Linear Regression

36-600

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

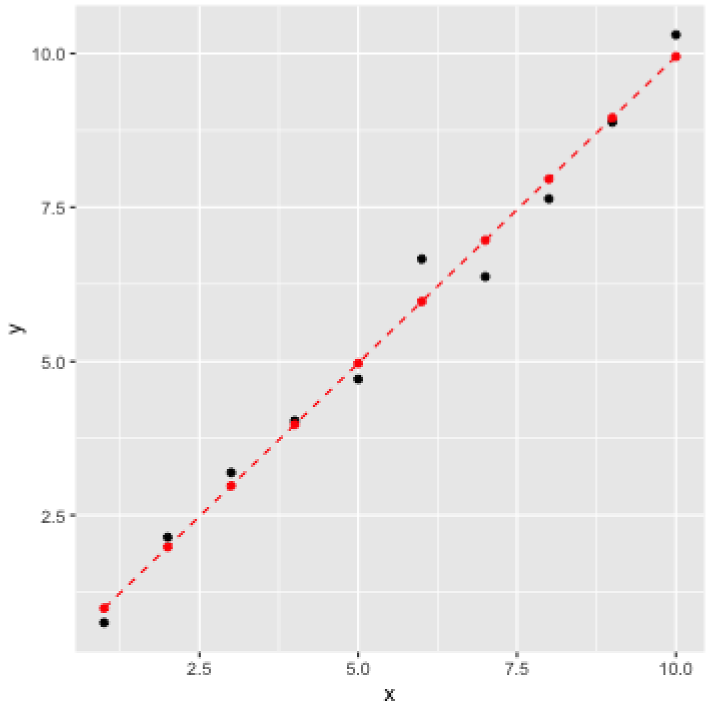
```r
set.seed(303)
x <- 1:10
y <- x + rnorm(10,sd=0.5)
out.lm <- lm(y~x)
summary(out.lm)
```

```
##
## 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 below) to determine which subset of the predictors should be included in your final model.

Linear Regression: What if I Have Categorical Predictors?

Let's add a factor variable with favorite ice-cream flavor to what we had before:

```
names(df) ; levels(df$ic) ; summary(lm(y~.,data=df))
```

```
## [1] "x"  "y"  "ic"
```

```
## [1] "Chocolate" "Vanilla"
```

```
## 
## Call:
## lm(formula = y ~ ., data = df)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.61956 -0.24318  0.01021  0.20864  0.64890 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.02231    0.34101   0.065    0.950    
## x            0.98112    0.09458  10.374 1.68e-05 ***
## icVanilla    0.09901    0.54331   0.182    0.861    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.423 on 7 degrees of freedom
## Multiple R-squared:  0.9849,	Adjusted R-squared:  0.9806 
## F-statistic: 228.8 on 2 and 7 DF,  p-value: 4.198e-07
```

Linear Regression: What if I Have Categorical Predictors?

In the output, we see a variable named `icVanilla`. What's up with that?

When a predictor is a factor variable with k levels, then $k-1$ so-called *dummy variables* are shown in the output. Here, `ic` has levels `Chocolate` and `Vanilla`, and so `Chocolate` (the first level) becomes the "reference level" and `icVanilla` is what gets output.

Mathematically, the predictive model is

$$\hat{Y}|\mathbf{x} = \beta_0 + \beta_x x + \beta_{vanilla}\mathbb{I}_{vanilla} \, .$$

$\mathbb{I}$ is an *indicator variable*: here, it takes on value 1 if the value of `ic` is `Vanilla` and 0 otherwise. So, for chocolate ice-cream eaters, the model is

$$\hat{Y}|\mathbf{x} = \beta_0 + \beta_x x \, ,$$

while for vanilla ice-cream eaters, the model is

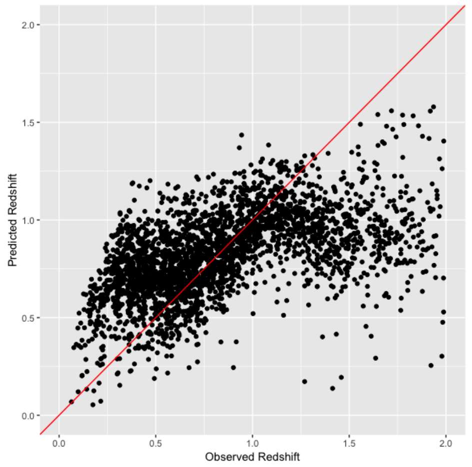
$$\hat{Y}|\mathbf{x} = \beta_0 + \beta_x x + \beta_{vanilla} \, .$$

According to the output, we would add 0.099 to $\hat{Y}$ if the datum is associated with vanilla. However, we do note that the p-value is high, and that the null hypothesis that $\beta_{vanilla} = 0$ is not rejected, so there's no statistical evidence for an offset between data associated with different flavors!

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: Multicollinearity and the 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 much more uncertain than `R` is indicating, 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).

Here's an example of vif output:

```
suppressMessages(library(car))
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 variables have high vif values, while `mag.u`'s value is marginally high.

Linear Regression: Variance Inflation Factor

If you choose 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 may end up removing too many variables.

```
# Feel free to STEAL THIS CODE!
#
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 in this particular case: 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.

Variable Selection

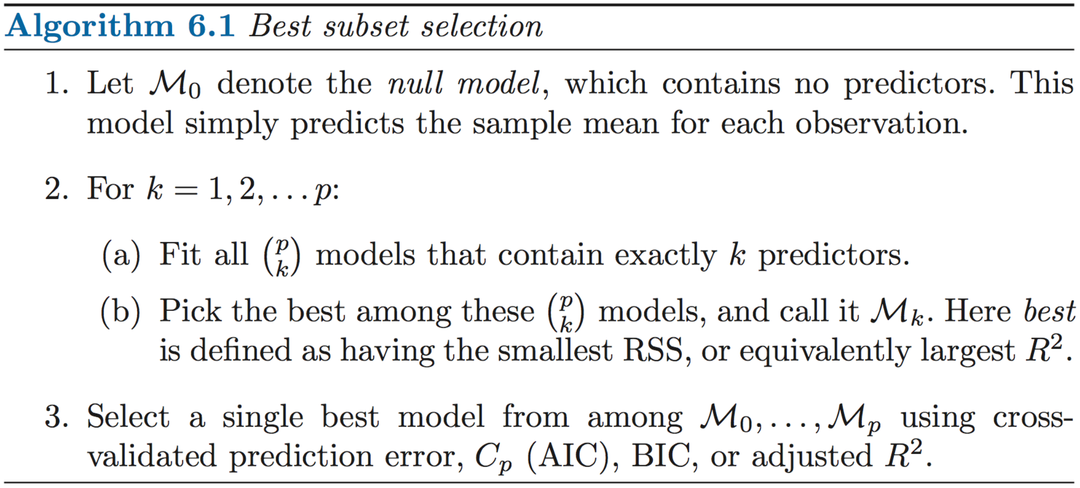
In subset selection, we attempt to select a subset s out of the p overall predictors in a linear model. Why?

1. *To improve prediction accuracy*. Eliminating uninformative predictors can lead to lower model variance, at the expense of a slight increase in bias, leading to lower test-set MSE values.

2. *To improve model interpretability*. Eliminating uninformative predictors is obviously a good thing when your goal is to tell the story of how your predictors are associated with your response.

Note that subset selection is useful and/or necessary if, e.g., $n \lesssim p$ (the sample size is roughly the same as, or less than, the number of predictor variables), but can still be helpful if $n > p$. As $n/p \rightarrow \infty$, subset selection will yield less and less "useful" results, i.e., it will eliminate fewer and fewer variables. However, you should still try it even when $n \gg p$: you won't know for sure how useful subset selection is otherwise!

Best Subset Selection



Algorithm 6.1 *Best subset selection*

```
1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors. This
   model simply predicts the sample mean for each observation.

2. For $k = 1, 2, \ldots p$:

   (a) Fit all $\binom{p}{k}$ models that contain exactly $k$ predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here best
       is defined as having the smallest RSS, or equivalently largest $R^2$.

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-
   validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

(Algorithm 6.1, *Introduction to Statistical Learning* by James et al.)

Note:

$$\binom{p}{k} = \frac{p!}{k!(p-k)!} .$$

For multiple linear regression, BSS works for $p \lesssim 25$; otherwise the total number of models is such that lack of computer memory becomes an issue. For logistic regression (i.e., for a categorical response), BSS is limited to $p \leq 15$ due to the computational costs of having to perform numerical optimization. (And BSS is *slow* near that upper limit.) We will present a mitigation strategy for the high-variable/logistic-regression case later when we cover logistic regression.

Best Subset Selection: Metrics

The functional forms of the metrics given in Step 3 are

$$\begin{aligned} C_p &= \frac{1}{n}(\text{RSS} + 2k\hat{\sigma}^2) \\ \text{AIC} &= \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2k\hat{\sigma}^2) = \frac{C_p}{\hat{\sigma}^2} \\ \text{BIC} &= \frac{1}{n}(\text{RSS} + \log(n)k\hat{\sigma}^2) \end{aligned}$$

RSS denotes the "residual sum-of-squares." The additive terms are penalty terms that increase with k and thus act to prevent overfitting. $\hat{\sigma}$ is an estimate of the standard deviation of the linear regression error term ϵ, i.e., the magnitude of the scatter of data around the regression line (thus the metrics implicitly assume constant error).

Best Subset Selection: Metrics

Typically, $\log(n) > 2$, so BIC (or "Bayesian Information Criterion") imposes a larger penalty relative to C_p (or "Mallow's C_p") or AIC (or "Akaike Information Criterion").

$\Rightarrow$ BIC tends to underfit (i.e., it will select as optimal those models that have *fewer* variables)

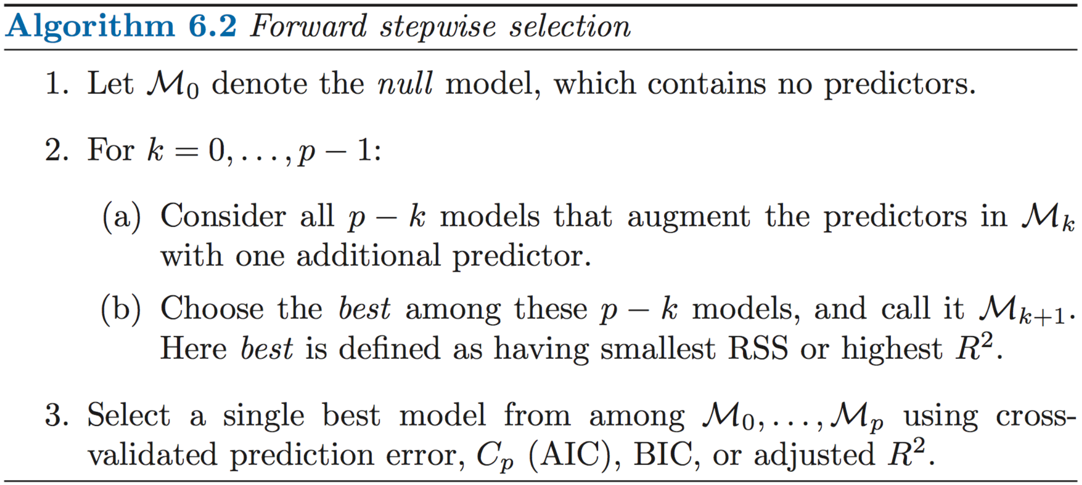
$\Rightarrow$ C_p and AIC tend to overfit (i.e., they will select models with *more* variables)

Which metric you choose is up to you; the choice should be motivated by your inferential goals. (This is another one of those "Embrace the Ambiguity" moments.)

- If you use BIC, then you can be confident that every selected variable is important, but other important variables might have been left out of the final list.
- If you use, e.g., AIC, then you can be confident that your selected variables include all the important ones, but the final list may also include some unimportant ones as well.

Forward and Backward Stepwise Selection

What if BSS is computationally infeasible? In that case, we might use either *forward* or *backward stepwise selection*. For instance:



Algorithm 6.2 *Forward stepwise selection*

```
1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
    (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$
        with one additional predictor.
    (b) Choose the *best* among these $p - k$ models, and call it $\mathcal{M}_{k+1}$.
        Here *best* is defined as having smallest RSS or highest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-
   validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

(Algorithm 6.2, *Introduction to Statistical Learning* by James et al.)

In words, forward stepwise selection starts with no predictor variables and adds one at a time; backward stepwise selection is similar, except that it starts with the full set of predictors and takes one out at a time. One can apply forward and backward stepwise selection using `regsubsets()` or `bestglm()` as above, but with the arguments `method="forward"` or `method="backward"`.

Forward and backward stepwise selection are examples of *greedy algorithms*: they make locally optimally choices that may collectively not yield a globally optimal solution. BSS is always to be preferred, if applying it is computationally feasible.

Regression Example

`df` is a data frame with 3,419 rows and 17 columns. The response variable is contained in a column dubbed `y`, which is what the `bestglm` package expects.

```r
# Perform 70-30 data splitting.
set.seed(404)
train <- sample(nrow(df),0.7*nrow(df))
df.train <- df[train,]
df.test  <- df[-train,]
suppressMessages(library(bestglm))
bg.out <- bestglm(df.train,family=gaussian,IC="BIC")
bg.out$BestModel
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(Xy[, c(bestset[-1], FALSE),
##     drop = FALSE], y = y))
##
## Coefficients:
## (Intercept)        mag.i       col.iJ       col.JH          J.G       J.size          H.G        H.M20          H.
##      1.4113       0.4540      -0.7420       0.4833       4.1496      -1.1839      -3.1846       0.3445       0.224
##      H.size
##      1.6232
```

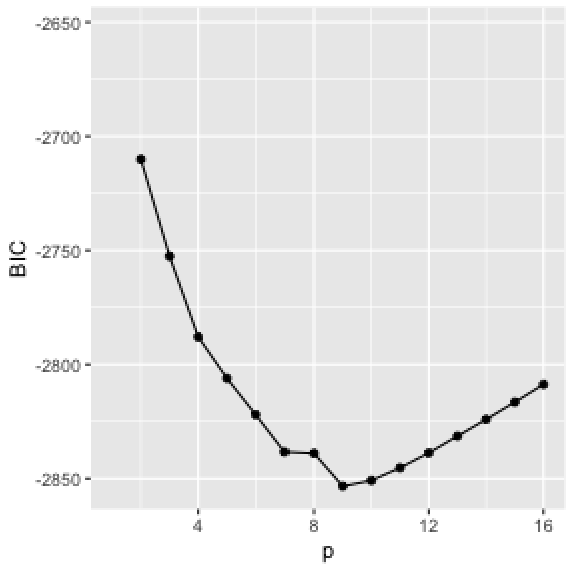
We observe that 9 of 16 predictor variables are retained when using BIC as the penalizing criterion. (If we were to use AIC instead? 11 variables are retained.)

Regression Example

```r
library(ggplot2)
df.bg <- data.frame(1:16,bg.out$Subsets$BIC[-1]) # the -1 gets rid of the BIC for a no-variable model
names(df.bg) <- c("p","BIC")
ggplot(data=df.bg,mapping=aes(x=p,y=BIC)) +
  geom_point() + geom_line() + ylim(min(df.bg$BIC),min(df.bg$BIC)+200)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Regression Example

The output of `bestglm()` contains, as you saw above, `BestModel`. According to the documentation for `bestglm()`, `BestModel` is "[a]n lm-object representing the best fitted algorithm." That means you can pass it to `predict()` in order to generate predicted response values (where the response is in the `y` column of your data frames).

```
resp.pred <- predict(bg.out$BestModel,newdata=df.test)
mean((df.test$y-resp.pred)^2)  # compare with 0.2658 for full predictor set
```

```
## [1] 0.2643661
```