Anderson's Flowers

Dataset:
- Sepal Length/Width
- Petal Length/Width
- Species

Plot:
- X axis is Petal Width
- Y axis is Petal Length
- Color is the Species

Anderson's Flowers

stream flowers
(sepalL, petalW, sepalW, petalL, species, obs)

Plot
- X axis is Petal Width
- Y axis is Petal Length
- Color is the Species
Anderson's Flowers

stream flowers (sepalL, petalW, sepalW, petalL, species, obs)

layer FlowerPlot from flowers
  X: Scale[0, 100](petalL)
  Y: Scale[0, 100](petalW)
  COLOR: BrewerColor(species)
     -> SetAlpha(50, _)

ID: obs
REGISTRATION: "CENTER"
Anderson's Flowers

stream flowers
(sepalL, petalW, sepalW, petalL, species, obs)

layer FlowerPlot
from flowers
  X:* Scale[0, 100](petalL)
  Y:* Scale[0, 100](petalW)
  COLOR: BrewerColor(species)
  -> SetAlpha(50, _)
ID: obs
REGISTRATION: "CENTER"
Goals

- Direct
- Compact
- Deterministic
- Incremental/Dynamic
- Plays well with others
- Easy is easy, Hard is possible
Flowers: guide trendline from FlowerPlot

JUNG Spring Force Embedding

import JUNG
stream VertexList (parent, child)
canvas Main
guide pointLabels from Nodes ID
COLOR: @Color{GRAY60}

stream Prime ()
from VertexList
() : Layout.add(parent, child)

layer Nodes
from VertexList
ID: child
REGISTRATION: "CENTER"
FILL_COLOR: @Color{BLUE}
(SHAPE, SIZE): ("ELLIPSE", 5)
(X,Y): Layout.query(child)

layer Edges[LINE]
from VertexList
ID: Concatenate(parent, child)
(X.1, Y.1): Layout.query(parent)
(X.2, Y.2): Layout.query(child)
PEN_COLOR: @Color{GRAY30,80}

operator Layout:FRLayout
Crimean Rose

See Test

ompi-nightly-trunk : 1.3a1r18049

Legend
- Green: Passed
- Red: Failed
- Yellow: Not run
- Orange: Missing output
- Black: Missing tests

Initial development
TextArc

Processes raw text from Project Guttenberg

Makes use of:
- Custom operators
- Map/Gather
- Stream/Stream transformer

TextArc: 55 lines (8pt font)

import Layouts
import Geometry as Geo

stream RawLines(text)
from RawLines
line: Counter()
words: text
stream RawWords (line, word)
from Lines
line: line
word: Split(text, "\s+") >> Strip(_) -> ToLower(_)
stream PrimeCenter ()
from Words
() : WordCount(word) -> ExtendTuple(_, word) -> LineIndex.put(word, *)
() : WordCount(word)

layer Border[TEXT]
from Lines
ID: line
TEXT: text
(X,Y):* Layout(line)

layer Center[TEXT]
from Words
ID: word
TEXT: word
(X,Y):* Centroid(word) -> Contract(X,Y)
REGISTRATION: "CENTER"
(COLOR, FONT):*
[count] WordCount.query(word) -> Max[range: ALL](_)
[freq] Divide[count[_],_]
[color] HeatScale[hot: "BLUE", cold: "GRAY30"](freq[_])

operator WordCount base Count
operator LineIndex base Dict[fields: "lines"]
operator Centroid (word) -> (X,Y)
(ALL) =>
(X: LineIndex(word) -> ToTuple(_) >> Layout.query(_)) ->
Select[field: 0(*)] -> Average[range: LAST](LAST)
(Y: LineIndex(word) -> ToTuple(_) >> Layout.query(_)) ->
Select[field: 1(*)] -> Average[range: LAST](LAST)
operator Layout base CircularLayout[start: 0, size: 10, ratio: .75]
operator Contract base Geo::Scale[by: .94]