Easy Data Visualization with Graph

Aja Hammerly
Background

- I work for DreamBox Learning
- We build adaptive educational software for children
- 700 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
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").label "B"
end
Adding Edges

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

end
Many Shapes

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

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
Help for the design impaired
Color Schemes

- Uses Brewer Color Schemes

- Preview schemes here
  - http://colorbrewer2.org
Color Scheme Example

digraph do
    node_attribs << filled
    node_attribs << "colorscheme=set14"
    node("A").attributes << "fillcolor=1"
    node("B").attributes << "fillcolor=2"
    node("C").attributes << "fillcolor=3"
    node("D").attributes << "fillcolor=4"
    edge "A", "B", "C", "D"
end
Compare 1 to 10

Build up to 10

Build up to 100

Ordering to 100

Addition to 10

Place Value to 99

Addition & Subtraction

Equality
Clustering
Into This
digraph do
  subgraph "cluster_1" do
    label "cluster 1"
    edge "A", "B"
  end
  subgraph "cluster_2" do
    label "cluster 2"
    edge "C", "D"
  end
  edge "B", "E"
  edge "D", "E"
end
Important Note

subgraph "cluster_1" do
  label "cluster 1"
  edge "A", "B"
end
Building from 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>
Extract Data

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
Putting it all together
Thank You

• Ryan Davis for graph
• Aaron Patterson for Nokogiri
• Ben and Shane for organizing the conference
• DreamBox; we're hiring
Thank You