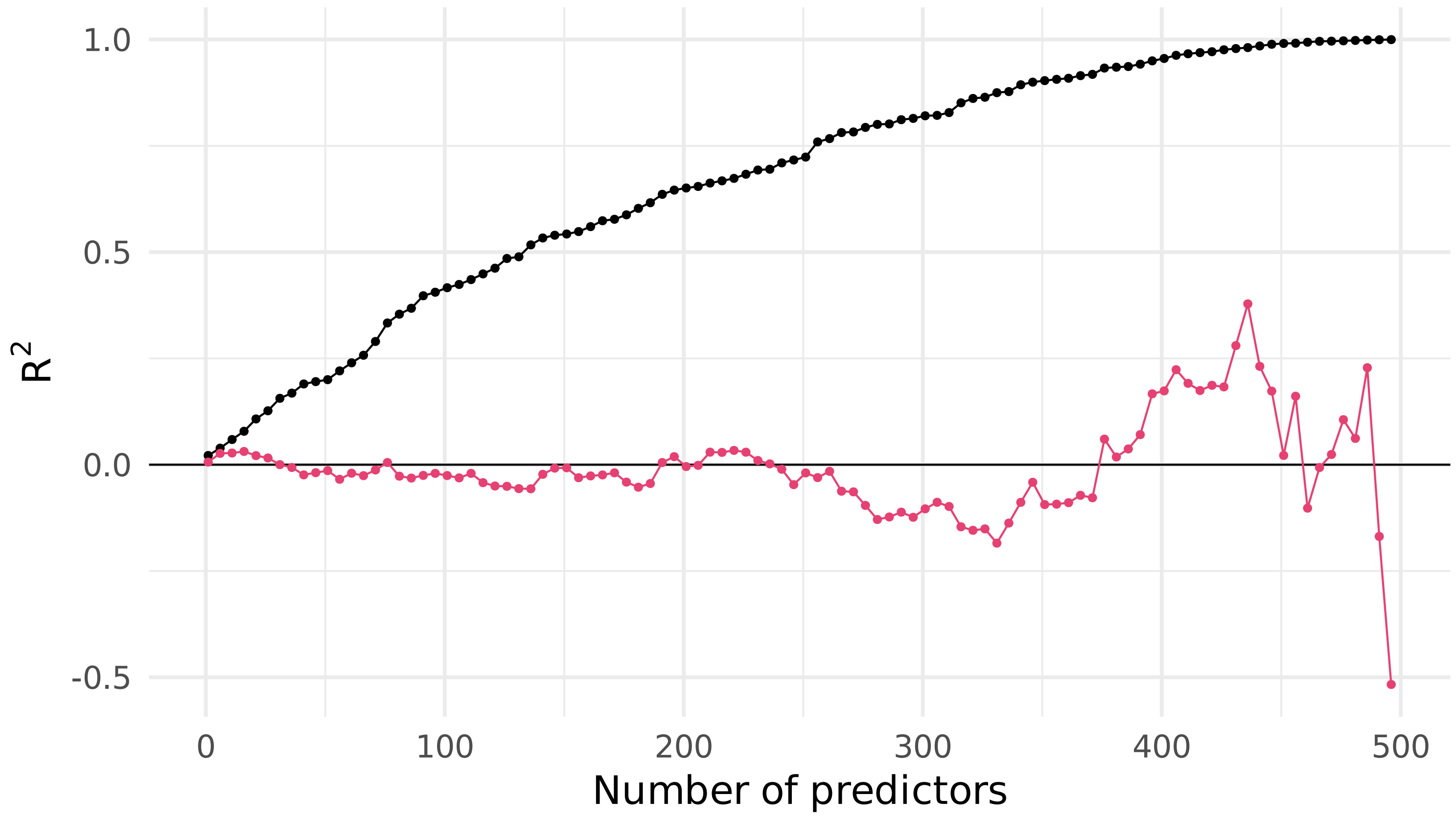
To address overfitting, we can compare **in-** *vs.* **out-of-sample** performance.

In-sample R^2 mechanically increases as we add predictors.



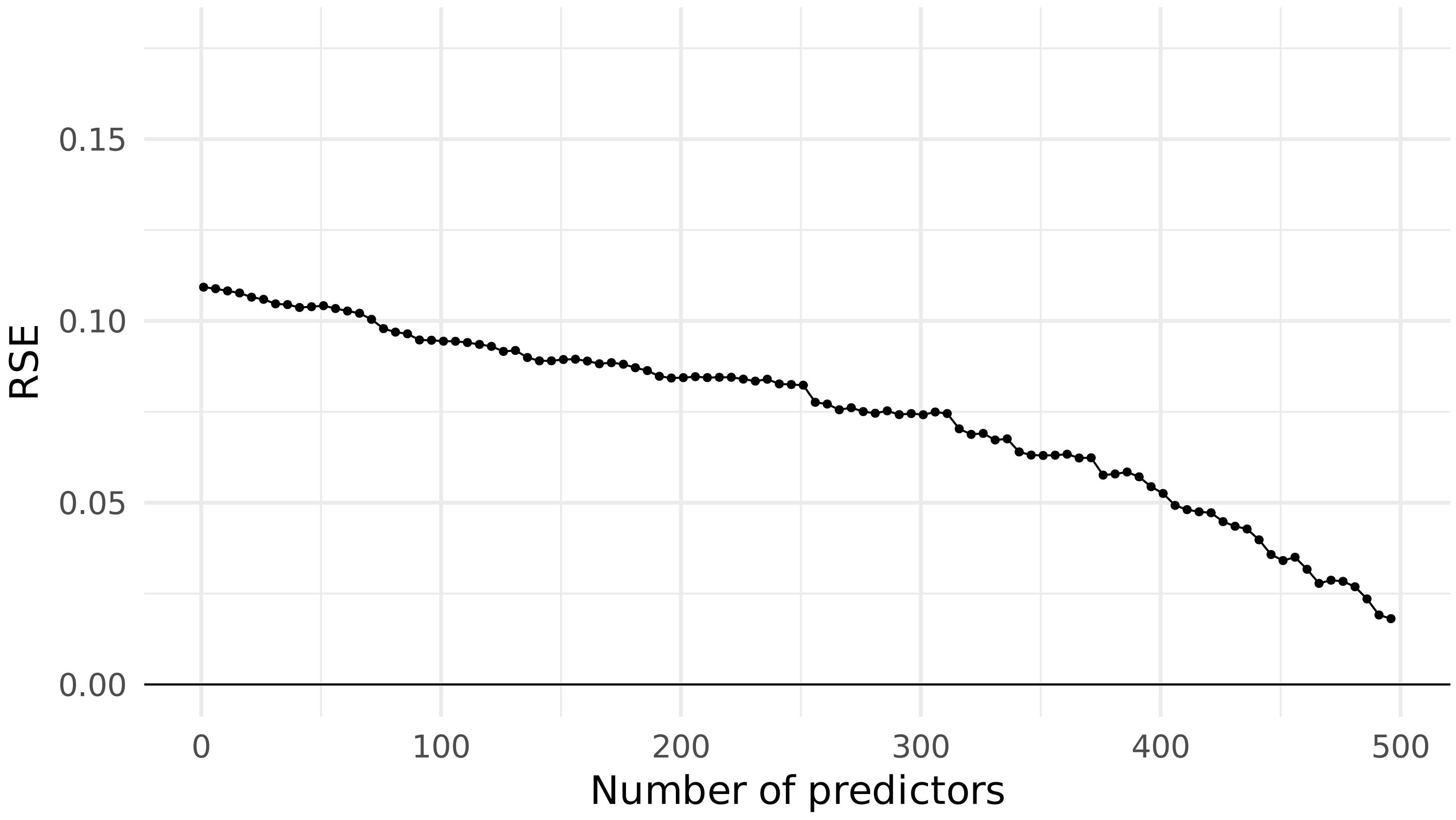
In-sample R^2 mechanically increases as we add predictors.

Out-of-sample R^2 does not.

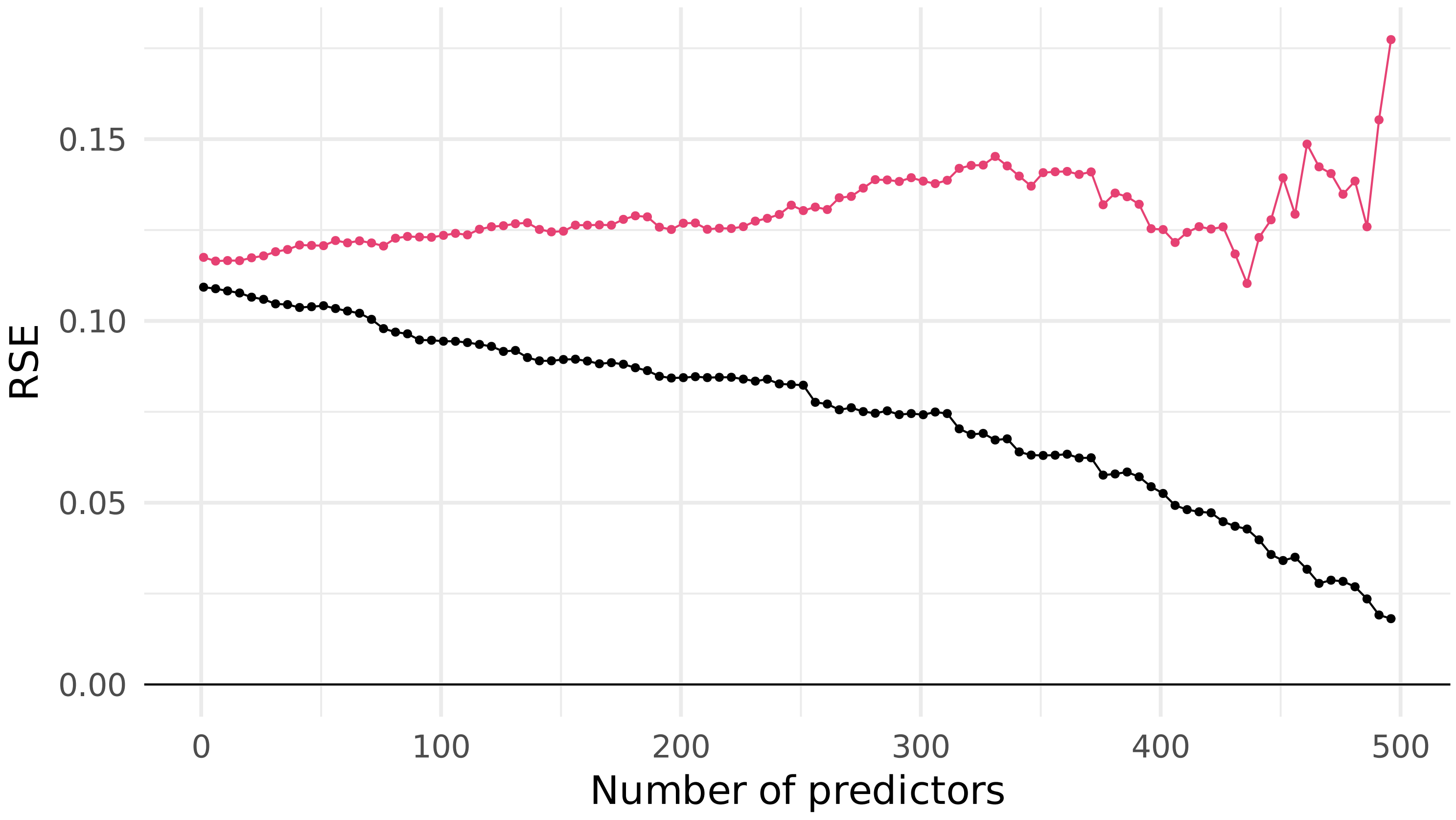


What about RSE? Does its penalty *help*?

Despite its penalty for adding variables, **in-sample RSE** still can overfit,



Despite its penalty for adding variables, **in-sample RSE** still can overfit, as evidenced by **out-of-sample RSE**.



Penalization

RSE is not the only way to penalization the addition of variables.[†]

Adjusted R^2 is another *classic* solution.

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$$

Adj. R^2 attempts to "fix" R^2 by **adding a penalty for the number of variables**.

† We'll talk about other penalization methods (LASSO and Ridge) shortly.

Penalization

RSE is not the only way to penalization the addition of variables.†

Adjusted R² is another *classic* solution.

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$$

Adj. R^2 attempts to "fix" R^2 by **adding a penalty for the number of variables**.

- RSS always decreases when a new variable is added.

† We'll talk about other penalization methods (LASSO and Ridge) shortly.

Penalization

RSE is not the only way to penalization the addition of variables.[†]

Adjusted R^2 is another *classic* solution.

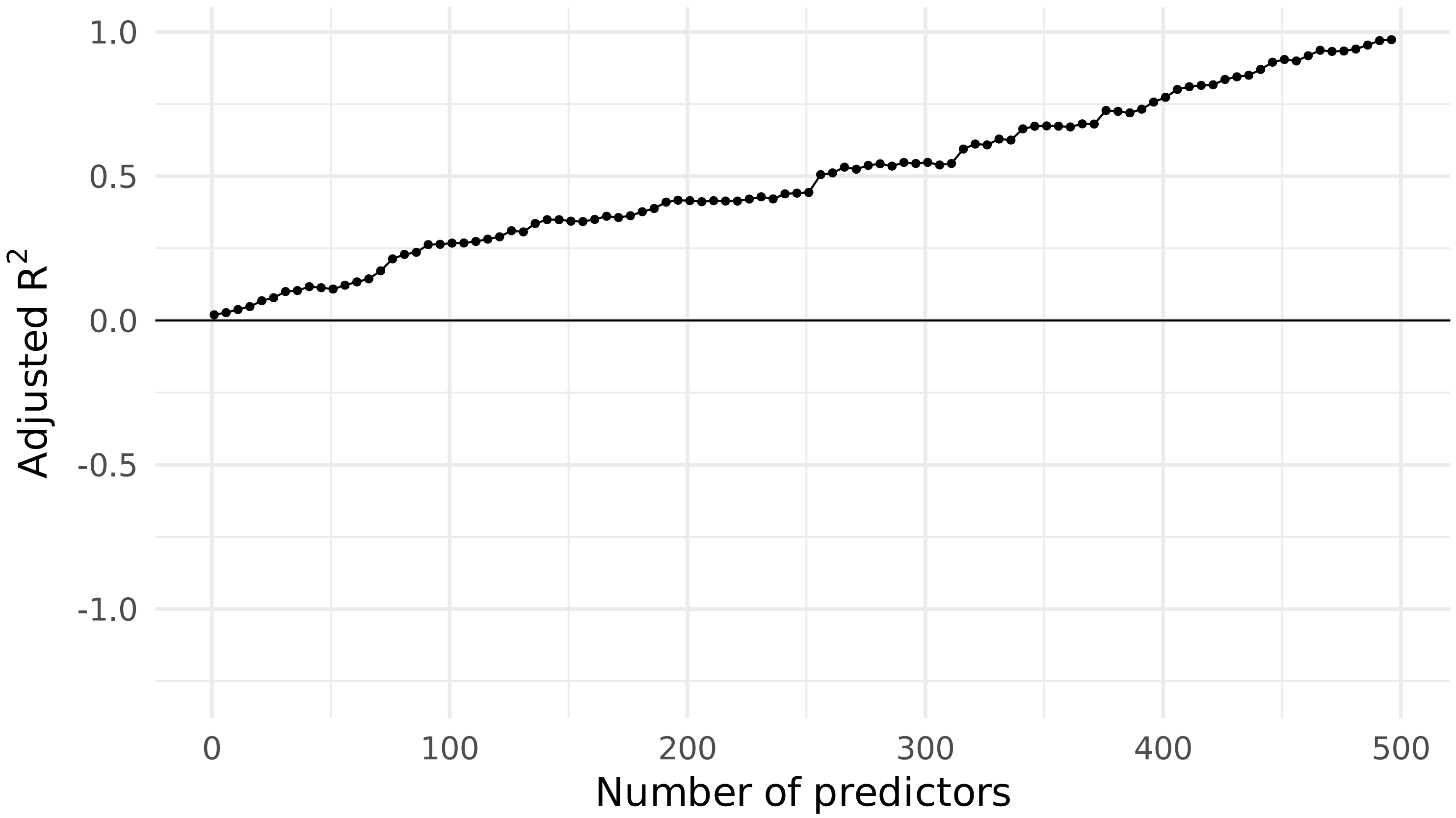
$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$$

Adj. R^2 attempts to "fix" R^2 by **adding a penalty for the number of variables**.

- RSS always decreases when a new variable is added.
- $\text{RSS}/(n-p-1)$ may increase or decrease with a new variable.

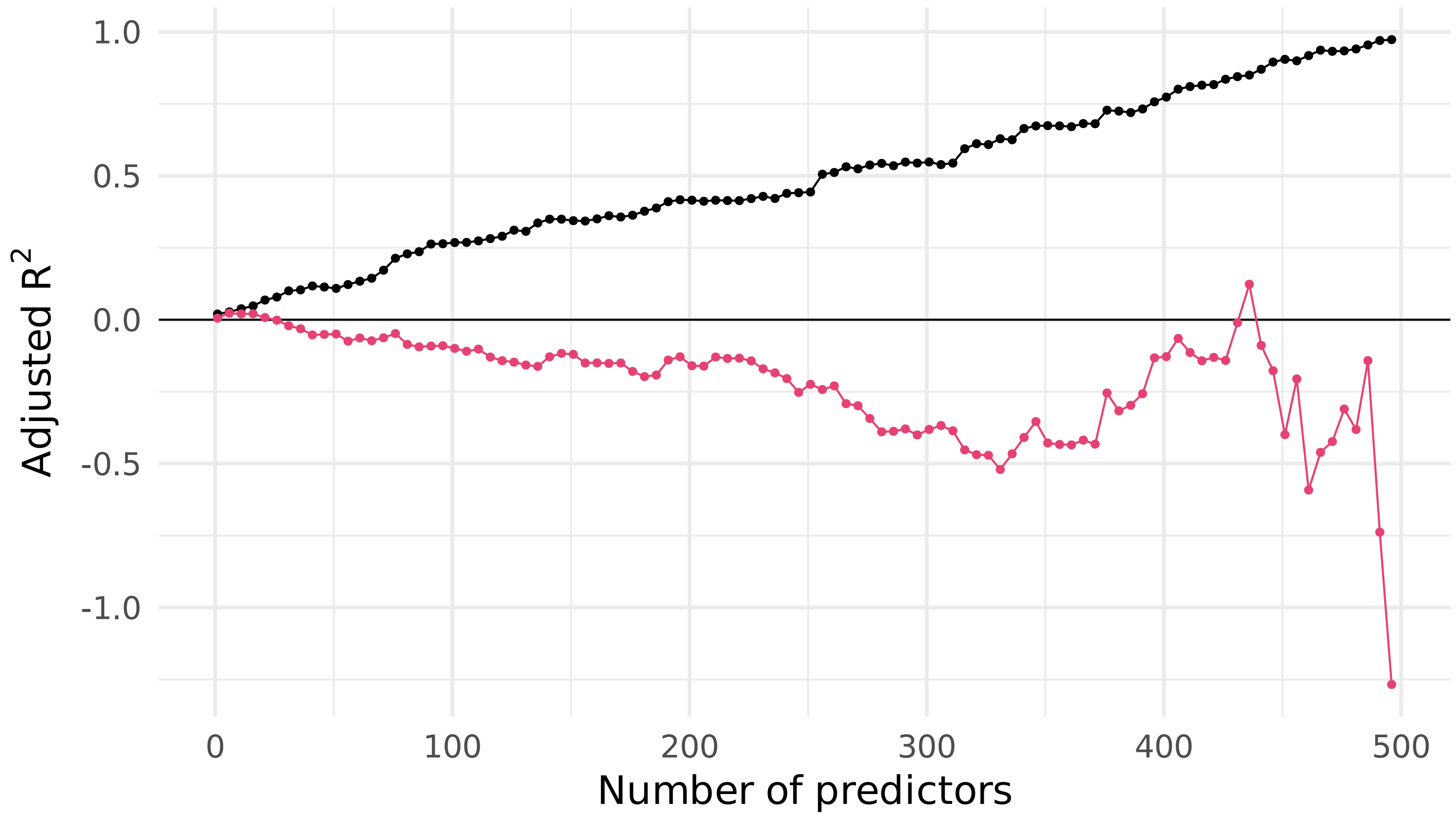
† We'll talk about other penalization methods (LASSO and Ridge) shortly.

However, **in-sample adjusted R^2** still can overfit.



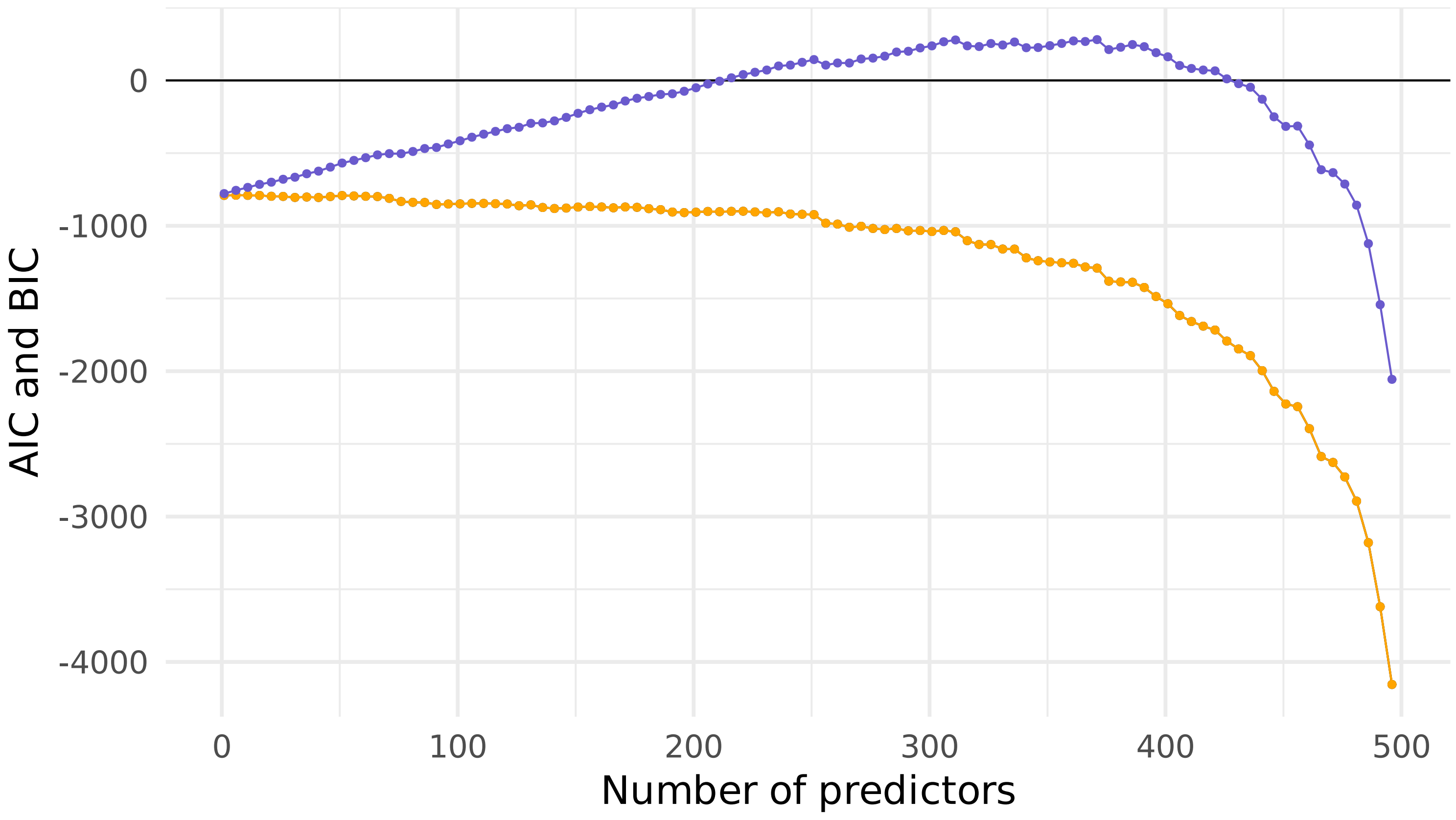
However, **in-sample adjusted R^2** still can overfit.

Illustrated by **out-of-sample adjusted R^2**.



Here are **in-sample** **AIC** and **BIC**.

Neither in-sample metric seems to entirely guard against overfitting.



Model selection

A better way?

R^2, adjusted R^2, and RSE each offer some flavor of model fit, but they appear **limited in their abilities to prevent overfitting**.

Model selection

A better way?

R^2, adjusted R^2, and RSE each offer some flavor of model fit, but they appear **limited in their abilities to prevent overfitting**.

We want a method to optimally select a (linear) model—balancing variance and bias and avoiding overfit.

Model selection

A better way?

R^2, adjusted R^2, and RSE each offer some flavor of model fit, but they appear **limited in their abilities to prevent overfitting**.

We want a method to optimally select a (linear) model—balancing variance and bias and avoiding overfit.

We'll discuss two (related) methods today:

1. **Subset selection** chooses a (sub)set of our p potential predictors

2. **Shrinkage** fits a model using all p variables but "shrinks" its coefficients

Subset selection

In **subset selection**, we

1. whittle down the p potential predictors (using some magic/algorithm)
2. estimate the chosen linear model using OLS

Model selection

Subset selection

In **subset selection**, we

1. whittle down the p potential predictors (using some magic/algorithm)
2. estimate the chosen linear model using OLS

How do we do the *whittling* (selection)?

Model selection

Subset selection

In subset selection, we

1. whittle down the p potential predictors (using some magic/algorithm)
2. estimate the chosen linear model using OLS

How do we do the *whittling* (selection)? We've got **options**.

- Best subset selection fits a model for every possible subset.
- Forward stepwise selection starts with only an intercept and tries to build up to the best model (using some fit criterion).
- Backward stepwise selection starts with all p variables and tries to drop variables until it hits the best model (using some fit criterion).
- Hybrid approaches are what their name implies (*i.e.*, hybrids).

Best subset selection

Best subset selection is based upon a simple idea: Estimate a model for every possible subset of variables; then compare their performances.

Model selection

Best subset selection

Best subset selection is based upon a simple idea: Estimate a model for every possible subset of variables; then compare their performances.

Q So what's the problem? (Why do we need other selection methods?)

Model selection

Best subset selection

Best subset selection is based upon a simple idea: Estimate a model for every possible subset of variables; then compare their performances.

Q So what's the problem? (Why do we need other selection methods?)
A "a model for **every possible subset**" can mean **a lot** (2^p) of models.

Model selection

Best subset selection

Best subset selection is based upon a simple idea: Estimate a model for every possible subset of variables; then compare their performances.

Q So what's the problem? (Why do we need other selection methods?)
A "a model for **every possible subset**" can mean **a lot** (2^p) of models.

E.g.,

- 10 predictors $\rightarrow$ 1,024 models to fit
- 25 predictors $\rightarrow$ >33.6 million models to fit
- 100 predictors $\rightarrow$ > 1 nonillion (>1.3 $\times$ 10^{30}) models to fit

Model selection

Best subset selection

Best subset selection is based upon a simple idea: Estimate a model for every possible subset of variables; then compare their performances.

Q So what's the problem? (Why do we need other selection methods?)
A "a model for **every possible subset**" can mean **a lot** (2^p) of models.

E.g.,

- 10 predictors $\rightarrow$ 1,024 models to fit
- 25 predictors $\rightarrow$ >33.6 million models to fit
- 100 predictors $\rightarrow$ > 1 nonillion (>1.3 $\times$ 10^{30}) models to fit

Even with plentiful, cheap computational power, we can run into barriers.

Model selection

Best subset selection

Computational constraints aside, we can implement **best subset selection** as

```
1. Define M_0 as the model with no predictors.
```

1. Define $\mathcal{M}_0$ as the model with no predictors.

2. For k in 1 to p:

   - Fit every possible model with k predictors.

   - Define $\mathcal{M}_k$ as the "best" model with k predictors.

3. Select the "best" model from $\mathcal{M}_0, \ldots, \mathcal{M}_p$.

As we've seen, RSS declines (and R^2 increases) with p, so we should use a cross-validated measure of model performance in step **3**.[†]

† Back to our distinction between test *vs.* training performance.

Model selection

Example dataset: `Credit`

We're going to use the `Credit` dataset from *ISL*'s R package `ISLR`.

Model selection

Example dataset: `Credit`

We're going to use the `Credit` dataset from *ISL*'s R package `ISLR`.

ID	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
1	14.9	3606	283	2	34	11	Male	No	Yes	Caucasian	333
2	106.0	6645	483	3	82	15	Female	Yes	Yes	Asian	903
3	104.6	7075	514	4	71	11	Male	No	No	Asian	580
4	148.9	9504	681	3	36	11	Female	No	No	Asian	964
5	55.9	4897	357	2	68	16	Male	No	Yes	Caucasian	331
6	80.2	8047	569	4	77	10	Male	No	No	Caucasian	1151
7	21.0	3388	259	2	37	12	Female	No	No	African American	203

The `Credit` dataset has 400 observations on 12 variables.

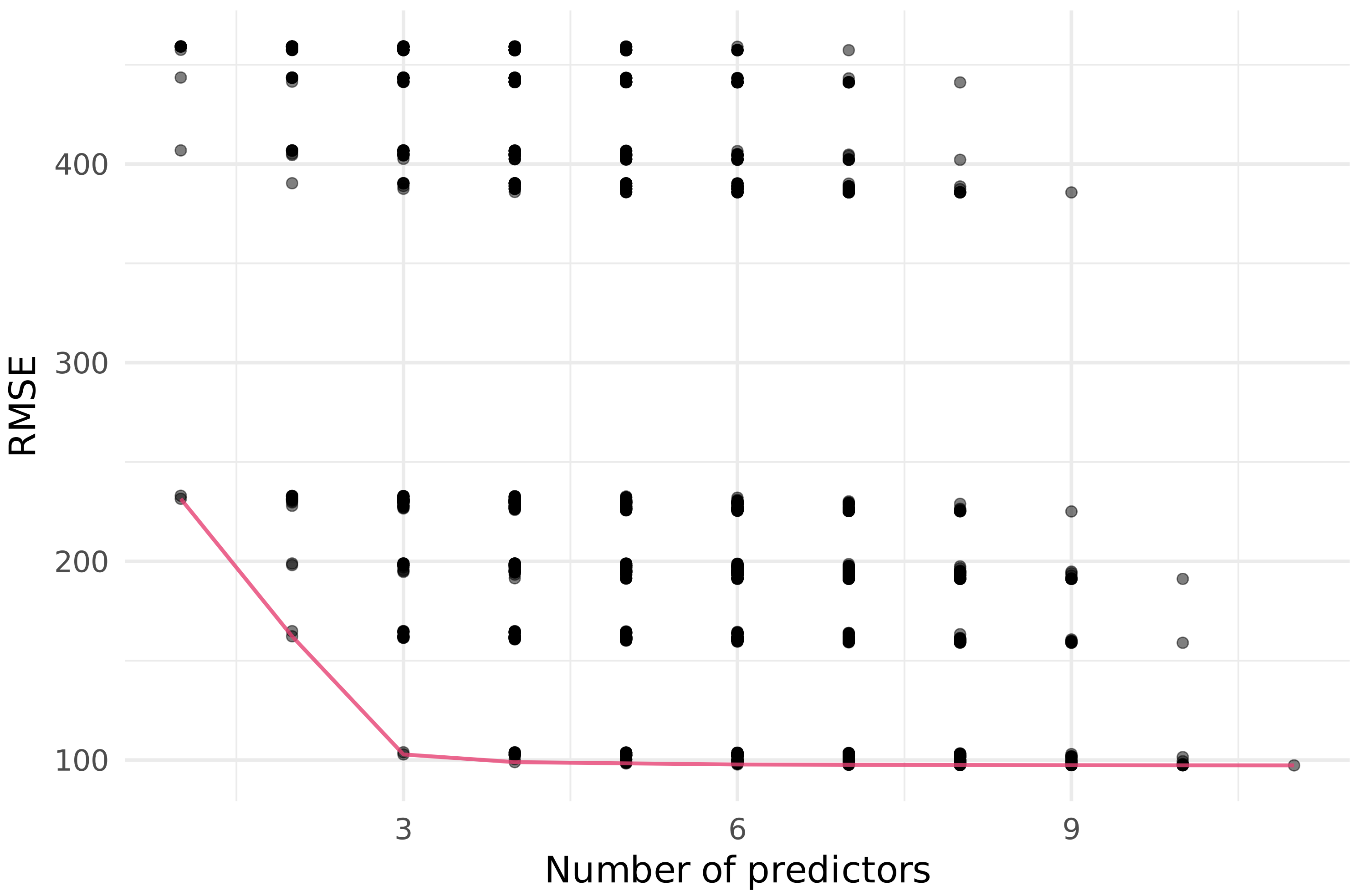
Model selection

Example dataset: `Credit`

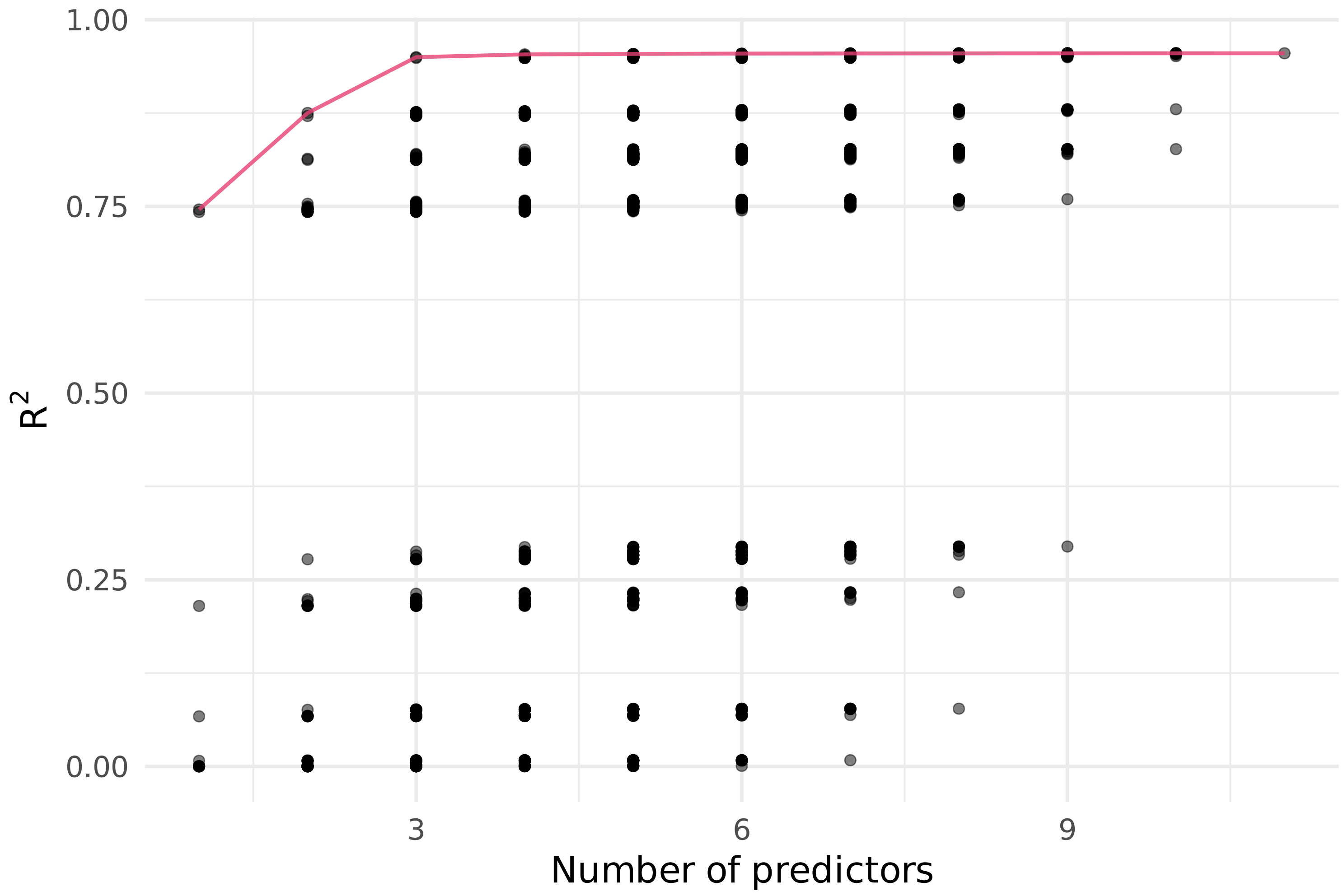
We need to pre-process the dataset before we can select a model...

income	limit	rating	cards	age	education	i_female	i_student	i_married	i_asian	i_african_american	balance
14.9	3606	283	2	34	11	0	0	1	0	0	333
106.0	6645	483	3	82	15	1	1	1	1	0	903
104.6	7075	514	4	71	11	0	0	0	1	0	580
148.9	9504	681	3	36	11	1	0	0	1	0	964
55.9	4897	357	2	68	16	0	0	1	0	0	331
80.2	8047	569	4	77	10	0	0	0	0	0	1151
21.0	3388	259	2	37	12	1	0	0	0	1	203

Now the dataset on has 400 observations on 12 variables (2,048 subsets).



400
300
200
100
RMSE
3
6
9
Number of predictors



1.00
0.75
0.50
0.25
0.00
R^2
3
6
9
Number of predictors

Model selection

Best subset selection

From here, you would

1. Estimate cross-validated error for each $\mathcal{M}_k$.

2. Choose the $\mathcal{M}_k$ that minimizes the CV error.

3. Train the chosen model on the full dataset.

Model selection

Best subset selection

Warnings

- Computationally intensive
- Selected models may not be "right" (squared terms with linear terms)
- You need to protect against overfitting when choosing across $\mathcal{M}_k$
- Also should worry about overfitting when p is "big"
- Dependent upon the variables (transformations) you provide

Benefits

- Comprehensive search across provided variables
- Resulting model—when estimated with OLS—has OLS properties
- Can be applied to other (non-OLS) estimators

Stepwise selection

Stepwise selection provides a less computational intensive alternative to best subset selection.

The basic idea behind **stepwise selection**

```
1. Start with an arbitrary model.
2. Try to find a "better" model by adding/removing variables.
3. Repeat.
4. Stop when you have the best model. (Or choose the best model.)
```

Stepwise selection

Stepwise selection provides a less computational intensive alternative to best subset selection.

The basic idea behind stepwise selection

```
1. Start with an arbitrary model.
2. Try to find a "better" model by adding/removing variables.
3. Repeat.
4. Stop when you have the best model. (Or choose the best model.)
```

The two most-common varieties of stepwise selection:

- Forward starts with only intercept ($\mathcal{M}_0$) and adds variables
- Backward starts with all variables ($\mathcal{M}_p$) and removes variables

Model selection

Forward stepwise selection

The process...

1. Start with a model with only an intercept (no predictors), $\mathcal{M}_0$.
2. For $k = 0, \ldots, p$:
    - Estimate a model for each of the remaining $p - k$ predictors, separately adding the predictors to model $\mathcal{M}_k$.
    - Define $\mathcal{M}_{k+1}$ as the "best" model of the $p - k$ models.
3. Select the "best" model from $\mathcal{M}_0, \ldots, \mathcal{M}_p$.

Forward stepwise selection

The process...

1. `Start with a model with only an intercept (no predictors),` $\mathcal{M}_0$.
2. `For` $k = 0, \ldots, p$:
   - `Estimate a model for each of the remaining` $p - k$ `predictors, separately adding the predictors to model` $\mathcal{M}_k$.
   - `Define` $\mathcal{M}_{k+1}$ `as the "best" model of the` $p - k$ `models.`
3. `Select the "best" model from` $\mathcal{M}_0, \ldots, \mathcal{M}_p$.

What do we mean by "best"?

`2:` *best* is often RSS or R^2.

`3:` *best* should be a cross-validated fit criterion.

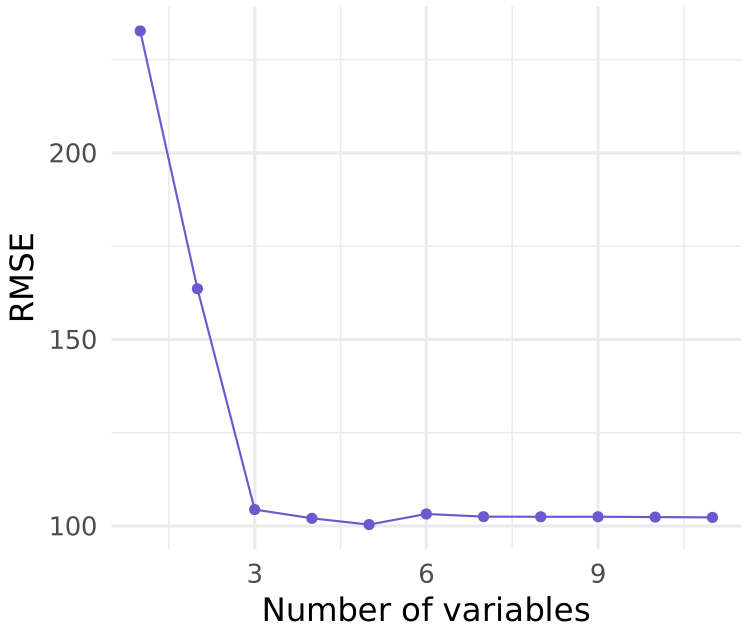
Forward stepwise selection with `caret` in R

```r
train_forward = train(
  y = credit_dt[["balance"]],
  x = credit_dt %>% dplyr::select(-balance),
  trControl = trainControl(method = "cv", number = 5),
  method = "leapForward",
  tuneGrid = expand.grid(nvmax = 1:11)
)
```

Forward stepwise selection with `caret` in R

```
train_forward = train(
  y = credit_dt[["balance"]],
  x = credit_dt %>% dplyr::select(-balance),
  trControl = trainControl(method = "cv", number = 5),
  method = "leapForward",
  tuneGrid = expand.grid(nvmax = 1:11)
)
```

N vars	RMSE	R2	MAE
1	232.71	0.750	175.9
2	163.60	0.873	122.2
3	104.41	0.950	84.6
4	102.05	0.953	82.3
5	100.37	0.955	80.0
6	103.22	0.953	82.5
7	102.50	0.953	82.1
8	102.47	0.953	82.0
9	102.47	0.953	82.3
10	102.38	0.953	82.2



Backward stepwise selection

The process for **backward stepwise selection** is quite similar...

Backward stepwise selection

The process for **backward stepwise selection** is quite similar...

1. Start with a model that includes all p predictors: $\mathcal{M}_p$.
2. For $k = p, p-1, \ldots, 1$:
   - Estimate k models, where each model removes exactly one of the k predictors from $\mathcal{M}_k$.
   - Define $\mathcal{M}_{k-1}$ as the "best" of the k models.
3. Select the "best" model from $\mathcal{M}_0, \ldots, \mathcal{M}_p$.

Model selection

Backward stepwise selection

The process for **backward stepwise selection** is quite similar...

1. Start with a model that includes all p predictors: $\mathcal{M}_p$.

2. For $k = p, p-1, \ldots, 1$:

    - Estimate k models, where each model removes exactly one of the k predictors from $\mathcal{M}_k$.

    - Define $\mathcal{M}_{k-1}$ as the "best" of the k models.

3. Select the "best" model from $\mathcal{M}_0, \ldots, \mathcal{M}_p$.

What do we mean by "best"?

2: *best* is often RSS or R^2.

3: *best* should be a cross-validated fit criterion.

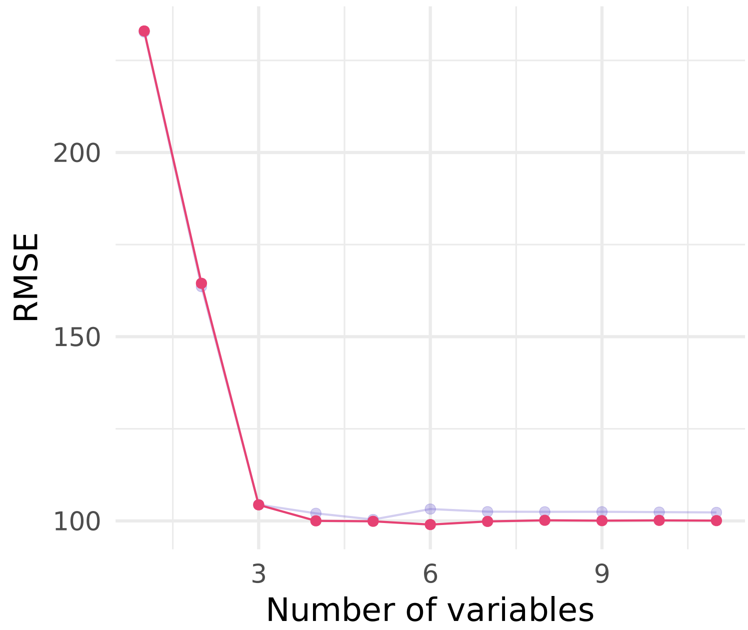
Backward stepwise selection with `caret` in R

```
train_backward = train(
  y = credit_dt[["balance"]],
  x = credit_dt %>% dplyr::select(-balance),
  trControl = trainControl(method = "cv", number = 5),
  method = "leapBackward",
  tuneGrid = expand.grid(nvmax = 1:11)
)
```

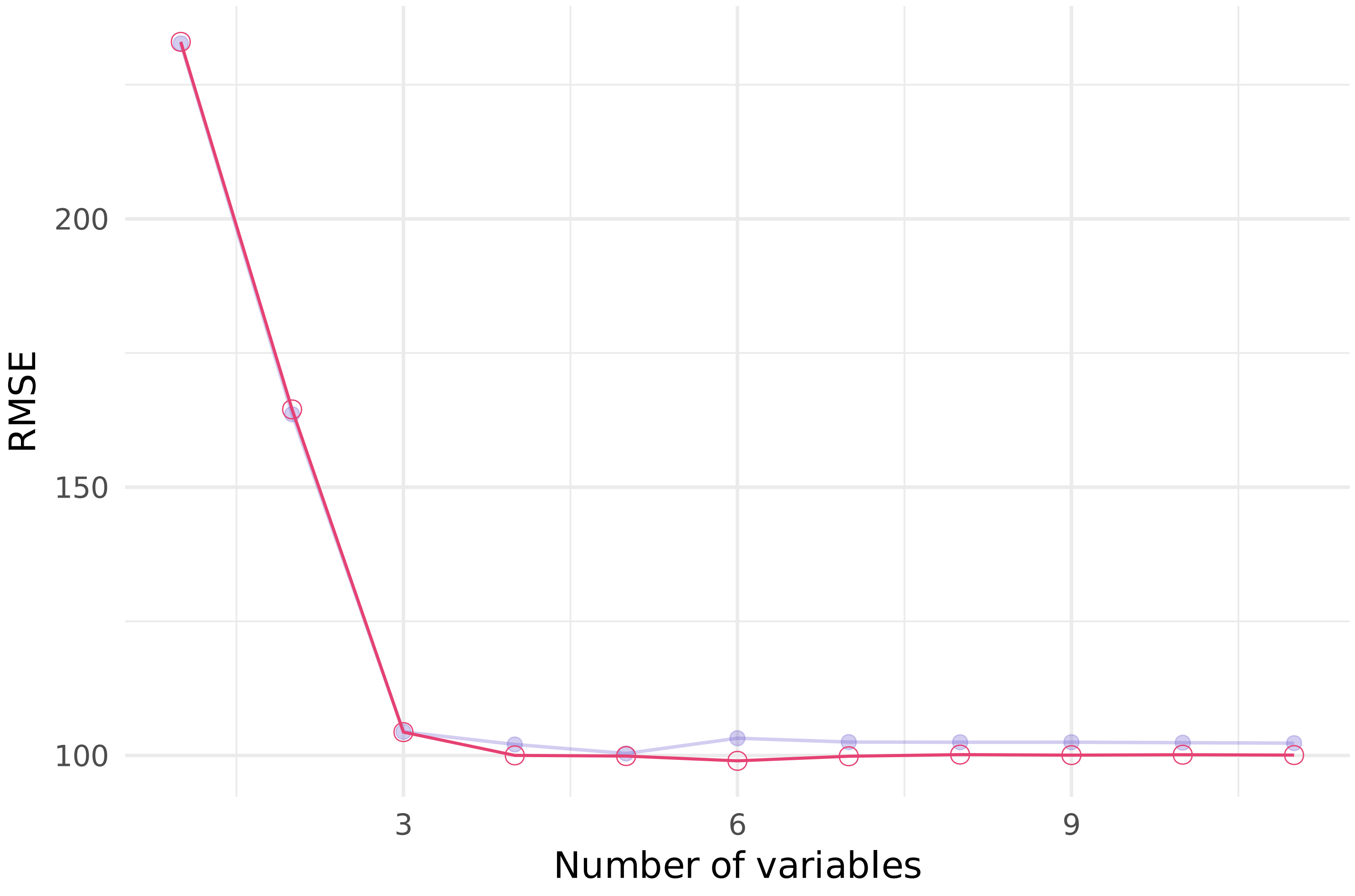
Backward stepwise selection with `caret` in R

```r
train_backward = train(
  y = credit_dt[["balance"]],
  x = credit_dt %>% dplyr::select(-balance),
  trControl = trainControl(method = "cv", number = 5),
  method = "leapBackward",
  tuneGrid = expand.grid(nvmax = 1:11)
)
```

N vars	RMSE	R2	MAE
1	232.99	0.743	178.0
2	164.49	0.872	124.9
3	104.35	0.949	83.9
4	100.01	0.953	79.5
5	99.89	0.953	79.4
6	99.00	0.954	79.1
7	99.85	0.953	79.8
8	100.16	0.953	80.1
9	100.06	0.953	80.2
10	100.14	0.953	80.3



Note: **forward** and **backward** step. selection can choose different models.



Stepwise selection

Notes on stepwise selection

- **Less computationally intensive** (relative to best subset selection)
    - With $p = 20$, BSS fits 1,048,576 models.
    - With $p = 20$, foward/backward selection fits 211 models.
- There is **no guarantee** that stepwise selection finds the best model.
- ***Best*** is defined by your fit criterion (as always).
- Again, **cross validation is key** to avoiding overfitting.

Model selection

Criteria

Which model you choose is a function of **how you define "best"**.

Model selection

Criteria

Which model you choose is a function of **how you define "best"**.

And we have many options...

Model selection

Criteria

Which model you choose is a function of **how you define "best"**.

And we have many options... We've seen RSS, (R)MSE, RSE, MA, R^2, Adj. R^2.

Model selection

Criteria

Which model you choose is a function of **how you define "best"**.

And we have many options... We've seen RSS, (R)MSE, RSE, MA, R^2, Adj. R^2.

Of course, there's more. Each **penalizes** the d predictors differently.

$$C_p = \frac{1}{n}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

$$\text{BIC} = \frac{1}{n\hat{\sigma}^2}\left(\text{RSS} + \log(n)d\hat{\sigma}^2\right)$$

Model selection

Criteria

> C_p, AIC, and BIC all have rigorous theoretical justifications… the adjusted R^2 is not as well motivated in statistical theory

ISL, p. 213

In general, we will stick with cross-validated criteria, but you still need to choose a selection criterion.

Sources

These notes draw upon

- An Introduction to Statistical Learning (*ISL*)
  James, Witten, Hastie, and Tibshirani

Table of contents