Chapter 3

Chapter 3. Data pre-processing

addition, deletion or transformation of training set data

3.1 Case study: cell segmentation in high-content screening

3.2 Data transformation for individual predictors

Centering and scaling

zero mean and standard deviation of one

Transformations to resolve skewness

log, square root, or inverse Boxcox

3.3 Data transformation for multiple predicors

Transformations to resolve outliers

Spatial sign

Data reduction and feature extraction

generate a smaller set of prefictors that seek to capture a majority of the information in the original variables

PCA PCA seeks predictor-set variation without regard to any further understanding of the predictors or to knowledge of the modeling objectives

3.4 Dealing with missing values

structurally missing informative missingness censored data

Imputation

Imputation is just another layer of modeling where we try to estimate values of the predictor variables based on other predictor variables

KNN for imputation

3.5 Removing predictors

Zero variance predictor Between predictor correlations

3.6 Adding predictors

dummy variables nonlinearality

3.7 Binning predictors

3.8 Computing

```
library(AppliedPredictiveModeling)
library(caret)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(e1071)
library(lattice)
library(caret)
```

```
apropos("confusion")
```

```
## [1] "confusionMatrix"       "confusionMatrix.train"
```

```
RSiteSearch("confusion",restrict='function')
```

```
## A search query has been submitted to http://search.r-project.org
## The results page should open in your browser shortly
```

```
data(segmentationOriginal)
segData <- subset(segmentationOriginal, Case=="Train")
```

```
cellID <- segData$Cell
class <- segData$Class
case <- segData$Case
segData <- segData[,-(1:3)]
```

```
statusColNum <- grep("Status",names(segData))
statusColNum
```

```
##  [1]   2   4   9  10  11  12  14  16  20  21  22  26  27  28  30  32  34
## [18]  36  38  40  43  44  46  48  51  52  55  56  59  60  63  64  68  69
## [35]  70  72  73  74  76  78  80  82  84  86  88  92  93  94  97  98 103
## [52] 104 105 106 110 111 112 114
```

```
segData <- segData[,-statusColNum]
```

Transformation

```
skewness(segData$AngleCh1)
```

```
## [1] -0.02426252
```

```
skewValues <- apply(segData, 2, skewness)
head(skewValues)
```

```
##    AngleCh1     AreaCh1 AvgIntenCh1 AvgIntenCh2 AvgIntenCh3 AvgIntenCh4
## -0.02426252  3.52510745  2.95918524  0.84816033  2.20234214  1.90047128
```

```
Ch1AreaTrans <- BoxCoxTrans(segData$AreaCh1)
Ch1AreaTrans
```

```
## Box-Cox Transformation
##
## 1009 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   150.0   194.0   256.0   325.1   376.0  2186.0
##
## Largest/Smallest: 14.6
## Sample Skewness: 3.53
##
## Estimated Lambda: -0.9
```

```
# The original data
head(segData$AreaCh1)
```

```
## [1] 819 431 298 256 258 358
```

```
# After transformation
predict(Ch1AreaTrans, head(segData$AreaCh1))
```

```
## [1] 1.108458 1.106383 1.104520 1.103554 1.103607 1.105523
```

```
(819^(-0.9)-1)/(-0.9)
```

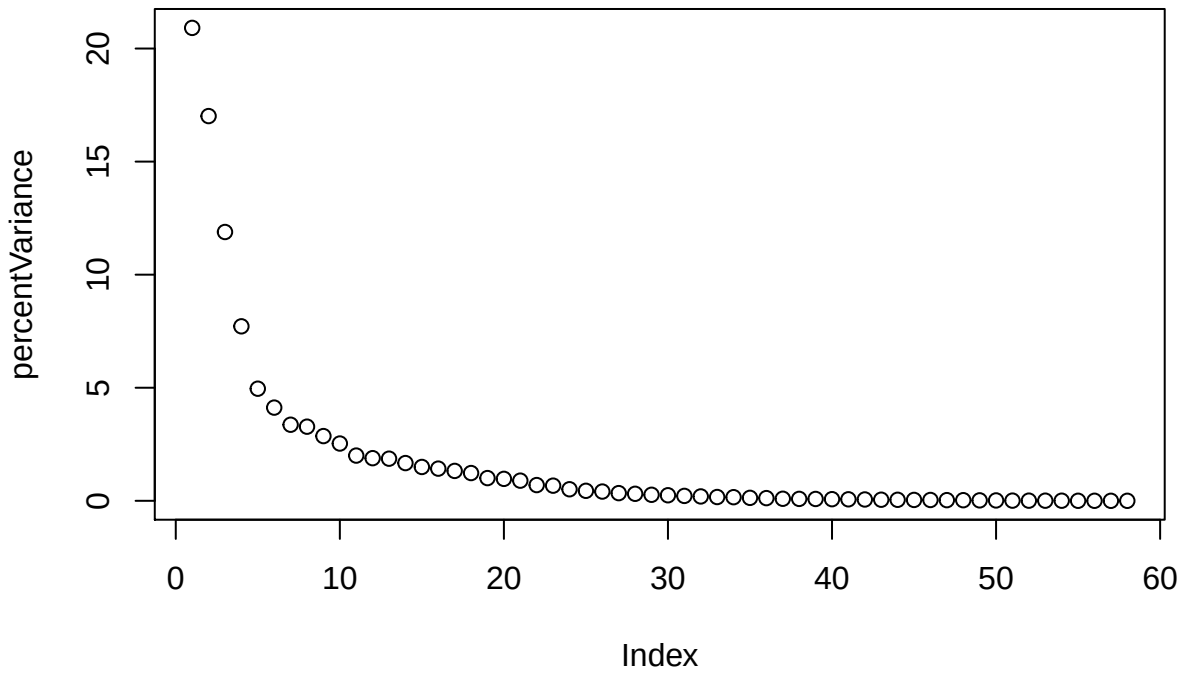
```
## [1] 1.108458
```

```
pcaObject <- prcomp(segData, center=TRUE, scale. = TRUE)

# Calculate the cumulative percentage of variance which each component accounts for
percentVariance <- pcaObject$sd^2/sum(pcaObject$sd^2)*100
percentVariance[1:3]
```

```
## [1] 20.91236 17.01330 11.88689
```

```
plot(percentVariance)
```



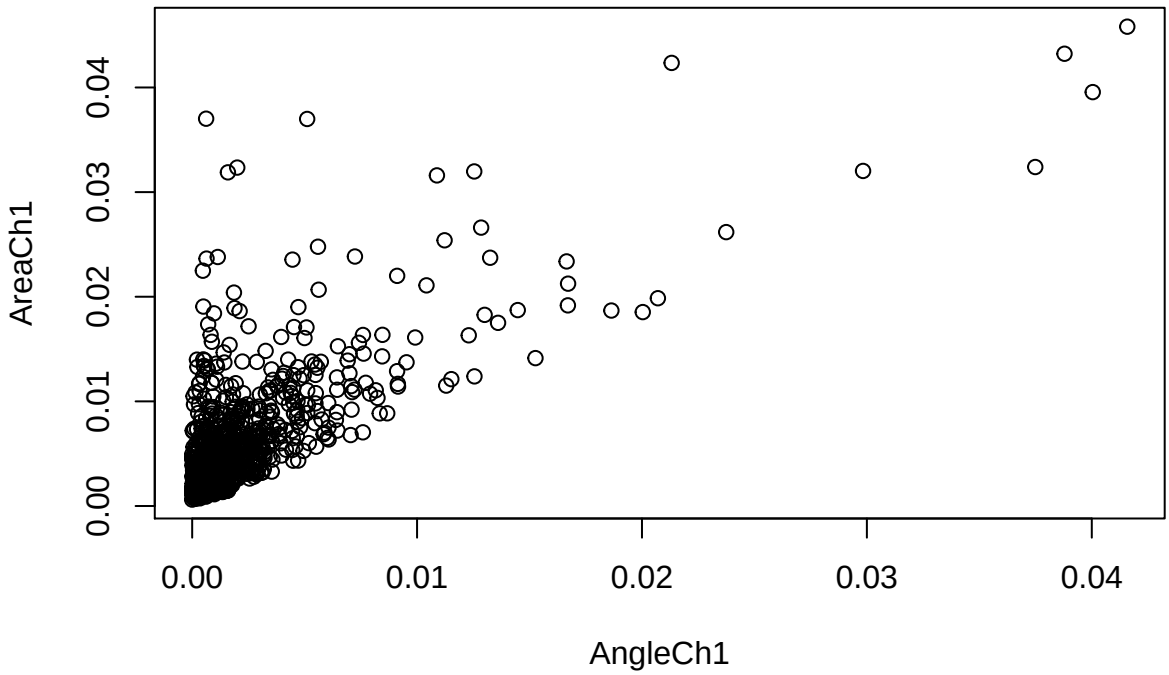

```
# The transformed values are stored in pcaObject as a sub-object called x
head(pcaObject$x[,1:5])
```

```
##           PC1        PC2         PC3       PC4        PC5
## 2   5.0985749  4.5513804 -0.03345155 -2.640339  1.2783212
## 3  -0.2546261  1.1980326 -1.02059569 -3.731079  0.9994635
## 4   1.2928941 -1.8639348 -1.25110461 -2.414857 -1.4914838
## 12 -1.4646613 -1.5658327  0.46962088 -3.388716 -0.3302324
## 15 -0.8762771 -1.2790055 -1.33794261 -3.516794  0.3936099
## 16 -0.8615416 -0.3286842 -0.15546723 -2.206636  1.4731658
```

```
head(pcaObject$rotation[,1:3])
```

```
##                       PC1         PC2          PC3
## AngleCh1      0.001213758 -0.01284461  0.006816473
## AreaCh1       0.229171873  0.16061734  0.089811727
## AvgIntenCh1 -0.102708778  0.17971332  0.067696745
## AvgIntenCh2 -0.154828672  0.16376018  0.073534399
## AvgIntenCh3 -0.058042158  0.11197704 -0.185473286
## AvgIntenCh4 -0.117343465  0.21039086 -0.105060977
```

```
SS <- spatialSign(segData)
plot(SS)
```




```
trans <- preProcess(segData, method=c("BoxCox","center","scale","pca"))
trans
```

```
## Created from 1009 samples and 58 variables
##
## Pre-processing:
##   - Box-Cox transformation (47)
##   - centered (58)
##   - ignored (0)
##   - principal component signal extraction (58)
##   - scaled (58)
##
## Lambda estimates for Box-Cox transformation:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.00000 -0.50000 -0.10000  0.05106  0.30000  2.00000
##
## PCA needed 19 components to capture 95 percent of the variance
```

```
# Apply the transformations
transformed <- predict(trans, segData)
# These values are different than the previous PCA components since they were transformed prior to PCA
head(transformed[,1:5])
```

```
##           PC1        PC2        PC3       PC4        PC5
## 2   1.5684742  6.2907855 -0.3333299 -3.063327 -1.3415782
## 3  -0.6664055  2.0455375 -1.4416841 -4.701183 -1.7422020
## 4   3.7500055 -0.3915610 -0.6690260 -4.020753  1.7927777
## 12  0.3768509 -2.1897554  1.4380167 -5.327116 -0.4066757
## 15  1.0644951 -1.4646516 -0.9900478 -5.627351 -0.8650174
## 16 -0.3798629  0.2173028  0.4387980 -2.069880 -1.9363920
```

```
# The order in which the possible transformation are applied is transformation, centering, scaling, imp
```

Filtering

```
nearZeroVar(segData)
```

```
## integer(0)
```

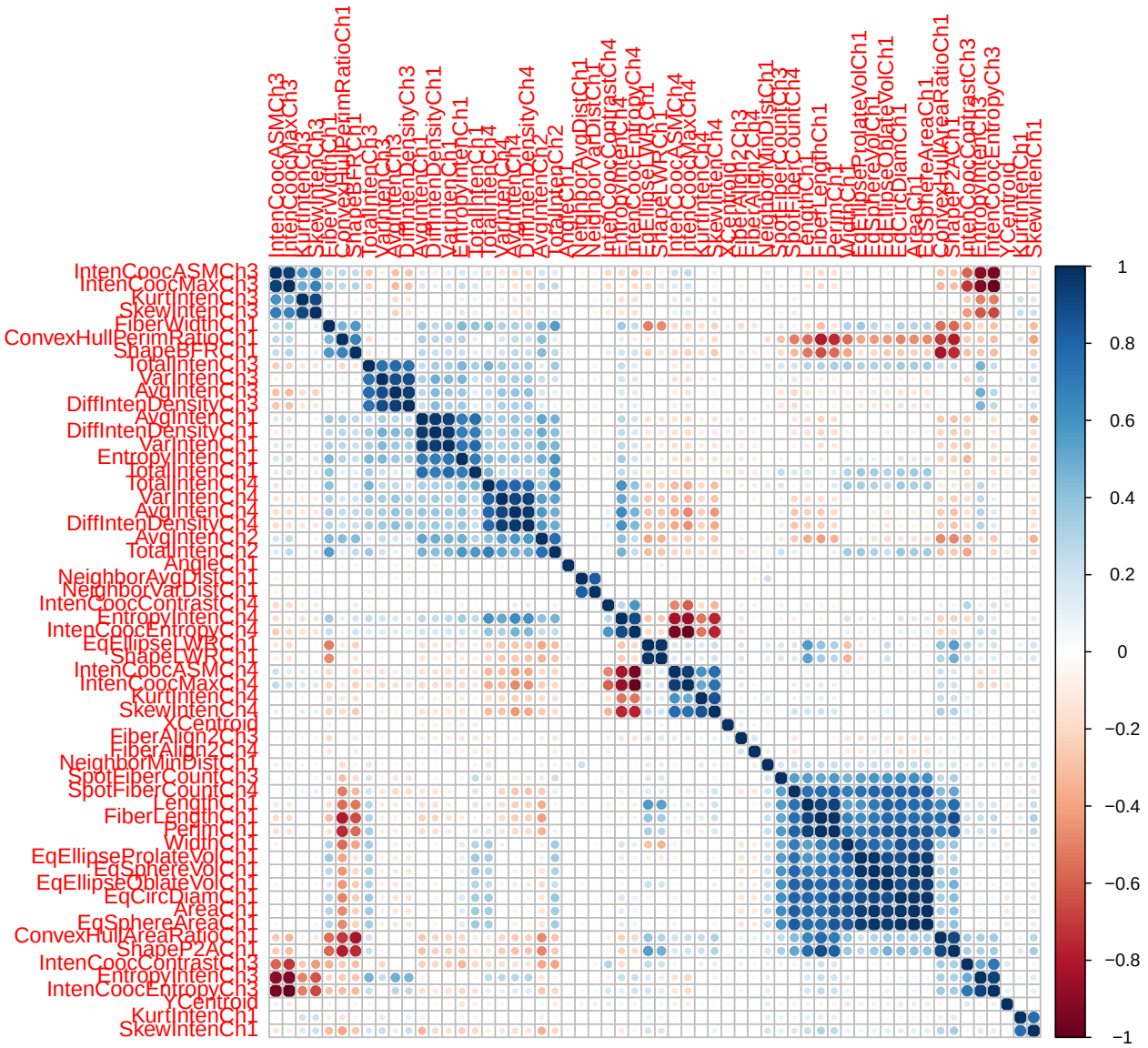
```
correlations <- cor(segData)
dim(correlations)
```

```
## [1] 58 58
```

```
correlations[1:4,1:4]
```

```
##                 AngleCh1      AreaCh1 AvgIntenCh1 AvgIntenCh2
## AngleCh1     1.000000000 -0.002627172 -0.04300776 -0.01944681
## AreaCh1     -0.002627172  1.000000000 -0.02529739 -0.15330301
## AvgIntenCh1 -0.043007757 -0.025297394  1.00000000  0.52521711
## AvgIntenCh2 -0.019446810 -0.153303007  0.52521711  1.00000000
```

```
library(corrplot)
corrplot(correlations, order="hclust")
```




```
highCorr <- findCorrelation(correlations, cutoff=.75)
length(highCorr)
```

```
## [1] 32
```

```
head(highCorr)
```

```
## [1] 23 40 43 36  7 15
```

```
filteredSegData <- segData[,-highCorr]
```

Creating dummy variables

```
data(cars)
type <- c("convertible", "coupe", "hatchback", "sedan", "wagon")
cars$Type <- factor(apply(cars[, 14:18], 1, function(x) type[which(x == 1)]))

carSubset <- cars[sample(1:nrow(cars), 20), c(1, 2, 19)]
```

```
head(carSubset)
```

```
##        Price Mileage  Type
## 612 17944.86   19592 sedan
## 322 12846.06   27560 sedan
## 81  11149.62   34447 coupe
## 233 24903.48   40719 wagon
## 120 19774.25   23359 sedan
## 752 35895.50   23056 sedan
```

```
levels(carSubset$Type)
```

```
## [1] "convertible" "coupe"       "hatchback"   "sedan"       "wagon"
```

```
simpleMod <- dummyVars(~Mileage+Type, data=carSubset, levelsOnly=TRUE)
simpleMod
```

```
## Dummy Variable Object
##
## Formula: ~Mileage + Type
## 2 variables, 1 factors
## Factor variable names will be removed
## A less than full rank encoding is used
```

```
predict(simpleMod,head(carSubset))
```

```
##     Mileage convertible coupe hatchback sedan wagon
## 612   19592           0     0         0     1     0
## 322   27560           0     0         0     1     0
## 81    34447           0     1         0     0     0
## 233   40719           0     0         0     0     1
## 120   23359           0     0         0     1     0
## 752   23056           0     0         0     1     0
```

```
withInteraction <- dummyVars(~Mileage + Type + Mileage:Type, data=carSubset, levelsOnly=TRUE)
withInteraction
```

```
## Dummy Variable Object
##
## Formula: ~Mileage + Type + Mileage:Type
## 2 variables, 1 factors
## Factor variable names will be removed
## A less than full rank encoding is used
```

```r
predict(withInteraction, head(carSubset))
```

```
##     Mileage convertible coupe hatchback sedan wagon
## 612   19592           0     0         0     1     0
## 322   27560           0     0         0     1     0
## 81    34447           0     1         0     0     0
## 233   40719           0     0         0     0     1
## 120   23359           0     0         0     1     0
## 752   23056           0     0         0     1     0
##     Mileage:Typeconvertible Mileage:Typecoupe Mileage:Typehatchback
## 612                       0                 0                     0
## 322                       0                 0                     0
## 81                        0             34447                     0
## 233                       0                 0                     0
## 120                       0                 0                     0
## 752                       0                 0                     0
##     Mileage:Typesedan Mileage:Typewagon
## 612             19592                 0
## 322             27560                 0
## 81                  0                 0
## 233                 0             40719
## 120             23359                 0
## 752             23056                 0
```