

# Seeing the Big Picture

Quick and Dirty Data Visualization with Ruby

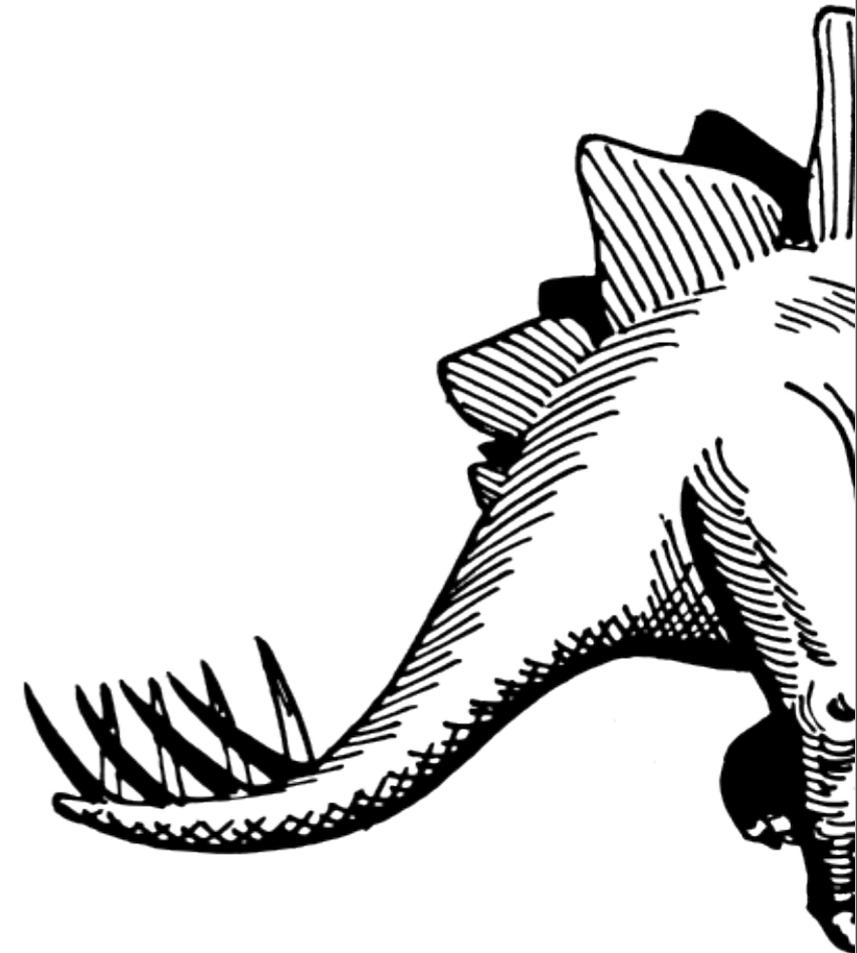
**Aja Hammerly**

@thagomizer\_rb

<http://github.com/thagomizer>

<http://www.thagomizer.com>

...L...E...S...T...P...A...T...T...I...P...A...L...



# Motivations



<http://www.flickr.com/photos/lumen850/5461476268/>



<http://www.flickr.com/photos/7147684@N03/1037533775>



<http://www.flickr.com/photos/kalavinka/4610308398>







**ORACLE®**



<http://www.flickr.com/photos/jamisonjudd/2433102356>

**WAIT**





<http://www.flickr.com/photos/magillicuddy/25074192>



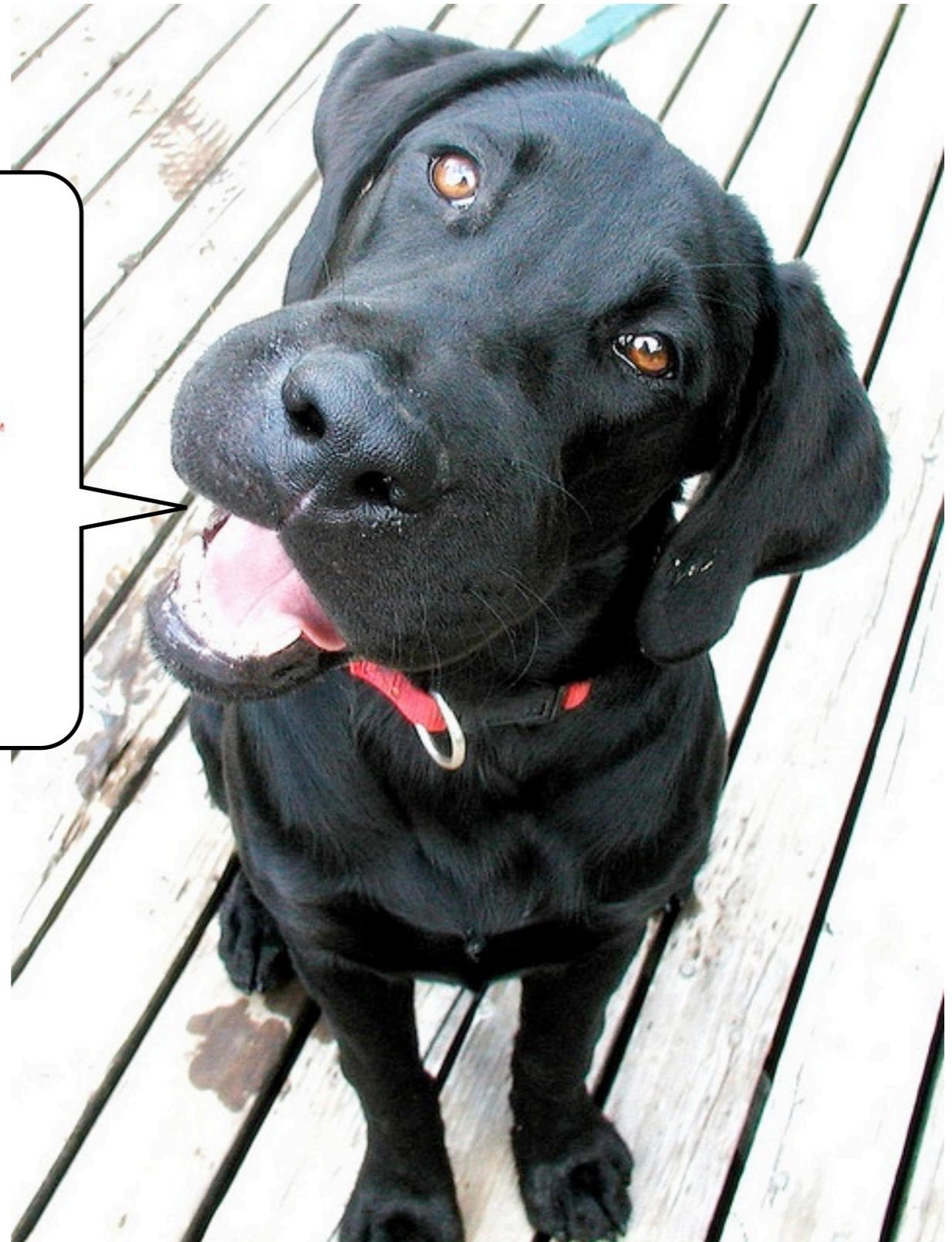
<http://www.flickr.com/photos/kalavinka/4610308398>



<http://www.flickr.com/photos/kalavinka/4610308398>



<http://www.flickr.com/photos/oakleyoriginals/3526895658>



<http://www.flickr.com/photos/oakleyoriginals/3526895658>

# Solution

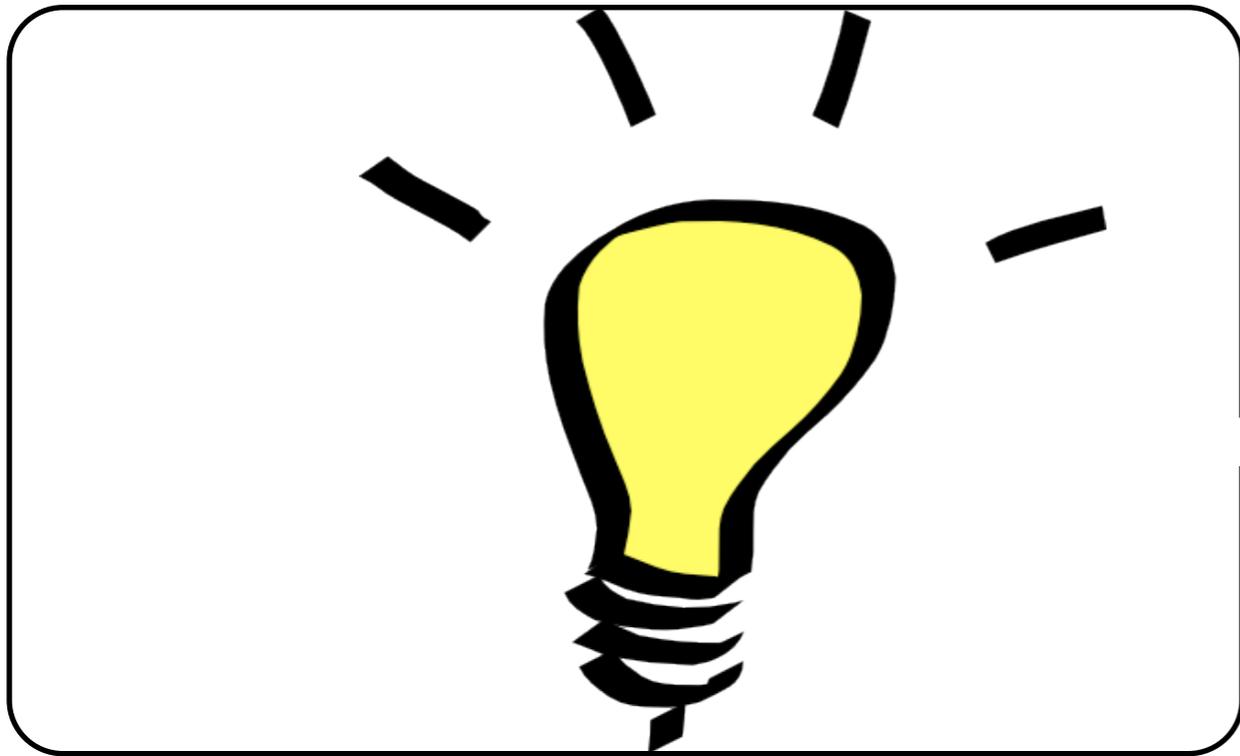
Pick a better graph,  
perhaps traffic versus  
backups with fake data

### Leg Curl





<http://www.flickr.com/photos/oakleyoriginals/3526895658>



<http://www.flickr.com/photos/oakleyoriginals/3526895658>

# Data Is Proof

# People Pattern Match

# Pictures Are Universal

# Overview

# Expectations

- 93 slides left (eep!)
- Code heavy
- Practitioner Focused
- Slides & Code Available

# Figure descriptive des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.

Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite. Paris, le 20 Novembre 1869.

Les hommes présents sont représentés par les largeurs des zones colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers de la carte. Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M. M. Chiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre. En jugeant à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davout qui avaient été détachés sur Minsk et Mohilow et qui rejoignirent vers Orscha et Witebsk, avaient toujours marché avec l'armée.

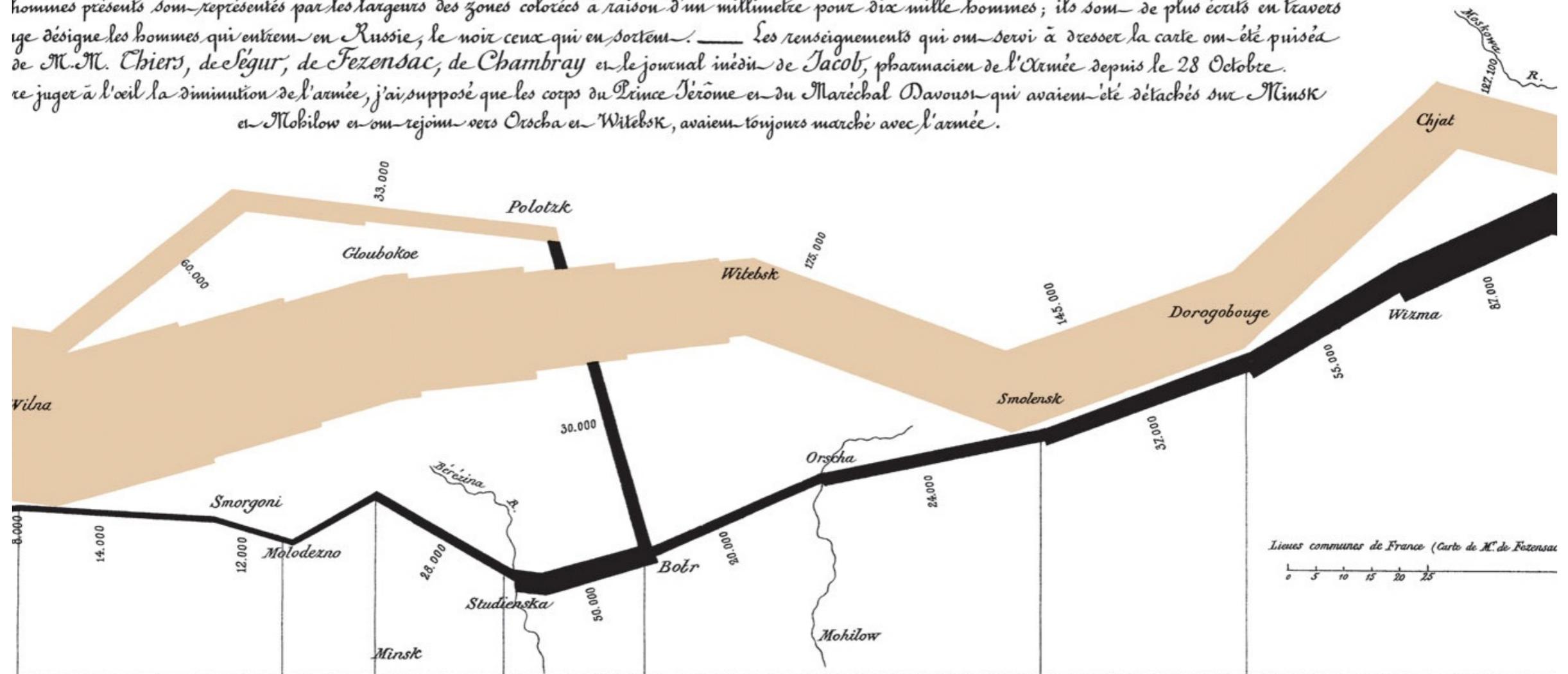
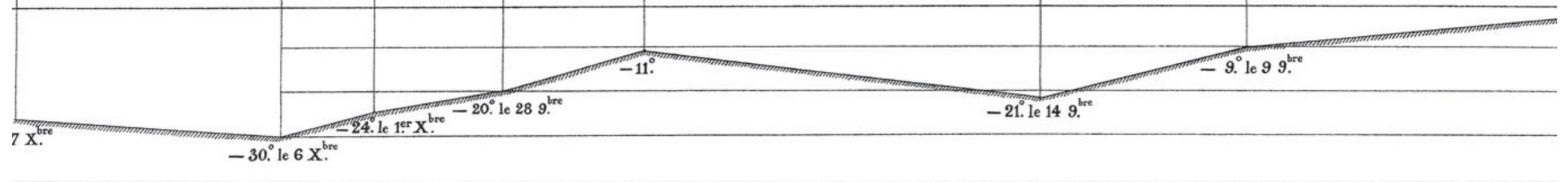
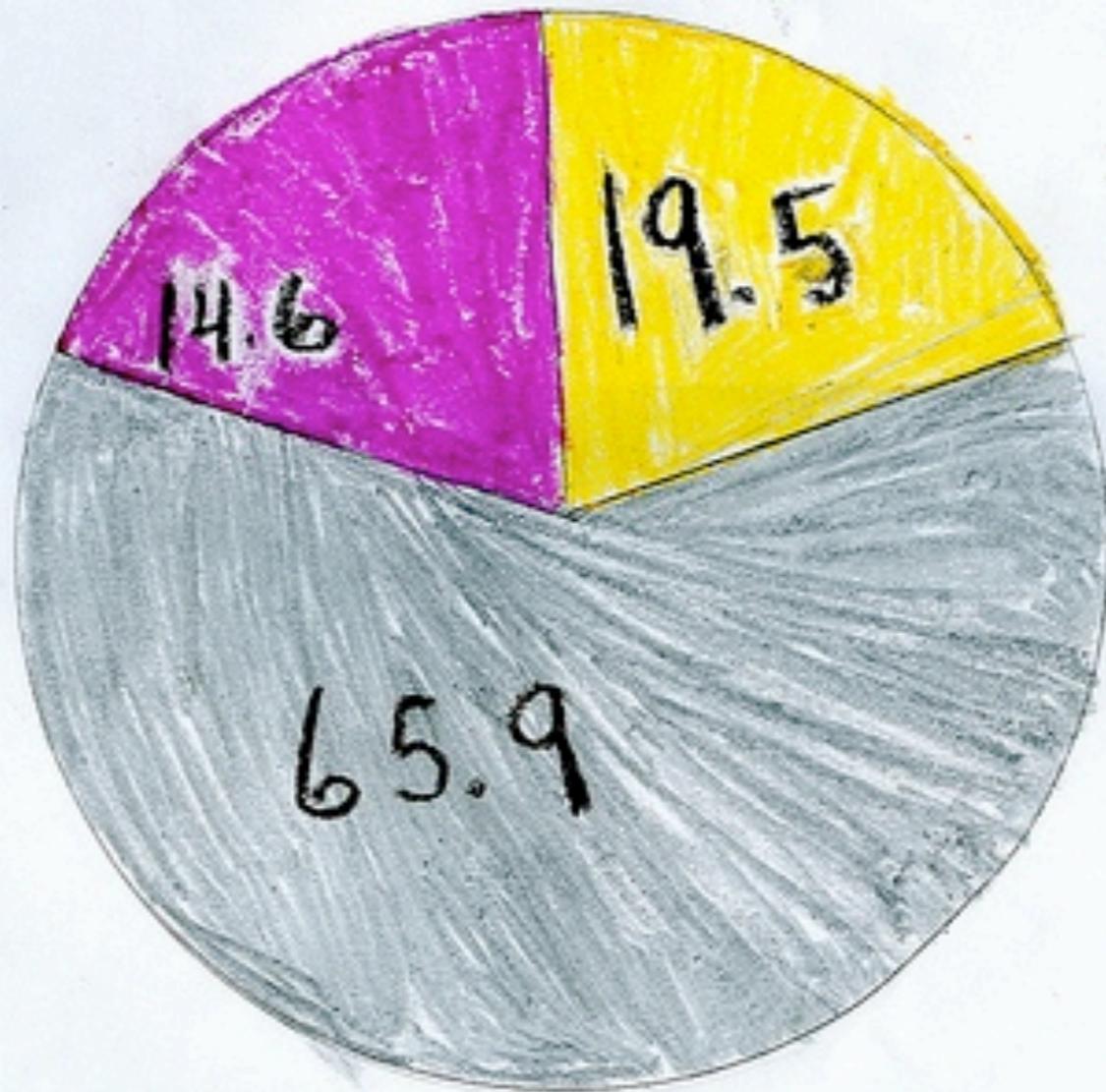


TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.



Do 41 Fifth Graders Love, Think he is ok,  
or strongly dislike Justin Bieber

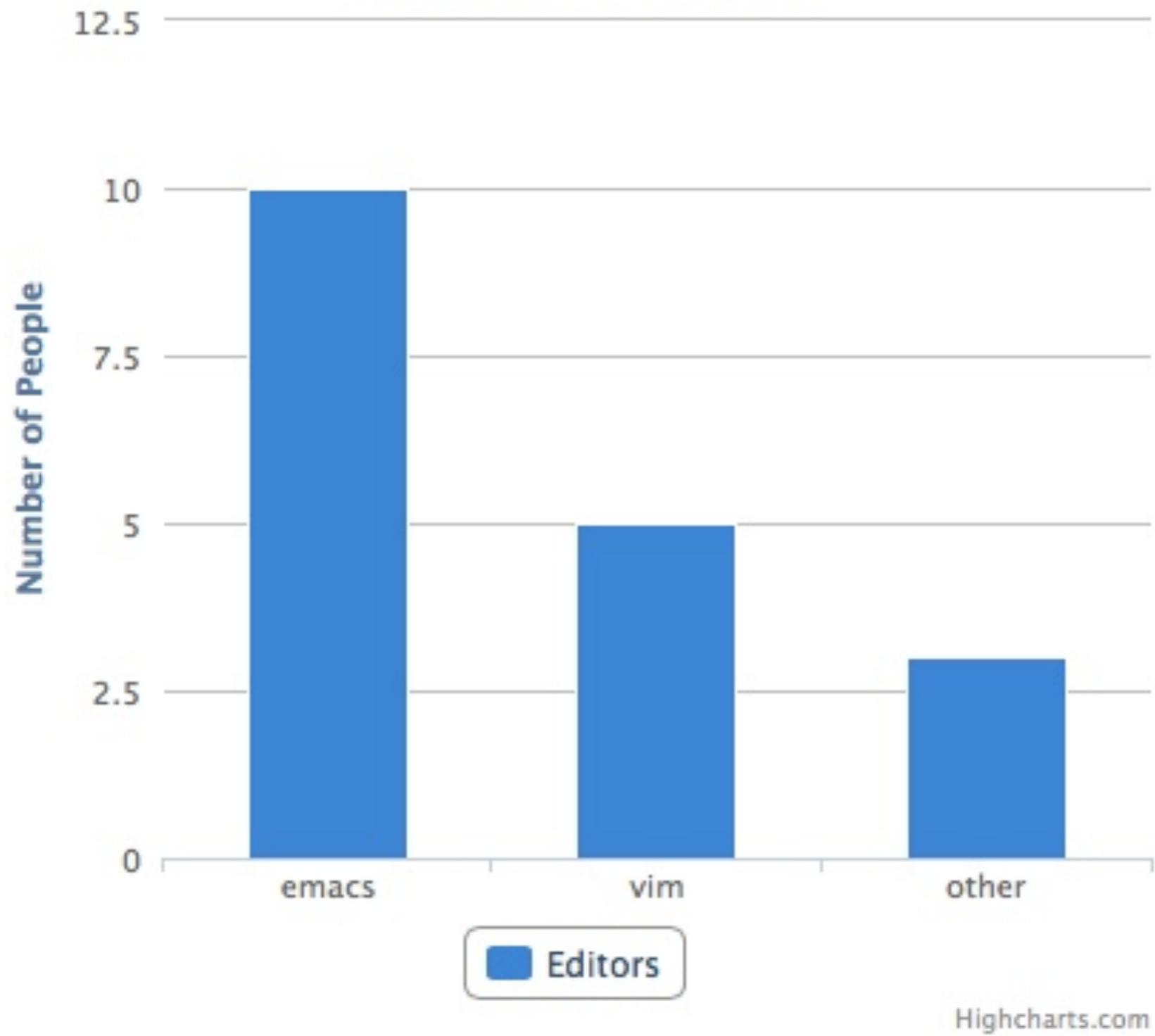


Key  
■ Love  
■ Think he is ok  
■ strongly dislike

Sophia G

<http://www.flickr.com/photos/dw2002/5451586738>

## Editors At Seattle.rb





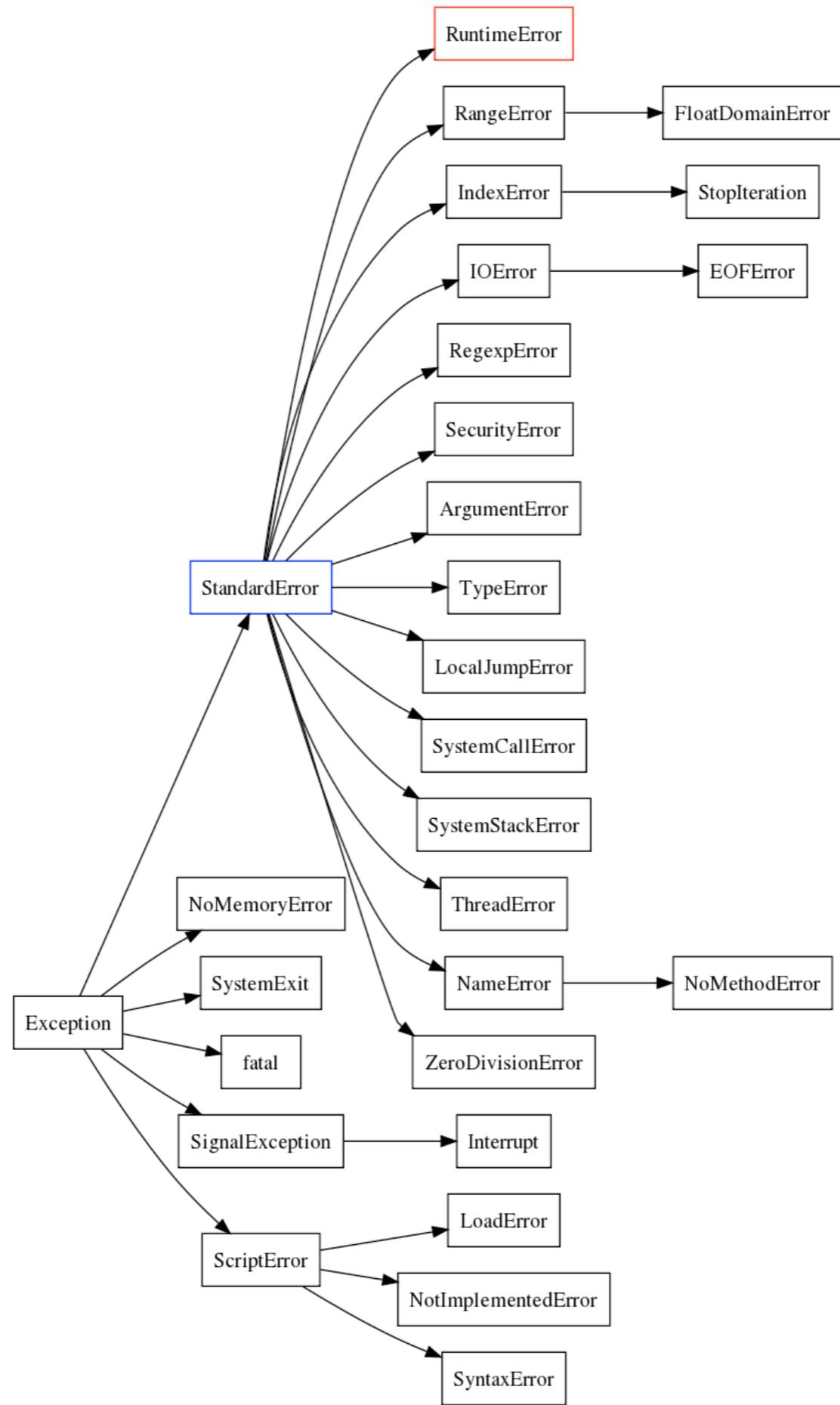
# Graph

```
sudo gem install graph
```

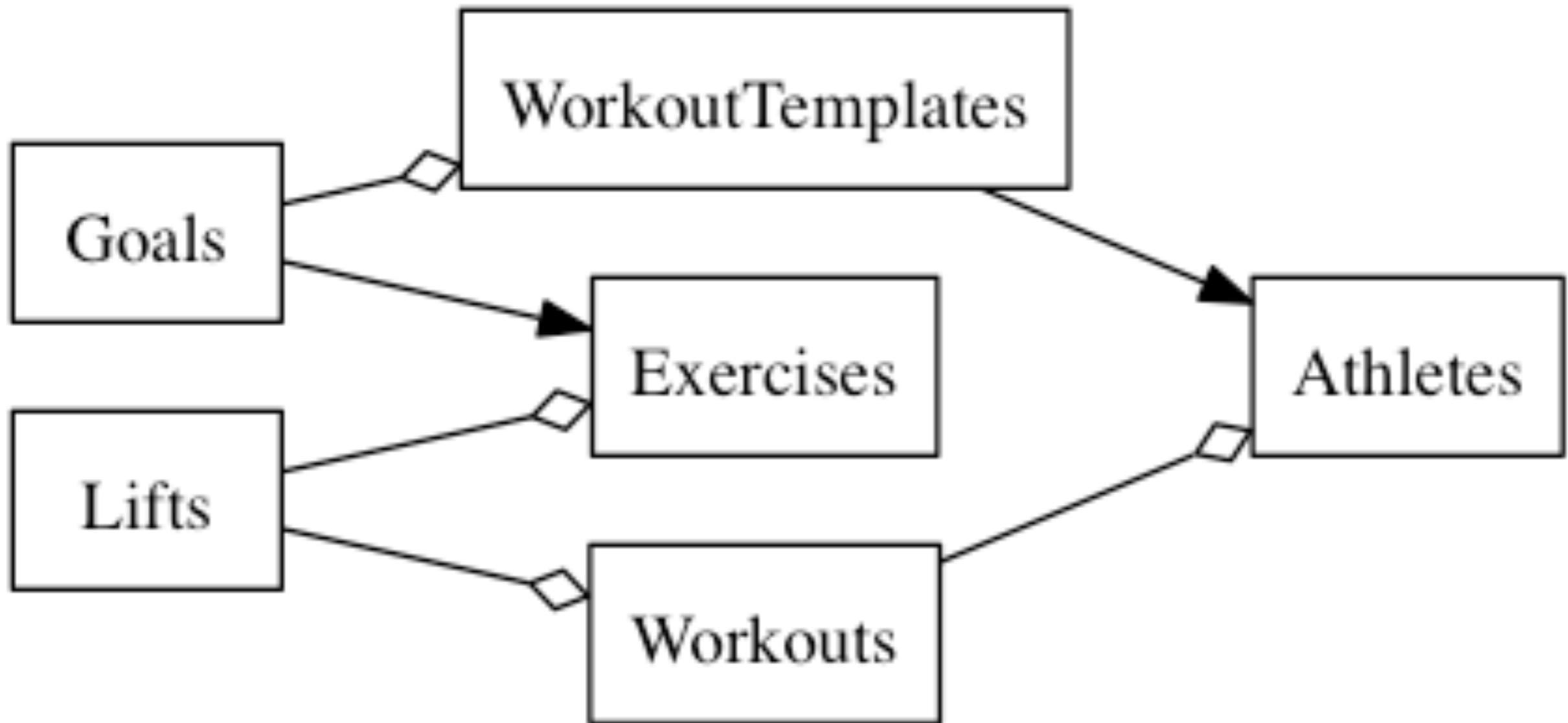
```
brew install graphviz
```

# Examples

# Ruby Exceptions



# Rails Associations



# CS Curriculum



# DOT

- Simple language to describe graphs
- Graphs are nodes and edges
- Can edit attributes such as color and shape

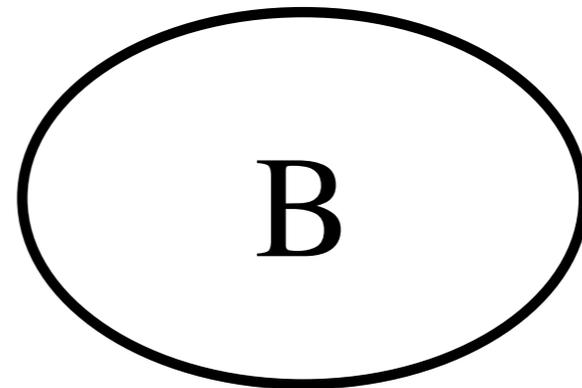
# Viewing DOT files

- GraphViz
- Tulip

# Basics

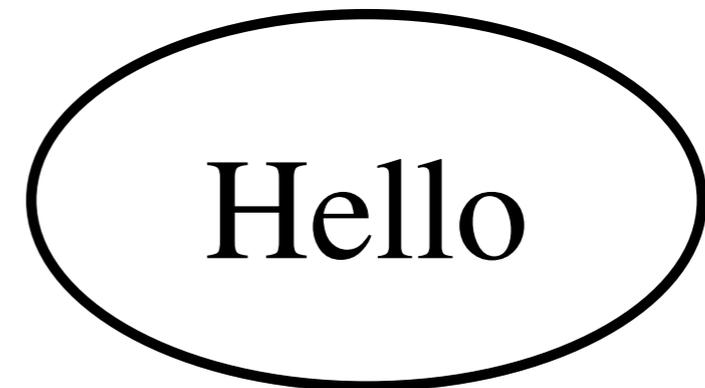
# Single Node

```
digraph do  
  node("B")  
end
```



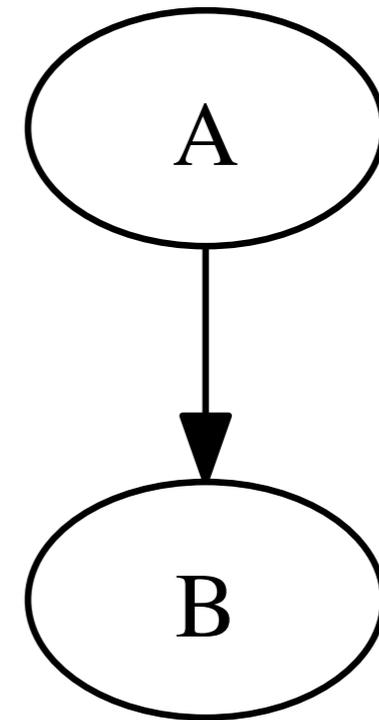
# Labels

```
digraph do  
  node("B").label "Hello"  
end
```



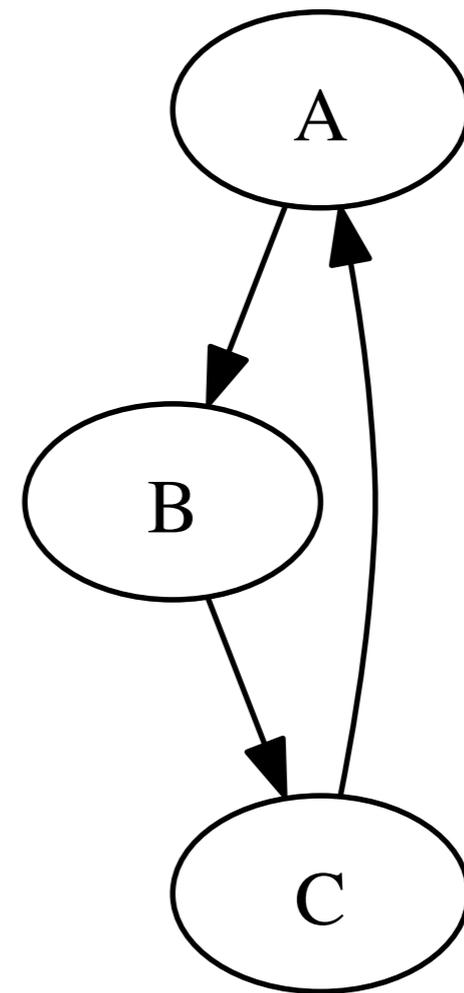
# Edges

```
digraph do  
  edge "A", "B"  
end
```



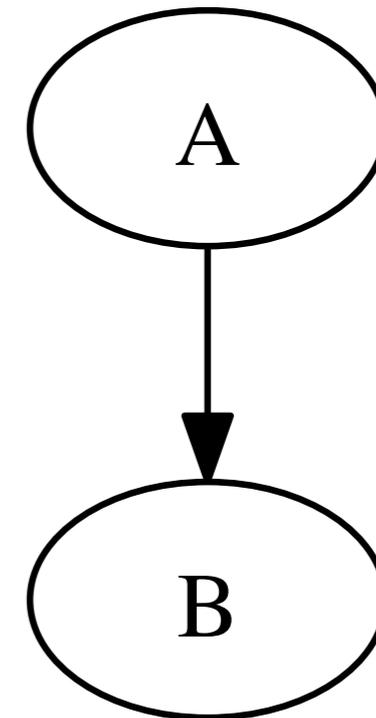
# Saving

```
digraph do
  edge "A", "B"
  edge "B", "C"
  edge "C", "A"
  save "cycle"
end
```



# Exporting

```
digraph do
  edge "a", "b"
  save "edge", "png"
  save "edge", "jpg"
  save "edge", "pdf"
  save "edge", "svg"
end
```



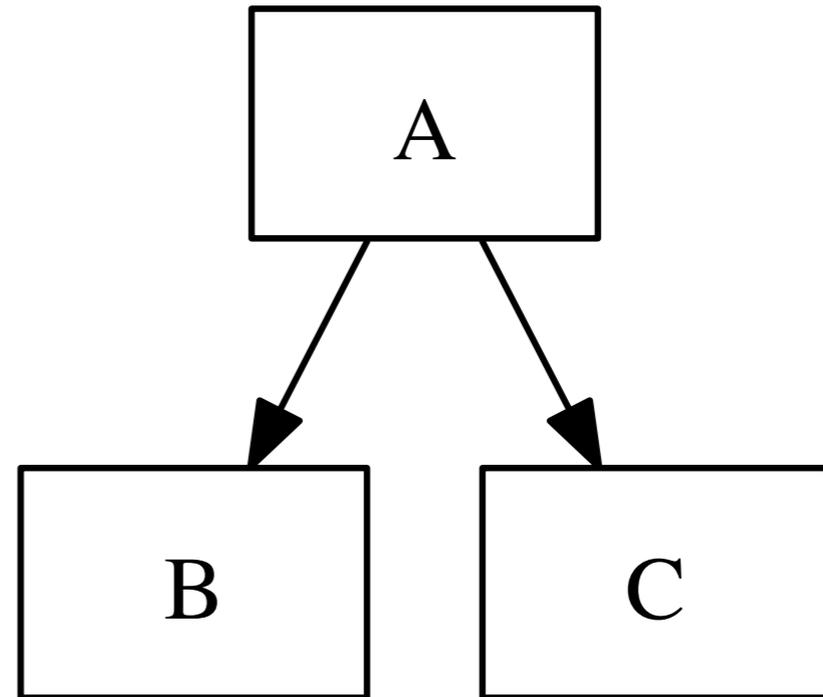
**Format list:** <http://www.graphviz.org/doc/info/output.html>

# Advanced

# Shapes

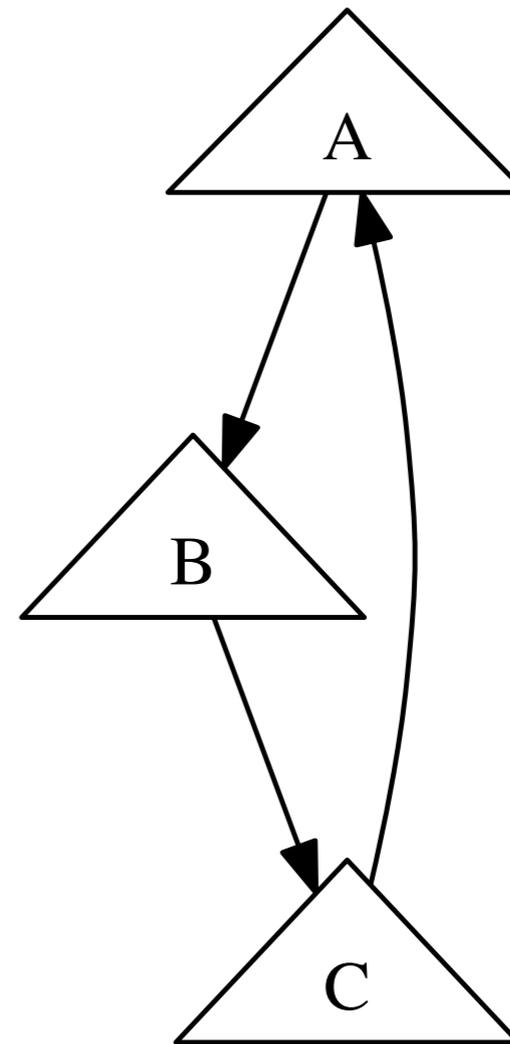
# Boxes

```
digraph do
  boxes
  edge "A", "B"
  edge "A", "C"
end
```



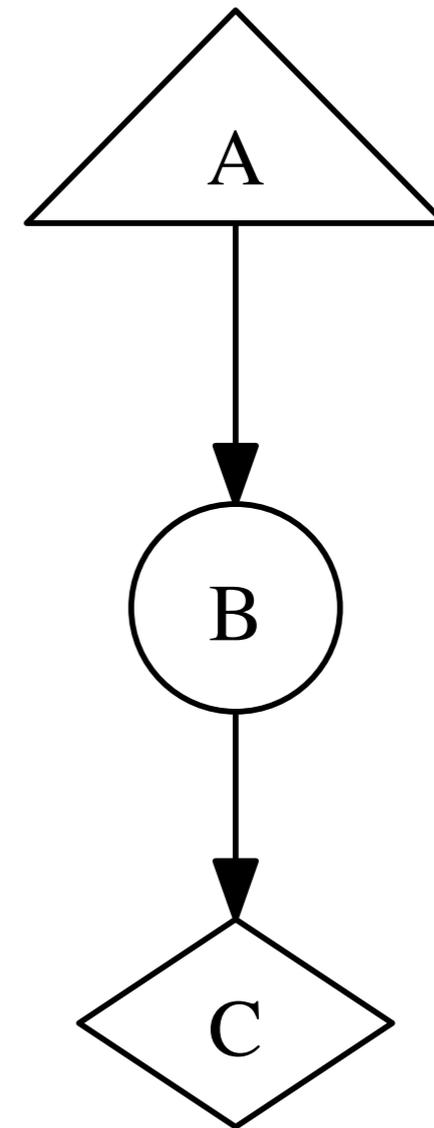
# Shapes

```
digraph do
  node_attribs <<
    triangle
  edge "A", "B"
  edge "B", "C"
  edge "C", "A"
end
```



# Many Shapes

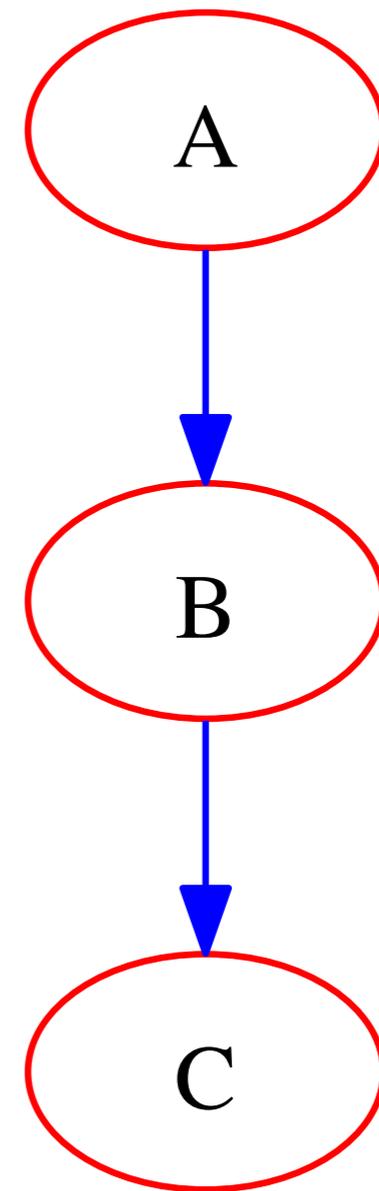
```
digraph do
  edge "A", "B", "C"
  triangle << node("A")
  circle << node("B")
  diamond << node("C")
end
```



# Color

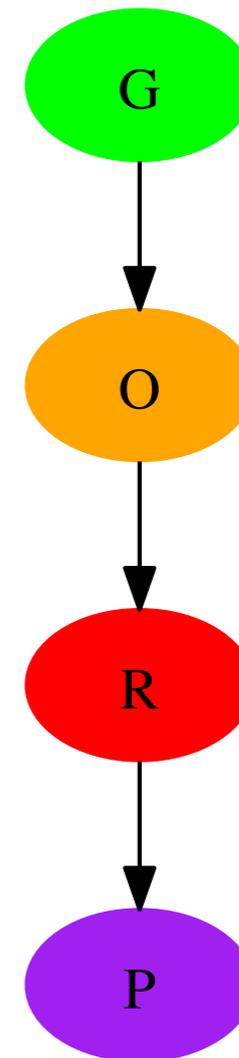
# Color Edges & Nodes

```
digraph do
  node_attribs << red
  edge_attribs << blue
  edge "A", "B", "C"
end
```



# Many Colors

```
digraph do
  node_attribs << filled
  edge "G", "O", "R", "P"
  green << node("G")
  orange << node("O")
  red << node("R")
  purple << node("P")
end
```



# Design Impaired?

[www.colorbrewer2.com](http://www.colorbrewer2.com)

Colorbrewer: Color Advice for Maps

http://www.colorbrewer2.com/

number of data classes on your map: **9** [learn more >](#)

the nature of your data: **diverging** [learn more >](#)

pick a color scheme: RdYlBu

(optional) only show schemes that are:

colorblind safe     print friendly  
 photocopy-able    [learn more >](#)

pick a color system

RGB     CMYK     HEX

adjust map context

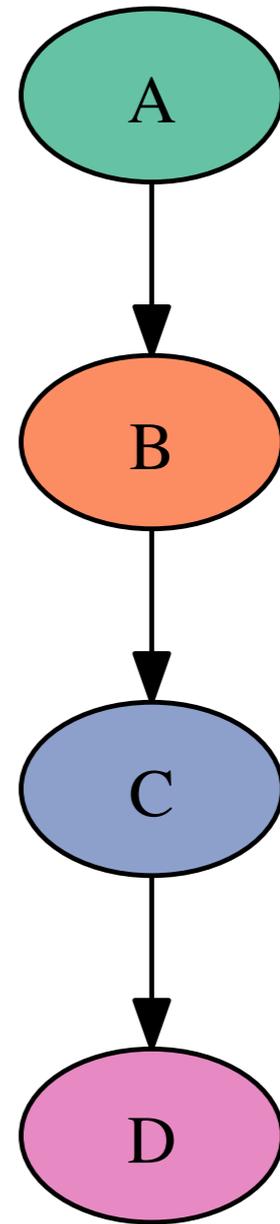
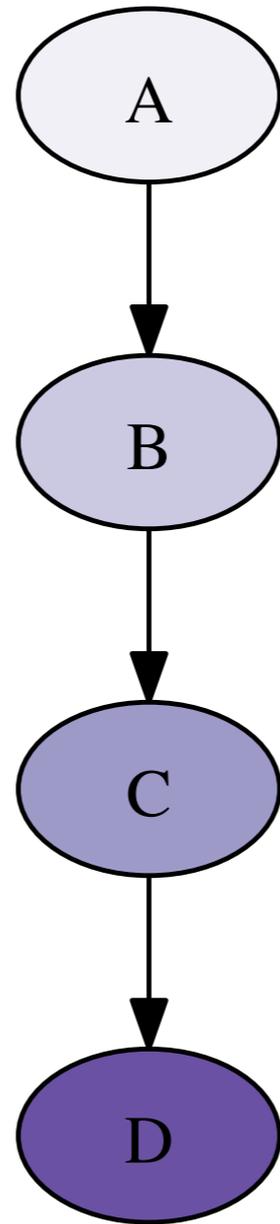
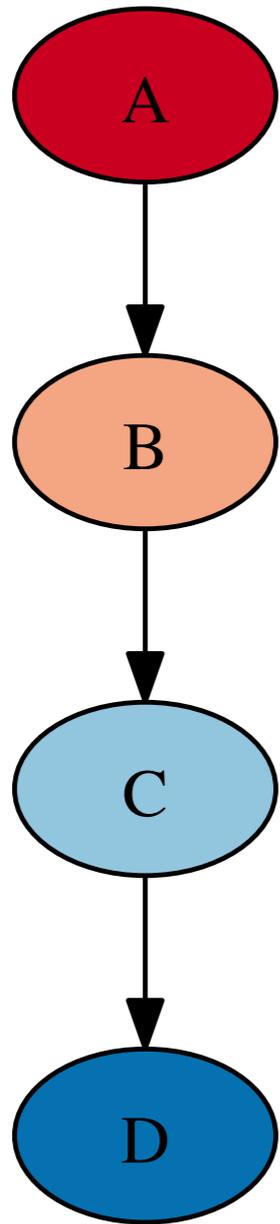
roads     cities     borders

Double check what color brewer calls the three categories

[www.graphviz.org/doc/info/  
colors.html](http://www.graphviz.org/doc/info/colors.html)

# Color Scheme Example

```
digraph do
  node_attribs << filled
  colorscheme(:set1, 4)
  c1 << node("A")
  c2 << node("B")
  c3 << node("C")
  c4 << node("D")
  edge "A", "B", "C", "D"
end
```

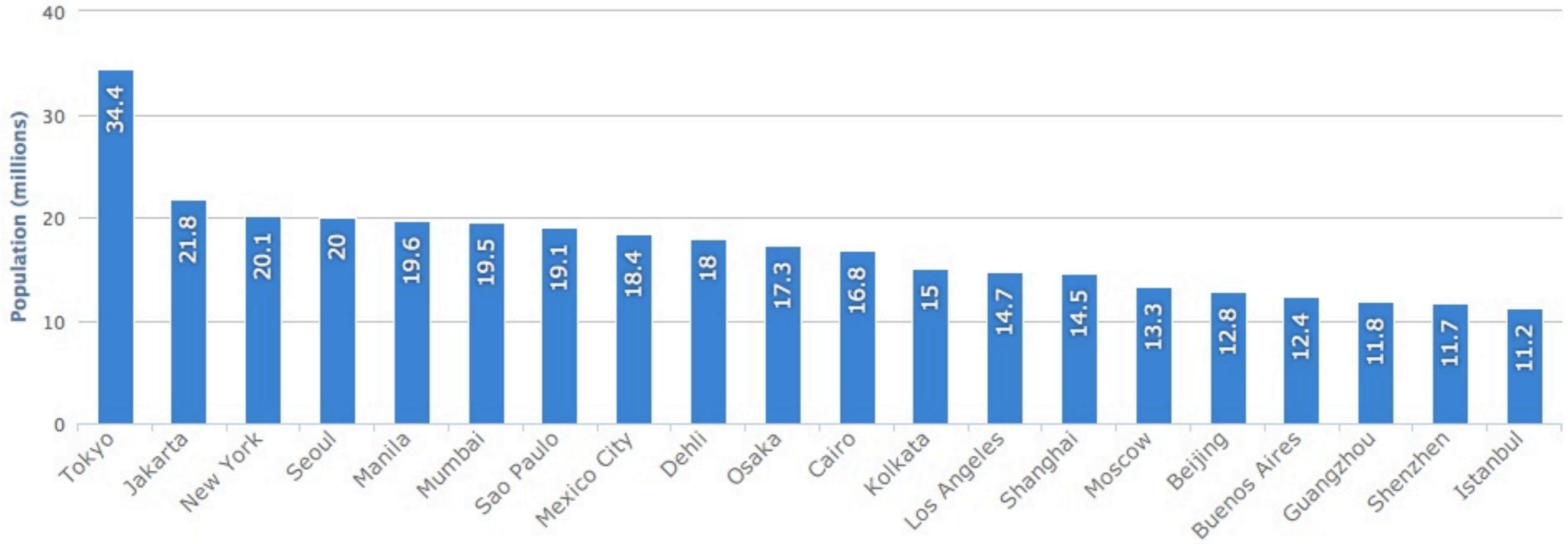


# Charts

# Examples



### World's largest cities per 2008



# Highcharts 101

# Basic Highcharts

## Page

# Chart title



Highcharts.com

# Include

JQuery  
Highcharts.js

```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script>
$(function () {
    $('#chart').highcharts({
        xAxis: {
            categories: ['emacs', 'vim', 'other']
        },
        series: [{
            data: [10, 5, 3]
        }]
    });
});
</script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<div id="chart"/>
</html>
```

**<html>**

```
<script type="text/javascript" src="http://  
ajax.googleapis.com/ajax/libs/jquery/1.7.1/  
jquery.min.js"></script>
```

```
<script>
```

```
$(function () {  
    $('#chart').highcharts({  
        xAxis: {  
            categories: ['emacs', 'vim', 'other']  
        },  
        series: [{  
            data: [10, 5, 3]  
        }]  
    });  
});
```

```
</script>
```

```
<script src="http://code.highcharts.com/  
highcharts.js"></script>
```

```
<div id="chart"/>
```

**</html>**

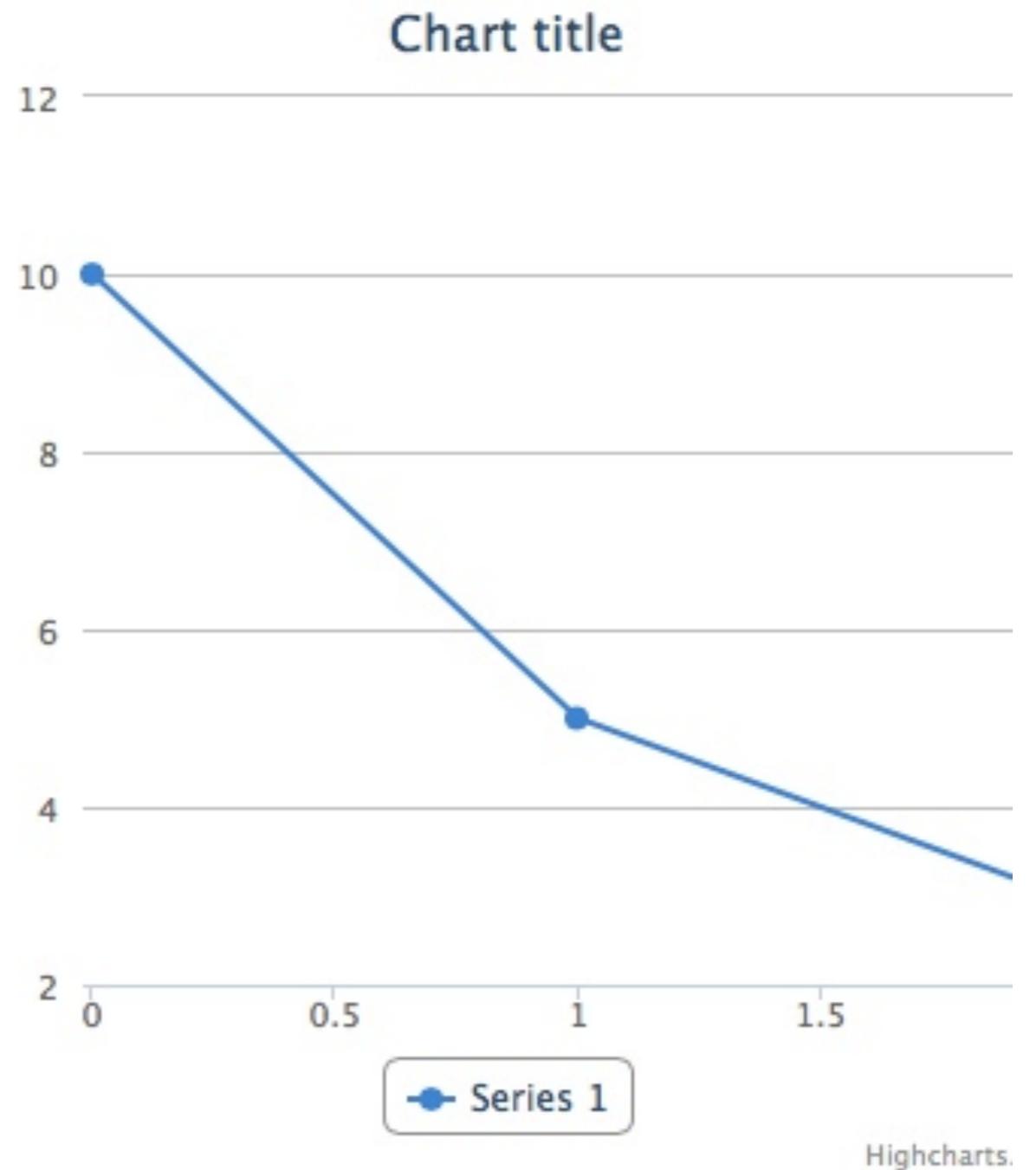
```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script>
$(function () {
    $('#chart').highcharts({
        xAxis: {
            categories: ['emacs', 'vim', 'other']
        },
        series: [{
            data: [10, 5, 3]
        }]
    });
});
</script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<div id="chart"/>
</html>
```

```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script>
$(function () {
    $('#chart').highcharts({
        xAxis: {
            categories: ['emacs', 'vim', 'other']
        },
        series: [{
            data: [10, 5, 3]
        }]
    });
});
</script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<div id="chart"/>
</html>
```

```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script>
$(function () {
    $('#chart').highcharts({
        xAxis: {
            categories: ['emacs', 'vim', 'other']
        },
        series: [{
            data: [10, 5, 3]
        }]
    });
});
</script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<div id="chart"/>
</html>
```

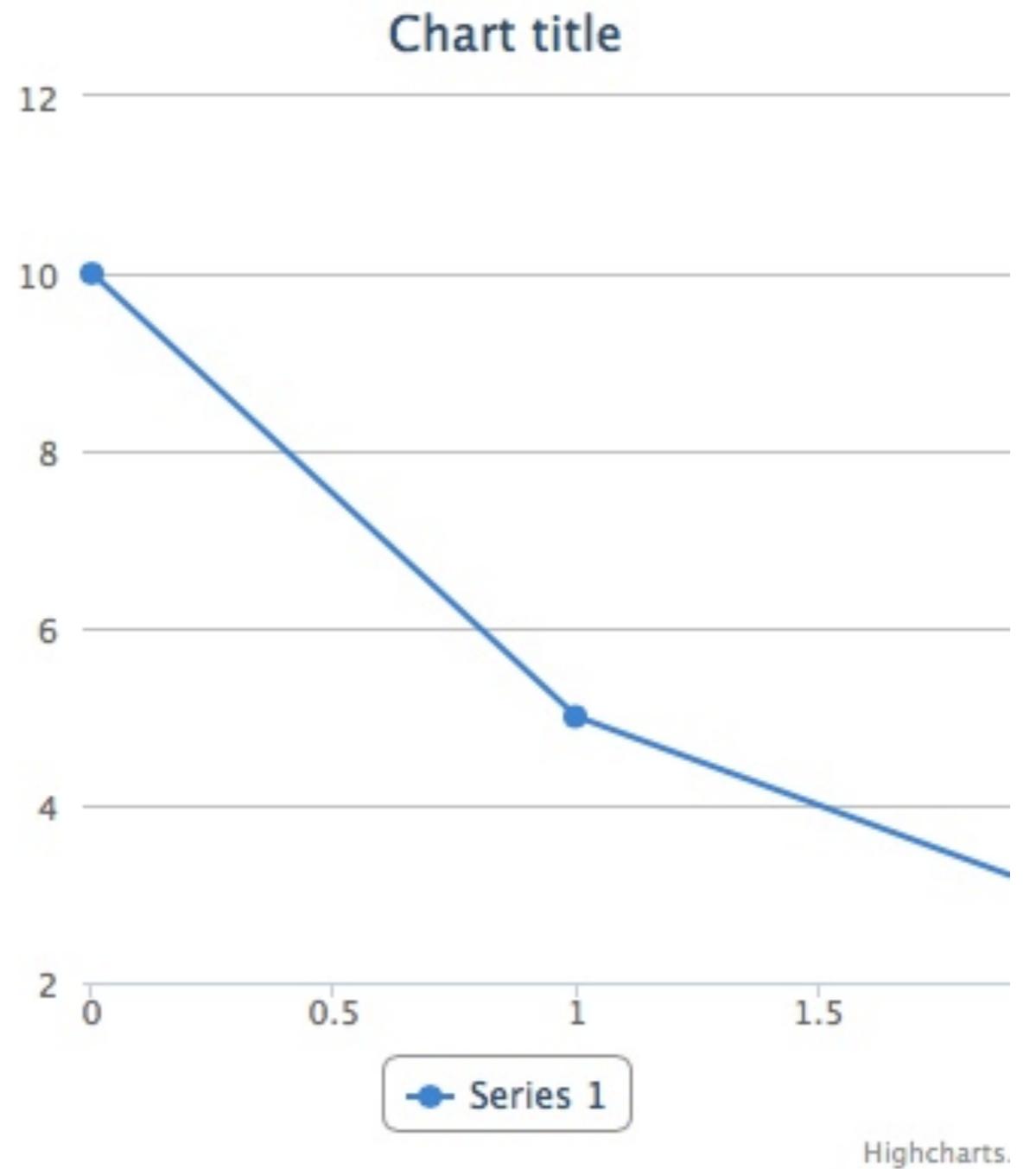
```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script>
$(function () {
    $('#chart').highcharts({
        xAxis: {
            categories: ['emacs', 'vim', 'other']
        },
        series: [{
            data: [10, 5, 3]
        }]
    });
});
</script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<div id="chart"/>
</html>
```

```
$(function () {  
  $('#chart').highcharts({  
    series: [{  
      data: [10, 5, 3]  
    }]  
  });  
});
```

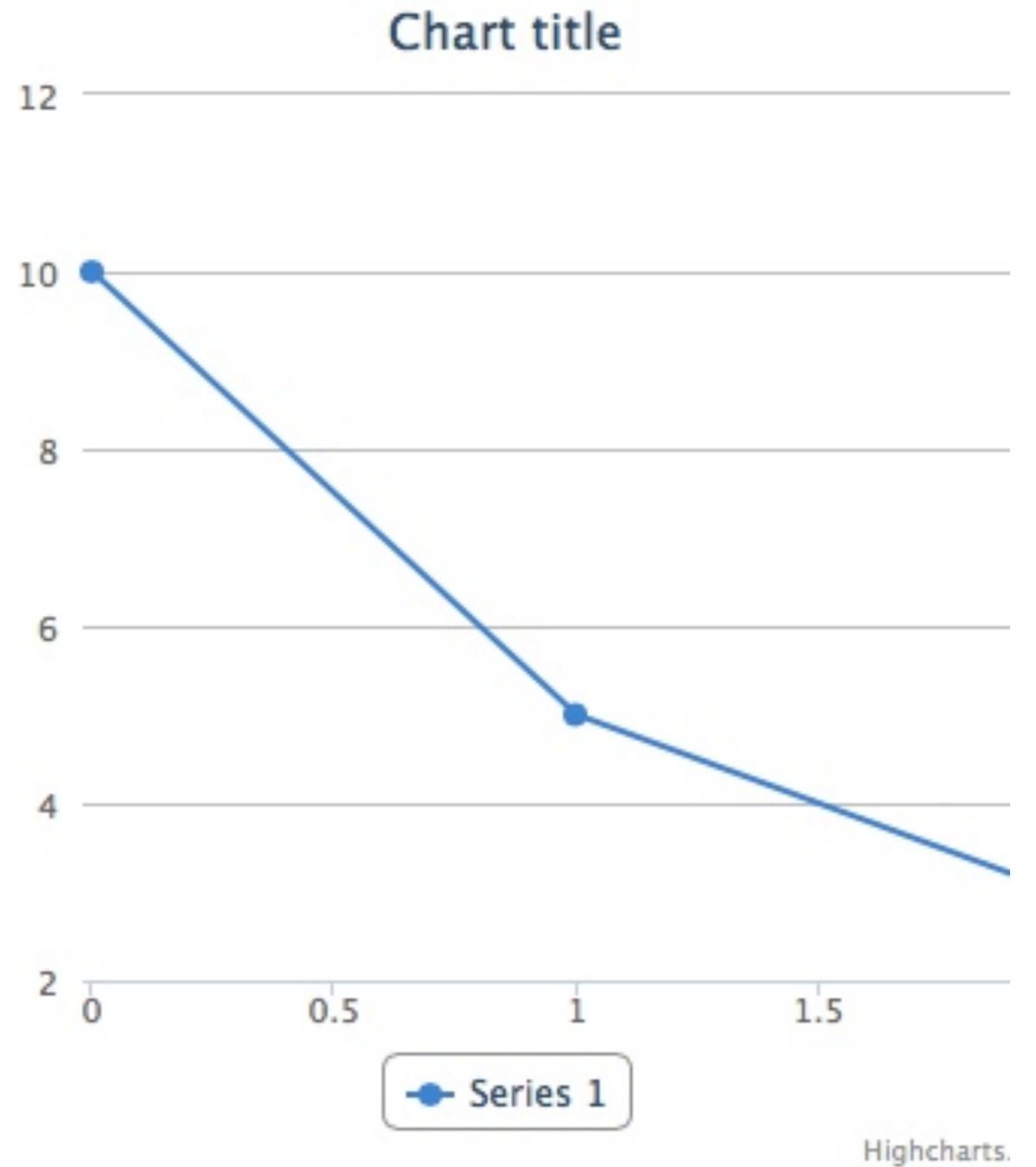


Highcharts.

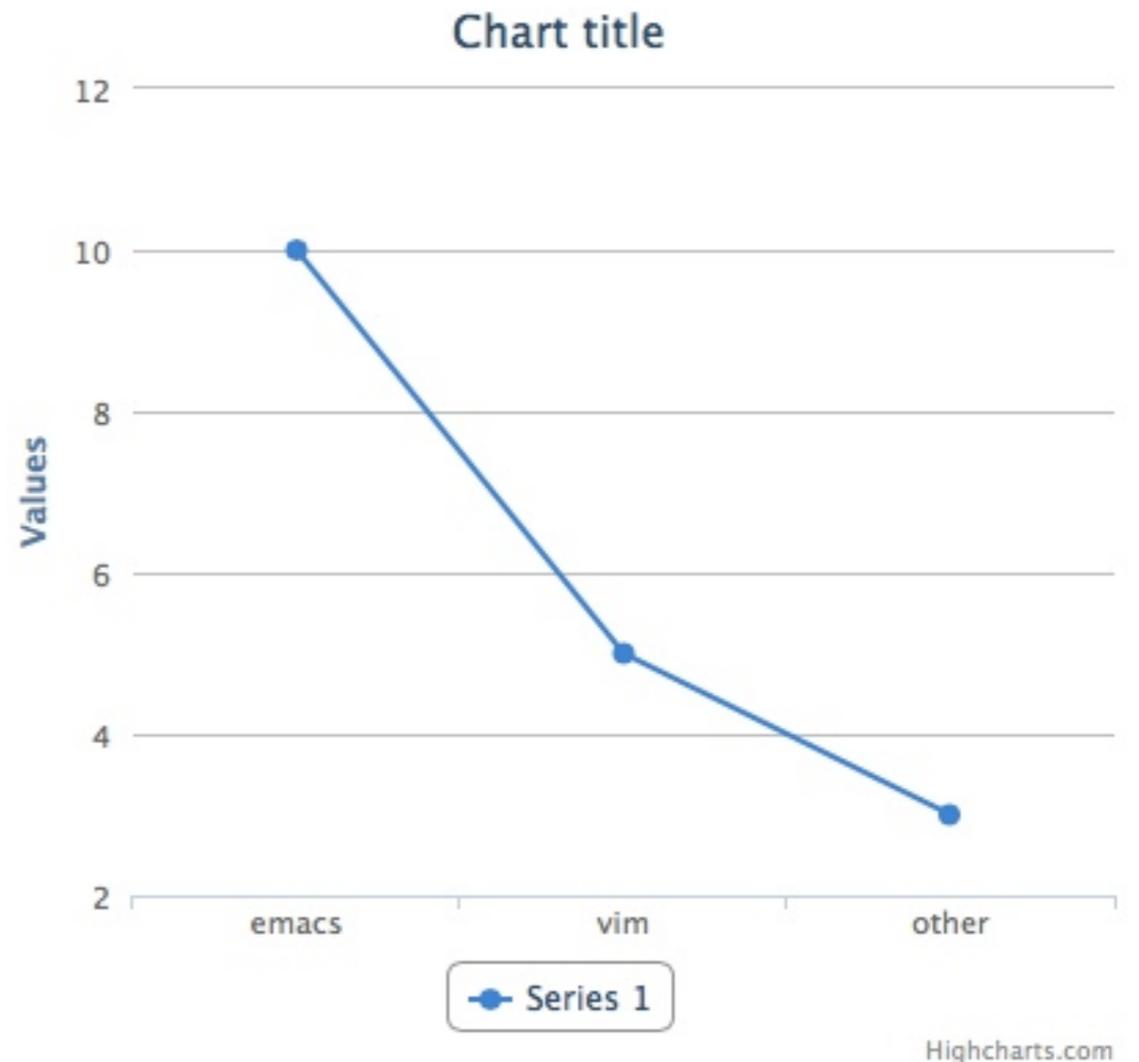
```
$(function () {  
  $('#chart').highcharts({  
    series: [{  
      data: [10, 5, 3]  
    }]  
  });  
});
```



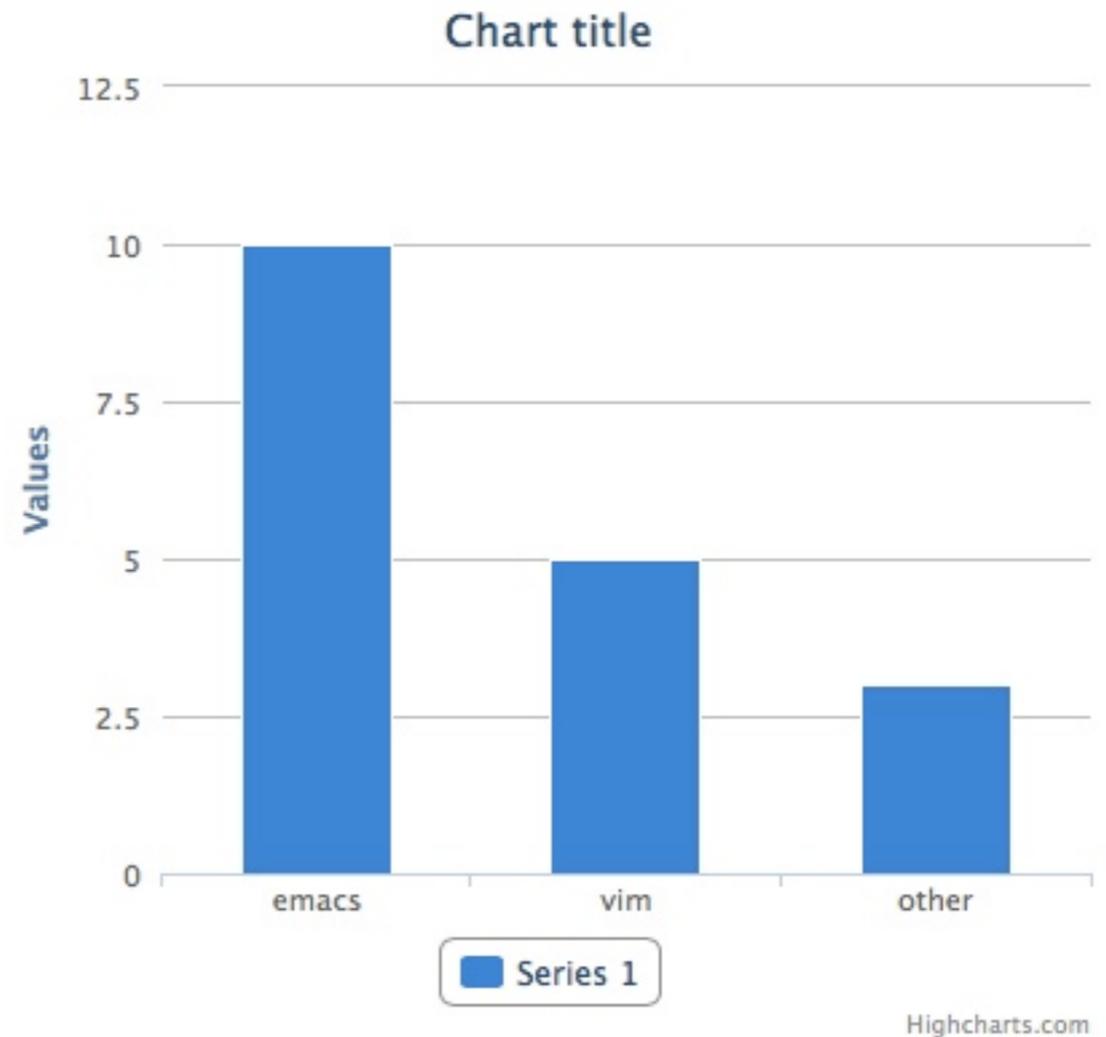
```
$(function () {  
  $('#chart').highcharts({  
    series: [{  
      data: [10, 5, 3]  
    }]  
  });  
});
```



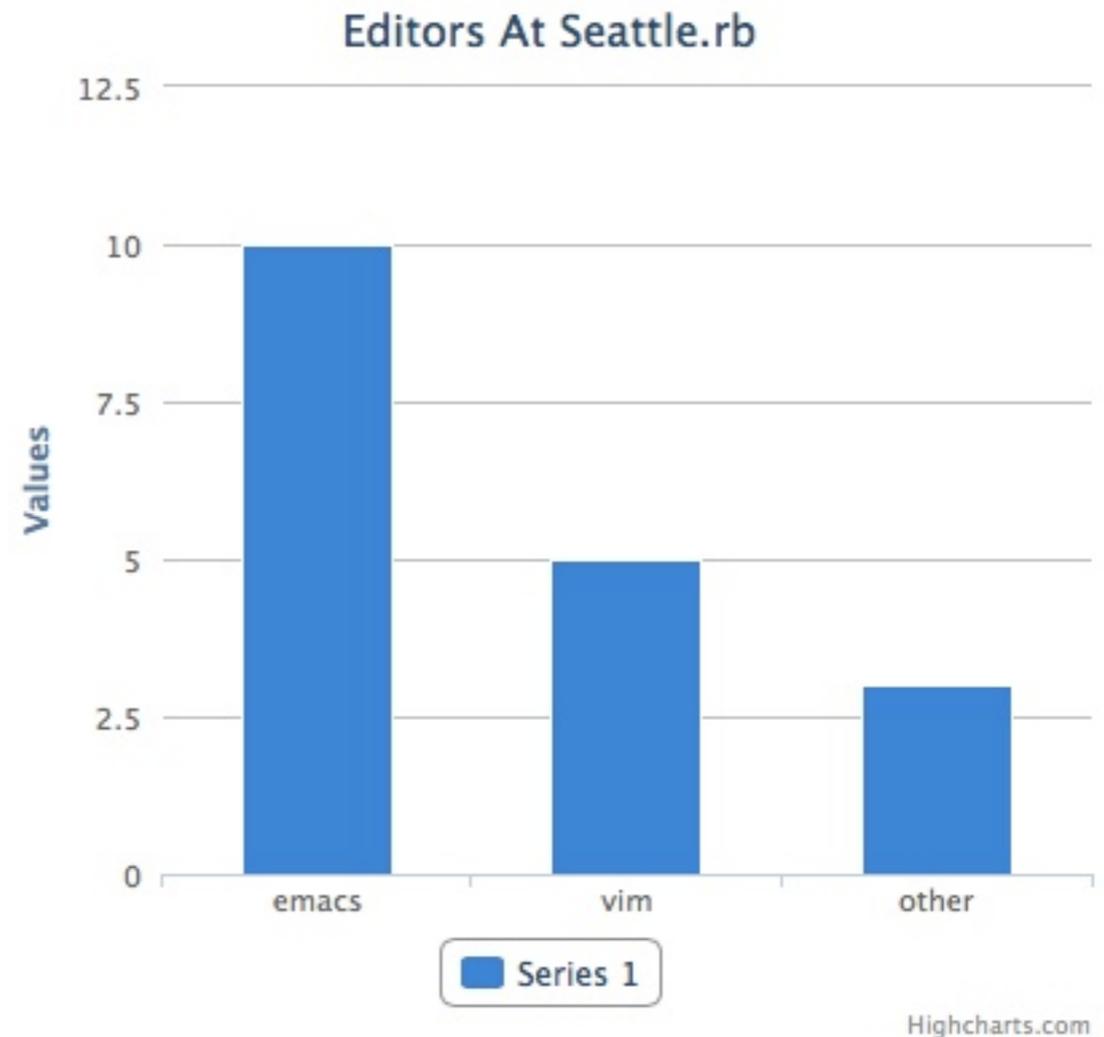
```
$(function () {  
  $('#chart').highcharts({  
    xAxis : {  
      categories:  
      ["emacs",  
       "vim",  
       "other"]  
    },  
    series: [{  
      data: [10, 5, 3]  
    }]  
  });  
});
```



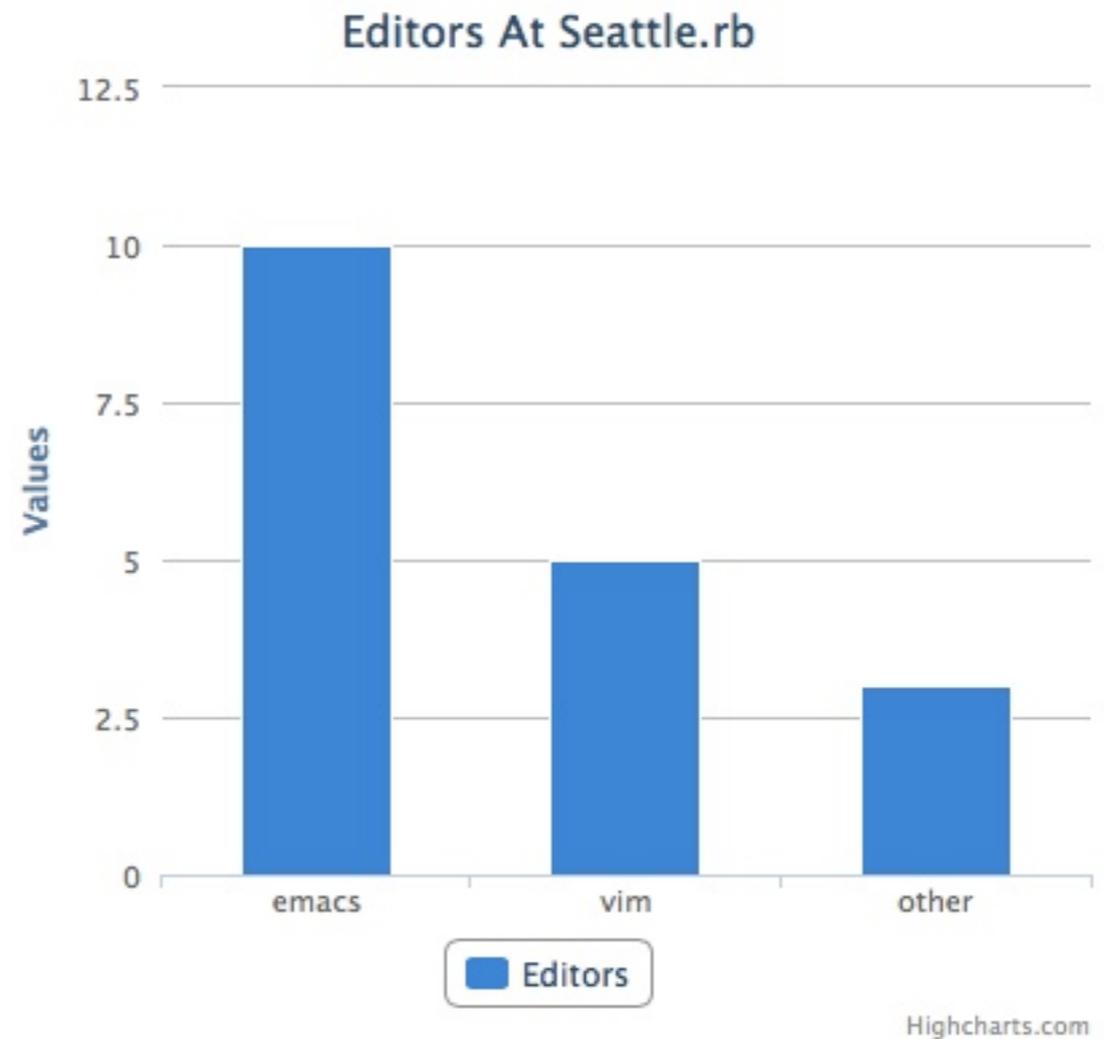
```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    xAxis : {
      categories:
        ["emacs",
         "vim",
         "other"]
    },
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



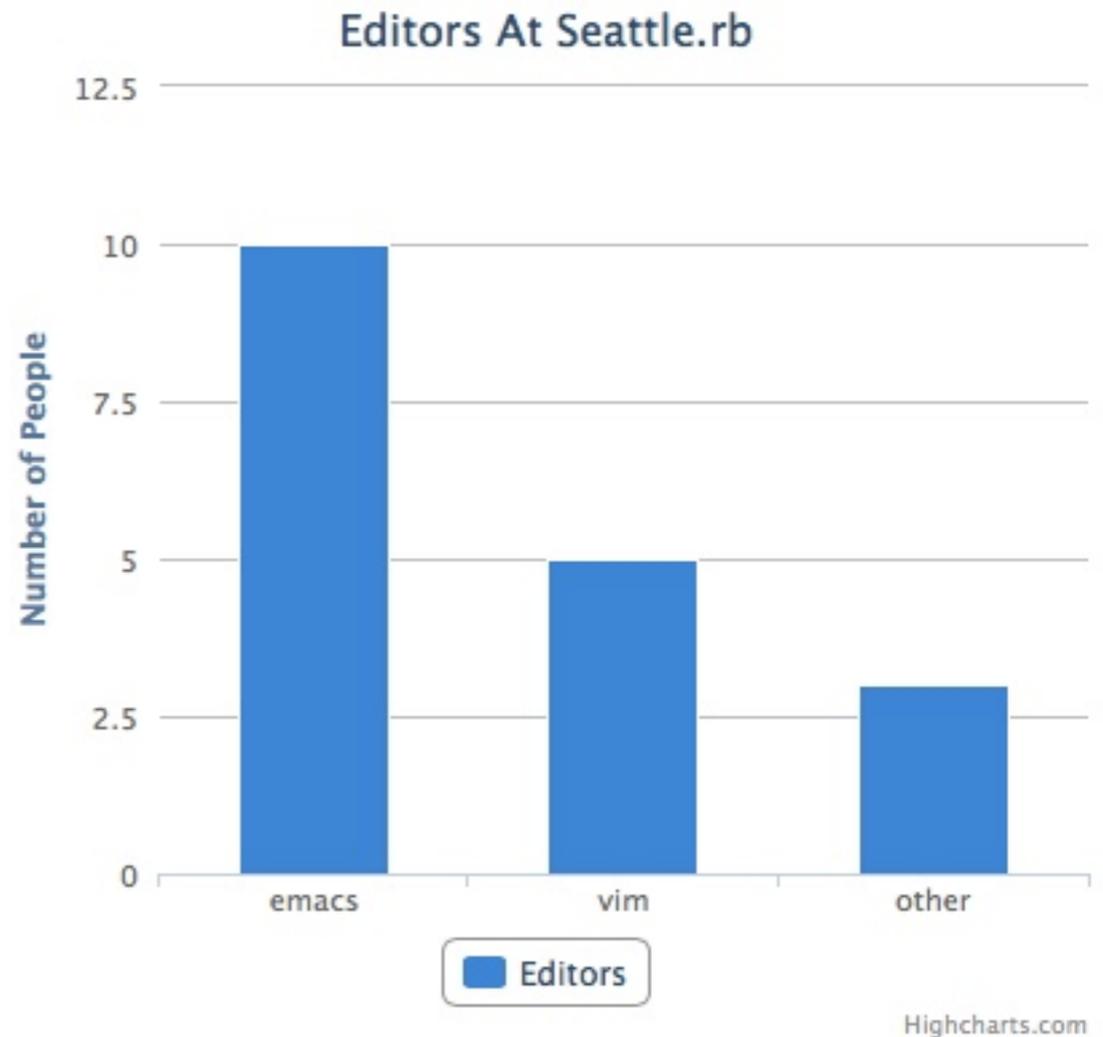
```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    title: {
      text: 'Editors At
Seattle.rb'
    },
    xAxis : {
      categories:
        ["emacs",
         "vim",
         "other"]
    },
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    title: { ... },
    xAxis : { ... },
    series: [{
      name: "Editors",
      data: [10, 5, 3]
    }]
  });
});
```

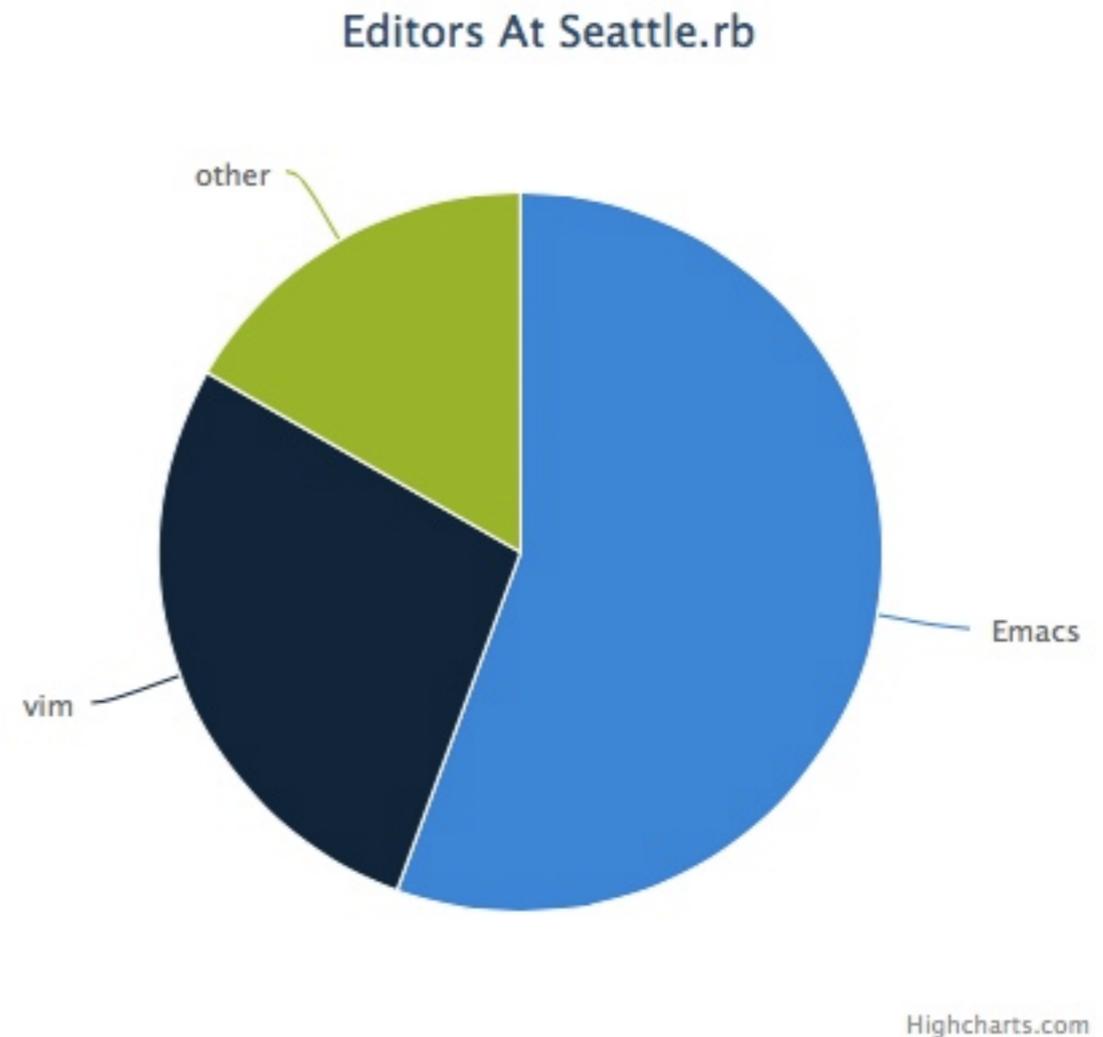


```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    title: { ... },
    yAxis : {
      title: {
text: "Number of People" }
      },
    xAxis : { ... },
    series: [{
      name: "Editors",
      data: [10, 5, 3]
    }]
  });
});
```



# Highcharts 102

```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'pie'
    },
    title: { ... },
    series: [{
      name: "Editors",
      data: [
        ["Emacs", 10],
        ["vim", 5],
        ["other", 3]
      ]
    }]
  });
});
```



line

spline

area

column

bar

pie

scatter

area range

area spline

column range



```
yAxis : [  
  { // Primary yAxis  
    labels : { style : { color : '#6666ff' },  
              align : 'left', x : 0, y : -2 },  
    title : { text : 'Weight',  
              style : { color : '#6666ff' } }  
  },  
  { // Secondary yAxis  
    labels : { style : { color : '#33aa33' },  
              align : 'right', x : 0, y : -2 },  
    title : { text : 'Time',  
              style : { color : '#33aa33' } },  
    opposite : true  
  }  
]
```

```
yAxis : [  
  {// Primary yAxis  
    labels : {style : {color : '#6666ff'},  
              align : 'left', x : 0, y : -2},  
    title : {text : 'Weight',  
             style : {color : '#6666ff'}}  
  },  
  {// Secondary yAxis  
    labels : {style : { color : '#33aa33' },  
              align : 'right', x : 0, y : -2},  
    title : {text : 'Time',  
             style : { color : '#33aa33' }},  
    opposite : true  
  }  
]
```

```
yAxis : [  
  { // Primary yAxis  
    labels : { style : { color : '#6666ff' },  
              align : 'left', x : 0, y : -2 },  
    title : { text : 'Weight',  
              style : { color : '#6666ff' } }  
  },  
  { // Secondary yAxis  
    labels : { style : { color : '#33aa33' },  
              align : 'right', x : 0, y : -2 },  
    title : { text : 'Time',  
              style : { color : '#33aa33' } },  
    opposite : true  
  }  
]
```

# Data Extraction

# Techniques

- e
- le
- act Info
- emic  
rams
- Major
- Math Major
- Math/Bio Major
- Minor
- cies
- urse  
criptions
- ple Schedules
- urse Precedence
- dy Abroad
- puter  
ineering
- s, Innovations,  
lts
- s Schedule
- c
- posium
- arch

# Course Descriptions

This is a listing of the all the courses we offer. For a graphical depiction of course precedence, click [here](#). To see the courses we are offering in the current semester, visit [this page](#).

## CS 5. Introduction to Computer Science

**Prerequisites** Permission by instructor.

**Credit Hours** 3.0

**Offered** Fall semester.

Introduction to elements of computer science. Students learn general computational problem-solving techniques and gain experience with the design, implementation, testing and documentation of programs in a high-level language. In addition, students learn to design digital devices, understand how computers work, and learn to program a computer in its own machine language. Finally, students are exposed to ideas in computability theory. The course includes discussions of societal and ethical issues related to computer science.

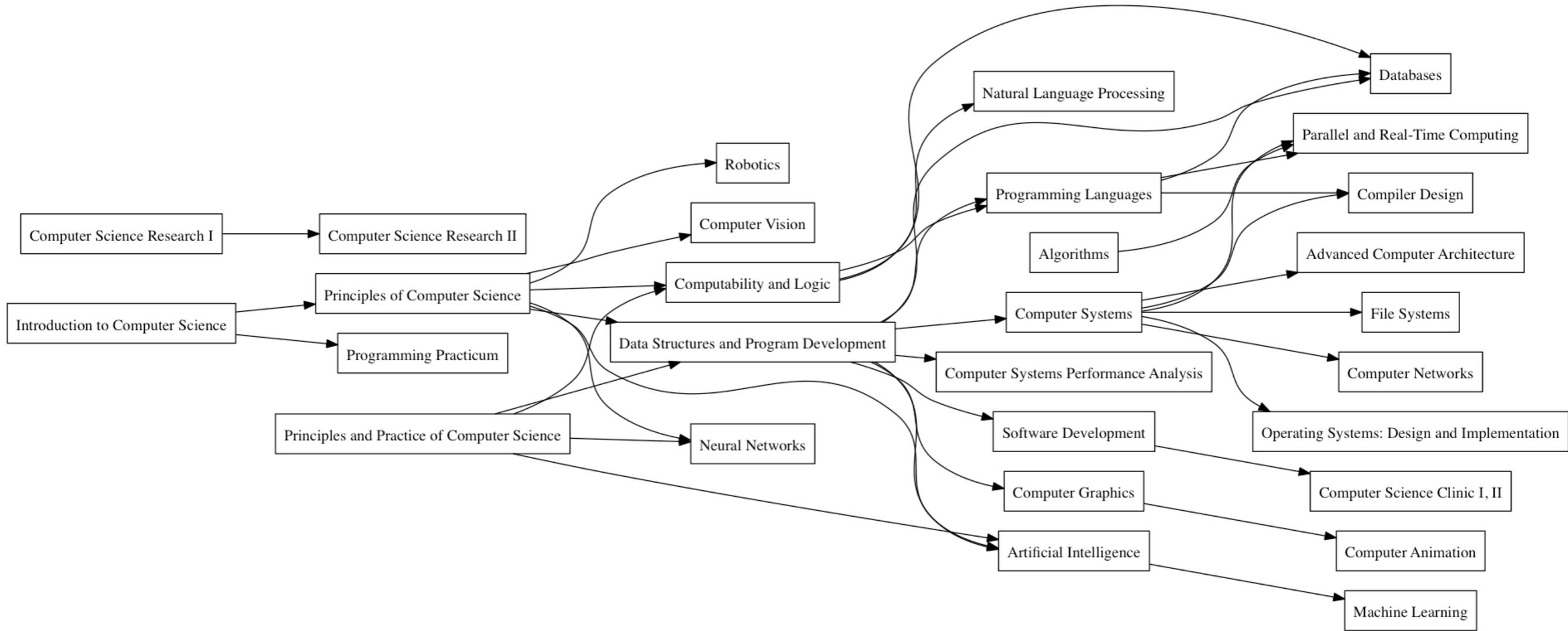
## CS 5GR. Introduction to Biology and Computer Science

**Prerequisites** Permission of instructor

**Credit Hours** 3.0

**Offered** Fall semester.

This course introduces fundamental concepts from the core course Computer Science 5 using biology as the context for those computational ideas. Students see both the intellectual and practical connections between these two disciplines and write computer programs to explore biological phenomena. Biology topics include the basics of biochemistry, the central dogma, population genetics, molecular evolution, metabolism, regulation, and phylogenetics. Computer science material includes basic data types and control structures, recursion, dynamic programming, and an introduction to automata and computability. This course fulfills the computer science core requirement at Harvey Mudd College. It does not fulfill the HMC biology core requirement.



```

#!/usr/bin/ruby -w
require 'rubygems'
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

courses = []

course_descriptions = doc.css(".crsdscrptn-header")

course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-title").text.split('.')

  course[:number] = normalize(number)
  course[:title]  = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end

digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end

```

```

#!/usr/bin/ruby -w
require 'rubygems'
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

courses = []

course_descriptions = doc.css(".crsdscrptn-header")

course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-title").text.split('.')

  course[:number] = normalize(number)
  course[:title]  = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end

digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end

```

```
require 'graph'  
require 'nokogiri'  
require 'open-uri'
```

```
uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"  
doc      = Nokogiri::HTML(open(uri))
```

```
def normalize(number)  
  number.strip.gsub(/\s/, '')  
end
```

```
courses = []
```

```
course_descriptions = doc.css(".crsdscrptn-header")
```

```
require 'graph'  
require 'nokogiri'  
require 'open-uri'
```

```
uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"  
doc      = Nokogiri::HTML(open(uri))
```

```
def normalize(number)  
  number.strip.gsub(/\s/, '')  
end
```

```
courses = []
```

```
course_descriptions = doc.css(".crsdscrptn-header")
```

```
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

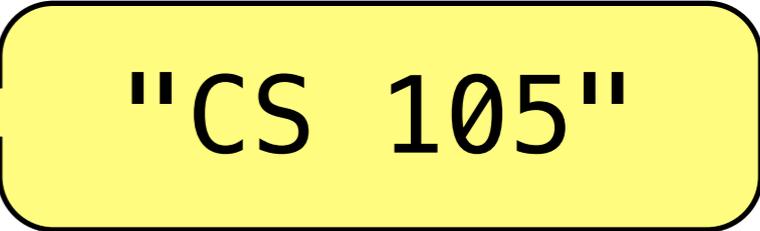
courses = []

course_descriptions = doc.css(".crsdscrptn-header")
```

```
require 'graph'  
require 'nokogiri'  
require 'open-uri'
```

```
uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"  
doc      = Nokogiri::HTML(open(uri))
```

```
def normalize(number)  
  number.strip.gsub(/\s/, '')  
end
```



"CS 105"

```
courses = []
```

```
course_descriptions = doc.css(".crsdscrptn-header")
```

```
require 'graph'  
require 'nokogiri'  
require 'open-uri'
```

```
uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"  
doc      = Nokogiri::HTML(open(uri))
```

```
def normalize(number)  
  number.strip.gsub(/\s/, '')  
end
```



"CS105"

```
courses = []
```

```
course_descriptions = doc.css(".crsdscrptn-header")
```

```
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-  
descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

courses = []

course_descriptions = doc.css(".crsdscrptn-header")
```

- e
- le
- act Info
- emic
- rams
- Major
- Math Major
- Math/Bio Major
- Minor
- cies
- urse
- criptions
- ple Schedules
- urse Precedence
- dy Abroad
- puter
- ineering
- s, Innovations,
- Its
- s Schedule
- c
- posium
- arch

# Course Descriptions

This is a listing of the all the courses we offer. For a graphical depiction of course precedence, click [here](#). To see the courses we are offering in the current semester, visit [this page](#).

## CS 5. Introduction to Computer Science

**Prerequisites** Permission by instructor.

**Credit Hours** 3.0

**Offered** Fall semester.

Introduction to elements of computer science. Students learn general computational problem-solving techniques and gain experience with the design, implementation, testing and documentation of programs in a high-level language. In addition, students learn to design digital devices, understand how computers work, and learn to program a computer in its own machine language. Finally, students are exposed to ideas in computability theory. The course includes discussions of societal and ethical issues related to computer science.

## CS 5GR. Introduction to Biology and Computer Science

**Prerequisites** Permission of instructor

**Credit Hours** 3.0

**Offered** Fall semester.

This course introduces fundamental concepts from the core course Computer Science 5 using biology as the context for those computational ideas. Students see both the intellectual and practical connections between these two disciplines and write computer programs to explore biological phenomena. Biology topics include the basics of biochemistry, the central dogma, population genetics, molecular evolution, metabolism, regulation, and phylogenetics. Computer science material includes basic data types and control structures, recursion, dynamic programming, and an introduction to automata and computability. This course fulfills the computer science core requirement at Harvey Mudd College. It does not fulfill the HMC biology core requirement.

- e
- le
- act Info
- emic
- rams
- Major
- Math Major
- Math/Bio Major
- Minor
- cies
- urse
- criptions
- ple Schedules
- urse Precedence
- dy Abroad
- puter
- ineering
- s, Innovations,
- Its
- s Schedule
- c
- posium
- arch

## Course Descriptions

This is a listing of the all the courses we are offering in the current semester.

`".crsdscrip-headers"`

### CS 5. Introduction to Computer Science

**Prerequisites** Permission by instructor.  
**Credit Hours** 3.0  
**Offered** Fall semester.

Introduction to elements of computer science. Students learn general computational problem-solving techniques and gain experience with the design, implementation, testing and documentation of programs in a high-level language. In addition, students learn to design digital devices, understand how computers work, and learn to program a computer in its own machine language. Finally, students are exposed to ideas in computability theory. The course includes discussions of societal and ethical issues related to computer science.

### CS 5GR. Introduction to Biology and Computer Science

**Prerequisites** Permission of instructor  
**Credit Hours** 3.0  
**Offered** Fall semester.

This course introduces fundamental concepts from the core course Computer Science 5 using biology as the context for those computational ideas. Students see both the intellectual and practical connections between these two disciplines and write computer programs to explore biological phenomena. Biology topics include the basics of biochemistry, the central dogma, population genetics, molecular evolution, metabolism, regulation, and phylogenetics. Computer science material includes basic data types and control structures, recursion, dynamic programming, and an introduction to automata and computability. This course fulfills the computer science core requirement at Harvey Mudd College. It does not fulfill the HMC biology core requirement.

```

#!/usr/bin/ruby -w
require 'rubygems'
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

courses = []

course_descriptions = doc.css(".crsdscrptn-header")

course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-title").text.split('.')

  course[:number] = normalize(number)
  course[:title]  = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end

digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end

```

```
course_descriptions.each do |description|  
  course = Hash.new  
  
  number, title = description.css(".crsdscrptn-  
title").text.split('.')  
  
  course[:number] = normalize(number)  
  course[:title] = title.strip  
  
  desc = description.css("dl dd")[0].text  
  pre_reqs = desc.scan(/CS\s*\d+/)  
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }  
  
  courses << course  
end
```

```
course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-
title").text.split('.')

  course[:number] = normalize(number)
  course[:title] = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end
```

```
number, title = description.css(".crsdscrptn-  
title").text.split('.')
```

```
course[:number] = normalize(number)  
course[:title] = title.strip
```

"CS 105. Computer Systems"

```
number, title = description.css(".crsdscrptn-  
title").text.split('.')
```

```
course[:number] = normalize(number)  
course[:title] = title.strip
```

```
course_descriptions.each do |description|
  course = Hash.new

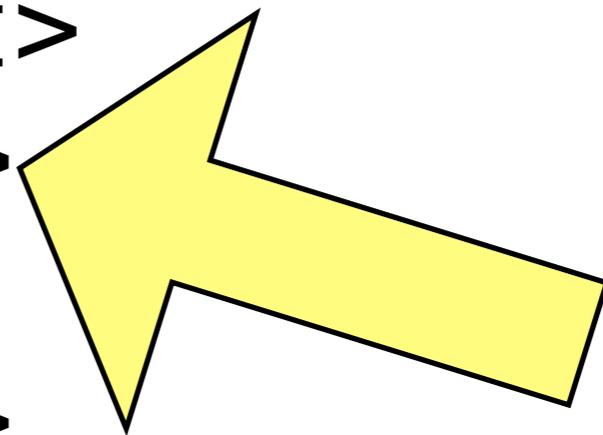
  number, title = description.css(".crsdscrptn-
title").text.split('.')

  course[:number] = normalize(number)
  course[:title] = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end
```

```
<dl>
  <dt>Prerequisites</dt>
  <dd>CS 42, CS 60</dd>
  <br />
  <dt>Credit Hours</dt>
  <dd>3.0</dd>
  <br />
</dl>
```



```
desc = description.css("dl dd")[0].text  
pre_reqs = desc.scan(/CS\s*\d+/  
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

"CS 42, CS 60"

```
desc = description.css("dl dd")[0].text  
pre_reqs = desc.scan(/CS\s*\d+/)  
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

```
desc = description.css("dl dd")[0].text
pre_reqs = desc.scan(/CS\s*\d+/)
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

`["CS 42", "CS 60"]`

```
desc = description.css("dl dd")[0].text  
pre_reqs = desc.scan(/CS\s*\d+/)  
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

```
desc = description.css("dl dd")[0].text
pre_reqs = desc.scan(/CS\s*\d+/)
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

```
desc = description.css("dl dd")[0].text
pre_reqs = desc.scan(/CS\s*\d+/)
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

**course[:pre\_req] = ["CS42", "CS60"]**

```
course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-
title").text.split('.')

  course[:number] = normalize(number)
  course[:title] = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end
```

```

#!/usr/bin/ruby -w
require 'rubygems'
require 'graph'
require 'nokogiri'
require 'open-uri'

uri      = "http://www.cs.hmc.edu/program/course-descriptions/"
doc      = Nokogiri::HTML(open(uri))

def normalize(number)
  number.strip.gsub(/\s/, '')
end

courses = []

course_descriptions = doc.css(".crsdscrptn-header")

course_descriptions.each do |description|
  course = Hash.new

  number, title = description.css(".crsdscrptn-title").text.split('.')

  course[:number] = normalize(number)
  course[:title]  = title.strip

  desc = description.css("dl dd")[0].text
  pre_reqs = desc.scan(/CS\s*\d+/)
  course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }

  courses << course
end

digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end

```

```
digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end
```

**digraph do**

```
rotate; boxes;
```

```
courses.each do |course|  
  node(course[:number]).label course[:title]  
end
```

```
courses.each do |course|  
  course[:pre_reqs].each do |pr|  
    edge pr , course[:number]  
  end  
end
```

```
orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}  
orphans.each { |name| remove_node(name) }
```

```
save "hmc_cs", "png"
```

**end**

```
digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end
```

```
digraph do
  rotate; boxes;

courses.each do |course|
  node(course[:number]).label course[:title]
end

courses.each do |course|
  course[:pre_reqs].each do |pr|
    edge pr , course[:number]
  end
end

orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
orphans.each { |name| remove_node(name) }

save "hmc_cs", "png"
end
```

```
digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end
```

```
courses.each do |course|
  course[:pre_reqs].each do |pr|
    edge pr , course[:number]
  end
end
```

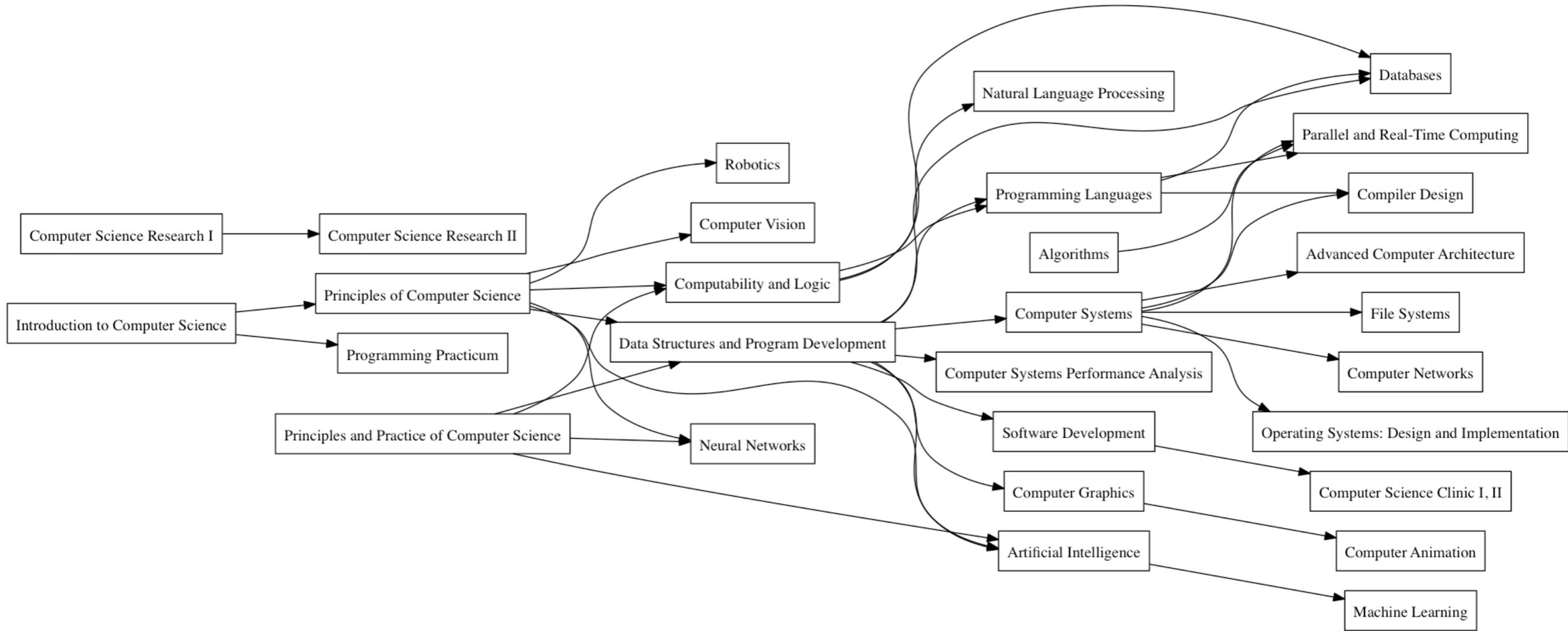
```
digraph do
  rotate; boxes;

  courses.each do |course|
    node(course[:number]).label course[:title]
  end

  courses.each do |course|
    course[:pre_reqs].each do |pr|
      edge pr , course[:number]
    end
  end

  orphans = nodes.map { |_, n| n.orphan? ? n.name : nil}
  orphans.each { |name| remove_node(name) }

  save "hmc_cs", "png"
end
```



# Special Thanks

- Ryan Davis for Graph & Highcharts assistance
- Aaron Patterson & Mike Dalessio for Nokogiri

**Thank You**