Easy Data Visualization with Graph

Aja Hammerly
Background

• I work for DreamBox Learning
• We build adaptive educational software for children
• 500 lessons with manually specified dependencies
• Need to find patterns and bugs in that data
Which is easier to comprehend?
Or This?
Making pictures by hand is easy
But it doesn't scale
• Time consuming

• Underlying data changes frequently

• Different people want different views
graphviz
to the rescue
DOT

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

digraph example {
    a -> b;
    b -> c;
    a[shape=box]
    b[color=red]
    c
}
Viewing DOT files

- GraphViz
- Tulip
But let's use Ruby
sudo gem install graph
A simple graph

digraph do
  node("B")
end
Nodes with Labels

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

Hello
Adding 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 "example", "png"
  save "example", "jpg"
end

Format list:  http://www.graphviz.org/doc/info/output.html
Now you can build this
Or This
But that's boring
Shapes
digraph do
  node_attribs << triangle
  edge "A", "B"
  edge "B", "C"
  edge "C", "A"
end
Boxes are Special

digraph do
  boxes
  edge "A", "B"
  edge "A", "C"
end
Many Shapes

digraph do
  edge "A", "B", "C"
  triangle << node("A")
  circle    << node("B")
  diamond   << node("C")
end
Color
One Color for All

digraph do
    node_attributes << red
    edge_attributes << blue
    edge "A", "B", "C"
end
digraph do
  node_attribs << filled
  edge "G", "O", "R", "P"
  green  << node("G")
  orange << node("O")
  red    << node("R")
  purple << node("P")
end
Help for the design impaired
Color Schemes

• Uses Brewer Color Schemes
  • http://www.graphviz.org/doc/info/colors.html

• Preview schemes here
  • http://colorbrewer2.com
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
Clustering
Into This
digraph do
  cluster "1" do
    label "cluster 1"
    edge "A", "B"
  end
  cluster "2" do
    label "cluster 2"
    edge "C", "D"
  end
  edge "B", "E"
  edge "D", "E"
end
Building from data
Data

<lessons>
   <lesson id="1" name="MoreLess1to5"/>
   <lesson id="2" name="MoreLess1to10"/>
   <lesson id="3" name="MoreLessEqual1to5"/>
   <lesson id="4" name="MoreLessEqual1to10"/>
   <sequence lesson_id="2" pre_req="1"/>
   <sequence lesson_id="3" pre_req="1"/>
   <sequence lesson_id="4" pre_req="2"/>
   <sequence lesson_id="4" pre_req="3"/>
</lessons>
File.open("sample.xml") do |f|
  doc = Nokogiri::XML(f)
  lessons = doc.xpath("//lesson")
  sequences = doc.xpath("//sequence")

  draw_graph(lessons, sequences)
end
def draw_graph(lessons, sequences):
    digraph do
        lessons.each do |l|
            node(l["id"]).label l["name"]
        end

        sequences.each do |s|
            edge s["pre_req"], s["lesson_id"]
        end
    end
end
On a Larger Scale
Automated Updates
Hudson/Jenkins

• When data changes:
  • Automatically regenerates the graphs
  • Copies graphs to the internal network
  • Sends mail
More Fun with Graph
Visualize Dependencies
Example

$ graph homebrew
Analyzers

- RubyGems
- Homebrew
- FreeBSD Ports
- MacPorts
Dynamic Schema Diagrams
Illustrate History
Animation
Thank You

- Ryan Davis for graph
- Aaron Patterson for Nokogiri
Thank You