Lab 3 - Parallelizing k-means Stat 215A, Fall 2017

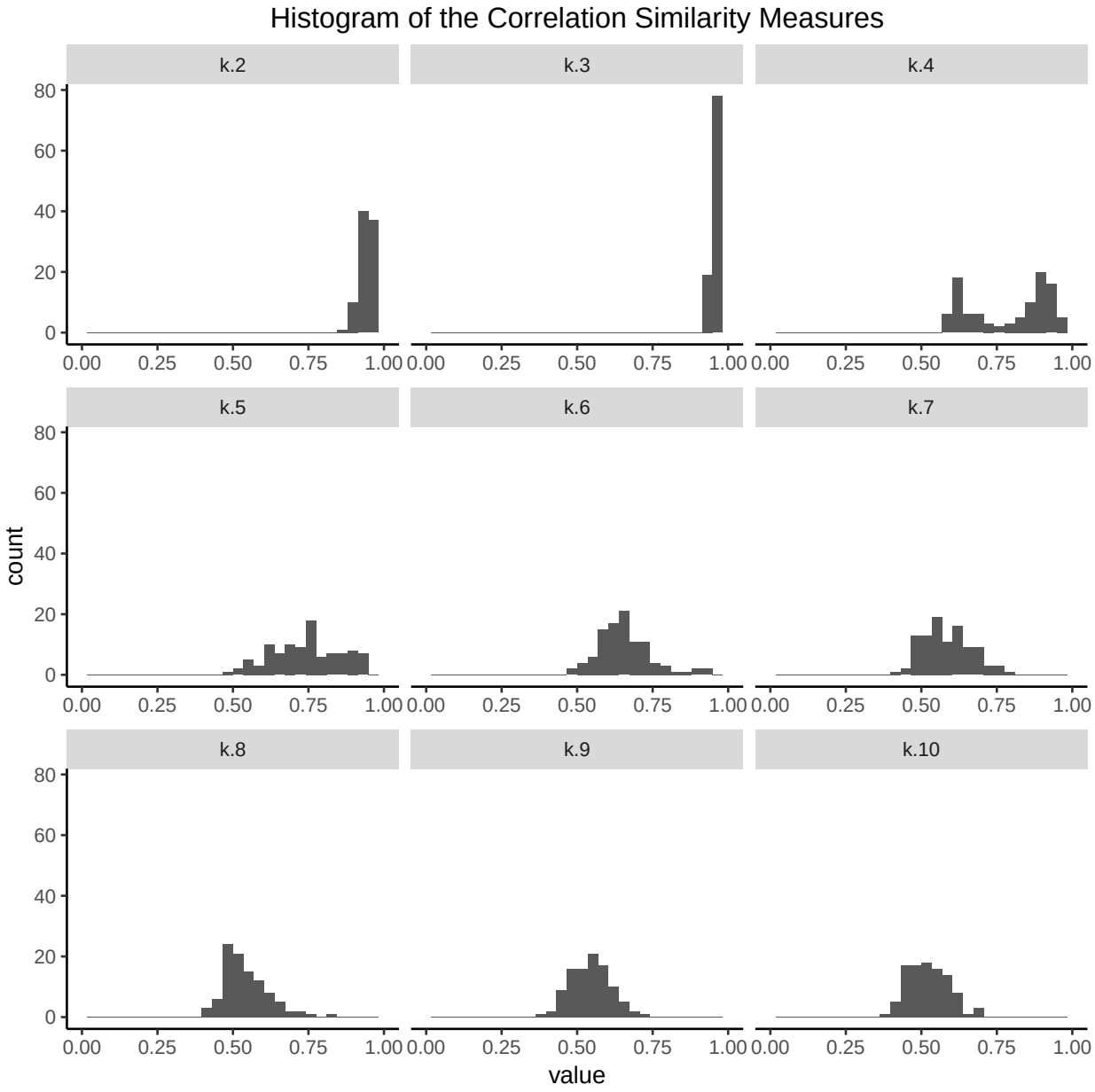
October 23, 2018

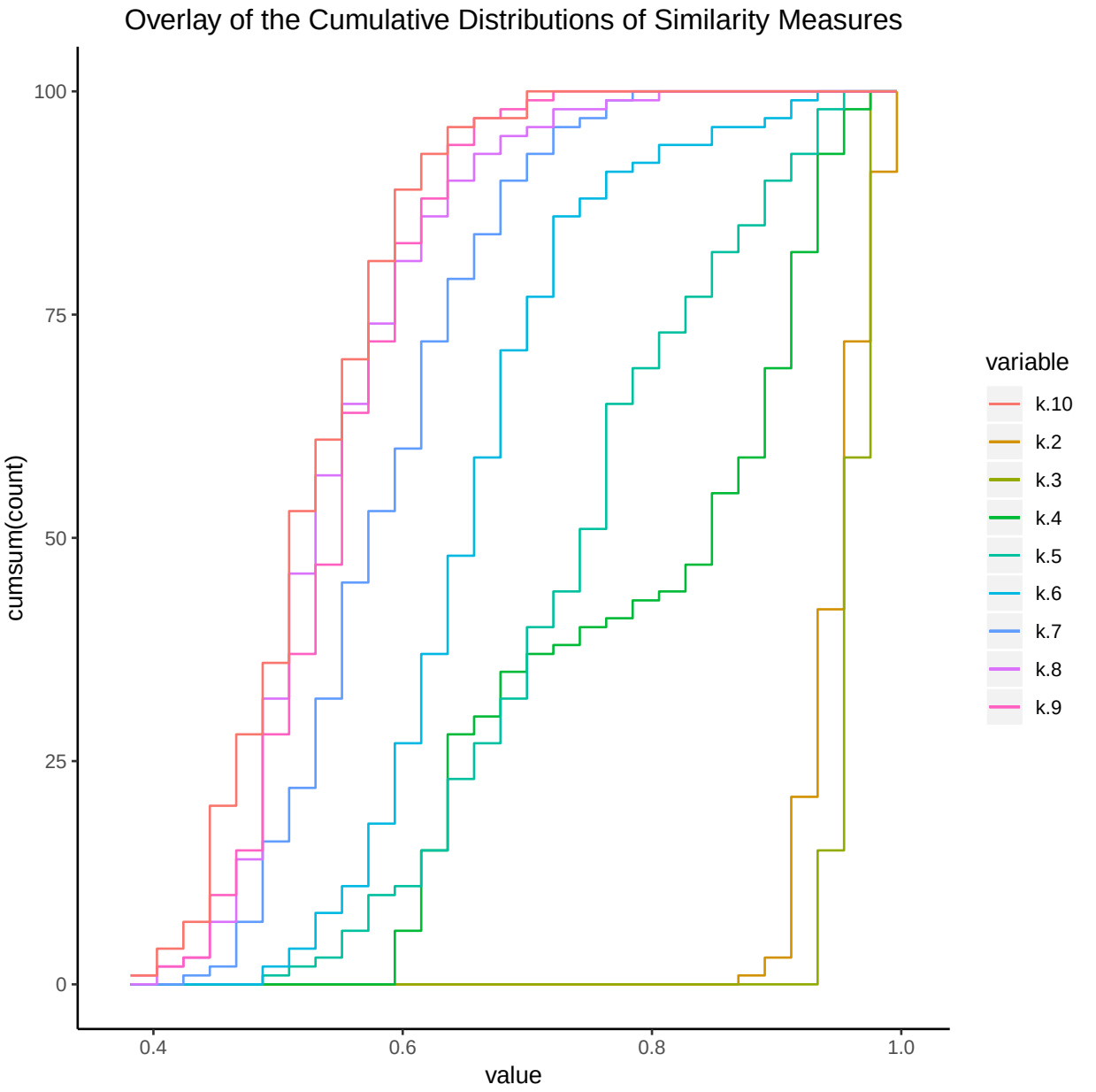
Compare C++ and R Version of Similarity Matrix

Based on the running time, apprently cpp version of the similarity matrix is more efficient in time. For cpp implementation, I didn't store the matrix but rather passed the vectors to similarity calculation function directly. In R, I stored the matrix first and then conducted column by column multiplication and addtion. Computing without storing the matrix is the major step that boosts up the speed. Also, since cpp stores vector as a piece of memory without abstraction, it hits the processor directly. Therefore, cpp runs for loops much faster than R.

Table 1: Running time of Similarity Measures (top:R bottom:Cpp)

min	lq	mean	median	uq	max	neval
0.8318798	0.8640928	0.9313386	0.9413814	0.9551097	1.470284	100
min	lq	mean	median	uq	max	neval
0.0008296	0.0022539	0.0026401	0.0023489	0.0027209	0.010583	100


Histogram of the Correlation Similarity Measures
k.2
k.3
k.4
k.5
k.6
k.7
k.8
k.9
k.10
count
value
0
20
40
60
80
0.00
0.25
0.50
0.75
1.00



Optimal Number of Clusters

Based on the plots, apprently k = 3 is the most optimal number of clusters. Since after k=3, the correlation starts to spread out rather than center around 1. This method is trustworthy because first the result makes sense based on our knowledge from last lab. Second, the way it accesses stability by evaluting the cosine distance between two cluster-index matrix is soild in a mathematical sense.