Text And Prediction, Part 3

Josh Clinton

2022-07-12

Who is Publius?

There are 11 essays for which the identity of the author is disputed How do we tell who wrote what? Authorship identification is a task in *forensic linguistics*.

To determine authorship we ideally need a source of textual information that we can use to predict authorship.

1. Unrelated to the topic of a text.
2. Difficult to be strategic with.
3. Idiosyncratic - reveals traits of author.

Using a non-topic is often important because it is not a trait that is typically associated with an individual author. Because anyone can write on any topic there is no way to guarantee that the topic being written about can identify authorship. We saw this in the relationship between our word-based clusters and the authorship from last time. Relatedly, we want to use predictors that are hard to be strategic with. Features that are easy to mimic – such as the topic being written about – are not useful because even if they perfectly predict past behavior it is relatively easy for people to disguise their identity.

So what we want to do is to use traits that are associated with an author (or speaker if we are analyzing spoken words) and which are hard to alter by the author – so that the pattern is likely to endure across the texts being analyzed – and which are also subtle and hard to mimic.

This often involves the use of relatively minor words.For example, how many times did I say "so" and "right" during lecture? I've no idea. Nor have you. Probably too many, but I cannot change that frequency without significant effort because it is a personal (idiosyncratic) speaking habit. When analyzing text, the best predictors can also be found when looking at how authors use words to express themselves. Not the topics, but the words.

Consider, for example, excerpts by Madison and Hamilton and how they differ in their use of written English to express themselves.

Hamilton writing on the militia (Federalist Paper 29) notes:

> The power of regulating the militia and of commanding its services in times of insurrection and invasion are natural incidents to the duties of superintending the common defense and of watching over the internal peace of the confederacy it requires no skill in the science of war to discern that uniformity[. . .]

Madison on the judiciary (Federalist Paper 49):

> The several departments being perfectly co-ordinate by the terms of their common commission, none of them, it is evident, can pretend to an exclusive or superior right of settling the boundaries between their respective powers[. . .]

Madison stops to make the remark "it is evident" whereas Hamilton just barrels ahead.

So our prediction task is to use text to figure how who wrote the contested Federalist papers. The task involves several steps:

1. Identify the dependent variable. What are we predicting?
2. Find predictors. What are we using to predict?
3. Fit the model.

4. Evaluate the model

With text analysis, the hardest steps are often the first two.

Let's load in the data and libraries we need for this unit.

```
library("tidyverse")
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'pillar'
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'hms'
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
library("tidymodels")
```

```
## Warning: package 'recipes' was built under R version 4.1.2
```

```
dtm <- readRDS(file="FederalistPaperDocumentTermMatrix.Rds")
```

Now create the outcome variable – the vector of authors associated with each of the 85 Federalist Papers. We are going to create some variables/vectors associated with the index of each Federalist paper. So, for example, Hamilton wrote papers 1, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17 and so on. John Jay wrote papers 2, 3, 4, 5 and 64 and the papers with contested authorship are 49 through 57, 62 and 63. Then we are going to create a variable called `author` and name the value associated with each position using the name of the author.

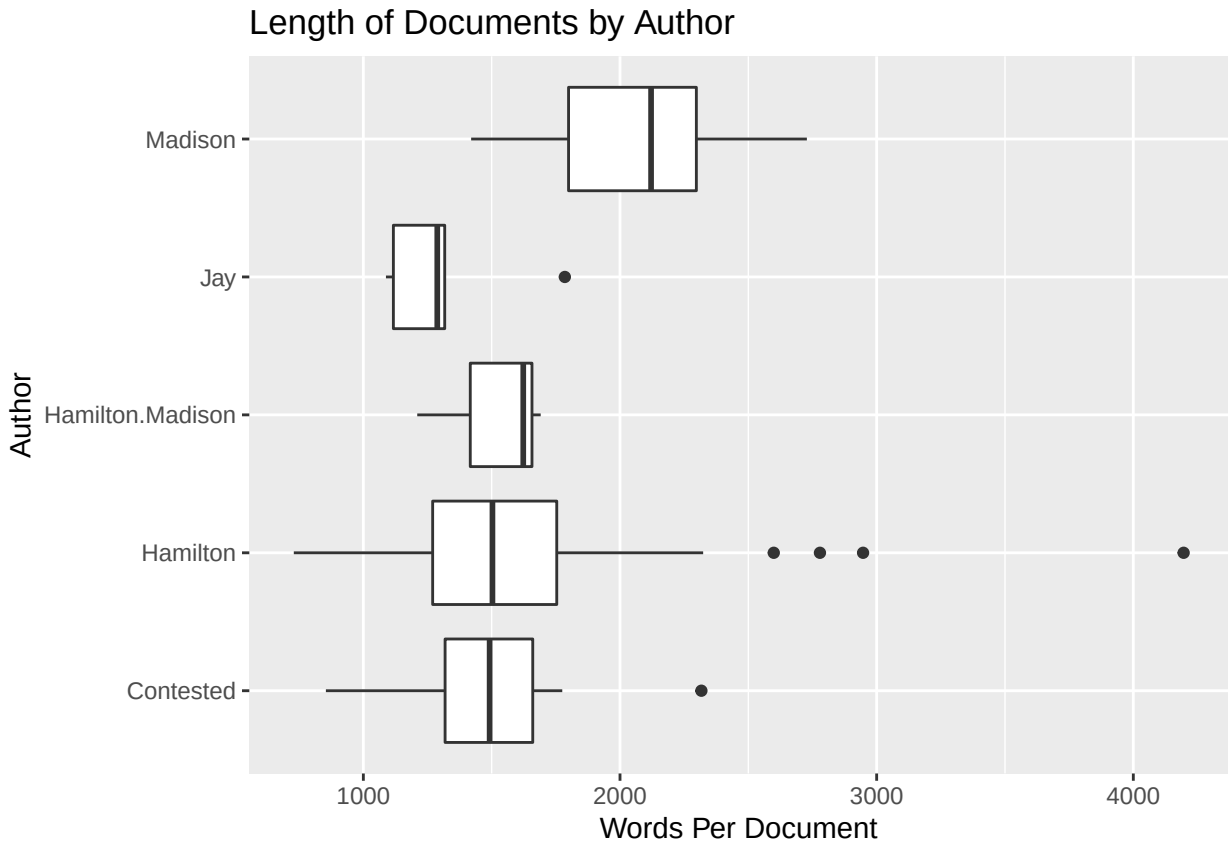
```
Hamilton <- c(1, 6:9, 11:13, 15:17, 21:36, 59:61, 65:85)
Madison <- c(10, 14, 37:48, 58)
Hamilton.Madison <- 18:20
Jay <- c(2:5, 64)
Contested <- c(49:57, 62, 63)

author <- rep(NA, 85)
author[Hamilton] <- 'Hamilton'
author[Madison] <- 'Madison'
author[Hamilton.Madison] <- 'Hamilton.Madison'
author[Jay] <- 'Jay'
author[Contested] <- 'Contested'
```

Classification: The 'bag of words'

But raw word counts are not reliable predictors because some authors may be more verbose than others. Let's consider the relationship between authorship and document length. Start by creating a tibble `worddoc` with 2 variables that combines the authorship with the total number words being used in each.

```
worddoc <- bind_cols(author = author, WordsPerDoc = rowSums(dtm))

worddoc %>%
  ggplot(aes(x = WordsPerDoc , y = author)) +
  geom_boxplot() +
  labs(x= "Words Per Document",
```

```
       y= "Author",
       title = "Length of Documents by Author")
```



How many possible predictors? Lots. Even if we used the use of words in isolation from one another (i.e., no interactions) we would have: 8594 possible predictors because that is how many words we have.

So how might we try to come up with some good predictors? Well lets see how frequently Hamilton and Madison are using certain words. (Note that I am using slightly different syntax then we have used to take a sum of papers written by Hamilton and Madison rather than using `group_by` because the `dtm` object is not a tidyverse format.) `colSums` is a function that takes the sum across rows for each column – so there are as many sums as there are columns. Recall that we are using `[]` to select elements of the `dtm` data matrix – here the documents that are associated with `Hamilton` or `Madison` based on the index we defined in the second code chunk.

```
HamiltonWords <- colSums(dtm[Hamilton,])
MadisonWords <- colSums(dtm[Madison,])
Ratio <- HamiltonWords/MadisonWords
Total <- HamiltonWords + MadisonWords

worddoc <- bind_cols(words = names(dtm[1,]), Ratio = Ratio, Total = Total)
```

Let's look for good predictors based on which "common" words are used more by Hamilton or Madison. For Hamilton, these are words with a high ratio. For Madison, just the opposite.

Let's start by looking at words used frequently.

```
worddoc %>%
  filter(Total > 10 & (Ratio > 5 | Ratio < 1)) %>%
  arrange(-Total)
```

```
## # A tibble: 440 x 3
##    words        Ratio Total
##    <chr>        <dbl> <dbl>
##  1 would        5.39   1092
##  2 there       10.8     412
##  3 upon        52.9     377
##  4 powers       0.758   225
##  5 between      5.21    205
##  6 men          6.83    180
##  7 legislative  0.942   167
##  8 courts      28.8     119
##  9 man          5.88    117
## 10 president    7.15    106
## # ... with 430 more rows
```

And/or by words with the largest ratio.

```
worddoc %>%
  filter(Total > 10 & (Ratio > 5 | Ratio < 1) & Ratio != "Inf") %>%
  arrange(Ratio)
```

```
## # A tibble: 388 x 3
##    words         Ratio Total
##    <chr>         <dbl> <dbl>
##  1 composing    0.0833    13
##  2 whilst       0.0833    13
##  3 coin         0.133     17
##  4 term         0.176     20
##  5 consequently 0.308     17
##  6 department   0.333     76
##  7 enumeration  0.364     15
##  8 address      0.375     11
##  9 ratification 0.375     11
## 10 again        0.385     18
## # ... with 378 more rows
```

So maybe let's try some that favor Hamilton and some that favor Madison. Note that there is some logic behind what we are doing, but we are not using a statistical "feature selection" model. (Why not? With 8594 predictors and 51 observations there are many specifications that will perfectly predict our outcome of interest. There are ways of dealing with this, but it is something that will be covered in DSCI 3100 and beyond.)

```
words <- c('would', 'there', 'upon', 'whilst')
```

Another set of words could be the following...

```
words2 <- c('president','department','courts','coin')
```

How else could you define another set of words?

```
# words3 <- c()
```

So how are these words used by these authors?

One hiccup is that `dtm` is a document-term-matrix that was created by `tm` and it is not in the `tidytext` universe so to manipulate it we need to use some functions available in base R. Again, base R should not be thought of as "basic", it is actually a very powerful set of tools that some prefer to the `tidy` framework (because it has close connections to C, it is far more stable, it is less reliant on (seemingly-ever-changing)

functions, etc.)

Using the indexing available in R, we can take the sum of the papers written by Hamilton over the columns containing the words defined by the object `words`. `colSums` is a function that takes the sum across rows for each column – so there are as many sums as there are columns. Recall that we are using `[]` to select elements of the `dtm` data matrix.

```
colSums(dtm[Hamilton,words])
```

```
##  would  there   upon whilst
##    921    377    370      1
```

```
colSums(dtm[Madison,words])
```

```
##  would  there   upon whilst
##    171     35      7     12
```

Now do the same for `words2`. What do you observe?

```
# INSERT CODE HERE
```

Since this may depend on how many words are being used, lets turn word these counts into the relative frequency with which each word is used in each documnt (so the count over the sum total number of words in each document) and then calulate that rates per 1000 words.

```
dtm <- dtm / rowSums(dtm) * 1000
```

Now let's take the average word rate for each word for Hamilton and Madison using the indexing and `colMeans` instead of `colSums`.

```
colMeans(dtm[Hamilton,words])
```

```
##        would        there         upon       whilst
## 11.441018746  4.417750184  4.398682789  0.007055719
```

```
colMeans(dtm[Madison,words])
```

```
##     would     there      upon    whilst
## 5.4589543 1.1132520 0.2000269 0.3801131
```

Our dependent variable is the identity of the author (if we know it):

```
table(author)
```

```
## author
##        Contested         Hamilton Hamilton.Madison              Jay
##               11               51                3                5
##          Madison
##               15
```

So let's organize the data for analysis by creating a new data frame that has the number of the Federalist Paper (`number`), the author (`author`), and the word frequency for the words listed in the `words` object. We are going to combined these variables in a new data.frame object we will call `dat`.

```
dat <- data.frame(number=1:nrow(dtm),
                  author=author,
                  dtm[, c(words,words2)])
```

What does this look like? Let's take a look at the first row of data....

```
dat[1,]
```

```
##          number   author    would    there     upon whilst president department
```

```
## fp01.txt      1 Hamilton 1.682086 1.682086 5.046257      0         0          0
##           courts coin
## fp01.txt      0    0
```

Building on the last unit we want to build a model using a training dataset and then apply it to a test dataset. In supervised learning we are using the training dataset to develop a prediction model that we can use to "learn" about the testing data based on the relationships we find in the training data. When we are working with the training dataset we are making in-sample predictions (often called fitted values).

In this application rather than randomly generating a training and testing dataset using the `split` function we are going to create the split using the fact that the authorship is known for some cases and unknown for others. In particular, our training dataset is the set of observations for which the author is known and the testing dataset is the dataset for which the author is unknown.

Now could we also use cross-validation techniques applied to the set of observations for which the author is known? Certainly. But for now let us put that aside.

So start by defining the training dataset `train.dat` using the set of Federalist Papers with known authorship and defining a `score` that we are going to predict. Note that `author` is a character variable and we want to create a numeric variable that we can predict using the regression model we are going to develop. The variable we are goign to predict (our dependent variable) is called `score` and it is going to take on the value of `1` if the paper was written by Hamilton and `-1` if written by Madison. This is useful because it means that postive values are associated with Hamilton and negative values are associated with Madison.

```
train.dat <- dat %>%
      filter(author=='Madison' | author=='Hamilton') %>%
      mutate(score = ifelse(author=='Hamilton',1, -1))
```

So now let us create the testing data by selecting the papers with an unknown author.

```
test.dat <- dat %>%
      filter(author=='Contested')
```

So now we have our testing and training dataset we need to develop the model we are going to use to predict the `score` in the training dataset to generate a predicted score for papers contained in the testing data.

To show you an alternative way of implementing linear regression in R, we are going to depart from the workflow that we discussed in the last unit to show you the steps in "base R" because the proces is simpler given what we need to do.

We are also going to abstract away from the question of "feature selection" – i.e., which variables to use — and instead use a model containing just 4 of the 8594 possible word rates to predict `score` based on the principles we discussed above. (The code required to do the more involved steps associated with selecting the features and using them to generate a model is contained below, but we want to focus on the process of extracting a prediction and then using that prediction to make inferences about authorship).

```
lm_fit <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

hm_formula<-as.formula("score ~ upon + there + would + whilst")
hm_recipe<-recipe(hm_formula,train.dat)

hm_wf<-workflow()%>%
  add_model(lm_fit)%>%
  add_recipe(hm_recipe)

hm_wf<-hm_wf%>%
  fit(train.dat)
```

```
tidy(hm_wf)
```

```
## # A tibble: 5 x 5
##   term        estimate std.error statistic     p.value
##   <chr>          <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)  -0.443    0.149       -2.98 0.00413
## 2 upon          0.158    0.0278       5.70 0.000000364
## 3 there         0.0914   0.0256       3.58 0.000688
## 4 would         0.0198   0.00904      2.19 0.0322
## 5 whilst       -0.999    0.251       -3.99 0.000182
```

So we now have a model that we can use to predict a score given the frequency of word use in a document. This is a prediction machine – you tell me the frequency of words (per 1000) of "upon", "there", "would", and "whilst" and we can use linear regression to predict a score for that document.

Now try the same with `words2`. What do you need to change?

```
# INSERT CODE HERE
```

So we have "supervised" the learning by giving the computer a relationship that is known – i.e., we know the authorship of these papers being analyzed and the computer is then using that to figure out predictors of that truth.

As part of the `lm` object we also have all of the predicted values (fitted values) associated with the data being predicted. We can access this the same was we accessed the object of coefficients.

```
hm_train <- hm_wf %>%
  predict(new_data=train.dat) %>%
  bind_cols(train.dat)

hm_train <- hm_train %>%
  mutate(PredictAuthor = ifelse(.pred > 0,"Hamilton","Madison"))

table(Actual = hm_train$author, Prediction = hm_train$PredictAuthor)
##           Prediction
## Actual     Hamilton Madison
##   Hamilton       51       0
##   Madison         0      15
```

Now do the same for model 2!

```
# INSERT CODE HERE
```

What happened and why?

If we looked at the `rmse`we get:

```
hm_train %>%
  rmse(truth=score, estimate=.pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.436
```

And in model 2?

```
# INSERT CODE HERE
```

Classification: Evaluation (out of sample) using `predict`

The real test of a classification model is to predict data it has not seen before. So long as the new data has exactly the same format as the old data (in terms of variable names and transformations), we can use the `predict` function to apply a `lm` model to the new data. To predict all we need to do is to specify the `object` (i.e., model) we are going to use and the `newdata`.

```
hm_test<- hm_wf%>%
  predict(new_data=test.dat)%>%
  bind_cols(test.dat)

hm_test
```

```
## # A tibble: 11 x 11
##       .pred number author   would there  upon whilst president department courts
##   *    <dbl>  <int> <chr>    <dbl> <dbl> <dbl>  <dbl>     <dbl>      <dbl>  <dbl>
##  1 -0.751       49 Contest~ 17.8  1.62   0     0.809     0          2.43       0
##  2 -0.00254     50 Contest~ 12.9  0      1.17  0         0          0          0
##  3 -1.46        51 Contest~  6.28 2.79   0     1.40      0          6.98       0
##  4 -0.330       52 Contest~  5.71 0      0     0         0          0          0
##  5 -0.864       53 Contest~  3.62 1.21   0     0.604     0          0          0
##  6 -0.0900      54 Contest~  4.02 0.670  1.34  0         0          0          0
##  7 -0.0199      55 Contest~  6.46 3.23   0     0         1.29       0          0
##  8 -0.975       56 Contest~  3.30 2.47   0     0.824     0          0          0
##  9 -1.95        57 Contest~  3.61 2.41   0     1.80      0.601      0          0
## 10 -0.387       62 Contest~  2.82 0      0     0         0          0          0
## 11 -0.465       63 Contest~  4.75 3.45   0     0.432     0          0.432      0
## # ... with 1 more variable: coin <dbl>
```

This will generate a predicted value for each observation in the `newdata`. Given this application, it is a predicted score where positive scores should be associated with Hamilton and negative scores shoudl be associated with Madison. Because there are only 11 contested papers we can inspect the scores individually by calling `pred.author`, but because we are interested in whether the prediction suggests that Hamilton or Madison wrote the paper we can also classify the prediction depending on whether the score is greater than 0 or not (given how we defined the meaning of `score` above!)

```
hm_test <- hm_test %>%
  mutate(PredictAuthor = ifelse(.pred > 0,"Hamilton","Madison"))

table(hm_test$PredictAuthor)
```

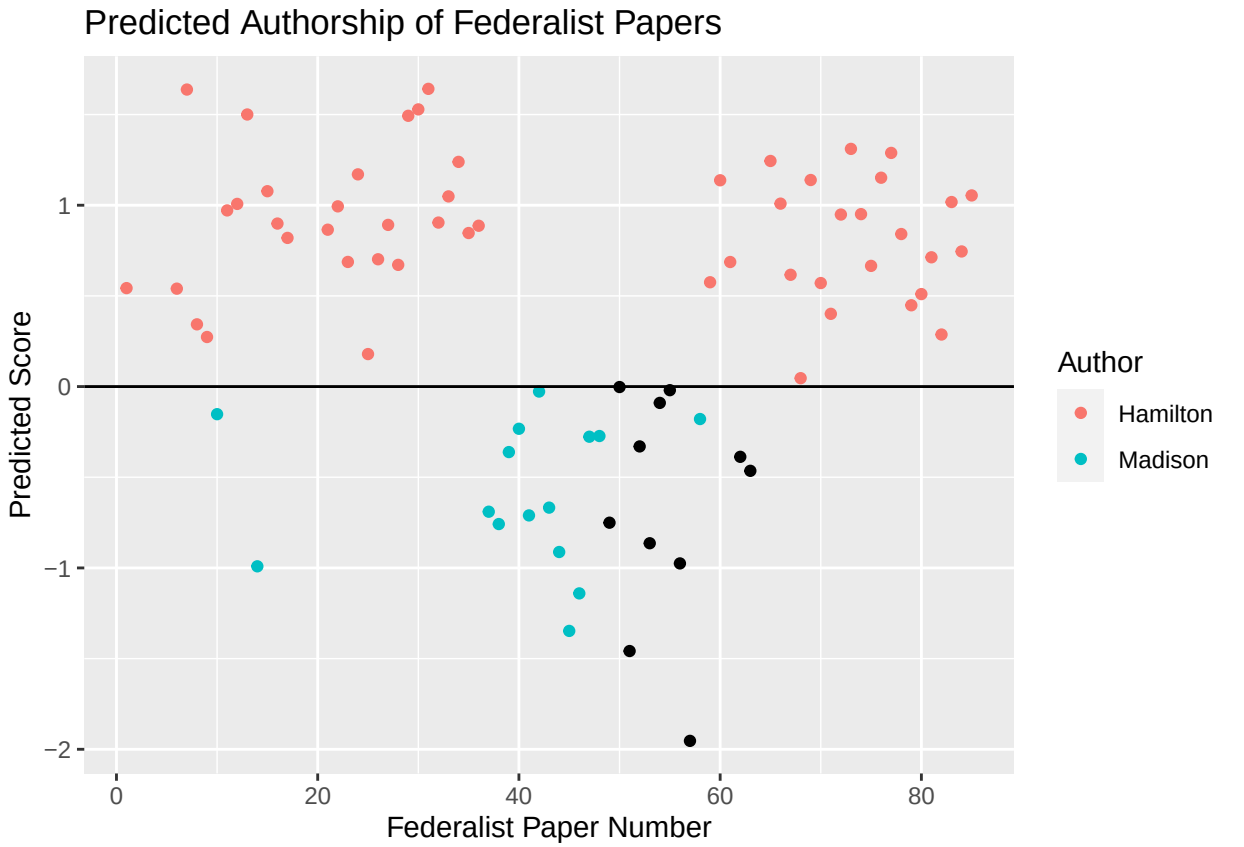
```
##
## Madison
##      11
```

So who probably wrote the contested Federalist papers? Based on this model, it seems like Madison was most likely responsible. (Note the contrast to the conclusion from the clustering analysis!)

We can depict this graphically by plotting the predicted value for every known and contested Federalist Paper written by Hamilton and Madison.

```
ggplot() +
  geom_point(data= hm_train, aes(x = number, y = .pred, color = author)) +
  geom_point(data= hm_test, aes(x = number, y = .pred)) +
  geom_hline(yintercept = 0) +
  labs(color = "Author",
       title = "Predicted Authorship of Federalist Papers",
```

```
       x = "Federalist Paper Number",
       y = "Predicted Score")
```



Higher values of `score` plotted on the y-axis indicate an increased likelihood of Hamilton writing the document. As noted earlier, the basic model we used correctly predicted every single one of the papers with known authorship.The contested papers are plotted in black and you can see that although the variation in scores suggests that some papers were more likely to be written by Madison than others – given the model and given the pattern of word use – every single contested Federalist paper was predicted to be authored by Madison.

As practice, can you get the predictions from our second model? Can you figure out what is going on?

```
# INSERT CODE HERE
```

So this is equally good? What does the data look like? Good lesson in the ways things can go sideways when doing out of sample prediction if the "data generating process" is different!

```
# Inspect test data?
```

We can do even more out of sample predictions. How about the joint-authored papers. Can we use our model to determine whether the writing style was closer to Hamilton or Madison? (i.e., who may have been more likely to have written them?) Of course. Simple. We just need to define our `newdata` object by `filter`ing the original data to select contested papers and then `predict` that `newdata` using the regression object we created.

```
ham.mad.dat <- dat %>%
    filter(author == 'Hamilton.Madison')

hm_wf%>%
  predict(new_data=ham.mad.dat) %>%
```

```
  bind_cols(ham.mad.dat)
```

```
## # A tibble: 3 x 11
##    .pred number author      would there  upon whilst president department courts
## *  <dbl>  <int> <chr>       <dbl> <dbl> <dbl>  <dbl>     <dbl>      <dbl>  <dbl>
## 1 -0.708     18 Hamilton.M~ 3.55  1.77  0.591  0.591         0          0  0
## 2 -0.953     19 Hamilton.M~ 2.46  0.616 0      0.616         0          0  0
## 3 -0.296     20 Hamilton.M~ 0.826 0     0.826  0             0          0  0.826
## # ... with 1 more variable: coin <dbl>
```

The fact that every predicted score is less than 0 suggests that the writing was more in the style of Madison than Hamilton – especially for Federalist Paper 19.

How about John Jay. Did he write more like Hamilton or more like Madison? To answer this question we can predict the papers written by Jay. Using the same steps as above...

```
# INSERT CODE HERE
```

So some of each! Federalist Paper 3 was written in a very "Madisonian" style whereas Federalist Paper 4 was more "Hamiltonian."

Sensitivity Analysis & Cross-Validation

In this application we had a well-defined training and testing dataset but as we talked about earlier we can also use our training dataset to see how sensitive our predictions might be given the data we have. While this is less a concern when you have millions of observations, when we have relatively few observations we may worry that our results are being driven by a few cases. We can use "cross-validation" to try to examine the sensitivity of our estimates.

The basic idea in cross-validation is that we want to:

1. Remove some of the training data
2. Test our model performance on what we left out
3. Summarize how well we did

We can certainly use this to help build a model – as we did last unit – but we can also do some other cool things.

One thing we can do is to see how reliable a predictor a particular variable is by determining how much the predicted relationship between the outcome and that predictor changes as we change the observations we analyze in the training dataset. If the coefficients – i.e., the marginal correlation coefficient – changes as a result of dropping some observations than this suggests that those observations are very important for the estimated correlation.

There are several ways to do this, but one way to do this in R in way that highlights the logic of what is being done is to use a loop to sequentially drop observations, fit a regression to observations excluding the dropped observation, and save the coefficient associated with that regression as follows:

```
tmp <- rep(NA, nrow(train.dat)) # Define a variable to hold the coefficient estimates as we drop observ

for (i in 1:nrow(train.dat)){   # Loop over every observation in our training set
  cv.train.dat <- train.dat %>%
          filter(number != i)   # drop observation i

  hm_cv_wf<-hm_wf %>%
    fit(cv.train.dat)
```

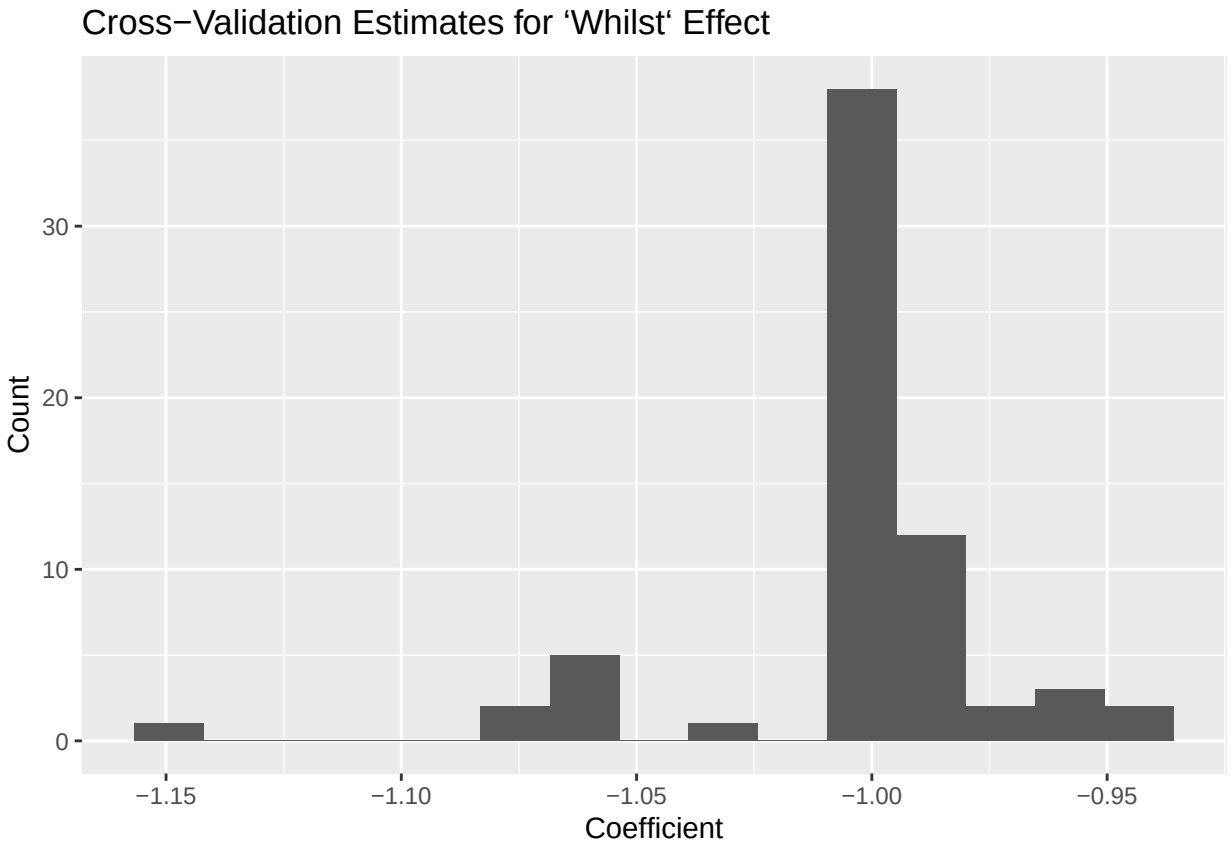
```
  tmp[i] <- tidy(hm_cv_wf)$estimate[5]  # save the coefficient associated with `whilst` from that regre
}
```

Now we can simply plot this

```
ggplot(tidy(tmp),aes(x =x)) +
  geom_histogram(bins=15) +
  labs(title = "Cross-Validation Estimates for `Whilst` Effect",
       x= "Coefficient",
       y = "Count")
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```



Now do this for a different variable!

```
#INSERT CODE HERE
```

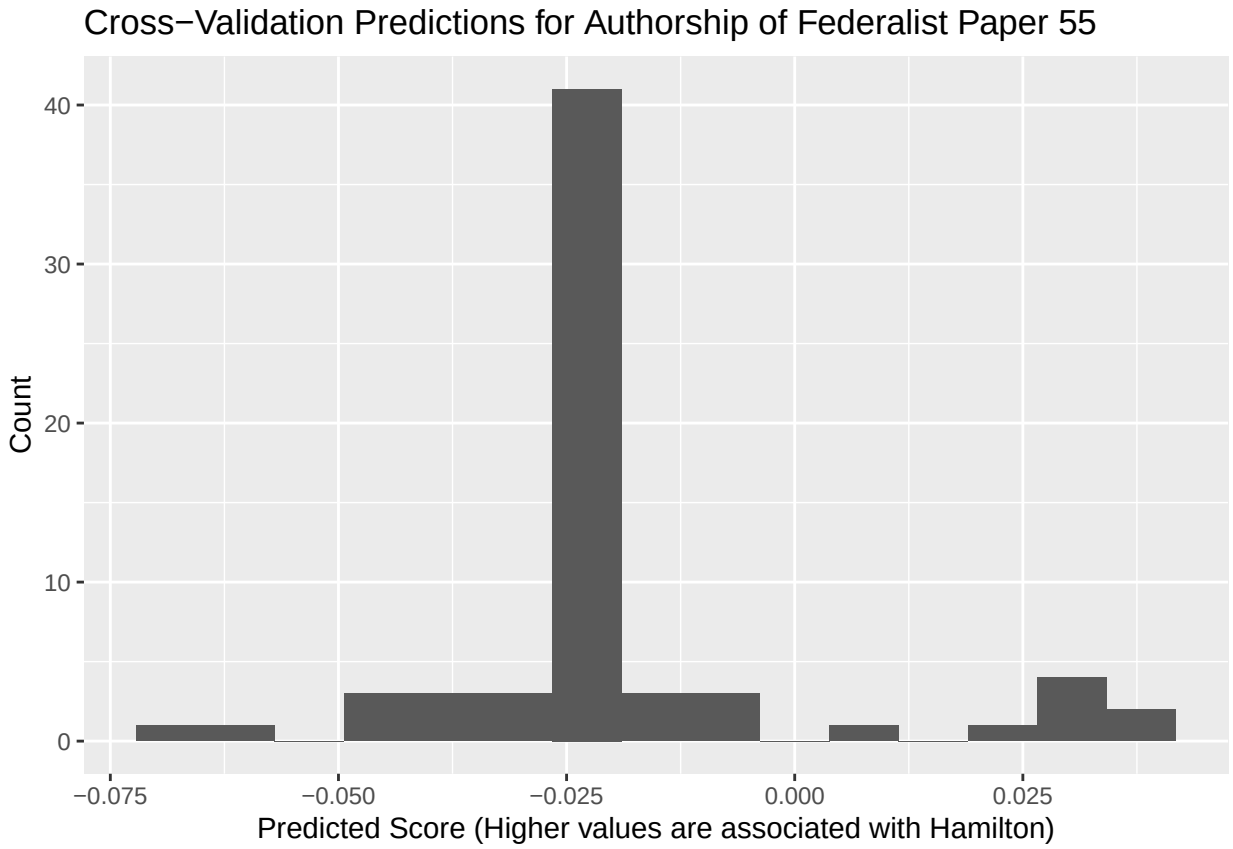
We can also ask the question – sure are we that Hamilton wrote Federalist Paper 55 rather than Madison? To do this we can do the same process but intead of saving the coefficient for `whilst` we use the results for each regression we run on each dataset that drops observation i to `predict` the `score` for Federalist Paper 55.

```
tmp <- rep(NA, nrow(train.dat)) # Define a variable to hold the coefficient estimates as we drop observ

for (i in 1:nrow(train.dat)){   # Loop over every observation in our training set
  cv.train.dat <- train.dat %>%
          filter(number != i)   # drop observation i
```

```
  hm_cv_wf<-hm_wf%>%
    fit(cv.train.dat)

  tmp[i] <- hm_cv_wf %>%
      predict(new_data=test.dat[test.dat$number==55,])
}
```

So now plot this!

```
tmp <- unlist(tmp) # because predict is a list object
ggplot(tidy(tmp),aes(x =x)) +
  geom_histogram(bins=15) +
  labs(title = "Cross-Validation Predictions for Authorship of Federalist Paper 55",
       x= "Predicted Score (Higher values are associated with Hamilton)",
       y = "Count")
```



The resulting predictions in `tmp` show how much our predicted score varies as we drop the obervations one at a time. If we see a lot of fluctuations then this suggests that our predictions are sensitive to the data we are using and this suggests caution.

Now do this for a different Federalist paper!

```
#INSERT CODE HERE
```

This helps emphasize that if history had been a little different our coefficients would have been a little different, and if our coefficients were a little different our predictions are a little different.

How different? Figuring out how different the outcome may have been without actually changing the world to see is one of the things that statistics studies.