Preprocessing

Jeffrey Leek

May 18, 2016

Why preprocess?

```
library(caret); library(kernlab); data(spam)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'kernlab'

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

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

Why preprocess?

```
mean(training$capitalAve)
```

```
## [1] 4.910247
```

```
sd(training$capitalAve)
```

```
## [1] 28.30757
```

Standardizing

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

```
## [1] -1.150422e-17
```

```
sd(trainCapAveS)
```

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

Standardizing - test set

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

```
## [1] 0.03975327
```

```
sd(testCapAveS)
```

```
## [1] 1.423405
```

Standardizing - *preProcess* function

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

```
## [1] -1.150422e-17
```

```
sd(trainCapAveS)
```

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

Standardizing - *preProcess* function

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

```
## [1] 0.03975327
```

```
sd(testCapAveS)
```

```
## [1] 1.423405
```

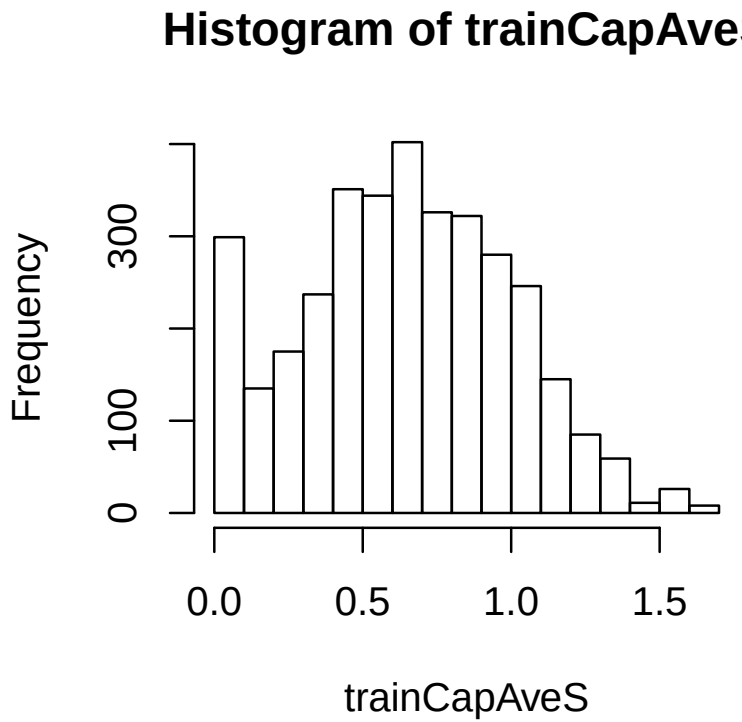
Standardizing - *preProcess* argument

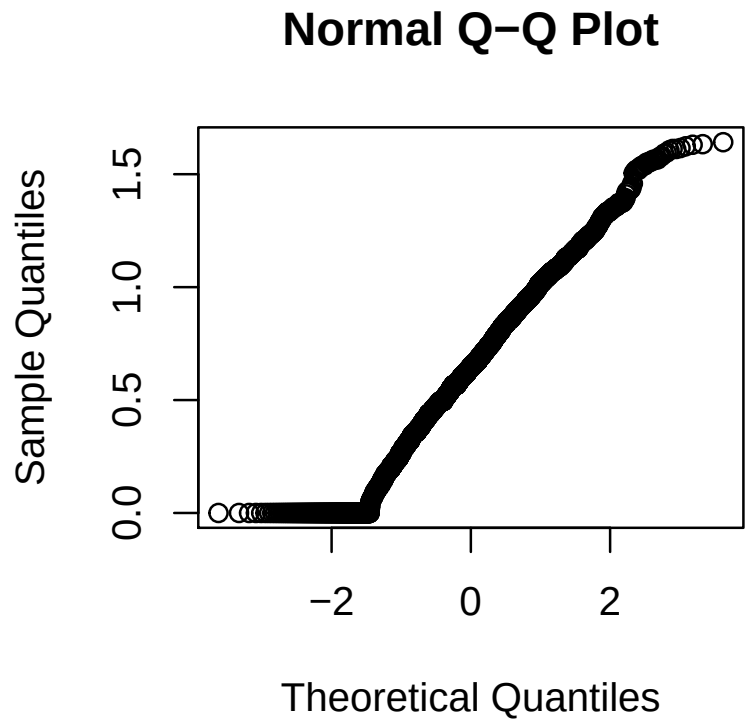
```
set.seed(32343)
modelFit <- train(type ~.,data=training,
                  preProcess=c("center","scale"),method="gl
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or
```

Standardizing - Box-Cox transforms

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





Standardizing - Imputing data

```
set.seed(13343)

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

# Impute 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(capAveTru
```

Standardizing - Imputing data

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

```
##             0%            25%            50%            75%
## -2.816863e+00 -1.317386e-03 -1.415817e-05  5.857060e-04
```

```
quantile((capAve - capAveTruth)[selectNA])
```

```
##             0%            25%            50%            75%
## -2.8168632080 -0.0169680286  0.0009679843  0.0218804267
```

```
quantile((capAve - capAveTruth)[!selectNA])
```

```
##             0%            25%            50%            75%
## -9.489725e-01 -1.232646e-03 -2.019562e-05  5.464626e-04
```

Notes and further reading

- Training and test must be processed in the same way
- Test transformations will likely be imperfect
- Especially if the test/training sets collected at different times
- Careful when transforming factor variables!
- preprocessing with caret