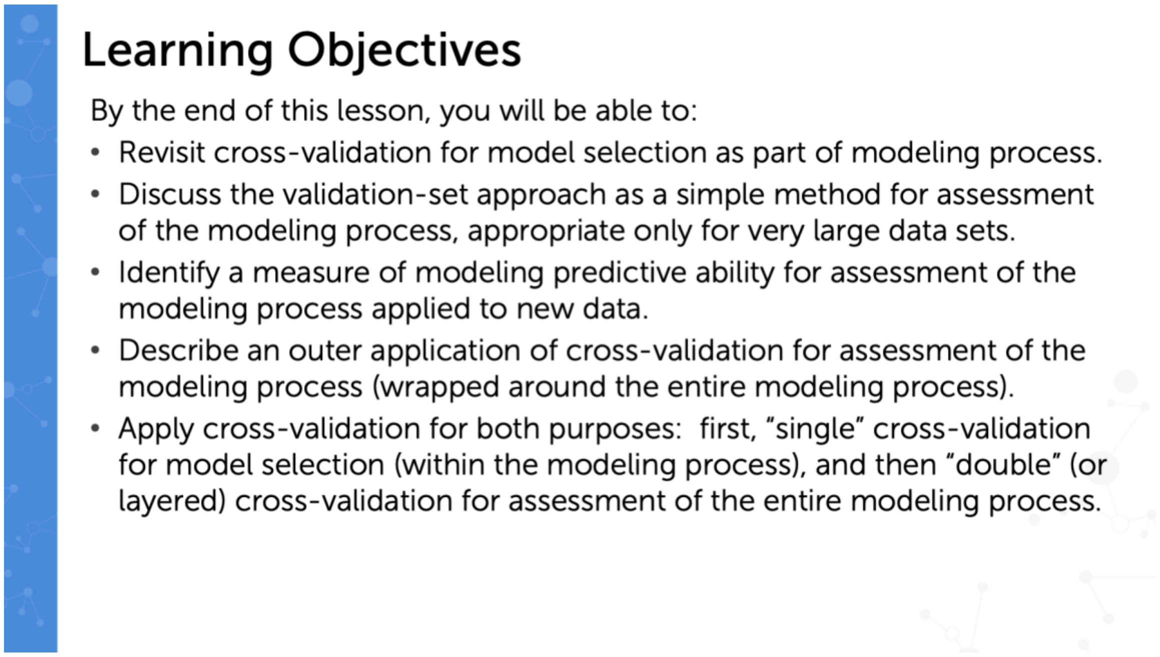
UNIVERSITY OF WISCONSIN
DATA SCIENCE

DS 740

Data Mining

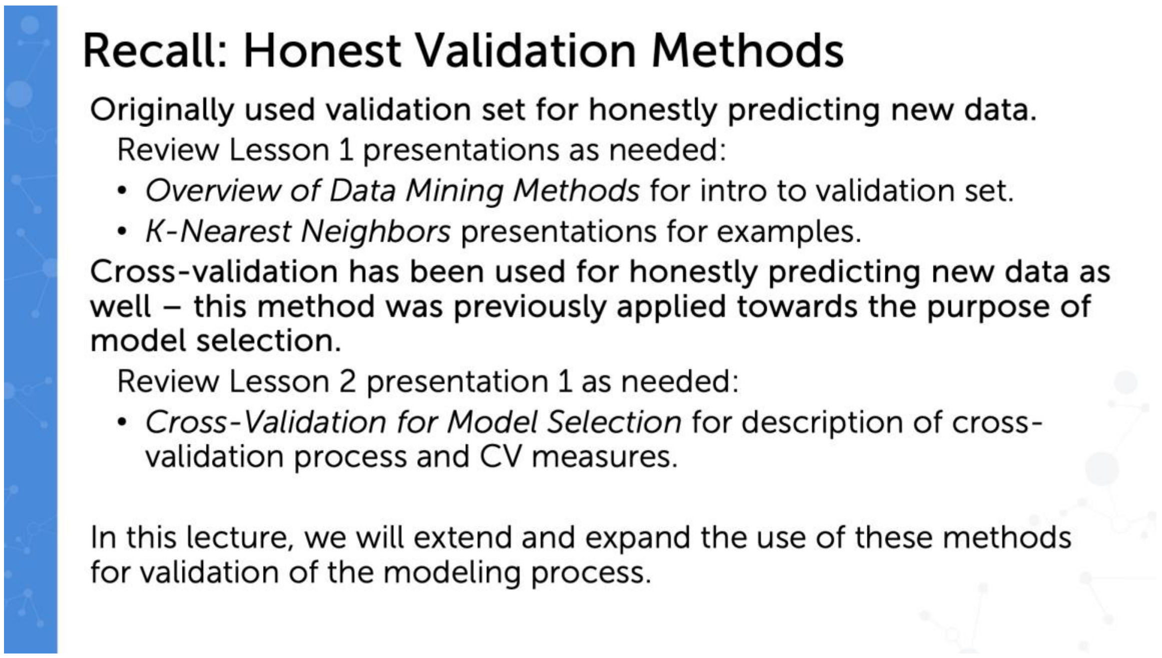
"Double" Cross-Validation:
Cross-Validation for the Assessment of Modeling Process



Learning Objectives

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

- Revisit cross-validation for model selection as part of modeling process.
- Discuss the validation-set approach as a simple method for assessment of the modeling process, appropriate only for very large data sets.
- Identify a measure of modeling predictive ability for assessment of the modeling process applied to new data.
- Describe an outer application of cross-validation for assessment of the modeling process (wrapped around the entire modeling process).
- Apply cross-validation for both purposes: first, “single” cross-validation for model selection (within the modeling process), and then “double” (or layered) cross-validation for assessment of the entire modeling process.



Recall: Honest Validation Methods

Originally used validation set for honestly predicting new data.

Review Lesson 1 presentations as needed:

- *Overview of Data Mining Methods* for intro to validation set.
- *K-Nearest Neighbors* presentations for examples.

Cross-validation has been used for honestly predicting new data as well – this method was previously applied towards the purpose of model selection.

Review Lesson 2 presentation 1 as needed:

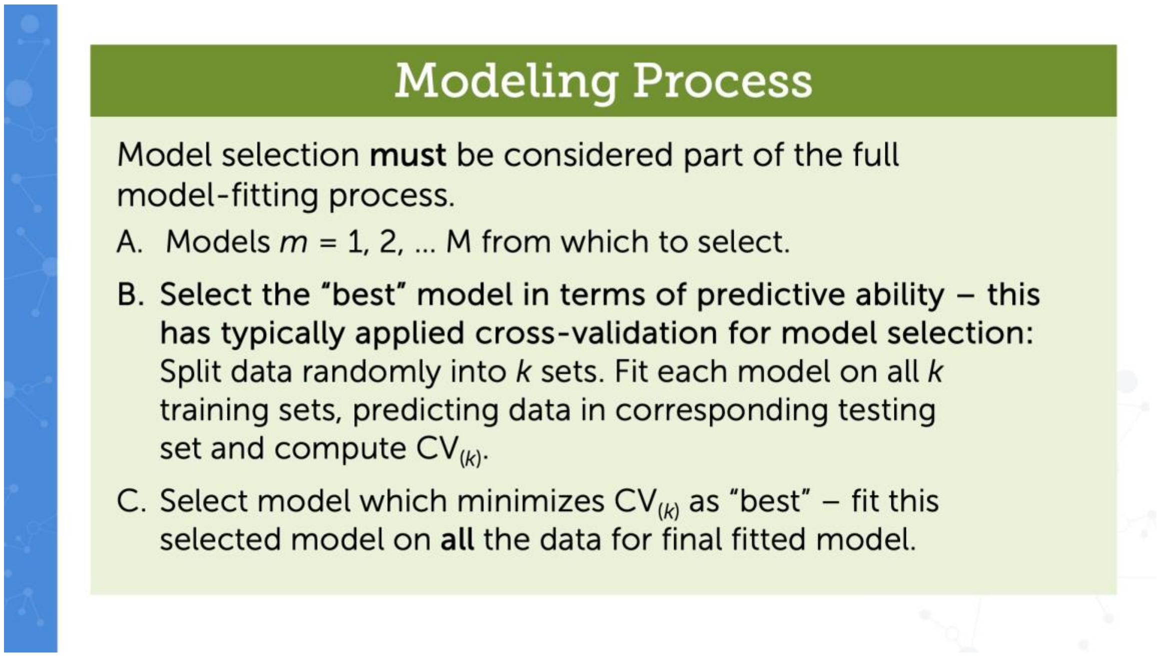
- *Cross-Validation for Model Selection* for description of cross-validation process and CV measures.

In this lecture, we will extend and expand the use of these methods for validation of the modeling process.

The first method covered early in this course for honest prediction was to simply split the data into a training set and a validation set. The training set was used to fit the model, which in turn was used to predict truly new data in the validation set. This is applicable and useful for very large data sets.

We also discussed cross validation for honest prediction. This can be applied to data sets of all sizes. Cross validation was originally introduced for use in model selection, and we revisited this in the caret presentation. We will now use cross validation for assessment purposes as well.

In this presentation, we discuss applying both validation set and cross validation to assess the entire modeling process. We note that as part of this modeling process that we are assessing may involve an inner cross validation for model selection, for example with caret. We must carefully apply and outer validation method to honestly assess the entire modeling process.


Modeling Process

Model selection **must** be considered part of the full model-fitting process.

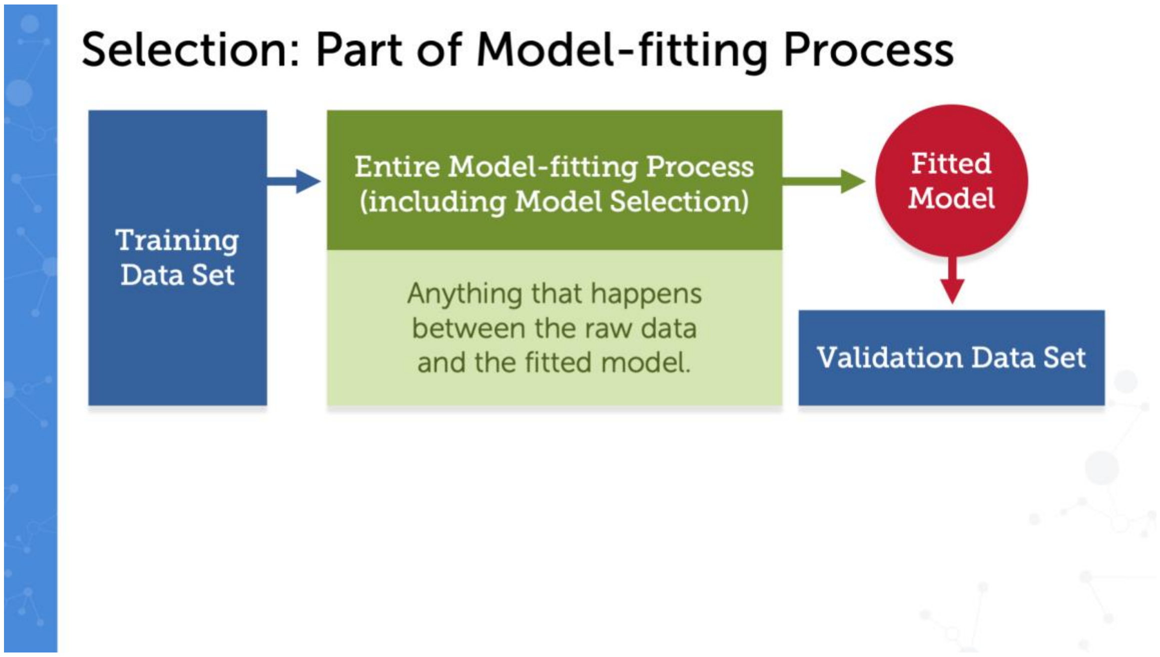
A. Models m = 1, 2, ... M from which to select.

B. **Select the "best" model in terms of predictive ability – this has typically applied cross-validation for model selection:** Split data randomly into k sets. Fit each model on all k training sets, predicting data in corresponding testing set and compute $CV_{(k)}$.

C. Select model which minimizes $CV_{(k)}$ as "best" – fit this selected model on **all** the data for final fitted model.

As we reviewed and expanded upon in the first presentation of this lesson eight, one of the possible uses of cross validation is for model selection. However, if we opt to use cross validation for model selection, that necessarily becomes part of the modeling process for choosing among the potential models.

Thus, it should be incorporated as part of the way we reach the final fitted model, and must be included in that full modeling process.

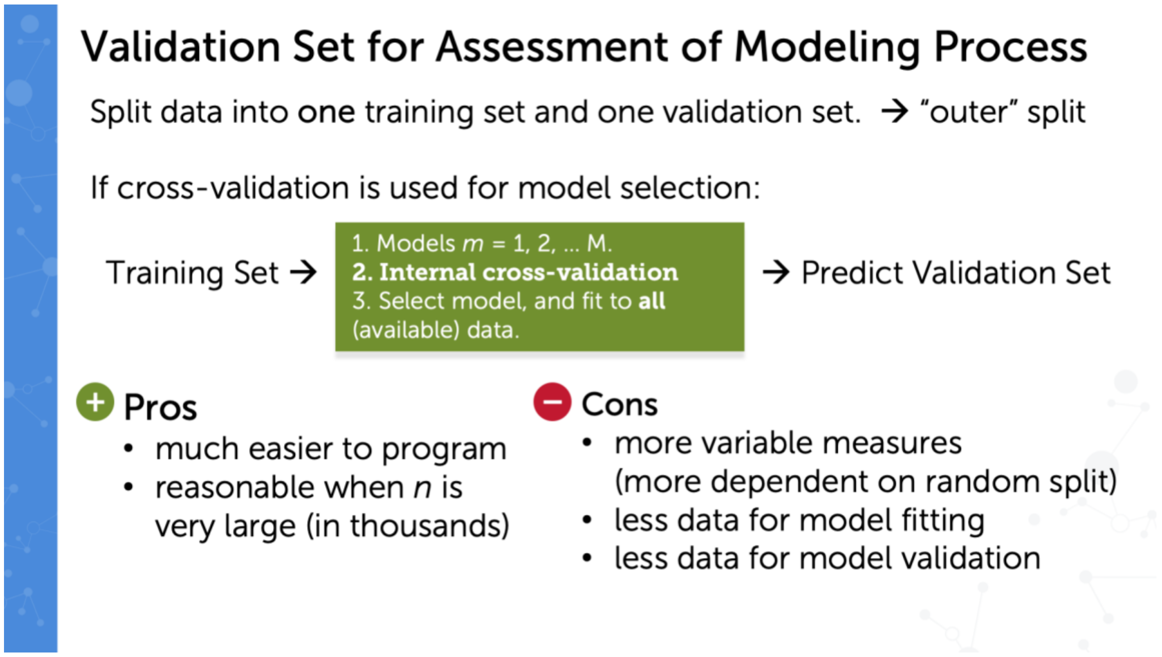


Selection: Part of Model-fitting Process

Training Data Set
Entire Model-fitting Process (including Model Selection)
Anything that happens between the raw data and the fitted model.
Fitted Model
Validation Data Set

So what is the meaning of the entire model fitting process? It literally includes all steps between the start, with the usable data, set until we get to the form of the final fitted model. Note that in addition to actually fitting one of our introduced models, this can include steps that aren't as easily recognizable as part of the model fitting process, such as parameter estimation or selection among several models.

Since all these steps can attune the fitted model to the training data, we must take care to incorporate those within the process. We would then apply the model that has been fit on the training data to some sort of validation set for assessment.



Validation Set for Assessment of Modeling Process

Split data into **one** training set and one validation set. → "outer" split

If cross-validation is used for model selection:

Training Set → 1. Models m = 1, 2, ... M. **2. Internal cross-validation** 3. Select model, and fit to **all** (available) data. → Predict Validation Set

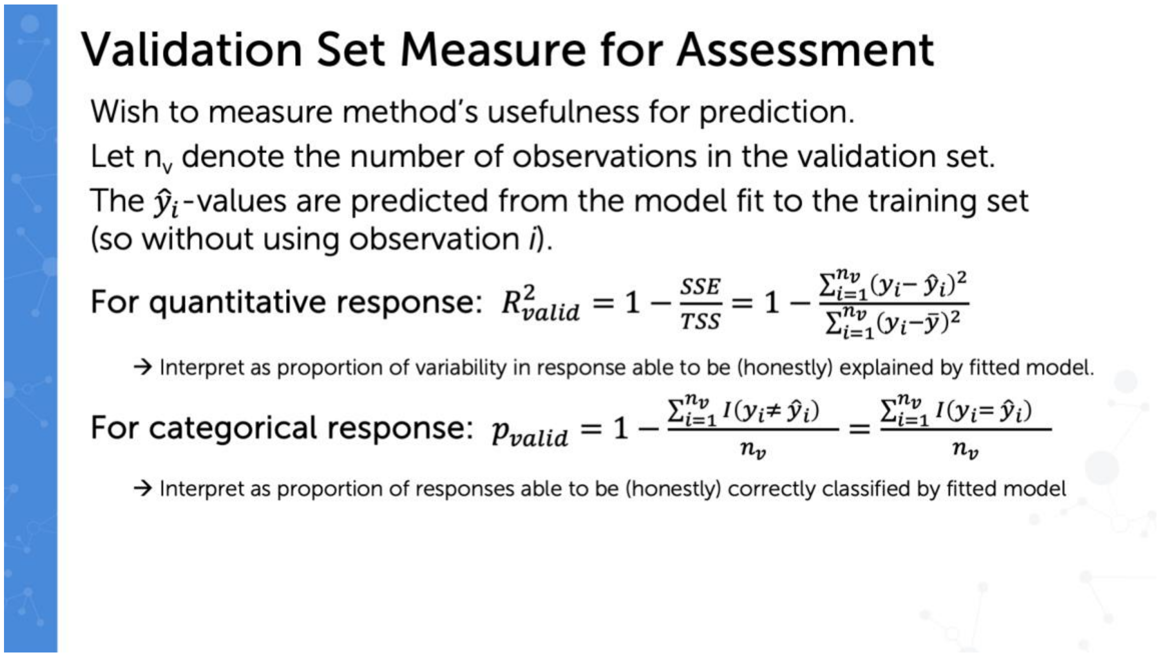
Pros

- much easier to program
- reasonable when n is very large (in thousands)

Cons

- more variable measures (more dependent on random split)
- less data for model fitting
- less data for model validation

If we have a really large data set, an easy to implement alternative for assessment is to use just one split of the data into a training set and a validation set. This turns out to be easier to program, but it also has the downsides of more variable measures for assessment due to fewer data used for validating the entire modeling process. Also less data is available for model fitting, which results in not as accurate of a fitted model. However, a very large data set can typically mitigate the impacts of these issues.



Validation Set Measure for Assessment

Wish to measure method's usefulness for prediction.

Let n_v denote the number of observations in the validation set.

The $\hat{y}_i$-values are predicted from the model fit to the training set (so without using observation *i*).

For quantitative response: $R^2_{valid} = 1 - \frac{SSE}{TSS} = 1 - \frac{\sum_{i=1}^{n_v}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_v}(y_i - \bar{y})^2}$

→ Interpret as proportion of variability in response able to be (honestly) explained by fitted model.

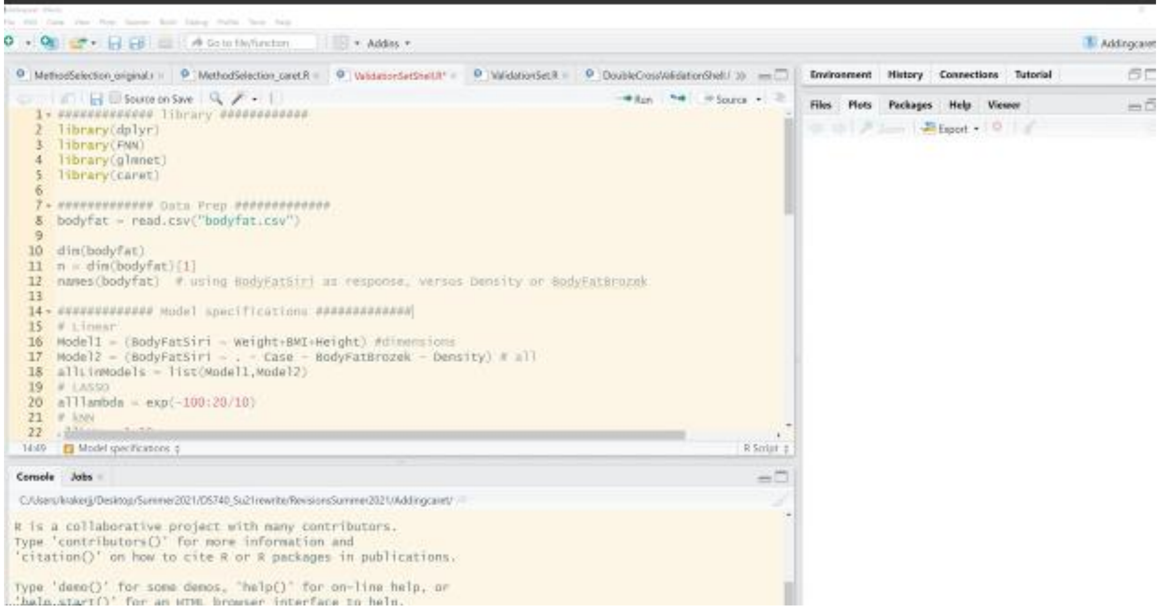
For categorical response: $p_{valid} = 1 - \frac{\sum_{i=1}^{n_v} I(y_i \neq \hat{y}_i)}{n_v} = \frac{\sum_{i=1}^{n_v} I(y_i = \hat{y}_i)}{n_v}$

→ Interpret as proportion of responses able to be (honestly) correctly classified by fitted model

The type of measure that we use to assess a modeling process' ability to predict new data depends on the type of response variable that we have. If we have a numeric response variable, we use errors. If we have a categorical response variable, we use classification.

For a quantitative response, we consider the sum of squared errors as a summary of the unexplained variability in predictions of new data. We then take this as a proportion out of the total sum of squared response values and subtracting this from one gives the proportion of variability in response that is able to be honestly explained by the fitted model. For a categorical response, the measure is even simpler. We take the proportion of responses that are honestly correctly classified by the fitted model.

DS740 - "Double" Cross-Validation



```
############## library ############
library(dplyr)
library(FNN)
library(glmnet)
library(caret)

############## Data Prep ##############
bodyfat = read.csv("bodyfat.csv")

dim(bodyfat)
n = dim(bodyfat)[1]
names(bodyfat)  # using BodyFatSiri as response, versus Density or BodyFatBrozek

############# Model specifications #############
# Linear
Model1 = (BodyFatSiri ~ Weight+BMI+Height) #dimensions
Model2 = (BodyFatSiri ~ . - Case - BodyFatBrozek - Density) # all
allLinModels = list(Model1,Model2)
# LASSO
alllambda = exp(-100:20/10)
# kNN
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
```

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

As discussed on the previous slide, we would like to get an assessment measure based on the prediction of truly new data in a validation set.

In order to follow this process, we will begin with the same data opening process and model specifications that we did in the multiple methods selection in the first presentation of this lesson. We'll talk through a couple items as we take a look at the process for model selection to be added in to this validation assessment.

So we'll begin by running through our list of necessary libraries by making sure we're pointed to the correct direction and reading in the data and appropriate setup as well as the model specifications that were used in specifying our linear lasso and k-nearest neighbors models previously.

To conduct the validation set assessment of the entire modeling process, we need to split the data into our pieces. And so I begin by specifying n.train, which is the number of observations used in the training set, and then the remainder will be n.valid.

We're going to use approximately 2/3 of the data in the training set and approximately one third in the validation set. We'll set a seed and then which train specifies which of the original set of n observations, where n was 250. Which of those are going to be used in our training set?

In order to stay consistent with prior specifications of subsets, we'll make this into a true/false vector called include.train, which specifies either a true/false that is a logical vector of the whichtrain elements occurring in the list one up to n and, similarly, an include.valid the true/false vector.

Since we have just the one split into training and validation sets, our train data will be the rows that are true from our include.train true/false vector. And so if we just want to double-check down here, and I always recommend checking dimensions to make sure we're staying consistent. Yes, it has the 168 rows that we intended for our training set.

And because we may need these as well, we'll have our training x, our training y, and then we have to get the subset for our validation data. And so that takes the rows that are true in the include.valid logical vector. And specify the valid x and the valid y vectors or matrices, as well.

Now, in order to run our process, we're going to be using-- that is, the data being used is just going to be your training data. And so data used is applied to our training data only. That is, our valid data will be held out entirely from this model selection process.

And if we go back to our model selection process, which we're going to pick from our caret application back in the first presentation of this lesson, we're going to go in to the section in which we set up the training method, set up our different cross-validations, and then took a look at the best possible model.

In order to actually fully go through the model selection process, we'll have to go through all of this. And so I am going to pull all this information, including getting the one_best_Type, the one_best_Pars, and the one_best_Model over to this validation set shell. And we won't need a one_best_Order out of this.

So what I did in this process was literally everything that I did in my prior method selection with caret. The only thing that really was updated was that the data being used was only the training data set.

And now I'm going to have to be a little bit careful as to what eventually comes out of this. That is, I'm going to have to be a little bit careful once I get the one_best_Type, one_best_Pars, and one_best_Model. I need to make sure that my predictions are being done with the proper method.

And in many cases, we can use the predict function, which will then identify the type of model that has been fit and ultimately selected as the best model. That is, it will be able to identify if it's a linear model, if it is a penalized regression model, if it is some sort of tree model. Whatever type of model it is can often be used with the appropriate inputs to make a prediction.

It doesn't work quite so directly with the kNN simply because we have to take the validation set and predict observations in the validation set according to the data that was used to fit the one_best_Model. And so, using some piping notation, which requires that we have opened up the dplyr library, we are able to smoothly come up with predicted values by taking the one best-- in this case, it would be the k-nearest neighbors if that turns out to be the one_best_Type. We could take that one_best_Mode and pipe it through to predict our our validation data.

It will turn out, in this case, that is not, in fact, the best model. And then we do some wrap-up computations-- numeric measures, in this case-- to complete our assessment. So I've combined all of this for you in the provided data set called validationset.r. That is, I took the shell-- what was the original shell-- put in the relevant inner model selection and then brought out the best-fitting model from that.

So again, we can run our setup, as we've done previously. We can make sure that we have split the data, as we did previously. Data used equals training data. Now I'm going to use the-- I'm going to reset the seed at 82 before specifying the training method-- cv number 10.

And then I'm going to run my various train functions applied just to the training data set according to the method we specified, which is cross-validation with 10 folds. Again, this is on the inner process being used for model selection. So that one will run. Next, do that for linear model two and then for lasso.

And when we get to lasso, we get a little bit of a question mark here. And that question mark has to do-- if we take a look at the output. It has to do with the fact that basically we have-- we're looking at hyperparameters that go outside the bounds of what the model is able to fit.

So what this means is, if we take a look back all the way up to the very, very top of this, the lambda, as high as we specified, is not allowing us to fit all of our splits on the data on this inner cross-validation. So what that simply means is we're going to re-specify just using a few fewer lambda values.

So I'm only going to run my lambda across 116 values. Still a lot and still will allow us to pick out an optimal model. But when we repeat this process, we now will not get a warning. And the reason we don't get a warning is because it was able to fit across all our inner cross-validation splits. It was able to fit for all lambda and produce measures for all such lambda.

So that's something to keep in mind when you are specifying hyperparameters. We'll similarly run our fit caret kNN. Again, all this is done using data used, which was specified at the beginning to only include our training data.

And finally, at the end of this, we'll do something similar-- in fact, the same thing copied directly over from our method selection via caret. Setting up our list of all best types, all best parameters, all best models, all best RMSE, we're selecting among these best models from these four fits.

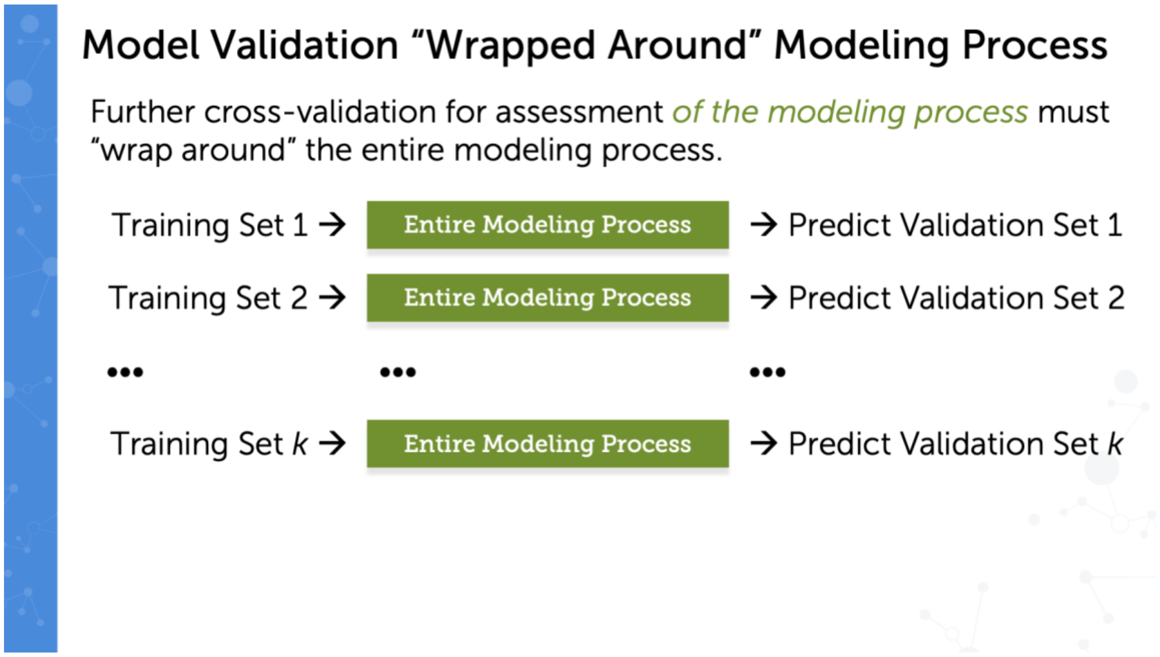
And so if I take a quick look-- and in fact, maybe just take a look at our all_best_RMSE, all best root mean square error, this tells me that the third model, which is my lasso, my best lasso model is going to be my best model. So all best types, all best parameters, all best models, and picking the best of them. So one best type is lasso with parameters alpha of one, of course, and lambda of 0.04076.

And then the model-- if we take a look at this, this is going to include a real ugly specification of the model, as we talked about previously. But the positive of having stored this is we can actually use it for prediction. And so you might ask, well, if we just have to come out of this process with selecting one model and using that for prediction, why don't we just observe that, in fact, the one best type is lasso and run only these two lines?

And that would be accurate for validation set. That is, we could simply run those two lines and get our predicted values-- that is, our predicted y's-- only for the validation set. However, what this allows us to do is, if we did a new split of the data-- and this is looking ahead a little bit. If we did a new split of the data, it would allow us to run through and check, depending on the best type, how we should make the predictions.

All right, so our final step here is to take a look at how these predicted values from the validation set-- if our actual observed y's line up reasonably well. Reasonably well is accurate. We can take a look at the root mean square measure coming out of this, except that I have to fix this to say valid y.

Or, more directly, we can get our r squared analogy, which will allow us to describe the fit of the model. So I would add something like this at the end of the description-- about 70% of variability and truly new predictions is explained by this modeling process. That is, we are able to get an honest assessment of the predictive power of this modeling process.



Model Validation "Wrapped Around" Modeling Process

Further cross-validation for assessment *of the modeling process* must "wrap around" the entire modeling process.

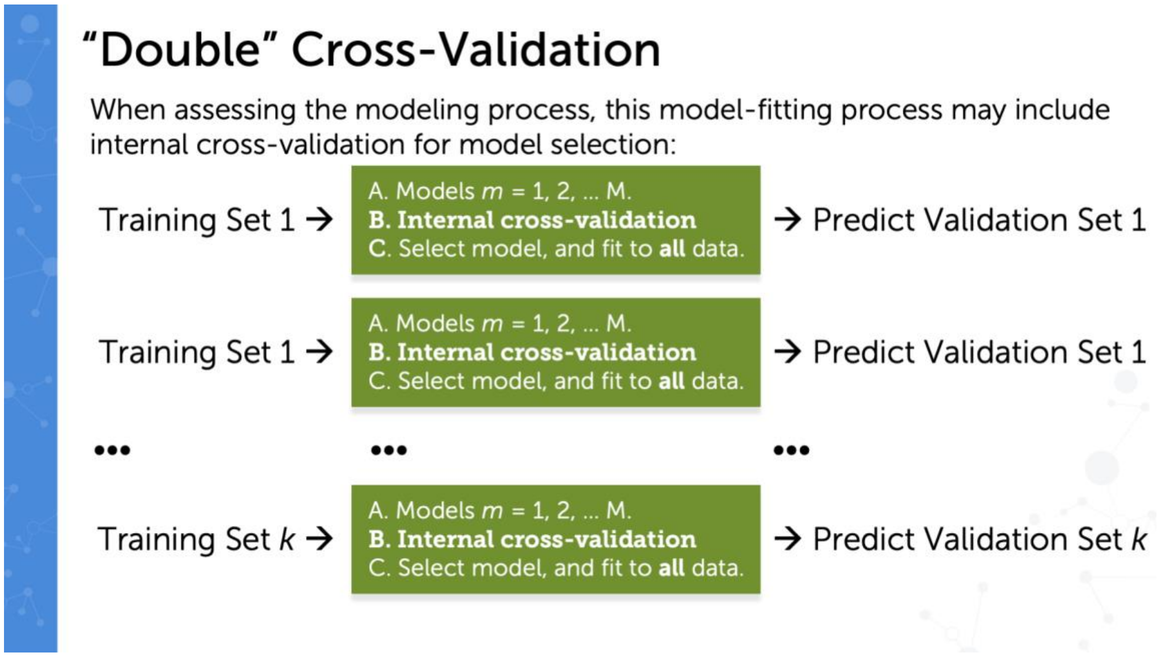
Training Set 1 → Entire Modeling Process → Predict Validation Set 1

Training Set 2 → Entire Modeling Process → Predict Validation Set 2

••• ••• •••

Training Set *k* → Entire Modeling Process → Predict Validation Set *k*

We will be discussing how to practically apply cross-validation for validation around the entire modeling process. Along with this, we will introduce numeric measures for the methods usefulness for prediction. The key here is for every training set on which we fit a model, we must implement the entire process at each split. This means that we implement the entire modeling process k times for the k splits of the data into training and testing sets for the outer cross-validation.



"Double" Cross-Validation

When assessing the modeling process, this model-fitting process may include internal cross-validation for model selection:

Training Set 1 → A. Models *m* = 1, 2, ... M. **B. Internal cross-validation** C. Select model, and fit to **all** data. → Predict Validation Set 1

Training Set 1 → A. Models *m* = 1, 2, ... M. **B. Internal cross-validation** C. Select model, and fit to **all** data. → Predict Validation Set 1

••• ••• •••

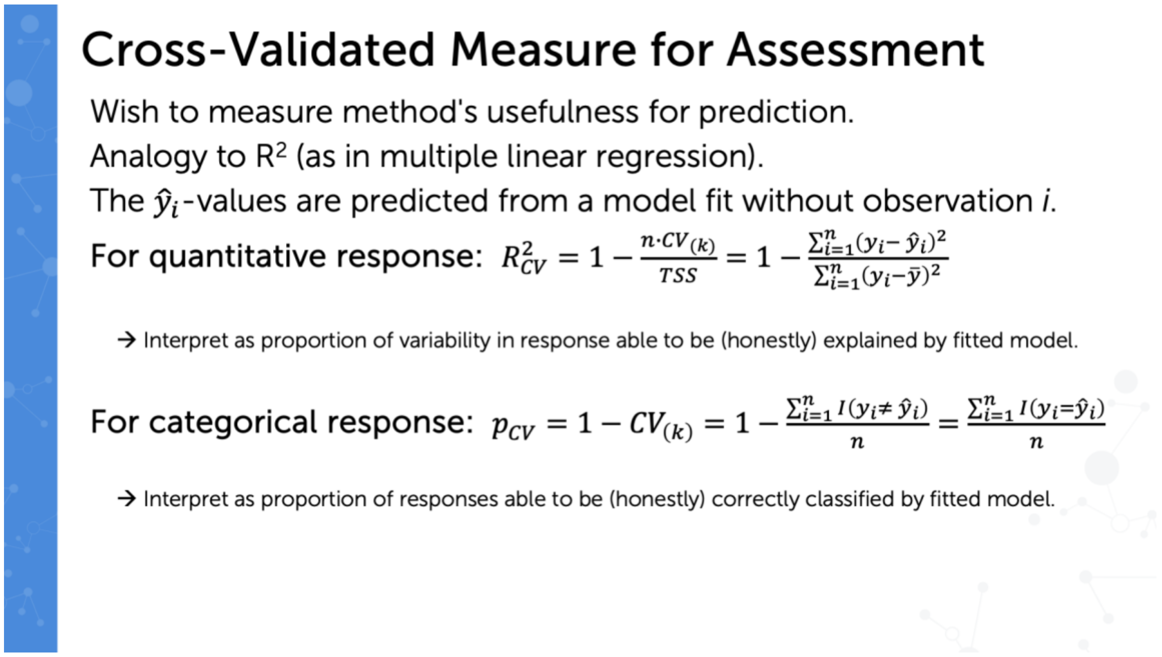
Training Set *k* → A. Models *m* = 1, 2, ... M. **B. Internal cross-validation** C. Select model, and fit to **all** data. → Predict Validation Set *k*

Note that the modeling process may include an inner cross validation used for model selection, as revisited earlier in this lesson. This means that if we use cross validation, such as with caret for model selection, and then further wish to use cross validation for assessment of this entire modeling process, we must place the cross validation for model selection that is the inner cross validation as part of that model fitting process.

Around that entire modeling process, we wrap a so-to-speak outer cross validation for assessment of that modeling process. Thus is pictured here when we use the training set one to fit the model. The process that we go through to get that fitted model includes the inner cross validation.

We then use this produced fitted model to predict validation set one, and we continue on in that process, applying the full modeling process to fit the model for each training set of the case splits of the data, and then predict each validation set in turn.

While this concept is reasonably simple, the application can be a bit tricky to implement in programming, so we spend some time examining that on the next few slides.



Cross-Validated Measure for Assessment

Wish to measure method's usefulness for prediction.

Analogy to R^2 (as in multiple linear regression).

The $\hat{y}_i$-values are predicted from a model fit without observation *i*.

For quantitative response: $R^2_{CV} = 1 - \frac{n \cdot CV_{(k)}}{TSS} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

→ Interpret as proportion of variability in response able to be (honestly) explained by fitted model.

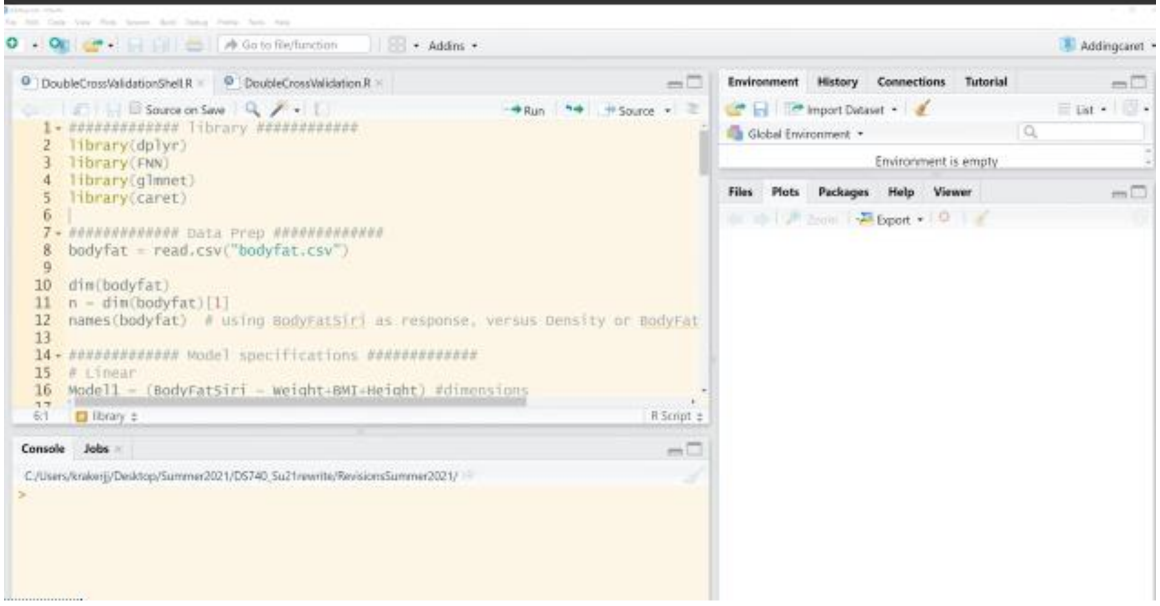
For categorical response: $p_{CV} = 1 - CV_{(k)} = 1 - \frac{\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)}{n} = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n}$

→ Interpret as proportion of responses able to be (honestly) correctly classified by fitted model.

If we are planning to use cross-validation to assess the usefulness of the modeling process for predicting new data, how do we quantify this? Recall that the CV sub k measure is our cross-validated analogy to mean square error. As we did with the validation set measure, we can extend this to an analogy of the r squared statistic for numeric data or a proportion for categorical response. For numeric responses, interpretations are similar to those from before. We consider the R squared CV measure to be interpreted as the proportion of variability in the response that was explained in an honest way by the fitted model.

And for a categorical response, we use the PCV measure interpreted as the proportion of responses that are correctly classified in an honest way by the fitted model. We can then use these measures to discuss the usefulness of the modeling process for predicting truly new data.

DS740 - "Double" Cross-Validation


```
############# library ############
library(dplyr)
library(FNN)
library(glmnet)
library(caret)

############# Data Prep #############
bodyfat = read.csv("bodyfat.csv")

dim(bodyfat)
n = dim(bodyfat)[1]
names(bodyfat)  # using BodyFatSiri as response, versus Density or BodyFat

############# Model specifications #############
# Linear
Model1 = (BodyFatSiri ~ Weight+BMI+Height) #dimensions
```

```
C:/Users/krakerjj/Desktop/Summer2021/DS740_Su21rewrite/RevisionsSummer2021/
>
```

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

We now use an outer layer of cross-validation. That is splitting into folds with a training and validation set to apply our prior process only across multiple folds. So our setup is going to be very similar to what we did with the validation set. And I'll start by talking through the basic shell.

I've additionally opened our MethodSelection_caret to be put inside that shell. So with our shell, we're going to be opening up libraries, reading in our data frame, setting up our models. All is done previously. And then we're going to be doing something very similar to a validation set split, only with multiple splits.

And I'm going to do so with 10 splits or folds of the data. As I've done before, I'm going to to set up my groups and randomize the order to get my cvgroups. I'll then take the set up storage for the allpredictedCV. And this is going to be coming from this outer split at each inner application I'm going to be predicting my validation set, which is going to be a different set at each split.

And I'll also want to keep track of the bestTypes and bestPars. Not that we're going to be directly using those, but it can be a useful follow-up to keep track of those and take a look for consistency. Finally, I'm going to be looping throughout our splits. Now this is going to be effectively what we've done before at a-- with a one layer cross-validation.

So we're going to go for j in 1 up to nfolds. Now I'm using j as a tracker. Because we're thinking about this as an outer fold, not as our inner fold. So I want to use a different index, j not i. I'm going to specify my group, traindata, trainx, trainy, as those observations that are not in group j.

I'm only using the traindata. Validdata is the observations that are in group j as we move through our different groupings. j equals 1 up to 10. We're then going to take-- and this is an extremely important step. The data we're going to be using are the traindata.

So what we're doing is-- we normally would in cross-validation, is we split up the data. Then we take the traindata and we fit our model-fitting process. Well, the unique thing here is that this entire model-fitting process is going to include an inner cross-validation via caret, as we saw previously.

And so we're going to pull that process into this spot where it says, INCLUDING ALL CONSIDERED MODELS. And then output will be the best_Type, best_Pars best_Model, and not the best_Order. We do not need that item in this application.

And then we're going to check at each split-- each of those in our example, we'll be doing 10 splits or folds of the data. We're going to get a best type with a certain bestPars. We're going to keep track of that from this inner model-fitting process.

Then as we did with the validation set application, we'll check. If it's linear, we make our predictions based on the linear model and using the output in the appropriate way. If it's less so, we'll make the predictions using the output in the appropriate way.

And finally, if we're using kNN, we actually have to use a dplyr function and take the one_best_Model, piping it through a prediction of the valid data. So it's a little bit different of an application. And so I wanted to make sure that you had a syntax example of how to make the prediction using a kNN model fit output from caret.

And if we want to, we're going to display which models were best within each loop. But our primary-- and this is why it lands right here at the end. Our primary focus is to get a measure of assessment. And of course, we can get a usual RMSE or MSE. But primarily, we're going to be looking for that R squared, that honest measure of the percent of variability in the response that's explained by our model-fitting process.

So as we did with the validation set, we're just going to pop into this space, the model-fitting process. Now we've got all our definitions made. We have our data used, specified. Again, the important thing is inside this outer cross-validation loop, we're going to only use the training data to fit the modeling process.

So we'll take this entire set of information and we'll just pop that in here. And if we've called upon things carefully-- for example, if we've set this up so that the dataused-- or data referenced is dataused, which is the traindata in this case-- we'll be properly applying the inner cross-validation via our caret functions, such as train, to this-- only the training data from the outer split.

So I've done that in our DoubleCrossValidation.R. And the one thing I will mention, in addition to a setup-- and I do need to include the setup here, set up the training method. In addition to that setup of the training method, and the running of the different fits, and identification of the output, I have also included a visualization of the output for each inner split.

Now this visualization is optional. That is, I'm going through and I'm tracking. At each of the outer splits, when I apply the inner cross-validation, I am keeping track of all the results, all the measures from the inner cross-validation, and plotting those.

This is not a necessary part. And so if you find this plotting part to be unuseful, you can remove that part. But I do find it visually helpful as I'm running through this. So we will do all our setup, assuming I'm looking at the right location. So this is a repeat of the setup that we've seen a couple of times now.

I do want to talk through this outer cross-validation split, with the focus being on assessment. So folds. We set up our cvgroups. Of course, we could take a look at our cvgroups, which are going to be a random ordering of the values from up here, the 1 through 10 values.

We're going to set up our storage for our cross-- our predicted values, as well as storage for the bestTypes and bestPars coming out of each of the folds. And we set up our training method. And so I apologize. I did not have this in here originally because we have set that up externally.

Now should you do this externally or internally, it should not matter in terms of getting an actual validation split. But if you want to mimic precisely what we did in our inner split, you could leave that. Instead of on the outer split, you could leave that on the inner split.

So if we leave that on the inner split, we'll take this. And you can run this outer split if you would like to. Well, I often have to do some error checking. And so I might wish to set j equals 1 and run through some of these commands, just using the first outer split of the data.

And as a general check of whether my full process is working correctly, this is pretty good practice to make sure I'm not running into any difficulties on this inner split or on this inner application. So for my first split of the data, I got a one_best_Model. That looks like it was of Type LASSO with parameter-- well, of course, alpha equals 1. And lambda equals 0.165.

I would then take-- store that best type from my first outer split and store it among my old bestType storage, and also store the parameters as location j in my list. So this specifies a list location, allbestPars. So if I take a look at allbestTypes, I've got LASSO for my first and allbestPars. I have the parameters for my Lasso at that first split or coming out of that first split.

And then I'm going to run this if to get an application of these two lines. So in other words, if I took a look at allpredicted values, just the groupj locations, allpredicted-- excuse me-- CV, I should get the same values as if I simply ran the prediction for my lambda model.

So I'm storing those values just in the locations where groupj is true. So I'm storing the predictions at their proper locations. So now I want to run through the full loop of these outer splits. And what you'll see if you keep an eye over on the right, what you'll see is a new plot each time I run through a different split.

And what this plot is tracking is the RMSE for the different app-- different splits. For the different outer splits, it's tracking the RMSEs for the inner cross-validation. And so you'll see in this case, the linear model is actually best. And that's why there was a red line. And the next split, in the next outer split, the inner cross-validation, applied via caret, had a Lasso as the best model.

And so I do like keeping these visuals in there, simply because they allow me both to keep track of the processing of my outer split, how fast is this actually going, and helps me visualize what is the outcome of the inner cross-validation. And again, it's just a double check, that I'm tracking things correctly.

So I believe-- oh, not quite. So I believe that we've tracked two outer splits, where the best model chosen by the inner cross-validation is linear. So when I type in allbestTypes, I should get two of them having been linear. But most of the splits in that outer cross-validation resulted in a Lasso being chosen by the inner cross-validation.

I can also take a look at all the bestPars. Or if we wanted to take a look at this a little bit more nicely, we could print out-- not super visually appealing. We could do some rounding or other text-based applications to make this look a little bit prettier.

But it does allow me to see the parameterizations and the two linear models that were picked for the full linear models, which is not surprising. The subset linear model does not seem to work very well. Again, though, this double or outer cross-validation was not intended for selection.

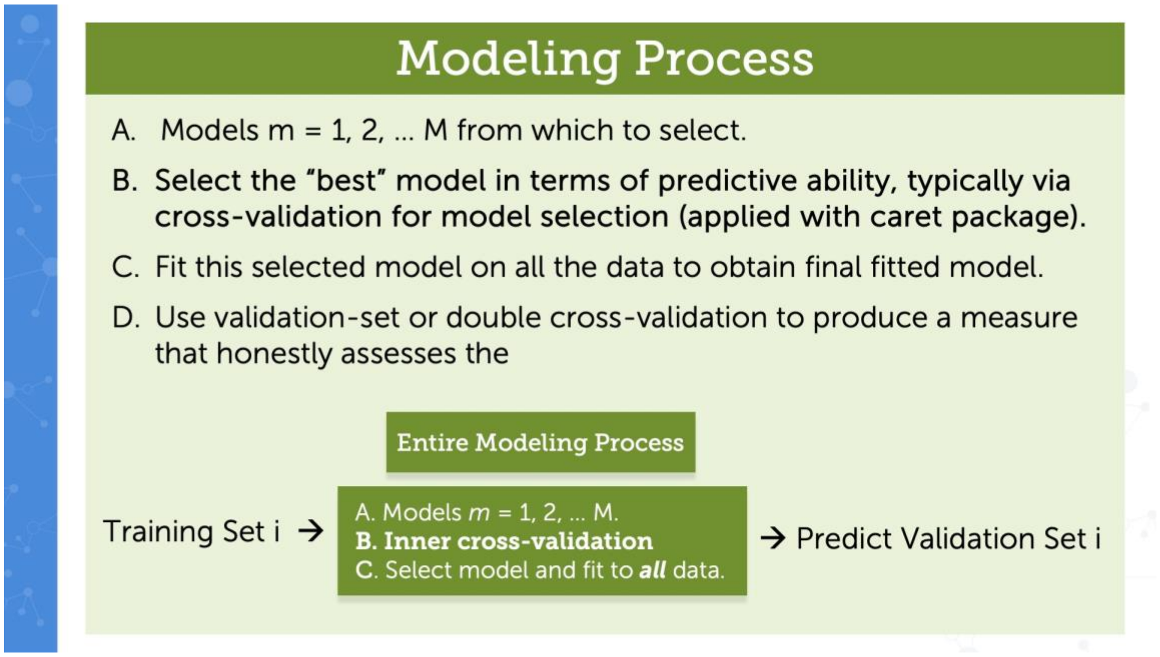
The purpose for this outer cross-validation was simply-- and it seems a little bit over the top. But really, we're trying to assess this process. And so that's our focus here. That's why this is the part we need to end on, should we get our response-- or true values and the response.

We've got our RMSE measure from the cross-validation predictions from that outer loop, which used an inner cross-validation in the model selection process applied to the training data. And so this finally results in an R squared.

And this R squared is not too dissimilar from what we saw with the validation set approach. That's not surprising. But because we were able to apply this to all the data, we have a little bit more data on which to base an accurate measure of R squared.

And so this tells us that about 68.5% of the variability in BodyFatSiri values is explained by this model-fitting process, where that model-fitting process encompasses everything we did over here. That is setting up the training set, running through the fit_caret on the linear model, fit_caret on linear model 2, fit_caret on LASSO, and ultimately picking the best model from that.

That whole process is the modeling process that we popped inside here and then assessed via this OUTER shell, this outer cross-validation, to get an accurate assessment measure. It's a big concept. And it will take some practice. Thanks for following along with me. And I hope that you are doing very well.



Modeling Process

A. Models m = 1, 2, … M from which to select.

B. Select the "best" model in terms of predictive ability, typically via cross-validation for model selection (applied with caret package).

C. Fit this selected model on all the data to obtain final fitted model.

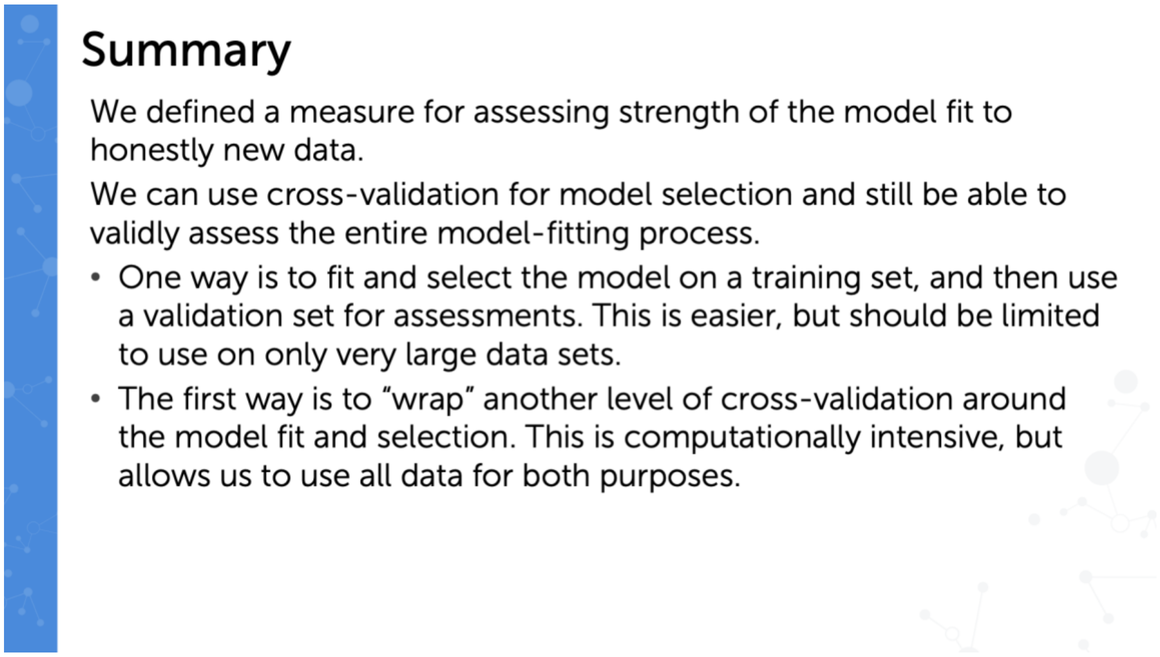
D. Use validation-set or double cross-validation to produce a measure that honestly assesses the

Entire Modeling Process
Training Set i →
A. Models *m* = 1, 2, … M.
B. Inner cross-validation
C. Select model and fit to ***all*** data.
→ Predict Validation Set i

We now summarize our entire approach, including both model selection and assessment of the model fitting process. The model fitting process starts off with a set of potential models, and typically includes a one level or inner cross validation for model selection, as introduced earlier in this course.

This is then used to select a model, which is considered to be the best model by the appropriate criterion, and we fit this model on all available data. We next take this entire model fitting process and assess it with an outer split of the data.

This can use an outer layer via validation set, or a cross validation assessment wrapped around the entire modeling process. If we use cross validation in this outer assessment, we call this double cross validation.



Summary

We defined a measure for assessing strength of the model fit to honestly new data.

We can use cross-validation for model selection and still be able to validly assess the entire model-fitting process.

- One way is to fit and select the model on a training set, and then use a validation set for assessments. This is easier, but should be limited to use on only very large data sets.
- The first way is to “wrap” another level of cross-validation around the model fit and selection. This is computationally intensive, but allows us to use all data for both purposes.