StatR 201: Winter 2013

Homework 04

Rod Doe

March 05, 2013

Data Preparation and Exploration

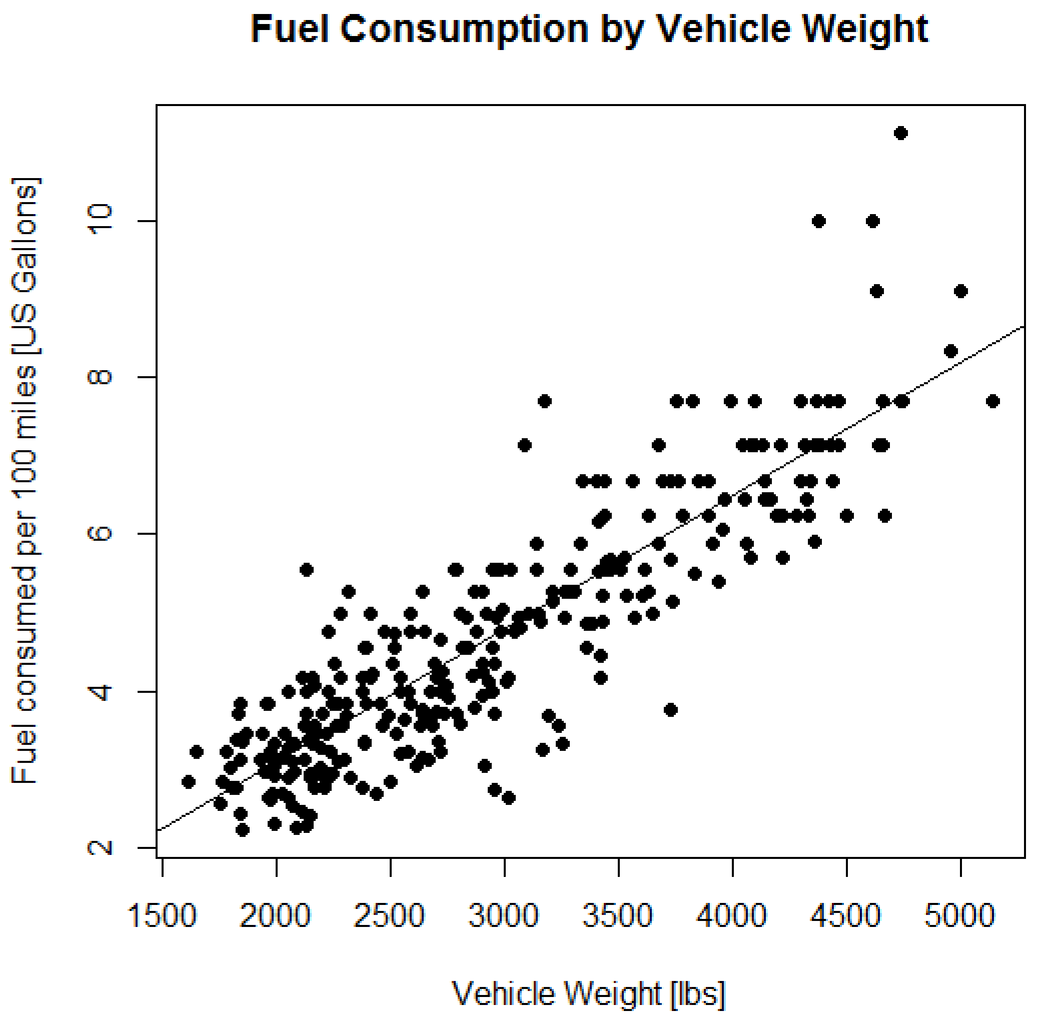
I implemented these transformations on the data:

- Created column gp100m as 100/(column mpg).
- I added column engine type after discovering that many of the rows in the data set referred to engines that used the Wankel rotary piston engine, in lieu of a typical reciprocating piston engine.
- Changed these columns to factors:
  - Continent
  - Diesel
  - Cylinders

The following are graphs I would use to present these data to an audience that may not contain subject matter experts.

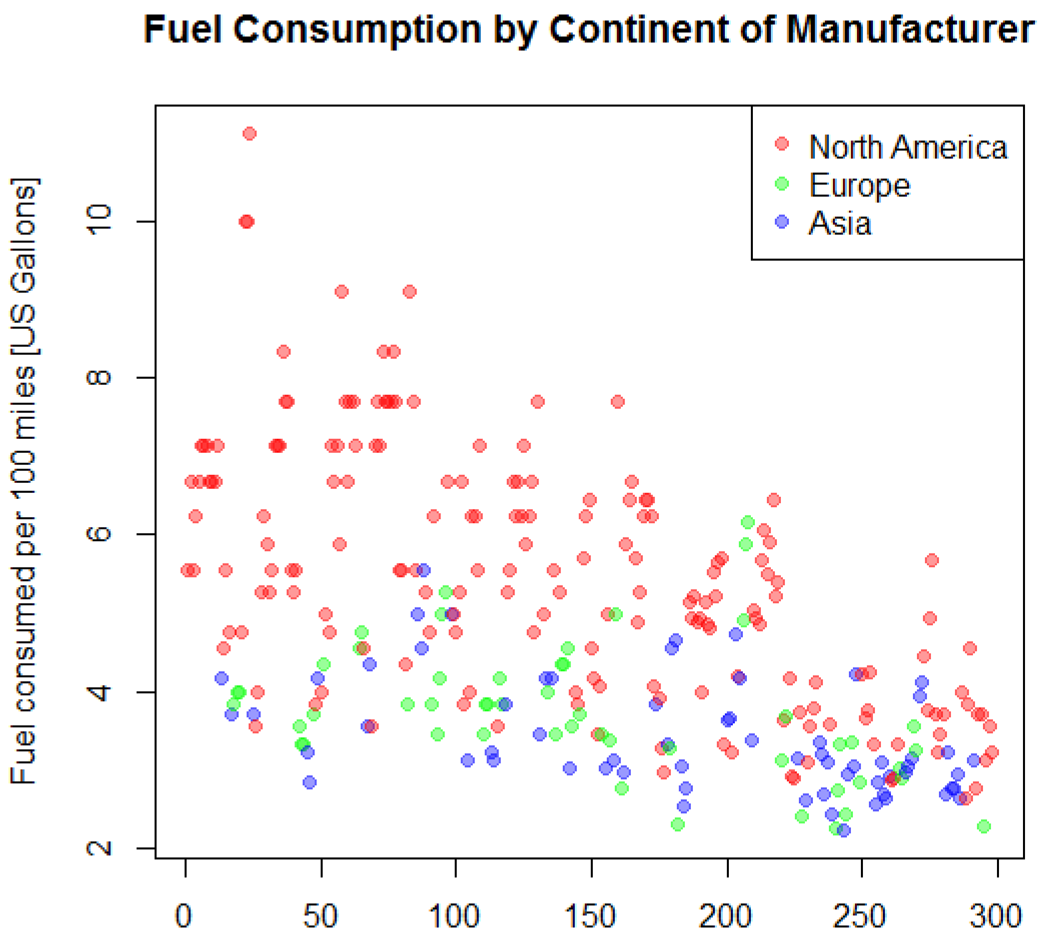
Fuel Consumption by Vehicle Weight

Instinctively, we would expect fuel consumption to increase with vehicle weight. Indeed, the data support this expectation.



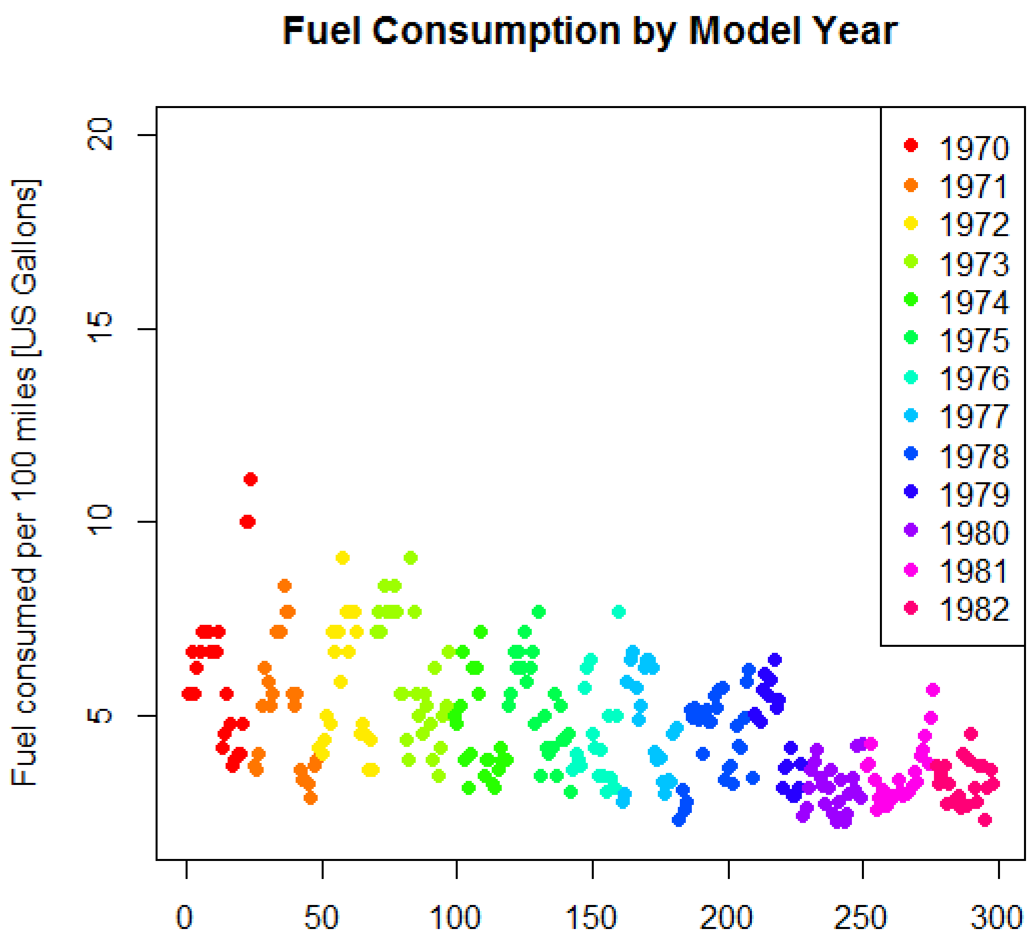
Fuel Consumption by Continent of Manufacturer

This graph shows that the cars made in North America tend to have higher fuel consumption, whereas cars made in Europe or Asia tend to have lower fuel consumption. This may be due to vehicle size or other factors.



Fuel Consumption by Model Year

The data show that fuel consumption generally decreased by model year. Historically, this was the period of the Arab Oil Embargo, so there was ample incentive for the industry to improve fuel efficiency.

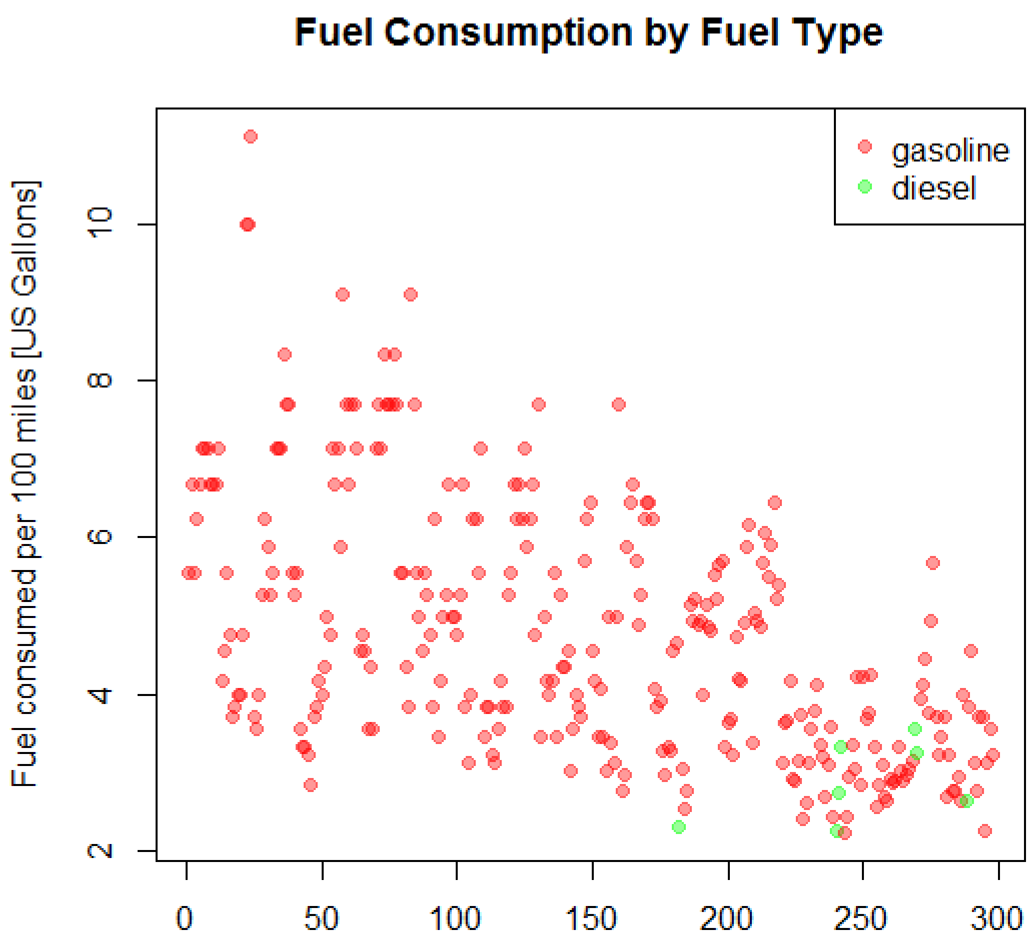


Fuel Consumption by Fuel Type

Some of the test vehicles used diesel fuel, and we should expect them to have superior fuel efficiency. This is because diesel fuel has a higher volumetric energy density. Here is an excerpt from WikiPedia on the subject of diesel fuel energy density:

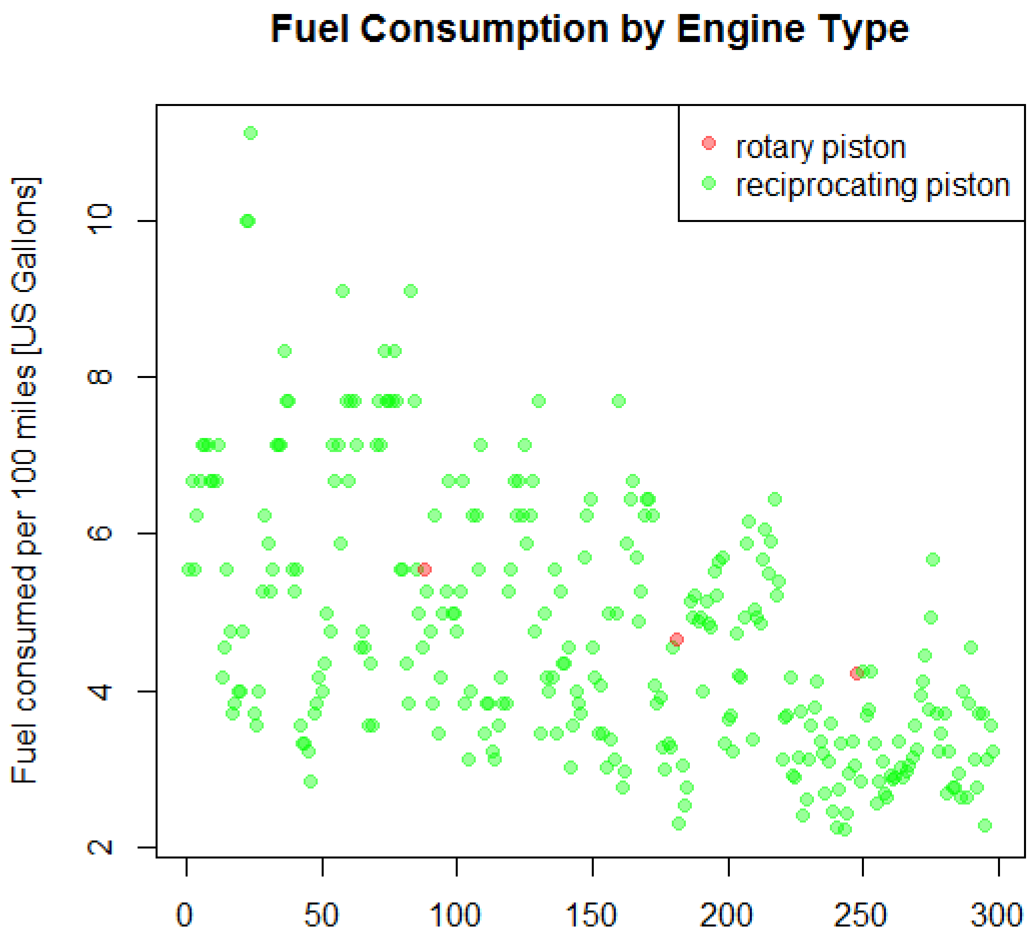
"…due to the higher density, diesel offers a higher volumetric energy density at 35.86 MJ/L (128 700 BTU/US gal) vs. 32.18 MJ/L (115 500 BTU/US gal) for gasoline, some 11% higher, which should be considered when comparing the fuel efficiency by volume."

Our data indicate that the diesel vehicles exhibited some of the lowest fuel consumption rates.



Fuel Consumption by Engine Type

Most of the vehicles in this dataset used reciprocating piston internal-combustion engines, in which a round piston travels up and down in a cylinder. However, some of the Mazda vehicles use Wankel rotary engines, in which triangular piston rotates an eccentric crankshaft in a lobed cylinder. The rotary engine in these vehicles used the same fuel, so the physics of the problem suggest (the same fuel exhausted to the same ambient temperature thereby limiting the amount of energy that can be extracted from the fuel) that we should not expect significant difference in fuel consumption. The rotary engine is more compact and probably lighter than the reciprocating engine, so there may be some benefits due to decreased vehicle weight. However, the data do not show any general trends that at first glance suggest any clear benefits.



Given the apparent unremarkable effect of engine type on fuel consumption, the engine type will not be included as a factor in further analysis.

Data Transformation Function

Given a test autos dataset, run the dataset through this function to prepare it for analysis:

```
transformAutos <- function(autos) {
  autos.clean = data.frame(cbind(100/autos$mpg, autos$cyl, autos$volume,
autos$hp, autos$weight, autos$accel ,autos$year, autos$continent,
autos$diesel))
  colnames(autos.clean) = c("gp100m", "cyl", "volume", "hp", "weight",
"accel", "year", "continent", "diesel")

  # Establish factors in data.
  autos.clean$continent = factor(autos.clean$continent)  # North America,
Europe, Asia
  autos.clean$diesel = factor(autos.clean$diesel) # gasoline, diesel
  autos.clean$cyl = factor(autos.clean$cyl) # Number of cylinders is non-
continuous
  autos.clean$year = factor(autos.clean$year) # Model years are non-
continuous.

  # And return the data frame.
  autos.clean
}
```

Troubling VIFs

```
transformAutos <- function(autos) {
  autos.clean = data.frame(cbind(100/autos$mpg, autos$cyl, autos$volume, autos$hp, autos$weight,
autos$accel ,autos$year, autos$continent, autos$diesel))
  colnames(autos.clean) = c("gp100m", "cyl", "volume", "hp", "weight", "accel", "year", "continent",
"diesel")

  # Establish factors in data.
  autos.clean$continent = factor(autos.clean$continent)  # North America, Europe, Asia
  autos.clean$diesel = factor(autos.clean$diesel) # gasoline, diesel
  autos.clean$cyl = factor(autos.clean$cyl) # Number of cylinders is non-continuous
  autos.clean$year = factor(autos.clean$year) # Model years are non-continuous.

  # And return the data frame.
  autos.clean
}

autos.clean = transformAutos(autos)
```

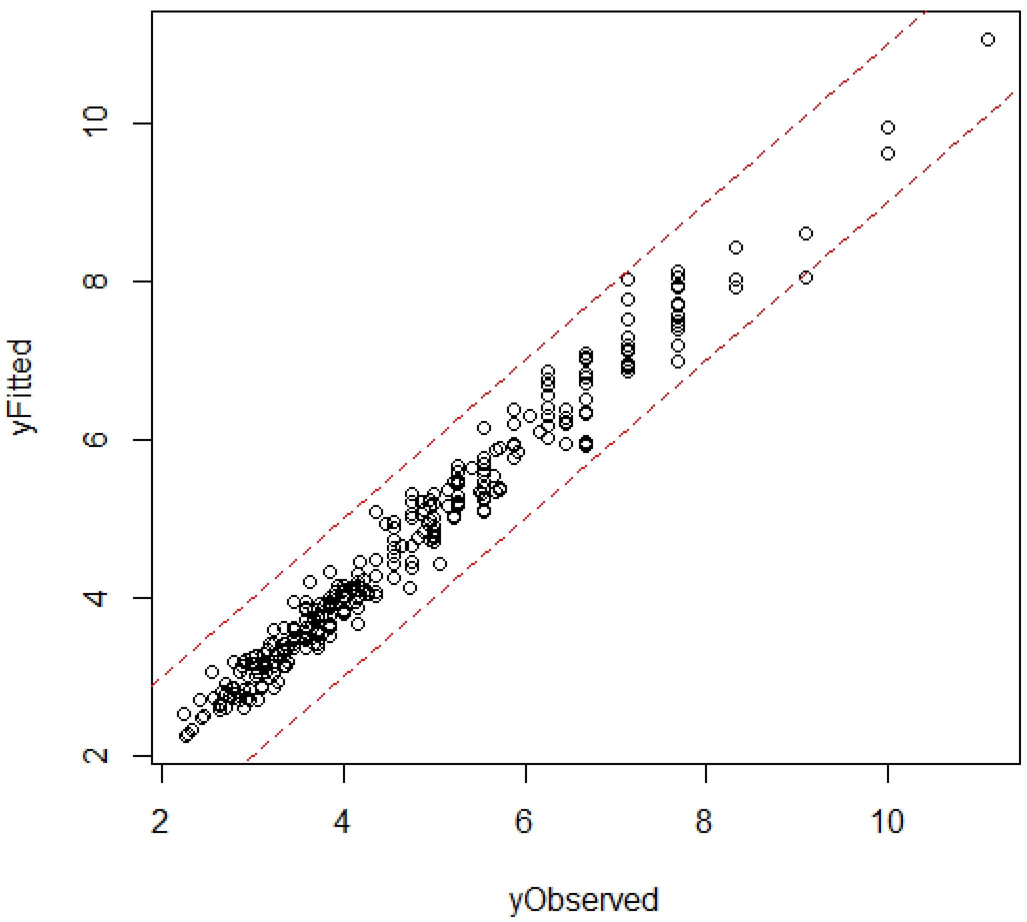
```
# Gets non-factor covariates only (no response)
autos.x = autos.clean[, !(colnames(autos.clean) %in% c("gp100m", "cyl", "year", "continent", "diesel" ))]

 vif(autos.x)
     covariate vif
[1,] "volume"  "10.9394383263869"
[2,] "hp"      "8.77933159369108"
[3,] "weight"  "9.3811706364795"
[4,] "accel"   "2.4990248104742"
```

These are quite high. My hands, however, are tied.

Fitting Models

Simple lm with excessive VIFs in data

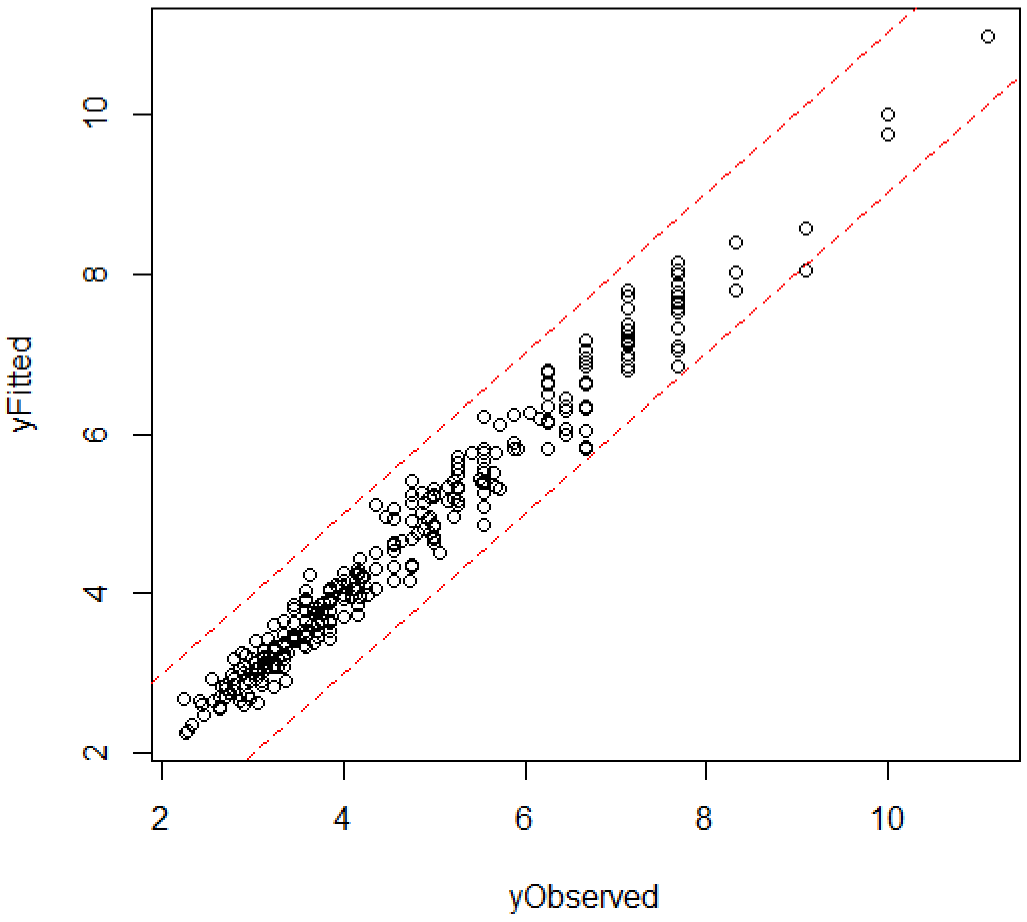
```
# Fit a simple lm model with excessive VIF.
fit.lm = lm(formula=formula, data=autos.clean)
plotFit(autos.clean$gp100m, fitted(fit.lm))
summary.lm(fit.lm)
```



```
Residual standard error: 0.3688 on 152 degrees of freedom
Multiple R-squared: 0.9739,     Adjusted R-squared: 0.949
F-statistic: 39.12 on 145 and 152 DF,  p-value: < 2.2e-16
```

Simple lm model with step="both", AIC

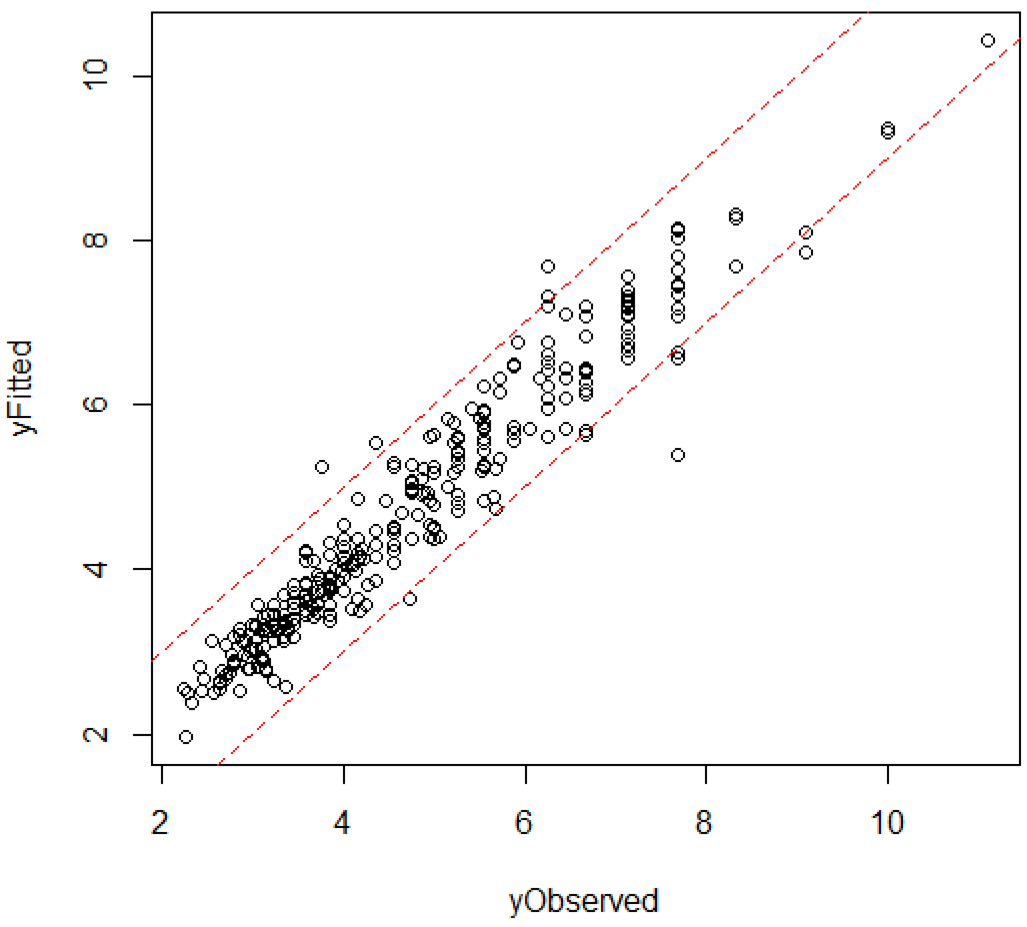
```
fit.lm.step.both.aic = step(fit.lm, direction=c("both"), k=2)
plotFit(autos.clean$gp100m, fitted(fit.lm.step.both.aic))
summary.lm(fit.lm.step.both.aic)
```



```
Residual standard error: 0.3591 on 180 degrees of freedom
Multiple R-squared: 0.9707,     Adjusted R-squared: 0.9517
F-statistic: 50.98 on 117 and 180 DF,  p-value: < 2.2e-16
```

Simple lm model with step="both", BIC

```
# Fit a step = both, with BIC and excessive VIF.
fit.lm.step.both.bic = step(fit.lm, direction=c("both"),
k=log(length(autos.clean[,1])))
plotFit(autos.clean$gp100m, fitted(fit.lm.step.both.bic))
summary.lm(fit.lm.step.both.bic)
```



```
Residual standard error: 0.4499 on 265 degrees of freedom
Multiple R-squared: 0.9323,     Adjusted R-squared: 0.9241
F-statistic: 114.1 on 32 and 265 DF,  p-value: < 2.2e-16
```

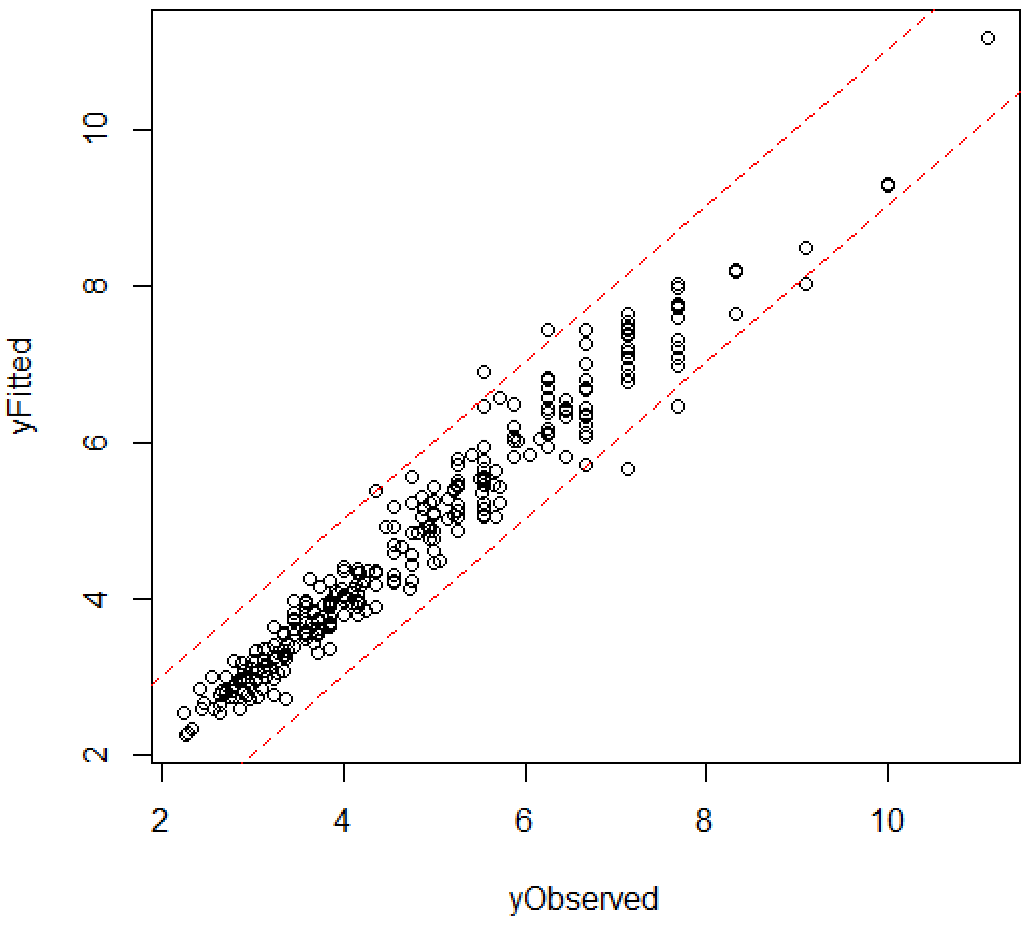
Filter autos.clean for excessive VIF

```
# New transform function without volume and hp
transformAutos.VIF <- function(autos) {
  autos.clean = data.frame(cbind(100/autos$mpg, autos$cyl, autos$weight,
autos$accel ,autos$year, autos$continent, autos$diesel))
  colnames(autos.clean) = c("gp100m", "cyl", "weight", "accel", "year",
"continent", "diesel")

  # Establish factors in data.
  autos.clean$continent = factor(autos.clean$continent)
  autos.clean$diesel = factor(autos.clean$diesel) # gasoline, diesel
  autos.clean$cyl = factor(autos.clean$cyl)  autos.clean$year =
factor(autos.clean$year)
  # And return the data frame.
  autos.clean
}
```

```
> autos.vifClean = transformAutos.VIF(autos)
> str(autos.vifClean)
'data.frame':   298 obs. of  7 variables:
 $ gp100m   : num  5.56 6.67 5.56 6.25 6.67 ...
 $ cyl      : Factor w/ 5 levels "3","4","5","6",..: 5 5 5 5 5 5 5 5 5 5 ...
 $ weight   : num  3504 3693 3436 3433 4341 ...
 $ accel    : num  12 11.5 11 12 10 9 8.5 10 8.5 10 ...
 $ year     : Factor w/ 13 levels "70","71","72",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ continent: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ diesel   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Simple lm with excessive VIFs removed

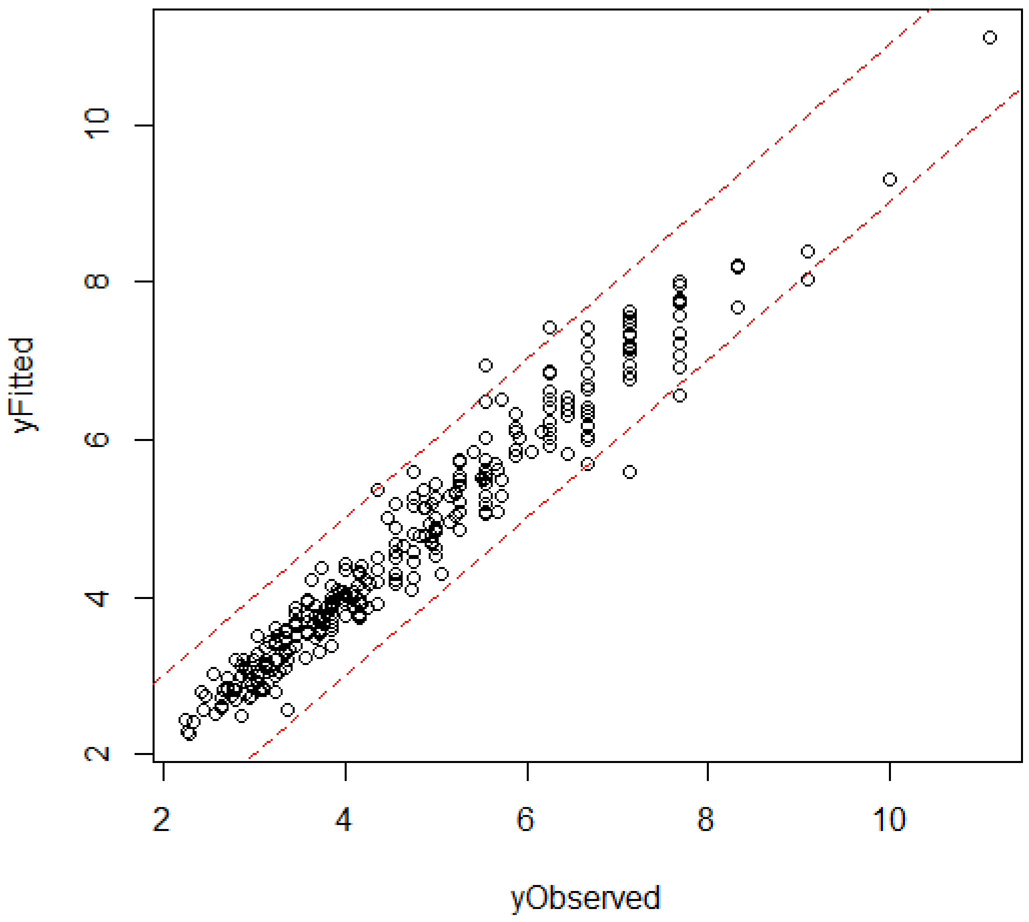
```
# Fit a simple lm model with excessive VIF.
fit.lm.vif.clean = lm(formula=formula, data= autos.vifClean)
plotFit(autos.clean$gp100m, fitted(fit.lm.vif.clean))
summary.lm(fit.lm.vif.clean)
```



```
Residual standard error: 0.4275 on 191 degrees of freedom
Multiple R-squared: 0.9559,     Adjusted R-squared: 0.9315
F-statistic:  39.1 on 106 and 191 DF,  p-value: < 2.2e-16
```

Simple lm model with step="both", excessive VIFs removed, AIC

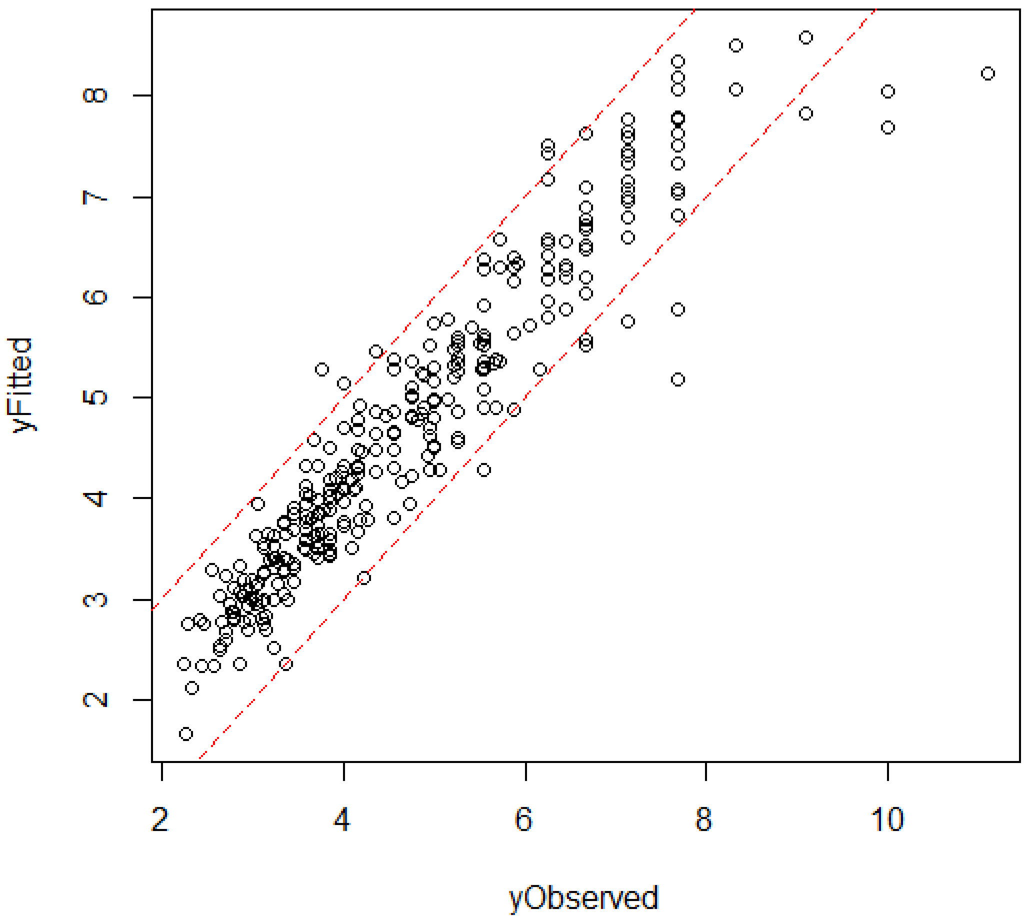
```
fit.lm.vif.clean.step.aic = step(fit.lm.vif.clean,
direction=c("both"), k=2)
plotFit(autos.vifClean$gp100m, fitted(fit.lm.vif.clean.step.aic))
summary.lm(fit.lm.vif.clean.step.aic)
```



```
Residual standard error: 0.4221 on 203 degrees of freedom
Multiple R-squared: 0.9544,     Adjusted R-squared: 0.9332
F-statistic: 45.15 on 94 and 203 DF,  p-value: < 2.2e-16
```

Simple lm model with step="both", excessive VIFs removed, BIC

```
fit.lm.vif.clean.step.bic = step(fit.lm.vif.clean,
direction=c("both"), k=log(length(autos.clean[,1])))
plotFit(autos.vifClean$gp100m, fitted(fit.lm.vif.clean.step.bic))
summary.lm(fit.lm.vif.clean.step.bic)
```



```
Residual standard error: 0.5464 on 283 degrees of freedom
Multiple R-squared: 0.8934,     Adjusted R-squared: 0.8881
F-statistic: 169.4 on 14 and 283 DF,  p-value: < 2.2e-16
```

Summary of Fit Results

Model summary	**Scaled**	**Covariates with high VIFs in data**	**Adjusted R-squared**	**Comments** [1]
Simple lm	No	Yes	0.9490	Nice fit
lm, step=”both”, AIC	No	Yes	0.9517	Nice fit
lm, step=”both”, BIC	No	Yes	0.9241	1 outlier
Simple lm	No	No	0.9315	5 outliers
lm, step=”both”, AIC	No	No	0.9332	5 outliers
lm, step=”both”, BIC	No	No	0.8881	14 outliers

[1] For the purpose of this discussion, an outlier is defined as a point graphed on an observed-fitted plot that falls outside of a-b lines with intercepts of 1 and -1 and slope of 1.

Discussion of fit results:

These results are somewhat counter-intuitive. Specifically:

- A simple linear model gave nice results.
- The step function, which is supposed to simplify the mode, did not.
- Removal of covariates with high Variance Inflation Factors (VIFs) did not improve the model.

Cross Validation

I tried to apply cross validation to my data according to a 5-fold Hastie model, and hit a problem. Factors in the Kth test set did not exist in the other K -1 sets, over which the model was generated. As such, the model could not predict values on factors it did not know about. This required me to remove factors from my data and refit.

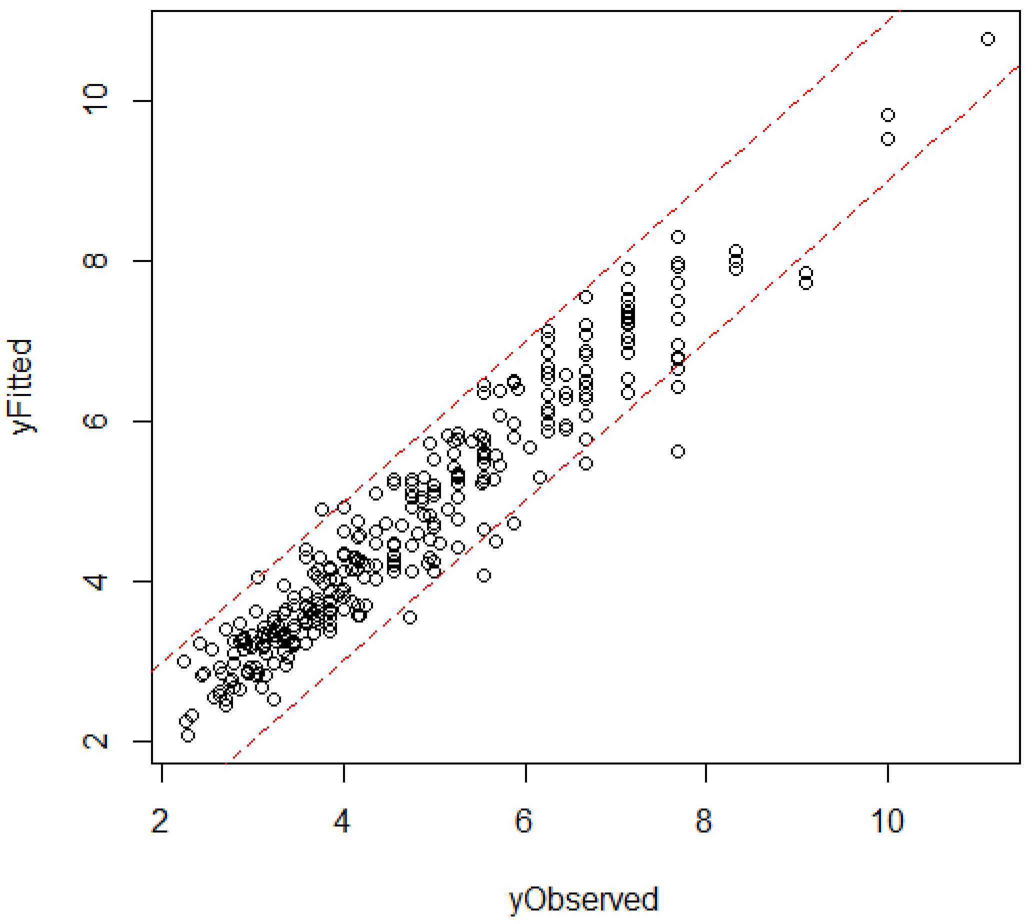
Here is the resultant transform function:

```
transformAutos.sans.factors <- function(autos) {
  autos.clean = data.frame(cbind(100/autos$mpg, autos$cyl,
autos$volume, autos$hp, autos$weight, autos$accel ,autos$year,
autos$continent, autos$diesel))
  colnames(autos.clean) = c("gp100m", "cyl", "volume", "hp", "weight",
"accel", "year", "continent", "diesel")

  # Establish factors in data.
  #autos.clean$continent = factor(autos.clean$continent)  # North
America, Europe, Asia
  #autos.clean$diesel = factor(autos.clean$diesel) # gasoline, diesel
  #autos.clean$cyl = factor(autos.clean$cyl) # Number of cylinders is
non-continuous
  #autos.clean$year = factor(autos.clean$year) # Model years are non-
continuous.

  # And return the data frame.
  autos.clean
}
```

Here are the new fits.

Simple lm without factors

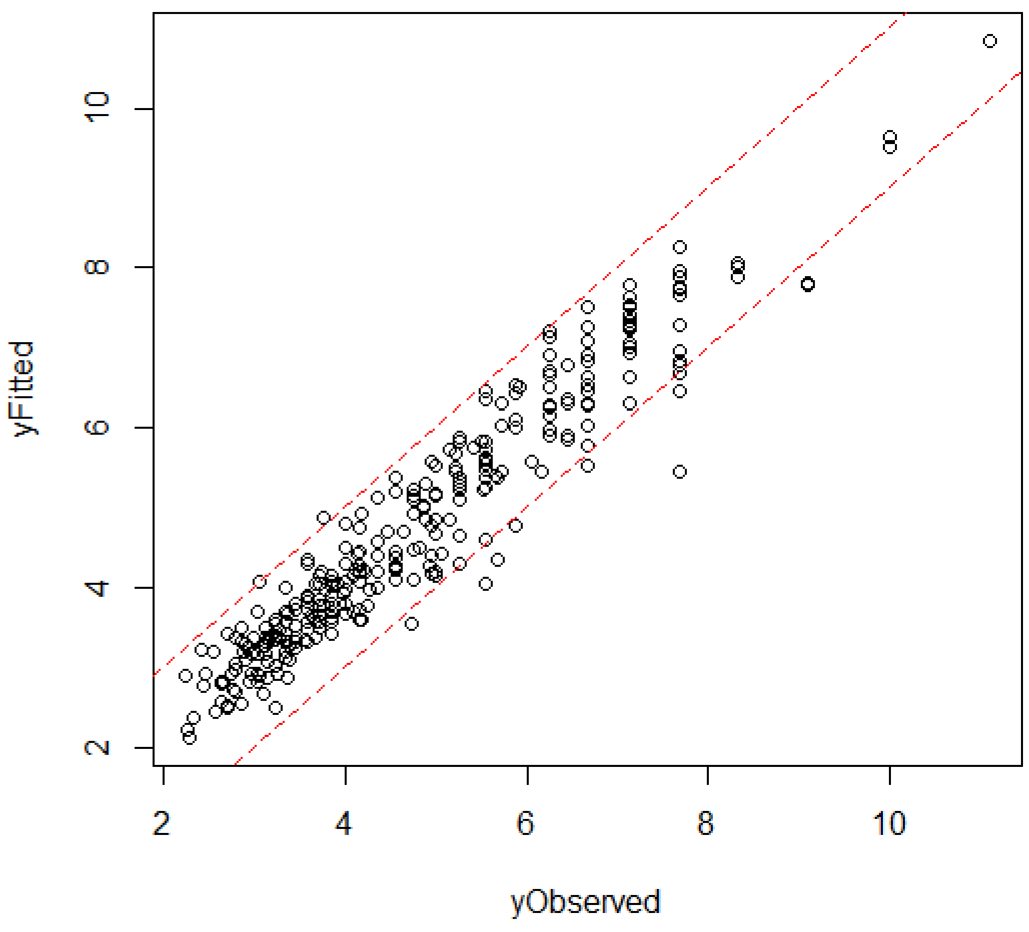
```
# Fit a simple lm model with excessive VIF.
fit.lm.factorless = lm(formula=formula, data=autos.clean.factorless)
plotFit(autos.clean.factorless$gp100m, fitted(fit.lm.factorless))
summary.lm(fit.lm.factorless)
```



```
Residual standard error: 0.4899 on 262 degrees of freedom
Multiple R-squared: 0.9206,     Adjusted R-squared:  0.91
F-statistic: 86.84 on 35 and 262 DF,  p-value: < 2.2e-16
```

lm with step = "both", high VIFs, AIC, and no factors

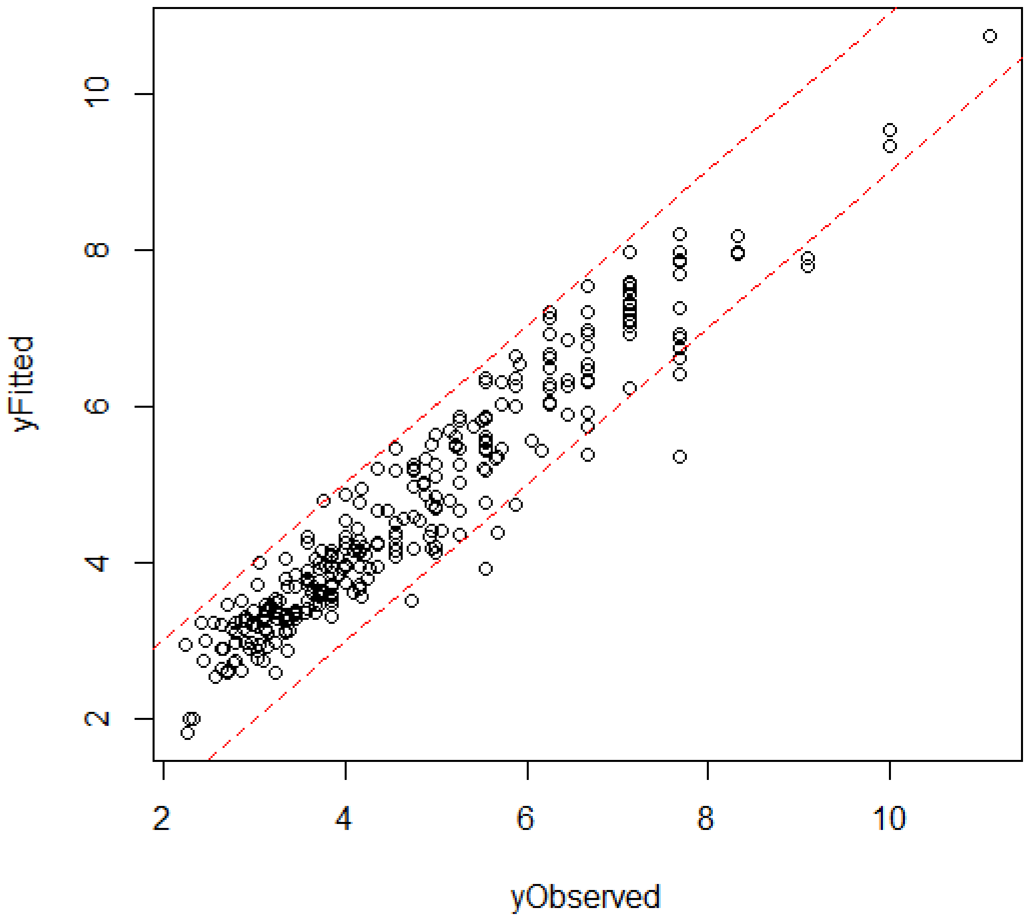
```
# Fit a step = both, with AIC and excessive VIF.
fit.lm.factorless.step.both.aic = step(fit.lm.factorless,
direction=c("both"), k=2)
plotFit(autos.clean$gp100m, fitted(fit.lm.factorless.step.both.aic))
summary.lm(fit.lm.factorless.step.both.aic)
```



Residual standard error: 0.4831 on 277 degrees of freedom
Multiple R-squared: 0.9184,   Adjusted R-squared: 0.9125
F-statistic: 155.9 on 20 and 277 DF, p-value: < 2.2e-16

lm with step = “both”, high VIFs, BIC, and no factors

```
# Fit a step = both, with BIC and excessive VIF.
fit.lm.factorless.step.both.bic = step(fit.lm.factorless,
direction=c("both"), k=log(length(autos.clean[,1])))
plotFit(autos.clean$gp100m, fitted(fit.lm.factorless.step.both.bic))
summary.lm(fit.lm.factorless.step.both.bic)
```



```
Residual standard error: 0.4888 on 281 degrees of freedom
Multiple R-squared: 0.9153,     Adjusted R-squared: 0.9104
F-statistic: 189.7 on 16 and 281 DF,  p-value: < 2.2e-16
```

Cross-Validation Function

Here is my cross-validation function:

```
# Cross-validation function
crossValidate <- function(ds, formula, nSets=5, yvalue) {
  nSets = ifelse(nSets == 0, 5, nSets) # no divide by zero...

  nRows = length(ds[,1])
  nRowsPerSet = floor(nRows / nSets)

  # Accumulate the total test error.
  total.test.err = 0

  # Iterate through the data set,
  # dividing into test and training data sets.)
  for (nSet in 1:nSets) {
      # Identify the test set indices.
      idxFirst = ((nSet - 1) * nRowsPerSet + 1)
      idxLast = nSet * nRowsPerSet

      # Don't read past the end of the data.
      idxLast = ifelse(idxLast > nRows, nRows, idxLast)
      idxSetTest = idxFirst:idxLast

      # The training set is everything but the test set.
      idxSetTrain = setdiff(1:nRows, idxSetTest)

      # Select data for training, testing.
      train = ds[idxSetTrain,]
      test = ds[idxSetTest, ]

      # Create the model for this training set.
      # Uncomment and iterate…
      model = lm(formula, data=train)
      #model.lm = lm(formula, data=train)
      #model = step(model.lm, direction=c("backward"))
      #model.lm = lm(formula, data=train)
      #model = step(model.lm, direction=c("forward"))
      #model.lm = lm(formula, data=train)
      #model = step(model.lm, direction=c("both"))
      #model = glm(formula, family=gaussian, data=train)
      #model.glm = glm(formula, family=gaussian, data=train)
      #model = step(model.glm, direction=c("backward"))
      #model.glm = glm(formula, family=gaussian, data=train)
      #model = step(model.glm, direction=c("forward"))
      #model.glm = glm(formula, family=gaussian, data=train)
      #model = step(model.glm, direction=c("both"))
      #model = lm(gp100m ~ cyl + weight + accel + year + continent +
diesel, data=train)
      #model = lm(gp100m ~ weight + year + continent + diesel,
data=train)

      # Calculate rms error for this test set.
      test.yhat <- predict.lm(object=model, newdata=test)
     test.y <- with(test, get(yvalue))
     test.err <- sqrt(mean((test.yhat - test.y)^2))

     # Accumulate test error * size of test set.
      total.test.err = total.test.err + test.err

  }

  mean.test.err = total.test.err / nSets
  # Return the mean rms error of the test sets.
  mean.test.err
}

#---Test invocation---
mean.test.err = crossValidate(ds, formula, 5, "gp100m")
```

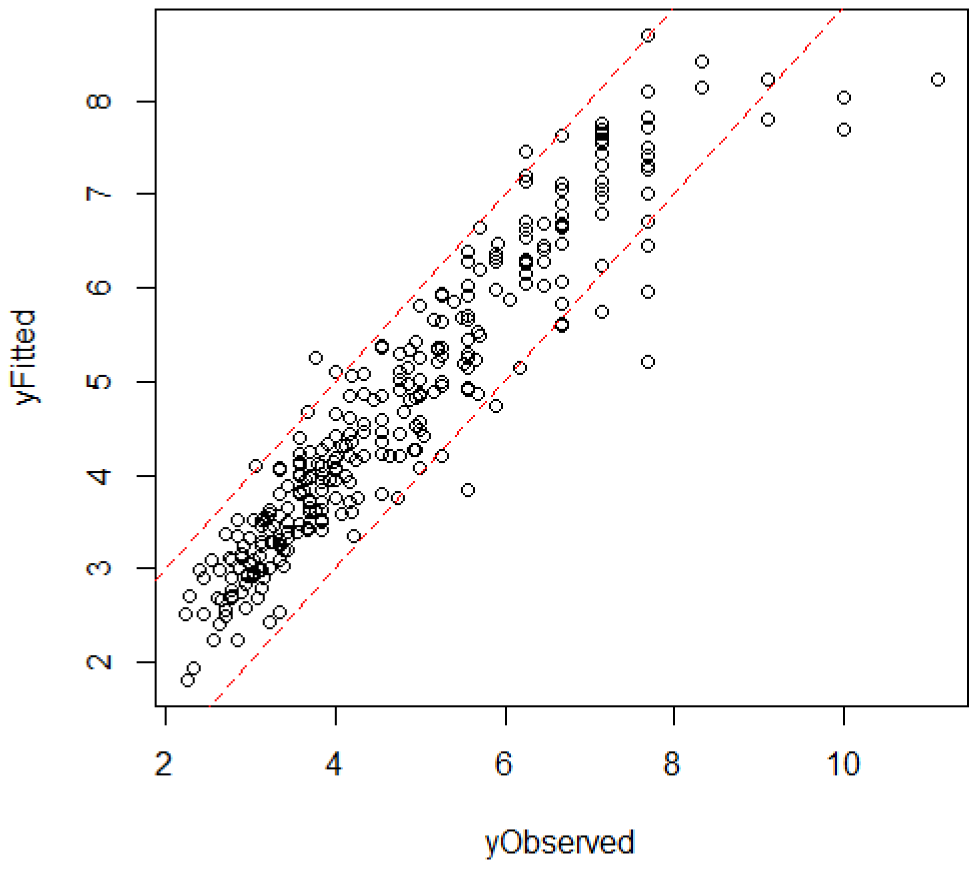
Model Cross-Validation Summary

Model	**Formula**	**K**	**RMS Error**
model = lm(formula, data=train)	gp100m ~ .^2	5	0.6871105
model.lm = lm(formula, data=train) model = step(model.lm, direction=c("backward"))	gp100m ~ .^2	5	0.6879766
model.lm = lm(formula, data=train) model = step(model.lm, direction=c("forward"))	gp100m ~ .^2	5	0.6871105
model.lm = lm(formula, data=train) model = step(model.lm, direction=c("both"))	gp100m ~ .^2	5	(Broke R)
model = lm(formula, data=train)	gp100m ~ .^2	5	0.6871105
model.glm = glm(formula, family=gaussian, data=train) model = step(model.glm, direction=c("backward"))	gp100m ~ .^2	5	0.6879766
model.glm = glm(formula, family=gaussian, data=train) model = step(model.glm,	gp100m ~ .^2	5	0.6871105

direction=c("forward"))			
model.glm = glm(formula, family=gaussian, data=train) model = step(model.glm, direction=c("both"))	gp100m ~ .^2	5	0.6876549
model = lm(formula, data=train)	gp100m ~ cyl + weight + accel + year + continent + diesel	5	0.5896428
model = lm(formula, data=train)	model = lm(gp100m ~ cyl + weight + year + continent + diesel, data=train)	5	0.5810698
model = lm(formula, data=train)	model = lm(gp100m ~ weight + year + continent + diesel, data=train)	5	0.5780315

Best Performing Parsimonious Model

```
Call:

lm(formula = gp100m ~ weight + year + continent + diesel, data = ds)
```



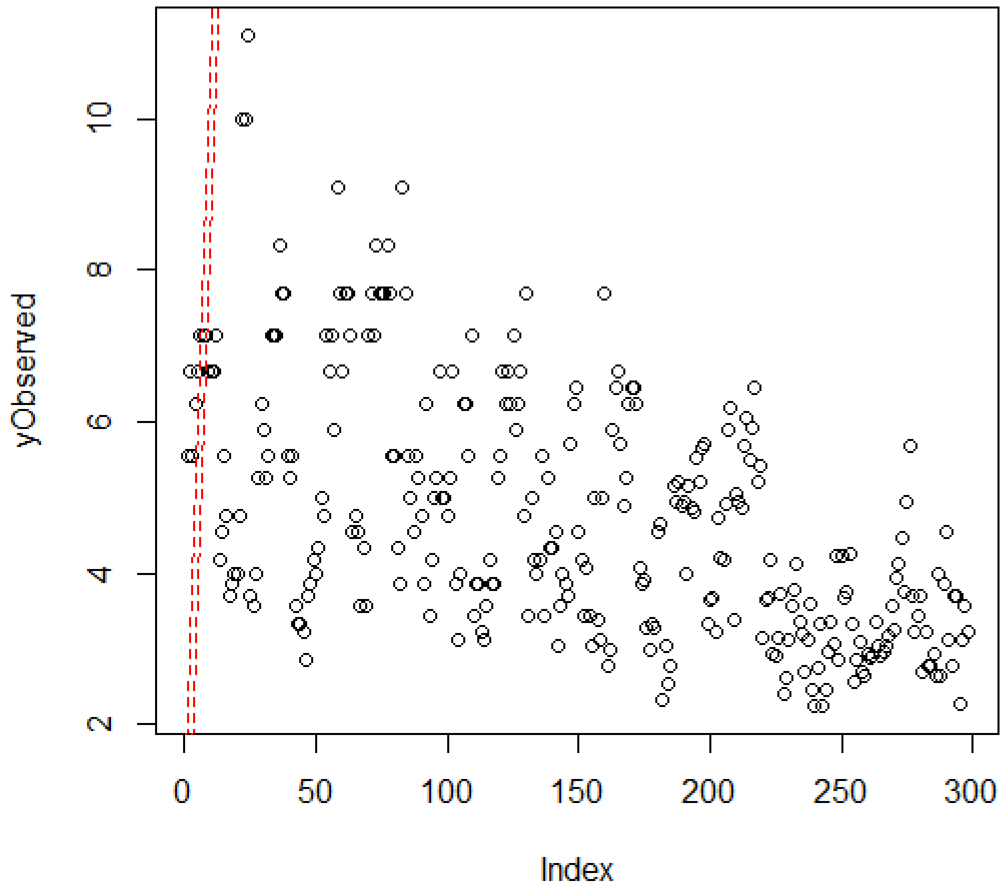
```
Residual standard error: 0.5641 on 293 degrees of freedom
Multiple R-squared: 0.8823,     Adjusted R-squared: 0.8807
F-statistic: 549.2 on 4 and 293 DF,  p-value: < 2.2e-16
```

Lasso

```
# Load data frame.
setwd("C:/Users/Rod/SkyDrive/R/201/Week07")  # SkyDrive from home
autos = read.csv("autoMPGtrain.csv", header=TRUE)
autos$gp100m = 100 / autos$mpg

transformAutos.sans.factors <- function(autos) {
  autos.clean = data.frame(cbind(100/autos$mpg, autos$cyl,
autos$volume, autos$hp, autos$weight, autos$accel ,autos$year,
autos$continent, autos$diesel))

  colnames(autos.clean) = c("gp100m", "cyl", "volume", "hp", "weight",
"accel", "year", "continent", "diesel")

  # Establish NO factors in data.
  #autos.clean$continent = factor(autos.clean$continent)  # North
America, Europe, Asia
  #autos.clean$diesel = factor(autos.clean$diesel) # gasoline, diesel
  #autos.clean$cyl = factor(autos.clean$cyl) # Number of cylinders is
non-continuous
  #autos.clean$year = factor(autos.clean$year) # Model years are non-
continuous.

# And return the data frame.
  autos.clean
}

ds = transformAutos.sans.factors(autos)
str(ds)

> str(ds.x)
'data.frame':   298 obs. of  8 variables:
 $ cyl      : num  8 8 8 8 8 8 8 8 8 8 ...
 $ volume   : num  307 350 318 304 429 454 440 455 390 383 ...
 $ hp       : num  130 165 150 150 198 220 215 225 190 170 ...
 $ weight   : num  3504 3693 3436 3433 4341 ...
 $ accel    : num  12 11.5 11 12 10 9 8.5 10 8.5 10 ...
 $ year     : num  70 70 70 70 70 70 70 70 70 70 ...
 $ continent: num  1 1 1 1 1 1 1 1 1 1 ...
 $ diesel   : num  0 0 0 0 0 0 0 0 0 0 ...
```

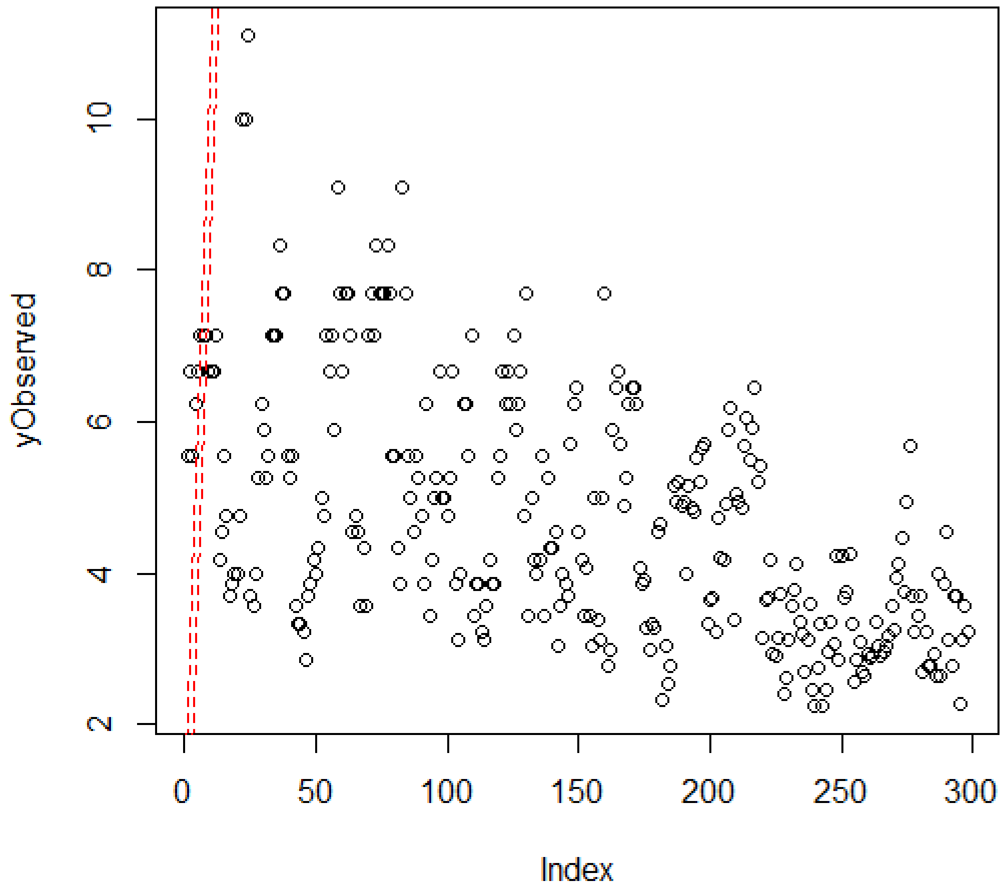
```
# A handy little function to visually render the fit.
 plotFit <- function(yObserved, yFitted) {
  plot(yObserved, yFitted)
  abline(1,1,lty=2, col=rgb(1,0,0))
  abline(-1,1,lty=2, col=rgb(1,0,0))
}


# Gets covariates only (no response)
ds.x = ds[, !(colnames(ds) %in% c("gp100m"))]
ds.y = ds$gp100m
> str(ds.x)
'data.frame':   298 obs. of  8 variables:
 $ cyl      : num  8 8 8 8 8 8 8 8 8 8 ...
 $ volume   : num  307 350 318 304 429 454 440 455 390 383 ...
 $ hp       : num  130 165 150 150 198 220 215 225 190 170 ...
 $ weight   : num  3504 3693 3436 3433 4341 ...
 $ accel    : num  12 11.5 11 12 10 9 8.5 10 8.5 10 ...
 $ year     : num  70 70 70 70 70 70 70 70 70 70 ...
 $ continent: num  1 1 1 1 1 1 1 1 1 1 ...
 $ diesel   : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
# Lasso from lars
library(lars)
lars.lasso = lars(as.matrix(ds.x), ds.y, type="lasso")
plotFit(ds.y, fitted(lars.lasso))
```



This doesn’t look quite right. Let’s try glmnet.

```
# Lasso from glmnet
> library(glmnet)
> glmnet.lasso = glmnet(x = as.matrix(ds.x), y = ds.y)
> plotFit(ds.y, fitted(glmnet.lasso))
```



The glmnet version looks equally incorrect. It's time to round the corner and submit.