Prediction with Multiple Regression

One important use of the linear regression model is prediction. For prediction problems, there are typically three data sets.

1. A **training set**. The researcher uses this data set to fit the model. In our case, this will be a linear regression model.
2. A **prediction set**. This data set contains the predictors, but not the outcome, for several new cases. This might be new people, states, or years. Based on the variables in this data set, the researcher produces a prediction.
3. A **test set**. The same as the prediction set, but with the outcome variable included. The researcher uses the test set to evaluate her modeling approach.

The taxes data have a training set and a prediction set. For these data, the outcome we're trying to predict is `tax_change` which is the legislated changes in tax revenue (in billions) for the 50 U.S. states from 1988 to 2009. The prediction set contains the 50 U.S. states for 2010 and 2011.

For our purposes, a good predictive model minimizes the r.m.s. error of the predictions in the *prediction set* (although we can't know the r.m.s. in the prediction set–there's no outcome to compare our predictions to. After all, it wouldn't be a prediction otherwise.)

```r
# load data
training_set <- readRDS("data/taxes-training.rds")
prediction_set <- readRDS("data/taxes-prediction.rds")

# quick look at data
tibble::glimpse(training_set)
```

```
## Observations: 1,055
## Variables: 21
## $ state                          <chr> "Alabama", "Alabama", "Alabama"...
## $ state_abbr                     <fctr> AL, AL, AL, AL, AL, AL, AL, AL...
## $ year                           <int> 1988, 1989, 1990, 1991, 1992, 1...
## $ tax_change                     <dbl> 0, 0, 47, 22, 100, 0, 0, 0, 0, ...
## $ personal_income                <dbl> 0.060, 0.063, 0.067, 0.072, 0.0...
## $ percent_change_personal_income <dbl> 7.84, 9.09, 5.00, 6.35, 7.46, 4...
## $ estimated_imbalance            <int> 8, 67, -3, 0, 32, 119, 0, 43, 4...
## $ taxes_last_year                <dbl> 3.4, 3.7, 3.8, 3.9, 4.2, 4.6, 4...
## $ gov_request                    <dbl> 0, 0, 55, 0, 520, 0, 0, 0, 0, 0...
## $ lag_tax_change                 <dbl> 0.0, 0.0, -4.8, 0.0, 0.0, 0.0, ...
## $ population                     <dbl> 4.000000, 4.100000, 4.100000, 4...
## $ percent_change_population      <dbl> 0.00, 0.00, 2.50, 0.00, 2.44, 0...
## $ gov_party                      <fctr> Republican, Republican, Republ...
## $ house_dem_share                <dbl> 0.8476190, 0.8333333, 0.8333333...
## $ senate_dem_share               <dbl> 0.8571429, 0.8235294, 0.8235294...
## $ citizen_ideology               <dbl> 45.02993, 37.47248, 33.83535, 3...
## $ year_of_biennium               <fctr> Second Year of Biennium, First...
## $ change_in_gov_party            <fctr> No Change, No Change, No Chang...
## $ change_in_house_dem_share      <dbl> 0.000000000, -0.014285716, 0.00...
## $ change_in_senate_dem_share     <dbl> 0.00000000, -0.03361351, 0.0000...
## $ change_in_citizen_ideology     <dbl> 6.51300, -7.55745, -3.63713, 5....
```

Two variables stand out as especially important. The first is `gov_request`, which is the amount of new taxes requested by the governor. The second is `estimated_imbalance`, which is the estimated budget imbalance for that year. Let's use each of these variables and fit two linear models.

Note that we can include multiple explanatory variables in a regression model by including each on the right-hand side of the formula and separating them with a `+`.

```
fit1 <- lm(tax_change ~ gov_request, data = training_set)
fit2 <- lm(tax_change ~ gov_request + estimated_imbalance, data = training_set)
```

Now here is the important question: How do we know which of these models will best predict the years 2010 and 2011, which are not in this data set, but in the prediction set. Here is the answer: we don't. We have to rely on a good theory or model of the budget making process. However, we can try to guess how well out model will predict out-of-sample cases (in this situation, 2010 and 2011) based on how well it predicts in-sample cases (in this situation, 1988-2009).

R.M.S. Error

First, we can look at the r.m.s. error for each regression. Of course, a lower r.m.s. error is preferred. However, a model that becomes too complex might fit the in-sample data extremely well, but out-of-sample data quite poorly. So bias yourself toward simpler models and models that make theoretical sense. We can use the function `residuals()` to extract the errors from the objects storing our model fits. By taking the square root of the average of the squares of these residuals, we can find the r.m.s. errors of the regressions.

```
sqrt(mean(residuals(fit1)^2))  # r.m.s. error for fit1
```

```
## [1] 383.471
```

```
sqrt(mean(residuals(fit2)^2))  # r.m.s. error for fit2
```

```
## [1] 380.558
```

BIC

We can also use a model summary called the BIC (Bayesian Information Criterion) to guess which model will predict out-of-sample data best. We can calculate the BIC for multiple models using the `BIC()` function and supplying multiple `lm()` outputs as unnamed arguments. Lower values of the BIC indicate a better model. However, the BIC will penalized you for a more complicated model, so it will start to increase for a sufficiently-complicated model (unlike the r.m.s. error).

```
BIC(fit1, fit2)
```

```
##      df      BIC
## fit1  3 15567.79
## fit2  4 15558.66
```

Prediction

Because both the r.m.s. error and BIC suggest that `fit1` is the better model, we might want to use that model to make predictions for the prediction set. For this, we can use the `predict()` function.

The predict function takes two arguments:

1. The first argument is the output of `lm()` for the model you want to use to make predictions. This argument is usually unnamed.
2. The second argument `newdata`, which is the data frame for which you'd like to make predictions. This data frame should include the predictors used the the model (the variable(s) on the right-hand side of the `~`), but it does not need to include the outcome variable. If this argument is not included, then `predict()` makes predictions for the data set used to fit the model.

By default, we store the predictions a new variable in the prediction data set.

```r
prediction_set$prediction <- predict(fit1, newdata = prediction_set)
```

Are these predictions any good? We don't know. In order to find out, we'd have to compare the predictions to the actual values, which I have safely hidden away on my computer.

Review Exercises

1. What does the function `residuals()` do? In particular, what is the first (and only) argument to `residuals()`? What does it output? How can you use the output to calculate the r.m.s. error of the regression?
2. What two model summaries can we use to guess the predictive ability of the model? How do we calculate each in R?
3. What does the function `predict()` do? In particular, what is the first argument to `predict()`? The second? What does it output? How do you store the output of `predict()` as a variable in the prediction set?
4. Does a lower r.m.s. error of the regression indicate that the regression will better predict (out-of-sample)?