Exercises for
Applied Predictive Modeling
Chapter 3 — Data Pre–Processing

Max Kuhn, Kjell Johnson

Version 1
January 8, 2015

The solutions in this file use several R packages not used in the text. To install all of the packages needed for this document, use:

```
> install.packages(c("AppliedPredictiveModeling", "car", "caret", "corrplot",
+                    "e1071", "mlbench", "subselect", "reshape2", "vcd"))
```

One note about these exercises: the type and amount tot pre–processing is dependent on the model being used. The results shown here are appropriate for models that have significant pre–processing requirements of the predictor variables.

Exercise 1

The UC Irvine Machine Learning Repository[1] contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are 9 predictors, including the refractive index and percentages of 8 elements: Na, Mg, Al, Si, K, Ca, Ba and Fe.

The data can be accessed via:

```
> library(mlbench)
> data(Glass)
> str(Glass)
```

```
'data.frame': 214 obs. of  10 variables:
 $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
```

[1] http://archive.ics.uci.edu/ml/index.html

```
$ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
$ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
$ Si  : num  71.8 72.7 73 72.6 73.1 ...
$ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
$ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
$ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
$ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
$ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

(a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

(b) Does there appear to be any outliers in the data? Are predictors skewed?

(c) Are there any relevant transformations of one or more predictors that might improve the classification model?

Solutions

To examine the predictor distributions, individual histograms or density plots are useful. To look at them in a single plot, we will first "melt" the data into a "long" format so that predictors are not in separate columns:

```
> library(reshape2)
> meltedGlass <- melt(Glass, id.vars = "Type")
> head(meltedGlass)
```

```
  Type variable value
1    1       RI 1.521
2    1       RI 1.518
3    1       RI 1.516
4    1       RI 1.518
5    1       RI 1.517
6    1       RI 1.516
```

Now, we can use the lattice function `densityplot` to look at each predictor. The code used to create Figure 1 is:

```
> library(lattice)
> densityplot(~value|variable,
+             data = meltedGlass,
+             ## Adjust each axis so that the measurement scale is
+             ## different for each panel
+             scales = list(x = list(relation = "free"),
```

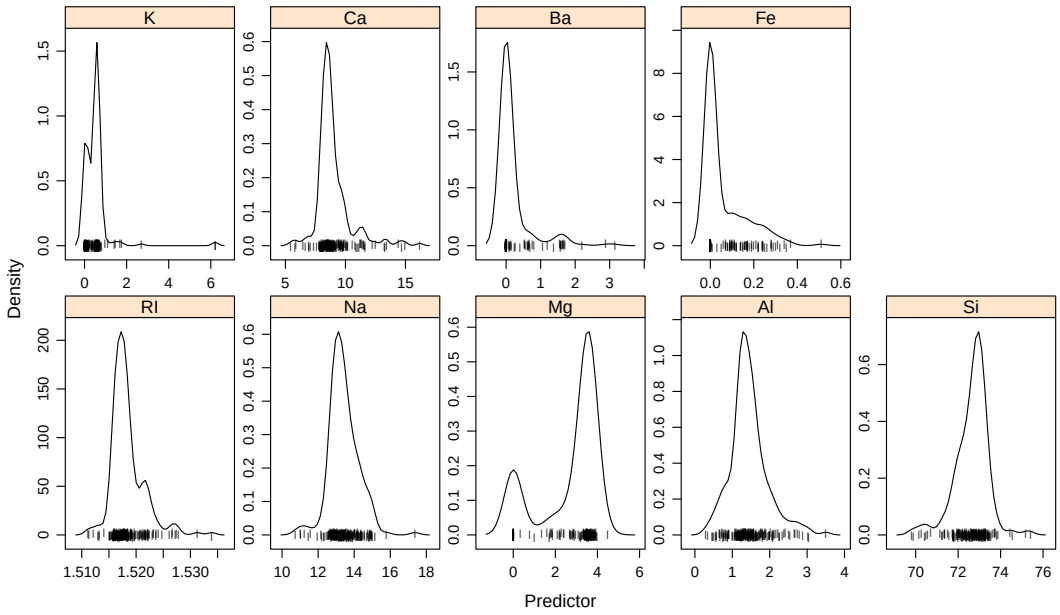

Figure 1: Density plots of each of the predictors in the original Glass data set. The points along the x–axis show the values of the individual samples.

```
+                       y = list(relation = "free")),
+             ## 'adjust' smooths the curve out
+             adjust = 1.25,
+             ## change the symbol on the rug for each data point
+             pch = "|",
+             xlab = "Predictor")
```

From Figure 1, we can see that `K` and `Mg` appear to have possible second modes around zero and that several predictors (`Ca`, `Ba`, `Fe` and `RI`) show signs of skewness. There may be one or two outliers in `K`, but they could simply be due to natueral skewness. Also, predictors `Ca`, `RI`, `Na` and `Si` have concentrations of samples in the middle of the scale and a small number of data points at the edges of the distribution. This characteristic is indicative of a "heavy–tailed" distribution.

A scatterplot matrix can also be helpful to visualize a data set of this size (i.e. 9 predictor variables). The lattice function `splom` was used to create the scatterplot matrix in Figure 2. This visualization highlights several other important characteristics of this data:

1. The measurements of some glass types, specifically `Fe`, `Ba`, `K` and `Mg`, are zero. This creates a "mixture distribution" of points; one distribution consists of glass types containing the element in question whereas the other does not. This finding implies that the samples in these distributions may not behave in the same manner.

2. Most predictors are uncorrelated with the exception of pairs `Ca`/`RI` and `Ca`/`Na`.

3. Many of the pair–wise combinations have very non–standard distributions (i.e. heavy tails or mixtures of distributions).

4. It is difficult to tell if the extreme point in the `K` data is an outlier or just a artifact of a skewed distribution that has not been sampled enough. In either case, this should be accounted for through the modeling, preferably by using models that are resistant to outliers.

Would transformations help these data? Based on our findings above, we need to investigate transformations of individual predictors that will resolve skewness and/or outliers (e.g. the spatial sign transformation).

For skewness, first note that several predictors have values of zero. This excludes transformations such as the log transformation or the Box–Cox family of transformations. When we are faced with predictors containing zero values, the Yeo–Johnson family of transformations can be helpful[2] (Yeo & Johnson 2000). This family of transformations is very similar to the Box–Cox transformation, but can handle zero or negative predictor values. The transformation can be estimated using caret's `preProcess` function:

```
> yjTrans <- preProcess(Glass[, -10], method = "YeoJohnson")
> yjData <- predict(yjTrans, newdata= Glass[, -10])
> melted <- melt(yjData)
```

The resulting density plots are shown in Figure 3. The only substantive change relative to the original distributions is that a second mode was induced for predictors `Ba` and `Fe`. Given these results, this transformation did not seem to improve the data (in terms of skewness).

Next, we will apply the spatial sign transformation to attempt to mitigate outliers. For this data, we first center and scale the data, then apply the transformation:

```
> centerScale <- preProcess(Glass[, -10], method = c("center", "scale"))
> csData <- predict(centerScale, newdata = Glass[, -10])
> ssData <- spatialSign(csData)
> splom(~ssData, pch = 16, col = rgb(.2, .2, .2, .4), cex = .7)
```

Figure 4 shows the results. Many of the possible outliers have been contracted into the mainstream of the data. This transformation did result in at least one new pattern: the samples with zero values for both `Fe` and `B` are now projected onto a straight line in these two dimensions.

While we were unable to resolve skewness in this data via transformations, we were able to minimize the number of unusually extreme observations. Note that attempts to pre–process data to resolve predictor distribution problems are not always successful. Our best efforts in pre–processing may not yield highly desirable transformed values. Under these kinds of circumstances, we will need to use models that are not unduly affected by skewed distributions (e.g. tree–based methods).

[2]We were not aware of this set of transformation at the time when the text was written.

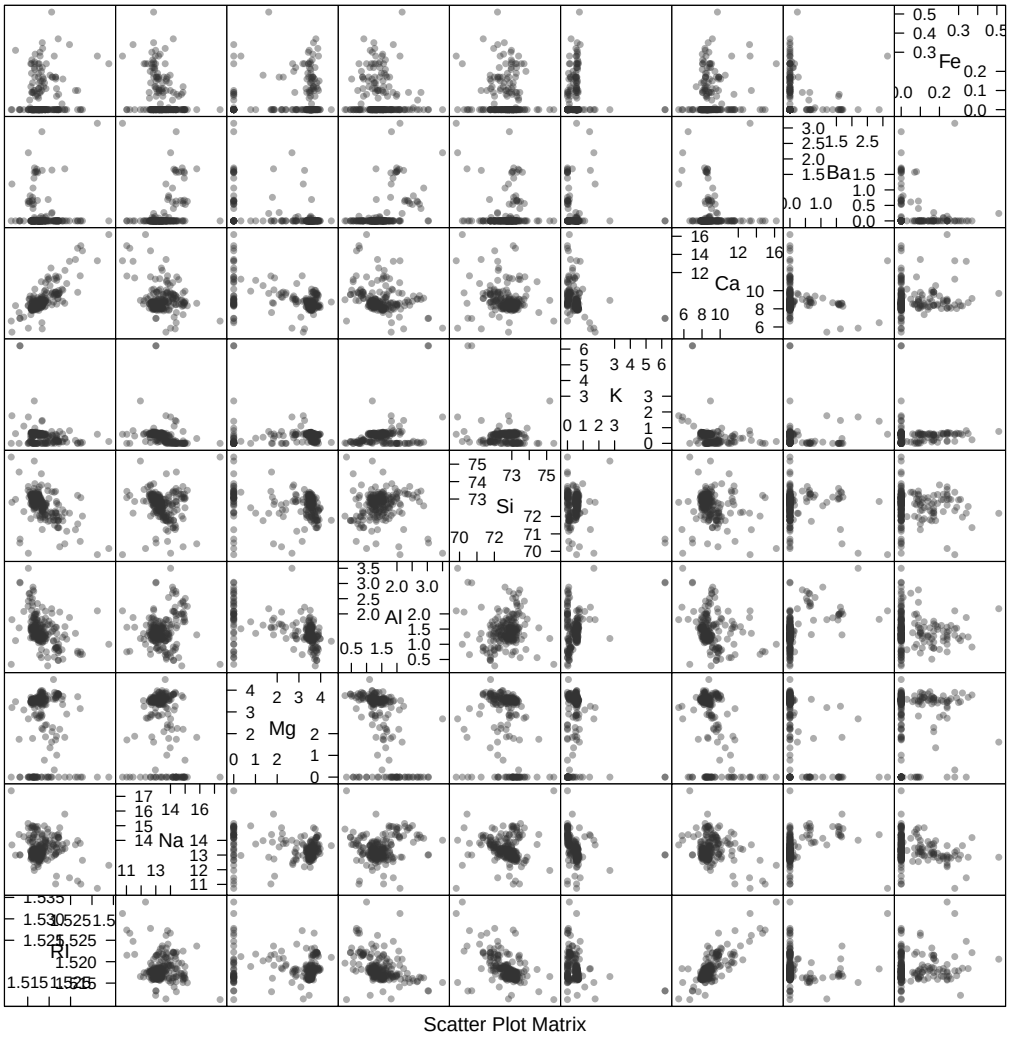

Figure 2: A scatterplot matrix of the predictors in the original Glass data set.

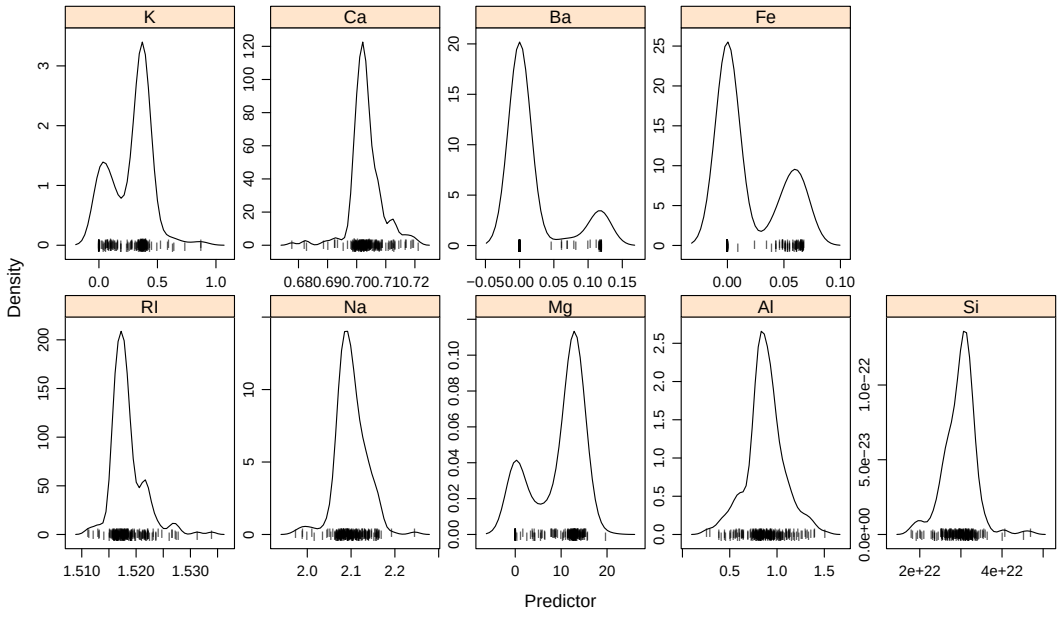

Figure 3: Density plots of the Glass predictors after a Yeo–Johnson transformation.

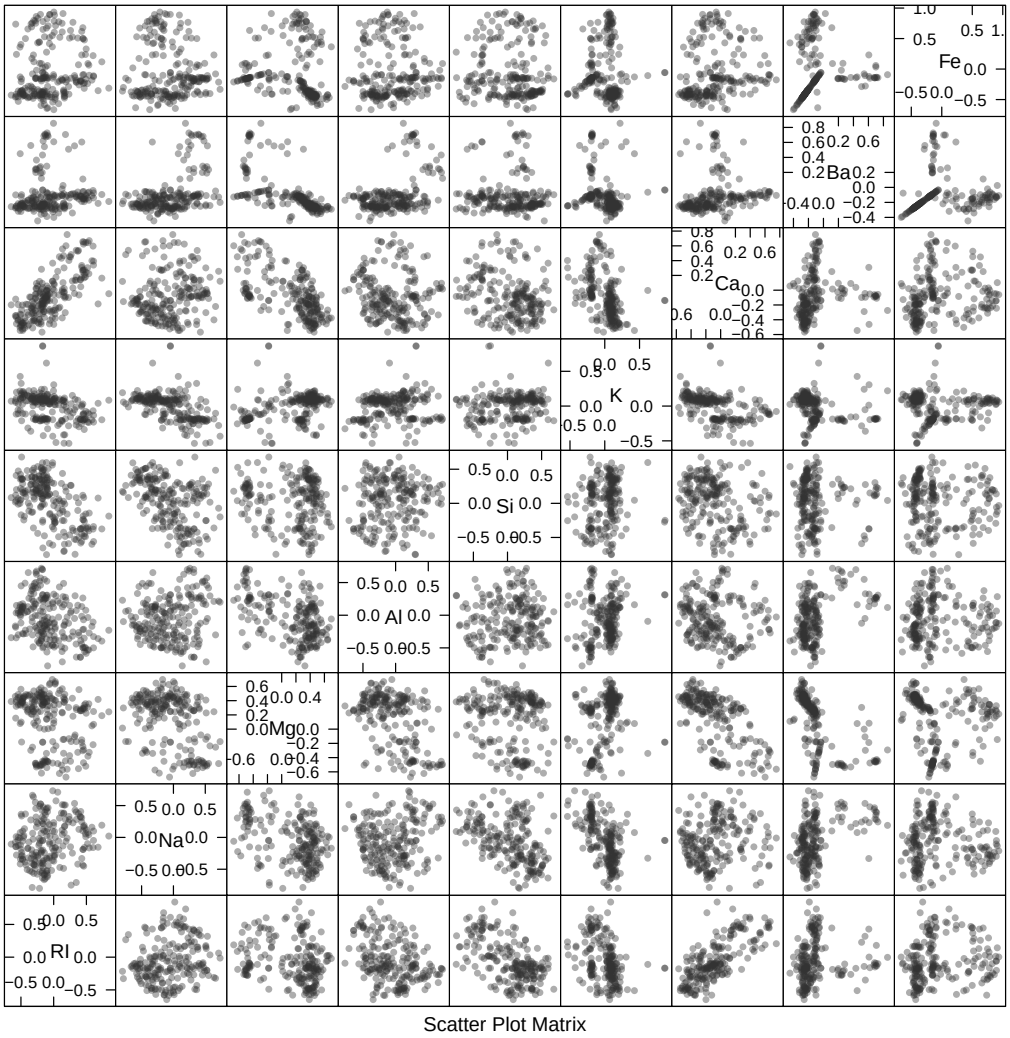

Figure 4: A scatterplot matrix of the Glass data after the spatial sign transformation.

Exercise 2

The Soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g. temperature, precipitation) and plant conditions (e.g. left spots, mold growth). The outcome labels consist of 19 distinct classes.

The data can be loaded via:

```
> library(mlbench)
> data(Soybean)
> ## See ?Soybean for details
```

(a) Investigate the frequency distributions for the categorical predictors. Are the distributions likely to cause issues for models.

(b) Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

(c) Develop a strategy for dealing with the missing data, either by eliminating predictors or imputation.

Solutions

The contents of the Soybean data frame are:

```
> str(Soybean)
```

```
'data.frame': 683 obs. of  36 variables:
 $ Class          : Factor w/ 19 levels "2-4-d-injury",..: 11 11 11 11 11 11 11 11 11 11 ...
 $ date           : Factor w/ 7 levels "0","1","2","3",..: 7 5 4 4 7 6 6 5 7 5 ...
 $ plant.stand    : Ord.factor w/ 2 levels "0"<"1": 1 1 1 1 1 1 1 1 1 1 ...
 $ precip         : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
 $ temp           : Ord.factor w/ 3 levels "0"<"1"<"2": 2 2 2 2 2 2 2 2 2 2 ...
 $ hail           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ crop.hist      : Factor w/ 4 levels "0","1","2","3": 2 3 2 2 3 4 3 2 4 3 ...
 $ area.dam       : Factor w/ 4 levels "0","1","2","3": 2 1 1 1 1 1 1 1 1 1 ...
 $ sever          : Factor w/ 3 levels "0","1","2": 2 3 3 3 2 2 2 2 2 3 ...
 $ seed.tmt       : Factor w/ 3 levels "0","1","2": 1 2 2 1 1 1 2 1 2 1 ...
 $ germ           : Ord.factor w/ 3 levels "0"<"1"<"2": 1 2 3 2 3 2 1 3 2 3 ...
 $ plant.growth   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ leaves         : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ leaf.halo      : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ leaf.marg      : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...
 $ leaf.size      : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
```

```
 $ leaf.shread    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ leaf.malf      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ leaf.mild      : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ stem           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ lodging        : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 1 1 1 ...
 $ stem.cankers   : Factor w/ 4 levels "0","1","2","3": 4 4 4 4 4 4 4 4 4 4 ...
 $ canker.lesion  : Factor w/ 4 levels "0","1","2","3": 2 2 1 1 2 1 2 2 2 2 ...
 $ fruiting.bodies: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ ext.decay      : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...
 $ mycelium       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ int.discolor   : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ sclerotia      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ fruit.pods     : Factor w/ 4 levels "0","1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ fruit.spots    : Factor w/ 4 levels "0","1","2","4": 4 4 4 4 4 4 4 4 4 4 ...
 $ seed           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ mold.growth    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ seed.discolor  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ seed.size      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ shriveling     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ roots          : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

When we look closely at this output, we see that the factor levels of some predictors are not informative. For example, the `temp` column contains integer values. These values correspond the relative temperature: below average, average and above average. For our understanding of the data, it would be very helpful to change these integer values to their actual values. This change can be done using the `recode` function in the `car` package. We can also make missing values an independent category so that we see them in tables:

```
> Soybean2 <- Soybean
> table(Soybean2$temp, useNA = "always")


   0    1    2 <NA>
  80  374  199   30
```

```
> library(car)
> Soybean2$temp <- recode(Soybean2$temp,
+                         "0 = 'low'; 1 = 'norm'; 2 = 'high'; NA = 'missing'",
+                         levels = c("low", "norm", "high", "missing"))
> table(Soybean2$temp)


    low    norm    high missing
     80     374     199      30
```

For this part of the solution to this problem, we will look at the relationship between the months, temperature and precipitation. To explore these relationships, we need to recode months and

precipitation:

```
> table(Soybean2$date, useNA = "always")


   0    1    2    3    4    5    6 <NA>
  26   75   93  118  131  149   90    1

> Soybean2$date <- recode(Soybean2$date,
+                         "0 ='apr';1='may';2='june';3='july';4='aug';5='sept';6='oct';NA = 'missing'",
+                         levels = c("apr", "may", "june", "july", "aug", "sept", "missing"))
> table(Soybean2$date)


    apr     may    june    july     aug    sept missing
     26      75      93     118     131     149       1

> table(Soybean2$precip, useNA = "always")


   0    1    2 <NA>
  74  112  459   38

> Soybean2$precip <- recode(Soybean2$precip,
+                         "0 = 'low'; 1 = 'norm'; 2 = 'high'; NA = 'missing'",
+                         levels = c("low", "norm", "high", "missing"))
> table(Soybean2$precip)


    low    norm    high missing
     74     112     459      38
```

To start, let's look at the date predictor. Are the months represented equally? From the table above, we can see that June through September have the most data and that there is a single missing value. For precipitation (ironically) most of the data are above average. In addition, the temperature and precipitation columns have missing value rates of about 5%.

Like the previous problems, we should examine the pair-wise or joint distributions of these predictors. Joint distributions of factor predictors are often displayed in a contingency table. There are also several ways that these distributions can be displayed in a graph. The `mosaic` function in the vcd package (Meyer, Zeileis & Hornik 2006) and the `barchart` function in the lattice package are two options. What does the joint distribution of temperature and month look like? First, we will use a mosaic plot:

```
> library(vcd)
> ## mosaic() can table a table or a formula:
> mosaic(~date + temp, data = Soybean2)
```

Alternatively, a bar chart can also be used:

```
> barchart(table(Soybean2$date, Soybean2$temp),
+          auto.key = list(columns = 4, title = "temperature"))
```

The results are shown in Figure 5. Note that in the bar chart, the bars are not cumulative (i.e. missing values are not the most frequent). Here we see which months are the most frequent. Additionally, we see that average temperatures are the most frequent category within each month, although high temperatures are also very likely in September. Missing values are most likely in July. One useful option to `barchart` is `stack` to create stacked bars.

To investigate higher–order relationships, predictors can be added to the table or formula to create more complex visualizations (e.g. panels in the lattice plots, etc).

What does the distribution look like per response class for the missing data? If we look at the frequency of *any* missing predictor value per class, the results show that some classes are more problematic than others:

```
> table(Soybean$Class, complete.cases(Soybean))
```

```
                           FALSE TRUE
  2-4-d-injury                16    0
  alternarialeaf-spot          0   91
  anthracnose                  0   44
  bacterial-blight             0   20
  bacterial-pustule            0   20
  brown-spot                   0   92
  brown-stem-rot               0   44
  charcoal-rot                 0   20
  cyst-nematode               14    0
  diaporthe-pod-&-stem-blight 15    0
  diaporthe-stem-canker        0   20
  downy-mildew                 0   20
  frog-eye-leaf-spot           0   91
  herbicide-injury             8    0
  phyllosticta-leaf-spot       0   20
  phytophthora-rot            68   20
  powdery-mildew               0   20
  purple-seed-stain            0   20
  rhizoctonia-root-rot         0   20
```

```
> hasMissing <- unlist(lapply(Soybean, function(x) any(is.na(x))))
> hasMissing <- names(hasMissing)[hasMissing]
> head(hasMissing)
```

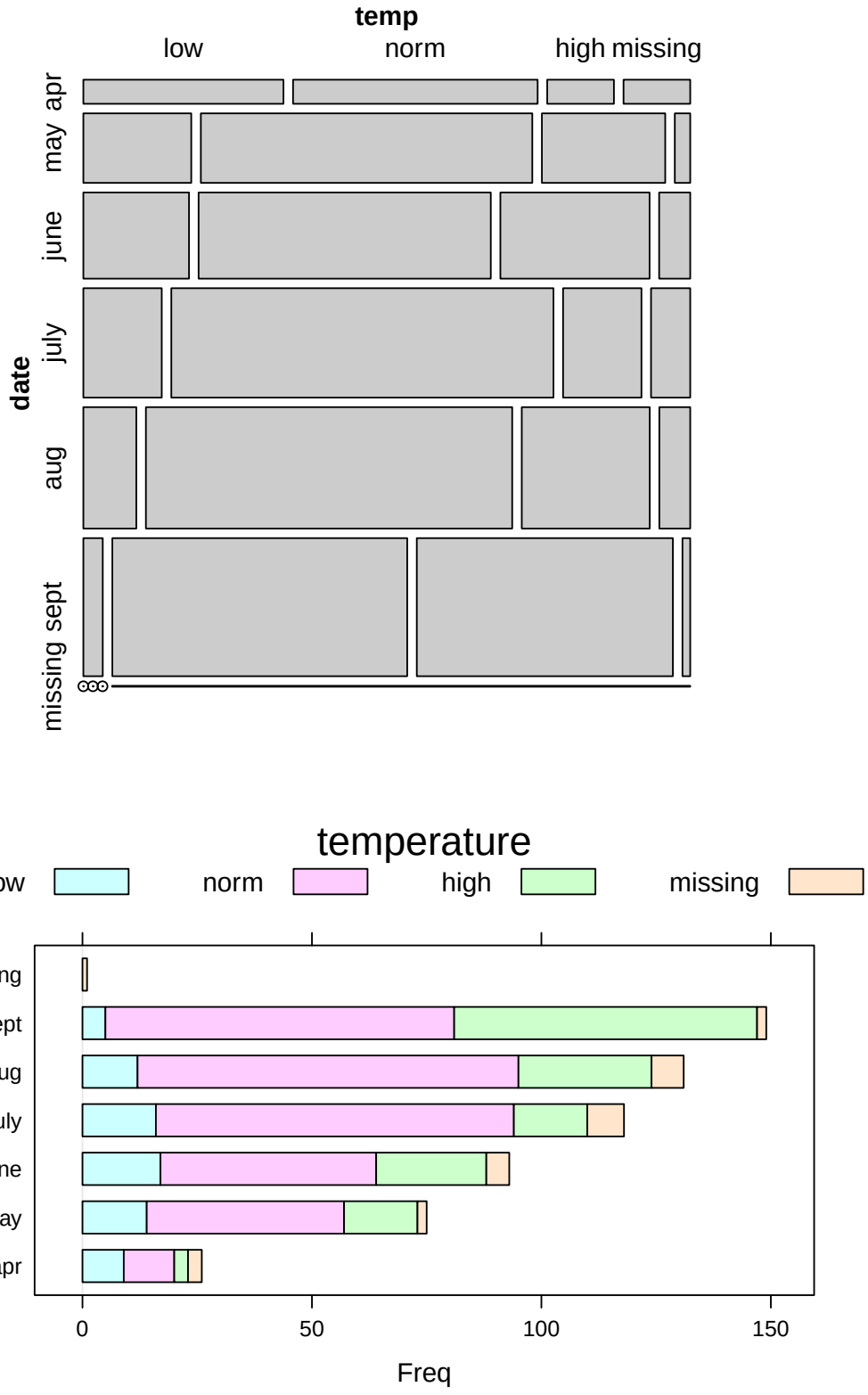

Figure 5: Mosaic and bar charts of the joint frequency distribution for month and temperature.

```
[1] "date"        "plant.stand" "precip"      "temp"        "hail"
[6] "crop.hist"
```

There are several classes where all of the samples have at least one missing predictor value. Are these concentrated in a single predictor that we could remove? We can get the percentage of missing values for each predictor by class using the following syntax:

```
> byPredByClass <- apply(Soybean[, hasMissing], 2,
+                        function(x, y) {
+                          tab <- table(is.na(x), y)
+                          tab[2,]/apply(tab, 2, sum)
+                        },
+                        y = Soybean$Class)
>
> ## The columns are predictors and the rows are classes. Let's eliminate
> ## any rows and columns with no missing values
>
> byPredByClass <- byPredByClass[apply(byPredByClass, 1, sum) > 0,]
> byPredByClass <- byPredByClass[, apply(byPredByClass, 2, sum) > 0]
>
> ## now print:
> t(byPredByClass)
```

```
               2-4-d-injury cyst-nematode diaporthe-pod-&-stem-blight
date                 0.0625             0                        0.0
plant.stand          1.0000             1                        0.4
precip               1.0000             1                        0.0
temp                 1.0000             1                        0.0
hail                 1.0000             1                        1.0
crop.hist            1.0000             0                        0.0
area.dam             0.0625             0                        0.0
sever                1.0000             1                        1.0
seed.tmt             1.0000             1                        1.0
germ                 1.0000             1                        0.4
plant.growth         1.0000             0                        0.0
leaf.halo            0.0000             1                        1.0
leaf.marg            0.0000             1                        1.0
leaf.size            0.0000             1                        1.0
leaf.shread          1.0000             1                        1.0
leaf.malf            0.0000             1                        1.0
leaf.mild            1.0000             1                        1.0
stem                 1.0000             0                        0.0
lodging              1.0000             1                        1.0
stem.cankers         1.0000             1                        0.0
canker.lesion        1.0000             1                        0.0
fruiting.bodies      1.0000             1                        0.0
ext.decay            1.0000             1                        0.0
mycelium             1.0000             1                        0.0
int.discolor         1.0000             1                        0.0
sclerotia            1.0000             1                        0.0
fruit.pods           1.0000             0                        0.0
fruit.spots          1.0000             1                        0.0
seed                 1.0000             0                        0.0
mold.growth          1.0000             0                        0.0
seed.discolor        1.0000             1                        0.0
seed.size            1.0000             0                        0.0
shriveling           1.0000             1                        0.0
roots                1.0000             0                        1.0
               herbicide-injury phytophthora-rot
date                          0           0.0000
```

```
plant.stand                 0          0.0000
precip                      1          0.0000
temp                        0          0.0000
hail                        1          0.7727
crop.hist                   0          0.0000
area.dam                    0          0.0000
sever                       1          0.7727
seed.tmt                    1          0.7727
germ                        1          0.7727
plant.growth                0          0.0000
leaf.halo                   0          0.6250
leaf.marg                   0          0.6250
leaf.size                   0          0.6250
leaf.shread                 0          0.6250
leaf.malf                   0          0.6250
leaf.mild                   1          0.6250
stem                        0          0.0000
lodging                     1          0.7727
stem.cankers                1          0.0000
canker.lesion               1          0.0000
fruiting.bodies             1          0.7727
ext.decay                   1          0.0000
mycelium                    1          0.0000
int.discolor                1          0.0000
sclerotia                   1          0.0000
fruit.pods                  0          0.7727
fruit.spots                 1          0.7727
seed                        1          0.7727
mold.growth                 1          0.7727
seed.discolor               1          0.7727
seed.size                   1          0.7727
shriveling                  1          0.7727
roots                       0          0.0000
```

From this output, we see that there are many predictors completely missing for the `2-4-d-injury`, `cyst-nematode` and `herbicide-injury` classes. The `phytophthora-rot` class has a high rate of missing data across many predictors and the `diaporthe-pod-&-stem-blight` has a more moderate pattern of missing data.

One approach to handling missing data is to use an imputation technique. However, it is unlikely that imputation will help since almost 100% of the predictor values will need to be imputed in a few cases. We could encode the missing as another level or eliminate the classes associated with the high rate of missing values from the data altogether.

How would the frequencies of the predictor values affect the modeling process? If we are using a model that is sensitive to sparsity then the low rate of some of the factor levels might be an issue. We can convert the factors to a set of dummy variables and see how good or bad the sparsity is.

```
> ## Some of the factors are ordinal. First convert them to unordered factors so
> ## that we get a set of binary indicators.
>
> orderedVars <- unlist(lapply(Soybean, is.ordered))
> orderedVars <- names(orderedVars)[orderedVars]
>
> ## Let's bypass the problem of missing data by removing the offending classes
>
```

```
> completeClasses <- as.character(unique(Soybean$Class[complete.cases(Soybean)]))
> Soybean3 <- subset(Soybean, Class %in% completeClasses)
> for(i in orderedVars) Soybean3[, i] <- factor(as.character(Soybean3[, i]))
>
> ## Use dummyVars to generate the binary predictors...
> dummyInfo <- dummyVars(Class ~ ., data = Soybean3)
> dummies <- predict(dummyInfo, Soybean3)
>
> ## ... then nearZeroVar to figure out which should be removed.
> predDistInfo <- nearZeroVar(dummies, saveMetrics = TRUE)
> head(predDistInfo)

       freqRatio percentUnique zeroVar   nzv
date.0    30.500        0.3175   FALSE  TRUE
date.1     8.265        0.3175   FALSE FALSE
date.2     6.326        0.3175   FALSE FALSE
date.3     4.727        0.3175   FALSE FALSE
date.4     4.081        0.3175   FALSE FALSE
date.5     3.500        0.3175   FALSE FALSE

> ## The number and percentage of predictors to remove:
> sum(predDistInfo$nzv)

[1] 16

> mean(predDistInfo$nzv)

[1] 0.1616
```

So if we wanted to remove sparse and unbalanced predictors, 16.2% of the dummy variables would be eliminated. One way around this is to use models that are not sensitive to this characteristic, such as tree– or rule–based models, or naïve Bayes.

Exercise 3

Chapter 5 introduces Quantitative Structure–Activity Relationship (QSAR) modeling where the characteristics of a chemical compound are used to predict other chemical properties. The caret package contains such a data set from (Mente & Lombardo 2005). Here, where the ability of a chemical to permeate the blood–brain barrier was experimentally determined for 208 compounds. 134 predictors were measured for each compound.

(a) Start R and use these commands to load the data:

```
> library(caret)
> data(BloodBrain)
> # use ?BloodBrain to see more details
```

The numeric outcome is contained in the vector `logBBB` while the predictors are in the data frame `bbbDescr`.

(b) Do any of the individual predictors have degenerate distributions?

(c) Generally speaking, are there strong relationships between the predictor data? If so, how could correlations in the predictor set be reduced? Does this have a dramatic effect on the number of predictors available for modeling?

Solutions

For these data, the first assessment looks for sparse and unbalanced predictors. The caret `nearZeroVar` function is used again but this time with the `saveMetrics` options that retains information about each predictor:

```
> ncol(bbbDescr)
```

```
[1] 134
```

```
> predictorInfo <- nearZeroVar(bbbDescr, saveMetrics = TRUE)
> head(predictorInfo)
```

```
         freqRatio percentUnique zeroVar   nzv
tpsa         2.143       61.5385   FALSE FALSE
nbasic       1.737        0.9615   FALSE FALSE
negative   207.000        0.9615   FALSE  TRUE
vsa_hyd      1.000       93.2692   FALSE FALSE
a_aro        1.189        5.7692   FALSE FALSE
weight       1.000       91.8269   FALSE FALSE
```

```
> ## Are there any near-zero variance predictors?
> rownames(predictorInfo)[predictorInfo$nzv]
```

```
[1] "negative"     "peoe_vsa.2.1" "peoe_vsa.3.1" "a_acid"       "vsa_acid"
[6] "frac.anion7." "alert"
```

```
> ## Examples:
> table(bbbDescr$a_acid)
```

```

  0   2   3
201   6   1
```

```
> table(bbbDescr$alert)
```

```

  0   1
206   2
```

```
> ## Let's get rid of these:
> filter1 <- bbbDescr[, !predictorInfo$nzv]
> ncol(filter1)
```

```
[1] 127
```

As mentioned in the text, there are some models that are resistant to near–zero variance predictors and, for these models, we would most likely leave them in.

What about the distributions of the remaining predictors? Although, it is time consuming to look at individual density plots of 127 predictors, we do recommend it (or at least looking at a sample of predictors). For example, the top panel of Figure 6 shows a random sample of eight predictors:

```
> set.seed(532)
> sampled1 <- filter1[, sample(1:ncol(filter1), 8)]
> names(sampled1)
```

```
[1] "o_sp3"         "adistm"        "peoe_vsa.6"    "o_sp2"
[5] "peoe_vsa.3"    "fpsa3"         "frac.cation7." "wpsa2"
```

A few of these predictors exhibit skewness and one (`frac.cation7.`) shows two distinct modes. Based on the rug plot of points in the panel for `o_sp2`, these data are also likely to be bimodal.

To numerically assess skewness, the function from the `e1071` package is used again:

```
> library(e1071)
> skew <- apply(filter1, 2, skewness)
> summary(skew)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.120   0.267   0.890   0.872   1.620   5.330
```

There are a number of predictors that are left– or right–skewed. We can again apply the Yeo–Johnson transformation to the data (some of the predictors are negative):

```
> yjBBB <- preProcess(filter1, method = "YeoJohnson")
> transformed <- predict(yjBBB, newdata = filter1)
> sampled2 <- transformed[, names(sampled1)]
```

The results for the sampled predictors are shown in the bottom panel of Figure 6. Although the distributions for `fpsa3` and `wpsa2` are more symmetric, the other predictors have either additional modes or more pronounced modes. One option would be to manually assess which predictors would benefit from this type of transformation.

Is there severe correlation between the predictors? Based on previous experience with these types of data, there are likely to be many relationships between predictors. For example, when we examine the predictor names we find that 24 are some type of surface area predictor. These are most likely correlated to some extent. Also, surface area is usually related to the size (or weight) of a molecule, so additional correlations may exist.

The correlation matrix of the predictors can be computed and examined. However, we know that many predictors are skewed in these data. Since the correlation is a function of squared values of the predictors, the samples in the tails of the predictor distributions may have a significant effect on the correlation structure. For this reason, we will look at the correlation structure three ways: the untransformed data, the data after the Yeo–Johnson transformation, and the data after a spatial sign transformation.

```
> rawCorr <- cor(filter1)
> transCorr <- cor(transformed)
>
> ssData <- spatialSign(scale(filter1))
> ssCorr <- cor(ssData)
```

```
> library(corrplot)
> ## plot the matrix with no labels or grid
> corrplot(rawCorr, order = "hclust", addgrid.col = NA, tl.pos = "n")
> corrplot(transCorr, order = "hclust", addgrid.col = NA, tl.pos = "n")
> ssData <- spatialSign(scale(filter1))
> ssCorr <- cor(ssData)
> corrplot(ssCorr, order = "hclust", addgrid.col = NA, tl.pos = "n")
```

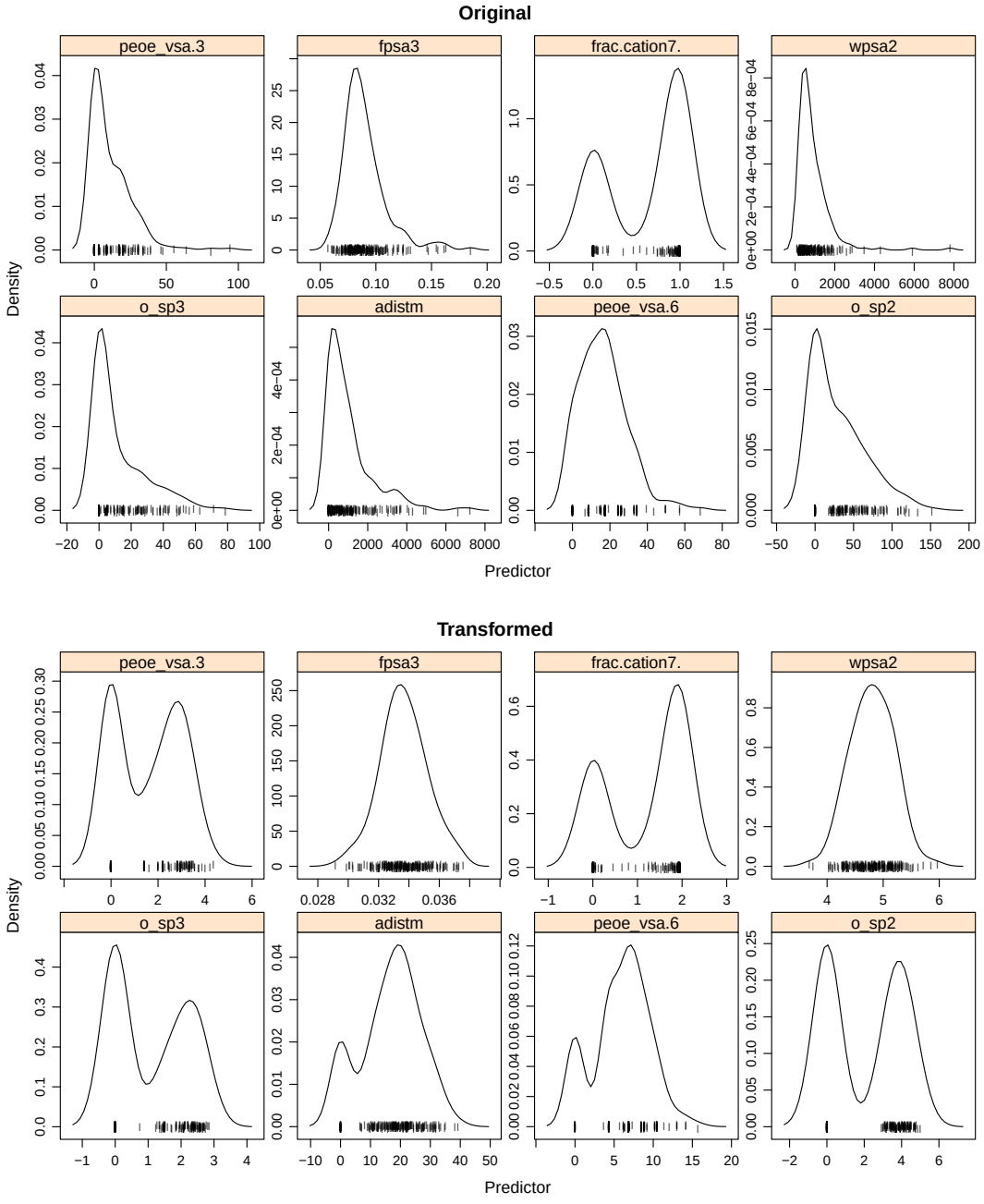

Figure 6: Density plots of the blood–brain barrier predictors before and after a Yeo–Johnson transformation.

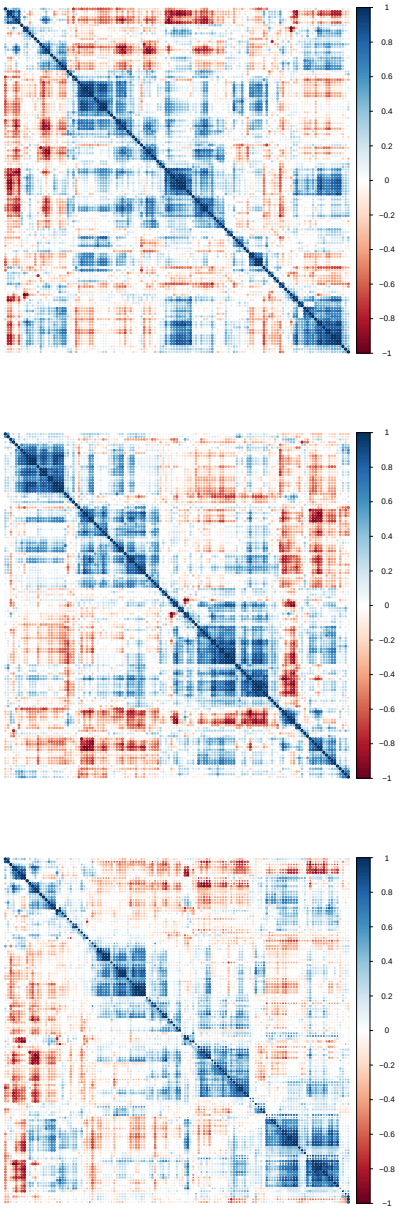

Figure 7: Correlation matrices for the raw data (top), transformed via the Yeo–Johnson transformation (middle) and the spatial sign transformation (bottom).

The results are in Figure 7. This visualization indicates that correlations lessen with increasing levels of transformations:

```
> corrInfo <- function(x) summary(x[upper.tri(x)])
> corrInfo(rawCorr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.0000 -0.1620  0.0643  0.0707  0.2860  1.0000

> corrInfo(transCorr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.0000 -0.1670  0.0608  0.0684  0.2910  1.0000

> corrInfo(ssCorr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.0000 -0.1610  0.0386  0.0556  0.2550  1.0000
```

Rather than transform the data to resolve between–predictor correlations, it may be a better idea to remove predictors. The caret function `findCorrelation` was described in the text. The user is required to state what level of pair–wise correlations that they are willing to accept. The code below shows (for these data) the trade–off between the correlation threshold, the number of retained predictors, and the average absolute correlation in the data. Figure 8 shows the results.

```
> thresholds <- seq(.25, .95, by = 0.05)
> size <- meanCorr <- rep(NA, length(thresholds))
> removals <- vector(mode = "list", length = length(thresholds))
>
> for(i in seq_along(thresholds)){
+   removals[[i]] <- findCorrelation(rawCorr, thresholds[i])
+   subMat <- rawCorr[-removals[[i]], -removals[[i]]]
+   size[i] <- ncol(rawCorr) -length(removals[[i]])
+   meanCorr[i] <- mean(abs(subMat[upper.tri(subMat)]))
+ }
>
> corrData <- data.frame(value = c(size, meanCorr),
+                        threshold = c(thresholds, thresholds),
+                        what = rep(c("Predictors",
+                                     "Average Absolute Correlation"),
+                                   each = length(thresholds)))
```

We can also try the subselect package (Cerdeira, Silva, Cadima & Minhoto 2014) to remove predictors. This package uses a different criterion to evaluate the quality of a subset and has less greedy methods to search the predictor space. First, we have to remove all linear dependencies from the

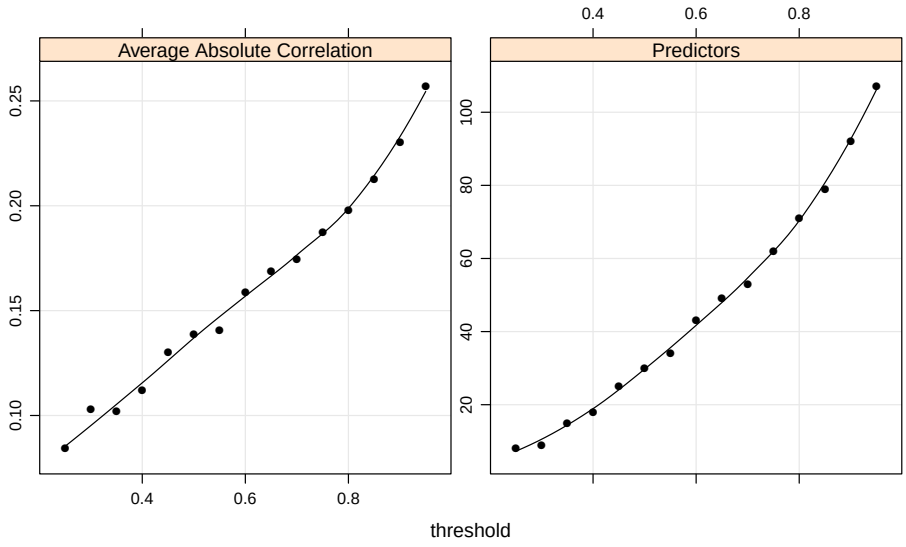

Figure 8: The average absolute correlation and the subset size for each value of the correlation filter threshold.

data. That includes perfect pair–wise correlations as well as relationships between three or more predictors. The `trim.matrix` function does that:

```
> library(subselect)
> ncol(rawCorr)
```

```
[1] 127
```

```
> trimmed <- trim.matrix(rawCorr, tolval=1000*.Machine$double.eps)$trimmedmat
> ncol(trimmed)
```

```
[1] 119
```

We can use simulated annealing and genetic algorithms to search for quality subsets. These techniques allow for lower and upper limits for the number of predictors. However, the functions get dramatically slower as the range increases. Here, we will look at one solution found by `findCorrelation` and, will subsequently use subselect to search within that subset size:

```
> set.seed(702)
> sa <- anneal(trimmed, kmin = 18, kmax = 18, niter = 1000)
> saMat <- rawCorr[sa$bestsets[1,], sa$bestsets[1,]]
>
> set.seed(702)
```

```
> ga <- genetic(trimmed, kmin = 18, kmax = 18, nger = 1000)
> gaMat <- rawCorr[ga$bestsets[1,], ga$bestsets[1,]]
>
> fcMat <- rawCorr[-removals[size == 18][[1]],
+                  -removals[size == 18][[1]]]
>
> corrInfo(fcMat)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.2540 -0.0976  0.0052  0.0124  0.0883  0.3800

> corrInfo(saMat)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.9540 -0.1200  0.0358  0.0583  0.2220  0.9560

> corrInfo(gaMat)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8890 -0.1680  0.0350  0.0333  0.2420  0.9220
```

The main difference between these results is that the greedy approach of `findCorrelation` is much more conservative than the techniques found in the subselect package.

Session Info

- R Under development (unstable) (2014-12-29 r67265), `x86_64-apple-darwin10.8.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: AppliedPredictiveModeling 1.1-6, car 2.0-21, caret 6.0-41, corrplot 0.73, e1071 1.6-3, ggplot2 1.0.0, knitr 1.6, lattice 0.20-29, mlbench 2.1-1, reshape2 1.4, subselect 0.12-3, vcd 1.3-2
- Loaded via a namespace (and not attached): BradleyTerry2 1.0-5, brglm 0.5-9, class 7.3-11, cluster 1.15.3, codetools 0.2-9, colorspace 1.2-4, COREelearn 0.9.43, digest 0.6.4, evaluate 0.5.5, foreach 1.4.2, formatR 0.10, gtable 0.1.2, gtools 3.4.1, highr 0.3, iterators 1.0.7, lme4 1.1-7, MASS 7.3-35, Matrix 1.1-4, minqa 1.2.3, munsell 0.4.2, nlme 3.1-118, nloptr 1.0.4, nnet 7.3-8, plyr 1.8.1, proto 0.3-10, Rcpp 0.11.2, rpart 4.1-8, scales 0.2.4, splines 3.2.0, stringr 0.6.2, tools 3.2.0

References

Cerdeira, J., Silva, P., Cadima, J., & Minhoto, M. (2014), *subselect: Selecting variable subsets*. **URL:** *http://CRAN.R-project.org/package=subselect*

Mente, S., & Lombardo, F. (2005), "A recursive-partitioning model for blood–brain barrier permeation," *Journal of Computer-Aided Molecular Design*, 19(7), 465–481.

Meyer, D., Zeileis, A., & Hornik, K. (2006), "The strucplot framework: Visualizing multi-way contingency tables with vcd," *Journal of Statistical Software*, 17(3), 1–48. **URL:** *http://www.jstatsoft.org/v17/i03/*

Yeo, I.-K., & Johnson, R. (2000), "A new family of power transformations to improve normality or symmetry," *Biometrika*, 87(4), 954–959.