Seeing the Big Picture

Quick and Dirty Data Visualization with Ruby
Aja Hammerly
@thagomizer_rb
http://github.com/thagomizer
http://www.thagomizer.com
SUBSTANTIAL
Motivations
http://www.flickr.com/photos/kalavinka/4610308398
WAIT
http://www.flickr.com/photos/oakleyoriginals/3526895658
Solution
Pick a better graph, perhaps traffic versus backups with fake data.
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
Do 4th Fifth Graders Love, Think he is ok, or strongly dislike Justin Bieber

Key
- Love
- Think he is ok
- strongly dislike

14.6
19.5
65.9

http://www.flickr.com/photos/dw2002/5451586738
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
digraph do
  node("B")
end
digraph do
  node("B").label "Hello"
end

Hello
digraph do
    edge "A", "B"
end
digraph do
    edge "A", "B"
    edge "B", "C"
    edge "C", "A"
    save "cycle"
end
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
Shapes
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?
Double check what color brewer calls the three categories.
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
Highcharts 101
Basic Highcharts Page
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]
    }]
  });
});
$(function () {
  $('#chart').highcharts({
    series: [
      {
        data: [10, 5, 3]
      }
    ]
  });
});
```javascript
$(function () {
    $('#chart').highcharts({'
        series: [{
            data: [10, 5, 3]
        }]
    });
});
```
$(function () {
    $('#chart').highcharts(
        
        chart : {
            type: 'column'
        },
        xAxis : {
            categories: [
                "emacs",
                "vim",
                "other" ] 
        },
        series: [
            {
                data: [10, 5, 3]
            }
        ]
    );
});

Chart title

<table>
<thead>
<tr>
<th></th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>emacs</td>
<td>10</td>
</tr>
<tr>
<td>vim</td>
<td>5</td>
</tr>
<tr>
<td>other</td>
<td>3</td>
</tr>
</tbody>
</table>

Highcharts.com
$(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]
      ]
    }]
  });
});
<table>
<thead>
<tr>
<th>line</th>
<th>pie</th>
</tr>
</thead>
<tbody>
<tr>
<td>spline</td>
<td>scatter</td>
</tr>
<tr>
<td>area</td>
<td>area range</td>
</tr>
<tr>
<td>column</td>
<td>area spline</td>
</tr>
<tr>
<td>bar</td>
<td>column range</td>
</tr>
</tbody>
</table>
Leg Curl

Weight

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
    }
]
```javascript
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
  }
]
```
```javascript
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 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

Friday, September 13, 13
#!/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")

"CS 105"
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(".crsdscrip-
tn-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 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 Descriptions

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

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

Friday, September 13, 13
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(".crsdscrptntitle").text.split('.')

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

    desc = description.css("dl dd")[0].text
    pre_reqs = desc.scan(/CS\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(".crsdsrptn-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) }
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+/)

```ruby
course[:pre_reqs] = pre_reqs.map { |s| normalize(s) }
```

```ruby
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\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(".crsdsrptn-header")

course_descriptions.each do |description|
  course = Hash.new
  number, title = description.css(".crsdsrptn-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

Friday, September 13, 13
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