STAT 5525: Homework 1

Adam Wells

Monday, June 15

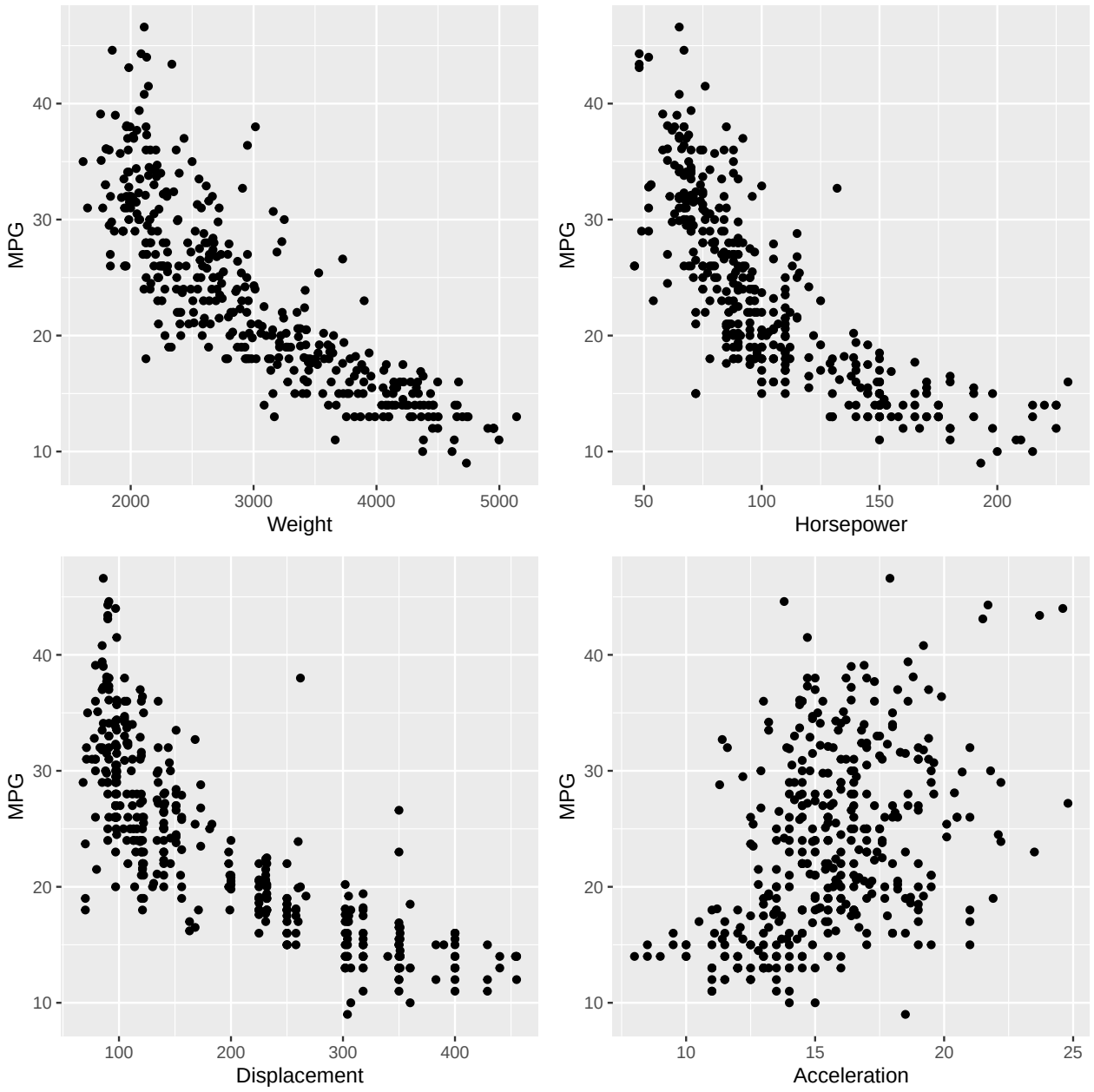
Assignment

In class we demonstrated how to use cross-validation to evaluate the performance of 3 polynomial regression models for predicting mpg from horsepower (hpw) using MSE as the criterion. Select an alternative predictor, such as weight for example (you are not required to use weight), to develop an alternative regression model and compare its performance to the quadratic model using hpw. First evaluate alternatives using the new predictor, such as polynomial models with the new predictor, to find the best model using that predictor, then compare it to the quadratic model. (If you have time, you could investigate using both hpw and your new predictor in a multiple regression model but you are not required to do so.) Report your results in an RMarkdown notebook similar to the lab5 notebook. Use ggplot graphics as well as a narrative to explain why you chose your particular predictor and to report your results. You are free to reuse the code in the lab5 notebook, and encouraged to do so, but are not required to do so.

Visualizing Predictors

The plots below illustrate the relationship between mpg and four predictor variables: weight, horsepower, displacement, and acceleration. Weight, horsepower, and displacement appear to have a similar relationship to mpg : all three are *negatively* correlated with mpg, the spread of the data is similar, and the plots are curved (non-linear). Acceleration (time 0-60), on the other hand, is *positively* correlated with mpg, and the data is more diffuse. In this assignment, I will compare various models using horsepower and weight as predictors.

```
auto <- Auto %>%
  as_tibble() %>%
  rename(cyls = cylinders, dpl = displacement, hpw = horsepower, accl = acceleration)

p<-ggplot(data=auto, mapping=aes(x = weight, y = mpg))
p<-p+geom_point()
p<-p+labs(x="Weight",y="MPG")

q<-ggplot(data=auto, mapping=aes(x = hpw, y = mpg))
q<-q+geom_point()
q<-q+labs(x="Horsepower",y="MPG")

r<-ggplot(data=auto, mapping=aes(x = dpl, y = mpg))
r<-r+geom_point()
r<-r+labs(x="Displacement",y="MPG")

s<-ggplot(data=auto, mapping=aes(x = accl, y = mpg))
s<-s+geom_point()
s<-s+labs(x="Acceleration",y="MPG")

grid.arrange(p,q,r,s,nrow=2,ncol=2)
```



Setting up Test Data

20% of the data are randomly selected to serve as test data. The remaining 80% will be used for training models.

```
set.seed(99)
test.indices <- sample(1:nrow(auto), 0.2 * nrow(auto))
test.data <- auto[test.indices, ]
training.data <- auto[-test.indices, ]
```

Setting up Models

I will evaluate 7 models to see which models are better predictors of MPG. In subsequent sections I will refer to the model numbers listed below:

1. A quadratic model using hpw: mpg~hpw^2+hpw
2. A simple linear model using weight: mpg~weight
3. A quadratic model using weight: mpg~weight^2+weight
4. A cubic model using weight: mpg~weight^3+weight^2+weight
5. A simple linear multiregression model using weight and hpw: mpg~hpw+weight
6. A quadratic multiregression model using weight and hpw:mpg~weight^2+weight+hpw^2+hpw
7. A cubic multiregression model using weight and hpw: mpg~weight^3+weight^2+weight+hpw^3+hpw^2+hpw

```
model1<-lm(mpg ~ poly(hpw, 2, raw = T), training.data)
model2 <- lm(mpg ~ weight, training.data)
model3 <- lm(mpg ~ poly(weight, 2, raw = T), training.data)
model4 <- lm(mpg ~ poly(weight, 3, raw = T), training.data)
model5<-lm.fit <- lm(mpg ~ weight+hpw, training.data)
model6<-lm(mpg ~ poly(weight, 2, raw = T)+poly(hpw, 2, raw = T), training.data)
model7<-lm(mpg ~ poly(weight, 3, raw = T)+poly(hpw, 3, raw = T), training.data)
```

Validating Models (K-fold Cross Validation)

10-fold cross validation was performed on all of the above models. The following chart and graph show the training and validation MSE values for each model. The quadratic model using weight (model 3) has a lower training and validation MSE than the quadratic model using hpw (model 1). Additionally, the multiple regression quadratic and cubic models (models 6 and 7) are the best fit for the training data.

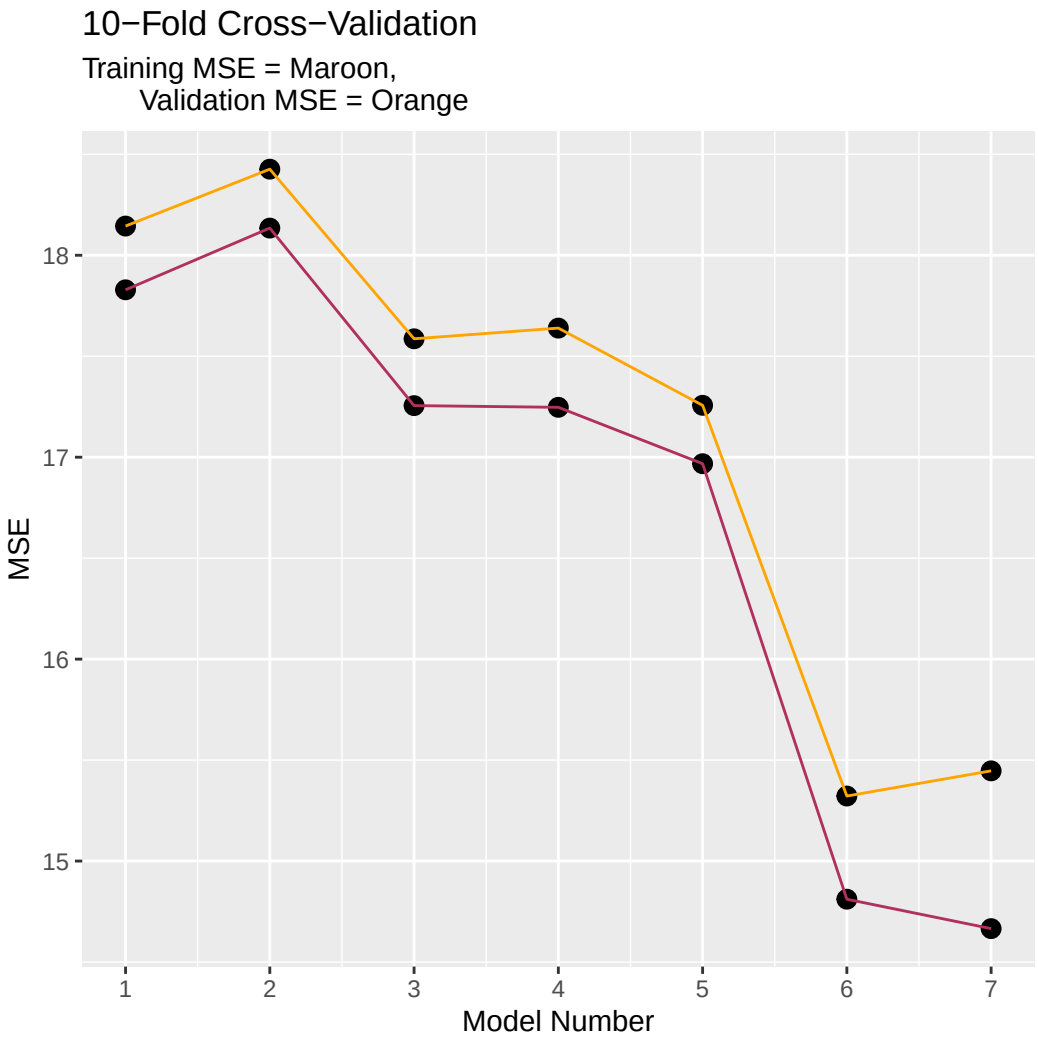
```
k.fold.cross.validator <- function(df, model.formula, K) {
  set.seed(99)

  # this function calculates the MSE of a single fold using the fold as the holdout data
  fold.mses <- function(df, model.formula, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]
    fit <- glm(model.formula, data = train.data)
    tibble(train.mse = mse(fit, train.data), valid.mse = mse(fit, holdout.data))
  }

  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))
  # if argument K == 1 we want to do LOOCV
  if (K == 1) {
    K <- nrow(df)
  }
  folds <- cut(indices, breaks = K, labels = F)
  # convert "model.formula" from, character to a formula
  model.formula <- formula(model.formula)
  # set error to 0 to begin accumulation of fold MSEs
  mses <- tibble()
  # iterate on the number of folds
  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.mse <- fold.mses(df, model.formula, holdout.indices)
    mses <- mses %>%
```

```
      bind_rows(folded.mse)
  }
  mses %>%
    summarize(train.mse = mean(train.mse), valid.mse = mean(valid.mse))
}
```

model	train.mse	valid.mse
1	17.83	18.14
2	18.13	18.43
3	17.26	17.59
4	17.25	17.64
5	16.97	17.26
6	14.81	15.32
7	14.67	15.45

Here is a plot of the results:

```
df2 <- df %>%
  pivot_longer(c(train.mse, valid.mse), names_to = "test", values_to = "mse")

df2 %>% ggplot()+
  geom_point(aes(x = model, y = mse), data = df2, size = 3) +
  geom_line(aes(x = model, y = mse), data = df2 %>% filter(test == "train.mse"),
            color = "maroon") +
  geom_line(aes(x = model, y = mse), data = df2 %>% filter(test == "valid.mse"),
            color = "orange") +
  xlab("Model Number") +
  ylab("MSE") +
  scale_x_continuous(breaks = c(1:7))+
  labs(title = "10-Fold Cross-Validation", subtitle = "Training MSE = Maroon,
       Validation MSE = Orange")
```



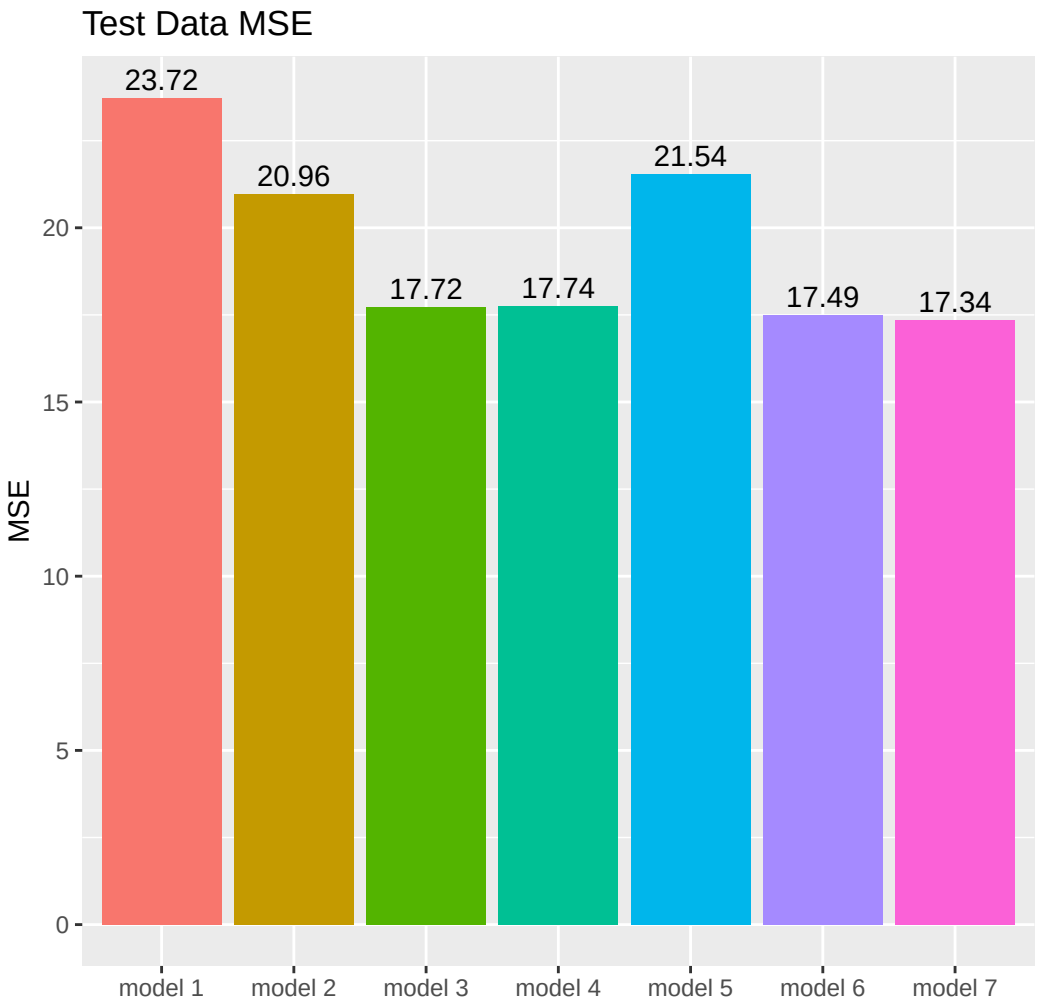
Testing Models

In this section, I will compare the MSE of each model when applied to the testing data. I will also use a re-sampling procedure to evaluate the range of possible MSEs for different subsets of testing data.

The graph below shows that models 3, 4, 6, and 7 have very similar MSEs when applied to the test data. Model 7 (cubic multiregression model using weight and hpw) has the lowest MSE. It is, however, important to note that the MSE is highly dependent on the selection of test data and training data. If the data were randomly re-divided into testing and training data, the MSE values could change dramatically.

```
mse<-data.frame(mse(model1,test.data),
                mse(model2,test.data),
                mse(model3,test.data),
                mse(model4,test.data),
                mse(model5,test.data),
                mse(model6,test.data),
                mse(model7,test.data))

colnames(mse)<-c("model 1","model 2","model 3", "model 4", "model 5", "model 6",
                 "model 7")
df3 <- pivot_longer(mse, cols=c("model 1","model 2","model 3", "model 4", "model 5",
```

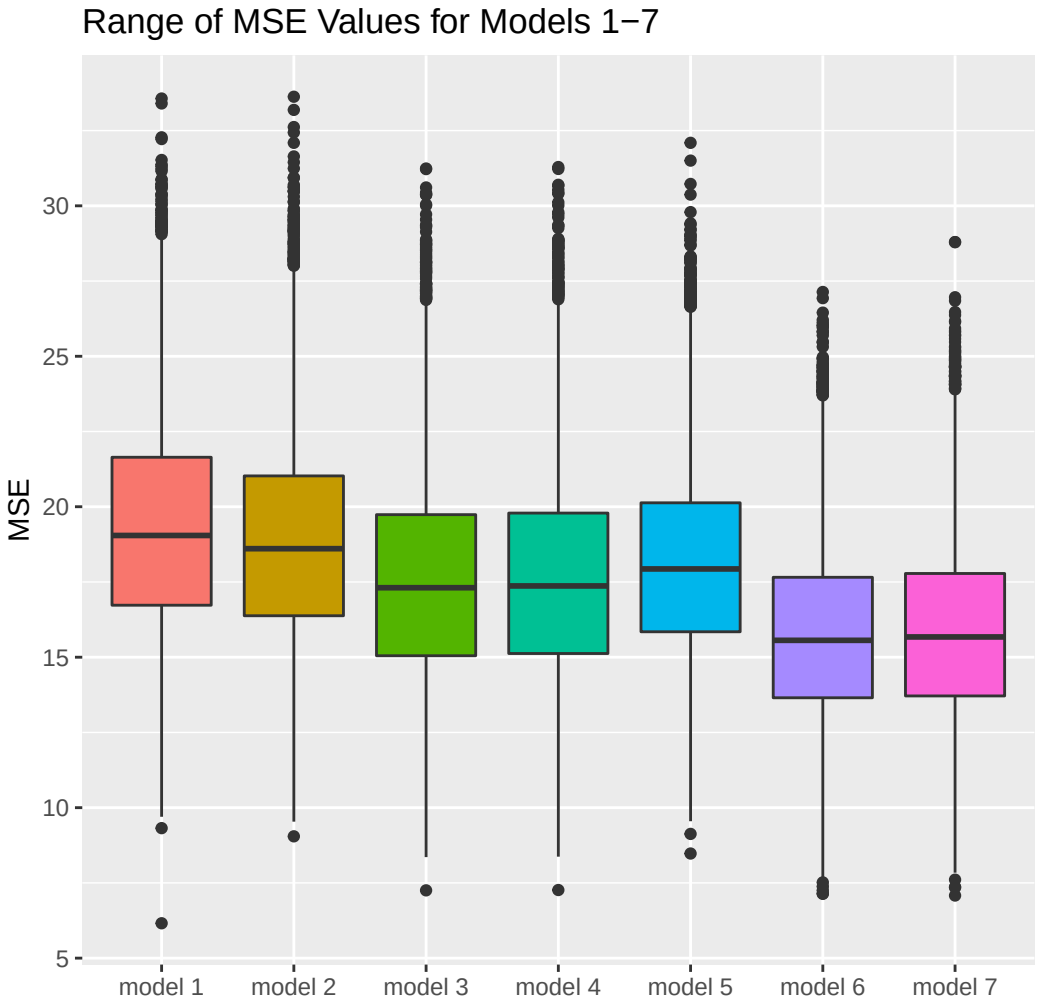
```
                                "model 6", "model 7"),
                    names_to='Models',
                    values_to="MSE")

df3%>%ggplot(aes(x=Models, y=MSE, fill=Models))+
geom_bar(stat='identity')+
geom_text(aes(label=round(MSE,2), position="stack", vjust=-0.4))+
theme(legend.position="none")+
labs(title = "Test Data MSE",x="")
```



The MSEs in the previous graph are very dependent on the particular subsets of data selected to test and train the models. We originally set aside test data by randomly sampling 20% of the auto data. If we were to perform that sampling procedure again, thereby selecting a different subset of the data, we would get different MSE values. The graph below shows the range of MSE values produced when the testing and training subsets are re-sampled and the models are refitted. The process was repeated 10,000 times. This gives a sense of the range of possible MSE values for each model.

```r
#This function is a bootstrap-like function (without replacement) that divides the data
#into testing and training data.  It creates a linear model based on the training data
#and determines the MSE of that model when applied to the test data.

Test_boot<-function(data,formula,R){
xbar.boot <- rep(0, R)
set.seed(99)
for (i in 1:R){
  test.indices2 <- sample(1:nrow(data), 0.2*nrow(data), replace = F)
  test.data <- data[test.indices2, ]
  training.data <- data[-test.indices2, ]
  model<-lm(data = training.data,formula = formula)
  xbar.boot[i] <- mse(model,test.data)
}
return(xbar.boot)
}

#This function takes a character vector of linear model formulas and calls the Test_boot
#function on each member of the vector.  It aggregates the results into a dataframe.

MSE_boot<-function(data,formulas,R){
MSE<-c()
for(i in 1:length(formulas)){
  MSE[[i]]<-Test_boot(data,formulas[i],R)
}
MSE<-data.frame(MSE)
colnames(MSE)<-c(1:length(formulas))
return(MSE)
}

formulas<-c("mpg ~ poly(hpw, degree = 2, raw = T)",
            "mpg ~ poly(weight,degree=1,raw=T)",
            "mpg ~ poly(weight, degree = 2, raw = T)",
            "mpg ~ poly(weight, degree = 3, raw = T)",
            "mpg ~ poly(weight,degree=1,raw=T)+poly(hpw,degree=1,raw=T)",
            "mpg ~ poly(weight, degree = 2, raw = T)+poly(hpw,degree=2,raw=T)",
            "mpg ~ poly(weight, degree = 3, raw = T)+poly(hpw,degree=3,raw=T)")

MSEsummary<-MSE_boot(auto,formulas = formulas,R=10000)
colnames(MSEsummary)<-c("model 1","model 2","model 3", "model 4", "model 5", "model 6",
                        "model 7")
```

```r
data_long<-pivot_longer(MSEsummary,
                        cols=c("model 1","model 2","model 3", "model 4", "model 5",
                               "model 6","model 7"),
                        names_to="models")

p<-ggplot(data=data_long,mapping=aes(x=as.factor(models), y=value, fill=models))
p<-p+geom_boxplot()
p<-p+scale_y_continuous(breaks = seq(5,35,5))
p<-p+labs(x="",y="MSE",title = "Range of MSE Values for Models 1-7")
p<-p+theme(legend.position="none")
p
```



Conclusion

Based on the training, testing, and validation procedures used above, we can draw the following two conclusions. 1) Of the four models with single predictors (hpw *or* weight), models 3 and 4 (quadratic and cubic models using weight) tend to have have lower median MSEs than model 1 (quadratic model using hpw). Weight therefore performs better than horsepower as an individual predictor of mpg. 2) Out of all the models, 6 and 7 have the lowest median MSEs. Consequently, if I were choosing models, I would choose the simpler of th two–namely, model 6, which is a quadratic multiregression model using weight and hpw: mpg~weight^2+weight+hpw^2+hpw.