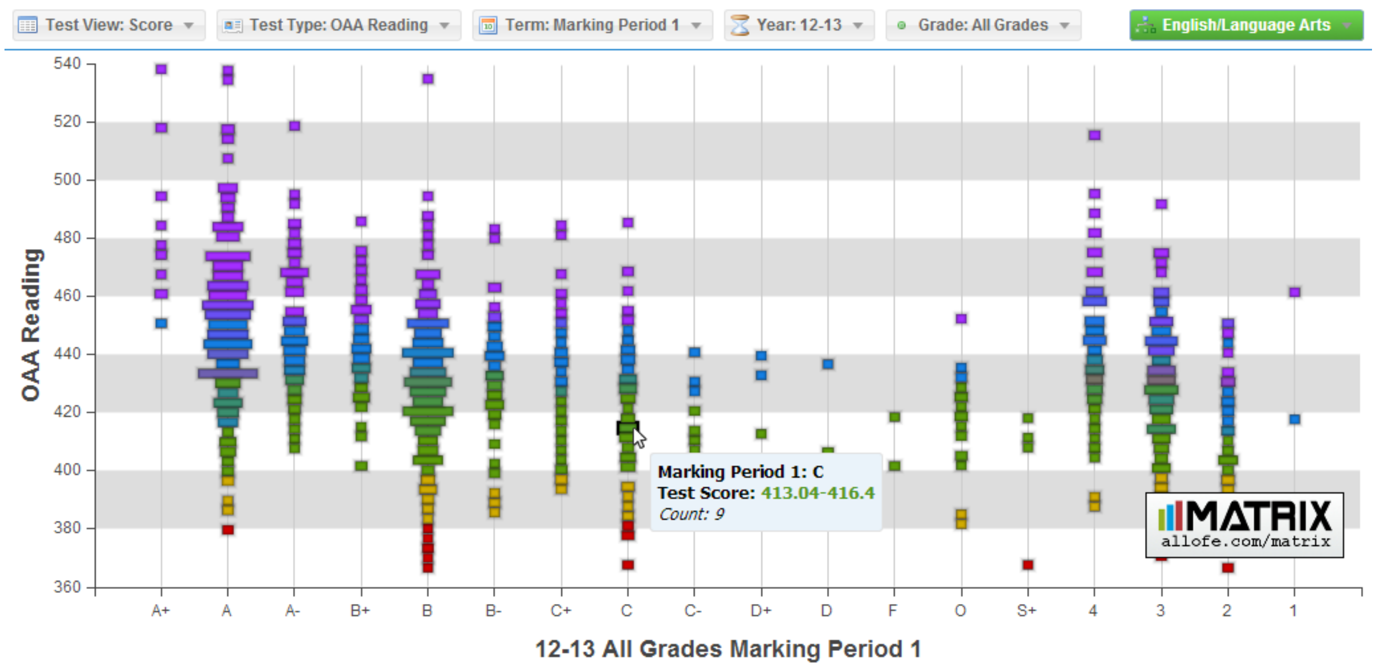


The Problem

One of our clients came to us with a problem: they suspected a correlation between letter grades received in the classroom and standardized test performance, but they had no way of identifying which tests had a strong correlation and which had a weak correlation. We needed a report that would make it easy to qualitatively show how many students were doing well in their classes but poorly on tests, or vice versa. All of the data we needed was already in Matrix so we set out to create a report which would display that data in an intuitive and visually striking way.

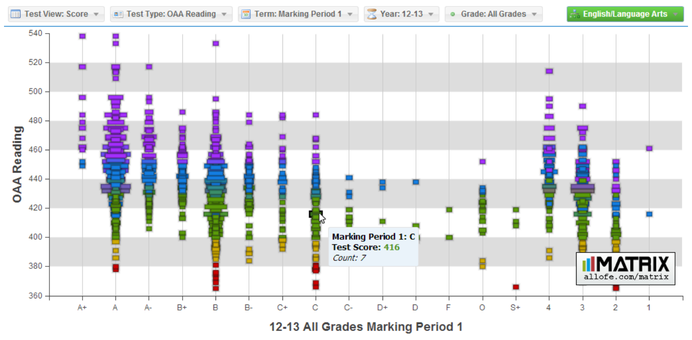
The Solution

Matrix is all about making information as beautiful as it is useful and to do that we rely on charts built with ExtJS, a powerful JavaScript framework for web applications. For this particular solution a basic scatterplot could not demonstrate the relationship between classroom grades and standardized test performance in the way our client needed. The missing piece was a way to visualize the number of scores that fell at every intersection of a test score and letter grade. To get there, we had to push the chart's behavior a bit: we tweaked the Scatter series so the width of each chart marker is proportional to the number of scores it represents.
We think it turned out really well. Let us know what you think, in the comments or on Twitter! (@allofek12)

(If you want to compare an early version of the chart to the final, or you're curious how exactly we modified the ExtJS Scatter series, there are detailed samples after the jump.)

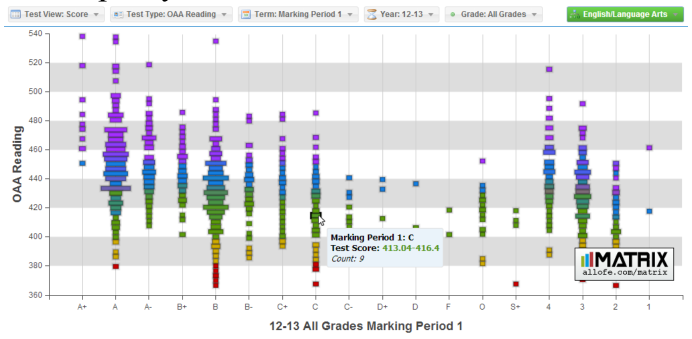
Our First Try

The first version of this chart had one chart point for every combination of test score and letter grade, but some of the tests have such large score ranges that the markers wound up overlapping significantly. It's an interesting visual effect, but it makes it hard for a user to locate a particular marker to get more specific information.


Test View: Score | Test Type: OAA Reading | Term: Marking Period 1 | Year: 12-13 | Grade: All Grades | English/Language Arts

Marking Period 1: C
Test Score: 416
Count: 7

In order to keep markers from overlapping, we have the server group similar test scores into "buckets" before returning data to the chart's proxy.


Test View: Score | Test Type: OAA Reading | Term: Marking Period 1 | Year: 12-13 | Grade: All Grades | English/Language Arts

Marking Period 1: C
Test Score: 413.04-416.4
Count: 9

The Technical Part

The Data

For a given set of filters, the provider will return a JSON-formatted object that looks like this. Note `letter_map`, which we use to generate a renderer for the bottom axis that maps `letter_score_num` to normal letter grades.

```
{
    "letter_map": ["A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-", "D+", "D", "F", "O",
         // All the letter grades in this set, in sort order.
    "rows": [{ // Each of these corresponds to one point on the chart.
        "year": "12-13",
        "end_year_int": 2013,
        "letter_score": "2",
        "letter_score_num": 17,
        "test_score": 433.88, // Midpoint of the bucket, used to place the point.
        "test_score_min": 432.2, // These define the size of the bucket.
        "test_score_max": 435.56,
        "name": "Marking Period 1",
        "count": 3,
        "rows": [/* These inner rows are used to generate aggregate data about a point, an
    },/* More rows... */],
    "total": 320
}
```

The Series Config

This is an excerpt from the chart config object, defining our `scatter` series. Note that we overrode `isItemInPoint` to account for the variable width of the markers.

```
series: [
    {
        type: 'scatter',
        axis: ['left','bottom'],
        xField: 'letter_score_num',
        yField: 'test_score',
        markerConfig: { type: 'square' },
        renderer: classroomGradesCorrelationScatterRenderer,
        highlight: { stroke: 'black' },
        isItemInPoint: function (x, y, item) {
            var point = item.point,
                xTolerance = item.sprite.attr.width / 2,
                yTolerance = item.sprite.attr.height / 2;
            return (
                point[0] - xTolerance <= x &&
                point[0] + xTolerance >= x &&
                point[1] - yTolerance <= y &&
                point[1] + yTolerance >= y
            );
        },
        tips:{
            constrainPosition:true,
            trackMouse:true,
            dismissDelay: 0,
            renderer: classroomGradesCorrelationTipRenderer
        }
    }
]
```

The Series Renderer

This is where the work of setting the markers' width is done. It also sets their fill colors based on the test achievement level.

```
function classroomGradesCorrelationScatterRenderer(s,r,a,i,st) {
    var chart            = Ext.getCmp(s.surface.el.up('div.x-surface').id),
        gradeAxis        = chart.axes.getByKey('bottom'),
        letterScoreNum   = r.get('letter_score_num'),
        testScore        = r.get('test_score'),
        testAchievements = Ext.Array.pluck(r.get('rows'), 'test_achievement'),
        testTypeId       = st.proxy.extraParams.test_type_ids.split(',')[0],
        axisWidth        = gradeAxis.axisBBox.width,
        maxCount         = st.max('count'),
        letterCount      = st.proxy.reader.rawData.letter_map.length,
        maxWidth         = Math.floor(axisWidth/(letterCount + 1)) - 12,
        aColor;

    // First determine the sprite's color. (See averageColor below.)
    aColor = averageColor(
        Ext.Array.filter(
            Ext.Array.map(
                testAchievements,
                function(a) { return top.TEST_TYPE_ID_X_ACHIEVEMENT_X_COLOR[testTypeId][a]
            ),
            Ext.identityFn
        )
    );
    if (aColor) a.fill = '#' + aColor;

    // Next determine the sprite's width.
    // A count of zero would be 6px wide, and the point(s) with the highest count will tak
    a.width = 6 + maxWidth * r.get('count') / maxCount;

    // Move the sprite over so that it remains centered in its column.
    if (a.hidden !== false) {
        // We test a.hidden because the renderer gets called for marker shadows too, which
        //   inherited attributes from their parent markers, and we don't want to translat
        a.translate.x = 3 + a.translate.x - a.width/2;
    }
    return a;
}
```

Bonus Chart Hacking: The Color Averager

Colors are a function of achievement level, and achievement level is a function of both score and grade level. This chart can display data from multiple grade levels, and a score that's Advanced for a third grader might only be Proficient for a fourth grader, so two students with the same score may have different achievement levels. The `averageColor` function produces a color that represents a blend of the different achievement level colors for a point.

```
function averageColor(colorArr) {
    var r = [],
        g = [],
        b = [],
        a = [];

    // First split the individual colors into their red, green, and blue components.
    Ext.each(colorArr, function(c) {
        r.push(parseInt(c.substring(0,2),16));
        g.push(parseInt(c.substring(2,4),16));
        b.push(parseInt(c.substring(4,6),16));
    });

    // The average color will be composed of the mean red value, the mean green value, and
    a.push(Ext.String.leftPad(Math.round(Ext.Array.mean(r)).toString(16), 2, '0'));
    a.push(Ext.String.leftPad(Math.round(Ext.Array.mean(g)).toString(16), 2, '0'));
    a.push(Ext.String.leftPad(Math.round(Ext.Array.mean(b)).toString(16), 2, '0'));

    return a.join('');
}
```

Any Questions?

We want to hear from you! If you've got questions or comments about this post, sound off in the comments or drop us a line on Twitter (@allofek12).