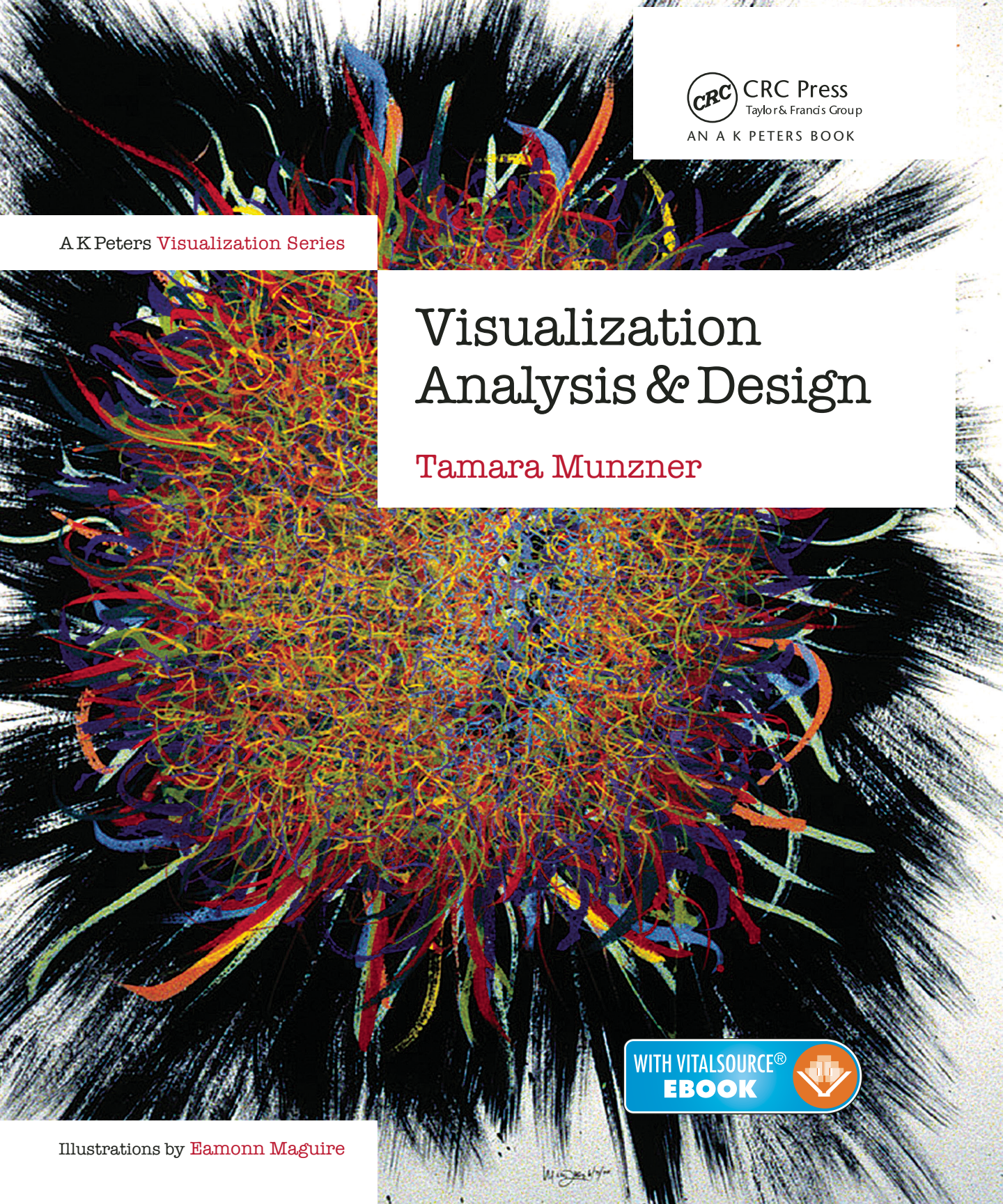

CRC Press
Taylor & Francis Group
AN A K PETERS BOOK
A K Peters Visualization Series
Visualization Analysis & Design
Tamara Munzner
WITH VITALSOURCE®
EBOOK
Illustrations by Eamonn Maguire

Visualization Analysis & Design

Tamara Munzner
Department of Computer Science
University of British Columbia

Illustrations by Eamonn Maguire

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN A K PETERS BOOK

Cover art: *Genesis 6-3-00*, by Aribert Munzner. Casein on paperboard, 26" × 20", 2000. http://www.aribertmunzner.com

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20140909

International Standard Book Number-13: 978-1-4665-0893-4 (eBook - PDF)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface xv
- Why a New Book? . . . xv
- Existing Books . . . xvi
- Audience . . . xvii
- Who's Who . . . xviii
- Structure: What's in This Book . . . xviii
- What's Not in This Book . . . xx
- Acknowledgments . . . xx

1 What's Vis, and Why Do It? 1
- 1.1 The Big Picture . . . 1
- 1.2 Why Have a Human in the Loop? . . . 2
- 1.3 Why Have a Computer in the Loop? . . . 4
- 1.4 Why Use an External Representation? . . . 6
- 1.5 Why Depend on Vision? . . . 6
- 1.6 Why Show the Data in Detail? . . . 7
- 1.7 Why Use Interactivity? . . . 9
- 1.8 Why Is the Vis Idiom Design Space Huge? . . . 10
- 1.9 Why Focus on Tasks? . . . 11
- 1.10 Why Focus on Effectiveness? . . . 11
- 1.11 Why Are Most Designs Ineffective? . . . 12
- 1.12 Why Is Validation Difficult? . . . 14
- 1.13 Why Are There Resource Limitations? . . . 14
- 1.14 Why Analyze? . . . 16
- 1.15 Further Reading . . . 18

2 What: Data Abstraction 20
- 2.1 The Big Picture . . . 21
- 2.2 Why Do Data Semantics and Types Matter? . . . 21
- 2.3 Data Types . . . 23
- 2.4 Dataset Types . . . 24
  - 2.4.1 Tables . . . 25
  - 2.4.2 Networks and Trees . . . 26
    - 2.4.2.1 Trees . . . 27

2.4.3 Fields . . . 27
2.4.3.1 Spatial Fields . . . 28
2.4.3.2 Grid Types . . . 29
2.4.4 Geometry . . . 29
2.4.5 Other Combinations . . . 30
2.4.6 Dataset Availability . . . 31
2.5 Attribute Types . . . 31
2.5.1 Categorical . . . 32
2.5.2 Ordered: Ordinal and Quantitative . . . 32
2.5.2.1 Sequential versus Diverging . . . 33
2.5.2.2 Cyclic . . . 33
2.5.3 Hierarchical Attributes . . . 33
2.6 Semantics . . . 34
2.6.1 Key versus Value Semantics . . . 34
2.6.1.1 Flat Tables . . . 34
2.6.1.2 Multidimensional Tables . . . 36
2.6.1.3 Fields . . . 37
2.6.1.4 Scalar Fields . . . 37
2.6.1.5 Vector Fields . . . 37
2.6.1.6 Tensor Fields . . . 38
2.6.1.7 Field Semantics . . . 38
2.6.2 Temporal Semantics . . . 38
2.6.2.1 Time-Varying Data . . . 39
2.7 Further Reading . . . 40

3 Why: Task Abstraction 42
3.1 The Big Picture . . . 43
3.2 Why Analyze Tasks Abstractly? . . . 43
3.3 Who: Designer or User . . . 44
3.4 Actions . . . 45
3.4.1 Analyze . . . 45
3.4.1.1 Discover . . . 47
3.4.1.2 Present . . . 47
3.4.1.3 Enjoy . . . 48
3.4.2 Produce . . . 49
3.4.2.1 Annotate . . . 49
3.4.2.2 Record . . . 49
3.4.2.3 Derive . . . 50
3.4.3 Search . . . 53
3.4.3.1 Lookup . . . 53
3.4.3.2 Locate . . . 53
3.4.3.3 Browse . . . 53
3.4.3.4 Explore . . . 54

3.4.4 Query . . . 54
3.4.4.1 Identify . . . 54
3.4.4.2 Compare . . . 55
3.4.4.3 Summarize . . . 55
3.5 Targets . . . 55
3.6 How: A Preview . . . 57
3.7 Analyzing and Deriving: Examples . . . 59
3.7.1 Comparing Two Idioms . . . 59
3.7.2 Deriving One Attribute . . . 60
3.7.3 Deriving Many New Attributes . . . 62
3.8 Further Reading . . . 64

4 Analysis: Four Levels for Validation 66
4.1 The Big Picture . . . 67
4.2 Why Validate? . . . 67
4.3 Four Levels of Design . . . 67
4.3.1 Domain Situation . . . 69
4.3.2 Task and Data Abstraction . . . 70
4.3.3 Visual Encoding and Interaction Idiom . . . 71
4.3.4 Algorithm . . . 72
4.4 Angles of Attack . . . 73
4.5 Threats to Validity . . . 74
4.6 Validation Approaches . . . 75
4.6.1 Domain Validation . . . 77
4.6.2 Abstraction Validation . . . 78
4.6.3 Idiom Validation . . . 78
4.6.4 Algorithm Validation . . . 80
4.6.5 Mismatches . . . 81
4.7 Validation Examples . . . 81
4.7.1 Genealogical Graphs . . . 81
4.7.2 MatrixExplorer . . . 83
4.7.3 Flow Maps . . . 85
4.7.4 LiveRAC . . . 87
4.7.5 LinLog . . . 89
4.7.6 Sizing the Horizon . . . 90
4.8 Further Reading . . . 91

5 Marks and Channels 94
5.1 The Big Picture . . . 95
5.2 Why Marks and Channels? . . . 95
5.3 Defining Marks and Channels . . . 95
5.3.1 Channel Types . . . 99
5.3.2 Mark Types . . . 99

5.4 Using Marks and Channels . . . 99
5.4.1 Expressiveness and Effectiveness . . . 100
5.4.2 Channel Rankings . . . 101
5.5 Channel Effectiveness . . . 103
5.5.1 Accuracy . . . 103
5.5.2 Discriminability . . . 106
5.5.3 Separability . . . 106
5.5.4 Popout . . . 109
5.5.5 Grouping . . . 111
5.6 Relative versus Absolute Judgements . . . 112
5.7 Further Reading . . . 114

6 Rules of Thumb 116
6.1 The Big Picture . . . 117
6.2 Why and When to Follow Rules of Thumb? . . . 117
6.3 No Unjustified 3D . . . 117
6.3.1 The Power of the Plane . . . 118
6.3.2 The Disparity of Depth . . . 118
6.3.3 Occlusion Hides Information . . . 120
6.3.4 Perspective Distortion Dangers . . . 121
6.3.5 Other Depth Cues . . . 123
6.3.6 Tilted Text Isn't Legibile . . . 124
6.3.7 Benefits of 3D: Shape Perception . . . 124
6.3.8 Justification and Alternatives . . . 125
Example: Cluster–Calendar Time-Series Vis . . . 125
Example: Layer-Oriented Time-Series Vis . . . 128
6.3.9 Empirical Evidence . . . 129
6.4 No Unjustified 2D . . . 131
6.5 Eyes Beat Memory . . . 131
6.5.1 Memory and Attention . . . 132
6.5.2 Animation versus Side-by-Side Views . . . 132
6.5.3 Change Blindness . . . 133
6.6 Resolution over Immersion . . . 134
6.7 Overview First, Zoom and Filter, Details on Demand . . . 135
6.8 Responsiveness Is Required . . . 137
6.8.1 Visual Feedback . . . 138
6.8.2 Latency and Interaction Design . . . 138
6.8.3 Interactivity Costs . . . 140
6.9 Get It Right in Black and White . . . 140
6.10 Function First, Form Next . . . 140
6.11 Further Reading . . . 141

7 Arrange Tables 144
7.1 The Big Picture . . . 145
7.2 Why Arrange? . . . 145
7.3 Arrange by Keys and Values . . . 145
7.4 Express: Quantitative Values . . . 146
Example: Scatterplots . . . 146
7.5 Separate, Order, and Align: Categorical Regions . . . 149
7.5.1 List Alignment: One Key . . . 149
Example: Bar Charts . . . 150
Example: Stacked Bar Charts . . . 151
Example: Streamgraphs . . . 153
Example: Dot and Line Charts . . . 155
7.5.2 Matrix Alignment: Two Keys . . . 157
Example: Cluster Heatmaps . . . 158
Example: Scatterplot Matrix . . . 160
7.5.3 Volumetric Grid: Three Keys . . . 161
7.5.4 Recursive Subdivision: Multiple Keys . . . 161
7.6 Spatial Axis Orientation . . . 162
7.6.1 Rectilinear Layouts . . . 162
7.6.2 Parallel Layouts . . . 162
Example: Parallel Coordinates . . . 162
7.6.3 Radial Layouts . . . 166
Example: Radial Bar Charts . . . 167
Example: Pie Charts . . . 168
7.7 Spatial Layout Density . . . 171
7.7.1 Dense . . . 172
Example: Dense Software Overviews . . . 172
7.7.2 Space-Filling . . . 174
7.8 Further Reading . . . 175

8 Arrange Spatial Data 178
8.1 The Big Picture . . . 179
8.2 Why Use Given? . . . 179
8.3 Geometry . . . 180
8.3.1 Geographic Data . . . 180
Example: Choropleth Maps . . . 181
8.3.2 Other Derived Geometry . . . 182
8.4 Scalar Fields: One Value . . . 182
8.4.1 Isocontours . . . 183
Example: Topographic Terrain Maps . . . 183
Example: Flexible Isosurfaces . . . 185
8.4.2 Direct Volume Rendering . . . 186
Example: Multidimensional Transfer Functions . . . 187

8.5 Vector Fields: Multiple Values . . . 189
8.5.1 Flow Glyphs . . . 191
8.5.2 Geometric Flow . . . 191
Example: Similarity-Clustered Streamlines . . . 192
8.5.3 Texture Flow . . . 193
8.5.4 Feature Flow . . . 193
8.6 Tensor Fields: Many Values . . . 194
Example: Ellipsoid Tensor Glyphs . . . 194
8.7 Further Reading . . . 197

9 Arrange Networks and Trees 200
9.1 The Big Picture . . . 201
9.2 Connection: Link Marks . . . 201
Example: Force-Directed Placement . . . 204
Example: sfdp . . . 207
9.3 Matrix Views . . . 208
Example: Adjacency Matrix View . . . 208
9.4 Costs and Benefits: Connection versus Matrix . . . 209
9.5 Containment: Hierarchy Marks . . . 213
Example: Treemaps . . . 213
Example: GrouseFlocks . . . 215
9.6 Further Reading . . . 216

10 Map Color and Other Channels 218
10.1 The Big Picture . . . 219
10.2 Color Theory . . . 219
10.2.1 Color Vision . . . 219
10.2.2 Color Spaces . . . 220
10.2.3 Luminance, Saturation, and Hue . . . 223
10.2.4 Transparency . . . 225
10.3 Colormaps . . . 225
10.3.1 Categorical Colormaps . . . 226
10.3.2 Ordered Colormaps . . . 229
10.3.3 Bivariate Colormaps . . . 234
10.3.4 Colorblind-Safe Colormap Design . . . 235
10.4 Other Channels . . . 236
10.4.1 Size Channels . . . 236
10.4.2 Angle Channel . . . 237
10.4.3 Curvature Channel . . . 238
10.4.4 Shape Channel . . . 238
10.4.5 Motion Channels . . . 238
10.4.6 Texture and Stippling . . . 239
10.5 Further Reading . . . 240

11 Manipulate View 242
11.1 The Big Picture . . . 243
11.2 Why Change? . . . 244
11.3 Change View over Time . . . 244
Example: LineUp . . . 246
Example: Animated Transitions . . . 248
11.4 Select Elements . . . 249
11.4.1 Selection Design Choices . . . 250
11.4.2 Highlighting . . . 251
Example: Context-Preserving Visual Links . . . 253
11.4.3 Selection Outcomes . . . 254
11.5 Navigate: Changing Viewpoint . . . 254
11.5.1 Geometric Zooming . . . 255
11.5.2 Semantic Zooming . . . 255
11.5.3 Constrained Navigation . . . 256
11.6 Navigate: Reducing Attributes . . . 258
11.6.1 Slice . . . 258
Example: HyperSlice . . . 259
11.6.2 Cut . . . 260
11.6.3 Project . . . 261
11.7 Further Reading . . . 261

12 Facet into Multiple Views 264
12.1 The Big Picture . . . 265
12.2 Why Facet? . . . 265
12.3 Juxtapose and Coordinate Views . . . 267
12.3.1 Share Encoding: Same/Different . . . 267
Example: Exploratory Data Visualizer (EDV) . . . 268
12.3.2 Share Data: All, Subset, None . . . 269
Example: Bird's-Eye Maps . . . 270
Example: Multiform Overview–Detail Microarrays . . . 271
Example: Cerebral . . . 274
12.3.3 Share Navigation: Synchronize . . . 276
12.3.4 Combinations . . . 276
Example: Improvise . . . 277
12.3.5 Juxtapose Views . . . 278
12.4 Partition into Views . . . 279
12.4.1 Regions, Glyphs, and Views . . . 279
12.4.2 List Alignments . . . 281
12.4.3 Matrix Alignments . . . 282
Example: Trellis . . . 282
12.4.4 Recursive Subdivision . . . 285
12.5 Superimpose Layers . . . 288

12.5.1 Visually Distinguishable Layers . . . 289
12.5.2 Static Layers . . . 289
Example: Cartographic Layering . . . 289
Example: Superimposed Line Charts . . . 290
Example: Hierarchical Edge Bundles . . . 292
12.5.3 Dynamic Layers . . . 294
12.6 Further Reading . . . 295

13 Reduce Items and Attributes 298
13.1 The Big Picture . . . 299
13.2 Why Reduce? . . . 299
13.3 Filter . . . 300
13.3.1 Item Filtering . . . 301
Example: FilmFinder . . . 301
13.3.2 Attribute Filtering . . . 303
Example: DOSFA . . . 304
13.4 Aggregate . . . 305
13.4.1 Item Aggregation . . . 305
Example: Histograms . . . 306
Example: Continuous Scatterplots . . . 307
Example: Boxplot Charts . . . 308
Example: SolarPlot . . . 310
Example: Hierarchical Parallel Coordinates . . . 311
13.4.2 Spatial Aggregation . . . 313
Example: Geographically Weighted Boxplots . . . 313
13.4.3 Attribute Aggregation: Dimensionality Reduction . . . 315
13.4.3.1 Why and When to Use DR? . . . 316
Example: Dimensionality Reduction for Document Collections . . . 316
13.4.3.2 How to Show DR Data? . . . 319
13.5 Further Reading . . . 320

14 Embed: Focus+Context 322
14.1 The Big Picture . . . 323
14.2 Why Embed? . . . 323
14.3 Elide . . . 324
Example: DOITrees Revisited . . . 325
14.4 Superimpose . . . 326
Example: Toolglass and Magic Lenses . . . 326
14.5 Distort . . . 327
Example: 3D Perspective . . . 327
Example: Fisheye Lens . . . 328
Example: Hyperbolic Geometry . . . 329

Example: Stretch and Squish Navigation 331
Example: Nonlinear Magnification Fields 333
14.6 Costs and Benefits: Distortion 334
14.7 Further Reading 337

15 Analysis Case Studies 340
15.1 The Big Picture 341
15.2 Why Analyze Case Studies? 341
15.3 Graph-Theoretic Scagnostics 342
15.4 VisDB 347
15.5 Hierarchical Clustering Explorer 351
15.6 PivotGraph 355
15.7 InterRing 358
15.8 Constellation 360
15.9 Further Reading 366

Figure Credits 369

Bibliography 375

This page intentionally left blank

Chapter 2

What: Data Abstraction

2.1 The Big Picture

Figure 2.1 shows the abstract types of *what* can be visualized. The four basic dataset types are tables, networks, fields, and geometry; other possible collections of items include clusters, sets, and lists. These datasets are made up of different combinations of the five data types: items, attributes, links, positions, and grids. For any of these dataset types, the full dataset could be available immediately in the form of a static file, or it might be dynamic data processed gradually in the form of a stream. The type of an attribute can be categorical or ordered, with a further split into ordinal and quantitative. The ordering direction of attributes can be sequential, diverging, or cyclic.

2.2 Why Do Data Semantics and Types Matter?

Many aspects of vis design are driven by the kind of data that you have at your disposal. What kind of data are you given? What information can you figure out from the data, versus the meanings that you must be told explicitly? What high-level concepts will allow you to split datasets apart into general and useful pieces?

Suppose that you see the following data:

```
14, 2.6, 30, 30, 15, 100001
```

What does this sequence of six numbers mean? You can't possibly know yet, without more information about how to interpret each number. Is it locations for two points far from each other in three-dimensional space, $14, 2.6, 30$ and $30, 15, 100001$? Is it two points closer to each other in two-dimensional space, $14, 2.6$ and

30, 30, with the fifth number meaning that there are 15 links between these two points, and the sixth number assigning the weight of '100001' to that link?

Similarly, suppose that you see the following data:

```
Basil, 7, S, Pear
```

These numbers and words could have many possible meanings. Maybe a food shipment of produce has arrived in satisfactory condition on the 7th day of the month, containing basil and pears. Maybe the Basil Point neighborhood of the city has had 7 inches of snow cleared by the Pear Creek Limited snow removal service. Maybe the lab rat named Basil has made seven attempts to find his way through the south section of the maze, lured by scent of the reward food for this trial, a pear.

To move beyond guesses, you need to know two crosscutting pieces of information about these terms: their semantics and their types. The **semantics** of the data is its real-world meaning. For instance, does a word represent a human first name, or is it the shortened version of a company name where the full name can be looked up in an external list, or is it a city, or is it a fruit? Does a number represent a day of the month, or an age, or a measurement of height, or a unique code for a specific person, or a postal code for a neighborhood, or a position in space?

The **type** of the data is its structural or mathematical interpretation. At the data level, what kind of thing is it: an item, a link, an attribute? At the dataset level, how are these data types combined into a larger structure: a table, a tree, a field of sampled values? At the attribute level, what kinds of mathematical operations are meaningful for it? For example, if a number represents a count of boxes of detergent, then its type is a quantity, and adding two such numbers together makes sense. If the number represents a postal code, then its type is a *code* rather than a *quantity*—it is simply the name for a *category* that happens to be a number rather than a textual name. Adding two of these numbers together does not make sense.

Table 2.1 shows several more lines of the same dataset. This simple example table is tiny, with only nine rows and four columns. The exact semantics should be provided by the creator of the dataset; I give it with the column titles. In this case, each person has a unique identifier, a name, an age, a shirt size, and a favorite fruit.

Sometimes types and semantics can be correctly inferred simply by observing the syntax of a data file or the names of variables

ID	Name	Age	Shirt Size	Favorite Fruit
1	Amy	8	S	Apple
2	Basil	7	S	Pear
3	Clara	9	M	Durian
4	Desmond	13	L	Elderberry
5	Ernest	12	L	Peach
6	Fanny	10	S	Lychee
7	George	9	M	Orange
8	Hector	8	L	Loquat
9	Ida	10	M	Pear
10	Amy	12	M	Orange

Table 2.1. A full table with column titles that provide the intended semantics of the attributes.

within it, but often they must be provided along with the dataset in order for it to be interpreted correctly. Sometimes this kind of additional information is called **metadata**; the line between data and metadata is not clear, especially given that the original data is often derived and transformed. In this book, I don't distinguish between them, and refer to everything as *data*.

► Deriving data is discussed in Section 3.4.2.3.

The classification below presents a way to think about dataset and attribute types and semantics in a way that is general enough to cover the cases interesting in vis, yet specific enough to be helpful for guiding design choices at the abstraction and idiom levels.

2.3 Data Types

Figure 2.2 shows the five basic **data types** discussed in this book: items, attributes, links, positions, and grids. An **attribute** is some specific property that can be measured, observed, or logged.* For example, attributes could be salary, price, number of sales, protein expression levels, or temperature. An **item** is an individual entity that is discrete, such as a row in a simple table or a node

★ Synonyms for *attribute* are **variable** and **data dimension**, or just **dimension** for short. Since *dimension* has many meanings, in this book it is reserved for the visual channels of spatial position as discussed in Section 6.3.

Figure 2.2. The five basic data types: items, attributes, links, positions, and grids.

in a network. For example, items may be people, stocks, coffee shops, genes, or cities. A **link** is a relationship between items, typically within a network. A **grid** specifies the strategy for sampling continuous data in terms of both geometric and topological relationships between its cells. A **position** is spatial data, providing a location in two-dimensional (2D) or three-dimensional (3D) space. For example, a position might be a latitude–longitude pair describing a location on the Earth's surface or three numbers specifying a location within the region of space measured by a medical scanner.

2.4 Dataset Types

★ The word *dataset* is singular. In vis the word **data** is commonly used as a singular mass noun as well, in contrast to the traditional usage in the natural sciences where *data* is plural.

A **dataset** is any collection of information that is the target of analysis.* The four basic **dataset types** are tables, networks, fields, and geometry. Other ways to group items together include clusters, sets, and lists. In real-world situations, complex combinations of these basic types are common.

Figure 2.3 shows that these basic dataset types arise from combinations of the data types of items, attributes, links, positions, and grids.

Figure 2.4 shows the internal structure of the four basic dataset types in detail. Tables have cells indexed by items and attributes, for either the simple flat case or the more complex multidimensional case. In a network, items are usually called nodes, and they are connected with links; a special case of networks is trees. Continuous fields have grids based on spatial positions where cells contain attributes. Spatial geometry has only position information.

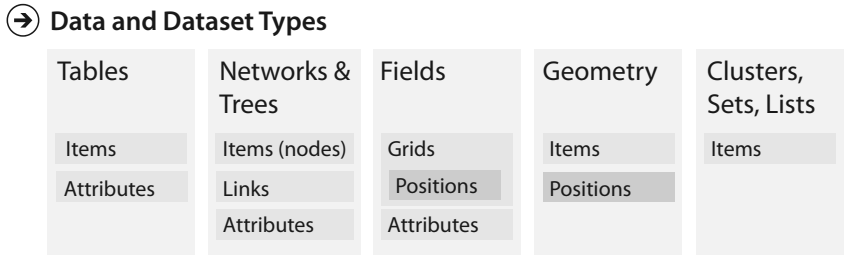


Figure 2.3. The four basic dataset types are tables, networks, fields, and geometry; other possible collections of items are clusters, sets, and lists. These datasets are made up of five core data types: items, attributes, links, positions, and grids.

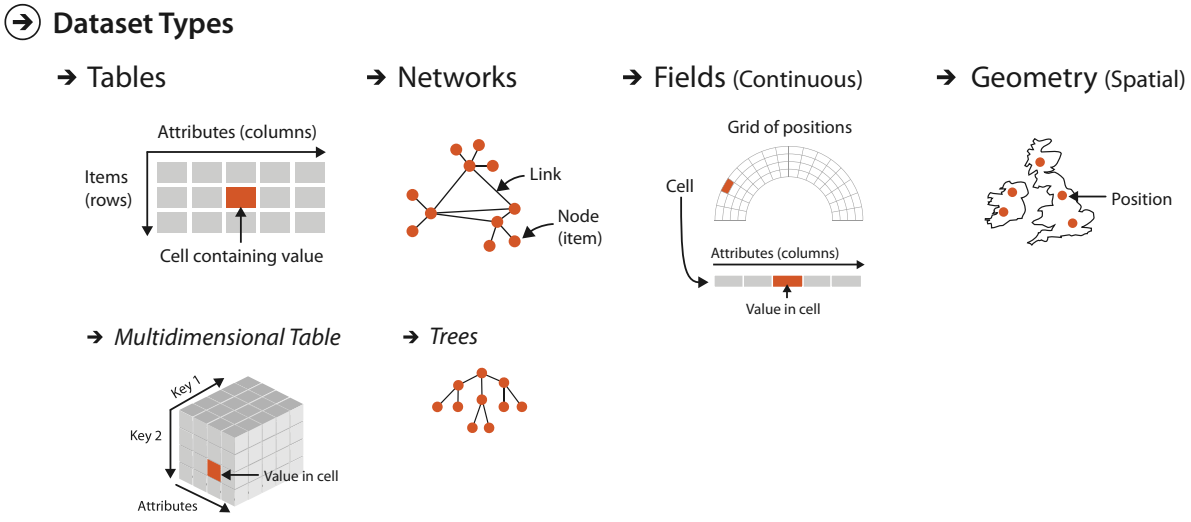


Figure 2.4. The detailed structure of the four basic dataset types.

2.4.1 Tables

Many datasets come in the form of **tables** that are made up of rows and columns, a familiar form to anybody who has used a spreadsheet. In this chapter, I focus on the concept of a table as simply a type of dataset that is independent of any particular visual representation; later chapters address the question of what visual representations are appropriate for the different types of datasets.

► Chapter 7 covers how to arrange tables spatially.

For a simple **flat table**, the terms used in this book are that each row represents an **item** of data, and each column is an **attribute** of the dataset. Each **cell** in the table is fully specified by the combination of a row and a column—an item and an attribute—and contains a **value** for that pair. Figure 2.5 shows an example of the first few dozen items in a table of orders, where the attributes are order ID, order date, order priority, product container, product base margin, and ship date.

A **multidimensional table** has a more complex structure for indexing into a cell, with multiple keys.

► Keys and values are discussed further in Section 2.6.1.

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box		7/17/07
32	7/16/07	2-High	Medium Box		7/18/07
32	7/16/07	2-High	Medium Box		7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	1/20/05	5-Low	Wrap Bag	0.56	1/20/05
69	5	4-Not Specified	Small Pack	0.44	6/6/05
69	5	4-Not Specified	Wrap Bag	0.6	6/6/05
70	12/18/06	5-Low	Small Box	0.59	12/23/06
70	12/18/06	5-Low	Wrap Bag	0.82	12/23/06
96	4/17/05	2-High	Small Box	0.55	4/19/05
97	1/29/06	3-Medium	Small Box	0.38	1/30/06
129	11/19/08	5-Low	Small Box	0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

Figure 2.5. In a simple table of orders, a row represents an *item*, a column represents an *attribute*, and their intersection is the *cell* containing the value for that pairwise combination.

★ A synonym for *networks* is **graphs**. The word *graph* is also deeply overloaded in vis. Sometimes it is used to mean *network* as we discuss here, for instance in the vis subfield called *graph drawing* or the mathematical subfield called *graph theory*. Sometimes it is used in the field of statistical graphics to mean **chart**, as in bar graphs and line graphs.

★ A synonym for *node* is **vertex**.

★ A synonym for *link* is **edge**.

2.4.2 Networks and Trees

The dataset type of **networks** is well suited for specifying that there is some kind of relationship between two or more items.★ An item in a network is often called a **node**.★ A **link** is a relation between two items.★ For example, in an articulated social network the nodes are people, and links mean friendship. In a gene interaction network, the nodes are genes, and links between them mean that these genes have been observed to interact with each other. In a computer network, the nodes are computers, and the links represent the ability to send messages directly between two computers using physical cables or a wireless connection.

Network nodes can have associated attributes, just like items in a table. In addition, the links themselves could also be considered to have attributes associated with them; these may be partly or wholly disjoint from the node attributes.

It is again important to distinguish between the abstract concept of a network and any particular visual layout of that network where the nodes and edges have particular spatial positions. This chapter concentrates on the former.

▶ The spatial arrangement of networks is covered in Chapter 9.

2.4.2.1 Trees

Networks with hierarchical structure are more specifically called **trees**. In contrast to a general network, trees do not have cycles: each child node has only one parent node pointing to it. One example of a tree is the organization chart of a company, showing who reports to whom; another example is a tree showing the evolutionary relationships between species in the biological tree of life, where the child nodes of humans and monkeys both share the same parent node of primates.

2.4.3 Fields

The **field** dataset type also contains attribute values associated with cells.[1] Each **cell** in a field contains measurements or calculations from a **continuous** domain: there are conceptually infinitely many values that you might measure, so you could always take a new measurement between any two existing ones. Continuous phenomena that might be measured in the physical world or simulated in software include temperature, pressure, speed, force, and density; mathematical functions can also be continuous.

For example, consider a field dataset representing a medical scan of a human body containing measurements indicating the density of tissue at many sample points, spread regularly throughout a volume of 3D space. A low-resolution scan would have 262,144 cells, providing information about a cubical volume of space with 64 bins in each direction. Each cell is associated with a specific region in 3D space. The density measurements could be taken closer together with a higher resolution grid of cells, or further apart for a coarser grid.

Continuous data requires careful treatment that takes into account the mathematical questions of **sampling**, how frequently to

[1] My use of the term *field* is related to but not identical to its use in the mathematics literature, where it denotes a mapping from a domain to a range. In this case, the domain is a Euclidean space of one, two, or three dimensions, and the adjective modifying *field* is a statement about the range: **scalars**, **vectors**, or **tensors**. Although the term *field* by itself is not commonly found in the literature, when I use it without an adjective I'm emphasizing the continuous nature of the domain, rather than specifics of the ranges of scalars, vectors, or tensors.

take the measurements, and **interpolation**, how to show values in between the sampled points in a way that does not mislead. Interpolating appropriately between the measurements allows you to **reconstruct** a new view of the data from an arbitrary viewpoint that's faithful to what you measured. These general mathematical problems are studied in areas such as signal processing and statistics. Visualizing fields requires grappling extensively with these concerns.

In contrast, the table and network datatypes discussed above are an example of **discrete** data where a finite number of individual items exist, and interpolation between them is not a meaningful concept. In the cases where a mathematical framework is necessary, areas such as graph theory and combinatorics provide relevant ideas.[2]

2.4.3.1 Spatial Fields

Continuous data is often found in the form of a **spatial field**, where the cell structure of the field is based on sampling at spatial positions. Most datasets that contain inherently spatial data occur in the context of tasks that require understanding aspects of its spatial structure, especially shape.

For example, with a spatial field dataset that is generated with a medical imaging instrument, the user's task could be to locate suspected tumors that can be recognized through distinctive shapes or densities. An obvious choice for visual encoding would be to show something that spatially looks like an X-ray image of the human body and to use color coding to highlight suspected tumors. Another example is measurements made in a real or simulated wind tunnel of the temperature and pressure of air flowing over airplane wings at many points in 3D space, where the task is to compare the flow patterns in different regions. One possible visual encoding would use the geometry of the wing as the spatial substrate, showing the temperature and pressure using size-coded arrows.

The likely tasks faced by users who have spatial field data constrains many of the choices about the use of space when designing visual encoding idioms. Many of the choices for **nonspatial data**, where no information about spatial position is provided with the dataset, are unsuitable in this case.*

★ A synonym for *nonspatial data* is **abstract data**.

[2] Technically, all data stored within a computer is discrete rather than continuous; however, the interesting question is whether the underlying semantics of the bits that are stored represents samples of a continuous phenomenon or intrinsically discrete data.

Thus, the question of whether a dataset has the type of a spatial field or a nonspatial table has extensive and far-reaching implications for idiom design. Historically, vis diverged into areas of specialization based on this very differentiation. The subfield of **scientific visualization**, or **scivis** for short, is concerned with situations where spatial position is *given* with the dataset. A central concern in scivis is handling continuous data appropriately within the mathematical framework of signal processing. The subfield of **information visualization**, or **infovis** for short, is concerned with situations where the use of space in a visual encoding is *chosen* by the designer. A central concern in infovis is determining whether the chosen idiom is suitable for the combination of data and task, leading to the use of methods from human–computer interaction and design.

2.4.3.2 Grid Types

When a field contains data created by sampling at completely regular intervals, as in the previous example, the cells form a **uniform grid**. There is no need to explicitly store the **grid geometry** in terms of its location in space, or the **grid topology** in terms of how each cell connects with its neighboring cells. More complicated examples require storing different amounts of geometric and topological information about the underlying grid. A **rectilinear grid** supports nonuniform sampling, allowing efficient storage of information that has high complexity in some areas and low complexity in others, at the cost of storing some information about the geometric location of each each row. A **structured grid** allows curvilinear shapes, where the geometric location of each cell needs to be specified. Finally, **unstructured grids** provide complete flexibility, but the topological information about how the cells connect to each other must be stored explicitly in addition to their spatial positions.

2.4.4 Geometry

The **geometry** dataset type specifies information about the shape of items with explicit spatial positions. The items could be points, or one-dimensional lines or curves, or 2D surfaces or regions, or 3D volumes.

Geometry datasets are intrinsically spatial, and like spatial fields they typically occur in the context of tasks that require shape understanding. Spatial data often includes hierarchical structure at multiple scales. Sometimes this structure is provided intrinsically

with the dataset, or a hierarchy may be derived from the original data.

Geometry datasets do not necessarily have attributes, in contrast to the other three basic dataset types. Many of the design issues in vis pertain to questions about how to encode attributes. Purely geometric data is interesting in a vis context only when it is derived or transformed in a way that requires consideration of design choices. One classic example is when contours are derived from a spatial field. Another is when shapes are generated at an appropriate level of detail for the task at hand from raw geographic data, such as the boundaries of a forest or a city or a country, or the curve of a road. The problem of how to create images from a geometric description of a scene falls into another domain: computer graphics. While vis draws on algorithms from computer graphics, it has different concerns from that domain. Simply showing a geometric dataset is not an interesting problem from the point of view of a vis designer.

► Section 3.4.2.3 covers deriving data.

► Section 8.4 covers generating contours from scalar fields.

Geometric data is sometimes shown alone, particularly when shape understanding is the primary task. In other cases, it is the backdrop against which additional information is overlaid.

2.4.5 Other Combinations

Beyond tables, there are many ways to group multiple *items* together, including sets, lists, and clusters. A **set** is simply an unordered group of items. A group of items with a specified ordering could be called a **list**.* A **cluster** is a grouping based on attribute similarity, where items within a cluster are more similar to each other than to ones in another cluster.

★ In computer science, **array** is often used as a synonym for *list*.

There are also more complex structures built on top of the basic network type. A **path** through a network is an ordered set of segments formed by links connecting nodes. A **compound network** is a network with an associated tree: all of the nodes in the network are the leaves of the tree, and interior nodes in the tree provide a hierarchical structure for the nodes that is different from network links between them.

Many other kinds of data either fit into one of the previous categories or do so after transformations to create derived attributes. Complex and hybrid combinations, where the complete dataset contains multiple basic types, are common in real-world applications.

The set of basic types presented above is a starting point for describing the *what* part of an analysis instance that pertains to

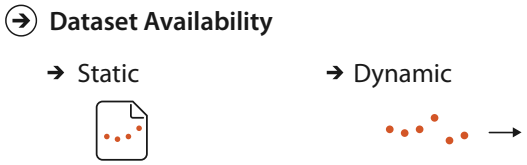


Figure 2.6. Dataset availability can be either static or dynamic, for any dataset type.

data; that is, the **data abstraction**. In simple cases, it may be possible to describe your data abstraction using only that set of terms. In complex cases, you may need additional description as well. If so, your goal should be to translate domain-specific terms into words that are as generic as possible.

2.4.6 Dataset Availability

Figure 2.6 shows the two kinds of dataset availability: *static* or *dynamic*.

The default approach to vis assumes that the entire dataset is available all at once, as a **static file**. However, some datasets are instead **dynamic streams**, where the dataset information trickles in over the course of the vis session.* One kind of dynamic change is to add new items or delete previous items. Another is to change the values of existing items.

★ A synonym for *dynamic* is **online**, and a synonym for *static* is **offline**.

This distinction in availability crosscuts the basic dataset types: any of them can be static or dynamic. Designing for streaming data adds complexity to many aspects of the vis process that are straightforward when there is complete dataset availability up front.

2.5 Attribute Types

Figure 2.7 shows the attribute types. The major disinction is between categorical versus ordered. Within the ordered type is a further differentiation between ordinal versus quantitative. Ordered data might range sequentially from a minimum to a maximum value, or it might diverge in both directions from a zero point in the middle of a range, or the values may wrap around in a cycle. Also, attributes may have hierarchical structure.

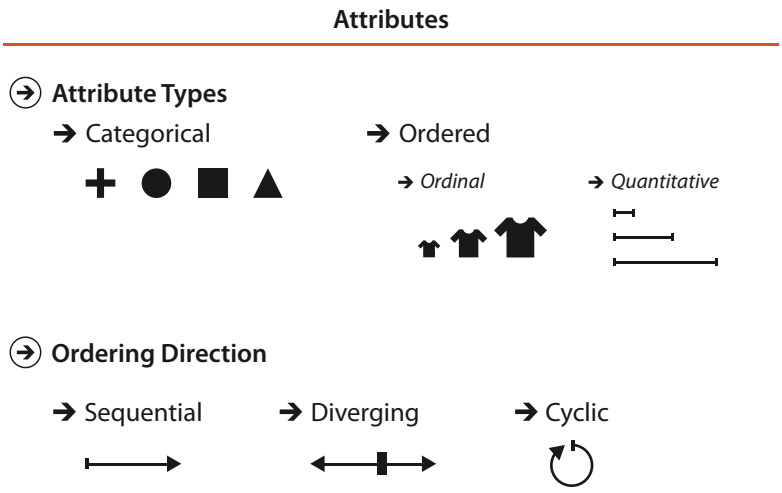


Figure 2.7. Attribute types are categorical, ordinal, or quantitative. The direction of attribute ordering can be sequential, diverging, or cyclic.

2.5.1 Categorical

★ A synonym for *categorical* is **nominal**.

The first distinction is between categorical and ordered data. The type of **categorical** data, such as favorite fruit or names, does not have an implicit ordering, but it often has hierarchical structure.* Categories can only distinguish whether two things are the same (apples) or different (apples versus oranges). Of course, any arbitrary external ordering can be imposed upon categorical data. Fruit could be ordered alphabetically according to its name, or by its price—but only if that auxiliary information were available. However, these orderings are not implicit in the attribute itself, the way they are with quantitative or ordered data. Other examples of categorical attributes are movie genres, file types, and city names.

2.5.2 Ordered: Ordinal and Quantitative

All **ordered** data does have an implicit ordering, as opposed to unordered *categorical* data. This type can be further subdivided. With **ordinal** data, such as shirt size, we cannot do full-fledged arithmetic, but there is a well-defined ordering. For example, large minus medium is not a meaningful concept, but we know that medium falls between small and large. Rankings are another kind

of ordinal data; some examples of ordered data are top-ten lists of movies or initial lineups for sports tournaments depending on past performance.

A subset of ordered data is **quantitative** data, namely, a measurement of magnitude that supports arithmetic comparison. For example, the quantity of 68 inches minus 42 inches is a meaningful concept, and the answer of 26 inches can be calculated. Other examples of quantitative data are height, weight, temperature, stock price, number of calling functions in a program, and number of drinks sold at a coffee shop in a day. Both integers and real numbers are quantitative data.[3]

In this book, the *ordered* type is used often; the ordinal type is only occasionally mentioned, when the distinction between it and the quantitative type matters.

2.5.2.1 Sequential versus Diverging

Ordered data can be either **sequential**, where there is a homogeneous range from a minimum to a maximum value, or **diverging**, which can be deconstructed into two sequences pointing in opposite directions that meet at a common zero point. For instance, a mountain *height* dataset is sequential, when measured from a minimum point of sea level to a maximum point of Mount Everest. A *bathymetric* dataset is also sequential, with sea level on one end and the lowest point on the ocean floor at the other. A full *elevation* dataset would be diverging, where the values go up for mountains on land and down for undersea valleys, with the zero value of sea level being the common point joining the two sequential datasets.

2.5.2.2 Cyclic

Ordered data may be **cyclic**, where the values wrap around back to a starting point rather than continuing to increase indefinitely. Many kinds of time measurements are cyclic, including the hour of the day, the day of the week, and the month of the year.

2.5.3 Hierarchical Attributes

There may be hierarchical structure within an attribute or between multiple attributes. The daily stock prices of companies collected

[3] In some domains the quantitative category is further split into **interval** versus **ratio** data [Stevens 46]; this distinction is typically not useful when designing a visual encoding, so in this book these types remain collapsed together into this single category.

over the course of a decade is an example of a time-series dataset, where one of the attributes is time. In this case, time can be aggregated hierarchically, from individual days up to weeks, up to months, up to years. There may be interesting patterns at multiple temporal scales, such as very strong weekly variations for weekday versus weekend, or more subtle yearly patterns showing seasonal variations in summer versus winter. Many kinds of attributes might have this sort of hierarchical structure: for example, the geographic attribute of a postal code can be aggregated up to the level of cities or states or entire countries.

► Section 13.4 covers hierarchical aggregation in more detail, and Section 7.5 covers the visual encoding of attribute hierarchies.

2.6 Semantics

Knowing the type of an attribute does not tell us about its semantics, because these two questions are crosscutting: one does not dictate the other. Different approaches to considering the semantics of attributes that have been proposed across the many fields where these semantics are studied. The classification in this book is heavily focused on the semantics of keys versus values, and the related questions of spatial and continuous data versus nonspatial and discrete data, to match up with the idiom design choice analysis framework. One additional consideration is whether an attribute is temporal.

2.6.1 Key versus Value Semantics

A **key** attribute acts as an index that is used to look up **value** attributes.* The distinction between key and value attributes is important for the dataset types of tables and fields, as shown in Figure 2.8.

★ A synonym for *key attribute* is **independent attribute**. A synonym for *value attribute* is **dependent attribute**. The language of independent and dependent is common in statistics. In the language of data warehouses, a synonym for *independent* is **dimension**, and a synonym for *dependent* is **measure**.

2.6.1.1 Flat Tables

A simple **flat table** has only one key, where each item corresponds to a row in the table, and any number of value attributes. In this case, the key might be completely implicit, where it's simply the index of the row. It might be explicit, where it is contained within the table as an attribute. In this case, there must not be any duplicate values within that attribute. In tables, keys may be categorical or ordinal attributes, but quantititive attributes are typically unsuitable as keys because there is nothing to prevent them from having the same values for multiple items.

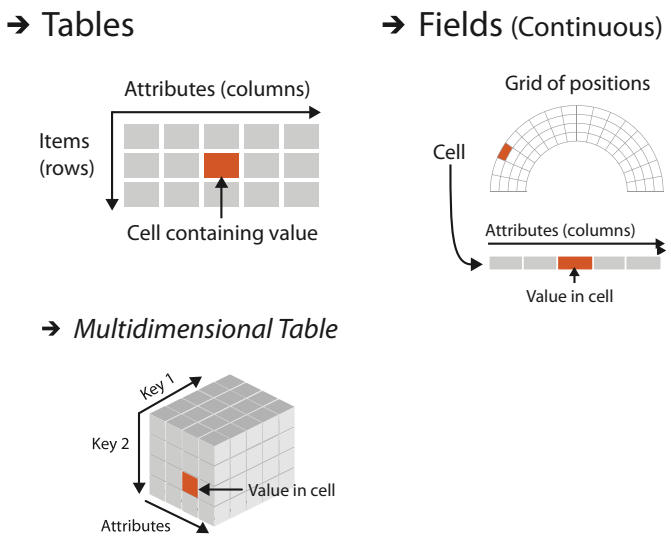


Figure 2.8. Key and value semantics for tables and fields.

For example, in Table 2.1, *Name* is a categorical attribute that might appear to be a reasonable key at first, but the last line shows that two people have the same name, so it is not a good choice. *Favorite Fruit* is clearly not a key, despite being categorical, because *Pear* appears in two different rows. The quantitative attribute of *Age* has many duplicate values, as does the ordinal attribute of *Shirt Size*. The first attribute in this flat table has an explicit unique identifier that acts as the key.[4] This key attribute could either be ordinal, for example if the order that the rows were entered into the table captures interesting temporal information, or categorical, if it's simply treated as a unique code.

Figure 2.9 shows the order table from Figure 2.5 where each attribute is colored according to its type. There is no explicit key: even the *Order ID* attribute has duplicates, because orders consist of multiple items with different container sizes, so it does not act as a unique identifier. This table is an example of using an implicit key that is the row number within the table.

[4] It's common to store the key attribute in the first column, for understandability by people and ease of building data structures by computers.

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	0.72	7/17/07
32	7/16/07	2-High	Medium Box	0.6	7/18/07
32	7/16/07	2-High	Medium Box	0.65	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	1/20/05	5-Low	Wrap Bag	0.56	1/20/05
69	6/4/05	4-Not Specified	Small Pack	0.44	6/6/05
69	6/4/05	4-Not Spec		0.6	6/6/05
70	12/18/06	5-Low		0.59	12/23/06
70	12/18/06	5-Low		0.82	12/23/06
96	4/17/05	2-High		0.55	4/19/05
97	1/29/06	3-Medium		0.38	1/30/06
129	11/19/08	5-Low		0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

Figure 2.9. The order table with the attribute columns colored by their type; none of them is a key.

2.6.1.2 Multidimensional Tables

The more complex case is a **multidimensional table**, where multiple keys are required to look up an item. The combination of all keys must be unique for each item, even though an individual key attribute may contain duplicates. For example, a common multidimensional table from the biology domain has a gene as one key and time as another key, so that the value in each cell is the activity level of a gene at a particular time.

The information about which attributes are keys and which are values may not be available; in many instances determining which attributes are independent keys versus dependent values is the goal of the vis process, rather than its starting point. In this case, the successful outcome of analysis using vis might be to recast a flat table into a more semantically meaningful multidimensional table.

2.6.1.3 Fields

Although fields differ from tables a fundamental way because they represent continuous rather than discrete data, keys and values are still central concerns. (Different vocabulary for the same basic idea is more common with spatial field data, where the term *independent variable* is used instead of *key*, and *dependent variable* instead of *value*.)

Fields are structured by sampling in a systematic way so that each grid cell is spanned by a unique range from a continuous domain. In spatial fields, spatial position acts as a quantitative key, in contrast to a nonspatial attribute in the case of a table that is categorical or ordinal. The crucial difference between fields and tables is that useful answers for attribute values are returned for locations throughout the sampled range, not just the exact points where data was recorded.

Fields are typically characterized in terms of the number of keys versus values. Their **multivariate** structure depends on the number of value attributes, and their **multidimensional** structure depends on the number of keys. The standard multidimensional cases are 2D and 3D fields for static measurements taken in two or three spatial dimensions,[5] and fields with three or four keys, in the case where these measurements are time-varying. A field can be both multidimensional and multivariate if it has multiple keys and multiple values. The standard classification according to multivariate structure is that a **scalar field** has one attribute per cell, a **vector field** has two or more attributes per cell, and a **tensor field** has many attributes per cell.★

★ These definitions of *scalar*, *vector*, and *tensor* follow the common usage in vis. In a strict mathematical sense, these distinctions are not technically correct, since scalars and vectors are included as a degenerate case of tensors. Mapping the mathematical usage to the vis usage, **scalars** mean mathematical tensors of order 0, **vectors** mean mathematical tensors of order 1, and **tensors** mean mathematical tensors of order 2 or more.

2.6.1.4 Scalar Fields

A **scalar field** is univariate, with a single value attribute at each point in space. One example of a 3D scalar field is the time-varying medical scan above; another is the temperature in a room at each point in 3D space. The geometric intuition is that each point in a scalar field has a single value. A point in space can have several different numbers associated with it; if there is no underlying connection between them then they are simply multiple separate scalar fields.

2.6.1.5 Vector Fields

A **vector field** is multivariate, with a list of multiple attribute values at each point. The geometric intuition is that each point in a vector

[5] It's also possible for a spatial field to have just one key.

field has a direction and magnitude, like an arrow that can point in any direction and that can be any length. The length might mean the speed of a motion or the strength of a force. A concrete example of a 3D vector field is the velocity of air in the room at a specific time point, where there is a direction and speed for each item. The dimensionality of the field determines the number of components in the direction vector; its length can be computed directly from these components, using the standard Euclidean distance formula. The standard cases are two, three, or four components, as above.

2.6.1.6 Tensor Fields

A **tensor field** has an array of attributes at each point, representing a more complex multivariate mathematical structure than the list of numbers in a vector. A physical example is stress, which in the case of a 3D field can be defined by nine numbers that represent forces acting in three orthogonal directions. The geometric intution is that the full information at each point in a tensor field cannot be represented by just an arrow and would require a more complex shape such as an ellipsoid.

2.6.1.7 Field Semantics

This categorization of spatial fields requires knowledge of the attribute semantics and cannot be determined from type information alone. If you are given a field with multiple measured values at each point and no further information, there is no sure way to know its structure. For example, nine values could represent many things: nine separate scalar fields, or a mix of multiple vector fields and scalar fields, or a single tensor field.

2.6.2 Temporal Semantics

A **temporal** attribute is simply any kind of information that relates to time. Data about time is complicated to handle because of the rich hierarchical structure that we use to reason about time, and the potential for periodic structure. The time hierarchy is deeply multiscale: the scale of interest could range anywhere from nanoseconds to hours to decades to millennia. Even the common words *time* and *date* are a way to partially specify the scale of temporal interest. Temporal analysis tasks often involve finding or verifying periodicity either at a predetermined scale or at some scale not known in advance. Moreover, the temporal scales of interest do not all fit into a strict hierarchy; for instance, weeks do not fit

cleanly into months. Thus, the generic vis problems of transformation and aggregation are often particularly complex when dealing with temporal data. One important idea is that even though the dataset semantics involves change over time, there are many approaches to visually encoding that data—and only one of them is to show it changing over time in the form of an animation.

Temporal attributes can have either value or key semantics. Examples of temporal attributes with dependent value semantics are a duration of elapsed time or the date on which a transaction occurred. In both spatial fields and abstract tables, time can be an independent key. For example, a time-varying medical scan can have the independent keys of x, y, z, t to cover spatial position and time, with the dependent value attribute of density for each combination of four indices to look up position and time. A temporal key attribute is usually considered to have a quantitative type, although it's possible to consider it as ordinal data if the duration between events is not interesting.

► Section 3.4.2.3 introduces the problem of data transformation. Section 13.4 discusses the question of aggregation in detail.

► Vision versus memory is discussed further in Section 6.5.

2.6.2.1 Time-Varying Data

A dataset has **time-varying** semantics when time is one of the key attributes, as opposed to when the temporal attribute is a value rather than a key. As with other decisions about semantics, the question of whether time has key or value semantics requires external knowledge about the nature of the dataset and cannot be made purely from type information. An example of a dataset with time-varying semantics is one created with a sensor network that tracks the location of each animal within a herd by taking new measurements every second. Each animal will have new location data at every time point, so the temporal attribute is an independent key and is likely to be a central aspect of understanding the dataset. In contrast, a horse-racing dataset covering a year's worth of races could have temporal value attributes such as the race start time and the duration of each horse's run. These attributes do indeed deal with temporal information, but the dataset is not time-varying.

A common case of temporal data occurs in a **time-series** dataset, namely, an ordered sequence of time–value pairs. These datasets are a special case of tables, where time is the key. These time-value pairs are often but not always spaced at uniform temporal intervals. Typical time-series analysis tasks involve finding trends, correlations, and variations at multiple time scales such as hourly, daily, weekly, and seasonal.

The word **dynamic** is often used ambiguously to mean one of two very different things. Some use it to mean a dataset has *time-varying* semantics, in contrast to a dataset where time is not a key attribute, as discussed here. Others use it to mean a dataset has *stream* type, in contrast to an unchanging file that can be loaded all at once. In this latter sense, items and attributes can be added or deleted and their values may change during a running session of a vis tool. I carefully distinguish between these two meanings here.

▶ The dataset types of dynamic streams versus static files are discussed in Section 2.4.6.

2.7 Further Reading

The Big Picture The framework presented here was inspired in part by the many taxonomies of data that have been previously proposed, including the synthesis chapter at the beginning of an early collection of infovis readings [Card et al. 99], a taxonomy that emphasizes the division between continuous and discrete data [Tory and Möller 04a], and one that emphasizes both data and tasks [Shneiderman 96].

Field Datasets Several books discuss the spatial field dataset type in far more detail, including two textbooks [Telea 07, Ward et al. 10], a voluminous handbook [Hansen and Johnson 05], and the *vtk* book [Schroeder et al. 06].

Attribute Types The attribute types of categorical, ordered, and quantitative were proposed in the seminal work on scales of measurement from the psychophysics literature [Stevens 46]. Scales of measurement are also discussed extensively in the book *The Grammar of Graphics* [Wilkinson 05] and are used as the foundational axes of an influential vis design space taxonomy [Card and Mackinlay 97].

Key and Value Semantics The Polaris vis system, which has been commercialized as Tableau, is built around the distinction between key attributes (independent dimensions) and value attributes (dependent measures) [Stolte et al. 02].

Temporal Semantics A good resource for time-oriented data vis is a recent book, *Visualization of Time-Oriented Data* [Aigner et al. 11].

This page intentionally left blank

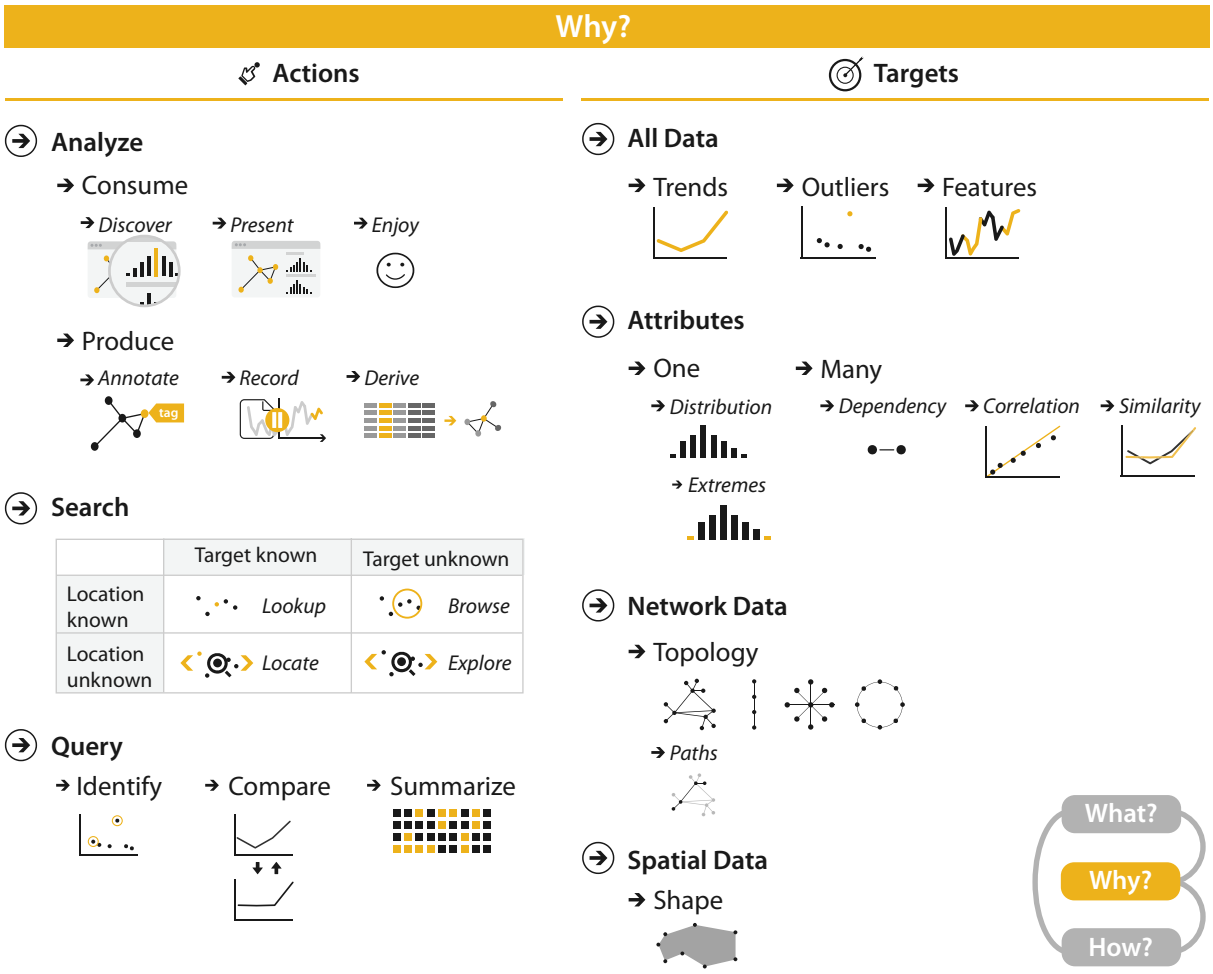


Figure 3.1. *Why* people are using vis in terms of actions and targets.

Chapter 3

Why: Task Abstraction

3.1 The Big Picture

Figure 3.1 breaks down into actions and targets the reasons *why* a vis tool is being used. The highest-level actions are to use vis to consume or produce information. The cases for consuming are to present, to discover, and to enjoy; discovery may involve generating or verifying a hypothesis. At the middle level, search can be classified according to whether the identity and location of targets are known or not: both are known with lookup, the target is known but its location is not for locate, the location is known but the target is not for browse, and neither the target nor the location are known for explore. At the low level, queries can have three scopes: identify one target, compare some targets, and summarize all targets. Targets for all kinds of data are finding trends and outliers. For one attribute, the target can be one value, the extremes of minimum and maximum values, or the distribution of all values across the entire attribute. For multiple attributes, the target can be dependencies, correlations, or similarities between them. The target with network data can be topology in general or paths in particular, and with spatial data the target can be shape.

3.2 Why Analyze Tasks Abstractly?

This framework encourages you to consider tasks in abstract form, rather than the domain-specific way that users typically think about them.

Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them. Otherwise, it's hard to make useful comparisons between domain situations, because if you don't do this kind of translation then everything just appears to be different. That apparent difference is misleading: there are a lot of similar-

ities in what people want to do once you strip away the surface language differences.

For example, an epidimiologist studying the spread of a new strain of influenza might initially describe her task as "contrast the prognosis of patients who were intubated in the ICU more than one month after exposure to patients hospitalized within the first week", while a biologist studying immune system response might use language such as "see if the results for the tissue samples treated with LL-37 match up with the ones without the peptide". Even if you know what all the specialized vocabulary means, it's still hard to think about what these two descriptions have in common because they're using different words: "contrast" versus "match up". If you transform these into descriptions using a consistent set of generic terms, then you can spot that these two tasks are just two instances of the same thing: "compare values between two groups".

The analysis framework has a small set of carefully chosen words to describe *why* people are using vis, designed to help you crisply and concisely distinguish between different goals. This set has verbs describing *actions*, and nouns describing *targets*. It's possible that you might decide to use additional terms to completely and precisely describe the user's goals; if so, strive to translate domain-specific terms into words that are as generic as possible.

The same vis tool might be usable for many different goals. It is often useful to consider only one of the user's goals at a time, in order to more easily consider the question of *how* a particular idiom supports that goal. To describe complex activities, you can specify a chained sequence of tasks, where the output of one becomes the input to the next.

Another important reason to analyze the task is to understand whether and how to transform the user's original data into different forms by deriving new data. That is, the task abstraction can and should guide the data abstraction.

3.3 Who: Designer or User

It's sometimes useful to augment an analysis instance specification by indicating *who* has a goal or makes a design choice: the designer of the vis or the end user of the vis. Both cases are common.

Vis tools fall somewhere along a continuum from specific to general. On the specific side, tools are narrow: the designer has built many choices into the design of the tool itself in a way that the user cannot override. These tools are limited in the kinds of data and tasks that they can address, but their strength is that users are not faced with an overwhelming array of design choices. On the general side, tools are flexible and users have many choices to make. The breadth of choices is both a strength and a limitation: users have a lot of power, but they also may make ineffective choices if they do not have a deep understanding of many vis design issues.

Specialized vis tools are designed for specific contexts with a narrow range of data configurations, especially those created through a problem-driven process. These specialized datasets are often an interesting mix of complex combinations of and special cases of the basic data types. They also are a mix of original and derived data. In contrast, general vis tools are designed to handle a wide range of data in a flexible way, for example, by accepting any dataset of a particular type as input: tables, or fields, or networks. Some particularly broad tools handle multiple dataset types, for instance, supporting transformations between tables and networks.

► Dataset types are covered in Section 2.4.

3.4 Actions

Figure 3.2 shows three levels of **actions** that define user goals. The high-level choices describe how the vis is being used to *analyze*, either to consume existing data or to also produce additional data. The mid-level choices cover what kind of *search* is involved, in terms of whether the target and location are known or not. The low-level choices pertain to the kind of *query*: does the user need to identify one target, compare some targets, or summarize all of the targets? The choices at each of these three levels are independent from each other, and it's usually useful to describe actions at all three of them.

3.4.1 Analyze

At the highest level, the framework distinguishes between two possible goals of people who want to **analyze** data using a vis tool: users might want only to *consume* existing information or also to actively *produce* new information.

The most common use case for vis is for the user to **consume** information that has already been generated as data stored in a

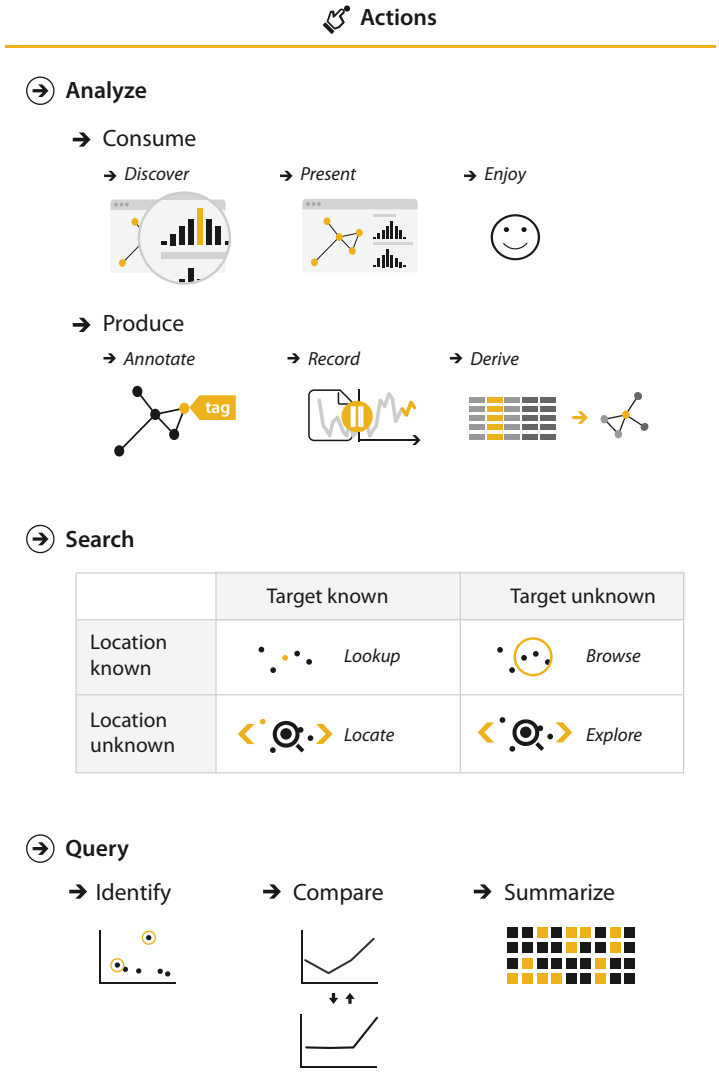


Figure 3.2. Three levels of actions: analyze, search, and query.

format amenable to computation. The framework has three further distinctions within that case: whether the goal is to present something that the user already understands to a third party, or for the user to discover something new or analyze information that

is not already completely understood, or for users to enjoy a vis to indulge their casual interests in a topic.

3.4.1.1 Discover

The **discover** goal refers to using vis to find new knowledge that was not previously known. Discovery may arise from the serendipitous observation of unexpected phenomena, but investigation may be motivated by existing theories, models, hypotheses, or hunches. This usage includes the goal of finding completely new things; that is, the outcome is to **generate** a new hypothesis. It also includes the goal of figuring out whether a conjecture is true or false; that is, to **verify**—or disconfirm—an existing hypothesis.

While vis for discovery is often associated with modes of scientific inquiry, it is not restricted to domain situations that are formally considered branches of science. The discover goal is often discussed as the classic motivation for sophisticated interactive idioms, because the vis designer doesn't know in advance what the user will need to see.* The fundamental motivation of this analysis framework is to help you separate out the questions of *why* the vis is being used from *how* the vis idiom is designed to achieve those goals, so I will repeatedly emphasize that *why* doesn't dictate *how*.

★ This distinction between the goals of presentation of the known and discovery of the unknown is very common in the vis literature, but other sources may use different terms, such as **explain** versus **explore**.

3.4.1.2 Present

The **present** goal refers to the use of vis for the succinct communication of information, for telling a story with data, or guiding an audience through a series of cognitive operations. Presentation using vis may take place within the context of decision making, planning, forecasting, and instructional processes. The crucial point about the *present* goal is that vis is being used by somebody to communicate something specific and already understood to an audience.

Presentation may involve collaborative or pedagogical contexts, and the means by which a presentation is given may vary according to the size of the audience, whether the presentation is live or prerecorded, and whether the audience is in the same place as the presenter. One classic example of a *present* vis is static information graphics, such as a diagram in a newspaper or an image in a blog. However, the *present* goal is not intrinsically limited to a static visual encoding idiom; it's very possible to pursue this goal with dynamic vis idioms that include interaction and animation. Once again, the decision about *why* is separable from *how* the id-

iom is designed: presentation can be supported through a wide variety of idiom design choices.

A crucial aspect of presentation is that the knowledge communicated is already known to the presenter in advance. Sometimes the presenter knows it before using vis at all and uses the vis only for communication. In other cases, the knowledge arose from the presenter's previous use of vis with the goal of discovery, and it's useful to think about a chained sequence of tasks where the output of a discover session becomes the input to a present session.

3.4.1.3 Enjoy

The **enjoy** goal refers to casual encounters with vis. In these contexts, the user is not driven by a previously pressing need to verify or generate a hypothesis but by curiosity that might be both stimulated and satisfied by the vis. Casual encounters with vis for enjoyment can be fleeting, such as when looking at an infographic while reading a blog post. However, users can become sufficiently engaged with an enjoyable vis tool that they use it intensively for a more extended period of time.

One aspect of this classification that's tricky is that the goals of the eventual vis user might not be a match with the user goals conjectured by the vis designer. For example, a vis tool may have been intended by the designer for the goal of discovery with a particular audience, but it might be used for pure enjoyment by a different group of people. In the analyses presented in this book I'll assume that these goals are aligned, but in your own experience as a designer you might need to consider how they might diverge.

Figure 3.3 shows the Name Voyager, which was created for expectant parents deciding what to name their new baby. When the user types characters of a name, the vis shows data for the popularity of names in the United States since 1900 that start with that sequence of characters. The tool uses the visual encoding idiom where each name has a stripe whose height corresponds to popularity at a given time. Currently popular names are brighter, and gender is encoded by color. The Name Voyager appealed to many people with no interest in having children, who analyzed many different historical trends and posted extensively about their findings in their personal blogs, motivated by their own enjoyment rather than a pressing need [Wattenberg 05].

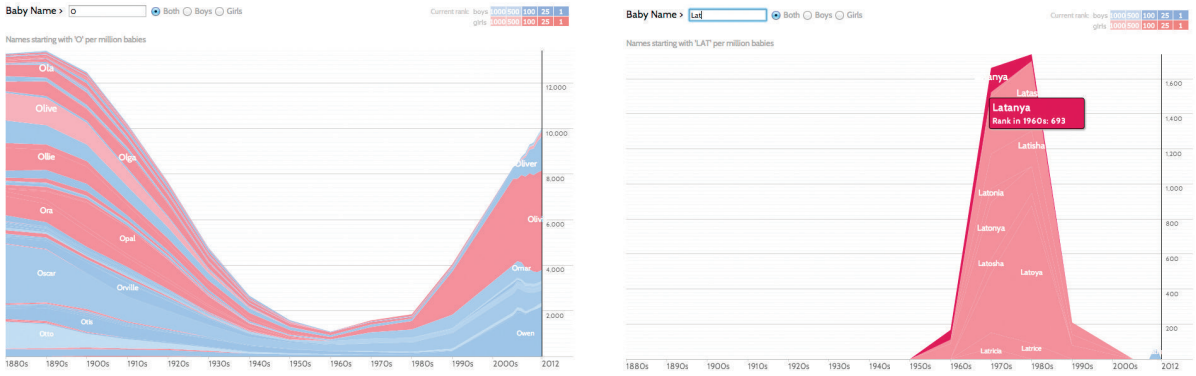


Figure 3.3. Name Voyager, a vis tool originally intended for parents focused deciding on what to name their expected baby, ended up being used by many nonparents to analyze historical trends for their own enjoyment. Left: Names starting with 'O' had a notable dip in popularity in the middle of the century. Right: Names starting with 'LAT' show a trend of the 1970s. After [Wattenberg 05, Figures 2 and 3], using http://www.babynamewizard.com.

3.4.2 Produce

In contrast to using vis only for consuming existing information, in the **produce** case the intent of the user is to generate new material. Often the goal with *produce* is to produce output that is used immediately, as input to a next instance. Sometimes the user intends to use this new material for some other vis-related task later on, such as discovery or presentation. Sometimes the intended use of the new material is for some other purpose that does not require vis, such as downstream analysis using nonvisual tools. There are three kinds of produce goals: *annotate*, *record*, and *derive*.

3.4.2.1 Annotate

The **annotate** goal refers to the addition of graphical or textual annotations associated with one or more preexisting visualization elements, typically as a manual action by the user. When an annotation is associated with data items, the annotation could be thought of as a new attribute for them. For example, the user could annotate all of the points within a cluster with a text label.

► Attributes are covered in Chapter 2.

3.4.2.2 Record

The **record** goal saves or captures visualization elements as persistent artifacts. These artifacts include screen shots, lists of book-

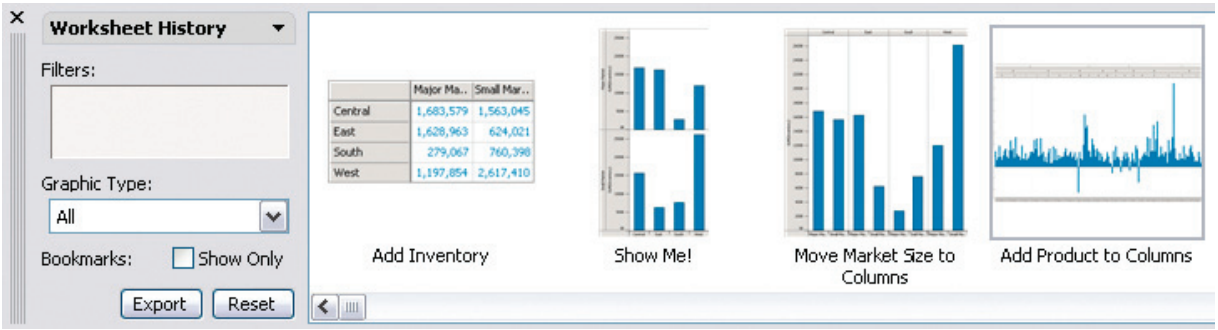


Figure 3.4. Graphical history recorded during an analysis session with Tableau. From [Heer et al. 08, Figure 1].

marked elements or locations, parameter settings, interaction logs, or annotations. The *record* choice saves a persistent artifact, in contrast to the *annotate*, which attaches information temporarily to existing elements; an annotation made by a user can subsequently be recorded. One interesting example of a *record* goal is to assemble a *graphical history*, in which the output of each task includes a static snapshot of the view showing its current state, and these snapshots accumulate in a branching meta-visualization showing what occurred during the user's entire session of using the vis tool. Figure 3.4 shows an example from the Tableau vis tool [Heer et al. 08]. Recording and retaining artifacts such as these are often desirable for maintaining a sense of analytical provenance, allowing users to revisit earlier states or parameter settings.

3.4.2.3 Derive

The **derive** goal is to produce new data elements based on existing data elements. New attributes can be derived from information contained within existing ones, or data can be transformed from one type into another. Deriving new data is a critical part of the vis design process. The common case is that deriving new data is a choice made by vis designers, but this choice could also be driven by a user of a vis tool.

When you are faced with a dataset, you should always consider whether to simply use it as is, or to transform it to another form: you could create newly derived attributes from the original ones, or even transform the dataset from the original type to another one.

There is a strong relationship between the form of the data—the attribute and dataset types—and what kinds of vis idioms are

effective at displaying it. The good news is that your hands are not tied as a designer because you can transform the data into a form more useful for the task at hand. Don't just draw what you're given; decide what the right thing to show is, create it with a series of transformations from the original dataset, and draw that!

The ability to derive new data is why the data abstraction used in a vis tool is an active choice on the part of the designer, rather than simply being dictated by what the user provides. Changing the dataset to another form by deriving new attributes and types greatly expands the design space of possible vis idioms that you can use to display it. The final data abstraction that you choose might simply be the dataset in its original form, but more complex data abstractions based on deriving new attributes and types are frequently necessary if you're designing a vis tool for a complex, real-world use case. Similarly, when you consider the design of an existing vis system, understanding how the original designer chose to transform the given dataset should be a cornerstone of your analysis.

A dataset often needs to be transformed beyond its original state in order to create a visual encoding that can solve the desired problem. To do so, we can create **derived attributes** that extend the dataset beyond the original set of attributes that it contains.[⋆]

★ A synonym for *derive* is **transform**.

In some cases, the derived attribute encodes the same data as the original, but with a change of type. For example, a dataset might have an original attribute that is quantitative data: for instance, floating point numbers that represent temperature. For some tasks, like finding anomalies in local weather patterns, that raw data might be used directly. For another task, like deciding whether water is an appropriate temperature for a shower, that quantitative attribute might be transformed into a new derived attribute that is ordered: hot, warm, or cold. In this transformation, most of the detail is aggregated away. In a third example, when making toast, an even more lossy transformation into a binary categorical attribute might suffice: burned or not burned.

In other cases, creating the derived attribute requires access to additional information. For a geographic example, a categorical attribute of city name could be transformed into two derived quantitative attributes containing the latitude and longitude of the city. This transformation could be accomplished through a lookup to a separate, external database.

A new derived attribute may be created using arithmetic, logical, or statistical operations ranging from simple to complex. A common simple operation is subtracting two quantitative attributes

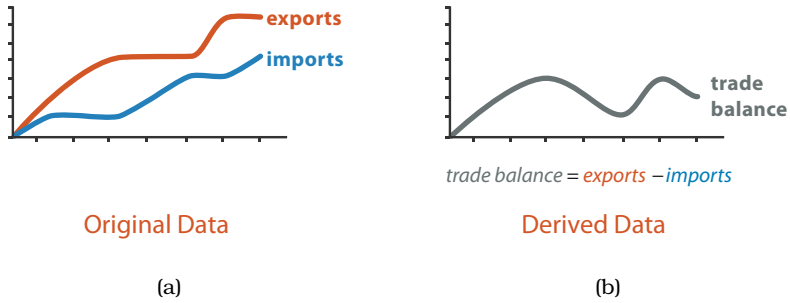


Figure 3.5. Derived attributes can be directly visually encoded. (a) Two original data attributes are plotted, imports and exports. (b) The quantitative derived attribute of trade balance, the difference between the two originals, can be plotted directly.

to create a new quantitative difference attribute, which can then be directly visually encoded. Figure 3.5 shows an example of encoding two attributes directly, versus encoding the derived variable of the difference between them. For tasks that require understanding this difference, Figure 3.5(b) is preferable because it encodes the difference directly. The user can interpret the information by judging position along a common frame. In contrast, in Figure 3.5(a) the user must judge the difference in heights between the two original curves at each step, a perceptual operation that is more difficult and demanding. This operation is simple because it is localized to a pair of attribute values; a more complex operation would require global computations across all values for an attribute, such as averaging for a single attribute or the correlation between two of them.

Datasets can be transformed into new ones of a different type, just as new attributes can be derived from existing ones. The full process of creating derived data may involve multiple stages of transformation.

For example, the VxInsight system transforms a table of genomics data into a network through a multistage derivation process by first creating a quantitative derived attribute of similarity, and then creating a derived network with links only between the most similar items [Davidson et al. 01]. The table had 6000 rows of yeast genes, and 18 columns containing measurements of the gene activity level in a specific experimental condition. The values in the columns were used to derive a new attribute, the similar-

ity score, defined between each pair of genes. The similarity score was computed using sophisticated statistical processing to be robust in the presence of nonlinear and noisy data, as occurs in this sort of biological application. This derived attribute was then used to create a derived network, where the nodes in the network were genes. A link was established between two genes when the similarity score was high; specifically, links were created only for the top 20 similarity scores.

3.4.3 Search

All of the high-level *analyze* cases require the user to **search** for elements of interest within the vis as a mid-level goal.[⋆] The classification of search into four alternatives is broken down according to whether the identity and location of the search target is already known or not.

★ The verb **find** is often used as a synonym in descriptions of *search* tasks, implying a successful outcome.

3.4.3.1 Lookup

If users already know both what they're looking for and where it is, then the search type is simply **lookup**. For example, a user of a tree vis showing the ancestral relationships between mammal species might want to look up humans, and can get to the right spot quickly by remembering how humans are classified: they're in the group that has live young rather than laying eggs like a platypus or having a pouch like kangaroos, and within that group humans fall into the category of primates.

3.4.3.2 Locate

To find a known target at an unknown location, the search type is **locate**: that is, find out where the specific object is. In a similar example, the same user might not know where to find rabbits, and would have to look around in a number of places before locating them as lagomorphs (not rodents)!

3.4.3.3 Browse

In contrast, the exact identity of a search target might not be known in advance; rather, it might be specified based on characteristics. In this case, users are searching for one or more items that fit some kind of specification, such as matching up with a particular range of attribute values. When users don't know exactly what they're looking for, but they do have a location in mind

of where to look for it, the search type is **browse**. For instance, if a user of a tree vis is searching within a particular subtree for leaf nodes having few siblings, it would be an instance of *browse* because the location is known in advance, even though the exact identity of the search target isn't. Another example of browsing is a user of a vis tool with the visual encoding idiom of a line graph displaying the share price of multiple companies over the past month, who examines the share price of each line on June 15.

3.4.3.4 Explore

When users are not even sure of the location, the search type is **explore**. It entails searching for characteristics without regard to their location, often beginning from an overview of everything. Examples include searching for outliers in a scatterplot, for anomalous spikes or periodic patterns in a line graph of time-series data, or for unanticipated spatially dependent patterns in a choropleth map.

3.4.4 Query

Once a target or set of targets for a search has been found, a low-level user goal is to **query** these targets at one of three scopes: *identify*, *compare*, or *summarize*. The progression of these three corresponds to an increase in the amount of search targets under consideration: one, some, or all. That is, **identify** refers to a single target, **compare** refers to multiple targets, and **summarize** refers to the full set of possible targets.

For a concrete example, consider different uses of a choropleth map of US election results, where each state is color-coded by the party that won. A user can *identify* the election results for one state, *compare* the election results of one state to another, or *summarize* the election results across all states to determine how many favored one candidate or the other or to determine the overall distribution of margin of victory values.

3.4.4.1 Identify

The scope of **identify** is a single target. If a search returns known targets, either by *lookup* or *locate*, then *identify* returns their characteristics. For example, a user of a static map that represents US election results by color coding each state red or blue, with the saturation level of either hue showing the proportion, can *identify*

the winning party and margin of victory for the state of California. Conversely, if a search returns targets matching particular characteristics, either by *browse* or *explore*, then *identify* returns specific references. For instance, the election map user can *identify* the state having the highest margin of victory.

3.4.4.2 Compare

The scope of **compare** is multiple targets. Comparison tasks are typically more difficult than identify tasks and require more sophisticated idioms to support the user. For example, the capability of inspecting a single target in detail is often necessary, but not sufficient, for comparison.

3.4.4.3 Summarize

The scope of **summarize** task is all possible targets. A synonym for *summarize* is **overview**, a term is frequently used in the vis literature both as a verb, where it means to provide a comprehensive view of everything, and as a noun, where it means a summary display of everything. The goal of providing an overview is extremely common in visualization.

▶ Section 6.7 discusses the question of how and when to provide overviews.

3.5 Targets

Figure 3.6 shows four kinds of abstract targets. The actions discussed above refer to a **target**, meaning some aspect of the data that is of interest to the user. Targets are nouns, whereas actions are verbs. The idea of a target is explicit with search and query actions. It is more implicit with the use actions, but still relevant: for example, the thing that the user presents or discovers.

Three high-level targets are very broadly relevant, for all kinds of data: *trends*, *outliers*, and *features*. A **trend** is a high-level characterization of a pattern in the data.⋆ Simple examples of trends include increases, decreases, peaks, troughs, and plateaus. Almost inevitably, some data doesn't fit well with that backdrop; those elements are the **outliers**.⋆ The exact definition of **features** is task dependent, meaning any particular structures of interest.

Attributes are specific properties that are visually encoded. The lowest-level target for an attribute is to find an individual value. Another frequent target of interest is to find the extremes: the minimum or maximum value across the range. A very common

★ Indeed, a synonym for *trend* is simply **pattern**.

★ There are many other synonyms for *outliers*, including **anomalies**, **novelties**, **deviants**, and **surprises**.

▶ Attributes are discussed in detail in Chapter 2.

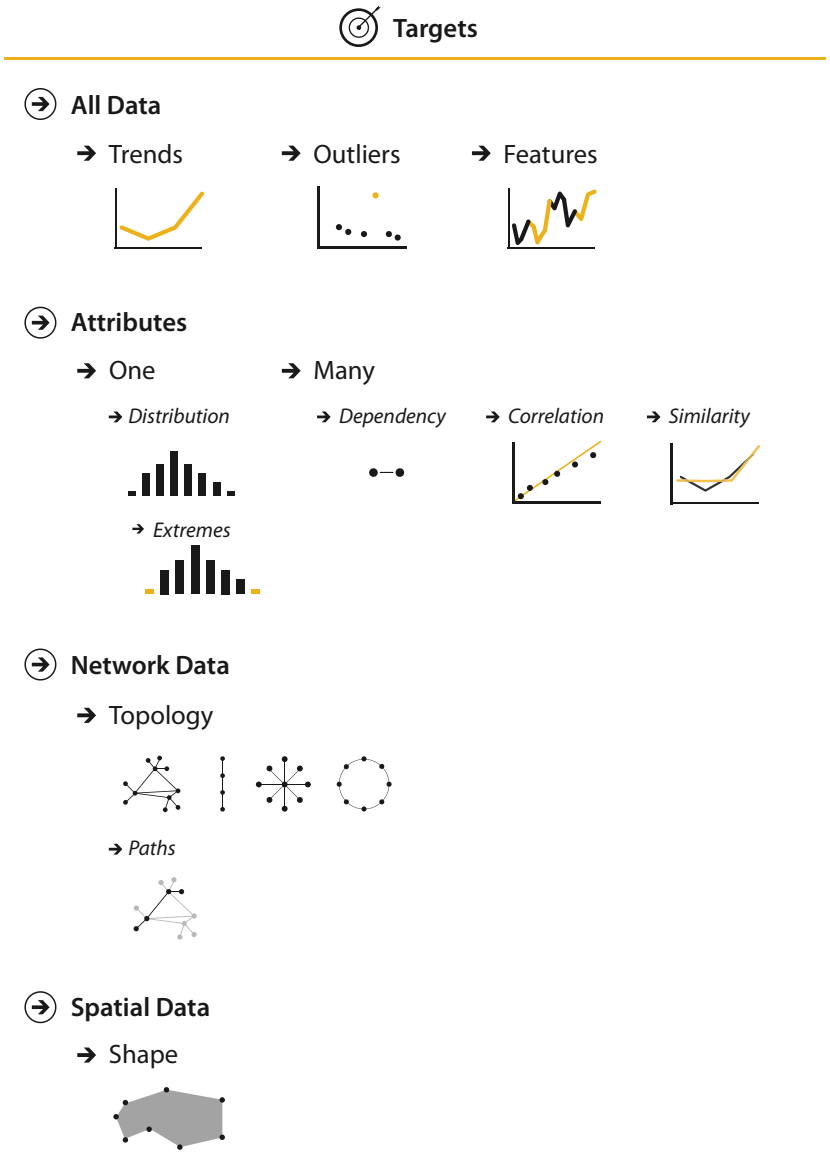


Figure 3.6. The goals of the user might be to find or understand specific aspects of the data: trends and outliers for all kinds of data; individual values, the minimum or maximum extremes of the range, or the entire distribution of a single attribute; or the dependencies, correlations, or similarities between multiple attributes; topology or paths for network data, and shape for spatial data.

target that has high-level scope is the distribution of all values for an attribute.

Some targets encompass the scope of multiple attributes: *dependencies*, *correlations*, and *similarities* between attributes. A first attribute can have a **dependency** on a second, where the values for the first directly depend on those of the second. There is a **correlation** between one attribute and another if there is a tendency for the values of second to be tied to those of the first. The **similarity** between two attributes can be defined as a quantitative measurement calculated on all of their values, allowing attributes to be ranked with respect to how similar, or different, they are from each other.

The abstract tasks of understanding trends, outliers, distributions, and correlations are extremely common reasons to use vis. Each of them can be expressed in very diverse terms using domain-specific language, but you should be on the lookout to recognize these abstractions.

Some targets pertain to specific types of datasets. Network data specifies relationships between nodes as links. The fundamental target with network data is to understand the structure of these interconnections; that is, the network's **topology**. A more specific topological target is a **path** of one or more links that connects two nodes. For spatial data, understanding and comparing the geometric **shape** is the common target of user actions.

► The network datatype is covered in Section 2.4.2, and choices for how arrange networks are covered in Chapter 9.

► Section 2.4.3.1 covers the dataset type of spatial fields, and Section 2.4.4 covers geometry. Choices for arranging spatial data are covered in Chapter 8.

3.6 How: A Preview

The third part of an analysis instance trio is *how* a vis idiom can be constructed out of a set of design choices. Figure 3.7 provides a preview of these choices, with a high-level breakdown into four major classes.

The family of how to encode data within a view has five choices for how to arrange data spatially: express values; separate, order, and align regions; and use given spatial data. This family also includes how to map data with all of the nonspatial visual channels including color, size, angle, shape, and many more. The manipulate family has the choices of change any aspect of the view, select elements from within the view, and navigate to change the viewpoint within the view—an aspect of change with a rich enough set of choices to merit its own category. The family of how to facet data between views has choices for how to juxtapose and coordinate multiple views, how to partition data between views, and how to superimpose layers on top of each other. The family of how to

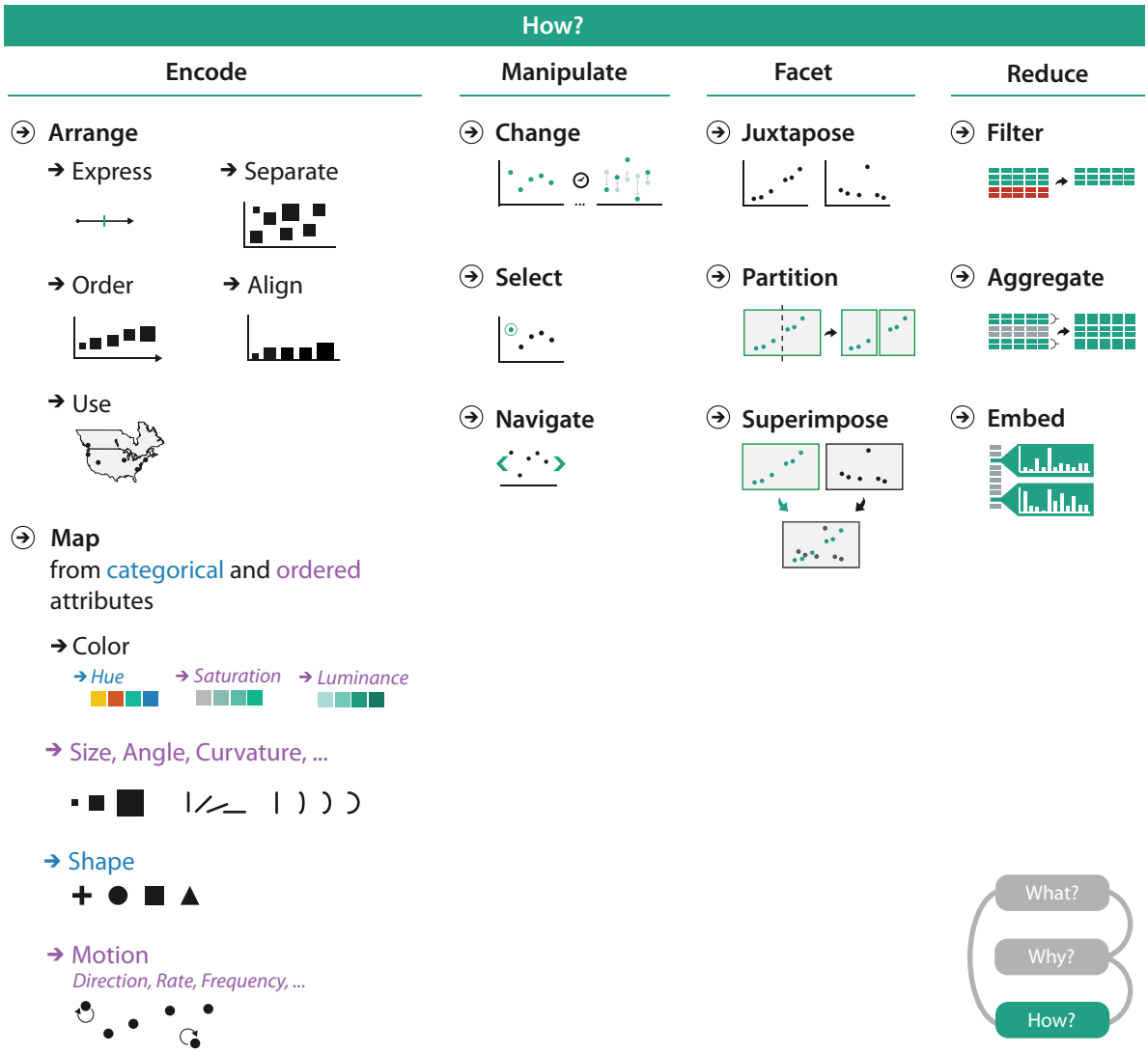


Figure 3.7. *How* to design vis idioms: encode, manipulate, facet, and reduce.

reduce the data shown has the options of filter data away, aggregate many data elements together, and embed focus and context information together within a single view.

The rest of this book defines, describes, and discusses these choices in depth.

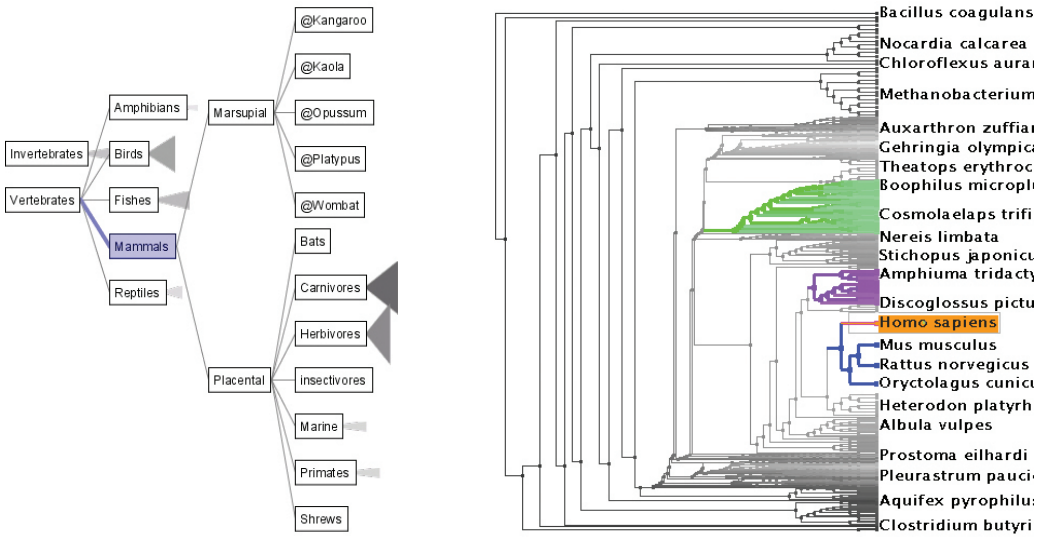
3.7 Analyzing and Deriving: Examples

The three analysis and derivation examples below give a taste of how this what–why–how framework can be used right away. The first example covers comparative analysis between two vis tools. The second example discusses deriving a single attribute, an importance measure for trees to decide which branches to show to summarize its topological structure. The third example covers deriving many new attributes and augmenting a spatial fluid dynamics dataset by creating derived spaces in which features of interest are easy to find.

3.7.1 Comparing Two Idioms

The what–why–how analysis framework is useful for comparative analysis, for example, to examine two different vis tools that have different answers for the question of *how* the idiom is designed when used for exactly the same context of *why* and *what* at the abstraction level.

SpaceTree [Plaisant et al. 02], shown in Figure 3.8(a), and TreeJuxtaposer [Munzner et al. 03], shown in Figure 3.8(b), are tree vis



(a) (b)

Figure 3.8. Comparing two idioms. (a) SpaceTree [Plaisant et al. 02]. (b) TreeJuxtaposer. From http://www.cs.umd.edu/hcil/spacetree and [Munzner et al. 03, Figure 1].

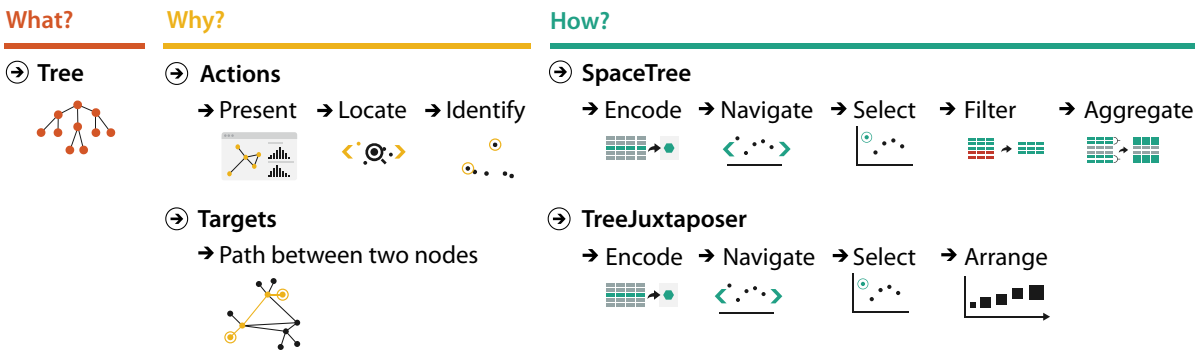


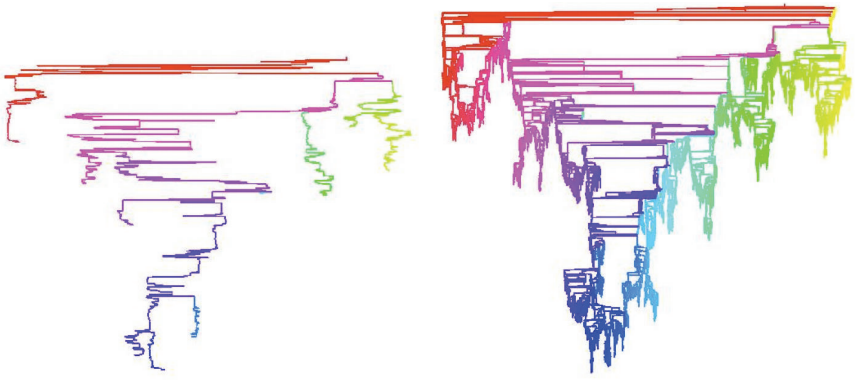
Figure 3.9. Analyzing what–why–how comparatively for the SpaceTree and TreeJuxtaposer idioms.

tools that use somewhat different idioms. What these tools take as input data is the same: a large tree composed of nodes and links. Why these tools are being used is for the same goal in this scenario: to present a path traced between two nodes of interest to a colleague. In more detail, both tools can be used to locate paths between nodes and identify them.

Some aspects of idioms are the same: both systems allow the user to navigate and to select a path, with the result that it's encoded differently from the nonselected paths through highlighting. The systems differ in how elements of the visualization are manipulated and arranged. SpaceTree ties the act of selection to a change of what is shown by automatically aggregating and filtering the unselected items. In contrast, TreeJuxtaposer allows the user to arrange areas of the tree to ensure visibility for areas of interest. Figure 3.9 summarizes this what–why–how analyis.

3.7.2 Deriving One Attribute

In a vis showing a complex network or tree, it is useful to be able to filter out most of the complexity by drawing a simpler picture that communicates the key aspects of its topological structure. One way to support this kind of summarization is to calculate a new derived attribute that measures the importance of each node in the graph and filter based on that attribute. Many different approaches to calculating importance have been proposed; **centrality metrics** do so in a way that takes into account network topology. The Strahler number is a measure of node importance originally



(a) (b)

Figure 3.10. The derived quantitative attribute of Strahler numbers is used to filter the tree in order to create a recognizable summary. (a) The important skeleton of a large tree is visible when only 5000 of the highest-ranked nodes are drawn. (b) The full tree has over a half million nodes. From [Auber 02, Figures 10 and 13].

developed in hydrogeology to characterize the branching structure of rivers that has been adapted and extended for use visualizing trees and networks [Auber 02]. Very central nodes have large Strahler numbers, whereas peripheral nodes have low values. The Strahler number is an example of a derived attribute for network data that is the result of a complex and global computation, rather than simply a local calculation on a small neighborhood around a node.

Figure 3.10 shows an example of filtering according to the Strahler derived attribute to summarize a tree effectively. The result of drawing only the top-ranked 5000 nodes and the links that connect them is a recognizable skeleton of the full tree, shown in Figure 3.10(a), while over a half million nodes are shown in Figure 3.10(b). In contrast, if the 5000 nodes to draw were picked randomly, the structure would not be understandable. Both versions of the network are also colored according to the Strahler number, to show how the centrality measure varies within the network.

To summarize this example concisely in terms of a what–why–how analysis, as shown in Figure 3.11, a new quantitative attribute is derived and used to filter away the peripheral parts of a tree, in support of the task of summarizing the tree's overall topology. As in the previous example, the tree is encoded as a node–link diagram, the most common choice for tree and network arrangment.

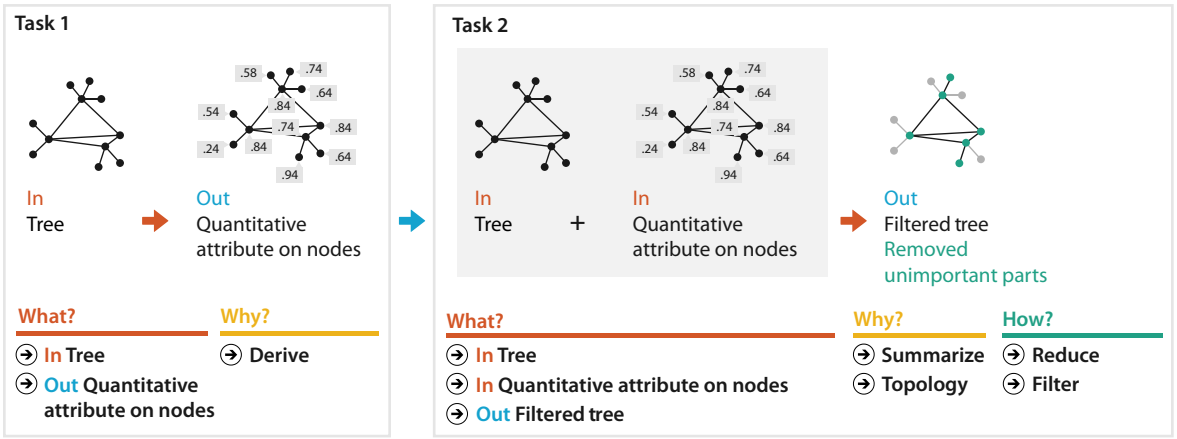


Figure 3.11. Analyzing a chained sequence of two instances where an attribute is derived in order to summarize a tree by filtering away the unimportant parts.

3.7.3 Deriving Many New Attributes

Data transformations can shed light into spatial data as well. In an example from computational fluid dynamics, linked derived spaces are used for feature detection [Henze 98]. The vis system shown in Figure 3.12 allows the user to quickly create plots of any two original or derived variables from the palette of variables shown in the upper left *derived fields* pane. The views are linked together with color highlighting. The power of this idiom lies in seeing where regions that are contiguous in one view fall in the other views.

► Multiple views are discussed further in Chapter 12.

The original dataset is a time-varying spatial field with measurements along a curvilinear mesh fitted to an airfoil. The plot in the *physical space* pane on the upper right of Figure 3.12 shows the data in this physical space, using the two spatial field variables. Undisturbed airflow enters the physical space from the left, and the back tip of the airfoil is on the right. Two important regions in back of the airfoil are distinguished with color: a red recirculation region and a yellow wake region. While these regions are not easy to distinguish in this physical view, they can be understood and selected more easily by interaction with the four other derived views. For example, in the derived space of *vorticity vs enthalpy* in the upper middle of Figure 3.12, the recirculation zone is distinguishable as a coherent spatial structure at the top, with the yellow wake also distinguishable beneath it. As the white box shows, the

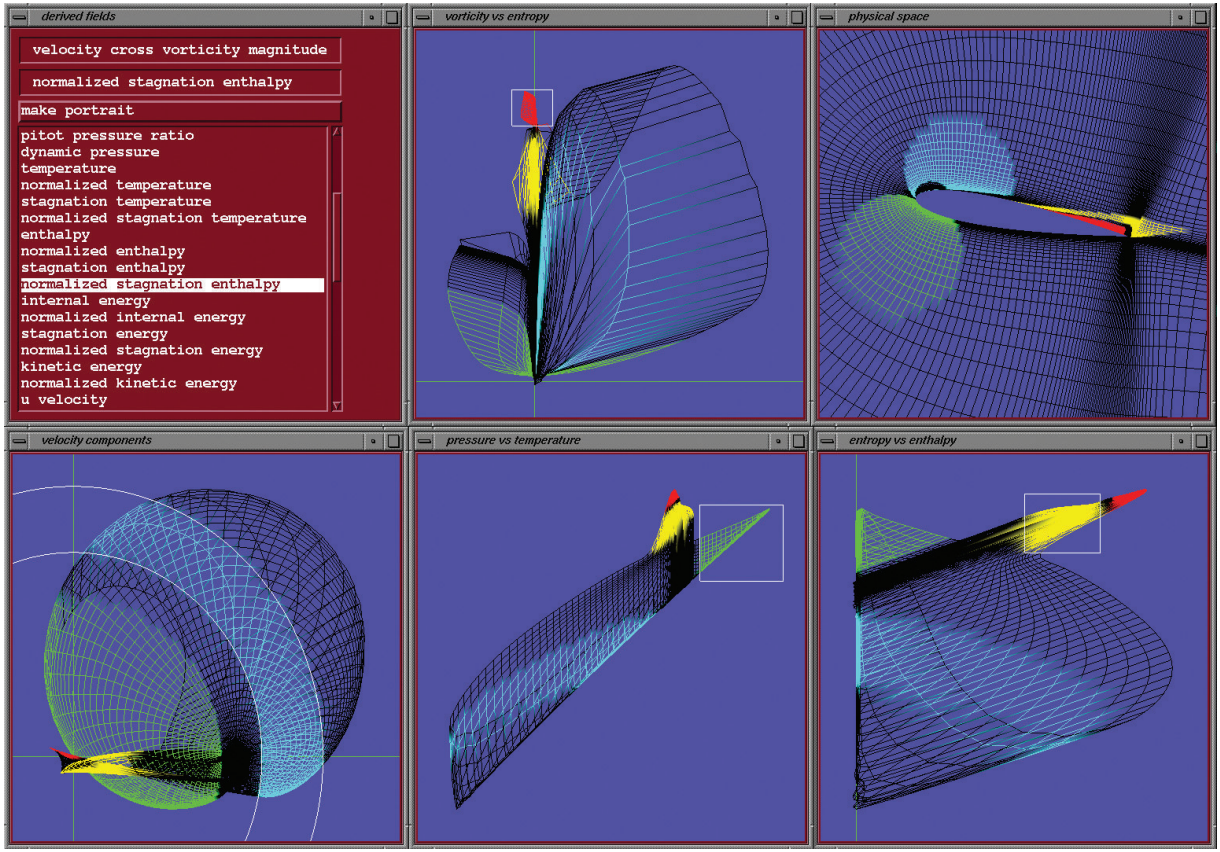


Figure 3.12. Computational fluid dynamics vis showing the list of many derived attributes (top left), one view of the original spatial field (top right), and four other views showing pairs of selected derived attributes. The multiple juxtaposed views are coordinated with shared colored highlights. From [Henze 98, Figure 5].

recirculation zone can easily be selected in this view. The *pressure vs temperature* pane in the bottom middle of Figure 3.12 shows another derived space made by plotting the pressure versus the temperature. In this view, the red recirculation zone and the yellow wake appear where both the pressure and temperature variables are high, in the upper right. Without getting into the exact technical meaning of the derived variables as used in fluid dynamics (vorticity, entropy, enthalpy, and so on), the point of this example is that many structures of interest in fluid dynamics can be seen more easily from layouts in the derived spaces.

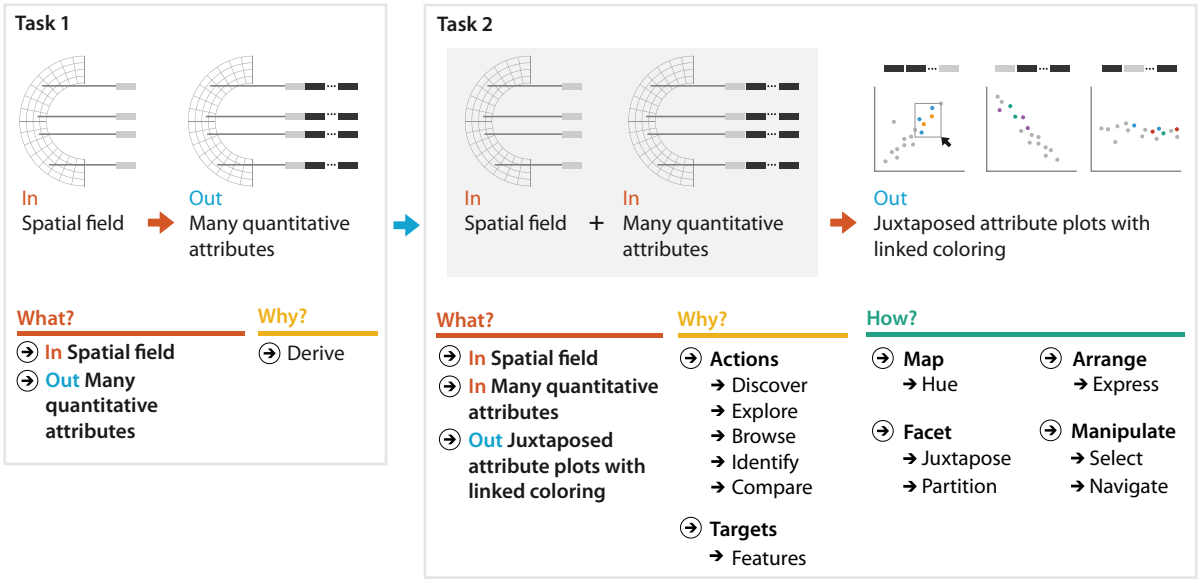


Figure 3.13. Analyzing a chained sequence, where many attributes are derived and visually encoded.

To summarize this example in terms of a what–why–how analysis, as shown in Figure 3.13, many new quantitative attributes are derived from an original spatial field dataset. Each pair of them is visually encoded into a view, as is the original spatial data, and the multiple juxtaposed views are coordinated with shared color coding and highlighting.

3.8 Further Reading

The Big Picture An earlier version of the what–why–how framework was first presented as a paper [Brehmer and Munzner 13], which includes a very detailed discussion of its relationship to the extensive previous work in classifications of tasks and interaction idioms. That discussion covers 30 previous classifications and 20 relevant references, ranging from a characterization of the scientific data analysis process [Springmeyer et al. 92], to an influential low-level task classification [Amar et al. 05], to a taxonomy of tasks for network datasets [Lee et al. 06], to a recent taxonomy of interaction dynamics [Heer and Shneiderman 12].

Who: Designers versus Users Some of the challenges inherent in bridging the gaps between vis designers and users are discussed in an influential paper [van Wijk 06].

Derive Many vis pipeline models discuss the idea of data transformation as a critical early step [Card et al. 99, Chi and Riedl 98], and others also point out the need to transform between different attribute types [Velleman and Wilkinson 93]. A later taxonomy of vis explicitly discusses the idea that data types can change as the result of the transformation [Tory and Möller 04b].

Examples The analysis examples are SpaceTree [Plaisant et al. 02], TreeJuxtaposer [Munzner et al. 03], Strahler numbers for tree simplification [Auber 02], and linked derived spaces for feature detection [Henze 98].

Visualization Analysis & Design

Tamara Munzner

"A must read for researchers, sophisticated practitioners, and graduate students."
—Jim Foley, College of Computing, Georgia Institute of Technology
Author of *Computer Graphics: Principles and Practice*

"Munzner's new book is thorough and beautiful. It belongs on the shelf of anyone touched and enriched by visualization."
—Chris Johnson, Scientific Computing and Imaging Institute, University of Utah

"This is the visualization textbook I have long awaited. It emphasizes abstraction, design principles, and the importance of evaluation and interactivity."
—Jim Hollan, Department of Cognitive Science, University of California, San Diego

"Munzner is one of the world's very top researchers in information visualization, and this meticulously crafted volume is probably the most thoughtful and deep synthesis the field has yet seen."
—Michael McGuffin, Department of Software and IT Engineering, École de Technologie Supérieure

"Munzner elegantly synthesizes an astounding amount of cutting-edge work on visualization into a clear, engaging, and comprehensive textbook that will prove indispensable to students, designers, and researchers."
—Steven Franconeri, Department of Psychology, Northwestern University

"Munzner shares her deep insights in visualization with us in this excellent textbook, equally useful for students and experts in the field."
—Jarke van Wijk, Department of Mathematics and Computer Science, Eindhoven University of Technology

"The book shapes the field of visualization in an unprecedented way."
—Wolfgang Aigner, Institute for Creative Media Technologies, St. Pölten University of Applied Sciences

"This book provides the most comprehensive coverage of the fundamentals of visualization design that I have found. It is a much-needed and long-awaited resource for both teachers and practitioners of visualization."
—Kwan-Liu Ma, Department of Computer Science, University of California, Davis

This book's unified approach encompasses information visualization techniques for abstract data, scientific visualization techniques for spatial data, and visual analytics techniques for interweaving data transformation and analysis with interactive visual exploration. Suitable for both beginners and more experienced designers, the book does not assume any experience with programming, mathematics, human–computer interaction, or graphic design.

CRC Press
Taylor & Francis Group
an **informa** business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

K14708

ISBN: 978-1-4665-0891-0
90000
9 781466 508910

Visualization/Human–Computer Interaction/Computer Graphics