Lecture 16: Polynomial and Categorical Regression

1 Review

We predict a scalar random variable Y as a linear function of p different predictor variables $X_1, \ldots X_p$, plus noise:

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p + \epsilon$$

and assume that $\mathbb{E}[\epsilon|X] = 0$, $\mathrm{Var}[\epsilon|X] = \sigma^2$, with ϵ being uncorrelated across observations. In matrix form,

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

the design matrix $\mathbf{X}$ including an extra column of 1s to handle the intercept, and $\mathbb{E}[\epsilon|\mathbf{X}] = \mathbf{0}$, $\mathrm{Var}[\epsilon|\mathbf{X}] = \sigma^2\mathbf{I}$.

If we add the Gaussian noise assumption, $\epsilon \sim MVN(\mathbf{0}, \sigma^2\mathbf{I})$, independent of all the predictor variables.

The least squares estimate of the coefficient vector, which is also the maximum likelihood estimate if the noise is Gaussian, is

$$\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

These are unbiased, with variance $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$. Under the Gaussian noise assumption, $\widehat{\beta}$ itself has a Gaussian distribution. The standard error $\widehat{\mathrm{se}}\left[\widehat{\beta}_j\right] = \sigma\sqrt{(\mathbf{X}^T\mathbf{X})^{-1}_{jj}}$. Fitted values are given by $\mathbf{X}\widehat{\beta} = \mathbf{H}\mathbf{Y}$, and residuals by $\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{Y}$. Fitted values $\widehat{\mathbf{m}}$ and residuals $\mathbf{e}$ are also unbiased and have Gaussian distributions, with variance matrices $\sigma^2\mathbf{H}$ and $\sigma^2(\mathbf{I} - \mathbf{H})$, respectively.

The maximum likelihood estimator of σ^2 is the in-sample mean-squared error, $n^{-1}(\mathbf{e}^T\mathbf{e})$ and $\mathbb{E}\left[\widehat{\sigma}^2\right] = \sigma^2\frac{n-q}{n}$. Under the Gaussian noise assumption, $n\widehat{\sigma}^2/\sigma^2 \sim \chi^2_{n-q}$. Also under the Gaussian noise assumption, the Gaussian sampling distribution of any particular coefficient or conditional mean can be converted into a t distribution, with $n-q$ degrees of freedom, by using the appropriate standard error, obtained by plugging in the de-biased estimate of σ^2.

2 Adding Curvature: Polynomial Regression

If the relationship between Y ansd X is non-linear, we could try to capture that fact with a polynomial. For example, we could use a model like

$$Y = \beta_0 + \beta_1 X + \beta_1 X^2 + \beta_1 X^3 + \epsilon.$$

The same is true if there are many covariates. For example, we could a model like

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \ldots \beta_d X_1^d + \beta_{d+1} X_2 + \ldots \beta_{p+d-1} X_p + \epsilon$$

where instead of Y being linearly related to X_1, it's polynomially related, with the order of the polynomial being d. We just add $d-1$ columns to the design matrix $\mathbf{X}$, containing $x_1^2, x_1^3, \ldots x_1^d$, and treat them just as we would any other predictor variables. With this expanded design matrix, it's still true that $\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$, that fitted values are $\mathbf{H}\mathbf{Y}$ (using the expanded $\mathbf{X}$ to get

H), etc. The number of degrees of freedom for the residuals will be $n - q$ where, in this case, $q = p + 1 + (d - 1)$. Of course, there are many other such models we can fit. For example,

$$Y = \beta_0 + \sum_{i=1}^{p} \sum_{j=1}^{d_i} \beta_{i,j} X_i^j + \epsilon.$$

There are many mathematical and statistical points to make about polynomial regression, but let's take a look at how we'd actually estimate one of these models in R first.

2.1 R Practicalities

There are a couple of ways of doing polynomial regression in R.

The most basic is to manually add columns to the data frame with the desired powers, and then include those extra columns in the regression formula.

```
xsq = x*x
out = lm(y ~ x + xsq)
```

A second method is:

```
out = lm(y ~ x + I(x^2))
```

The function `I()` is the **identity function**, which tells R "leave this alone". We use it here because the usual symbol for raising to a power, `^`, has a special meaning in linear-model formulas, relating to interactions.

The third method is:

```
out = lm(y ~ poly(x,2))
```

which tells R to use a second order polynomial.

2.2 Example

Look at the data in Figure 1. The first fit is linear. It does not look good. Then I tried a cubic polynomial fit which works much better.

Here is the code.

```
> pdf("example.pdf")
> par(mfrow=c(2,2))
>
> ### linear regression
> out = lm(y ~ x)
> summary(out)



Coefficients:
```

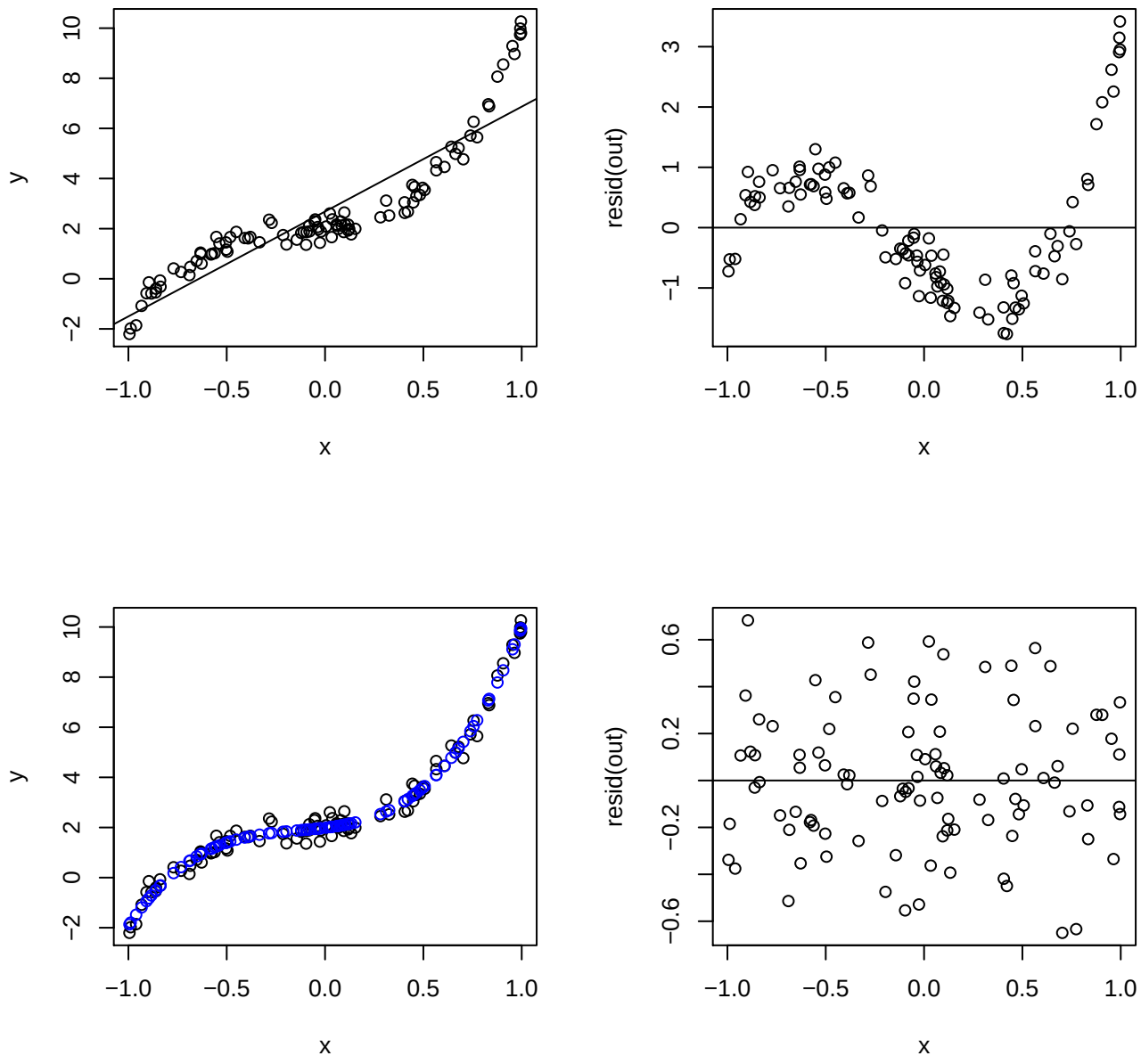

FIGURE 1: *Top left: data and fitted line. Top right: residuals. Bottom left: a cubic polynomial fit. Bottom right: residuals.*

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.6823     0.1124   23.86   <2e-16 ***
x             4.1847     0.1981   21.12   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.124 on 98 degrees of freedom
Multiple R-squared:  0.8199,Adjusted R-squared:  0.8181
F-statistic: 446.3 on 1 and 98 DF,  p-value: < 2.2e-16

> plot(x,y)
> abline(out)
> plot(x,resid(out))
> abline(h=0)
>
> ### cubic regression
> out = lm(y ~ poly(x,3))
> summary(out)



Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.63305    0.02976   88.48   <2e-16 ***
poly(x, 3)1 23.73862    0.29759   79.77   <2e-16 ***
poly(x, 3)2  7.22625    0.29759   24.28   <2e-16 ***
poly(x, 3)3  7.93886    0.29759   26.68   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.2976 on 96 degrees of freedom
Multiple R-squared:  0.9876,Adjusted R-squared:  0.9872
F-statistic:  2555 on 3 and 96 DF,  p-value: < 2.2e-16

> plot(x,y)
> points(x,fitted(out),col="blue")
> plot(x,resid(out))
> abline(h=0)
>
> dev.off()
```

2.3 Another Example

Suppose we use the `mobility` data. Now we fit the model:

```
out = lm(Mobility ~ Commute + poly(Latitude,2) + Longitude,data=mobility)
```

The output looks like this:

```
summary(out)
```

```
##
## Call:
## lm(formula = Mobility ~ Commute + poly(Latitude, 2) + Longitude,
##     data = mobility)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12828 -0.02384 -0.00691  0.01722  0.32190
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.0261223  0.0121233  -2.155   0.0315
## Commute             0.1898429  0.0137167  13.840  < 2e-16
## poly(Latitude, 2)1  0.1209235  0.0475524   2.543   0.0112
## poly(Latitude, 2)2 -0.2596006  0.0484131  -5.362 1.11e-07
## Longitude          -0.0004245  0.0001394  -3.046   0.0024
##
## Residual standard error: 0.04148 on 724 degrees of freedom
## Multiple R-squared:  0.3828,Adjusted R-squared:  0.3794
## F-statistic: 112.3 on 4 and 724 DF,  p-value: < 2.2e-16
```

We can use `predict` as usual:

```
new = data.frame(Commute = 0.298,Latitude = 40.57, Longitude = -79.58)
predict(out, newdata = new)
```

```
## 0.07079416
```

2.4 Which Polynomial?

Smoothness. Polynomial functions vary continuously in all their arguments. In fact, they are "smooth" in the sense in which mathematicians use that word, meaning that all their derivatives exist and are continuous, too. This is desirable if you think the real regression function you're trying to model is smooth, but not if you think there are sharp thresholds or jumps. Polynomials *can* approximate thresholds arbitrarily closely, but you end up needing a very high order polynomial.

Over-fitting and wiggliness. A polynomial of degree d can exactly fit any d points. (Any two points lie on a line, any three on a parabola, etc.) Using a high-order polynomial, or even summing a large number of low-order polynomials, can therefore lead to curves which come very close to the data we used to estimate them, but predict very badly. In particular, high-order polynomials can

display very wild oscillations in between the data points. Plotting the function in between the data points (using `predict`) is a good way of noting this. We will also look at more formal checks when we cover cross-validation later in the course.

Picking the polynomial order. The best way to pick the polynomial order is on the basis of some actual scientific theory which says that the relationship between Y and X_i should, indeed, by a polynomial of order d_i. Failing that, carefully examining the diagnostic plots is your next best bet. Finally, the methods we'll talk about for variable and model selection in forthcoming lectures can also be applied to picking the order of a polynomial, though as we will see, you need to be very careful about what those methods actually do, and whether that's really what you want.

2.5 Orthogonal Polynomials

There are actually many different polynomials that you can use. Suppose that $x \in [-1, 1]$. Here are some polynomials:

$$f_0(x) = 1, \quad f_1(x) = x, \quad f_2(x) = x^2, \quad f_3(x) = x^3, \ldots$$

But here is another set of polynomials:

$$g_0(x) = 1, \quad g_1(x) = x, \quad g_2(x) = \frac{1}{2}(3x^2 - 1), \quad g_3(x) = \frac{1}{2}(5x^3 x), \ldots$$

The latter polynomials are called Legendre polynomials. These polynomials are orthogonal, meaning that

$$\int_{-1}^{1} g_j(x) g_k(x) dx = 0$$

of $j \neq k$. The advantage of using orthogonal polynomials is that the least squares calculations are more stable and the standard errors of the coefficients are smaller. However, the final fitted regression function will be the same whether you use one set of polynomials or another set. In fact, the `poly` command in R is actually creating orthogonal polynomials for you.

2.6 Non-Polynomial Basis Functions

The polynomial functions form a basis. Any well-behaved function can be approximated by a linear combination of polynomials. There are other smooth functons we can use besides polynomials. For example, we can use sines and cosines:

$$\sum_{j=1}^{d} \gamma_{i1j} \sin(j\omega X_i) + \gamma_{i2j} \cos(j\omega X_i).$$

In this course, we will mostly use polynomials.

3 Categorical Predictors

We often have variables which we think are related to Y which are not real numbers, but are qualitative rather than quantitative — answers to "what kind?" rather than to "how much?". For people, these might be things like gender, race and occupation.

Some of these are purely qualitative, coming in distinct types, but with no sort of order or ranking implied; these are often specifically called "categorical", and the distinct values "categories". (The values are also called "levels", though that's not a good metaphor without an order.) Other have distinct levels which can be put in a sensible order, but there is no real sense that the *distance* between one level and the next is the same — they are **ordinal** but not **metric**. When it is necessary to distinguish non-ordinal categorical variables, they are often called **nominal**, to indicate that their values have names but no order.

In R, categorical variables are represented by a special data type called `factor`, which has as a sub-type for ordinal variables the data type `ordered`.

In this section, we'll see how to include both categorical and ordinal variables in multiple linear regression models, by **coding** them as numerical variables, which we know how to handle.

3.1 Binary Categories

The simplest case is that of a binary variable, one which comes in two qualitatively different types. Suppose, for example, that $X_1 \in \{0, 1\}$. We just include X_1 in the regression as usual. This is called an **indicator variable** or **dummy variable**. That is, we **code** the qualitative categories as 0 and 1.

We then regress on the indicator variable, along with all of the others, getting the model

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p + \epsilon.$$

The coefficient β_1 is the expected difference in Y between two units which are identical, except that one of them has $X_1 = 0$ and the other has $X_1 = 1$. That is, it's the expected difference in the response between members of the reference category and members of the other category, all else being equal. For this reason, β_1 is often called the **contrast** between the two classes.

In R. If a data frame has a column which is a two-valued factor already, and it's included in the right-hand side of the regression formula, `lm` handles creating the column of indicator variables internally.

Here, for instance, we use a classic data set to regress the weight of a cat's heart on its body weight and its sex. (If it worked, such a model would be useful in gauging doses of veterinary heart medicines.)

```
library(MASS)
data(cats)
out = lm(Hwt ~ Sex + Bwt,data=cats)
summary(out)
```

```
##
## Call:
## lm(formula = Hwt ~ Sex + Bwt, data = cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.5833 -0.9700 -0.0948  1.0432  5.1016
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4149     0.7273  -0.571    0.569
## SexM         -0.0821     0.3040  -0.270    0.788
## Bwt           4.0758     0.2948  13.826   <2e-16
##
## Residual standard error: 1.457 on 141 degrees of freedom
## Multiple R-squared:  0.6468,Adjusted R-squared:  0.6418
## F-statistic: 129.1 on 2 and 141 DF,  p-value: < 2.2e-16
```

`Sex` is coded as `F` and `M`, and R's output indicates that it chose `F` as the reference category.

Diagnostics. The mean of the residuals within each category is guaranteed to be zero but they should also have the same variance and otherwise the same distribution, so there is still some point in plotting residuals against X_1.

Inference. There is absolutely nothing special about the inferential statistics for the estimated contrast $\widehat{\beta}_1$. It works just like inference for any other regression coefficient.

Why not two columns? It's natural to wonder why we have to pick out one level as the reference, and estimate a contrast. Why not add *two* columns to $\mathbf{X}$, one indicating each class? The problem is that then those two columns will be linearly dependent (they'll always add up to one), so the data would be collinear and the model in-estimable.

Why not two slopes? The model we've specified has two parallel regression surfaces, with the same slopes but different intercepts. We could also have a model with the same intercept across categories, but different slopes for each variable. Geometrically, this would mean that the regression surfaces weren't parallel, but would meet at the origin (and elsewhere). We'll see how to make that work when we deal with interactions in a few lectures. If we wanted different slopes and intercepts, we might as well just split the data.

3.2 Categorical Variables with More than Two Levels

Suppose our categorical variable has more than two levels, say k of them. We can handle it in almost exactly the same way as the binary case. We pick one level — it really doesn't matter which — as the reference level. We then introduce $k-1$ columns into the design matrix $\mathbf{X}$, which are indicators for the other categories. If, for instance, $k = 3$ and the classes are `North`, `South`, `West`, we pick one level, say `North`, as the reference, and then add a column $X_{\texttt{South}}$ which is 1 for data points in class `South` and 0 otherwise, and another column $X_{\texttt{West}}$ which is 1 for data points in that class and 0 otherwise.

Having added these columns to the design matrix, we regress as usual, and get $k-1$ contrasts. The over-all β_0 is really the intercept for the reference class; the contrasts are added to β_0 to get the intercept for each class. Geometrically, we now have k parallel regression surfaces, one for each level of the variable. Diagnostics and inference are the same as in the binary case.

If you have a variable `x` in R and you want R to treat it as a factor, use this command:
`x = as.factor(x)`

Why not k columns? Because, just like in the binary case, that would make all those columns for that variable sum to 1, causing problems with collinearity.

3.3 Two, Three, Many Categorical Predictors

Nothing in what we did above requires that there be only one categorical predictor; the other variables in the model could be indicator variables for other categorical predictors. Nor do all the categorical predictors have to have the same number of categories. The only wrinkle with having multiple categories is that β_0, the over-all intercept, is now the intercept for individuals where *all* categorical variables are in their respective reference levels. Each combination of categories gets its own regression surface, still parallel to each other.

With multiple categories, it is natural to want to look at interactions — to let their be an intercept for left-handed little-endian plumber, rather than just adding up contrasts for being left-handed and being a little-endian and being a plumber. We'll look at that when we deal with interactions.

3.4 Analysis of Variance: Only Categorical Predictors

A model in which there are *only* categorical predictors is, for historical reasons, often called an **analysis of variance** model. Estimating such a model presents absolutely no special features beyond what we have already covered. An "analysis of covariance" model is just a regression with both qualitative and quantitative predictors.

3.5 Ordinal Variables

An ordinal variable is one where the qualitatively-distinct levels can be put in a sensible order, but there's no implication that the distance from one level to the next is constant. At our present level of sophistication, we have basically two ways to handle them:

1. Ignoring the ordering and treat them like nominal categorical variables.
2. Ignoring the fact that they're only ordinal, assign them numerical codes (say 1, 2, 3, ...) and treat them like ordinary numerical variables.

The first procedure is unbiased, but can end up dealing with a lot of distinct coefficients. It also has the drawback that if the relationship between Y and the categorical variable is monotone, that may not be respected by the coefficients we estimate. The second procedure is very easy, but usually without any substantive or logical basis. It implies that each step up in the ordinal variable will predict exactly the *same* difference in Y, and why should that be the case? If, after treating an ordinal variable like a nominal one, we get contrasts which are all (approximately) equally spaced, we might then try the second approach.

3.6 R Example

Let's revisit the `mobility` data. Let's add `State` to the model.

```
nlevels(mobility$State)
```

```
## [1] 51
```

```
levels(mobility$State)
```

```
##  [1] "AK" "AL" "AR" "AZ" "CA" "CO" "CT" "DC" "DE" "FL" "GA" "HI" "IA" "ID"
## [15] "IL" "IN" "KS" "KY" "LA" "MA" "MD" "ME" "MI" "MN" "MO" "MS" "MT" "NC"
## [29] "ND" "NE" "NH" "NJ" "NM" "NV" "NY" "OH" "OK" "OR" "PA" "RI" "SC" "SD"
## [43] "TN" "TX" "UT" "VA" "VT" "WA" "WI" "WV" "WY"
```

There are 51 levels for `State`, as there should be, corresponding to the 50 states and the District of Columbia.

Running a model with `State` and `Commute` as the predictors, we therefore expect to get 52 coefficients (1 intercept, 1 slope, and 51-1 = 50 contrasts). R will calculate contrasts from the first level, which here is `AK`, or Alaska.

```
out = lm(Mobility ~ Commute + State, data=mobility)
coefficients(out)
```

```
## (Intercept)     Commute     StateAL     StateAR     StateAZ     StateCA
##    0.018400    0.126000   -0.005600    0.001840    0.007290    0.031500
##     StateCO     StateCT     StateDC     StateDE     StateFL     StateGA
##    0.044100    0.021500    0.071300    0.007460    0.004160   -0.022000
##     StateHI     StateIA     StateID     StateIL     StateIN     StateKS
##    0.029100    0.052200    0.029700    0.013200    0.010000    0.042400
##     StateKY     StateLA     StateMA     StateMD     StateME     StateMI
##    0.011900    0.021100    0.001230    0.018700    0.004710    0.005230
##     StateMN     StateMO     StateMS     StateMT     StateNC     StateND
##    0.055300    0.011900   -0.018700    0.045200   -0.011400    0.146000
##     StateNE     StateNH     StateNJ     StateNM     StateNV     StateNY
##    0.060400    0.032200    0.062700    0.006670    0.045400    0.022300
##     StateOH     StateOK     StateOR     StatePA     StateRI     StateSC
##   -0.000559    0.036000    0.013800    0.035000    0.022400   -0.019300
##     StateSD     StateTN     StateTX     StateUT     StateVA     StateVT
##    0.042300    0.000761    0.032200    0.060500    0.014100    0.017300
##     StateWA     StateWI     StateWV     StateWY
##    0.025800    0.031700    0.057800    0.061200
```

When the categorical variable has so many levels you might want to simplify the variable. For example, you could replace `State` with a simple variable with a new variable that takes two values: North and South. This leads to something called a *bias-variance tradeoff* that we will discuss later.