



# DSCI 554 LECTURE 5

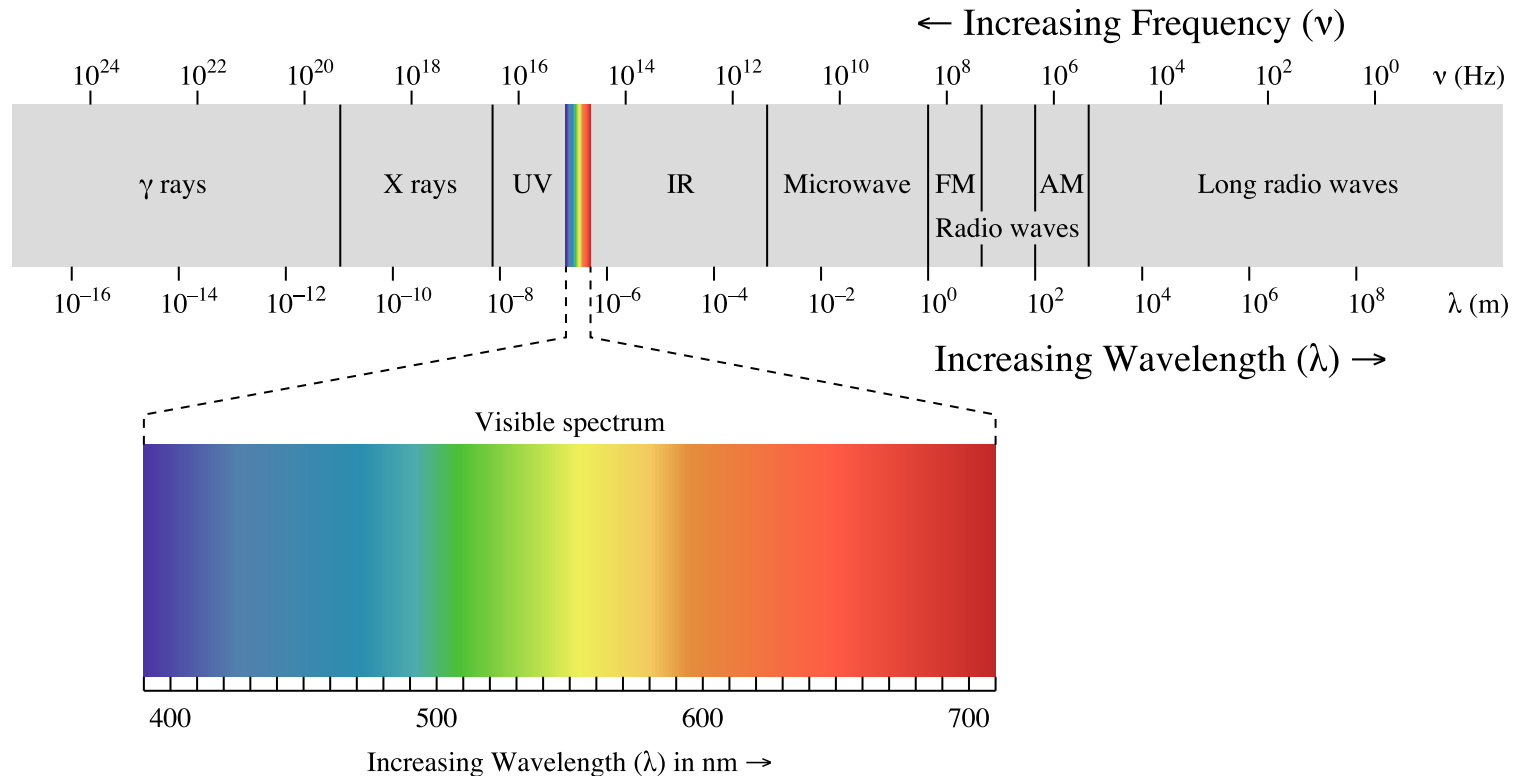
## THE EYE AND THE VISUAL BRAIN, D3 SCALES AND AXES

Dr. Luciano Nocera

# OUTLINE

- The eye and the visual brain
- D3 scales and axes

# VISIBLE SPECTRUM



Visible spectrum wavelengths from 400-700nm (in nanometers)

**MOST PEOPLE SEE RED**  
**CLOSER THAN BLUE**  
**BUT SOME SEE THE**  
**OPPOSITE EFFECT**

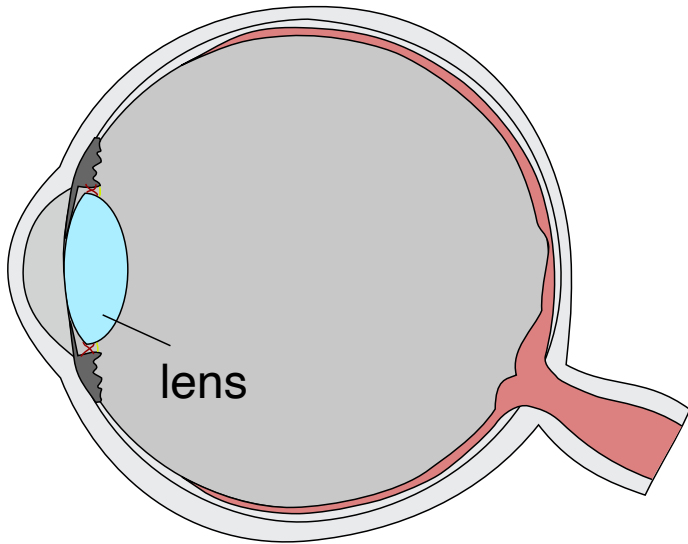
**MOST PEOPLE SEE RED**

**CLOSER THAN BLUE**

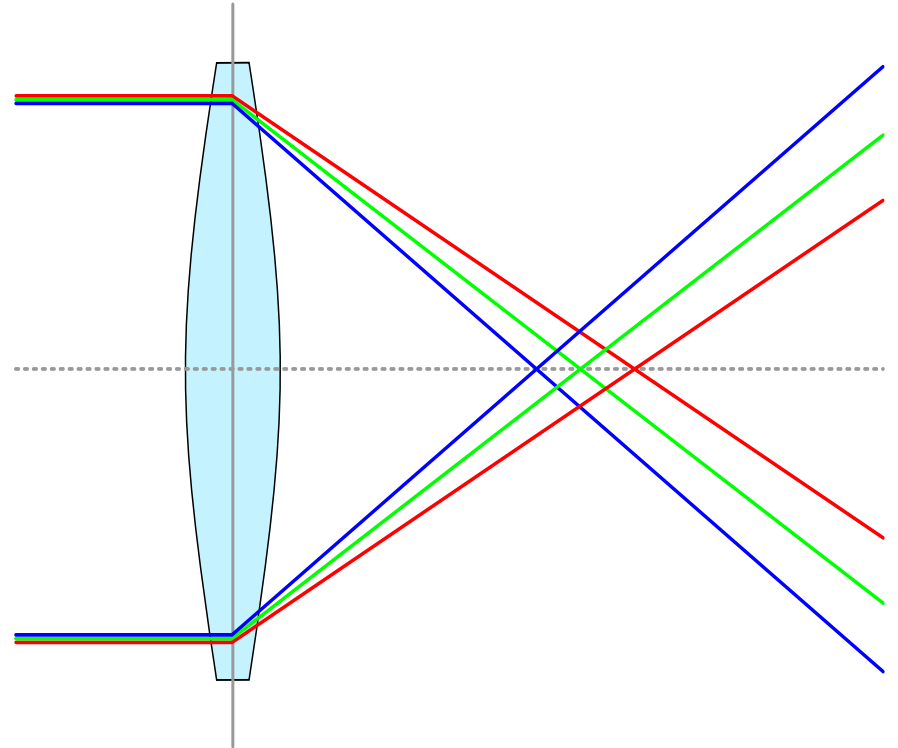
**BUT SOME SEE THE**

**OPPOSITE EFFECT**

# THE LENS



Eye: organ of the visual system that transforms light in signals that travel to the brain



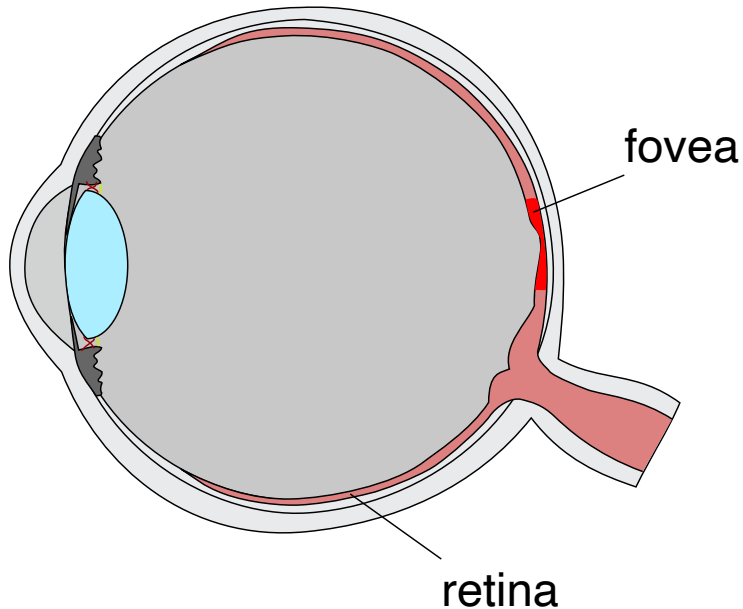
Chromatic aberration ([Chromostereopsis](#)), optical illusion caused by refraction and binocular vision



# THE RETINA

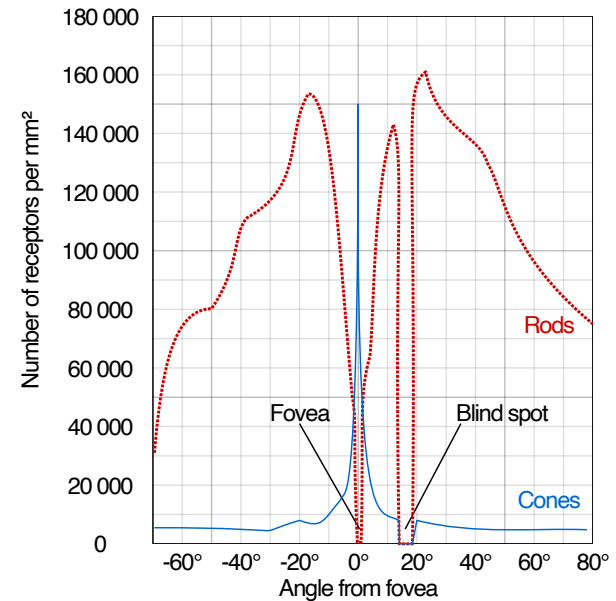
*Central high-res vision at Fovea with Cone cells*

**FOVEA  $\approx$  CENTRAL 1-2° OF FIELD OF VIEW**



Retina: layer containing cells sensitive to light

## PHOTORECEPTORS DISTRIBUTION



Distribution of rods and cones along a line passing through the fovea and the blind spot of a human eye. -- Foundations of Vision, Brian A. Wandell.



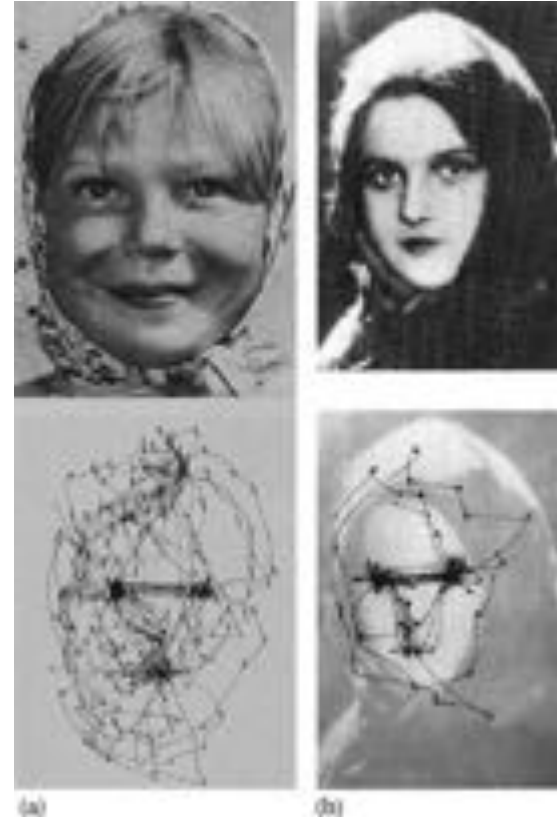
# SACCADES & FIXATIONS

## SACCADES

- Eye movements (about 3 each second)
- Accompanied by periods of blindness
- $> 200ms$  to initiate
- Fastest movements in body (up to  $900^{\circ}s^{-1}$ )

## FIXATIONS

- A glimpse
- When visual information is acquired
- Task dependent

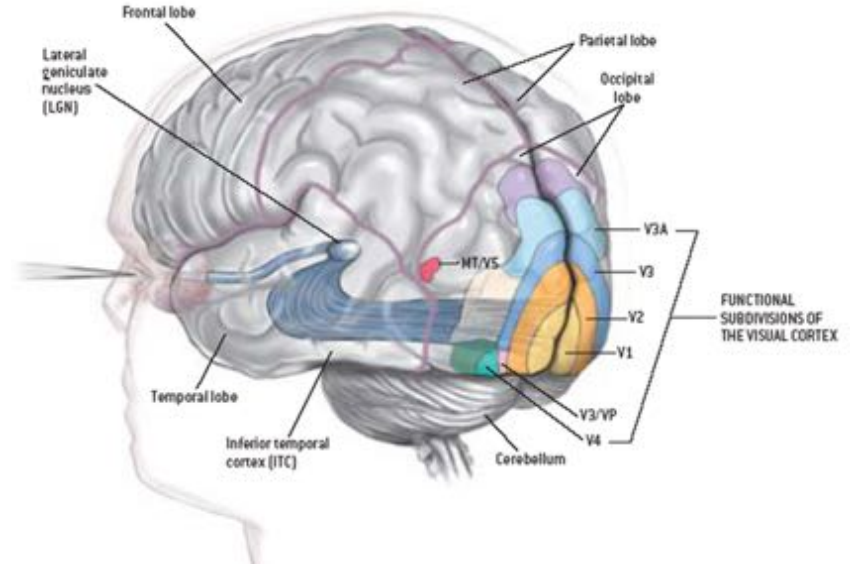


Yarbus A L. Eye Movements and Vision.  
New York: Plenum Press; 1967

# THE VISUAL SYSTEM

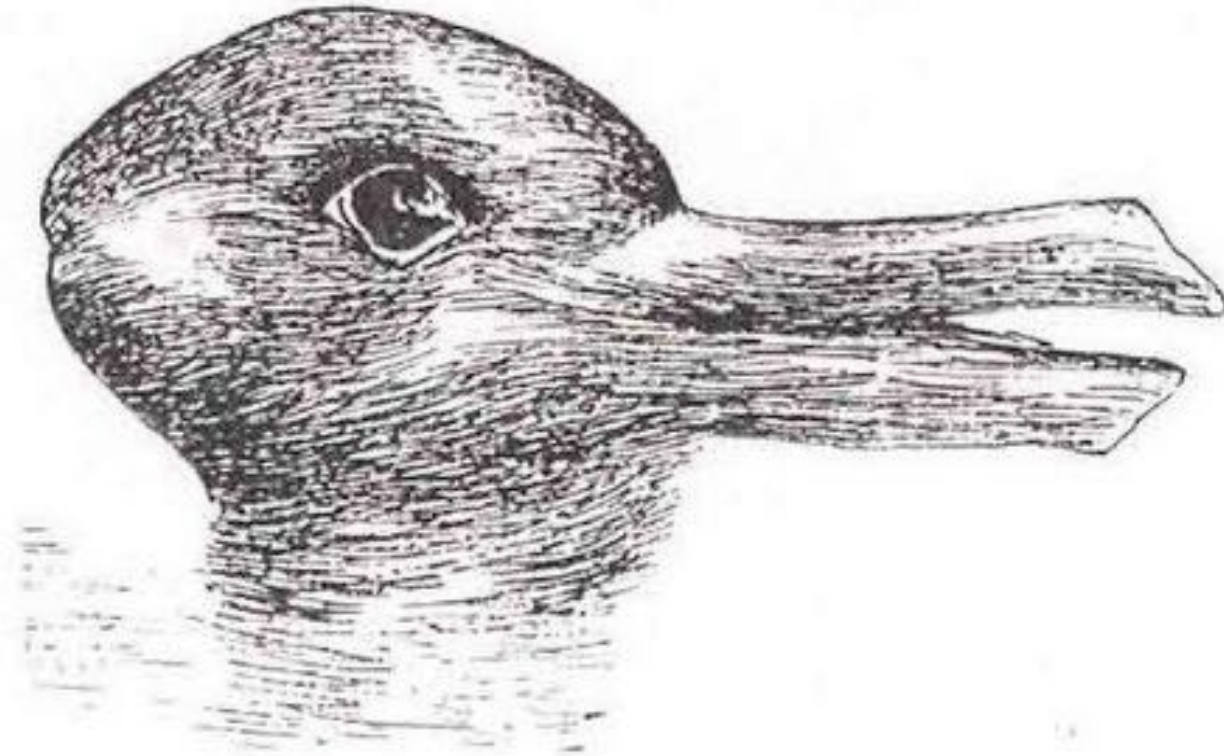
Comprised of:

- Eyes
- Nerves (optic nerve from eyes to brain)
- Visual cortex (areas V1-5)



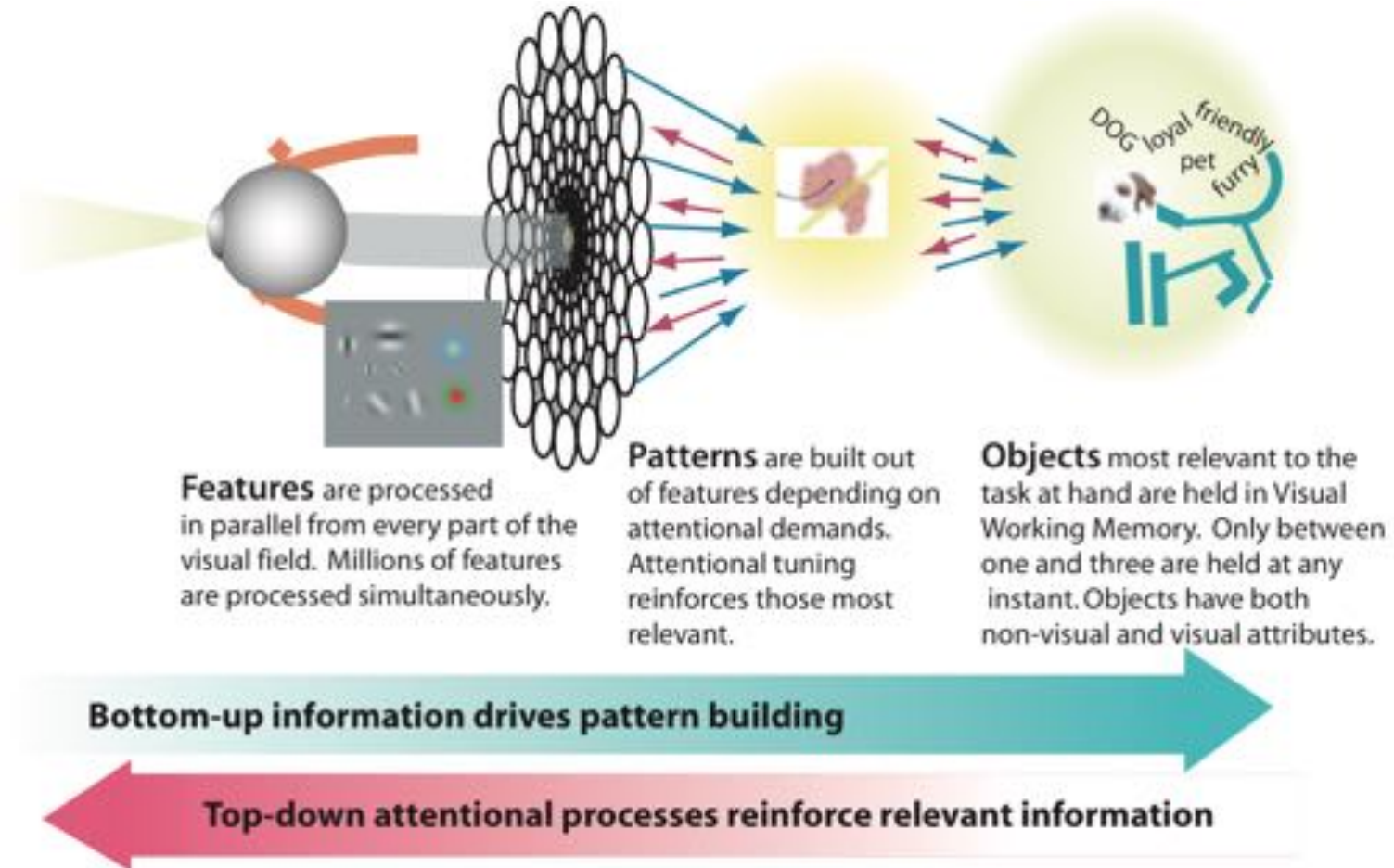
Expert Reviews in Molecular Medicine by Cambridge University Press 2004

# BOTTOM-UP & TOP-DOWN PROCESSING



Duck rabbit illusion. The figure is perceived as changing between two interpretations. As the brain interprets visual signals it provides continuity between fixations, adds information at the blind spot and performs object recognition.

# BOTTOM-UP & TOP-DOWN PROCESSING



Colin Ware. Visual thinking: For design. Morgan Kaufmann, 2010.

# BOTTOM-UP & TOP-DOWN PROCESSING

## Bottom-up or data-driven

---

Perception results from transforming sensory input into higher level information

---

Knowledge and attention influence perception

## Top-down or schema-driven

---

Signals from the eye are integrated and compared to examples in memory

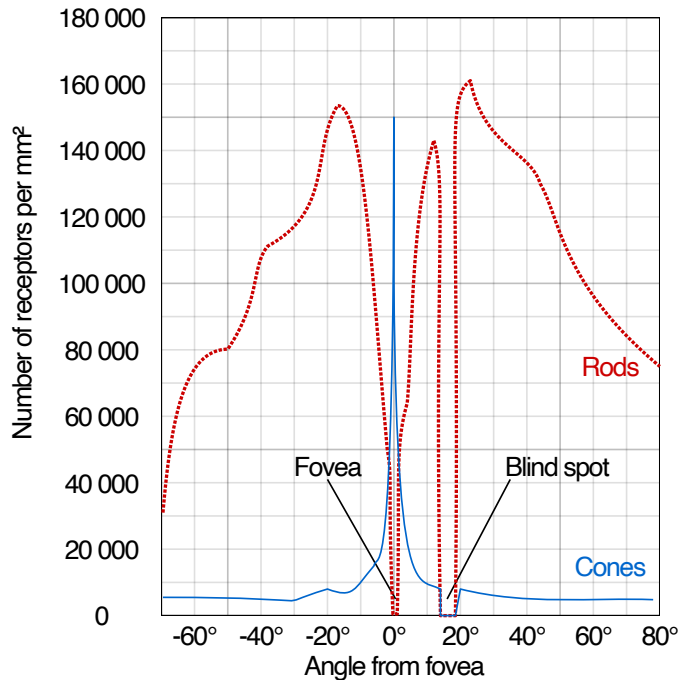
---

Object knowledge directs the eyes and the feature extraction processing

# VISUAL QUERIES

WHICH PHOTORECEPTOR IS PRESENT IN THE FOVEA?

Answer: almost exclusively cones



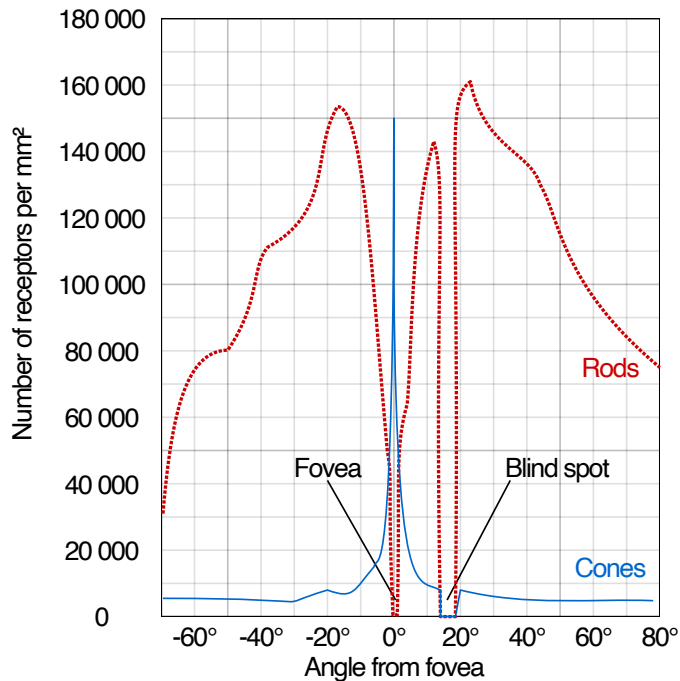
# VISUAL QUERIES

WHICH PHOTORECEPTOR IS PRESENT IN THE FOVEA?

Answer: almost exclusively cones

RELEVANT VISUAL QUERIES

- Find “Fovea” label
- Find Fovea region on x axis
- Find curve with larger y value (use grid lines)
- Follow corresponding curve and read label



# VISUAL QUERIES

WHICH PHOTORECEPTOR IS PRESENT IN THE FOVEA?

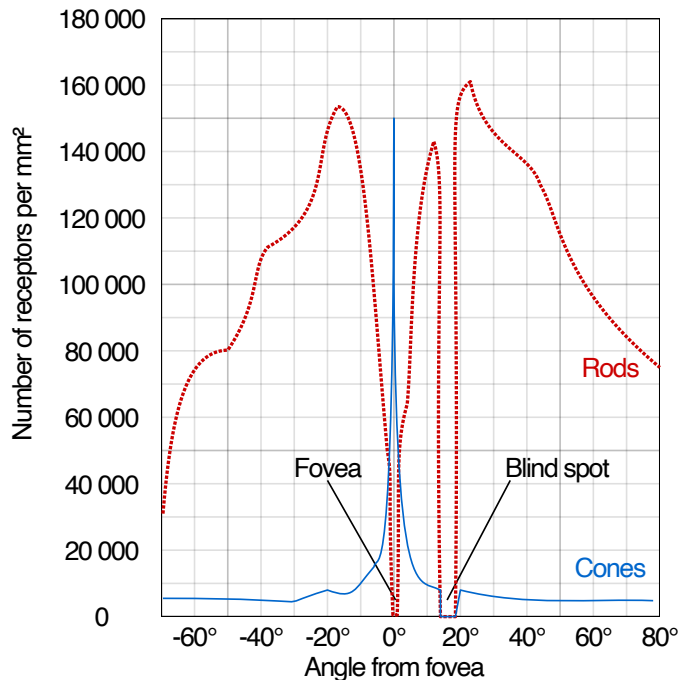
Answer: almost exclusively cones

RELEVANT VISUAL QUERIES

- Find “Fovea” label
- Find Fovea region on x axis
- Find curve with larger y value (use grid lines)
- Follow corresponding curve and read label

EXAMPLE OF BOTTOM-UP PROCESSING

Perceiving the lines, e.g., perceived dots in dotted line are integrated into a line representation





# VISUAL QUERIES

WHICH PHOTORECEPTOR IS PRESENT IN THE FOVEA?

Answer: almost exclusively cones

RELEVANT VISUAL QUERIES

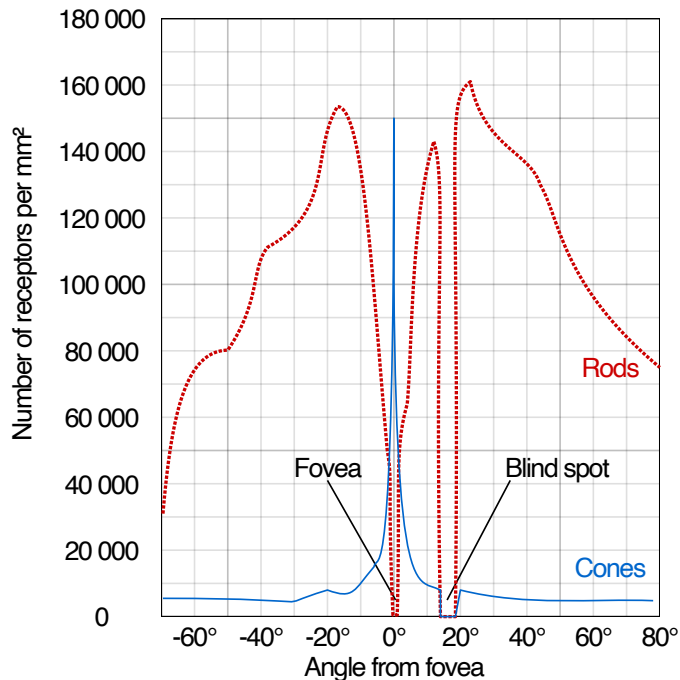
- Find “Fovea” label
- Find Fovea region on x axis
- Find curve with larger y value (use grid lines)
- Follow corresponding curve and read label

EXAMPLE OF BOTTOM-UP PROCESSING

Perceiving the lines, e.g., perceived dots in dotted line are integrated into a line representation

EXAMPLE OF TOP-DOWN PROCESSING

Search for text, e.g., “Fovea”



# VISUAL QUERIES AND DESIGN CONSIDERATIONS

*Visual query: a pattern cognitively specified, that if found in the display will contribute to the solution of a problem*

*-- C. Ware*

*Visual thinking consists of a series of acts of attention, driving eye movements and tuning our pattern-finding circuits*

*-- C. Ware*

# VISUAL QUERIES AND DESIGN CONSIDERATIONS

*Visual query: a pattern cognitively specified, that if found in the display will contribute to the solution of a problem*

*-- C. Ware*

*Visual thinking consists of a series of acts of attention, driving eye movements and tuning our pattern-finding circuits*

*-- C. Ware*

*Carefully craft visualizations to optimize visual queries*

*-- C. Ware*



The attention test



The Monkey business illusion. Daniel J. Simons.

# INATTENTIONAL BLINDNESS

- Failure to detect an unexpected stimulus that is fully visible
- Limited attention allows to focus on one thing at the time
- The brain prioritizes what to focus on

# INATTENTIONAL BLINDNESS & IMPLICATIONS FOR DESIGN

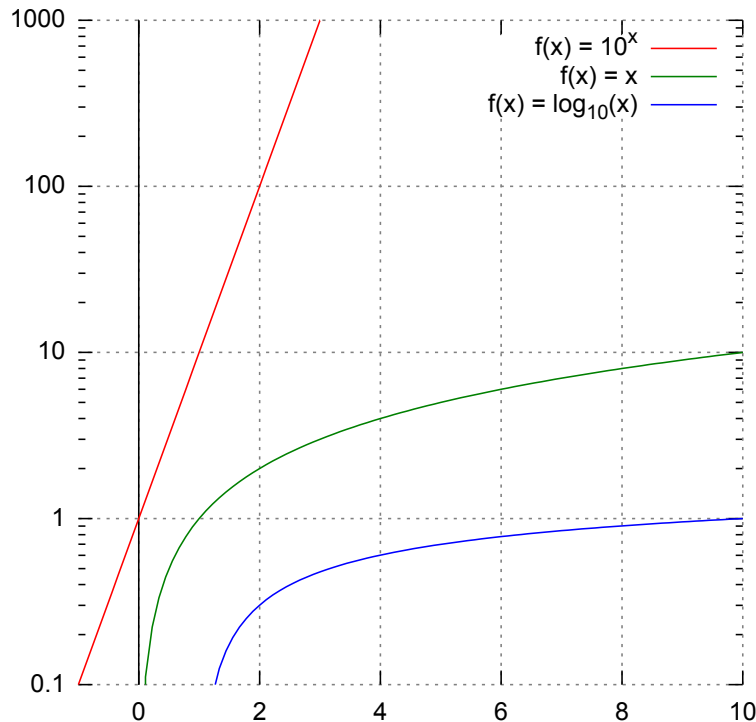
- For designers ... it is important to know what kinds of visual information the brain can process efficiently [Ware].
- Be aware of inattentional blindness. Never show simultaneous animations on different parts of the screen [Cairo].
- Do not use too many competing stimuli. Filling graphic with objects colored in pure accent tones disorients users [Cairo].

# OUTLINE

- The eye and the visual brain
- D3 scales and axes



# SCALE & AXES



Linear and log scales

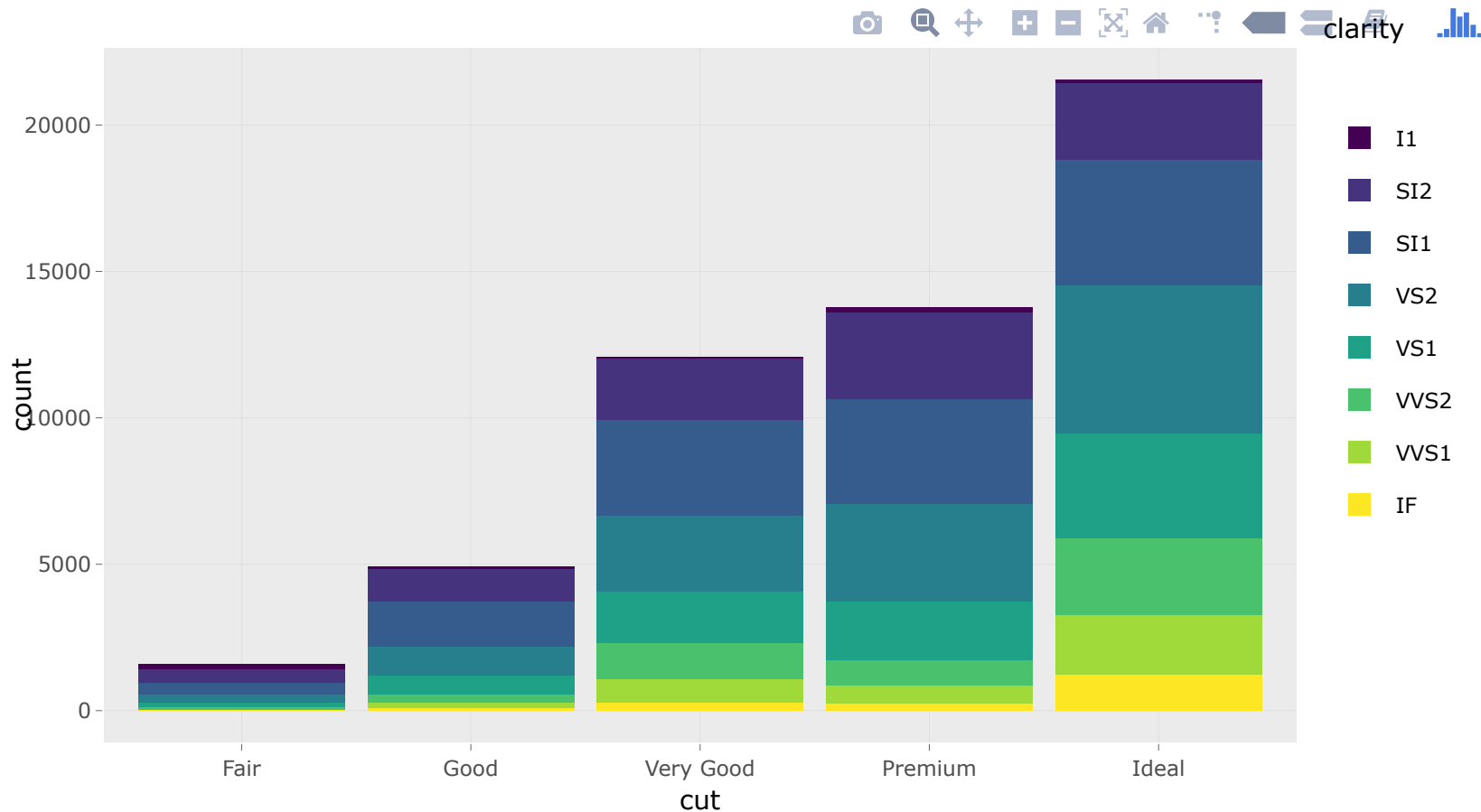
## SCALES

Encodings used to map data to visual representations

## AXES

Visual representations of the scale that let us read data values

# ANNOTATIONS



ggplot2 plot exported with Plotly

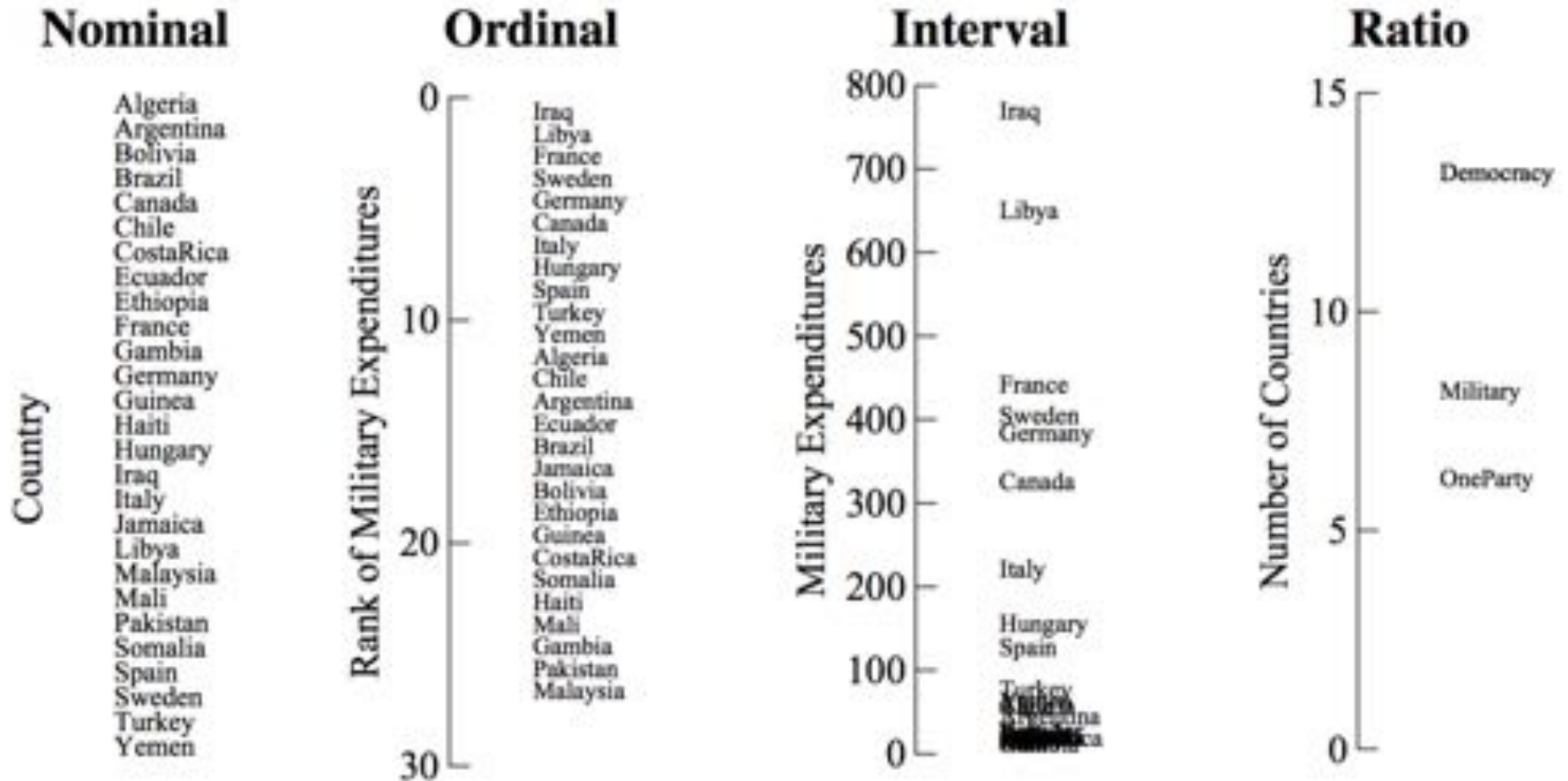
Graph include: title, data annotations

Axes include: axis labels, tick marks, tick mark labels, grid lines

Legends include: title, keys, key labels

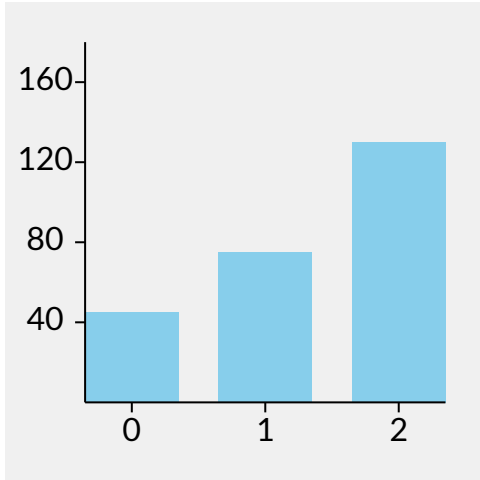


# TYPES OF SCALES [STEVENS 1946]



On the theory of scales of measurement, Stevens 1946

# D3 CHART WITH HARD CODED SCALES, AXES & LABELS WITH INSERT/APPEND



```
var dataset = [['0', 45], ['1', 75], ['2', 130]];

// define variables to use for chart sizing
var w = 200;
var h = 200;
var pad = 20;

var svg = d3.select('#svg00')
  .attr('width', w + 2 * pad)
  .attr('height', h + 2 * pad)
  .style('background-color', 'rgb(240, 240, 240)');

svg.selectAll('rect')
  .data(dataset)
  .enter()
  .append('rect')
  .attr('x', function(d, i) { return 2 * pad + i * (w / dataset.length); })
  .attr('y', function(d) { return h - d[1]; })
  .attr('width', w / dataset.length - pad)
  .attr('height', function(d) { return d[1]; })
  .style('fill', 'skyblue');

svg.selectAll('text')
  .data(dataset)
  .enter()
  .append('text')
  .text(function(d) { return d[0]; })
  .attr('x', function(d, i) {
    return 2 * pad + i * (w / dataset.length) + (w / dataset.length - pad) / 2; })
  .attr('y', function(d) { return h + 15; })
  .style('fill', 'black')
  .style('font-size', '16px')
  .style('text-anchor', 'middle')
  .style('alignment-baseline', 'middle');

svg.selectAll('line')
  .data(dataset)
  .enter()
  .append('line')
  .attr('x1', function(d, i) {
    return 2 * pad + i * (w / dataset.length) + (w / dataset.length - pad) / 2; })
  .attr('y1', function(d) { return h + 5; })
  .attr('x2', function(d, i) {
    return 2 * pad + i * (w / dataset.length) + (w / dataset.length - pad) / 2; })
  .attr('y2', function(d) { return h; })
  .style('stroke', 'black');

svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h)
  .attr('x2', w + pad)
  .attr('y2', h)
  .style('stroke', 'black');

svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h)
  .attr('x2', 2 * pad)
  .attr('y2', pad)
  .style('stroke', 'black');
```

```
svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h - 40)
  .attr('x2', 2 * pad - 5)
  .attr('y2', h - 40)
  .style('stroke', 'black');

svg.append('text')
  .text('40')
  .attr('x', pad)
  .attr('y', h - 40)
  .style('fill', 'black')
  .style('font-size', '16px')
  .style('text-anchor', 'middle')
  .style('alignment-baseline', 'middle');

svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h - 80)
  .attr('x2', 2 * pad - 5)
  .attr('y2', h - 80)
  .style('stroke', 'black');

svg.append('text')
  .text('80')
  .attr('x', pad)
  .attr('y', h - 80)
  .style('fill', 'black')
  .style('font-size', '16px')
  .style('text-anchor', 'middle')
  .style('alignment-baseline', 'middle');

svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h - 120)
  .attr('x2', 2 * pad - 5)
  .attr('y2', h - 120)
  .style('stroke', 'black');

svg.append('text')
  .text('120')
  .attr('x', pad)
  .attr('y', h - 120)
  .style('fill', 'black')
  .style('font-size', '16px')
  .style('text-anchor', 'middle')
  .style('alignment-baseline', 'middle');

svg.append('line')
  .attr('x1', 2 * pad)
  .attr('y1', h - 160)
  .attr('x2', 2 * pad - 5)
  .attr('y2', h - 160)
  .style('stroke', 'black');

svg.append('text')
  .text('160')
  .attr('x', pad)
  .attr('y', h - 160)
  .style('fill', 'black')
  .style('font-size', '16px')
  .style('text-anchor', 'middle')
  .style('alignment-baseline', 'middle');
```

# D3 SCALES AND AXES TO THE RESCUE!

## d3 / d3-scale

- Simplify mapping data to representation
- Facilitates complex data transformations

## d3 / d3-axis

- Simplify drawing of axes

# HOW D3 SCALES WORK

Input (Data)

Domain

Output (Representation)

Range

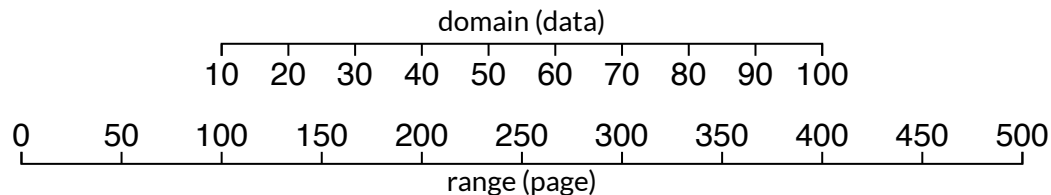


```
data = [10, 20, 30, 50, 80, 100];
```

```
min = d3.min(data); //10  
max = d3.max(data); //100
```

```
domain = [10, 100];
```

```
range = [100, 400];
```



# TYPES OF D3 SCALES

## Continuous

- Quantitative data
  - Continuous domain
- 
- Linear
  - Time
  - Power
  - Log
  - Quantize (rounds to set of discrete values)
  - Quantile (computes quantiles)
  - Sequential
  - Threshold (specifies arbitrary breaks)

## Ordinal

- Qualitative data
  - Discrete domain
- 
- Ordinal
  - Band
  - Point

# D3.SCALELINEAR()

$$y = a x + b$$

```
dataset = [100, 120, 150];  
  
var x = d3.scaleLinear() //function are objects in js  
    .domain([d3.min(dataset), d3.max(dataset)]) //extent of the data  
    .range([0, 100]); //range is the extent of the svg in pixels  
  
x(125); //50
```



# D3.SCALETIME()

$$y = a t + b$$

```
//use Date() to specify time in milliseconds  
var x = d3.scaleTime()  
    .domain([new Date(2018, 8, 20), new Date(2018, 11, 12)])  
    .range([0, 960]);  
  
x(new Date(2018, 11, 30)); //evaluate for date  
x(Date.now()); //evaluates scale for today's date
```

[MDN Web docs: Date](#)

# D3.SCALELOG()

$$y = a \log(x) + b$$

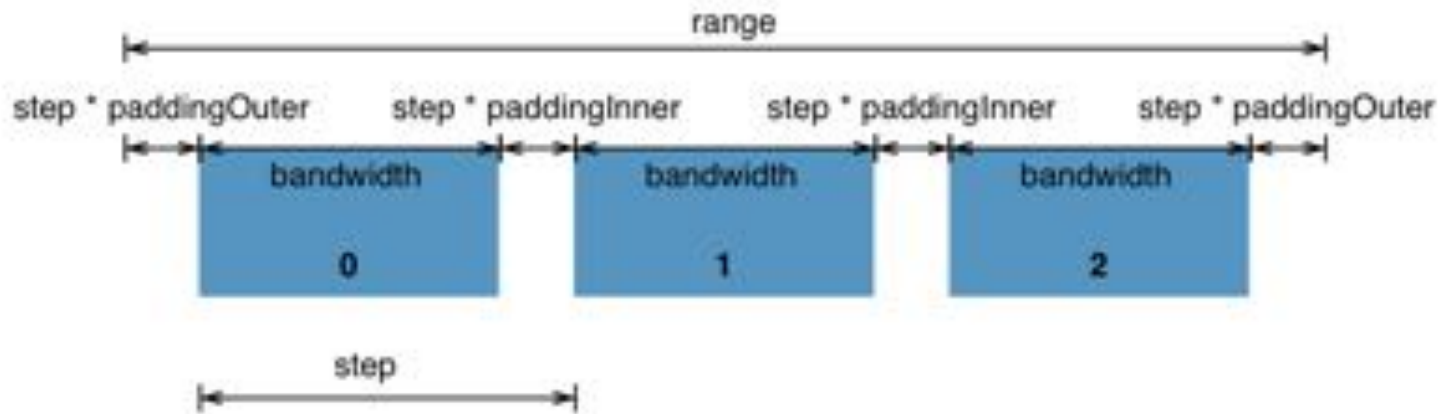
```
var x = d3.scaleLog()  
    .domain([1, 10000])  
    .range([0, 1000])  
    .base(10);
```

```
x(10);    //250  
x(10000); //1000
```

# D3.SCALEORDINAL()

```
var x = d3.scaleOrdinal()  
  .domain(['0', '1', '2']) //discrete domain  
  .range([0, 1, 2]);  
  
x('0'); //0  
x('1'); //1
```

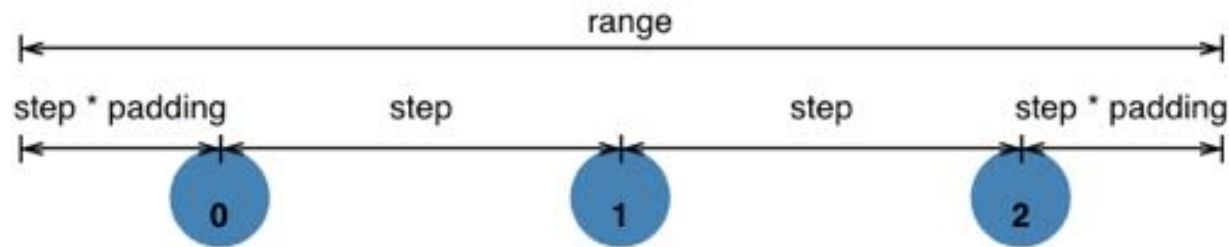
# D3.SCALEBAND()



```
var x = d3.scaleBand()  
  .domain(['0', '1', '2']) //discrete domain  
  .range([0, 600])  
  .paddingInner(0.05); //set inner padding in [0, 1], defaults to 0.  
  
x('0'); //0  
x('1'); //303.4  
x.bandwidth(); //193.2
```

# D3.SCALEPOINT()

Same as scaleBand() with bandwidth = 0



```
var x = d3.scalePoint()  
  .domain(['0', '1', '2']) //discrete domain  
  .range([0, 600]);  
  
x('0'); //0  
x('1'); //300
```

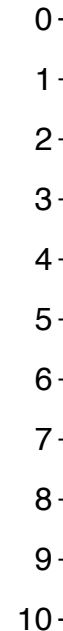
# BOILERPLATE CODE FOR DRAWING AXES

```
var svg = d3.select('#svg02')
  .attr("width", 400)
  .attr("height", 400);

// 1. CREATE SCALE FOR AXIS
var scale = d3.scaleLinear()
  .domain([0, 10])
  .range([0, 300]);

// 2. CREATE AXIS AND SET THE SCALE
var axis = d3.axisLeft(scale);

// 3. ADD AXIS IN A GROUP AND PLACE
svg.append("g")
  .attr("transform", "translate(50,50)")
  .call(axis); //call axis to draw
```



# 4 TYPES OF D3 AXES

```
var svg = d3.select('#svg03')
  .attr("width", 400)
  .attr("height", 400);

var scale = d3.scaleLinear()
  .domain([0, 10])
  .range([0, 300]);

//vertical axis with ticks on the left
var axis_l = d3.axisLeft(scale);

//vertical axis with ticks on the right
var axis_r = d3.axisRight(scale);

//horizontal axis with ticks on the top
var axis_t = d3.axisTop(scale);

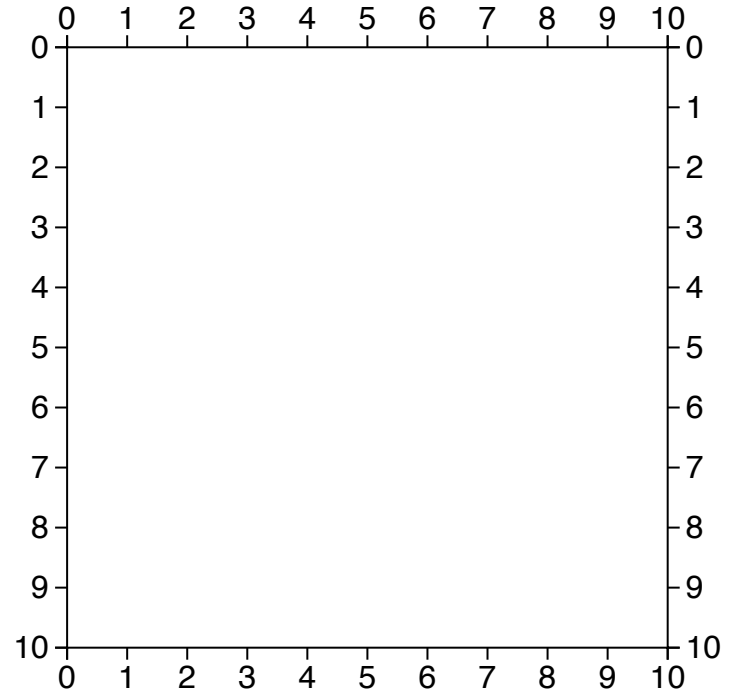
//vertical axis with ticks on the bottom
var axis_b = d3.axisBottom(scale);

svg.append("g")
  .attr("transform", "translate(50,50)")
  .call(axis_l);

svg.append("g")
  .attr("transform", "translate(350,50)")
  .call(axis_r);

svg.append("g")
  .attr("transform", "translate(50,50)")
  .call(axis_t);

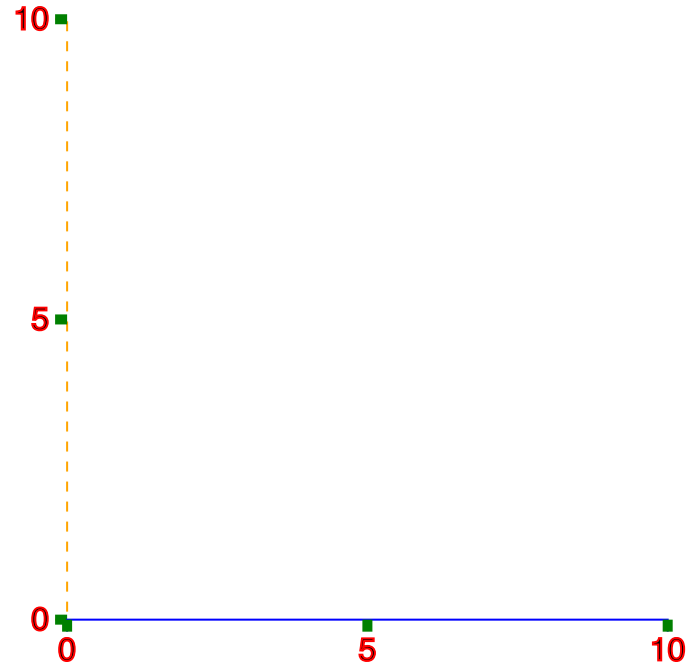
svg.append("g")
  .attr("transform", "translate(50,350)")
  .call(axis_b);
```



# BOILERPLATE CODE FOR STYLING AXES

```
<style type="text/css">
  #svg04 path {
    stroke: blue;
  }
  #svg04 .dashed-axis path { //axis is <path>
    stroke: orange;
    stroke-dasharray: 5,5;
  }
  #svg04 .tick line { //ticks are <line>
    stroke: green;
    stroke-width: 5px;
    shape-rendering: crispEdges;
  }
  #svg04 .tick text {
    stroke: red;
    font-family: sans-serif;
    font-size: 16px;
  }
</style>

<script>
  var svg = d3.select('#svg04')
    .attr("width", 400)
    .attr("height", 400);
  var y = d3.scaleLinear().domain([0, 10]).range([300, 0]);
  var x = d3.scaleLinear().domain([0, 10]).range([0, 300]);
  var axis_l = d3.axisLeft(y).ticks(3);
  var axis_b = d3.axisBottom(x).ticks(3);
  svg.append("g")
    .attr("transform", "translate(50,50)")
    .attr('class', 'dashed-axis')
    .call(axis_l);
  svg.append("g")
    .attr("transform", "translate(50,350)")
    .call(axis_b);
</script>
```





# BOILERPLATE CODE FOR AXES LABELS AND GRID LINES

```
<style>
  .label {
    font-size: 18px;
    text-anchor: middle;
    alignment-baseline: middle;
  }

  .dashed-axis path {
    stroke-dasharray: 3, 3;
  }
</style>
<script>
  var svg = d3.select('#svg05').attr("width", 400).attr("height", 400);

  var y = d3.scaleLinear().domain([0, 10]).range([300, 0]);
  var x = d3.scaleLinear().domain([0, 10]).range([0, 300]);
  var axis_l = d3.axisLeft(y)
  var axis_b = d3.axisBottom(x)

  svg.append("g")
    .attr("transform", "translate(50,50)")
    .call(axis_l);

  svg.append("g")
    .attr("transform", "translate(50,350)")
    .call(axis_b);

  var axis_hg = d3.axisBottom(x) //create and place grid lines
    .tickSize(0)
    .ticks(0);

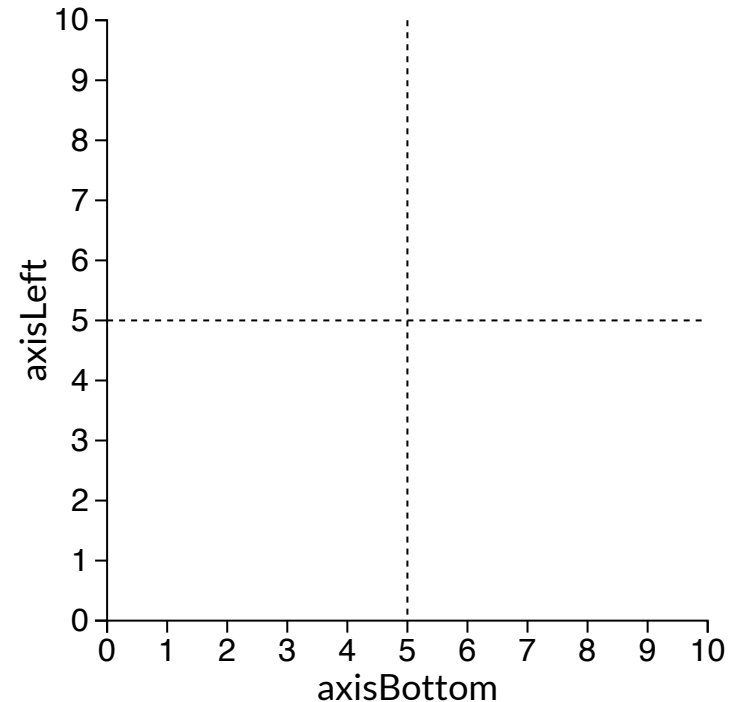
  var axis_vg = d3.axisLeft(x)
    .tickSize(0)
    .ticks(0);

  svg.append("g")
    .attr("transform", "translate(50,200)")
    .attr('class', 'dashed-axis')
    .call(axis_hg);

  svg.append("g")
    .attr("transform", "translate(200,50)")
    .classed('dashed-axis', true)
    .call(axis_vg);

  svg.append("text") //create and place labels
    .attr("x", 200)
    .attr("y", 385)
    .classed('label', true)
    .text("axisBottom");

  svg.append("text")
    .attr("x", -200)
    .attr("y", 15)
    .classed('label', true)
    .attr("transform", "rotate(-90)")
    .text("axisLeft");
</script>
```



# MARGIN CONVENTION

```
var margin = {top: 20, right: 20, bottom: 20, left: 20}; //step1: set margin
var width = 600 - margin.left - margin.right, //step2: set width and height
    height = 300 - margin.top - margin.bottom;

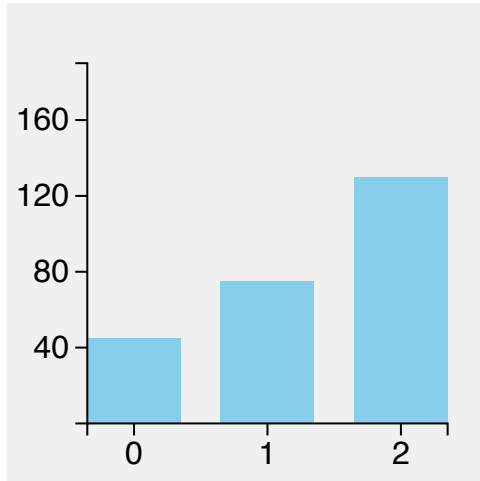
var svg = d3.select("body").append("svg") //step3: set-up svg
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

## Facilitates plots:

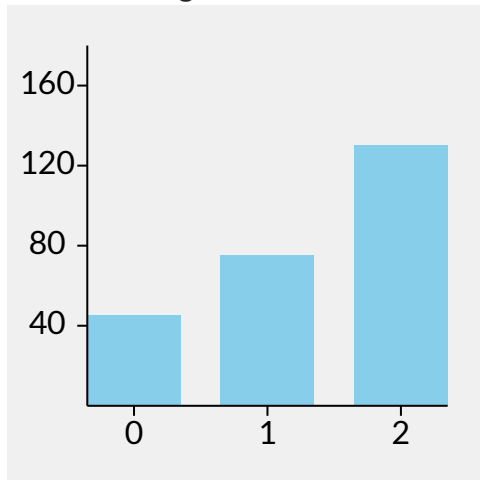
- Moves origin to bottom-left
- Adds padding for labels
- Inverts the y axis



# BAR CHART WITH D3 SCALES, AXES AND MARGIN CONVENTION



Using D3 scales and axes



Manually added scales and axes

```
<svg id="p_39_1" style="background-color: #f0f0f0; margin-right: 40px"></svg>

<style>
  path { stroke: black; }
  .tick line { stroke: black; }
  .tick { stroke: black; }
  .tick text {
    stroke: none;
    fill: black;
    font-size: 16px;
  }
</style>

<script>
  var data = [['0', 45], ['1', 75], ['2', 130]];
  var margin = { top: 30, right: 20, bottom: 30, left: 40 };
  var width = 240 - margin.left - margin.right,
      height = 240 - margin.top - margin.bottom;

  var x = d3.scaleBand()
    .domain(data.map(d => d[0]))
    .range([0, width])
    .paddingInner(.3);

  var y = d3.scaleLinear()
    .domain([0, d3.max(data, d => d[1]) + margin.top + margin.bottom])
    .range([height, 0]);

  var xAxis = d3.axisBottom().scale(x);

  var yAxis = d3.axisLeft().scale(y)
    .tickValues([40, 80, 120, 160]);

  var svg = d3.select("#p_39_1")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

  svg.selectAll('rect')
    .data(data)
    .enter()
    .append('rect')
    .attr('x', (d) => x(d[0]))
    .attr('y', (d) => y(d[1]))
    .attr('width', x.bandwidth)
    .attr('height', (d) => height - y(d[1]))
    .style('fill', 'skyblue');

  svg.append("g")
    .attr("transform", "translate(0," + height + ")")
    .call(xAxis);

  svg.append("g").call(yAxis);
</script>
```

Complete code including CSS!

