

Project: Map Reduce and Hadoop
Unity Id : akagrawa

Problem 3: The goal of this problem is to make sure how you could compose more sophisticated analytical pipelines and apply them to large-scale real-world graphs such as those from Amazon, Wiki, Facebook, etc. Specifically, you are asked to expand the solution to Problem #2 from calculating the degree of each vertex to calculating the FREQUENCY of vertex degree (or degree distribution), namely, the number of vertices of a certain degree.

Code: Please refer README.md for the code description and commands for the execution.

Approach: Implemented a java hadoop map-reduce program to count the degree of the vertices of the given graph. With the class VertexDegree.java configured a map-reduce job which call VertexDegreeMapper.java for mapper action and VertexDegreeReducer.java for reducer action. In VertexDegree.java the key-value pair is accessed and an intermediate key-value pair is produced. This intermediate key value pair is sorted and combined using VertexDegreeReducer.java functions and from there it is written in the output folder. This output folder is served as an input for DegreeFrequency.java for same set of mapper-reducer functions to calculate the degree frequency. The final output is saved as degree-frequency key value pair. Single mapper and reducer were configured for the above jobs. A master controller file Executor.java is called for calling of the above two jobs. In order to load separate graphs in HDFS, the input file must contain that particular graph for a job execution period.

Input Files: The graph data-set given for amazon, google, facebook etc were used for the performance evaluation. All the degree frequency computation is performed separately for all the individual graphs.

Job Analysis : Following is the portion of the job analysis output of the hadoop framework at port 50030. For more details please find hadoop-jobXXX-history.pdf file inside files folder. Two jobs are performed for the above task, hence two job-history files were generated.

Time taken for Job 1 (Vertex-Degree) Computation : 1 min 34 secs

Time taken for Job 2 (Degree-Count) Computation : 28 secs

a) Vertex Degree Job

Hadoop job_201409180041_0001 on localhost

User: akagrawa

Job Name: vertexDegreeCount

Job File:

hdfs://localhost:54310/opt/hadoop/tmp/mapred/staging/akagrawa/.staging/job_201409180041_0001/job.xml

Submit Host: bn20-75.dcs.mcn.org

Submit Host Address: 152.46.20.75

Job-ACLs: All users are allowed

Job Setup: [Successful](#)

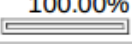
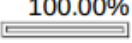
Status: Succeeded

Started at: Thu Sep 18 00:42:32 EDT 2014

Finished at: Thu Sep 18 00:44:07 EDT 2014

Finished in: 1mins, 34sec

Job Cleanup: [Successful](#)

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00% 	6	0	0	6	0	0 / 0
reduce	100.00% 	1	0	0	1	0	0 / 0

b) Degree Frequency Job

Hadoop job_201409180041_0002 on localhost

User: akagrawa

Job Name: degreeFrequencyCount

Job File:

hdfs://localhost:54310/opt/hadoop/tmp/mapred/staging/akagrawa/.staging/job_201409180041_0002/job.xml

Submit Host: bn20-75.dcs.mcn.org

Submit Host Address: 152.46.20.75

Job-ACLs: All users are allowed

Job Setup: [Successful](#)

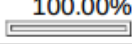
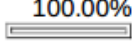
Status: Succeeded

Started at: Thu Sep 18 00:44:08 EDT 2014

Finished at: Thu Sep 18 00:44:37 EDT 2014

Finished in: 28sec

Job Cleanup: [Successful](#)

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00% 	2	0	0	2	0	0 / 0
reduce	100.00% 	1	0	0	1	0	0 / 0

(1) For the Stanford graphs provided to you, test and report which graphs are scale-free, namely whose degree distribution follows a power law, at least asymptotically. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as

$$P(k) \sim k^{-\gamma}$$

where γ is a parameter whose value is typically in the range $2 < \gamma < 3$, although occasionally it may lie outside these bounds.

A scale-free graph is one with a power-law degree distribution. For an undirected network, we can just write the degree distribution as

$$P_{\text{deg}}(k) \sim k^{-\gamma}$$

where γ is an exponent between 2 and 3. This form of $P_{\text{deg}}(k)$ decays slowly as the degree k increases, increasing the likelihood of finding a node with a very large degree.

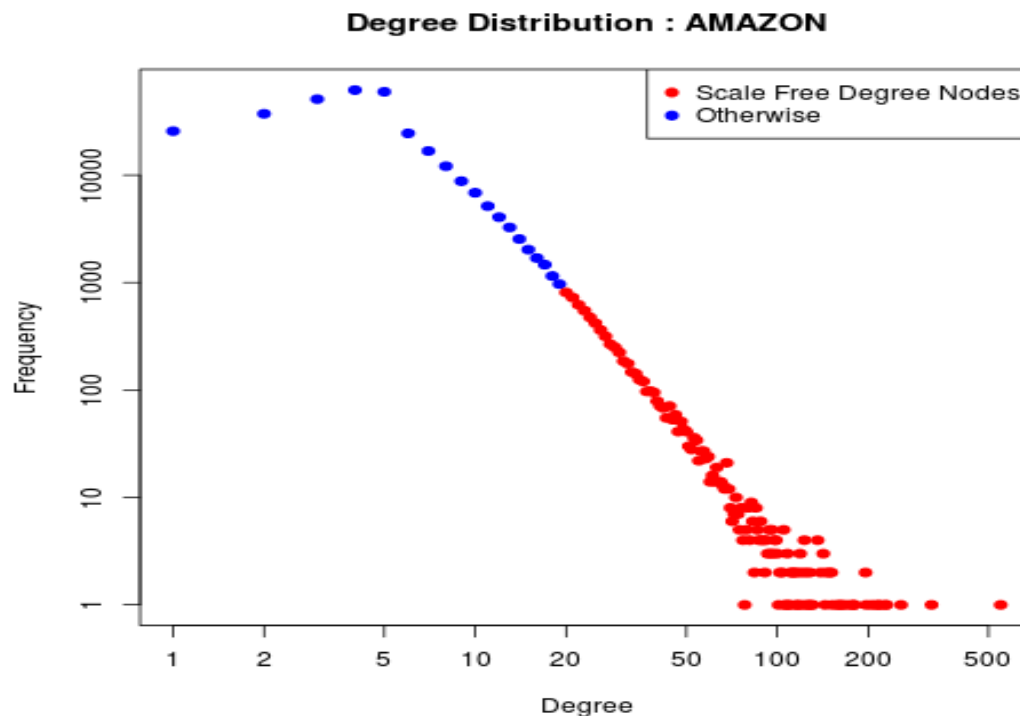
In order to check, if the graphs above are scale free, the following steps are performed.

1. Computation of degree frequency of the graphs (facebook, google, amazon etc) using the map-reduce jobs described in the first part of this problem.
2. Plotted a log-log plot using R script where each degree points is tested against power law, i.e. the ratio of a k th degree frequency wrt to all the degree frequency should lie between $(1/k)$ power of 2 and 3.
3. The degree points following this power law criterion are marked **red** and those which do not lie in this boundation are marked **blue**.

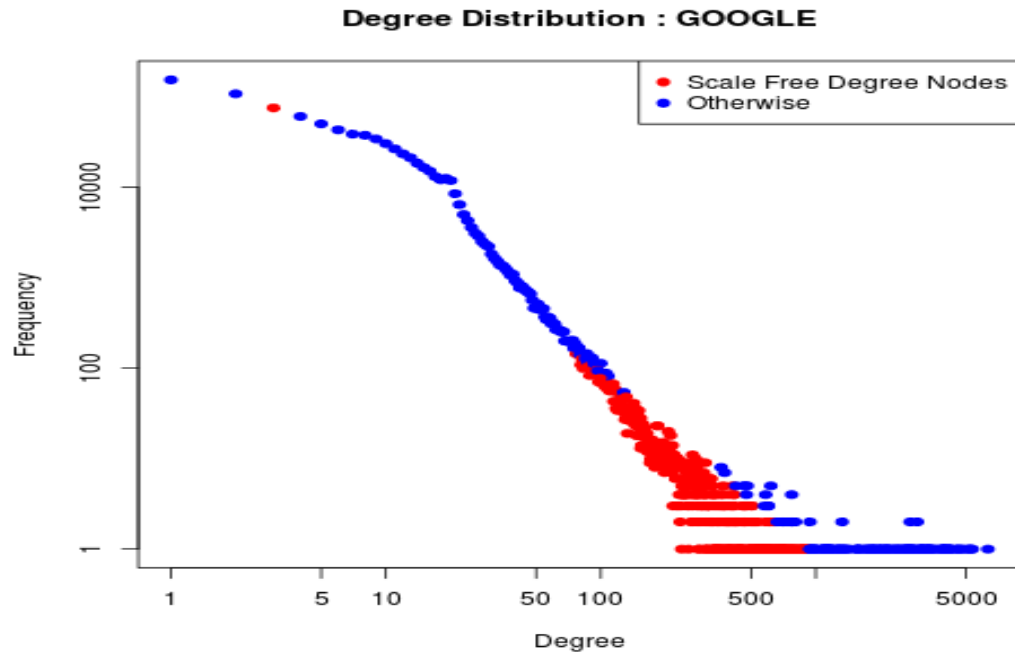
In conclusion, graphs **log-log plot** of degree distribution will reveal that the graph is scale free or not.

For Details, please find the same plots in the files folder.

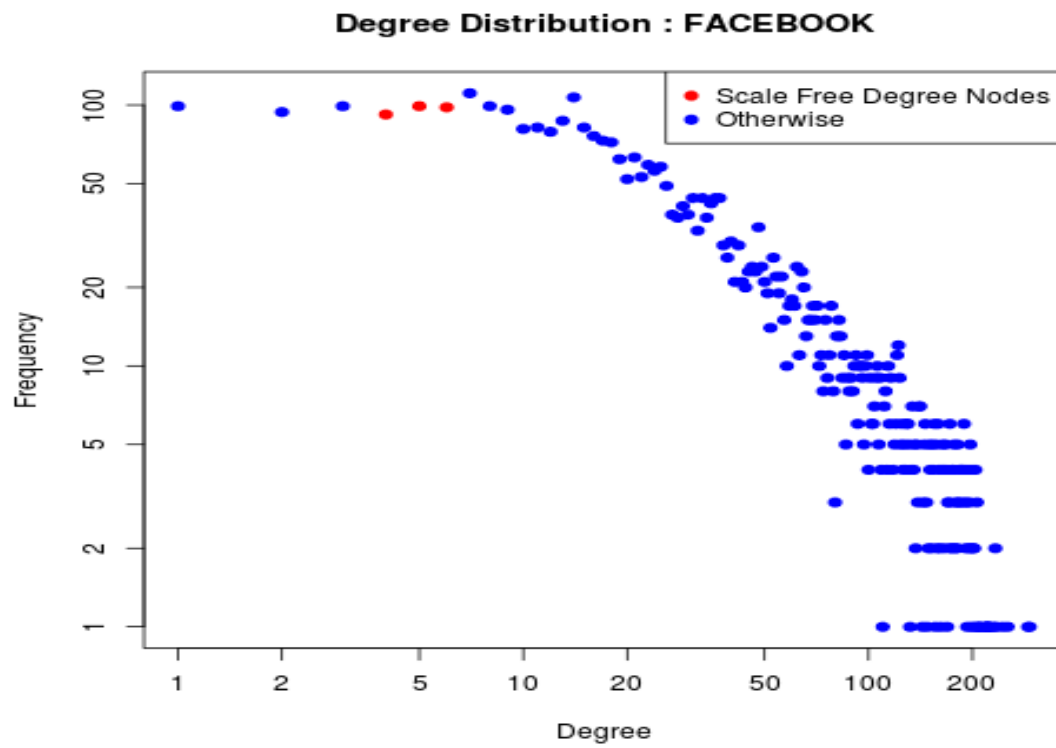
a) **Amazon Graph Plot : Scale Free (Contains Red points)**



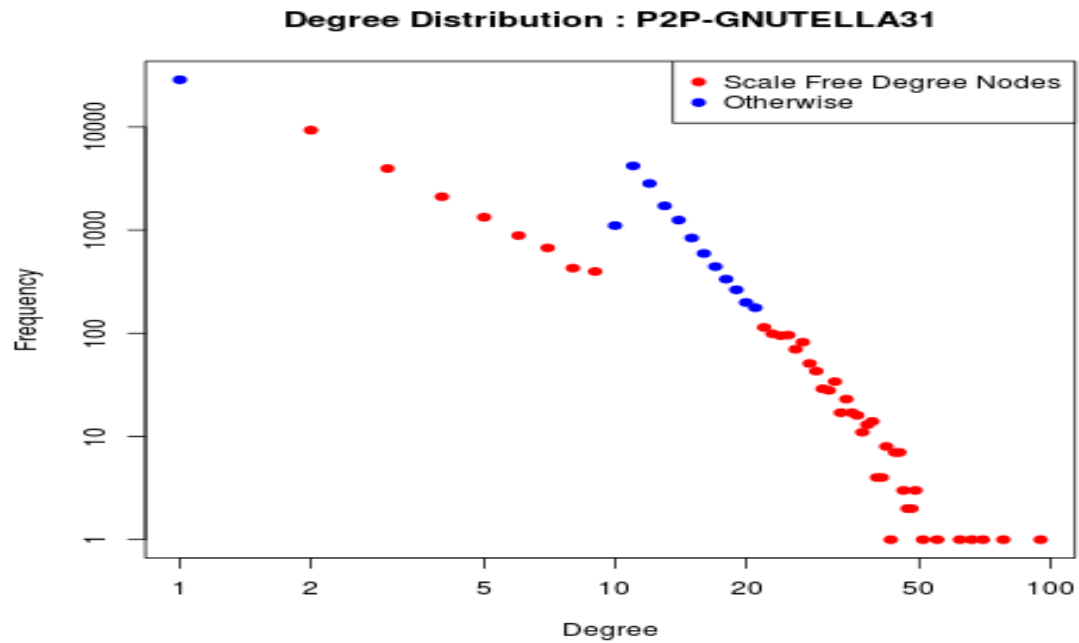
b) **Google Graph Plot** : Scale Free (Contains Red points)



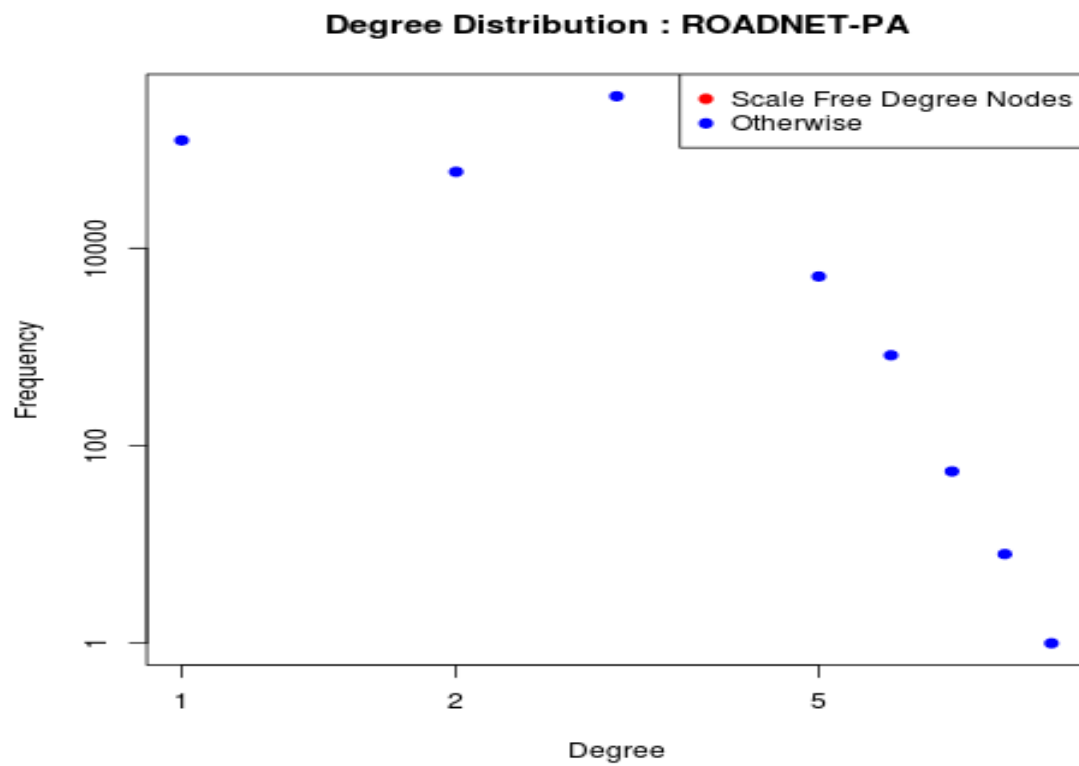
c) **Facebook Graph** : Not scale free for this sample (Very less red points)



d) **p2p-Gnutella31 graph** : Scale Free (Contains Red points)



e) **RoadNet-PA graph** : Not scale free (Contains no red points)



These graphs are generated using R plot functions. Please refer project1_p3.R script in the code folder and README.md for the code description and execution.

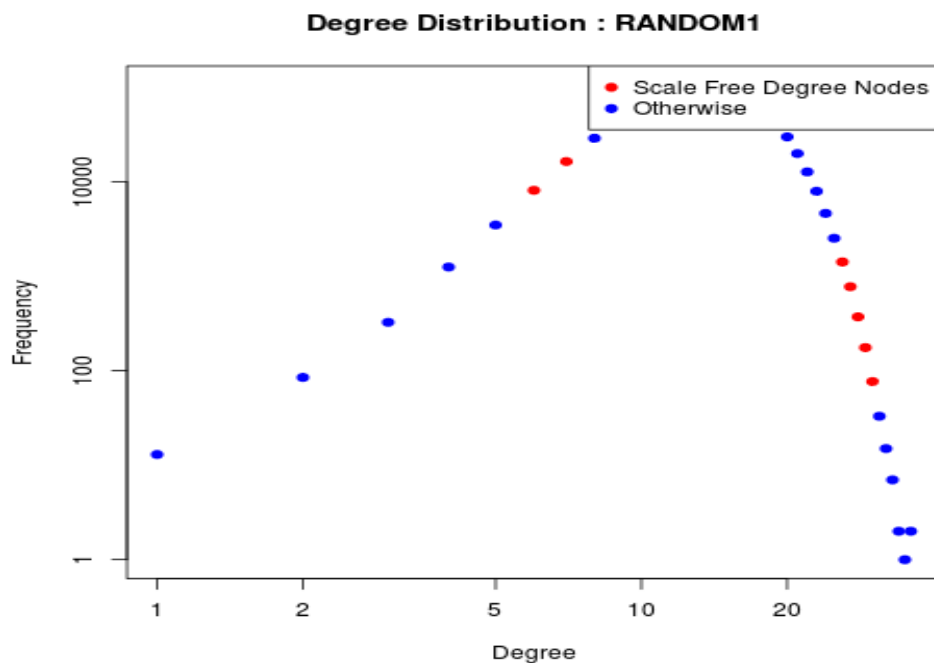
(2) Generate one million node random graphs (multiple) using R-MAT graph generator with default parameters (see, announcement 'VCL Image for Project 1' for location of R-MAT). Are those generated graphs scale-free?

RMAT is a graph generator framework which exposes certain APIs to generate random, SSscale and scale-free graphs. In order to generate those graphs, we need to modify its config for the no of vertices and edges. It also provides probabilistic distribution configuration of the edges for the graph to be generated which leads us to generate scale free graphs as well.

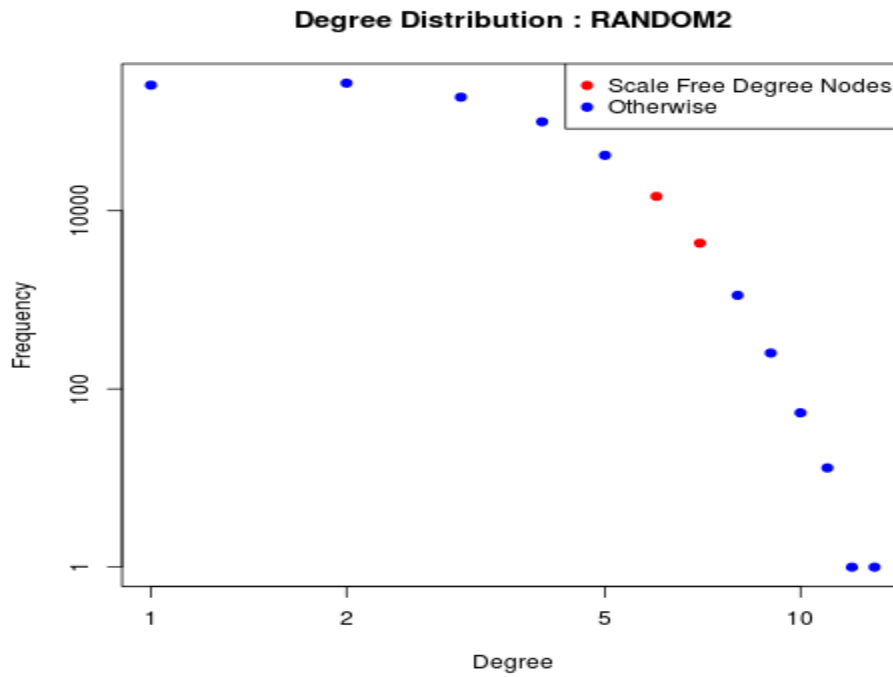
Steps are as follows :

1. I have generated 2 random graphs and 1 rmat graph using the RMAT commands with different config. Please refer README.md for the RMAT commands.
2. After that, the transformation of the above output graphs from the RMAT api was needed since it contains many extra information. Hence the transform() function in project1_p3_RMAT.R is loaded in R session for the vertex-vertex transformation. Again, please refer README.md for the same.
3. These transformed graph files are compatible for the input of DegreeFrequency problem in the first part of this problem. Hence we use map-reduce programs to transform it again to obtain degree frequency of each graphs.
4. These graphs are then again plotted in log-log plot to test for scale-free graphs and hence the below plots are generated with n = no of vertices and m = no of edges.

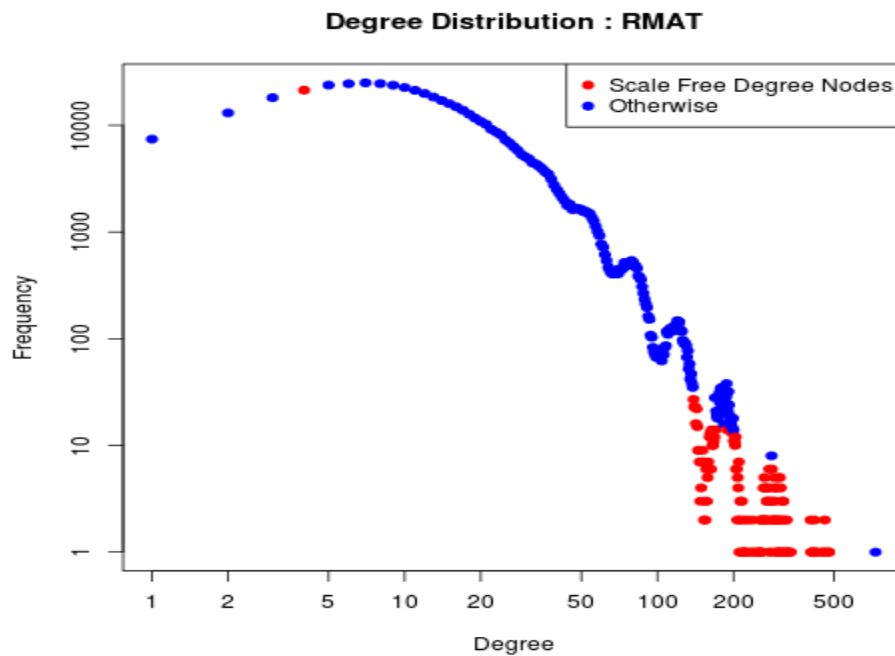
a) **Random Graph 1** : n = 1 million, m = 7.05 million = **somewhat Scale Free** since many points are red. But higher degrees are very low, and degree distribution is somewhat the same.



b) **Random Graph 2** : $n = 1$ million, $m = 1.05$ million = **Not a scale free graph** (Very few red points)



c) **Rmat Graph 1** : $n = 1$ million, $m = 5$ million (This is with construct a **scale free** graph)



References:

- <http://www.cse.psu.edu/~madduri/software/GTgraph/gen.pdf>
- http://mathinsight.org/scale_free_network
- <http://blogs.msdn.com/b/avkashchauhan/archive/2012/03/29/how-to-chain-multiple-mapreduce-jobs-in-hadoop.aspx>
- <http://stackoverflow.com/questions/17551193/r-color-scatter-plot-points-based-on-values>