1η Εργασία

Διαχείρηση Σύνθετων Δεδομένων
27/03/2023 – Εαρινό Εξάμηνο 2023

By

ΤΟΔΡΙ ΑΓΓΕΛΟ (ΑΜ. 3090)
Καθηγητής: Ν. Μαμουλής

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΡΓΑΣΙΑ 1 – ΙΣΤΟΓΡΑΜΜΑΤΑ – ΛΙΓΑ ΛΟΓΙΑ ΣΧΕΤΙΚΑ ΜΕ ΑΥΤΑ

Τα ιστογράμματα είναι ένα χρήσιμο εργαλείο για την οπτικοποίηση της κατανομής ενός συνόλου δεδομένων. Είναι ένα πολύτιμο εργαλείο για τη διερευνητική ανάλυση δεδομένων. Σε αυτή την άσκηση χρησιμοποιώντας Java θα εξασκηθούμε στην εργασία με δεδομένα πραγματικού κόσμου για να αποκτήσουμε κατανόηση της λειτουργίας των ιστογραμμάτων και τον τρόπο χρήσης τους για την ανάλυση και την οπτικοποίηση δεδομένων.

ΣΗΜΕΙΩΣΗ ΓΙΑ ΤΟΝ ΑΞΙΟΛΟΓΗΤΗ

Για δικιά σας διευκόλυνση όλος ο κώδικας της άσκησης και για τα 2 μέρη εξηγείται αναλυτικά και επιδεικνύεται στα παρακάτω Screenshots. Πρακτικά μπορείτε εάν το επιθυμείται να παραλείψεται την αξιολόγηση των αρχείων καθώς τα screenshots καλύπτουν όλη την έκταση του κώδικα και για τα 2 αρχεία. Για του λογου το αληθές μπορείτε να το εξακριβώσετε αυτό και οι ίδιοι.

ΜΕΡΟΣ 1: ΔΗΜΙΟΥΡΓΙΑ EQUI-WIDHT & EQUI-DEPTH HISTOGRAM

- Στο πρώτο μέρος της άσκησης, ζητήθηκε να γράψουμε ένα πρόγραμμα, το οποιο γράφηκε σε Java που διαβάζει ένα αρχείο CSV που περιέχει δεδομένα εισοδήματος και δημιουργεί δύο τύπους ιστογραμμάτων
- Ενα ιστόγραμμα ίσου πλάτους(equi-widht) και ένα ιστόγραμμα ίσου βάθους(equi-depth)
- Ένα ιστόγραμμα ίσου πλάτους χωρίζει το εύρος των εισοδημάτων σε ίσου μεγέθους bin, ενώ ένα ιστόγραμμα ίσου βάθους χωρίζει το εύρος των εισοδημάτων έτσι ώστε κάθε bin να περιέχει περίπου τον ίδιο αριθμό τιμών εισοδήματος
- Το πρόγραμμα εκτυπώνει επίσης τις ελάχιστες και μέγιστες τιμές εισοδήματος, τον αριθμό των έγκυρων τιμών εισοδήματος και τον αριθμό των bins που χρησιμοποιήθηκαν για τα ιστογράμματα

ΕΞΗΓΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΓΙΑ ΤΟ ΜΕΡΟΣ 1

Το πρόγραμμα χρησιμοποιεί μία μόνο μέθοδο (την μεθοδο main της Java) για να επιτελέσει την λειτουργικότητα που ζητείται απο το μέρος 1. Παρακάτω θα υπάρχουν Screenshot καθως και περαιτέρω εξήγηση του κώδικα οπου χρειάζεται.

ΔΙΑΒΑΣΜΑ ΑΡΧΕΙΟΥ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΛΙΣΤΑΣ ΜΕ INCOMES

```java
public class Histogram {
    public static void main(String[] args) {
        String csvFile = "C:\\Users\\a.todhri\\Desktop\\data1.csv";
        String line;
        String csvSplitBy = ",";
        ArrayList<Double> incomes = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {
            br.readLine(); // skip header line
            while ((line = br.readLine()) != null) {
                String[] fields = line.split(csvSplitBy);
                if (!fields[13].equals("")) { // check if Income value is recorded
                    incomes.add(Double.parseDouble(fields[13]));
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            return;
        }
```

Εικόνα 1 - screenshot1

Σχόλια:

- το ***csvFile*** location μπορεί είτε να είναι hard-coded είτε να δίνεται παραμετρικά απο την εκτέλεση του προγράμματος
- τα cvs αρχεια συνήθως γίνονται split με κόμματα(,) παρόλα αυτά εαν το csv γίνεται split με κάτι διαφορετικό μπορούμε να τροποποιήθσουμε την μεταβλητή ***csvSplitBy***
- χρησιμοποιούμαι ενα ArrayList απο double για να κρατήσουμε/διαχειριστούμε το σύνολο των δεδομένων για τα Incomes. Αυτό βέβαια σε μεγαλύτερους όγκους δεδομένων μπορεί να δημιουργήσει προβήματα μνήμης και να κανει το πρόγραμμα μας αργό. Για λόγους απλότητας δεν δόθηκε βαρύτητα σε θέματα Performance και μνήμης
- διατρέχοντας το αρχείο και εισάγουμε κάθε τιμή income (13^{η} στήλη στο csv) μέσα στην λίστα incomes

ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΕΚΤΥΠΩΣΗ EQUI-WIDHT HISTOGRAM

```
int numValidIncomes = incomes.size();
System.out.println(numValidIncomes + " valid income values");

double minIncome = Collections.min(incomes);
double maxIncome = Collections.max(incomes);
System.out.println("minimum income = " + minIncome + " maximum income = " + maxIncome);

int numBins = 100;
double binWidth = (maxIncome - minIncome) / numBins;

int[] equiwidthHist = new int[numBins];
for (double income : incomes) {
    int binIndex = (int) ((income - minIncome) / binWidth);
    if (binIndex == numBins) { // handle edge case of income equal to maxIncome
        binIndex--;
    }
    equiwidthHist[binIndex]++;
}

System.out.println("equiwidth:");
for (int i = 0; i < numBins; i++) {
    double lowerBound = minIncome + i * binWidth;
    double upperBound = minIncome + (i+1) * binWidth;
    System.out.printf(Locale.US, format: "range: [%.2f,%.2f), numtuples: %d\n", lowerBound, upperBound, equiwidthHist[i]);
}
```

Εικόνα 2 - screenshot2

Σχόλια:

- Τα valid incomes έγιναν filter (με βάση το εάν έχουν κάποια υπαρκτή τιμή) στην Εικόνα 1. Το σύνολο (counter) αποθηκευτηκε στην μεταβλητή ***numValidIncomes***
- Γίνεται εκτύπωση των min, max απο την λίστα με τα Incomes χρησιμοποιόντας την μέθοδο min,max απο την δομή/utility class Collections της Java (java 8)
- ***numBins*** με την χρήση αυτής της μεταβλητής ορίζουμε τον αριθμό των bins. Για μικρότερα σύνολα δεδομένων μπορούμε να τροποποιήσουμε την τιμή αυτή (πχ με τον αριθμό 10)
- έπειτα υπολογίσαμε το bin width με βάση την μεγιστη/ελαχιστη τιμή εισοδήματος
- δημιουργείται το ιστόγραμμα κάνοντας Increment το αντίστοιχο bin στο οποίο ανήκει το Income στο οποίο διερευνά η for
- Το όρισμα Locale.US χρησιμοποιείται για να διασφαλιστεί ότι ο δεκαδικός διαχωριστής είναι τελεία

ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΕΚΤΥΠΩΣΗ EQUI-DEPTH HISTOGRAM

```
        Collections.sort(incomes); // sort the incomes ArrayList in ascending order
        int[] equidepthHist = new int[numBins];
        int numItemsPerBin = numValidIncomes / numBins;
        int remainder = numValidIncomes % numBins;
        int startIndex = 0;

        System.out.println("equidepth:");
        for (int i = 0; i < numBins; i++) {
            int numItems = numItemsPerBin;
            if (i < remainder) {
                numItems++;
            }
            int endIndex = startIndex + numItems;
            double lowerBound = incomes.get(startIndex);
            double upperBound = incomes.get(endIndex - 1);
            equidepthHist[i] = numItems;
            System.out.printf(Locale.US, format: "range: [%.2f,%.2f), numtuples: %d\n", lowerBound, upperBound, numItems);
            startIndex = endIndex;
        }
    }
}
```

Εικόνα 3 – screenshot3

Σχόλια:

- Τα δεδομένα εισοδήματος ταξινομούνται χρησιμοποιώντας τη μέθοδο Collections.sort()
- δημιουργείται ένα κενό ιστόγραμμα ίσου βάθους ως έναν ακέραιο πίνακα με τον αριθμό των bins που καθορίζεται από τον χρήστη
- υπολογίζεται ο αριθμός των στοιχείων που πρέπει να τοποθετηθούν σε κάθε bin διαιρώντας τον συνολικό αριθμό των έγκυρων εισοδημάτων με τον αριθμό των bins που ορίζει ο χρήστης
- η μεταβλητή startIndex τίθεται σε 0 για να παρακολουθείται ο δείκτης έναρξης κάθε bin
- η for χρησιμοποιείται για την επανάληψη κάθε bin και τη συμπλήρωση του ιστογράμματος equi-depth
- η μεταβλητή endIndex υπολογίζεται ως το άθροισμα των startIndex και numItems, το οποίο δίνει το δείκτη του τελευταίου στοιχείου στο τρέχον bin
- οι μεταβλητές lowerBound και upperBound υπολογίζονται ως η ελάχιστη και η μέγιστη τιμή στο τρέχον bin, αντίστοιχα
- Τέλος, το εύρος τιμών και ο αριθμός των στοιχείων στο τρέχον bin εκτυπώνονται με τη μέθοδο printf(). Το όρισμα Locale.US χρησιμοποιείται για να διασφαλιστεί ότι ο δεκαδικός διαχωριστής είναι τελεία. Ο startIndex ενημερώνεται σε endIndex για την επόμενη επανάληψη του βρόχου.

ΜΕΡΟΣ 2: ΕΚΤΙΜΗΣΗ ΚΑΙ ΑΚΡΙΒΕΙΑ ΤΩΝ ΙΣΤΟΓΡΑΜΜΑΤΩΝ ΝΑ ΕΚΤΙΜΟΥΝ ΠΛΗΘΟΣ ΤΙΜΩΝ ΠΟΥ ΠΕΦΤΟΥΝ ΣΕ ΔΙΑΣΤΗΜΑ [A,B]

- Σε ενα μεγάλο μέρος το πρόγραμμα είναι επέκταση του προγράμματος στο μέρος Α. Γίνεται με κοινό τρόπο η δημιουργία των equi-depth & equi-widht ιστογραμμάτων.
- Η διαφορά είναι οτι ζητείται απο το input (program arguments) 2 τιμές income A, B
- Το πρόγραμμα θα προσπαθήσει να κάνει εκτίμηση το πλήθος των τιμών που βρίσκεται σε αυτό το εύρος τιμών [A,B] χρησιμοποιόντας τα 2 ιστογράμματα που εχουν παραχθεί απο το μέρος 1 της προγραμματιστικής άσκησης
- Τέλος θα εκτυπώσει εκτίμηση equi-widht, εκτίμηση equi-depth και πραγματικό πλήθος τιμών στο dataset

ΕΙΣΟΔΟΣ ΠΑΡΑΜΕΤΡΩΝ ΑΠΟ ΤΟ INPUT (PROGRAM ARGUMENTS)

```
if (args.length != 2) {
    System.out.println("Usage: java Histogram2 <a> <b>");
    return;
}
double a = Double.parseDouble(args[0]);
double b = Double.parseDouble(args[1]);
```

Εικόνα 4 - screenshot4

Σχόλια:

- Το πρόγραμμα ελέγχει αν του έχουν δοθεί ακριβώς δύο ορίσματα και αν όχι, τερματίζει με μήνυμα σφάλματος.

ΔΗΜΙΟΥΡΓΙΑ EQUI-DEPTH & EQUI-WIDHT ΙΣΤΟΓΡΑΜΜΑΤΩΝ

Τα σχόλια παραλείπονται καθώς έχει γίνει επεξήγηση παραπάνω στο μέρος 1 της προγραμματιστικής άσκησης. Δείτε εδώ.

ΔΗΜΙΟΥΡΓΙΑ ΕΚΤΙΜΗΣΗΣ ΠΛΗΘΟΥΣ ΤΙΜΩΝ ΣΤΟ ΙΣΤΟΓΡΑΜΜΑ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ EQUI-WIDTH ΙΣΤΟΓΡΑΜΜΑ

```
        // estimate result using equiwidth histogram
        int lowerBinIndex = (int) ((a - minIncome) / binWidth);
        if (lowerBinIndex == numBins) {
            lowerBinIndex--;
        }
        int upperBinIndex = (int) ((b - minIncome) / binWidth);
        if (upperBinIndex == numBins) {
            upperBinIndex--;
        }
        double equiwidthResult = 0;
        for (int i = lowerBinIndex; i <= upperBinIndex; i++) {
            double lowerBound = minIncome + i * binWidth;
            double upperBound = minIncome + (i + 1) * binWidth;
            double overlap = Math.min(upperBound, b) - Math.max(lowerBound, a);
            double binFraction = overlap / binWidth;
            equiwidthResult += equiwidthHist[i] * binFraction;
        }
        System.out.printf(Locale.US, format: "equiwidth estimated results: %.12f%n", equiwidthResult);
```

Εικόνα 5 - screenshot5

Σχόλια:

- Δημιουργεί ένα ιστόγραμμα equiwidth αναθέτοντας κάθε τιμή εισοδήματος σε ένα bin με βάση την τιμή του σε σχέση με τις ελάχιστες και μέγιστες τιμές, και στη συνέχεια υπολογίζει το κλάσμα κάθε bin που εμπίπτει στο δεδομένο εύρος [a, b]. Στη συνέχεια προσθέτει αυτά τα κλάσματα για να εκτιμήσει τον συνολικό αριθμό των εισοδημάτων εντός του εύρους

ΔΗΜΙΟΥΡΓΙΑ ΕΚΤΙΜΗΣΗΣ ΠΛΗΘΟΥΣ ΤΙΜΩΝ ΣΤΟ ΙΣΤΟΓΡΑΜΜΑ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ EQUI-DEPTH ΙΣΤΟΓΡΑΜΜΑ

```
    // estimate equidepth results
    double equidepthEstimate = 0.0;
    startIndex = 0;
    for (int i = 0; i < numBins; i++) {
        double lowerBound = incomes.get(startIndex);
        double upperBound = incomes.get(startIndex + equidepthHist[i] - 1);
        if (lowerBound < b && upperBound >= a) {
            double binPercentage = 1.0;
            if (lowerBound < a) {
                binPercentage = (upperBound - a) / (upperBound - lowerBound);
            } else if (upperBound >= b) {
                binPercentage = (b - lowerBound) / (upperBound - lowerBound);
            }
            equidepthEstimate += equidepthHist[i] * binPercentage;
        }
        startIndex += equidepthHist[i];
    }
    System.out.printf(Locale.US, format: "equidepth estimated results: %.12f%n", equidepthEstimate);

    // print actual results for incomes within the range
    System.out.println("actual results: " + incomes.stream().filter(income -> income >= a && income < b).count());
}
```

Εικόνα 6 - screenshot6

Σχόλια:

- Δημιουργεί ένα ιστόγραμμα ίσου βάθους ταξινομώντας πρώτα την ArrayList εισοδημάτων σε αύξουσα σειρά και στη συνέχεια αναθέτοντας ίσο αριθμό εισοδημάτων σε κάθε bin. Στη συνέχεια εκτιμά τον συνολικό αριθμό των εισοδημάτων εντός του εύρους υπολογίζοντας το ποσοστό κάθε δοχείου που εμπίπτει στο εύρος και προσθέτοντας τα κλάσματα που προκύπτουν.
- Τέλος, εκτυπώνει τον εκτιμώμενο αριθμό εισοδημάτων εντός του εύρους με βάση τις δύο μεθόδους ιστογράμματος, καθώς και τον πραγματικό αριθμό εισοδημάτων εντός του εύρους.

ΜΕΡΟΣ 2: ΠΟΡΙΣΜΑΤΑ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

Καταληκτική ερώτηση: Μέσω πειραματισμού αποφανθείτε για το αν το equiwidth histogram υπερτερεί ή υστερεί του equi-depth histogram αφού δοκιμάσετε έναν μεγάλο αριθμό ερωτήσεων με διάφορα εύρη

Η απάντηση εξαρτάται από διάφορους παράγοντες, όπως η κατανομή των δεδομένων, το μέγεθος του συνόλου δεδομένων και τα συγκεκριμένα εύρη που δοκιμάζονται. Σε γενικές γραμμές, τα ιστογράμματα ίσου βάθους τείνουν να αποδίδουν καλύτερα όταν έχουν να κάνουν με ασυμετρικές κατανομές δεδομένων, ενώ τα ιστογράμματα ίσου πλάτους μπορεί να αποδίδουν καλύτερα όταν έχουν να κάνουν με πιο ομοιόμορφες κατανομές. Κατα τις δοκιμές φαίνεται να είναι πιο κοντά στα πραγματικά αποτελέσματα τα αποτελέσματα των εκτιμήσεων **equi-depth** αλλα αυτό απο μόνο του δεν είναι αρκετό για να δείξει οτι τα διαγράμματα αυτά υπερτερούν εναντι των equi-widht.

2 τυχαίες εκτελέσεις του προγράμματος μας 2 με διάφορα εύρη τιμών:

java Histogram2 1000 60000

equiwidth estimated results: 46456.891378448770

equidepth estimated results: 46387.300417246170

actual results: 46316

java Histogram2 55000 60000

equiwidth estimated results: 5142.437017295106

equidepth estimated results: 5099.601787109189

actual results: 5063

java Histogram2 19000 55000

equiwidth estimated results: 39354.366524606020

equidepth estimated results: 39334.042957313766

actual results: 39361

Εκτέλεση output μέρους 1ου της προγραμματιστικής άσκησης:

C:\Users\a.todhri\Desktop\MYE041>java Histogram

72901 valid income values

minimum income = 2611.0 maximum income = 248750.0

equiwidth:

range: [2611.00,5072.39), numtuples: 13

range: [5072.39,7533.78), numtuples: 23

range: [7533.78,9995.17), numtuples: 85

range: [9995.17,12456.56), numtuples: 240

range: [12456.56,14917.95), numtuples: 386

range: [14917.95,17379.34), numtuples: 687

range: [17379.34,19840.73), numtuples: 799

…

equidepth:

range: [2611.00,14814.00), numtuples: 730

range: [14821.00,17456.00), numtuples: 729

range: [17457.00,19731.00), numtuples: 729

range: [19732.00,21265.00), numtuples: 729

range: [21266.00,22475.00), numtuples: 729

range: [22478.00,23874.00), numtuples: 729

range: [136250.00,154583.00), numtuples: 729

…

Εκτέλεση output μέρους 2ου της προγραμματιστικής άσκησης:

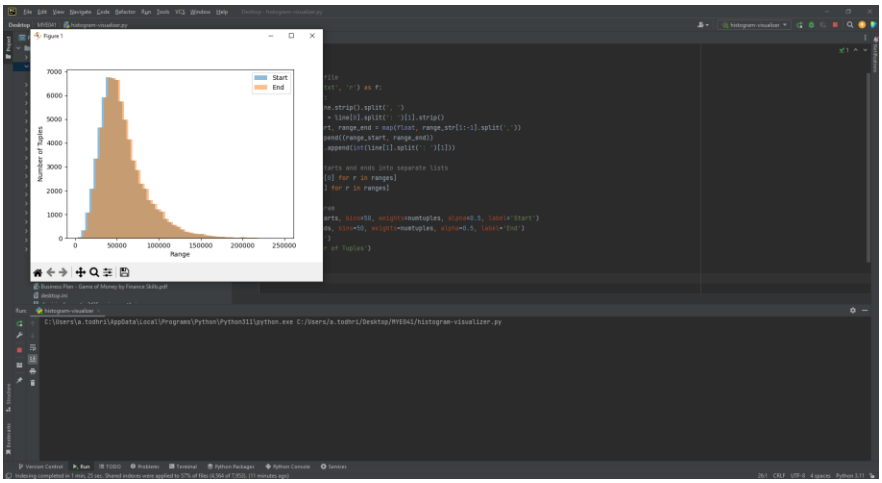
C:\Users\a.todhri\Desktop\MYE041>java Histogram2 19000 55000

equiwidth estimated results: 39354.366524606020

equidepth estimated results: 39334.042957313766

actual results: 39361

ΑΠΕΙΚΟΝΙΣΗ EQUI-WIDHT ΙΣΤΟΓΡΑΜΜΑΤΟΣ ΜΕ ΧΡΗΣΗ PYTHON ΚΑΙ MATPLOTLIB.PYPLOT




```
import matplotlib.pyplot as plt

ranges = []
numtuples = []

# Read data from file
with open('input.txt', 'r') as f:
    for line in f:
        line = line.strip().split(', ')
        range_str = line[0].split(': ')[1].strip()
        range_start, range_end = map(float, range_str[1:-
1].split(','))
        ranges.append((range_start, range_end))
        numtuples.append(int(line[1].split(': ')[1]))

# Extract range starts and ends into separate lists
range_starts = [r[0] for r in ranges]
range_ends = [r[1] for r in ranges]

# Generate histogram
plt.hist(range_starts, bins=50, weights=numtuples,
alpha=0.5, label='Start')
plt.hist(range_ends, bins=50, weights=numtuples, alpha=0.5,
label='End')
plt.xlabel('Range')
plt.ylabel('Number of Tuples')
plt.legend()
plt.show()
```