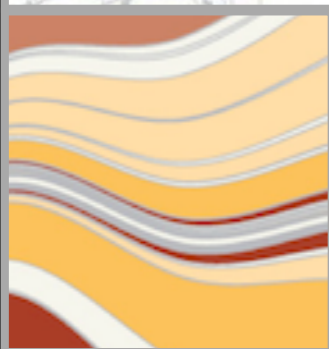
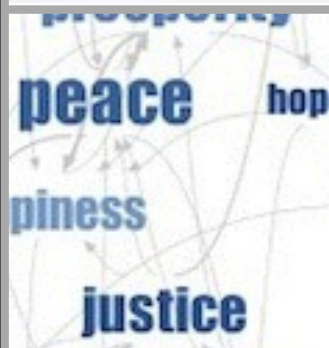
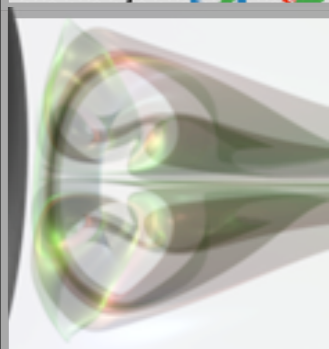
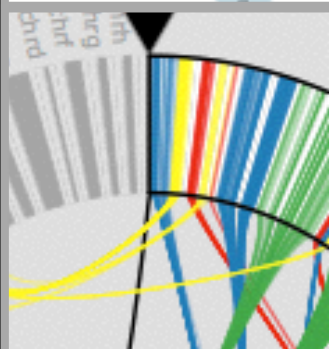
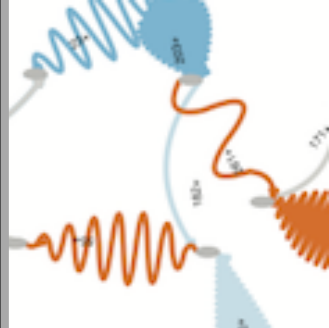


cs6630 | September 23 2014

# TASKS & INTERACTION

Miriah Meyer  
*University of Utah*



administrivia . . .

-any questions/problems with Processing?

last time . . .










# Processing

# what is it?

-programming environment

-visually oriented applications

-targets artists, designers, etc.

 <p><b>Avena+ Test Bed</b> by Benedikt Groß</p> <p>Avena+ Test Bed is a project that explores the relationship between landscape, agriculture and digital fabrication by intercepting the process of precision farming by generative design.</p> <p>Links: <a href="#">Benedikt Groß</a></p>	 <p><b>Kinograph</b> by Matthew Epler</p> <p>Kinograph is an open source project that makes film digitisation affordable and scalable. It uses components available on the internet, a few 3D printed parts, and a consumer level camera and it produces high quality video with sound.</p> <p>Links: <a href="#">Kinograph</a></p>	 <p><b>fluid</b> by Hannes Jung</p> <p>Created by Hannes Jung, fluid is a concept study of an interacting, changing surface that uses non-newtonian fluid, an Arduino board, a speaker and Processing to allow surface to change from liquid to solid, from plain to three-dimensional symmetric patterns.</p> <p>Links: <a href="#">Hannes Jung</a></p>
 <p><b>3D Printed Record</b> by Amanda Ghassaei</p> <p>Created using Processing, ModelBuilder Library by Marius Watz and a 3D printer, Amanda Ghassaei at instructables managed to print a 33rpm music record that actually doesn't sound too bad considering the limitations of currently available 3d printing technologies.</p> <p>Links: <a href="#">Instructables</a></p>	 <p><b>Digital Natives and Glitched Realities</b> by Matthew Plummer-Fernandez</p> <p>Digital Natives are everyday items such as toys and detergent bottles that are 3D scanned using a digital camera, subjected to algorithms that distort and finally 3D printed in colour resin/sandstone.</p> <p>Links: <a href="#">Matthew Plummer-Fernandez</a></p>	 <p><b>Stone Spray</b> by Petr Novikov, Inder Shergill and Anna Kulik</p> <p>Stone Spray is a construction method which uses soil as the base material and a liquid binder to solidify the soil granules. The device uses an Arduino UNO, Processing application and a custom built jet spray system to deposit the mix of soil and binder, for constructing architectural shapes.</p> <p>Links: <a href="#">Petr Novikov, Inder Shergill and Anna Kulik</a></p>
 <p><b>City Symphonies</b> by Mark McKeague</p> <p>Mark McKeague explores an</p>	 <p><b>Silenc</b> by Maras Karambelkar, Momo Miyazaki and Kenneth A. Robertsen</p>	 <p><b>unnamed soundsculpture</b> by Daniel Franke &amp; Cedric Kiefer</p> <p>Produced by onformative and</p>

# why Processing?

-difficulty to sketch with other languages

-complicated setup

-not easy to learn

-repetitive code

```
100 // draw loop
101 // draw loop
102 // draw loop
103 // draw loop
104 // draw loop
105 // draw loop
106 // draw loop
107 // draw loop
108 // draw loop
109 // draw loop
110 // draw loop
111 // draw loop
112 // draw loop
113 // draw loop
114 // draw loop
115 // draw loop
116 // draw loop
117 // draw loop
118 // draw loop
119 // draw loop
120 // draw loop
121 // draw loop
122 // draw loop
123 // draw loop
124 // draw loop
125 // draw loop
126 // draw loop
127 // draw loop
128 // draw loop
129 // draw loop
130 // draw loop
131 // draw loop
132 // draw loop
133 // draw loop
134 // draw loop
135 // draw loop
136 // draw loop
137 // draw loop
138 // draw loop
139 // draw loop
140 // draw loop
141 // draw loop
142 // draw loop
143 // draw loop
144 // draw loop
145 // draw loop
146 // draw loop
147 // draw loop
148 // draw loop
149 // draw loop
150 // draw loop
151 // draw loop
152 // draw loop
153 // draw loop
154 // draw loop
155 // draw loop
156 // draw loop
157 // draw loop
158 // draw loop
159 // draw loop
160 // draw loop
161 // draw loop
162 // draw loop
163 // draw loop
164 // draw loop
165 // draw loop
166 // draw loop
167 // draw loop
168 // draw loop
169 // draw loop
170 // draw loop
171 // draw loop
172 // draw loop
173 // draw loop
174 // draw loop
175 // draw loop
176 // draw loop
177 // draw loop
178 // draw loop
179 // draw loop
180 // draw loop
181 // draw loop
182 // draw loop
183 // draw loop
184 // draw loop
185 // draw loop
186 // draw loop
187 // draw loop
188 // draw loop
189 // draw loop
190 // draw loop
191 // draw loop
192 // draw loop
193 // draw loop
194 // draw loop
195 // draw loop
196 // draw loop
197 // draw loop
198 // draw loop
199 // draw loop
200 // draw loop
```



# why Processing?

- Java-based



- similar syntax & portability

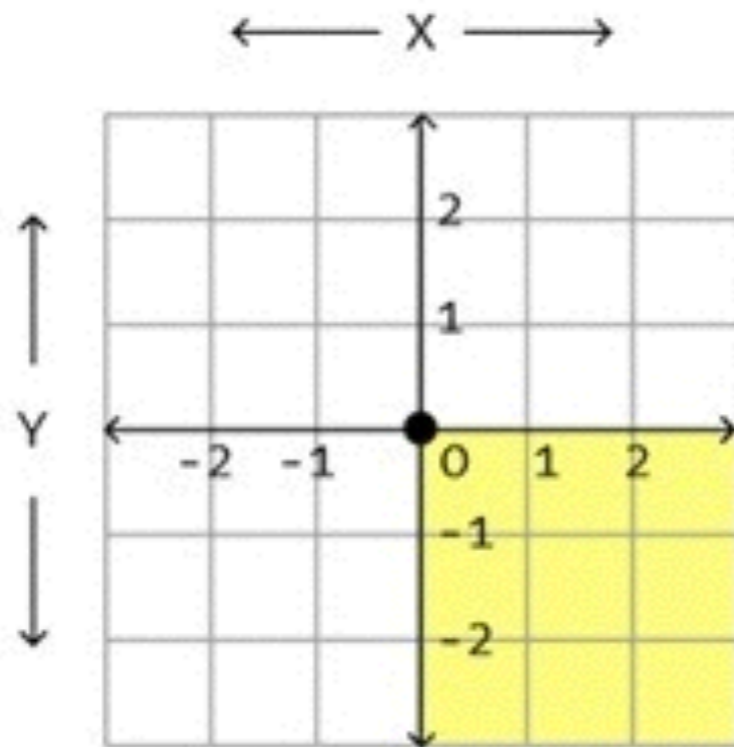
## **3 Billion Devices Run Java**

Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...



# graphics

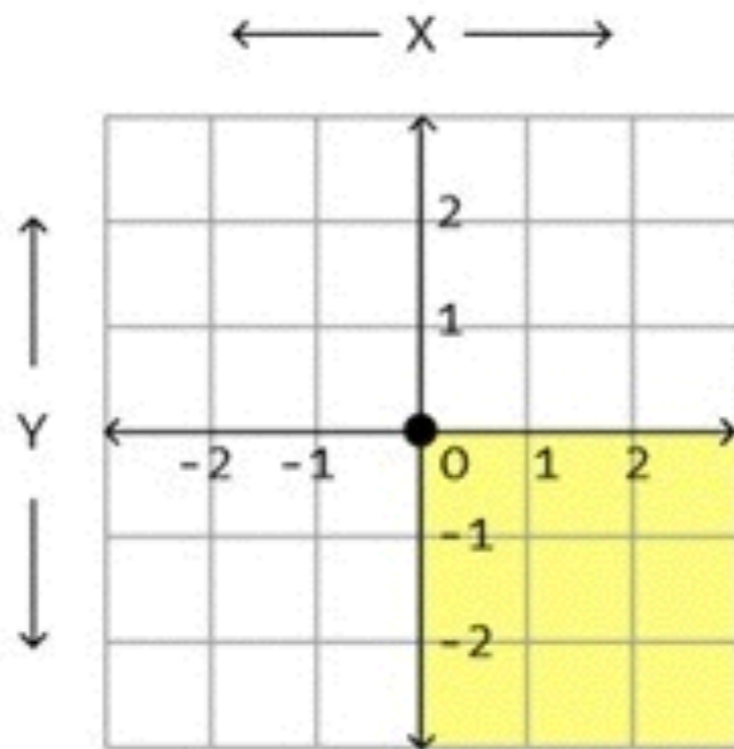
-grid of pixels



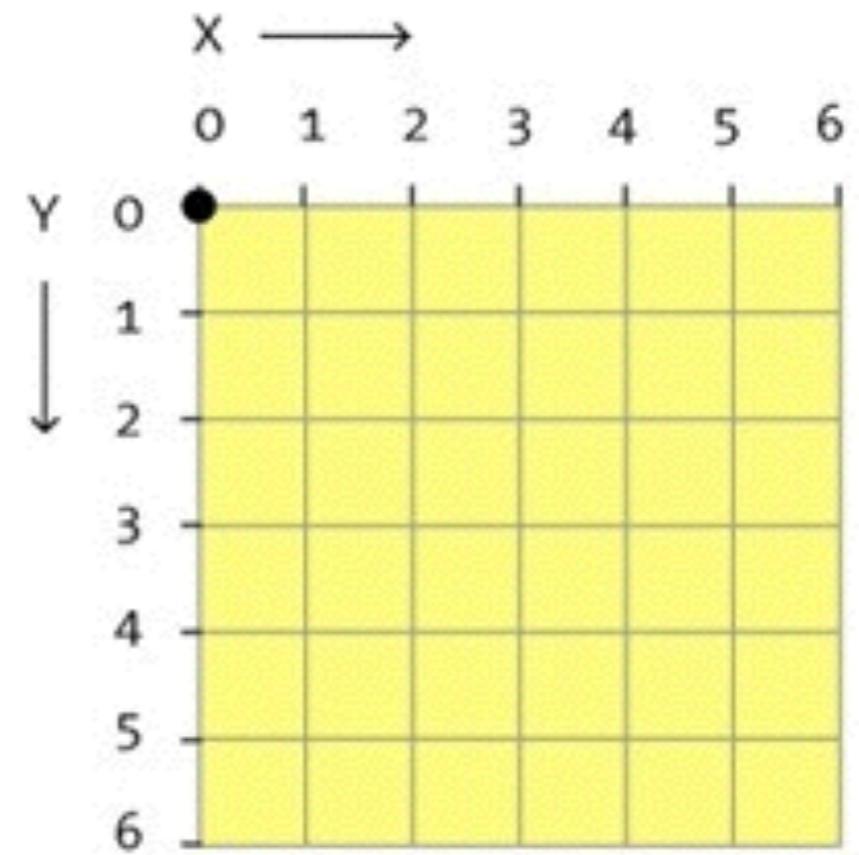
Eighth Grade

# graphics

-grid of pixels



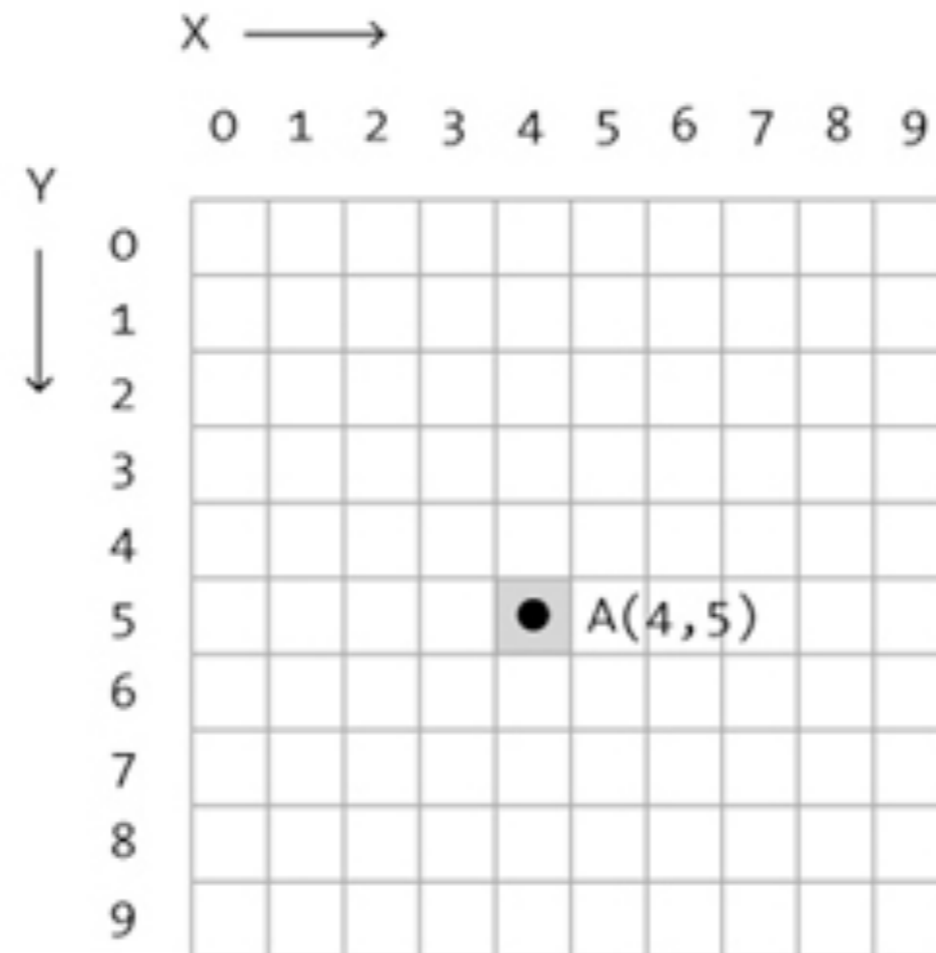
Eighth Grade



Computer

# shape

`point(x, y);`

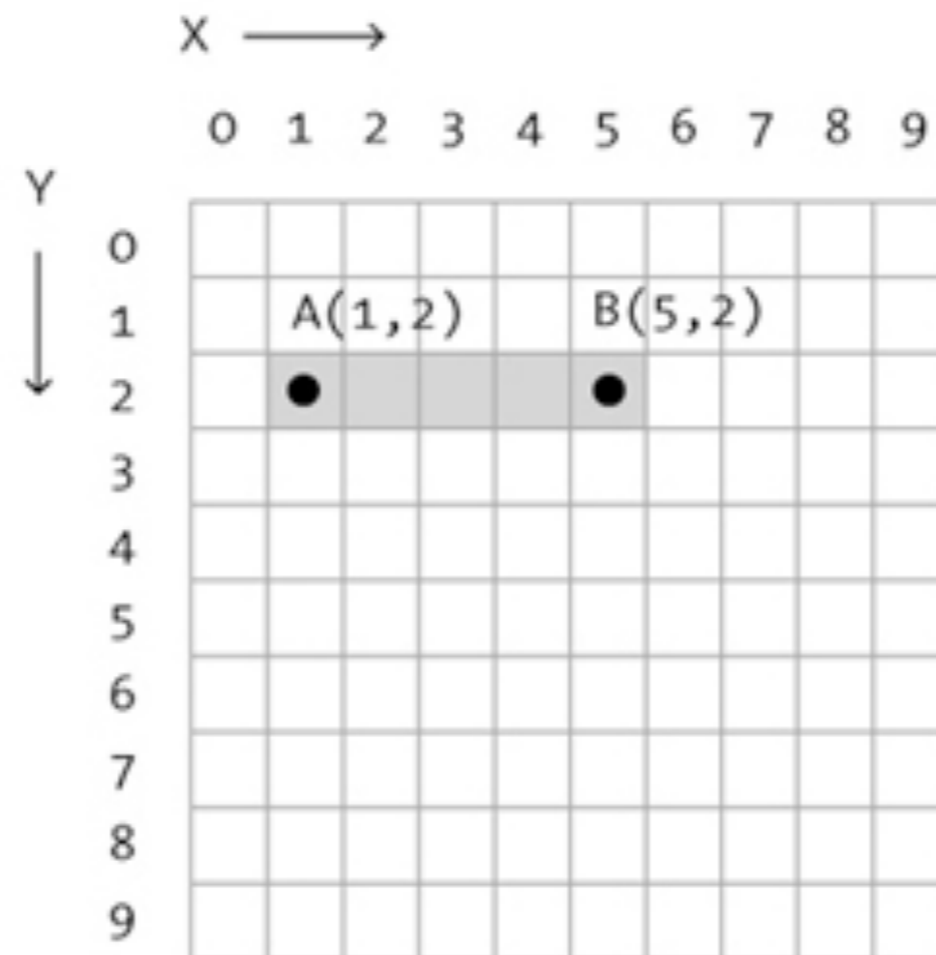


`point(x,y);`

Example:  
`A(4,5);`

# shape

```
line(x1, y1, x2, y2);
```



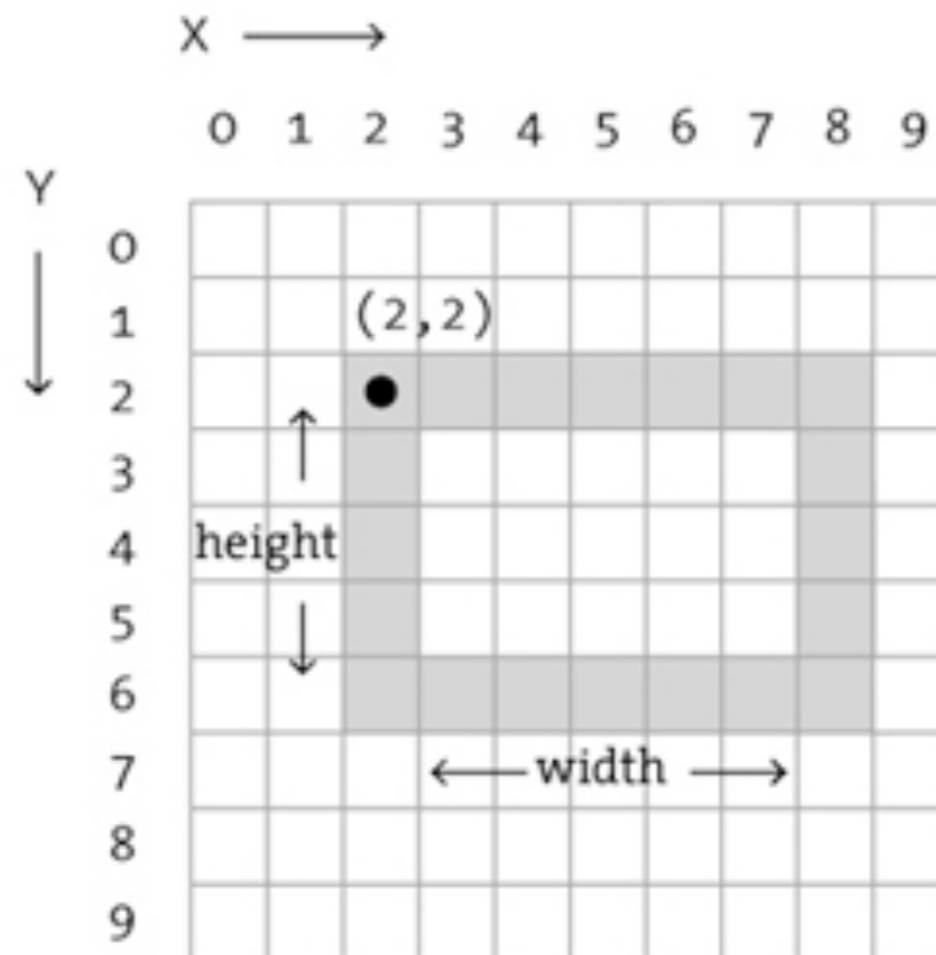
```
line(x1,y1,x2,y2);
```

Point A   Point B

Example:  
`line(1,2,5,2);`

# shape

```
rect(x, y, width, height);
```



```
rect(x,y,width,height);
```

Example:

```
rect(2,2,7,5);
```

today . . .

**-task abstraction**

**-interaction**

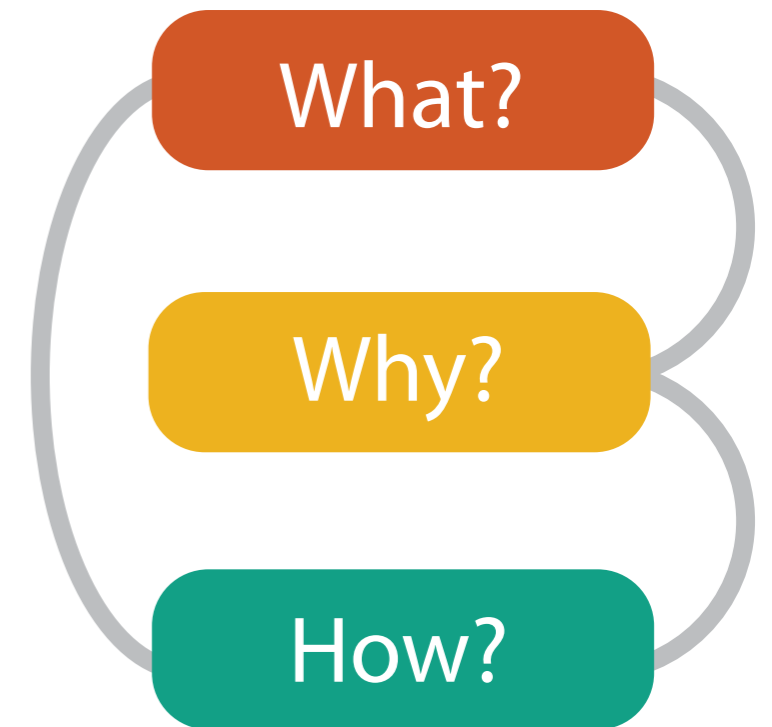
**-task abstraction**

-interaction



# analysis: what, why, and how

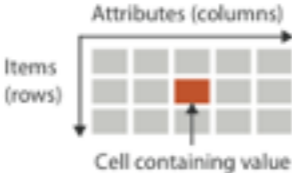
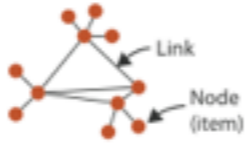
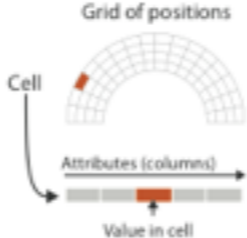
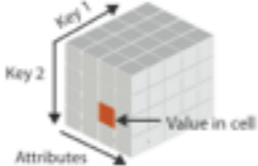




- **what** is shown?
- **why** is the user looking at it?
- **how** is it shown?

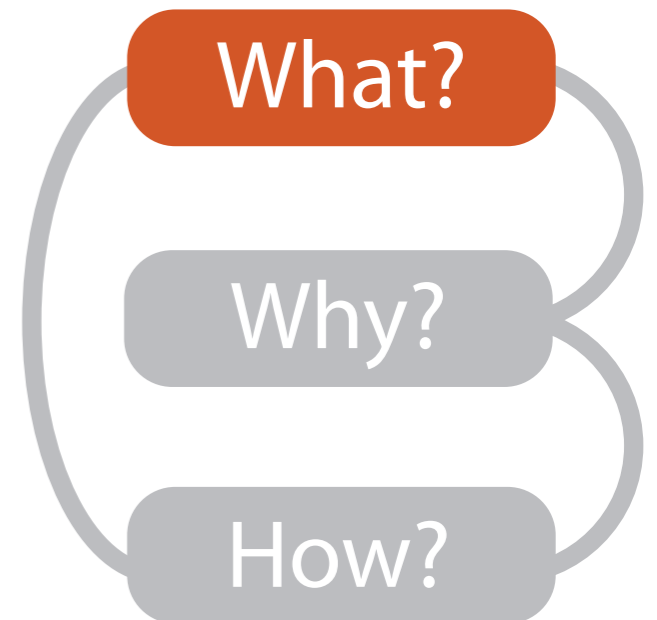


- abstract vocabulary avoids domain-specific terms
- what-why-how analysis framework as scaffold to think systematically about design space

# data abstraction

**What?**

Datasets	Attributes																				
<p>→ Data Types</p> <p>→ Items → Attributes → Links → Positions → Grids</p> <p>→ Data and Dataset Types</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>Tables</td> <td>Networks &amp; Trees</td> <td>Fields</td> <td>Geometry</td> <td>Clusters, Sets, Lists</td> </tr> <tr> <td>Items</td> <td>Items (nodes)</td> <td>Grids</td> <td>Items</td> <td>Items</td> </tr> <tr> <td>Attributes</td> <td>Links</td> <td>Positions</td> <td>Positions</td> <td></td> </tr> <tr> <td></td> <td>Attributes</td> <td>Attributes</td> <td></td> <td></td> </tr> </table>	Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists	Items	Items (nodes)	Grids	Items	Items	Attributes	Links	Positions	Positions			Attributes	Attributes			<p>→ Attribute Types</p> <p>→ Categorical</p> <p style="text-align: center;">+ ● ■ ▲</p> <p>→ Ordered</p> <p>→ Ordinal</p> <p style="text-align: center;">↑ ↑↑ ↑↑↑</p> <p>→ Quantitative</p> <p style="text-align: center;"> ----- ----- ----- </p>
Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists																	
Items	Items (nodes)	Grids	Items	Items																	
Attributes	Links	Positions	Positions																		
	Attributes	Attributes																			
<p>→ Dataset Types</p> <p>→ Tables</p>  <p>→ Networks</p>  <p>→ Fields (Continuous)</p>  <p>→ Multidimensional Table</p>  <p>→ Trees</p>  <p>→ Geometry (Spatial)</p> 	<p>→ Ordering Direction</p> <p>→ Sequential</p> <p style="text-align: center;">→</p> <p>→ Diverging</p> <p style="text-align: center;">←→</p> <p>→ Cyclic</p> <p style="text-align: center;">↻</p>																				
<p>→ Dataset Availability</p> <p>→ Static</p>  <p>→ Dynamic</p> 																					



# How?

## Encode

### → Arrange

→ Express



→ Separate



→ Order



→ Align



→ Use



### → Map

from **categorical** and **ordered** attributes

→ Color

→ Hue



→ Saturation



→ Luminance



→ Size, Angle, Curvature, ...



→ Shape



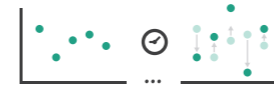
→ Motion

Direction, Rate, Frequency, ...



## Manipulate

### → Change



### → Select



### → Navigate



## Facet

### → Juxtapose



### → Partition



### → Superimpose

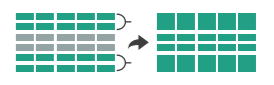


## Reduce

### → Filter



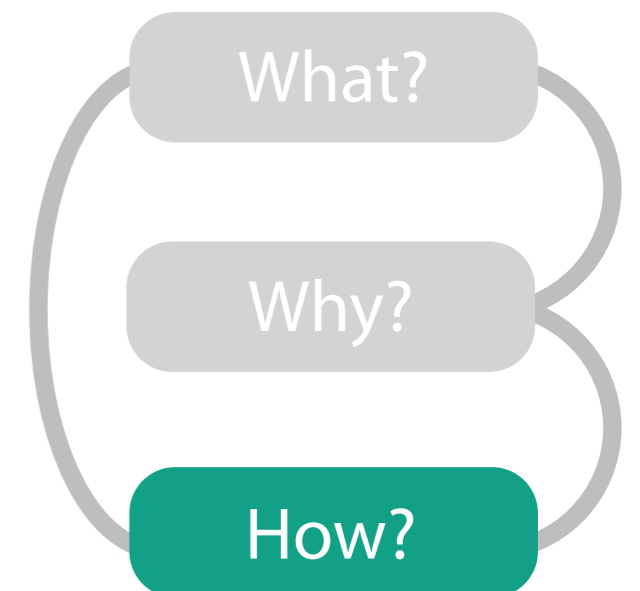
### → Aggregate



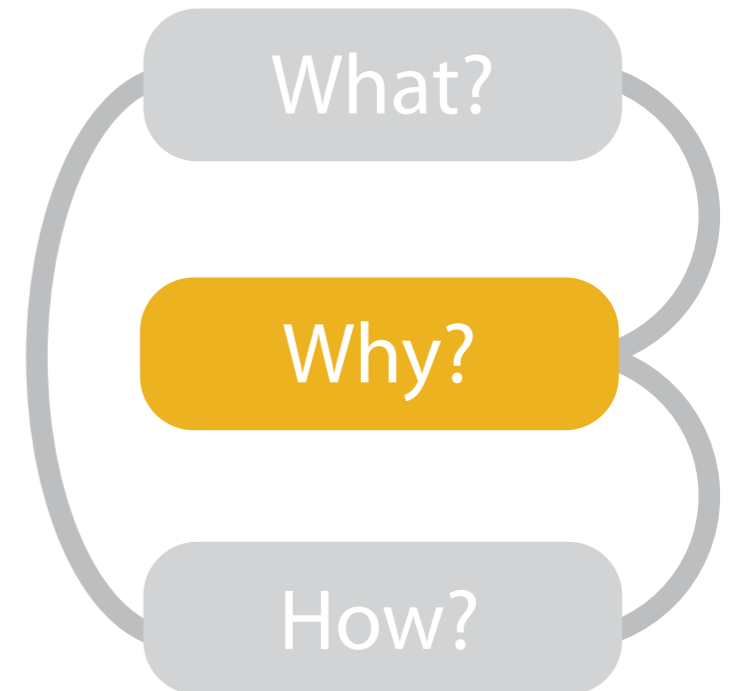
### → Embed



# visual encoding



# task abstraction



# **{action, target} pairs**

*discover distribution*

*compare trends*

*locate outliers*

*browse topology*

→ Analyze

{**action**, target}

→ Search

→ Query

**{action, target}**

→ Analyze

→ Consume

→ Discover



→ Present



→ Enjoy



→ Produce

→ Annotate



→ Record



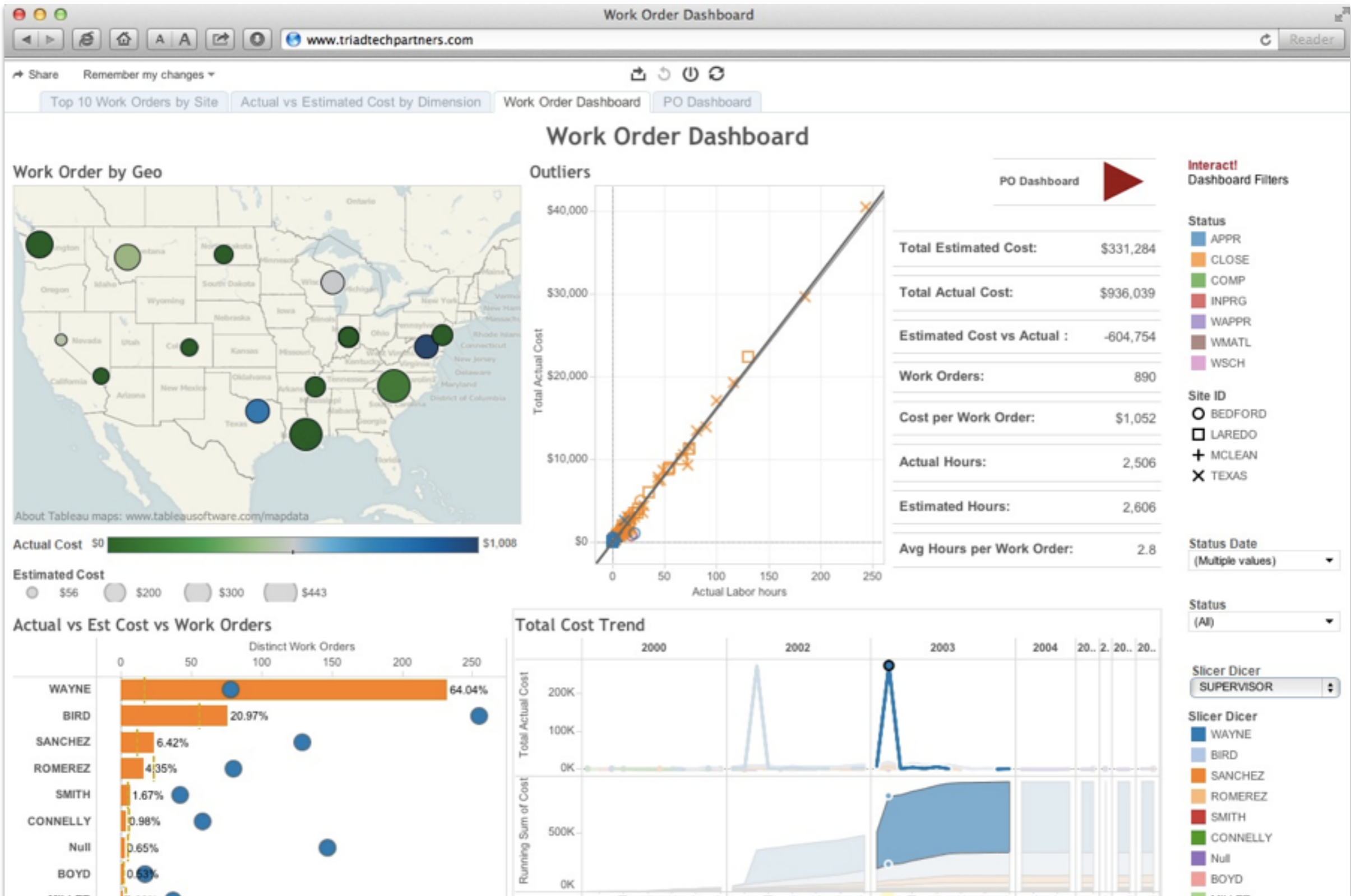
→ Derive



→ Search

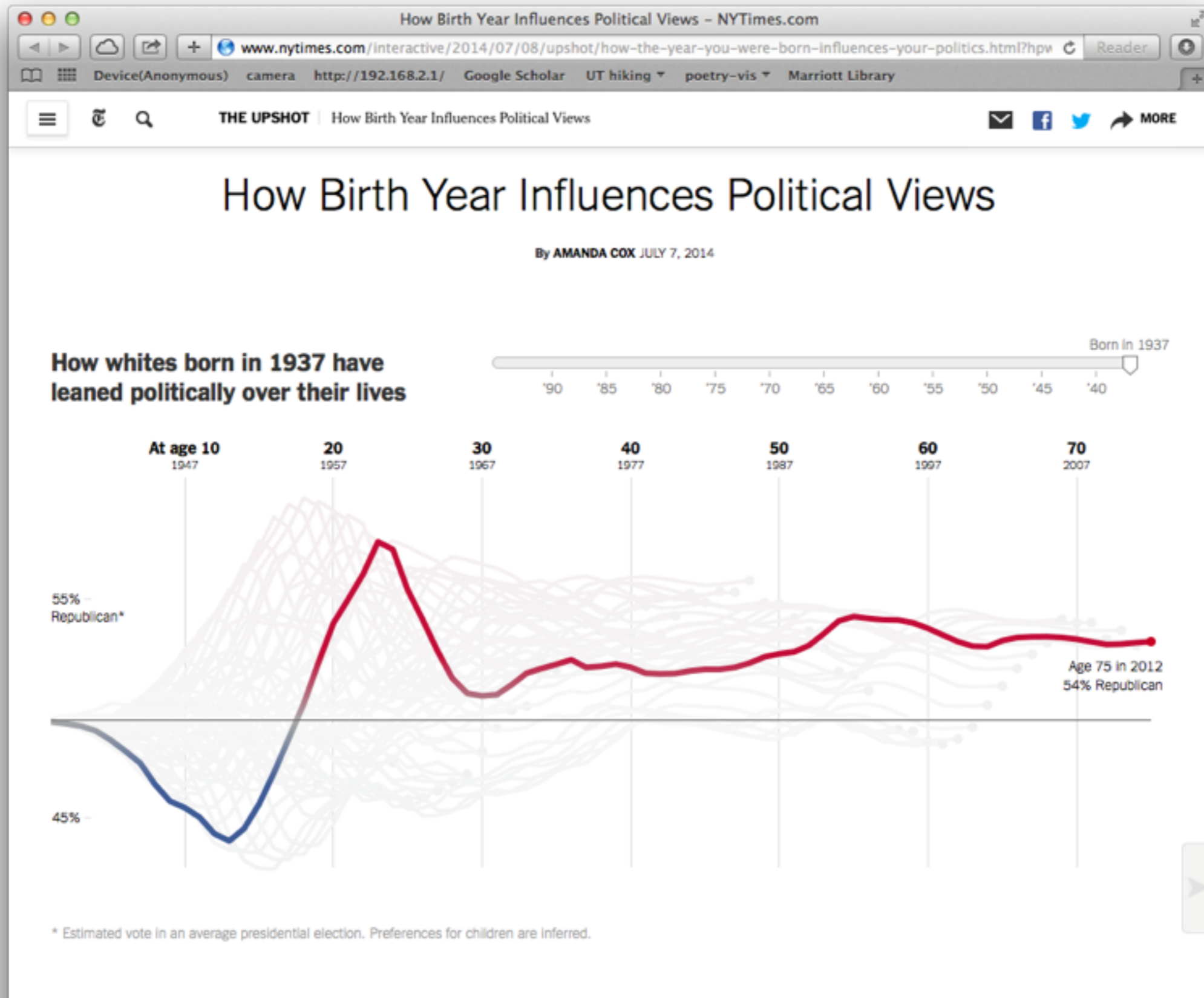
→ Query

# discover





# present



# enjoy



{**action**, target}

→ Analyze

→ Consume

→ Discover



→ Present



→ Enjoy



→ Produce

→ Annotate



→ Record



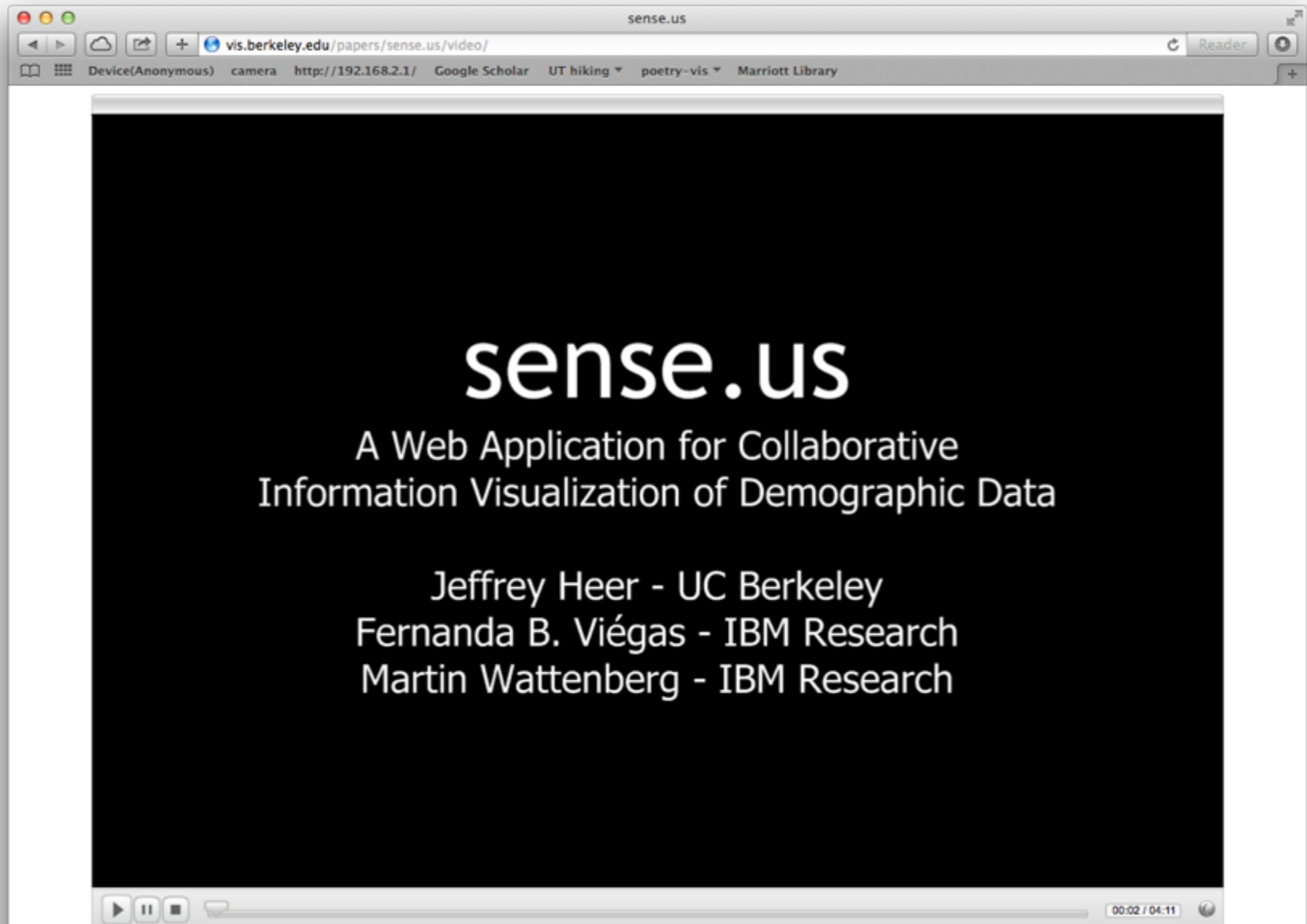
→ Derive



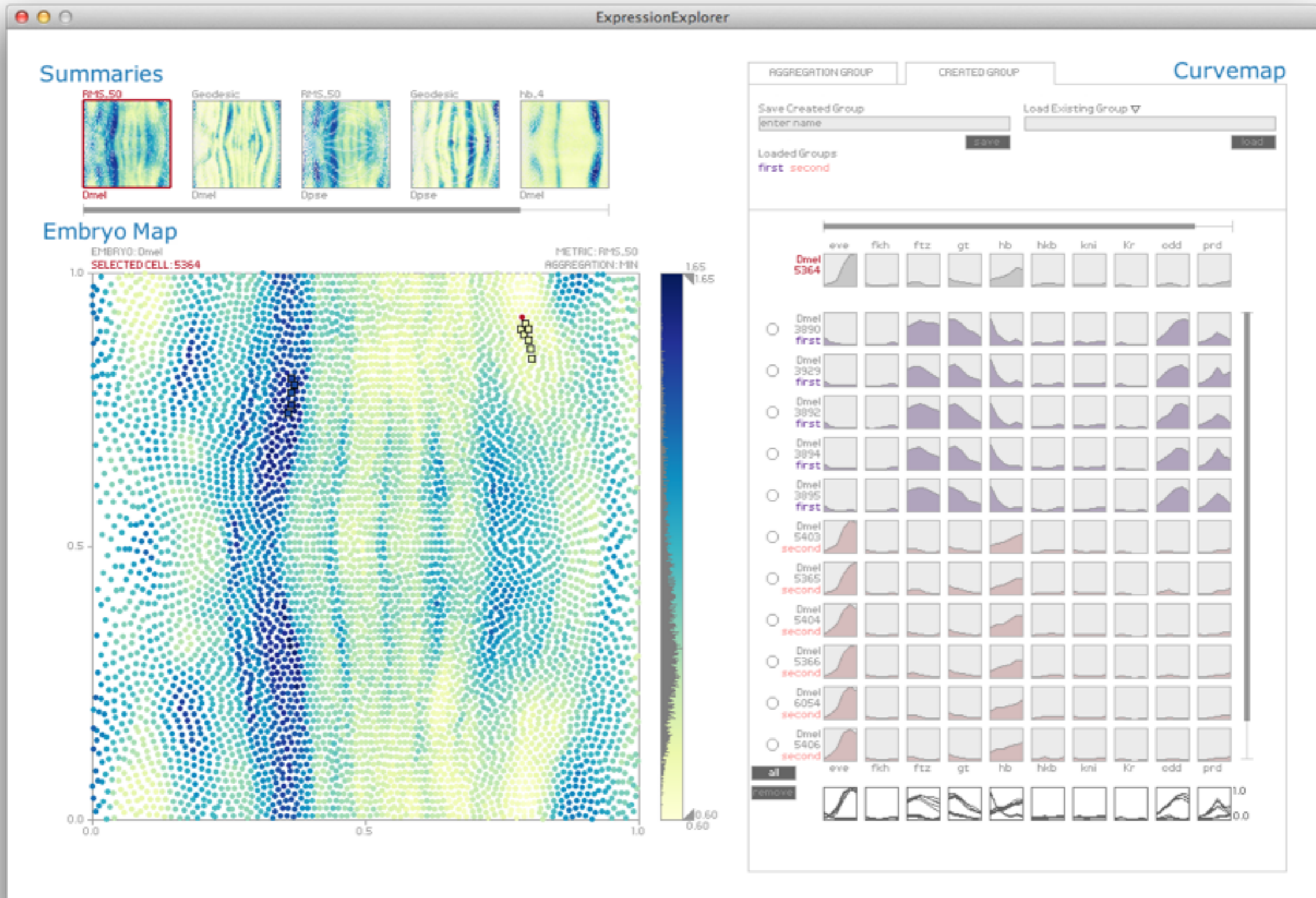
→ Search

→ Query

# annotate & record



# derive



{**action**, target}

→ Analyze

→ Consume

→ Discover



→ Present



→ Enjoy



→ Produce

→ Annotate







→ Record



→ Derive



→ Search

	Target known	Target unknown
Location known	 <i>Lookup</i>	 <i>Browse</i>
Location unknown	 <i>Locate</i>	 <i>Explore</i>

→ Query

{**action**, target}

→ Analyze

→ Consume

→ Discover



→ Present



→ Enjoy



→ Produce

→ Annotate



→ Record



→ Derive



→ Search

	Target known	Target unknown
Location known	<i>Lookup</i>	<i>Browse</i>
Location unknown	<i>Locate</i>	<i>Explore</i>

→ Query

→ Identify



→ Compare



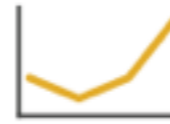
→ Summarize



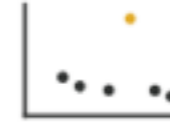
{action, **target**}

→ All Data

→ Trends



→ Outliers



→ Features



→ Attributes

→ One

→ *Distribution*



→ *Extremes*

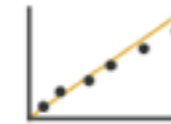


→ Many

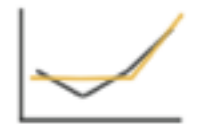
→ *Dependency*



→ *Correlation*



→ *Similarity*

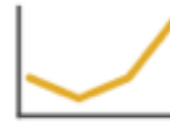




{action, **target**}

→ All Data

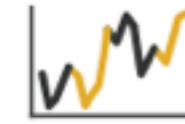
→ Trends



→ Outliers



→ Features



→ Attributes

→ One

→ *Distribution*



→ *Extremes*

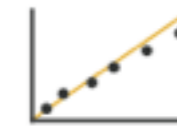


→ Many

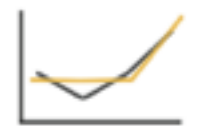
→ *Dependency*



→ *Correlation*

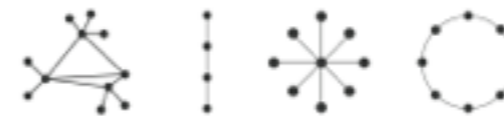


→ *Similarity*



→ Network Data

→ Topology



→ *Paths*



→ Spatial Data

→ Shape



why does this matter?

thought experiment...

# Why?

## 👉 Actions

### ➔ Analyze

➔ Consume

➔ Discover



➔ Present



➔ Enjoy



➔ Produce

➔ Annotate



➔ Record



➔ Derive



### ➔ Search

	Target known	Target unknown
Location known	•••• Lookup	•••• Browse
Location unknown	<••••> Locate	<••••> Explore

### ➔ Query

➔ Identify



➔ Compare



➔ Summarize



## 🎯 Targets

### ➔ All Data

➔ Trends



➔ Outliers



➔ Features



### ➔ Attributes

➔ One

➔ Distribution



➔ Extremes

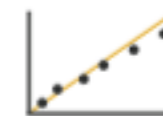


➔ Many

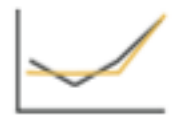
➔ Dependency



➔ Correlation



➔ Similarity



### ➔ Network Data

➔ Topology



➔ Paths



### ➔ Spatial Data

➔ Shape



**{action, target}**

# A Multi-Level Typology of Abstract Visualization Tasks

Matthew Brehmer and Tamara Munzner, *Member, IEEE*

**Abstract**—The considerable previous work characterizing visualization usage has focused on low-level tasks or interactions and high-level tasks, leaving a gap between them that is not addressed. This gap leads to a lack of distinction between the ends and means of a task, limiting the potential for rigorous analysis. We contribute a multi-level typology of visualization tasks to address this gap, distinguishing *why* and *how* a visualization task is performed, as well as *what* the task inputs and outputs are. Our typology allows complex tasks to be expressed as sequences of interdependent simpler tasks, resulting in concise and flexible descriptions for tasks of varying complexity and scope. It provides abstract rather than domain-specific descriptions of tasks, so that useful comparisons can be made between visualization systems targeted at different application domains. This descriptive power supports a level of analysis required for the generation of new designs, by guiding the translation of domain-specific problems into abstract tasks, and for the qualitative evaluation of visualization usage. We demonstrate the benefits of our approach in a detailed case study, comparing task descriptions from our typology to those derived from related work. We also discuss the similarities and differences between our typology and over two dozen extant classification systems and theoretical frameworks from the literatures of visualization, human-computer interaction, information retrieval, communications, and cartography.

**Index Terms**—Typology, visualization models, task and requirements analysis, qualitative evaluation

---

## 1 INTRODUCTION

Consider a person who encounters a choropleth map while reading a blog post in the aftermath of last year’s American presidential election. This particular map is static and visually encodes two attributes, candidate and margin of victory, encoded for each state using a bivariate colour mapping. This person decides to compare the election results of Texas to those of California, motivated not by an explicit need to generate or verify some hypothesis, nor by a need to present the visualization to an audience, but rather by a casual interest in American politics and its two most populous states. How might we describe this person’s *task* in an abstract rather than domain-specific way?

According to the nested model for visualization design and validation [43], abstract tasks are domain- and interface-agnostic operations performed by users. Disappointingly, there is little agreement as to the appropriate granularity of an abstract task among the many extant classifications of user behaviour in the visualization, human-computer interaction, cartography, and information retrieval literature [2, 3, 5, 10, 11, 12, 13, 14, 15, 19, 23, 29, 31, 37, 39, 42, 50, 51, 56, 57, 59, 61, 64, 66, 72, 73, 75, 78, 82, 83]. One of the more frequently cited of these [2] would classify the above example as being a series of value retrieval tasks. This low-level characterization does not describe the user’s context or motivation; nor does take into account prior experience and background knowledge. For instance, a description of this task might differ if the user was unfamiliar with American

describable in an abstract way across multiple levels.

The primary contribution of this paper is a multi-level typology of abstract visualization tasks that unites the previously disconnected scopes of low-level and high-level classification systems by proposing multiple levels of linkage between them. Our typology provides a powerful and flexible way to describe complex tasks as a sequence of interdependent simpler ones. While this typology is very much informed by previous work, it is also the result of new thinking and has many points of divergence with previous models. Central to the organization of our typology are three questions that serve to disambiguate the means and ends of a task: *why* the task is performed, *how* the task is performed, and *what* are the task’s inputs and outputs. We have found that no prior characterization of tasks satisfactorily answers all of these questions simultaneously at multiple levels of abstraction. Typically, low-level classification systems provide a sense of *how* a task is performed, but not *why*; high-level classification systems are the converse. One major advantage of our typology over prior work is in providing linkage between these two questions. Another advantage is the ability to link sequences of tasks, made possible by the consideration of *what* tasks operate on.

Our typology provides a consistent lexicon for description that supports making precise comparisons of tasks between different visualization tools and across application domains. Succinct and abstract

# A Design Space of Visualization Tasks

Hans-Jörg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann

**Abstract**—Knowledge about visualization tasks plays an important role in choosing or building suitable visual representations to pursue them. Yet, tasks are a multi-faceted concept and it is thus not surprising that the many existing task taxonomies and models all describe different aspects of tasks, depending on what these task descriptions aim to capture. This results in a clear need to bring these different aspects together under the common hood of a general design space of visualization tasks, which we propose in this paper. Our design space consists of five design dimensions that characterize the main aspects of tasks and that have so far been distributed across different task descriptions. We exemplify its concrete use by applying our design space in the domain of climate impact research. To this end, we propose interfaces to our design space for different user roles (developers, authors, and end users) that allow users of different levels of expertise to work with it.

**Index Terms**—Task taxonomy, design space, climate impact research, visualization recommendation



## 1 INTRODUCTION

As the field of information visualization matures, a phase of consolidation sets in that aims to pull together multiple individual works of research under a common conceptual hood. This hood can take on different shapes and forms, one of which is the *design space*. Such a design space realizes a descriptive generalization that permits to specify a concrete instance – be it a layout [8], a visualization [46], or a combination of visualizations [28] – by making design choices along a number of independent design dimensions. Even last year’s InfoVis conference recognized the increasing importance of design spaces by dedicating an entire session to them.

Yet, information visualization is more than the visual representation alone. It also takes into account the tasks the user wishes to pursue with the visual representation. The literature contains a wealth of classifications, taxonomies, and frameworks that describe these tasks: lists of verbal task descriptions, mathematical task models, domain-specific task collections, and procedural task combinations into workflows. All of these serve the respective purpose well for which they have been developed. However, the research question of how to consolidate them under the hood of one common design space is still open, even though it has been shown on a smaller scale that such a combination into a common framework can be a useful endeavor [9, 21].

In this paper, we aim to give a first answer to this research question by contributing such a design space for visualization tasks. This contribution is twofold. First, it derives an abstract design space that brings together the different aspects of the existing task taxonomies and models. It serves to clarify the somewhat fuzzy notion of visualization tasks and to provide a framework for the design of visualization tasks.

a visualization task design space for climate impact research based on structured interviews with eight domain experts and two visualization developers. This design space is then utilized to recommend visualizations that are suitable to pursue a given task in that field.

The remainder of this paper is organized as follows: The related work is summarized in Section 2 and from its discussion, we derive our task design space in Section 3. We then debate its properties, limitations, and applications in Section 4. This also includes examples of how some of the existing task taxonomies can be expressed as parts of our design space. After this conceptual part, Section 5 details the use case example of how to apply the general design space to the application domain of climate impact research and how to draw concrete benefits from it. With this example, we aim to show a feasible way for the adaptation of the design space that can be transferred to other application domains as well. We conclude this paper by briefly sharing our personal experience from working with the design space and pointing out directions for future work in Section 6.

## 2 RELATED WORK

The concept of *tasks* exhibits numerous facets that are also reflected in the existing body of research on that topic. Commonly, *visualization tasks* are understood as activities to be carried out interactively on a visual data representation for a particular reason. The investigation of visualization tasks has the aim to **establish recurring tasks** in order to use the knowledge about them for improving the **design and evaluation of visualizations**. Existing research for both of these aspects is

# A Design Space of Visualization Tasks

Hans-Jörg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann

**Abstract**—Knowledge about visualization tasks plays an important role in choosing or building suitable visual representations to pursue them. Yet, tasks are a multi-faceted concept and it is thus not surprising that the many existing task taxonomies and models all describe different aspects of tasks, depending on what these task descriptions aim to capture. This results in a clear need to bring these different aspects together under the common hood of a general design space of visualization tasks, which we propose in this paper. Our design space consists of five design dimensions that characterize the main aspects of tasks and that have so far been distributed across different task descriptions. We exemplify its concrete use by applying our design space in the domain of climate impact research. To this end, we propose interfaces to our design space for different user roles (developers, authors, and end users) that allow users of different levels of expertise to work with it.

**Index Terms**—Task taxonomy, design space, climate impact research, visualization recommendation



## 1 INTRODUCTION

As the field of information visualization matures, a phase of consolidation sets in that aims to pull together multiple individual works of

a visualization task design space for climate impact research based on structured interviews with eight domain experts and two visualization

- **WHY** is a task pursued? This specifies the task's **goal**.
- **HOW** is a task carried out? This specifies the task's **means**.
- **WHAT** does a task seek? This specifies the data **characteristics**.
- **WHERE** in the data does a task operate? This specifies the **target**, as well as the **cardinality** of data entities within that target.
- **WHEN** is a task performed? This specifies the order of tasks.
- **WHO** is executing a task? This specifies the (type of) user.

brings together the different aspects of the existing task taxonomies and models. It serves to clarify the somewhat fuzzy notion of visualization tasks and to provide a structured framework for their description.

to use the knowledge about them for improving the **design and evaluation of visualizations**. Existing research for both of these aspects is

-task abstraction

**-interaction**

-change over time

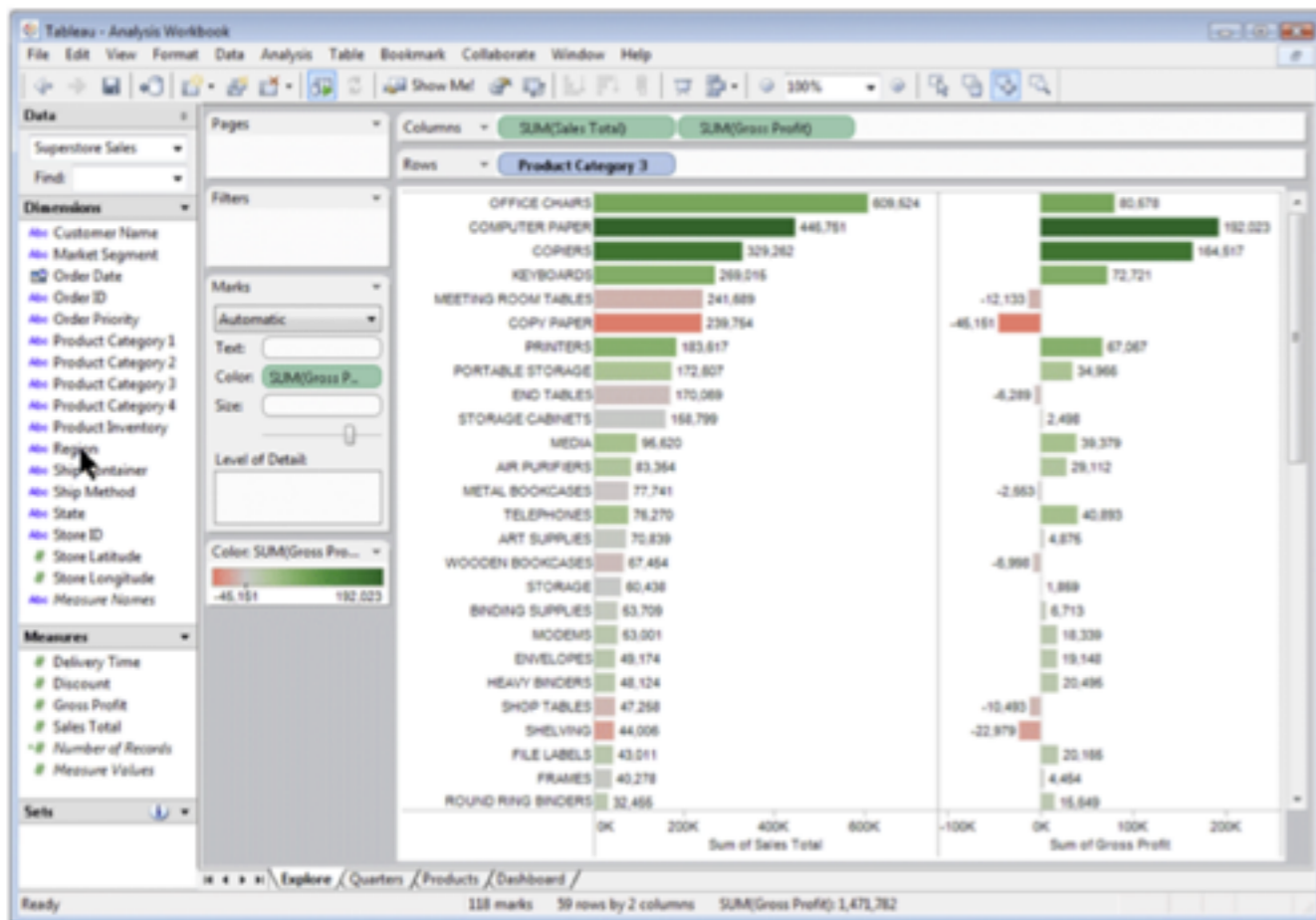
-selection

-highlighting

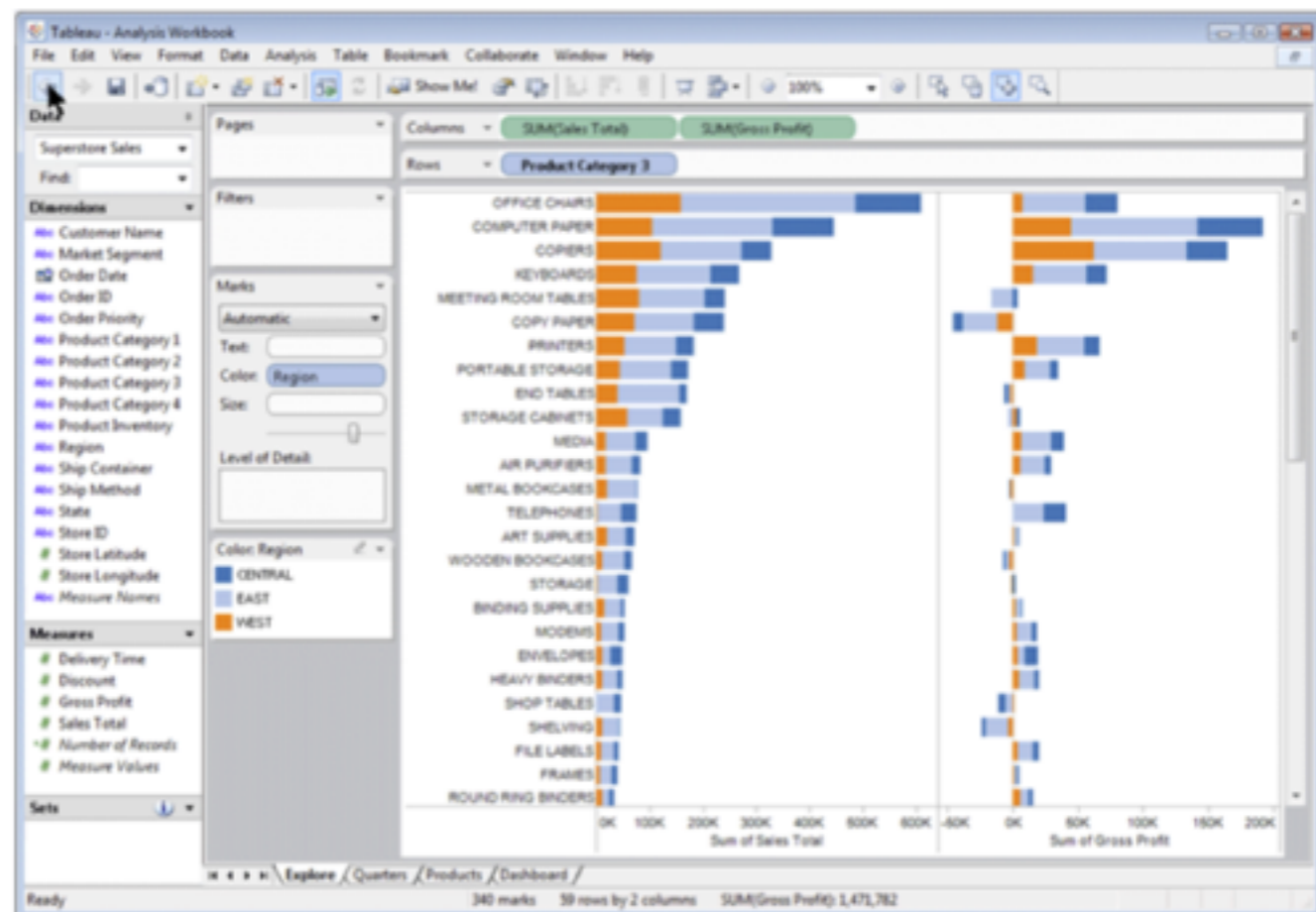
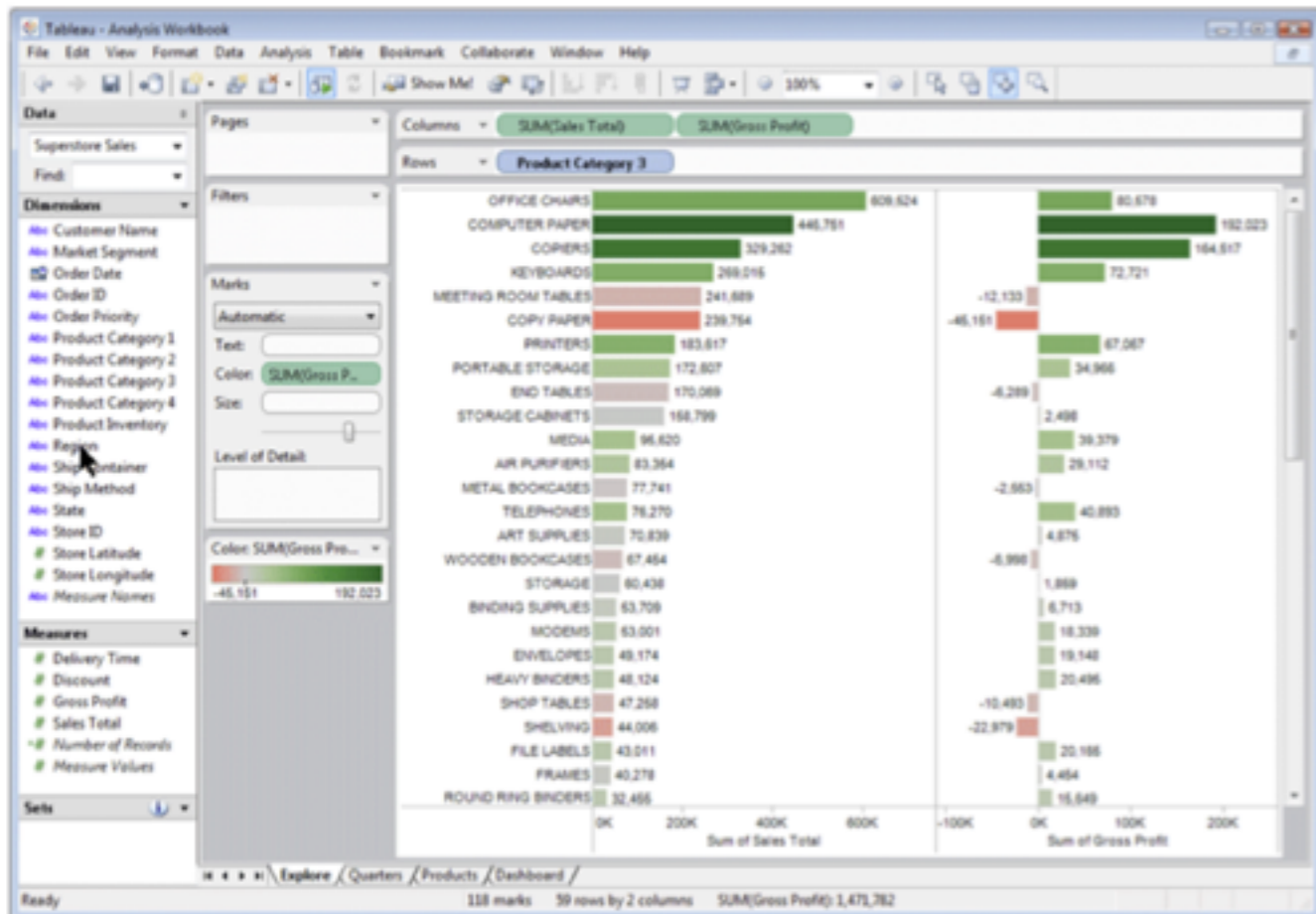
-navigation

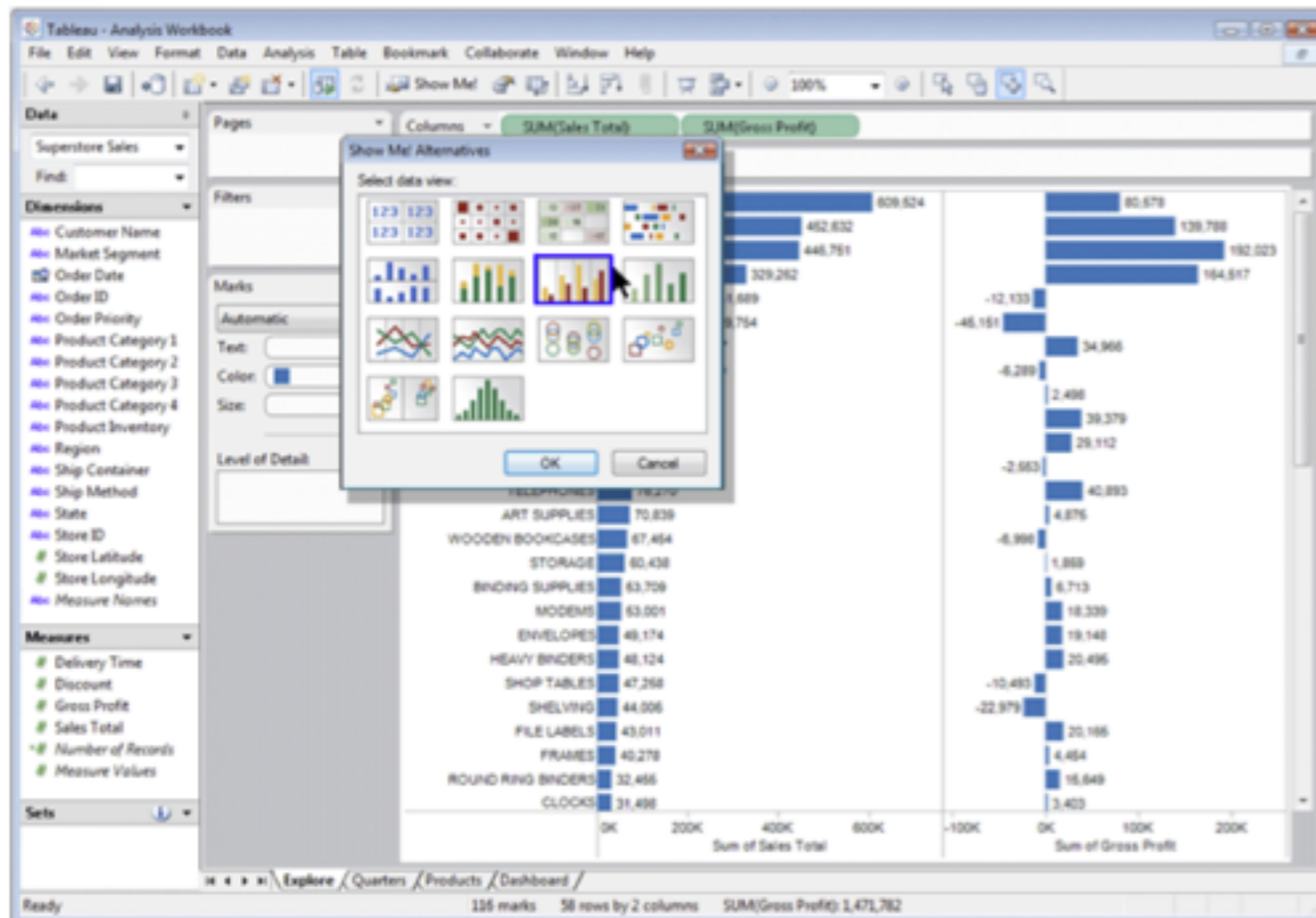
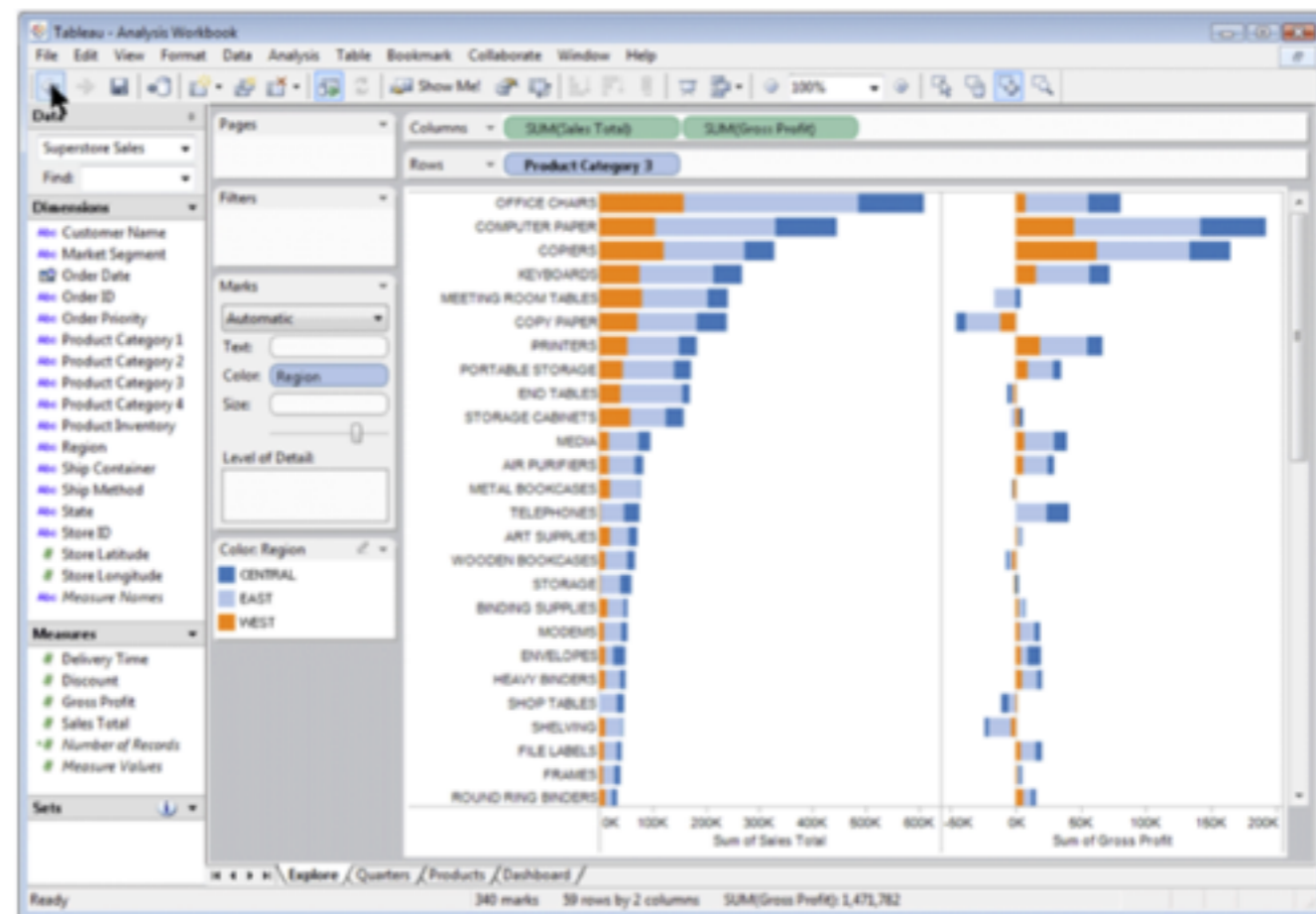
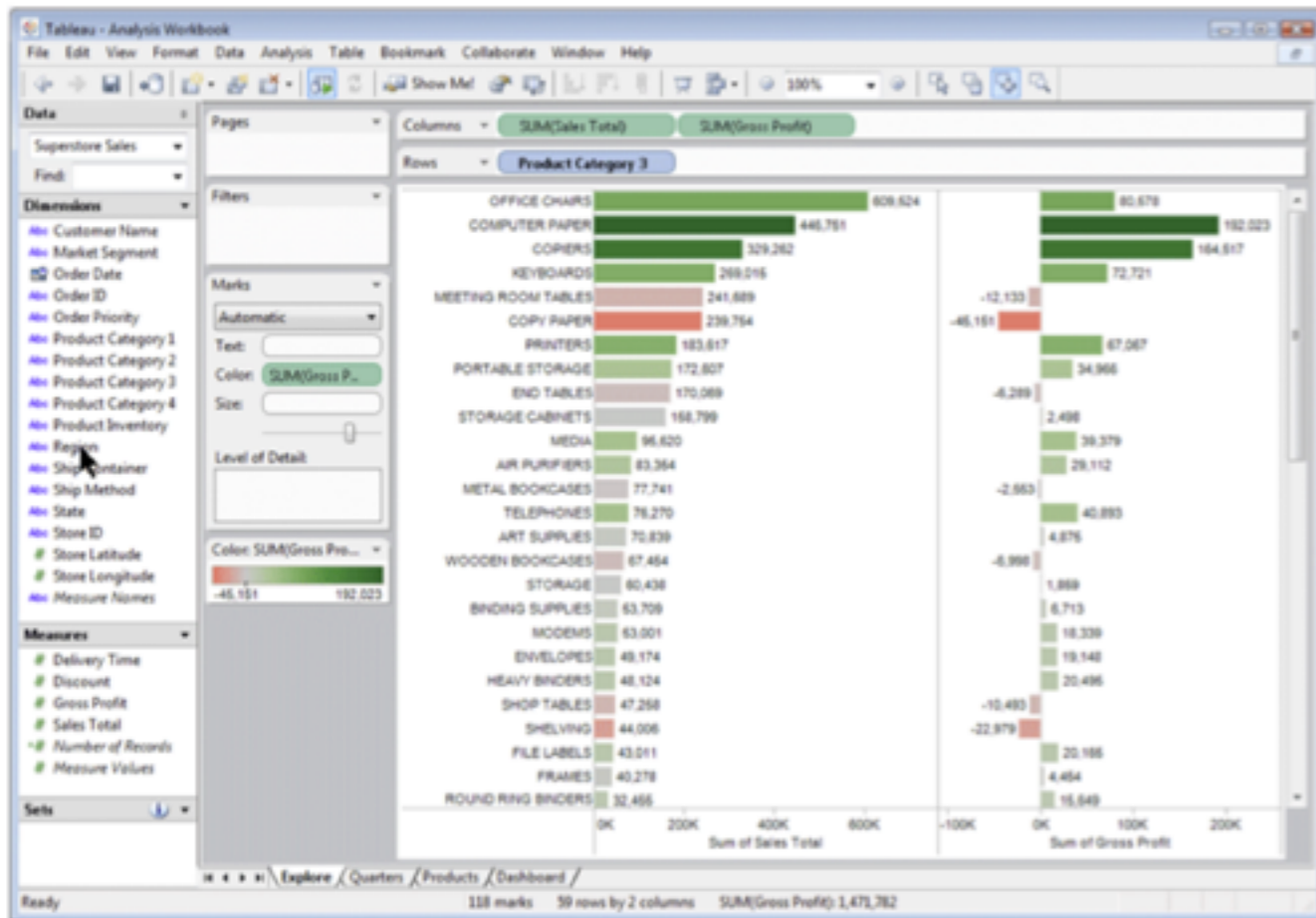


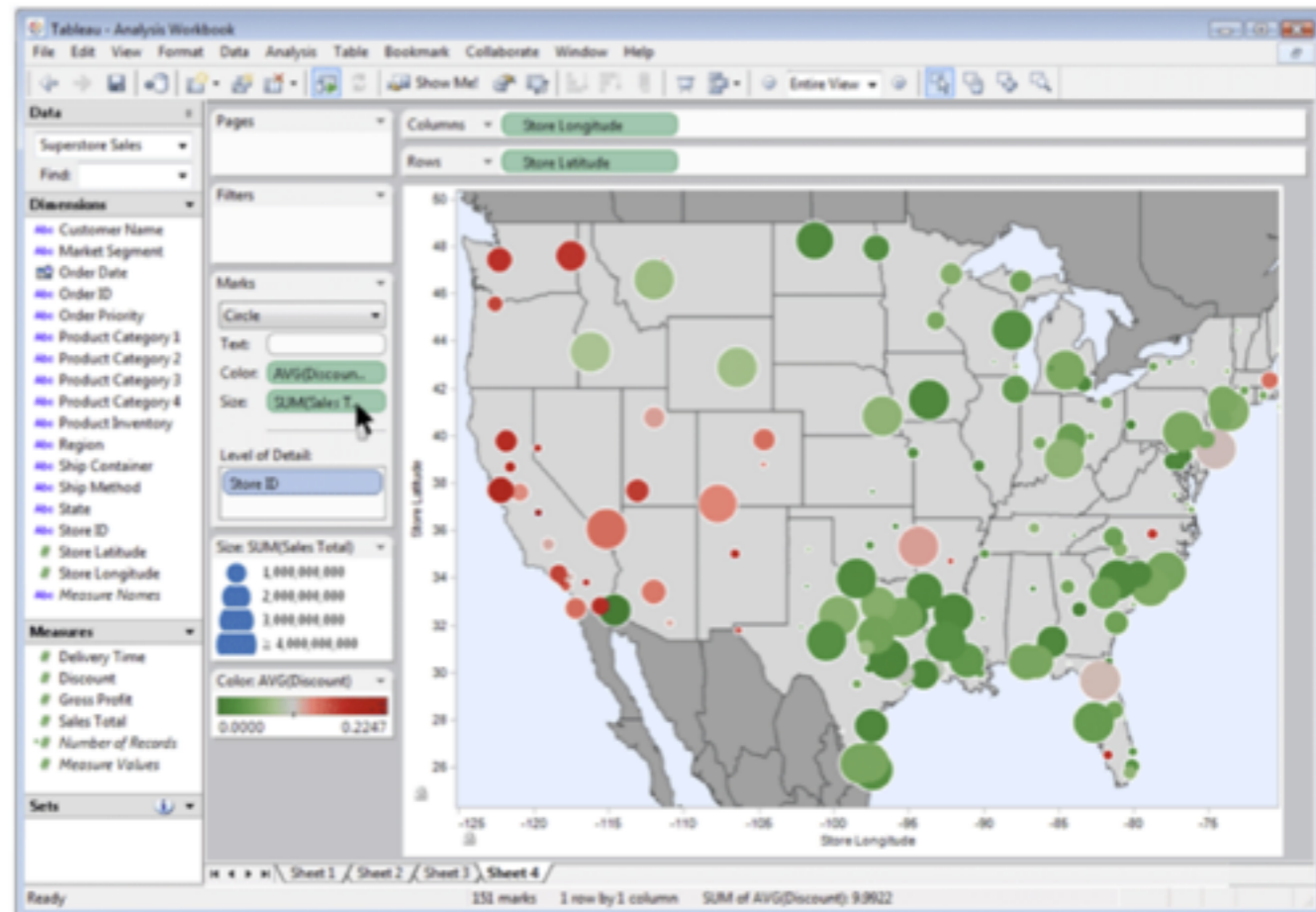
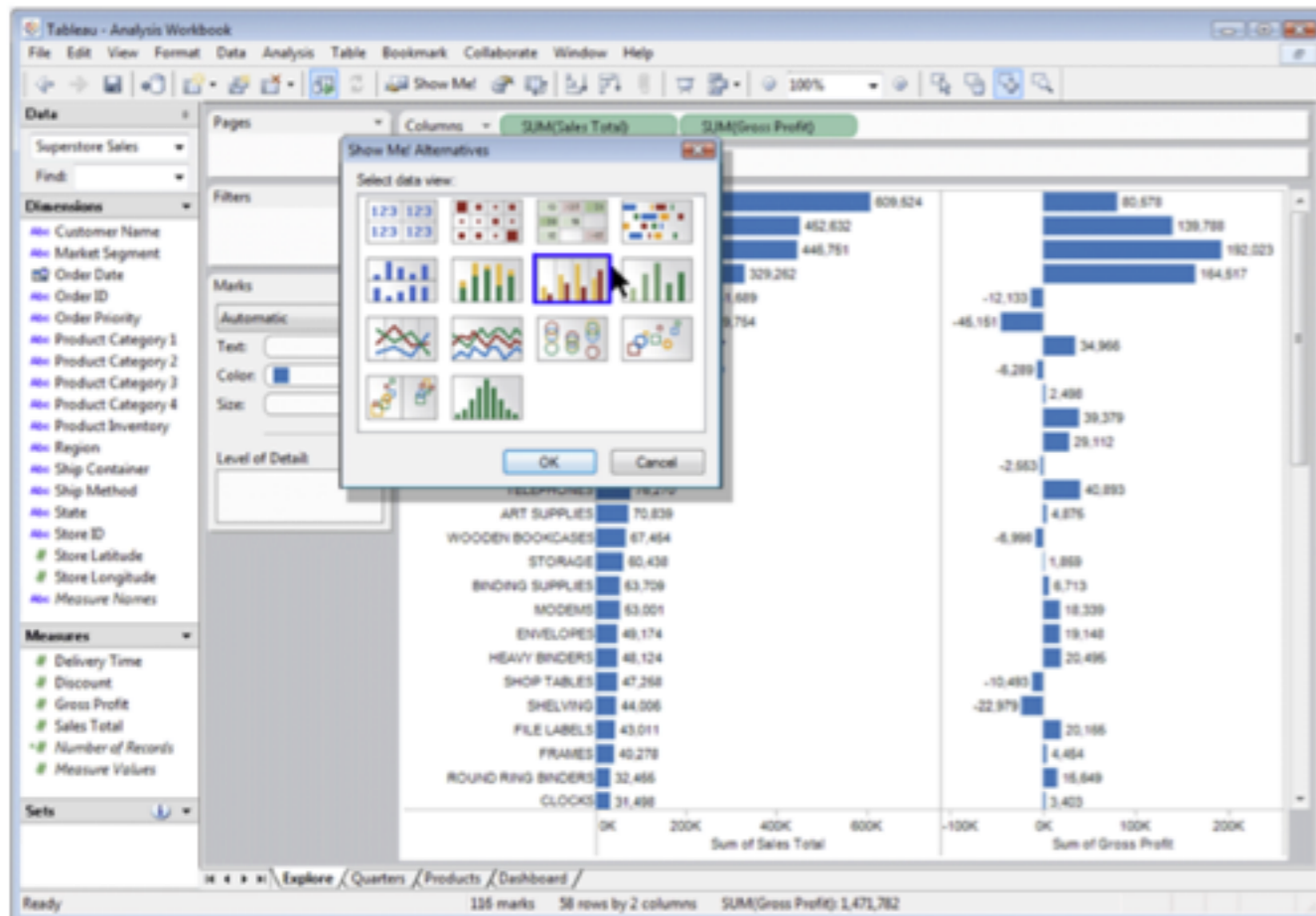
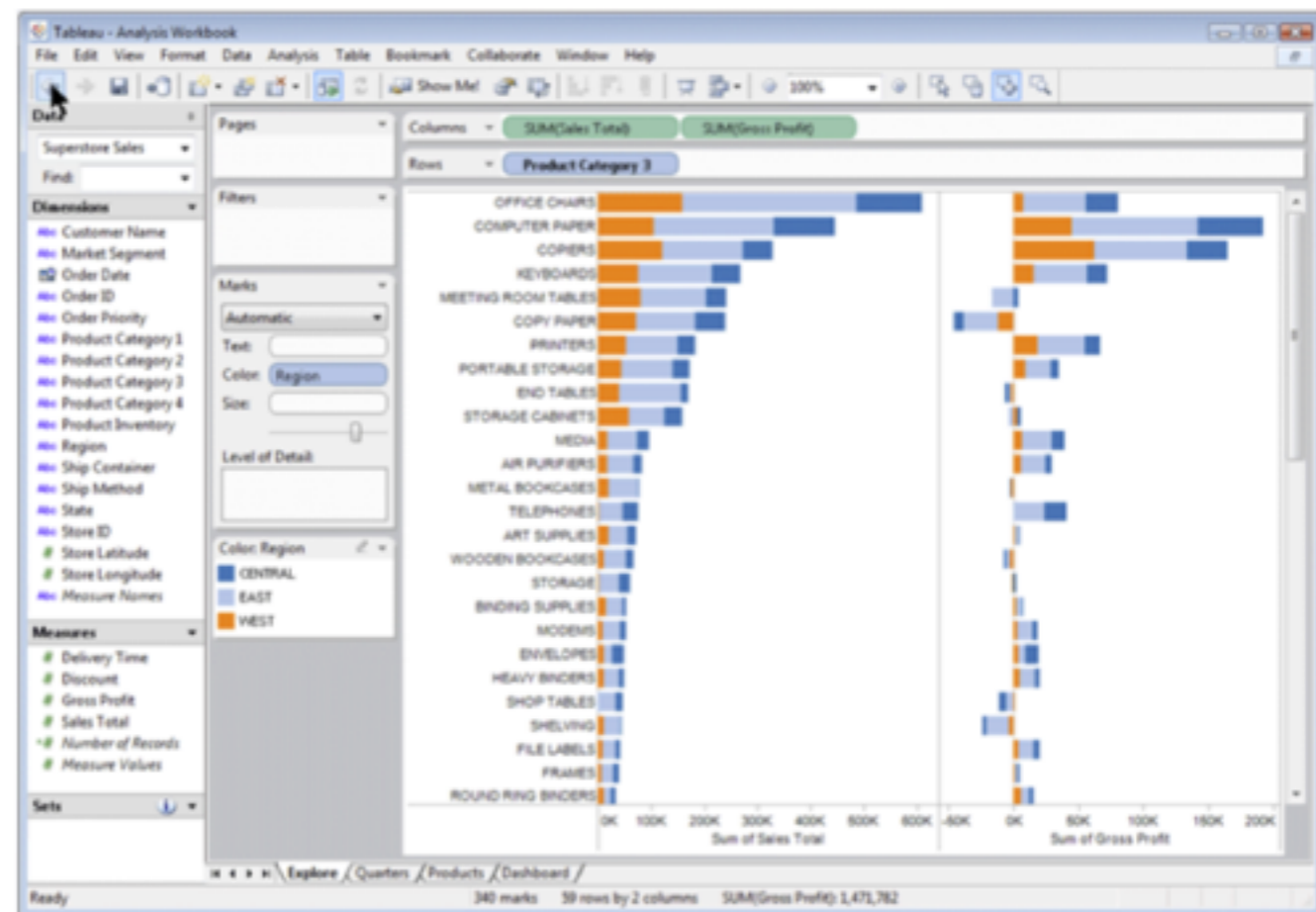
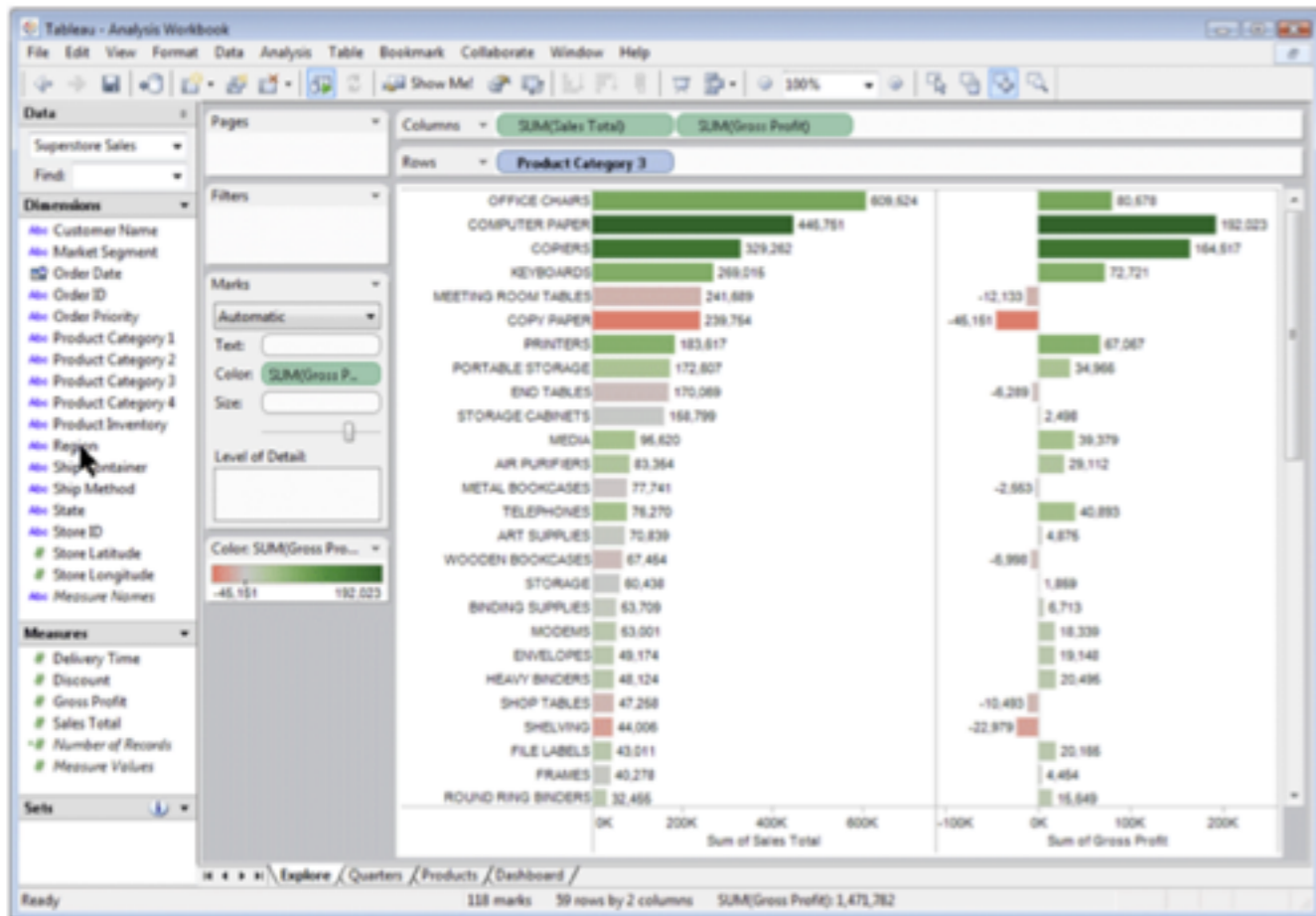
# CHANGE OVER TIME



change encoding







# animated transitions

Stacked-to-Grouped Bars

mbostock's block #3943967 October 24, 2012

## Stacked-to-Grouped Bars

Grouped  Stacked



Switch between stacked and grouped layouts using sequenced transitions! Animations preserve object constancy and allow the user to follow the data across views. Animation design by Heer and Robertson. Colors and data generation inspired by Byron and Wattenberg.

[Open in a new window.](#)

# index.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

body {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  margin: auto;
  position: relative;
  width: 960px;
}

text {
  font: 18px sans-serif;
}

.axis path,
.axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}

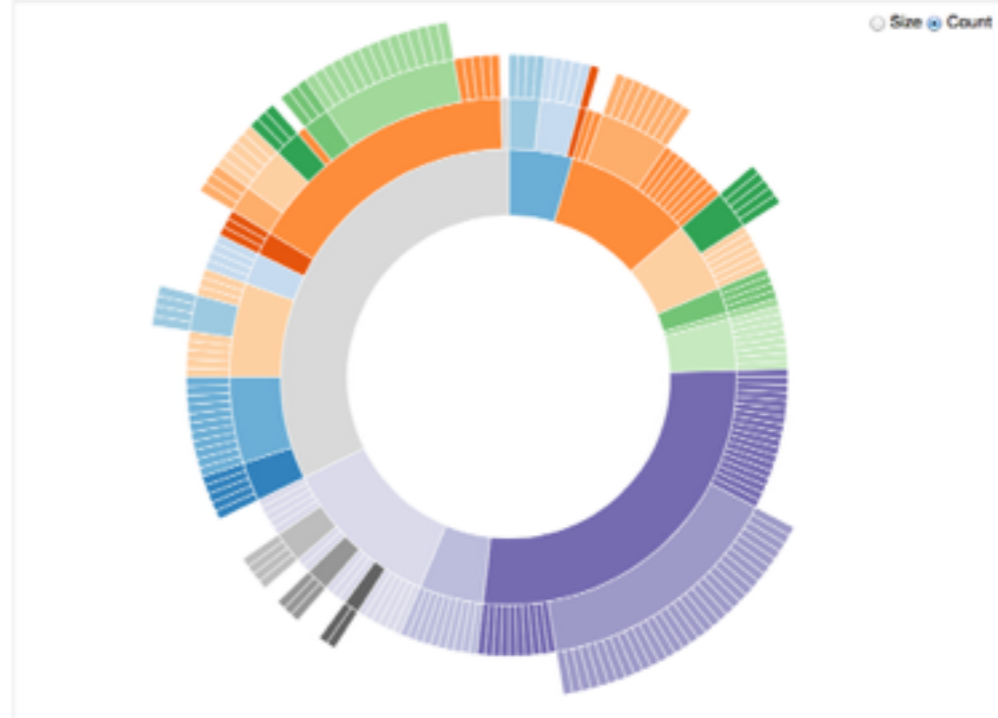
form {
  position: absolute;
  right: 10px;
  top: 10px;
}
```

Sunburst Partition

mbostock's block #4063423 November 13, 2012

## Sunburst Partition

Size  Count



A sunburst is similar to the treemap, except it uses a radial layout. The root node of the tree is at the center, with leaves on the circumference. The area (or angle, depending on implementation) of each arc corresponds to its value. Sunburst design by John Stasko. Data courtesy Jeff Heer.

[Open in a new window.](#)

# index.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

body {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  margin: auto;
  position: relative;
  width: 960px;
}

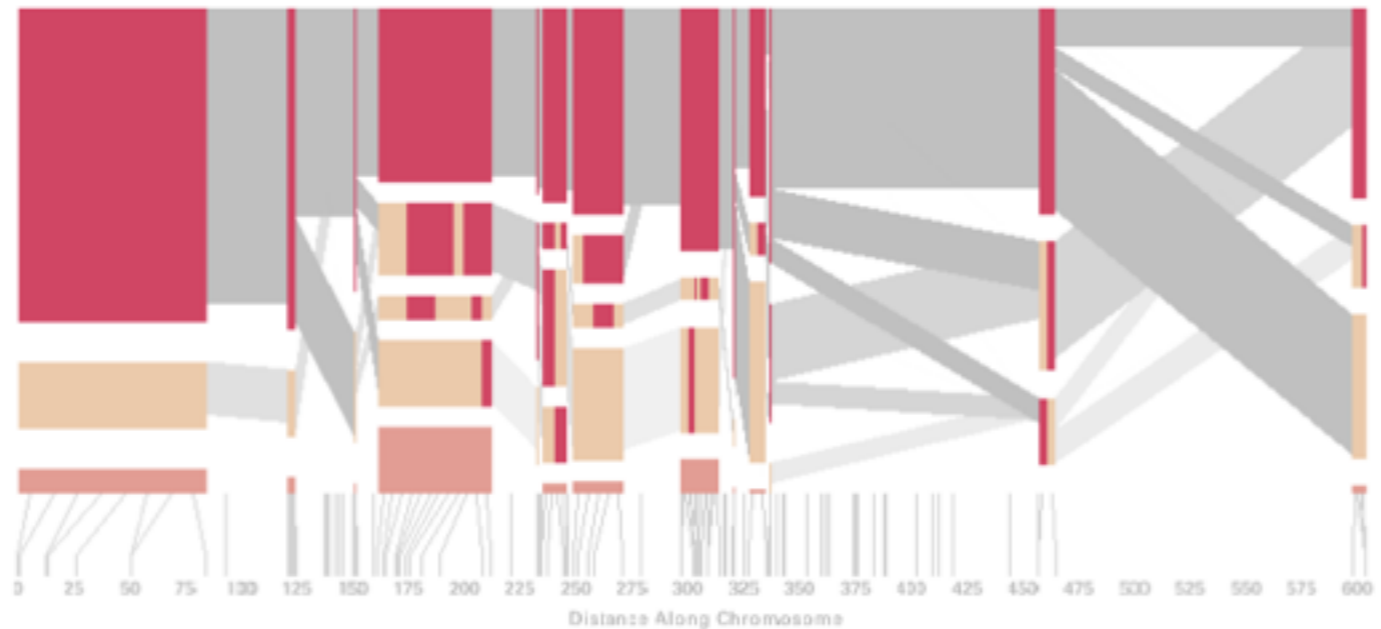
form {
  position: absolute;
  right: 10px;
  top: 10px;
}
```

<< [BENFRY](#)

## isometricblocks

When comparing the genome of two different people, you'll see single letter changes (called SNPs, pronounced "snips") every few thousand letters. An interesting feature of SNPs is that their ordering has distinct patterns, where sets of consecutive changes are most often found together. There are many methods for looking at this data, so this piece combines several of them into a single visual display. The project is described in greater detail in my [dissertation](#), starting in chapter four.

View  2D  2D Even Spacing  2D Quantitative  3D  3D with LD Units  LD Units from above



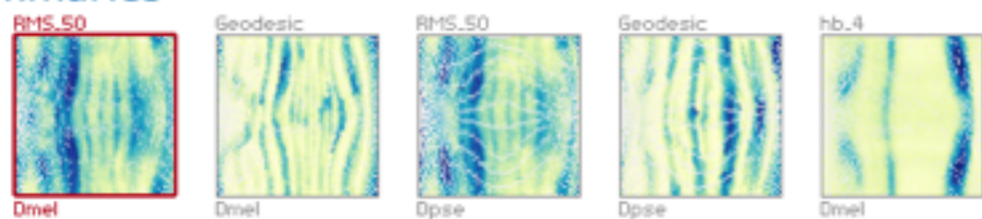
Cut High CI   98  
Cut Low CI   70  
Rec High CI   90  
Min Strong LD   95

The groupings of patterns are sometimes referred to as "haplotype blocks".

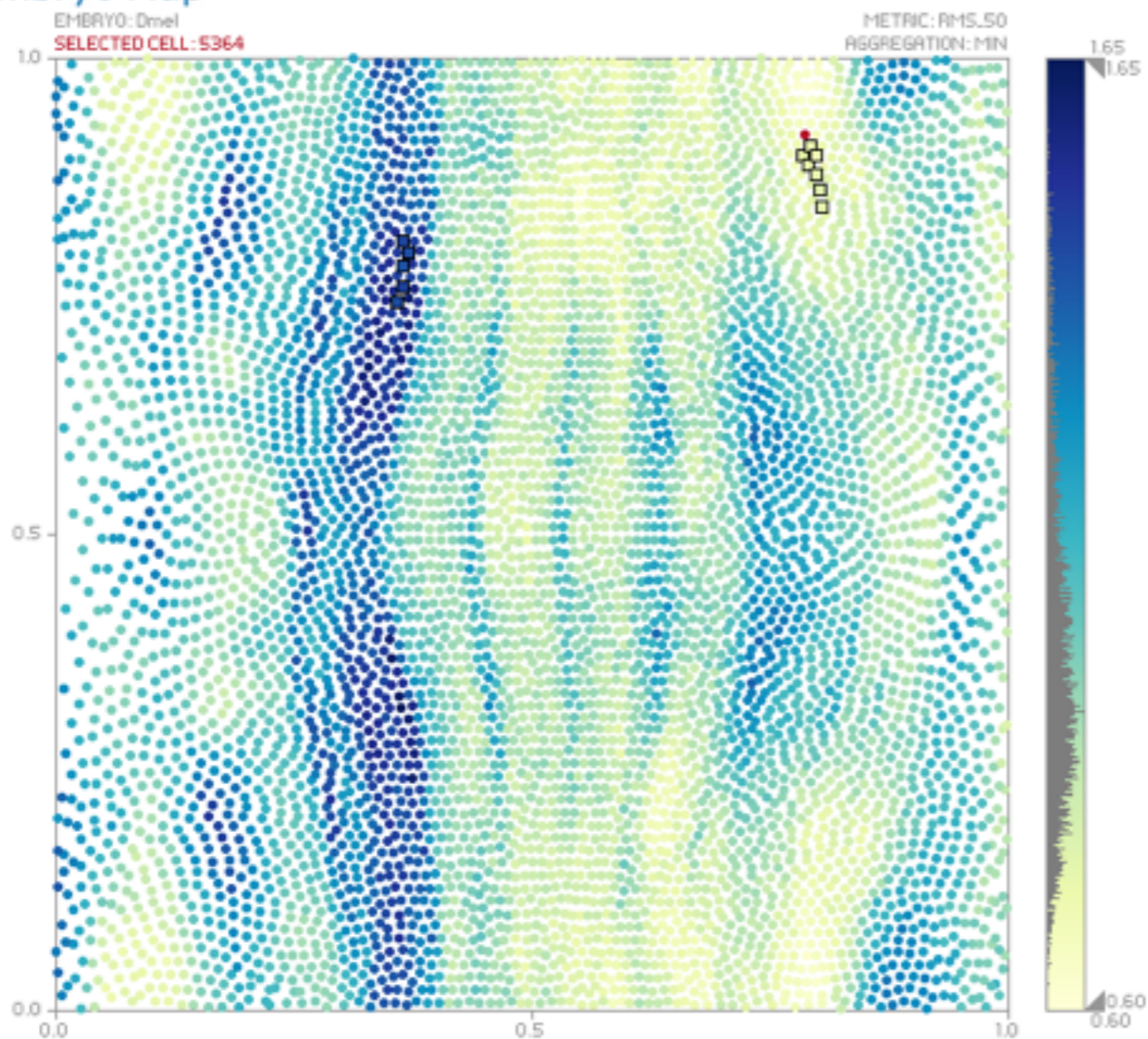
# SELECTION



### Summaries



### Embryo Map



### Curvemap

AGGREGATION GROUP | CREATED GROUP

Save Created Group: enter name [input] save [button]

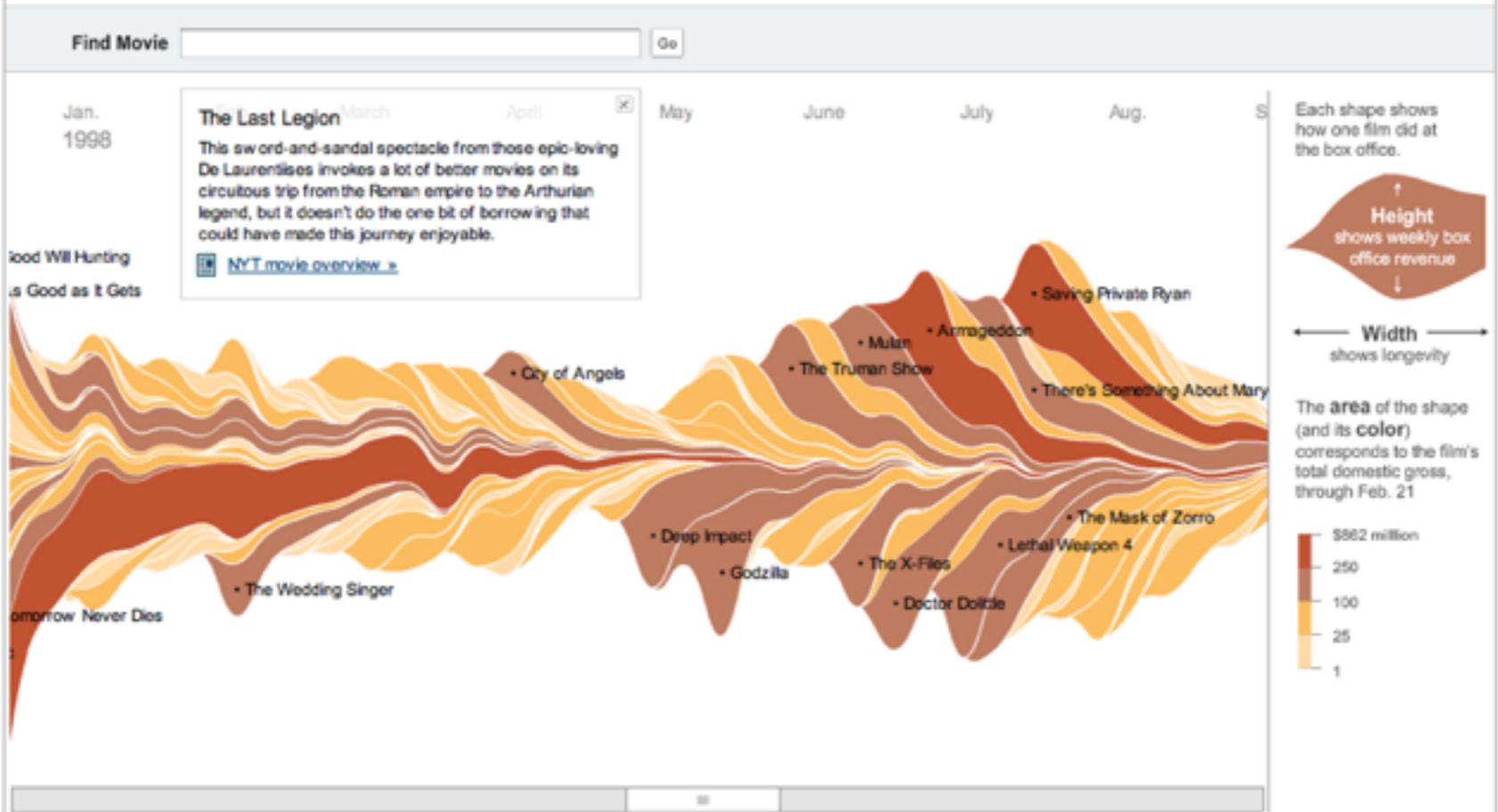
Load Existing Group: [input] load [button]

Loaded Groups: first second



# HIGHLIGHTING

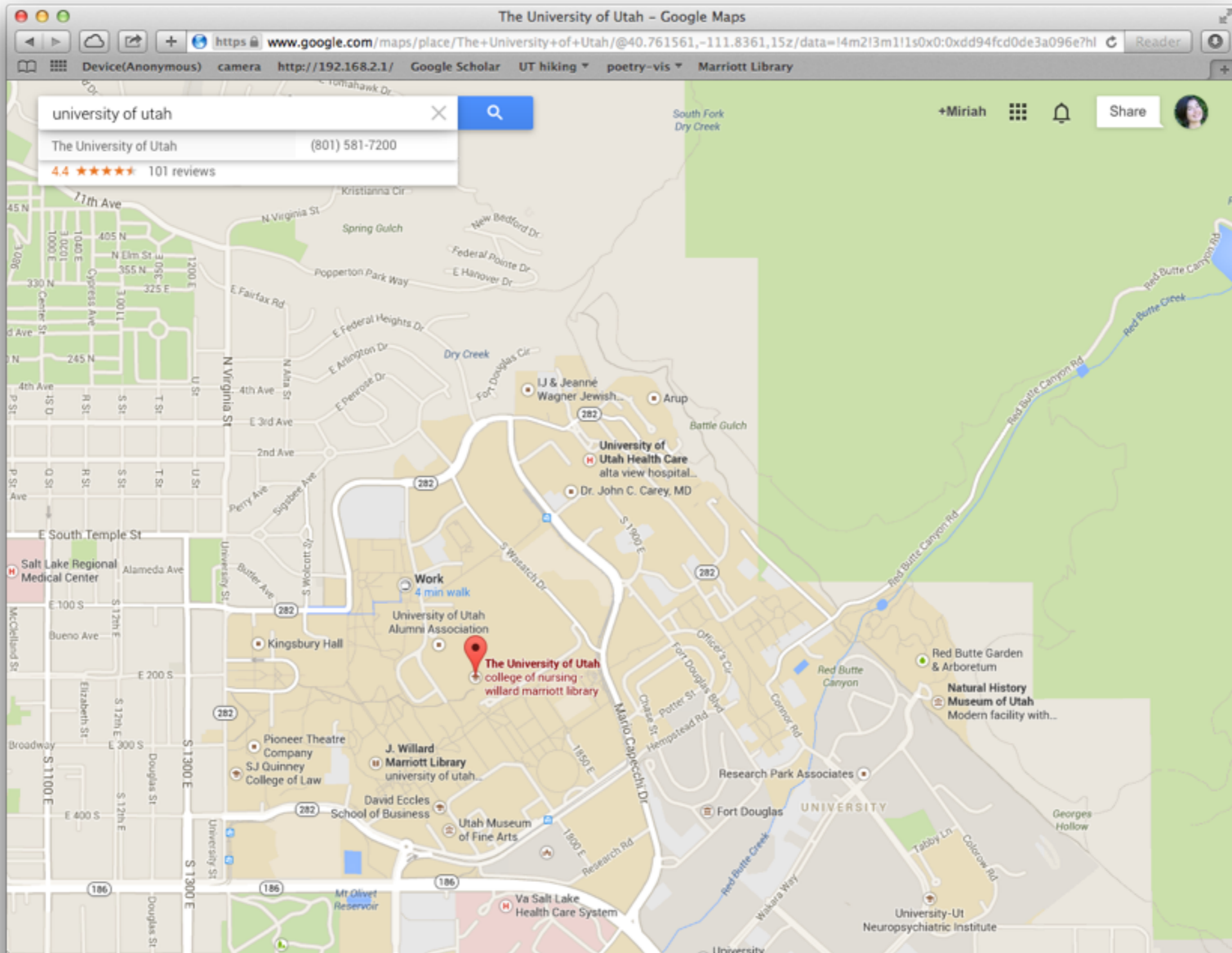
February 23, 2008    
**The Ebb and Flow of Movies: Box Office Receipts 1986 – 2008**  
 Summer blockbusters and holiday hits make up the bulk of box office revenue each year, while contenders for the Oscars tend to attract smaller audiences that build over time. Here's a look at how movies have fared at the box office, after adjusting for inflation.



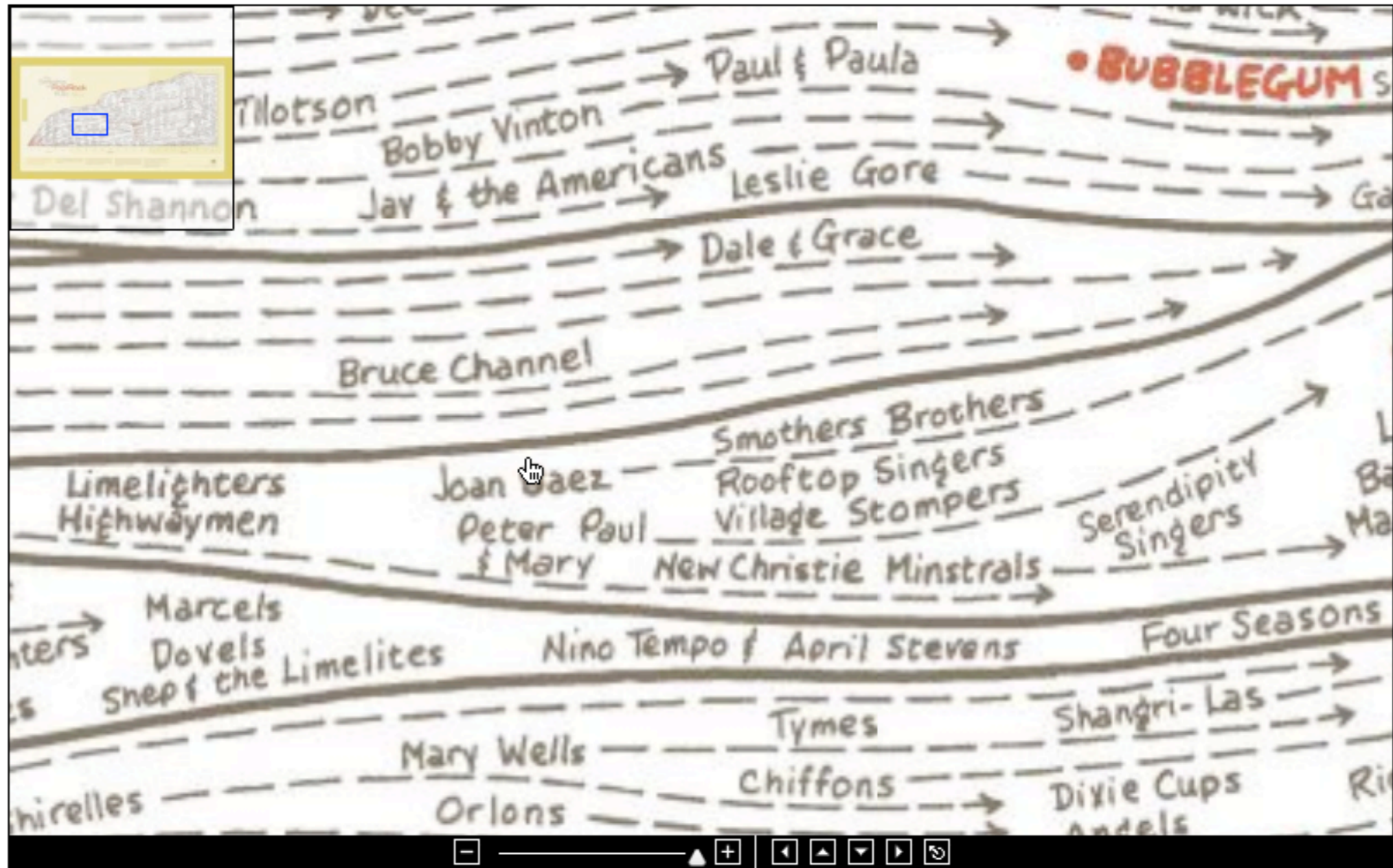


# NAVIGATION

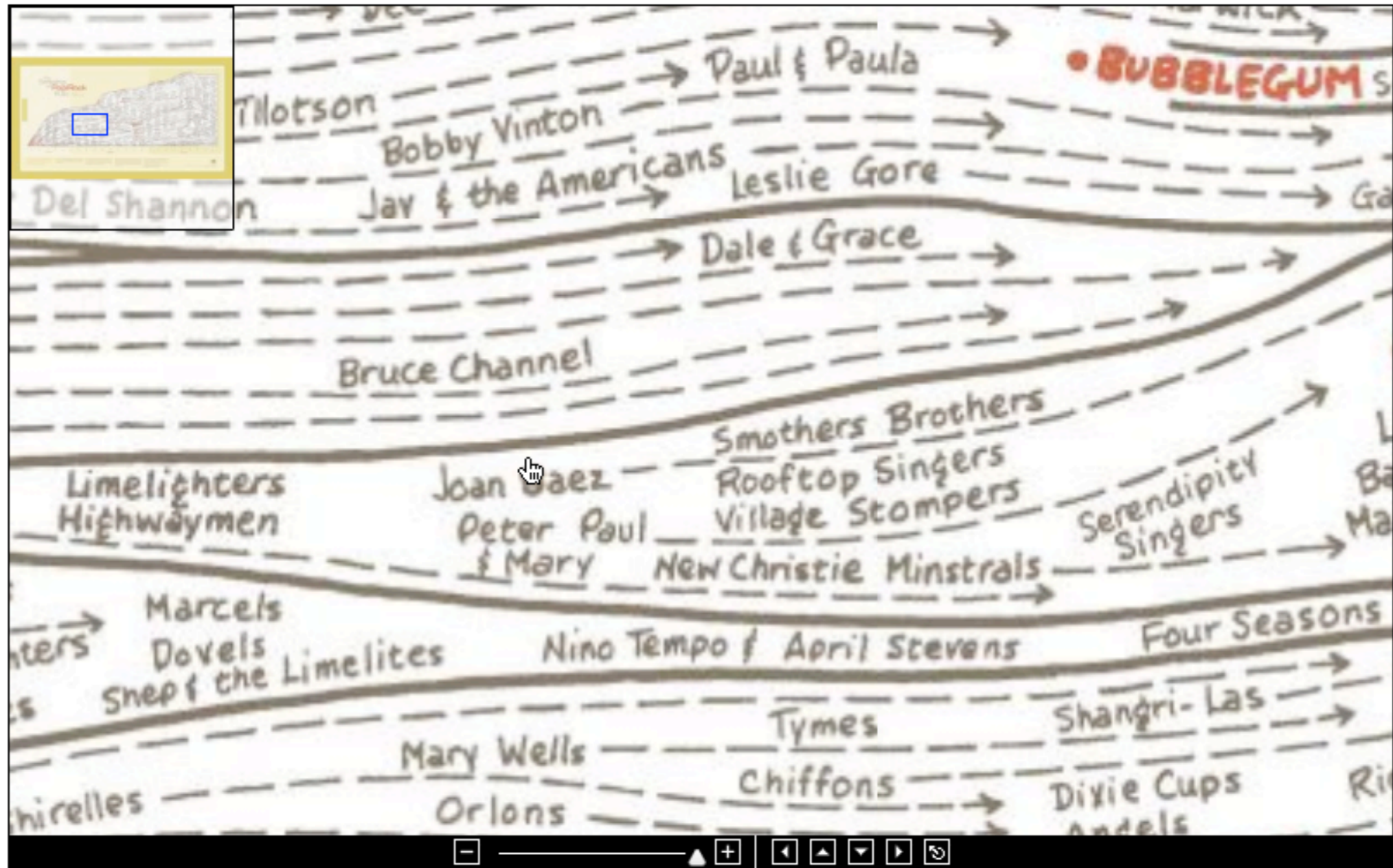
# pan (and translate)



# pan (and translate)

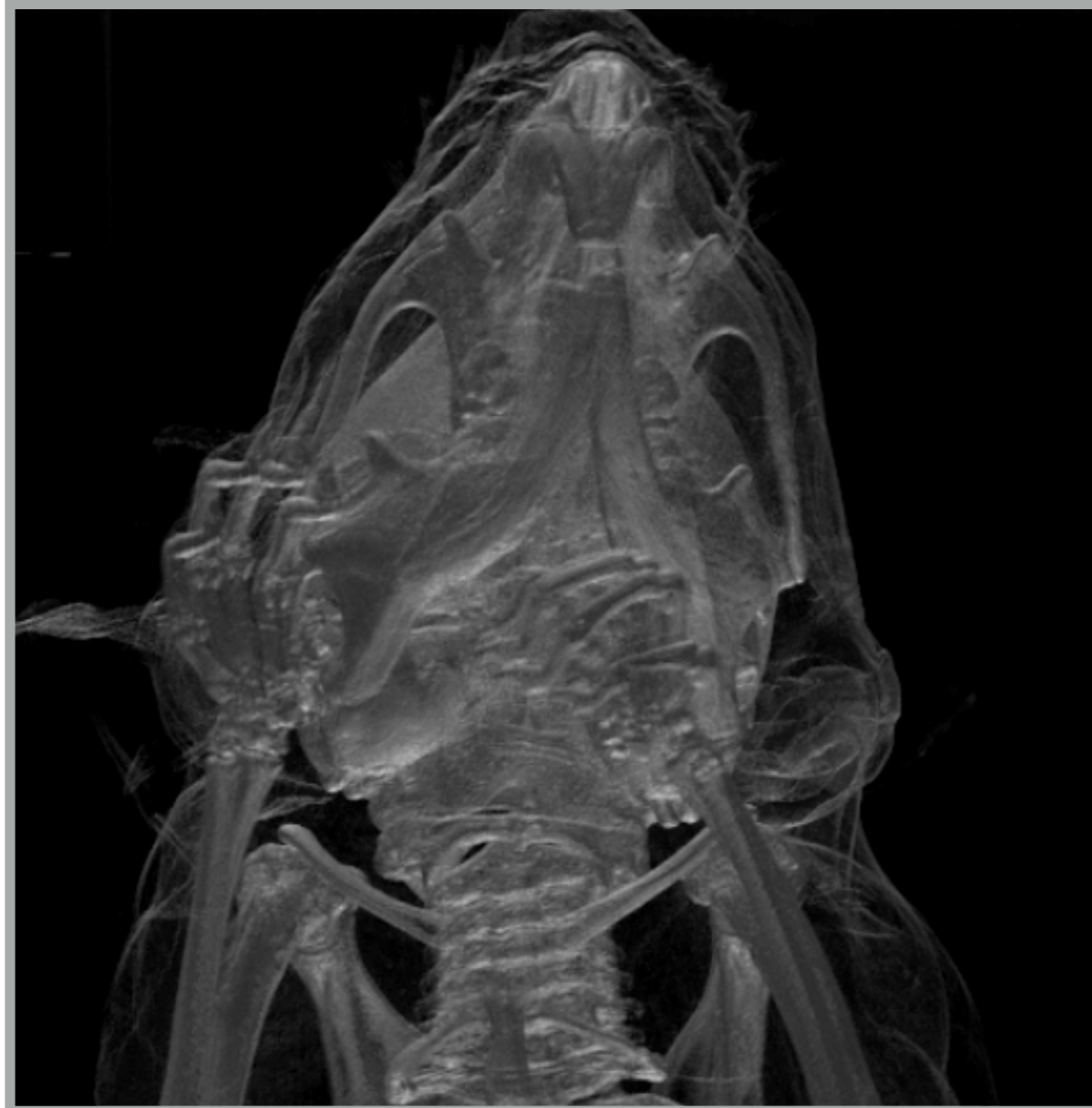


# pan (and translate)

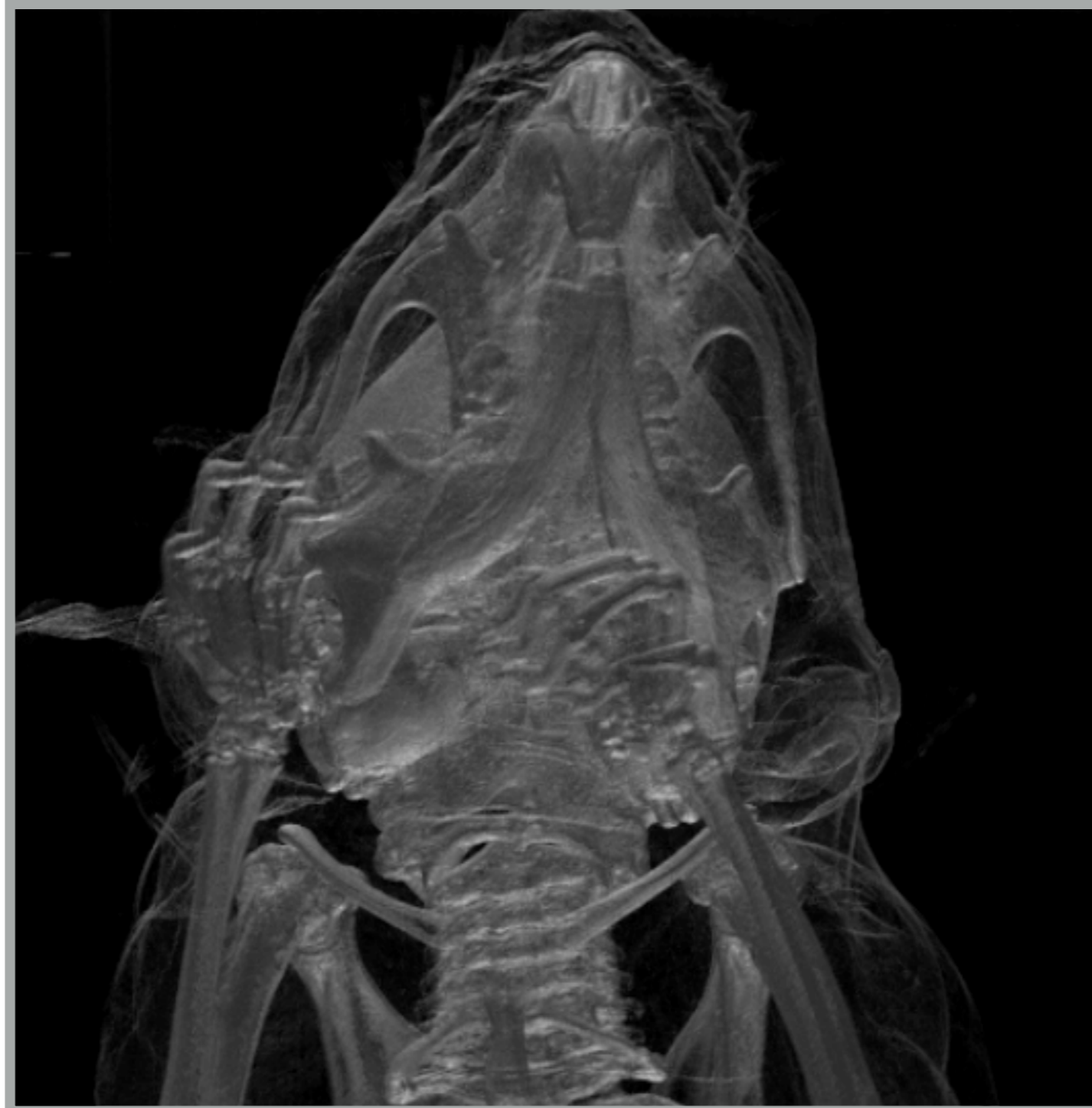




**rotate**

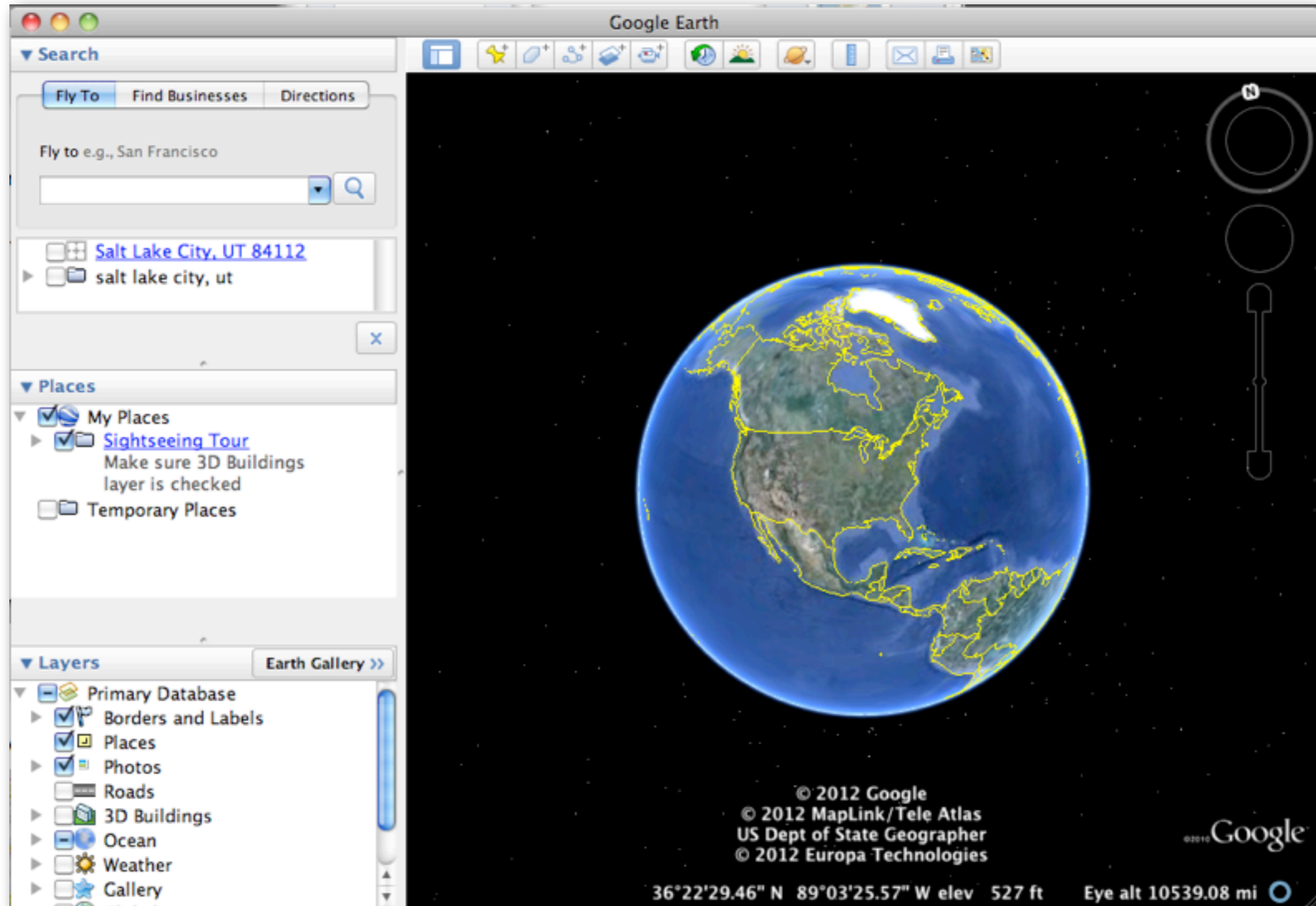


**rotate**

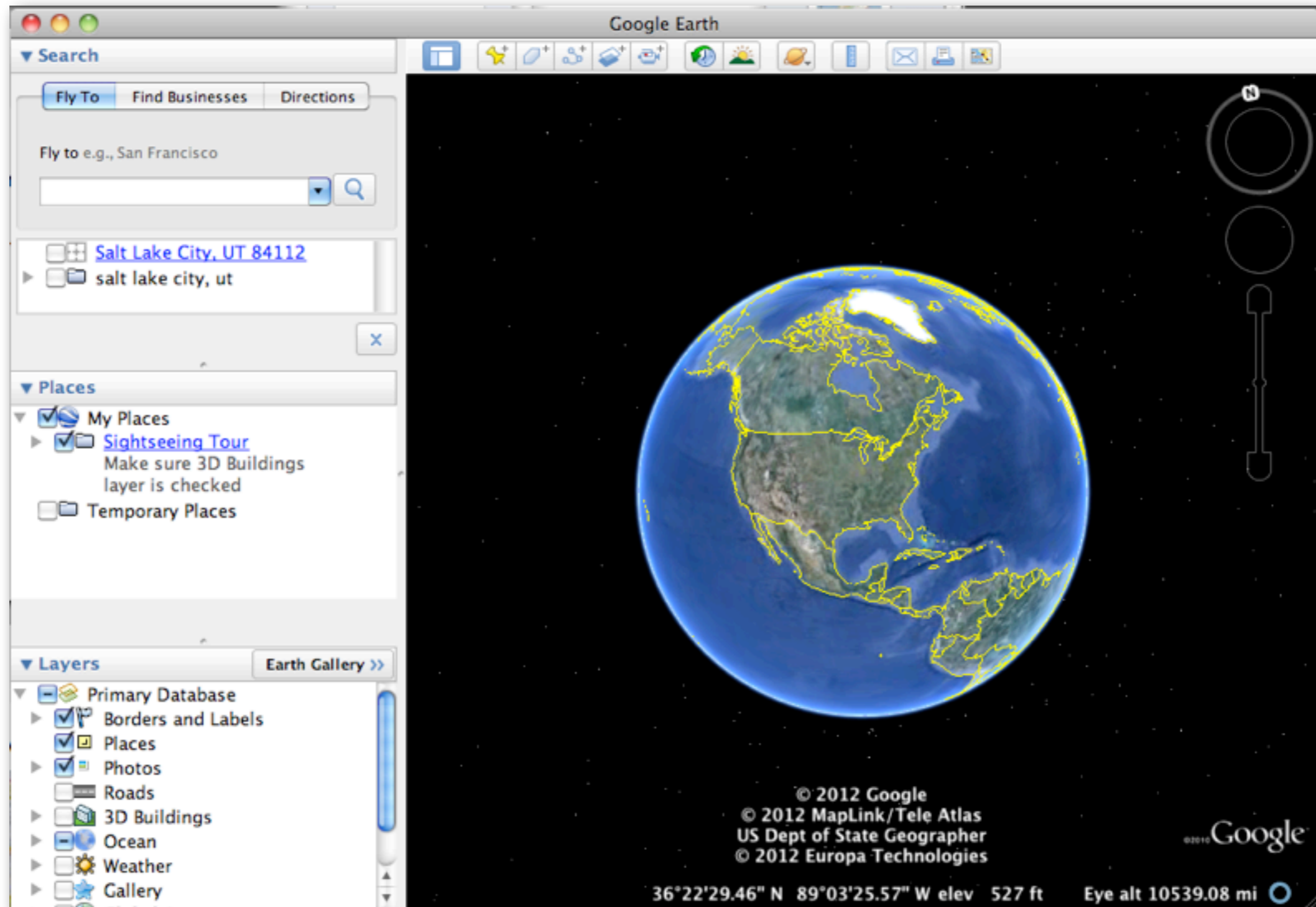


# GEOMETRIC vs SEMANTIC ZOOMING

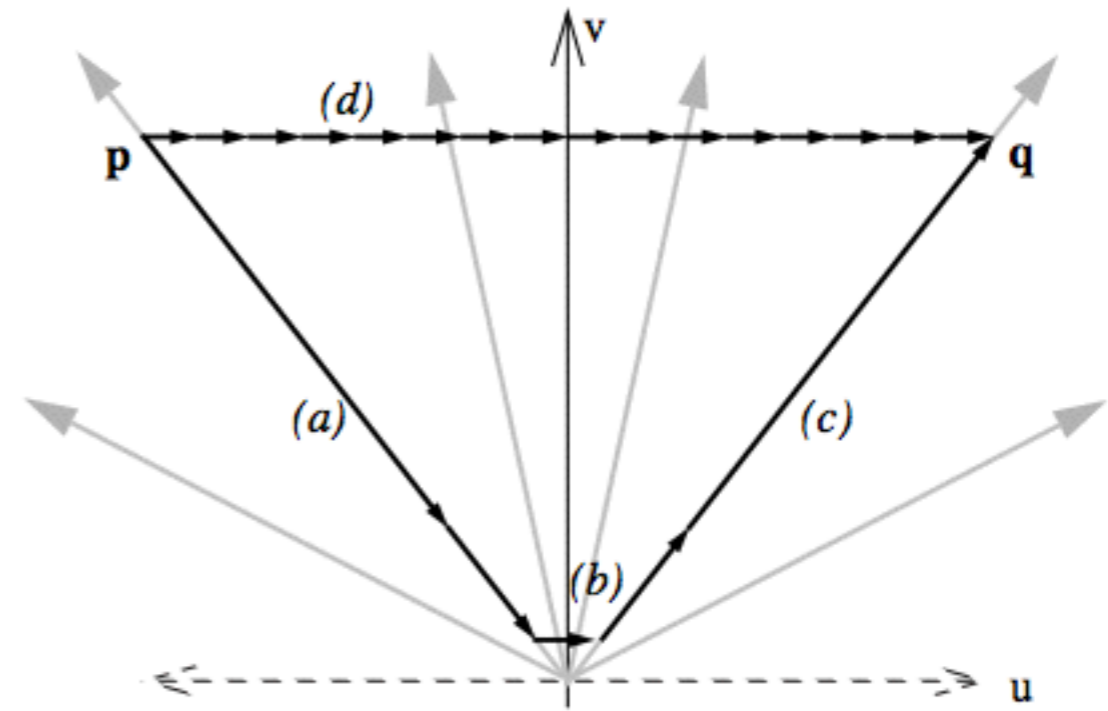
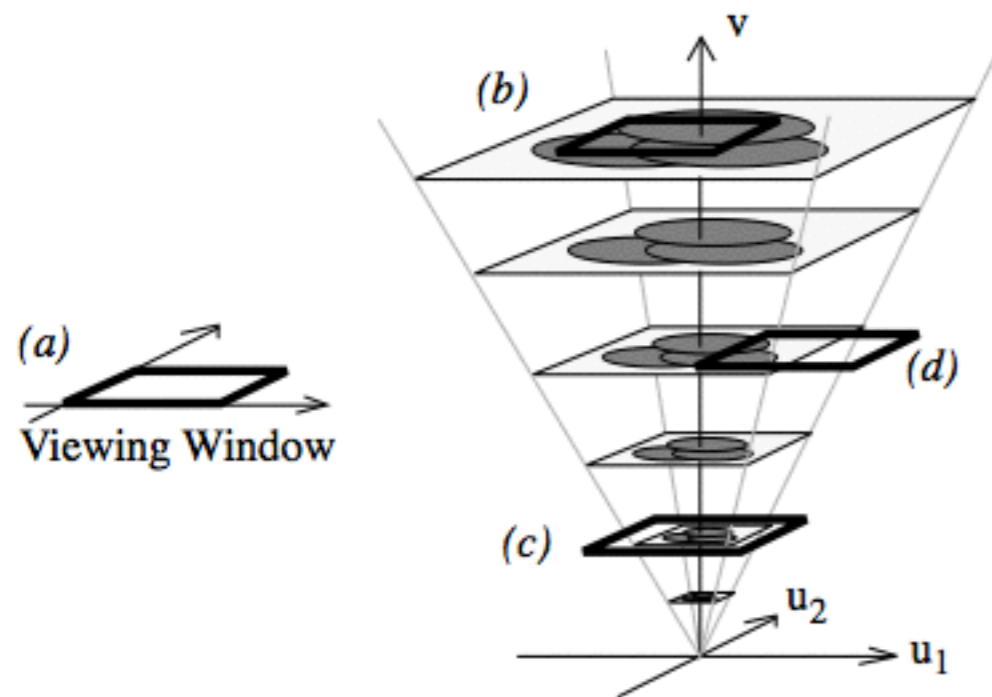
# geometric



# geometric



# SPACE-SCALE



# SPACE-SCALE DIAGRAMS: UNDERSTANDING MULTISCALE INTERFACES

*George W. Furnas*

Bellcore, 445 South Street  
Morristown, NJ 07962-1910  
(201) 829-4289  
gwf@bellcore.com

*Benjamin B. Bederson\**

Bellcore, 445 South Street  
Morristown, NJ 07962-1910  
(201) 829-4871  
bederson@bellcore.com

## ABSTRACT

Big information worlds cause big problems for interfaces. There is too much to see. They are hard to navigate. An armada of techniques has been proposed to present the many scales of information needed. Space-scale diagrams provide a framework for much of this work. By representing both a spatial world and its different magnifications explicitly, the diagrams allow the direct visualization and analysis of important scale related issues for interfaces.

**KEYWORDS:** Zoom views, multiscale interfaces, fisheye views, information visualization, GIS, visualization, user interface components, formal methods, design, etc.

## INTRODUCTION

For more than a decade there have been efforts to devise satisfactory techniques for viewing very large information worlds. (See, for example, [6] and [9] for recent reviews and analyses). The list of techniques for viewing 2D layouts alone is quite long: the Spatial Data Management System [3], Bifocal Display[1], Fisheye Views [4][12], Perspective Wall [8], the Document Lens [11], Pad [10], and Pad++ [2], the MacroScope[7], and many others.

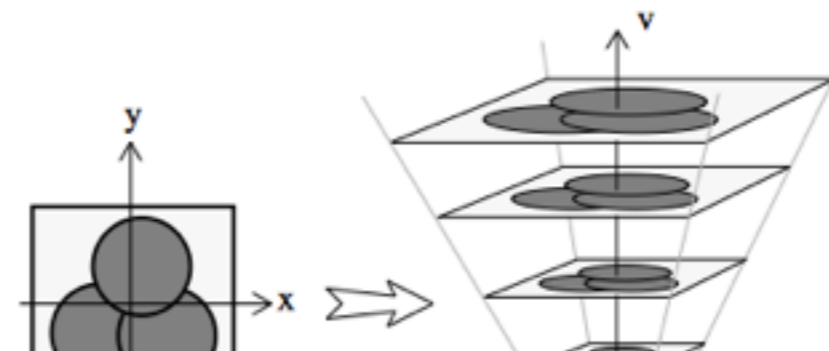
Central to most of these 2D techniques is a notion of what might be called multiscale viewing. Information objects and the structure embedding them can be displayed at many different magnifications, or scale. An interface technique is devised that allows users to manipulate which objects, or which part of the structure will be shown at what scale. The scale may be constant and manipulated over time as with a zoom metaphor, or varying over a single view as in the distortion techniques (e.g., fisheye or bifocal metaphor). In either

This paper introduces *space-scale* diagrams as a technique for understanding such multiscale interfaces. These diagrams make scale an explicit dimension of the representation, so that its place in the interface and interactions can be visualized, and better analyzed. We are finding the diagrams useful for trying to understand such interfaces geometrically, for guiding the design of code, and perhaps even as interfaces to authoring systems for multiscale information.

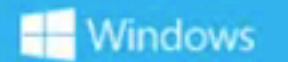
This paper will first present the necessary material for understanding the basic diagram and its properties. Subsequent sections will then use that material to show several examples of their uses.

# RECOMMENDED

The basic idea of a space-scale diagram is quite simple. Consider, for example, a square 2D picture (Figure 1a). The space-scale diagram for this picture would be obtained by creating many copies of the original 2-D picture, one at each possible magnification, and stacking them up to form an inverted pyramid (Figure 1b). While the horizontal axes rep-



# semantic

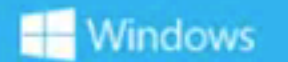


## Semantic Zoom

**Adam Barlow**, Program Manager  
Developer Experience



# semantic



## Semantic Zoom

**Adam Barlow**, Program Manager  
Developer Experience

# **semantic**

**LiveRAC: Interactive Visual Exploration of  
System Management Time-Series Data**

# **semantic**

**LiveRAC: Interactive Visual Exploration of  
System Management Time-Series Data**

L9. Views

**REQUIRED READING**

# Chapter 12

## Facet into Multiple Views

### 12.1 The Big Picture

This chapter covers choices about how to facet data across multiple views, as shown in Figure 12.1. One option for showing views is juxtapose them side by side, leading to many choices of how to coordinate these views with each other. The other option is to superimpose the views as layers on top of each other. When the views show different data, a set of choices covers how to partition data across multiple views.

The main design choices for juxtaposed views cover how to coordinate them: which visual encoding channels are shared between them, how much of the data is shared between them, and whether the navigation is synchronized. Other juxtaposition choices are when to show each view and how to arrange them. The design choices for partitioning are how many regions to use, how to divide the data up into regions, the order in which attributes are used to split, and when to stop. The design choices for how to superimpose include how elements are partitioned between layers, how many layers to use, how to distinguish them from each other, and whether the layers are static or dynamically constructed.

### 12.2 Why Facet?

The verb **facet** means to split: this chapter covers the design choices

▶ The other two ap-

► The need for justifying 3D for abstract data is covered in Section 6.3.

presence is worth the penalties of lower resolution and no workflow integration. It is very rare that immersion would be necessary for nonspatial, abstract data. Using 3D for visual encoding of abstract data is the uncommon case that needs careful justification. The use of an immersive display in this case would require even more careful justification.

## 6.7 Overview First, Zoom and Filter, Details on Demand

Ben Shneiderman's influential mantra of **Overview First, Zoom and Filter, Details on Demand** [Shneiderman 96] is a heavily cited design guideline that emphasizes the interplay between the need for overview and the need to see details, and the role of data reduction in general and navigation in particular in supporting both.

A vis idiom that provides an **overview** is intended to give the user a broad awareness of the entire information space. Using the language of the what-why-how analysis framework, it's an idiom with the goal of *summarize*. A common goal in overview design is to show all items in the dataset simultaneously, without any need for navigation to pan or scroll. Overviews help the user find regions where further investigation in more detail might be productive. Overviews are often shown at the beginning of the exploration process, to guide users in choosing where to drill down to inspect in more detail. However, overview usage is not limited to initial reconnaissance; it's very common for users to interleave the use of overviews and detail views by switching back and forth between them many times.

When the dataset is sufficiently large, some form of *reduce* action must be used in order to show everything at once. Overview creation can be understood in terms of both filtering and aggre-