Linear Regression

36-290 – Statistical Research Methodology

Week 5 Tuesday – Fall 2021

The Setting

Linear regression is an inferential (read: inflexible) model in which we assume that Y is related to the predictor variables $\mathbf{x}$ via the model

$$Y|\mathbf{x} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon\,,$$

where ϵ represents the scatter of data around the regression line; it is a random variable that is assumed to be normally distributed with mean zero and constant variance σ^2.

Why would we use linear regression?

- While it is inflexible, it is also readily interpretable. (If x_1 changes by one unit, $Y|\mathbf{x}$ changes by β_1 units, on average.) Note that it is not necessarily the case that there is an *a priori* belief that $\mathbf{x}$ and Y are exactly linearly related.
- It is a fast model to learn: the β's can be computed via formula.

Some points to remember:

- The response variable is assumed to be distributed normally. (To be more precise, $Y|\mathbf{x}$ is assumed to be distributed normally, not necessarily Y itself.) You thus may need to transform your response data prior to running linear regression.
- There is no assumption of normality for predictor variables but it is possible that transformations may lead to better fits (i.e., better predictions of the response variable values).
- In the end, linear regression outputs an estimate of the conditional mean: $E[Y|\mathbf{x}]$, i.e., the average value of Y given $\mathbf{x}$.

Linear Regression: Output

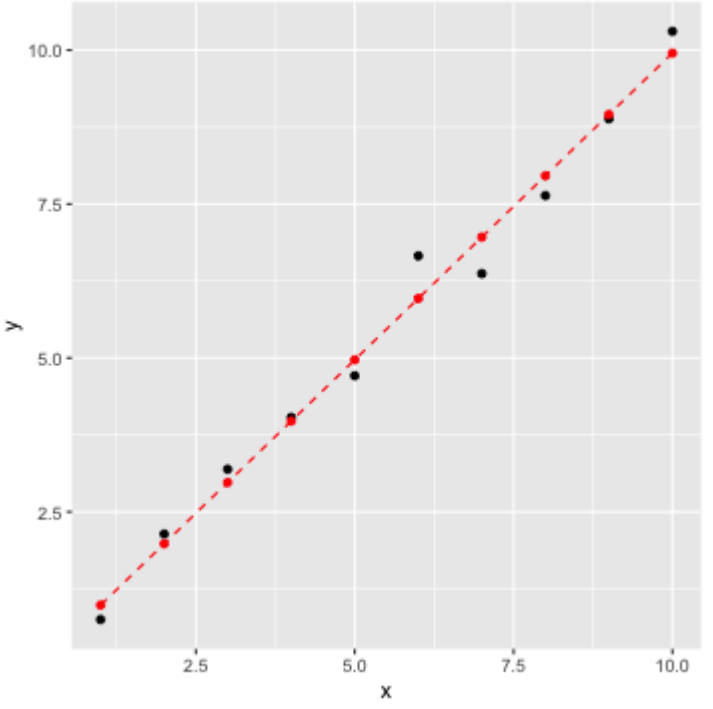
```
set.seed(303)
x = 1:10
y = x + rnorm(10,sd=0.5)
out.lm = lm(y~x)
summary(out.lm)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.59256 -0.25274 -0.00479  0.20039  0.69090
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01069    0.27092  -0.039    0.969
## x            0.99612    0.04366  22.814 1.44e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3966 on 8 degrees of freedom
## Multiple R-squared:  0.9849,    Adjusted R-squared:  0.983
## F-statistic: 520.5 on 1 and 8 DF,  p-value: 1.445e-08
```

In this simple example, the coefficient for variable x is estimated to be 0.996, and the estimated probability that one would observe a value of 0.996 or larger (or -0.996 or smaller) is 1.45×10^{-8}. Since this is less than the conventional decision threshold of 0.05, we conclude that the true value of the coefficient is not zero, i.e., there is a significant association between x and y.

Linear Regression: Output

In addition to the p values, you will note in the example output a value dubbed "Adjusted R-squared" (which has value 0.983). The adjusted R^2 has a value between 0 and 1 and is an estimate of the proportion of the variance of the data along the y-axis that is explained by the linear regression model. Adjusted R^2 provides intuition as to how well a linear model fits to the data, as opposed to the mean-squared error, which is "just a number."




```
## Variance of y =  9.235526   Variance of predicted y =  9.095721   Raw R^2 =  0.9848623
```

Linear Regression: Output

Caveats to keep in mind regarding p values:

- If the true value of a coefficient β_i is equal to zero, then the p value is sampled from a Uniform(0,1) distribution (i.e., it is just as likely to have value 0.45 as 0.16 or 0.84). Thus there's a 5% chance that you'll conclude there's a significant association between x and y even when there is none.
- As the sample size n gets large, the estimated coefficient uncertainty goes to zero, and *all* predictors are eventually deemed significant.

$\Rightarrow$ While the p values might be informative, use variable selection methods (covered later) to determine which subset of the predictors should be included in your final model.

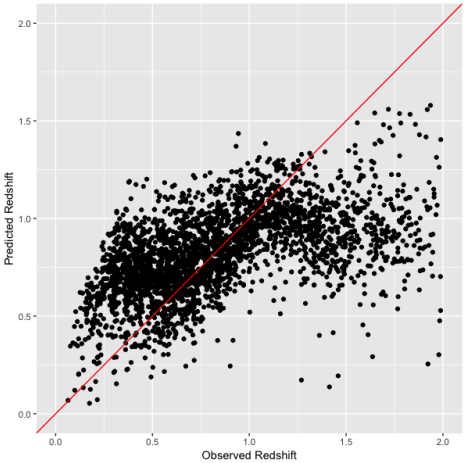
Some other points to keep in mind:

- Outliers may adversely affect your regression estimates. While presumably you would have identified outliers during EDA and removed them at that point, you can in a linear regression setting identify them via the "Cook's distance," which we do not cover here.
- Beware multicollinearity! This arises when one or more predictor variables is linearly related to other predictor variables. We'll come back to this below.

Linear Regression: Model Diagnostics

Since the MSE is unit-dependent, you cannot use its value alone to determine the quality of the underlying model.

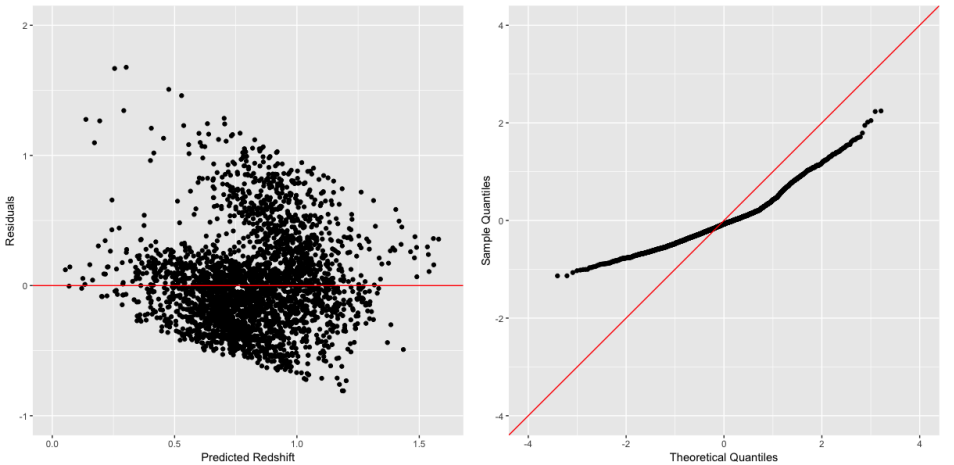
A useful diagnostic (for *any* regression model, not just a linear regression model!) is to plot predicted responses (for the test-set data only) versus the observed responses:



- If the data are completely uninformative, the data will lie on a horizontal locus: every input will generate the same prediction, the average observed value.
- If the model is "perfect," the data will lie along the diagonal line.
- Real-life models will generate plots with behaviors between this two extremes, with additional intrinsic scatter.

Linear Regression: Model Diagnostics

A variation on the diagnostic presented on the previous slide is to plot model residuals (observed response minus predicted response) versus the predicted responses:



If the model reflects well any underlying association, the residuals should have zero mean as a function of the predicted response. Furthermore, for models such as linear regression that have a built-in assumption of normality with constant variance, the standardized residuals should be normally distributed with variance 1. (Normality of the residuals may be checked using, e.g., the `qqnorm()` function.)

Linear Regression: Multicollinearity

For simplicity, assume that we have two predictor variables:

$$Y|x_1, x_2 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \, .$$

Furthermore, assume that it is the case that x_2 is exactly linearly related to x_1:

$$x_2 = \beta_0' + \beta_1' x_1 \, .$$

Then,

$$Y|x_1, x_2 = \beta_0 + \beta_1 x_1 + \beta_2(\beta_0' + \beta_1' x_1) = \beta_0 + \beta_2\beta_0' + (\beta_1 + \beta_2\beta_1')x_1 = \beta_0'' + \beta_1'' x_1 \, .$$

We can determine Y from x_1 alone! The value of x_2 does not matter.

What this means in practice is that if we were to attempt linear regression with such data, we would get a coefficient for x_1 and (in `R`) an `NA` for x_2's coefficient.

Thus multicollinearity affects statistical inference, because we cannot quantitatively relate, e.g., x_2 and Y. (At least, without separately modeling the relationship between x_1 and x_2.)

But: multicollinearity does not affect prediction!

Linear Regression: Variance Inflation Factor

In words: the variance inflation factor, or vif, is the amount by which the estimated variance for a coefficient is inflated because of multicollinearity. For instance, if a modeled regression line is

$$\hat{Y} = 5 + 4x_1 - 2x_2 \, ,$$

the estimated standard deviations for $\hat{\beta}_1$ and $\hat{\beta}_2$ are 2, and the vif's are 4 and 9, then we should view the actual estimate of the standard deviation for $\hat{\beta}_1$ to be $2 \times \sqrt{4} = 4$ and for $\hat{\beta}_2$ to be $2 \times \sqrt{9} = 6$.

In other words, if a vif value for a given predictor variable is very high, then the coefficient output by `lm()` for that variable can be very uncertain, making inferences involving that variable difficult.

How high is too high? The usual rules of thumb are to eliminate predictor variables with vif values above either 5 (more conservative) or 10 (less conservative).

Linear Regression: Variance Inflation Factor

In the following example, we analyze the `sp.passive` dataset from the `GALAXY_PROPERTIES` folder on the 36-290 `GitHub` site. (We remove all columns representing estimates of uncertainty.)

```
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:gtools':
##
##     logit
```

```
lm.out = lm(mass~.,data=predictors)
vif(lm.out)
```

```
##        ra       dec         z     mag.u     mag.g     mag.r     mag.i     mag.z
##  1.024113  1.030077  4.168698  5.980876 25.059826 34.179161 25.224718 14.476612
```

The last four magnitudes have high vif values, while `mag.u`'s value is marginally high.

Linear Regression: Variance Inflation Factor

To mitigate multicollinearity, take out one variable at a time (specifically, the variable with the highest vif value), iterating until all the vif values are below your chosen threshold (conventionally either 5 or 10). If you simply remove all the offending predictor variables at once, you will possibly remove too many of them.

```
THRESHOLD = 10
pred.vif = predictors
istop = 0
while ( istop == 0 ) {
  lm.out = lm(mass~.,data=pred.vif)
  v = vif(lm.out)
  if ( max(v) > THRESHOLD ) {
    pred.vif = pred.vif[,-which.max(v)]
  } else {
    istop = 1
  }
}
print(v)
```

```
##       ra      dec        z    mag.u    mag.z
## 1.023755 1.029297 2.972296 3.388196 1.860109
```

We see that `mag.g`, `mag.r`, and `mag.i` are removed, and that `mag.z` is *not* removed, despite the fact that its initial vif value was >10.

Linear Regression: Variance Inflation Factor

Below we learn linear models with full and "vif-reduced" sets of predictors:

```
# Full Dataset
lm.out = lm(mass~.,data=predictors)
summary(lm.out)$adj.r.squared
```

```
## [1] 0.8306293
```

```
mean((mass-predict(lm.out))^2)
```

```
## [1] 0.0481017
```

```
# vif-Reduced Dataset
lm.out = lm(mass~.,data=pred.vif)
summary(lm.out)$adj.r.squared
```

```
## [1] 0.8120881
```

```
mean((mass-predict(lm.out))^2)
```

```
## [1] 0.05336819
```

Eliminating variables impacts the ability to predict mass, but not too badly: the adjusted R^2 goes from 0.831 to 0.812, and the training-set MSE rises from 0.048 to 0.053. However, one does not know *a priori* the amount by which prediction will suffer...so one should always explore this angle, in any analysis involving linear (or logistic!) models.

Why PCA?

This is a good place to make a final point. Let's assume

- you *really*, **really** want to learn an inferential model, but
- your dataset has issues with multicollinearity.

One option that is open to you: determine principal components for your predictor variables, then run principal components regression (a topic covered by ISLR). Why? Because, by definition, there is no correlation between the data along different principal component axes. Your linear regression analysis will be "clean," and what will be left is the inferential step: the need to determine how the predictor variables map to the retained PCs. (This part will be "fuzzier.")