

.....

---

# IISME Educational Transfer Plan

## Using Treemap Visualizations to Understand Student Assessment Data



*Sponsored by Intel Corporation*

Brendan Creane  
IISME Fellow  
Elementary

August, 2005

---

# Using Treemap Visualizations to Understand Student Assessment Data

## *An Instructional Guide and Professional Development Outline*

### **Abstract**

**Testing students** is a fact of the American educational landscape. All too often, teachers are mandated to administer frequent assessments in core subject areas such as math, literacy, science, and history, but they are not given the tools to make good use of the large amounts of data that result. Treemaps are a novel visualization tool that enables people to take in large amounts of data quickly, and further to manipulate how the data are viewed with an eye towards identifying current trends and associated historical antecedents.

The goal of this Educational Transfer Plan is to create a customized Treemap visualization that allows teachers in San Francisco Unified School District to view the past year of math assessment data. Teachers will be given the tools they need to understand Treemaps and how to manipulate the visualization in order to identify trends, needs, and strengths within their classroom. Ultimately, teachers will be encouraged to use the Treemaps to drive instruction in their classroom.

My intention is to make a Treemap available using Open Source software that will serve as a template for adventurous teachers to create visualizations that help them in their own teaching. Finally, it is my hope that school districts and the private software companies they employ (e.g. Little Schoolhouse Software) expand their current assessment data tools to include Treemap visualizations.

### **National Board Standards**

*VIII. Assessment: Accomplished teachers understand the strengths and weaknesses of different assessment methods, base their instruction on ongoing assessment, and encourage students to monitor their own learning.*

The tree-map view of assessment data facilitates the task of monitoring student progress as seen through the lens of ongoing assessments by making the process convenient and quick.

*X. Reflection: Accomplished teachers regularly analyze, evaluate, reflect on, and strengthen the effectiveness and quality of their practice.*

The tree-map view of test data supports teachers in analyzing the effectiveness of their teaching practice by high-lighting particular areas of strength and weakness as they correlate to curricular strands.

*XI. Contributions to the Profession:*

*Accomplished teachers work with colleagues to improve schools and to advance knowledge and practice in their field.*

By publishing assessment data by student, class, and school, and then allowing people to further filter the data by curricular strand, gender, age, home language, etc. Tree-maps will provide a high-level view of how teachers are supporting student learning in a broader sense than individual achievement.

## Objectives

As a result of completing the professional development event and consulting the associated instruction manual, teachers will:

1. Understand the basic features of Treemaps including: how to group, filter, and highlight assessment data. In addition, teachers will be able to control features such as the size of a Treemap cell.
2. Know how to manipulate control features of a Treemap in order to identify trends, determine the significance of demographic data, and identify historical antecedents to current student performance.
3. Formulate strategies to remediate specific curricular strands with an eye towards improving student achievement.

## Resources and Materials

- Teachers at Harvey Milk Civil Rights Academy will have access to last year's math assessment data, viewable on a custom Treemap.
- Accompanying the training event, teachers will receive a brief instructional manual highlighting the features of the Treemap, as well as strategies for identifying specific trends in their classroom.
- In order to present the Treemap visualization to teachers, HMCRA will install an open source Treemap server from the University of Maryland, Human-Computer Interaction lab.

## Procedure

1. Setup a computer lab environment that has internet-capable computers as well as an LCD projector.

2. Introduce the topic of using Treemaps to help understand student test data.
3. Introduce the pre-training assessment and ask teachers to respond to the questions.
4. Display a sample Treemap and explain the basic features – cells, text on cells, mouseover boxes, filters, indicators, and Mouseclick menus. Make this as brief as possible since teachers will be doing their own scavenger hunt in the next step.
5. Instruct teachers that they'll be playing a scavenger hunt that requires playing with and understanding two Treemaps. The Treemaps have been created by Honeycomb to explore their use within a consumer product context. Examples include: Peet's Coffee's, iTunes Top 100 songs, amazon.com product showcase, and an interactive population map from Wikipedia. See the Appendix for a copy of the Scavenger Hunt.
6. Pass out a copy of the Treemap Scavenger Hunt game. The first pair of teachers to correctly complete the scavenger hunt will receive a small prize. Encourage early finishers to come up with their own interesting questions and strategies for find an answer (e.g. what is the densest country in terms of population in the African continent?)
7. Tell teachers that they've just mined a large amount of data for useful information regarding coffee, plasma screen HDTV's, and demographic information about Europe. In a similar manner, teachers can mine their own classroom assessment data (in this case math) for trends regarding student performance.
8. Display a Treemap illustrating one of the teacher's class data and point out salient features, including: overall performance by gender, ELL status. Trends from assessments 1 through 6 for different curricular strands, for different demographic groups. Quantify how much better the top 50% is doing compared to the lowest quartile.
9. Instruct teachers that they're going to be looking at their own classroom's math assessment data from the 2004/2005 school year. They can use the worksheet that I've created to help identify trends, both positive and negative, and to identify other useful characteristics of their students' performance.
10. Pass out sheet with guiding questions regarding classroom math assessment data as it relates to a Treemap visualization. Tell teachers that they'll be asked to "report out" the top trends or insights that they've gained by studying their classroom data. In addition, teachers will have the opportunity to share strategies for utilizing the Treemap visualization to mine their assessment data for useful information.
11. Point teachers to consider how they might capitalize on the strengths and remediate the weaknesses that they've identified in the coming school year. What part of the curriculum might they emphasize, deemphasize, reorder, etc.

12. Pass out assessment sheet and solicit suggestions and comments regarding this teacher development event. See appendix for copy of assessment form.

## Evaluation

In order to effectively evaluate the usefulness of Treemap visualizations of student assessment data, as well as the long term impact on behavior as a result of the training event, the evaluation process will contain three parts:

1. A pre-training assessment where teachers are asked to report information about how they use student test data.
2. A post-training assessment that asks teachers to report subjective experiences with Treemaps (how confident are they about using one, understanding one, etc.). In addition, teachers will be asked how likely they are to use Treemaps to help understand student test data in the future, and ultimately whether they will use their insights to shape curricular decisions.
3. Finally, teachers will be asked a follow-up questionnaire that focuses on real behavior over the past two months – how often have they consulted their classroom Treemap, and what problems and successes have they experienced.

All three of these assessments/questionnaires are in the Appendix.

## Related Activities

There is extensive student test data documented on the “OARS” (online assessment reporting system) system in SFUSD. Test data include literacy, science, CAT 6, and several additional categories of data. A related activity would help teachers access and begin to make use of some of this data in designing their year-long curriculum plans.

## Ability to Replicate

Although creating Treemaps from student test data requires detailed knowledge of a unique application (or scripting language), motivated teachers and district professionals may receive training in creating a Treemap “View” from “The Hive Group” – a for-profit organization that creates the commercial Treemap software that Intel uses. Alternatively, teachers can access the University of Maryland, Human-Computer Interaction lab’s open-source software which is well-documented, and surprisingly easy to use.

Finally, it is my goal to focus the attention of district leaders on the capabilities of Treemaps with regards to helping teachers understand and act upon the vast quantity of data they regularly create.

# Treemap Scavenger Hunt

Be the first to answer the following questions ...  
correctly ... and win a valuable prize!



## The Wikipedia Country Population Treemap

1. *What country has the densest population?*
2. *What is the least dense country in the Asian Continent?*
3. *How does the population density compare in the two most populous countries?*

## iTunes Top 100 Treemap

1. *What is the newest song to make it on the top 100? How old is it?*
2. *What is the oldest song with an upward trend in popularity?*
3. *Name all the songs that are trending down in popularity that are in the bottom 10% of popularity? For example, find a song in the 90...100 range of popularity that is going down in popularity.*

## Make up your own Brain Teasers

1. \_\_\_\_\_  
\_\_\_\_\_  
Answer: \_\_\_\_\_
2. \_\_\_\_\_  
\_\_\_\_\_  
Answer: \_\_\_\_\_

**Pre-Training Assessment**

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I have a plan for storing and retrieving my students' test data.					
2. I regularly examine the past <b>week</b> of student test data.					
3. I regularly look at the past <b>month</b> of student test data.					
4. I regularly look at <b>all</b> my students' test data.					
5. I intend / would like to use test data to help me plan my curriculum.					
6. I regularly use test data to help me plan my curriculum.					
7. Students are tested too much.					

## Post-Training Assessment

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I feel confident in my ability to use Treemaps to search for information.					
2. I understand how to use Filters, Groups, and Text Size Controls to discover trends and patterns in data.					
3. I feel comfortable helping other teachers learn how to use Treemaps.					
4. There is a strong likelihood that I'll use my class Treemap to view my students' assessment data.					
5. I will regularly use my class Treemap to look for trends and patterns in how my students are mastering the curriculum.					
6. This training event was a good use of my time.					

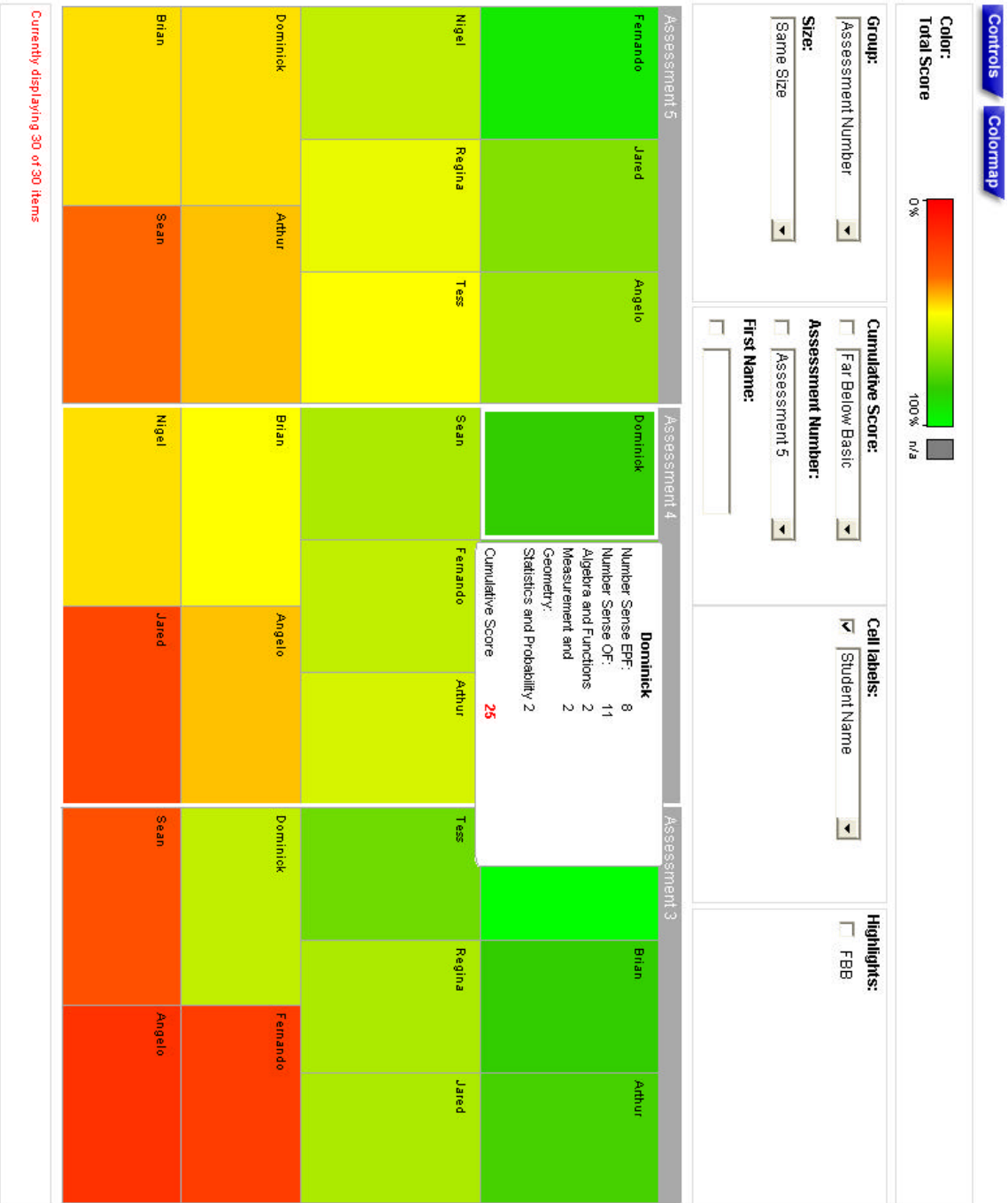
**Comments:**

## Follow-up Assessment

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. Over the past month, I've used my class Treemap to view my student assessment data.					
2. I feel comfortable using my class Treemap to discern trends and patterns in how well my class is mastering the curriculum.					
3. I have developed new strategies for interpreting data in my class Treemap.					
4. I use the insights gained from viewing my class Treemap to help guide and prioritize curriculum.					
5. Treemaps are having a long-term effect on how I interpret and act on class assessment data.					

Comments:

# Sample Treemap of Student Math Assessment Data



## How to build a Treemap using an XML transformation

*Author: Brendan Creane*

*Date: July 28, 2005*

*LastModified: July 29, 2005*

### Introduction

This document describes how to create a Treemap (a data visualization tool) by customizing an XML source document. The XML source contains about 900 lines, but you'll only need to set up about a dozen lines to get a basic Treemap working.

There are two main files that you'll use: your XML source file and the HTML output file. Think of the XML source file as a human-readable description of a Treemap, and the HTML file as the "machine code" that creates the Treemap. The XML source file is transformed into an HTML file using a collection of XSLT (XML transformation) files. These XSLT files tell an XML parser how to generate the `<applet>` and `<param>` values that the Honeycomb Treemap server understands.

The intention of this project is to facilitate rapid development of Treemaps. The alternative is to use the Honeycomb Designer; this is a java program with a WYSIWIG interface. Using the designer is time-intensive, but mostly foolproof. I recommend using the Designer once or twice to get a feel for the overall process of creating a Treemap.

If you have questions about what the effect of a particular node (e.g. `<font_color>`) does, there are extensive comments within the XML source file. In addition, each of the XSLT files is associated with a specific applet. For example, the file "applet\_group\_control.xml" has code associated with the group control. These XSLT files also contain extensive comments.

Finally, if there is some behavior in the Treemap which is unexpected, you may consult the "Honeycomb 4.0 WebSDK User's Manual." This documents the syntax and semantics of all the several hundred `<param>` values.

The easiest way to use the XML source is to just open up the file in Internet Explorer 6. Alternatively, you can use an XML IDE to apply the XSLT transformation to your source file. This latter method lets you see the raw `<param>` statements that come from the XML source.

### Setup your Development Environment

1. Copy the following files into a directory under your webserver's root folder (e.g. "C:\Inetpub\wwwroot\treemap\`</code>")
  - a. applet_cell_label_control.xml
  - b. applet_color.xml
  - c. applet_field.xml
  - d. applet_filters.xml`

- e. applet\_group\_control.xml
  - f. applet\_indicator\_control.xml
  - g. applet\_size\_control.xml
  - h. applet\_status\_control.xml
  - i. generic\_attributes.xml
  - j. generic\_parameters.xml
  - k. test\_map01.xml
  - l. test\_map\_data.csv
2. Copy the Honeycomb jar files to a directory under your webserver. The default location is “/Honeycomb/applet” For example: “C:\Inetpub\wwwroot\Honeycomb\applet. The jar files are:
    - a. Honeycomb.jar
    - b. Loader.jar
  3. Copy “test\_map01.xml” to another xml file (e.g. “my\_treemap.xml). This file will serve as your XML source. The remainder of the instructions assumes you are editing your XML source file.
  4. Confirm that you can view your XML source file through IE6. For example, point your browser to [http://localhost/treemap/my\\_treemap.xml](http://localhost/treemap/my_treemap.xml) You should see a reasonably complete Treemap with Group, Size, Filter, Indicator, Status, and Field applets.
  5. One strategy for ensuring your XML is well-formed (at least until there’s an XML Schema in place) is to continually reload your browser and make sure that there aren’t any XML syntax bugs.

### Setup Default Values for Applets

1. There are two nodes that you may need to change under the “default\_values” hc\_applet:
  - a. <applet\_group\_id> -- This **MUST** be unique for a view. The id is the glue that holds the half dozen HC applets together ... and it distinguishes one view from another view.
  - b. <code\_base> -- Specifies the path to the Honeycomb applets. If you want to store the applets with the “HTML” file, just set this to “.”

### Setup your Data Source

1. Decide where you’re going to pull the CSV (comma separated values) data from. This can be a URL accessed through http://, or it can be a relative pathname (relative to the htm file). Modify the <data>/<data\_source> node to reflect the location of the CSV file.

2. Create as many <field> nodes as you need (one for each column in your CSV file) and set <field\_name> to the name of the field (no whitespace in <field\_name>.) Also set <field @type> to “numeric” for numbers (floats, integers), or “string.”
3. Set up one or more <color\_map> nodes under a <field> node. The comments should guide you. Basically you decide if you want a continuous (range) or discrete (enum) color\_map. Then set up two or more <color\_point> nodes.

### Setup your Groups

1. A group is a way to aggregate your cells according to one of your data fields. So if you have an “area” field with the values: “heaven, earth, walmart” then your Treemap will group the cells into those three groups.
2. If you want to skip this for now, just leave the “GROUP ... NONE” group under the <groups> node.
3. Otherwise, create a <group> node and fill out the label, field\_name, and menu\_label nodes.
4. Create one or more <group\_specs> . These are conditions that must be met for a particular group to be displayed.
5. If you want to create a <sub\_group> (a group within a group), put the <group\_label> from one group into the <sub\_group> of another ... if that’s clear.

### Test your Treemap

1. At this point, you should have a basic (and functioning) Treemap. Point your browser to your XML file and confirm you have a “field” applet. For example: [http://localhost/treemap/my\\_treemap.xml](http://localhost/treemap/my_treemap.xml)
2. Common problems include a poorly formed <color\_map> node, and mistyped <field\_name> nodes.
3. Get your groups and colors working before you add a size filter, indicators, filters, mouseclicks, or text on cells.

### Setup your Sizes

1. The size control lets you change how cells are sized. You can specify one (and one only) default\_size. For testing, just leave the “Same Size” node in place. Add sizes one by one, testing each one.
2. An easy mistake here is to copy and paste the first size node – then you’ll have TWO “default\_size” attributes set to ‘true’. Remember, only one “default\_size” attribute can be ‘true’. This is true for other controls (e.g. group) as well. The XSLT transform will catch it, but you may not get a helpful error message from IE6.

### Setup your Filters

1. Filters let you set up conditions which restrict the number of cells you view at one time. The XML code is lengthy for a filter, but don't be intimidated, it has a simple syntax.
2. Decide what type of filter you'll have. Examples include a check box + drop down menu, or a check box + text filter.
3. There are samples for both these types in the sample XML code. Basically, you'll set up one or more <spec> nodes. These are like "if" statements in a programming language.
4. Remember to keep testing your code.

### Setup your Text on Cells

1. This section specifies what data is displayed on each cell. You can only have one data field displayed at a time, but using the "Text on Cell" control, you can change which field is shown.
2. Go to the "TEXT on CELLS Applet" code. You'll create one or more <option> nodes with a menu\_label and a field\_name. Note that the <field\_name> node are the same as you specified up in the <data> node.
3. Remember to make exactly one <option> node be the default option (@default\_option='true').

### Setup your Mouseover Box

1. The Mouseover box is a little window that hovers near the mouse as you move over the data cells. You can put <field\_name> values and labels, as well as an image into the Mouseover box.
2. Add a <line> node, specify a <mo\_label>, and then put in a <field\_name> node. Because of limitations with XSLT, you **MUST** specify the @type attribute for each <field\_name>. This can be "string," "numeric," or "image."
3. If you want to add an image to your Mouseover Box, do the following:
  - a. Add a column in your CSV file with the URL of the image associated with each row in your CSV file. The <field\_name> associated with this column **MUST** start with "img" For example: "img\_ImageURL"
  - b. Set up the <mouseover>/<default\_image> node with the maximum size in pixels of your images. You can also specify a "now loading ..." image to keep your visitors busy while the browser fetches the real image.

### Setup your Mouseclick Menu

1. The Mouse click menu provides additional lines that show up in your mouseover box when you click on the box. Use this for default Honeycomb actions (e.g. Zoom), and also to put a hyperlink to something associated with this cell.

2. Go to the <mouseclicks> section of your XML file.
3. Each <line\_mc>node contains a label (<menu\_item>) and an action (<action>). The <action> node **MUST** have a @type attribute set to either “builtin” or “url”.
4. In order to pass the value of a field to a web page, embed your field\_name in the url and bracket the field\_name with curly braces. For example:  
[http://imo-trees/cool\\_stuff.aspx?Entity={fd\\_Entity}](http://imo-trees/cool_stuff.aspx?Entity={fd_Entity})
5. If you need to use a reserved character in your url (e.g. &, <, >, ‘, “), then use the escape characters that are documented in the XML file.

### Setup your Indicators

1. Indicators highlight cells that meet certain conditions. HC places an icon inside each cell that matches the indicator’s condition.
2. The syntax is almost identical to that of <filters> with the exception that you specify an <icon\_path>. An <icon\_path> is just a url for an image (e.g. .png, or .gif).

### Setup your Sliders

1. Sliders are a different control interface for filters and indicators. You can have multiple sliders, each associated with a particular filter or indicator. The one tricky setting is the <filter\_or\_indicator\_group> node. Each slider must be associated with either a filter or an indicator. So the format for <filter\_or\_indicator\_group> is: [filter|indicator][1..n]. Some examples:
  - a. 1<sup>st</sup> indicator: <filter\_or\_indicator\_group>indicator1<filter\_or\_indicator\_group>
  - b. 3<sup>rd</sup> filter: <filter\_or\_indicator\_group>filter3<filter\_or\_indicator\_group>

### Customize the size and location of your applets

1. You can play with the <width>, <height>, <title>, <control\_position>, and labels inside each of the groups to change the characteristics of each applet.
2. If you want to change the actual layout (the HTML structure that contains each of the applets), modify “main.xsl”. Right now the applets are inside a simple HTML table.