

Data Visualization (CIS 490/680)

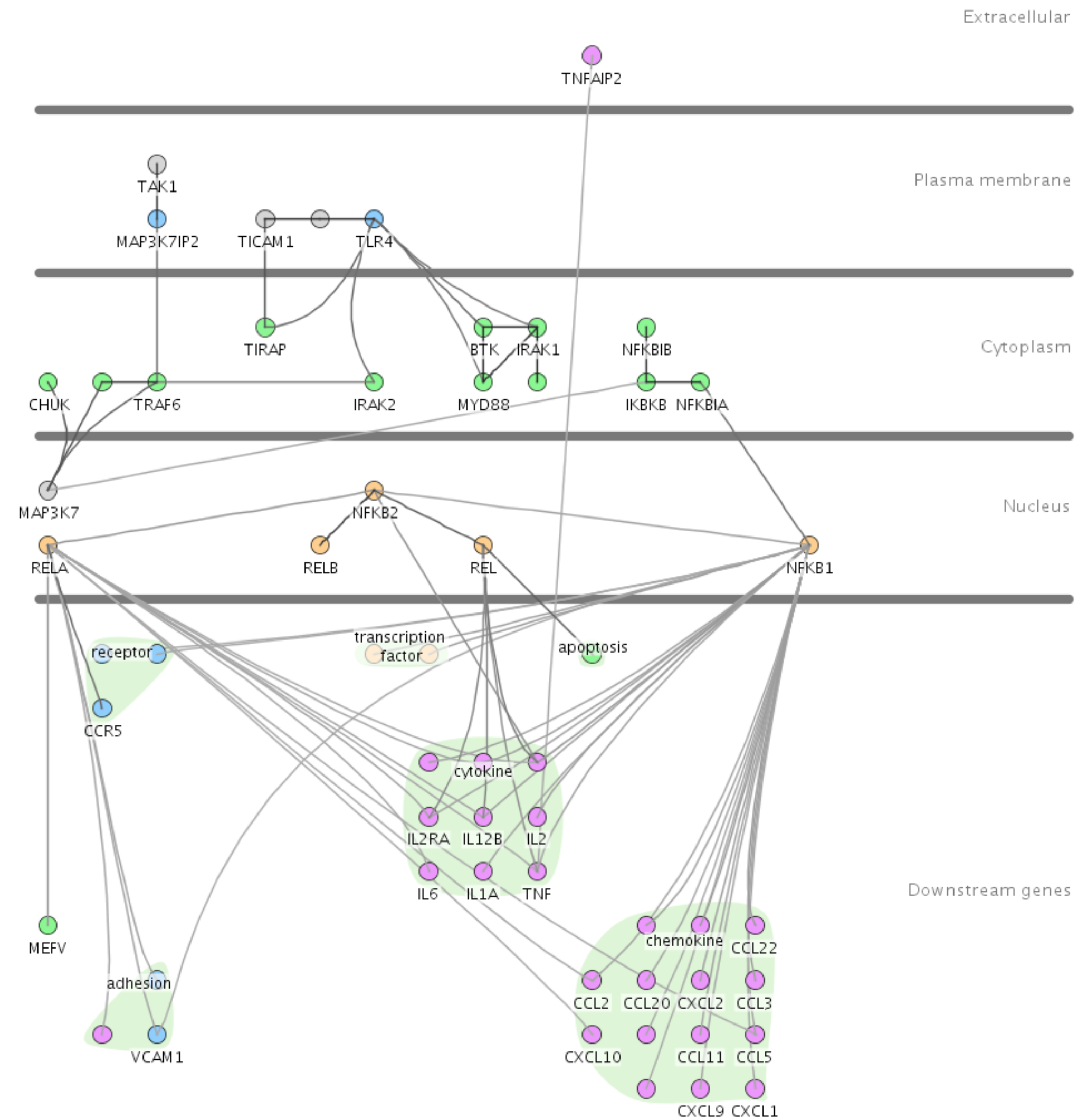
Web Programming

Dr. David Koop

Definition

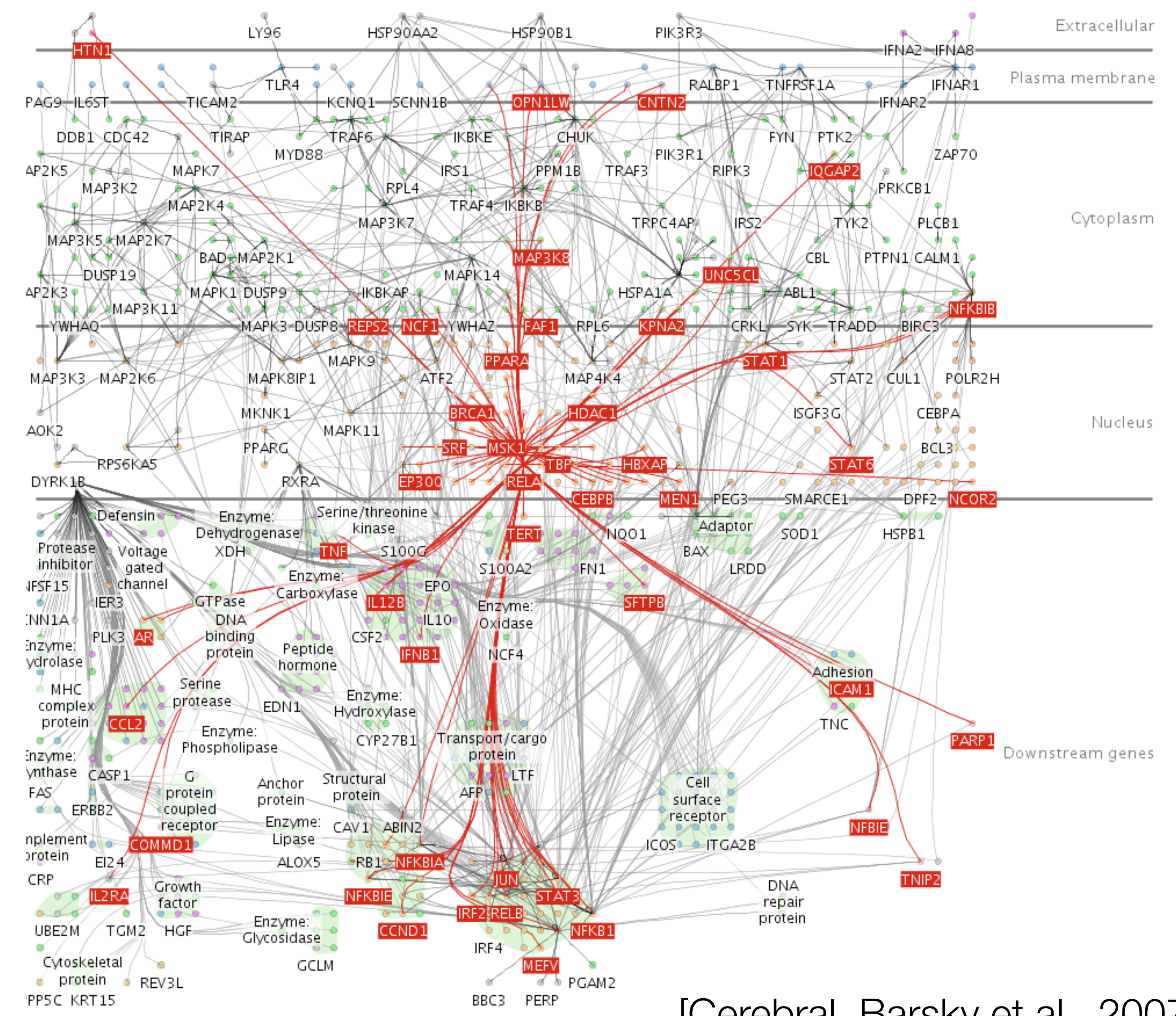
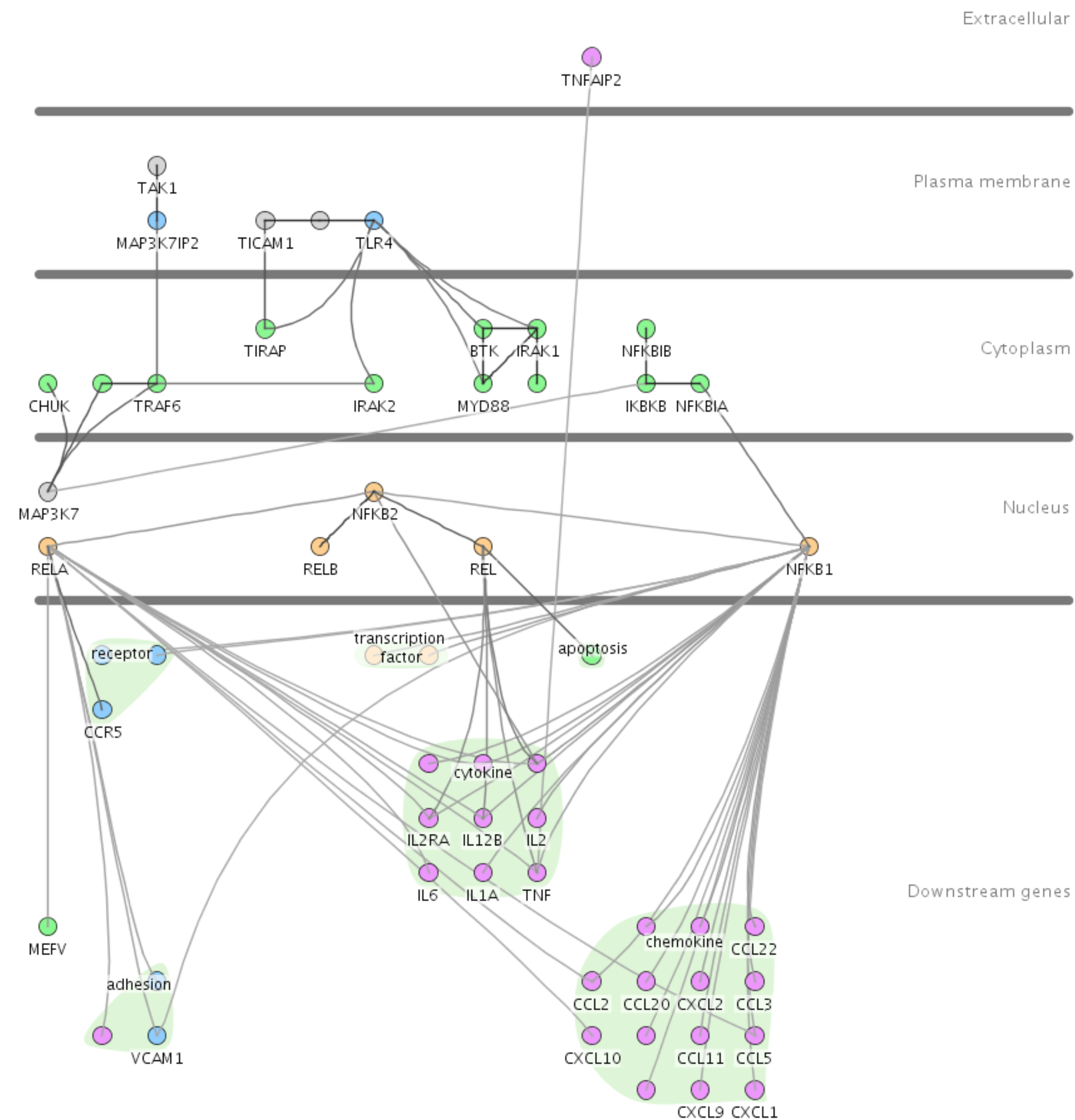
“Computer-based visualization systems provide **visual** representations of datasets designed to help people carry out tasks more effectively”

Why Computers?



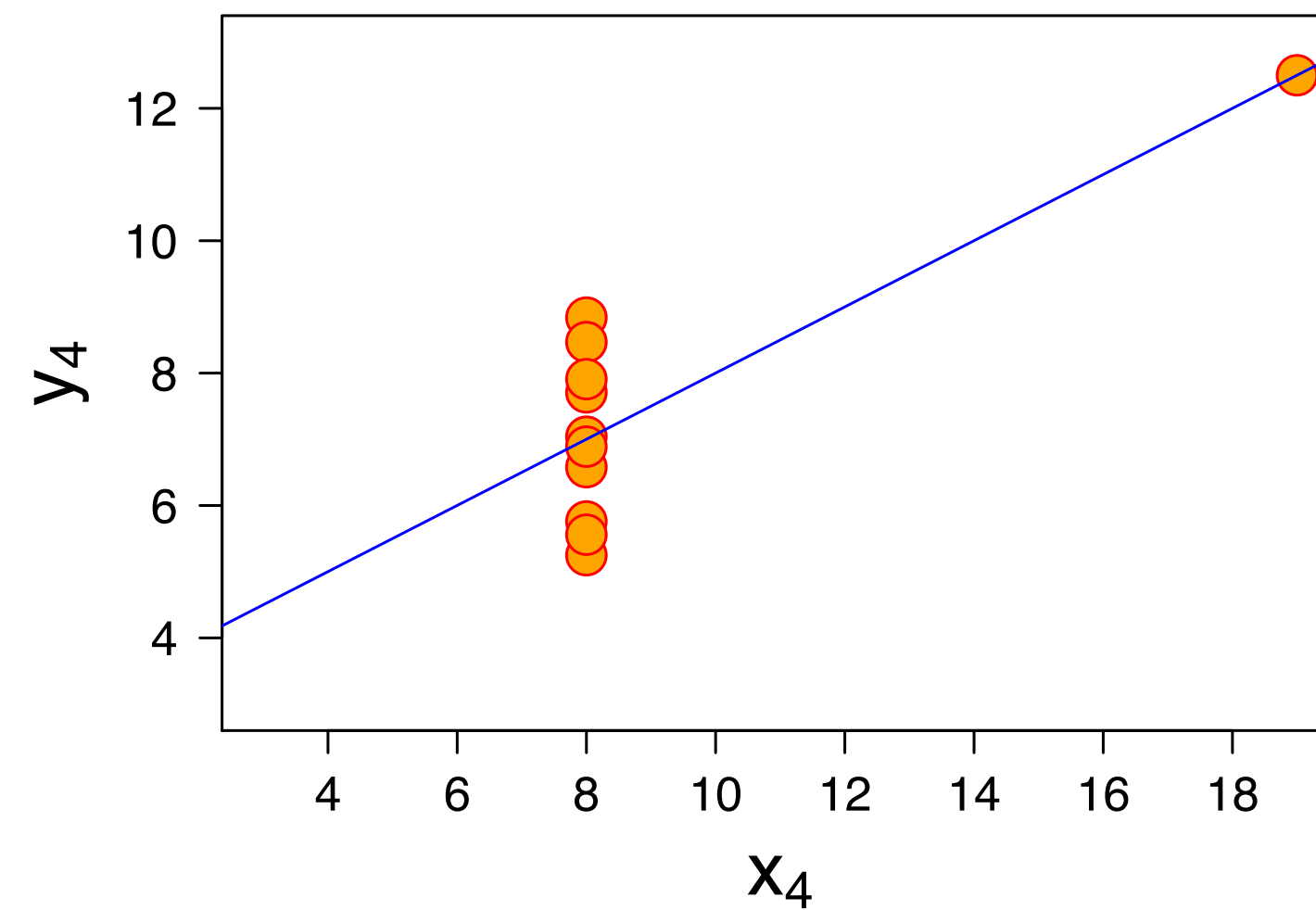
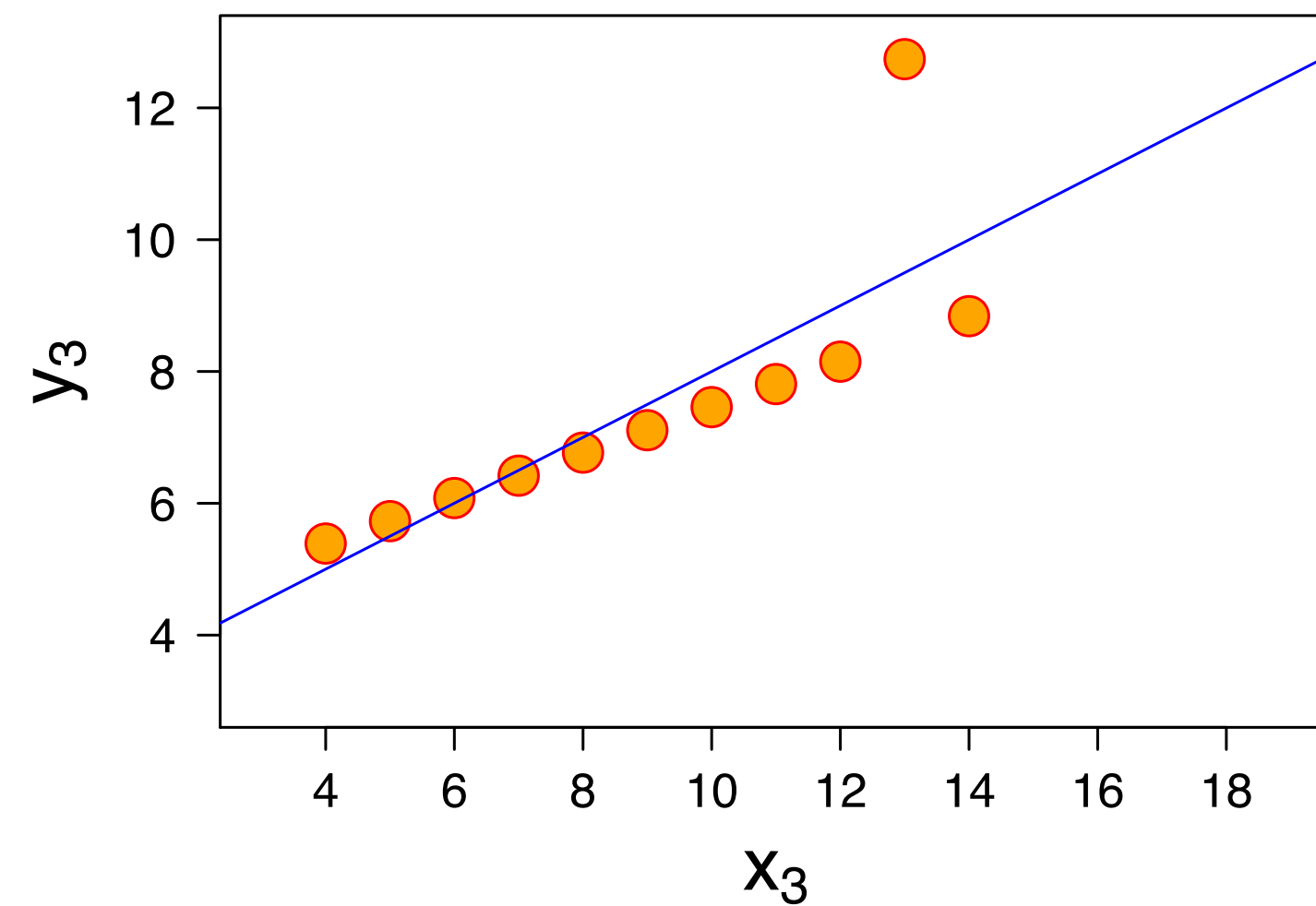
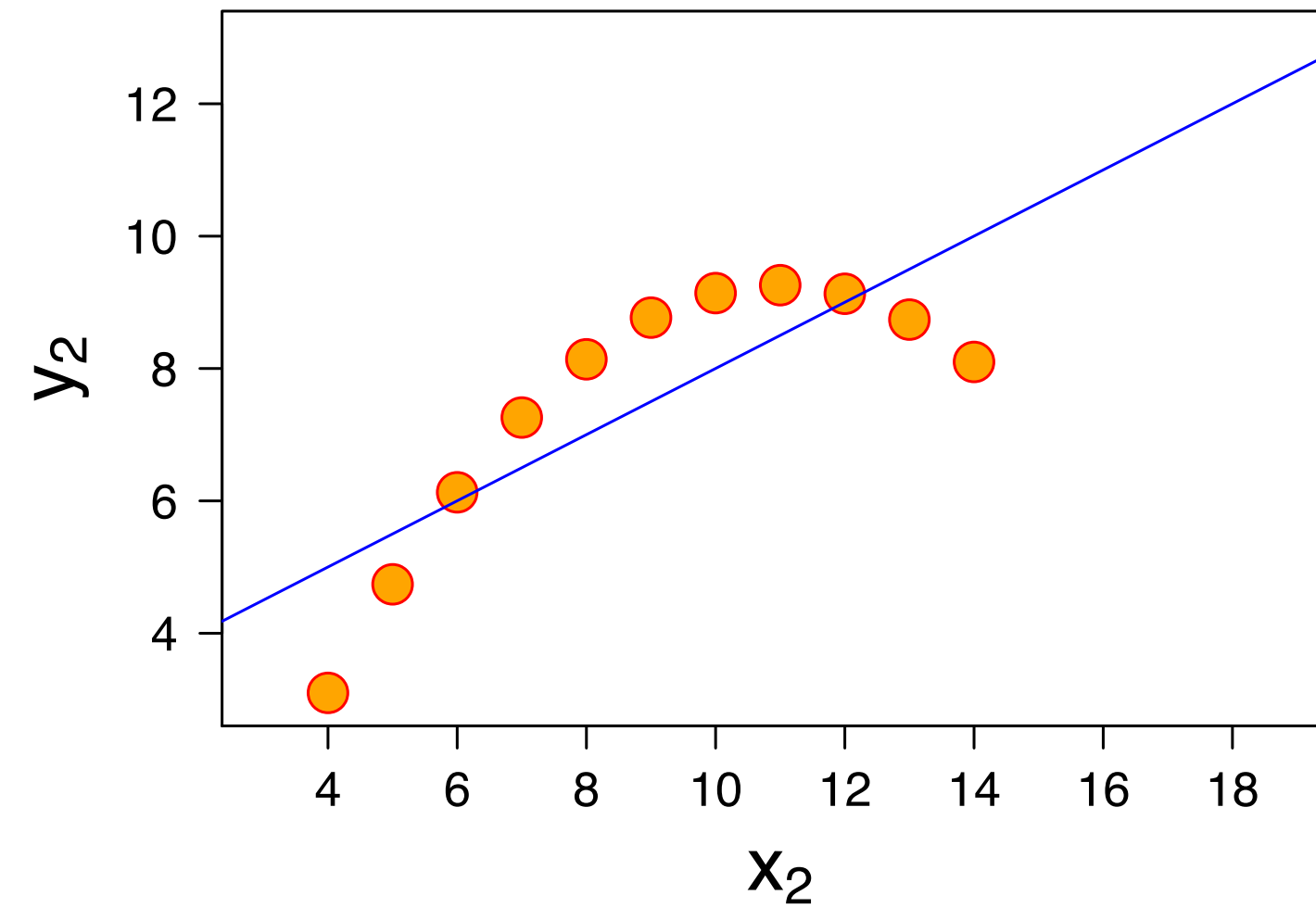
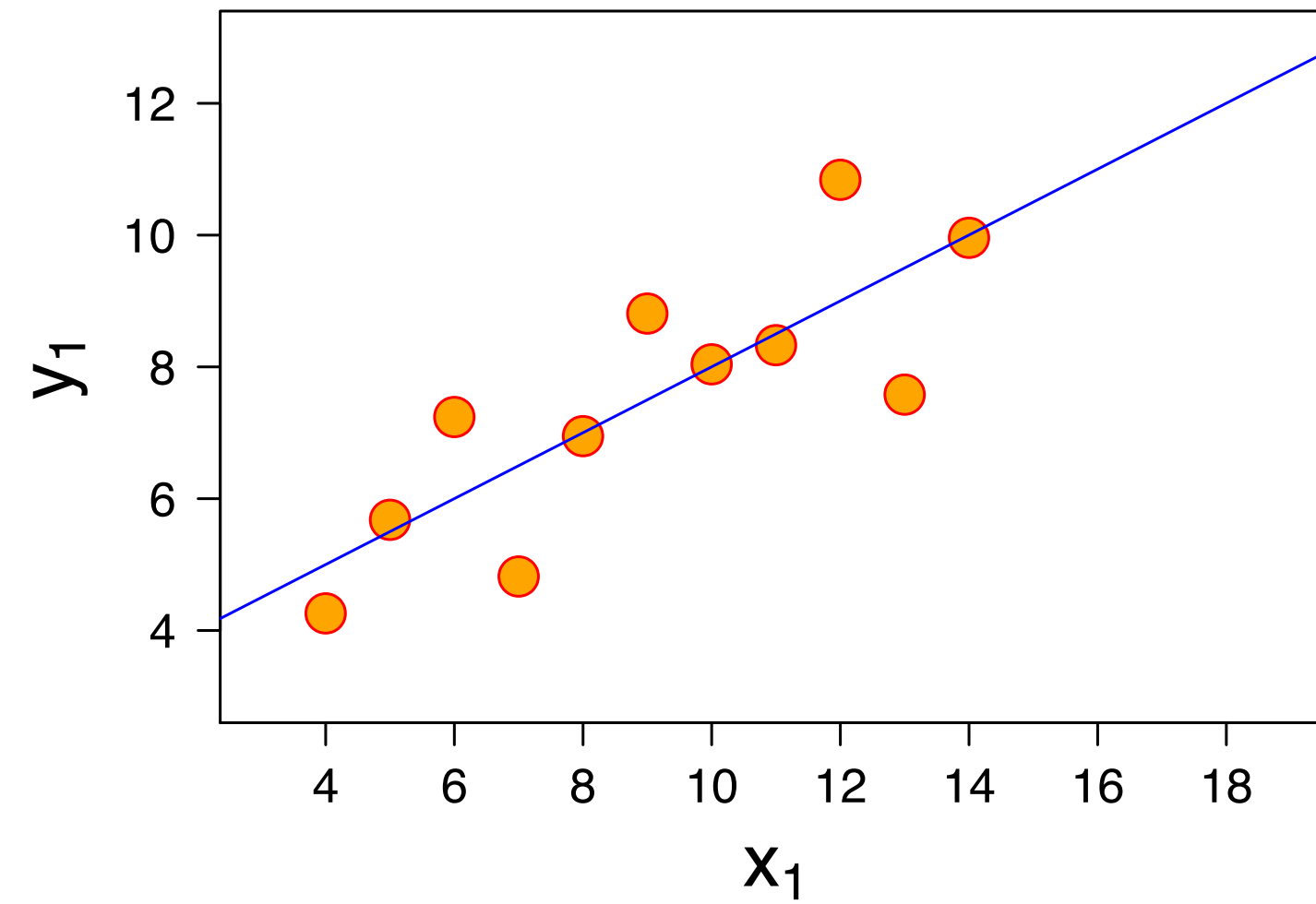
[Cerebral, Barsky et al., 2007]

Why Computers?



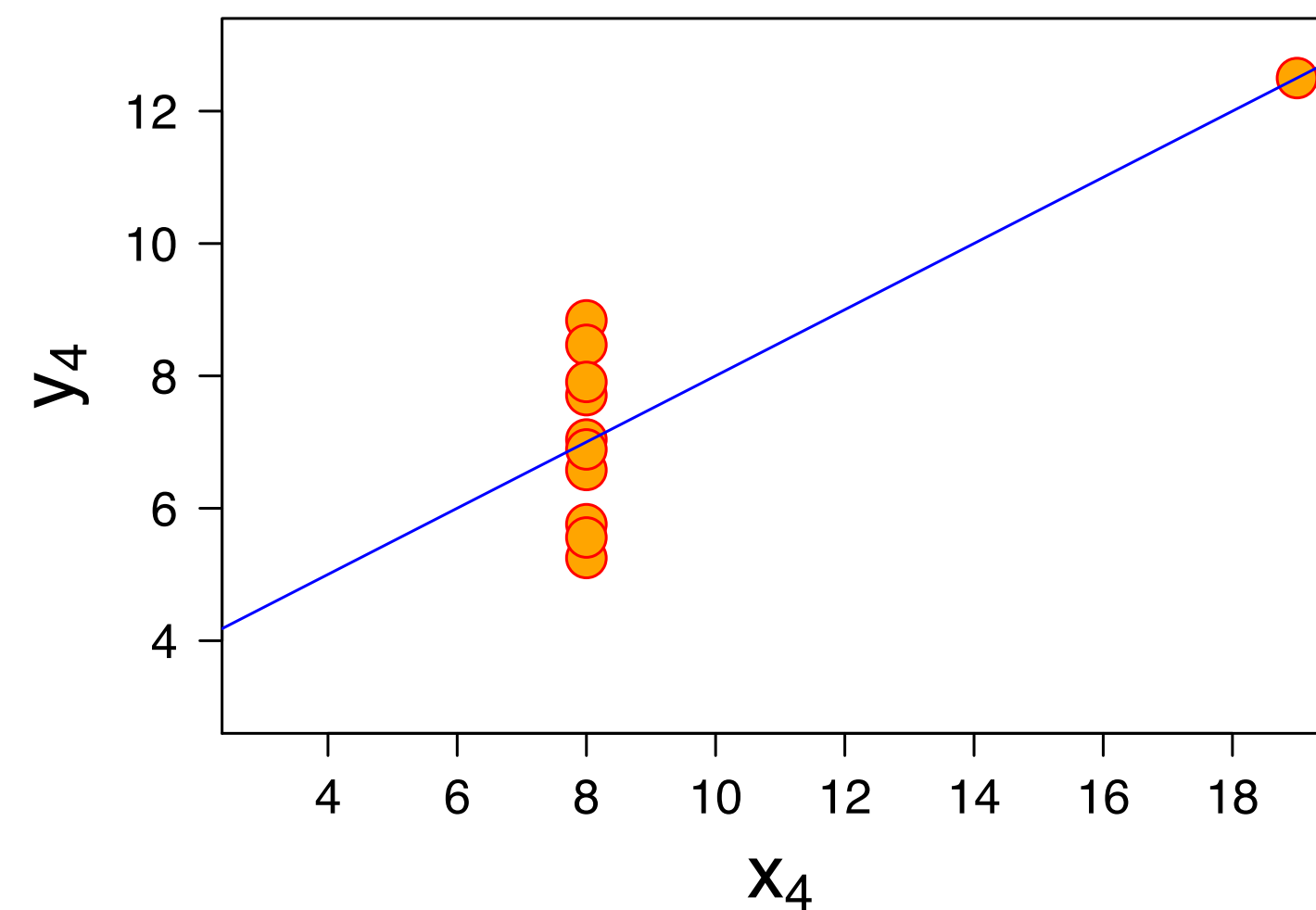
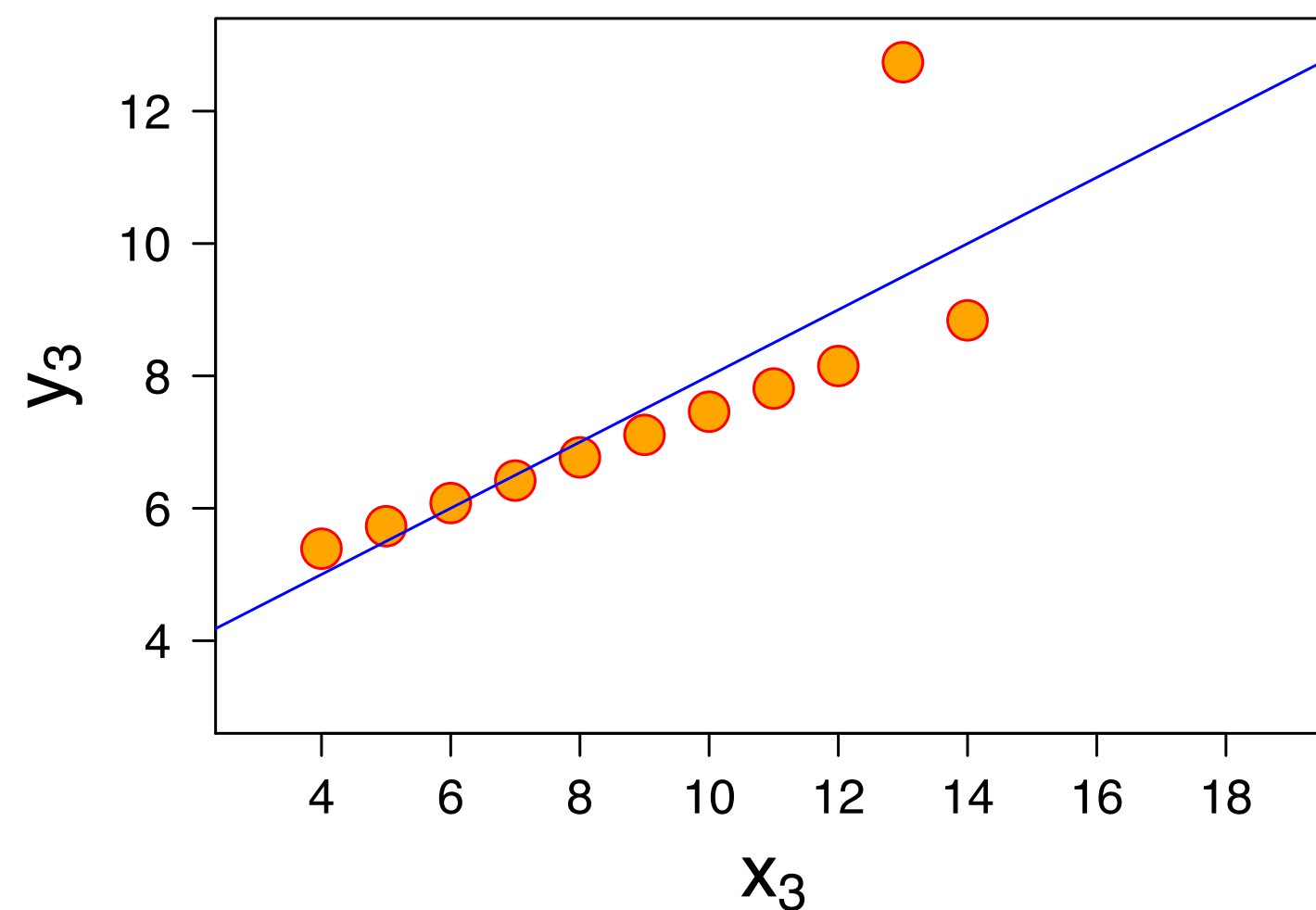
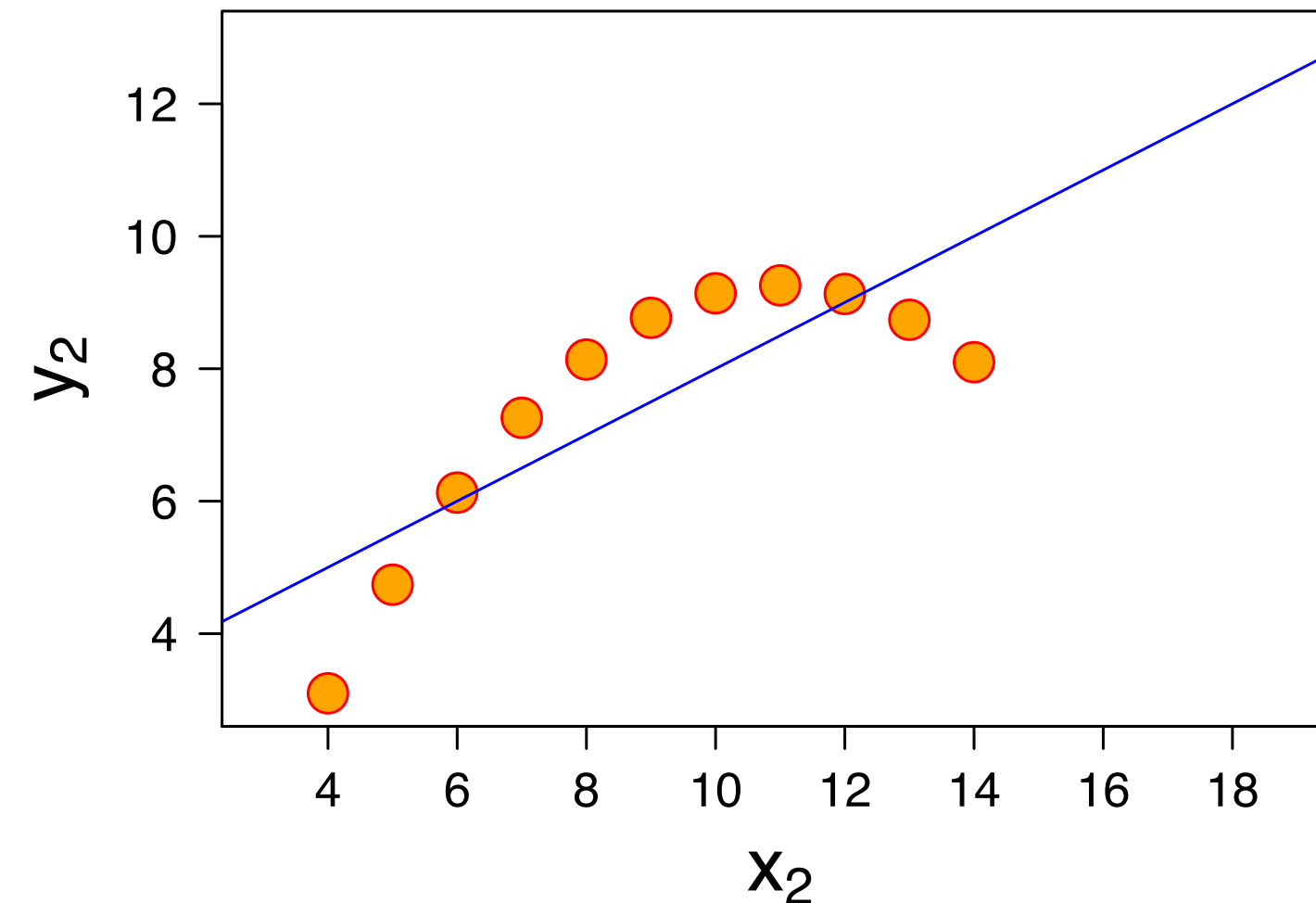
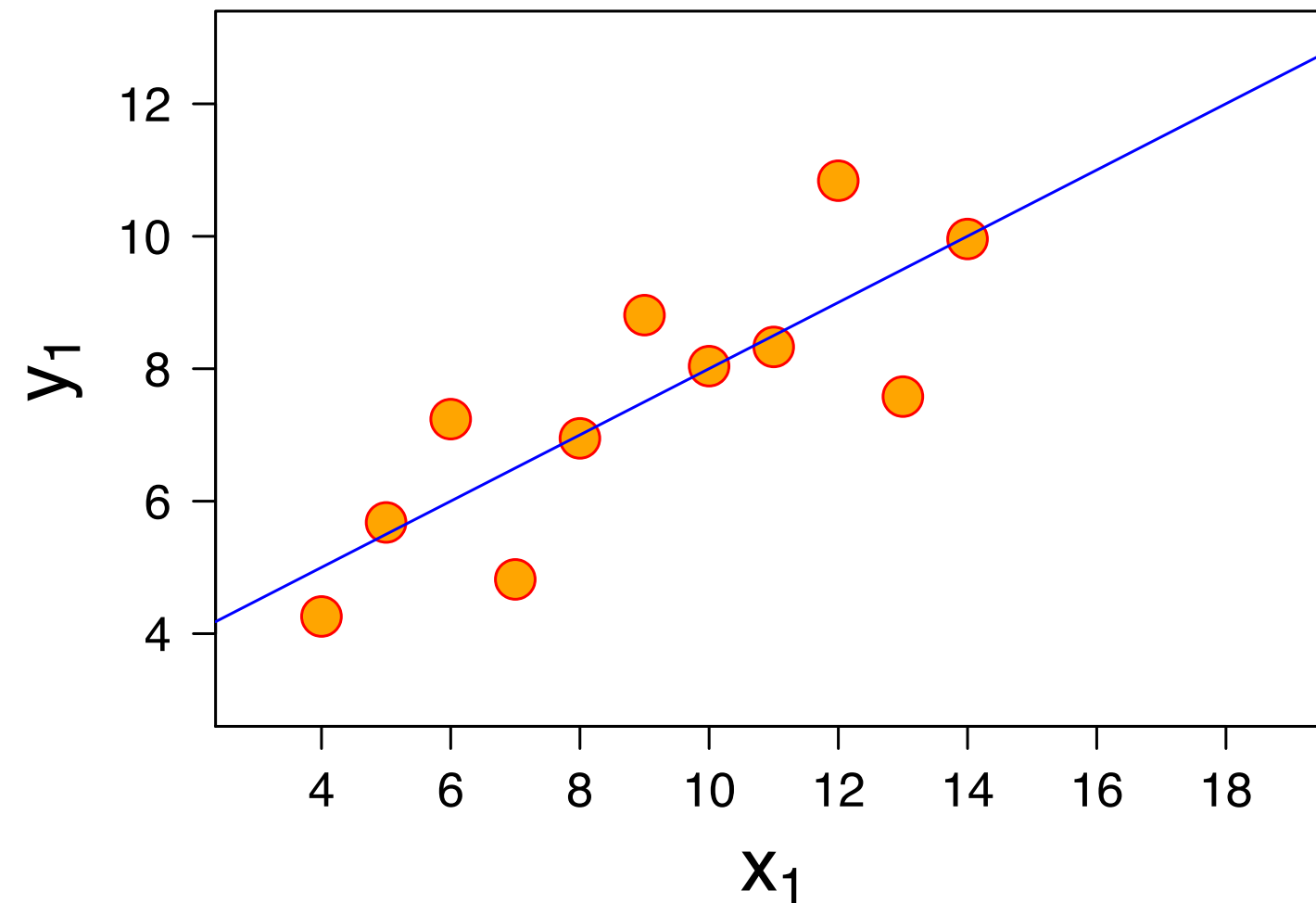
[Cerebral, Barsky et al., 2007]

Why Visual?



[F. J. Anscombe]

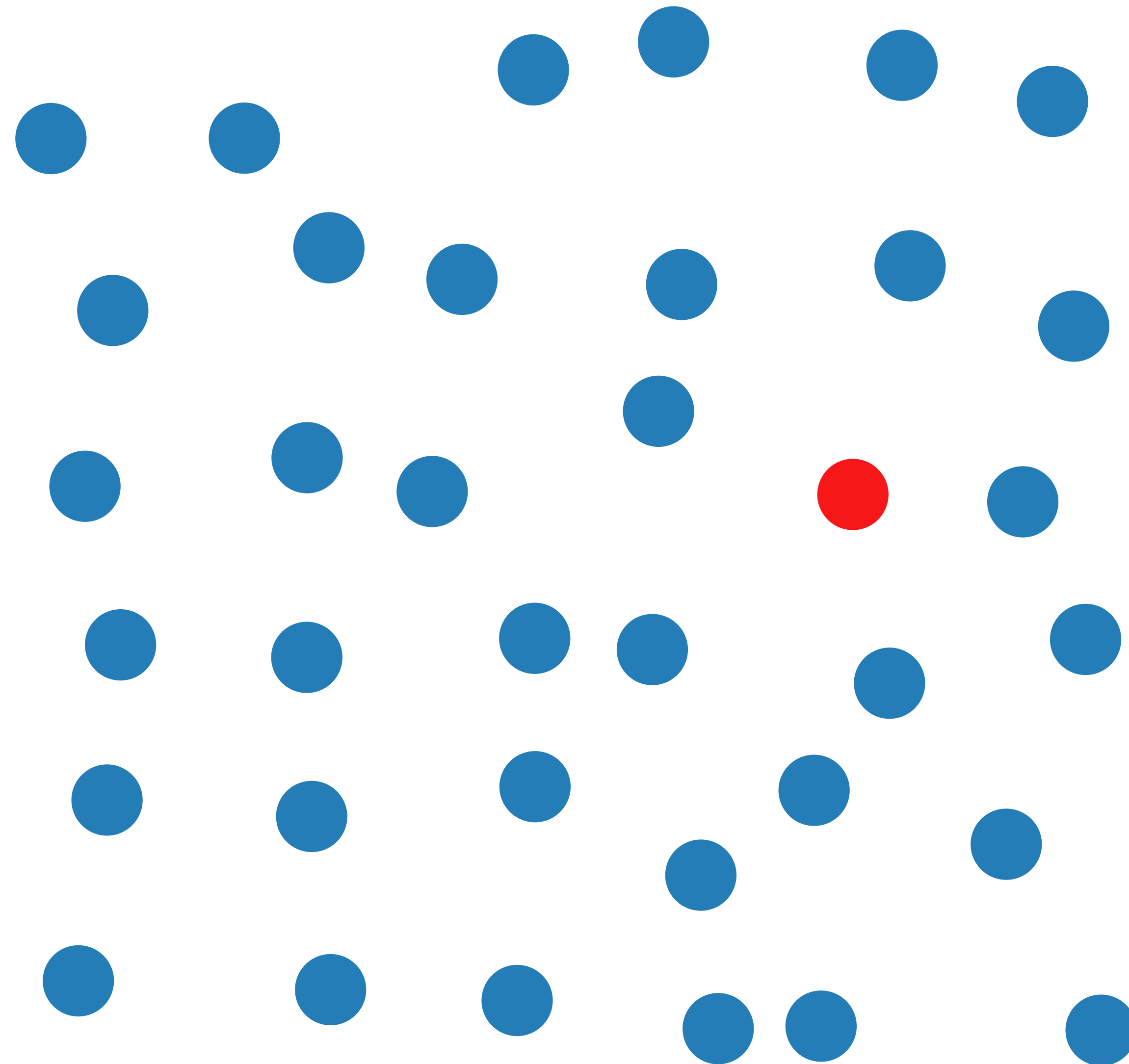
Why Visual?



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

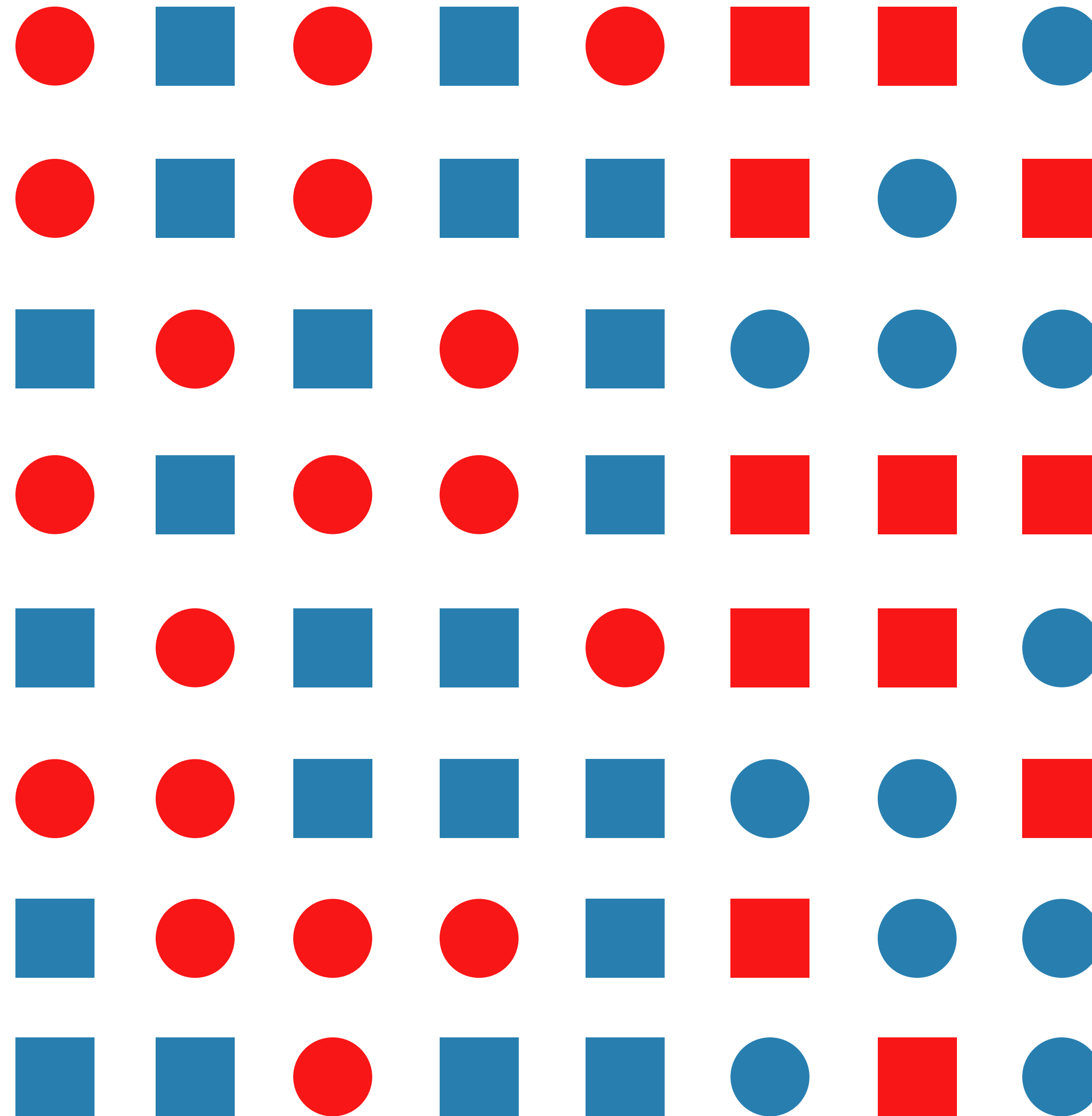
[F. J. Anscombe]

Visual Pop-out



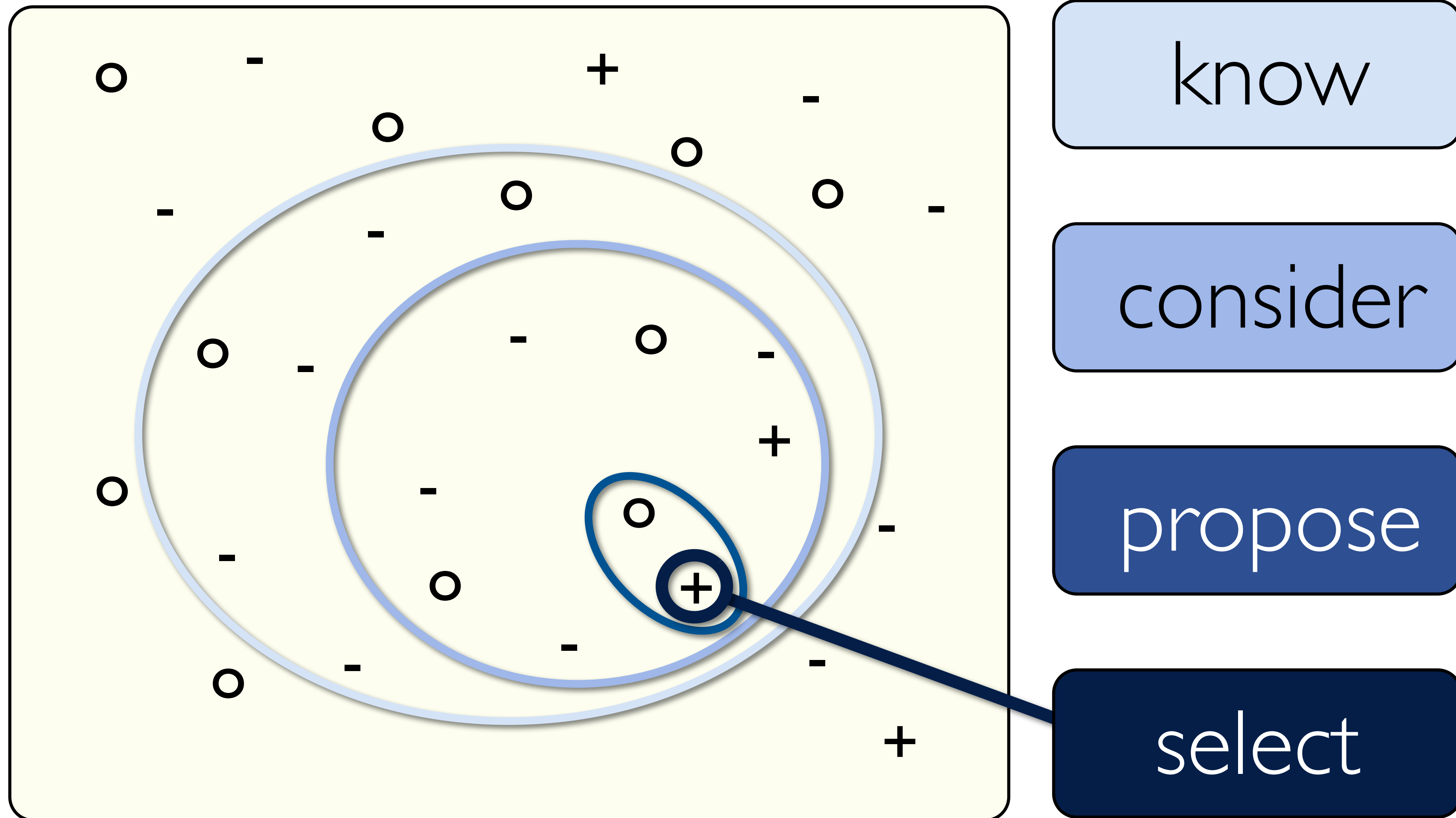
[C. G. Healey]

Visual Perception Limitations



[C. G. Healey]

Design Space: Think Broad



[Design Study Methodology, Sedlmair et al., 2012]

Assignment 1

- Link
- HTML, CSS, and SVG
- Due Friday
- Questions?



Languages of the Web

- HTML
- CSS
- SVG
- JavaScript
 - Versions of Javascript: ES6, ES2015, ES2017...
 - Specific frameworks: react, jQuery, bootstrap, D3

Web Programming Tools

- Basic: Text editor and Modern Browser
- Developer Tools: Built in to browsers (e.g. Chrome Developer Tools)
- Web Environments: CodePen, JSFiddle, Liveweave, CodeSandbox, etc.
- IDEs: WebStorm, etc.

Basic HTML File

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

- <https://codepen.io/dakoop/pen/PdRKEL>

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](https://developer.mozilla.org/en-US/docs/Web/CSS)

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>`
- <https://codepen.io/dakoop/pen/ErNJvJ>

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
- <https://codepen.io/dakoop/pen/MLbRQz>

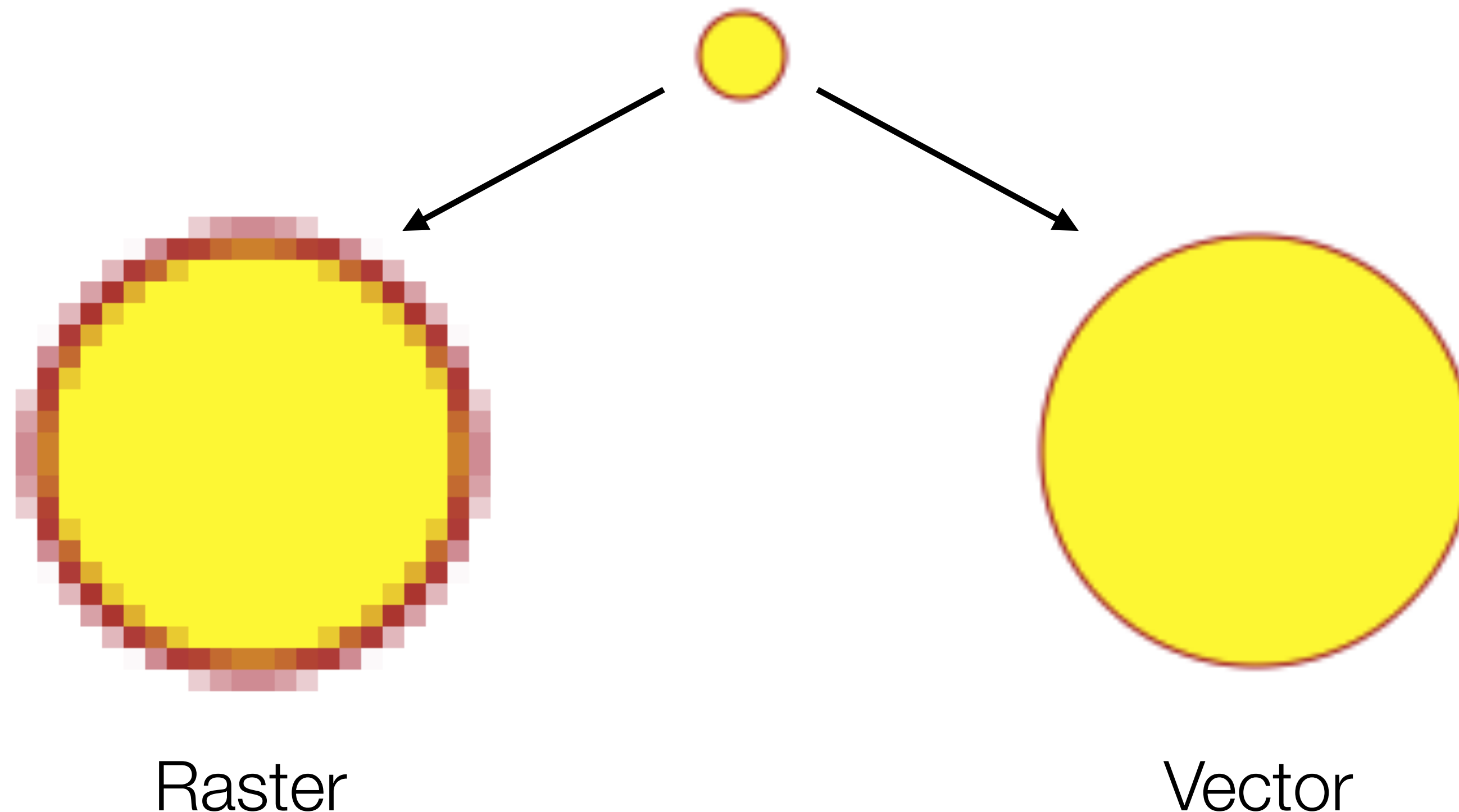
How to add CSS to HTML

- 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>`

What is the difference between
vector and raster graphics?

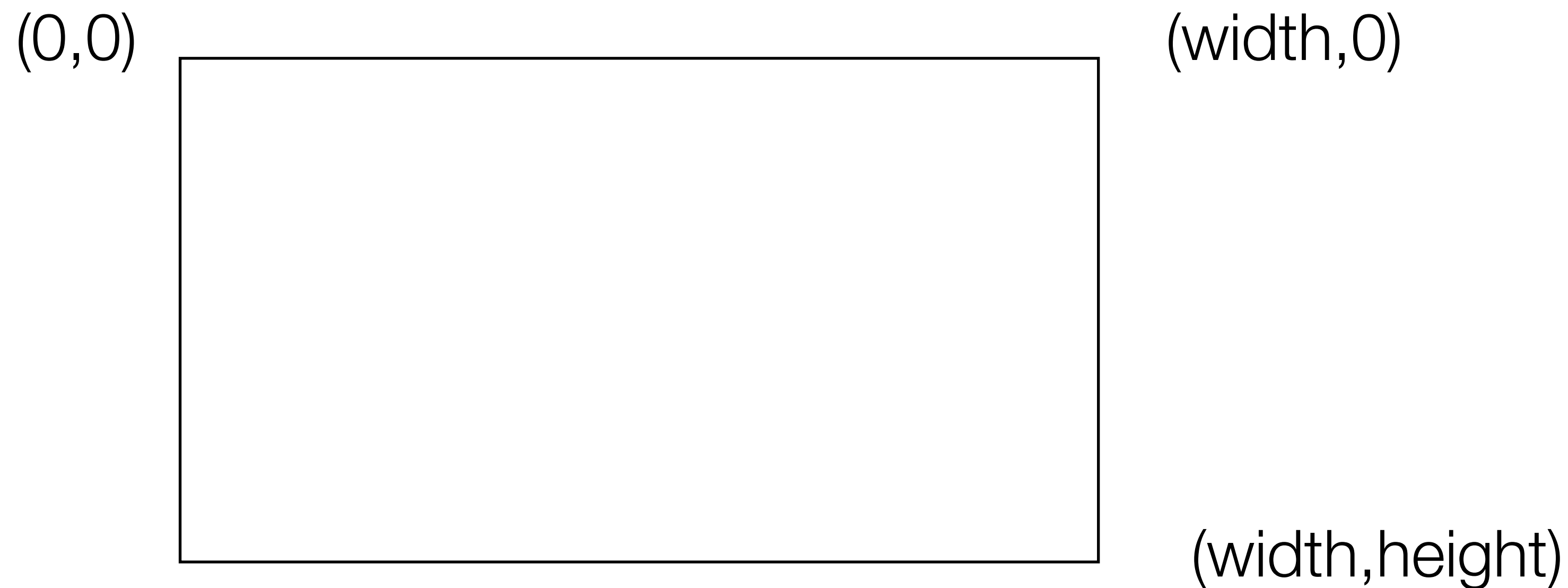
Scalable Vector Graphics (SVG)

- 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

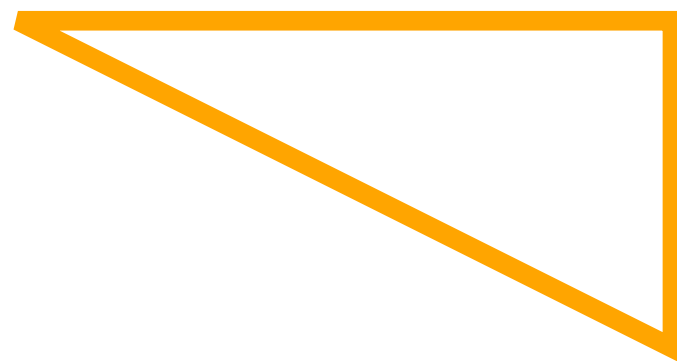
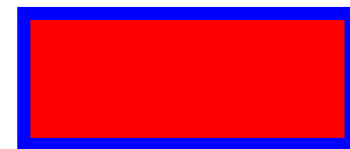
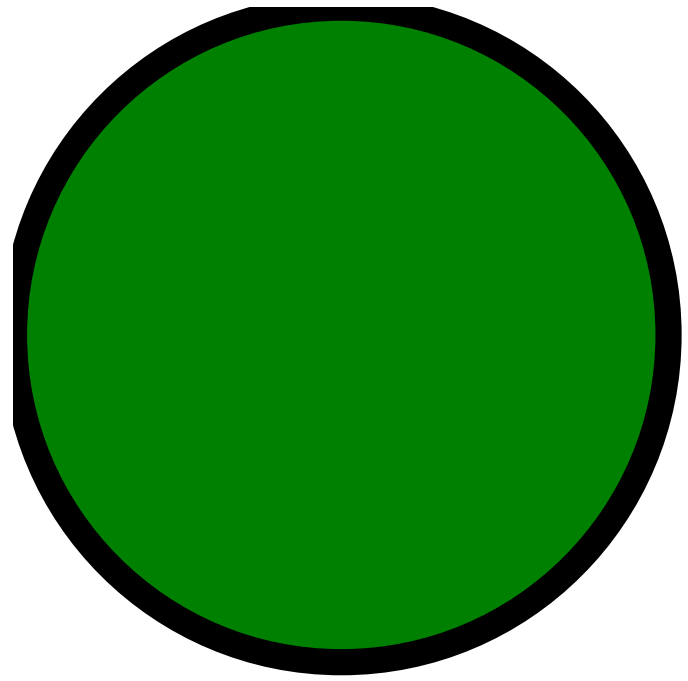
- Drawing primitives:
 - Lines, Circles, Rects, Ellipses, Text, Polylines, Paths
 - Work by specifying information about **how** to draw the shape
 - Lots more: see [MDN Documentation](#)
- Ordering/Stacking:
 - SVG Elements are drawn in the order they are specified
- Paths: directions for drawing
 -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

SVG Styles

- We can specify styles on 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



- <http://codepen.io/dakoop/pen/yexVXb>

```
<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; }
```


SVG Grouping

- Very powerful, useful for animations and transformations
- `<g> <circle .../> <circle ... /> <circle ... /></g>`
- Can add transforms to the group:
 - <http://codepen.io/dakoop/pen/rjpdXp>

```
<svg width="200" height="200">
  <g transform="translate(0, 200) scale(1, -1)">
    <circle cx="50" cy="50" r="10"/>
    <circle cx="80" cy="80" r="10"/>
    <circle cx="110" cy="50" r="10"/>
    <circle cx="140" cy="90" r="10"/>
  </g>
</svg>
```



[SVG Example, Scheidegger, 2016]

JavaScript in one slide

- Interpreted and Dynamically-typed Programming Language
- Statements end with semi-colons, normal blocking with brackets
- Variables: `var a = 0; let b = 2;`
- Operators: `+, -, *, /, [ ]`
- Control Statements: `if (<expr>) {...} else {...}, switch`
- Loops: `for, while, do-while`
- Arrays: `var a = [1,2,3]; a[99] = 100; console.log(a.length);`
- Functions: `function myFunction(a,b) { return a + b; }`
- Objects: `var obj; obj.x = 3; obj.y = 5;`
 - Prototypes for instance functions
- Comments are `/* Comment */` or `// Single-line Comment`

JavaScript References

- [Interactive Data Visualization for the Web](#), Murray
- [MDN Tutorials](#)