Outlier detection

Let's *try* to define outliers first. Outliers are data values that, by some measure, appear not to belong to our data set. For example, maybe you have the following 10 data points (sorted):

3.5 3.7 4.2 5.5 5.8 6.7 6.9 7.1 7.3 45.6

Just by looking at the numbers, you should be able to tell that the last data point (45.6) is strange. With small data sets, it might be as easy as this to figure out what makes up an outlier.

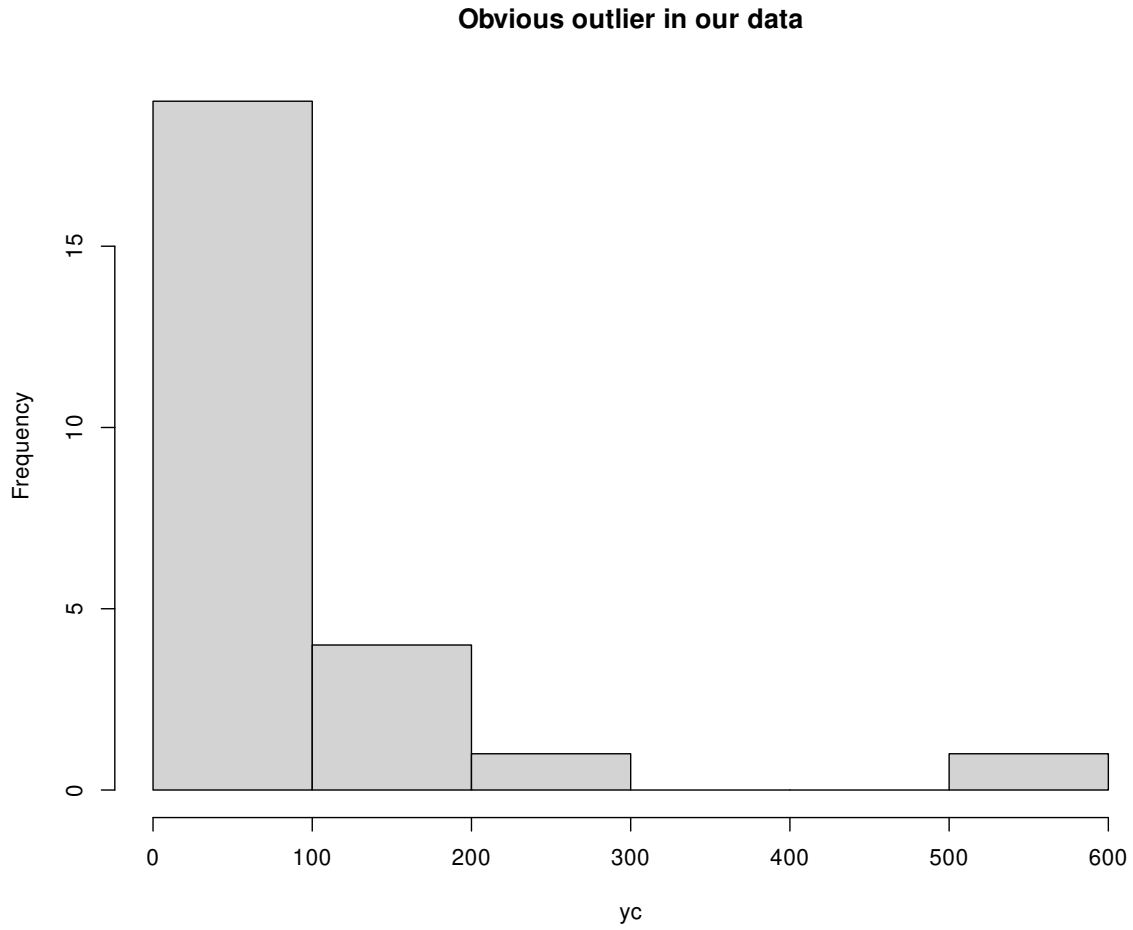
But in general, we might want to have better approaches. There are two reasonable methods of detecting outliers. The first is simply a variety of plots that let us visually examine the data. the second is based on hypothesis tests. We'll start with graphical methods.

1 Simple graphs

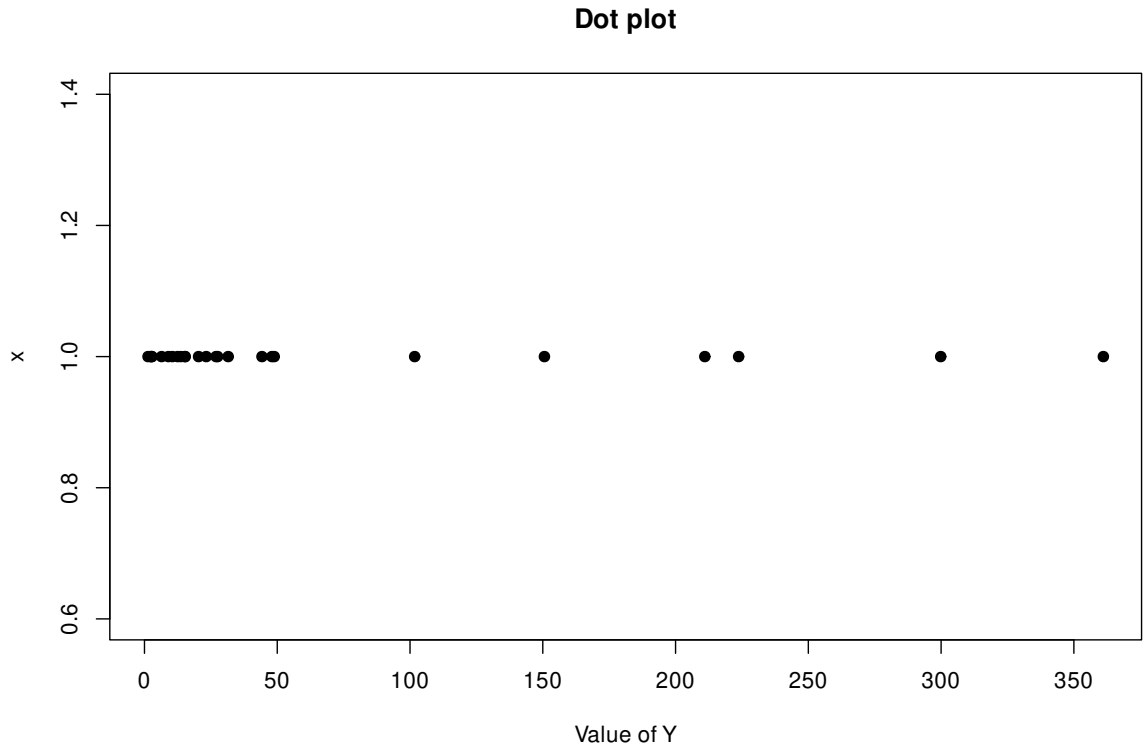
Sometimes very simple graphs like histograms will show you if there are any outliers present. Let's use the following data set for the next few examples:

3.2 9.2 10.7 12.0 14.7 14.7 14.8 15.7 23.6 24.9 26.4 26.5 31.9 34.0
40.0 44.3 57.2 59.2 95.9 113.2 126.6 141.7 176.3 251.1 591.1

Here, for example, is a histogram showing an obvious outlier:



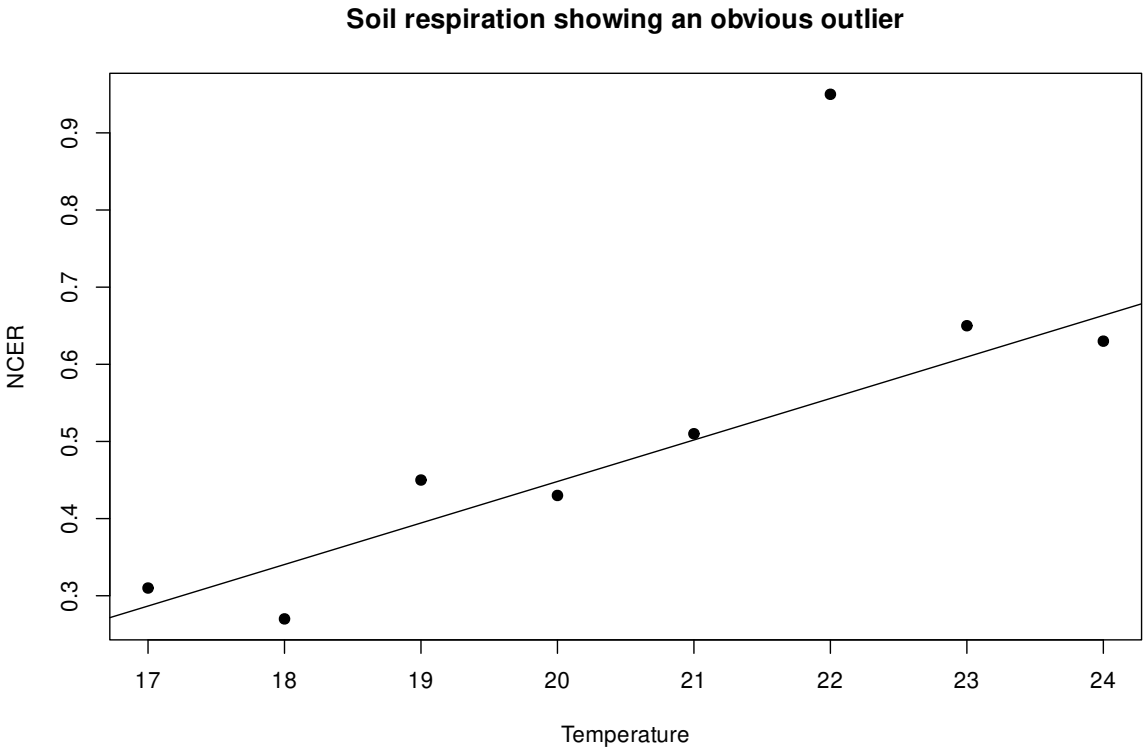
Similarly, you could construct a dot plot. A dot plot is simply our data on one axis showing the location of each data point. Here's an example using the same data set as for the histogram plot:



Looking at this plot, you might think there are at least three outliers, and maybe more.

This does raise the question - at what point should a data point be considered an outlier? Or to put it another way, how do we decide what makes an outlier (if it's not obvious)?

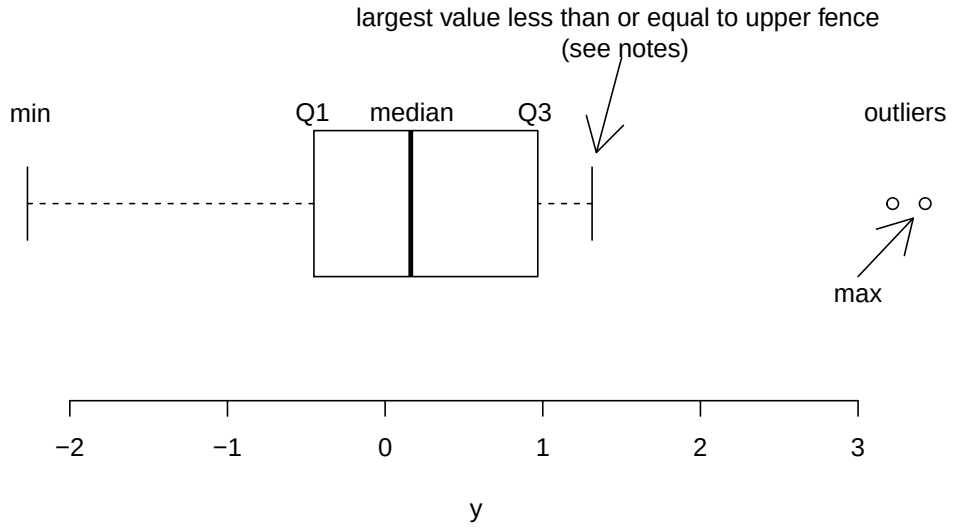
We'll give a more objective answer below, but before we do we should briefly look at outliers in 2 dimensions. Suppose we have the following values for NCER (the amount of CO_2) given off by the soil at various temperatures. If we just do a scatter plot (and maybe add a line), it should be obvious that the NCER value for 22 degrees is way too high - again, it's an obvious outlier.



Without showing the graphs, in dimensions higher than two, detecting outliers becomes more complicated. But there are some methods. For three dimensions, one can plot a three dimensional scatterplot and let it rotate (R can do this). Another option is to plot all possible combination of scatterplots (this doesn't completely guarantee we'll detect an outlier). For instance, if we have four variables W, X, Y and Z, we would plot W vs. X, W vs. Y, W vs. Z, X vs. Y, X vs. Z, and Y vs. Z. There are ways of doing this easily in R (look up the `pairs` function).

2 Boxplots

A boxplot is a very nice way of taking a look at our data and figuring out how spread out the data are, where most of the data are, and if there are any outliers.
Here's an example of a boxplot:



Some of these items you should be familiar with: median, min, max, outliers; but what about the rest (more details below)?

Q_1 = median of the lower half of the data (not including the median).

Q_3 = median of the upper half of the data (not including the median).

(One could label the median as Q_2. although that's usually not done.)

What is particularly important for us is that any points outside the "whiskers" are identified as *outliers*. Let's go through this again and provide some of the details in constructing a boxplot:

1) determine the median.

2) determine Q_1 and Q_3.

   a) divide your data into two halves, upper and lower. Do ***not*** include the median when you do this (just *eliminate* the median from your data when you do this).

      i) if you have an odd number of observations, just leave out the median.

      ii) if you have an even number of observations, the median isn't part of your data, so you can just ignore it.

   b) Q_1 = median of the lower half of your data (not including the median).

   c) Q_3 = median of the upper half of your data (not including the median).

3) calculate the IQR: $IQR = Q_3 - Q_1$.

4) now calculate the upper and lower fences:

   a) get $1.5 \times IQR$.

   b) lower fence $= lf = Q_1 - 1.5 \times IQR$.

   c) upper fence $= uf = Q_3 + 1.5 \times IQR$.

5) now draw the actual boxplot:

   a) draw a horizontal (or vertical) line (an axis) going from somewhere below the minimum value to somewhere above the maximum value.

   b) don't forget to add tick marks and actual y-values on the axis.

   c) draw lines (perpendicular to the axis) for the median, Q_1 and Q_3.

   d) draw a box extending from Q_1 to Q_3.

   e) draw a line (parallel to the axis) going to the minimum data value that is $\geq lf$. Then add a short perpendicular line at the end of this line.

   f) draw a line (parallel to the axis) going to the maximum data point that is $\leq uf$. Again, add a short perpendicular line at the end of this line.

      Important do *NOT* draw the fences on your plot.

   g) any values that are outside the fences (values bigger than the upper fence or smaller than the lower fence) are outliers.

      Draw individual dots for any outliers.

This is not the way most people make boxplots (see box), but it is pretty close and the calculations are much easier. If your sample size is reasonable, the differences in the resulting plot are pretty minor.

Optional: There are many methods of calculating Q_1 and Q_3. In fact, R has eight different methods! The basic problem is that some people believe that we shouldn't just eliminate the median if we have an odd number of observations. For example, if we have 7 data points, the median is the fourth data value. To calculate Q_1, we assign 1/2 of the median to each half of the data. So we now have 3.5 data points in the lower half, and 3.5 data points in the upper half. Now we need to find the median of 3.5 data points - ss you can tell, the math gets rather more complicated, which is why we're taking a simpler approach here.

Let's do an example using some (real) data on the length of radish seedlings exposed to caffeine (measured in mm):

14 5 13 10 12 16 6 24 13 33 16 12.5 13.5 1.5 15.5 30

The first step is to sort the data:

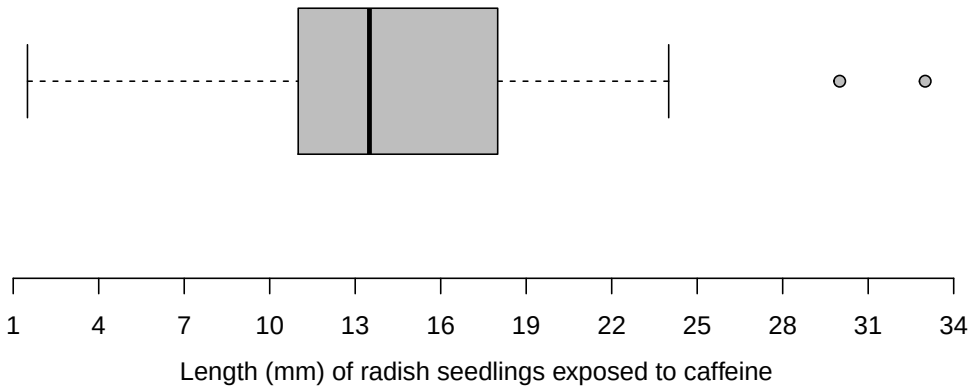
1.5 5 6 10 12 12.5 13 13 **13.5** 14 15.5 16 16 20 24 30 33

And we find the median is 13.5. We then get $Q_1 = (10+12)/2 = 11$ and $Q3 = (16+20)/2 = 18$ (we do not include the median in these calculations).

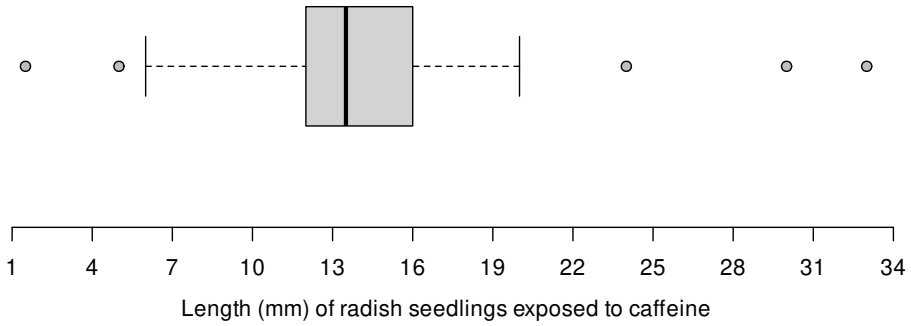
Then we get: $IQR = 18 - 11 = 7$ and $1.5 \times IQR = 1.5 \times 7 = 10.5$. This gives us the fences: $lf = 11 - 10.5 = 0.5$ and $uf = 18 + 10.5 = 28.5$.

We notice that the minimum = 1.5 which is greater than the lf, so we have no outliers on the low end.

On the other hand, we notice that there are two values that are greater than the uf (30 and 33), so we have two outliers on the upper end. Let's draw the actual boxplot:



This plot is drawn using our method of doing boxplots. It is obviously possible to get R to draw boxplots our way, but it isn't easy. From here on, we'll let R do things the way it wants to. Below is a boxplot using R's native method.



As you can see our radish boxplot does look different from the default method in R. Of particular concern to us is that R shows 5 outliers and our method shows 2 outliers. What causes this difference? The two main causes are:

> The multiplier we use for IQR. We used 1.5, which is a fairly standard multiplier, although obviously other values can be used. If you make this smaller, you get more outliers. If you make this bigger you will get less outliers.
>
> The method used to calculate IQR. Obviously this can vary a lot (see box), so this can affect the number of outliers we have.

It is probably best to use our method as a way of understanding boxplots, but then use R's default method for most boxplots for which you want to find outliers.

3 Statistical tests for outliers

There are actually quite a few tests for outliers. Most of these have several drawbacks, so we will restrict our attention to something called the Generalized Extreme Studentized Deviate (or Generalized ESD) test. This is fairly simple to use although we'll let R do most of the work once we've outlined the procedure.

The only thing this test requires is a *guess* at the maximum number of outliers that are suspected (you can always give a higher maximum if you're not sure). Some of the other tests require that you make exact guesses at the number of outliers you suspect.

The test works by examining how far each data point is away from the average, adjusting this by dividing by the standard deviation, and then comparing it to how far away each

point is expected to be - assuming the data are roughly normal.

In general, the hypotheses are H_0: there are no outliers and H_1: there are as many as k outliers. As usual we pick α and then proceed as follows:

1. We calculate the *studentized* distances. Basically this means we calculate the distance of each point from the sample mean and divide by the standard deviation:

$$R_i = \frac{max_i|y_i - \bar{y}|}{s}$$

   In other words, calculate the distance of *each* point from the mean and divide by the sample standard deviation (s).

2. Pick the largest R_i and remove the observation that goes with it.

3. Repeat step 1, but now with $n - 1$ observations.

4. Repeat steps 1 - 3 until you've removed k observations. This gives you k test statistics $(R_1, R_2, ..., R_k))$.

5. Now you need k critical values (CV's). You calculate each one as follows:

$$CV = \frac{(n-1)t_{p,n-i-1}}{\sqrt{(n-i-1+t^2_{p,n-i-1})(n-i+1)}}$$

where

$$p = 1 - \frac{\alpha}{2(n-i+1)}$$

   Okay, so now you're worried. But let's do an example. Suppose you have $n = 10$, $i = 2$, and $\alpha = 0.05$. Then you have:

$$p = 1 - \frac{0.05}{2(10-2-1)} = 0.00277$$

which we then substitute into:

$$CV = \frac{(10-2)t_{0.00277,10-2-1}}{\sqrt{(10-2-1+t^2_{0.00277,10-2-1})(10-2+1)}}$$

$$= \frac{8 \times t_{0.00277,7}}{\sqrt{(7+t^2_{0.00277,7}) \times 9}}$$

using R we find $t_{0.00277,7} = 3.23$ so we get:

$$\frac{8 \times 3.23}{\sqrt{(7+3.23^2) \times 9}} = 2.06$$

And this would be our critical value for R_2.

6. Finally(!), we compare each of our R_i's to each of the CV_i's, but we look for the *largest* i for which $R_i \geq CV_i$. This, *and all smaller* i's are we consider outliers.

This is not all that difficult, but it is extremely tedious, so we will let R do all the work as usual. To do this we need to install the `EnvStats` package. In RStudio, select the `Packages` tab on the lower right window, click on `Install`, and type the name of the package in the text box that pops up. Make sure you have the `Install dependencies` box checked or you may get a real mess when you try to use it. As soon as RStudio has found your package (keep typing the name - it will come up), click the `Install` button and you're off and running.

To actually use the package, you have to tell R to load it. For any package, you would simply do:

```
library(package-name)
```

so for our package you would do:

```
library(EnvStats)
```

As an aside, if you haven't noticed, there are probably over 1,000 packages available for R that can do almost anything you can imagine.

Now that we have our package installed and loaded, we can simply give R our dataset together with the maximum number of outliers we expect. The command is:
`rosnerTest(x, k)`

Where `x` is our data set and `k` is the maximum number of outliers we are looking for.

Here's an example using the dataset we introduced above, testing for a maximum of 5 outliers:

```
yc <- scan(nlines = 2)
3.2 9.2 10.7 12.0 14.7 14.7 14.8 15.7 23.6 24.9 26.4 26.5 31.9 34.0
40.0 44.3 57.2 59.2 95.9 113.2 126.6 141.7 176.3 251.1 591.1
rosnerTest(yc,5)
```

R will print a lot of information, but the easiest way to figure things out is to look near the bottom of the results for the following:

```
  i    Mean.i       SD.i Value Obs.Num    R.i+1 lambda.i+1 Outlier
1 0 78.35600 123.23722 591.1      25 4.160626   2.821681    TRUE
2 1 56.99167  62.77532 251.1      24 3.092112   2.801551    TRUE
3 2 48.55217  48.29870 176.3      23 2.644954   2.780277   FALSE
4 3 42.74545  40.39066 141.7      22 2.449936   2.757735   FALSE
5 4 38.03333  34.64190 126.6      21 2.556634   2.733780   FALSE
```

And from this we can see that we have two outliers (outliers are indicated in the last column, the actual values are in the `Value` column.

Finally, note that it is possible to change α and some other details using the `rosnerTest` function. Use `help(rosnerTest)` to see how to change these.