Lab 3-Parallelize
Stat 215A, Fall 2014

Xiang (Lisha) Li

October 21, 2014

1 Introduction

This lab will be evaluated primarily on your code. This writeup can be very minimal. Of course, feel free to write more if you feel it is necessary.

1.1 Similarity Measures

The paper[1] gives a good summary of similarity measures used in the literature in the language of linear algebra, thus clarifying what these measures are really doing and how they compare to each other. To paraphrase some of their exposition in order to motivate our choices of similarity measure, let's first define the cluster adjacency matrix C for a particular clustering with k clusters:

Definition 1.1 (Cluster Adjacency Matrix)**.**

$$C_{ij} = \begin{cases} 1 & \textit{if } i \textit{ and } j \textit{ are in the same cluster.} \\ 0 & \textit{otherwise} \end{cases}$$

Let C^1, C^2 be two cluster adjacency matrices from the same dataset. Then $< C^1, C^2 >$ counts the number of points clustered together. Since this is a dot product, we have a natural way to normalize the quantity, otherwise known as defining the correlation between the two sets of labeling (if we view the columns of C^i as points in a vectorspace). That is

$$cor(C^1, C^2) := \frac{< C^1, C^2 >}{||C^1|| \cdot ||C^2||},$$

where $|| \cdot ||$ is the norm induced by the inner product. This similarity measure is credited to Fowlkes and Mallows. Based on our linear algebra perspective/geometric, one can clearly see what it strives to measure. Two further similarity measures, called the Jaccard coefficient and matching coefficient, stay close to this geometric definition.

Definition 1.2 (Jaccard coefficient)**.**

$$J(C^1, C^2) = \frac{< C^1, C^2 >}{||C^1|| + ||C^2|| - < C^1, C^2 >}$$

Definition 1.3 (Matching coefficient)**.**

$$M(C^1, C^2) = 1 - \frac{1}{n^2}||C^1 - C^2||^2$$

I will use the original similarity correlation measure for this lab. For the code, I realized that creating a copy of C^i was expensive computationally, and not necessary. Instead, I used C++ to read the adjacency information straight off of the data frames, and computed a running sum of the lower triangular regions (relying on the fact that C^i are symmetric so their point wise products are as well).

2 Results

2.1 Parallelizing Code

I choose to parallelize my code at several points, an outline of the function calls given below. These can also be found in the comments of "optimize.R" with code lines.

OUTLINE OF FUNCTIONS

1. Dataframe subsample: Df.subsample().
   (a) input: (data frame, number of subsamples, subsample size in fractions). example: (lingBinarize, 40, 0.8)
   (b) output: list of dataframes. Each dataframe is a subsample of original data frame.
   (c) this code was parallelized with new random seed per foreach iteration.
2. k means: k.means().
   (a) input: (dataframe, number of subsamples, cluster size, subsample size). same as Df.subsample()
   (b) output: a list of dataframes per subsample, that only contains an ID column, and a "k.means" column, giving the clustering information. We apply kmeans with 25 random starts and max.iter 1000.
   (c) This function is also parallelized with different random seeds.
3. correlation similarity: cor.sim().
   (a) input: two dataframes, with the cluster information in a column named "k.means".
   (b) output: similarity between clustering of the two subsamples, defined using similarity correlation above. There is no writing of the clustering matrix, instead two for loops cary a total sum of the lower triangle of the element wise product.
   (c) Calls on clusteradjacency.cpp file to do the above two for loops.
4. k.stabalize().
   (a) input: (dataframe, number of subsamples, cluster size, subsample size)
   (b) output: list of correlation measures for each unique pair of subsamples for a given k.
   (c) this function calls on all 3 functions above and parallelized the for loops.
5. Shell script to parallelize application of k.stablize() for k in 1:k.max.

2.2 Pictures and Graphs for results

I choose to evaluate this stability algorithm on k means clustering with L^2 penalty. This was used on the large lingBinary for my my lab 3, so it is a continuation of checking robustness of that lab. Of course, the code can be adapted to other clustering methods, where extra boosts in runtime can be achieved by using hierarchical methods (since only one cluster call needs to be done for all k per subsample).

First I looked at 20 subsamples of 80%of the original dataset. This generated 190 unique pair comparisons. Looking at the histogram, regardless of how I adjusted the bin width, I was not satisfied and ran the program again this time taking 40 subsamples, thus generating 780 unique pair comparisons per choice of k. What I learned from the first run of the program was that it was not useful to look beyond 9 clusters. Instead, on the second run, I focused on 2-9 clusters. The numbers in the title of each histogram plot indicate the choice of k (number of clusters).

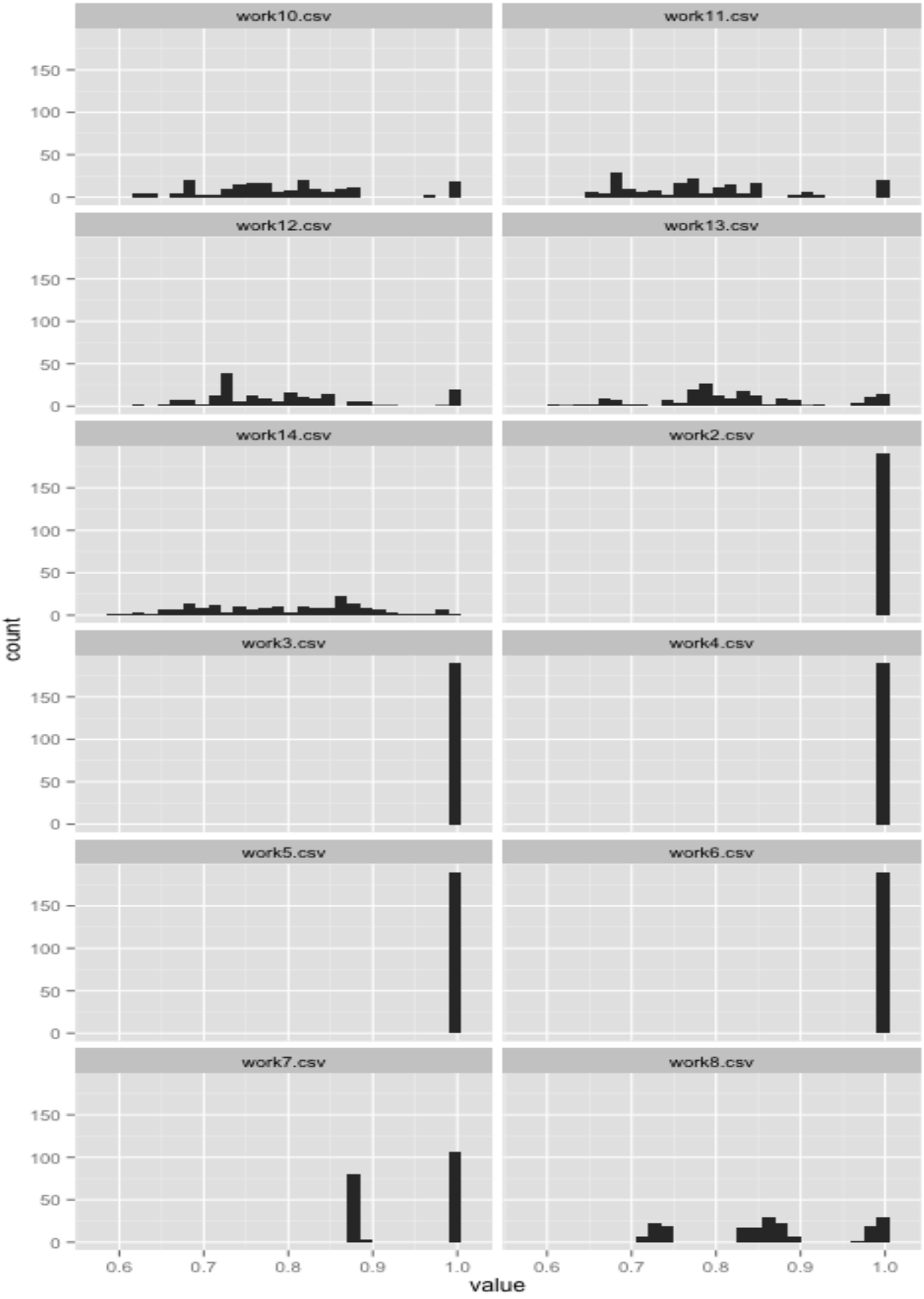

Figure 1: 20 subsamples per k: 190 distinct comparisons

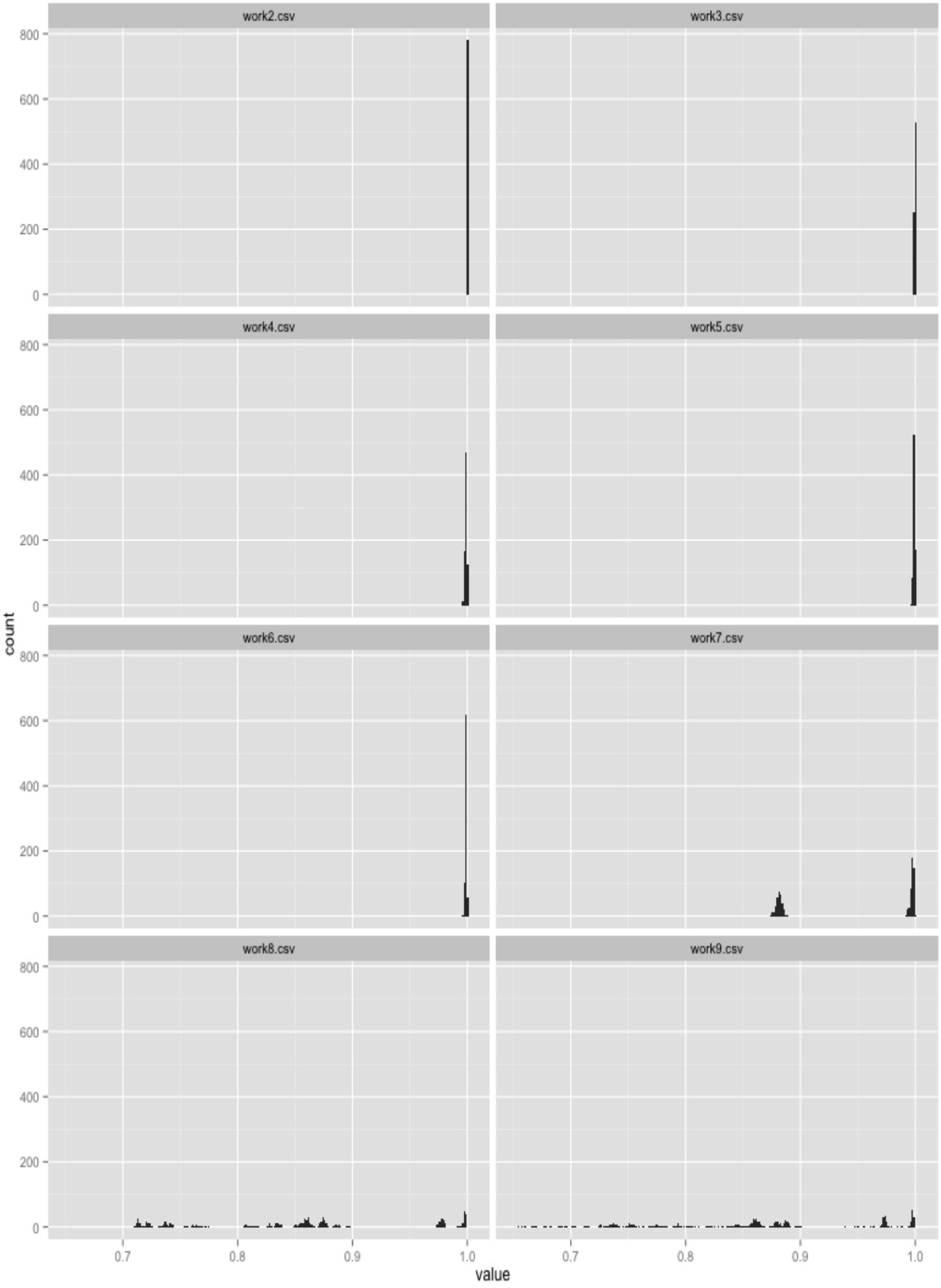

Figure 2: 40 subsamples per k: 780 distinct comparisons: Finer bin width

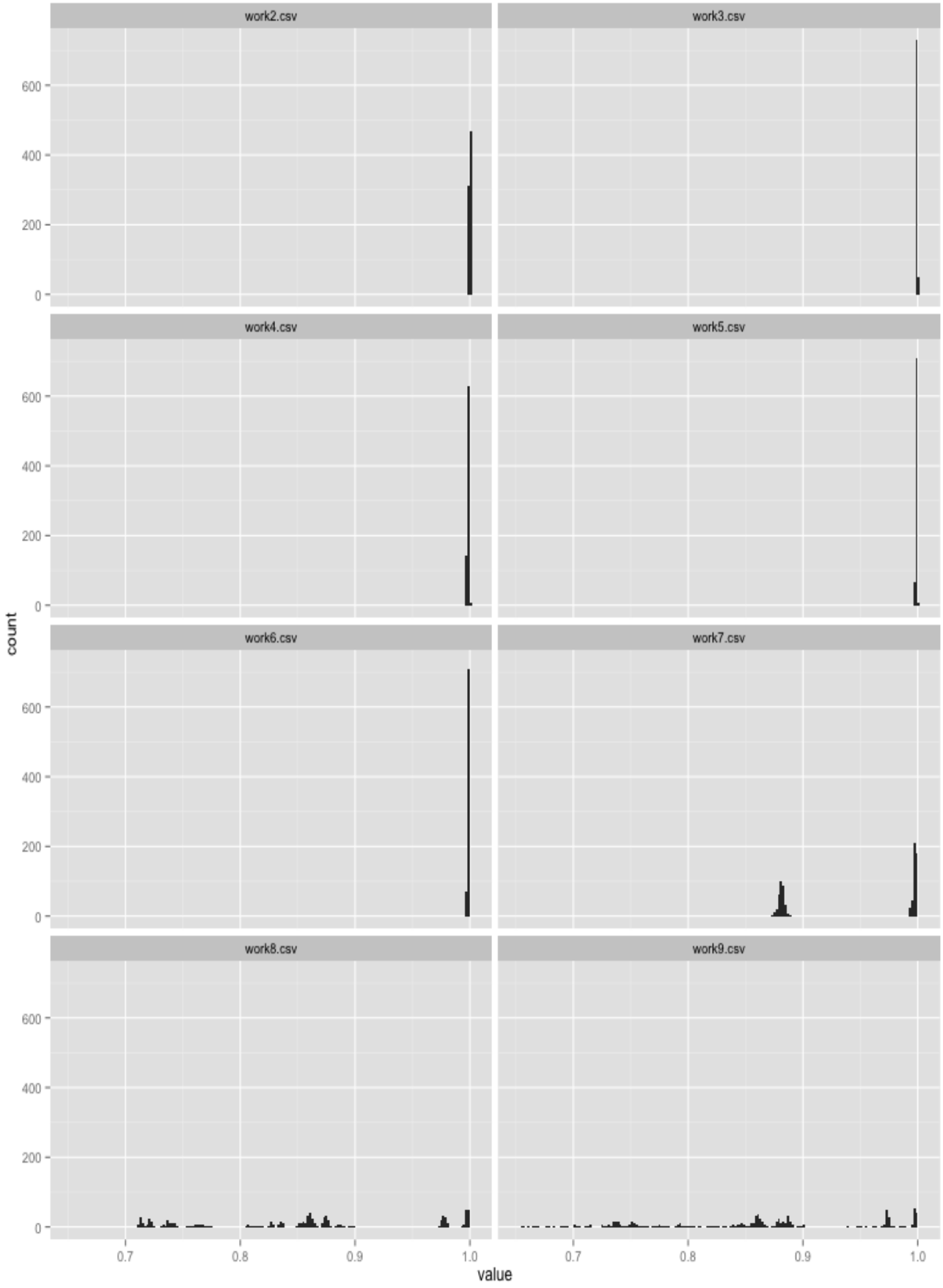

Figure 3: 40 subsamples per k: 780 distinct comparisons: Coarser bin width

2.3 How many clusters would you choose, and why?

Judging by both sets of histograms, There is a transition between 6 and 7 clusters. It would be even nicer if I can run it on 0.9 proportion and 0.7 proportion samples just to check robustness of that choice.

3 Do you trust this method? Why or why not?

The algorithm provided by Ben-Hur, et al., is an incredibly robust way to judge cluster results. I applaud the authors of the paper for striving to generate a distribution from the correlation similarities, and using major shifts in this distribution as a measure of how stable a clustering is. The transition between the histogram distributions from being concentrated at 1 is a good indicator of how arbitrary the clustering gets after k grows to beyond a transitional point.

One important caveat is that it is theoretically possible for the histograms to have more than one phase transition, that is, we can notice the clustering results concentrate at 1 for more than one k. For instance, there could be 20 "true" clusters, but 10 good bigger clusters that subsumes the 20 finer clusters. Then the histogram information can potentially look random between 10 and 20 depending on the dataset. However, because we are familiar with the domain from which the data comes, I trust that 6 is a reasonable cluster size, and informative for regional dialect relationships with geography. If there was more computational time, I would try run this algorithm for bigger k.max just to be there is not interesting tight clustering behaviour for larger k.

References

[1] Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Pacific Symposium on Biocomputing, pp. 617.