Coursera_Practical_Machine_Learning_JHU

Anyi Guo

24/12/2018

This week's videos take a lot of time to go through & writing notes on.

Week 2

Preprossing data with caret

```
library(caret)
```

```
## Loading required package: lattice
```

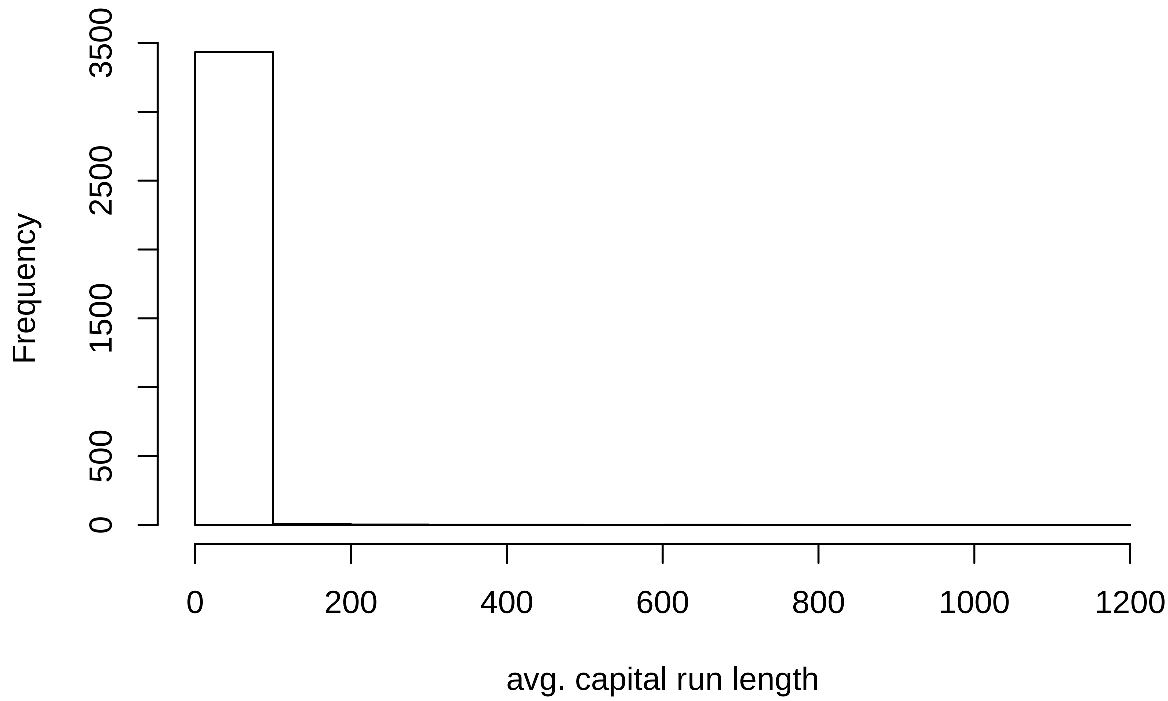
```
## Loading required package: ggplot2
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
data(spam)
inTrain<-createDataPartition(y=spam$type,p=0.75,list=FALSE)
training<-spam[inTrain,]
testing<-spam[-inTrain,]
hist(training$capitalAve,main="",xlab="avg. capital run length")
```



The histogram shows that the data are heavily skewed to the left.

Standardizing the variables (so that they have `mean = 0` and `sd=1`)

```
trainCapAve<-training$capitalAve
trainCapAveS<-(trainCapAve-mean(trainCapAve))/sd(trainCapAve)
mean(trainCapAveS)
```

```
## [1] 5.918213e-18
```

```
sd(trainCapAveS)
```

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

Standardizing the test set, using mean and sd of the training set. This means that the standardized test cap will not be exactly the same as that of the training set, but they should be similar.

```
testCapAve<-testing$capitalAve
testCapAveS<-(testCapAve-mean(trainCapAve))/sd(trainCapAve)
mean(testCapAveS)
```

```
## [1] -0.03865693
```

Use preprocess() function to do the standardization on the training set. The result is the same as using the above functions

```
preObj<-preProcess(training[,-58],method=c("center","scale"))
trainCapAveS<-predict(preObj,training[,-58])$capitalAve
mean(trainCapAveS)
```

```
## [1] 5.918213e-18
```

```
sd(trainCapAveS)
```

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

Use `preProcess()` to do the same on the testing dataset. Note that `preObj` (which was created based on the training set) is also used to predict on the testing set.

Note that `mean()` is not equal to 0 on the testing set, and `sd` is not equal to 1.

```
testCapAveS<-predict(preObj,testing[,-58])$capitalAve
mean(testCapAveS)
```

```
## [1] -0.03865693
```

```
sd(testCapAveS)
```

```
## [1] 0.3020468
```

Use `preProcess()` directly when building a model

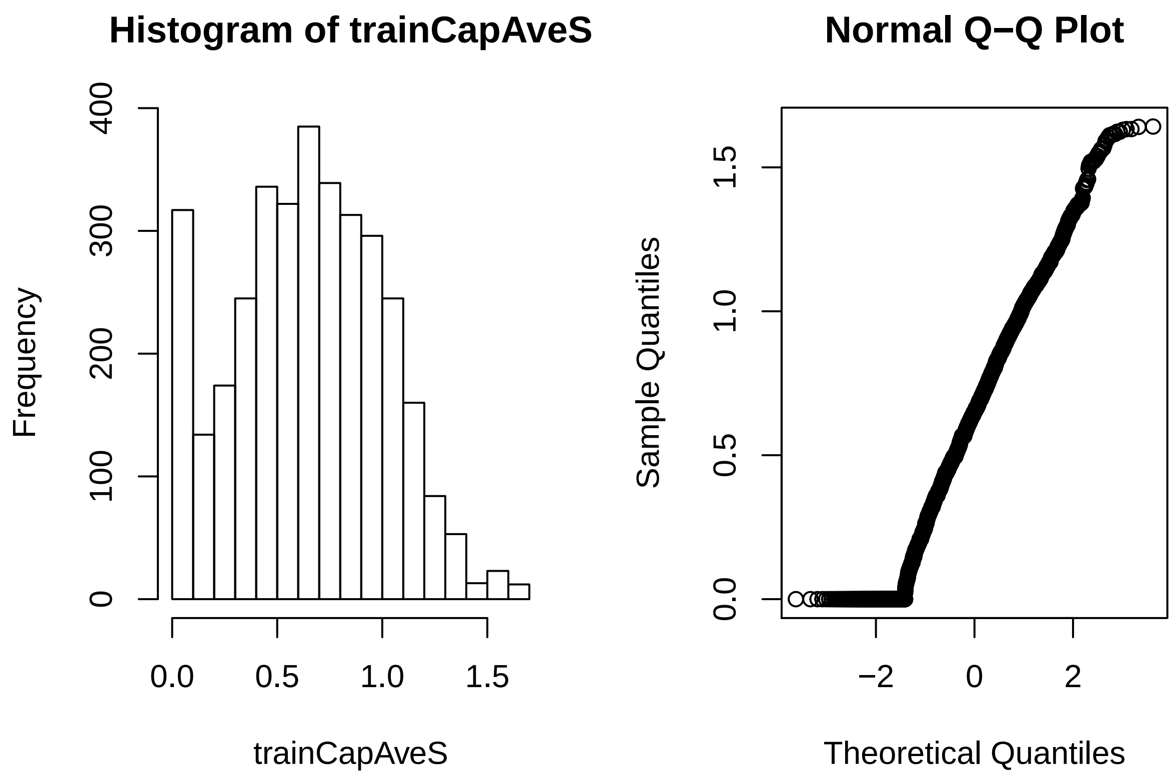
```
set.seed(1)
model<-train(type ~.,data=training,preProcess=c("center","scale"),method="glm")
model
```

```
## Generalized Linear Model
##
## 3451 samples
##   57 predictor
##    2 classes: 'nonspam', 'spam'
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9196191  0.8318209
```

Standardising - Box-Cox Transforms

This transforms the data into normal shape - i.e. bell shape

```r
preObj<-preProcess(training[,-58],method=c("BoxCox"))
trainCapAveS<-predict(preObj,training[,-58])$capitalAve
par(mfrow=c(1,2))
hist(trainCapAveS)
qqnorm(trainCapAveS)
```



Standardization: Imputing data where it is NA using `knnImpute`

`knnImpute` uses the average of the k-nearest neighbours to impute the data where it's not available.

```r
set.seed(1)

# Make some value NAs
training$capAve<-training$capitalAve
selectNA<-rbinom(dim(training)[1],size=1,prob=0.05)==1
training$capAve[selectNA]<-NA

# Impute data when it's NA, and standardize
preObj<-preProcess(training[,-58],method="knnImpute")
capAve<-predict(preObj,training[,-58])$capAve

# Standardize true values
capAveTruth<-training$capitalAve
capAveTruth<-(capAveTruth-mean(capAveTruth))/sd(capAveTruth)
```

Look at the difference at the imputed value (`capAve`) and the true value (`capAveTruth`), using `quantile()` function.

If the values are all relatively small, then it shows that imputing data works (i.e. doesn't change the dataset too much).

```r
quantile(capAve-capAveTruth)
```

```
##            0%           25%           50%           75%          100%
## -0.8110186882 -0.0017893307 -0.0006845495 -0.0001306859  0.1052364077
```

Some notes on preprocessing data

- training and testing must be processed in the same way (i.e. use the same `preObj` in `predict()` function)

Covariate/Predictor/Feature Creation

1. Step 1: raw data -> features (e.g. free text -> data frame) Google "Feature extraction for [data type]" Examples:

- Text files: frequency of words, frequency of phrases, frequency of capital letters
- Images: Edges, corners, ridges
- Webpages: # and type of images, position of elements, colors, videos (e.g. A/B testing)
- People: Height, weight, hair color, gender etc.

2. Step 2: features -> new, useful features

- more useful for some models (e.g. regression, SVM) than others(e.g. decision trees)
- should be done **only on the training set**
- new features should be added to data frames

3. An example of feature creation

```
```r
library(ISLR)
library(caret)
data(Wage)
inTrain<-createDataPartition(y=Wage$wage,p=0.7,list=FALSE)
training<-Wage[inTrain,]
testing<-Wage[-inTrain,]
```
```

- Convert factor variables to dummy variables

  The `jobclass` column is chacracters, so we can convert it to dummy variable with `dummyVars` function

```r
dummies<-dummyVars(wage ~ jobclass,data=training)
head(predict(dummies,newdata=training))
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                      1                       0
## 86582                       0                       1
## 161300                      1                       0
## 155159                      0                       1
## 11443                       0                       1
## 376662                      0                       1
```

- Remove features which is the same throughout the dataframe, using `nearZeroVar`

  If nsv (`nearZeroVar`) returns TRUE, then this feature is not important and thus can be removed.

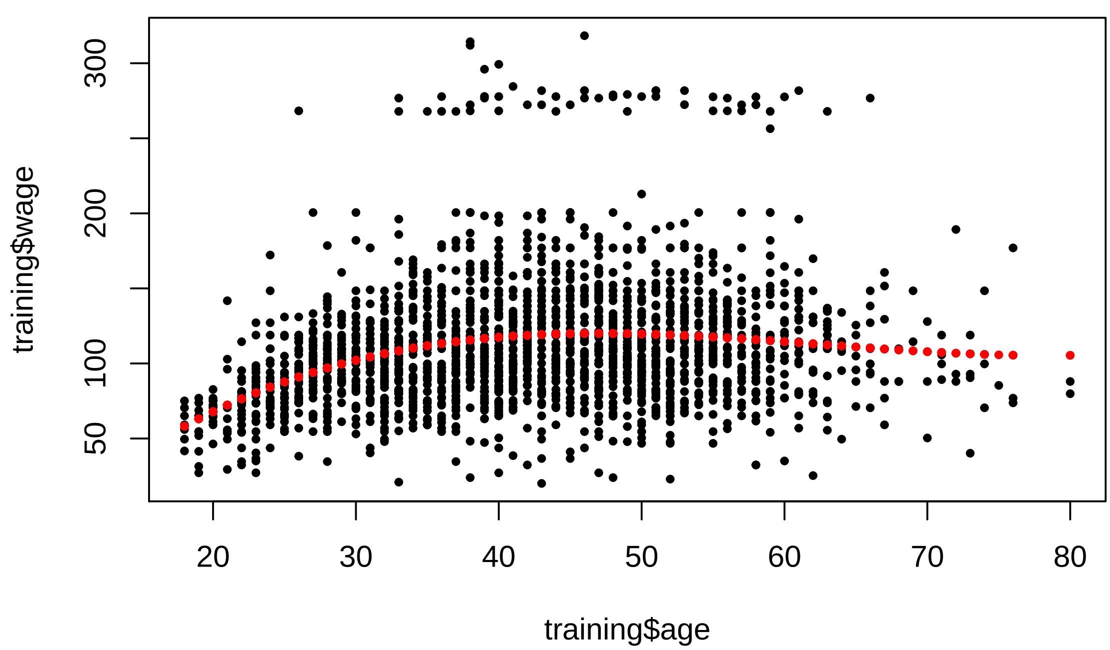
```
nsv<-nearZeroVar(training,saveMetrics = TRUE)
```

  - Spline basis `df=3` says that we want a 3rd-degree polynomial on this variable `training$age`. First column means `age` Second column means `age^2` Third column means `age^3`

```
library(splines)
bsBasis<-bs(training$age,df=3)
#bsBasis
```

Fitting curves with splines

```
lm1<-lm(wage~bsBasis,data=training)
plot(training$age,training$wage,pch=19,cex=0.5)
points(training$age,predict(lm1,newdata=training),col="red",pch=19,cex=0.5)
```



splines on the test set. Note that we are using the same `bsBasis` as is created in the training dataset

```r
p<-predict(bsBasis,age=testing$age)
```

PCA (Principal Components Analysis), mostly useful for linear-type models

1. Find features which are correlated

`which()` returns the list of features with correlation > 0.8

```r
library(caret)
library(kernlab)
data(spam)
set.seed(1)
inTrain<-createDataPartition(y=spam$type,p=0.75,list=FALSE)
training<-spam[inTrain,]
testing<-spam[-inTrain,]

M<-abs(cor(training[,-58]))
diag(M)<-0
which(M>0.8,arr.ind=T)
```

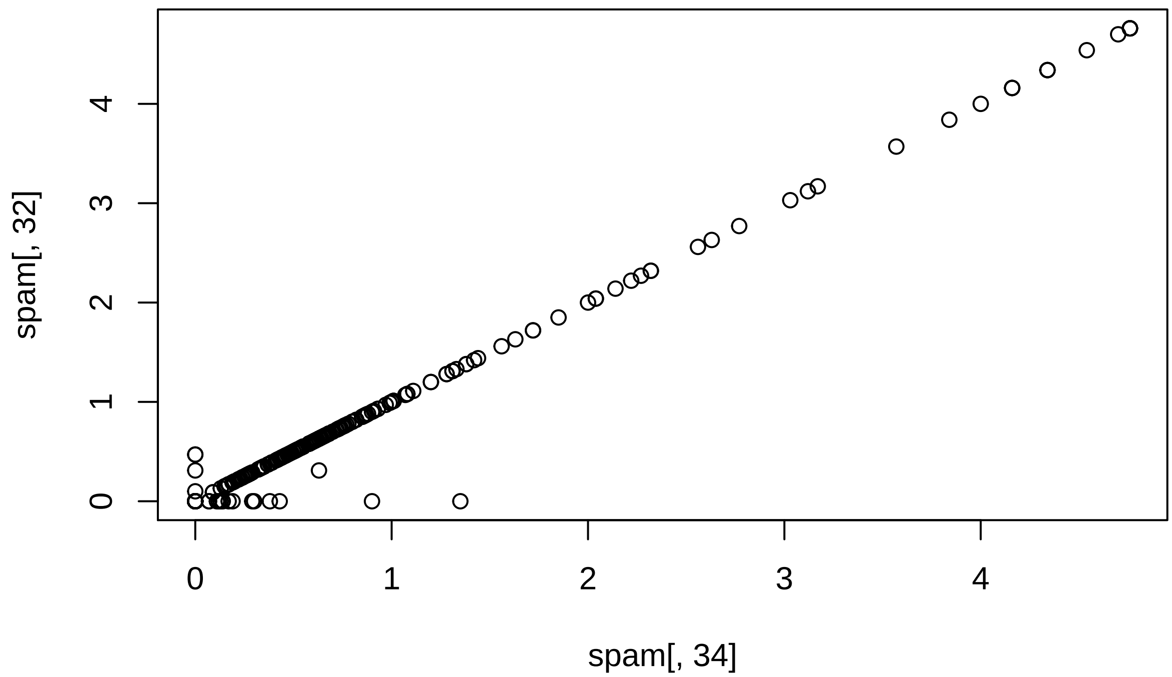
```
##        row col
## num415  34  32
## direct  40  32
## num857  32  34
## direct  40  34
## num857  32  40
## num415  34  40
```

Take a look at the correlated features:

```r
names(spam)[c(34,32,40)]
```

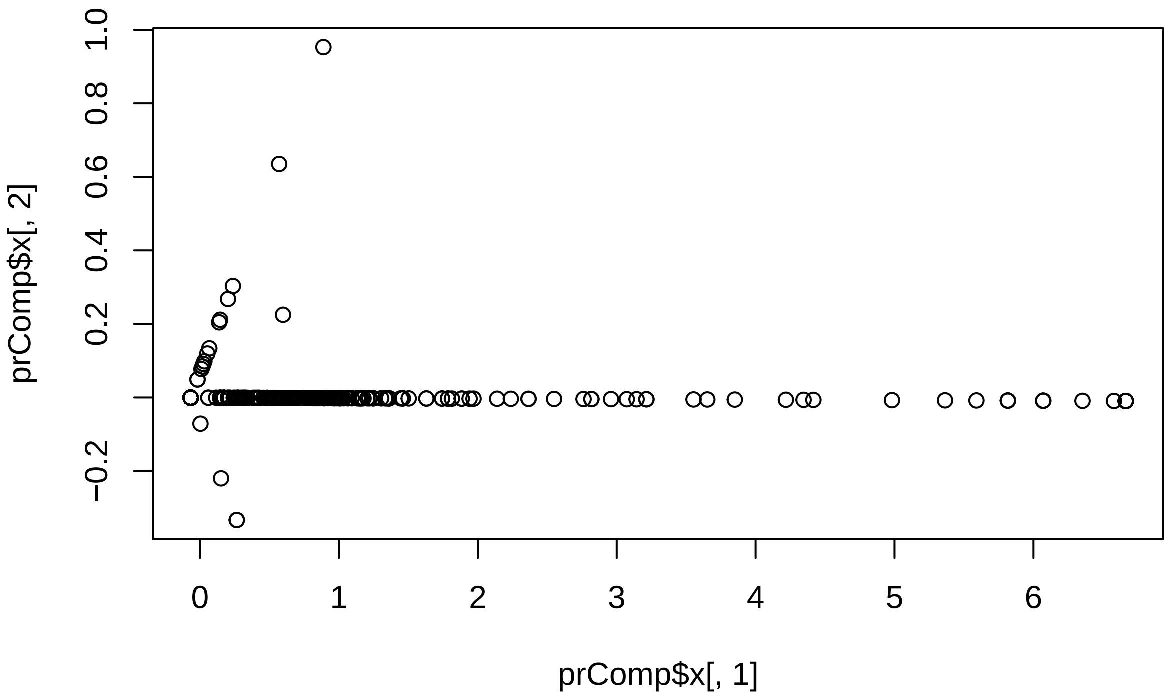
```
## [1] "num415" "num857" "direct"
```

```r
plot(spam[,34],spam[,32])
```



Apply PCA in R: `prcomp()`

```
smallSpam<-spam[,c(34,32)]
prComp<-prcomp(smallSpam)
plot(prComp$x[,1],prComp$x[,2])
```

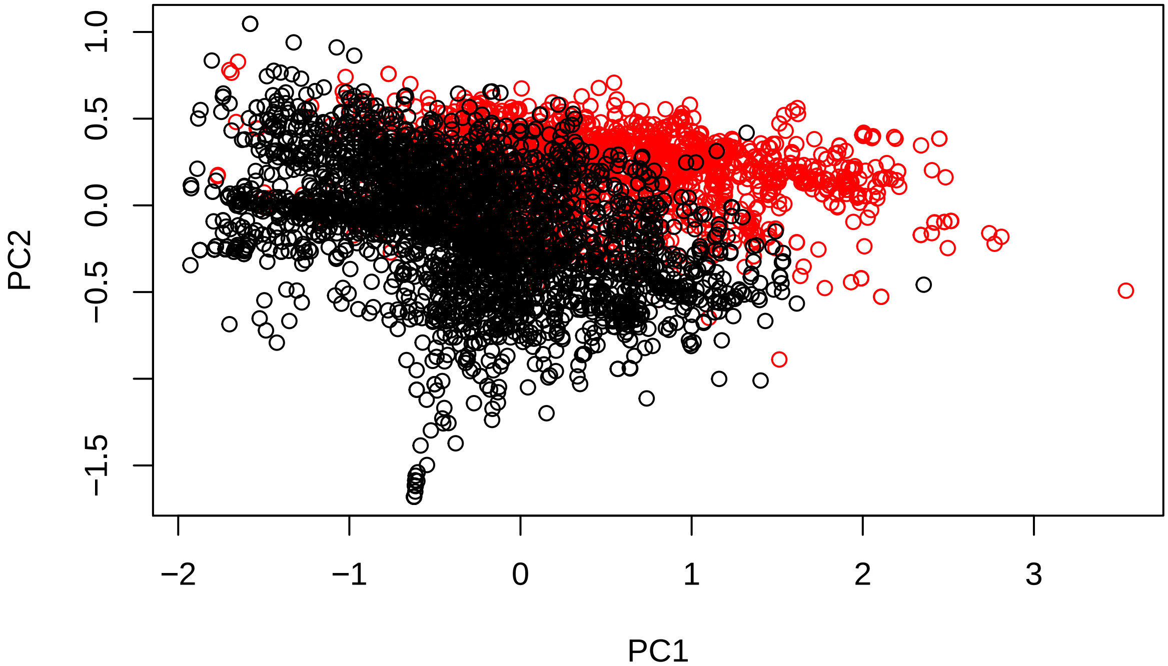



```r
prComp$rotation
```

```
##                PC1        PC2
## num415 0.7080625  0.7061498
## num857 0.7061498 -0.7080625
```

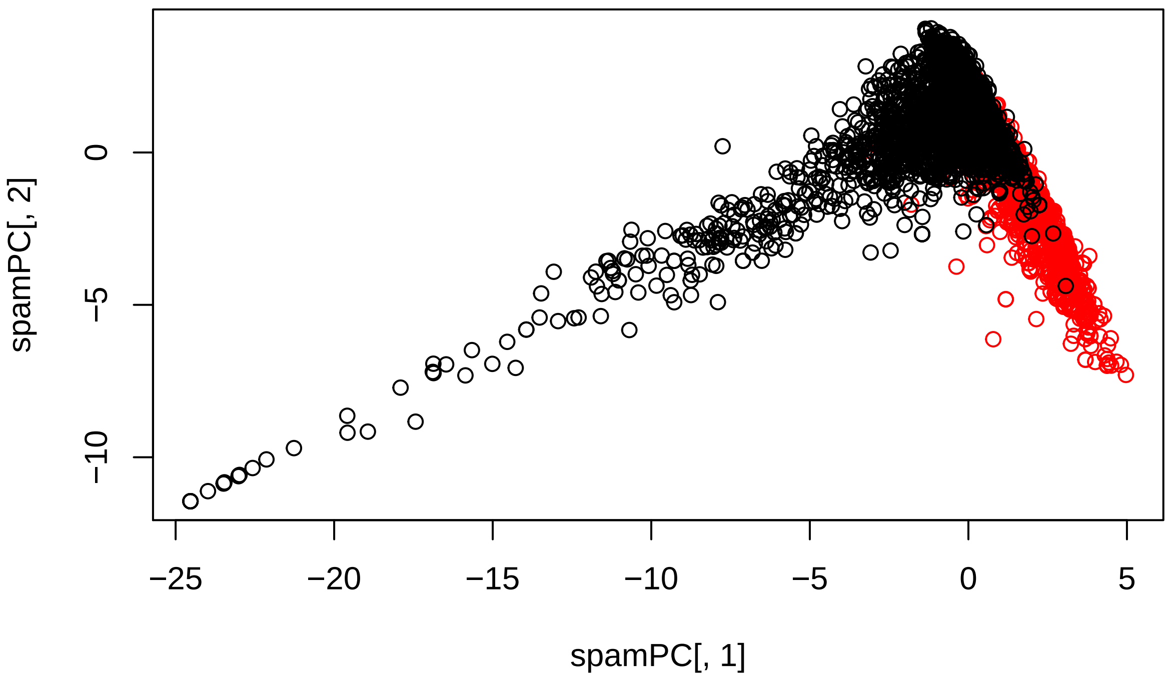
PCA on spam data

```r
typeColor<-((spam$type=="spam")*1+1)
prComp<-prcomp(log10(spam[,-58]+1))
plot(prComp$x[,1],prComp$x[,2],col=typeColor,xlab="PC1",ylab="PC2")
```



PCA with caret, preProcess()

```
preProc<-preProcess(log10(spam[,-58]+1),method="pca",pcaComp = 2)
spamPC<-predict(preProc,log10(spam[,-58]+1))
plot(spamPC[,1],spamPC[,2],col=typeColor)
```



Preprocessing with PCA to create model based on the training set

```
preProc<-preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC<-predict(preProc,log10(training[,-58]+1))
modelFit <- train(x = trainPC, y = training$type,method="glm")
```

Preprocessing with PCA to use on the testing set Note that we should use the same PCA procedure (`preProc`) when using predict() on the testing set

```
testPC<-predict(preProc,log10(testing[,-58]+1))
confusionMatrix(testing$type,predict(modelFit,testPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     654   43
##    spam         69  384
##
##                Accuracy : 0.9026
##                  95% CI : (0.884, 0.9191)
##     No Information Rate : 0.6287
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.794
```

```
## Mcnemar's Test P-Value : 0.01816
##
##             Sensitivity : 0.9046
##             Specificity : 0.8993
##          Pos Pred Value : 0.9383
##          Neg Pred Value : 0.8477
##              Prevalence : 0.6287
##          Detection Rate : 0.5687
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.9019
##
##        'Positive' Class : nonspam
##
```

Accuracy is > 0.9!

Alternative: preProcess with PCA during the training process (instead of doing PCA first, then do the training)

```
modelFit <- train(x = trainPC, y = training$type,method="glm",preProcess="pca")
confusionMatrix(testing$type,predict(modelFit,testPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     697    0
##    spam        453    0
##
##                Accuracy : 0.6061
##                  95% CI : (0.5772, 0.6345)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6061
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.6061
##    Detection Prevalence : 0.6061
##       Balanced Accuracy :     NA
##
##        'Positive' Class : nonspam
##
```

Predicting with Regression

Use the fainthful eruption data in caret

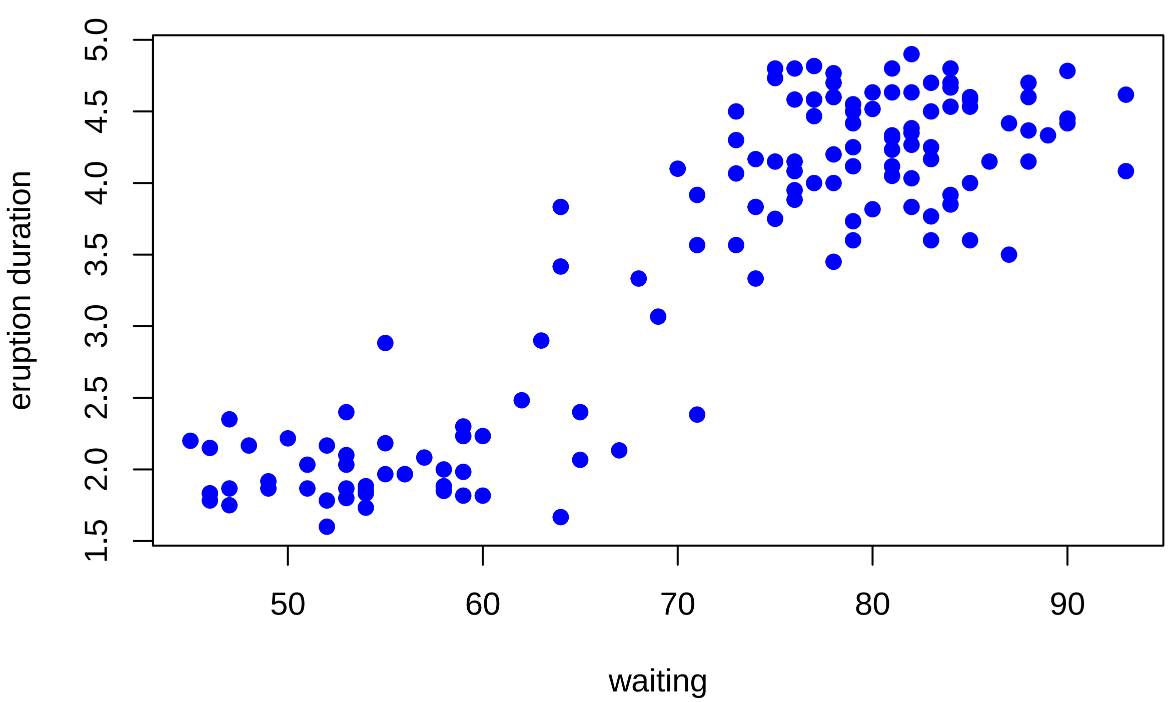
```r
library(caret)
data(faithful)
set.seed(333)
inTrain<-createDataPartition(y=faithful$waiting,p=0.5,list=FALSE)
trainFaith<-faithful[inTrain,]
testFaith<-faithful[-inTrain,]
head(trainFaith)
```

```
##   eruptions waiting
## 1     3.600      79
## 3     3.333      74
## 5     4.533      85
## 6     2.883      55
## 7     4.700      88
## 8     3.600      85
```

Plot eruption duration vs. waiting time.

You can see that there's a roughly linear relationship between the two variables.

```r
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="waiting",ylab="eruption duration")
```



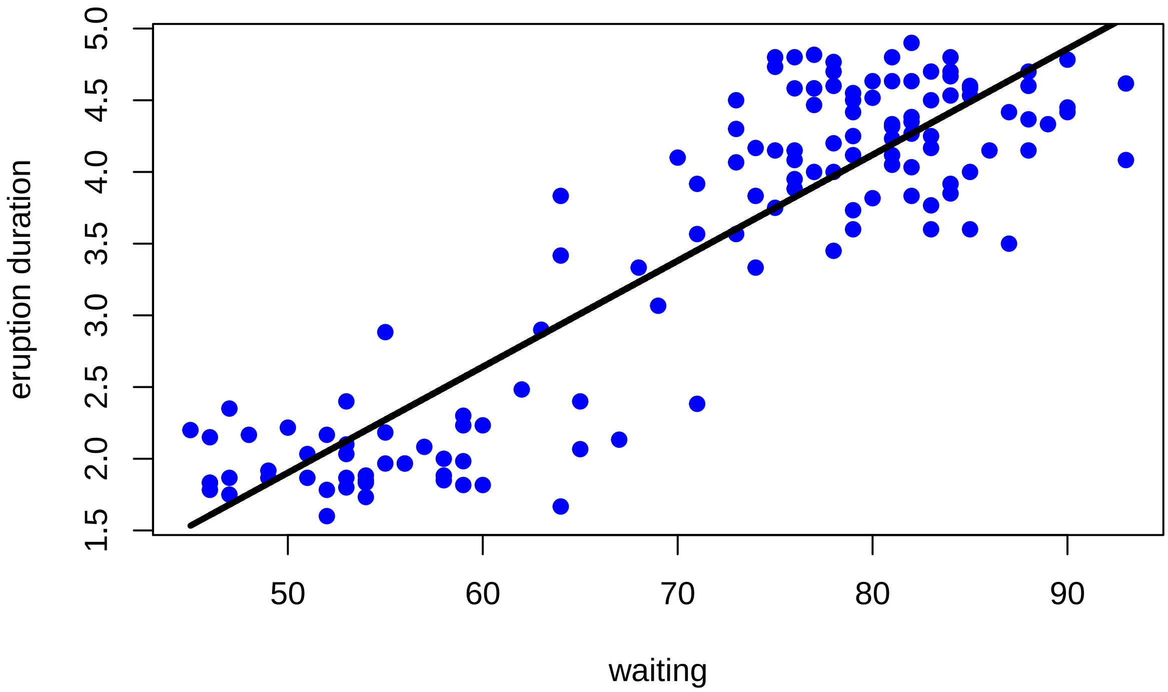
Fit a linear regression model

```r
lm1<-lm(eruptions~waiting,data=trainFaith)
summary(lm1)
```

```
##
## Call:
## lm(formula = eruptions ~ waiting, data = trainFaith)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.26990 -0.34789  0.03979  0.36589  1.05020
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.792739   0.227869  -7.867 1.04e-12 ***
## waiting      0.073901   0.003148  23.474  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.495 on 135 degrees of freedom
## Multiple R-squared:  0.8032, Adjusted R-squared:  0.8018
## F-statistic:   551 on 1 and 135 DF,  p-value: < 2.2e-16
```

Plot the model fit

```r
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="waiting",ylab="eruption duration")
lines(trainFaith$waiting,lm1$fitted,lwd=3)
```



Predicting a new value with the linear regression model

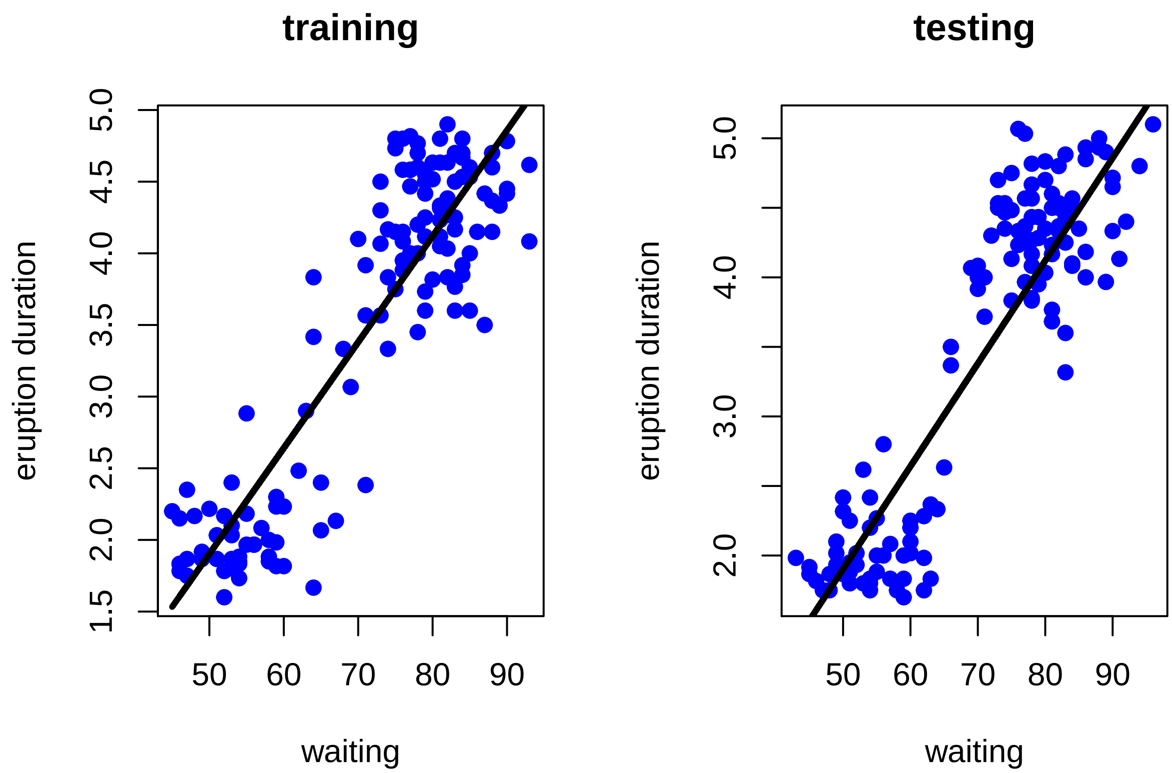
When `waiting time = 80`

```
newdata<-data.frame(waiting=80)
predict(lm1,newdata)
```

```
##        1
## 4.119307
```

Plot predictions - training vs testing set

```
par(mfrow=c(1,2))
# training
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",main="training",xlab="waiting",ylab="eru
lines(trainFaith$waiting,predict(lm1),lwd=3)
# testing
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue",main="testing",xlab="waiting",ylab="erupti
lines(testFaith$waiting,predict(lm1,newdata=testFaith),lwd=3)
```



Get training & testing errors

```
# RMSE on training
sqrt(sum((lm1$fitted-trainFaith$eruptions)^2))
```

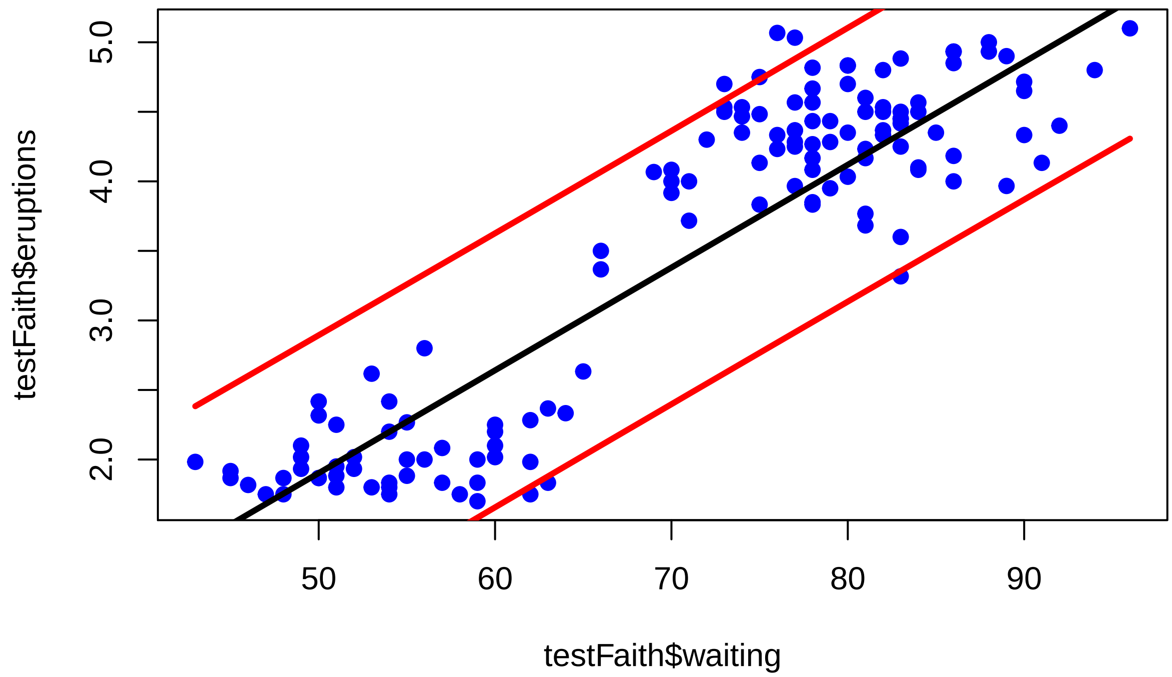
```
## [1] 5.75186
```

```
# RMSE on testing
sqrt(sum((predict(lm1,newdata=testFaith)-testFaith$eruptions)^2))
```

```
## [1] 5.838559
```

Prediction intervals

```
pred1<-predict(lm1,newdata=testFaith,interval="prediction")
ord<-order(testFaith$waiting)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue")
matlines(testFaith$waiting[ord],pred1[ord,],type="l",col=c(1,2,2),lty=c(1,1,1),lwd=3)
```



Same process with caret

```
modFit<-train(eruptions~waiting,data=trainFaith,method="lm")
summary(modFit$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.26990 -0.34789  0.03979  0.36589  1.05020
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.792739   0.227869  -7.867 1.04e-12 ***
## waiting      0.073901   0.003148  23.474  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.495 on 135 degrees of freedom
## Multiple R-squared:  0.8032, Adjusted R-squared:  0.8018
## F-statistic:   551 on 1 and 135 DF,  p-value: < 2.2e-16
```

Predicting with regression, multiple covariates

Use the wages dataset in ISLR package

```
library(ISLR)
library(ggplot2)
library(caret)
data(Wage)
Wage<-subset(Wage,select=-c(logwage))
summary(Wage)
```

```
##       year           age                       maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##               education                      region
##  1. < HS Grad      :268   2. Middle Atlantic   :3000
##  2. HS Grad        :971   1. New England       :   0
##  3. Some College   :650   3. East North Central:   0
##  4. College Grad   :685   4. West North Central:   0
##  5. Advanced Degree:426   5. South Atlantic    :   0
##                           6. East South Central:   0
##                           (Other)              :   0
##            jobclass               health      health_ins
##  1. Industrial :1544   1. <=Good     : 858   1. Yes:2083
##  2. Information:1456   2. >=Very Good:2142   2. No : 917
##
##
##
##
##
##       wage
##  Min.   : 20.09
##  1st Qu.: 85.38
##  Median :104.92
##  Mean   :111.70
##  3rd Qu.:128.68
##  Max.   :318.34
##
```

```
inTrain<-createDataPartition(y=Wage$wage,p=0.7,list=FALSE)
training<-Wage[inTrain,]
testing<-Wage[-inTrain,]
dim(training)
```

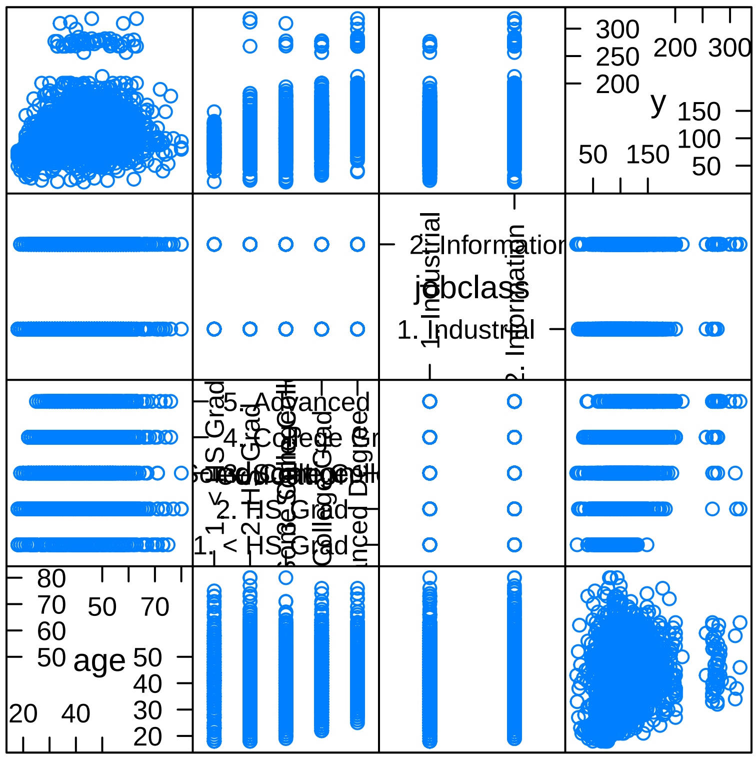
```
## [1] 2102   10
```

```r
dim(testing)
```

```
## [1] 898  10
```

Feature plot on the wages dataset

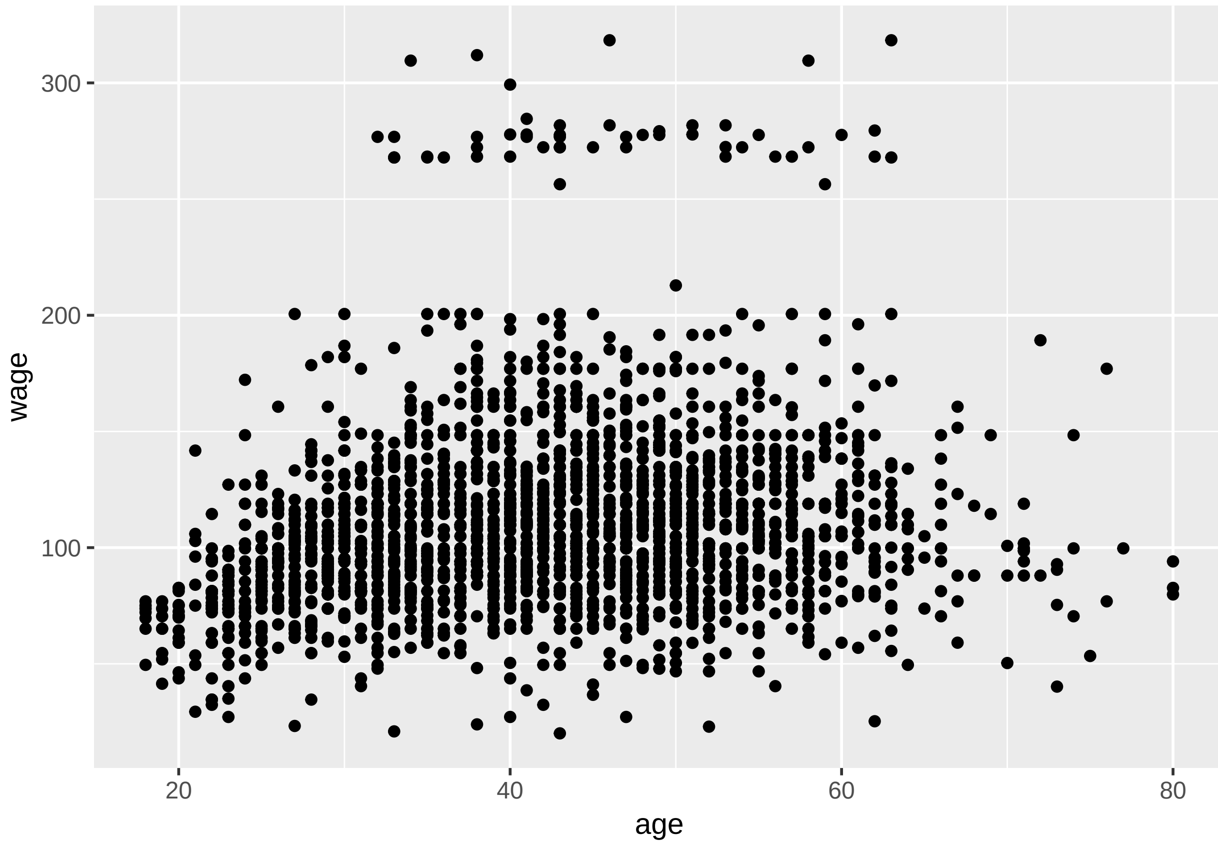
```r
featurePlot(x=training[,c("age","education","jobclass")],y=training$wage,plot="pairs")
```



Scatter Plot Matrix

Plot age vs. wage

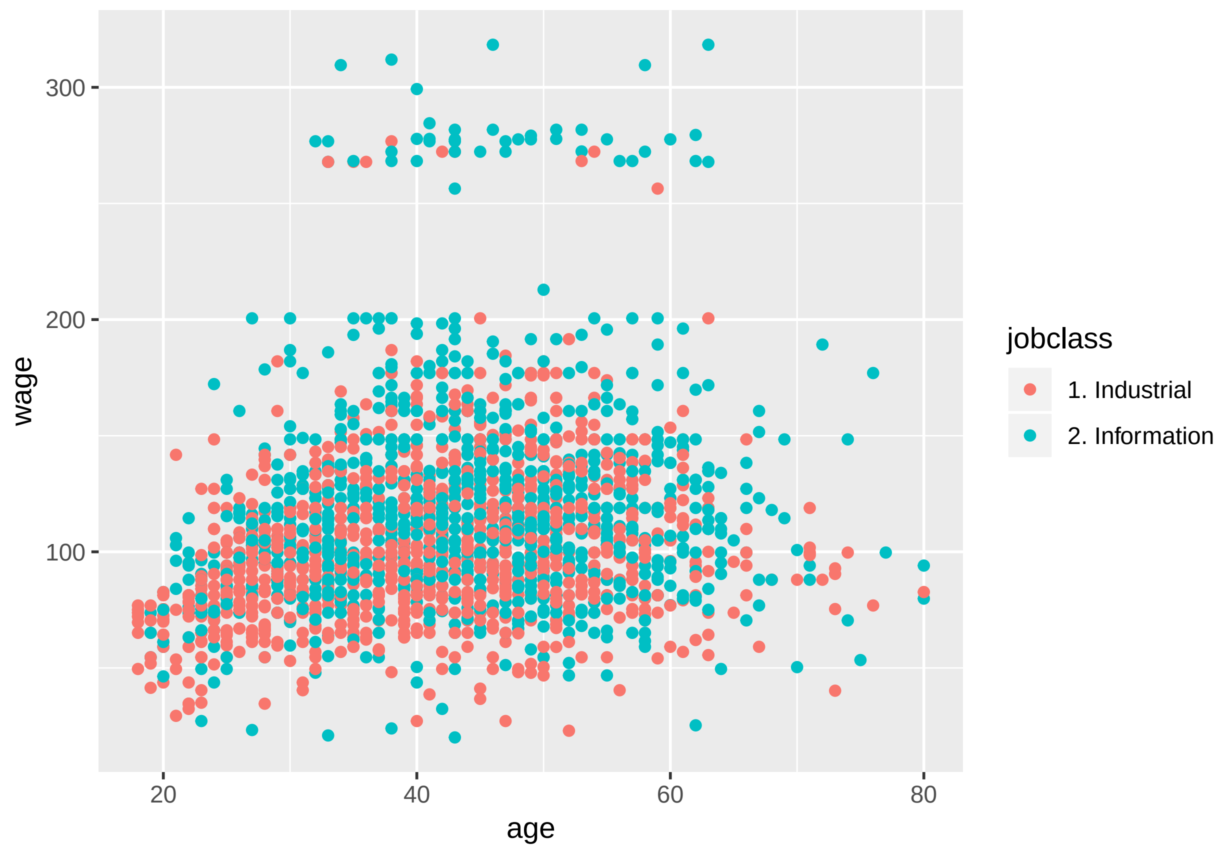
```r
qplot(age,wage,data=training)
```



Plot age vs wage, color by jobclass

We can see that the outliners are mostly for people in informational jobclass

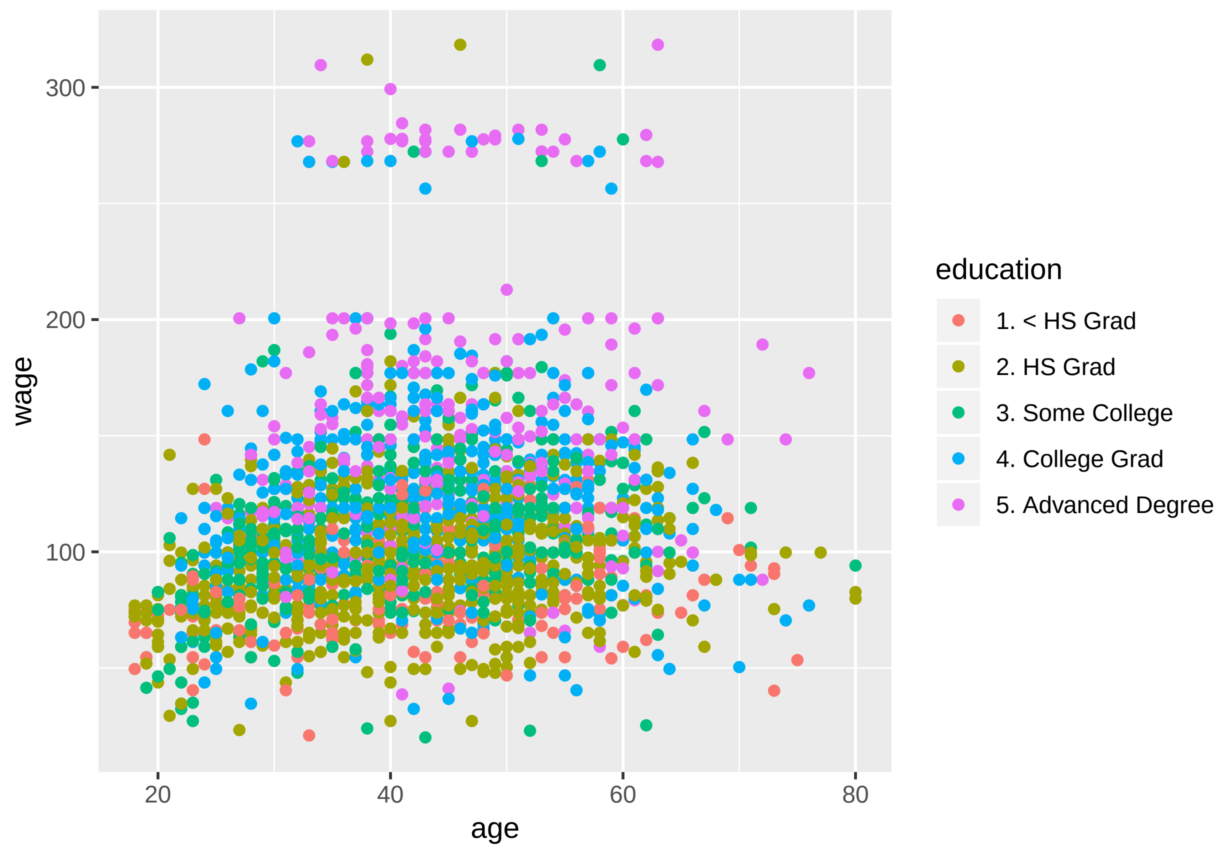
```
qplot(age,wage,color=jobclass,data=training)
```



Plot age vs. wage, color by education

You can see that the outliners are mostly advance degree education

```
qplot(age,wage,color=education,data=training)
```

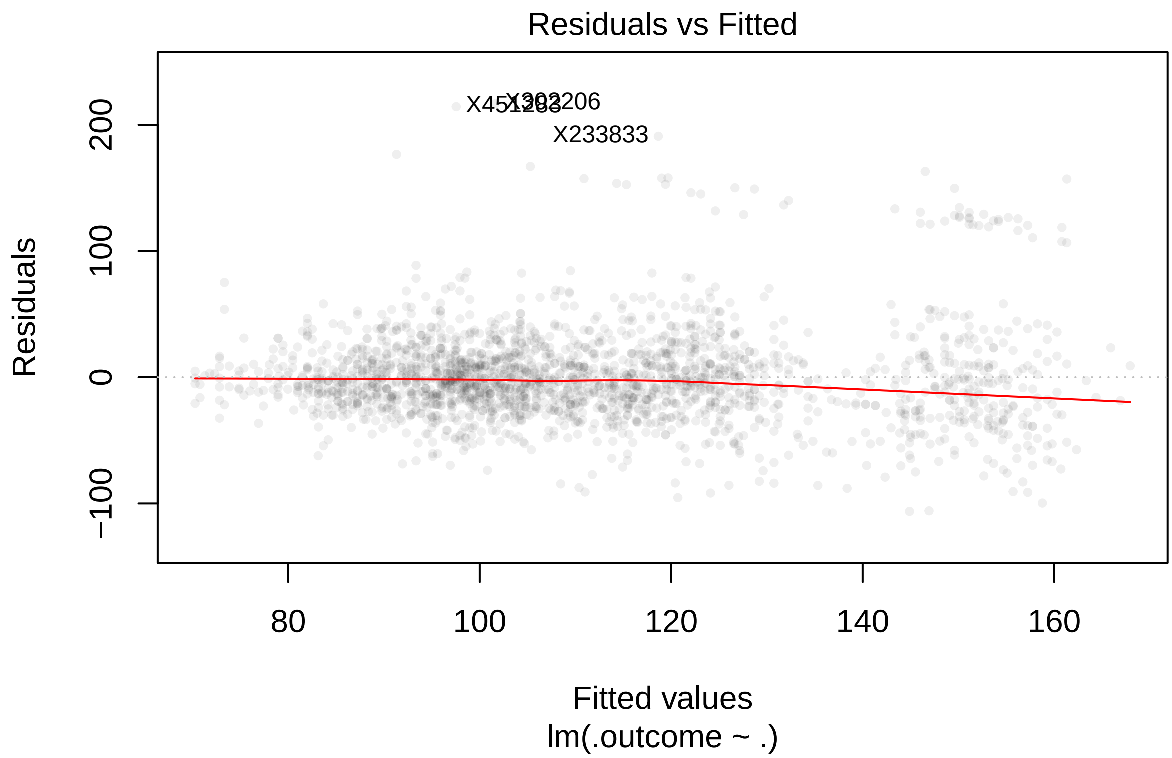


Fit a linear model

```
modFit<-train(wage~age+jobclass+education,method="lm",data=training)
finMod<-modFit$finalModel
print(modFit)
```

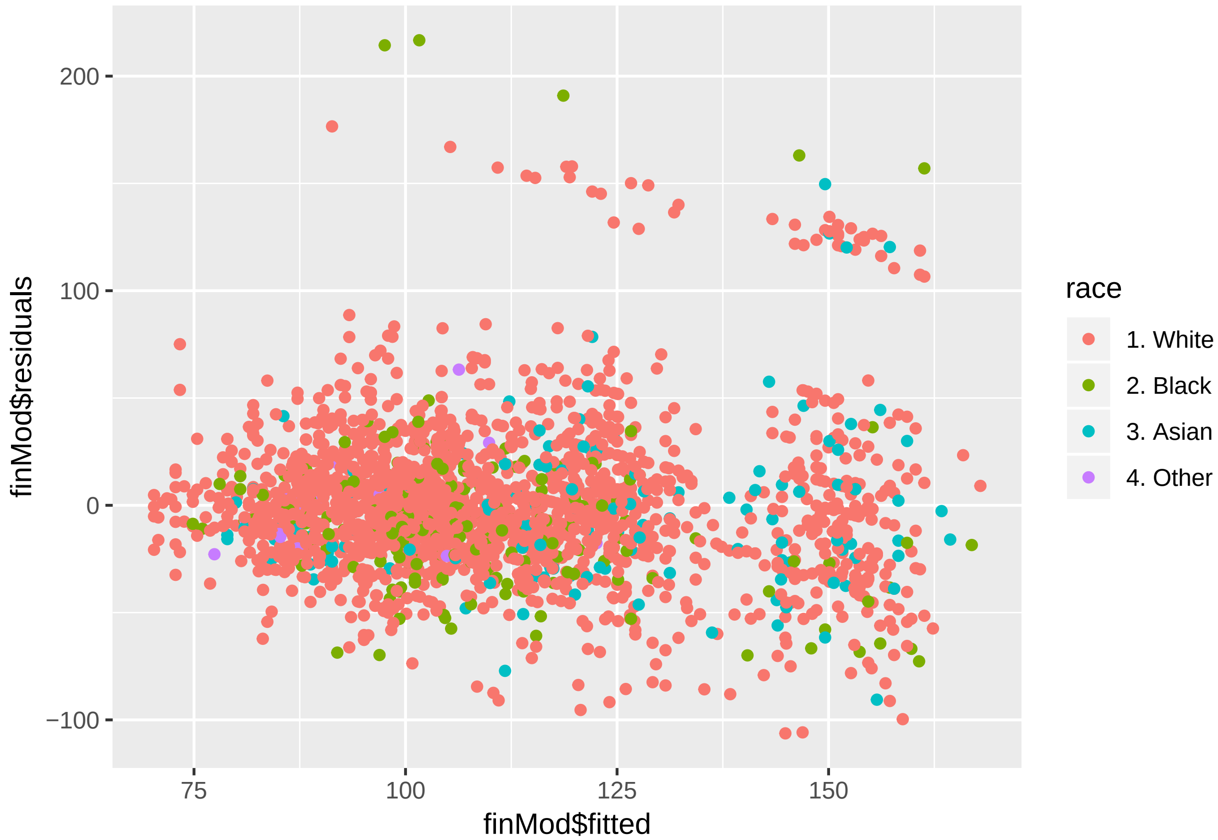
```
## Linear Regression
##
## 2102 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   35.79066  0.248127  24.68782
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
plot(finMod,1,pch=19,cex=0.5,col="#00000010")
```



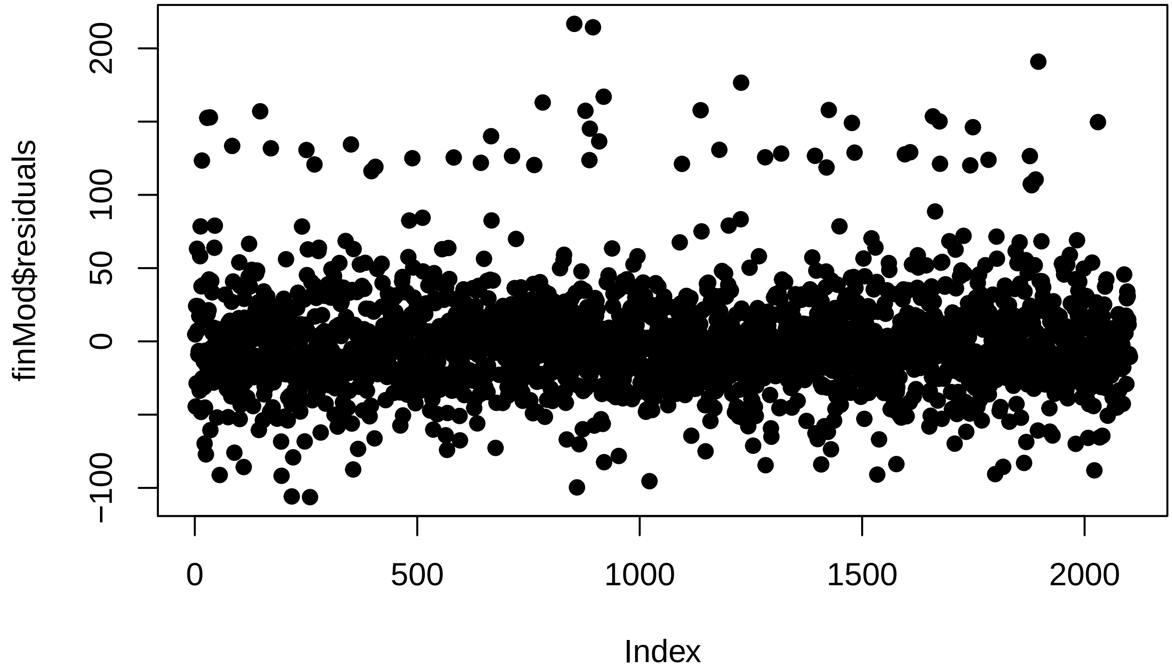
Color by variables not used in the model

```
qplot(finMod$fitted,finMod$residuals,color=race,data=training)
```



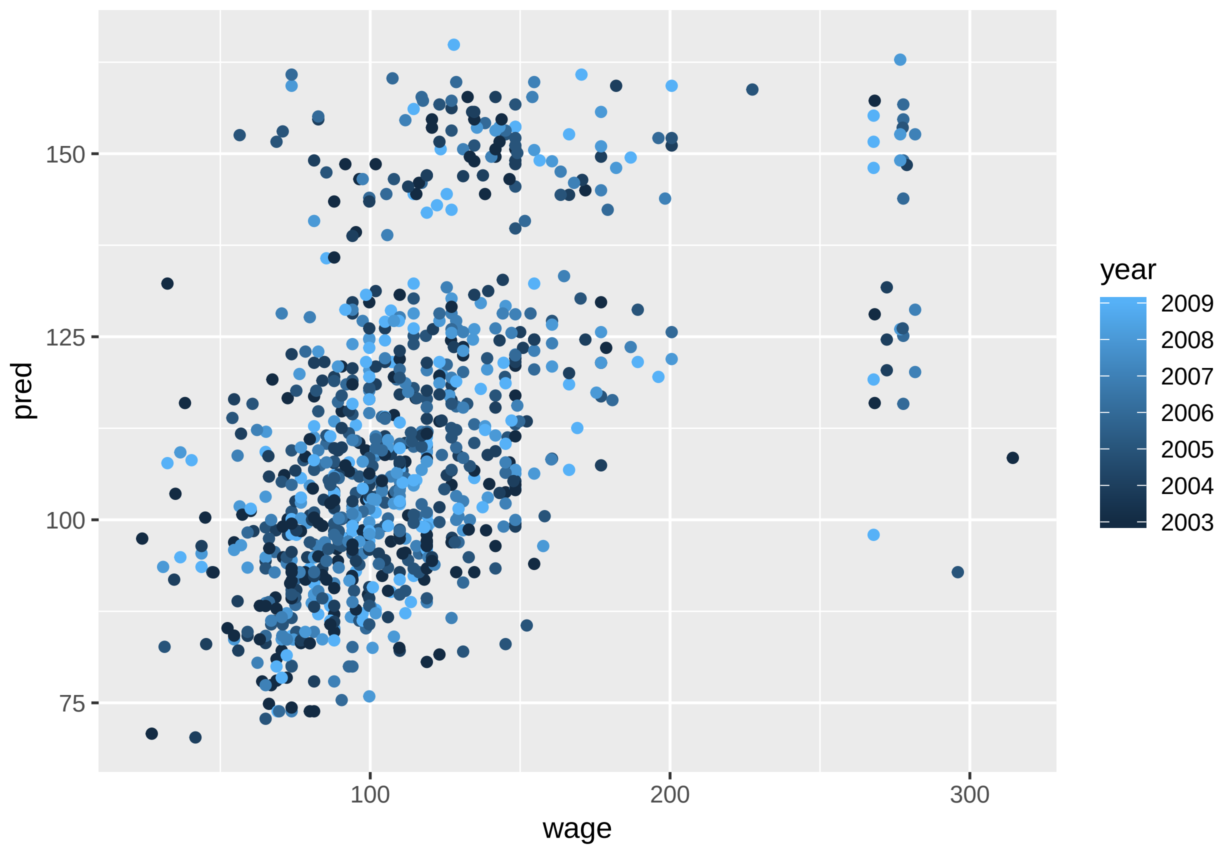
Plot by index (i.e. which rows in the dataframe they are at)

```
plot(finMod$residuals,pch=19)
```



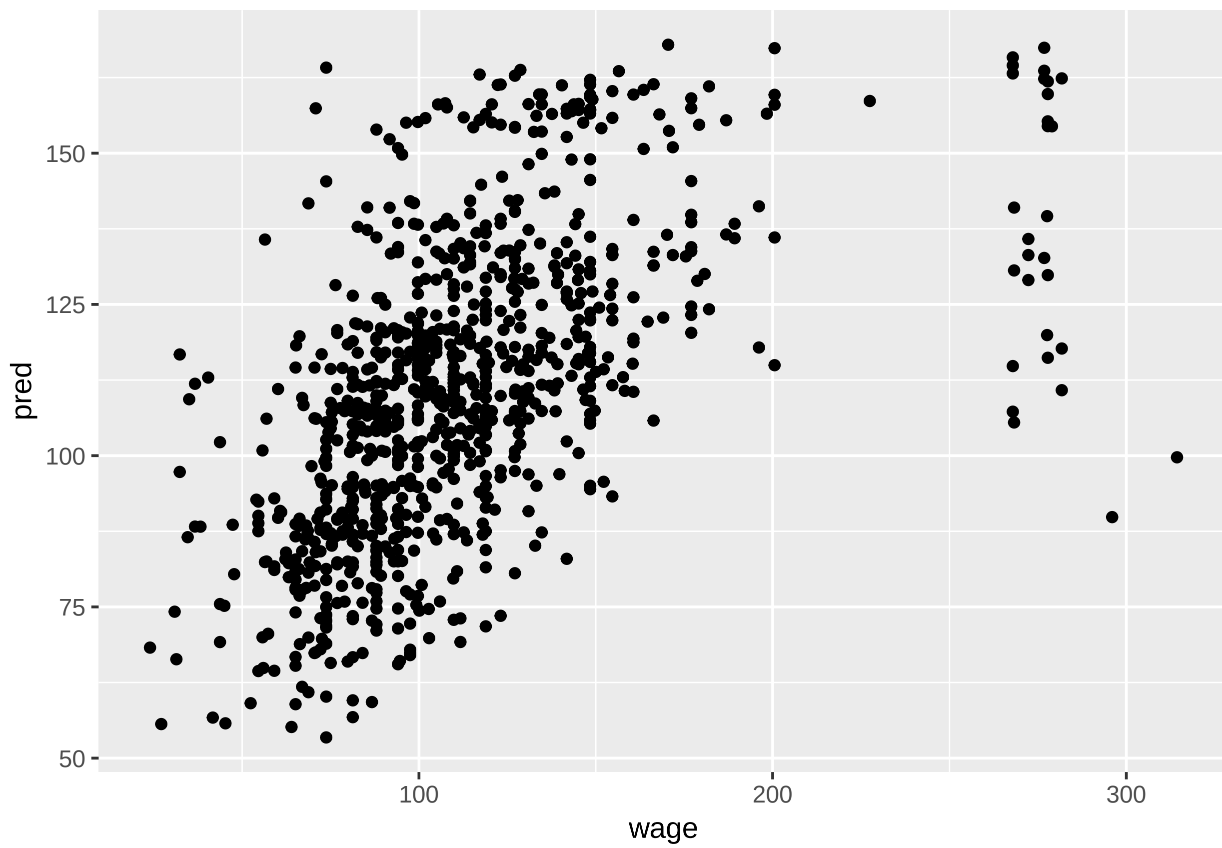
Predicted vs. truth in test set

```
pred<-predict(modFit,testing)
qplot(wage,pred,color=year,data=testing)
```



If you want to use all covariates (variables)

```
modFitAll<-train(wage~.,data=training,method="lm")
pred<-predict(modFitAll,newdata=testing)
qplot(wage,pred,data=testing)
```


pred
150
125
100
75
50
100
200
300
wage