Chapter 7

Outlier Detection

This chapter presents examples of outlier detection with R. At first, it demonstrates univariate outlier detection. After that, an example of outlier detection with LOF (Local Outlier Factor) is given, followed by examples on outlier detection by clustering. At last, it demonstrates outlier detection from time series data.

7.1 Univariate Outlier Detection

This section shows an example of univariate outlier detection, and demonstrates how to apply it to multivariate data. In the example, univariate outlier detection is done with function `boxplot.stats()`, which returns the statistics for producing boxplots. In the result returned by the above function, one component is `out`, which gives a list of outliers. More specifically, it lists data points lying beyond the extremes of the whiskers. An argument of `coef` can be used to control how far the whiskers extend out from the box of a boxplot. More details on that can be obtained by running `?boxplot.stats` in R. Figure 7.1 shows a boxplot, where the four circles are outliers.

```
> set.seed(3147)
> x <- rnorm(100)
> summary(x)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-3.3150 -0.4837  0.1867  0.1098  0.7120  2.6860

> # outliers
> boxplot.stats(x)$out

[1] -3.315391  2.685922 -3.055717  2.571203

> boxplot(x)
```

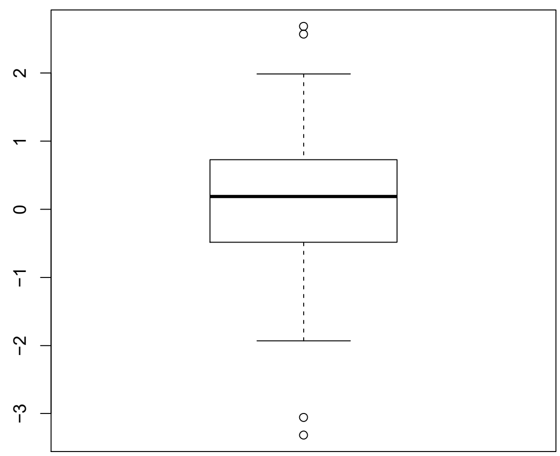

Figure 7.1: Univariate Outlier Detection with Boxplot

The above univariate outlier detection can be used to find outliers in multivariate data in a simple ensemble way. In the example below, we first generate a dataframe `df`, which has two columns, `x` and `y`. After that, outliers are detected separately from `x` and `y`. We then take outliers as those data which are outliers for both columns. In Figure 7.2, outliers are labeled with "+" in red.

```
> y <- rnorm(100)
> df <- data.frame(x, y)
> rm(x, y)
> head(df)

            x          y
1 -3.31539150  0.7619774
2 -0.04765067 -0.6404403
3  0.69720806  0.7645655
4  0.35979073  0.3131930
5  0.18644193  0.1709528
6  0.27493834 -0.8441813
```

```
> attach(df)
> # find the index of outliers from x
> (a <- which(x %in% boxplot.stats(x)$out))

[1]  1 33 64 74

> # find the index of outliers from y
> (b <- which(y %in% boxplot.stats(y)$out))

[1] 24 25 49 64 74

> detach(df)

> # outliers in both x and y
> (outlier.list1 <- intersect(a,b))

[1] 64 74

> plot(df)
> points(df[outlier.list1,], col="red", pch="+", cex=2.5)
```

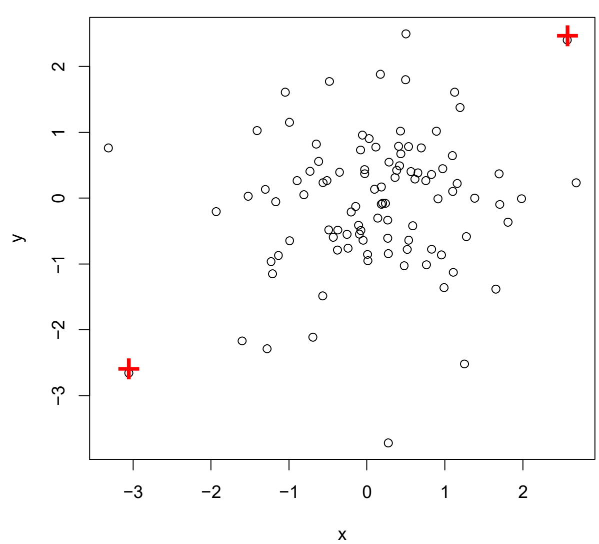

Figure 7.2: Outlier Detection - I

Similarly, we can also take outliers as those data which are outliers in either `x` or `y`. In Figure 7.3, outliers are labeled with "x" in blue.

```
> # outliers in either x or y
> (outlier.list2 <- union(a,b))

[1]  1 33 64 74 24 25 49

> plot(df)
> points(df[outlier.list2,], col="blue", pch="x", cex=2)
```

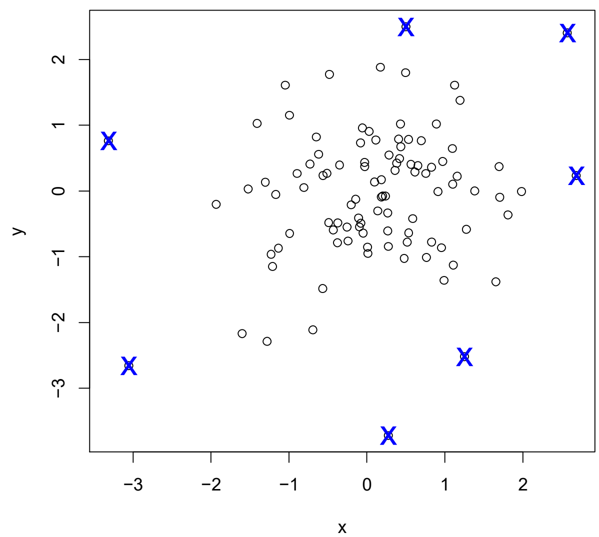

Figure 7.3: Outlier Detection - II

When there are three or more variables in an application, a final list of outliers might be produced with majority voting of outliers detected from individual variables. Domain knowledge should be involved when choosing the optimal way to ensemble in real-world applications.

7.2 Outlier Detection with LOF

LOF (Local Outlier Factor) is an algorithm for identifying density-based local outliers [Breunig et al., 2000]. With LOF, the local density of a point is compared with that of its neighbors. If the former is significantly lower than the latter (with an LOF value greater than one), the point is in a sparser region than its neighbors, which suggests it be an outlier. A shortcoming of LOF is that it works on numeric data only.

Function `lofactor()` calculates local outlier factors using the LOF algorithm, and it is available in packages *DMwR* [Torgo, 2010] and *dprep*. An example of outlier detection with LOF is given below, where k is the number of neighbors used for calculating local outlier factors. Figure 7.4 shows a density plot of outlier scores.

```
> library(DMwR)
> # remove "Species", which is a categorical column
> iris2 <- iris[,1:4]
> outlier.scores <- lofactor(iris2, k=5)
> plot(density(outlier.scores))
```

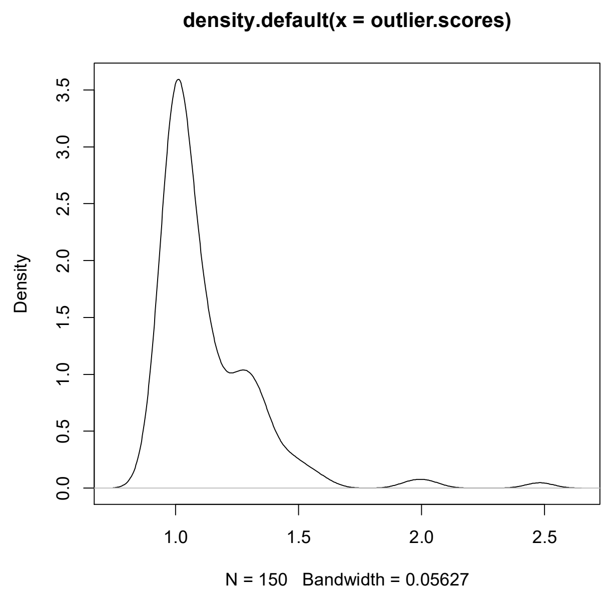

Figure 7.4: Density of outlier factors

```
> # pick top 5 as outliers
> outliers <- order(outlier.scores, decreasing=T)[1:5]
> # who are outliers
> print(outliers)
```

```
[1]  42 107  23 110  63
```

```
> print(iris2[outliers,])
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width
42           4.5         2.3          1.3         0.3
107          4.9         2.5          4.5         1.7
23           4.6         3.6          1.0         0.2
110          7.2         3.6          6.1         2.5
63           6.0         2.2          4.0         1.0
```

Next, we show outliers with a biplot of the first two principal components (see Figure 7.5).

```
> n <- nrow(iris2)
> labels <- 1:n
> labels[-outliers] <- "."
> biplot(prcomp(iris2), cex=.8, xlabs=labels)
```

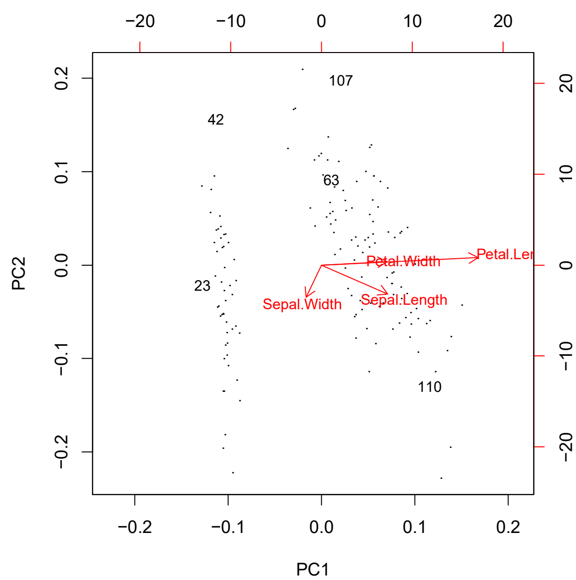

Figure 7.5: Outliers in a Biplot of First Two Principal Components

In the above code, `prcomp()` performs a principal component analysis, and `biplot()` plots the data with its first two principal components. In Figure 7.5, the x- and y-axis are respectively the first and second principal components, the arrows show the original columns (variables), and the five outliers are labeled with their row numbers.

We can also show outliers with a pairs plot as below, where outliers are labeled with "+" in red.

```
> pch <- rep(".", n)
> pch[outliers] <- "+"
> col <- rep("black", n)
> col[outliers] <- "red"
> pairs(iris2, pch=pch, col=col)
```

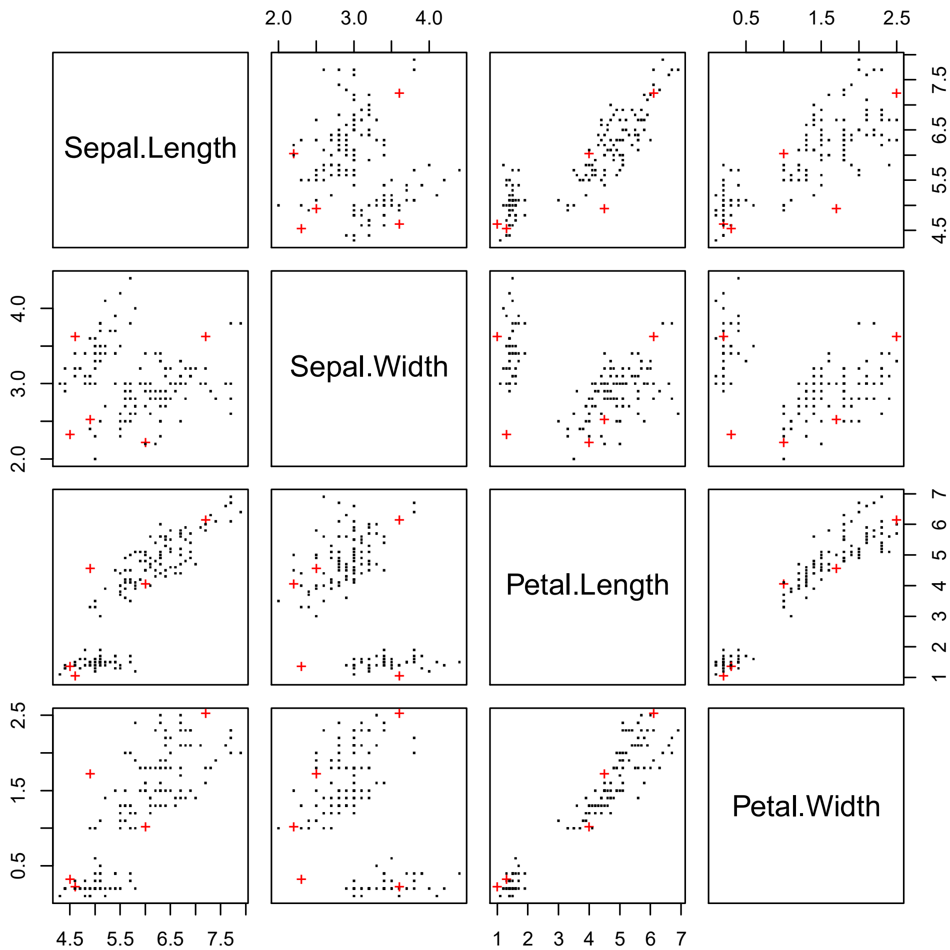

Figure 7.6: Outliers in a Matrix of Scatter Plots

Package *Rlof* [Hu et al., 2015] provides function `lof()`, a parallel implementation of the LOF algorithm. Its usage is similar to `lofactor()`, but `lof()` has two additional features of supporting multiple values of k and several choices of distance metrics. Below is an example of `lof()`. After computing outlier scores, outliers can be detected by selecting the top ones. Note that the current version of package *Rlof* (v1.0.0) works under MacOS X and Linux, but does not work under Windows, because it depends on package *multicore* for parallel computing.

```
> library(Rlof)
> outlier.scores <- lof(iris2, k=5)
```

```
> # try with different number of neighbors (k = 5,6,7,8,9 and 10)
> outlier.scores <- lof(iris2, k=c(5:10))
```

7.3 Outlier Detection by Clustering

Another way to detect outliers is clustering. By grouping data into clusters, those data not assigned to any clusters are taken as outliers. For example, with density-based clustering such as DBSCAN [Ester et al., 1996], objects are grouped into one cluster if they are connected to one another by densely populated area. Therefore, objects not assigned to any clusters are isolated from other objects and are taken as outliers. An example of DBSCAN be found in Section 6.4 Density-based Clustering.

We can also detect outliers with the k-means algorithm. With k-means, the data are partitioned into k groups by assigning them to the closest cluster centers. After that, we can calculate the distance (or dissimilarity) between each object and its cluster center, and pick those with largest distances as outliers. An example of outlier detection with k-means from the `iris` data (see Section 1.3.1 for details of the data) is given below.

```
> # remove species from the data to cluster
> iris2 <- iris[,1:4]
> kmeans.result <- kmeans(iris2, centers=3)
> # cluster centers
> kmeans.result$centers

  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     6.850000    3.073684     5.742105    2.071053
3     5.901613    2.748387     4.393548    1.433871

> # cluster IDs
> kmeans.result$cluster

  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [39] 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [77] 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2 2 2 2 2 2 3
[115] 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2 2 2 3 2 2 2 3 2 2 2 3 2 2 3

> # calculate distances between objects and cluster centers
> centers <- kmeans.result$centers[kmeans.result$cluster, ]
> distances <- sqrt(rowSums((iris2 - centers)^2))
> # pick top 5 largest distances
> outliers <- order(distances, decreasing=T)[1:5]
> # who are outliers
> print(outliers)

[1]  99  58  94  61 119

> print(iris2[outliers,])

    Sepal.Length Sepal.Width Petal.Length Petal.Width
99           5.1         2.5          3.0         1.1
58           4.9         2.4          3.3         1.0
94           5.0         2.3          3.3         1.0
61           5.0         2.0          3.5         1.0
119          7.7         2.6          6.9         2.3
```

```
> # plot clusters
> plot(iris2[,c("Sepal.Length", "Sepal.Width")], pch="o",
+      col=kmeans.result$cluster, cex=0.3)
> # plot cluster centers
> points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3,
+        pch=8, cex=1.5)
> # plot outliers
> points(iris2[outliers, c("Sepal.Length", "Sepal.Width")], pch="+", col=4, cex=1.5)
```

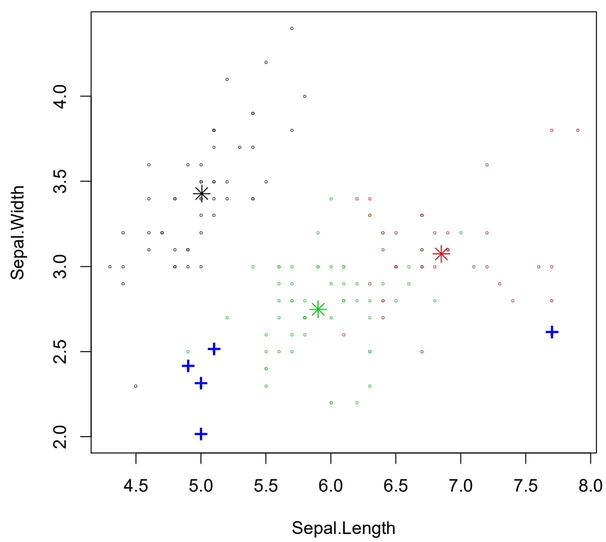

Figure 7.7: Outliers with k-Means Clustering

In the above figure, cluster centers are labeled with asterisks and outliers with "+".

7.4 Outlier Detection from Time Series

This section presents an example of outlier detection from time series data. In the example, the time series data are first decomposed with robust regression using function `stl()` and then outliers are identified. An introduction of STL (Seasonal-trend decomposition based on Loess) [Cleveland et al., 1990] is available at `http://cs.wellesley.edu/~cs315/Papers/stl%20statistical%20model.pdf`. More examples of time series decomposition can be found in Section 8.2.

```
> # use robust fitting
> f <- stl(AirPassengers, "periodic", robust=TRUE)
> (outliers <- which(f$weights<1e-8))

 [1]  79  91  92 102 103 104 114 115 116 126 127 128 138 139 140

> # set layout
> op <- par(mar=c(0, 4, 0, 3), oma=c(5, 0, 4, 0), mfcol=c(4, 1))
> plot(f, set.pars=NULL)
> sts <- f$time.series
> # plot outliers
> points(time(sts)[outliers], 0.8*sts[,"remainder"][outliers], pch="x", col="red")
> par(op) # reset layout
```

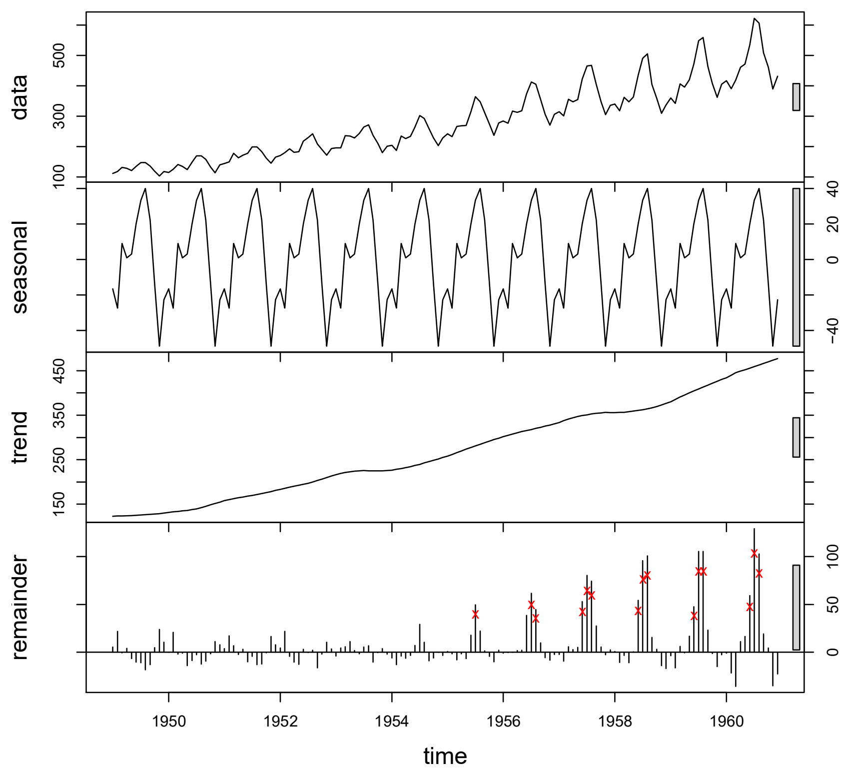

Figure 7.8: Outliers in Time Series Data

In above figure, outliers are labeled with "x" in red.

7.5 Discussions

The LOF algorithm is good at detecting local outliers, but it works on numeric data only. Package *Rlof* relies on the *multicore* package, which does not work under Windows. A fast and scalable outlier detection strategy for categorical data is the Attribute Value Frequency (AVF) algorithm [Koufakou et al., 2007].

Some other R packages for outlier detection are:

- Package *extremevalues* [van der Loo, 2010]: univariate outlier detection;
- Package *mvoutlier* [Filzmoser and Gschwandtner, 2015]: multivariate outlier detection based on robust methods; and
- Package *outliers* [Komsta, 2011]: tests for outliers.