Seeing the Big Picture

Quick and Dirty Data Visualization with Ruby

Aja Hammerly

@thagomizer_rb

http://github.com/thagomizer

http://www.thagomizer.com

Motivations

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

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

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


Java™


Java™


ORACLE®

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

WAIT


COMO HOPPE

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

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

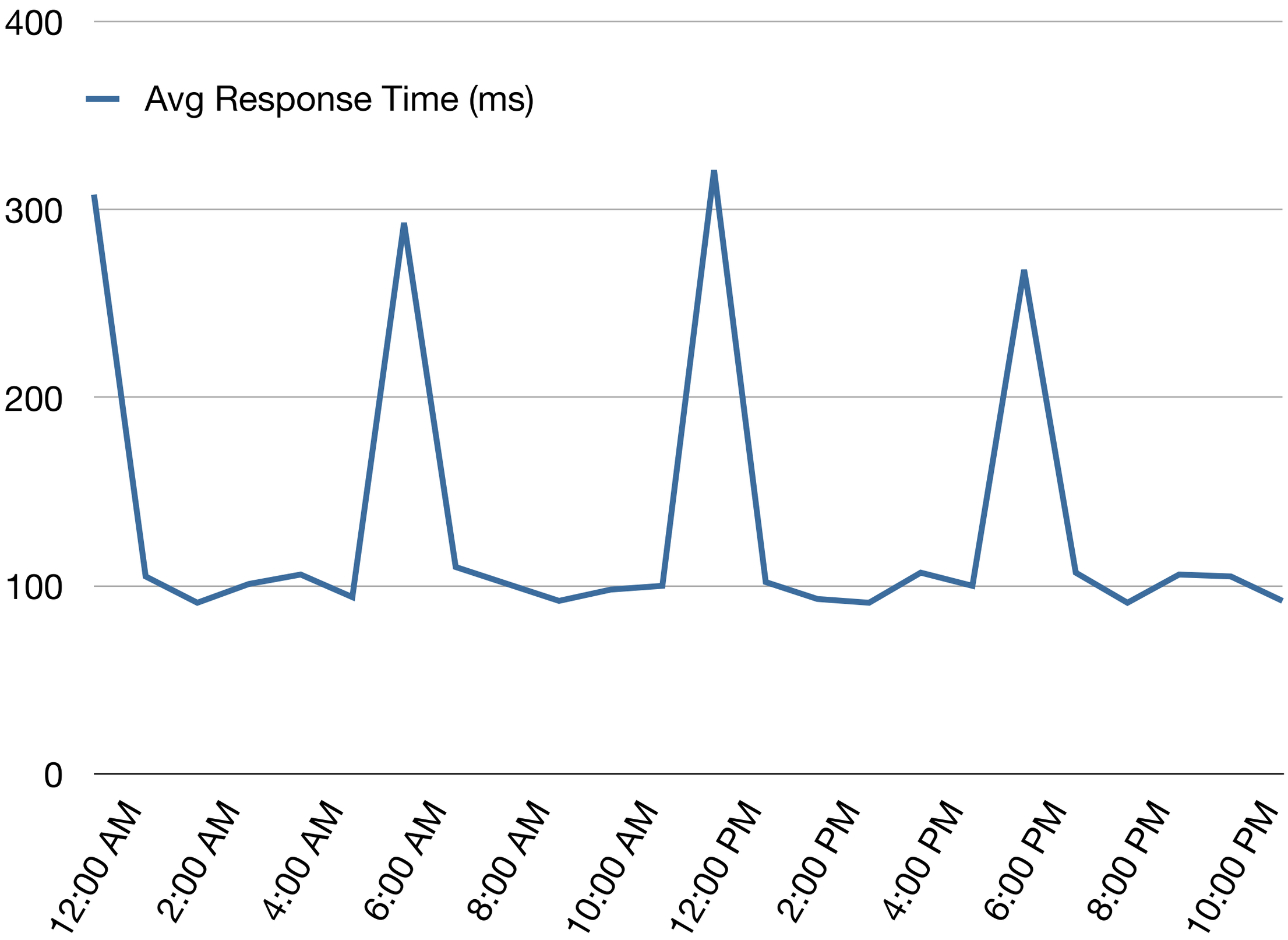
http://www.flickr.com/photos/kalavinka/4610308398

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

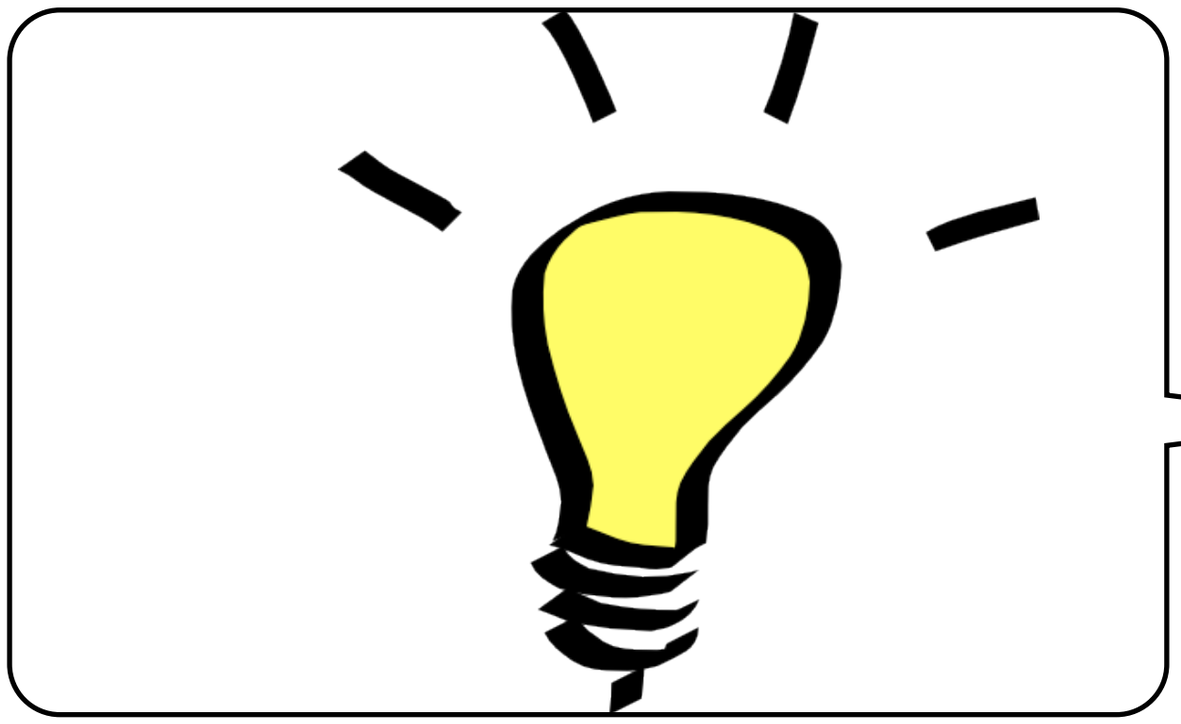
http://www.flickr.com/photos/oakleyoriginals/3526895658

ARGH

Solution


400
Avg Response Time (ms)
300
200
100
0
12:00 AM
2:00 AM
4:00 AM
6:00 AM
8:00 AM
10:00 AM
12:00 PM
2:00 PM
4:00 PM
6:00 PM
8:00 PM
10:00 PM

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



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

Data Is Proof

People Pattern Match

Pictures Are Universal

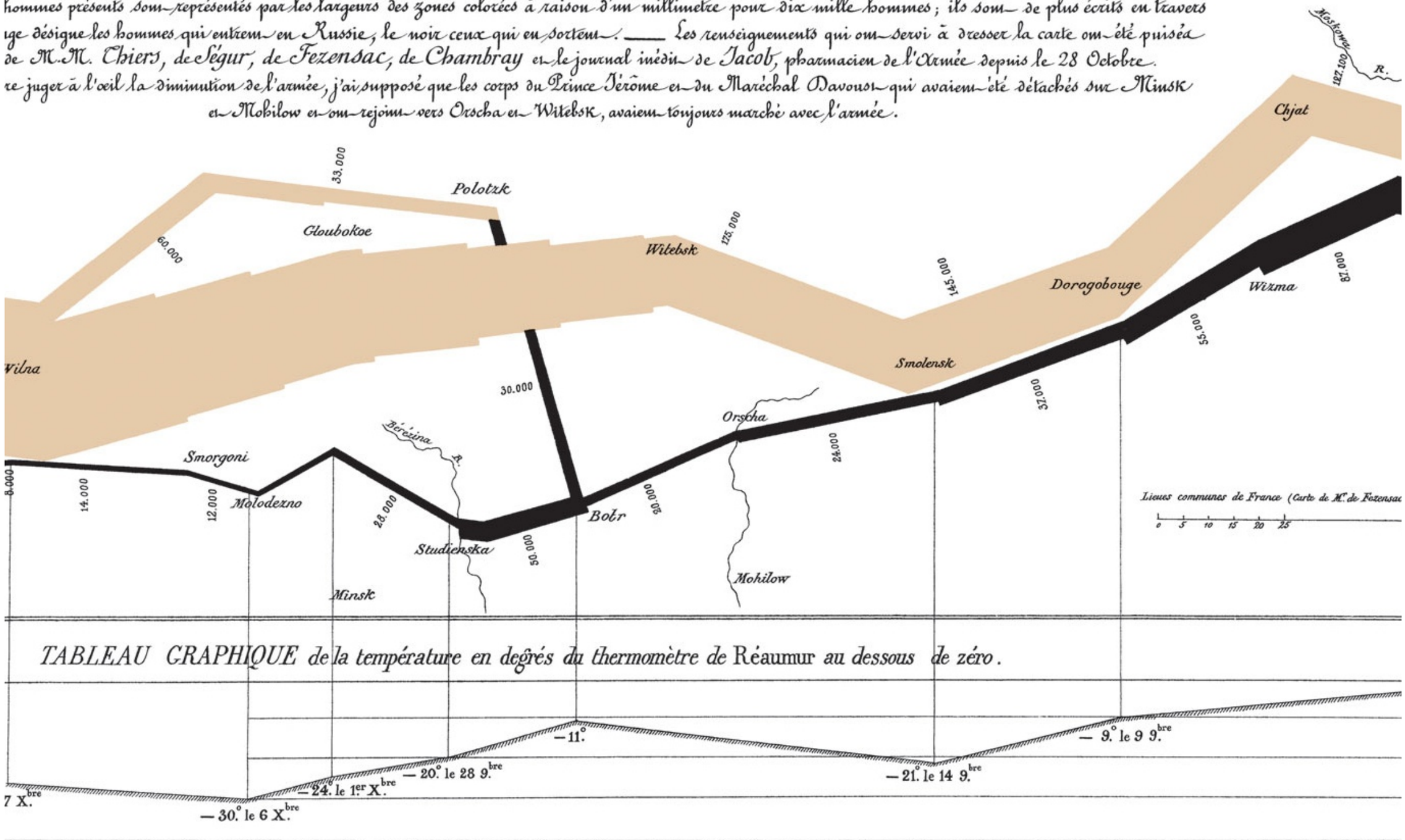
Overview

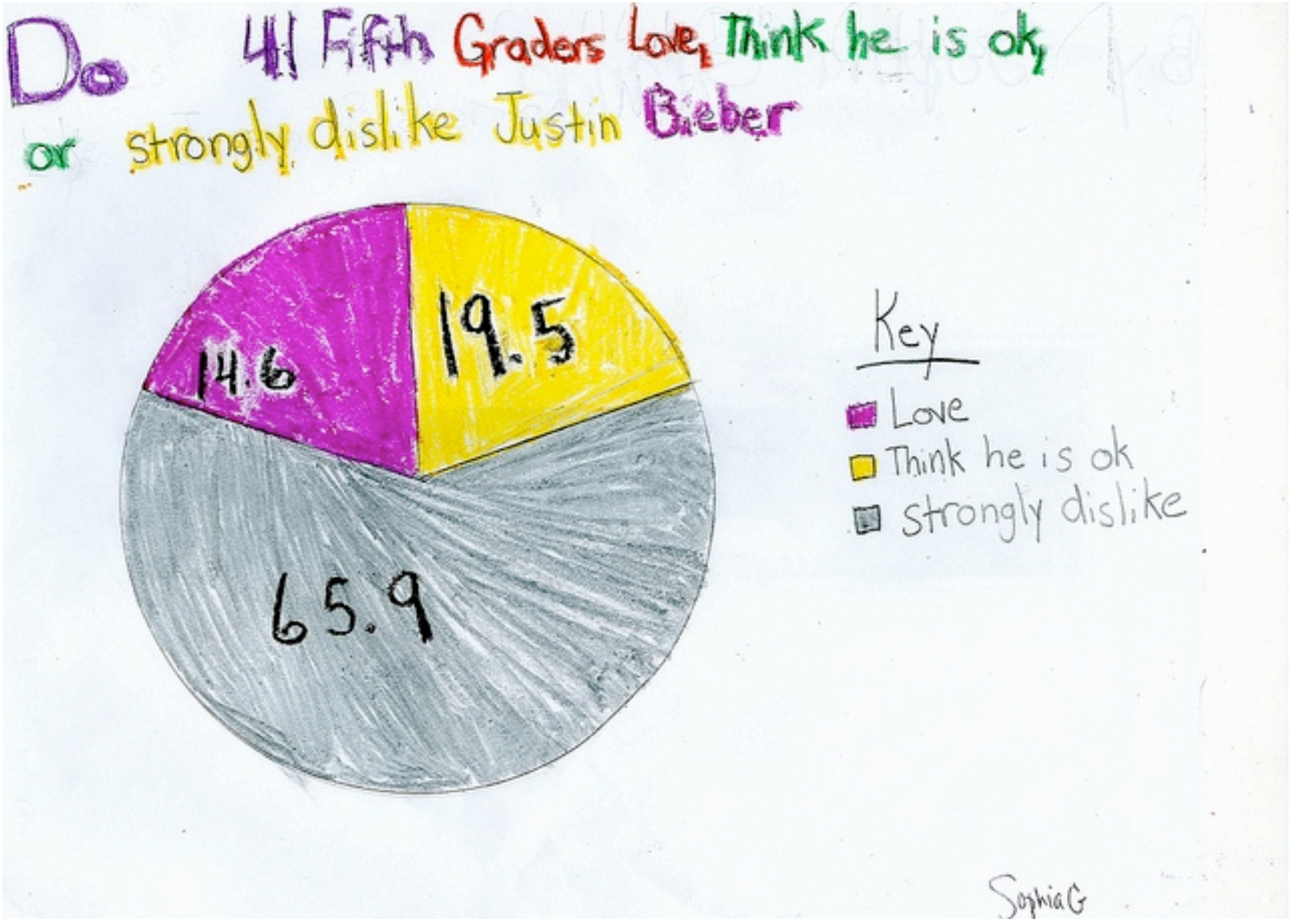
figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.

Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.

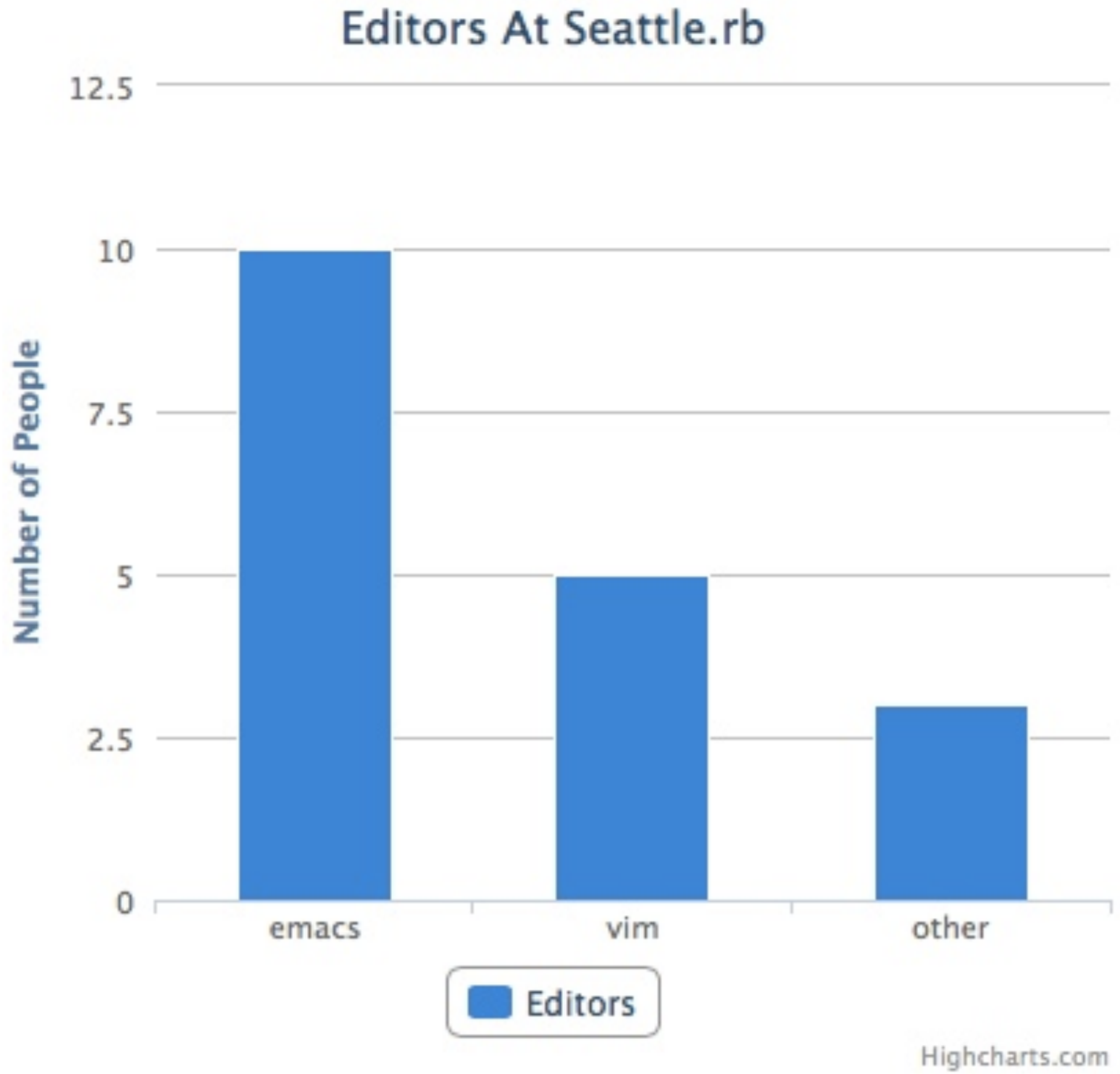
Paris, le 20 Novembre 1869.

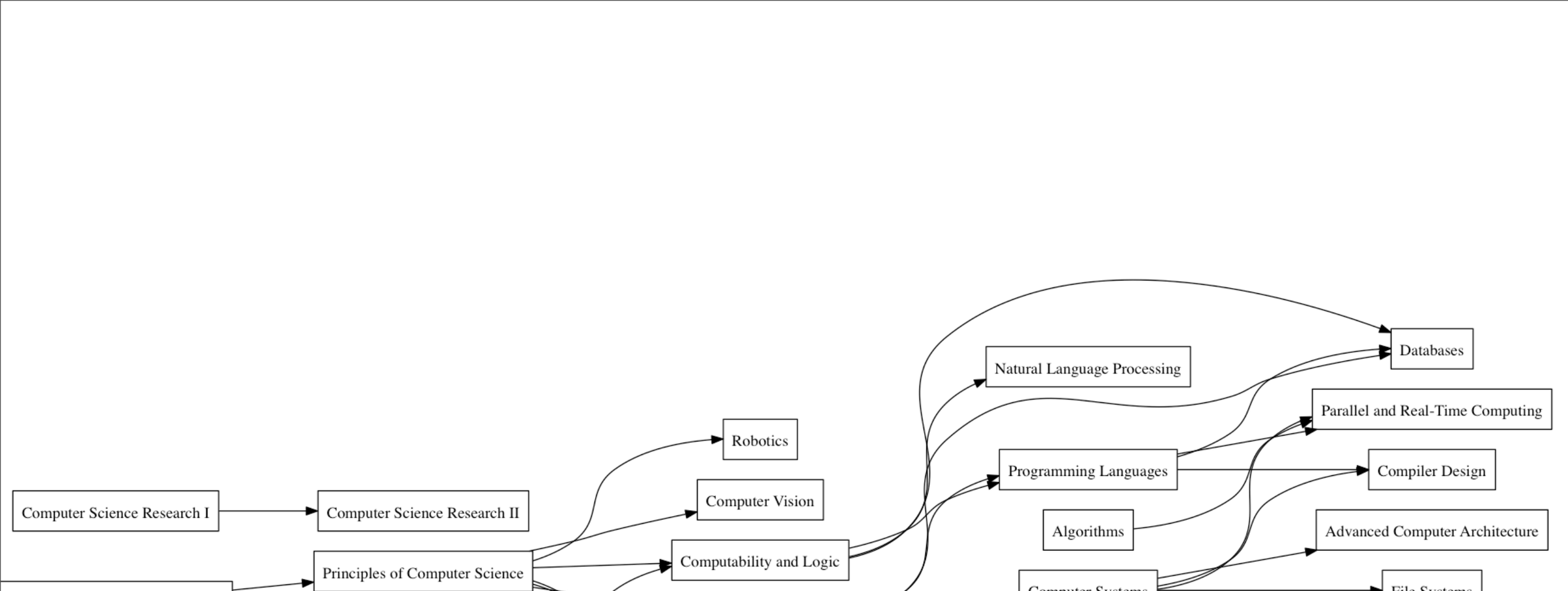
Les hommes présents sont représentés par les largeurs des zônes colorées à raison d'un millimètre pour dix mille hommes ; ils sont de plus écrits en travers
ge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. ___ Les renseignements qui ont servi à dresser la carte ont été puisés
de M.M. Thiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.
re juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davoust qui avaient été détachés sur Minsk
et Mobilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.





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


Editors At Seattle.rb
Number of People
12.5
10
7.5
5
2.5
0
emacs
vim
other
Editors
Highcharts.com


Computer Science Research I
Computer Science Research II
Introduction to Computer Science
Programming Practicum
Principles of Computer Science
Principles and Practice of Computer Science
Robotics
Computer Vision
Computability and Logic
Data Structures and Program Development
Neural Networks
Natural Language Processing
Programming Languages
Algorithms
Computer Systems
Computer Systems Performance Analysis
Software Development
Computer Graphics
Artificial Intelligence
Databases
Parallel and Real-Time Computing
Compiler Design
Advanced Computer Architecture
File Systems
Computer Networks
Operating Systems: Design and Implementation
Computer Science Clinic I, II
Computer Animation
Machine Learning

Expectations

- 95 slides left
- Code heavy
- Practitioner Focused
- Slides & Code Available

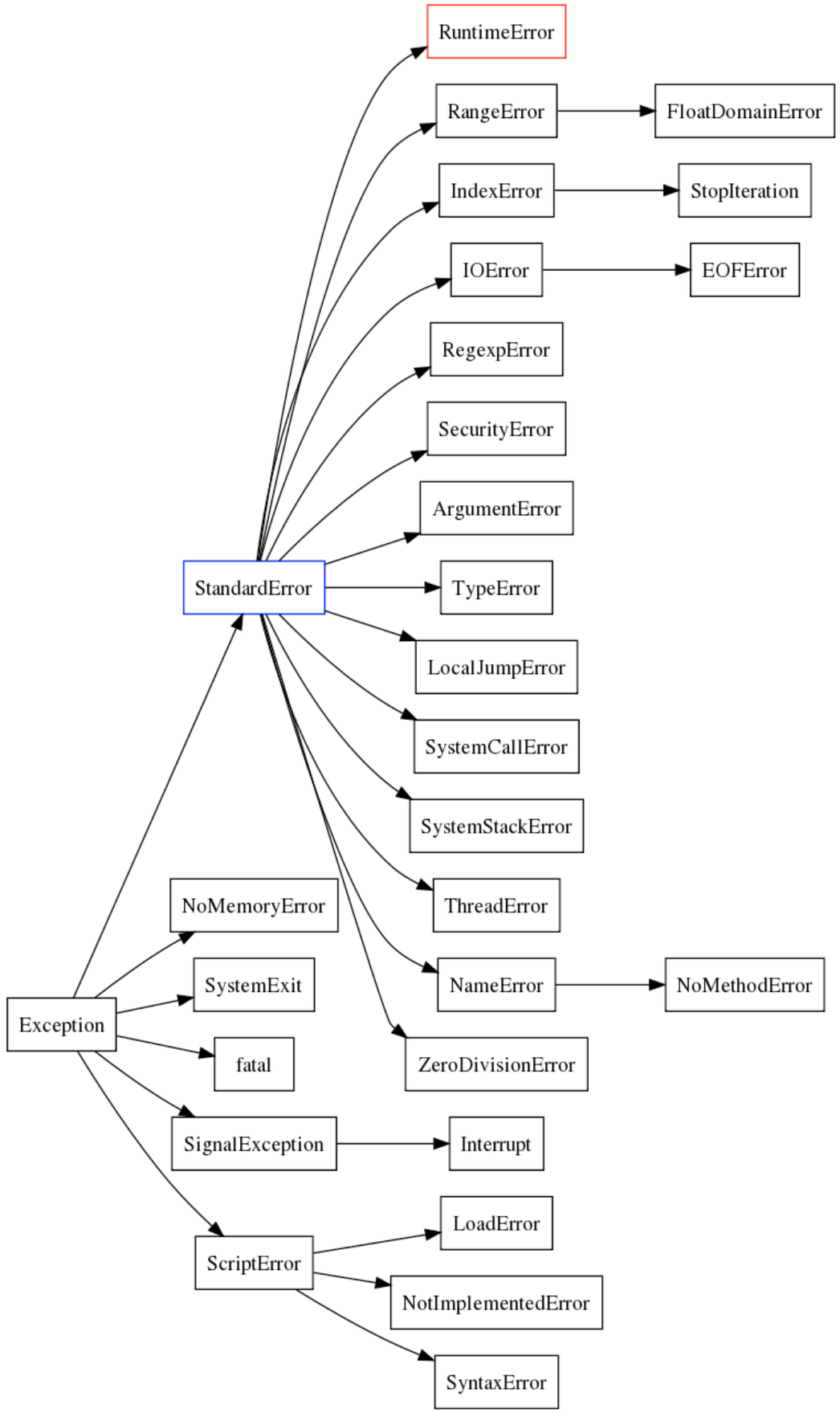
Graph

```
sudo gem install graph
```

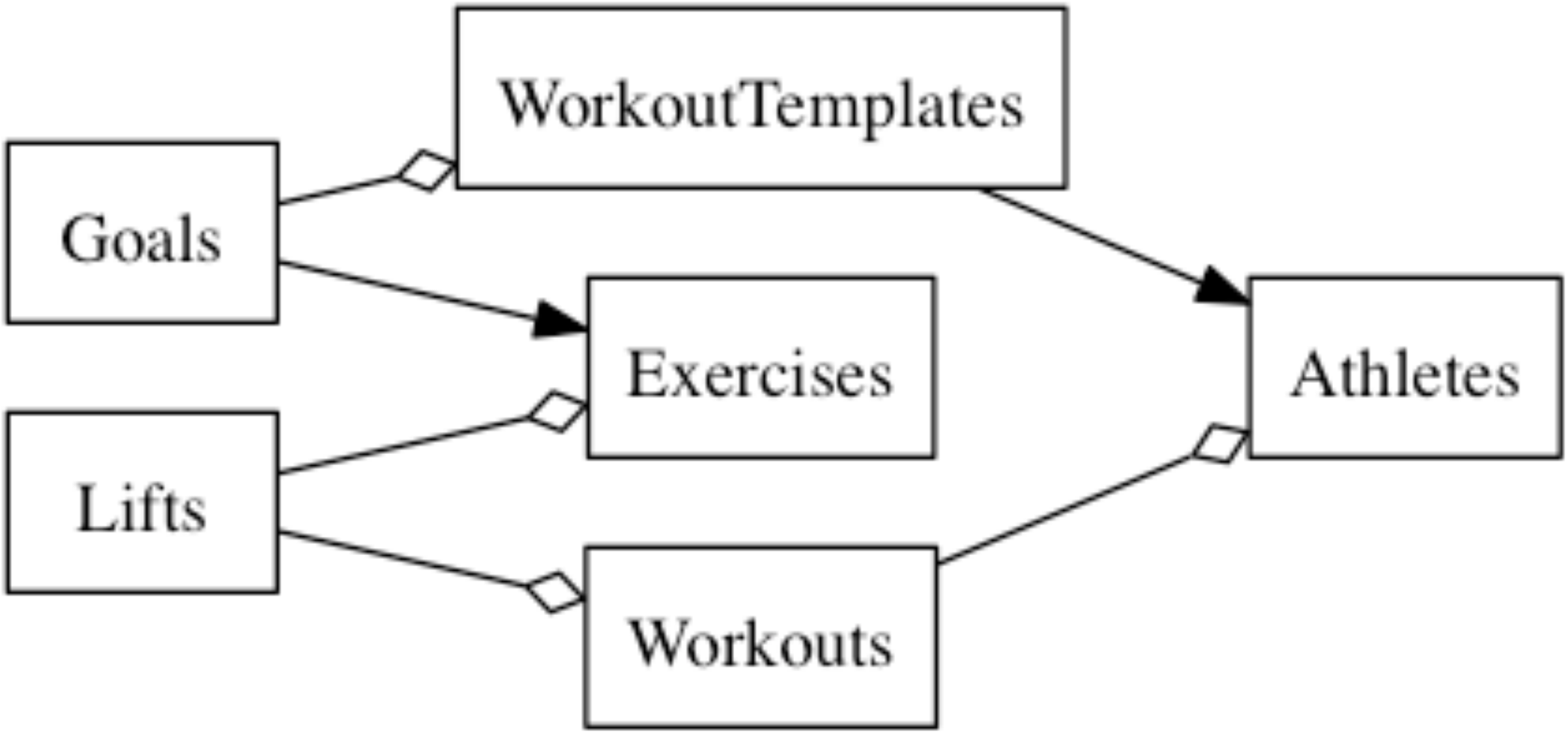
```
brew install graphviz
```

Examples

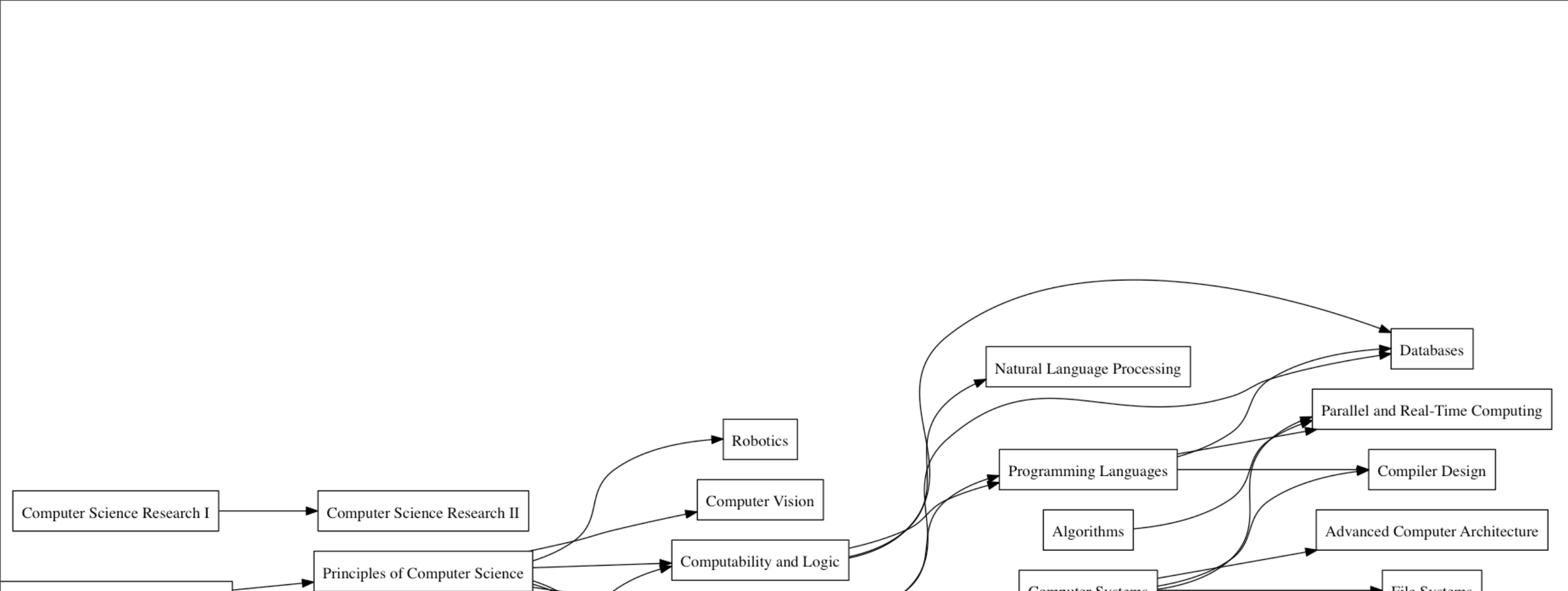
Ruby Exceptions


RuntimeError
RangeError
FloatDomainError
IndexError
StopIteration
IOError
EOFError
RegexpError
SecurityError
ArgumentError
StandardError
TypeError
LocalJumpError
SystemCallError
SystemStackError
NoMemoryError
ThreadError
SystemExit
NameError
NoMethodError
Exception
fatal
ZeroDivisionError
SignalException
Interrupt
LoadError
ScriptError
NotImplementedError
SyntaxError

Rails Associations


WorkoutTemplates
Goals
Exercises
Athletes
Lifts
Workouts

CS Curriculum


Computer Science Research I
Computer Science Research II
Introduction to Computer Science
Programming Practicum
Principles of Computer Science
Principles and Practice of Computer Science
Robotics
Computer Vision
Computability and Logic
Data Structures and Program Development
Neural Networks
Natural Language Processing
Programming Languages
Algorithms
Computer Systems
Computer Systems Performance Analysis
Software Development
Computer Graphics
Artificial Intelligence
Databases
Parallel and Real-Time Computing
Compiler Design
Advanced Computer Architecture
File Systems
Computer Networks
Operating Systems: Design and Implementation
Computer Science Clinic I, II
Computer Animation
Machine Learning

DOT

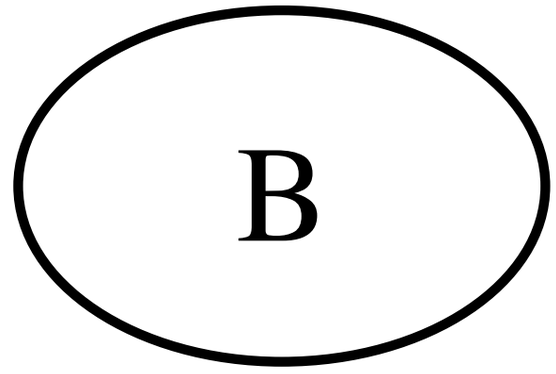
Viewing DOT files

- GraphViz
- Tulip
- Omnigraffle

Basics

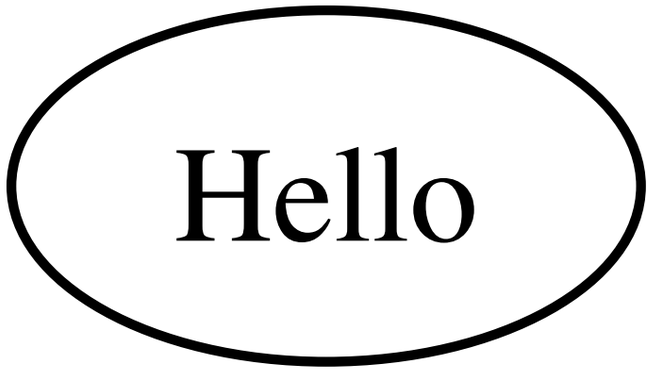
Single Node

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



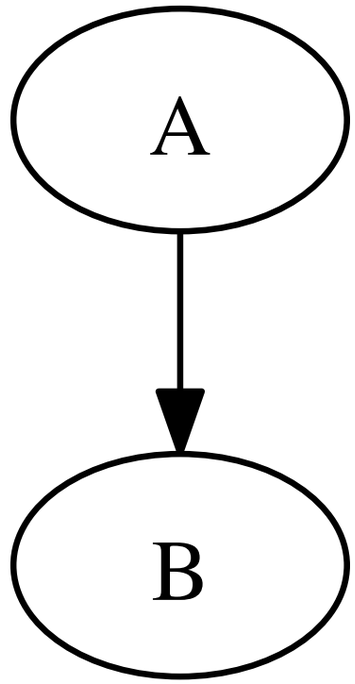
Labels

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



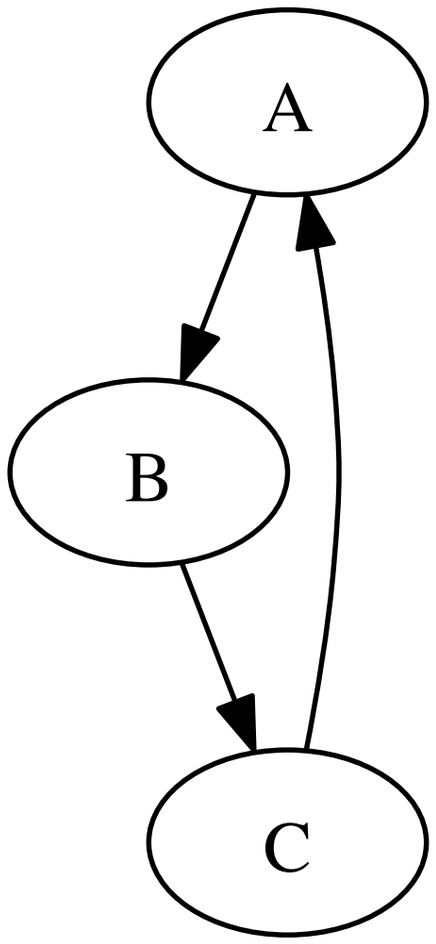
Edges

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



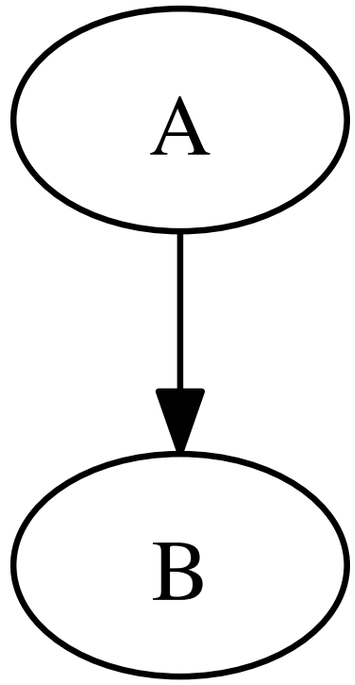
Saving

```
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
```



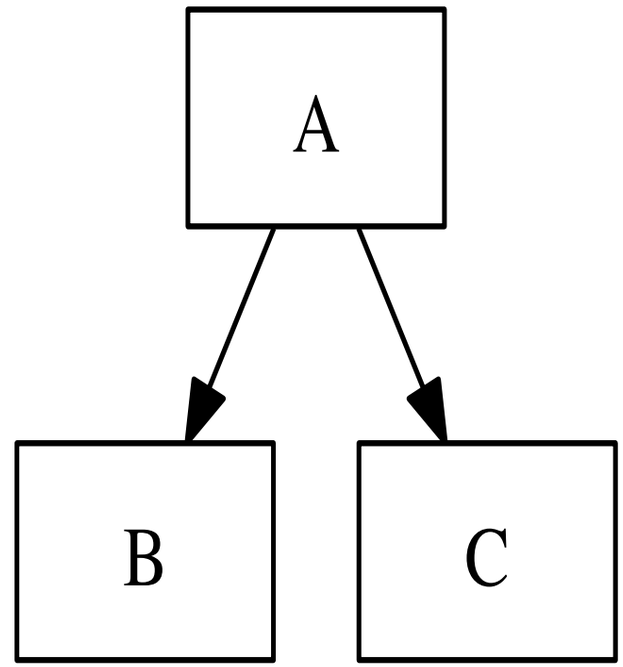
Format list: http://www.graphviz.org/doc/info/output.html

Advanced

Shapes

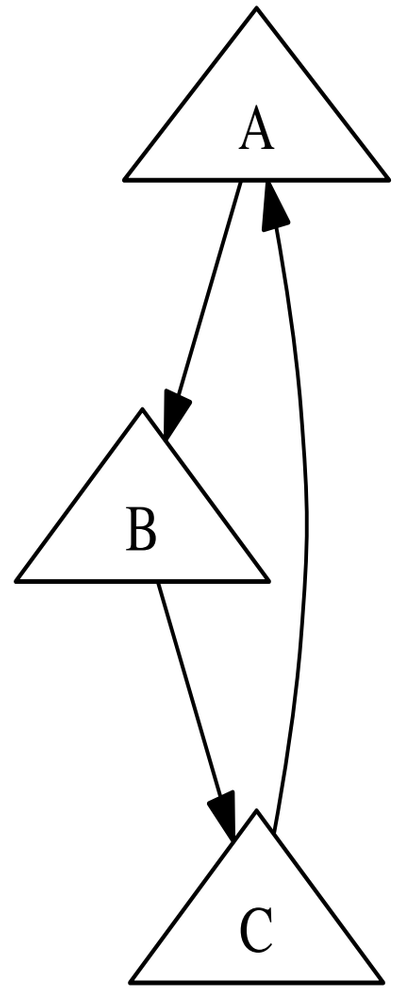
Boxes

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



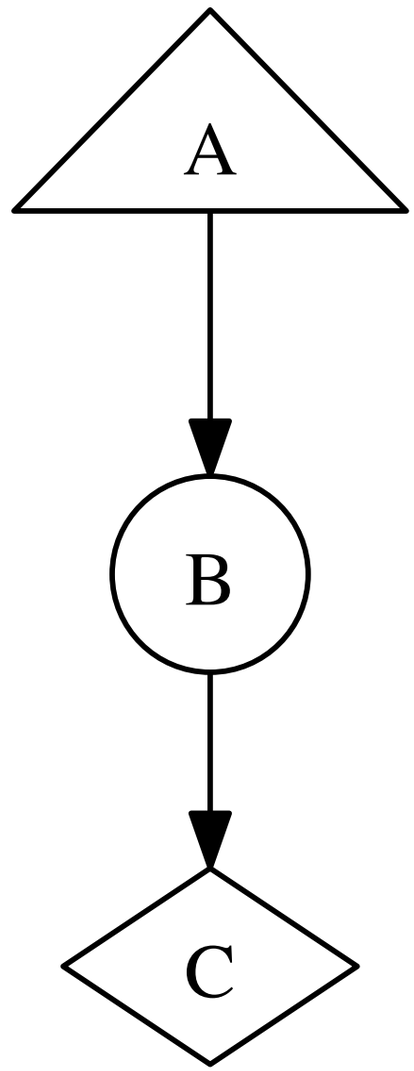
Shapes

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



Many Shapes

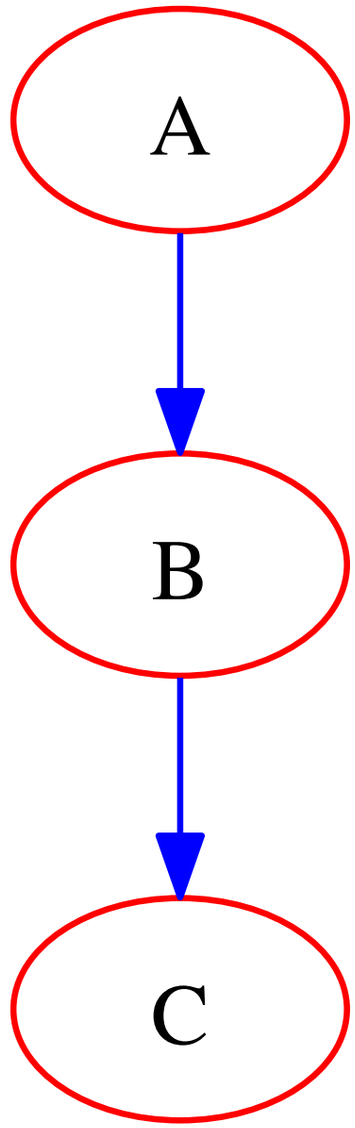
```
digraph do
  edge "A", "B", "C"
  triangle << node("A")
  circle   << node("B")
  diamond  << node("C")
end
```



Color

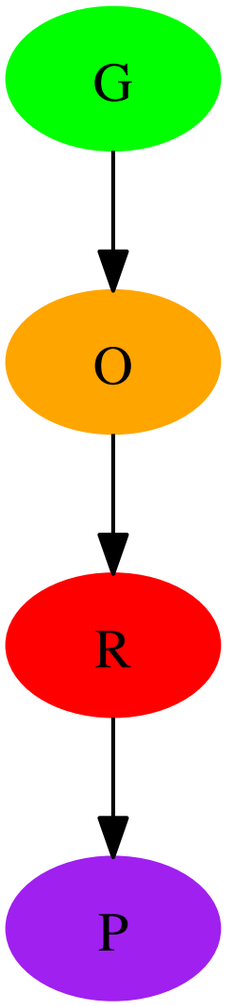
Color Edges & Nodes

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



Many Colors

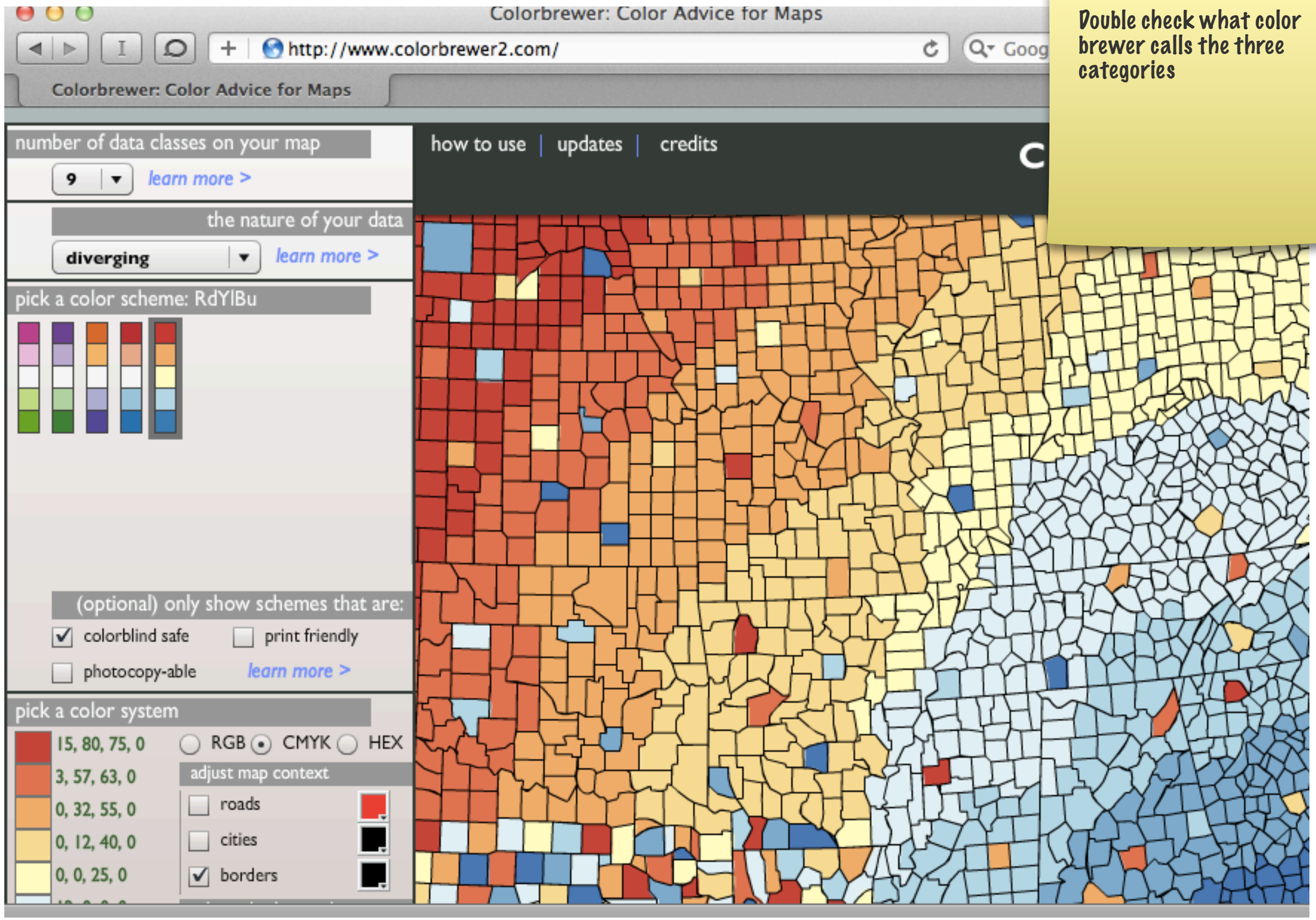
```
digraph do
  node_attribs << filled
  edge "G", "O", "R", "P"
  green  << node("G")
  orange << node("O")
  red    << node("R")
  purple << node("P")
end
```



Design Impaired?

www.colorbrewer2.com

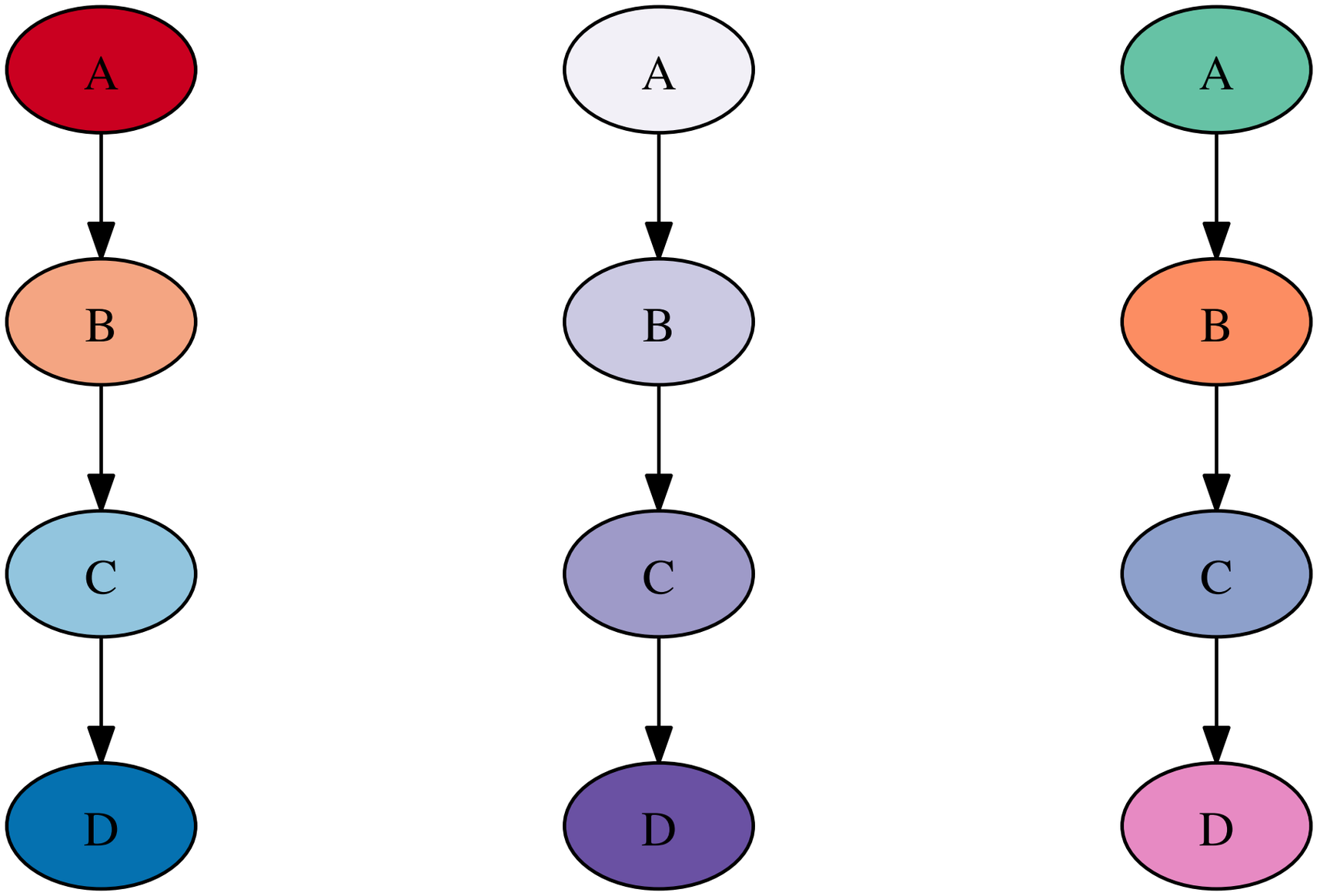
Double check what color brewer calls the three categories



www.graphviz.org/doc/info/colors.html

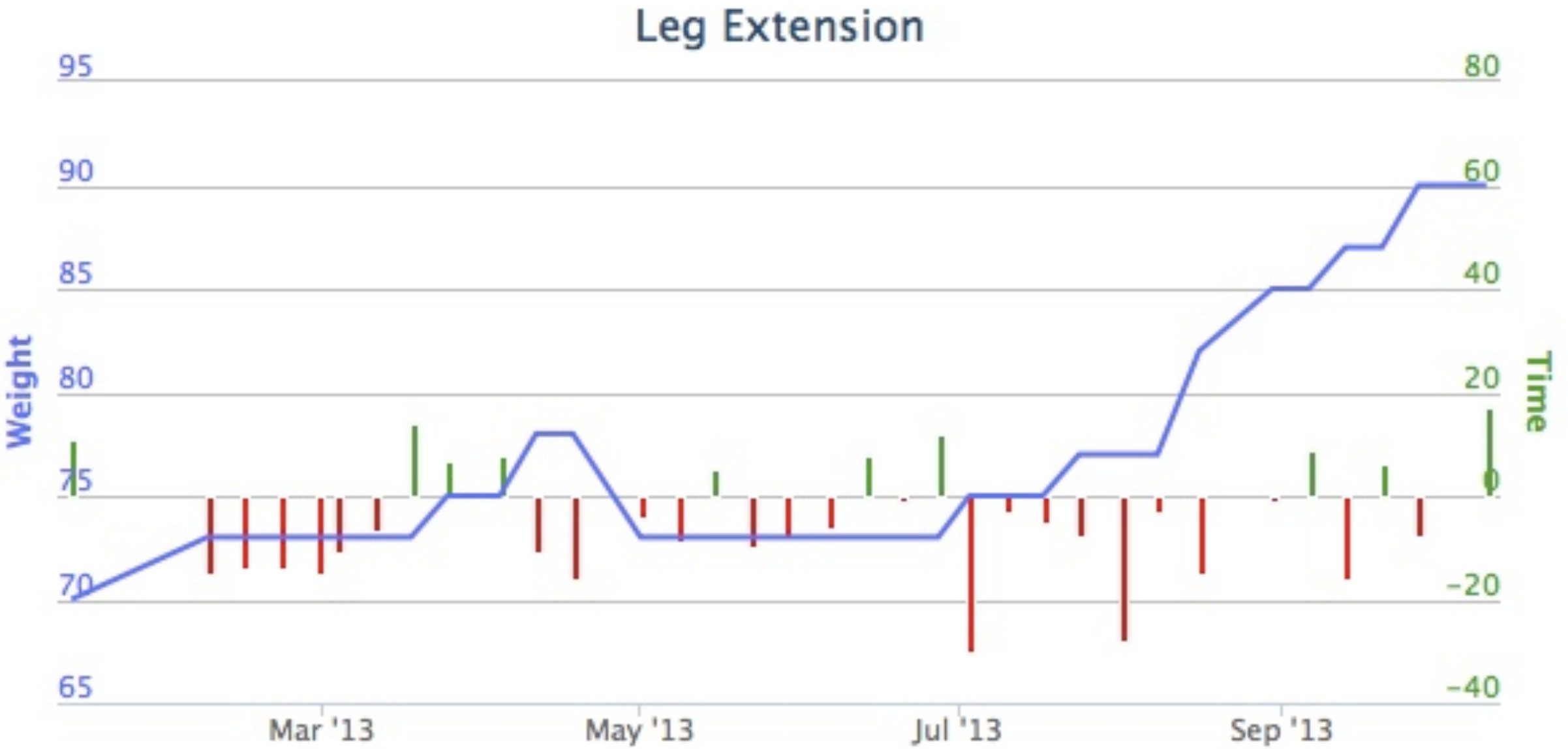
Example

```
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
```


A
B
C
D
A
B
C
D
A
B
C
D

Charts

Examples


Leg Extension
95
90
85
80
75
70
65
80
60
40
20
0
-20
-40
Mar '13
May '13
Jul '13
Sep '13


World's largest cities per 2008
Population (millions)
40
30
20
10
0
34.4
21.8
20.1
20
19.6
19.5
19.1
18.4
18
17.3
16.8
15
14.7
14.5
13.3
12.8
12.4
11.8
11.7
11.2
Tokyo
Jakarta
New York
Seoul
Manila
Mumbai
Sao Paulo
Mexico City
Dehli
Osaka
Cairo
Kolkata
Los Angeles
Shanghai
Moscow
Beijing
Buenos Aires
Guangzhou
Shenzhen
Istanbul

Highcharts 101

Basic Page

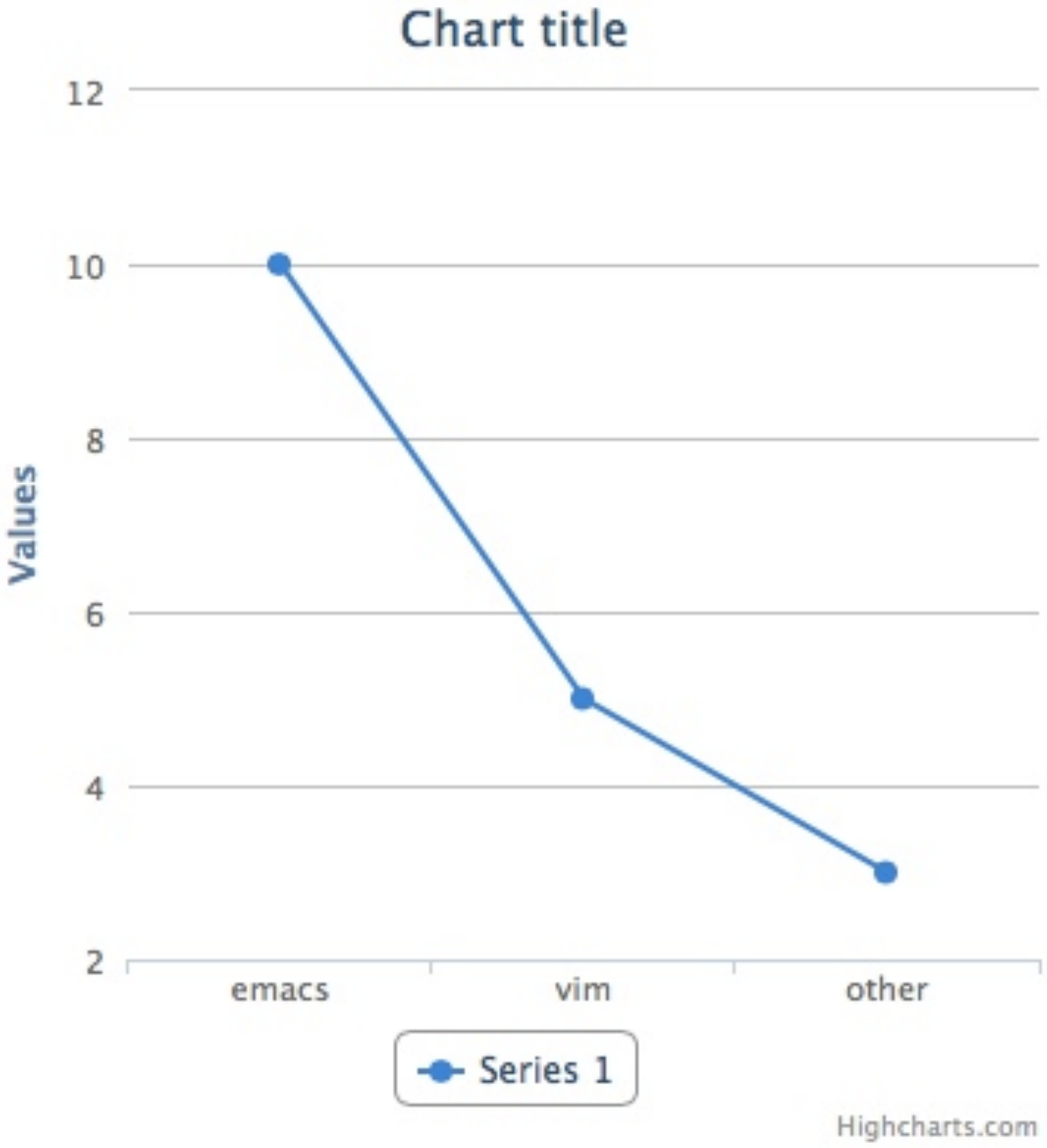


Chart title
12
10
8
6
4
2
Values
emacs
vim
other
Series 1
Highcharts.com

Requirements

JQuery
Highcharts.js

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

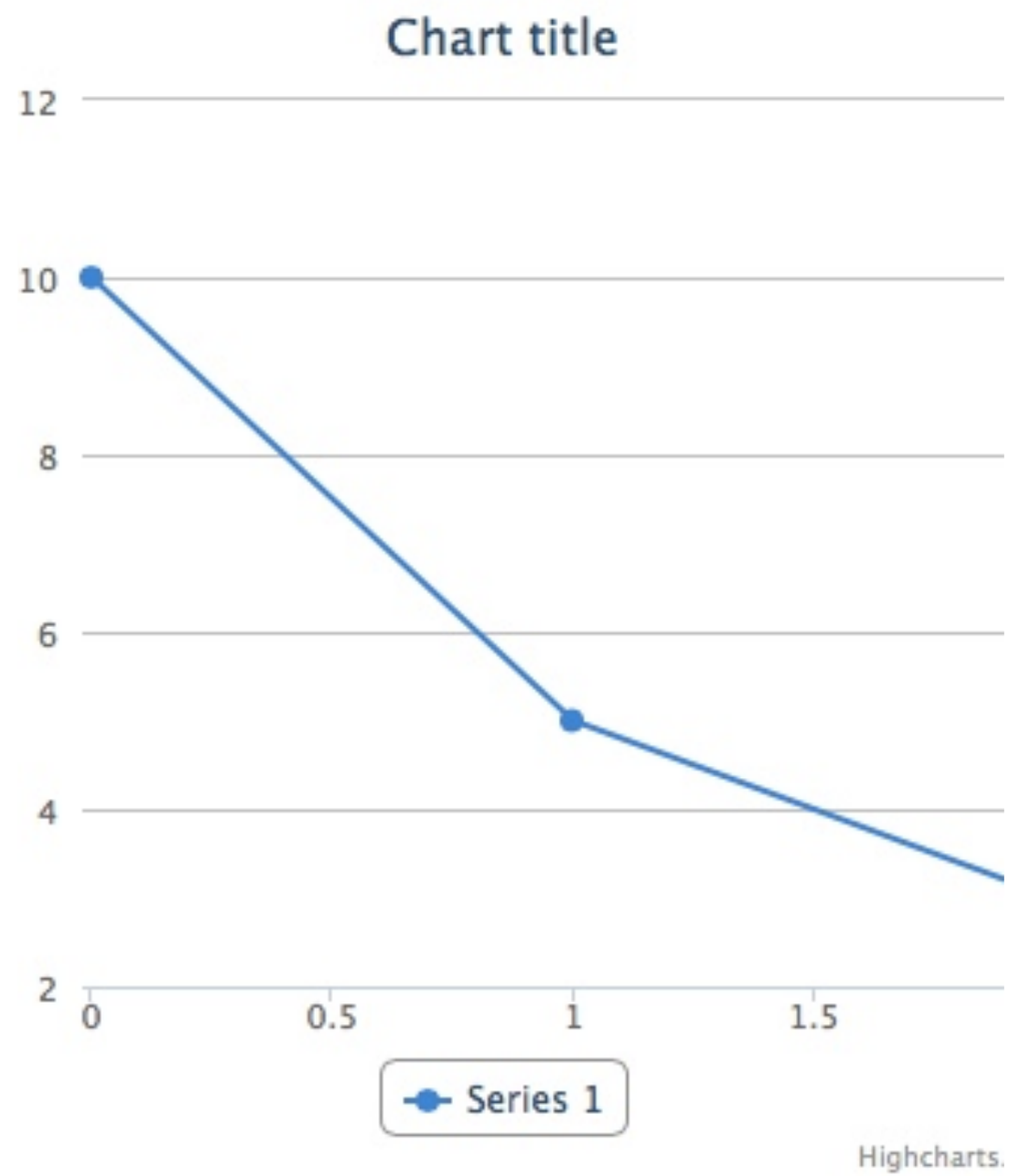
```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<script>
$(function () {
        $('#chart').highcharts({
            xAxis: {
                categories: ['emacs',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

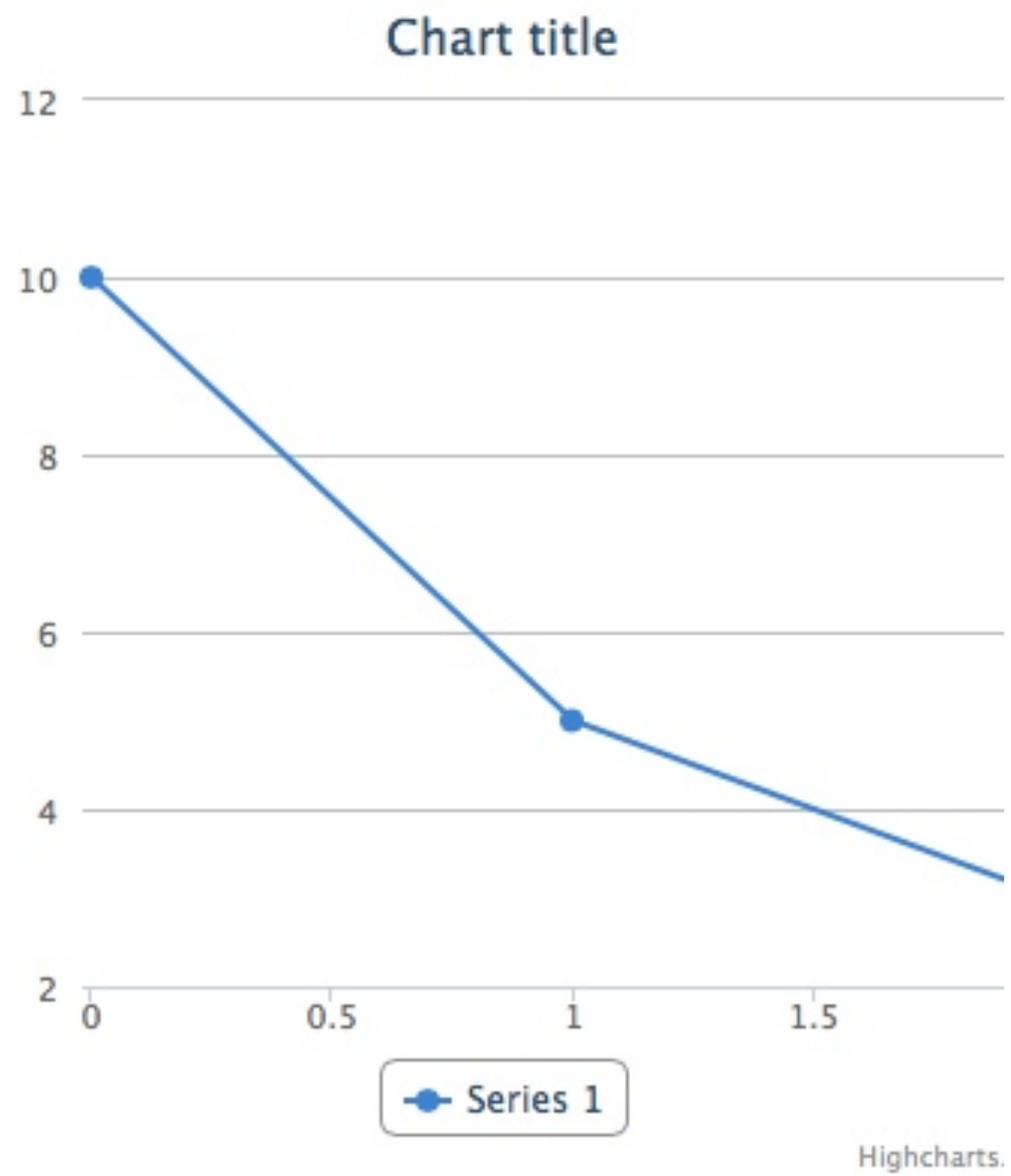
```
<html>
<script type="text/javascript" src="http://
ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"></script>
<script src="http://code.highcharts.com/
highcharts.js"></script>
<script>
$(function () {
        $('#chart').highcharts({
            xAxis: {
                categories: ['emacs',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

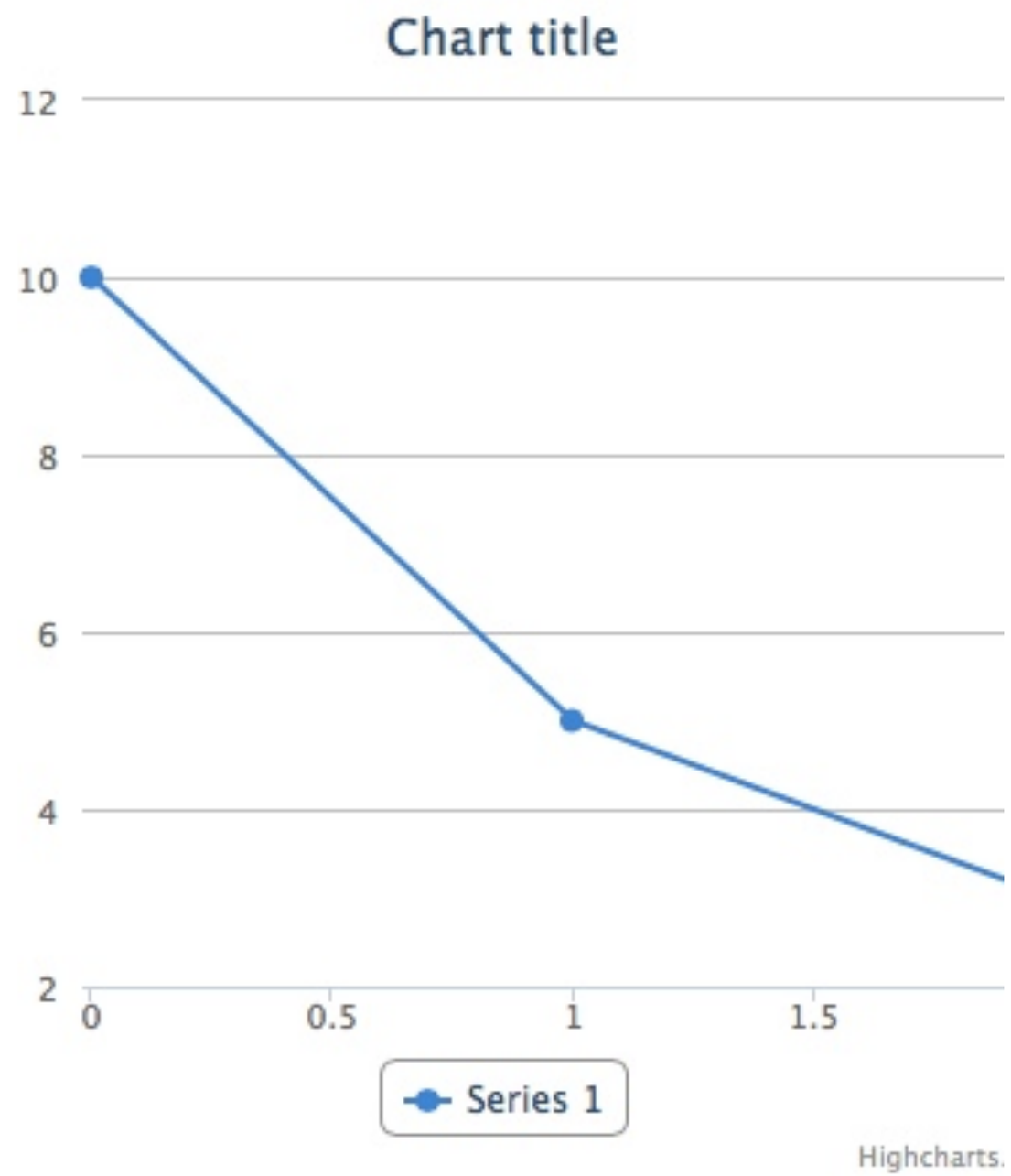
```
$(function () {
  $('#chart').highcharts({
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



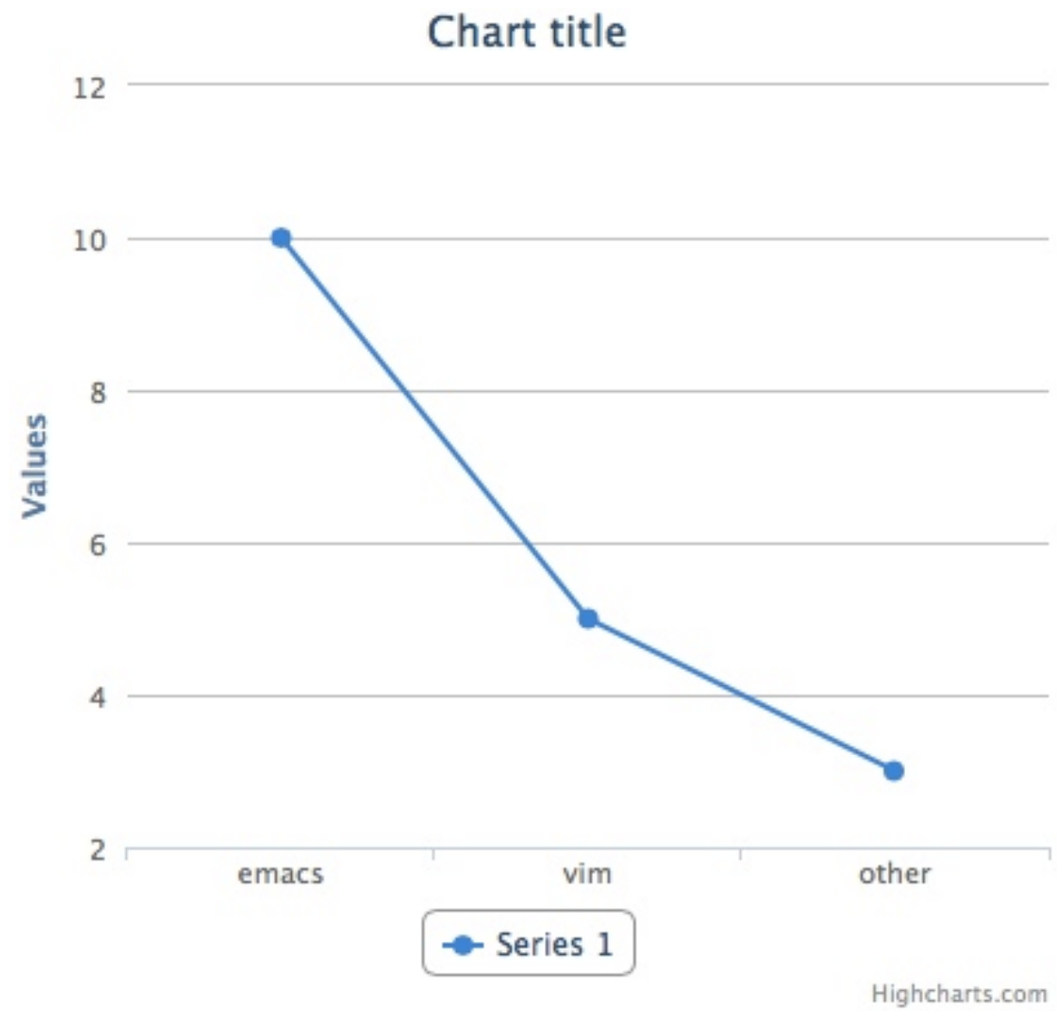
```
$(function () {
  $('#chart').highcharts({
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



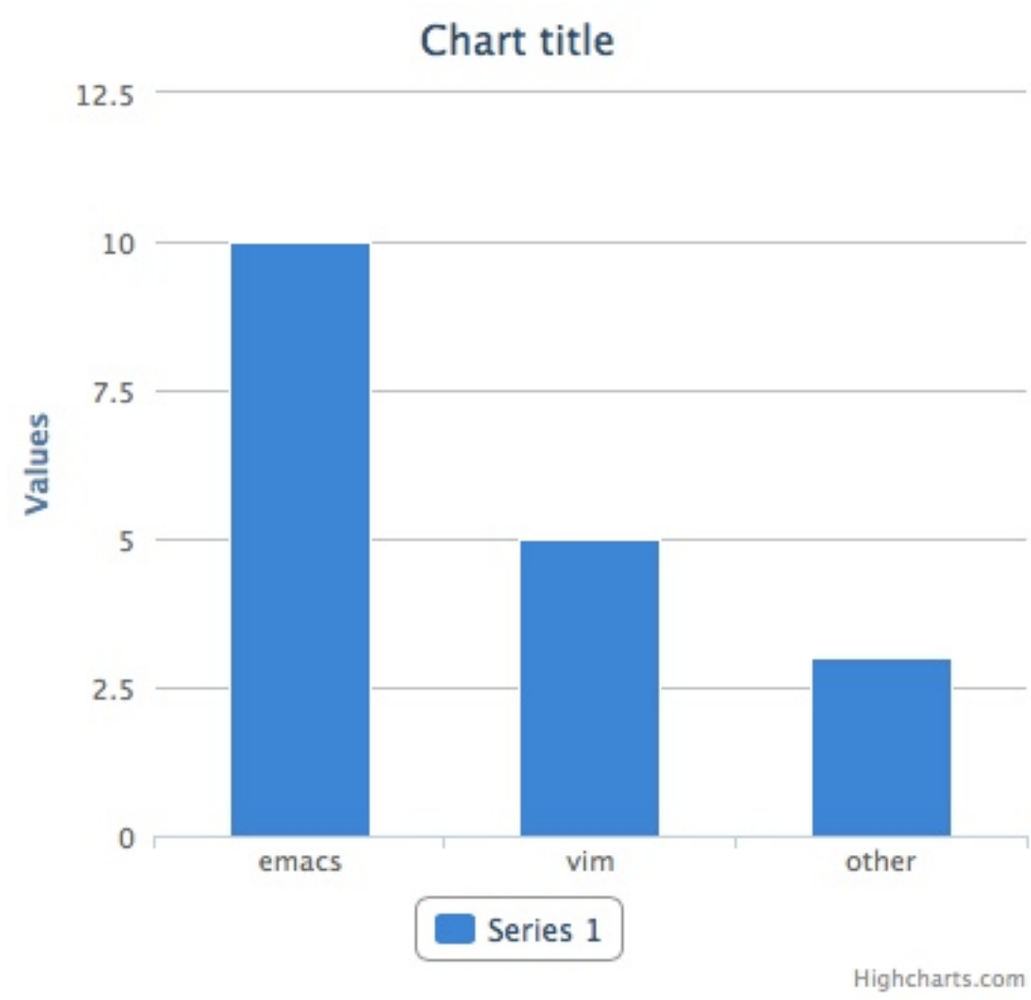
```
$(function () {
  $('#chart').highcharts({
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



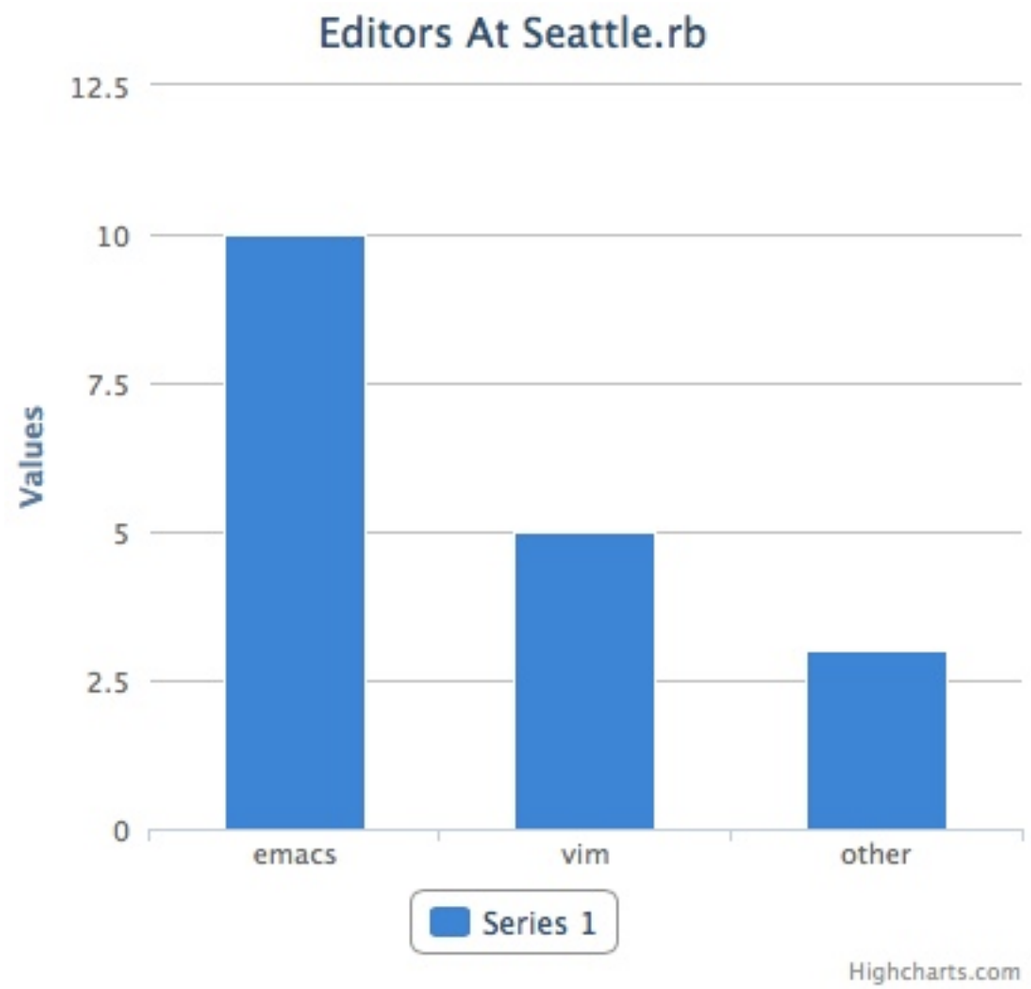
```
$(function () {
  $('#chart').highcharts({
    xAxis : {
      categories:
       ["emacs",
        "vim",
        "other"]
    },
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



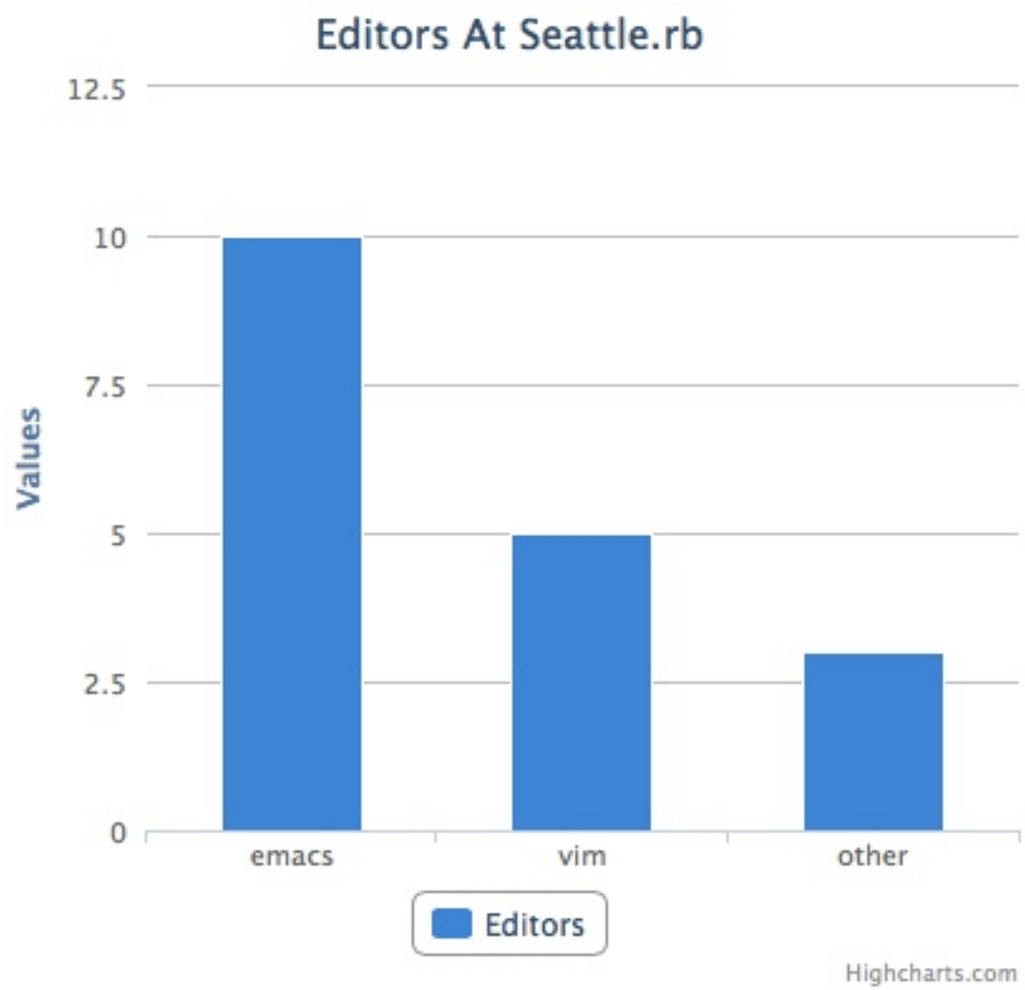
```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    xAxis : {
      categories:
        ["emacs",
         "vim",
        "other"]
    },
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



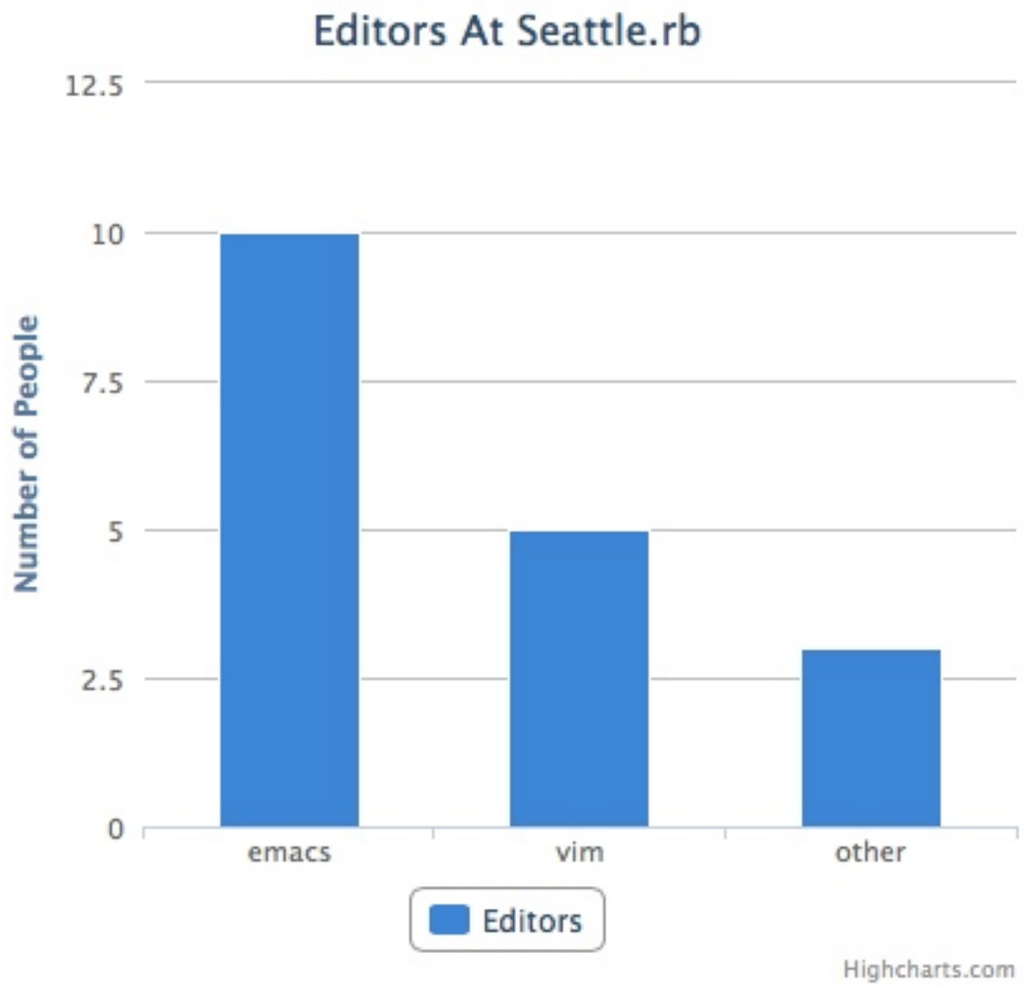
```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    title: {
      text: 'Editors At
Seattle.rb'
    },
    xAxis : {
      categories:
        ["emacs",
         "vim",
         "other"]
    },
    series: [{
      data: [10, 5, 3]
    }]
  });
});
```



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

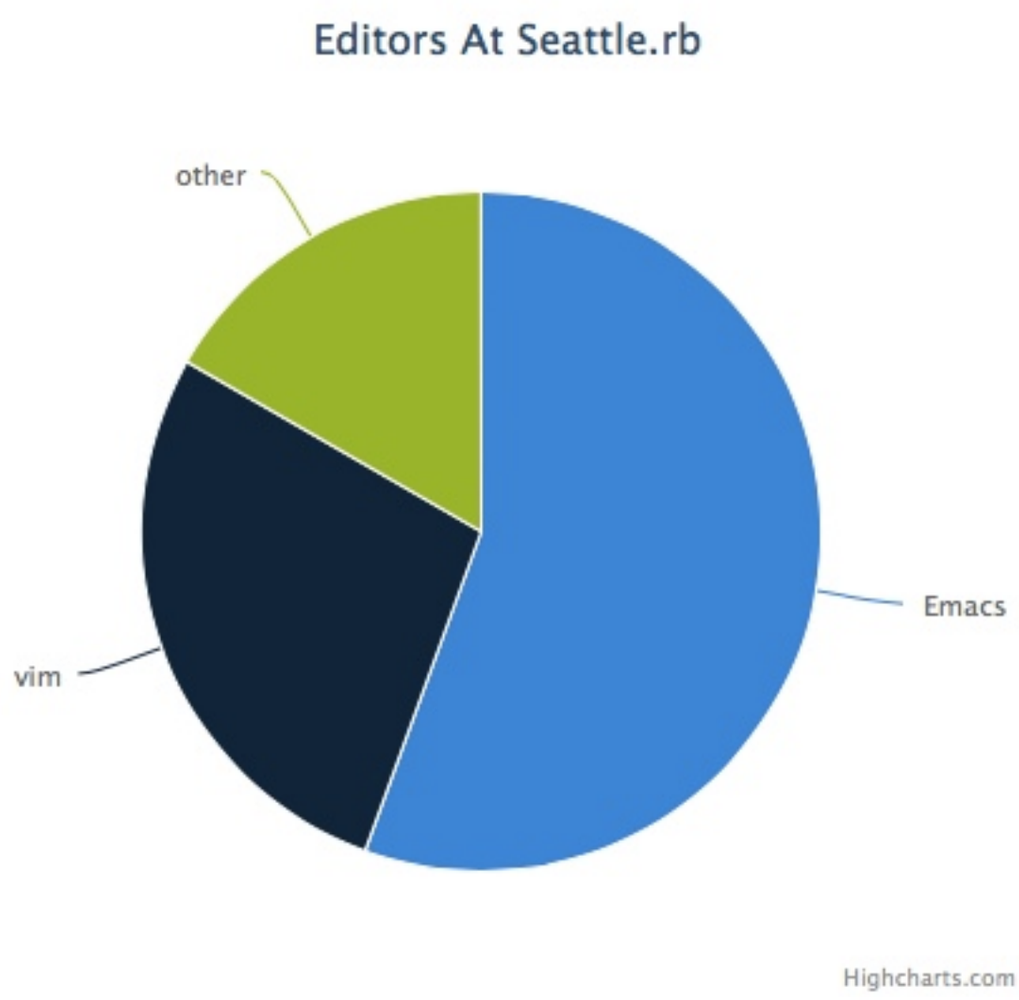


```
$(function () {
  $('#chart').highcharts({
    chart : {
      type: 'column'
    },
    title: { ... },
    yAxis : {
      title: {
text: "Number of People" }
    },
    xAxis : { ... },
    series: [{
      name: "Editors",
      data: [10, 5, 3]
    }]
  });
});
```



Highcharts 102

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



line

spline

area

column

bar

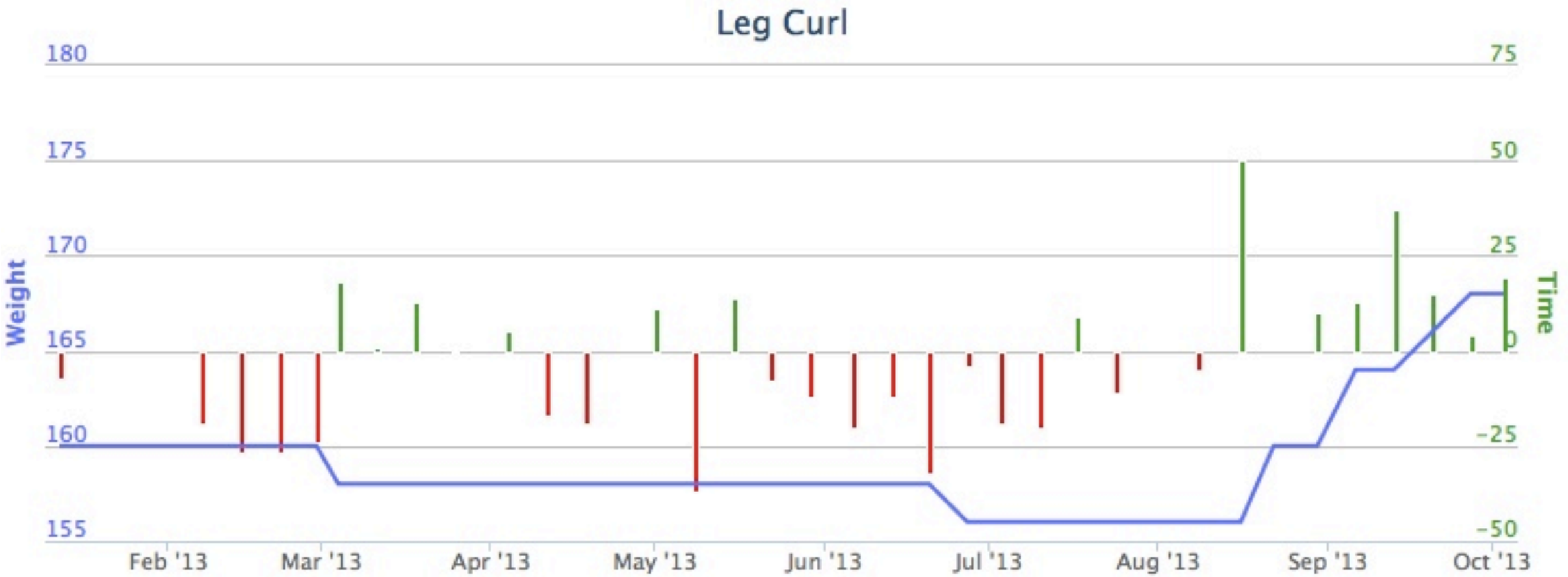
pie

scatter

area range

area spline

column range


Leg Curl
180
175
170
165
160
155
Weight
75
50
25
0
-25
-50
Time
Feb '13
Mar '13
Apr '13
May '13
Jun '13
Jul '13
Aug '13
Sep '13
Oct '13

```
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

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

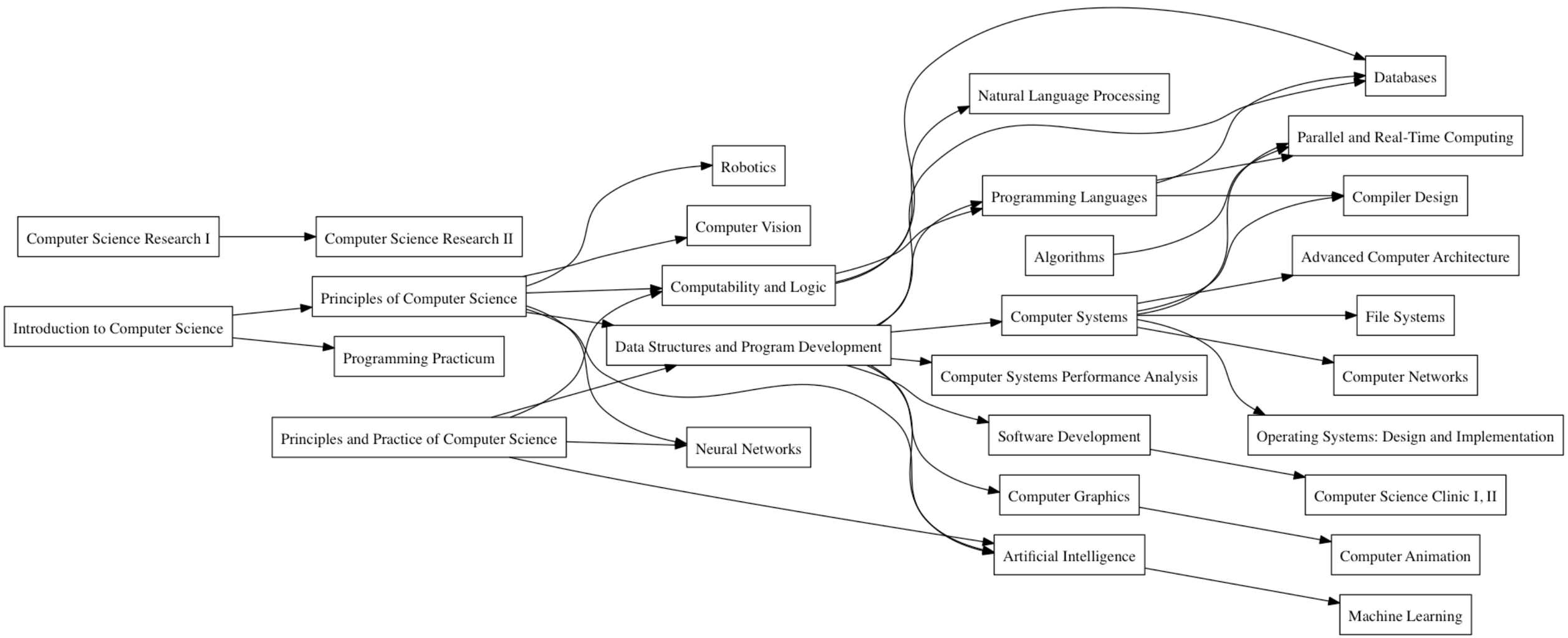
Prerequisites Permission by instructor.
Credit Hours 3.0
Offered Fall semester.

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.


Computer Science Research I
Computer Science Research II
Introduction to Computer Science
Programming Practicum
Principles of Computer Science
Principles and Practice of Computer Science
Robotics
Computer Vision
Computability and Logic
Neural Networks
Data Structures and Program Development
Natural Language Processing
Programming Languages
Algorithms
Computer Systems
Computer Systems Performance Analysis
Software Development
Computer Graphics
Artificial Intelligence
Databases
Parallel and Real-Time Computing
Compiler Design
Advanced Computer Architecture
File Systems
Computer Networks
Operating Systems: Design and Implementation
Computer Science Clinic I, II
Computer Animation
Machine Learning

```ruby
#!/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
```

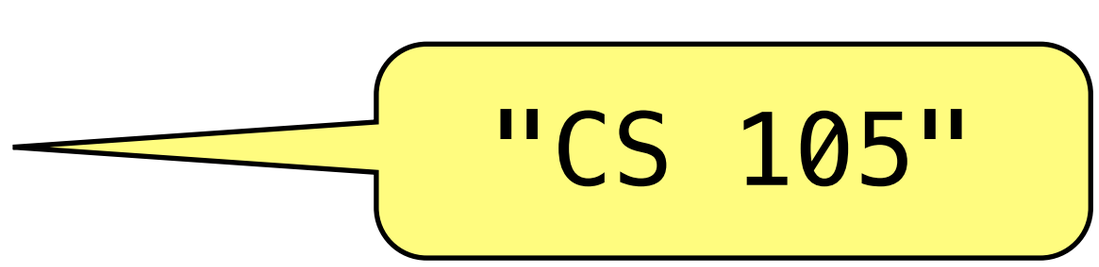
```ruby
#!/usr/bin/ruby -w
require 'rubygems'
require 'graph'
require 'nokogiri'
require 'open-uri'

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")
```

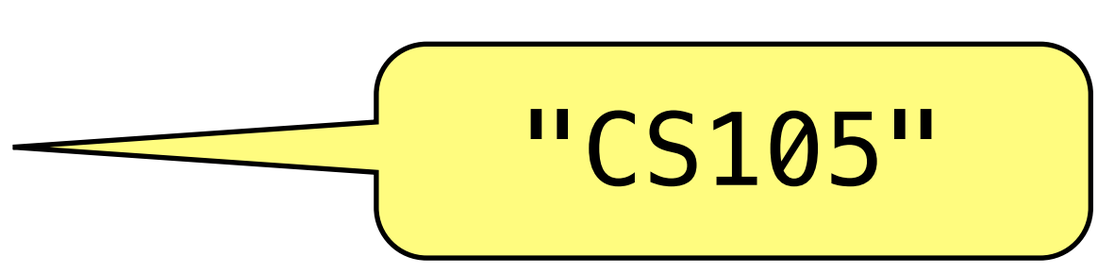
```
require 'graph'
require 'nokogiri'
require 'open-uri'

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

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")
```

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.

Course Descri

This is a listing of the all the cou
are offering in the current semes

".crsdscrptn-header"

CS 5. Introduction t 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.

```ruby
#!/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
```

```ruby
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
```

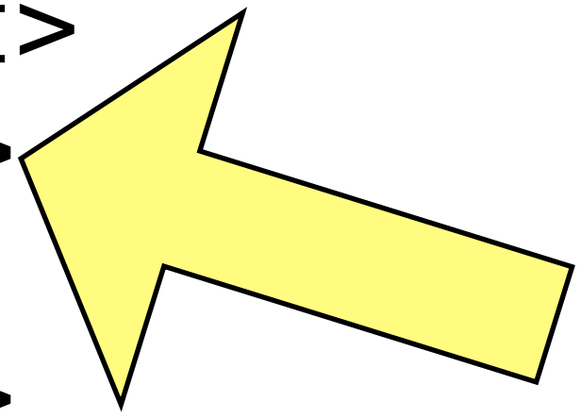
```
    number, title = description.css(".crsdscrptn-
    title").text.split('.')

    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
```

```ruby
#!/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
```

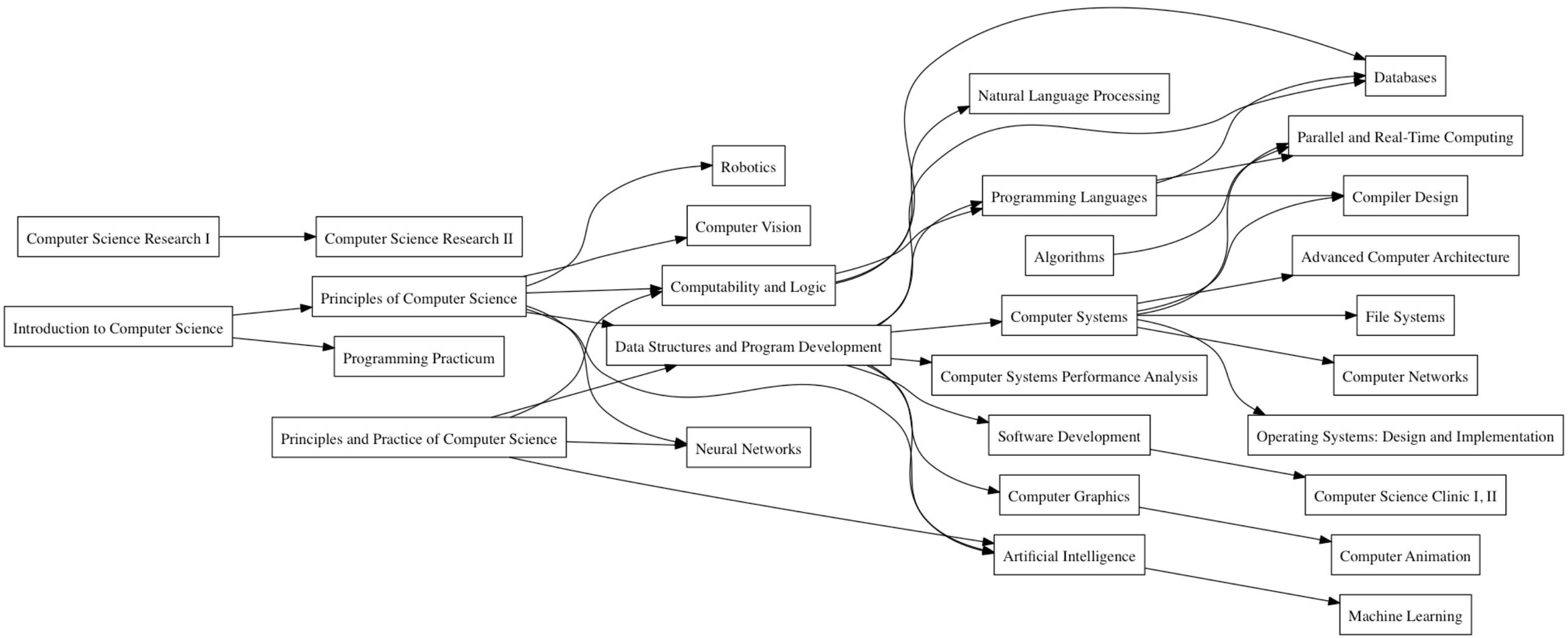
```
digraph do
  rotate; boxes;

  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
```

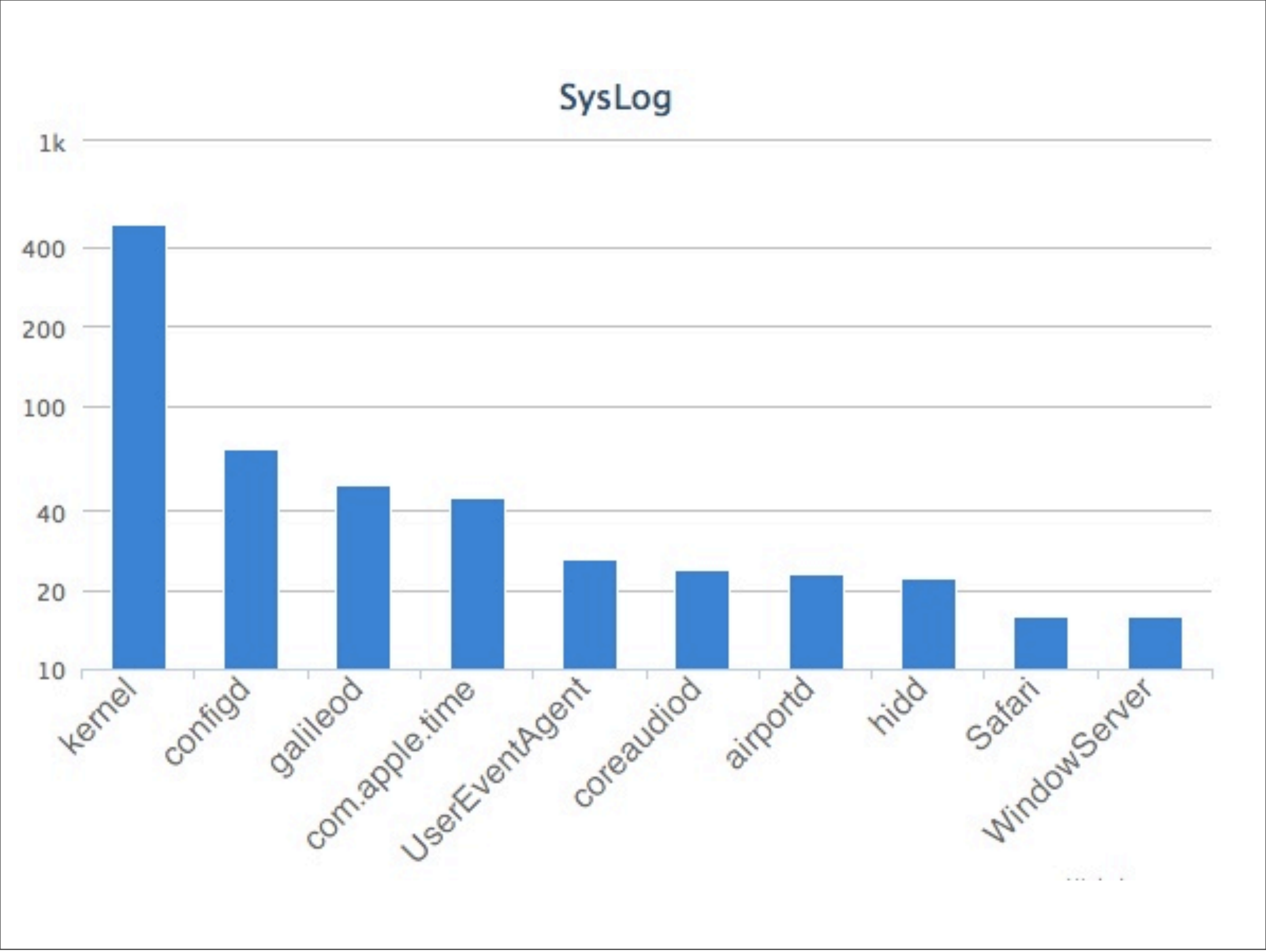
```
courses.each do |course|
  course[:pre_reqs].each do |pr|
    edge pr , course[:number]
  end
end
```

```
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
```

```
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
```


Computer Science Research I
Computer Science Research II
Introduction to Computer Science
Programming Practicum
Principles of Computer Science
Principles and Practice of Computer Science
Robotics
Computer Vision
Computability and Logic
Neural Networks
Data Structures and Program Development
Natural Language Processing
Programming Languages
Algorithms
Computer Systems
Computer Systems Performance Analysis
Software Development
Computer Graphics
Artificial Intelligence
Databases
Parallel and Real-Time Computing
Compiler Design
Advanced Computer Architecture
File Systems
Computer Networks
Operating Systems: Design and Implementation
Computer Science Clinic I, II
Computer Animation
Machine Learning

```
counts = Hash.new 0

ARGV.each do |path|
  File.foreach path do |line|
    if line =~ date_regexp
      counts[$1] += 1
    end
  end
end
```


SysLog
1k
400
200
100
40
20
10
kernel
configd
galileod
com.apple.time
UserEventAgent
coreaudiod
airportd
hidd
Safari
WindowServer

Special Thanks

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

Thank You