UNIVERSITY OF WISCONSIN
DATA SCIENCE

DS 740

Data Mining

Multiple Linear Regression

Model Comparison and Best Subset Selection

Important note: Transcripts are **not** substitutes for textbook assignments.

Learning Objectives

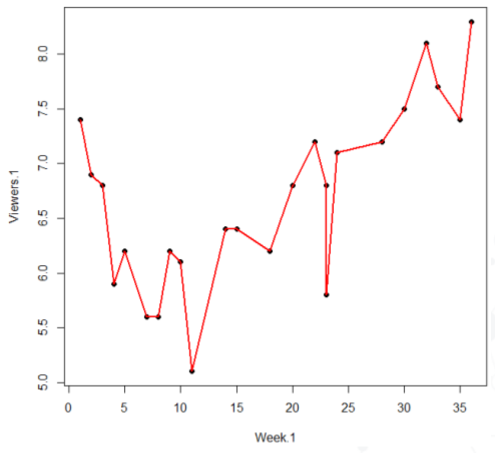
By the end of this lesson, you will be able to:

- Explain what the Variance Inflation Factor measures.
- Compare models based on adjusted R^2, Mallows' C_p, AIC, and BIC.
- Use Best Subsets Selection to choose an optimal model.
- Use the One-standard-error-rule to choose an optimal model.

Tradeoff Between Bias and Variance in Linear Regression

More predictors ➡ Lower bias, higher variance

Interpretability difficult



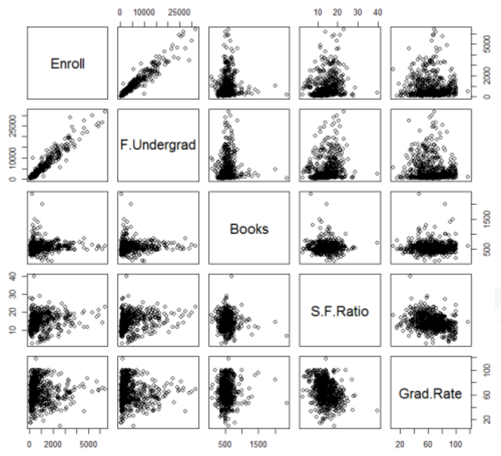
Using too many predictor variables in your linear regression results in a model with very low bias but very high variance. This means that the model is great at fitting your training data but terrible at predicting new data points. And it's also difficult to interpret.

Multicollinearity: A Check to Reduce Variables

When 2 or more predictor variables are highly correlated.

Inflates standard error for estimated regression coefficients ➡ larger p-values for individual coefficients.

```
pairs(myColleges)
```



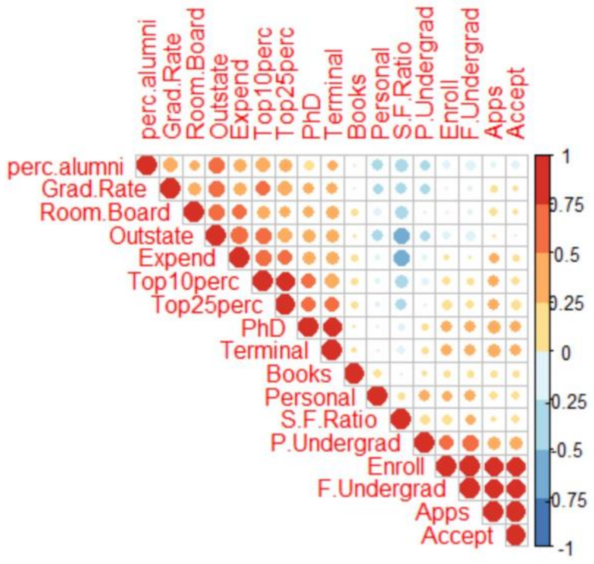
One simple step to avoid having too many predictor variables is to check for collinearity or multi-collinearity, which refers to when two or more predictor variables are highly correlated. This can inflate the standard error for your estimated regression coefficients, which results in larger P values for testing whether your individual coefficients are different from 0.

One way to check for collinearity is by looking at a matrix of scatter plots using the pairs function in R. Exam this matrix for any sign of a linear relationship between two of the variables. For example, in this data set, the total enrollment at colleges is highly correlated with the number of full-time undergraduates.

Notes:

library(ISLR)
my_colleges = College[-96,] # Remove an observation with an unrealistic graduation rate (> 100%)
my_colleges = my_colleges[,c(4, 7,11, 15,18)] # Examine a subset of variables so the scatterplots aren't too small to read
pairs(my_colleges)

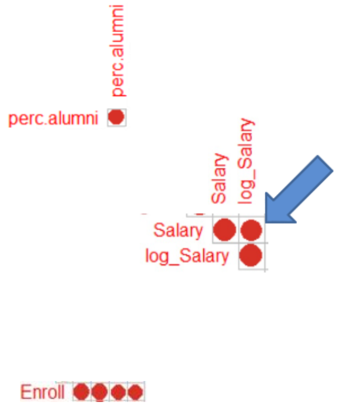
Correlation plots



A correlation plot is another good way to quickly check for correlations among variables. Look for predictor variables whose correlations are close to either positive 1 or negative 1.

Correlations that are not a cause for concern

- A variable always has correlation 1 with itself.
- It’s expected to have high correlation between a variable and its transformation.
  - Often, you were going to remove the original variable from the model anyway.
- Strong correlations (positive or negative) with the response variable are good.
  - Those predictors will help us make good predictions.



Some correlations that you see in the correlation graph are not a cause for concern. For example, a variable always has correlation 1 with itself. So you don't need to worry about the large number of red circles you see on the main diagonal of the correlation graph.

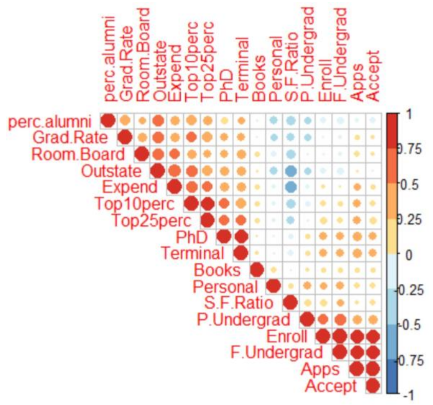
It's also expected to have high correlation between a variable and its transformation. For example, if you were using a variable salary and decided to log transform it to reduce its right skew, then we expect to have high correlation between those two variables. This isn't really a surprise, so it's not really a concern. Often, you were going to remove the original salary variable from the model anyway because of its skew.

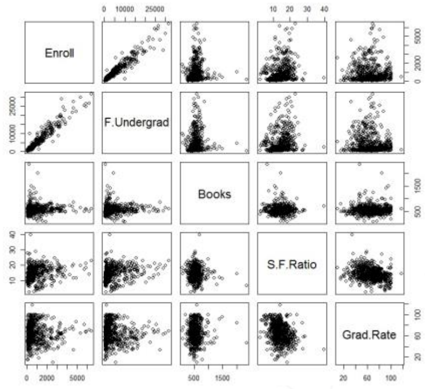
Finally, strong correlations either positive or negative with the response variable are good because those predictors will help us make good predictions. So for example, if we're trying to predict enrollment, then we should be happy to see a large number of predictor variables with strong correlations with enrollment.

Multicollinearity: Graph Limitations

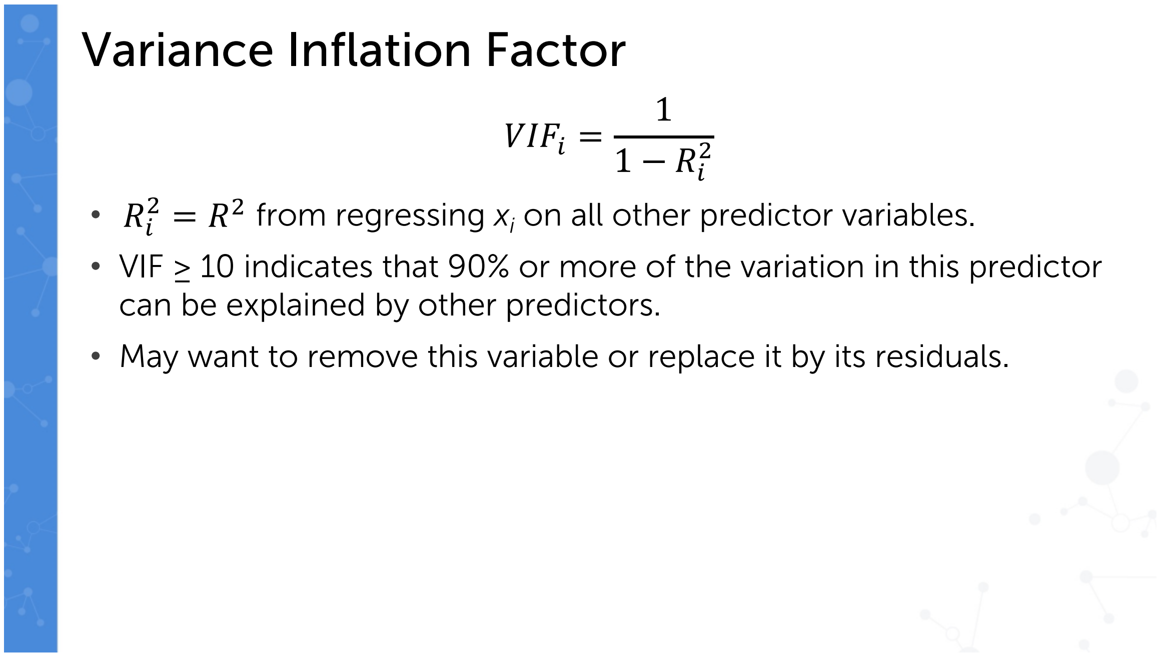
2 variables: easy to see

3+ variables: more difficult





The correlation plot and matrix of scatter plots can tell you about the collinearity between two variables, but it can't tell you about multicollinearity among three or more variables. For example, the total enrollment at a college might be only weakly correlated with the number of data science majors. But if you start to include the number of biology majors, business majors, psychology majors, and English majors, then the association starts to increase



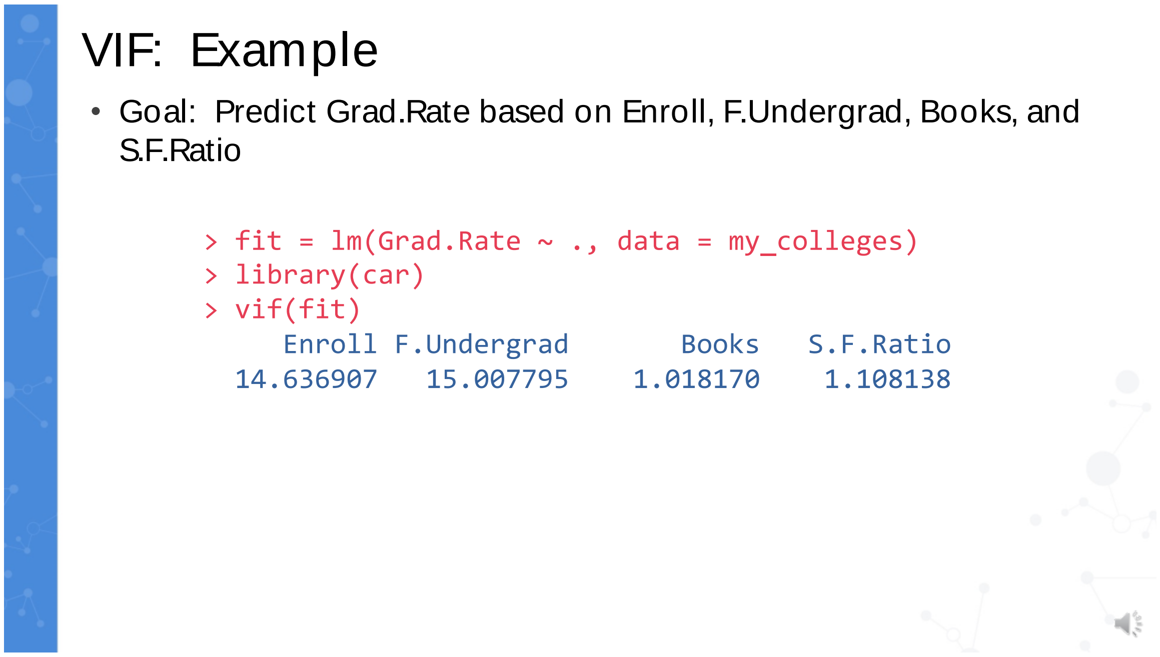
Variance Inflation Factor

$$VIF_i = \frac{1}{1 - R_i^2}$$

- $R_i^2 = R^2$ from regressing x_i on all other predictor variables.
- VIF $\geq$ 10 indicates that 90% or more of the variation in this predictor can be explained by other predictors.
- May want to remove this variable or replace it by its residuals.

To check for multi-collinearity, we can use the variance inflation factor, which is 1 over 1 minus the coefficient of determination that we get from regressing a particular predictor variable as a function of all the other predictor variables, ignoring the response variable.

A variance inflation factor of 10 or more indicates that 90% or more of the variation in this predictor variable can be explained by the other predictor variables. So in this case, you may want to remove that predictor variable or replace it by its residuals from that regression.



VIF: Example

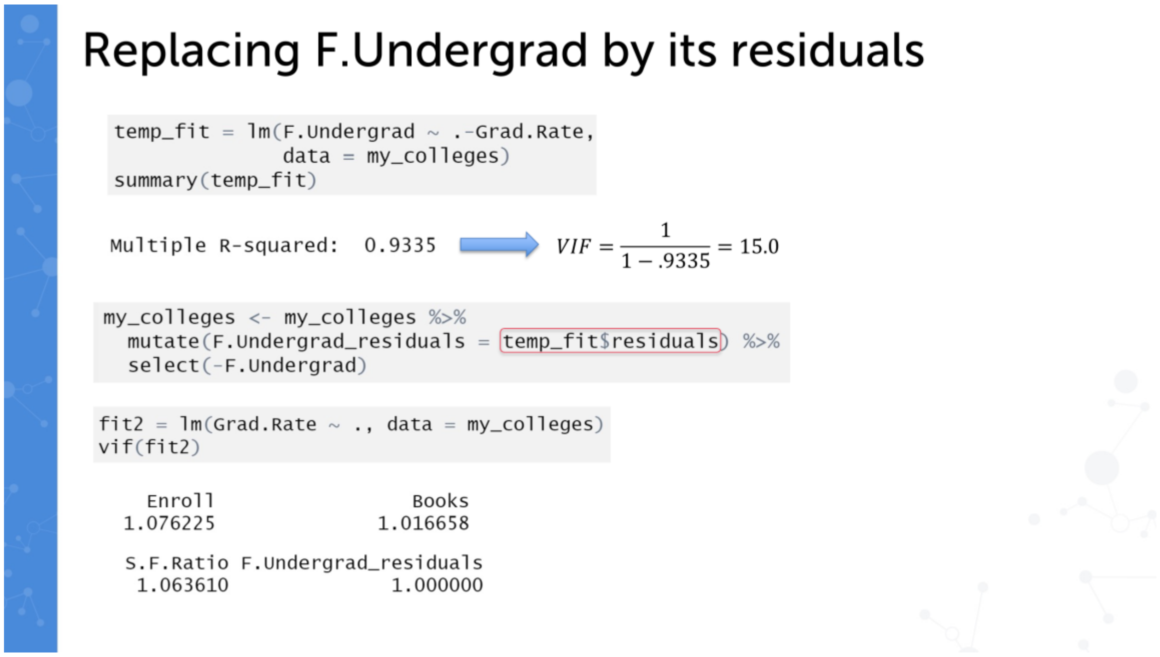
- Goal: Predict Grad.Rate based on Enroll, F.Undergrad, Books, and S.F.Ratio

```
> fit = lm(Grad.Rate ~ ., data = my_colleges)
> library(car)
> vif(fit)
   Enroll F.Undergrad       Books   S.F.Ratio
14.636907   15.007795    1.018170    1.108138
```

For example, suppose our goal was to predict graduation rate based on a college's enrollment, number of full-time undergraduates, cost of books, and student-faculty ratio. We would start by using a linear model to predict graduation rate like usual, and then we would use the VIF function applied to the output from that linear model.

In this case, we see that both enrollment and the number of full-time undergraduates have very high variance inflation factors. That makes sense because we expect that the number of full-time undergraduates is closely correlated with a school's enrollment. This means that enrollment and number of full-time undergraduates are giving us mostly the same information. So we could remove one of them from our model.

I'd probably remove number of full-time undergraduates because it has a slightly higher variance inflation factor. This would decrease our collinearity, which would tend to decrease the standard errors on our coefficient estimates in our linear model. And that in turn is likely to give us smaller p values. However, if we remove the number of full-time undergraduates from our model, we are removing information only some of which is duplicated by including the variable enrollment.



Replacing F.Undergrad by its residuals

```
temp_fit = lm(F.Undergrad ~ .-Grad.Rate,
                data = my_colleges)
summary(temp_fit)
```

```
Multiple R-squared:  0.9335
```

$$VIF = \frac{1}{1 - .9335} = 15.0$$

```
my_colleges <- my_colleges %>%
  mutate(F.Undergrad_residuals = temp_fit$residuals) %>%
  select(-F.Undergrad)
```

```
fit2 = lm(Grad.Rate ~ ., data = my_colleges)
vif(fit2)
```

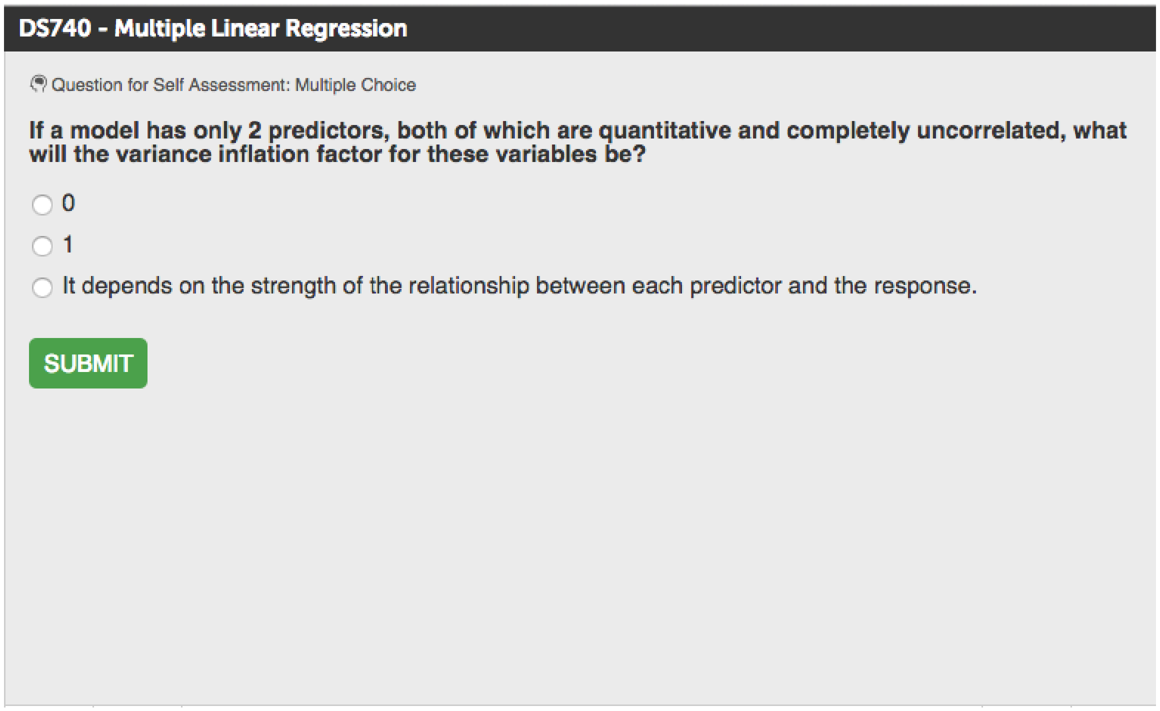
```
   Enroll                 Books
 1.076225              1.016658

  S.F.Ratio F.Undergrad_residuals
   1.063610              1.000000
```

An alternative is to regress the number of full-time undergraduates on the other predictor variables and then replace it by its residuals. What this does is it extracts all of the information that's in the number of full-time undergraduates that's not already contained in the other predictor variables.

The way to do this is by creating a temporary linear model where the number of full-time undergraduates is the response variable and using all the other predictor variables as the predictor variables, so everything except for graduation rate, which was the original response variable.

If we look at the summary of this temporary linear regression model, we find that it has a multiple R squared of 0.9335, which corresponds to a variance inflation factor of 15, just as we saw by using the VIF function. We can then take our temporary regression model dollar sign residuals and store them as a new column in our data set and remove the original number of full-time undergraduates. Then we can proceed with fitting our linear regression model for graduation rate. You'll notice that the variance inflation factors of the resulting model are much lower.

Self-Assessment Question 1



DS740 - Multiple Linear Regression

Question for Self Assessment: Multiple Choice

If a model has only 2 predictors, both of which are quantitative and completely uncorrelated, what will the variance inflation factor for these variables be?

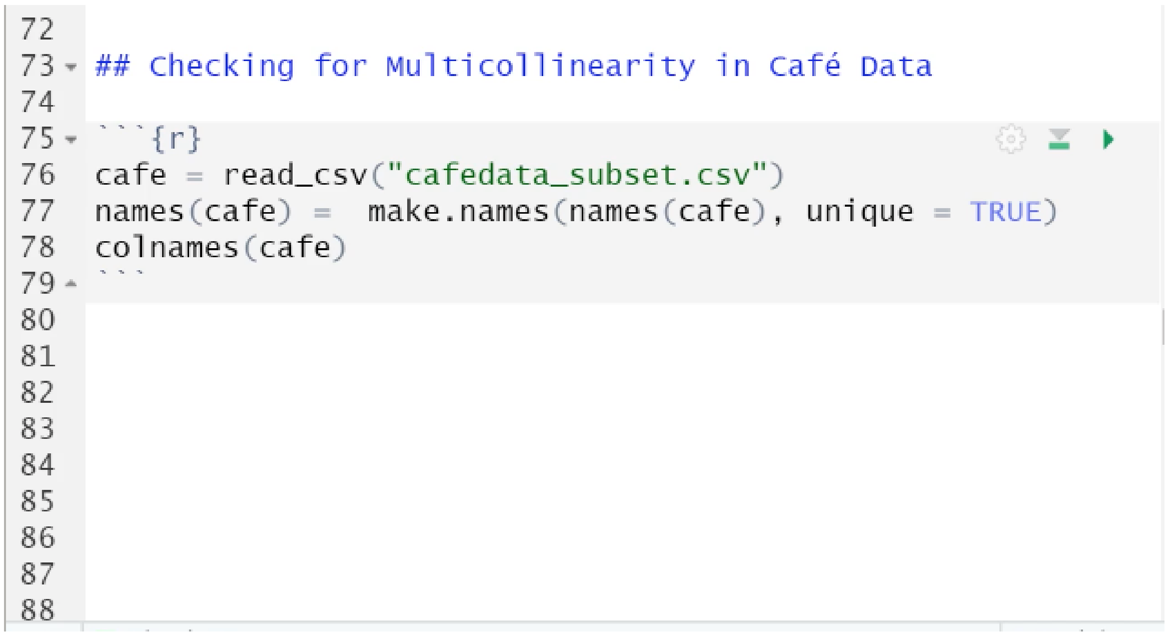
- ○ 0
- ○ 1
- ○ It depends on the strength of the relationship between each predictor and the response.

Answer is at the end of this transcript

Sample Problem

Check for multicollinearity in sales data from a student- run café.

```
> cafe = read.csv("cafedata_subset.csv")
> colnames(cafe)
 [1] "t"                    "Date"                 "Day.code"
 [4] "Day.of.Week"          "Bread.Sand.Sold"      "Bread.Sand.Waste"
 [7] "Wraps.Sold"           "Wraps.Waste"          "Muffins.Sold"
[10] "Muffins.Waste"        "Cookies.Sold"         "Cookies.Waste"
[13] "Fruit.Cup.Sold"       "Fruit.Cup.Waste"      "Chips"
[16] "Juices"               "Sodas"                "Coffees"
[19] "Total.Soda.and.Coffee" "Sales"               "Max.Temp"
[22] "Total.Items.Wasted"
```



~~~
## Checking for Multicollinearity in Café Data

```{r}
cafe = read_csv("cafedata_subset.csv")
names(cafe) =  make.names(names(cafe), unique = TRUE)
colnames(cafe)
```
~~~

This slide represents a video/screencast in the lecture. The transcript does not substitute video content.

Let's investigate whether multicollinearity is likely to be an issue when predicting sales at a student run cafe. I've already read in the data, and I'm using the function make.names to take all of the variable names and replace the spaces with periods. That will make them easier to refer to later on in our analysis.

This data set contains 22 variables. So a pairs plot of scatter plots ends up being pretty hard to read. We could make scatter plots of subsets of variables at a time, but I'm going to make a correlation plot instead. I've started by using the select if function from the dplyr package to select only the numeric variables for which we're able to compute the correlations. Then I'll use the cor function to compute the correlation matrix of pairwise correlations between each pair of numeric variables. And I'll make the correlation plot using the cor plot function from the cor plot library.

I'm using the function brewer.pal for Brewer palette to get the color scheme for my graph. And this is from the R color Brewer package. I'd like to reverse this color scheme so that red represents large values, correlations close to positive 1, and blue represents large negative values close to negative 1. To me it's more intuitive for red to be high and blue to be low.

Here's our correlation plot. Here we're trying to predict sales. So we're happy to see that sales is reasonably strongly correlated with a number of other variables in the data set. The other correlations that aren't with sales might be of concern, indicating multicollinearity. It looks like coffees is strongly negatively correlated with the maximum temperature on a given day. That makes sense that people would tend to want to buy coffee when it's cold out.

We also see that the maximum temperature is correlated with t, which represents the day on which the data was gathered. That makes sense also because this data set was gathered during one semester, the spring semester, at a university. So day is going to roughly start in January and end in April. It makes sense that it would be positively associated with the temperature.

We do see some groups of variables that are correlated with each other, such as the numbers of items sold of different types and the numbers of items wasted of different types. So it's going to be worth looking at the variance inflation factors to see if there's enough multicollinearity there that we need to be concerned.

The correlation graph and pairs graph of scatter plots can only tell us about relationships between pairs of numeric variables. So it's a good idea to also look at the data dictionary and think about possible interpretations to get a sense of what relationships might exist among the categorical variables and among groups of more than two variables at a time.

For the cafe data set, right away you might notice that we might have an association going on among these first four variables. Let's check this out using R. We can get a better sense of what's in these variables by looking at the first few entries using the head function.

So we see that the first few entries of the t variable are just the numbers from 1 to 6. And by looking at the first few entries of a table of dates, it looks like each date only occurs once. We can verify this by looking at the length of the date variable and then reducing that to just the unique elements of date and comparing the length of that vector. We see that both of those vectors have length 42. So yes, indeed, the date vector is just a list of 42 different dates with no dates repeated. So it looks like the t variable is just a numeric version of that same information. So we don't gain anything by including both of those variables.

Between these two variables, t is definitely the better choice to use in our model. That's because date would be treated as a categorical variable. Recall that linear regression, like many other modeling types, will automatically one-hot encode categorical variables, meaning that our 42 different dates will be turned into 41 different 0, 1 indicator variables. That would massively increase our number of predictor variables, which would lead to overfitting. In fact, because we had as many different dates as we had observations in our data set, we would end up with a model that perfectly predicted the sales of the data in our training set but which was useless for predicting the sales on any new data.

Back in the data dictionary, it looked like day code was simply a numeric representation of the day of the week. We can verify this by making a table of these two variables together. And here we can see that yes, 5 represents Friday, 1 represents Monday, and so on. So these two variables are giving us the exact same information as well. We don't need both of them in our analysis.

Between these two variables, I actually am going to choose to use day of week instead of day code. That's because this is a linear regression model. So using day code as a numeric variable would model the effect of the day as being either linearly increasing or linearly decreasing as you go from Monday to Friday throughout the week. In comparison, using day of week, which is a categorical variable, will allow us to estimate the effect separately of Mondays, Tuesdays, Wednesdays, Thursdays, and Fridays. So maybe at this university because of the way classes are scheduled, sales are higher on Mondays, Wednesdays, and Fridays than they are on Tuesdays or Thursdays. Using a categorical variable, day of week, would better allow us to model that.

So let's go ahead and fit a linear regression model of sales as a function of all the other variables in the data set except for date and day code. When we look at our model, we see that two of our variables are listed as NA, meaning that our model was not able to estimate a coefficient for these two variables. This happened because our model was completely singular. That is, these two variables were perfectly predictable based on the other predictor variables in our model.

If we look back at our data dictionary, we can see that the total soda and coffee variable probably is closely related to the variables sodas and coffee. And similarly, the total items wasted is probably closely related to some of these other variables, such as the amount of fruit cup waste, the amount of cookies waste, and so on.

Let's investigate whether these are in fact perfectly associated. Here I'm using the mutate function from dplyr to create a temporary new variable called waste sum. So I'm adding together all of the other waste variables. And then I'll use gf point to make a graph of total items wasted and waste sum. So we can see if total items wasted is, in fact, exactly the same as the sum of the other wastes.

Here we can see that the variables are perfectly on a line with each other. So this explains the perfectly singular result we got in our linear regression. To keep things simple, let's go ahead and use total items wasted and remove the other waste variables from our analysis.

To investigate total soda and coffee, we could make a graph just like we did for the total waste, but I'll try it a little bit differently. I'm still using mutate, but this time I'm creating a variable called is equal, which will be equal to true if sodas plus coffees is exactly equal to the total soda and coffee. Then I'll group by the value of that new variable is equal, and I'll count how many rows are equal to true, meaning sodas and coffees added together gives the total soda and coffee.

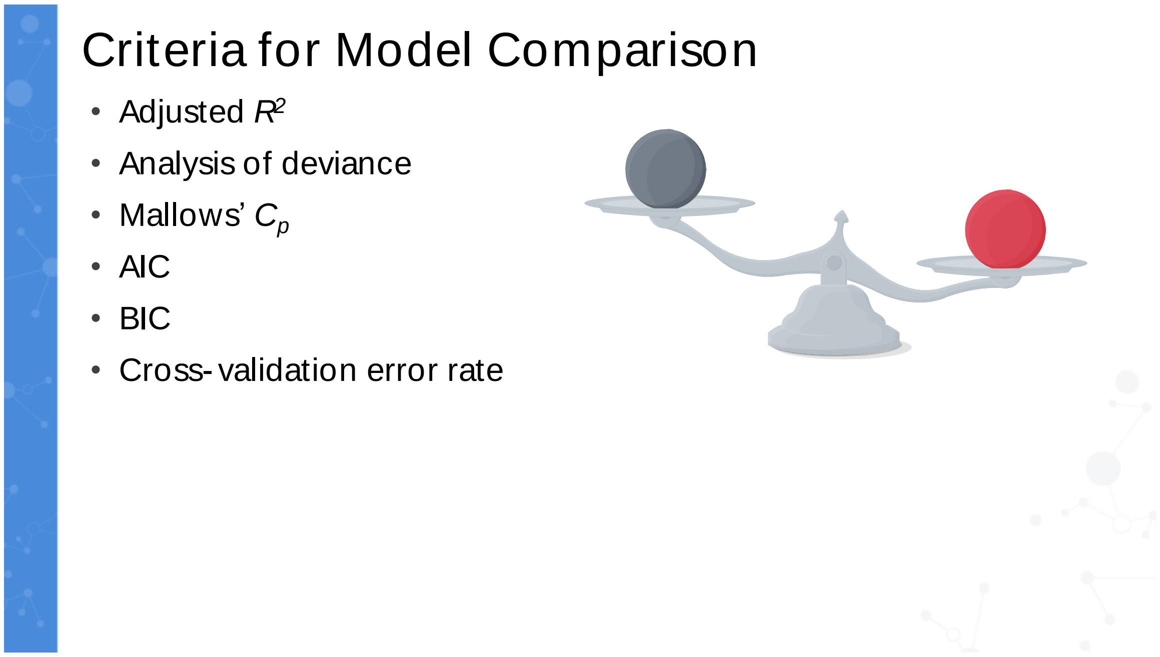
Here we see that all 42 rows in the data set give a value of true, meaning that we can perfectly predict the value of total soda and coffee based on the value of sodas and coffees. So for the purposes of our analysis, let's remove total soda and coffee and we'll just use sodas and coffees. So here's our new linear model where we're predicting sales based on all of the other variables except for the ones that we decided to exclude for reasons of multicollinearity. Linearity

Here we can see that we no longer have any NAs in our model. But some of our standard errors are still pretty large compared to the size of the coefficient that we're estimating. This could be a sign of additional multicollinearity that we haven't detected yet. To investigate this, let's look at the variance inflation factor.

Here we see that the variable day of week has a large variance inflation factor of 14.88. However, remember that that was a categorical variable with five possible levels, one for each day of the week, meaning that it had four degrees of freedom. This means that a variance inflation factor of 14 isn't quite as bad as it might seem for a variable that only had one degree of freedom, such as a quantitative variable.

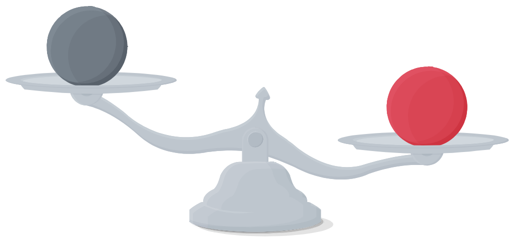
To get a better comparison among variables with different degrees of freedom, we can look at this third column, which contains the variance inflation factor raised to a power that depends on the degrees of freedom. Here we can see that day of week isn't looking so extreme anymore. And sodas has a variance inflation factor of 3.05.

Generally, you want to keep this third column less than the square root of 10, which is about 3.16. So the variance inflation factor for sodas isn't quite over that threshold. But I would be more concerned about sodas than I would be about day of week. If this were all of the investigation of multicollinearity that I was planning to do with this data set, I might consider either excluding sodas or regressing it on the other predictor variables and then replacing.

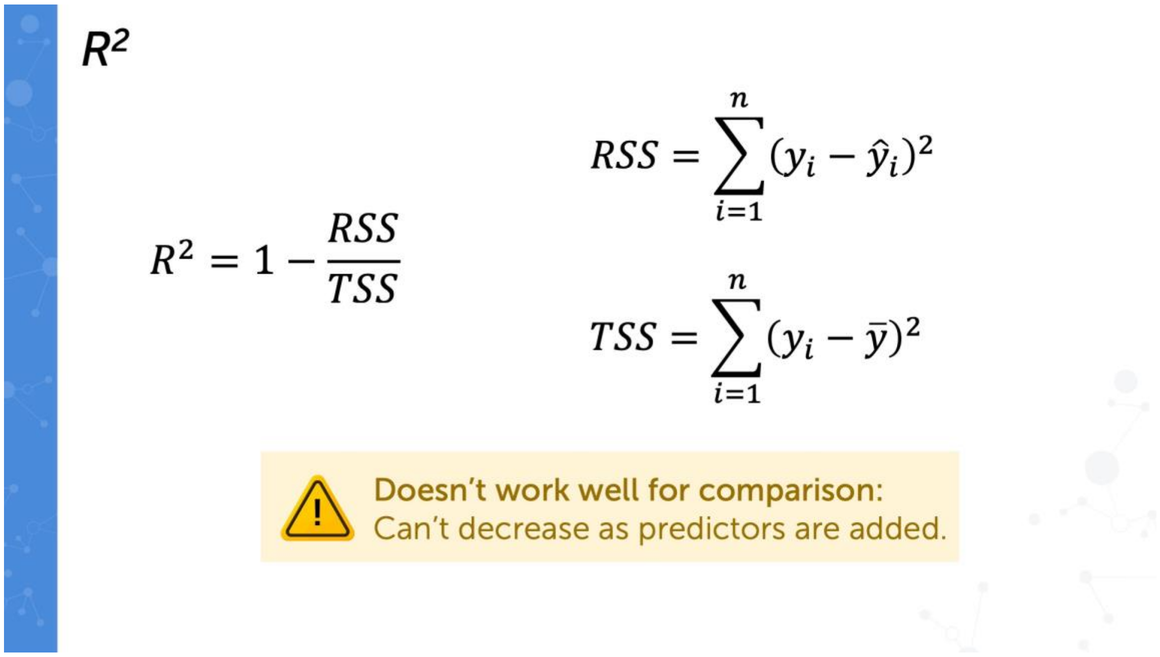


Criteria for Model Comparison

- Adjusted R^2
- Analysis of deviance
- Mallows' C_p
- AIC
- BIC
- Cross-validation error rate



To go deeper than checking for multi-collinearity, we need a way of comparing models to determine where they fall in the trade-off between bias and variance.



R^2

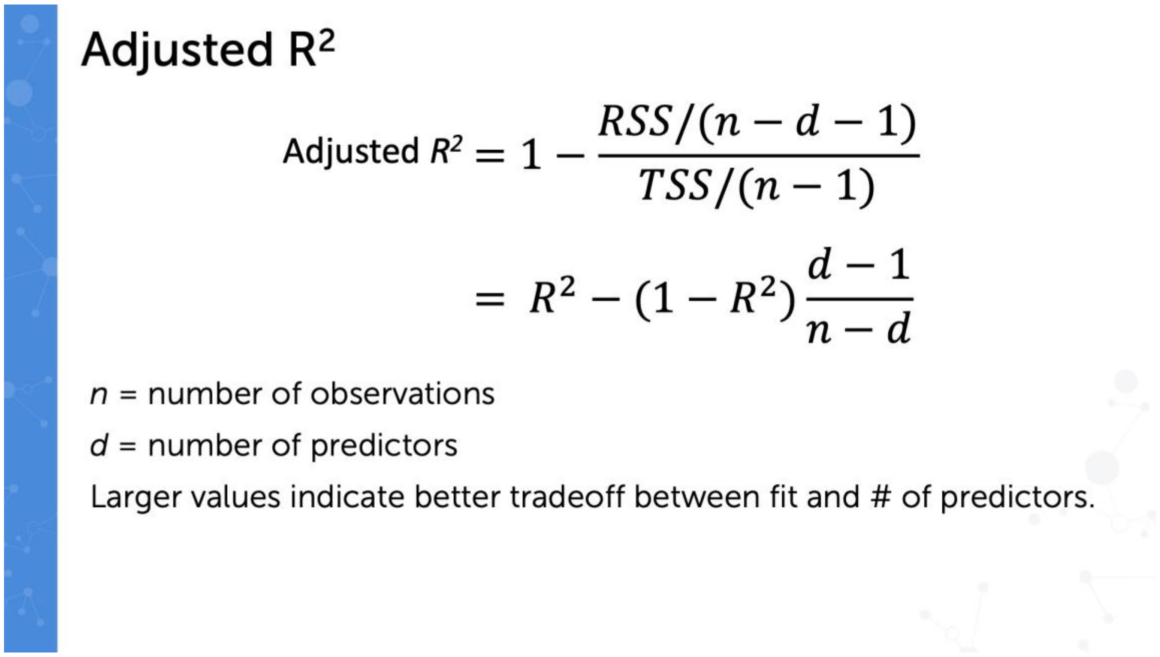
$$R^2 = 1 - \frac{RSS}{TSS}$$

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

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

⚠ **Doesn’t work well for comparison:**
Can’t decrease as predictors are added.

Recall that the coefficient of determination is 1 minus the residual sum of squares divided by the total sum of squares. This is not a good criterion for model comparison, because it can't decrease as predictor variables are added. So if you simply try to choose the model that maximizes r-squared, you'll always end up choosing the most complicated model.



Adjusted R²

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

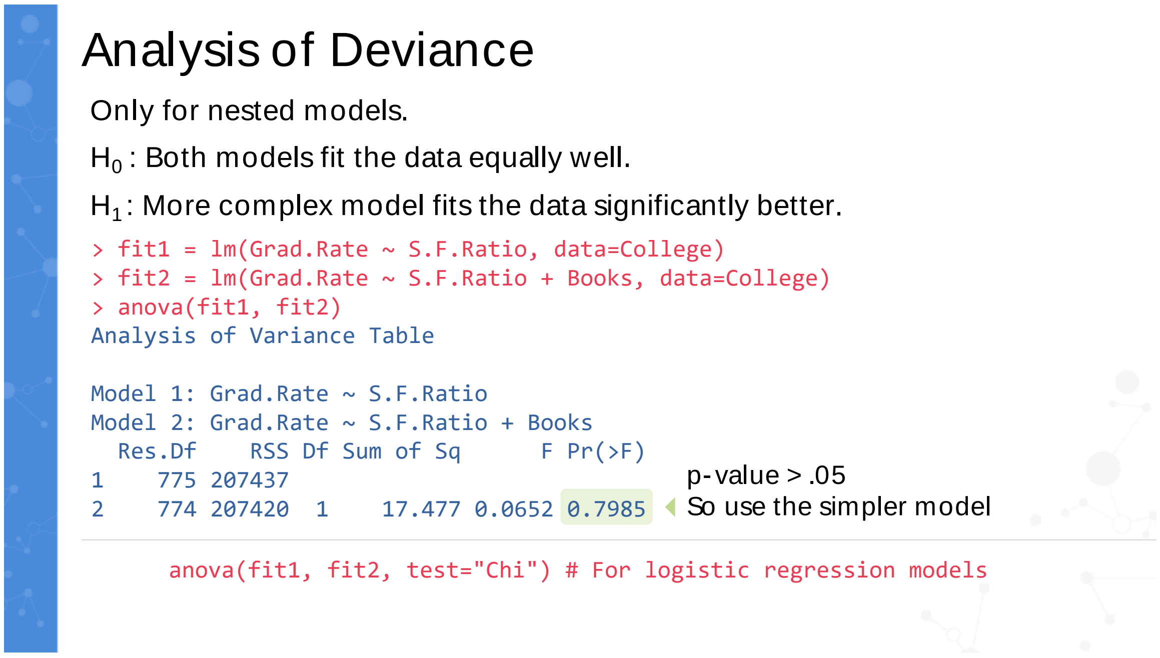
$$= R^2 - (1 - R^2)\frac{d-1}{n-d}$$

n = number of observations

d = number of predictors

Larger values indicate better tradeoff between fit and # of predictors.

The adjusted r-squared takes r-squared and subtracts a penalty based on d, the number of predictors, not counting the y-intercept. This means that larger values of the adjusted r-squared indicate a better trade-off between fit to the training data and the number of predictors.



Analysis of Deviance

Only for nested models.

H_0 : Both models fit the data equally well.

H_1 : More complex model fits the data significantly better.

```
> fit1 = lm(Grad.Rate ~ S.F.Ratio, data=College)
> fit2 = lm(Grad.Rate ~ S.F.Ratio + Books, data=College)
> anova(fit1, fit2)
Analysis of Variance Table

Model 1: Grad.Rate ~ S.F.Ratio
Model 2: Grad.Rate ~ S.F.Ratio + Books
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    775 207437
2    774 207420  1    17.477 0.0652 0.7985
```

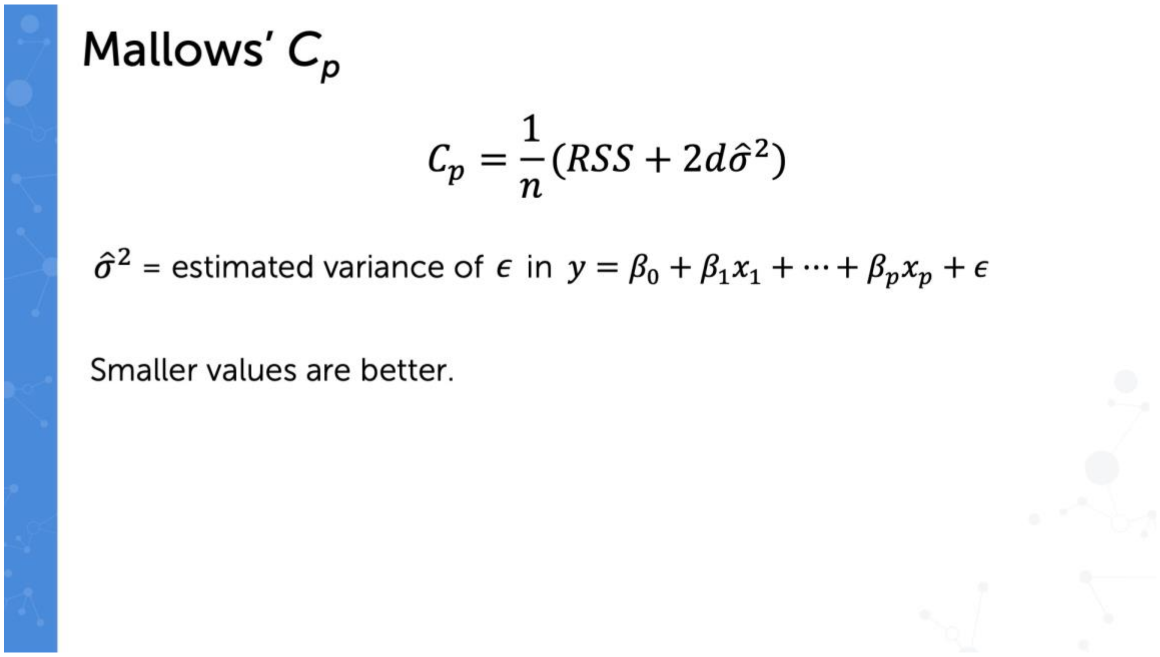
p-value > .05
So use the simpler model

```
anova(fit1, fit2, test="Chi") # For logistic regression models
```

The analysis of deviance is a hypothesis test for comparing models. Unlike adjusted r-squared and the other criteria for model comparison that we'll discuss in this lesson, the analysis of deviance should only be used for comparing nested models, ones in which the predictor variables of one model are a subset of the predictor variables of the other model.

The null hypothesis is that both models fit the data equally well. The alternative hypothesis is that the more complex model fits the data significantly better. In this example, we get a large P-value, so we retain the null hypothesis that both models fit the data equally well.

So in this case, there's not any good reason to use the more complex model, which increases the variance. So we would prefer to use the simpler model. The analysis of deviance can also be used for logistic regression models by including the argument test equals chi.



Mallows' C_p

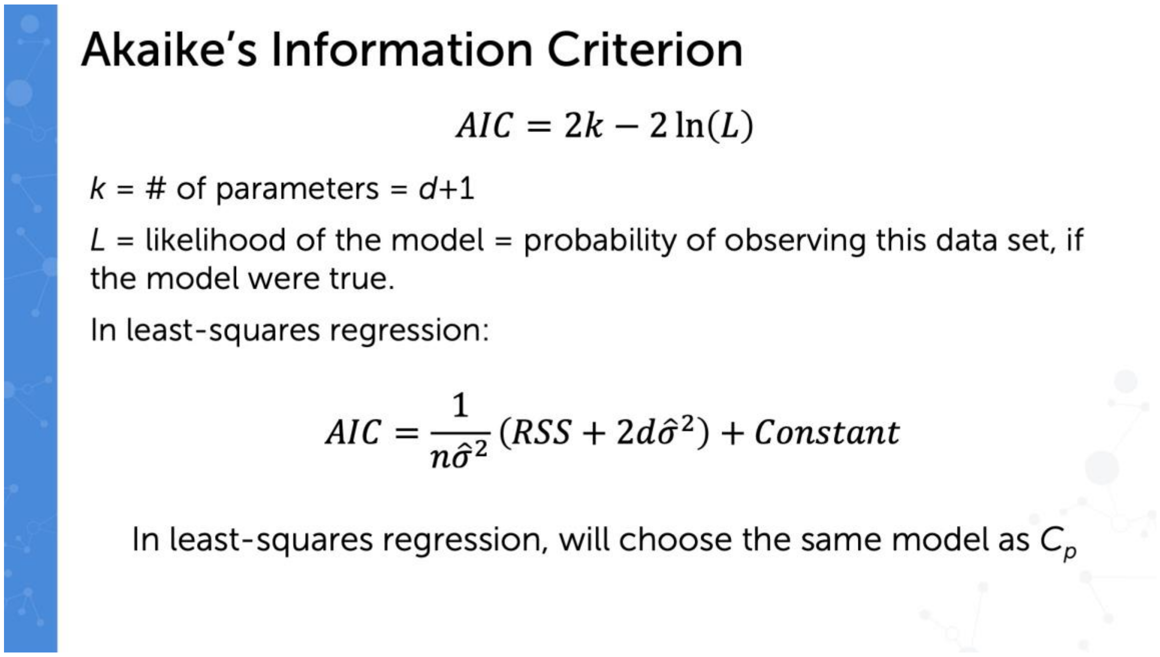
$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$

$\hat{\sigma}^2$ = estimated variance of ϵ in $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon$

Smaller values are better.

Mallows' Cp is another criterion for model comparison. It's based on the residual sum of squares plus a penalty term based on d, the number of predictor variables and sigma squared hat, the estimated variance of the noise terms in the linear regression.

A better model will tend to fit the training data well, so it will have a small residual sum of squares, and it will have a small penalty term. So smaller values of Mallows' Cp indicate a better model.



Akaike's Information Criterion

$$AIC = 2k - 2\ln(L)$$

k = # of parameters = d+1

L = likelihood of the model = probability of observing this data set, if the model were true.

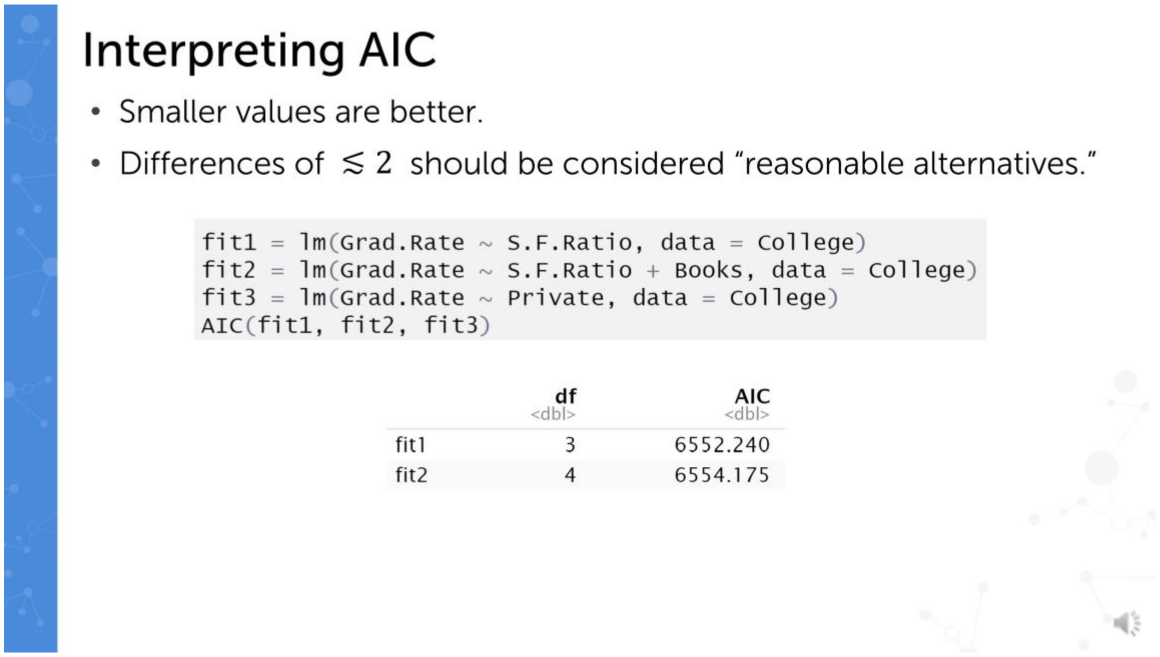
In least-squares regression:

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

In least-squares regression, will choose the same model as C_p

Akaike's Information Criterion is equal to 2 times the number of parameters minus 2 times the log likelihood, meaning the probability of observing this data set if the model were true. I really like Akaike's Information Criterion, because it's based on the likelihood. That means it's easy to apply to other types of models besides linear regression.

If you can write a statistical likelihood for the model, you can compute the AIC. However, AIC also has a special form when it's applied to least squares regression. You'll notice that this form is very similar to the form for Mallows' Cp. In fact, in least squares regression, AIC will always choose the same preferred model as Mallows' Cp.



Interpreting AIC

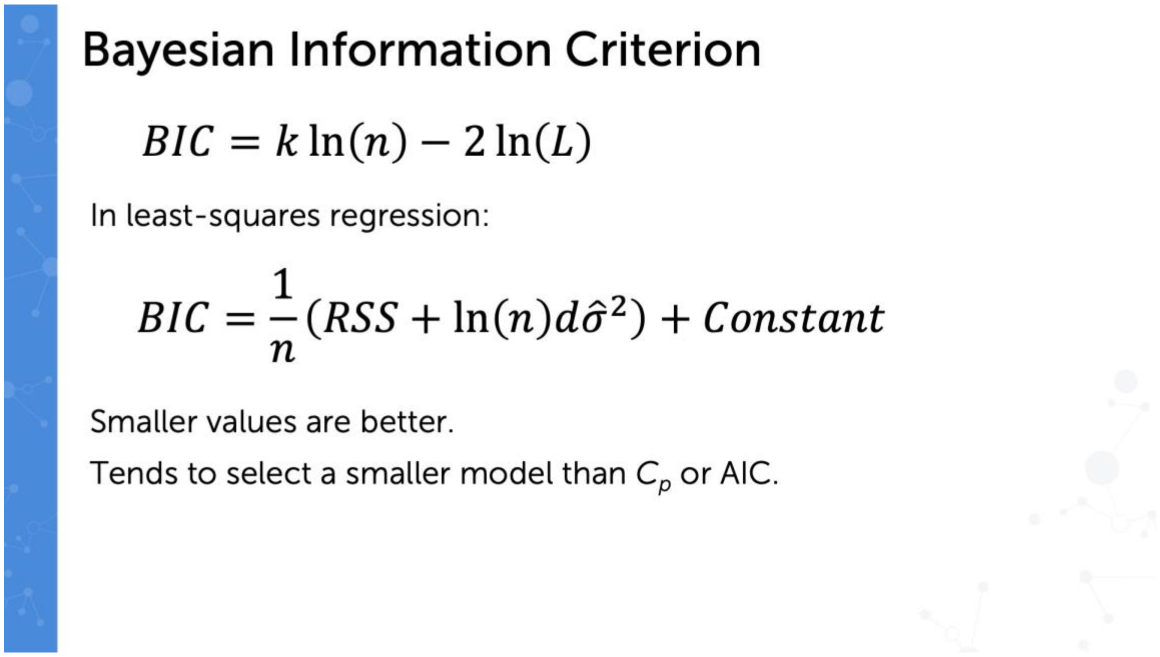
- Smaller values are better.
- Differences of $\lesssim 2$ should be considered "reasonable alternatives."

```
fit1 = lm(Grad.Rate ~ S.F.Ratio, data = College)
fit2 = lm(Grad.Rate ~ S.F.Ratio + Books, data = College)
fit3 = lm(Grad.Rate ~ Private, data = College)
AIC(fit1, fit2, fit3)
```

	df <dbl>	AIC <dbl>
fit1	3	6552.240
fit2	4	6554.175

Like Mallows' CP, smaller values of AIC indicate better models. Models with an AIC difference of less than about 2 are generally considered to be reasonable alternatives. For example, in the models shown here, fit 1 and fit 2 are reasonable alternatives to each other because their AICs are within 2 of each other, but fit 3 has a much lower AIC. So based on AIC, this model is the best trade-off between accuracy and parsimony.
Notes:

Styliano, Pickles, and Roberts recommend requiring AIC differences of 6 or more before rejecting a model:
Using Bonferroni, BIC and AIC to assess evidence for alternative biological pathways: covariate selection for the multilevel Embryo-Uterus model (2013)



Bayesian Information Criterion

$$BIC = k\ln(n) - 2\ln(L)$$

In least-squares regression:

$$BIC = \frac{1}{n}(RSS + \ln(n)d\hat{\sigma}^2) + Constant$$

Smaller values are better.

Tends to select a smaller model than C_p or AIC.

The Bayesian Information Criterion, or BIC, is similar to AIC except that the penalty for extra parameters depends on the sample size n. The larger the sample size, the bigger the penalty. So BIC tends to select smaller models than either Mallows' Cp or AIC.

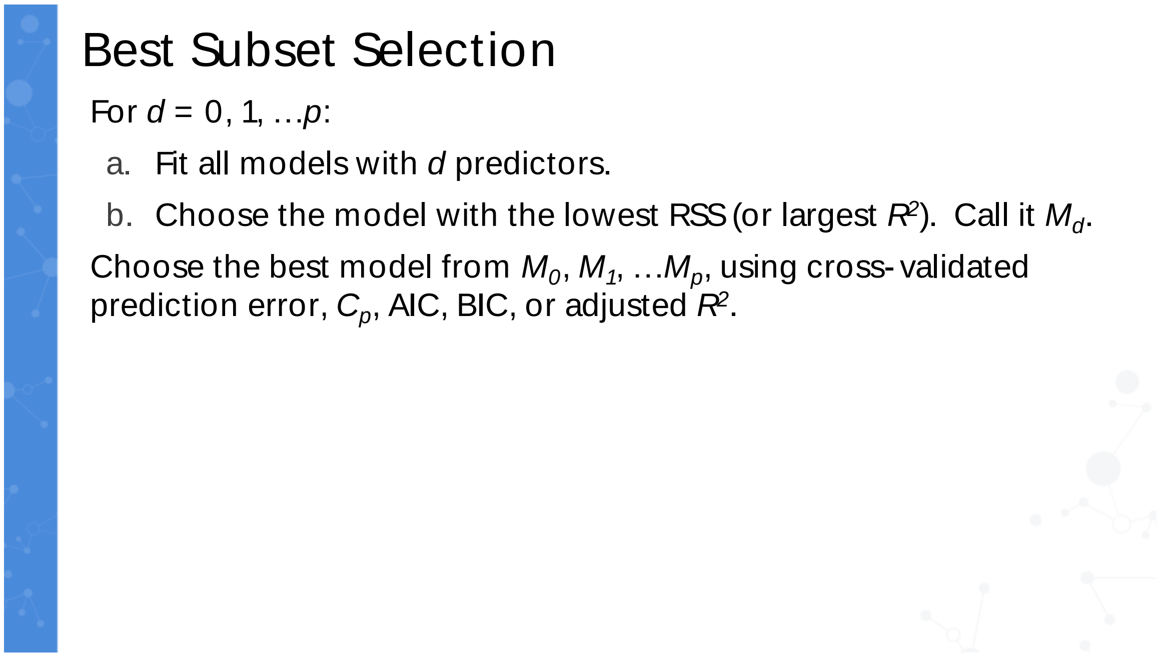
⚠ Caution

C_p, AIC, BIC involve constants which some functions omit.

When comparing models, always use the **same function for both models**.

```
> AIC(fit1)
[1] 6552.24
> extractAIC(fit1)
[1]    2.00 4345.21
```

It's important to note that all three of these criteria for model comparison involve constants which some functions omit. For example, the r functions AIC and extract AIC produce very different values for the AIC of the same model. So when you're comparing two models, you should always use the same data set and the same function to compute the model comparison criterion for both of the models.



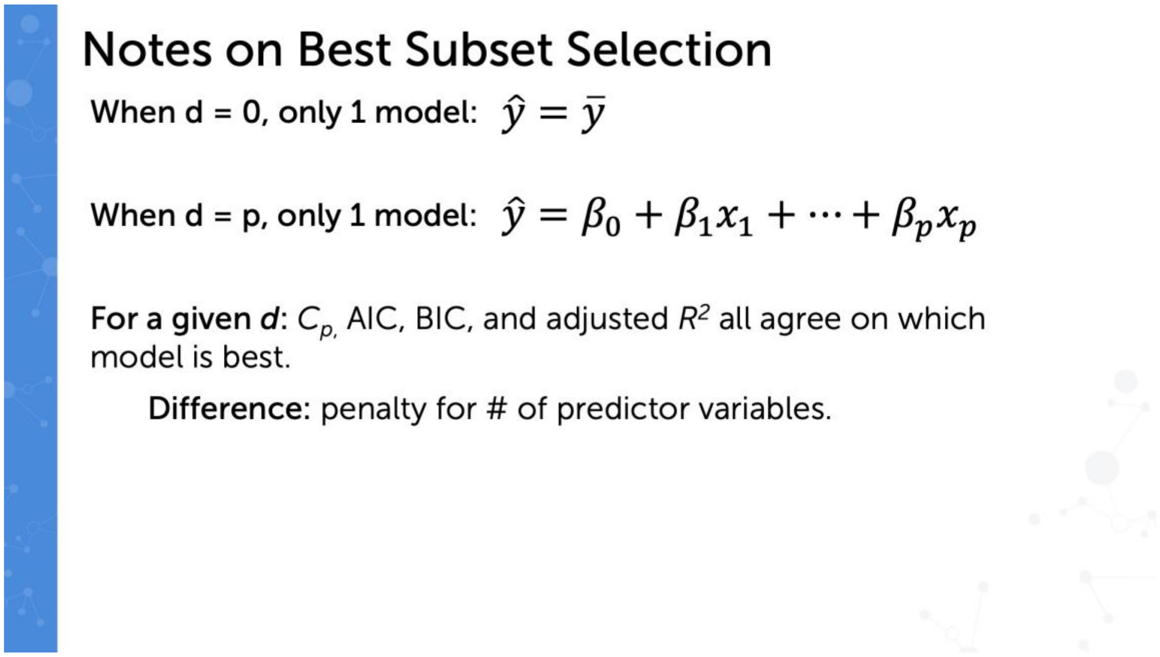
Best Subset Selection

For $d = 0, 1, \ldots p$:

a. Fit all models with d predictors.
b. Choose the model with the lowest RSS (or largest R^2). Call it M_d.

Choose the best model from $M_0, M_1, \ldots M_p$, using cross-validated prediction error, C_p, AIC, BIC, or adjusted R^2.

Best subset selection refers to a systematic approach for choosing the best possible model. Suppose we have P predictor variables available to us. Then for each possible number of predictors, from 0 up to P, will fit all of the models with that number of predictors. And we'll choose the one with the lowest residual sum of squares or, equivalently, the largest coefficient of determination. We'll call this model m sub d. That the best model of size d. Then we'll choose the best model from m sub 0 up through m sub p using one of our criteria for model selection.



Notes on Best Subset Selection

When d = 0, only 1 model: $\hat{y} = \bar{y}$

When d = p, only 1 model: $\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$

For a given *d*: $C_{p,}$ AIC, BIC, and adjusted R^2 all agree on which model is best.

Difference: penalty for # of predictor variables.

When d is equal to 0, there's only one model to consider-- the null model, in which each point's response value is predicted to equal the mean response value for the whole sample. When d is equal to p, we also only have one model to consider-- the full model, in which all of the predictor variables are used.

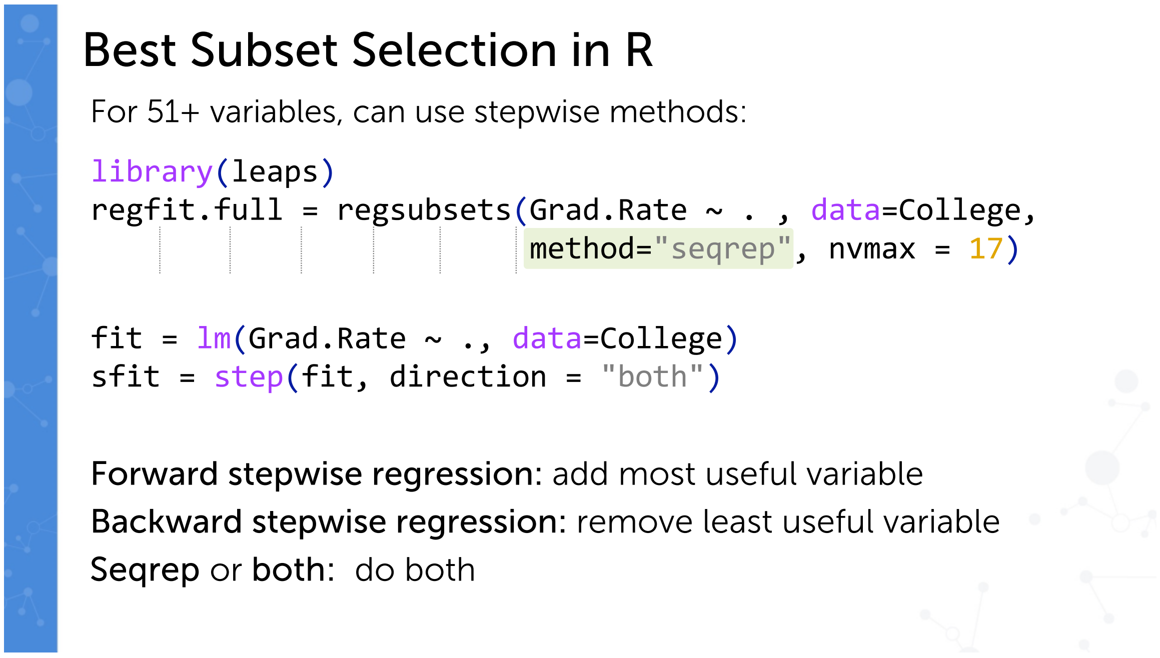
Also notice that Mallows' Cp, AIC, BIC, and adjusted r-squared all rely on the residual sum of squares. They only differ based on how they penalize the number of predictor variables. That means that for any given number of predictor variables d, these four criteria will all agree on which model is best. They'll only differ in their decisions in the last step of best subset selection when we're comparing models of different sizes.

Best Subset Selection in R

`regsubsets()` in `leaps` library has efficient algorithm for $p \leq 50$.

```
library(leaps)
regfit.full = regsubsets(Grad.Rate ~ . , data=College,
                         method="exhaustive", nvmax = 17)
```

To perform best subset selection in R, we can use the regsubsets function. If we do this using the argument method equals exhaustive, regsubsets will consider all possible models with numbers of variables between 1 and nvmax. In fact, exhaustive is the default value for the method argument. So we could omit this argument entirely and consider all possible models.



Best Subset Selection in R

For 51+ variables, can use stepwise methods:

```
library(leaps)
regfit.full = regsubsets(Grad.Rate ~ . , data=College,
                         method="seqrep", nvmax = 17)

fit = lm(Grad.Rate ~ ., data=College)
sfit = step(fit, direction = "both")
```

Forward stepwise regression: add most useful variable

Backward stepwise regression: remove least useful variable

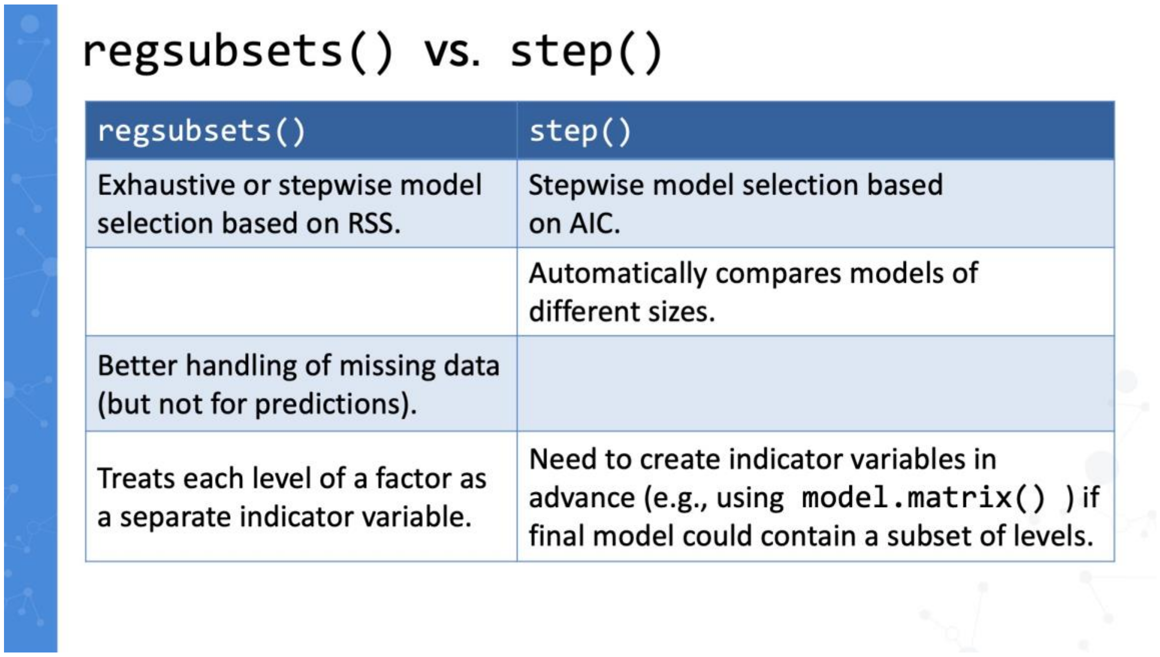
Seqrep or **both:** do both

If there are more than 50 predictor variables, we can use a stepwise approach either by changing the method argument in the regsubsets function or by using the step function, which is based on AIC. With this approach, instead of performing an exhaustive search of all possible models, at each stage we're doing one of two things-- either adding the most useful predictor variable to the model in forward stepwise regression or removing the least useful variable in backwards stepwise regression.

The argument seqrep for regsubsets or direction equals both in the step function allows us to do either forward or backward stepwise regression at each stage. This approach is not guaranteed to achieve the best possible model, but it can be useful if you have a lot of predictor variables. Forward stepwise regression can even be used if the number of predictor variables is larger than your sample size.

Notes:

```
regfit.full = regsubsets(Grad.Rate ~ ., data = College,
        method = "seqrep", nvmax = 17)

fit = lm(Grad.Rate ~ ., data = College)
sfit = step(fit, direction = "both")
```



`regsubsets()` vs. `step()`

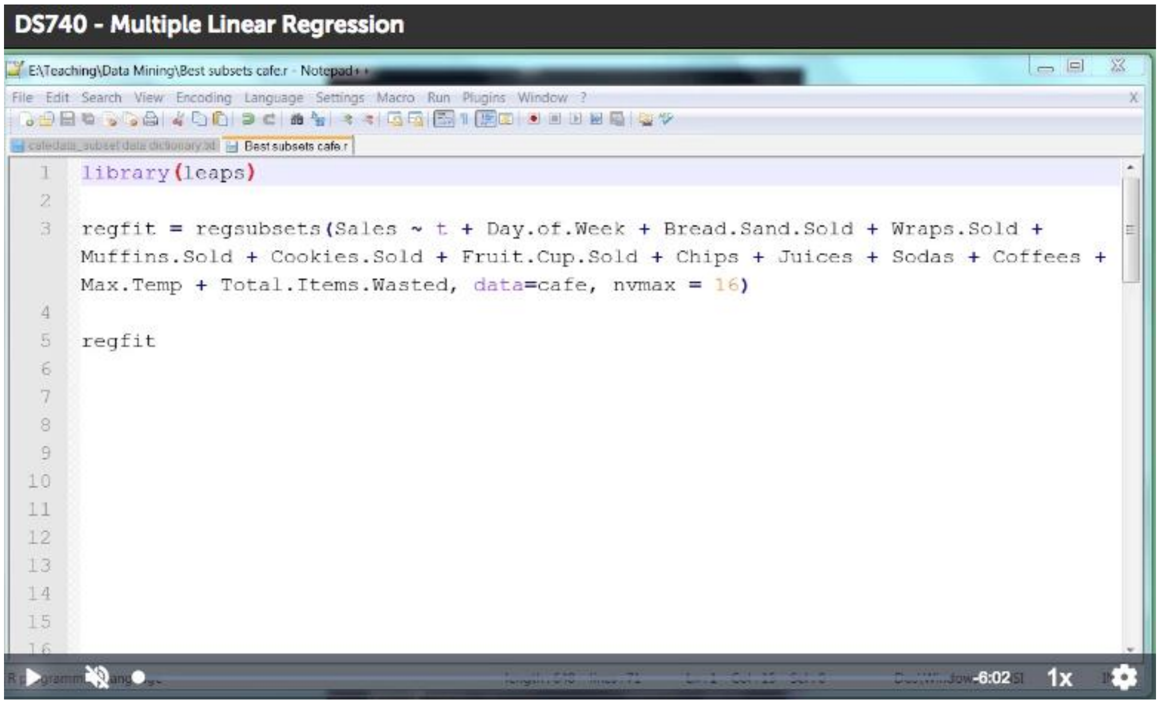
`regsubsets()`	`step()`
Exhaustive or stepwise model selection based on RSS.	Stepwise model selection based on AIC.
	Automatically compares models of different sizes.
Better handling of missing data (but not for predictions).	
Treats each level of a factor as a separate indicator variable.	Need to create indicator variables in advance (e.g., using `model.matrix()`) if final model could contain a subset of levels.

The step function is nice because it automatically compares models of different sizes, whereas reg subsets will just report the best model of each size and then leave the comparison between different sizes up to you. However, reg subsets has better handling of missing data, and it also treats each level of a factor as a separate indicator variable.

This is nice, for example, if you have sales data from different days of the week, and you want to consider the possibility that sales are different on Fridays but the same Monday through Thursdays.

Sample Problem, Continued

Apply best subset selection to the café sales data.


DS740 - Multiple Linear Regression

```
library(leaps)

regfit = regsubsets(Sales ~ t + Day.of.Week + Bread.Sand.Sold + Wraps.Sold +
Muffins.Sold + Cookies.Sold + Fruit.Cup.Sold + Chips + Juices + Sodas + Coffees +
Max.Temp + Total.Items.Wasted, data=cafe, nvmax = 16)

regfit
```

This slide represents a video/screencast in the lecture. The transcript does not substitute video content.

Let's perform Best Subset Selection on our cafe data, using an exhaustive search. To do this, we'll apply the Regsubsets function, which is in the Leaps library. The syntax of the Regsubsets function is very similar to the lm function, which we use for linear regression. The only additional argument here is nvmax. This sets the maximum number of predictor variables you want to include in any of the models.

I generally set this equal to the maximum number of predictor variables that you have in your dataset, remembering to add additional predictor variables for any additional levels of factor variables that you may have. If we simply look at the name of the object where we stored the regsubsets output, we get a summary of the analysis. And it's telling us that including all levels of the factor level variables, we had 16 variables. So double checking this against the nvmax that we used, we see that yes, we included all potential variables.

If we want to know more about the models that it considered, we can use plot regfit. And here we get a plot summarizing each of the variables that were possible predictor variables along the x-axis and the BIC of different models along the y-axis. And a dark rectangle indicates that that variable was included in that model. A lower BIC is better, which is indicated up at the top of this graph.

So here you can see that the best model with only a single predictor variable included the variable wraps.sold. But that model had a fairly high BIC, so it's down at the bottom of the graph. The model that was slightly better than that included all the possible predictor variables, the full model, with a BIC of about negative 16. If we want to change the y-axis, we can instead use the scale argument inside our plot function. Here I'll plot the adjusted r squared. And here we get that in terms of adjusted r squared, the full model actually didn't do too badly. It's shown up here. We can extract more information by using the Summary function. And here I'll store the summary results inside a new object.

Then we can extract information from that summary, using the dollar sign to extract pieces of that object. And I'll extract the BIC. Here we get the BIC of each model for one predictor variable up to 16 predictor variables. So we can plot this as a function of the number of variables. Here we see that as the number of predictor variables included in the model increases, the BIC of the best possible model of that size first decreases and then increases. We can use the which.min function to pick out which element of this vector of BICs is the lowest. It's the 10th element, which indicates that the model with 10 variables had the lowest BIC, which matches what we saw on the graph.

To extract the best model in terms of Mallows's Cp, we can use which.min again but extracting the Cp part of the summary object. Or we can extract the best model in terms of adjusted r squared, using which.max. Based on adjusted r squared, the model with 12 predictor variables was the best. But Mallows's Cp in this case agreed with BIC that the model with 10 predictor variables was the best. We can see what the coefficients of that model were by using the function coef with arguments regfit, which was our regsubsets object, and 10, the number of variables we wanted to be included in that model. So here we see the variables included in the model and the estimated coefficients for each of those. So for example, the number of sodas is positively associated with the total sales for the day.

Notes:

Number of variables = # of quantitative predictors + (# of levels – 1) for each categorical predictor.
Dataset: "Student-run Café Business Data," submitted by Concetta A. DePaolo and David F. Robinson, Indiana State University. Dataset obtained from the Journal of Statistics Education (http://www.amstat.org/publications/jse). Accessed 2 August 2016. Used by permission of author.

Cross-Validation for Best Subset Selection

regsubsets() returns adjusted R^2, C_p, and BIC for training data.

Use cross-validation to estimate error on new data.

```r
# Define a predict() function for regsubsets objects
predict.regsubsets <- function(object, newdata, id, ...){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id=id)
    xvars = names(coefi)
    mat[ , xvars] %*% coefi
} # end fuction predict.regsubsets
```

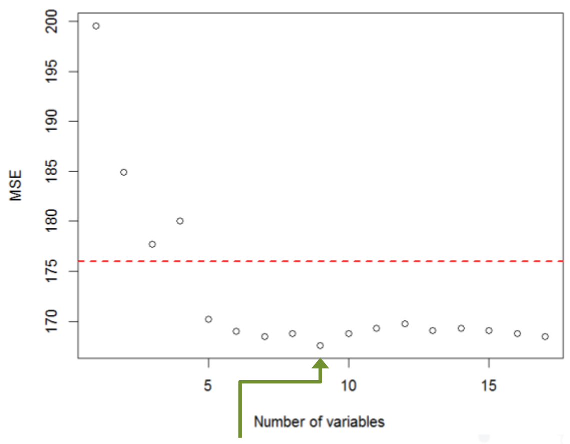
Reg subsets returns criteria for model comparison computed on the training data. However, if we want to estimate model performance on new data points, it's a good idea to use cross-validation. To do this, we need to be able to predict the response value for each of the data points in the validation set. Unfortunately, there's not a built-in predict function for models that are the output from reg subsets. So we need to write our own.

The function shown here will do this by first extracting the appropriate columns from the validation set and then multiplying them by the coefficients of the model.

One-Standard-Error Rule

All models with MSE within 1 standard error of the minimum are “near-optimal.”

Choose the one with the fewest predictors.

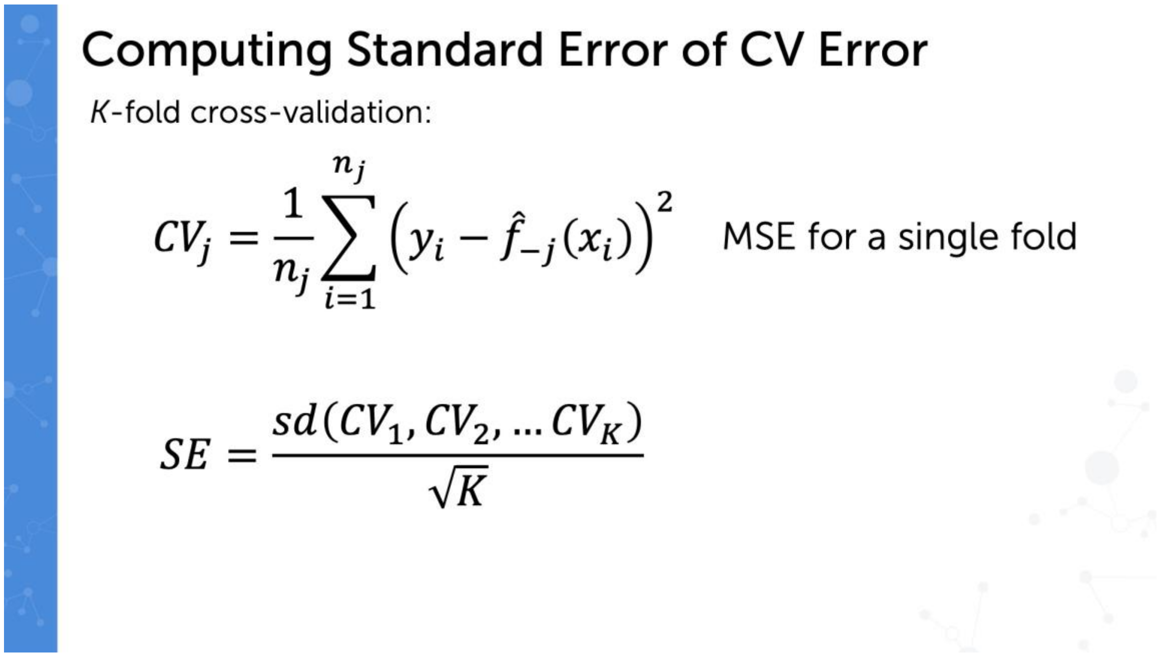


Lowest error this time, but could change…

A common occurrence is that as we increase the number of variables in the model, the cross-validation error will drop sharply at first and then level off. So we get a large number of models with very similar cross-validation error rates. In the example shown here, the model with nine variables had the lowest cross-validation error, but other models were very similar.

It might well be that if we repeated the cross-validation with a different random seed, we might find that the model with 8 variables or 10 variables was best. In this case, it doesn't really make sense to say that 9 is the absolute for certain optimal number of variables.

Instead, it makes more sense to say that all of the models with a mean squared error within a certain range, say, one standard error of the minimum, are near optimal. So we might prefer to choose the most parsimonious model, the one with the fewest predictors.



Computing Standard Error of CV Error

K-fold cross-validation:

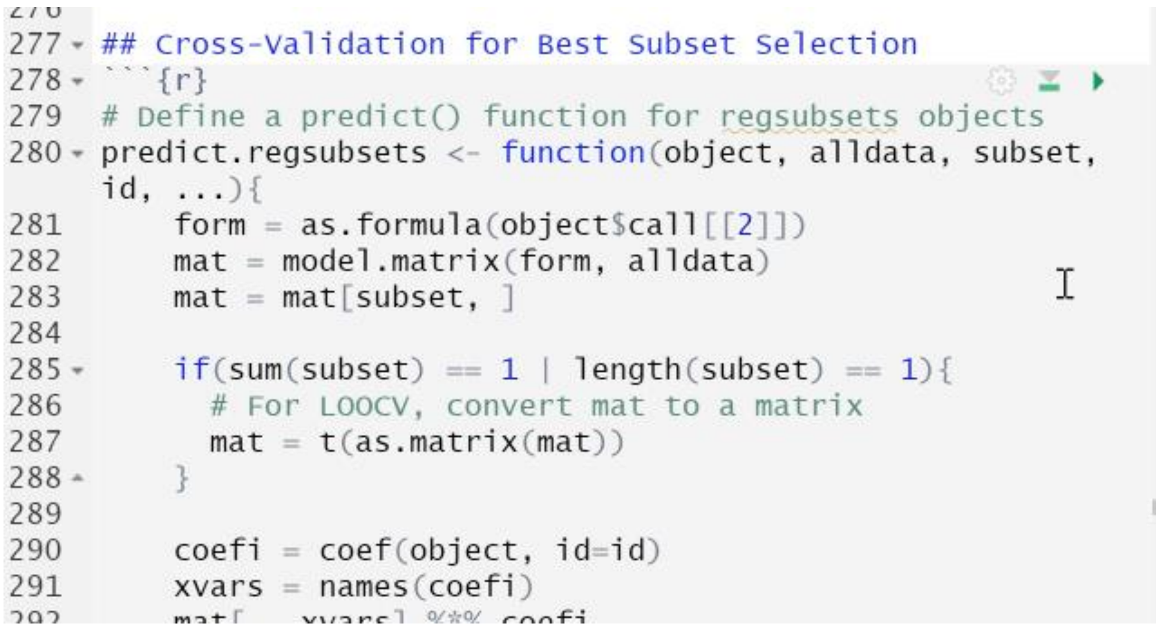
$$CV_j = \frac{1}{n_j}\sum_{i=1}^{n_j}\left(y_i - \hat{f}_{-j}(x_i)\right)^2$$ MSE for a single fold

$$SE = \frac{sd(CV_1, CV_2, \dots CV_K)}{\sqrt{K}}$$

Here's the formula for computing the standard error of the cross-validation error. Notice that the first formula, cv sub j, is exactly like computing the mean squared error, but we're doing it for a single fold of our k-fold cross-validation.

Sample Problem, Continued

Use cross-validation with the one-standard-error rule to choose a model for the café sales data.



~~~
## Cross-Validation for Best Subset Selection
```{r}
# Define a predict() function for regsubsets objects
predict.regsubsets <- function(object, alldata, subset,
id, ...){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, alldata)
    mat = mat[subset, ]

    if(sum(subset) == 1 | length(subset) == 1){
      # For LOOCV, convert mat to a matrix
      mat = t(as.matrix(mat))
    }

    coefi = coef(object, id=id)
    xvars = names(coefi)
~~~

This slide represents a video/screencast in the lecture. The transcript does not substitute video content.

To perform cross-validation for best subset selection I'm going to start by defining a predict function that will work for objects that are the output of the regsubsets function. Then, I'll perform cross-validation as usual, setting n equal to the number of rows in the cafe data set, and setting my value of n groups.

This is a fairly small data set, so I'm going to use leave-one-out cross-validation by setting n groups equal to n, the size of the data set. Then I'm creating my groups vector, setting my seed, and creating the cvgroups as a random permutation of the groups vector. Because we're doing leave-one-out cross-validation, we wouldn't really need to randomize the order of the groups vector. We could have just said cvgroups equals the vector from 1 up to n. I've defined the variable nvar equal to 16, the maximum number of variables that I might want to include as predictors in my model.

And here, instead of a vector called all_predicted, I'm defining a matrix called group_error. So this is set up so that each row will represent one fold, and each column will represent one model size, or number of variables.

Then we have our for loop, iterating over the folds as usual. We're setting up groupii, train_data and test_data as usual, and then I'm using regsubsets to fit my model.

Here we have data equals train_data, and nvmax equals nvar, which was 16. Here's where the difference comes in. Instead of simply computing the predictions for our test set, I'm now going to do a second for loop inside the first one, that's going to iterate over the different model sizes, from 1 up to 16. Inside that for loop, I'll make the predictions using my new predict function that I just wrote.

Recall that I wrote a function called predict dot regsubsets. Here I'm just calling predict. That's because this is a generalized function, where if you call the function predict it will automatically look at the object type of the first argument, see that it's a regsubsets object, and then look for a function called predict dot regsubsets.

So according to the way the arguments were set up in the function that I wrote, this first argument is our regsubsets object. The second argument is alldata equals cafe, the entire data set. Subset equals groupii, so the trues and falses defining which elements are in the test data set. And id equals jj, so that's the value of how many variables I want to include as predictors.

So this gives us our predictions for the test set. Normally we would just want to store this value in our all_predicted variable. But in this case, we're going to do it a little bit differently, and that's because we're going to want to compute the standard error of our mean squared errors. And the best way to do that is to compute the mean squared error separately for each of the folds.

So here I'm going to take the test data and the sales column, subtract my predictions, and then square it and take the mean to compute the mean squared error for this particular fold and this number of variables, jj. And then I'm storing that value in the row that corresponds to this fold, and the column that corresponds to this number of variables in my group_error matrix.

So now I'm going to come down here and actually compute the overall mean squared error. So our for loop was filling up this group_error matrix with each entry being the mean squared error from one fold. Now I want to take the mean over all the folds for a given number of variables. So I want to take the mean over all the columns. So that's index two of my group_error matrix.

That will give us our mean squared error overall, which should be a vector of 16 numbers. So one mean squared error for each number of variables. And then we can graph that as a function of the number of variables.

In this case, we see that the mean squared error starts out high, and then drops down to a minimum at two variables. If we want to programmatically determine which number of variables is best, we can use the function which dot min, which tells us that indeed two variables is the best model in terms of mean squared error.

In this case, it's pretty clear that having two variables both gives us a very low mean squared error and a very parsimonious model. But what if the best model had been one of these up here, with 12 or 14 variables? In that case, we might want to check if there was a more parsimonious model with a mean squared error within one standard error of the minimum.

We would do that by first computing the standard errors. We do that by taking the standard deviation of each column of the mean squared errors, and then dividing by the square root of the number of folds. Then we'll use this to define a threshold by taking the mean squared error overall from the lowest mean squared error model, and then adding the standard error of that model. And we want to take the most parsimonious model whose overall mean squared error is less than one standard error above the overall mean squared error of the lowest model.

So here we see that models 2, 10, and 12 through 16 all had mean squared errors within one standard error of the minimum. So we would want to pick the most parsimonious model from this list, which is the model with two variables.

To see which two variables are in this best model, and what their coefficients were, we want to go back to the model from the full data set, not the models from the cross-validation, which gave us a different model for each fold. So that was our regfit object from the previous video.

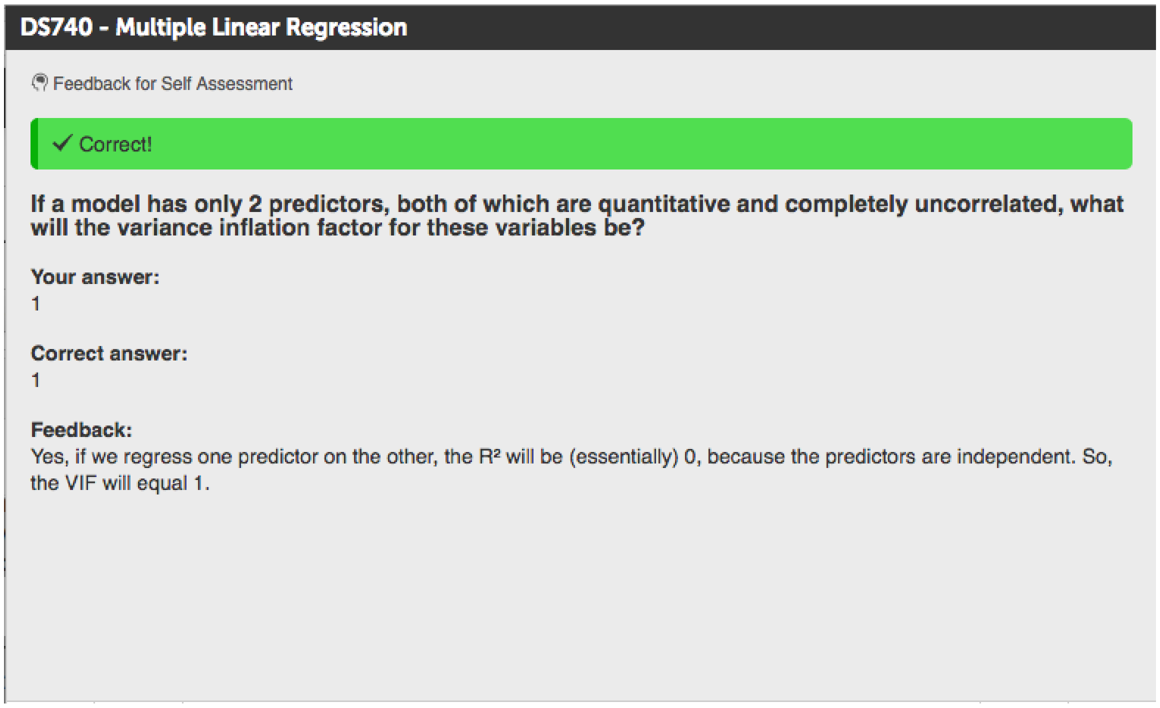
Notes:

Dataset: "Student-run Café Business Data," submitted by Concetta A. DePaolo and David F. Robinson, Indiana State University. Dataset obtained from the Journal of Statistics Education (http://www.amstat.org/publications/jse). Accessed 2 August 2016. Used by permission of author.

Summary

- Choosing a subset of variables for a linear regression model can improve interpretability and tradeoff between bias and variance.
- Adjusted R^2, Mallows' C_p, AIC, BIC agree on which model is best among all models of the same size.
  - They may disagree about which model is best when comparing models with different numbers of variables.
- The one-standard-error rule refers to choosing the simplest model with a cross-validation error rate within 1 standard error of the minimum.
- To perform best subset selection based on RSS in R, use `regsubsets()`.

Question 1 Answer



DS740 - Multiple Linear Regression

Feedback for Self Assessment

✓ Correct!

If a model has only 2 predictors, both of which are quantitative and completely uncorrelated, what will the variance inflation factor for these variables be?

Your answer:
1

Correct answer:
1

Feedback:
Yes, if we regress one predictor on the other, the R^2 will be (essentially) 0, because the predictors are independent. So, the VIF will equal 1.