Predictive Modeling

Chapter 3: Data Pre-processing

STA 6543

The University of Texas at San Antonio

What is data pre-processing?

- Data pre-processing generally refers to the addition, deletion, or transformation of the *training* data.
- Different predictive models usually have different sensitivities to the predictors.
- The need for data pre-processing is determined by the type of models being used.
  - Some models, such as *tree-based models*, are notably insensitive to the characteristics of the predictor data. Others, like *linear regression*, are not.
  - Care should be paid to check which, if any, pre-processing techniques can be useful.

A summary of models and some of their characteristics

Table A.1: A summary of models and some of their characteristics

Model	Allows $n < p$	Pre-processing	Interpretable	Automatic feature selection	# Tuning parameters	Robust to predictor noise	Computation time
Linear regression†	×	CS, NZV, Corr	✓	×	0	×	✓
Partial least squares	✓	CS	✓	○	1	×	✓
Ridge regression	×	CS, NZV	✓	×	1	×	✓
Elastic net/lasso	×	CS, NZV	✓	✓	1–2	×	✓
Neural networks	✓	CS, NZV, Corr	×	×	2	×	×
Support vector machines	✓	CS	×	×	1–3	×	×
MARS/FDA	✓		○	✓	1–2	○	○
K-nearest neighbors	✓	CS, NZV	×	×	1	○	✓
Single trees	✓		○	✓	1	✓	✓
Model trees/rules†	✓		○	✓	1–2	✓	✓
Bagged trees	✓		×	✓	0	✓	○
Random forest	✓		×	○	0–1	✓	×
Boosted trees	✓		×	✓	3	✓	×
Cubist†	✓		×	○	2	✓	×
Logistic regression*	×	CS, NZV, Corr	✓	×	0	×	✓
{LQRM}DA*	×	NZV	○	×	0–2	×	✓
Nearest shrunken centroids*	✓	NZV	○	✓	1	×	✓
Naïve Bayes*	✓	NZV	×	×	0–1	○	○
C5.0*	✓		○	✓	0–3	✓	×

†regression only *classification only

Symbols represent affirmative (✓), negative (×), and somewhere in between (○)

- Table A1 of the textbook.

Motivating example: cell segmentation in high-content screening

- Medical researchers often assess the cell characteristics of a living organism or plant to understand the effects of medicines or diseases on the size, shape, development status, and number of cells.
- There are two ways to do this:
  1) Experts can examine the target serum or tissue under a microscope and manually assess the desired cell characteristics. This work is tedious and requires expert knowledge of the cell type and characteristics.
  2) Another way to measure the cell characteristics from these kinds of samples is by using high-content screening.

High-content screening

- A sample is first dyed with a substance that will bind to the desired characteristic of the cells.
- The sample is then interrogated by an instrument (such as a confocal microscope),where the dye deflects light and the detectors quantify the degree of scattering for that specific wavelength.
- The light scattering measurements are then processed through imaging software to quantify the desired cell characteristics.
- Using an automated, high-throughput approach to assess samples' cell characteristics can *sometimes produce misleading results*.

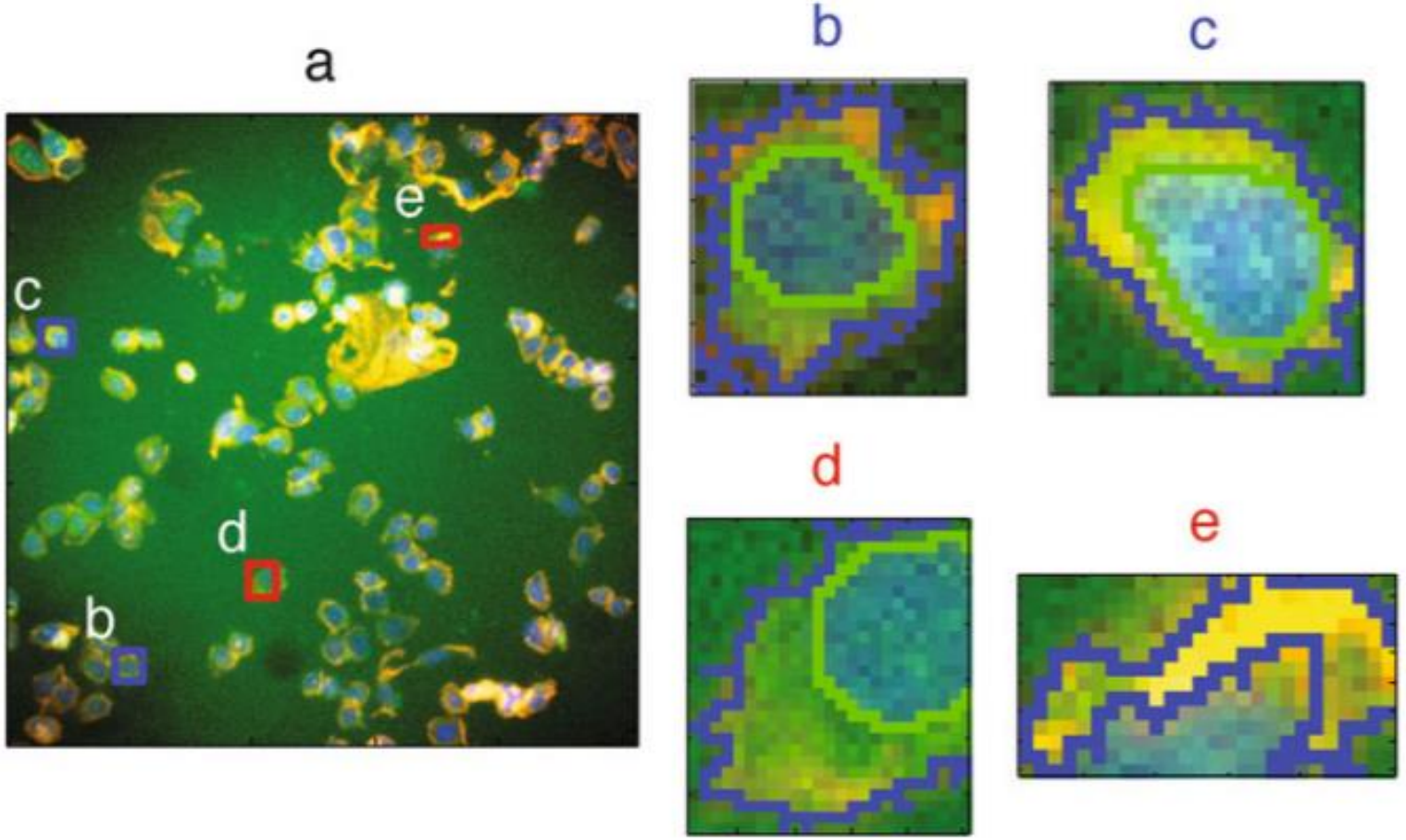


Fig. 3.1: An image showing cell segmentation from Hill et al. (2007). The *red boxes* [panels (**d**) and (**e**)] show poorly segmented cells while the cells in the *blue boxes* are examples of proper segmentation

Data pre-processing techniques

- *Data transformations* (e.g., centering and scaling, resolving skewness, resolving outliers, data reduction, etc)
- *Dealing with missing values* (e.g., removal of missing sample, data imputations)
- *Removing predictors* (e.g., near-zero variance predictor, multicollinearity)
- *Creating dummy variables* (e.g., categorical variables)
- *Binning predictors* (e.g., binning a numerical predictor to two or more groups)
- ……

Centering and scaling

- The most straightforward and common data transformation is to center and scale the data.
- To center the data, the average variable value is subtracted from all the values to make the predictors have a zero mean.
- To scale the data, each value of the variable is divided by its standard deviation to make the predictors have a common standard deviation of one.

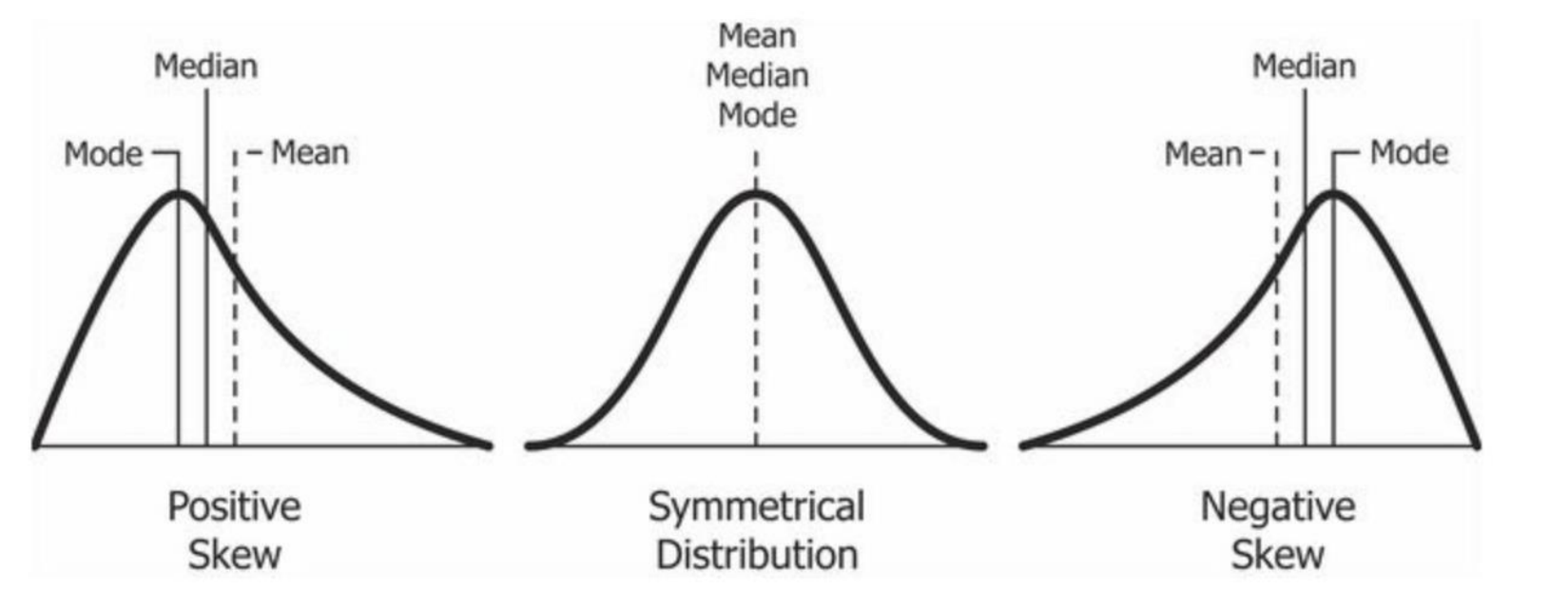
Centering and scaling

- Pros:
  - Improve the numerical stability of some calculations
  - Some methods, such as PCA or PLS benefit from the data being on a common scale
- Cons:
  - Loss of interpretability of the individual value.

In R Usage: scale(x, center = TRUE, scale = TRUE)

Transformation to resolve skewness

- Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.



Transformation to resolve skewness

- If the distribution is roughly *symmetric*, the skewness values will be close to *zero*.
- If the distribution becomes more *right skewed*, the skewness statistic becomes *larger*.
- If the distribution becomes more *left skewed*, the value becomes *negative*.

- The rule of thumb: skewed data whose ratio of the highest value to the lowest value is greater than 20 have significant skewness.

Transformation to resolve skewness

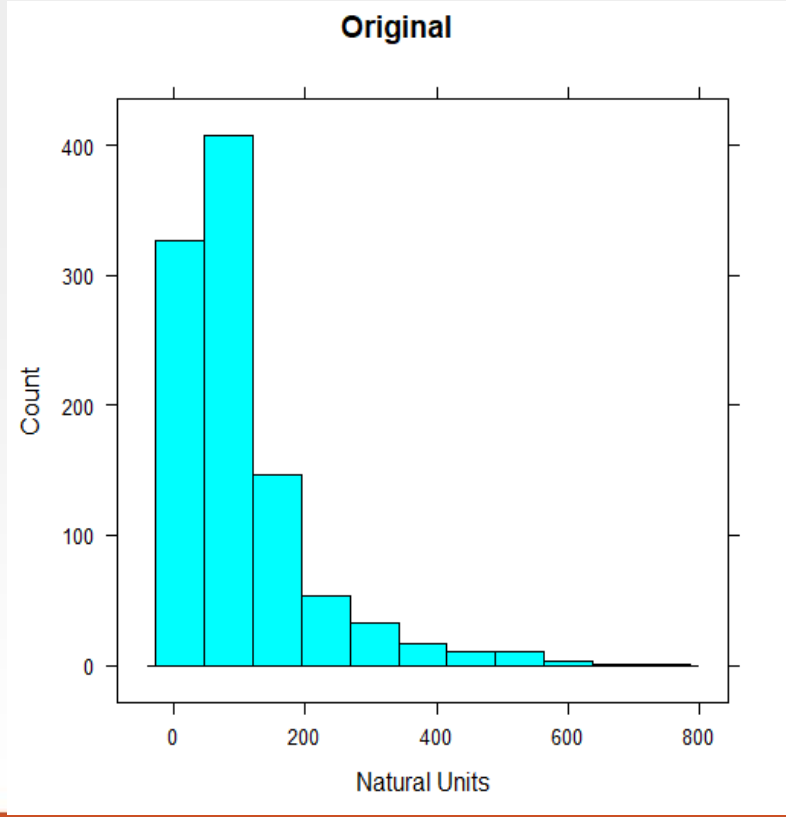
- Box-cox transformation

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

- It includes square transformation (λ = 2), square root (λ = 0.5), and inverse (λ = -1). Often times, some approximations are implemented.

Motivating example: cell segmentation data

- The cell segmentation data contain a predictor that measures the standard deviation of the intensity of the pixels in the actin filaments.



Motivating example: cell segmentation data

```
library(AppliedPredictiveModeling)
data(segmentationOriginal)

## Retain the original training set
segTrain <- subset(segmentationOriginal, Case == "Train")

## Remove the first three columns (identifier columns)
segTrainX <- segTrain[, -(1:3)]
segTrainClass <- segTrain$Class #two levels: PS and WS
```

Motivating example: cell segmentation data

```
#Rule of thumbs > 20
max(segTrainX$VarIntenCh3)/min(segTrainX$VarIntenCh3)

#calculate the skewness of a predictor
library(e1071)
skewness(segTrainX$VarIntenCh3)

#Box-cox transformation
library(caret)
BoxCoxTrans(segTrainX$VarIntenCh3)
```

Box Cox transformation for one predictor

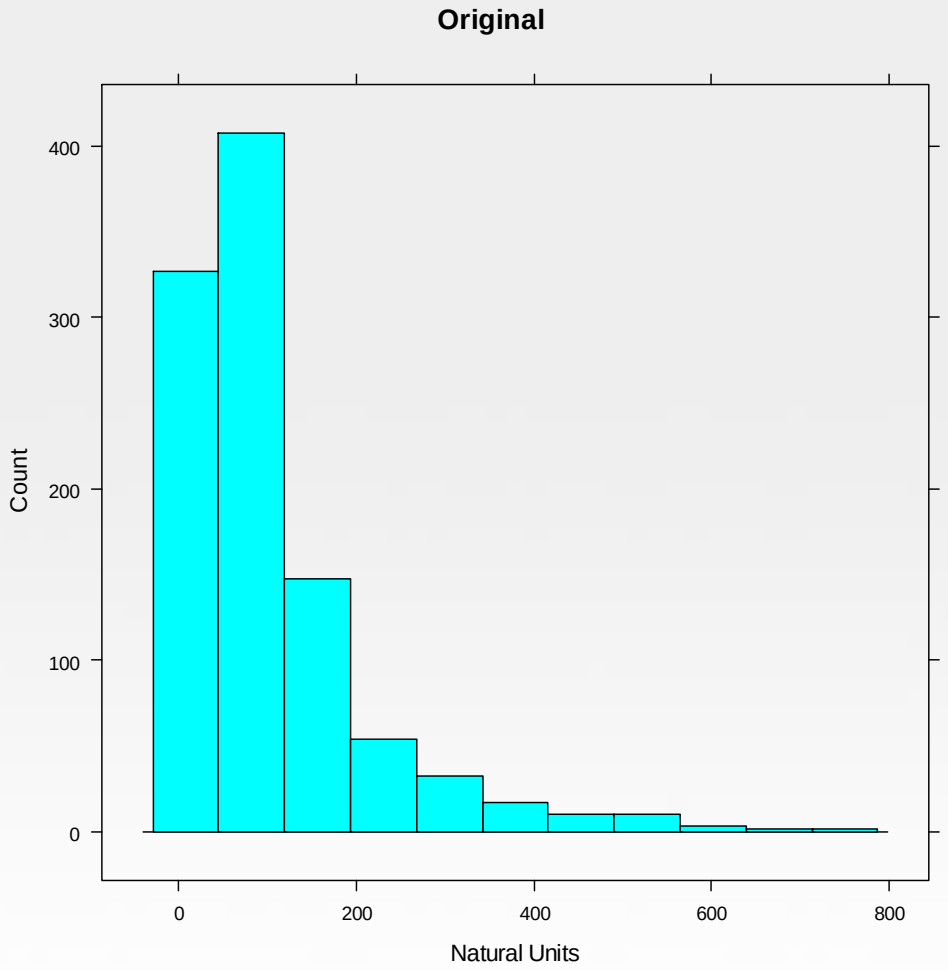
```
> library(caret)
> BoxCoxTrans(segTrainX$VarIntenCh3)
Box-Cox Transformation

1009 data points used to estimate Lambda

Input data summary:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.8693  37.0615  68.1316 101.6718 124.9899 757.0210

Largest/Smallest: 871
Sample Skewness: 2.39

Estimated Lambda: 0.1
With fudge factor, Lambda = 0 will be used for transformations
```

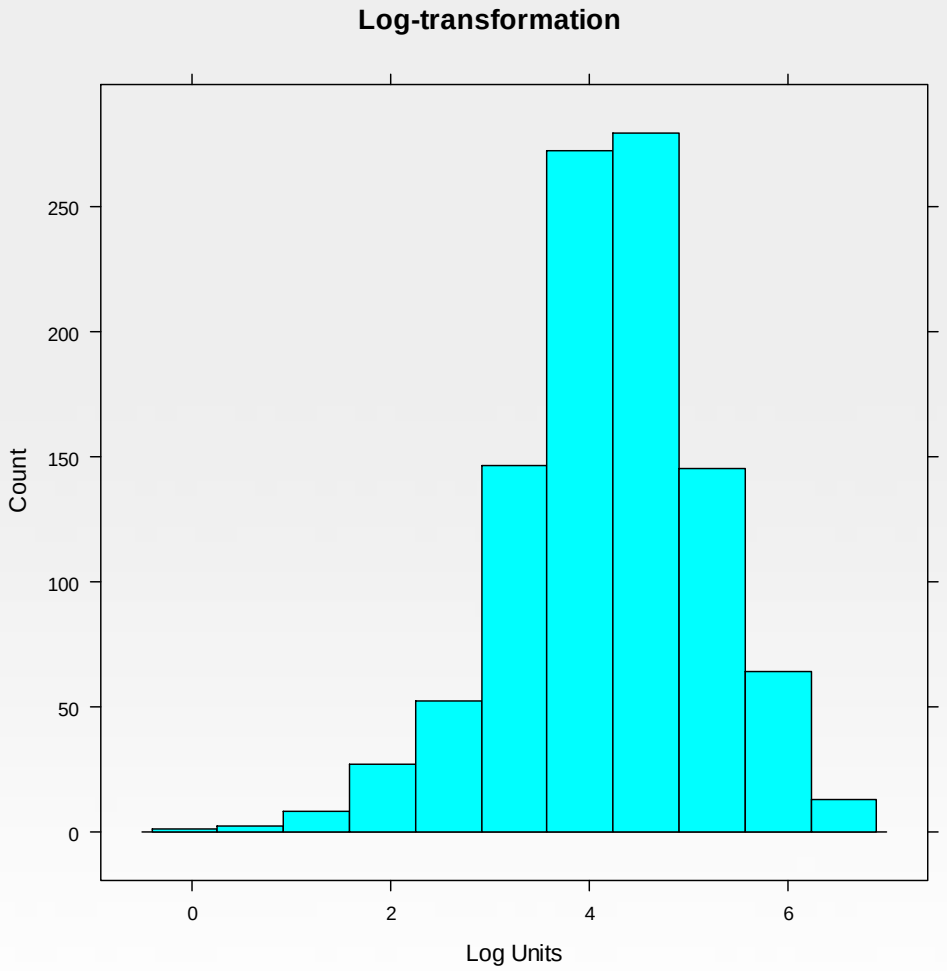
Conclusion: we may let λ = 0 indicating a log-transformation.

Box Cox transformation for one predictor

```
## Apply the transformations for predictor VarIntenCh3
VarIntenCh3BoxCox <- BoxCoxTrans(segTrainX$VarIntenCh3)
VarIntenCh3Trans <- predict(VarIntenCh3BoxCox, segTrainX$VarIntenCh3)

#Histgram comparisons before and after transformation
histogram(segTrainX$VarIntenCh3, xlab = "Natural Units",type = "count",main="Original")
histogram(VarIntenCh3Trans, xlab = “Log Units",type = "count", main="Log-transformation")
```

Histograms of before and after trans.





Box Cox transformation for another predictor PerimCh1

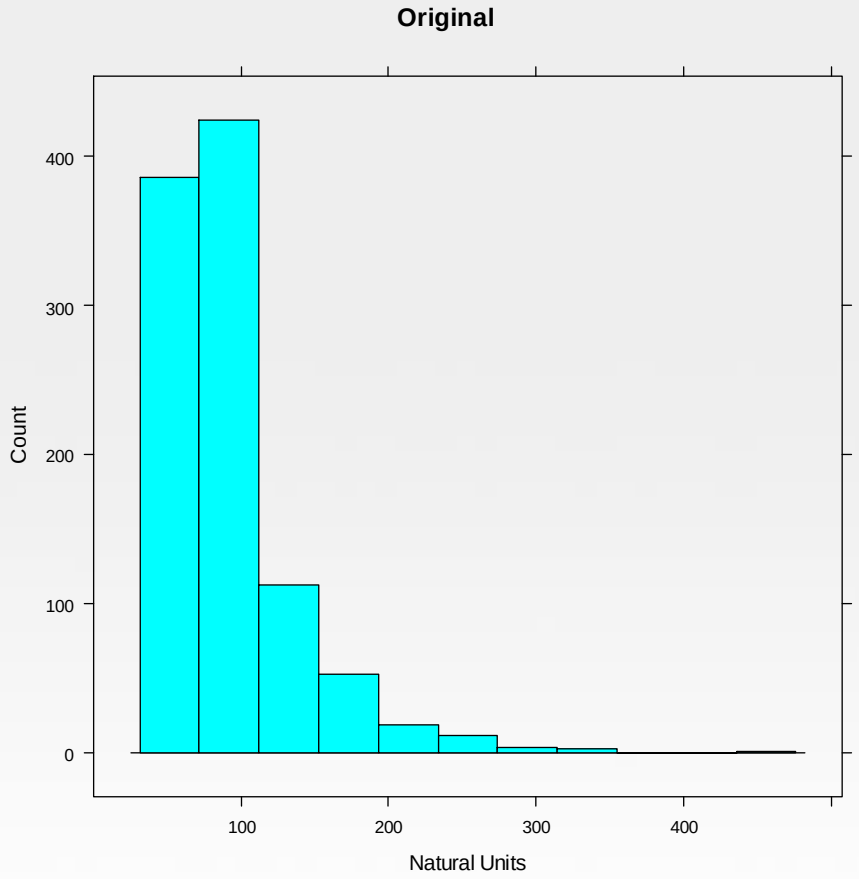
```
## Apply the transformations for predictor VarIntenCh3
PerimCh1BoxCox <- BoxCoxTrans(segTrainX$PerimCh1)
PerimCh1Trans <- predict(PerimCh1BoxCox, segTrainX$PerimCh1)

#Histgram comparisons before and after transformation
histogram(segTrainX$PerimCh1, xlab = "Natural Units",type = "count",main="Original")
histogram(PerimCh1Trans, xlab = “Log Units",type = "count", main="Log-transformation")
```

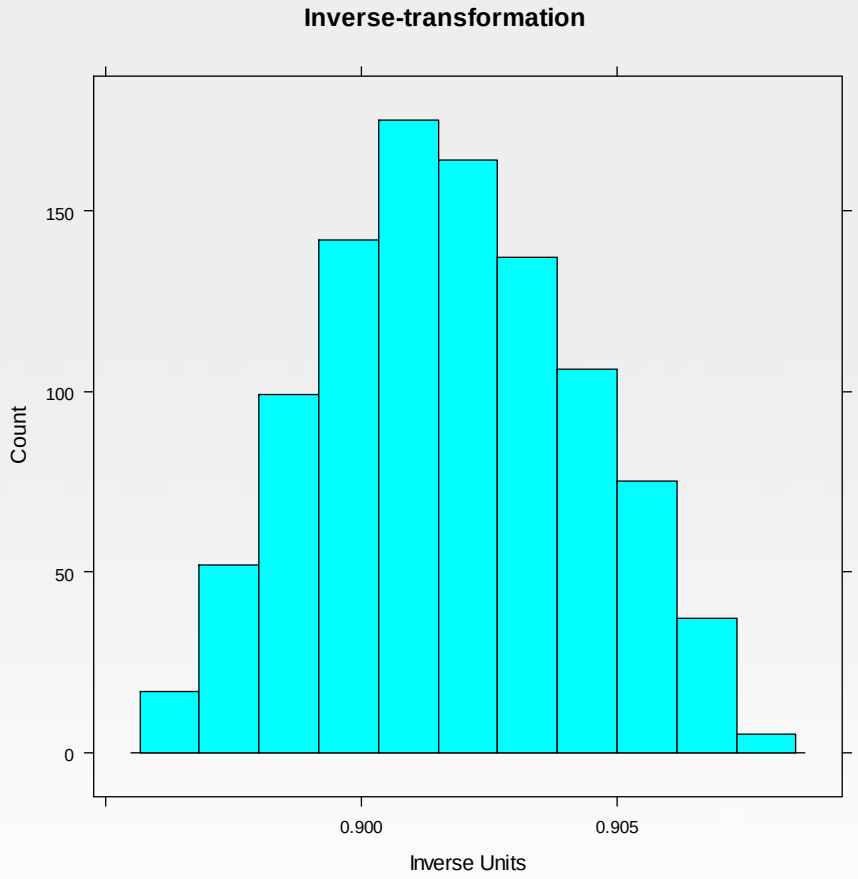
Box Cox transformation for another predictor PerimCh1

```
> BoxCoxTrans(segTrainX$PerimCh1)
Box-Cox Transformation

1009 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.74   64.37   79.02   91.61  103.24  459.77

Largest/Smallest: 9.63
Sample Skewness: 2.59

Estimated Lambda: -1.1
```

Conclusion: we may let λ = -1.1 indicating an inverse transformation.

Histograms of before and after trans.

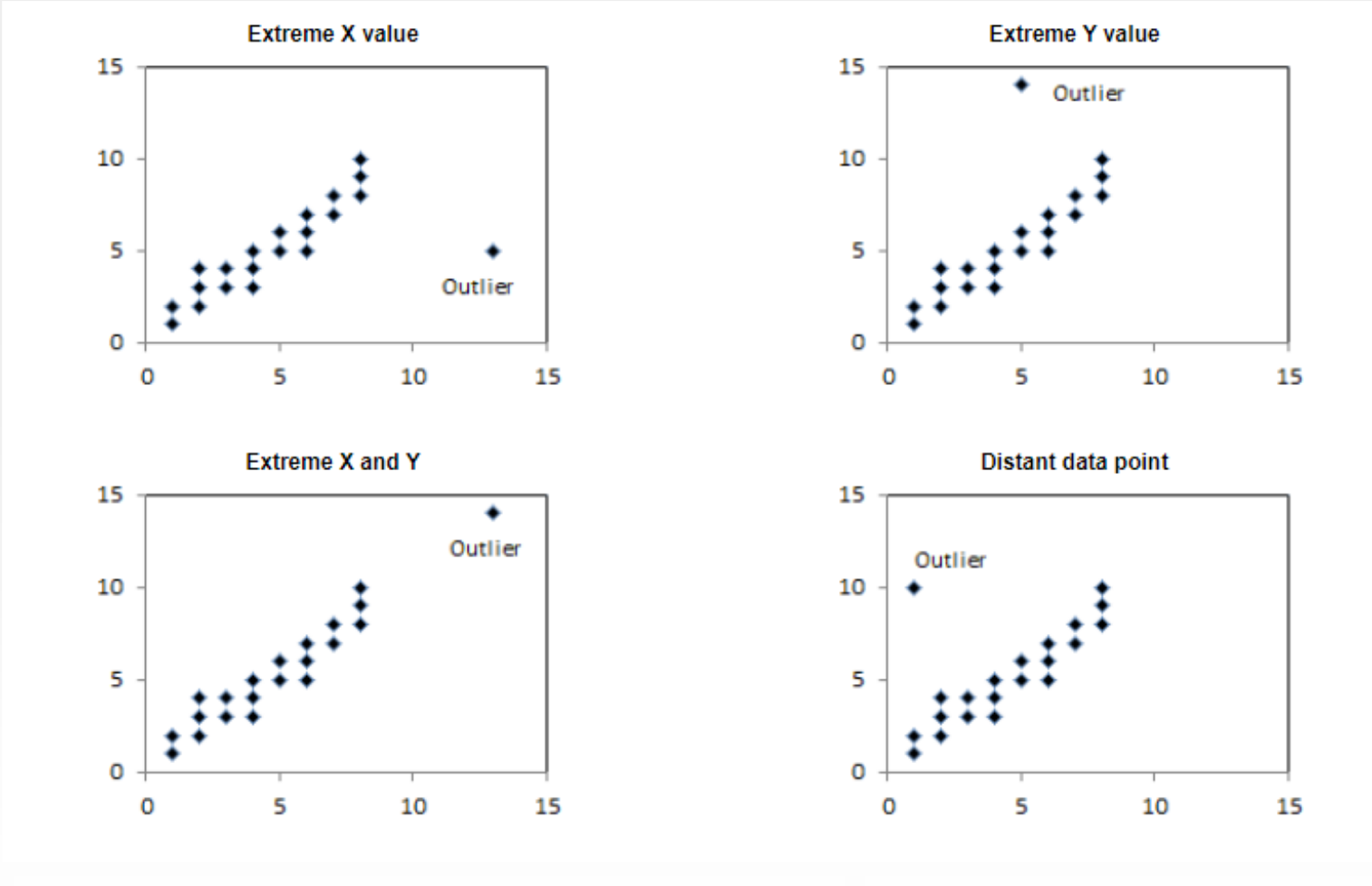




Outliers

- Outliers are defined as samples that are exceptionally far from the mainstream of the data.
- The outlying data may be an indication of a special part of the population under study that is just starting to be sampled.
- There are four ways that a data point might be considered an outlier.
  - It could have an extreme *X* value compared to other data points.
  - It could have an extreme *Y* value compared to other data points.
  - It could have extreme *X* and *Y* values.
  - It might be distant from the rest of the data, even without extreme *X* or *Y* values.

Outliers

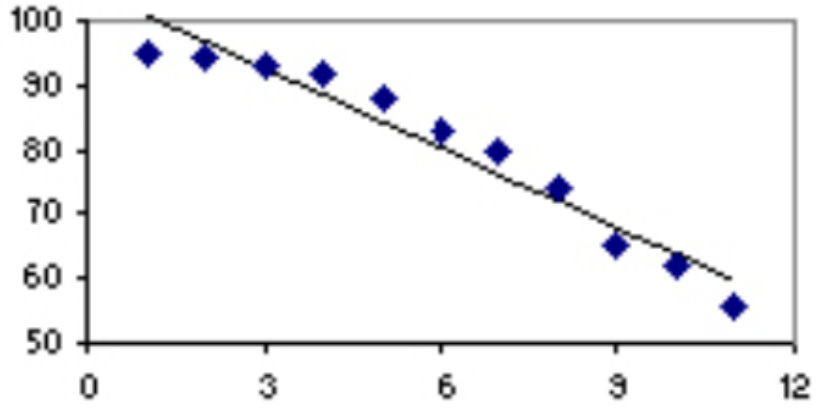

Extreme X value
Outlier
Extreme Y value
Outlier
Extreme X and Y
Outlier
Distant data point
Outlier

Influential points

- An influential point is an outlier that greatly affects the *slope* of the regression line.
- One way to test the influence of an outlier is to compute the regression equation with and without the outlier.

Influential points

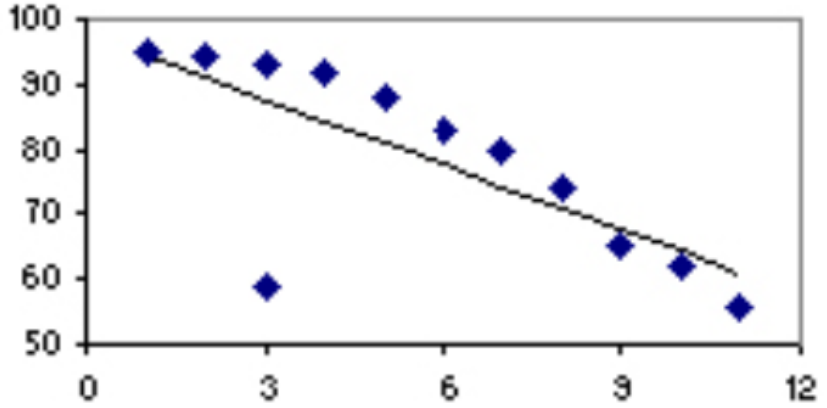
Without Outlier



Regression equation: $\hat{y} = 104.78 - 4.10x$

Coefficient of determination: $R^2 = 0.94$

With Outlier



Regression equation: $\hat{y} = 97.51 - 3.32x$

Coefficient of determination: $R^2 = 0.55$

- The scatterplots are identical, except that one plot includes an outlier. When the outlier is present, the slope is flatter (–4.10 vs. –3.32); so this outlier would be considered an influential point.

Spatial sign to resolve outliers

- There are several predictive models that are resistant to outliers; such as tree-based classification models, support vector machines (SVM) for classification.
- If a model is sensitive to outliers, a data transformation that can minimize the problem is the spatial sign.
- It projects the predictor values onto a multidimensional sphere. This has the effect of making all the samples the same distance from the center of the sphere. Mathematically, each sample is divided by its squared norm:

$$x_{ij}^* = \frac{x_{ij}}{\sum_{j=1}^{P} x_{ij}^2}.$$

Remark:

- It is important to center and scale the data prior to using this transformation, since the denominator is intended to measure the squared distance to the center of the distribution.
- The *spatial sign* transforms the predictors as a group. Removing predictors after applying the spatial sign may be problematic.

In R Usage: spatial.sign(X, center = TRUE, shape = TRUE, na.action = na.fail, ...)

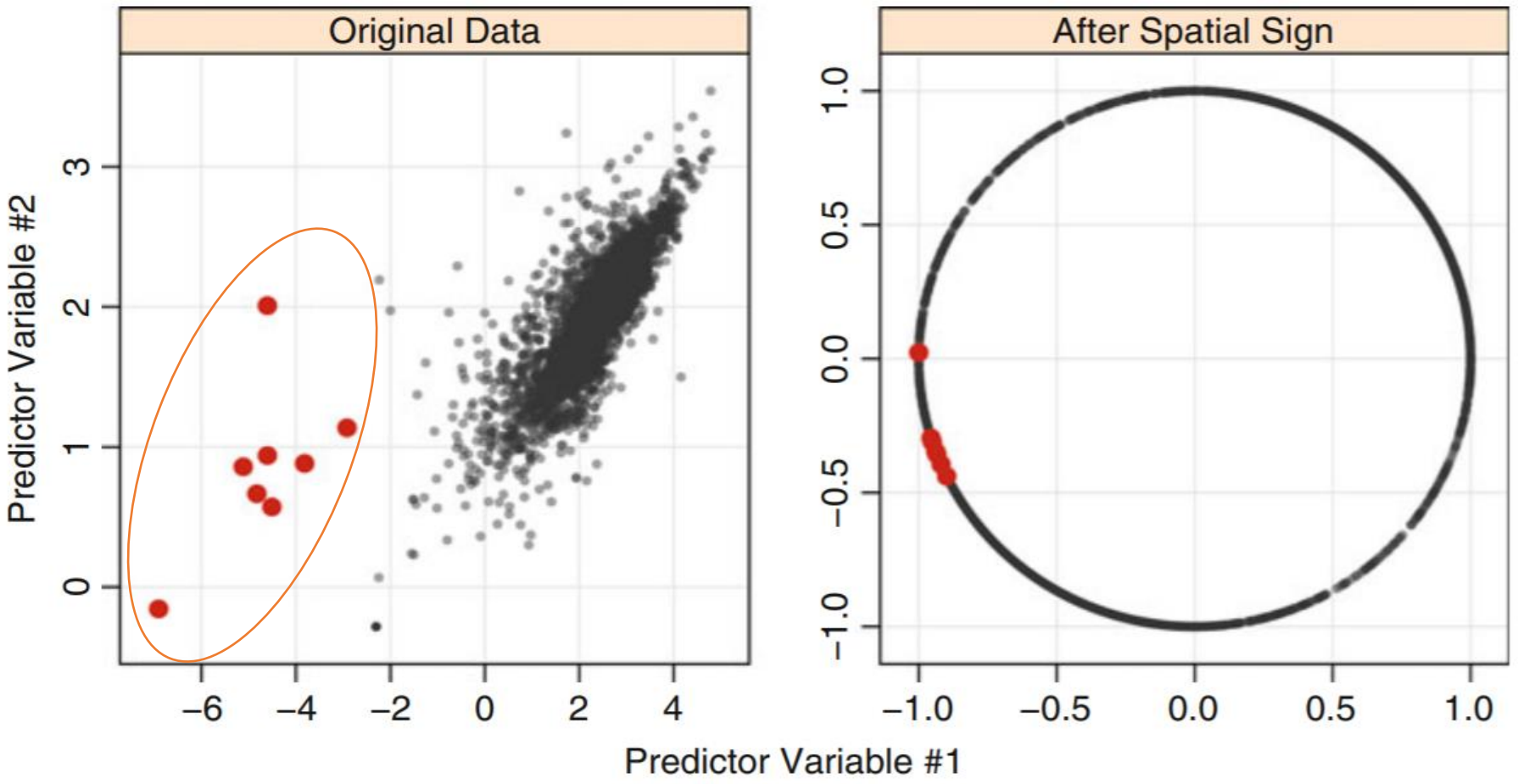


Fig. 3.4: *Left*: An illustrative example with a group of outlying data points. *Right*: When the original data are transformed, the results bring the outliers towards the majority of the data

R demonstration 3(1)

- R demonstration 3(1): Data scaling, transformation, and outliers

Data reduction (PCA)

- Principal component analysis (PCA) is an *unsupervised* technique that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (PCs).
- Mathematically, the *j*th PC can be written as

$$\mathrm{PC}_j = (a_{j1} \times \text{Predictor } 1) + (a_{j2} \times \text{Predictor } 2) + \cdots + (a_{jP} \times \text{Predictor } P)$$

where *P* is the number of predictors.

Remarks of PCA

- When predictors are on different scales and/or have skewed distributions, we need to
  - transform skewed data
  - center and scale data
- PCA is blind to the response (unsupervised learning)
- Use a *scree plot* to determine the number of principal components to retain.
  - The rule of thumb: the component number prior to the tapering off of variation is the maximal component that is retained.

R codes for PCA

```
library(caret)
## Use caret's preProcess function to transform for skewness
segPP <- preProcess(segTrainX, method = "BoxCox")

## Apply the transformations
segTrainTrans <- predict(segPP, segTrainX)
## R's prcomp is used to conduct PCA
pr <- prcomp(~AvgIntenCh1 + EntropyIntenCh1,
             data = segTrainTrans,  scale. = TRUE)

xyplot(AvgIntenCh1 ~ EntropyIntenCh1, data = segTrainTrans,
       groups = segTrain$Class,
       xlab = "Channel 1 Fiber Width",
       ylab = "Intensity Entropy Channel 1",
       auto.key = list(columns = 2), type = c("p", "g"),
       main = "Original Data", aspect = 1)
```

Check the plot when you specify

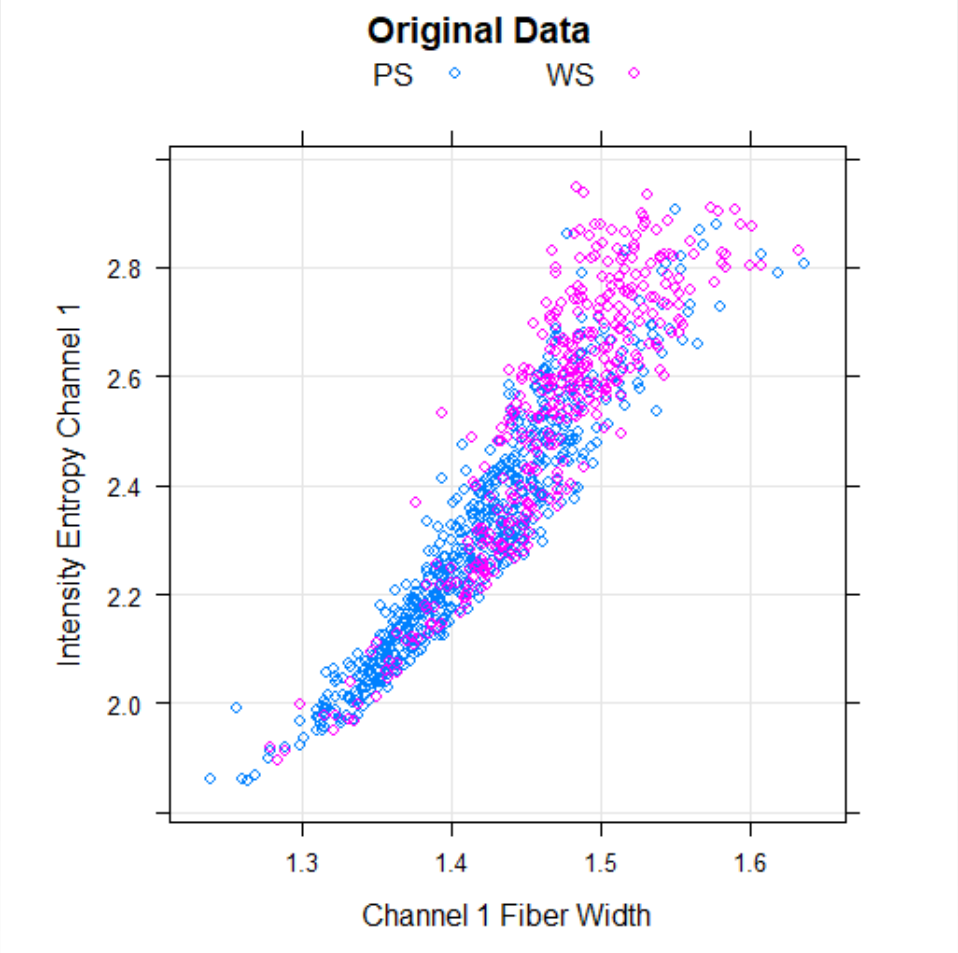
- auto.key =FALSE
- auto.key = TRUE

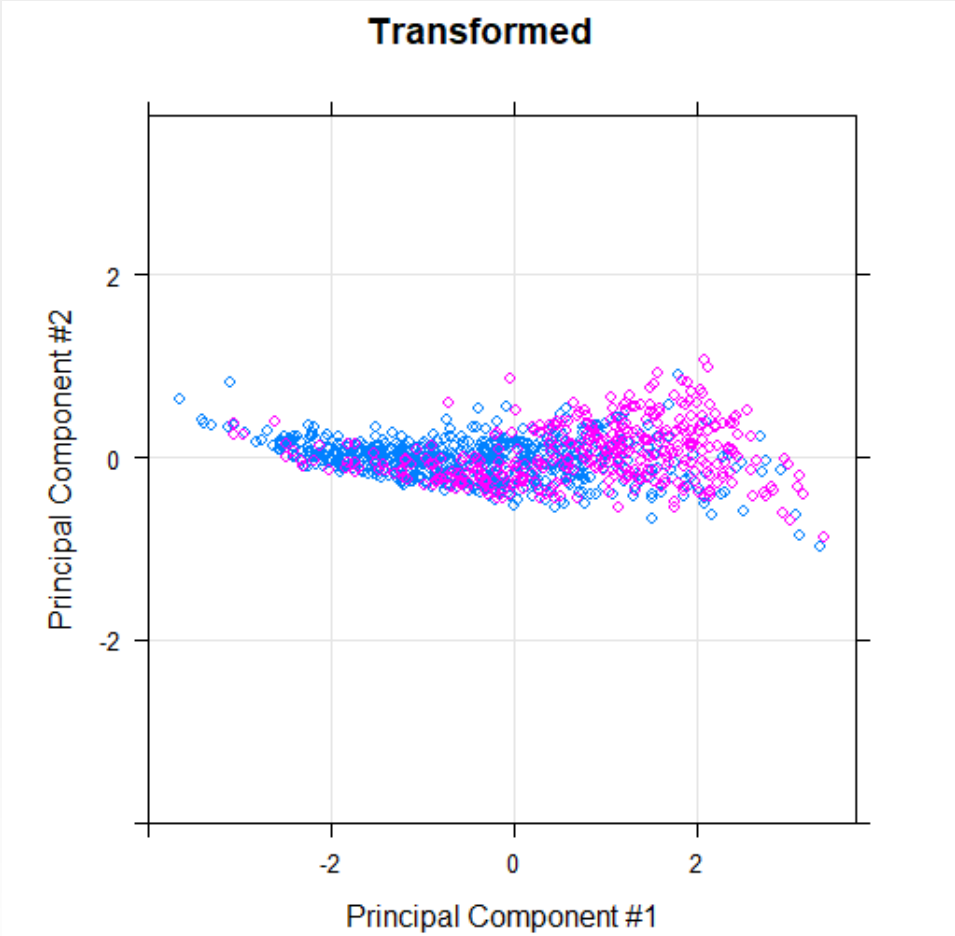
R codes for PCA

```
xyplot(PC2 ~ PC1,
      data = as.data.frame(pr$x),
      groups = segTrain$Class,
      xlab = "Principal Component #1",
      ylab = "Principal Component #2",
      main = "Transformed",
      xlim = extendrange(pr$x),
      ylim = extendrange(pr$x),
      type = c("p", "g"), aspect = 1)
```

For xyplot's type, you may refer to https://stat.ethz.ch/R-manual/R-devel/library/lattice/html/panel.xyplot.html
Extendrange: Extends a numerical range by a small percentage

PC transformation for the cell segmentation data

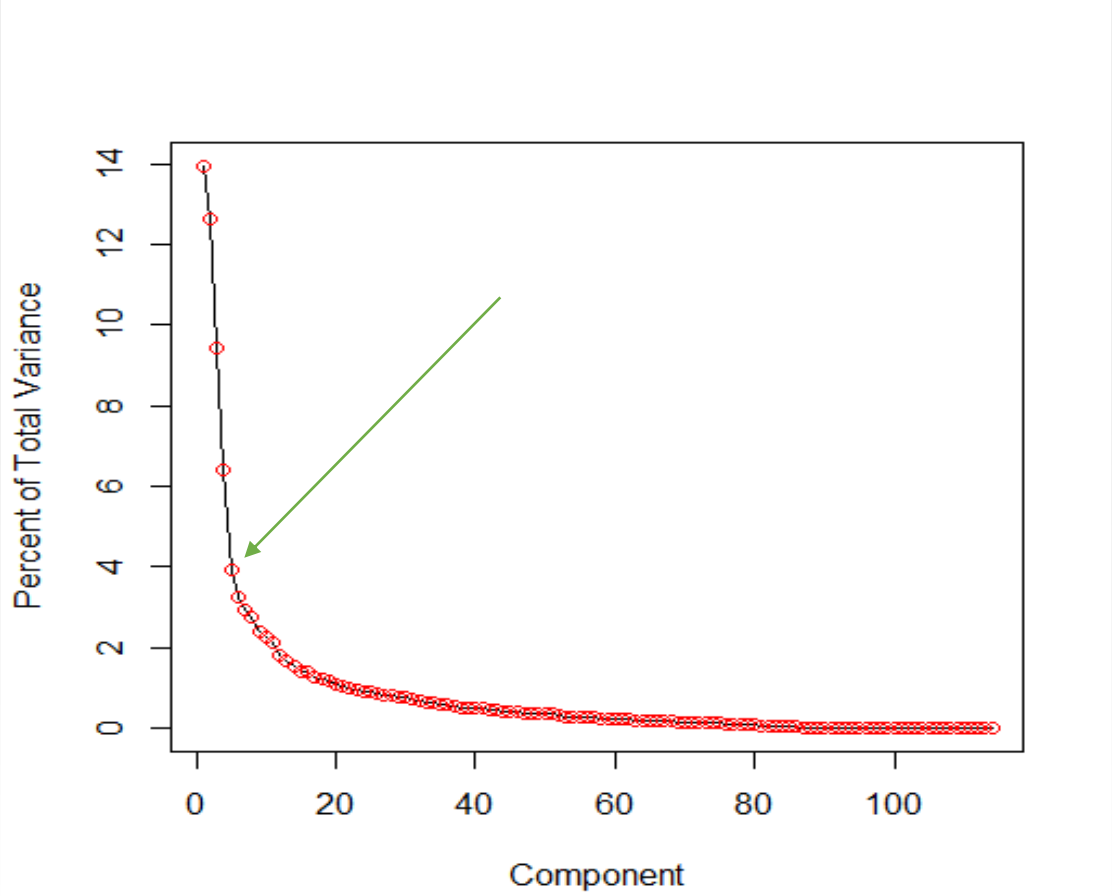




Fit PCA to entire set of segmentation data

```
## There are a few predictors with only a single value, so we remove these first
## (since PCA uses variances, which would be zero)

isZV <- apply(segTrainX, 2, function(x) length(unique(x)) == 1)  #identify the predictor with a single value
segTrainX <- segTrainX[, !isZV]

segPP <- preProcess(segTrainX, c("BoxCox", "center", "scale"))
segTrainTrans <- predict(segPP, segTrainX)

segPCA <- prcomp(segTrainTrans, center = TRUE, scale. = TRUE)

#Scree plot
PTotalVariance = (segPCA$sdev^2)/sum(segPCA$sdev^2)*100
ts.plot(PTotalVariance, xlab='Component', ylab='Percent of Total Variance')
points(PTotalVariance, col=2)
```

A “scree plot” where the percentage of the total variance explained by each component

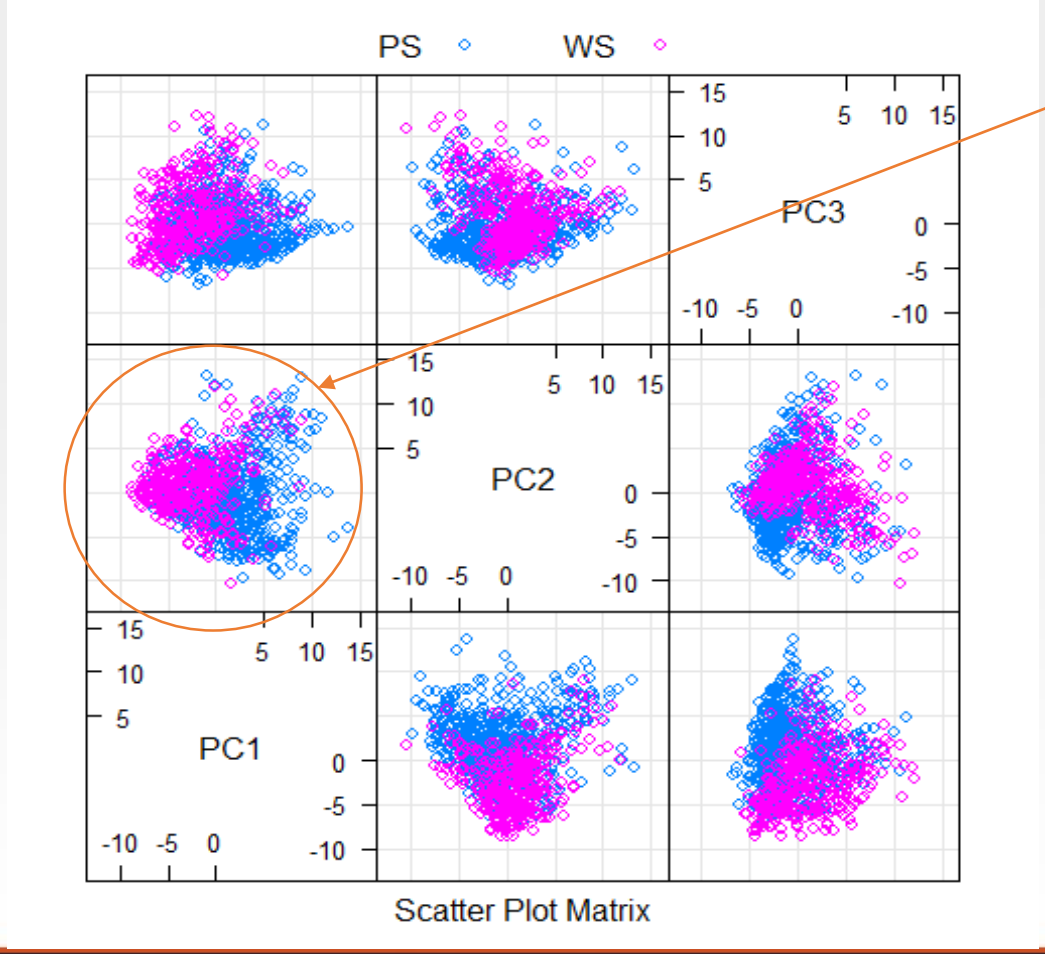


Plot a scatterplot matrix of the first three components

```
panelRange <- extendrange(segPCA$x[, 1:3])
splom(as.data.frame(segPCA$x[, 1:3]), #a data matrix consisting of three PCs
    groups = segTrainClass,
    type = c("p", "g"),
    auto.key = list(columns = 2),
    prepanel.limits = function(x) panelRange)
```

- *splom*: draw conditional scatter plot matrices and parallel coordinate plots.
- You may refer to http://127.0.0.1:29519/library/lattice/html/splom.html

A plot of the first three principal components for the cell segmentation data, colored by cell type



Some separation between the class when plotting the first and second components

Implement a series of transformations to multiple data

- The *caret* class *preProcess* has the ability to transform, center, scale, or impute values, as well as apply the spatial sign transformation and feature extraction.
- The *preProcess* function can be integrated into the *train* function for constructing predictive models.
- For example, to Box–Cox transform, center, and scale the data, then execute PCA for signal extraction.

R code

```
#Implement a series of transformations to multiple data
trans <- preProcess(segTrainX, method = c("BoxCox", "center", "scale", "pca"))
trans

# Apply the transformations:
transformed <- predict(trans, segTrainX)
# These values are different than the previous PCA components since
# they were transformed prior to PCA
head(transformed[, 1:6])
```

Implement a series of transformations to multiple data

```
> #Implement a series of transformations to multiple data
> trans <- preProcess(segTrainX, method = c("BoxCox", "center", "scale", "pca"))
> trans
Created from 1009 samples and 114 variables

Pre-processing:
  - Box-Cox transformation (47)
  - centered (114)
  - ignored (0)
  - principal component signal extraction (114)
  - scaled (114)

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 55 components to capture 95 percent of the variance
```

Implement a series of transformations to multiple data

```
> # Apply the transformations:
> transformed <- predict(trans, segTrainX)
> # These values are different than the previous PCA components since
> # they were transformed prior to PCA
> head(transformed[, 1:6])
          PC1        PC2        PC3        PC4        PC5        PC6
2   4.3560119 10.1198090  0.2870062 -0.8592671 -5.2499525  0.7331065
3  -0.5444723  1.6283869 -1.6533073 -3.8354232 -1.2555453  1.2558205
4   3.5512811 -0.5033273 -1.4852323 -1.0083940 -1.1571055 -3.2644940
12 -0.4318782 -1.7221518  0.9324858 -4.1005766 -2.4346531 -1.3982350
15  0.4826446 -0.6522716 -1.4394427 -4.6004477 -1.6368873 -0.7714049
16 -0.7522627  0.5447819 -0.3864860 -3.3520698 -0.1278511  1.7248114
```

Dealing with missing values

- In many cases, some predictors have no values for a given sample.
- Rubin (1976) classified missing data into three mechanisms, namely,
  - Missing completely at random (MCAR), where the missing process is completely independent from observed and missing quantities;
  - Missing at random (MAR), where the missing process depends on observed quantities, but not on missing quantities;
  - Nonignorable missing or not missing at random (NMAR), where the missing process may depend on both observed and missing quantities.
- The missing data mechanism could be ignored if the missingness is either MCAR or MAR. However, if the data are NMAR then the mechanism is not ignorable.

Dealing with missing values

- Understand *why* the values are missing
- Some methods
  - Remove the missing data
  - Some predictive models, such as tree-based techniques, can account for missing data
  - Imputing the missing data (Use MICE package in R)
  - Please refer to https://www.r-bloggers.com/imputing-missing-data-with-r-mice-package/

Removing predictors

- Few predictors means decreased computational time and complexity.
- Removing highly correlated predictors could promise the performance of the model and might lead to a more parsimonious and interpretable model.
- Removing predictors with degenerate distributions (e.g., near-zero variance predictor) could be a significant improvement in model performance and/or stability without the problematic variables.

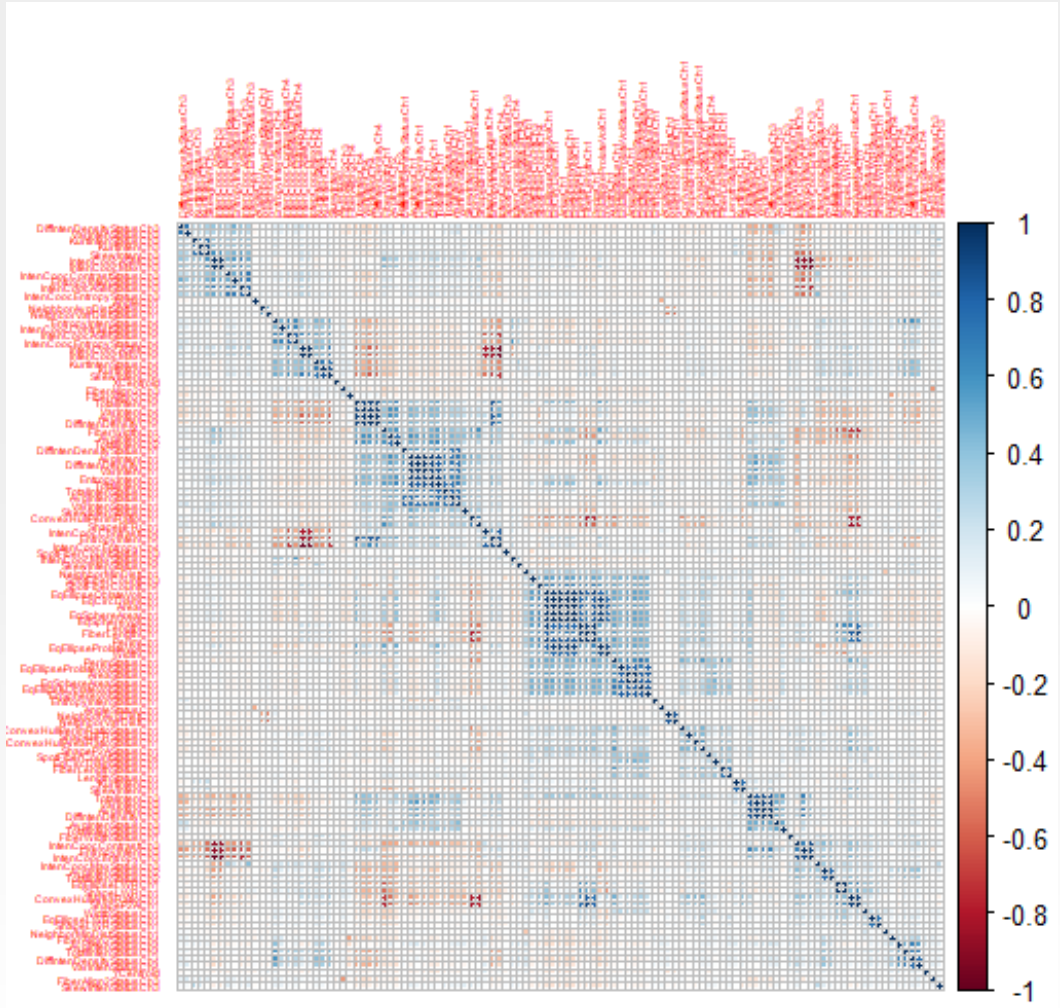
Algorithm for removing highly correlated predictors

1. Calculate the correlation matrix of the predictors.
2. Determine the two predictors associated with the largest absolute pairwise correlation (call them predictors A and B).
3. Determine the average correlation between A and the other variables. Do the same for predictor B.
4. If A has a larger average correlation, remove it; otherwise, remove predictor B.
5. Repeat Steps 2–4 until no absolute correlations are above the threshold.

R-codes for removing predictors

```
#Near zero variance predictor
nearZeroVar(segTrainTrans)
#Remove the near zero variance predictor
segTrainTrans1 = segTrainTrans[, -nearZeroVar(segTrainTrans)]
nearZeroVar(segTrainTrans1)

## To filter on correlations, we first get the correlation matrix for the
## predictor set
segCorr <- cor(segTrainTrans1)

library(corrplot)
corrplot(segCorr, order = "hclust", tl.cex = .35)
#tl.cexfor the size of text label (variable names).
## caret's findCorrelation function is used to identify columns to remove.
highCorr <- findCorrelation(segCorr, .75)
highCorr
```

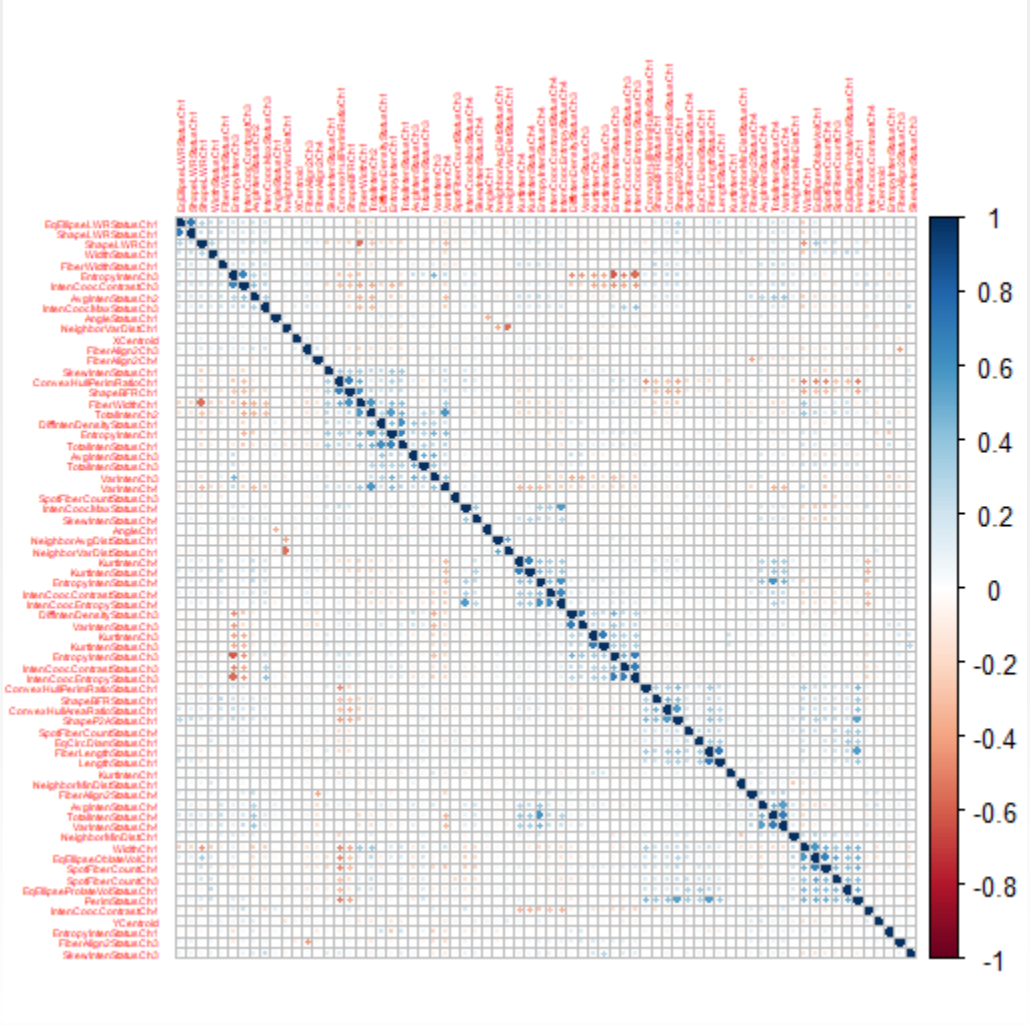
A visualization of the cell segmentation correlation matrix



R-codes for removing highly correlated predictors

```
## caret's findCorrelation function is used to identify columns to remove.
highCorr <- findCorrelation(segCorr, .75)
highCorr

#Removing highly correlated predictors
segCorr1 <- cor(segTrainTrans1[,-highCorr])
corrplot(segCorr1, order = "hclust", tl.cex = .35)
```

A visualization of the cell segmentation correlation matrix



Creating dummy variables

- When a predictor is categorical, such as gender, car type, it is common to decompose the predictor into a set of more specific variables.
- The categories are re-encoded into smaller bits of information called "dummy variables."
- Usually, each category get its own *dummy variable* that is a *zero/one indicator* for that group.

R-codes for 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)
levels(carSubset$Type)

simpleMod <- dummyVars(~Mileage + Type,
                       data = carSubset,
                       ## Remove the variable name from the
                       ## column name
                       levelsOnly = TRUE)
simpleMod
```

dummyVars: https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/dummyVars

R-codes for creating dummy variables with interaction

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

Creating dummy variables

```
> predict(withInteraction, head(carSubset))
    Mileage convertible coupe hatchback sedan wagon Mileage:Typeconvertible
107   24318           0     1         0     0     0                       0
181   21132           0     0         0     1     0                       0
30    21545           0     0         0     0     1                       0
75    30502           1     0         0     0     0                   30502
326   22152           0     1         0     0     0                       0
71     5239           1     0         0     0     0                    5239
    Mileage:Typecoupe Mileage:Typehatchback Mileage:Typesedan Mileage:Typewagon
107             24318                     0                0                0
181                 0                     0            21132                0
30                  0                     0                0            21545
75                  0                     0                0                0
326             22152                     0                0                0
71                  0                     0                0                0
```

Should we bin predictors?

- While there are recommended techniques for pre-processing data, there are also methods to avoid.
- One common approach to simplifying a data set is to take a numeric predictor and pre-categorize or “bin” it into two or more groups prior to data analysis.

SIRS

- Bone et al. (1992) define a set of clinical symptoms to diagnose Systemic Inflammatory Response Syndrome (SIRS). SIRS can occur after a person is subjected to some sort of physical trauma (e.g., car crash). A simplified version of the clinical criteria for SIRS are:
  - Temperature less than 36 ° C or greater than 38°C.
  - Heart rate greater than 90 beats per minute.
  - Respiratory rate greater than 20 breaths per minute.
  - White blood cell count less than 4,000 cells/mm^3 or greater than 12,000 cells/mm3.
- A person who shows two or more of these criteria would be diagnosed as having SIRS.

SIRS

The perceived *advantages* to this approach are:

- The ability to make seemingly simple statements, either for sake of having a simple decision rule (as in the SIRS example) or the belief that there will be a simple interpretation of the model.
- The modeler does not have to know the exact relationship between the predictors and the outcome.
- A higher response rate for survey questions where the choices are binned. For example, asking the date of a person's last tetanus shot is likely to have fewer responses than asking for a range (e.g., in the last 2 years, in the last 4 years).

SIRS

The perceived issues to this approach are:

- There can be a significant loss of performance in the model
- There is a loss of precision in the predictions when the predictors are categorized.
- The research has shown (Austin and Brunner 2004) that categorizing predictors can lead to a high rate of false positives (i.e., noise predictors determined to be informative).

Remark: The predictive models that are most powerful are usually the least interpretable.

R demonstration 3(2)

- R demonstration 3(2): PCA, creating dummy variables, removing and binning predictors.

Exercise 2