Liquid Diagrams: Information Visualisation Gadgets

Keith Andrews and Martin Lessacher
IICM, Graz University of Technology, Austria
kandrews@iicm.edu

Abstract

Information visualisation techniques have sometimes been slow to diffuse into more widespread public use. Recent advances in cloud computing have opened up opportunities to bring information visualisation to the masses in ways previously not possible.

Liquid diagrams are a suite of information visualisation gadgets written in Flex, which visualise live data contained in Google Docs spreadsheets through the Google Visualization API. Users can interactively configure the visualisation and any changes in the online spreadsheet data are reflected immediately in the display. In contrast to other solutions, liquid diagrams gadgets specifically support the printing and export of both vector (SVG) and raster (PNG) graphics versions of the visualisations, allowing users to construct and export high-quality diagrams for inclusion into other works.

The suite of visualisation gadgets currently available includes: area charts, bar charts, heat maps (choropleths), line charts, pie charts, treemaps, and parallel coordinates plots. Star plots and voronoi treemaps are coming soon.

1. Introduction

Information visualisation techniques have sometimes been slow to diffuse into more widespread public use. Information visualisation tools designed for desktop use typically have to be installed on the computers of end users, with all the associated issues of compatibility and maintentance, not to mention the initial hurdle of an installation dialogue. Recent advances in cloud computing have opened up opportunities to bring information visualisation to the masses in ways previously not possible. Google Docs, the online office suite by Google, had over 10 million unique visitors in January 2010, according to Compete [2].

Web sites allowing users to visualise their own data, such as IBM's Many Eyes [14] and Swivel [12] have shown the potential of online visualisation services. Google Docs promises a rapidly increasing user base and the new Google Visualization API [5] allows visualisations to be built which integrate seamlessly with the data contained in Google spreadsheets.

2. Google Spreadsheets Gadgets

Google spreadsheets can be linked to either static charts (Insert Chart) or interactive gadgets (Insert Gadget). Gadgets are specified by the URI of an XML file, which conforms to the Google Visualization API. The spreadsheet data and any parameters (settings) are basically passed to the gadget via Javascript, and the gadget itself can be built using purely Javascript or a mixture of Javascript and other technologies such as Flash or Flex.

Gadgets can be submitted to Google for verifcation, at which point they are listed in the "Add a Gadget" dialogue, and can be added to a spreadsheet without having to manually enter the URI of an XML file. Gadgets typically expect the spreadsheets which use them to conform to certain conventions. For example, a parallel coordinates gadget operating on multidimensional data might expect the spreadsheet rows to contain data items and each column to be a dimension (variable), with an optional first row containing lables. Thus each visualisation gadget requires some associated documentation of its settings and input data conventions.

The procedure for inserting a new gadget is illustrated in Figures 1 to 4. First of all, the user prepares their data in a Google Spreadsheet (Figure 1). Depending on the particular kind of gadget to be inserted, the data may have to conform to certain conventions. For example, a treemap gadget may require the first column to contain the name of an item, the second column to contain the name of its parent, and further columns to contain any data.

Next, the appropriate rows and columns in the spreadsheet are selected and the function Insert Gadget is called. This brings up the Add a Gadget panel (Figure 2), where a specific gadget can be selected. Those gadgets which have been provided by Google or verified by Google can simply be selected from the list. As yet unverified gadgets have to be inserted as custom gadgets. These are specified by en-

1550-6037/10 $26.00 © 2010 IEEE
DOI 10.1109/IV.2010.100

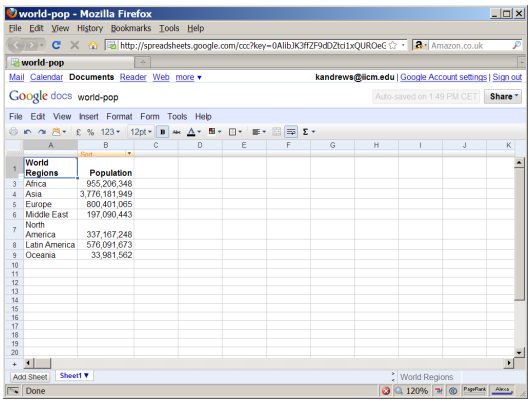


Figure 1: The user has entered some data about world populations into a Google spreadsheet. The data is selected and Insert Gadget is clicked (not shown).

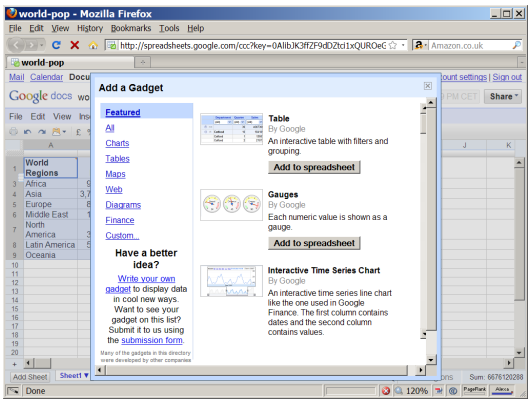


Figure 2: The Add a Gadget panel appears, listing available gadgets which have been verified by Google. Custom gadgets are specified by entering their URI (not shown).

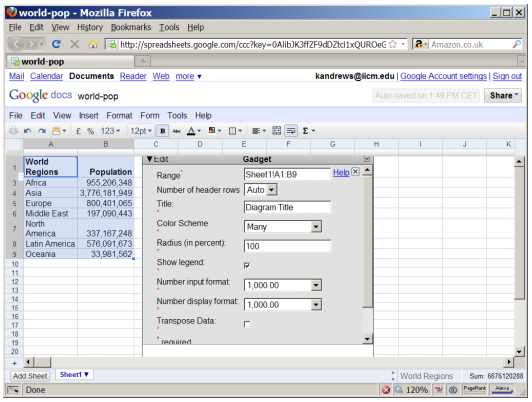


Figure 3: The gadget's settings (parameters) are entered.

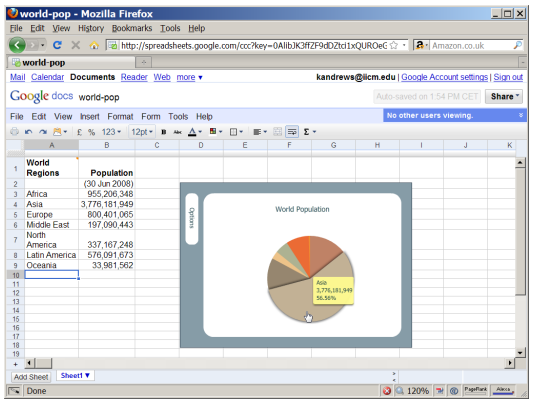


Figure 4: The gadget appears inside the spreadsheet. It responds immediately to any changes in data values.

tering the URI of their XML file. Unverified gadgets can be submitted for verification to Google, at which point they are listed.

Next, the gadget's Settings Panel appears (Figure 3). Here, various parameters and settings for the gadget can be entered. The Google Spreadsheet remembers these settings and uses them whenever the gadget is refreshed or redrawn. Finally, the gadget itself appears inside the Google spreadsheet (Figure 4). Liquid Diagrams gadgets can then further be interactively configured using the Options panel inside the visualisation.

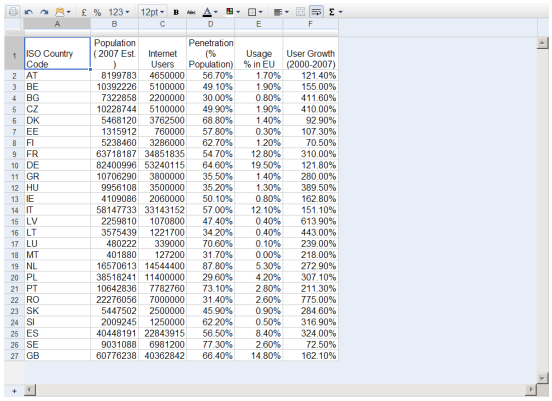
Gagdets can either reside within a standard spreadsheet sheet, say alongside data, or they can be given their own sheet (and thus have more room). Clicking on a gadget within a spreadsheet sheet reveals the gadget's title bar. Under the triangle at the top right are several functions, including Move to own sheet....

3. Liquid Diagrams

Seven information visualisations are currently available for Google Spreadsheets in the Liquid Diagrams suite: area chart, bar chart, heat map (choropleth), line chart, parallel coordinates, pie chart, and tree map. Two further visualisations are under development and will be added shortly: star plot and voronoi tree map.

Space restrictions do not allow all of them to be shown in full detail. Figures 5, 6, 7, and 8 show the Liquid Diagrams heatmap (choropleth) visualisation being applied to EU data about internet users. Once the user is happy with a visualisation, it can be exported in various formats. Figure 9 shows the final exported SVG vector graphics version of the diagram, which can be further edited and included in high-quality in other works.

Figure 10 shows an area chart of the most common girls names in Austria between 1997 and 2008. Figure 11 shows



ISO Country Code	Population (2007 Est.)	Internet Users	Penetration (% Population)	Usage % in EU	User Growth (2000-2007)
AT	8199783	4650000	56.70%	1.70%	121.40%
BE	10392226	5100000	49.10%	1.90%	155.00%
BG	7322858	2200000	30.00%	0.80%	411.60%
CZ	10228744	5100000	49.90%	1.90%	410.00%
DK	5468120	3762500	68.80%	1.40%	92.90%
EE	1315912	760000	57.80%	0.30%	107.30%
FI	5238460	3286000	62.70%	1.20%	70.50%
FR	63718187	34851835	54.70%	12.80%	310.00%
DE	82400996	53240115	64.60%	19.50%	121.80%
GR	10706290	3800000	35.50%	1.40%	280.00%
HU	9956108	3500000	35.20%	1.30%	389.50%
IE	4109086	2060000	50.10%	0.80%	162.80%
IT	58147733	33143152	57.00%	12.10%	151.10%
LV	2259810	1070800	47.40%	0.40%	613.90%
LT	3575439	1221700	34.20%	0.40%	443.00%
LU	480222	339000	70.60%	0.10%	239.00%
MT	401880	127200	31.70%	0.00%	218.00%
NL	16570613	14544400	87.80%	5.30%	272.90%
PL	38518241	11400000	29.60%	4.20%	307.10%
PT	10642836	7782760	73.10%	2.80%	211.30%
RO	22276056	7000000	31.40%	2.60%	775.00%
SK	5447502	2500000	45.90%	0.90%	284.60%
SI	2009245	1250000	62.20%	0.50%	316.90%
ES	40448191	22843915	56.50%	8.40%	324.00%
SE	9031088	6981200	77.30%	2.60%	72.50%
GB	60776238	40362842	66.40%	14.80%	162.10%

Figure 5: A Google spreadsheet containing data about internet users in EU countries. To save space, only the inside of the Google spreadsheet window is shown.

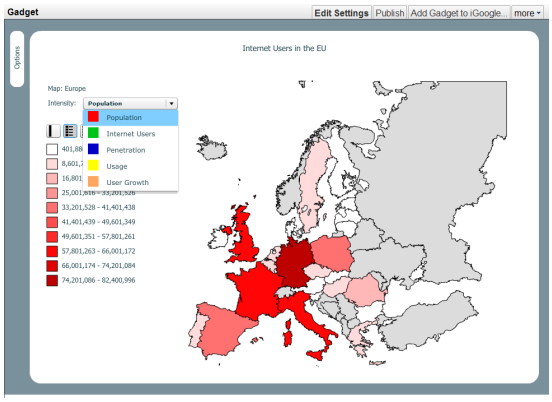


Figure 6: The Liquid Diagrams heatmap gadget has been inserted into the spreadsheet. The ISO country codes are matched to any available vector maps and a heatmap is created. Each data column is mapped to a different colour.

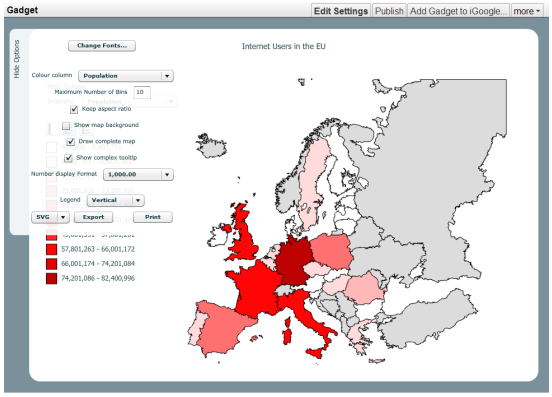


Figure 7: By default, the data is grouped into 10 bins (deciles) coded by colour intensity. The Options panel allows various settings to be changed and the final diagram to be exported.

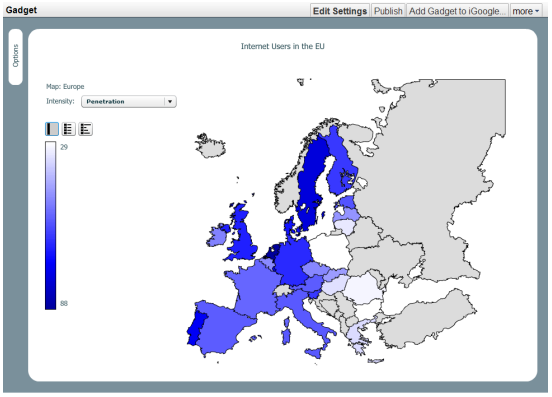


Figure 8: The user has selected internet penetration as the data column to be mapped and a continuous colour scale. This can then be exported as SVG.

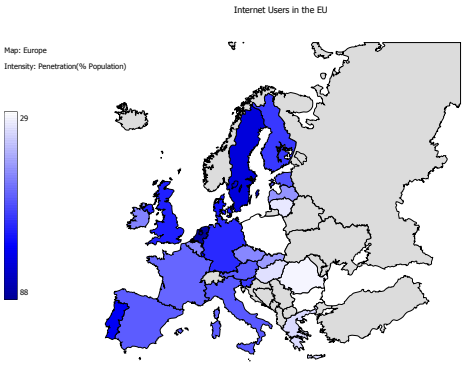


Figure 9: Penetration of the internet in the EU. Internet users as a percentage of the population. SVG vector graphics export of Liquid Diagrams heatmap.

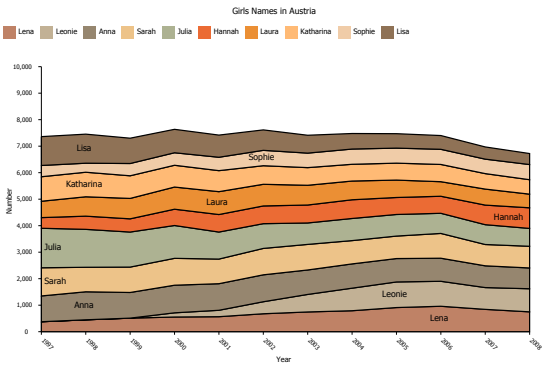


Figure 10: A Liquid Diagrams area chart showing the most common girls names in Austria between 1997 and 2008.

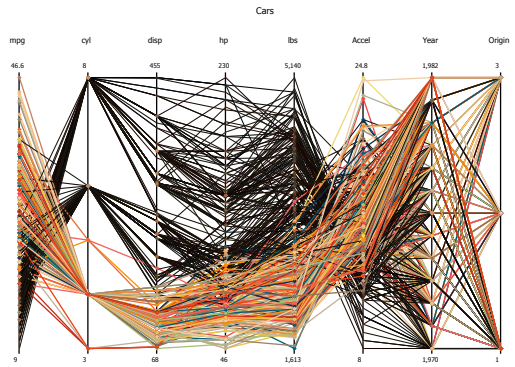


Figure 11: A Liquid Diagrams parallel coordinates visualisation of the well-known cars dataset.

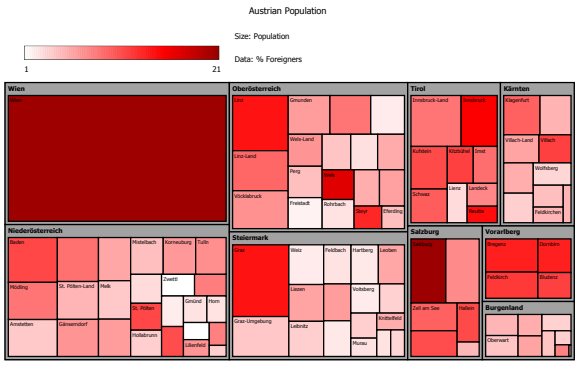


Figure 12: A Liquid Diagrams treemap visualisation of the nine provinces of Austria and their component districts. The size of each rectangle represents absolute population, the colour indicates the percentage of foreign nationals.

a parallel coordinates visualisation of the well-known cars dataset [13]. Figure 12 shows a treemap of Austrian population data. The size of each rectangle represents absolute population, the colour indicates the percentage of foreign nationals. It can clearly be seen that Vienna is the largest province in Austria in terms of population and also (with Salzburg city) has a high percentage of foreign nationals.

4. Interfacing to Liquid Diagrams

Liquid Diagrams gadgets are not, in fact, limited to any particular data source. The process of importing data into the Flex gadgets is to hand the data to the gadget by calling a function written in JavaScript. This means that any data source which supports access via JavaScript is suitable to be used with Liquid Diagrams gadgets. For example, standalone versions of the Liquid Diagrams visualisations could

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <mx:Application
3    xmlns:mx="http://www.adobe.com/2006/mxml"
4    layout="absolute"
5    creationComplete="uponCreation()">
6
7  public function uponCreation():void {
8    if (ExternalInterface.available) {
9      ExternalInterface.addCallback(
10       "getDataObjects", getDataObjects);
11     ExternalInterface.call("callbackReady");
12   } else {
13     Alert.show("Error: No External Interface!");
14   }
15 }
16
17 public function getDataObjects(
18   dataHeaders:Array, dataValues:Array,
19   dataOptions:Array, dataColours:Array):void {
20   // store and process the data
21   ...
22   drawVisualisation();
23 }
24 </mx:Application>
```

Listing 2: After initialisation, the application establishes an interface to its container and allows it to call the getDataObjects function. Afterwards, it calls the callbackReady function of the container to signal that initialisation is complete.

easily be built. However, for current development purposes, Google Spreadsheets are used as the data source to feed data into Liquid Diagrams visualisations.

The Flex gadget can be embedded into a web page using JavaScript. The code to embed the SWF object is written into the `innerHTML` element of a container specified by `div` tags in an HTML document, as shown in Listing 1. The embed operation causes the Flex object to be created.

After a gadget object has been created and initialised, it fires a `creationComplete` event, triggering the function uponCreation(). The association between event and function to be called is given inside an MXML file (see Listing 2), which is then compiled into the SWF object. Inside the uponCreation function, an external interface is established to enable communication between the Flex gadget and its container (lines 6–10). If the interface is available (line 6), functions can be defined which are callable by the container. The only externally callable function defined by Liquid Diagrams gadgets is the function getDataObjects, which will handle data transfer (line 7). Afterwards, the callbackReady function of the container is called by the Flex application to signal that its initialisation process has finished and it is ready to receive data (line 8). The function getDataObjects will later be called from outside to pass data from the data source into the gadget.

Listing 3 illustrates the final steps of the data exchange. The JavaScript function callbackReady (line 2) hands the data

```
<div id="chart"></div>

<script type="text/javascript">
  var containerElement = document.getElementById('chart');

  if(containerElement != null) {
    containerElement.innerHTML =
    '<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"' +
    'id="linechart" width="320" height="240"' +
    'codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab">' +
      '<param name="movie" value="linechart.swf" />' +
      '<param name="quality" value="high" />' +
      '<param name="bgcolor" value="#869ca7" />' +
      '<param name="allowScriptAccess" value="always" />' +
      '<embed src="linechart.swf" quality="high" bgcolor="#869ca7"' +
        'width="320" height="240" name="linechart"' +
        'align="middle"' +
        'play="true"' +
        'loop="false"' +
        'quality="high"' +
        'allowScriptAccess="always"' +
        'type="application/x-shockwave-flash"' +
        'pluginspage="http://www.adobe.com/go/getflashplayer">' +
      '</embed></object>';
}
</script>
```

Listing 1: A container is added to a web page and a SWF object is embedded into this container.

```
// Javascript file
function callbackReady() {
  getFlexApp('linechart').getDataObjects(
    vis.headers, vis.data,
    vis.options, vis.colours);
}
```

Listing 3: After an interface has been established, the getDataObjects function of the container is called which passes the spreadsheet data to the Flex gadget.

to the Flex application by calling the previously specified function getDataObjects (line 3). getDataObjects continues to process the data without requiring any further communication with the container.

The main file of a Google Gadget is the gadget XML file, shown in Listing 4. For custom gadgets, i.e. those not yet verified and listed by Google, the URL of the Gadget XML file must be entered when the gadget is inserted into a Google spreadsheet.

5. Related Work

Many Eyes [14] and Swivel [12] were the first efforts aimed at bringing information visualisation to the masses over the web. However, they can be somewhat cumbersome in their data upload facilities, have much less support for online spreadsheet operations than Google Docs, support only a limited number of visualisations, and do not allow users to export (scalable, high-quality) vector graphics versions of their visualisations.

Google themselves have a number of interactive charts and gadgets under the banner of Google Chart Tools [3], some of their own and some from third parties. Most are quite simple, although some are quite sophisticated. Some of them draw directly in SVG, although none of them to our knowledge allow interactive configuration and then export to SVG.

Hans Rosling's work at the gapminder project [11], since acquired by Google and rebranded as the Google Public Data Explorer [4], and especially his wonderful series of talks at the TED conferences [10] has done much to raise public awareness of the value of information visualisation. The OECD Explorer [9, 7], based on the NCVA eXplorer [8], allow public OECD data to be visualised online in a number of ways.

A number of information visualisation toolkits are available for use with Javascript and Flex, the best of which is flare [6]. flare makes it fairly easy to build interactive data visualisations. Another such project, birdeye [1], looks promising, but is at an earlier stage.

6. Concluding Remarks

Liquid Diagrams exploit the potential of cloud computing to allow information visualisations to be used by anyone with a web browser and a free Google Docs account. The main contributions of Liquid Diagrams are: 1) It is usable by anyone. Liquid Diagrams can be used in the cloud by

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="Line Chart"
    author="Martin Lessacher"
  </ModulePrefs>

  <UserPref name="_table_query_url"
    display_name="Data source URL"
    required="true" />
  <UserPref name="title"
    display_name="Title:"
    required="true"
    default_value="Diagram Title" />
  <UserPref name="line_thickness"
    display_name="Line thickness:"
    required="true"
    default_value="3" />

  <Content type="html"><![CDATA[
    <form>
      <div id="chart"></div>
    </form>
    <script type="text/javascript"
      src="http://www.google.com/jsapi">
    </script>
    <script type="text/javascript">
      google.load("visualization","1");
      google.setOnLoadCallback(initialize);

      var prefs = new _IG_Prefs();

      function initialize() {
        gadgetHelper = new google.visualization.
             GadgetHelper();
        var query = gadgetHelper.createQueryFromPrefs(
             prefs);
        query.send(handleQueryResponse);
      }

      function handleQueryResponse(response) {
        if (!gadgetHelper.validateResponse(response)) {
          return;
        }

        var data = response.getDataTable();
      }
    </script>
  ]]></Content>
</Module>
```

Listing 4: A sample Google Gadget XML file. The `UserPref` entries specifiy parameters for the gadget. In this case, the `Content` block conforms to the Google Visualization API and uses the `GadgetHelper` class to load data from the Google spreadsheet.

anyone with a web browser and a Google Docs account. Nothing has to be installed. 2) Live updates. Changes in the data are reflected immediately in the visualisation. 3) It is interactively configurable. The visualisations can be further interactively configured. 4) Vector graphics export. All Liquid Diagrams visualisations can be exported as Scalable Vector Graphics (SVG) or printed in vector PDF, for inclusion into other works. Vector graphics are freely scalable without loss of quality.

References

[1] birdeye. *birdeye Information Visualization and Visual Analytics Library*. March 2010. http://code.google.com/p/birdeye/ .

[2] Compete. *Compete*. March 2010. http://compete.com/ .

[3] Google. *Google Chart Tools*. March 2010. http://code.google.com/apis/charttools/ .

[4] Google. *Google Public Data Explorer*. March 2010. http://www.google.com/publicdata/ .

[5] Google. *Google Visualization API*. March 2010. http://code.google.com/apis/visualization/ .

[6] J. Heer. *flare*. March 2010. http://flare.prefuse.org/ .

[7] M. Jern. *Collaborative Web-Enabled GeoAnalytics Applied to OECD Regional Data*. In *Proc. 6th International Conference on Cooperative Design, Visualization, and Engineering (CDVE 2009)*, pages 32–43. Springer LNCS 5738, Luxembourg, Luxembourg, 2009. doi:10.1007/978-3-642-04265-2_5 .

[8] NCVA. *eXplorer*. March 2010. http://ncva.itn.liu.se/explorer .

[9] OECD. *OECD Regional Statistics*. March 2010. http://stats.oecd.org/OECDregionalstatistics/ .

[10] H. Rosling. *Debunking Myths About the "Third World"*. February 2006. http://www.ted.com/talks/view/id/92 .

[11] H. Rosling. *Gapminder*. March 2010. http://gapminder.org/ .

[12] Swivel. *Swivel*. March 2010. http://swivel.com/ .

[13] UCI. *Auto MPG Data Set*. UCI Machine Learning Repository, July 1993. http://archive.ics.uci.edu/ml/datasets/Auto+MPG .

[14] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. *Many Eyes: a Site for Visualization at Internet Scale*. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 13(6):1121–1128, November 2007. ISSN 1077-2626. doi:10.1109/TVCG.2007.70577 . http://www.research.ibm.com/visual/papers/viegasinfovis07.pdf .