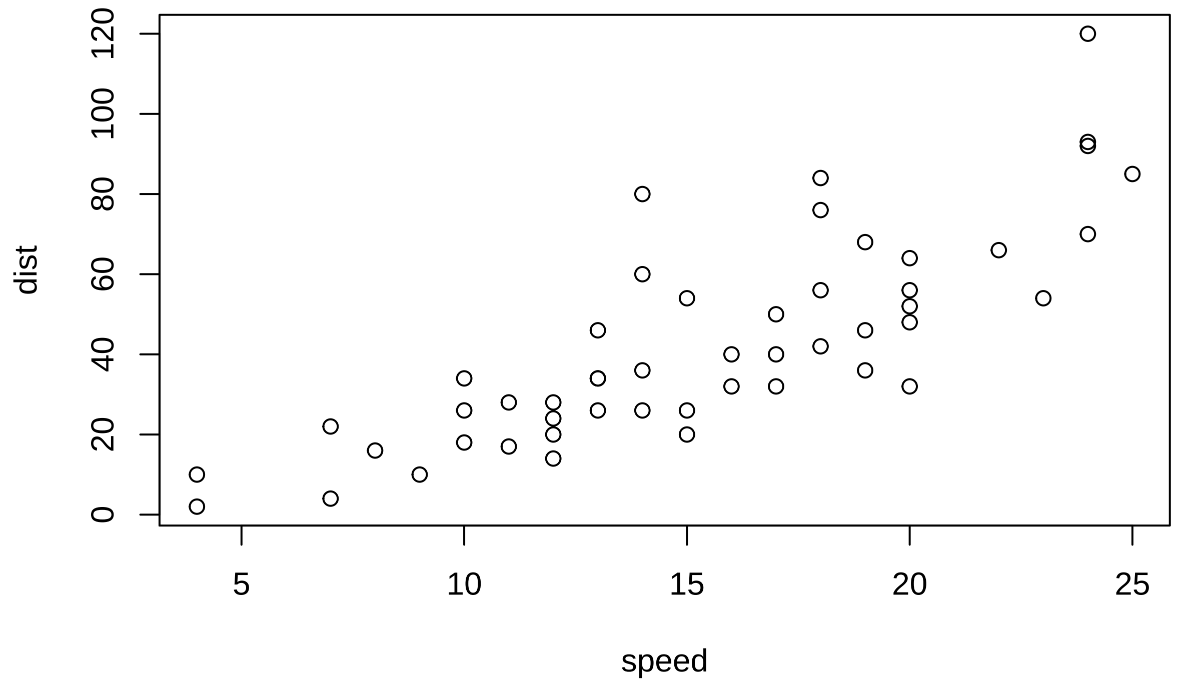
Cross Validation Testing

Chris Henson

The cars data set that we have worked with several times this summer is as simple as we can really get. We have fifty observations of speed and distance. This makes for a perfect data set to explore some of the subtleties of cross validation.

First, let's just look at a plot of our data:

```r
plot(cars)
```



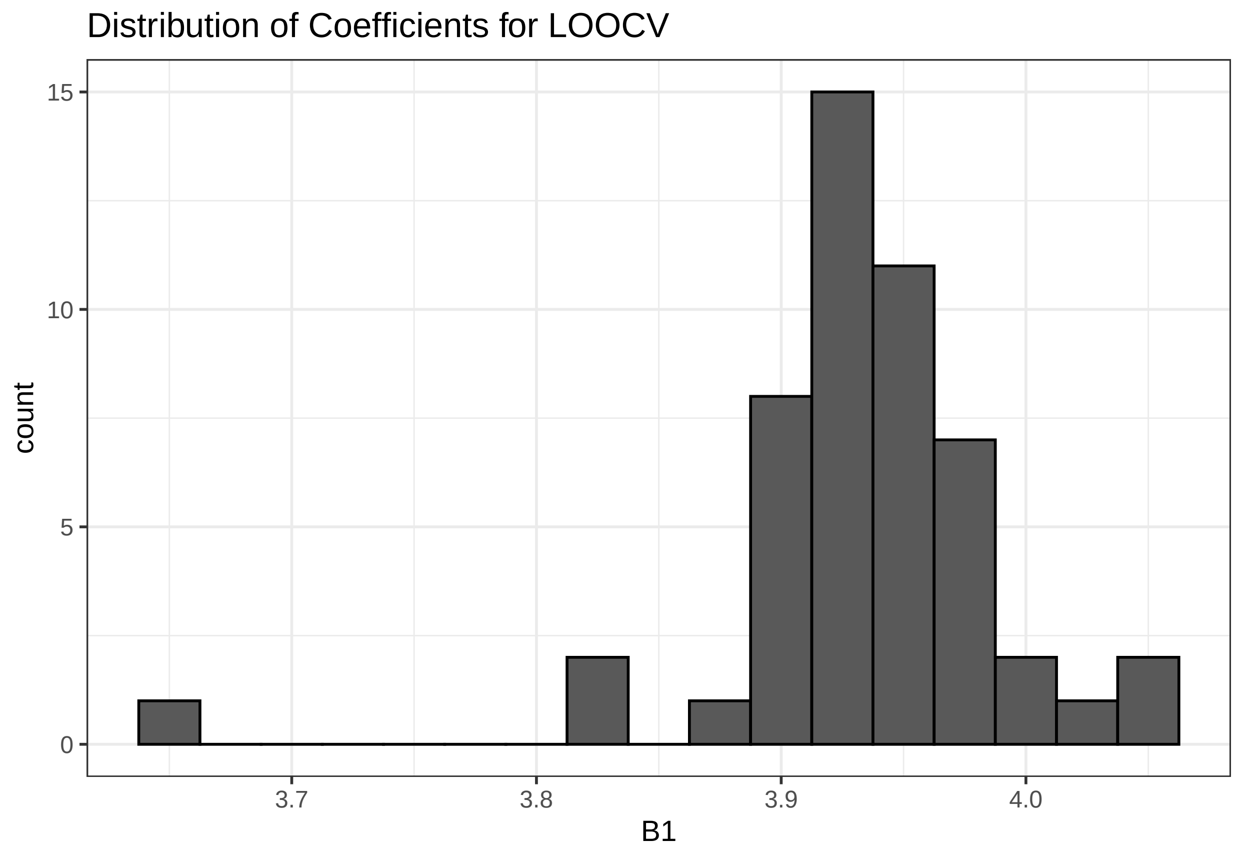
It looks like there's a pretty strong positive correlation, and it's tempting to run a linear regression for this data and leave it at that. However, we've been learning all about using cross validation to ensure that we're not just fitting to our training set. Our most robust method is the "leave-one-out" method, which I'll use first.

I will run a total of 50 regressions, each time taking out one point and running my regression on the remaining 49. I will also store the R squared and coefficient values so that I can take a closer look at them.

```r
#list to store slopes
LOOCVslopes = list()
#list to store R2 values
LOOCVr2 = list()

for (i in 1:50){
  #leave out one observation
  leaveOneOut = cars[-i,]
  linear = lm(dist ~ speed, data = leaveOneOut)

  LOOCVslopes[[i]] = linear$coefficients[2]
  LOOCVr2[[i]] = summary(linear)$r.squared
}

#create slope and R2 data frames
LOOCVslopes = data.frame(do.call("rbind", LOOCVslopes))
names(LOOCVslopes) <- c('B1')

LOOCVr2 = data.frame(do.call("rbind", LOOCVr2))
names(LOOCVr2) <- c('R2')
```

Let's take a look at some plots of our leave-one-out cross-validation. First I'll look at the distribution of coefficients:

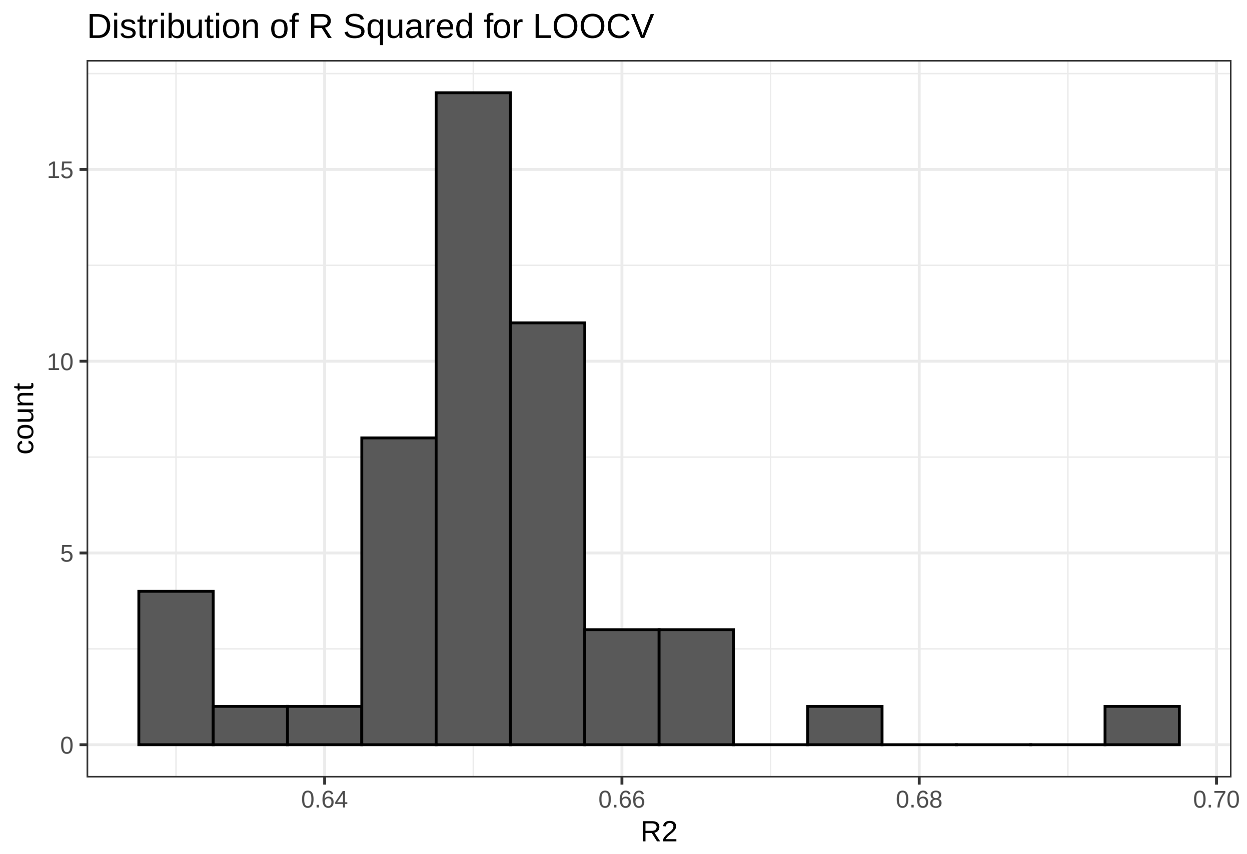
```r
library(ggplot2)

gg <- ggplot(LOOCVslopes,aes(B1)) +
      geom_histogram(color='black',binwidth=.025) + theme_bw() +
      labs(title = "Distribution of Coefficients for LOOCV")

plot(gg)
```



This looks pretty normally distributed, as we would expect. Since we are only taking out one data point at a time, our slope isn't going to differ very much from the slope of the regression for all 50 points. (The few values out in the tail are result of the slope changing more if we remove a high leverage point)

We can look at a similiar plot for R squared:

```
gg <- ggplot(LOOCVr2,aes(R2)) +
      geom_histogram(color='black',binwidth=.005) + theme_bw() +
      labs(title = "Distribution of R Squared for LOOCV")

plot(gg)
```

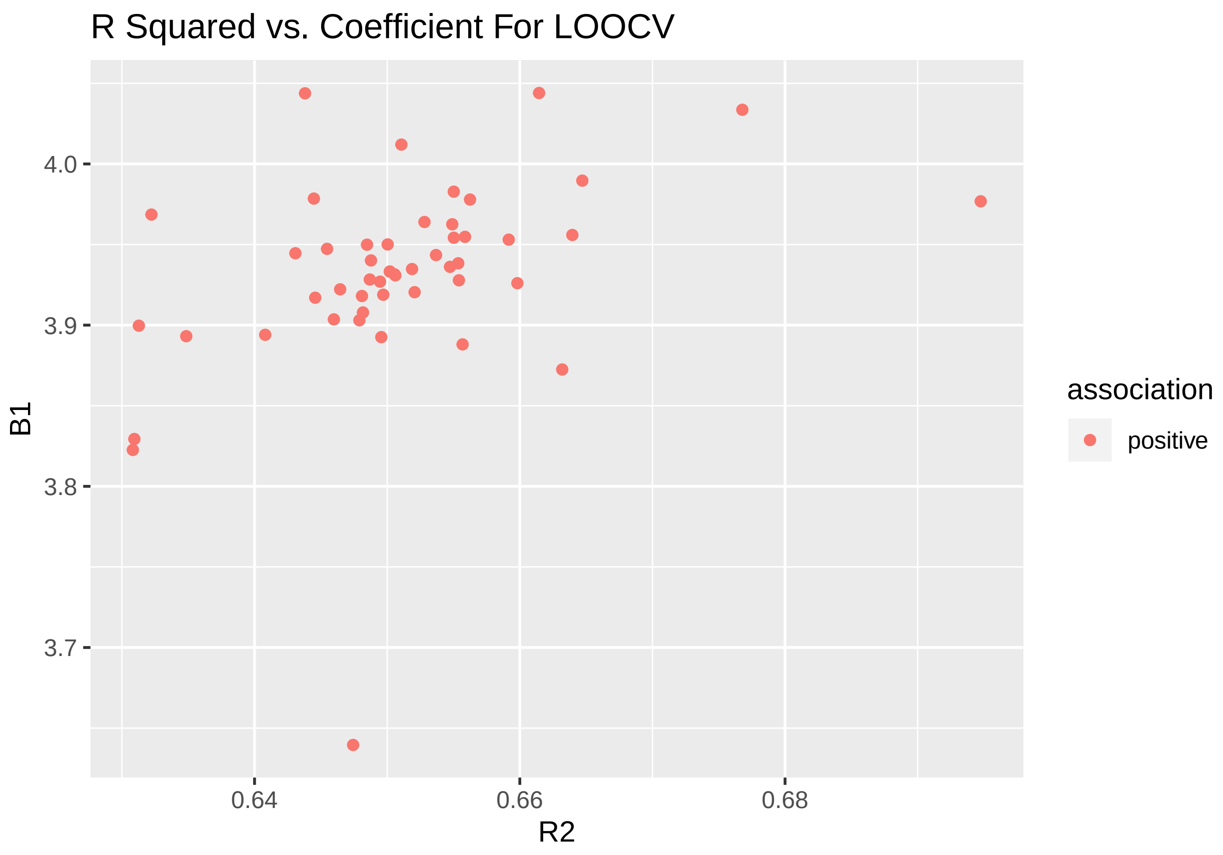


Again we see the same effect, and the few instance of a higher R squared value indicate that we probably removed a value that could be classified as an “outlier” or “noise” that is specific to the data set.

One final plot we can look at is the R squared value versus the coefficient for each regression. I've added a color to indicate association. (No surprise that they're all positive!) This highlights the trend that for this data set, we tend to get higher correlation for the cross validations that assign a larger coefficent.

```r
#bind slope and R2 to compare
compare <- cbind(LOOCVslopes, LOOCVr2)
compare$association <- ifelse(compare$B1 > 0, "positive", "negative")
compare <- compare[complete.cases(compare),]
```

```r
gg <- ggplot(compare, aes(x=R2, y=B1, color = association)) +
      scale_color_manual(values=c("#F8766D", "#00BA38"),breaks=c("positive","negative")) +
      geom_point() +
      labs(title = "R Squared vs. Coefficient For LOOCV")

plot(gg)
```



Now I want to repeat my entire process from above, but this time with every possible k-fold cross validation. The LOOCV method split our data into fifty different training/test splits. What I want to look at now is spliting our data into 49 traing/test splits, then 48, etc. all the way to just looking at the data with no training/test split, and running regressions for all of these training sets.

(Note that there will be some splits that don't work. For instance, I can't do regression with just one point! ggplot will automatically remove any invalid coefficients and r squared values before plotting, so it'll be fine)

```
for (k in 1:50){
  df = cars
  df    <- df[order(runif(nrow(df))), ]
  bins  <- rep(1:k, nrow(df) / k)
  suppressWarnings(assign(paste(k, 'fold_validation', sep = '.') , split(df, bins)))

  assign( paste('slopes', k, sep = '.'), list() )
  assign( paste('r.squared', k, sep = '.'), list() )
}
```

```
#collect validation frames into a list
folds<-grep("fold_validation",names(.GlobalEnv),value=TRUE)
folds_list<-do.call("list",mget(folds))
```

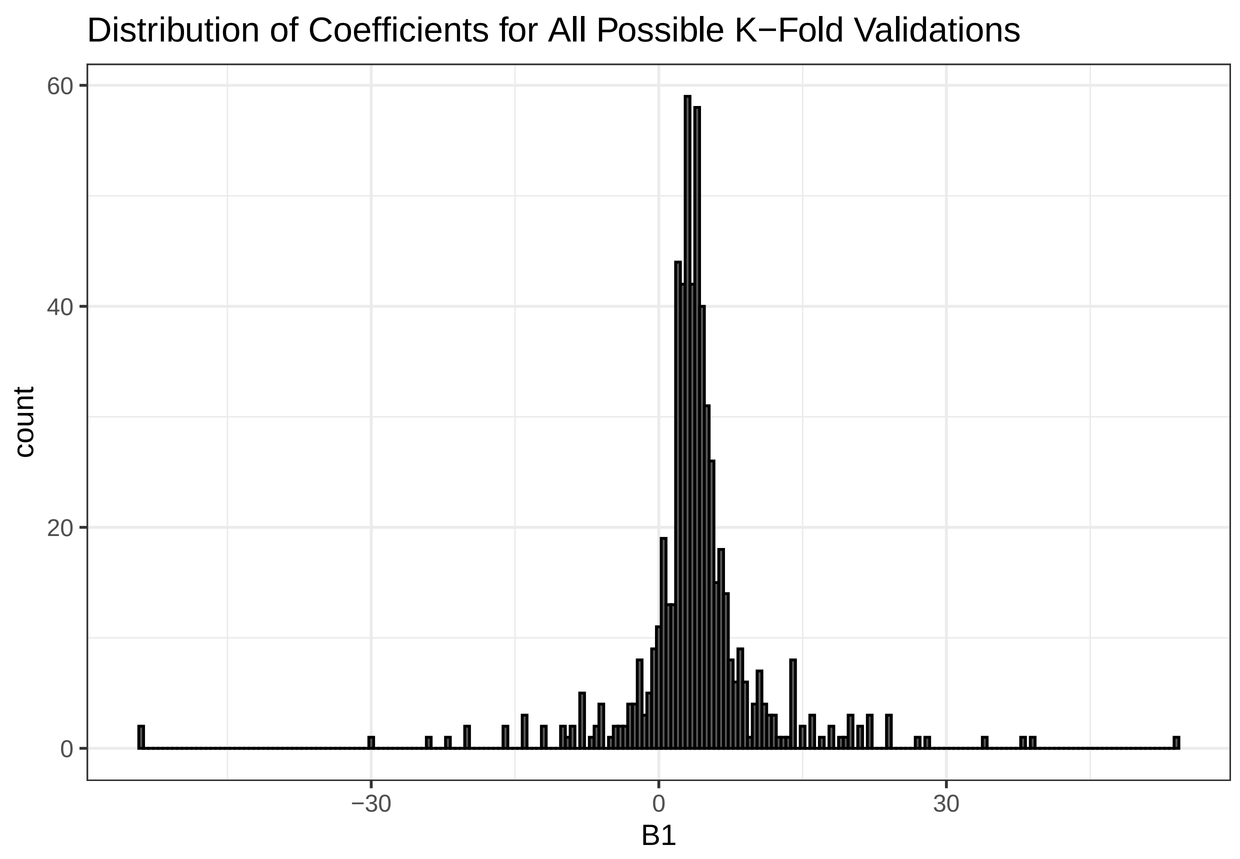
```
#list to store slopes
allSlope = list()
#lsit to store R2 values
allR2 = list()

#loop across each k validation
for (validation in folds_list){
  #loop across each training set
  for (frame in validation){
    linear = lm(dist ~ speed, data = frame)
    allSlope = c(allSlope, linear$coefficients[2])
    allR2 = c(allR2, summary(linear)$r.squared)
  }
}
```

```
#create slope and R2 frames
allSlope = data.frame(do.call("rbind", allSlope))
names(allSlope) <- c('B1')

allR2 = data.frame(do.call("rbind", allR2))
names(allR2) <- c('R2')
```

First I'll take a look at my distribution of coefficients:

```r
gg <- ggplot(allSlope,aes(B1)) +
      geom_histogram(color='black',binwidth=.5) + theme_bw() +
      labs(title = "Distribution of Coefficients for All Possible K-Fold Validations")

plot(gg)
```

```
## Warning: Removed 670 rows containing non-finite values (stat_bin).
```



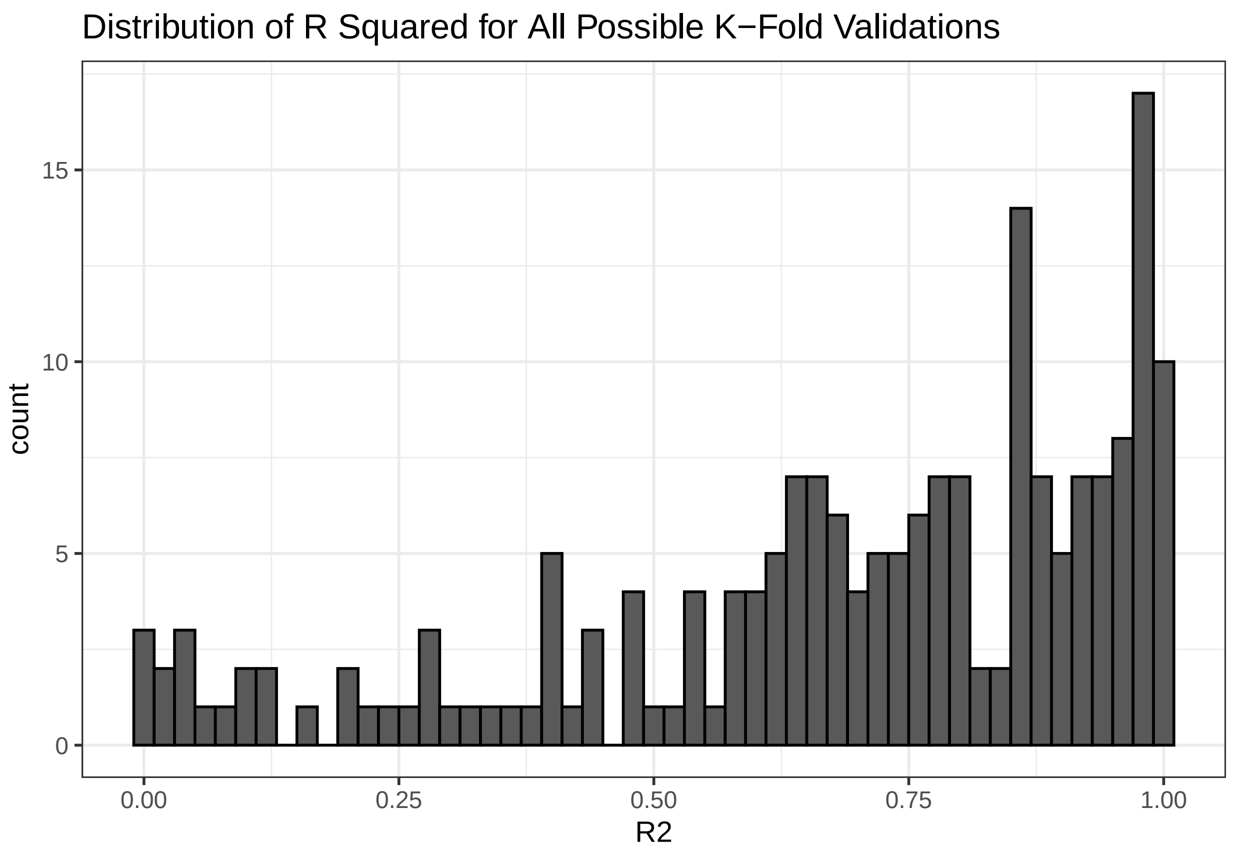
Again, I've gotten a pretty good normal distribution, especially with such a large sample size. What is concerning however, is how spread my coefficients are. It looks like now my coefficients could reasonably range anywhere from -30 to +30 if I get a value far in the tails of this distribution. how is this possible?

This highlights a key aspect of cross validation and averaging methods: **we do not have a garuntee that we didn't pick a pathological training/test split that misrepresents our predictors**. This is why it is preferable that we repeat our cross validation and arefully select k. While it is impractical to always use the better LOOCV method, we can at least repeat our cross validation for a moderate k to ensure that we are getting a good sampling of predictors from our distribution

Now let's take a look at our R Squared values for every possible cross validation, with near-perfect fit results omitted so that we can look at a reasonable histogram:

```r
#remove extreme R2 values
notPerfectFit <- data.frame(allR2[allR2$R2 > 0.001 & allR2$R2 < .999,])
names(notPerfectFit) <- c('R2')

gg <- ggplot(notPerfectFit ,aes(R2)) +
      geom_histogram(color='black',binwidth=.02) + theme_bw() +
            labs(title = "Distribution of R Squared for All Possible K-Fold Validations")

plot(gg)
```

```
## Warning: Removed 6 rows containing non-finite values (stat_bin).
```

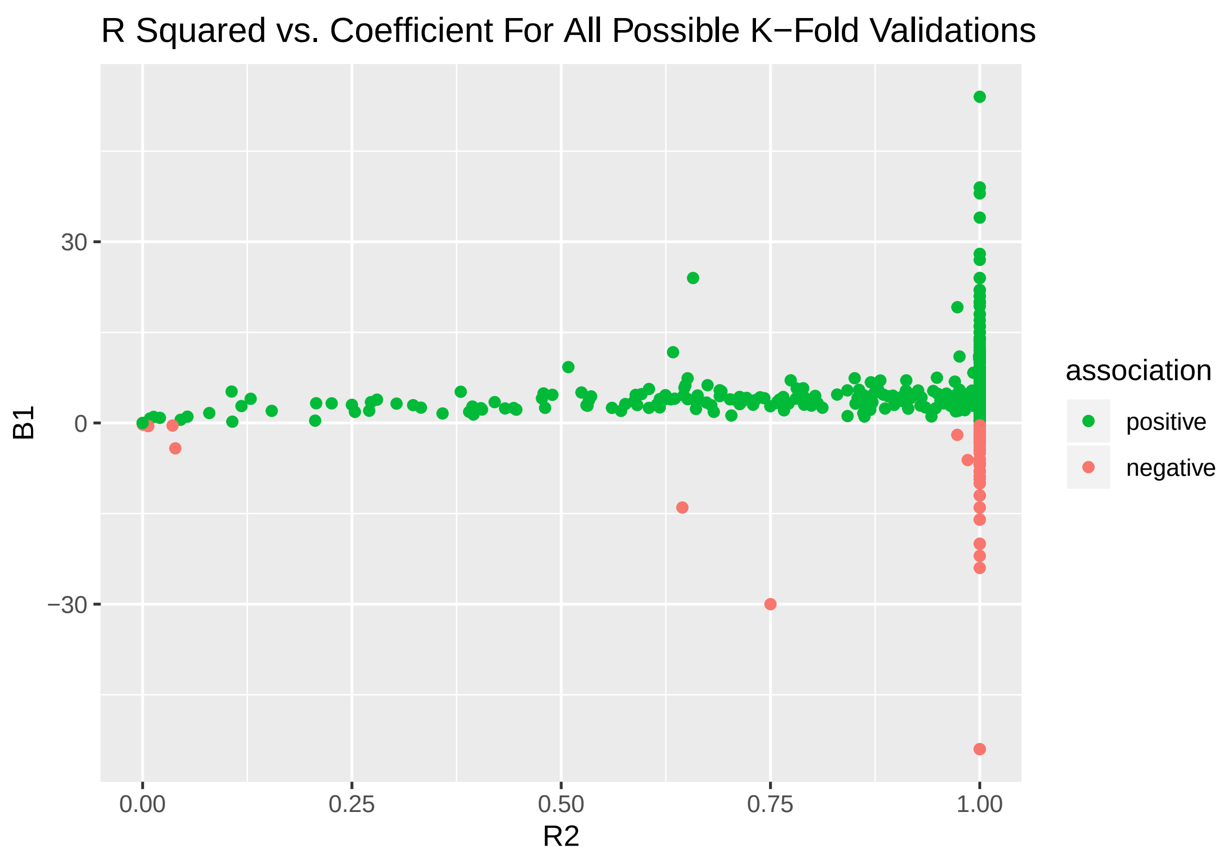


Here we can see the benefit of cross validation plainly illustrated. Across all possible k fold validations, our r square distribution is heavily skewed towards 1, representing the ability of the cross validation process to remove noise and form a strong predictive model.

Finally, let's take a look at the plot of R Squared values across all possible k fold validations:

```
#bind slope and R2 to compare
compare <- cbind(allSlope, allR2)
compare$association <- ifelse(compare$B1 > 0, "positive", "negative")
compare <- compare[complete.cases(compare),]
```

```
gg <- ggplot(compare, aes(x=R2, y=B1, color = association)) +
      scale_color_manual(values=c("#F8766D", "#00BA38"),breaks=c("positive","negative")) +
      geom_point() +
      labs(title = "R Squared vs. Coefficient For All Possible K-Fold Validations")

plot(gg)
```



Again, I have added color to indicate predicted positive/negative association. This time however, we see several regressions that predict negative association, which is obviously incorrect. So why is this?

Imagine if I took training sets of just two or three points, and they happened to have a negative association. Because we selected such a small training set, it was not indicative of the overall trends in our data. While this is a silly example, it is representative of the real issue of selecting a k value big enough to capture the trends in our data, while also small enough to ease computationaly complexity and bias.

In conclusion, even in the simplest possible case of linear regression with one predictor, cross validation requires careful thought towards the methods by which we select our validation method and careful use of tools such as repeated validation and averaging to mitigate anomalous predictions.