CS235 Data Mining Techniques
Assignment Report

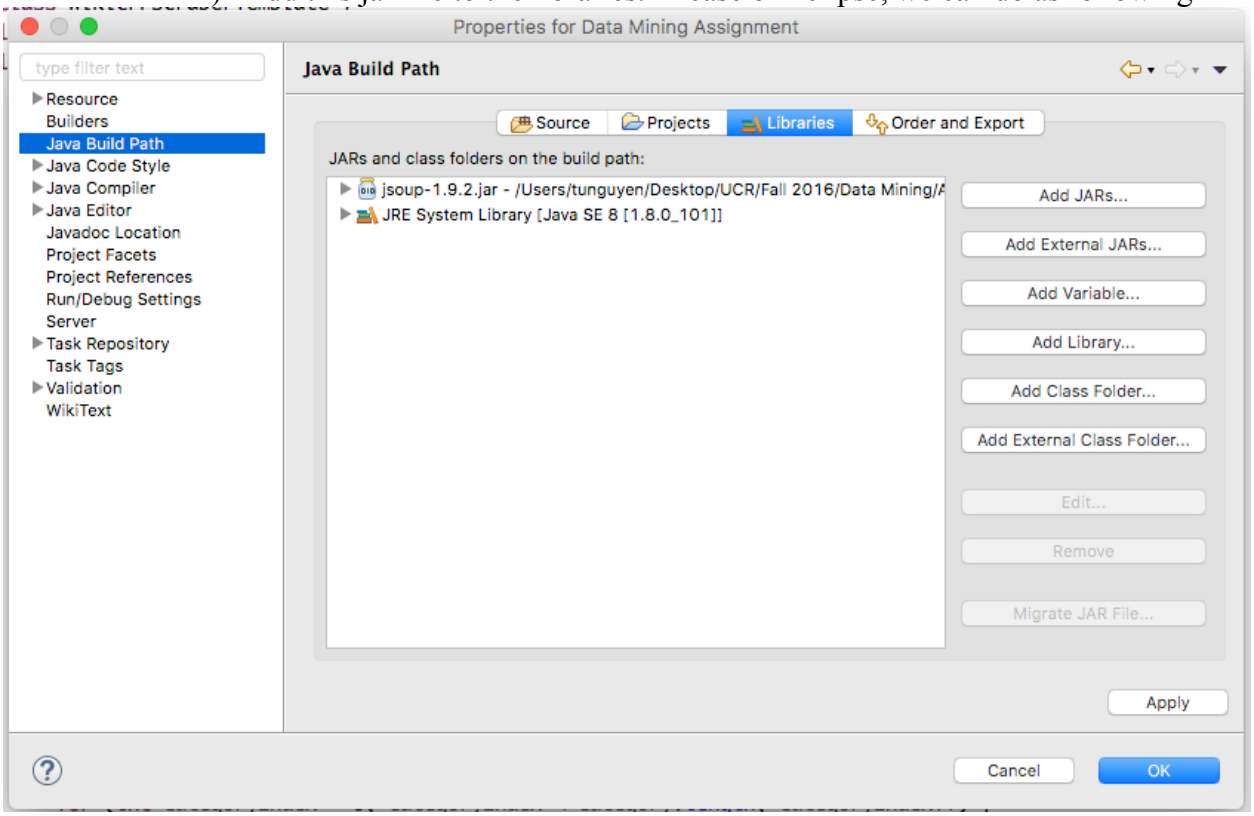
November 12, 2016

Student Name	Student ID
Tu Nguyen	861309226

I. Data Crawling

I used Jsoup library (https://jsoup.org/) to parse HTML pages. Here are the steps to use Jsoup

1) Download the jar file (jsoup-1.9.2.jar) and add it to the location of the project file
2) Add this jar file to the libraries. In case of Eclipse, we can do as following



3) Import the library so that we can use its APIs

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
```

I crawled all pages of all conferences and saved them to one file “wikicfp_crawl.txt”.

```
String category[] = {"data mining", "databases", "machine learning",
"artificial intelligence"};
```

```
                //String category = "databases";
                //String category = "machine learning";
                //String category = "artificial intelligence";

            int numOfPages = 20;

            //create the output file
            File file = new File("wikicfp_crawl.txt");
            file.createNewFile();
            FileWriter writer = new FileWriter(file);
```

The output file has 5 columns separated by tab as following

```
            /* Write columns' headers to the target file */
            writer.write("Conference's Acronym with year\t");
            writer.write("Conference's Acronym\t");
            writer.write("Year\t");
            writer.write("Conference's Name\t");
            writer.write("Location\n");
```

Here is the main code to parse the HTML pages, extract the necessary information, and save them to the output file

```
            //now start crawling the all 'numOfPages' pages
            for (int categoryIndex = 0; categoryIndex <
category.length; categoryIndex++) {
                for(int i = 1;i<=numOfPages;i++) {
                    //Create the initial request to read the first
page
                        //and get the number of total results
                    String linkToScrape =
"http://www.wikicfp.com/cfp/call?conference="+

URLEncoder.encode(category[categoryIndex], "UTF-8") +"&page=" + i;
                    String content = getPageFromUrl(linkToScrape);

                    Document doc = Jsoup.parse(content);

                    /* Navigate the content section */
                    Element cotentSec =
doc.select("div.contsec").first();

                    /* Select the first table, which contains
necessary info of conferences */
                    Element table =
cotentSec.select("table").get(0);
```

```
                        Elements rows = table.select("tr");

                        for (int j = 0; j < rows.size(); j++) {
                            Element row = rows.get(j);
                            Elements cols = row.select("td");

                            /* Search for column containing the key
word "Event" */
                            if (cols.get(0).text().contains("Event")) {
                                /* Target content identified */

                              /* The first table contains all info of
conferences */
                              Element tTable =
cols.get(0).select("table").first();
                              Elements tRows = tTable.select("tr");
                              int numRows = tRows.size();

                              /* Skip the first row, which contains
columns' titles */
                              int k = 1;
                              while (k < numRows) {
                                    /* Each conference record has two
consecutive rows:
                                     * First row: conference acronym +
conference name
                                     * Second row: when + where +
deadline */
                                    Element firstRow, secondRow;

                                    firstRow = tRows.get(k++);
                                    if (k < numRows) {
                                          Elements firstRowCols;
                                          firstRowCols =
firstRow.select("td");

                                          if (firstRowCols.size() == 1) {
                                                /* Skip this row since it
does not contain necessary info */

      System.out.println(firstRowCols.get(0).text());
                                          }
                                          else {
                                                Elements secondRowCols;
```

```
                                                /* We want to search for
pattern of "Conference's acronym + space or ' + conference's year" */
                                                Pattern p =
Pattern.compile("(.+)[\\s\\'](\\d+)");
                                                Matcher m;

                                                /* Get the second row of
a record */
                                                secondRow =
tRows.get(k++);
                                                secondRowCols =
secondRow.select("td");

                                                /* Write conference's
acronym with year to the target file */
      writer.write(firstRowCols.get(0).text() + "\t");

                                                /* Get conference's
acronym and year */
                                                m =
p.matcher(firstRowCols.get(0).text());
                                                //m =
p.matcher("ICNP2016");
                                                if (m.find()) {
                                                      /* Write the
conference's acronym to the target file */
      writer.write(m.group(1) + "\t");
      //System.out.println(m.group(1));

                                                      /* Write the
conference's year to the target file */
      writer.write(m.group(2) + "\t");
      //System.out.println(m.group(2));
                                                }
                                                else {
                                                      /* We need a new
pattern for this special case */
                                                      Pattern p1 =
```

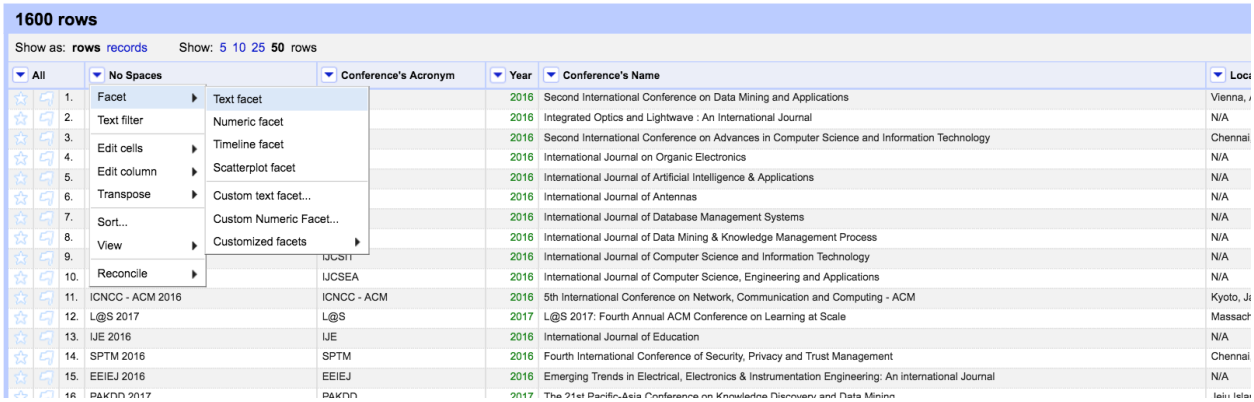
```
Pattern.compile("(\\D+)(\\d+)");
                                                    Matcher m1 =
p1.matcher(firstRowCols.get(0).text());

                                                    if (m1.find()) {
                                                          /* Write the
conference's acronym to the target file */

      writer.write(m1.group(1) + "\t");

      //System.out.println(m1.group(1));

                                                          /* Write the
conference's year to the target file */

      writer.write(m1.group(2) + "\t");

      //System.out.println(m1.group(2));
                                                    }
                                                    else {
                                                          /* Write
whatever we have for acronym to the target file */

      writer.write(firstRowCols.get(0).text() + "\t");

                                                          /* Year will
be NA */

      writer.write("NA" + "\t");

      System.out.println("Year is NA");
                                                    }
                                              }

                                              /* Write the conference's
name to the target file */

      writer.write(firstRowCols.get(1).text() + "\t");

                                              /* Write the conference's
location to the target file */

      writer.write(secondRowCols.get(1).text() + "\n");
                                        }
```

```
                                    }
                                    else {
                                          /* Invalid record */
                                          System.out.println("Error on
page #" + i + ", k = " + k);
                                          break;
                                    }
                               }

                               /* Already got all necessary info */
                                  break;
                              }
                         }

                         //System.out.println(content);

                         //IMPORTANT! Do not change the following:
                         Thread.sleep(DELAY*1000); //rate-limit the
queries
                    }
               }

        writer.close();
           } catch (IOException e) {
                e.printStackTrace();
           } catch (InterruptedException e) {
                e.printStackTrace();
           }

     }
```

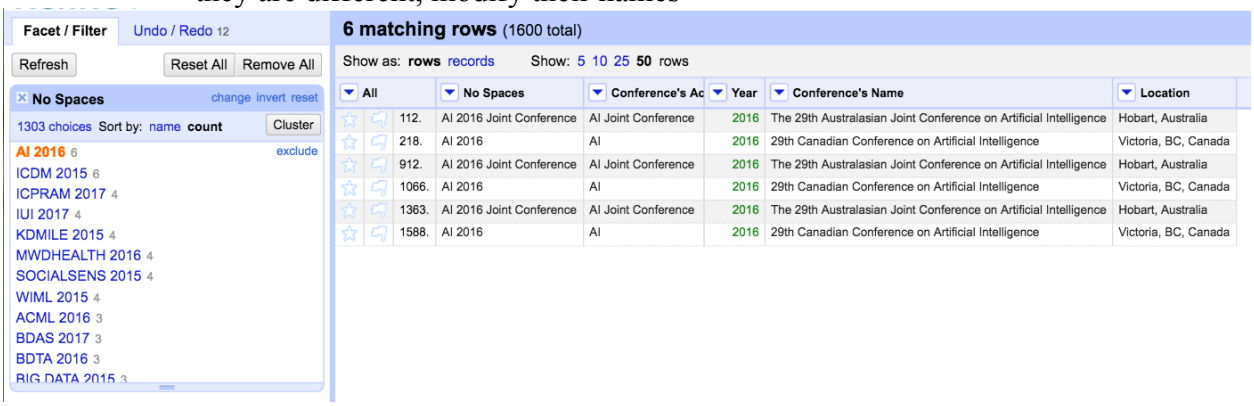
II. Data Cleaning

I used OpenRefine to clean the data. Here are the steps:

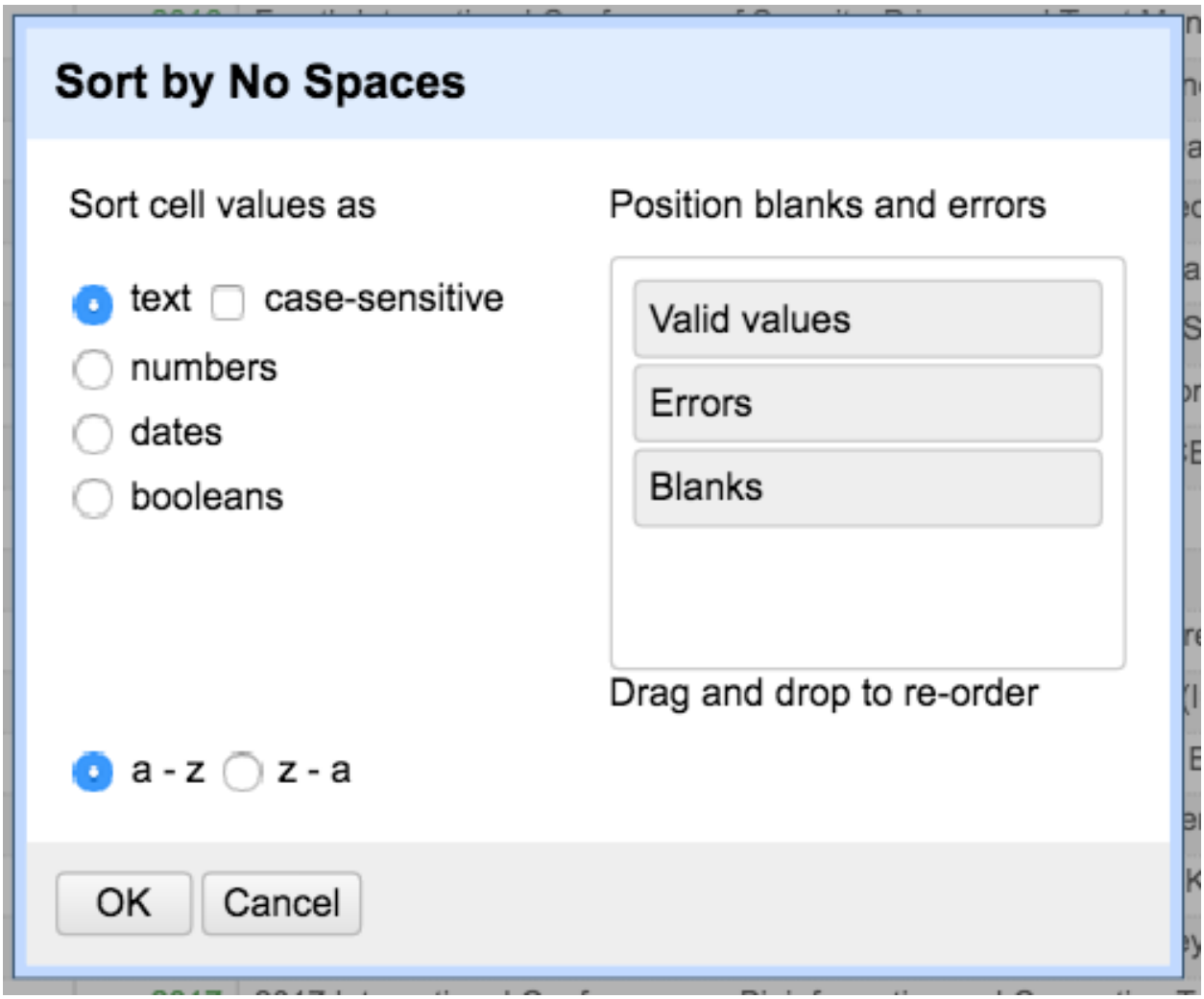
1. Capitalize and remove spaces at the beginning and end of the column "Conference's acronym with year"
2. Do "Text facet" on the column "Conference's acronym with year" ("No spaces")



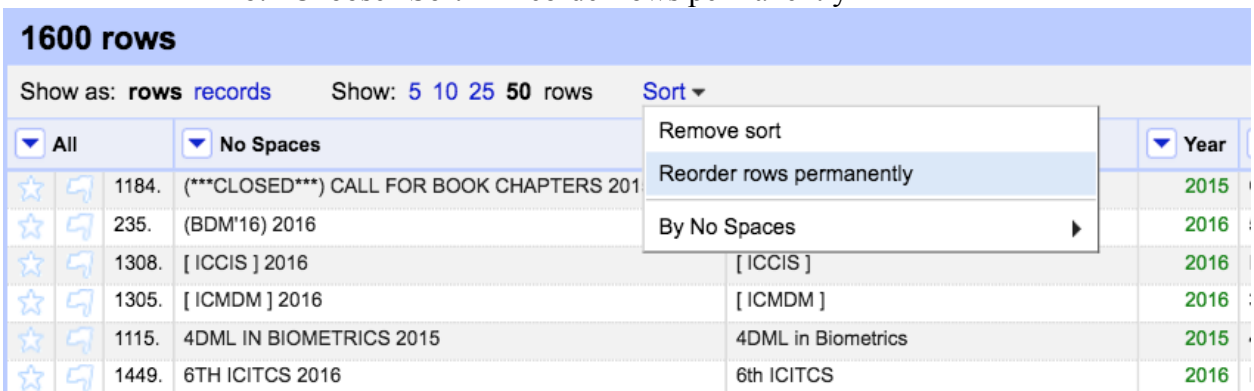
3. Choose “Sort by: count”
4. Manually browse each groups to see if they really belong to the same group. If they are different, modify their names



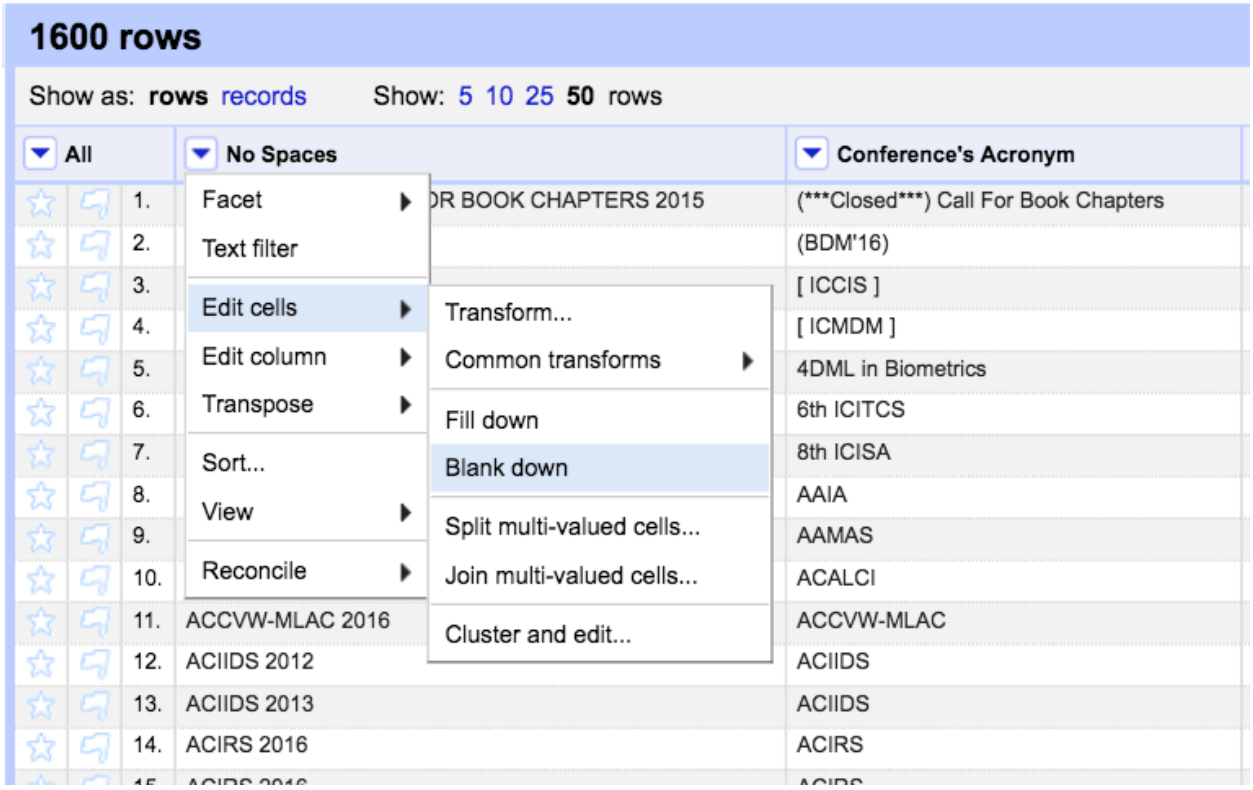
5. Remove duplicate rows by doing the following steps
    a. Sort the column “Conference’s acronym with year” (“No spaces”)



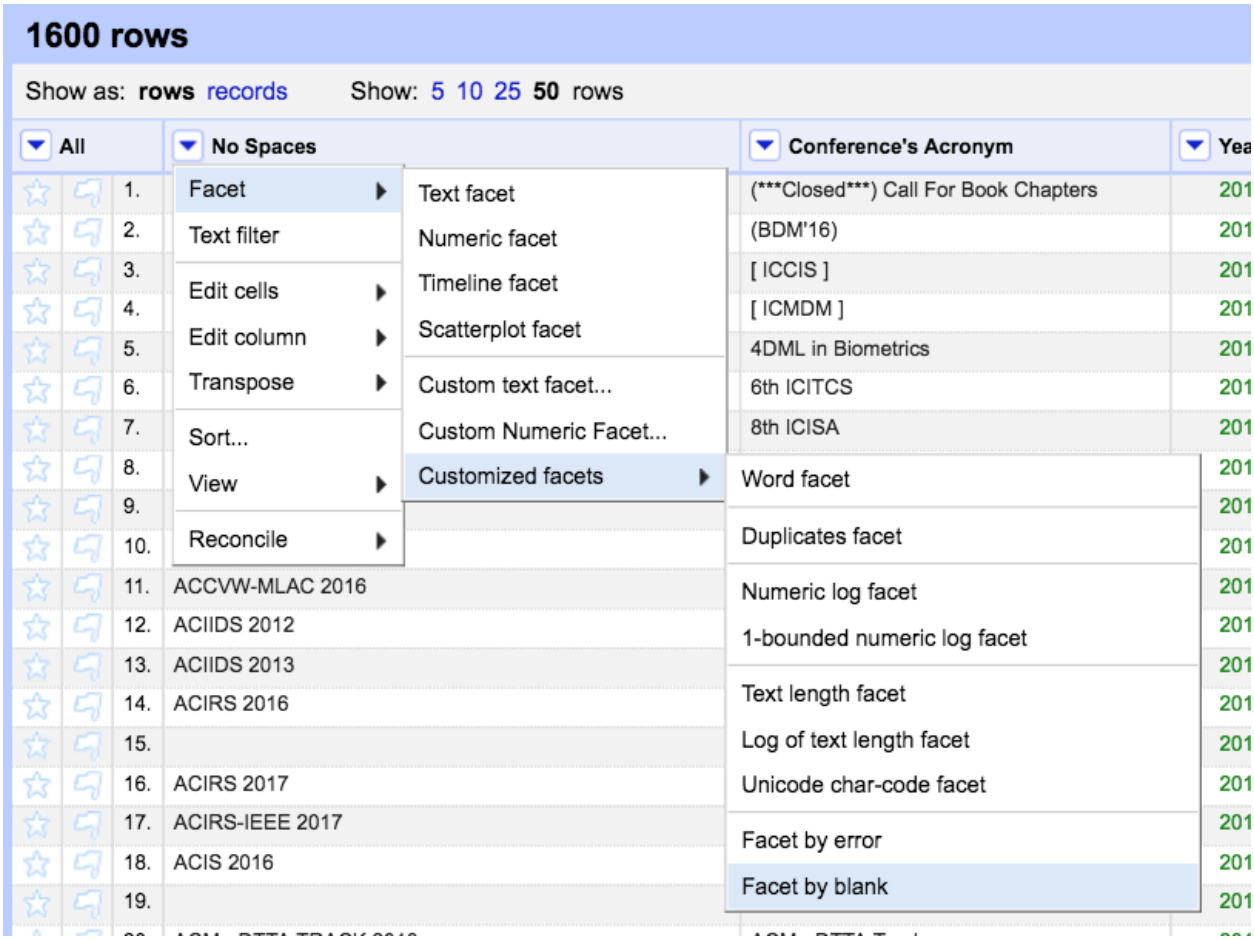
b. Choose “Sort -> Reorder rows permanently”



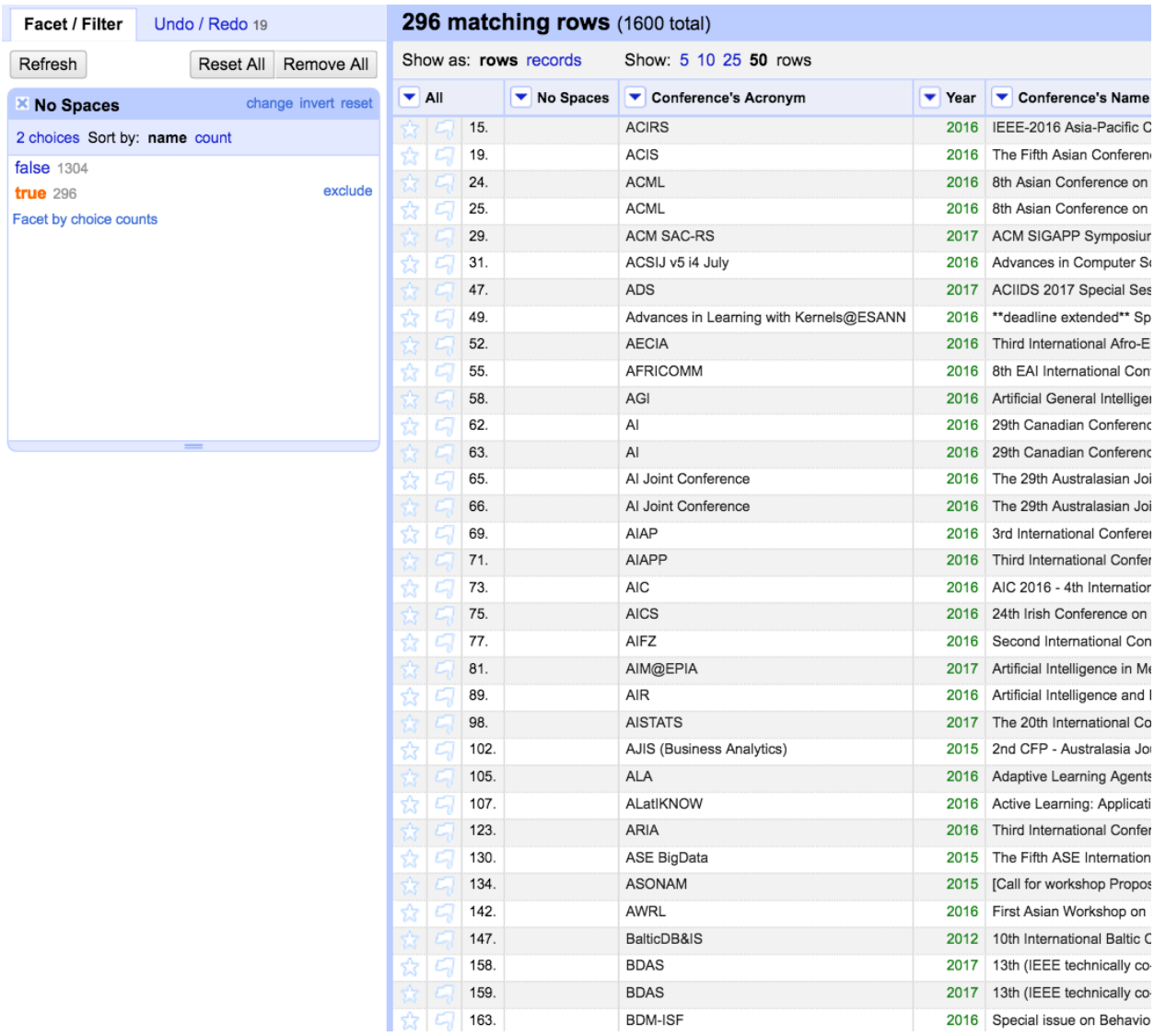
c. Edit cells -> Blank down



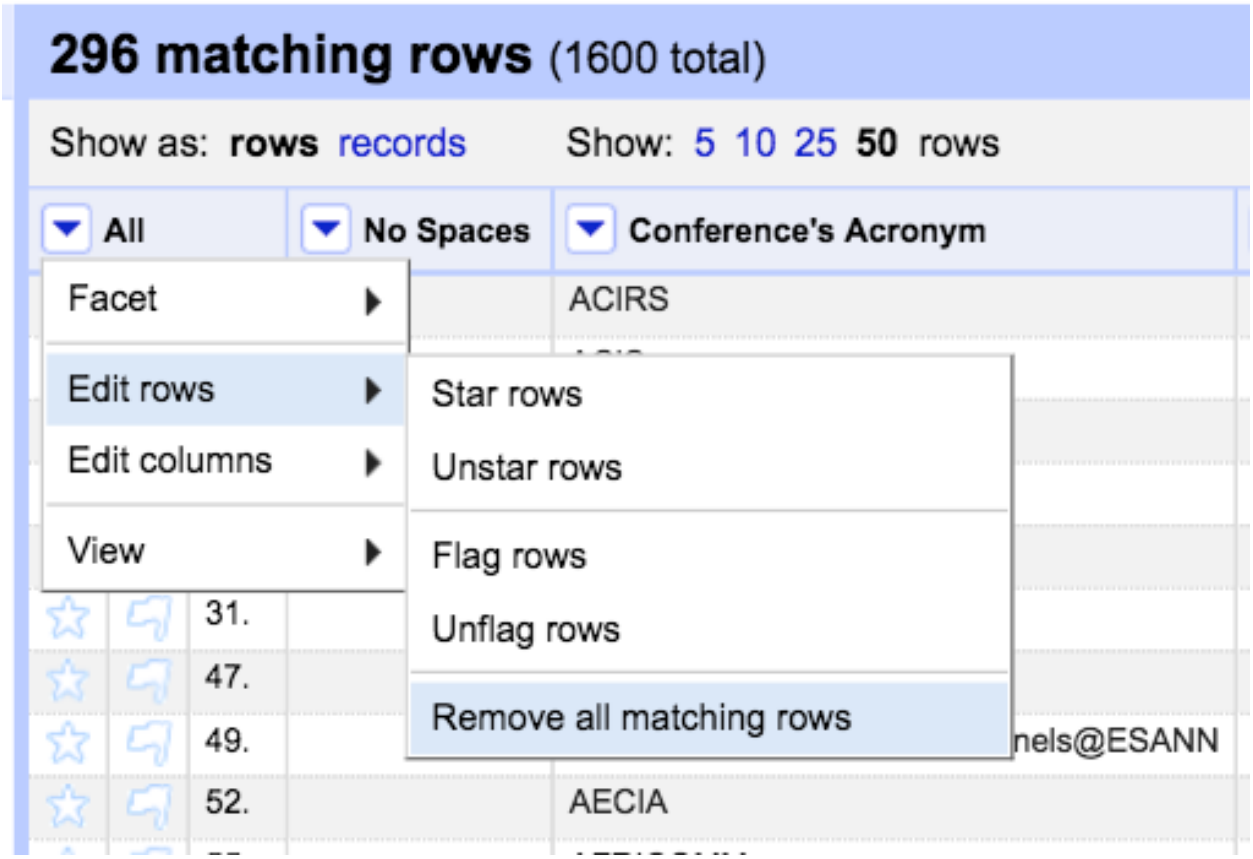
d. Facet -> Customized facet -> Facet by blank



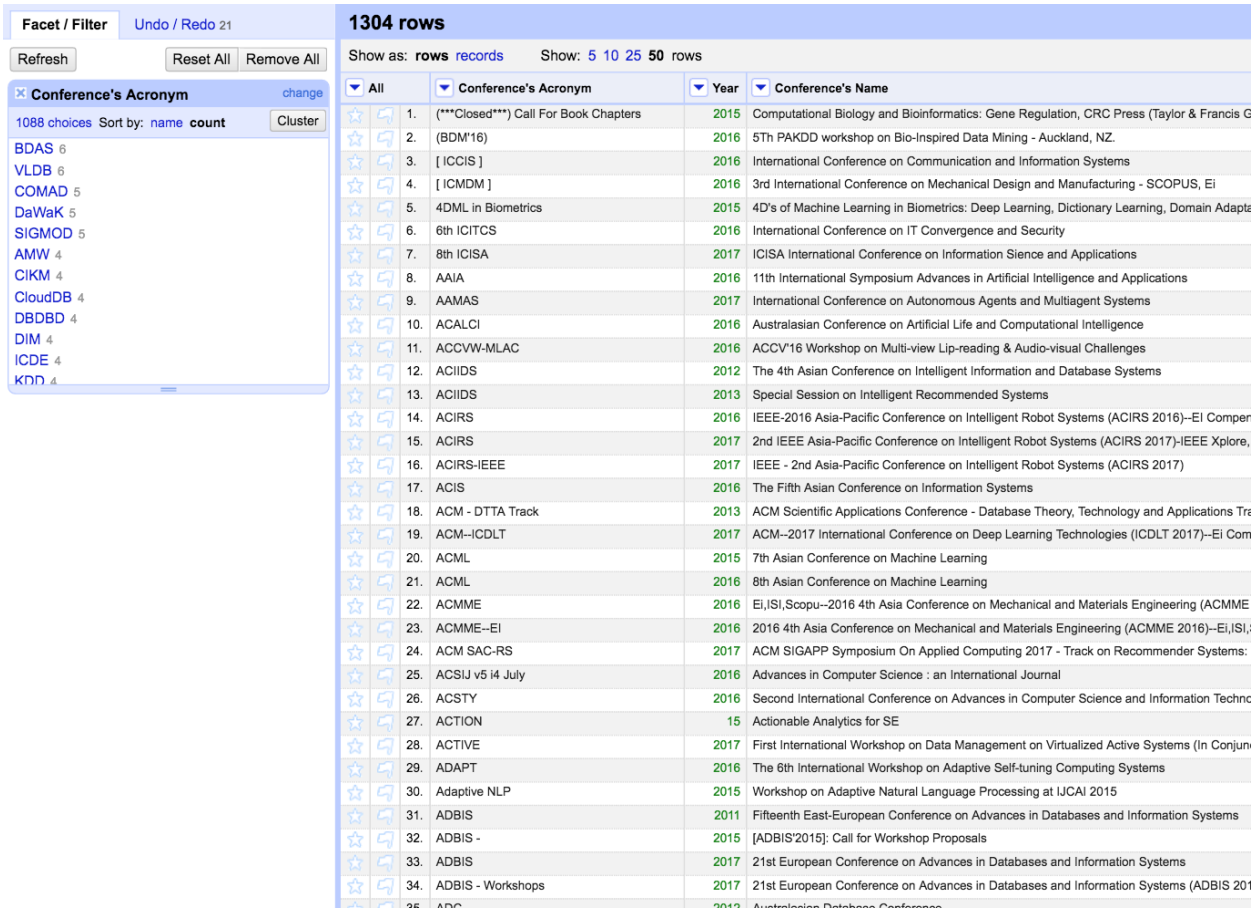
e. Click on “true”



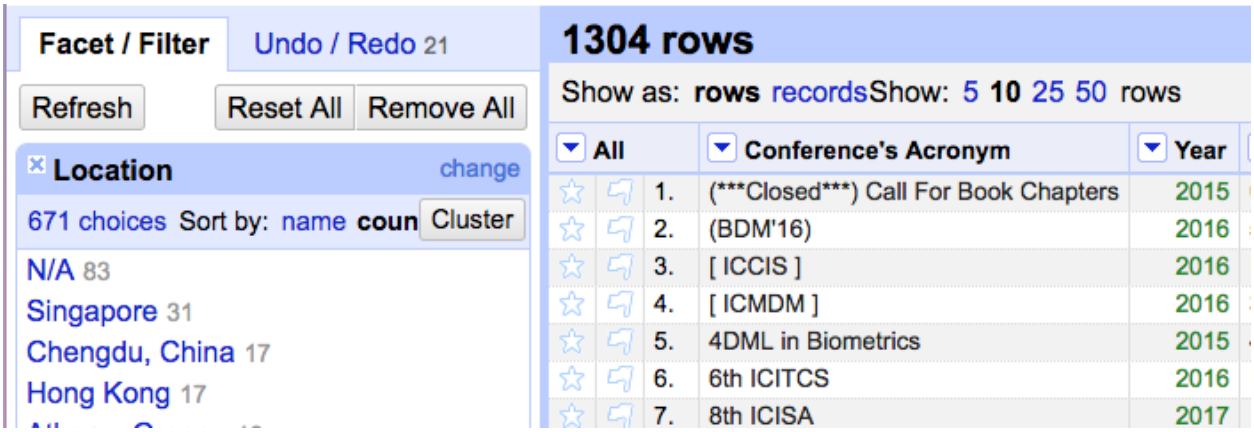
f. All -> Edit rows -> Remove all matching rows



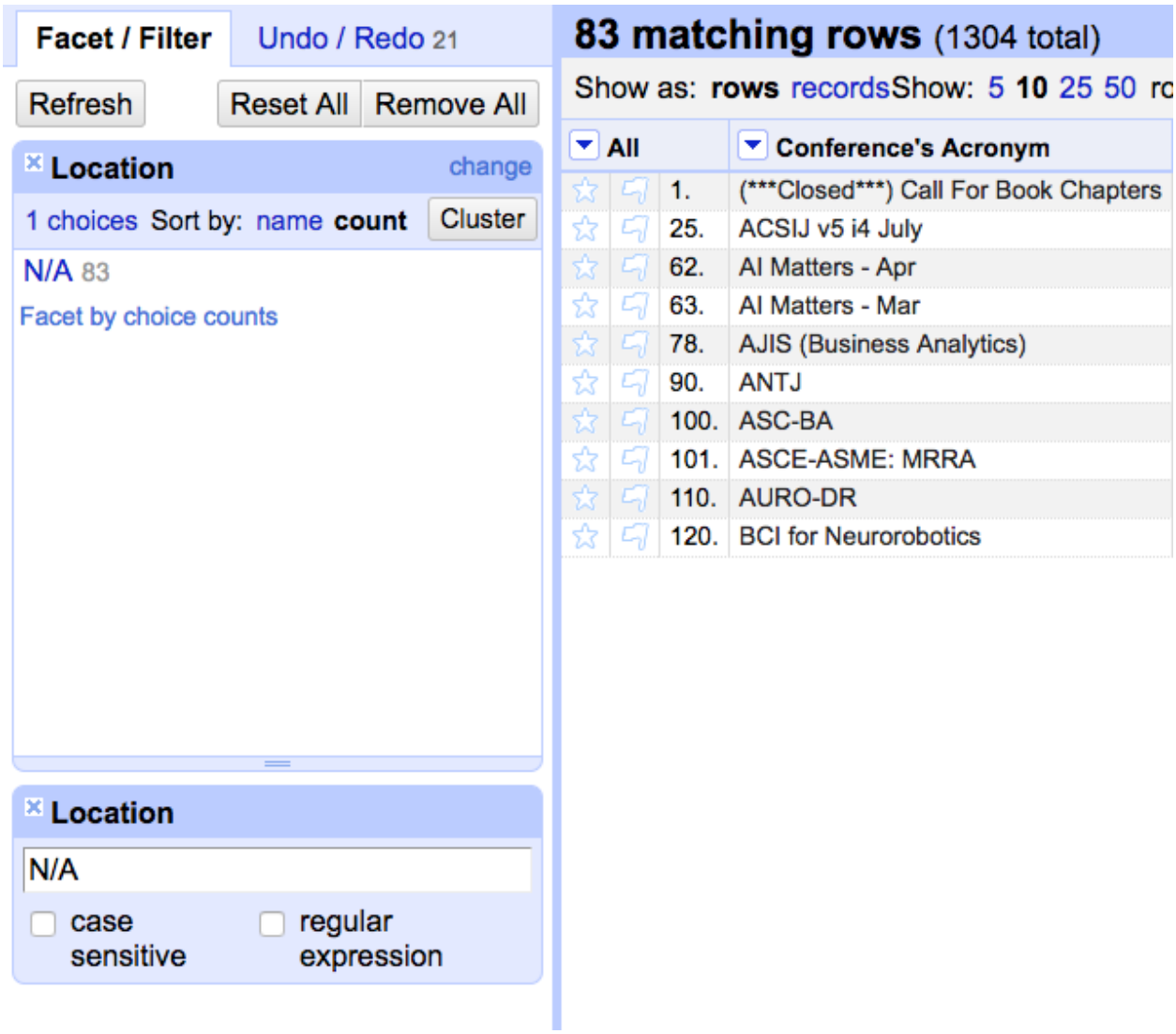
6. Remove the column “Conference’s acronym with year” (“No spaces”)
7. Apply “Text facet” on the column “Conference’s Acronym”
8. Choose “Sort by: count”
9. Remove special characters such as “-IEEE”, “IEE “, “-ACM”, “ACM”, “(”, “)”, “6th”, “8th”, “-”, … at the beginning or end of the field since these characters will make the “Conference’s Acronym” inconsistent.



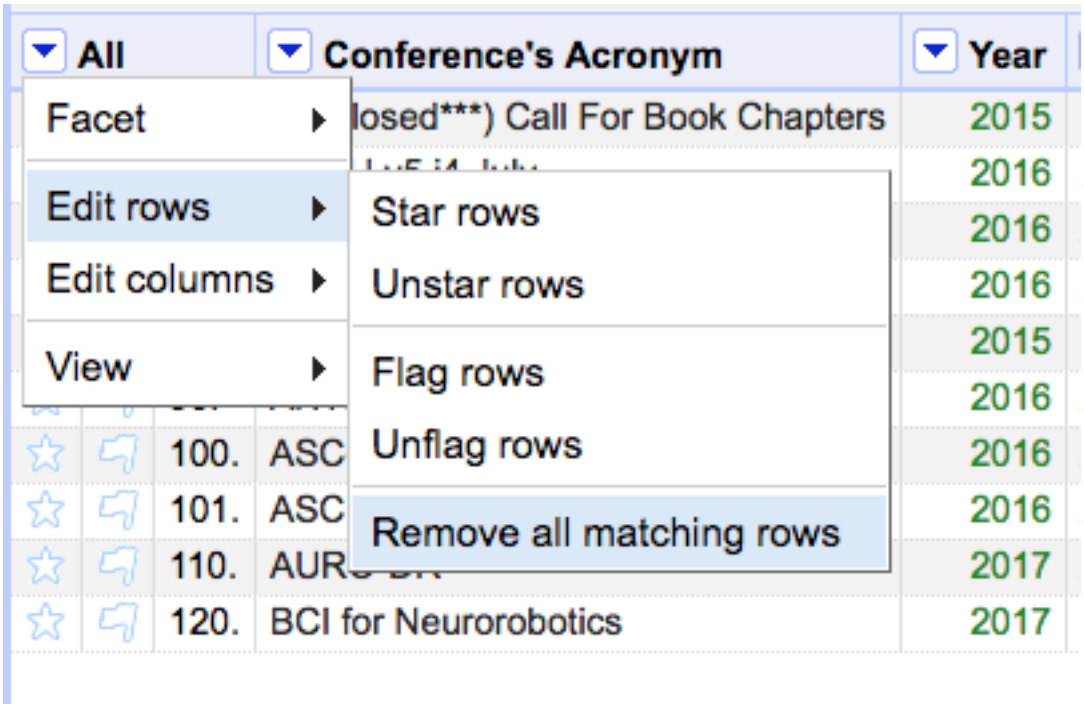
10. Do “Text Facet” on the column “Location”. We see that there are 83 N/A rows among 1304 rows. I think we can remove these rows since the counts for only 6.4%



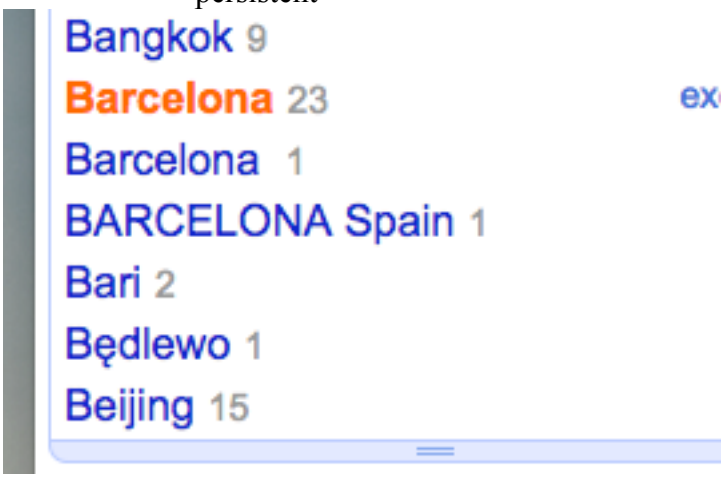
a) Do “Text filter” on the column “Location”, type “N/A” for the filter



b) All -> Edit rows -> Remove all matching rows



11. The column “Location” contains the information about city and country. However, we just need the city information for the statistics. Thus, we need to extract the city info from this column as following: Split the column “Location” by “,”, “(”, and “-”; Keep the first column and rename the column as “City”
12. Do “Text facet” on the column “City” and manually make the city names persistent



III. Hadoop

The input file, which contains list of conferences, has the following format

"Conference Acronym" \t "Year" \t "Conference Name" \t "Location"

1. Compute and plot the number of conferences per city
   Each line in the input file contains information of one conference. Thus, the number of occurrences of a city in the file is the number of conferences of that city.

To do this computation, I just slightly modify the WordCount example as following: Use "city" name as key and number of its occurrences as value since. The output file has two columns: “City” and “Number of conferences”. The map function of the Mapper looks like

```
        @Override
           public void map(Object key, Text value, Context context
                       ) throws IOException, InterruptedException {
             String line = (caseSensitive) ?
                value.toString() : value.toString().toLowerCase();
             for (String pattern : patternsToSkip) {
              line = line.replaceAll(pattern, "");
             }
             /* Format of the input data is
              * "Conference Acronym" \t "Year" \t "Conference Name" \t "Location" */
             /* Split by tab */
             StringTokenizer itr = new StringTokenizer(line,"\t");
             /* Skip the first three columns */
             for (int i = 0; i < 3; i++) {
                  if (itr.hasMoreTokens()) {
                           itr.nextToken();
                  } else {
                           System.err.println("Bad line: " + line + "\n");
                           break;
                  }
             }
             /* We need to process the "City" column */
             if (itr.hasMoreTokens()) {
               city.set(itr.nextToken());
               /* Use "city" column as key and number of occurrences as value */
        context.write(city, one);
               Counter counter = context.getCounter(CountersEnum.class.getName(),
                  CountersEnum.INPUT_WORDS.toString());
               counter.increment(1);
             }
           }
 }
```

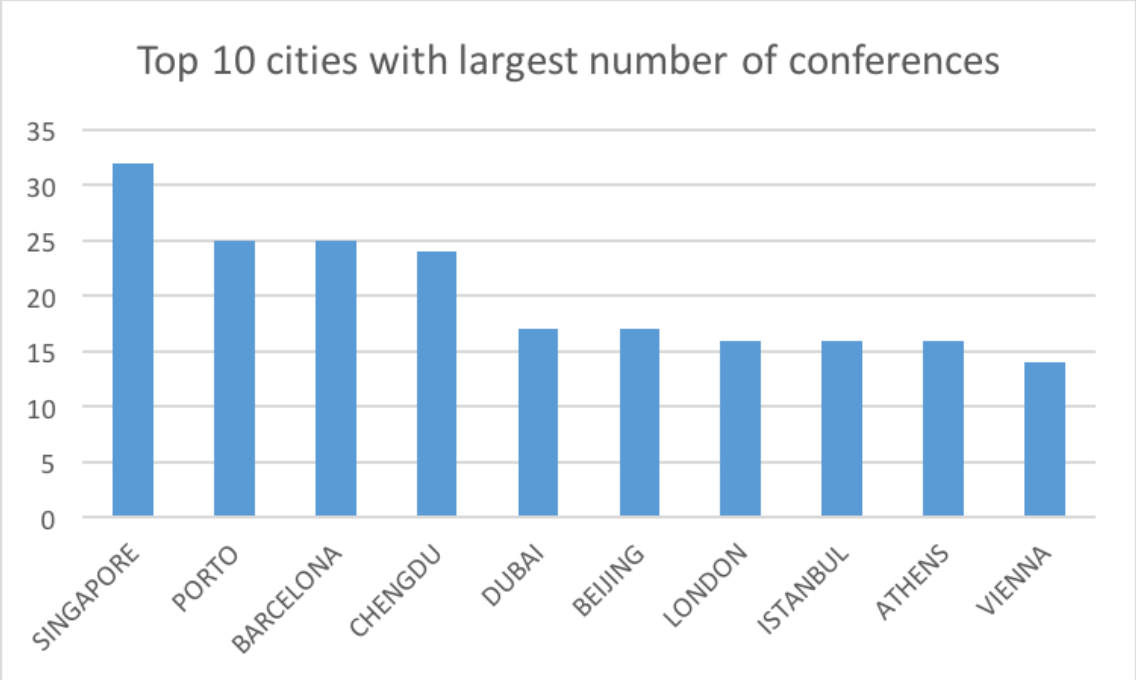
The reduce function of the Reducer looks like

```
        public static class IntSumReducer
             extends Reducer<Text,IntWritable,Text,IntWritable> {
           private IntWritable result = new IntWritable();

           public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
                           ) throws IOException, InterruptedException {
             int sum = 0;
```

```
            for (IntWritable val : values) {
                /* Sum up number of conferences (value) */
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
```

The top 10 locations are

City	Number of conferences
SINGAPORE	32
PORTO	25
BARCELONA	25
CHENGDU	24
DUBAI	17
BEIJING	17
LONDON	16
ISTANBUL	16
ATHENS	16
VIENNA	14



2. Output the list of conferences per city

I used the “city” name (Text) as key and “conference acronym” (Text) as value. The output file has two columns: “City” and “Conference list”, which contains list of conferences’ acronyms separated by “,”.

The map function of the Mapper looks like

```
public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
    String line = (caseSensitive) ?
       value.toString() : value.toString().toLowerCase();
    for (String pattern : patternsToSkip) {
      line = line.replaceAll(pattern, "");
    }
    /* Format of the input data is
    * "Conference Acronym" \t "Year" \t "Conference Name" \t "Location" */
    /* Split by tab */
    StringTokenizer itr = new StringTokenizer(line,"\t");
    /* We need to get the conference's acronym in the first column*/
    if (itr.hasMoreTokens()) {
        confAcronym.set(itr.nextToken());
    } else {
        System.err.println("Bad line: " + line + "\n");
    }
    /* Skip the next two columns */
    for (int i = 0; i < 2; i++) {
        if (itr.hasMoreTokens()) {
            itr.nextToken();
        } else {
            System.err.println("Bad line: " + line + "\n");
            break;
        }
    }
    /* We need to process the "City" in the fourth column */
    if (itr.hasMoreTokens()) {
      city.set(itr.nextToken());
      /* Use "city" as key, and "conference acronym" as value */
      context.write(city, confAcronym);
      Counter counter = context.getCounter(CountersEnum.class.getName(),
         CountersEnum.INPUT_WORDS.toString());
      counter.increment(1);
    } else {
        System.err.println("Bad line: " + line + "\n");
    }
   }
}
```

The reduce function of the Reducer looks like

```
/* We will use "Text" for the value field of the Reducer */
```

```
        public static class TextSumReducer
             extends Reducer<Text,Text,Text,Text> {
           private Text result = new Text();

           public void reduce(Text key, Iterable<Text> values,
                              Context context
                              ) throws IOException, InterruptedException {
            String sum = "";
            for (Text val : values) {
                String conf = val.toString();

                /* Remove spaces at the end and beginning of the string */
                conf = conf.trim();
                /* Add the conference to the list if it is not yet present in the list */
                if (sum.indexOf(conf) == -1) {
                        sum += conf + ", ";
                }
            }
            /* Remove the ", " at the end */
            sum = sum.trim();
            while ((sum.length() > 0) && (sum.charAt(sum.length()-1) == ',')) {
                sum = sum.substring(0,sum.length()-1);
            }
            result.set(new Text(sum));
            context.write(key, result);
           }
}
```

3. For each conference regardless of the year (e.g., KDD), output the list of cities
   I used "conference acronym" (Text) as key and "city" (Text) as value. The output file has two columns: “Conference acronym” and “City list”, which contains list of cities separated by “,”.
   The map function is as follows:

```
        public void map(Object key, Text value, Context context
                        ) throws IOException, InterruptedException {
            String line = (caseSensitive) ?
               value.toString() : value.toString().toLowerCase();
            for (String pattern : patternsToSkip) {
              line = line.replaceAll(pattern, "");
            }
            /* Format of the input data is
            * "Conference Acronym" \t ”Year" \t "Conference Name" \t "Location" */
            /* Split by tab */
            StringTokenizer itr = new StringTokenizer(line,"\t");
            /* We need to get the conference's acronym in the first column */
            if (itr.hasMoreTokens()) {
```

```
                confAcronym.set(itr.nextToken());
            } else {
                System.err.println("Bad line: " + line + "\n");
            }
            /* Skip the next two columns */
            for (int i = 0; i < 2; i++) {
                if (itr.hasMoreTokens()) {
                        itr.nextToken();
                } else {
                        System.err.println("Bad line: " + line + "\n");
                        break;
                }
            }
            /* We need to process the "City" in the fourth column */
            if (itr.hasMoreTokens()) {
              city.set(itr.nextToken());
              /* Use "conference acronym" as key and "city" as value */
        context.write(confAcronym, city);
              Counter counter = context.getCounter(CountersEnum.class.getName(),
                CountersEnum.INPUT_WORDS.toString());
              counter.increment(1);
            } else {
                System.err.println("Bad line: " + line + "\n");
            }
          }
}
```

The reduce function is as following:

```
        /* We will use "Text" for the value field of the Reducer */
        public static class TextSumReducer
              extends Reducer<Text,Text,Text,Text> {
           private Text result = new Text();

           public void reduce(Text key, Iterable<Text> values,
                        Context context
                        ) throws IOException, InterruptedException {
            String sum = "";
            for (Text val : values) {
                String conf = val.toString();

                conf = conf.trim();
                /* Add the current city to the list if it is not yet present in the list */
                if (sum.indexOf(conf) == -1) {
                        sum += conf + ", ";
                }
            }
```

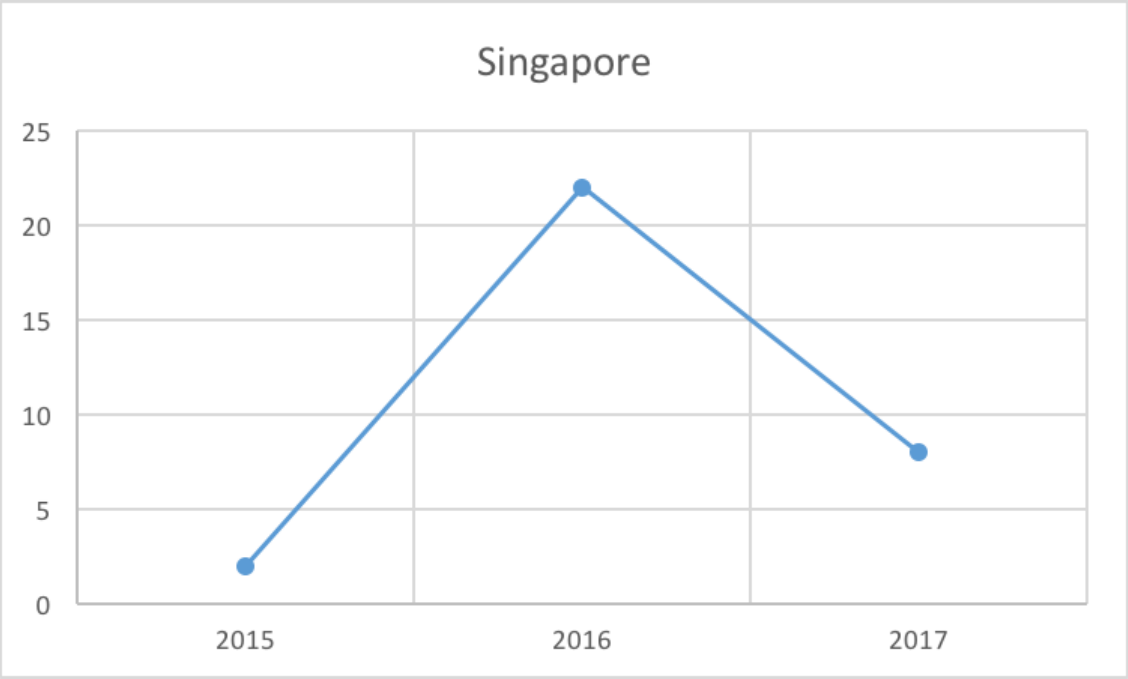
```
            /* Remove the ", " at the end */
            sum = sum.trim();
            while ((sum.length() > 0) && (sum.charAt(sum.length()-1) == ',')) {
                sum = sum.substring(0,sum.length()-1);
            }
            result.set(new Text(sum));
            context.write(key, result);
          }
}
```

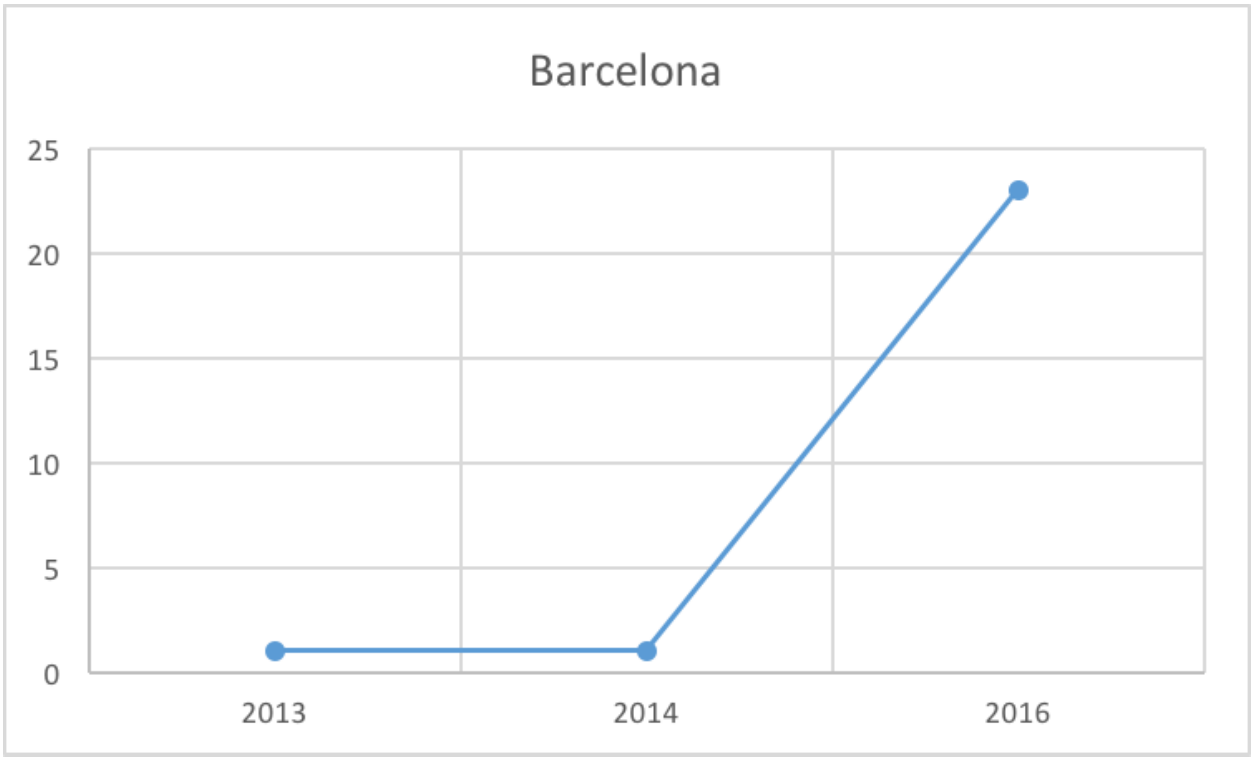
4. For each city compute and plot a time series of #conferences per year
   I use “City + Year” (Text) as key and it occurrences (IntWritable) as value. The output file has three columns: “City”, “Year”, and “Number of conferences”.
   The map function is as follows

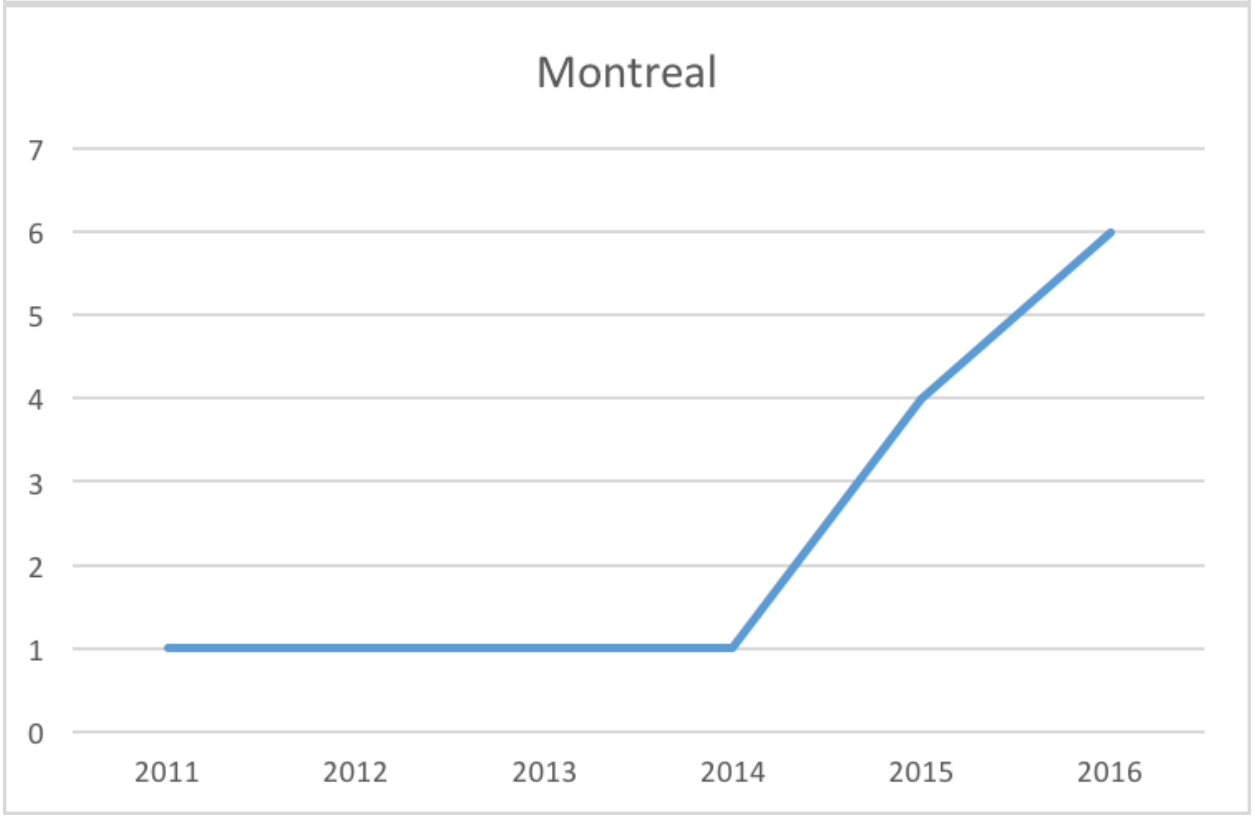
```
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
        String cityStr = "";
        String yearStr = "";
        String line = (caseSensitive) ?
           value.toString() : value.toString().toLowerCase();
        for (String pattern : patternsToSkip) {
          line = line.replaceAll(pattern, "");
        }
        /* Format of the input data is
        * "Conference Acronym" "Year" "Conference Name" "Location" */
        /* Split by tab */
        StringTokenizer itr = new StringTokenizer(line,"\t");
        /* Skip the first column */
        if (itr.hasMoreTokens()) {
                itr.nextToken();
          } else {
                System.err.println("Bad line: " + line + "\n");
        }
        /* Get the year */
        if (itr.hasMoreTokens()) {
                yearStr = itr.nextToken().toString();
          } else {
                System.err.println("Bad line: " + line + "\n");
        }
        /* Skip the third column */
        if (itr.hasMoreTokens()) {
                itr.nextToken();
          } else {
                System.err.println("Bad line: " + line + "\n");
        }
        /* We need to process the "City" column */
```

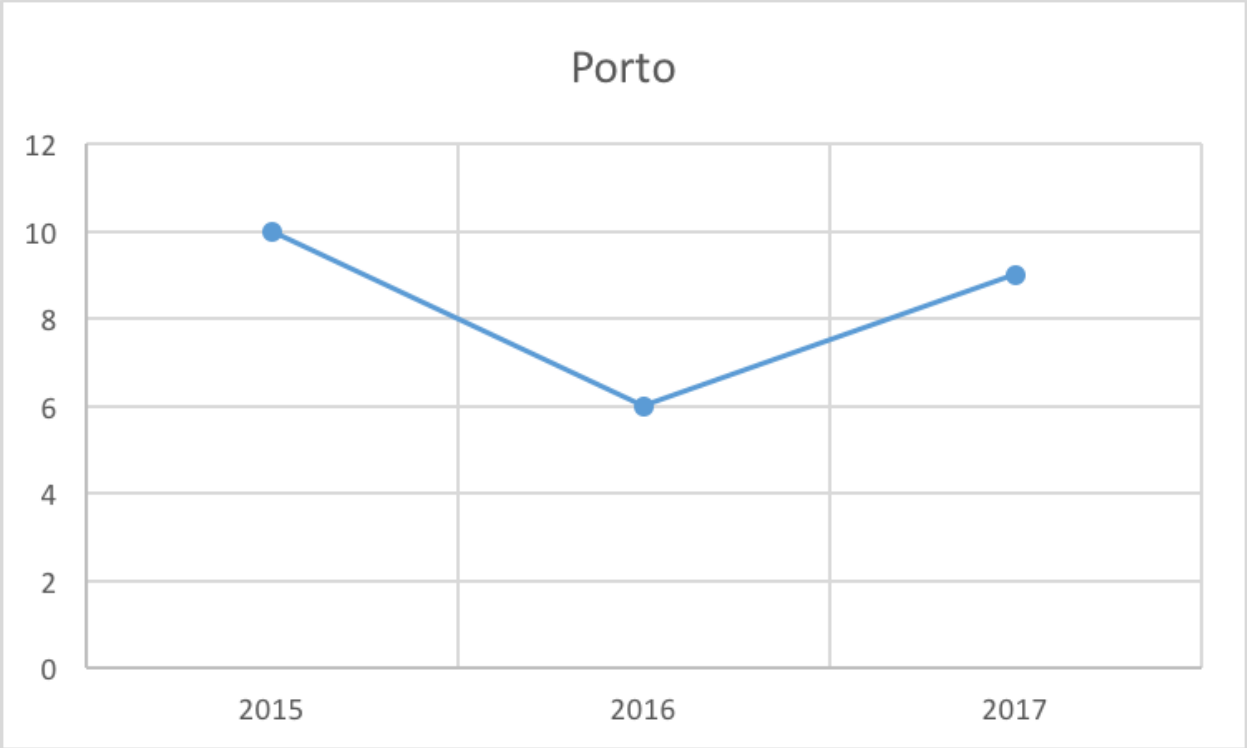
```
            if (itr.hasMoreTokens()) {
              cityStr = itr.nextToken().toString();
                 /* Use "city + year" as key and number of occurrences as value */
              context.write(new Text(cityStr + "\t" + yearStr), one);
              Counter counter = context.getCounter(CountersEnum.class.getName(),
                 CountersEnum.INPUT_WORDS.toString());
              counter.increment(1);
            }
          }
        }
```

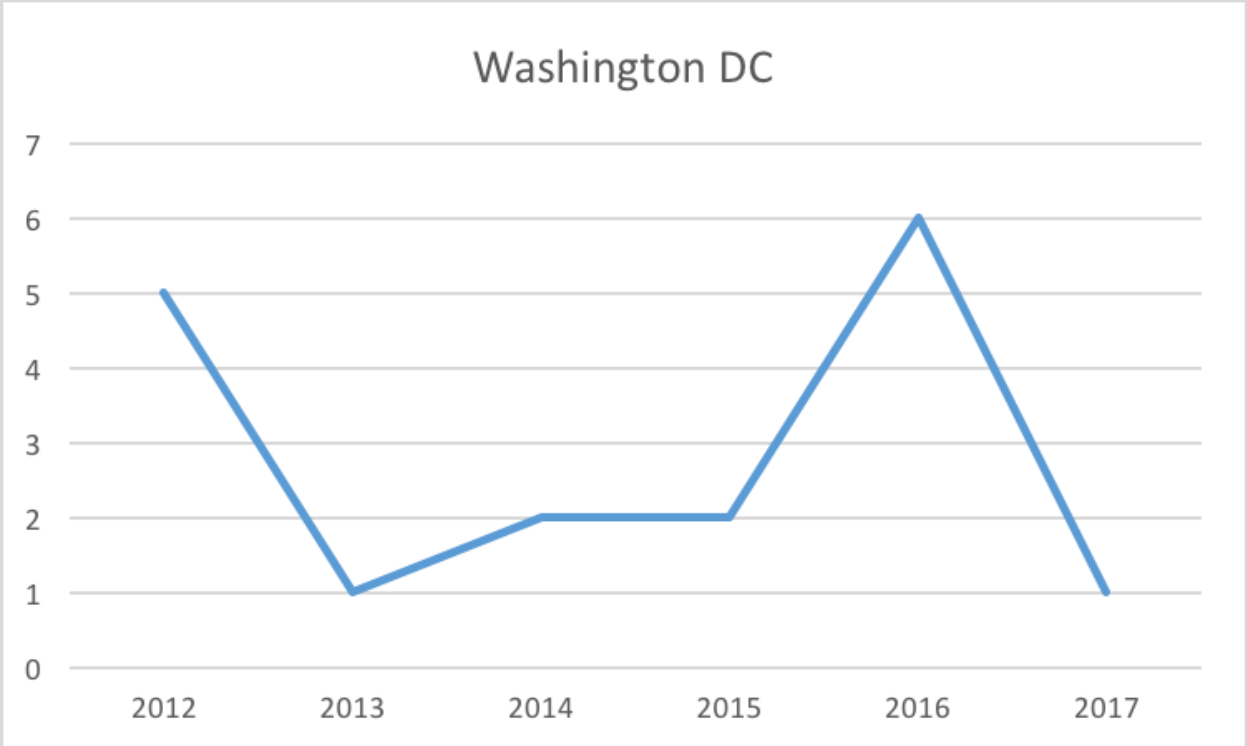
The time series plot of some cities




Barcelona
25
20
15
10
5
0
2013
2014
2016


Montreal
7
6
5
4
3
2
1
0
2011
2012
2013
2014
2015
2016


Porto
12
10
8
6
4
2
0
2015
2016
2017


Washington DC
7
6
5
4
3
2
1
0
2012
2013
2014
2015
2016
2017