TIE-22307 2018 Data-Intensive Programming

Programming assignment

Pauli Ahonen, pauli.ahonen@student.tut.fi, 245901

Olli Sola, olli.sola@student.tut.fi, 240327

Teemu Pöytäniemi, teemu.poytaniemi@student.tut.fi, 240056

General

We implemented all tasks. You can run this program in command line by going to project folder and using command: sbt "run -task 'task number'". Like sbt "run -task 1". **Notice that our assignment is named project.**

Workloads:

Pauli: tasks 4,6

Teemu: tasks 5,1,6

Olli: tasks 2,3,6

Task 1

We implemented k-means clustering in this task by getting x and y coordinates from data and using kmeans fit to find cluster centers. If one or both coordinate was missing, that data row was filtered and not used. Results are written to results/basic.csv.

Task 2

In this task we added third dimension which turned out to be fourth dimension, because day of a week does not fit into Cartesian coordinates. We used this solution https://datascience.stackexchange.com/questions/8799/boundary-conditions-for-clustering to transform day to x,y coordinates and scaled the resulting circle to approximately same scale as the crash coordinates are. After finding cluster centers we turned resulting day coordinates back to days by calculating distances to know day coordinates and choosing the nearest one. Results are written to results/task2.csv.

Task 3

We planned to implement k-means streaming by choosing some starting cluster centers and adjusting them as new data points are added. We need to keep records of every cluster center m_i and number of points every cluster has n_i. When new point is added, it is assigned to the cluster with nearest cluster center. n is incremented for that cluster and m is changed with formula **m = m + (1/n)*(x - m)**, where (x – m) is distance between added point and existing center. We can see from the formula that if there are many points already in that cluster, center is moved only slightly and if there are only a couple points, new point affects to the position of the center more.

Task 4

The basic idea of this task was to do the kmeans calculation with scala and then examine the data with MATLAB. We used KMeansModel's computeCost method which calculates the sum of squared distances of points to their nearest center. For basic case we used k values 2-400 and 250-400 as seen in figures 1 and 2 below.

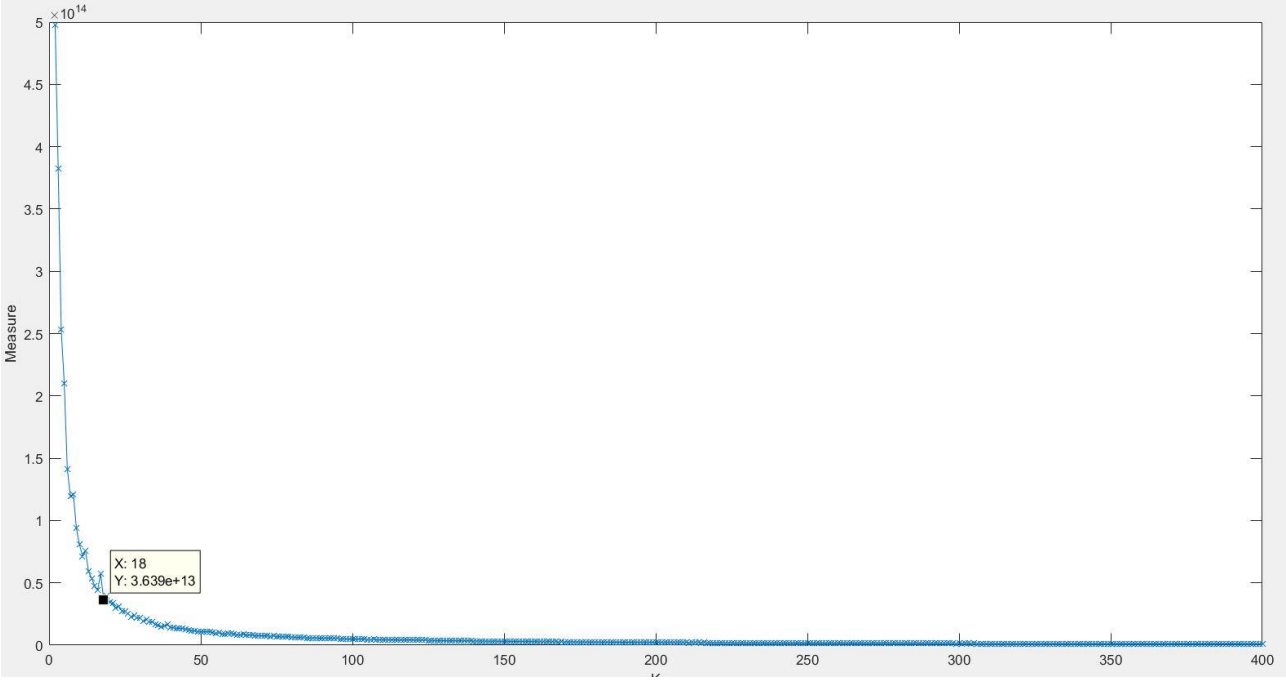

Figure 1. Elbow for basic case with k going from 2 to 400

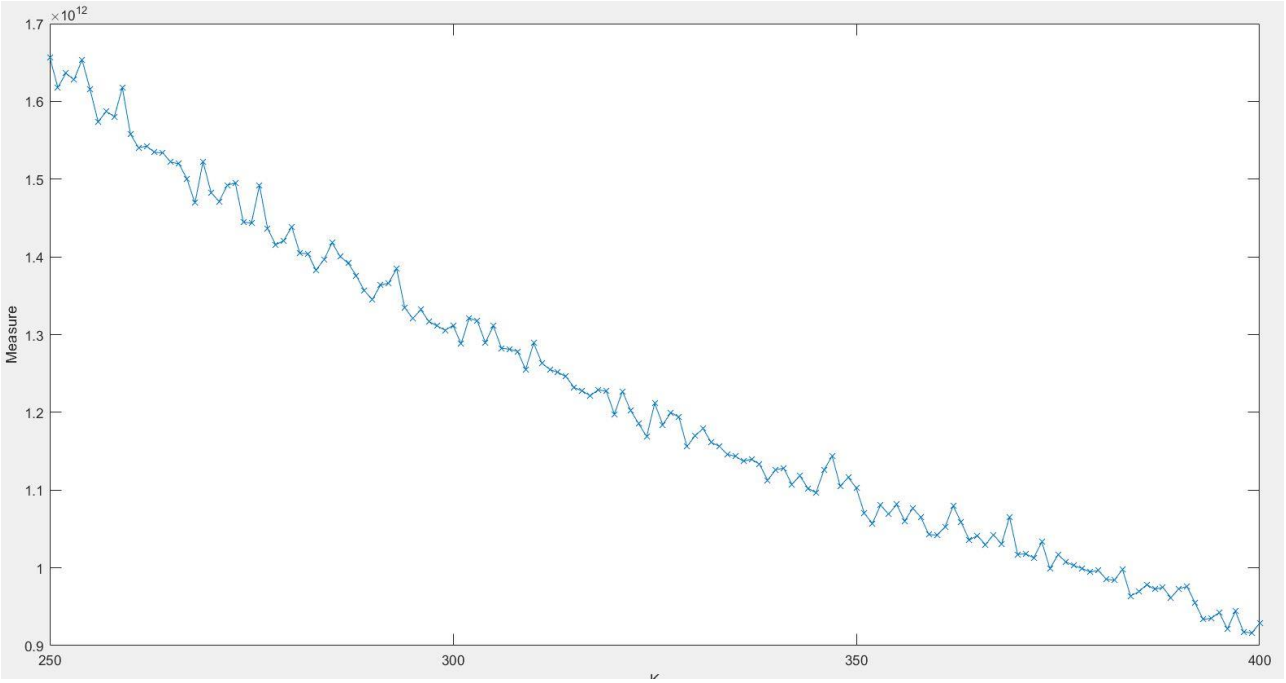

Figure 2. Elbow for basic case with k going from 150 to 400

As we can see results depend on what kinds of values you choose for k. Based on figure 1 we can say that the elbow is where k gets value 18. In figure 2 there is no elbow since it is more or less linear.

For the 3D case we used k values from 2 to 150 and from 30 to 150. We also used two different k value intervals for this case to see the difference for using different k ranges.

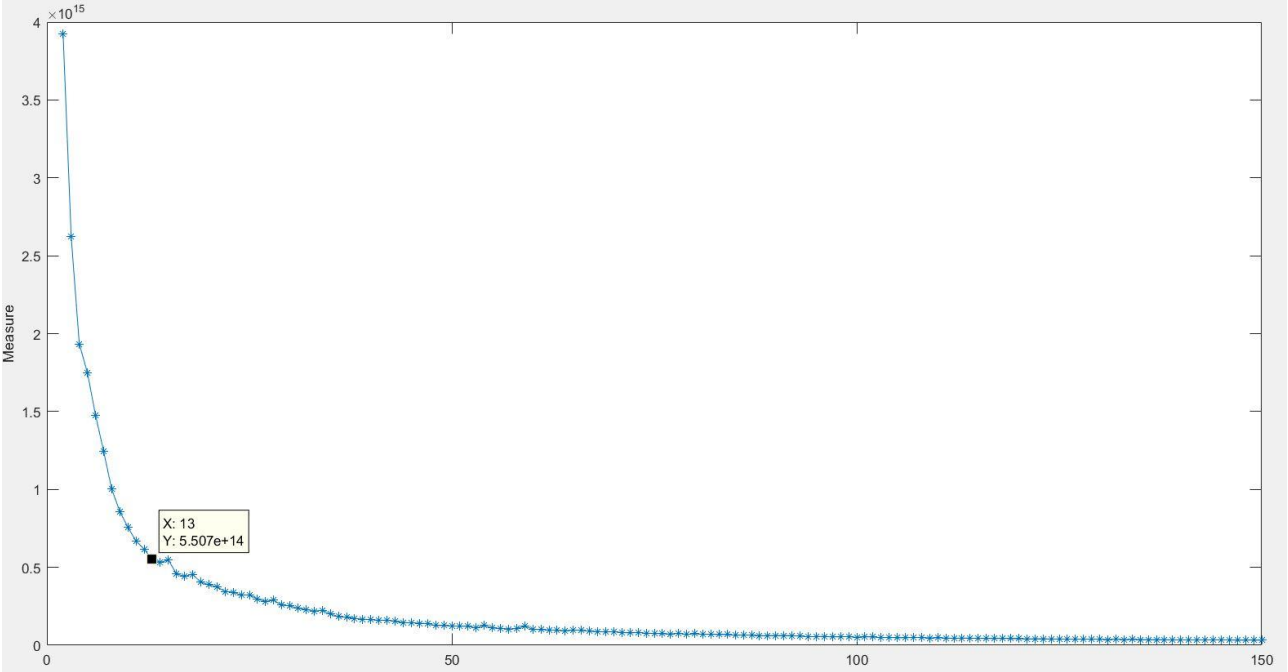

Figure 3. Elbow for 3D case with k going from 2 to 150

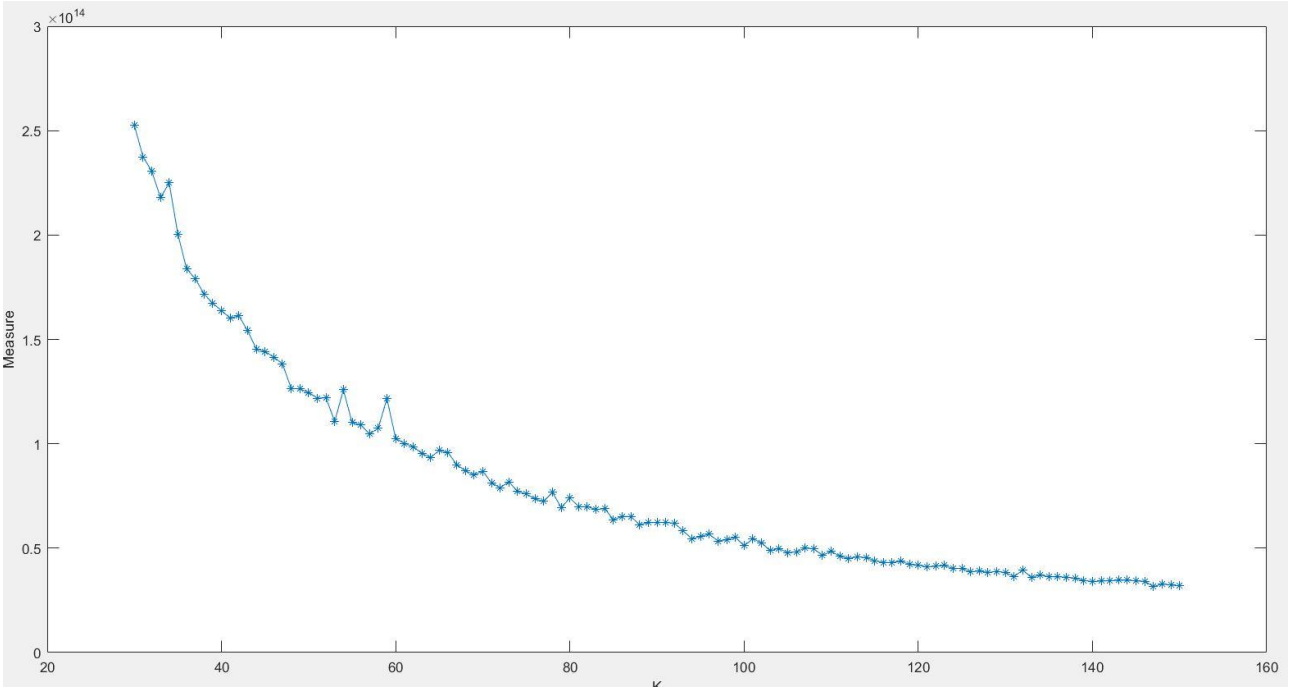

Figure 4. Elbow for 3D case with k going from 30 to 150

Results for this case are similar to basic case. In figure 3 we can see clear elbow around where k gets value 13. In figure it is harder to whether there is elbow or not, but since the slope is so gently sloping there is no elbow.

Task 5

The implementation is exactly the same than in task 2, but did the algorithm using RDDS. We had problems with Spark Context, so we read the file with data frames and implemented the algorithm with RDDS. The result is almost the same but we taught the difference is caused by random algorithm which makes the initial center points.

Task 6

In this task we modified task 5 kmeans function to return the sum of squared distances to the nearest cluster center. Then we looped that function with k values from 2 to 60 and plotted results to figure 5.

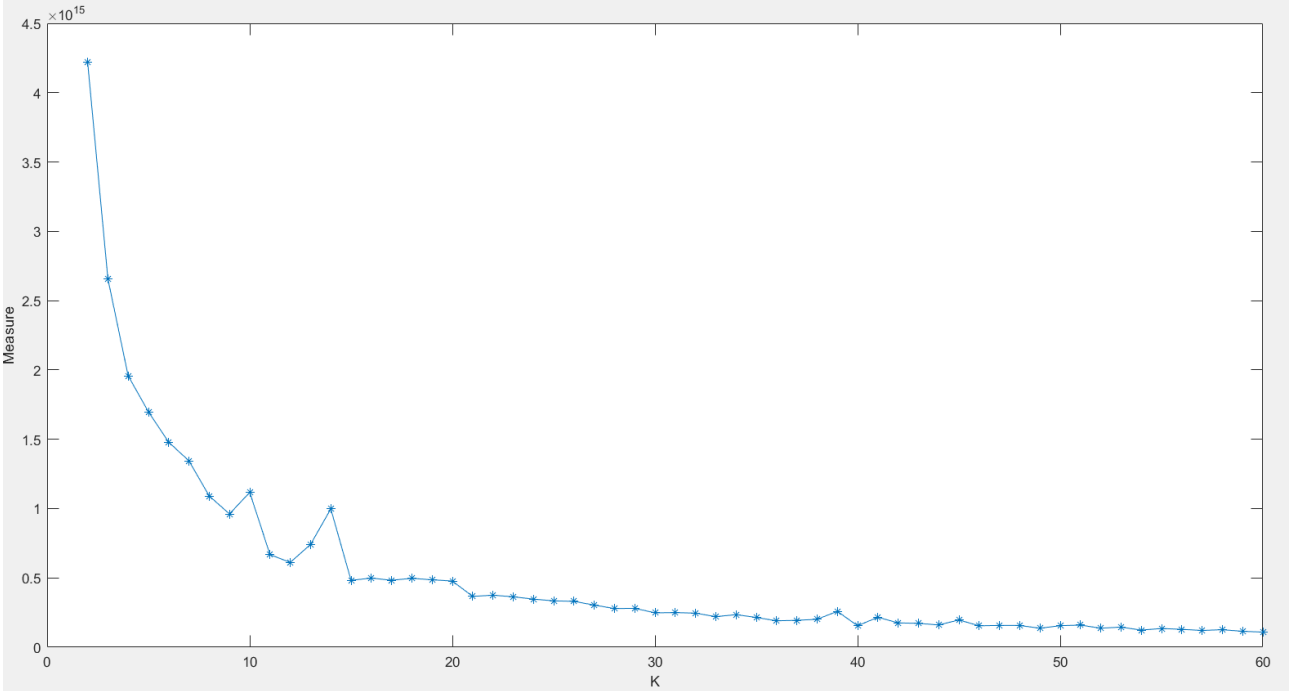

Figure 5. Results of task 6

As seen in figure 5 there are 2 peak points so it is hard to determine the exact spot of elbow but it could be near them.