Hands On: Plot the Degree Histogram Reading | Coursera

Hands On: Plot the Degree Histogram

1. Import the BreezeViz library
2. Define a function to calculate the degree histogram
3. Calculate the probability distribution for the degree histogram
4. Graph the results

Import the BreezeViz library

```
import breeze.linalg._
import breeze.plot._
```

Define a function to calculate the degree histogram

Define a function to create a histogram of the degrees. The function is nearly identical to how the histogram was computed in the previous hands on exercise. Remember to only include countries!

input:

```
def degreeHistogram(net: Graph[PlaceNode, Int]): Array[(Int, Int)] =
  net.degrees.
    filter { case (vid, count) => vid >= 100 }.
    map(t => (t._2,t._1)).
    groupByKey.map(t => (t._1,t._2.size)).
    sortBy(_._1).collect()
```

output:

```
degreeHistogram: (net:
org.apache.spark.graphx.Graph[PlaceNode,Int])Array[(Int, Int)]
```

Calculate the probability distribution for the degree histogram

Get the probability distribution (degree distribution) from the degree histogram by normalizing the node degrees by the total number of nodes, so that the degree probabilities add up to one.

input:

```
val nn = metrosGraph.vertices.filter{ case (vid, count) => vid >= 100
}.count()
```

output:

```
nn: Long = 28
```

input:

```
val metroDegreeDistribution = degreeHistogram(metrosGraph).map({case(d,n) =>
(d,n.toDouble/nn)})
```

output:

```
metroDegreeDistribution: Array[(Int, Double)] = Array((1,0.6428571428571429),
(2,0.14285714285714285),
(3,0.07142857142857142), (5,0.07142857142857142), (9,0.03571428571428571),
(14,0.03571428571428571))
```

Graph the results

Plot degree distribution and the histogram of vertex degrees.

input:

```
val f = Figure()
val p1 = f.subplot(2,1,0)
val x = new DenseVector(metroDegreeDistribution map (_._1.toDouble))
val y = new DenseVector(metroDegreeDistribution map (_._2))

p1.xlabel = "Degrees"
p1.ylabel = "Distribution"
p1 += plot(x, y)
p1.title = "Degree distribution"



val p2 = f.subplot(2,1,1)
val metrosDegrees = metrosGraph.degrees.filter { case (vid, count) => vid >=
```

```
100 }.map(_._2).collect()

p2.xlabel = "Degrees"
p2.ylabel = "Histogram of node degrees"
p2 += hist(metrosDegrees, 20)
```

output: