Following are instructions on how to use R to perform a statistical curve fit of experimental data. It assumes you have installed R and RStudio as described in the document "Setup for RStudio". See the document "Data management in RStudio" for how to manage your data.

In this project you have performance data consisting of execution counts as a function of data counts. You also have a body of computer science theory that predicts some of the performance data should follow a quadratic curve, *i.e.* $y = An^2 + Bn + C$, and some should follow an $n \lg n$ curve, *i.e.* $y = An \lg n + Bn + C$. Each of these curves has three coefficients, A, B, and C.

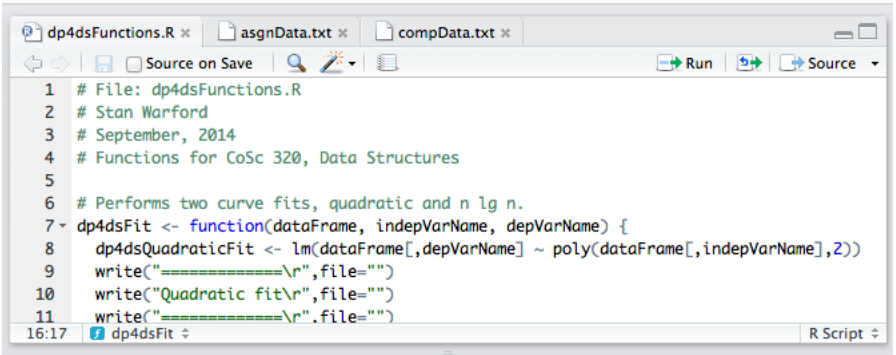
To determine whether a set of data more closely matches a quadratic curve or an $n \lg n$ curve, you perform a statistical computation called a *curve fit* to the data. With a curve fit, you specify the equation and a set of data. The computation adjusts the values of the coefficients so that the curve matches the data as close as possible. It also computes a residual standard error (*RSE*) that is a measure of how far off the best fit is. Hence, to determine whether a data set more closely matches a quadratic or an $n \lg n$ curve you must do two curve fits for the data — one for the quadratic equation and one for the logarithmic equation and record the *RSE* for each. The curve fit that has the smaller *RSE* is the one that more closely matches the data.

One of the questions you must address in your paper is whether the experimental data matches the theory. If theory predicts that the data should be quadratic, and if the *RSE* for the quadratic fit is smaller than the *RSE* for the logarithmic fit, then the data matches the theory. Otherwise, it does not.

1. *To install the dp4ds functions*
   Several functions written in the R language are provided for this project. The source code is contained in the file `dp4dsFunctions.R`, which you should download from our course web page into your `SortProject` directory.

   http://www.cslab.pepperdine.edu/warford/cosc320/dp4dsFunctions.R

   In RStudio, open the file in the text editor and click the Source button on the top right of the pane.



   The functions from the source file will be installed in your environment. You can see them listed in the environment tab.

2. *To run the curve fit*
   Suppose your data frame is called `compData` and you have an independent variable named `NumSorted` and a dependent variable named `InsertComp`.

```
> compData
   NumSorted InsertComp SelectComp HeapComp MergeComp QuickComp
1         500      63597     899999   799999    699999    599999
2        1000     254362    8444444  7444444   6444444   5444444
3        1500     554589    8888888  7888888   6888888   5888888
4        2000    1003667   81111111 71111111  61111111  51111111
5        2500    1606133   82222222 72222222  62222222  52222222
6        3000    2272433   83333333 73333333  63333333  53333333
7        3500    3083553   84444444 74444444  64444444  54444444
8        4000    3997696   85555555 75555555  65555555  55555555
9        4500    5066599   86666666 76666666  66666666  56666666
10       5000    6349081   87777777 77777777  67777777  57777777
11       5500    7670586   88888888 78888888  68888888  58888888
12       6000    9080159   89999999 79999999  69999999  59999999
```

The curve fit function `dp4dsFit()` has five parameters:

- A data frame.
- The name of the independent (x) variable enclosed in quotes.
- The name of the dependent (y) variable enclosed in quotes.
- The label for the x-axis.
- The label for the y-axis.

For example, to do a curve fit with the above data frame, enter the following on the RStudio command line. Do not enter the +. It is the R command line continuation prompt.

```
> dp4dsFit(compData, "NumSorted", "InsertComp",
+ "Number sorted", "Insert sort: Number of comparisons")
```

The function does a quadratic fit followed by an $n \lg n$ fit and prints a summary of the results to the console.

3. *To interpret the curve fit*
Here is a selection of the summary for the quadratic fit from the run with the above data.

```
Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                          3416871       8122  420.71  < 2e-16 ***
poly(dataFrame[, indepVarName], 2)1  9817068      28134  348.93  < 2e-16 ***
poly(dataFrame[, indepVarName], 2)2  2324549      28134   82.62 2.82e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 28130 on 9 degrees of freedom
Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
F-statistic: 6.429e+04 on 2 and 9 DF,  p-value: < 2.2e-16
```

In the Coefficients table,

- `(Intercept)` corresponds to coefficient C in $y = Ax^2 + Bx + C$,
- `poly(dataFrame[, indepVarName], 2)1` corresponds to coefficient B, and
- `poly(dataFrame[, indepVarName], 2)2` corresponds to coefficient A.

Hence, the best quadratic fit is $y = 2324549x^2 + 9817068x + 3416871$.

There are nine degrees of freedom, because there are three coefficients, A, B, and C, 12 data points, and $12 - 3 = 9$. The residual standard error (RSE) is 28,130 in this computation and is defined in general as

$$RSE = \sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{d.f.}}$$

where the sum is over all the data points, y_i is the y value of an individual data point, $\hat{y}_i$ is the y value of the point on the curve whose x value is the same as the x value of y_i, and $d.f.$ is the degrees of freedom. For the data with this experiment, you should always verify that you have three coefficients in the coefficient table, 12 data points, and nine degrees of freedom.

Here is a selection of the summary for the $n \lg n$ fit from the run with the above data.

```
Coefficients:
                                                          Estimate Std. Error t value
(Intercept)                                             1327296.56  241061.68   5.506
dataFrame[, indepVarName]                                -10547.80     831.28 -12.689
dataFrame[, indepVarName]:log(dataFrame[, indepVarName])    1357.47      92.53  14.671
                                                        Pr(>|t|)
(Intercept)                                             0.000377 ***
dataFrame[, indepVarName]                               4.78e-07 ***
dataFrame[, indepVarName]:log(dataFrame[, indepVarName]) 1.37e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 155300 on 9 degrees of freedom
Multiple R-squared:  0.9979,Adjusted R-squared:  0.9974
F-statistic:  2105 on 2 and 9 DF,  p-value: 9.571e-13
```

In the Coefficients table,

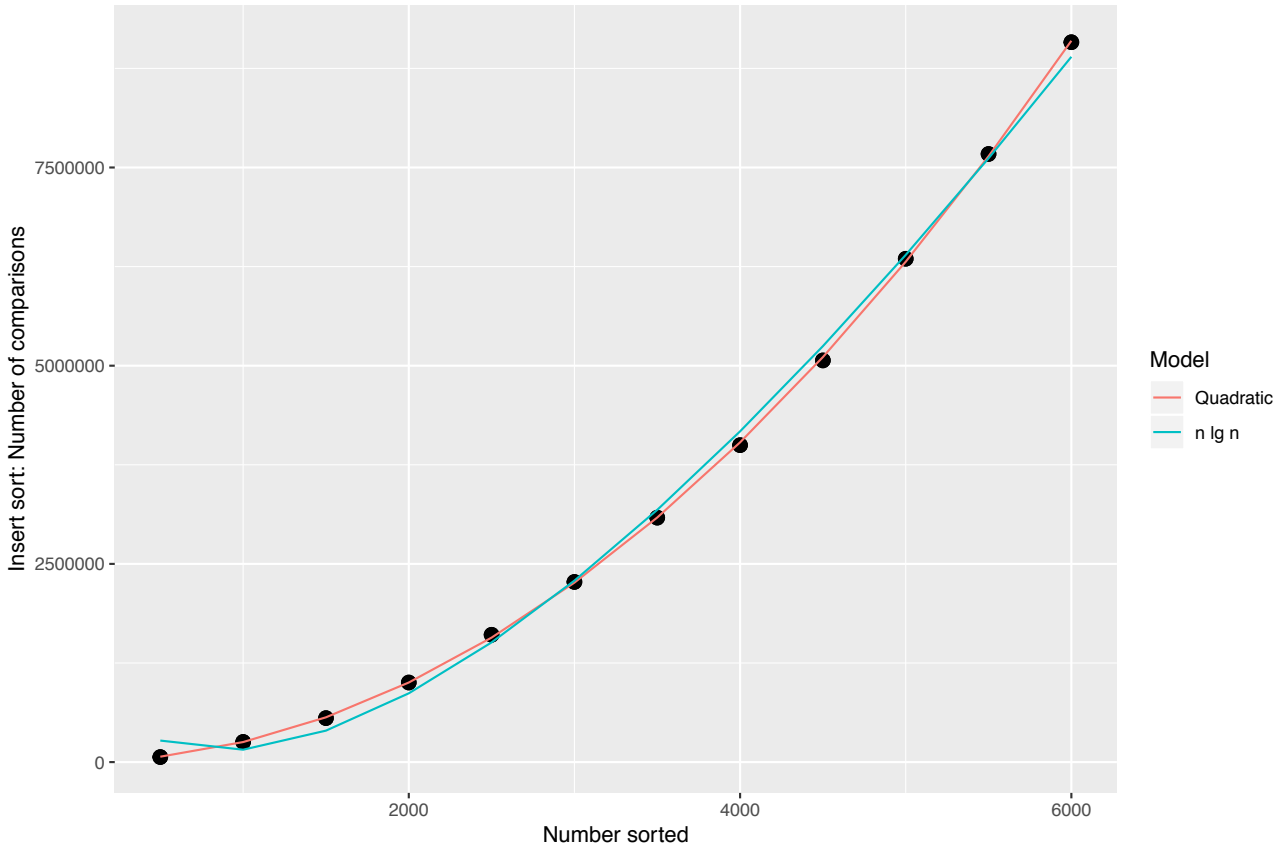
- `(Intercept)` corresponds to coefficient C in $y = An \log n + Bn + C$,
- `dataFrame[, indepVarName]` corresponds to coefficient B, and
- `dataFrame[, indepVarName]:log(dataFrame[, indepVarName])` corresponds to coefficient A.

Hence, the best $n \lg n$ fit is $y = 1357.47n \log n - 10547.80n + 1327296.56$.

As with the quadratic curve fit, there are nine degrees of freedom, because there are three coefficients, A, B, and C, 12 data points, and $12 - 3 = 9$. The RSE is 155,300. Because the RSE for the quadratic fit is 28,130, and 28,130 is less than 155,300, the data show that the number of element comparisons as a function of the data count in the insert sort algorithm is quadratic as opposed to $n \lg n$.

4. *To save the plot*

The `dp4dsFit()` function produces a plot of the data together with two curves, the least squares fit of a quadratic equation in red the least squares fit of an $n \lg n$ equation in blue. The following figure is produced by the above call to `dp4dsFit()`.



With the Plot tab selected and the plot showing in the pane, click Export → Save as PDF … . A useful feature is the ability to set a custom size for the plot in the PDF size field. This feature will scale the plot without distorting the text. You can also scale the PDF document in LaTeX when you include the graphic, but the LaTeX feature scales the complete figure including the text in the image.