

Data Visualization (CSCI 627/490)

Web Programming

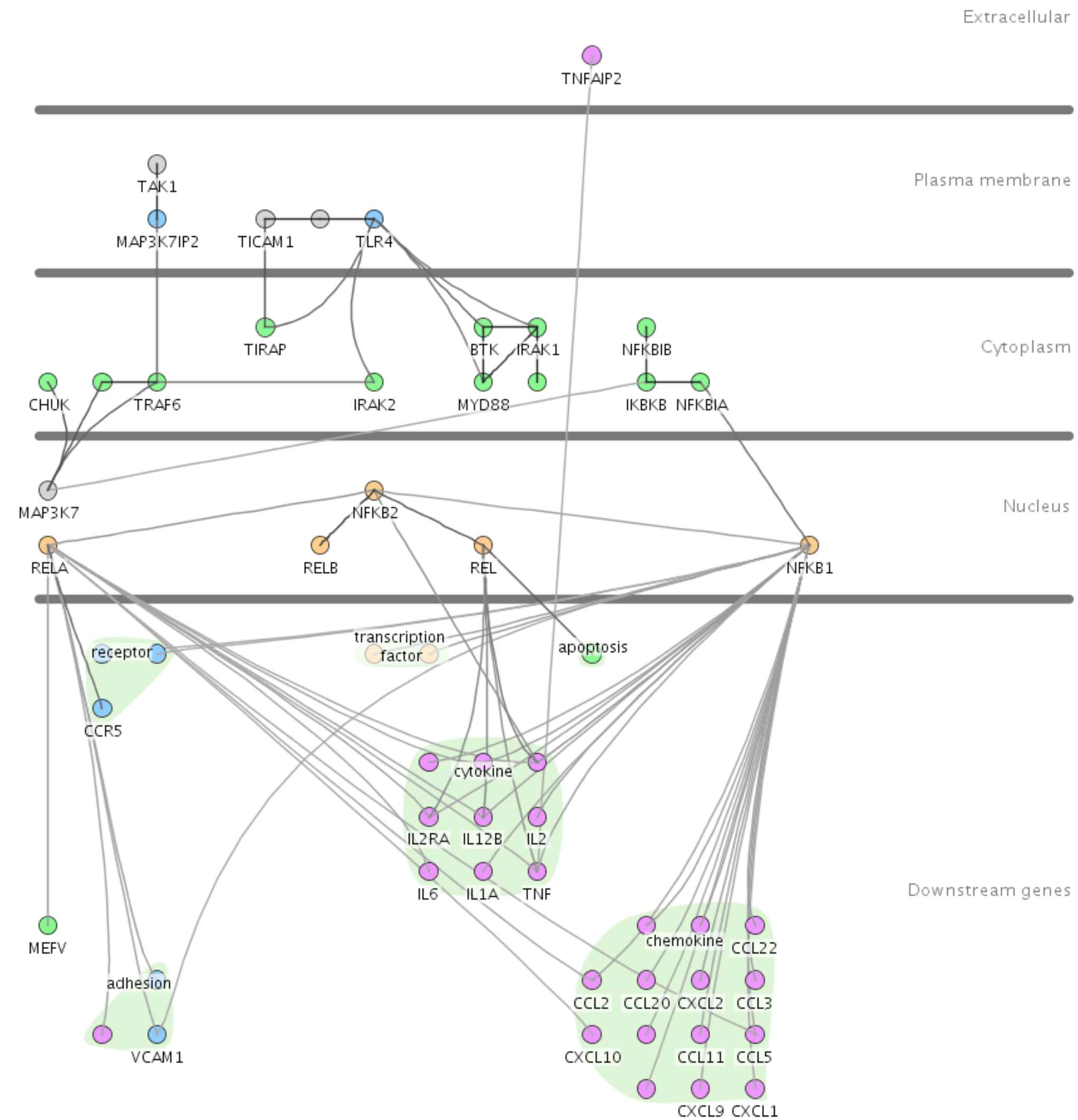
Dr. David Koop

Definition of Visualization

“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”

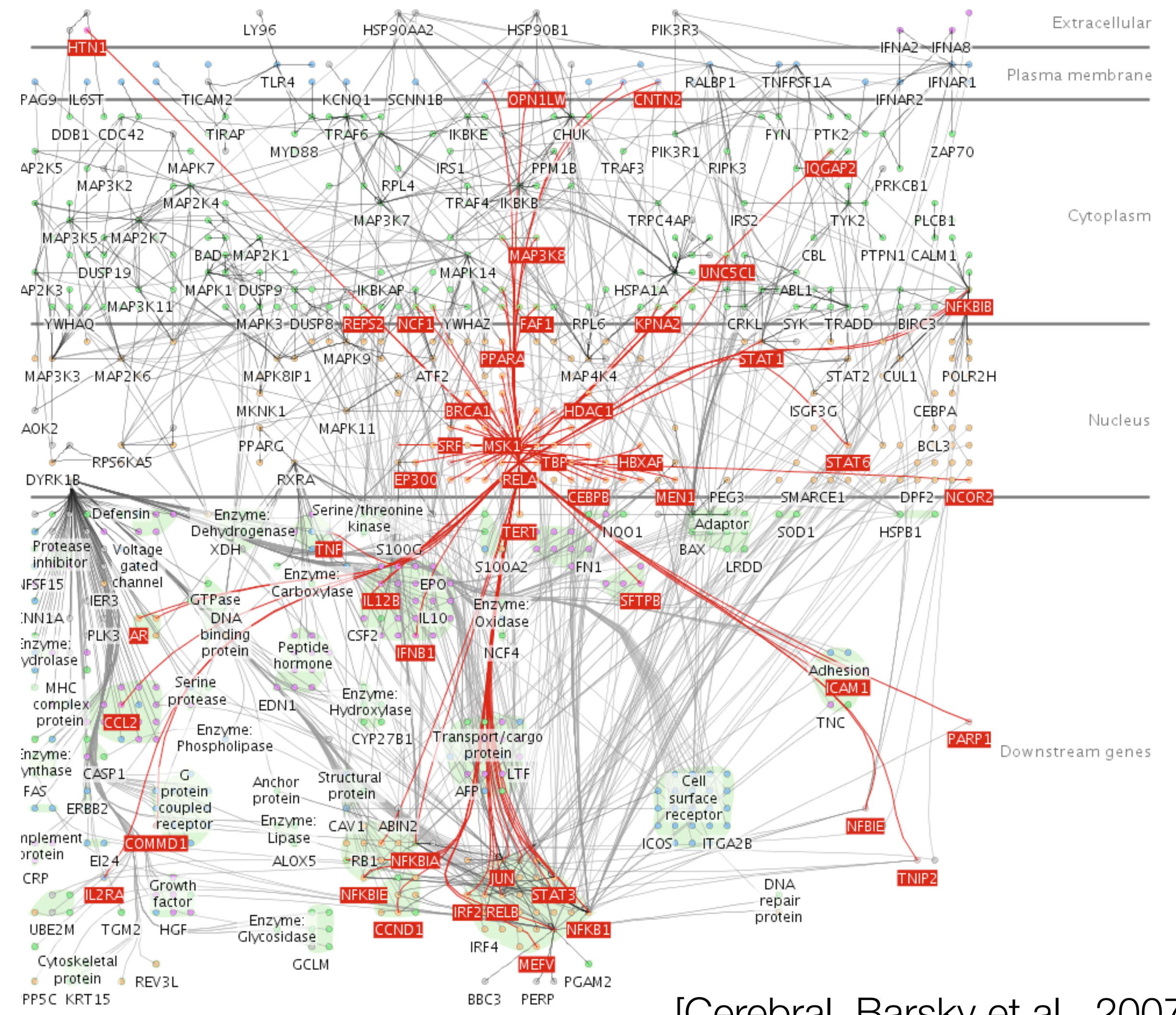
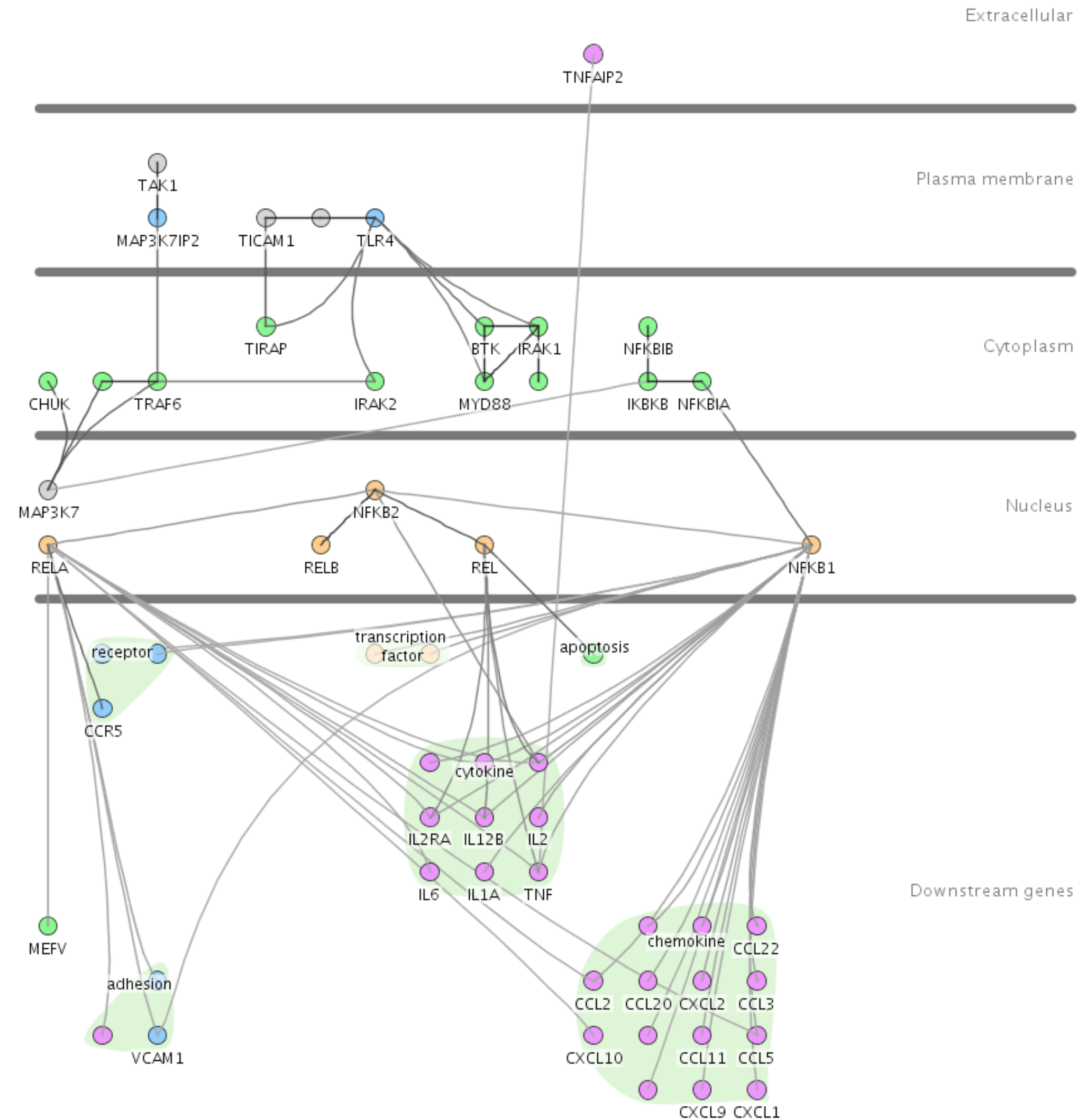
— T. Munzner

Why Computers?



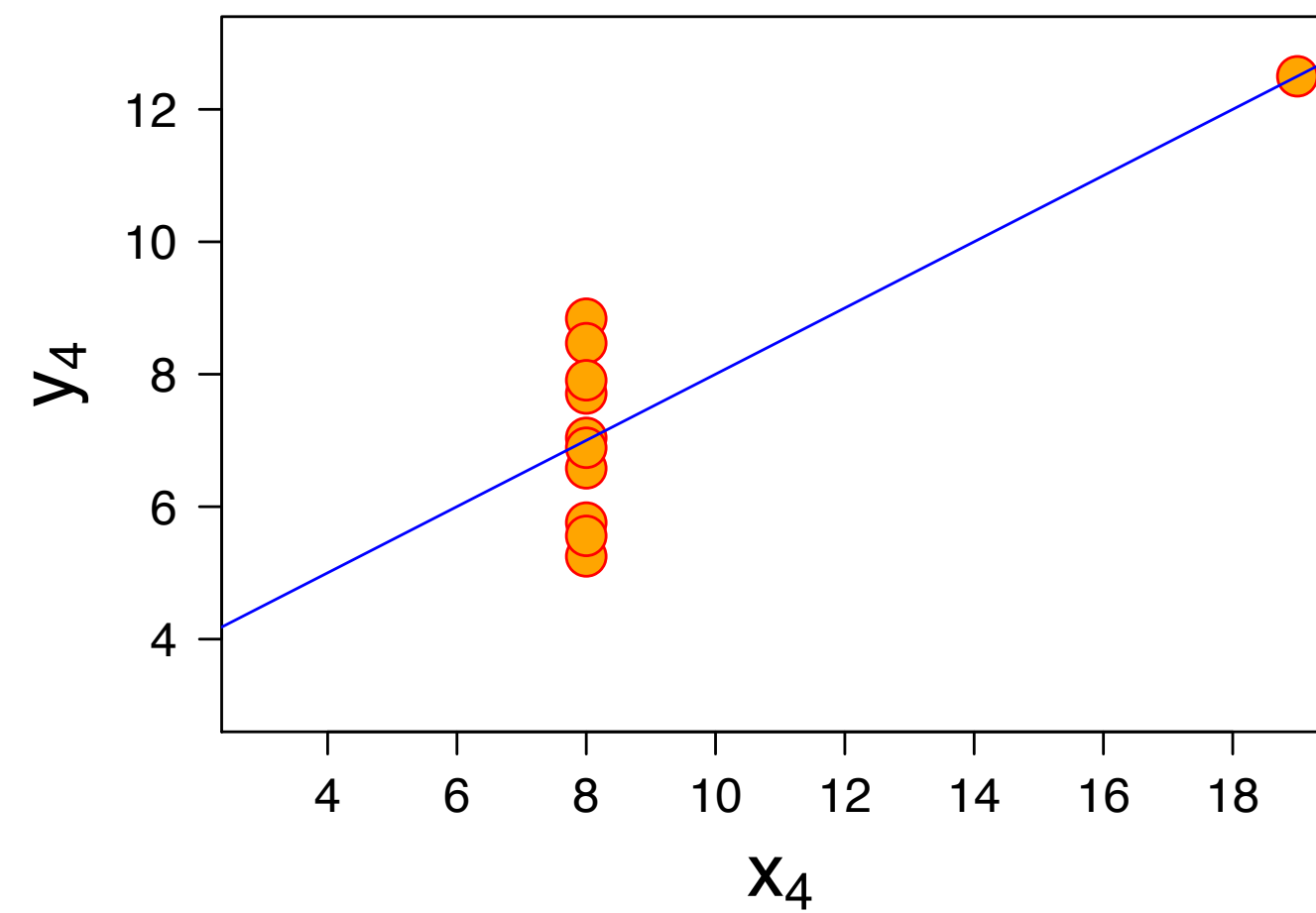
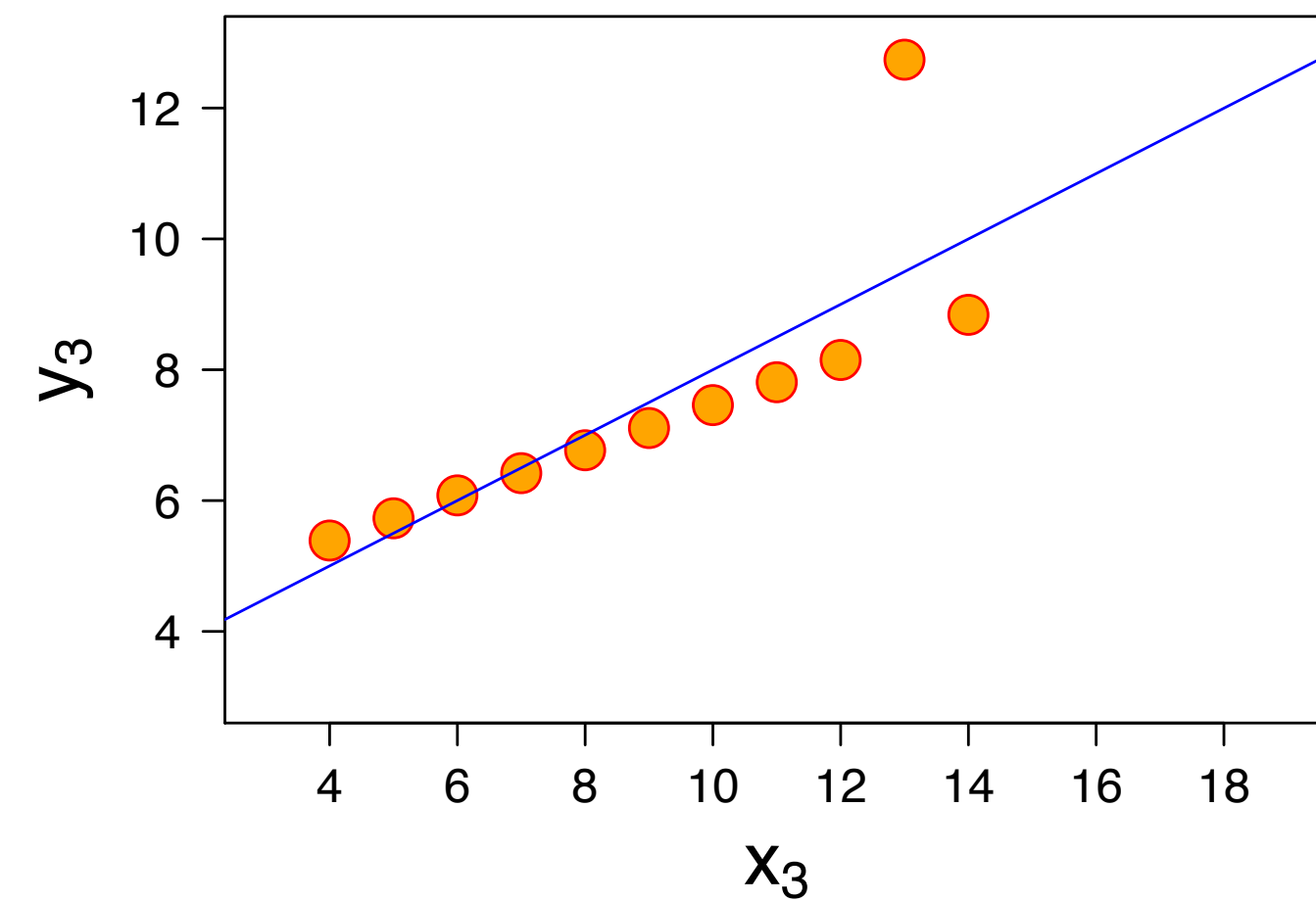
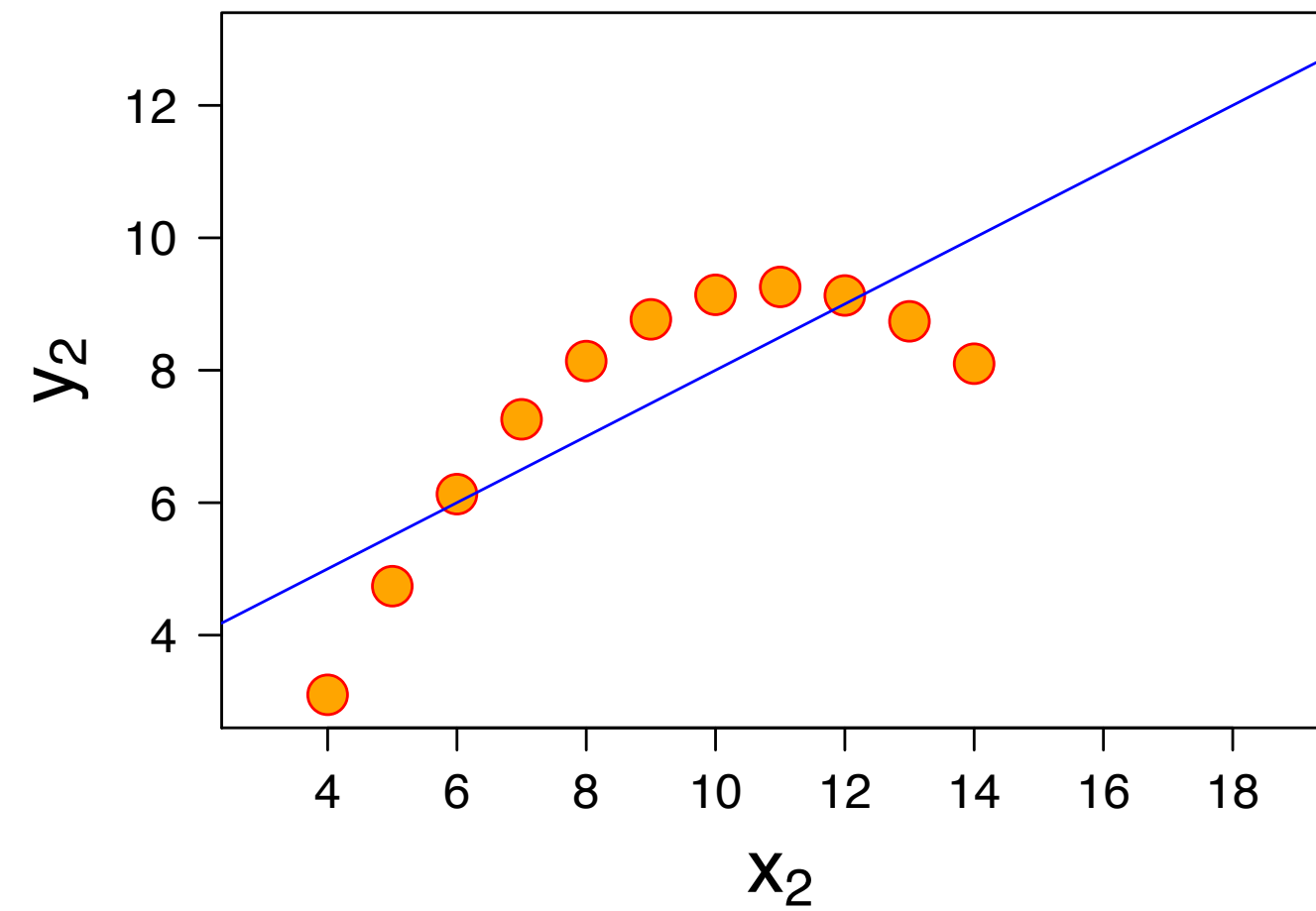
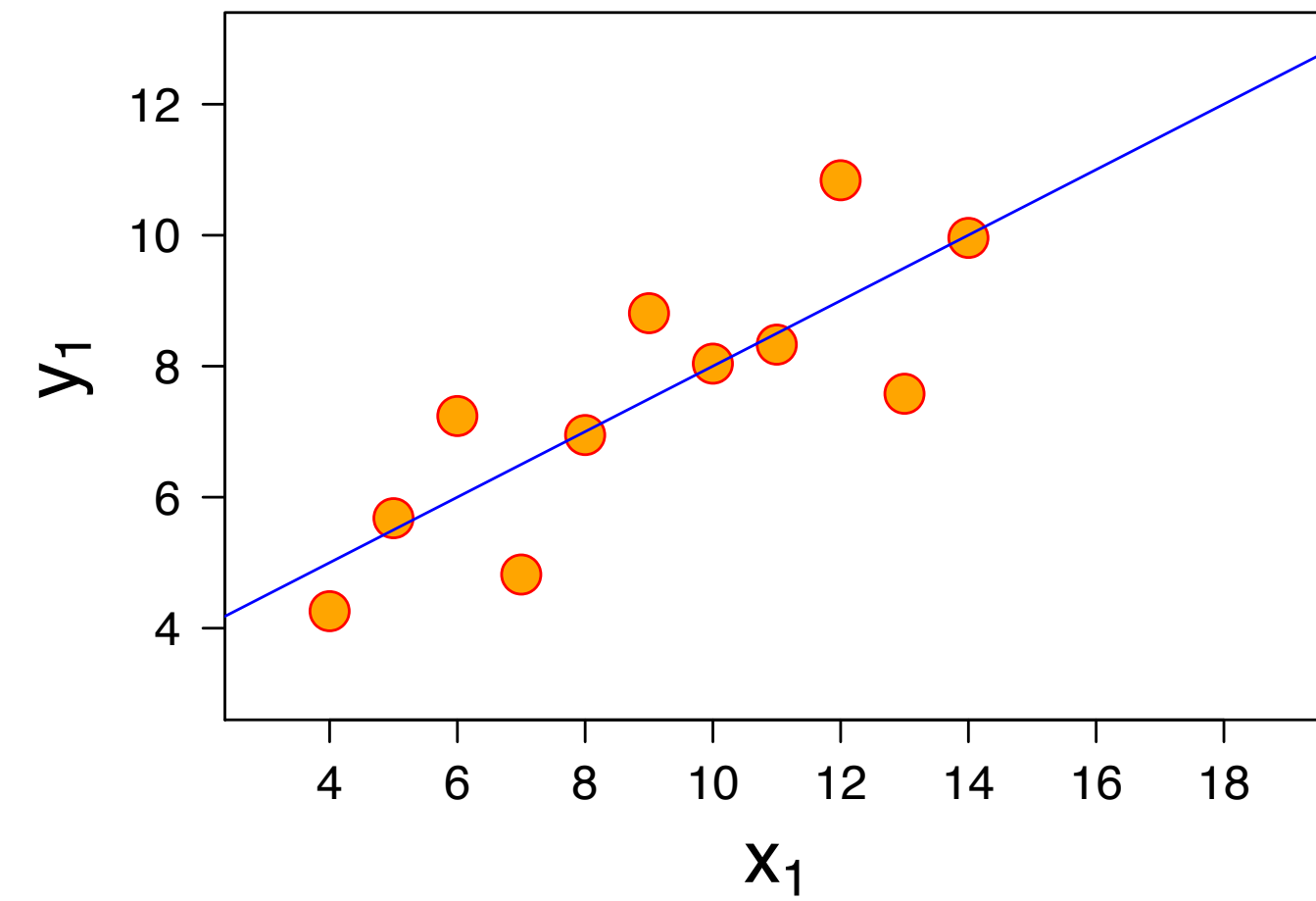
[Cerebral, Barsky et al., 2007]

Why Computers?



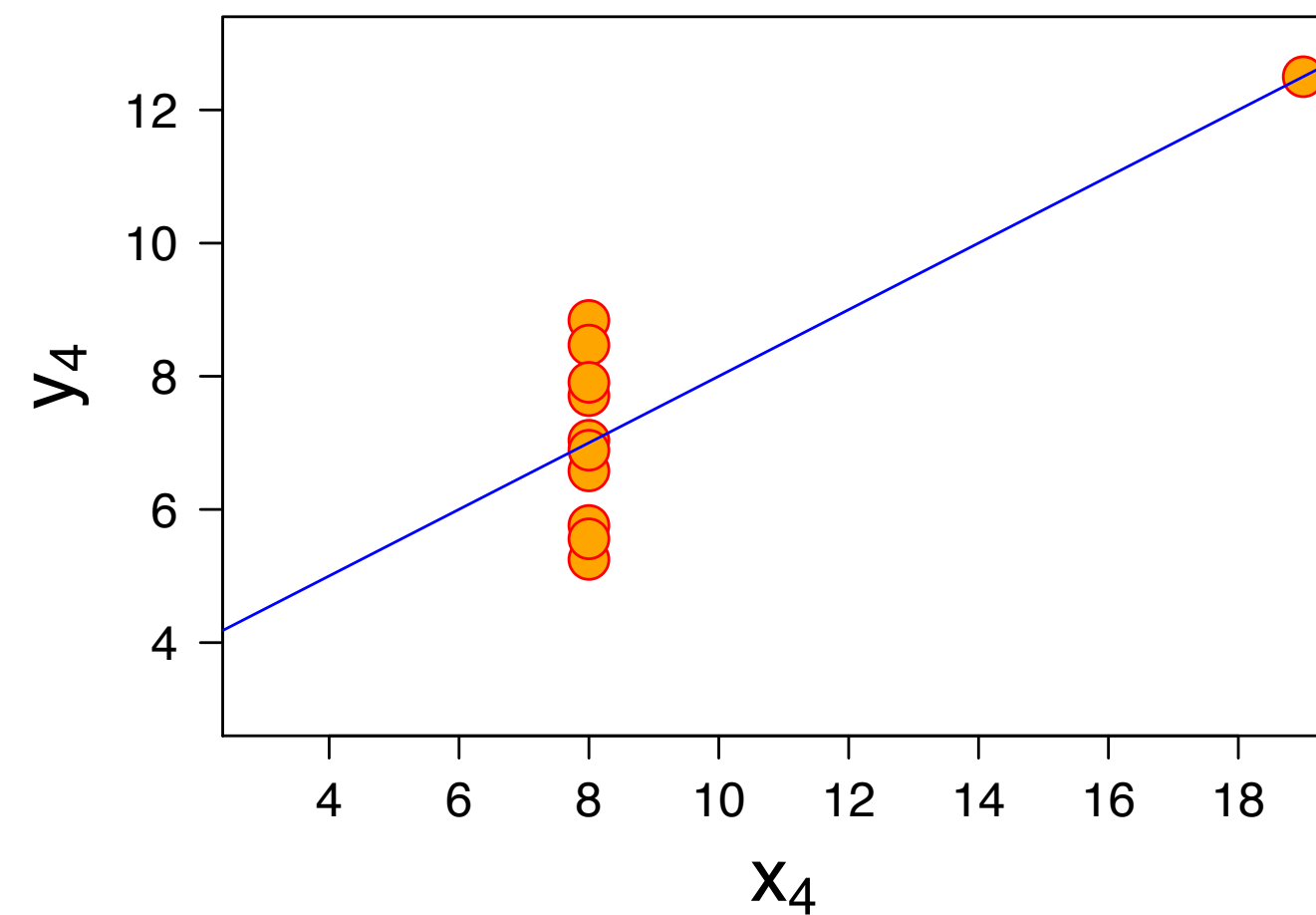
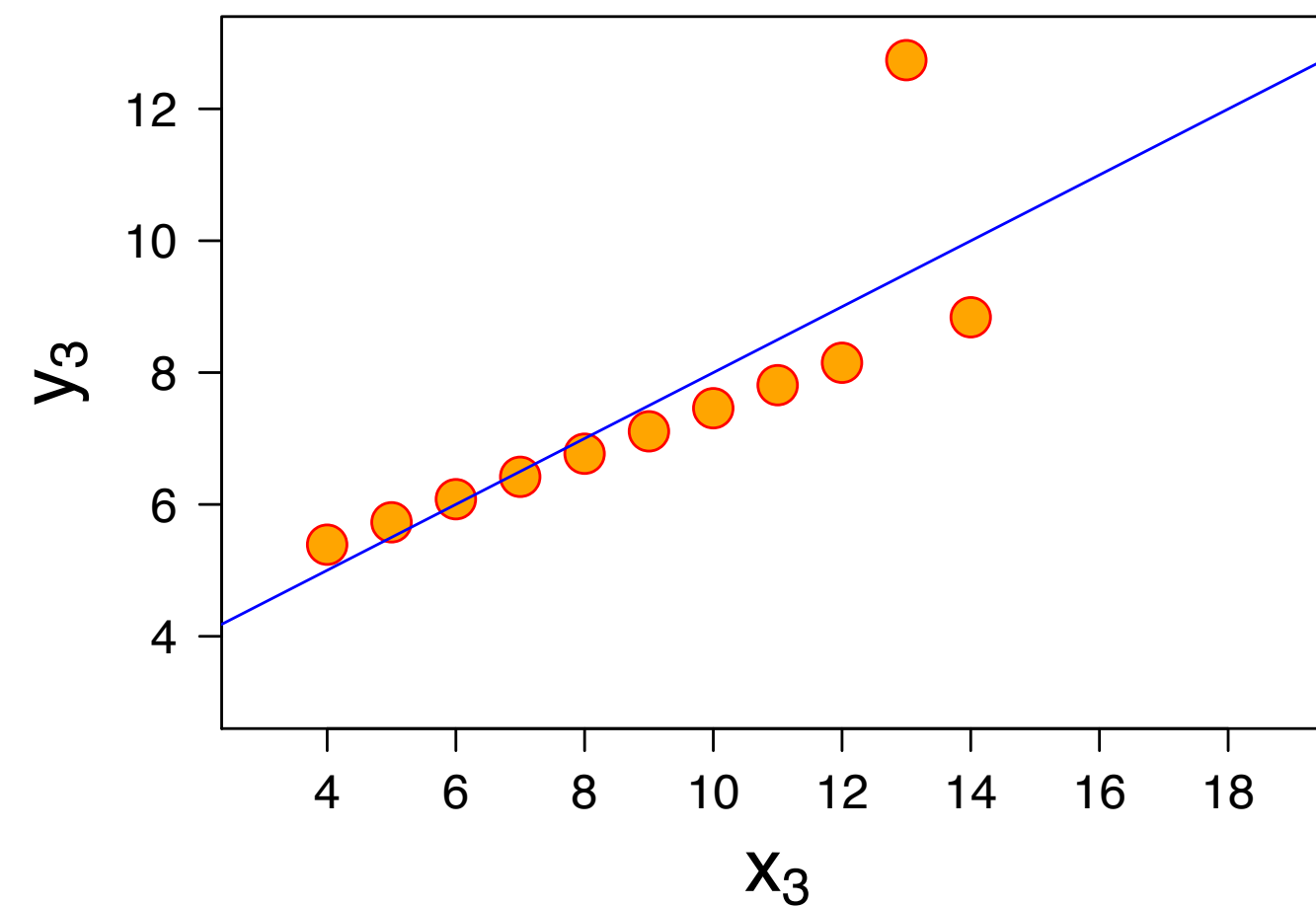
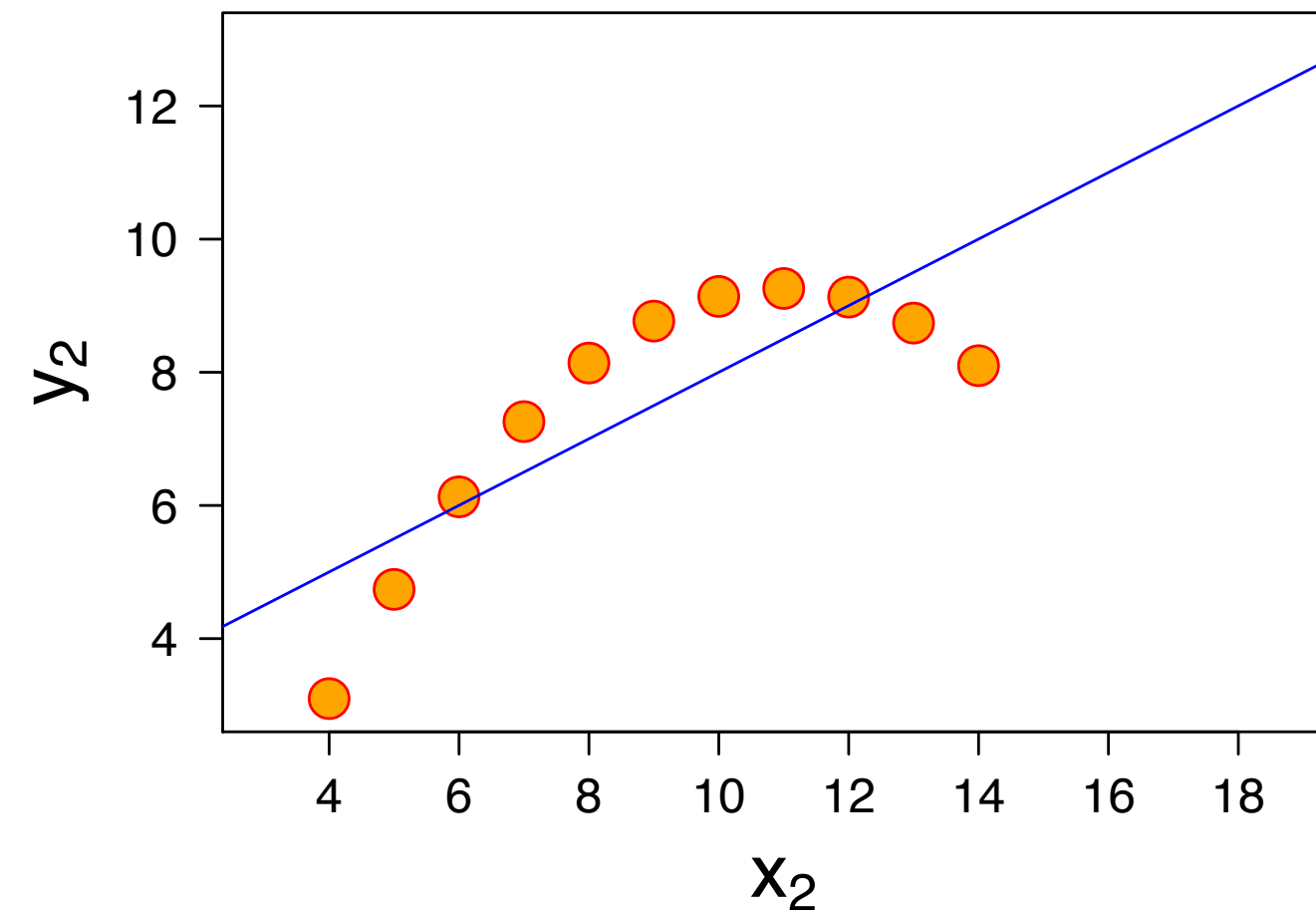
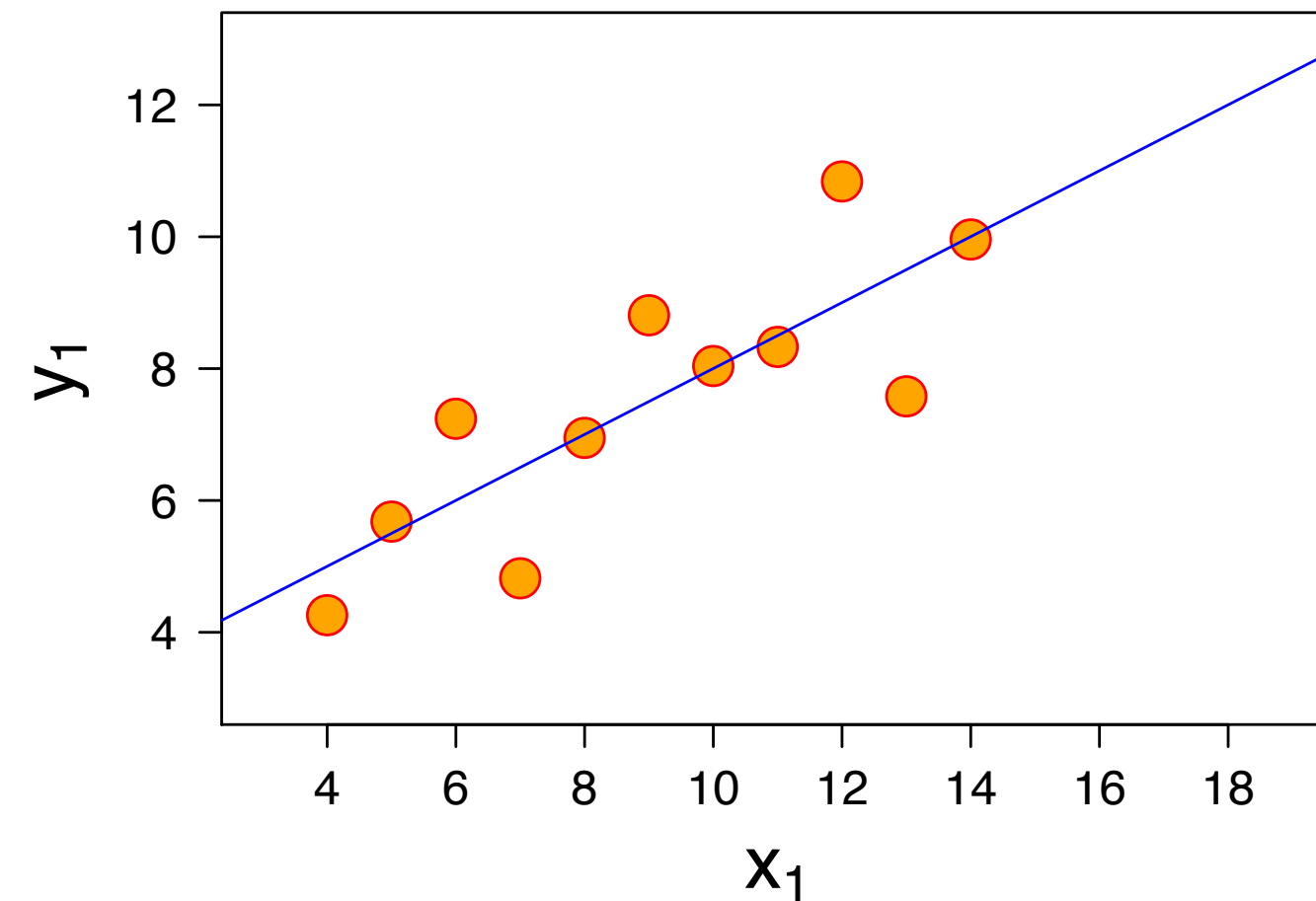
[Cerebral, Barsky et al., 2007]

Why Visual?



[F. J. Anscombe]

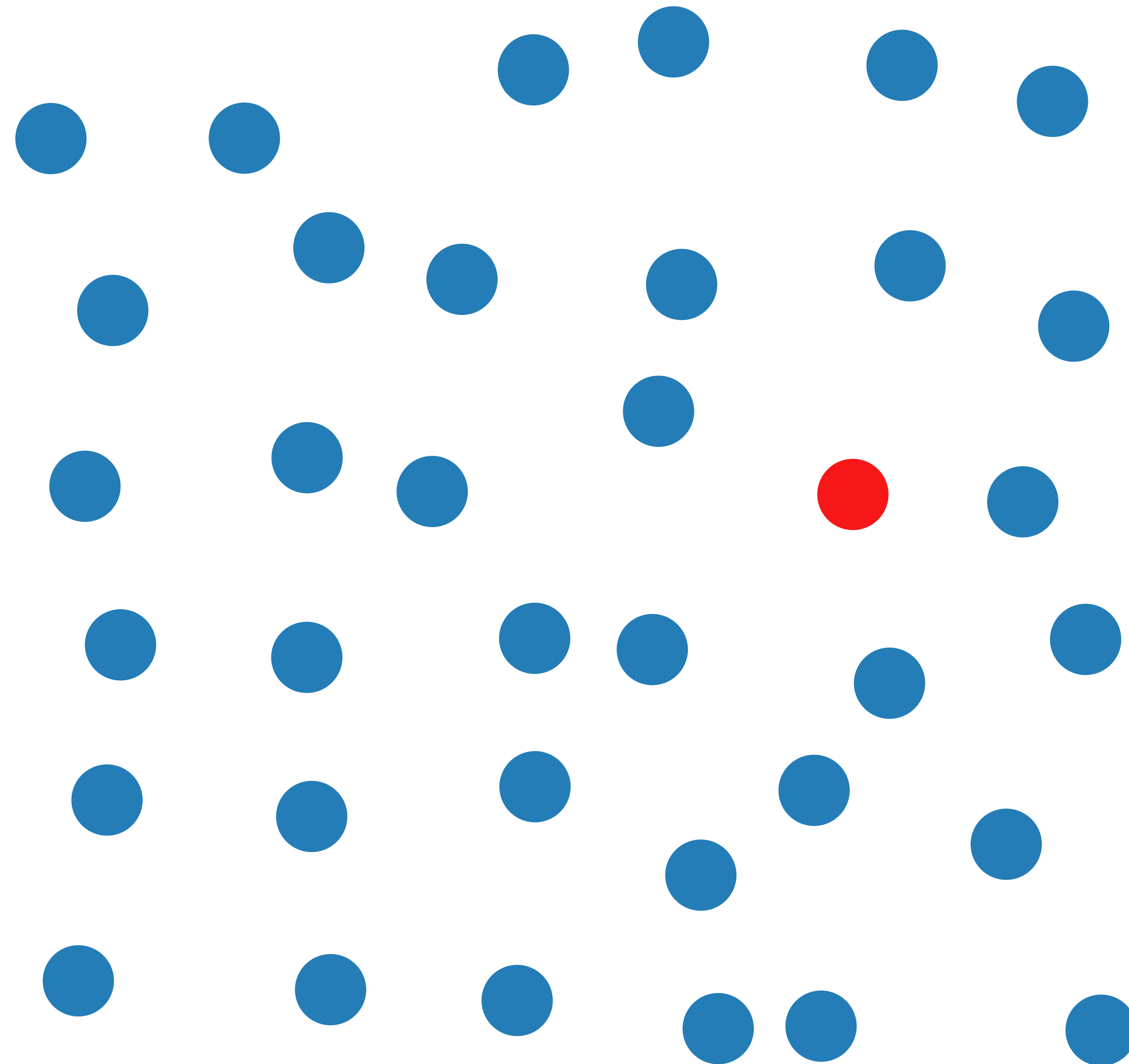
Why Visual?



Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816

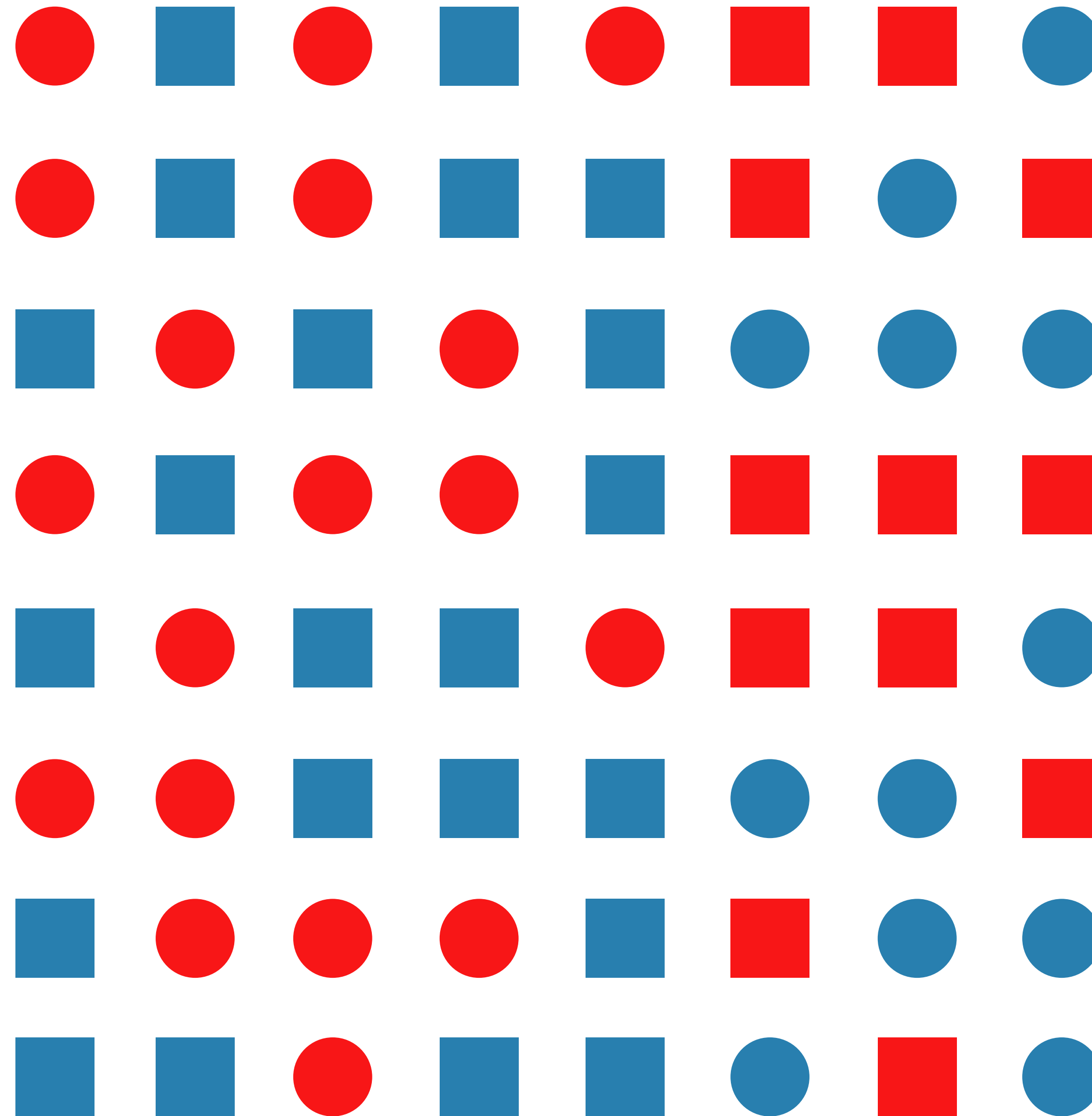
[F. J. Anscombe]

Visual Pop-out



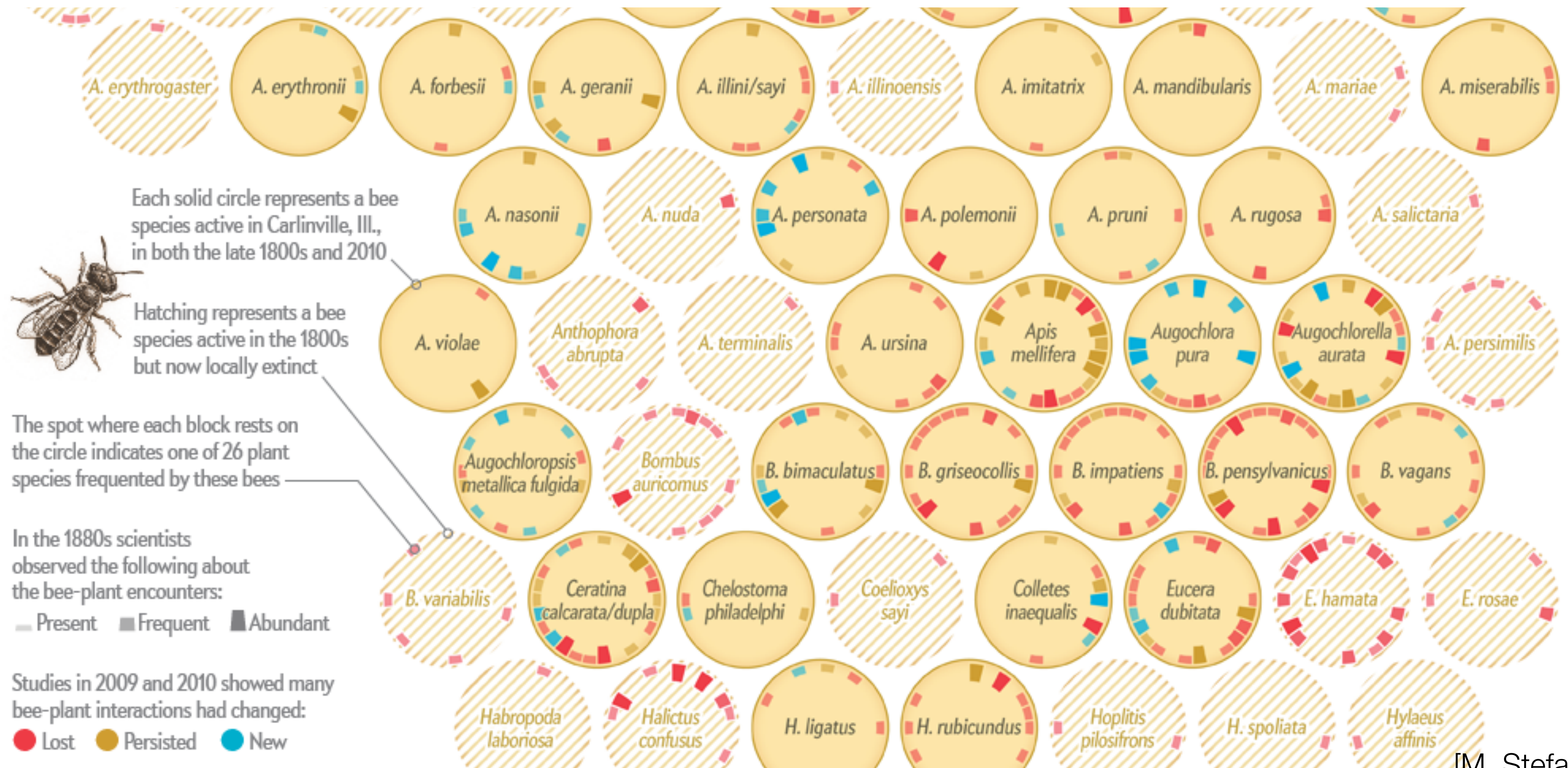
[C. G. Healey]

Visual Perception Limitations



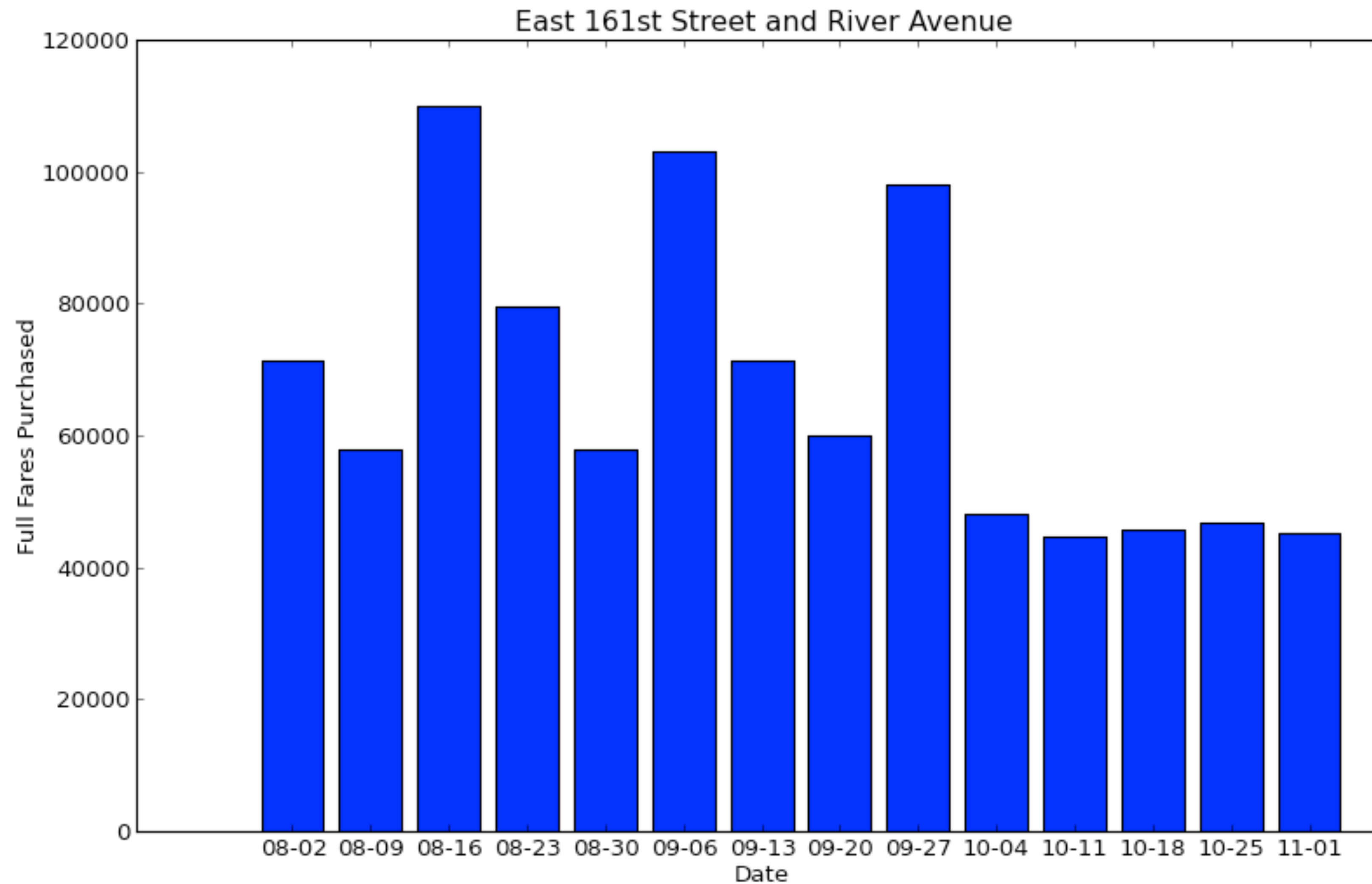
[C. G. Healey]

Visualization Design

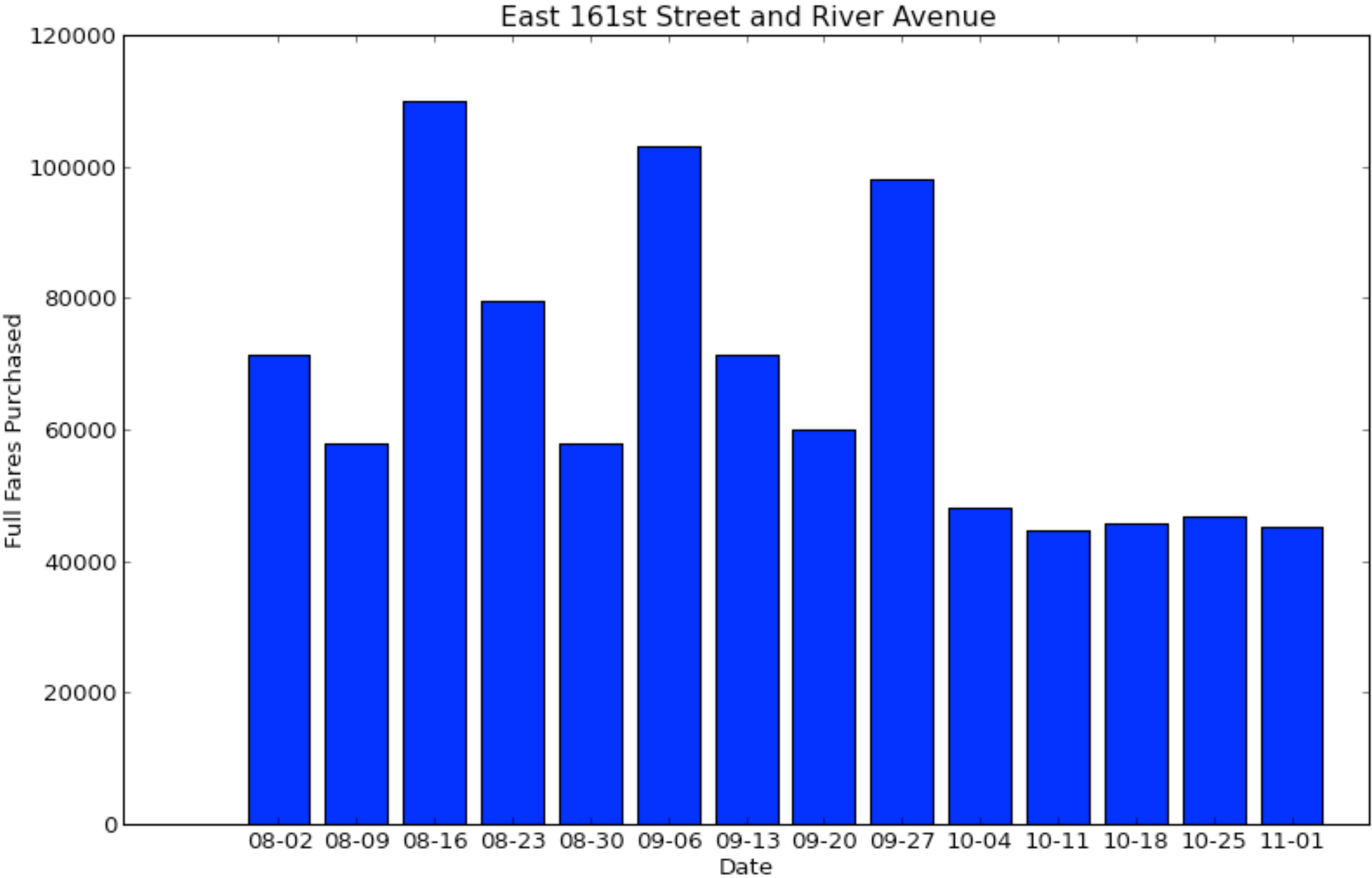


[M. Stefaner, 2013]

Data Exploration



Data Exploration



New York Yankees

AUGUST						
SUN	MON	TUE	WED	THU	FRI	SAT
	yankees.com			1	2	3
					SD	SD
4	5	6	7	8	9	10
SD	CHW	CHW	CHW		DET	DET
11	12	13	14	15	16	17
DET	LAA	LAA	LAA	LAA	BOS	BOS
18	19	20	21	22	23	24
BOS		TOR	TOR	TOR	TB	TB
25	26	27	28	29	30	31
TB	TOR	TOR	TOR		BAL	BAL

SEPTEMBER						
SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
BAL	CHW	CHW	CHW	BOS	BOS	BOS
8	9	10	11	12	13	14
BOS	BAL	BAL	BAL	BAL	BOS	BOS
15	16	17	18	19	20	21
BOS		TOR	TOR	TOR	SF	SF
22	23	24	25	26	27	28
SF		TB	TB	TB	HOU	HOU
29	30	ALL GAMES ARE EASTERN TIME.				
HOU						

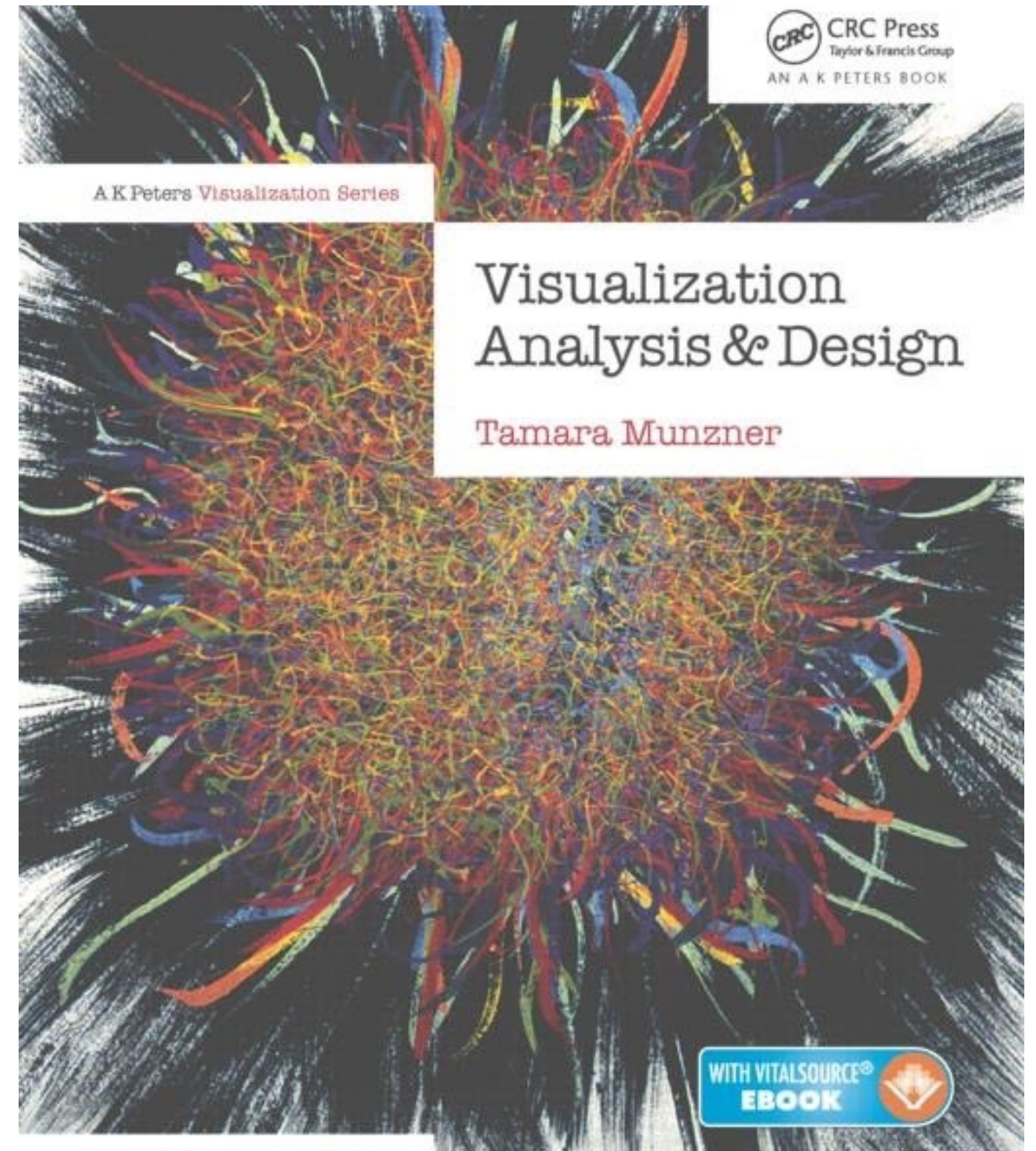
• 2013 REGULAR SEASON SCHEDULE •

Languages of the Web

- HTML
- CSS
- SVG
- JavaScript
 - ECMAScript is a JavaScript standard
 - Versions of Javascript: ES6 (ECMAScript 2015), ECMAScript 2024...
 - Specific frameworks: react, jQuery, bootstrap, Plot, D3

Administrivia

- Course Web Site
- Syllabus
 - Plagiarism
 - Accommodations
- Textbook:
 - Required: Munzner (VAD)
- Assignments
- Exams: Midterm (Mar. 4) and Final (May 4)

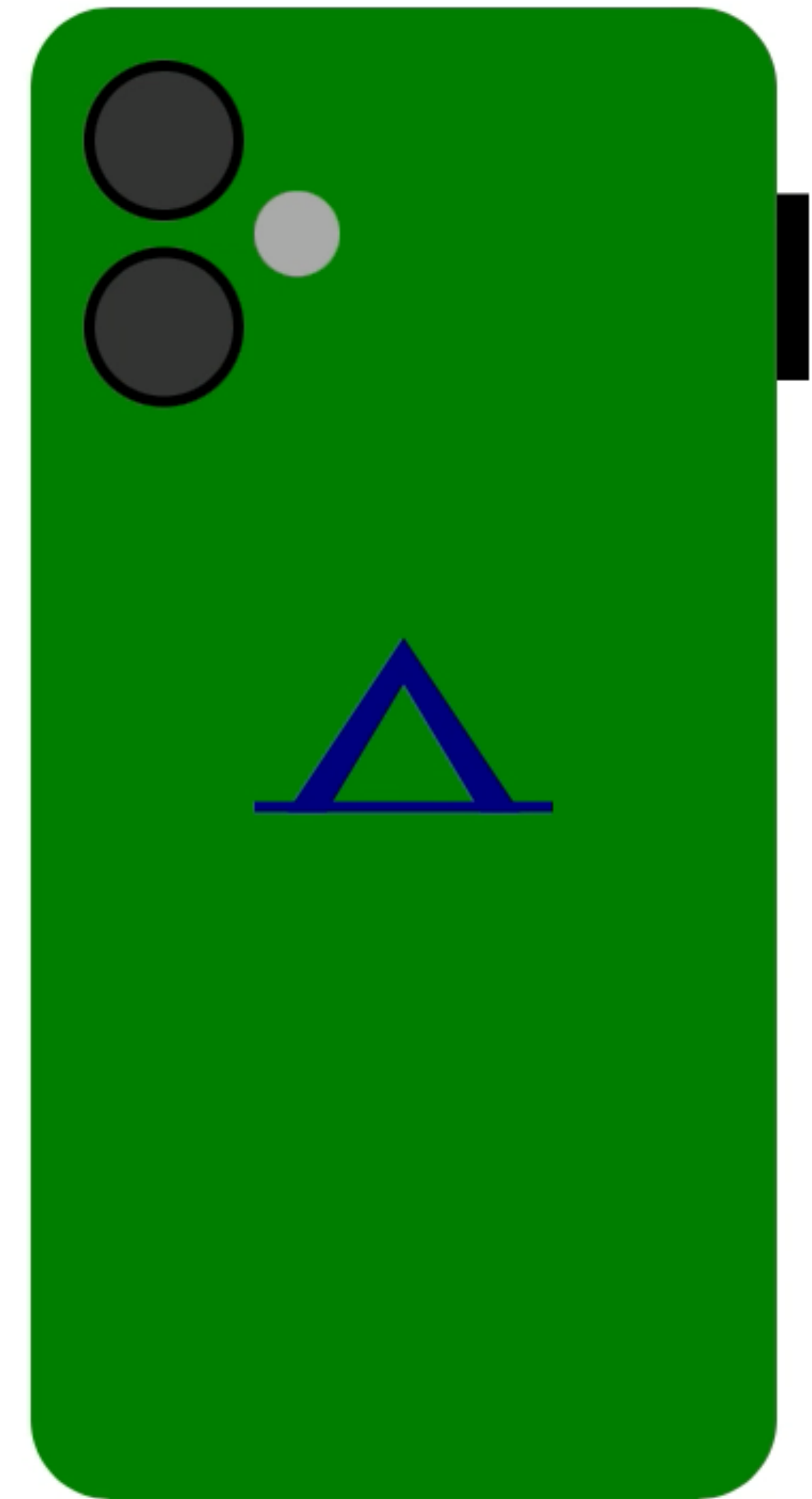


Observable

- Should have received an invitation to join NIU ReDAV team
 - Class examples will be available through that team
- Observable is a platform that allows users to create notebooks using JavaScript, Markdown, and HTML
- Introduction: Learning Observable
- Type markup/code, "execute" the cell, and results appear **above** the code
- Pin the cell to keep the code visible
- Can choose the type of cell (JavaScript, Markdown, or HTML)
- Can create an output (variable) in each cell that can be used in other cells
- Content is all **global scope!**

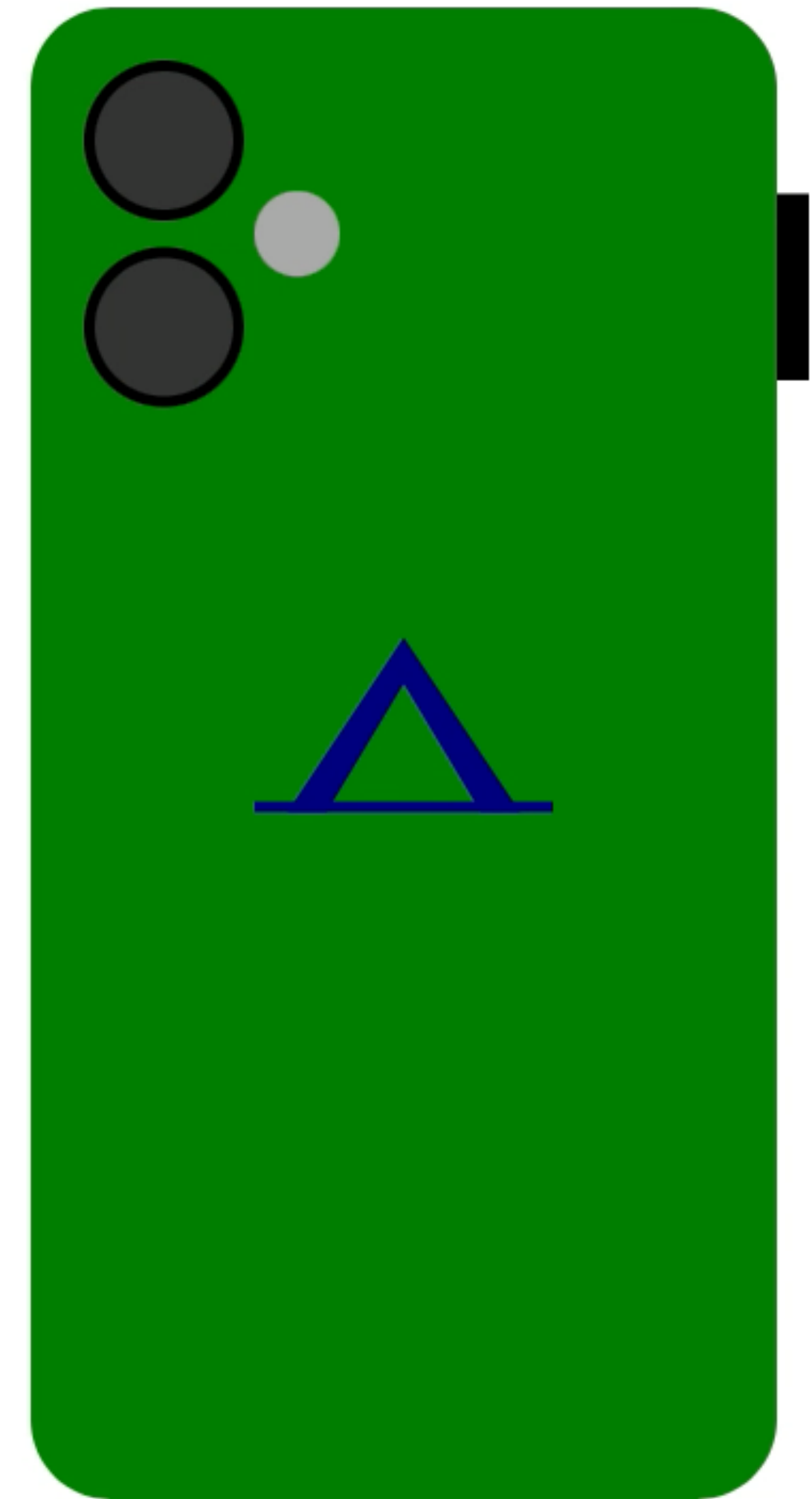
Assignment 1

- Write HTML, CSS, and SVG
- Text markup and styling (information)
- Drawing markup and styling (camera)
- Draw Bar chart using Plot library
- Due Wednesday, Jan. 28



Assignment 1

- Write HTML, CSS, and SVG
- Text markup and styling (information)
- Drawing markup and styling (camera)
- Draw Bar chart using Plot library
- Due Wednesday, Jan. 28



Hyper Text Markup Language (HTML)

- Markup languages allow users to encode the **semantics** of text
- Tags define the boundaries of the structures of the content
 - Tags are enclosed in angle brackets (e.g. `<html>`)
 - Most of the time, you have a start and end tag
 - End tags are just like start tags except that they have forward slash after the open bracket (e.g. `</html>`)
 - Tags may be nested but not mismatched
 - `<p>A <strong><em>very</em></strong> cool example</p>`
 - `<p>A <strong>very <em>cool</strong></em> example</p>`
 - What about ``?

HTML Elements and Attributes

- Tags denote **elements** of the content (e.g. sections, paragraphs, images)
- Each element may have **attributes** which define other information
 - An attribute has a **key** and **value** (`key="value"`)
 - e.g. ``
- Many different elements available
 - Common: headers (`h1`, ..., `h6`), paragraph (`p`), lists (`ul`, `ol`, `li`), emphasis (`em`, `strong`), link (`a`), spans & divisions (`span`, `div`)
 - Lots more (e.g. `abbr`): see [MDN Documentation](#)
- Many different attributes available
 - See [MDN Documentation](#): note that some are legacy due to CSS

HTML Element Structure & Naming

- Elements structure a document
 - Document Object Model (DOM)
 - We can visualize this information
 - More importantly, we can **navigate** this tree
- Identifying and Classifying elements: `id` and `class` attributes
 - `id` identifies a **single** element—use for a unique case
 - `class` may identify **multiple** elements—use for common cases
 - Each element may have multiple classes, separate by spaces
 - Use normal identifiers: don't start the name with a number

Other HTML Trivia

- `<`, `>`, `&`, and `"` are special characters, escape with `<`, `>`, `&`, and `"`; (note the semi-colon)
- Comments are enclosed by `<!--` and `-->`
 - `<!-- This is a comment -->`
- HTML Documents begin with a DOCTYPE declaration
 - For HTML5, this is easier `<!DOCTYPE html>`
- meta tag: `<meta charset="UTF-8"/>`
- HTML has audio and video tags, math equation support, and more

Basic HTML File

```
<!DOCTYPE html>
<html>
  <head>
    <title>A Basic Web Page</title>
  </head>
  <body>
    <h1>Da Bears</h1>
    <p><em>This is <strong>cool</strong>. What about
<u><strong>this?</strong></u></em></p>
    
  </body>
</html>
```

Basic HTML File

```
<!DOCTYPE html>
<html>
  <head>
    <title>A Basic Web Page</title>
  </head>
  <body>
    <h1>Da Bears</h1>
    <p><em>This is <strong>cool</strong>. What about
<u><strong>this?</strong></u></em></p>
    
  </body>
</html>
```

In many environments (e.g. Observable), only need this part

What is CSS?

Cascading Style Sheets (CSS)

- Separate from content, just specifies how to style the content
- Style information can appear in three places:
 - External file
 - In a style element at the beginning of the HTML file
 - In a specific element in the body of a document (least preferable)
- Why Cascading?
 - Don't want to have to specify everything over and over
 - Often want to use the same characteristics in a region of the DOM
 - Use inheritance: properties that apply to children cascade down

CSS Selectors

- How do we specify what part(s) of the page we want to style?
- The **element types** themselves (the HTML tag)
 - `strong { color: red; }`
- **Classes** of elements (ties to HTML `class` attribute)
 - `.cool { color: blue; }`
- A **specific** element (ties to HTML `id` attribute)
 - `#main-section { color: green; }`
- Relationships
 - Descendant: `p em { color: yellow; }`
 - Child: `p > em { color: orange; }`
- Pseudo-classes: `a:hover { color: purple; }`

Other CSS Bits

- Comments: `/* This is a comment in CSS */`
- Grouping Selectors: `p, li { font-size: 12pt; }`
- Multiple Classes: `.cool.temp { color: blue; }`
- Colors:
 - Names (Level 1, 2, & 3): `red, orange, antiquewhite`
 - Dash notation (3- & 6-character): `#fff, #00ff00`
 - Integer or % RGB and HSL Functions: `rgb( 255, 0, 0),`
`rgb(50%, 50%, 0%), hsl(120, 100%, 50%)`
 - Also `background-color`
- Watch out for multiple rules (look at how a web browser parses)
- Again, much more documentation at [MDN](#)

Example CSS

```
body {  
    font-face: sans-serif;  
    font-size: 12pt;  
}  
em { color: green; }  
em u { color: red; }  
em > strong { color: blue; }  
img { border: 4px solid red; }
```

- What colors are displayed for this HTML (with the above stylesheet)?
 - `This is cool. What about
<u>this?</u>`

CSS Specificity

- Example:
 - CSS:

```
p.exciting { color: red; }  
p { color: blue; }
```
 - What is the color of the paragraph
`<p class="exciting">Cool</p>?`
- Generally, last rule listed overrides previous rules
- ...but anytime a selector is **more specific**, it has precedence
- `p.exciting` is a more specific selector
- When in doubt, **test it** in a browser

How to add CSS to HTML

- External: a separate file via a link element (in the `<head>` section):
 - `<link rel="stylesheet" href="styles.css">`
- Embedded: in the header:
 - `<style type="text/css"> ... </style>`
- Inline: for a specific element: (**Discouraged!**)
 - `<p style="font-weight: bold;">Some text</p>`

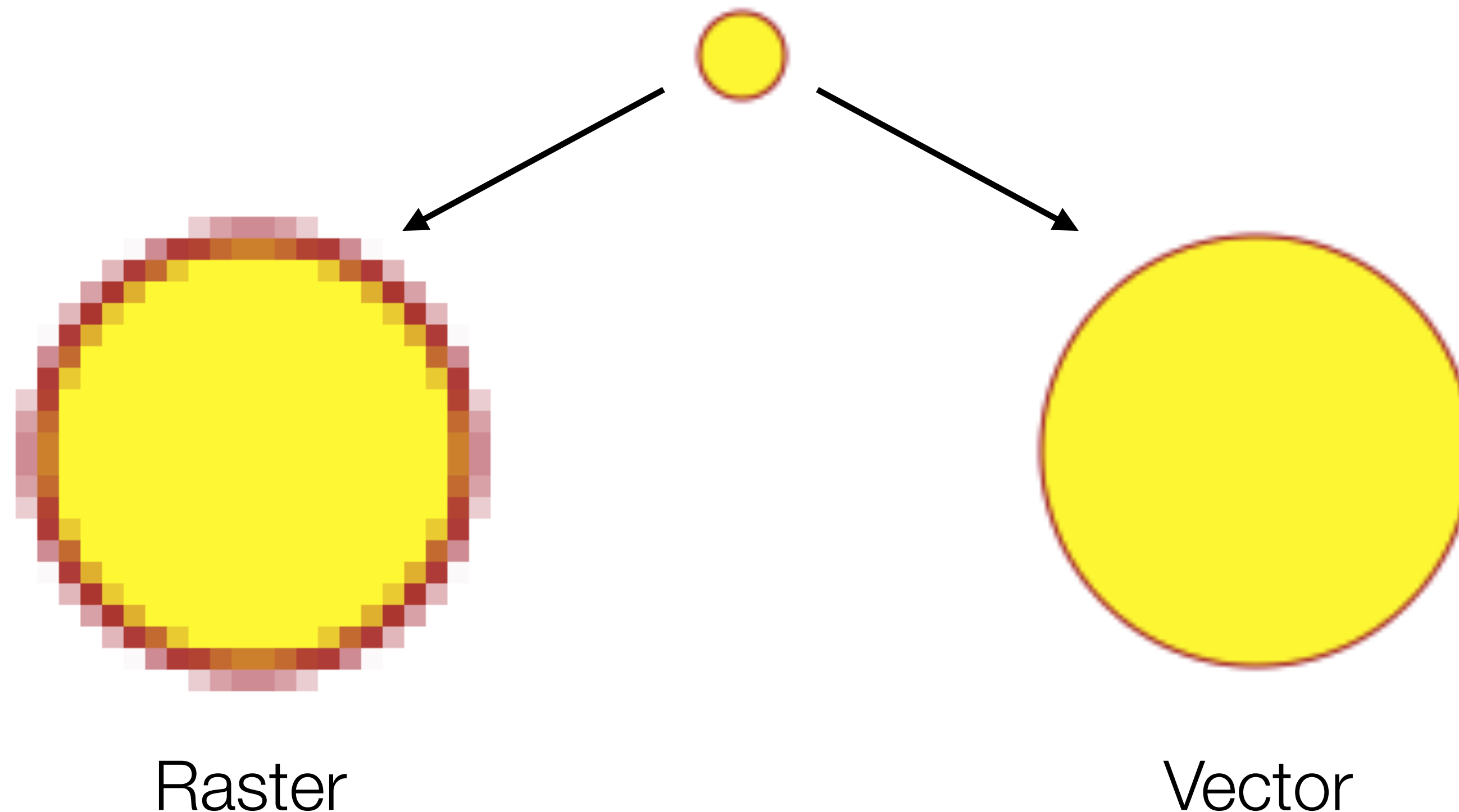
Observable

- Style rules are **global**
- Make sure you scope the rules
 - Add a `<style>` block to the cell for the rules for that cell
 - Put everything in a `<div>` with an `id` attribute
 - `<div id="mydiv">...</div>`
 - CSS rules should start with the div id unless you want to have rules apply to the entire notebook
 - `#mydiv p { font-family: sans-serif };`

What is the difference between
vector and raster graphics?

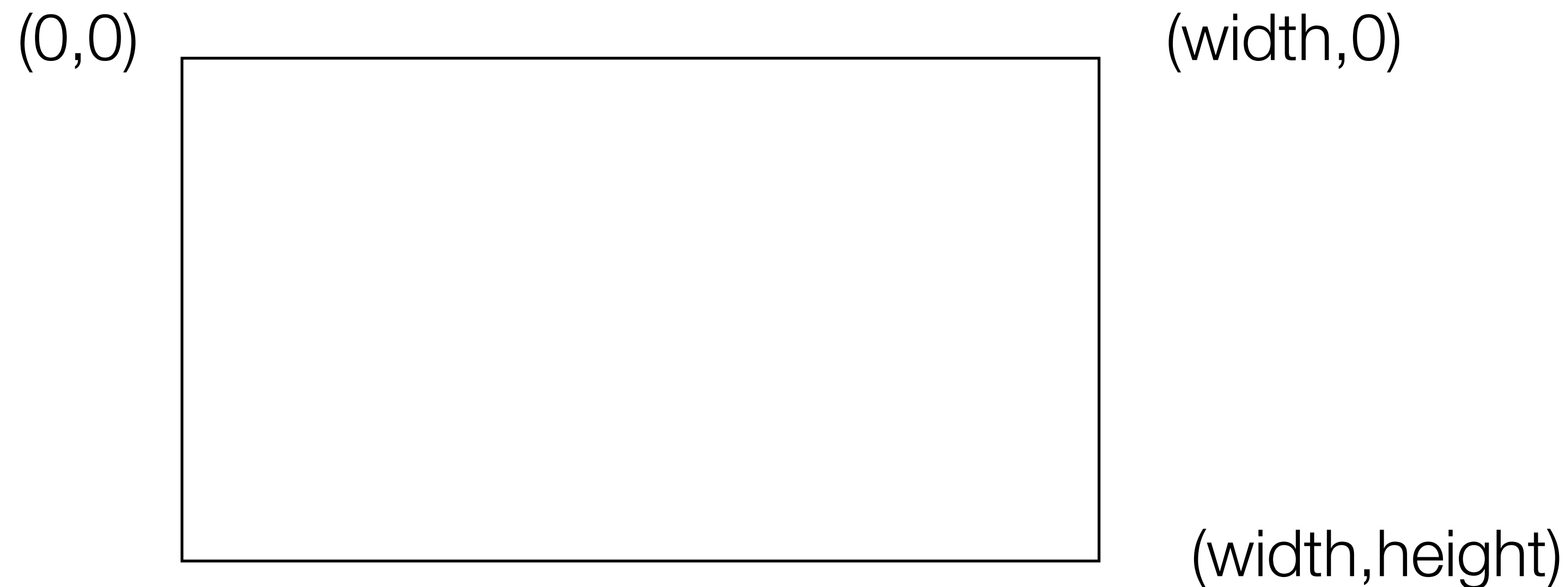
Scalable Vector Graphics (SVG)

- Vector graphics vs. Raster graphics
- Drawing commands versus a grid of pixels
- Why vector graphics?



SVG Background

- Another markup language:
 - Describe the shapes and paths by their endpoints, characteristics
- SVG can be embedded into HTML5 documents!
- Pixel Coordinates: **Top-left** origin



SVG Elements

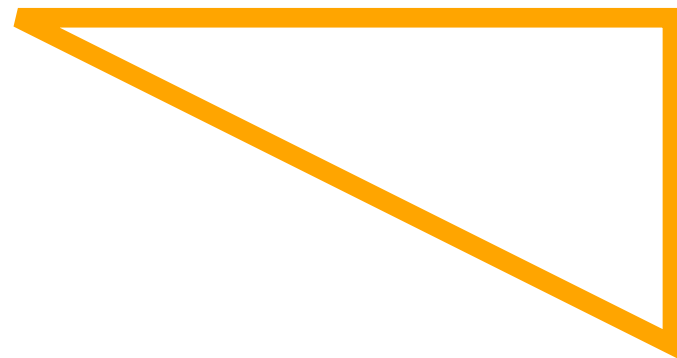
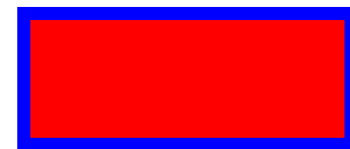
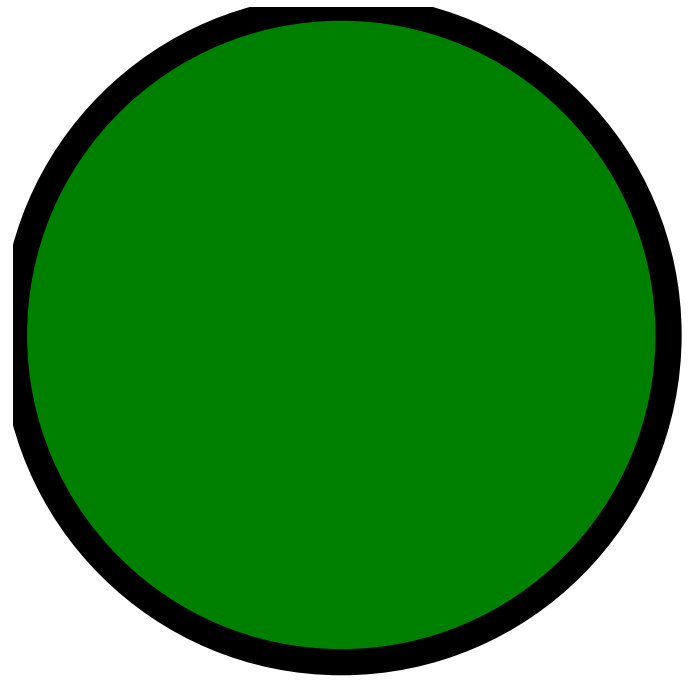
- Drawing primitives:
 - Lines, Circles, Rects, Ellipses, Text, Polylines, Paths
 - Work by specifying information about **how** to draw the shape
 - Lots more: see [MDN Documentation](#)
- Ordering/Stacking:
 - SVG Elements are drawn in the order they are specified
- Paths: directions for drawing
 - <https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths>

SVG Styles

- We can specify styles of SVG elements in CSS!
- Example:

```
circle { fill: green; stroke: black;  
        stroke-width: 4px; }  
.normal { fill: red; stroke: blue;  
         stroke-width: 2px; }  
#p1 { fill: none; stroke: red; stroke-width: 3px; }
```

SVG Example



```
<svg id="mysvg" width="300" height="600">  
  <circle cx="50" cy="50" r="50"/>  
  <rect class="lego" x="150" y="150"  
    width="50" height="20"/>  
  <path id="triangle" d="M 20 200  
    L 120 200 L 120 250 Z"/>
```

```
</svg>
```

```
circle { fill: green; stroke: black;  
  stroke-width: 4px; }  
.lego { fill: red; stroke: blue;  
  stroke-width: 2px; }  
#triangle { fill: none; stroke: orange;  
  stroke-width: 3px; }
```