Section 5.1 Model Selection

Jared S. Murray
The University of Texas at Austin
McCombs School of Business

Model Building Process

When building a model, remember: simplicity is your friend.

Every additional parameter represents a cost! Smaller models are easier to interpret and have fewer unknown parameters to be estimated (i.e., usually smaller standard errors).

The first step of every model building exercise is the selection of the the universe of X's to be potentially used. This task is entirely solved through your experience and context specific knowledge...

- Think carefully about the problem
- Consult subject matter research and experts
- Avoid including "too many" variables
- Consider transformations of the original variables – always make sure your assumptions appear to be valid!

Selecting a regression model

In a linear regression context, picking a model amounts to deciding which **terms** (X's) to include.

This is often called "variable selection". I don't like that terminology (for example, age is one *variable*, but age and age^2 are two *terms* or X's we might include) but I'm in the minority.

With a universe of possible terms in hand, the goal now is to select the model. Why not include all the terms?

Selecting a regression model

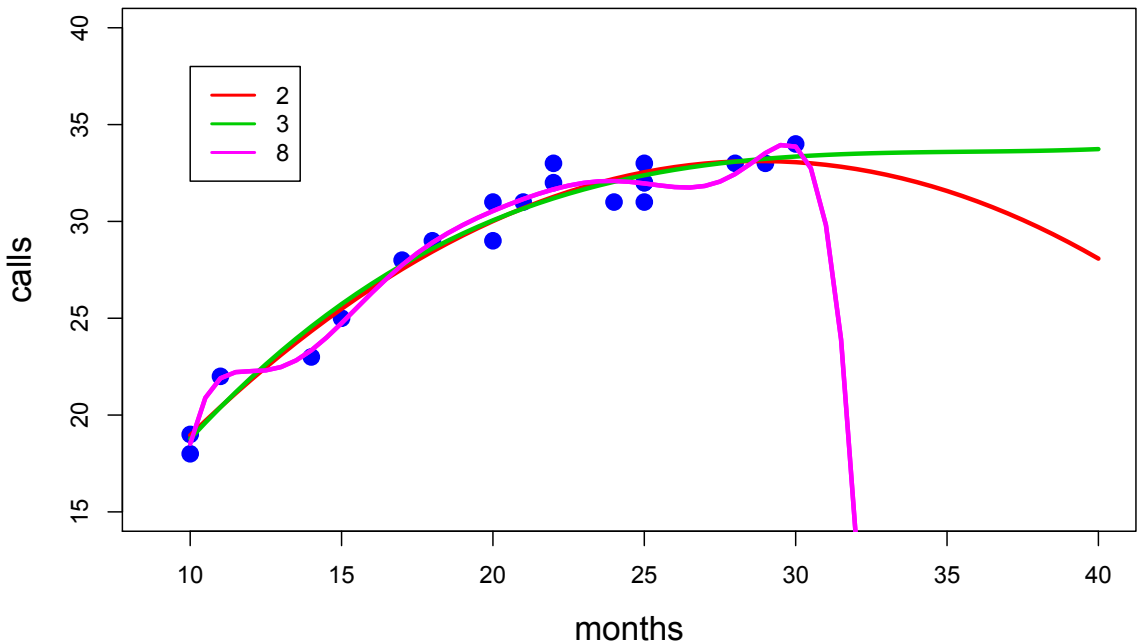
Big models tend to over-fit and find features that are specific to the data in hand... i.e., not generalizable relationships.

The result is bad prediction and bad science!

In addition, bigger models have more parameters and potentially more uncertainty about everything we are trying to learn...

Overfitting

Remember the effect of adding more polynomial terms in the telemarketing example?



The curves fit the sample better and better, but eventually generalize worse.

Adding Variables Can Increase Uncertainty

Compare the standard error of b_1 when fitting

$$nbeers = \beta_0 + \beta_1 weight + \beta_2 height + \epsilon$$

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.18709   10.76821  -1.039 0.304167
## weight        0.08530    0.02381   3.582 0.000806 ***
## height        0.07751    0.19598   0.396 0.694254
```

Versus

$$nbeers = \beta_0 + \beta_1 weight + \epsilon$$

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.02070    2.21329  -3.172  0.00264 **
## weight       0.09289    0.01399   6.642  2.6e-08 ***
```

Nearly 2x as large! And β_1 is (a little) harder to interpret.

What Makes a "Good" Model?

It depends a little on context, but a good model should:

- Only make assumptions that are reasonable (and not contradicted by the data)
- Contain enough variables to support the analysis
  - All potential "lurking variables" when making a causal claim
  - Important covariates when making predictions (for better predictions and smaller prediction intervals)
- Avoid including too many variables and/or complicated transformations.

We have graphical checks for assumptions, and subject knowledge about important and lurking variables. How can we assess predictive ability?

Out-of-Sample Prediction

How do we evaluate a forecasting model? Make predictions!

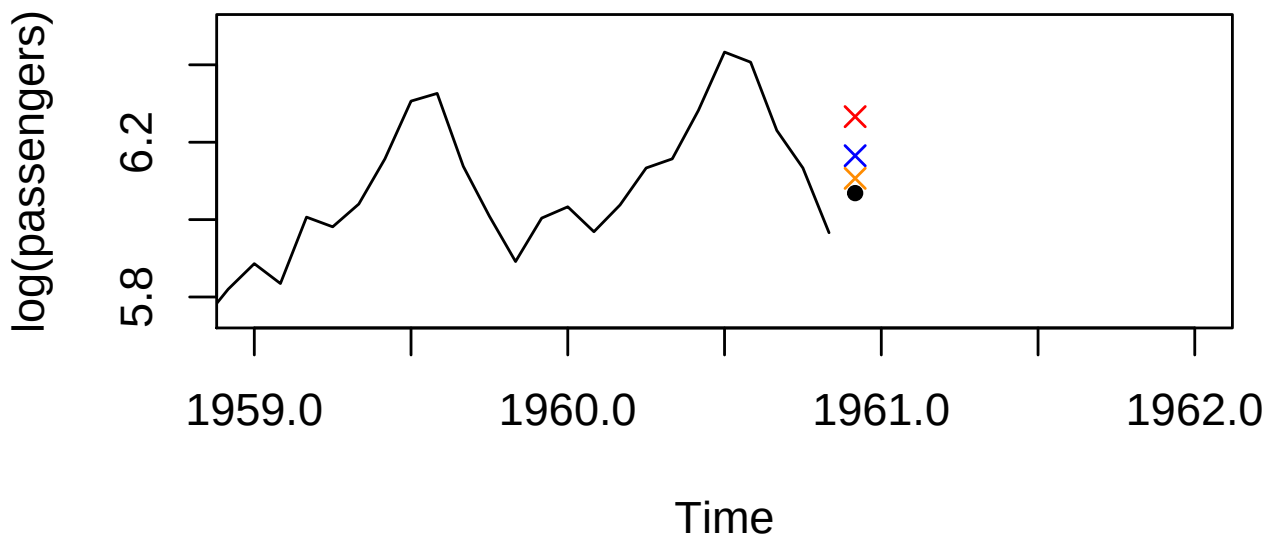
Basic Idea: We want to use the model to forecast outcomes for observations we have not seen before.

- Use the data to create a prediction problem **where we know the "right" answer**
- See how our candidate models perform.

We'll use most of the data for training the model,
and the left over part for validating the model.

Out-of-Sample Prediction

Let's revisit the airline data. We fit without the very last observation, and then generate a forecast for the known value under 3 different models (the x's; the true value is the black dot)

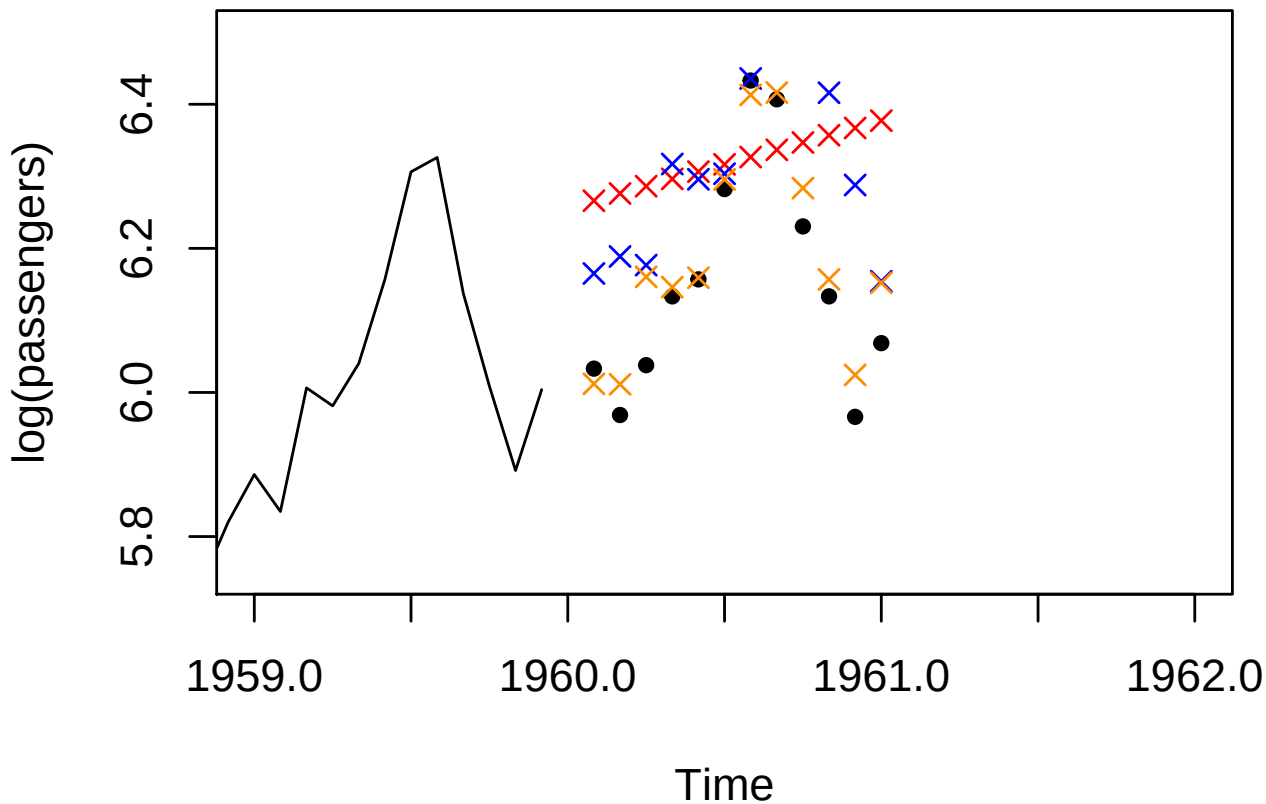


Which model do you prefer? Do you trust your choice?

Out-of-Sample Prediction

With one data point, a model might have a good prediction just by chance! Remember, the next data point includes a random error.

Let's repeat, but forecast a whole year:



Which model do you prefer? Do you trust your choice now?

Out-of-Sample Prediction

In a cross-validation scheme, you fit a bunch of models to most of the data (training sample) and choose the model that performed the best on the rest (left-out sample).

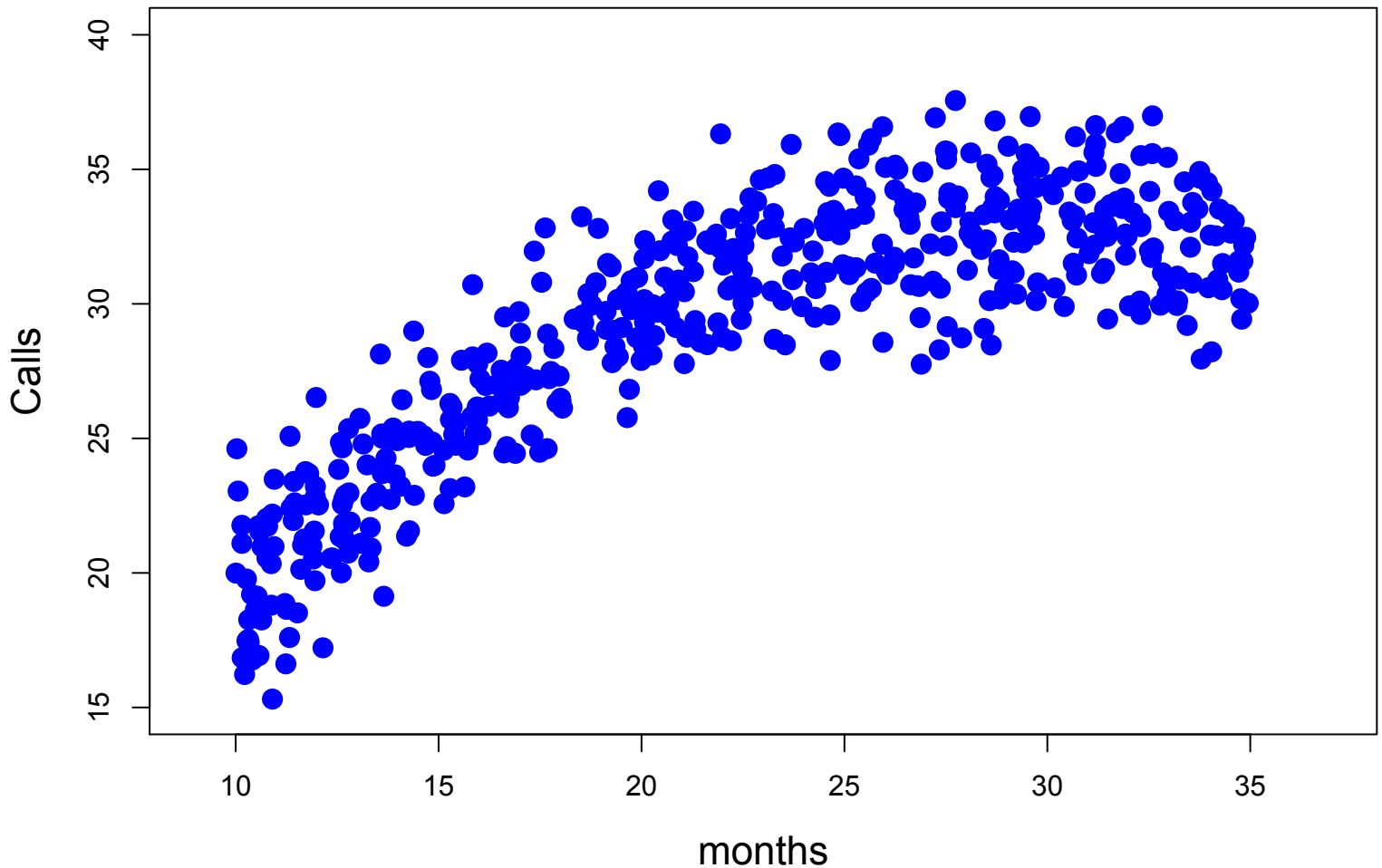
- Fit the model on the training data
- Use the model to predict $\hat{Y}_j$ values for all of the N_{LO} left-out data points
- Calculate the Mean Square Error for these predictions

$$MSE = \frac{1}{N_{LO}} \sum_{j=1}^{N_{LO}} (Y_j - \hat{Y}_j)^2$$

(For the airline data, we looked at $N_{LO} = 1$ and $N_{LO} = 12$)

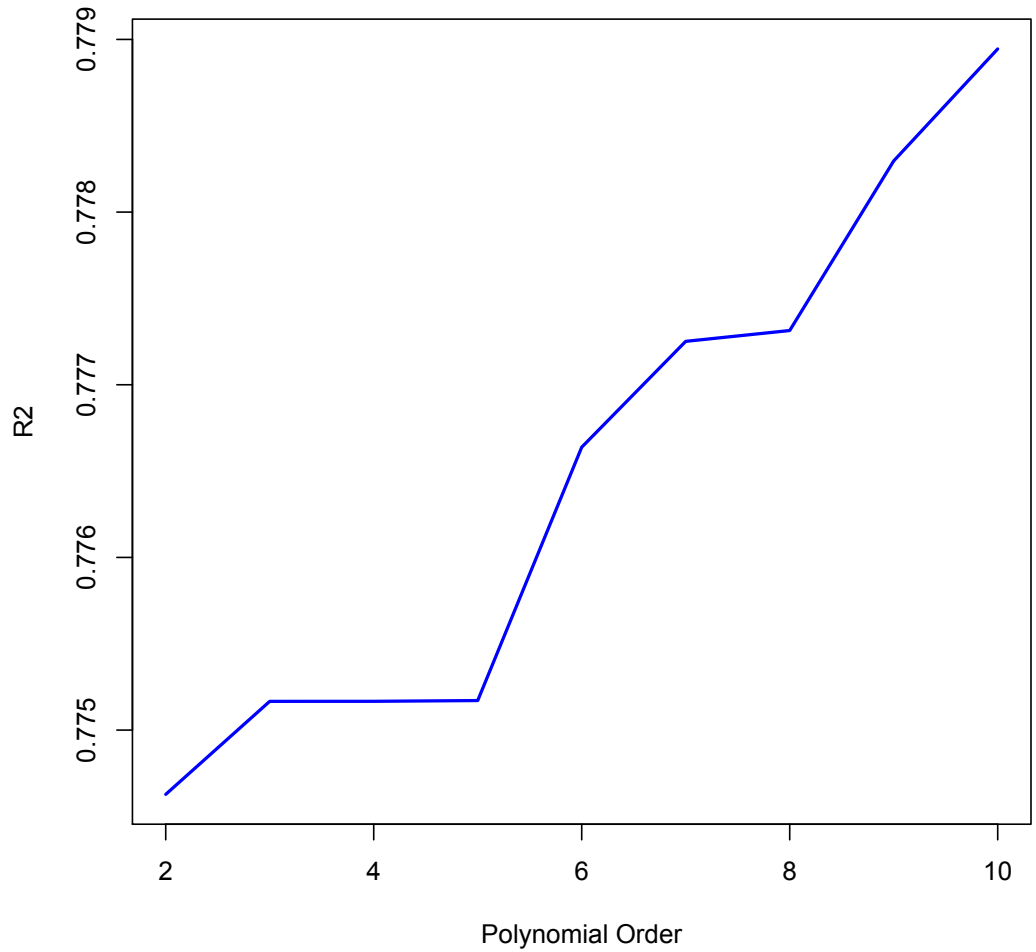
Example

To illustrate the potential problems of "over-fitting" the data, let's look again at the Telemarketing example... let's look at multiple polynomial terms...



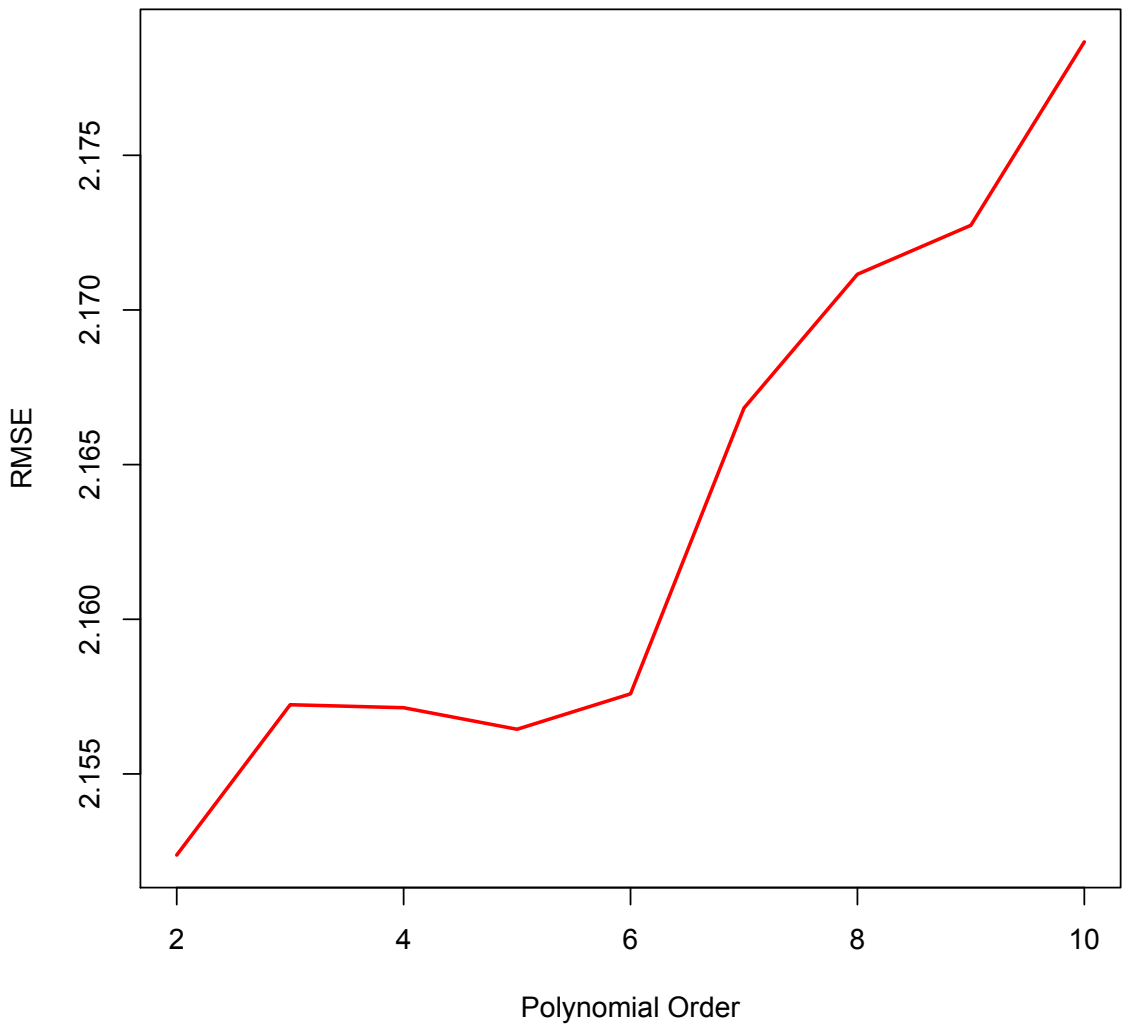
Example

Let's evaluate the fit of each model by their R^2, computed with the traning data - higher is (supposed to be) better.



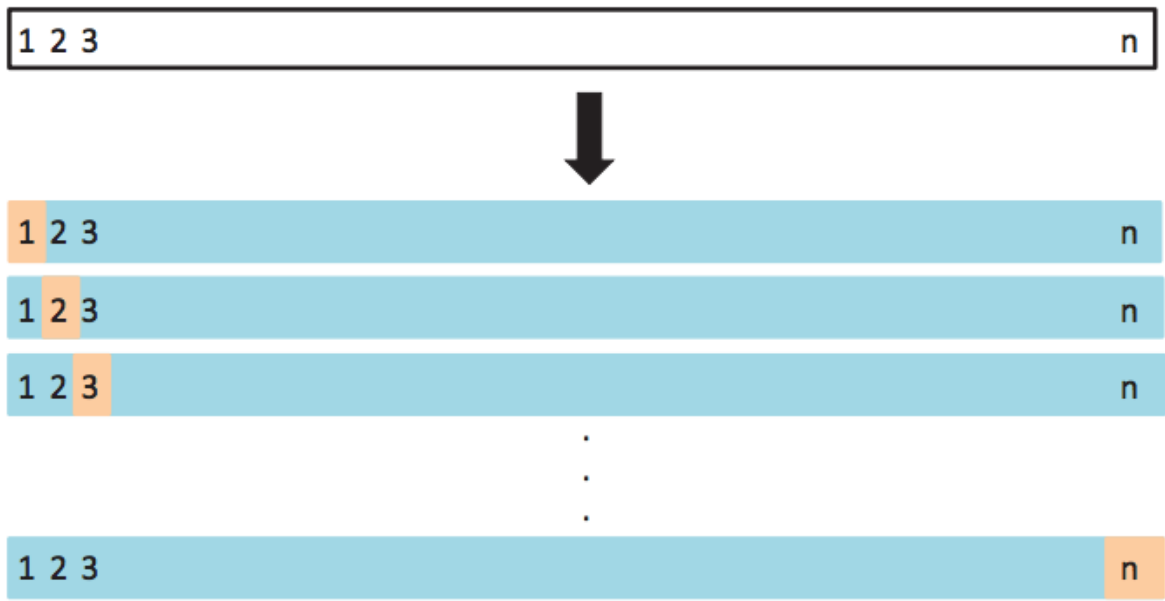
Example

How about the MSE on the left-out data? (Lower is better).



Leave One Out Cross Validation

In the last example, we randomly split the data into training and left-out data. Alternatively, we could do CV with each observation as the left-out data ($N_{LO} = 1$) and **average** the prediction errors



(Image from *Introduction to Statistical Learning with R*)

Leave One Out Cross Validation

This is **leave one out cross validation**:

For each observation i,

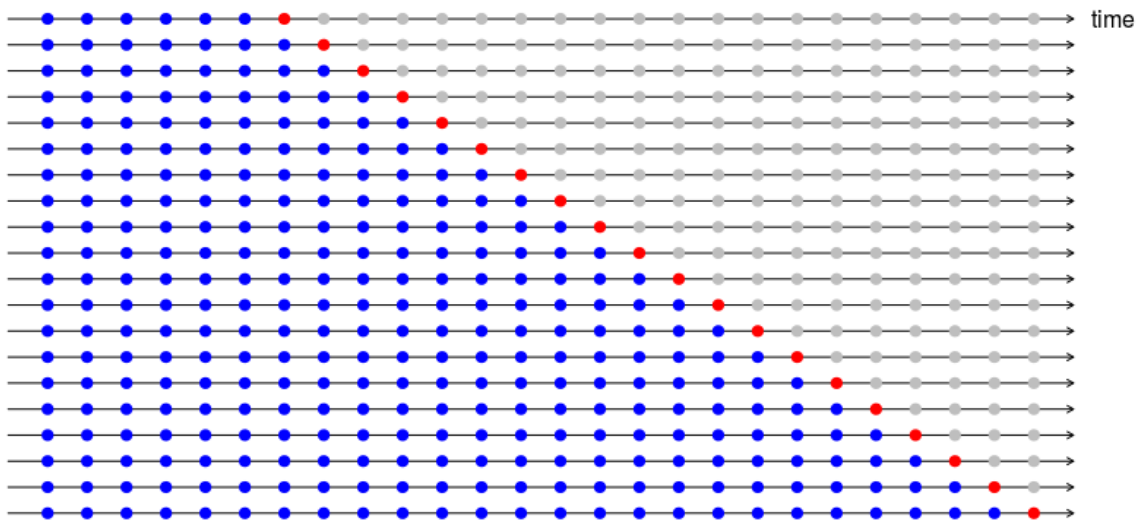
1. Fit the model using data points $1, 2, 3, \ldots, i-1, i+1\, dots, n$
2. Predict y_i with $\hat{y}_i^{(i)}$ (The (i) means that point i isn't used to generate the predicted value)
3. Save the out of sample prediction error $pe_i = y_i - \hat{y}_i^{(i)}$

Estimate the forecast error by

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

Leave One Out Cross Validation

For time series, things are a little more complicated. But we can use the first $j-1$ points (blue) to fit a model and compute the prediction error for y_j (red), ignoring the rest of the "future" $y_{j+1}, \ldots, y_T$ (gray).



Then we average the prediction errors just like before. This is implemented in the `tsCV` function in the forecast package.

AIC for Model Selection

Another way to evaluate a model is to use Information Criteria metrics which attempt to balance predictive fit and model complexity (number of covariates p). The Akakie Information Criterion (AIC)is one:

$$AIC \approx n \log(s^2) + 2p$$

The Bayesian Information Criterion (BIC) is another:

$$BIC \approx n \log(s^2) + p \log(n)$$

For either criterion lower is better.

AIC is about the same as leave-one-out CV in large samples (but faster to compute); BIC selects more parsimonious models.

Two Criteria I Don't Recommend

I don't recommend using R^2 (remember, it always goes up!) or *adjusted* R^2:

$$R^2_{adj} = 1 - \frac{s^2}{var(y)} = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)}$$

as it tends to overfit.

I also don't generally recommend blindly dropping variables when their $p-$value is below 0.05 (or 0.1 or some other arbitrary threshold)

Computing Model Selection Criteria

For regression models, you can compute the cross validation error, AIC, BIC, and adjusted R^2 using the forecast package:

```
CV(beerfit_hw)

##           CV          AIC         AICc          BIC        AdjR2
##    8.5337525 107.3087751 108.1976640 114.9568671    0.4585626

CV(beerfit_w)

##           CV          AIC         AICc          BIC        AdjR2
##    8.008227 105.474914 105.996653 111.210983    0.468078
```

Searching For a "Good" Model

What if we have many possible models to compare? With p possible X's, there are 2^p possible models.

For $p = 20$ that's over a million possibilities!

If we can't look at each possible model, we have to do some searching...

Stepwise Regression

One approach to build a regression model (semi-)automatically, step-by-step, is "stepwise regression" There are 3 options:

- Forward: adds one variable at the time until no remaining variable makes a significant contribution (or meet a certain criteria... could be out of sample prediction)
- Backwards: starts will all possible variables and removes one at the time until further deletions would do more harm them good
- Stepwise: just like the forward procedure but allows for deletions at each step

Auto MPG Example

Initial model:

$$MPG = \beta_0 + \beta_1 weight + \beta_2 horsepower + \beta_3 displacement + \beta_4 acceleration + \beta_5 cylinders + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \beta_9 origin2 + \epsilon$$

Backward elemination:

- Step1: delete *origin2*
- Step 2: delete *acceleration*
- Step 3: delete *cylinders*

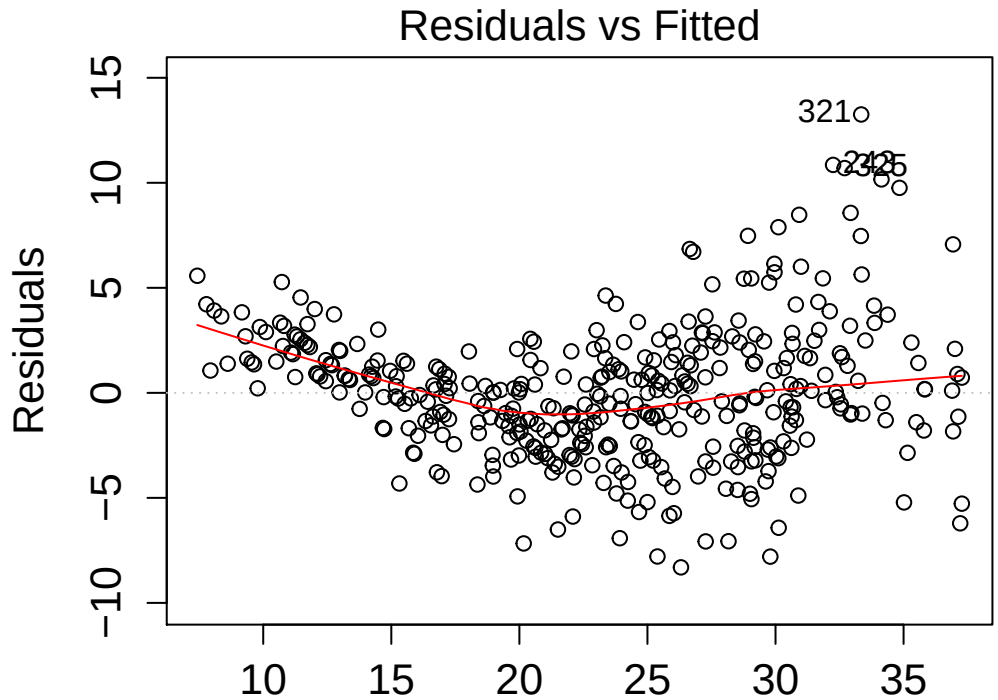
Selected model:

$$MPG = \beta_0 + \beta_1 weight + \beta_2 horsepower + \beta_3 displacement + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \epsilon$$

Auto MPG Example

Residual plot for:

$$MPG = \beta_0 + \beta_1 weight + \beta_2 horsepower + \beta_3 displacement + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \epsilon$$



Does this plot look good?

Auto MPG Example

New initial model, after diagnostics:

$$\log(MPG) = \beta_0 + \beta_1 + \log(weight) + \beta_2 \log(horsepower) + \beta_3 \log(displacement) + \beta_4 \log(acceleration) + \beta_5 cylinders + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \beta_9 origin2 + \epsilon$$

Backward elemination:

- Step1: delete $\log(displacement)$
- Step 2: delete $origin2$

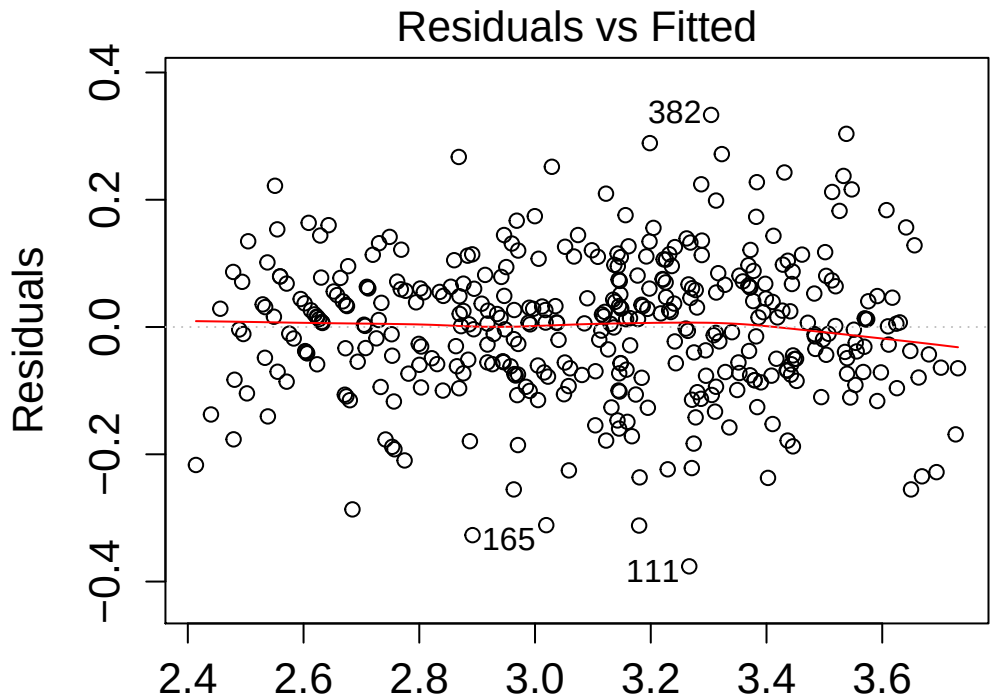
Selected model:

$$\log(MPG) = \beta_0 + \beta_1 + \log(weight) + \beta_2 \log(horsepower) + \beta_4 \log(acceleration) + \beta_5 cylinders + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \epsilon$$

Auto MPG Example

Residual plot for:

$$\log(MPG) = \beta_0 + \beta_1 + \log(weight) + \beta_2 \log(horsepower) +$$
$$\beta_4 \log(acceleration) + \beta_5 cylinders + \beta_6 year + \beta_7 year^2 + \beta_8 origin1 + \epsilon$$



How about this one?

Diagnose the residuals!

Auto MPG Example

In this example, forward/backward/stepwise give the same answer using AIC. This isn't always true! There is no button to push to get the "right" answer.

Also note that this process is hardly automatic! Fit, diagnose, transform, refit, diagnose... Model building is iterative.

Note also that after this process, **the p-values, t-statistics, and standard errors are no longer valid.** To get valid inference, we need to select a model using one part of our data (e.g. half) and compute SE's, etc on another "validation" dataset (or the remaining half of the original dataset).

One informal but very useful idea to put it all together...

I like to build models from the bottom, up...

- Set aside a set of points to be your validation set (if dataset large enough)
- Working on the training data, add one variable at the time deciding which one to add based on some criteria:
  1. CV, AIC, BIC, etc.
  2. Adjusted R^2 is OK as a heuristic
  3. p-value of the new variable – does it seem to add anything beyond what's already in the model?
- At every step, carefully analyze the output and check the residuals!
- Stop when no additional variable produces a "significant" improvement
- Always make sure you understand what the model is doing in the specific context of your problem