package com.twitter.search.earlybird.util;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import com.google.common.base.Preconditions;

/\*\*

\* A histogram of int values with arbitrary buckets.

\* Keeps a count for each bucket, and a sum of values for each bucket.

\* The histogram view is returned as a list of {@link Histogram.Entry}s.

\* <p/>

\* Bucket boundaries are inclusive on the upper boundaries. Given buckets of [0, 10, 100],

\* items will be places in 4 bins, { X <= 0, 0 < X <= 10, 10 < X <= 100, X > 100 }.

\* <p/>

\* This class is not thread safe.

\*

\*/

public class Histogram {

private final double[] buckets;

private final int[] itemsCount;

private final long[] itemsSum;

private int totalCount;

private long totalSum;

public static class Entry {

private final String bucketName;

private final int count;

private final double countPercent;

private final double countCumulative;

private final long sum;

private final double sumPercent;

private final double sumCumulative;

Entry(String bucketName,

int count, double countPercent, double countCumulative,

long sum, double sumPercent, double sumCumulative) {

this.bucketName = bucketName;

this.count = count;

this.countPercent = countPercent;

this.countCumulative = countCumulative;

this.sum = sum;

this.sumPercent = sumPercent;

this.sumCumulative = sumCumulative;

}

public String getBucketName() {

return bucketName;

}

public int getCount() {

return count;

}

public double getCountPercent() {

return countPercent;

}

public double getCountCumulative() {

return countCumulative;

}

public long getSum() {

return sum;

}

public double getSumPercent() {

return sumPercent;

}

public double getSumCumulative() {

return sumCumulative;

}

}

/\*\*

\* No buckets will put all items into a single bin.

\* @param buckets the buckets to use for binnning data.

\* An item will be put in bin i if item <= buckets[i] and > buckets[i-1]

\* The bucket values must be strictly increasing.

\*/

public Histogram(double... buckets) {

Preconditions.checkNotNull(buckets);

this.buckets = new double[buckets.length];

for (int i = 0; i < buckets.length; i++) {

this.buckets[i] = buckets[i];

if (i > 0) {

Preconditions.checkState(this.buckets[i - 1] < this.buckets[i],

"Histogram buckets must me strictly increasing: " + Arrays.toString(buckets));

}

}

this.itemsCount = new int[buckets.length + 1];

this.itemsSum = new long[buckets.length + 1];

this.totalCount = 0;

this.totalSum = 0;

}

/\*\*

\* Add the given item to the appropriate bucket.

\*/

public void addItem(double item) {

int i = 0;

for (; i < this.buckets.length; i++) {

if (item <= buckets[i]) {

break;

}

}

this.itemsCount[i]++;

this.totalCount++;

this.itemsSum[i] += item;

this.totalSum += item;

}

/\*\*

\* returns the current view of all the bins.

\*/

public List<Entry> entries() {

List<Entry> entries = new ArrayList<>(this.itemsCount.length);

double countCumulative = 0;

double sumCumulative = 0;

for (int i = 0; i < this.itemsCount.length; i++) {

String bucketName;

if (i < this.buckets.length) {

bucketName = "<= " + this.buckets[i];

} else if (this.buckets.length > 0) {

bucketName = " > " + this.buckets[this.buckets.length - 1];

} else {

bucketName = " \* ";

}

int count = this.itemsCount[i];

double countPercent = this.totalCount == 0 ? 0 : ((double) this.itemsCount[i]) / totalCount;

countCumulative += countPercent;

long sum = this.itemsSum[i];

double sumPercent = this.totalSum == 0 ? 0 : ((double) this.itemsSum[i]) / totalSum;

sumCumulative += sumPercent;

Entry e = new Entry(bucketName, count, countPercent, countCumulative,

sum, sumPercent, sumCumulative);

entries.add(e);

}

return entries;

}

/\*\*

\* Returns total number of items seen.

\*/

public int getTotalCount() {

return totalCount;

}

/\*\*

\* Returns sum of all the items seen.

\*/

public long getTotalSum() {

return totalSum;

}

}