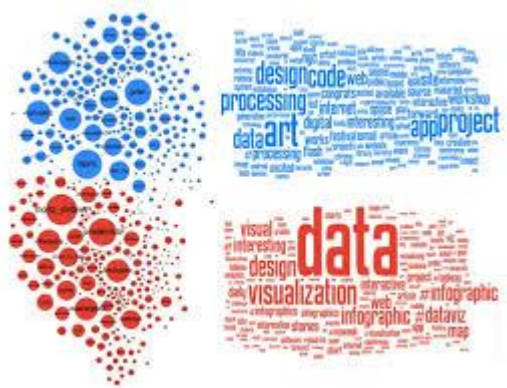
1010DATA

INSIGHT SUMMIT

Intro to Data Visualization for MBS Users



Adam Rothschild

May 19, 2016

Goals of this presentation

- Introduce and/or Review Features available to you as 1010data Users to Visualize your data and get started building Applications
- We will touch on:
  - Blocks and Parameters
  - Charting
  - Building a Simple Charting Quick App from Scratch
  - Technique to Produce Task Driven Number of Charts
  - Dropdown Widgets with product intelligence
  - Several Flavors of the gmap widget

Example 1:
An Easy way to Get Started with Quick Apps

- Basic CPR Query
- Create a Chart on the System
- Drop as Quick App (The system will convert this Chart to A Quick App for Us)
- Convert the CPR XML to a Block
- Add Features to the Quick APP

Example 1:
An Easy way to Get Started with Quick Apps

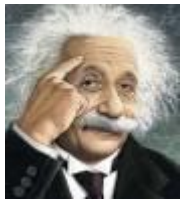
- New Tags As a result of Creating A Chart
  - <dynamic>: The Quick App Wrapper
  - <widget>: A graphical object similar to other Graphics Libraries
  - <graphspec>: Properties pertaining to the Charting Widget
- XML Code INTERNAL TO WIDGET!
- Well Documented in Reference

  https://www2.1010data.com/documentationcenter/beta/1010dataReferenceManual/1010dataReferenceManual.pdf

Review of Chart Example:

- Wrapped XML in <block> tags and replaced with an <insert>
- Added coupon,vintage,product parameters to the dynamic and block
- Added <if> logic to block to bypass NA inputs
- Added _require and _invmode options to the chart to prevent the code from running and the chart from rendering until we added the parameters
- Added a more informative title
- Enabled tooltips to allow drilldown on each point in the chart
- Add a submit button, named the chart widget and set update to manual

Task Driven #of Charts



- Most of the time a single chart will not suffice. What if you need to produce many charts quickly and you don't know exactly how many charts you will need to generate
- Provide a block to determine the number of charts we will require and the inputs to each chart before actually generating them using <do>
- Use a <for> loop to generate the charts
- Using class_= "nest" widget enables generating an arbitrary number of widgets

Generate a CPR chart for every GNMA[2] Cohort with an arbitrary high balance threshold

- Input generation block
  - Select GNM30 and GNMII30 product
  - Tabulate balance by product/coupon/productionyear
  - Find the high balance for each cohort

```
<sel value="g_hi(product coupon productionyear;;balance)>{@threshold}"/>
```

  - Compare against threshold
  - Create a string of inputs using g_splice

```
<willbe name="cpn_vintage" value="splice(product coupon productionyear;'_')"/>
<willbe name="all_combos" value="g_splice(;;;cpn_vintage;',';300)"/>
```

  - Use a <do> to pass inputs to the CPR/Chart routine

```
<do on_="init" value1_="@combo" row1_="1" col1_="1">
  <insert block="generate_cohorts" threshold="10000000000"/>
</do>
```

- Use a <foreach> to process each cohort from the cohort string produced by g_splice then decompose the cohort to get the product,coupon and productionyear

```
<foreach combo="{@combo}" tally_="k">
  <set prod="{strtake(@combo;'_';1)}"/>
  <set yr="{strtake(@combo;'_';-1)}"/>
  <set cpn="{strtake(strtake(@combo;'_';2);'_';-1)}"/>
  <set y="{425 * int((@k-1)/5)}"/>
  <set x="{625*mod(@k-1;5)}"/>
  <widget class_="graphics" width_="600" height_="400" cpn="
{@cpn}" vintage="{@yr}" product="{@prod}" tooltips_="1" abspos_="
{@x},{@y}">
```

- Insert the cohort parameters into a CPR block

```
<insert block="cpr" vintage="{@vintage}" coupon="{@cpn}" product="{@product}"/>
```

- Use the chart widget defined earlier using 1 Y-Variable
- Download the Charts to XLSX if desired

Dropdown widgets without product intelligence

- Basic Dropdown with no Product insight

```
<dynamic agency="1">
  <widget class_="dropdown" label_="agency" value_="@agency">
    <table>1,FHLMC;2,GNMA;3,FNMA
    </table>
  </widget>
  <widget class_="text" text_="{@agency}" label_="OUTPUT:"/>
</dynamic>
```

agency | FHLMC | OUTPUT: 1

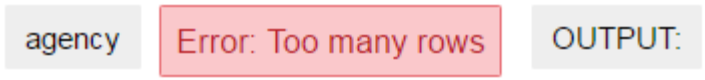
Lets implement a Dropdown for Non Agency Cusips

- Seems simple enough:

```
<dynamic pool_id="">
  <widget class_="dropdown" label_="agency" value_="@pool_id"
base_="pub.fin.lpmisc.poolcusip">
    <colord cols="pool_id,cusip"/>
  </widget>
  <widget class_="text" text_="{@pool_id}" label_="OUTPUT:"/>
</dynamic>
```

- There are 190K Cusips is there anything more useless then a dropdown with that many elements?

- Try Again



- Use a dropdownlist (Can select Multiple Inputs)
- Set serverfilter=1 (Enables the list of Cusips to be filtered before overwhelming the widget
- Add a beginswith filter
- Merge in another list if so desired(pool names in this instance)

```
<widget class_="dropdownlist" serverfilter_="1"
casesensitive_="0" filter_="beginswith" numhints_="500"
value_="@pool" label_="LOOKUP" width_="250" dropdownwidth_="250"
textcolor_="blue" margin_="20">
        <base table="pub.fin.lp.poollast"/>
        <colord cols="pool_id,name"/>
        <merge table2="pub.fin.lpmisc.poolcusip" match="order">
          <link table2="pub.fin.lp.poollast" col="pool_id"
col2="pool_id" cols="name"/>
          <willbe name="c2" value="splice(cusip pool_name;' ')"/>
          <colord cols="pool_id,c2"/>
        </merge>
        <sort col="name" dir="up"/>
      </widget>
```

- Once a Cusip is selected the Dropdown provides the associated pool_id
- Use that pool_id to generate a dropdown with Applicable dates by selecting that pool in the time series and tabulating on fmonth1

```
      <widget class_="dropdown" base_="pub.fin.lp.deep2"
label_="FMONTH" value_="@fdate" require_="{@pool<>NA}" margin_="20"
invmode_="hide">
        <link table2="pub.fin.lp.master" col="loan_nid"
col2="loan_nid" label="Master" cols="pool_id,deal_no"/>
        <link table2="pub.fin.lp.poollast" col="pool_id"
col2="pool_id"/>
        <sel value="pool_id = '{@pool}'"/>
        <tabu label="Tabulation on Loan History + Masters"
breaks="fmonth1">
          <tcol source="fmonth1" fun="cnt" label="Count"/>
        </tabu>
        <colord cols="fmonth1"/>
      </widget>
```

- Embed the same query which select by pool_id and fmonth1 into a grid widget and the Super Slick Excel button widget:

```
      <widget name="excel" class_="button" type_="export"
target_="xlsx" text_="Excel" require_="{@pool<>NA & @fdate <> NA}"
nvmode_="hide" base_="pub.fin.lp.deep2">
        <link table2="pub.fin.lp.master" col2="loan_nid"
ol="loan_nid"/>
        <link table2="pub.fin.lp.loanlast" col2="loan_nid"
ol="loan_nid" suffix="_sum"/>
        <link table2="pub.fin.lp.poollast" col2="pool_id"
ol="pool_id"/>
        <willbe name="pool_close_bal" value="c4"
abel="Pool`Closing`Balance"/>
        <sel value="pool_id='{@pool}'"/>
        <sel value="fmonth1={@fdate}"/>
      </widget>
```

The GMAP widget

Simplest example from the Reference manual

- Uses a Package to specify Latitude Longitude and Zoom
- Input for package to gmap is pzvalue_

```
<dynamic pzpackage="{pkg('lat' 'lng' 'zoom' 'nelat' 'nelng' 'swlat'
'swlng'; 40.753622 -73.972841 8 '' '' '' '')}">
```

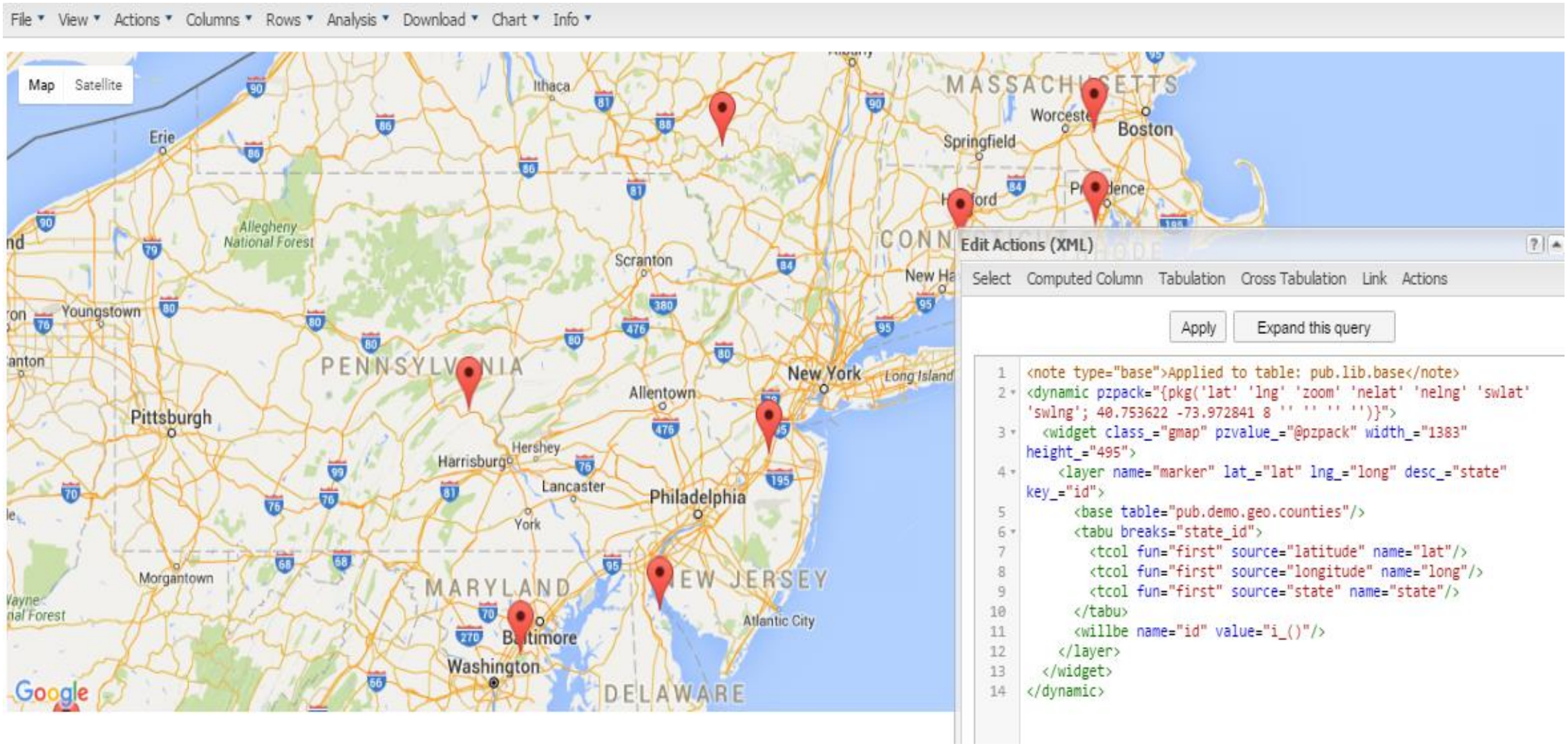
```
pzvalue_="@pzpackage"
```

Add a layer to the map widget

```
 <layer name="marker" lat_="lat" lng_="long" desc_="state"
_="id">
   <base table="pub.demo.geo.counties"/>
   <tabu breaks="state_id">
     <tcol fun="first" source="latitude" name="lat"/>
     <tcol fun="first" source="longitude" name="long"/>
     <tcol fun="first" source="state" name="state"/>
   </tabu>
   <willbe name="id" value="i_()"/>
 </layer>
```

- Layers can be used for heat maps or setting markers on the map
- Require Polygons or at a minimum Latitude and Longitude for each marker
- Polygon tables for state,cbsa,county/fips and zipcode are available on 1010

Gmap widget with <layer> to create state level markers



State Marker Data

Tabulation

Columns 1-5 of 5, Rows 1-17 of 50

state_id state_id	First latitude lat	First longitude long	First state state	id id
	45.7326	-93.9196		
MN	45.7326	-93.9196	Minnesota	1
WA	47.3917	-121.5708	Washington	2
ID	44.2394	-114.5103	Idaho	3
MT	46.9048	-110.3261	Montana	4
ND	47.5362	-99.793	North Dakota	5
ME	44.6074	-69.3977	Maine	6
WI	44.2563	-89.6385	Wisconsin	7
OR	44.5672	-122.1269	Oregon	8
SD	44.2853	-99.4632	South Dakota	9
NH	43.4108	-71.5653	New Hampshire	10
VT	44.0407	-72.7093	Vermont	11
NY	42.1497	-74.9384	New York	12
WY	42.7475	-107.2085	Wyoming	13
IA	42.0046	-93.214	Iowa	14
NE	41.1289	-98.2883	Nebraska	15
MA	42.2373	-71.5314	Massachusetts	16
IL	40.3363	-89.0022	Illinois	17

Loss Severity Heat Map Methodology

- Uses the Zip Code data from the CoreLogic Securities data set to map the loan to the appropriate county/state
- Incorporate a Severity block which computes the severity of loans on a given date and rolls this up to the state or county level
- Use the average and standard deviation of the loan level severity for each region as a weight value for each point in the <layer> to control the color
- Use a <do> to capture screen clicks and provide the location back to the application. This will trigger a popup which provides additional aggregate data for the region

Basics of Building a Heat Map Layer

```
    <layer name="polygon" lat_="{if(@pzvalue.zoom>7;'lat';'c3')}" lng_="
{if(@pzvalue.zoom>7;'long';'c2')}" groupby_="{if(@pzvalue.zoom>7;'fipscounty';'c1')}"
weight_="weight">
      <if test="{@pzvalue.zoom>7}">
        <then>
          <base table="x1010data.rothschilda.map_demo.county_polygons"/>
          <link table2="pub.lib.base" col="fipscounty" col2="fipsc">
            <insert block="severity" filters="" type="county"/>
            <sort col="t0" dir="down"/>
            <willbe name="std" value="g_std(;;t0)"/>
            <willbe name="avg" value="g_avg(;;t0)"/>
            <willbe name="upper" value="avg + 2*std"/>
            <willbe name="lower" value="avg - 2*std"/>
            <willbe name="weight" value="if(t0>=upper;1;t0<=lower;0;(t0-lower)/(upper-
lower))"/>
            <colord hide="std,avg,upper,lower"/>
            <sort col="county_fips_m" dir="up"/>
            <willbe name="fipsc" value="string(int(county_fips_m))"/>
          </link>
          <sel value="between(lat;{@pzvalue.swlat};{@pzvalue.nelat})"/>
          <sel value="between(long;{@pzvalue.swlng};{@pzvalue.nelng})"/>
        </then>
        <else>
          <base table="x1010data.rothschilda.map_demo.state_polygons"/>
          <link table2="pub.lib.base" col="c1" col2="state">
            <insert block="severity" filters="" type="state"/>
            <sort col="t0" dir="down"/>
            <willbe name="std" value="g_std(;;t0)"/>
            <willbe name="avg" value="g_avg(;;t0)"/>
            <willbe name="upper" value="avg + 2*std"/>
            <willbe name="lower" value="avg - 2*std"/>
            <willbe name="weight" value="if(t0>=upper;1;t0<=lower;0;(t0-lower)/(upper-
lower))"/>
            <colord hide="std,avg,upper,lower"/>
          </link>
        </else>
      </if>
    </layer>
```

- Based on the level of Zoom the layer will alternate between state and county polygons
- The County level layer select only latitude and longitude that is currently rendered
- Weight which is a gauge of the severity level is mapped to color

1010DATA

Revisit the Sub Prime Melt down at various points in time

- Use the fdate and cusip dropdowns created earlier
- Adds a county lookup dropdown
- Each Marker tags the delinquency status of a loan on the specified date
- Use the loan zip code directly <linked> to the zip code polygon table. Takes the first value
- Use class_=“nest” to support an arbitray number of charts

Give thanks where thanks is due

- Many thanks for attending this presentation. I urge everyone to get their feet wet if you haven't already done so. Remember we are here to assist you!
- Many thanks to my colleagues at 1010 that assisted me either directly or indirectly in this presentation.