## Module No-1

# THE SEVEN STAGES OF DATA VISUALIZATION

By Mangesh Balpande

# Why Data Display Requires Planning

- Each set of data has particular display needs, and the purpose for which you're using the data set has just as much of an effect on those needs as the data itself.
- Characteristics of a data set help determine what kind of visualization

1. **Too Much Information**

402-403 M TeraB

2. **Data Collection:**
   - We're getting better and better at collecting data, but we lag in what we can do with it. Lots of data is out there, but it's not being used to its greatest potential because it's not being visualized as well as it could be.
   - With all the data we've collected, we still don't have many satisfactory answers to the sort of questions that we started with. This is the greatest challenge of our information-rich era.

# Why Data Display Requires Planning(Cont..)

## 3. Thinking About Data

- When AOL released a data set containing the search queries of millions of users that had been "randomized" to protect the innocent, articles soon appeared about how people could be identified by—and embarrassed by—information regarding their search habits.

- With all the data we've collected, we still don't have many satisfactory answers to the sort of questions that we started with. This is the greatest challenge of our information-rich era.

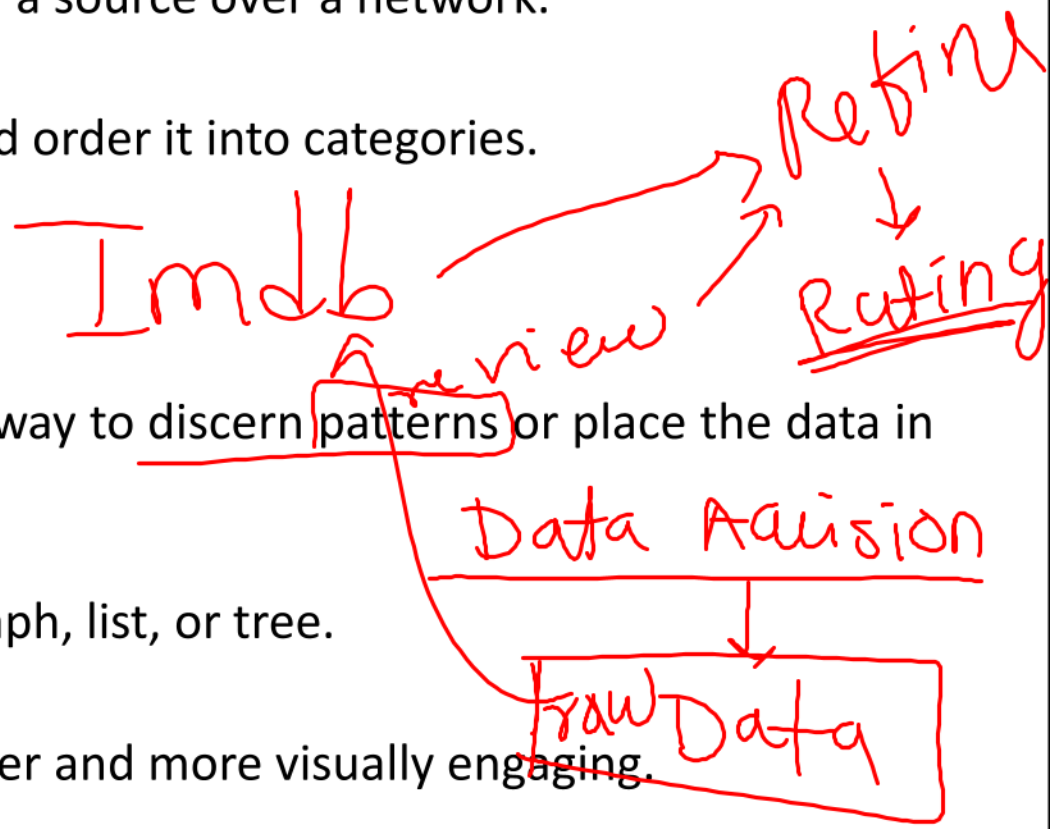
## 4. Data Never Stays the Same

- How do we build representations of data that adjust to new values every second, hour, or week? This is a necessity because most data comes from the real world, where there are no absolutes.

- The temperature changes, the train runs late, or a product launch causes the traffic pattern on a web site to change drastically.

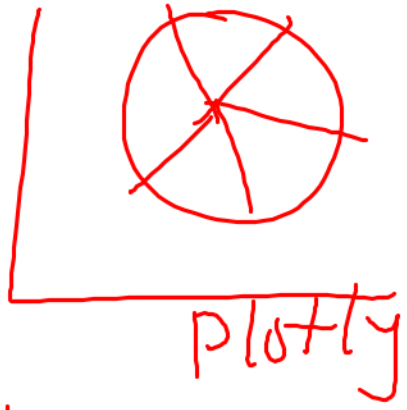
# Why Data Display Requires Planning(Cont..)

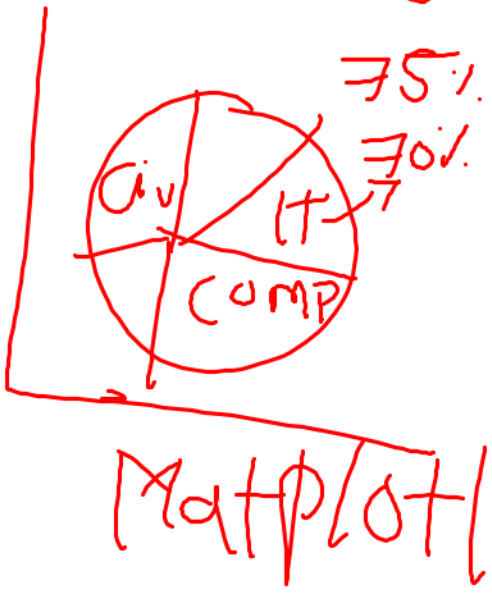
- **Process:** The process of understanding data begins with a set of numbers and a question. The following steps:
  1. **Acquire:** Obtain the data, whether from a file on a disk or a source over a network.
  2. **Parse:** Provide some structure for the data's meaning, and order it into categories.
  3. **Filter:** Remove all but the data of interest.
  4. **Mine:** Apply methods from statistics or data mining as a way to discern patterns or place the data in mathematical context.
  5. **Represent:** Choose a basic visual model, such as a bar graph, list, or tree.
  6. **Refine:** Improve the basic representation to make it clearer and more visually engaging.
  7. **Interact:** Add methods for manipulating the data or controlling what features are visible.



# Acquire:

- The acquisition step involves obtaining the data. Like many of the other steps, this can be either extremely complicated or very simple.



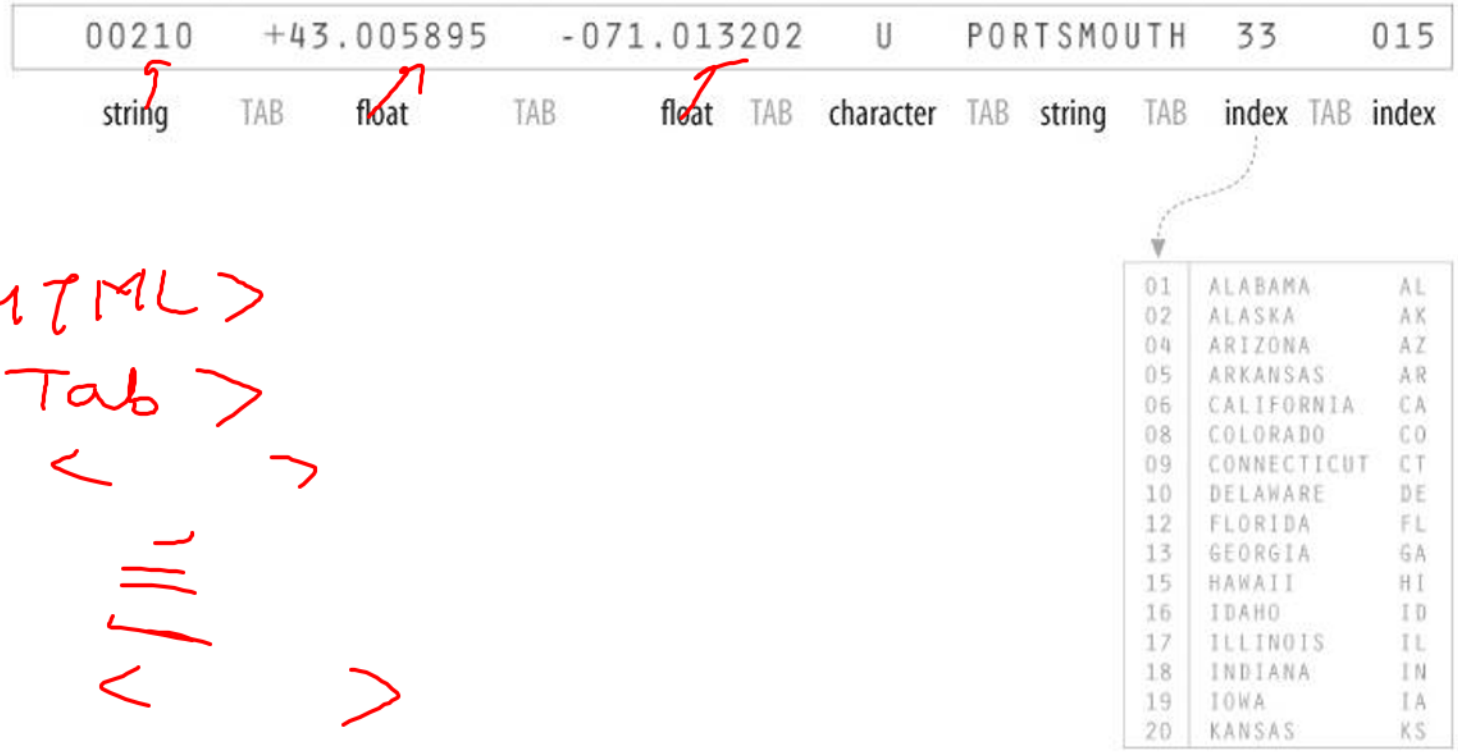




| | | | | | | |
|---|---|---|---|---|---|---|
| 00210 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00211 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00212 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00213 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00214 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00215 | +43.005895 | -071.013202 | U | PORTSMOUTH | 33 | 015 |
| 00501 | +40.922326 | -072.637078 | U | HOLTSVILLE | 36 | 103 |
| 00544 | +40.922326 | -072.637078 | U | HOLTSVILLE | 36 | 103 |
| 00601 | +18.165273 | -066.722583 | | ADJUNTAS | 72 | 001 |
| 00602 | +18.393103 | -067.180953 | | AGUADA | 72 | 003 |
| 00603 | +18.455913 | -067.145780 | | AGUADILLA | 72 | 005 |
| 00604 | +18.493520 | -067.135883 | | AGUADILLA | 72 | 005 |
| 00605 | +18.465162 | -067.141486 | P | AGUADILLA | 72 | 005 |
| 00606 | +18.172947 | -066.944111 | | MARICAO | 72 | 093 |
| 00610 | +18.288685 | -067.139696 | | ANASCO | 72 | 011 |
| 00611 | +18.279531 | -066.802170 | P | ANGELES | 72 | 141 |
| 00612 | +18.450674 | -066.698262 | | ARECIBO | 72 | 013 |
| 00613 | +18.458093 | -066.732732 | P | ARECIBO | 72 | 013 |
| 00614 | +18.429675 | -066.674506 | P | ARECIBO | 72 | 013 |
| 00616 | +18.444792 | -066.640678 | | BAJADERO | 72 | 013 |

*Figure 1-1. Zip codes in the format provided by the U.S. Census Bureau*

# Parse:

- After you acquire the data, it needs to be parsed—changed into a format that tags each part of the data with its intended use.



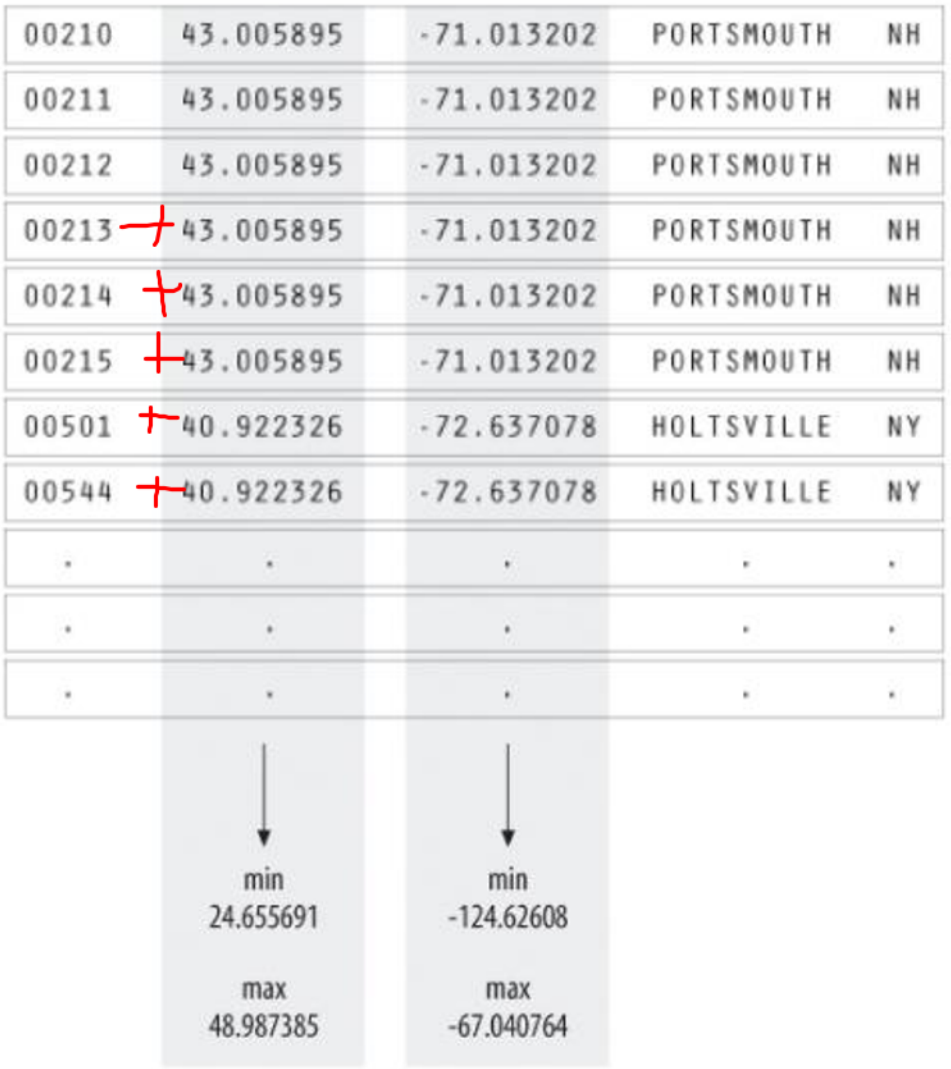
# Filter:

- involves filtering the data to remove portions not relevant to our use.

- Example: we'll be focusing on the contiguous 48 states, so the records for cities and towns that are not part of those states— Alaska, Hawaii, and territories such as Puerto Rico—are removed.

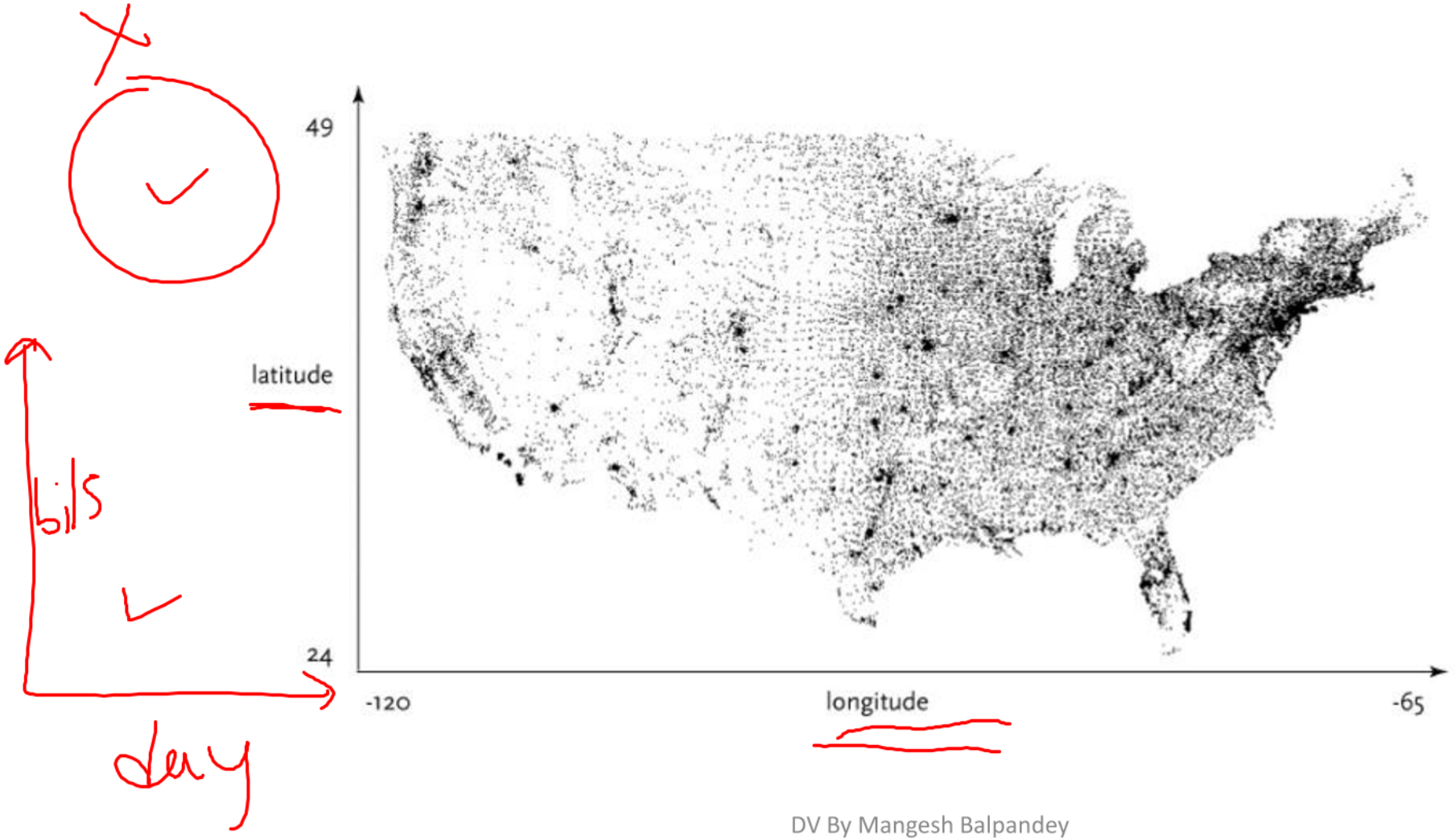
# Mine:

- This step involves math, statistics, and data mining.
- The program figure out the minimum and maximum values for latitude and longitude by running through the data, so that it can be presented on a screen at a proper scale.



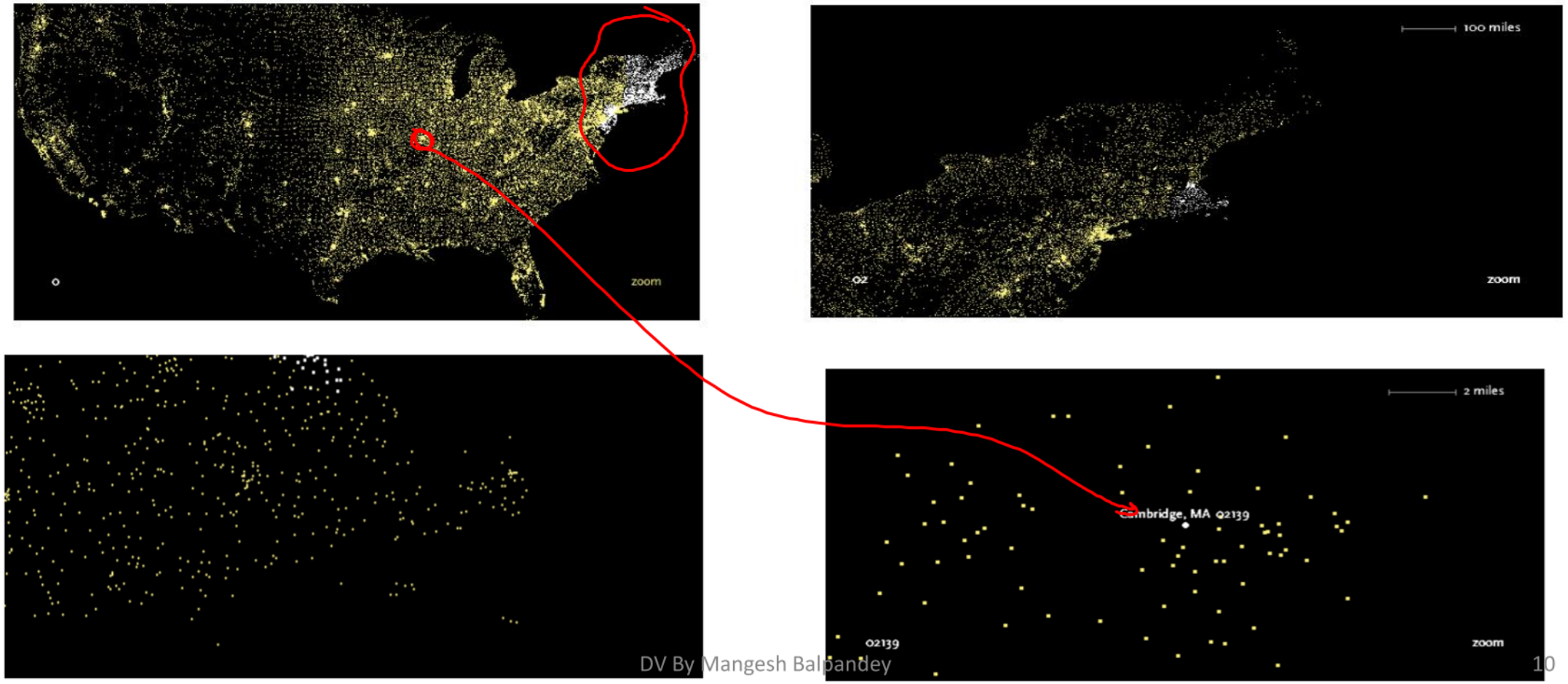
# Represent:

- This step determines the basic form that a set of data will take. Some data sets are shown as lists, others are structured like trees, and so forth.



# Interact:

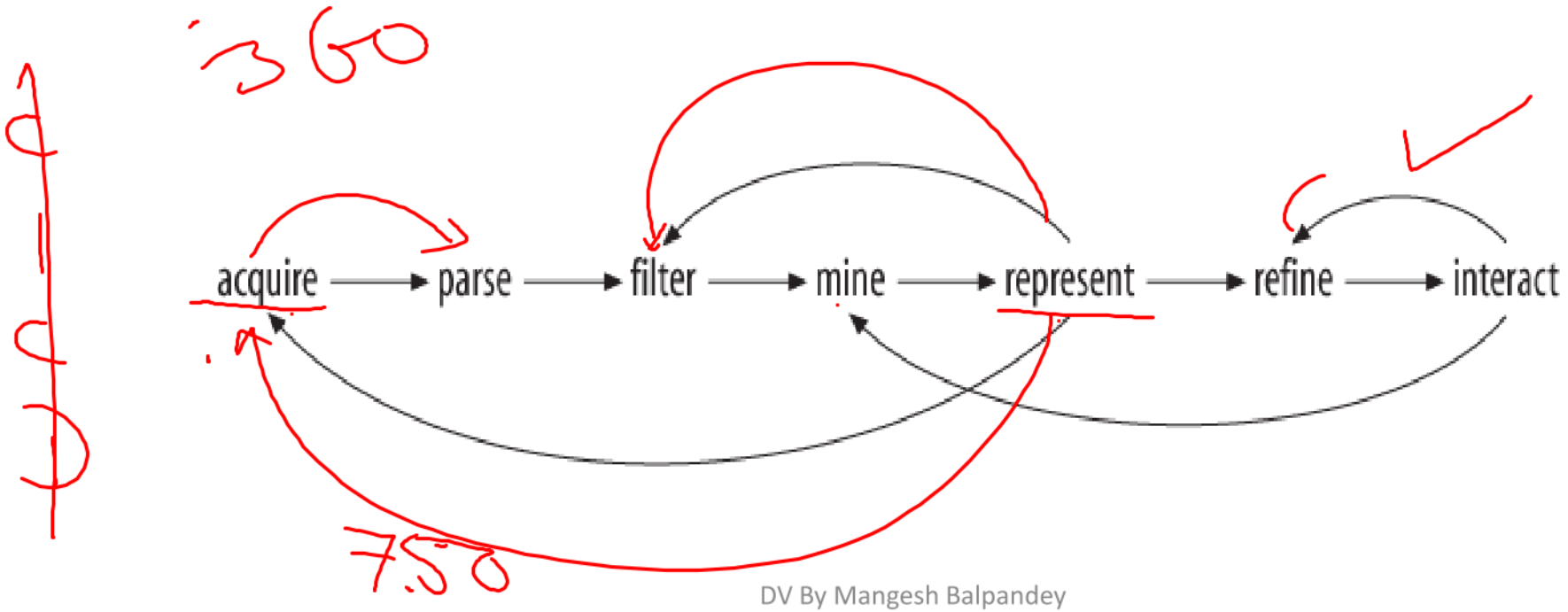
- The next stage of the process adds interaction, letting the user control or explore the data. Interaction might cover things like selecting a subset of the data or changing the viewpoint.



# Iteration and Combination

- The later decisions commonly reflect on earlier stages. Each step of the process is inextricably linked because of how the steps affect one another. In the Zipdecode application, for instance:

1) The need for a compact representation on the screen led to re-filter the data to include only the contiguous 48 states.



# Iteration and Combination(Cont…)

2) The representation step affected acquisition because after development if the application is modified it could show data that was downloaded over a slow Internet connection to the browser.

3) Change to the structure of the data allows the points to appear slowly, as they are first read from the data file, employing the data itself as a “progress bar.”

4) Interaction by typing successive numbers meant that the colors had to be modified in the visual refinement step to show a slow transition as points in the display are added or removed.

# Introduction to Processing

- The latest version of Processing can be downloaded at:

  http://processing.org/download

- Processing is based on Java, but because program elements in Processing are fairly simple, you can learn to use it from this book even if you don't know any Java.

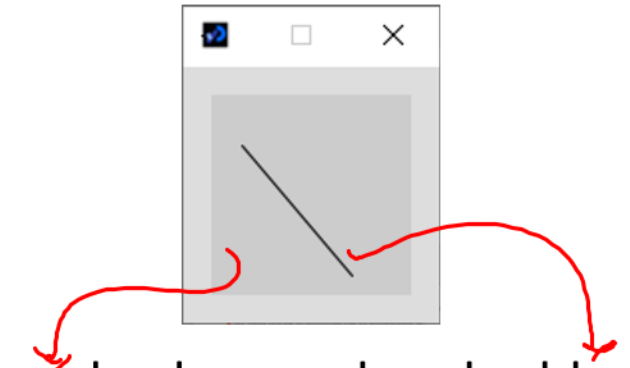
blood till

Computer Graphics
<include graphics.h>

# Introduction(Cont...)

- A Processing program is called a sketch.
- The idea is to make Java-style programming feel more like scripting, and adopt the process of scripting to quickly write code.
- Sketches are stored in the sketchbook, a folder that's used as the default location for saving all of your projects. When you run Processing, the sketch last used will automatically open.

```
line(15, 25, 70, 90);
```

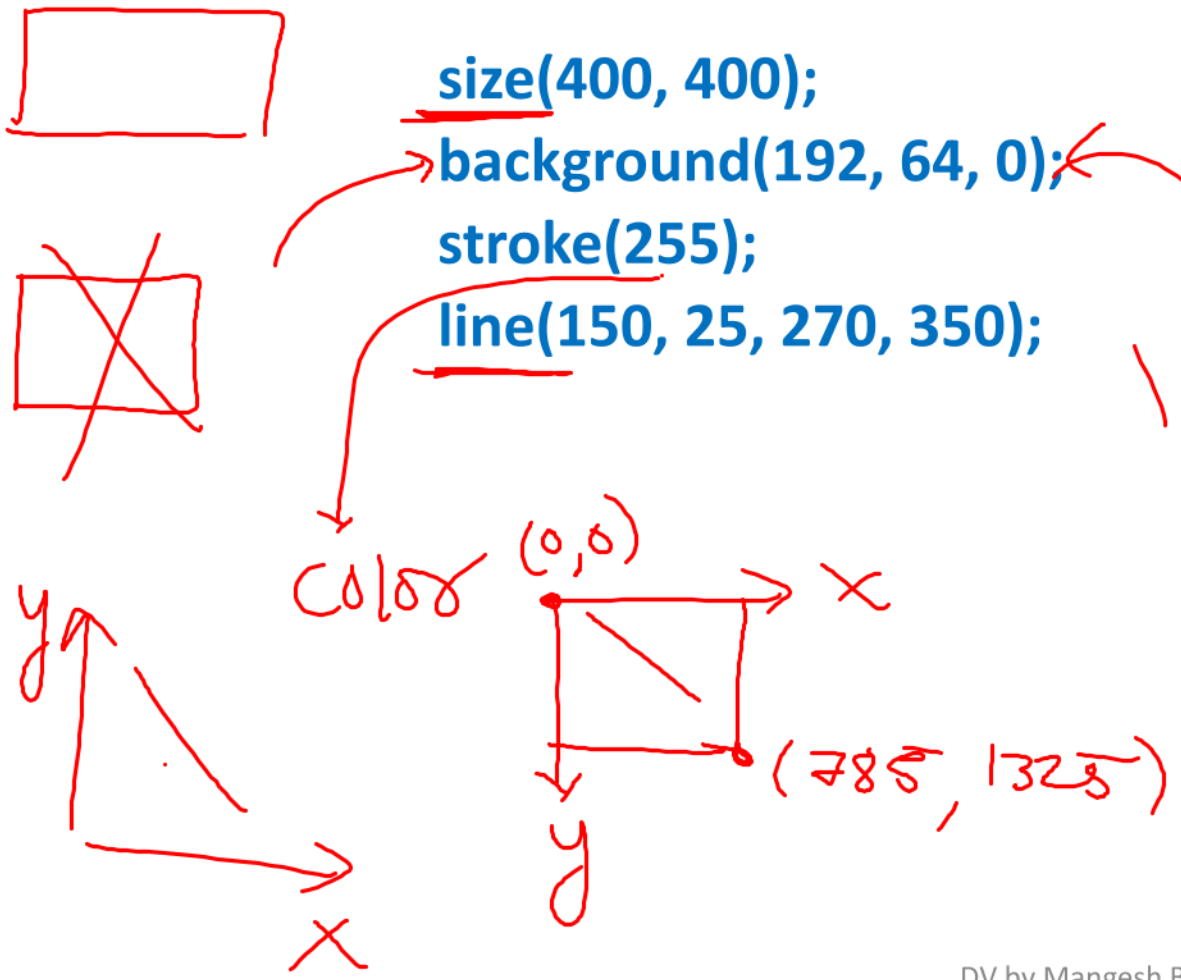


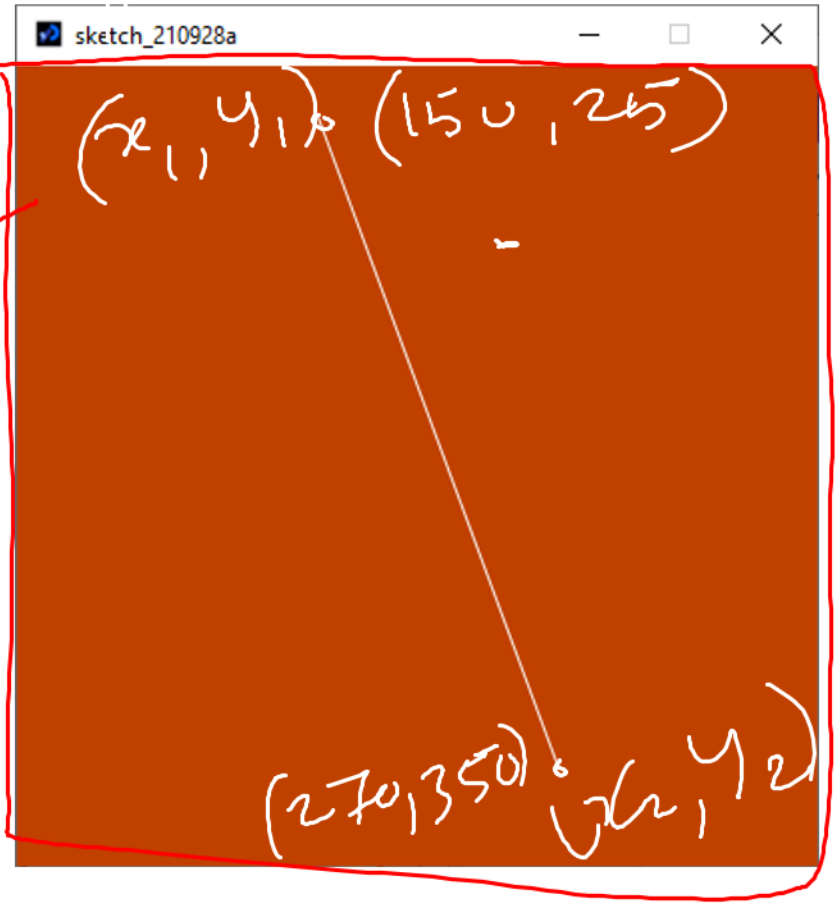
- The result will appear in a new window, with a gray background and a black line from coordinate (15, 25) to (70, 90).

# Introduction(Cont...)

- Building on this program to change the size of the display window and set the background color.

```
size(400, 400);
background(192, 64, 0);
stroke(255);
line(150, 25, 270, 350);
```



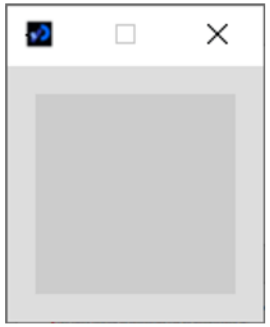



# Introduction(Cont...)

- By default, colors are specified in the range 0 to 255. Other variations of the parameters to the stroke( ) function

```
stroke(255);          // sets the stroke color to white
stroke(255, 255, 255);  // identical to stroke(255)
stroke(255, 128, 0);    // bright orange (red 255, green 128, blue 0)
stroke(#FF8000);        // bright orange as a web color
stroke(255, 128, 0, 128); // bright orange with 50% transparency
```

R - 255
G - 128
B - 0

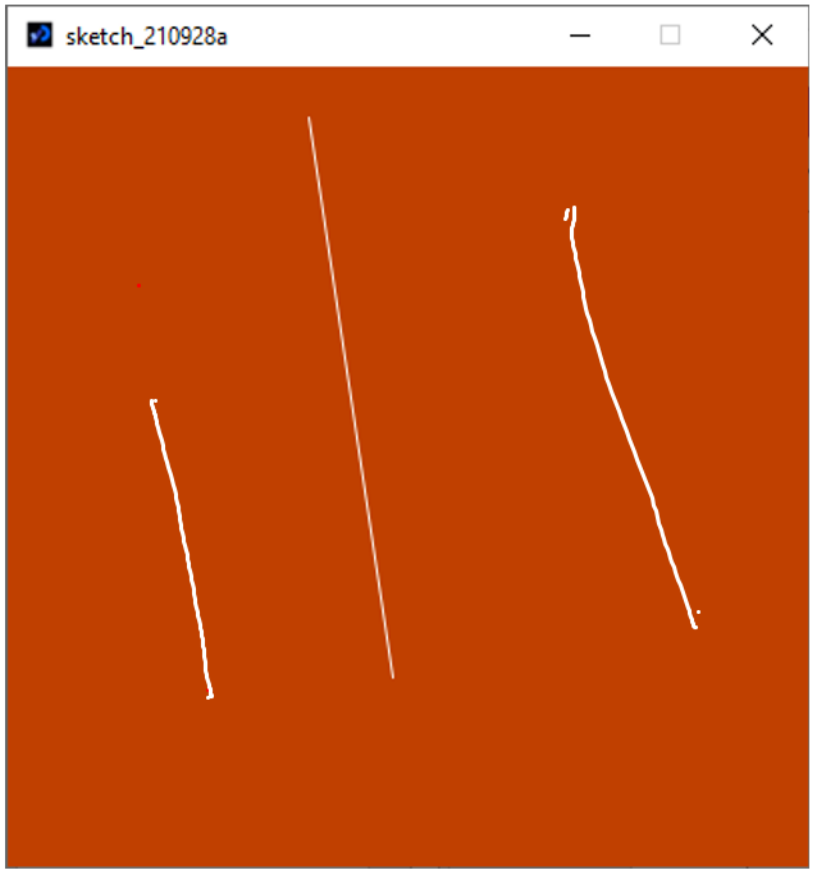


# Introduction(Cont...)

- To draw just a single line that follows the mouse, move the background( ) command to the draw( ) function, which will clear the display window (filling it with orange) each time draw( ) runs:

```
void setup( ) {
        size(400, 400);
        stroke(255);
}

void draw( ) {
        background(192, 64, 0);
        line(150, 25, mouseX, mouseY);
}
```



# Introduction(Cont...)

- More advanced mouse handling can also be introduced; for instance, the mousePressed( ) method will be called whenever the mouse is pressed.

- So, in the following example, when the mouse is pressed, the screen is cleared via the background( ) command:

```
void setup( ) {
        size(400, 400);
        stroke(255);
}

void draw( ) {
        line(150, 25, mouseX, mouseY);
}

void mousePressed( ) {
        background(192, 64, 0);
}
```