

# “Visualizing Open Source Histories”

**Nitin Madnani**

nmadnani@umiacs.umd.edu

## **Introduction:**

The goal of this project is to apply the TimeSearcher[1] tool to temporal data obtained from open source project website, such as <http://www.sourceforge.net> and <http://www.freshmeat.net>, and determine if it is capable enough to make discoveries and visualize patterns in this data. The data has been collected for all releases in the last 5 years (2000-2005) of about 100 open source projects and consists of attributes such as :

- **Total Lines of Code:** The number of lines in the source code in any release of the project.
- **Comment-Code Ratio:** The ratio of the number of lines of comments in any project to the number of lines of code.
- **Average Class Coupling:** This number represents the amount of “coupling” between any two classes in project. It measures the degree to which a class is related or connected to other classes (such as by one class calling another class or using a variable from another class). Coupling represents the degree of mutual interdependence and lower coupling is better.
- **Average Percent Lack of Cohesion:** “Cohesion”, crudely speaking, represents the degree to which the variables and methods in a particular class work together towards the same goal. A higher number means that the variables and methods are not working towards the same goal. Lower lack-of-cohesion is better, i.e., higher cohesion is better.

## **Motivation:**

The motivation behind this project is to thoroughly explore the dataset provided by using TimeSearcher and to determine prevalent temporal patterns in open source projects. This exploration is done in term of queries on the temporal data.

TimeSearcher seems to be a good tool for this project because it allows us to express these queries very easily and conveniently in terms of simple-to-understand graphical constructs directly on the data.

## **Methodology:**

The methodology for this project can be divided into two salient phases:

**1. Data Preparation & Normalization:** This phases dealt with taking the original data as provided in a CSV format, and converting it into the TQD (Temproral Query Data) format as required for input to TimeSearcher. The actual conversion is pretty straightforward but complications arise because of disparities between the way that the data has been collected and the way that It can be represented in TimeSearcher.

The collected data just consists of one line of comma-separated attributes for each release of each project. The problem is that not only do different projects have different dates of inception but their releases also occur on different dates. Since the data the needs to be represented has to have one global timeline for all attributes, this was a problem. To resolve this problem, I took the following steps:

1. The starting point of the global timeline was fixed to be January, 2000 and the end point was fixed at December, 2004.
2. Projects that started before the global starting point were assigned zero values to attributes at all time points between its start and the global starting point. All projects that ended before the global end point had the values of the release at its last time point assigned to all further time points.
3. At all time points between two releases, the attribute values were the same as the earlier of the two releases.

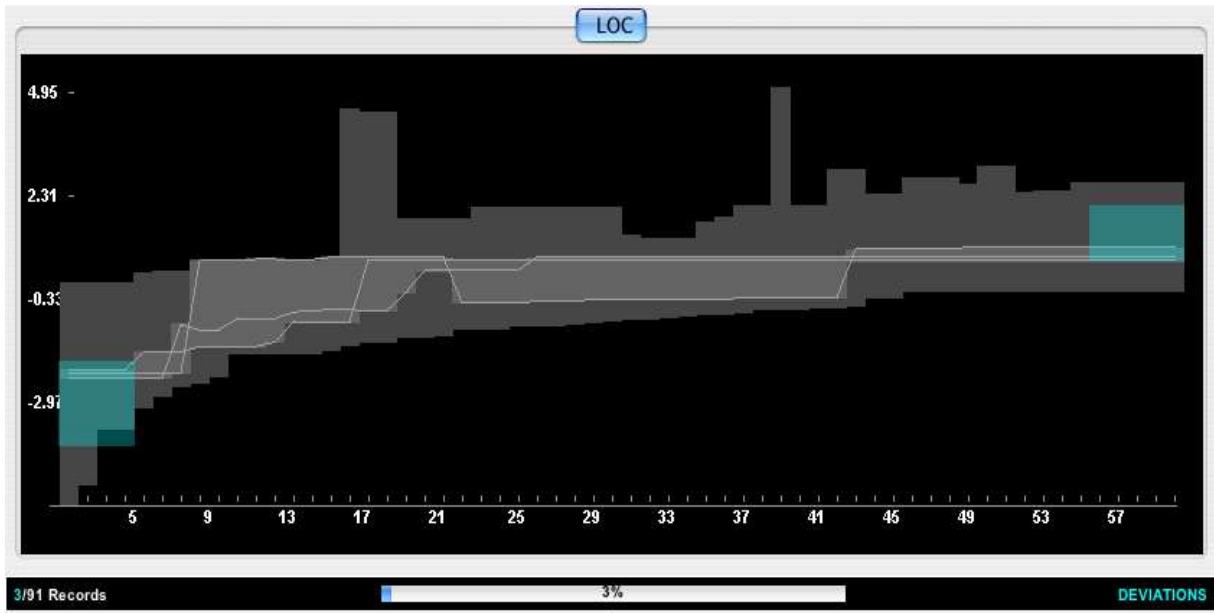
In my opinion this is not the most optimal scheme for data representation but this is what I could come up with given the data format. A python script, incorporating these decisions, parsed the original CSV file and created the required TQD files.

**2. Query Formulation:** This was the most important phase, even though this did not require as much programming effort as data preparation. According to me, the following dynamic attributes are the ones that warrant significant inspection for thorough exploration:

- a) Total lines of code for each release of each project.
- b) Average Coupling for each project.
- c) Percent Lack of Cohesion for each project.

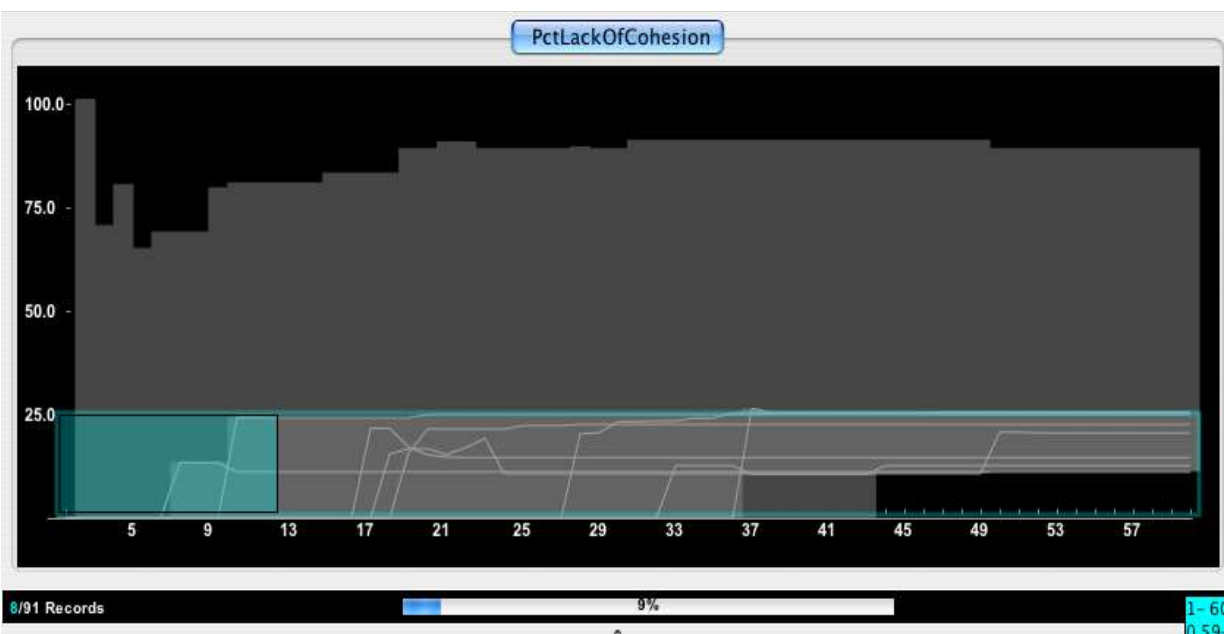
### **Query Results:**

This section describes the results for the queries formulated for each of the above attributes. Note that for most of these queries, deviation normalized data rather than raw data was used because the same attributes can have extremely diverse values for different projects, and hence it is important to to normalize these values to use the same data ranges for all projects.



*Figure 1. Only a few projects exhibit a very large code size increase*

- **Total lines of code (LOC):** Figure 1 shows the TimeSearcher with the LOC data for the projects. Only about 3 projects see a very large increase (2-4 times) in the number of lines of code. This indicates that, barring any anomalies, only a few projects made significantly major changes to their codebase.

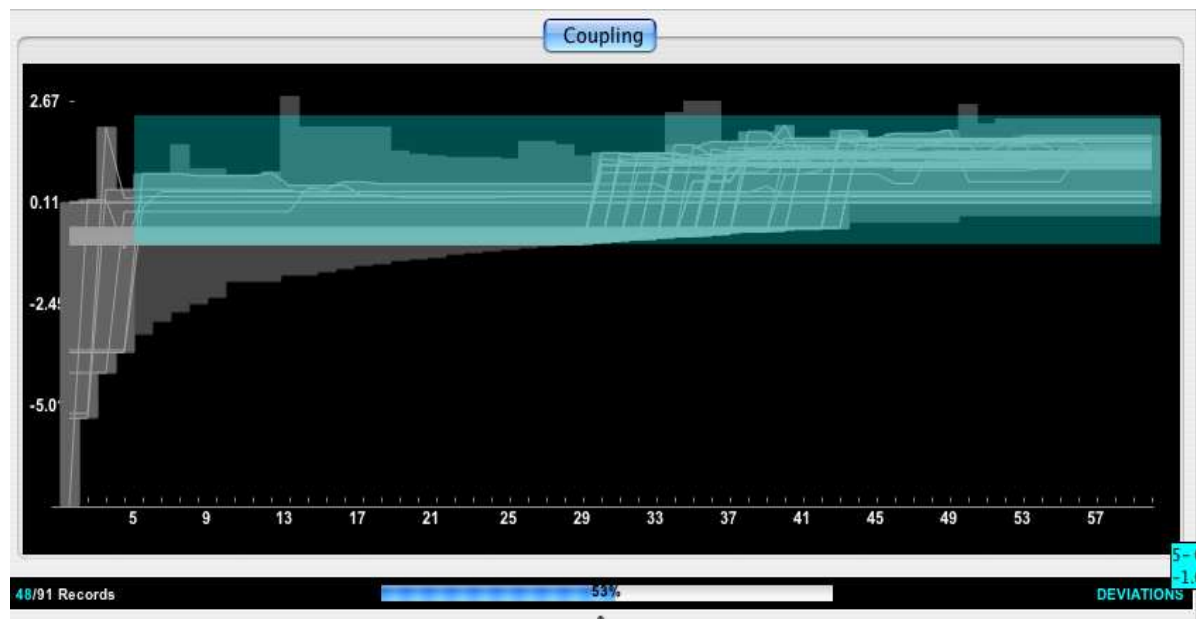


*Figure 2. Lack of cohesion is a significant problem in open source projects.*

- **Percent Lack Of Cohesion:** Figure 2, using a Variable Timebox shows that

there were less than 10 projects, in the whole 5 years of sampling, that had a lack-of-cohesion index less than 25% in any of the 5 years. In other words, More than 90% of the projects lacked cohesion significantly in our sampling period.

**Average Coupling:** Coupling is a more direct measure of complexity than anything else and lower coupling numbers are considered to be better. Figure 3 shows clearly that more than half of the projects under consideration have significantly high coupling numbers **and**, more importantly, they stay there in the whole sampling period.



*Figure 3. More than half the projects are in the red when it comes to coupling.*

### **Missing Data:**

The dataset used for this project only represented preliminary data collected and was missing some important information like, say, the number of downloads for each release of each project and the number of contributors for each release. Therefore, a completely thorough exploration of open source projects was not possible.

### **Conclusion:**

TimeSearcher is a very effective tool for exploring temporal data. Some suggestions (might already be implemented in version 2.0):

- Variable Angular Query?
- More robust (crashed due to a single extra space in TQD file)
- Allow viewing different attributes [already in 2.0]
- Allow construction of queries with different attributes?

**Other important questions that I missed ?**