Yuan Wang
440 Data Mining
Prof. Jiebo Luo
Homework 2

2.3
Calculated the accumulative frequency as below:

age	Frequency	accumulative frequency
1-5	200	200
6-15	450	650
16-20	300	950
21-50	1500	2450
51-80	700	3150
81-110	44	3194

N = 3194
$\frac{N}{2}$ = ½ * 3194 = 1597
So, L_l = 20
$\sum freq_l$ = 950
$freq_{median}$ = 1500
Width = 50 – 20 = 30

Thus,
Median = $L_l + \left(\frac{\frac{N}{2} - (\sum freq_l)}{freq_{median}}\right)$ width = 20 + $\left(\frac{1597 - 950}{1500}\right)$ * 30 = 32.94 years

2.6
A: Euclidean distance = $\sqrt{(22-20)^2 + (1-0)^2 + (42-36)^2 + (10-8)^2}$ = 6.7

B: Manhattan distance = $|22-20|$ + $|1-0|$ + $|42-36|$ + $|10-8|$ = 11

C: Minkowski distance = $\sqrt[3]{(22-20)^3 + (1-0)^3 + (42-36)^3 + (10-8)^3}$ =6.2

D: Supremum distance = 42 – 36 = 6

2.7
The median formula “Median = $L_l + \left(\frac{\frac{N}{2} - (\sum freq_l)}{freq_{median}}\right)$ width”, which is convenience to calculate a small size of data for the median value. However, when the data size is big, even divide the data into k equal groups, there still will be a big cost for calculating the median. The better way to do

it is as follows: first, hierarchically divide the whole data into k regions, find the region where median is contained. Then divide again this region into k sub-regions and find the sub-region which median resides. Iteratively doing this until the width of sub-region reaches a predefined threshold, then apply the median formula to get median value. In this way, we could avoid involving the whole data for the calculation which is expensive.

2.8

A:

The corresponding equations are as follow:

Euclidean distance = $\sqrt{sum_i(X_i - Y_i)^2}$

Manhattan distance = $sum_i|X_i - Y_i|$

Supremum distance = $max_i|X_i - Y_i|$

Cosine similarity = $\frac{x^t * y}{\|x\|\|y\|}$

The result of calculation is as below:

	A_1	A_2	Euclidean distance	Manhattan distance	Supremum distance	Cosine Similarity
X1	1.5	1.7	0.14	0.2	0.1	0.99999
X2	2	1.9	0.67	0.9	0.6	0.99575
X3	1.6	1.8	0.28	0.4	0.2	0.99997
X4	1.2	1.5	0.22	0.3	0.2	0.99903
X5	1.5	1	0.61	0.7	0.6	0.96536

X	1.4	1.6

B:

Use formula $A_1' = \frac{A_1}{\sqrt{(A_1^2 + A_2^2}}$, $A_2' = \frac{A_2}{\sqrt{(A_1^2 + A_2^2}}$ to normalize data set to make the norm of each data point equal to 1.

Then calculated the Euclidean distance accordingly:

	A_1	A_2	A_1'	A_2'	Euclidean distance
X1	1.5	1.7	0.6616	0.7498	0.0041
X2	2	1.9	0.7250	0.6887	0.0922
X3	1.6	1.8	0.6644	0.7474	0.0078
X4	1.2	1.5	0.6247	0.7809	0.0441
X5	1.5	1	0.8321	0.5547	0.2632

X	1.4	1.6	0.6585	0.7526

3.1
There are many examples in real life that prove the assessment of data quality can depend on the intended use of the data.
Accuracy. For example, Wegmans stores all the customer information in their system. If a marketing analyst wants to analyze the customer distribution in a particular store, zipcode is enough for the analysis. On the other hand, if customer service dept wants to send gifts to a rewarded customer, they will need accurate home address from system.
Completeness. When customer center verifies customer's identity, they would only check 4 digits of SSN. In same bank, when customer apply new credit card or mortgage loan, they have to provide 9 digits SSN.
Consistency. In AT&T account system, the information of mobile service cost must be consistent no matter customer or AT&T staff check it in their system or not.

The other dimensions can be used to assess data quality are such as **timeliness, believability.**
Timeliness: for example, the voice data transferred between two cell phone requires very small-time delay
Believability: The data value must be within a certain range.

3.3
A:

1. Sort the data
2. Using a bin depth of 3
   Bin 1: 13, 15, 16 Bin 2: 16, 19, 20 Bin 3: 20, 21, 22
   Bin 4: 22, 25, 25 Bin 5: 25, 25, 30 Bin 6: 33, 33, 35
   Bin 7: 35, 35, 35 Bin 8: 36, 40, 45 Bin 9: 46, 52, 70

3. Smoothing by bin means:
   Bin 1: 142/3, 142/3, 142/3 Bin 2: 181/3, 181/3, 181/3 Bin 3: 21, 21, 21
   Bin 4: 24, 24, 24 Bin 5: 262/3, 262/3, 262/3 Bin 6: 332/3, 332/3, 332/3
   Bin 7: 35, 35, 35 Bin 8: 401/3, 401/3, 401/3 Bin 9: 56, 56, 56

This method smooths a sorted data value by consulting to its "neighborhood". In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.[1]

B:
Outliers in the data may be detected by clustering, where similar values are organized into groups, or "clusters." Values that fall outside of the set of clusters may be considered outliers.[2]

C:
Besides the smoothing by bin mean, there are smoothing by bin median or boundaries. Furthermore, data smoothing can also be done by regression.

---

[1] *Ch 3.2.2, Data Mining: Concepts and Techniques", 3/E, by Jiawei Han, Michelin Kamber and Jian Pei.*
[2] *Ch 3.2.2, Data Mining: Concepts and Techniques", 3/E, by Jiawei Han, Michelin Kamber and Jian Pei*

3.5
A: Min-max normalization value range can be any value range. Between [new_min, new_max]
B:
$V_i' = \frac{V_i - \bar{A}}{\sigma_A}$, so
Z-score normalization value range is [$\frac{min_A - \bar{A}}{\sigma_A}$, $\frac{max_A - \bar{A}}{\sigma_A}$]

C:
$V_i' = \frac{V_i - \bar{A}}{S_A}$
the mean absolute deviation of A, denoted by S_A,
$S_A = \frac{1}{n}(|V_1 - \bar{A}| + |V_2 - \bar{A}| + \ldots + |V_n - \bar{A}|)$
The value range is [$\frac{min_A - \bar{A}}{S_A}$, $\frac{max_A - \bar{A}}{S_A}$]

D:
The value range of normalization by decimal scaling is [$\frac{min_A}{10^j}$, $\frac{max_A}{10^j}$]
where j is the smallest integer such that Max ($|\frac{V_i}{10^j}|$) < 1.

3.7
A:
min_A = 13, max_A = 70, new_min_A = 0, new_max_A = 1
V = 35,
V' = $\frac{35-13}{70-13}(1-0)+0$ = 0.39

B:
$\bar{A}$ = 809/27 = 29.96
σ_A = 12.94
V = 35,
V' = $\frac{35-29.96}{12.94}$ =0.39

C:
V= 35, j = 2
V' = $\frac{35}{10^2}$ = 0.35

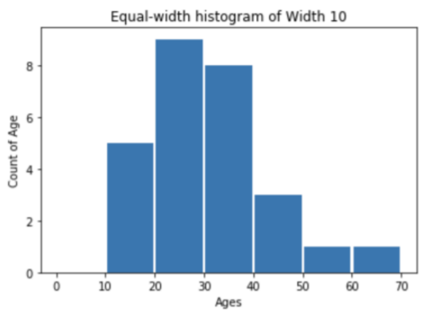
D:
Decimal scaling is the preferred to use for given data. It maintained the data distribution and is easy to understand. Comparing to decimal scaling, min-max normalization does not permit any future values to fall outside the current minimum and maximum values without encountering an "out of bounds error". For z-score normalization, it does not increase the information value of the attribute in terms of intuitiveness to users or in usefulness of mining results

3.11
A:

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [2]: age = [13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 33, 35, 35, 35, 36, 40, 45, 46, 52, 70]
        plt.hist(age, bins=[0, 10, 20, 30, 40, 50, 60, 70], rwidth=0.95, align='mid', label='Ages')
        plt.xlabel('Ages')
        plt.ylabel('Count of Age')
        plt.title('Equal-width histogram of Width 10')

Out[2]: Text(0.5, 1.0, 'Equal-width histogram of Width 10')
```



B:

Tuples

T1	13
T2	15
T3	16
T4	16
T5	19
T6	20
T7	20
T8	21
T9	22

T10	22
T11	25
T12	25
T13	25
T14	25
T15	30
T16	33
T17	33
T18	33

T19	33
T20	35
T21	35
T22	36
T23	40
T24	45
T25	46
T26	52
T27	70

SRSWOR vs. SRSWR

SRSWOR	(n = 5)
T3	16
T7	20
T11	25
T12	25
T27	70

SRSWR	(n = 5)
T3	16
T7	20
T7	20
T14	25
T18	33

Clustering sampling: Initial clusters											
T1	13	T6	20	T11	25	T16	33	T21	35	T26	52
T2	15	T7	20	T12	25	T17	33	T22	36	T27	70
T3	16	T8	21	T13	25	T18	33	T23	40		
T4	16	T9	22	T14	25	T19	33	T24	45		
T5	19	T10	22	T15	30	T20	35	T25	46		

Stratified Sampling

T1	13	Young	T10	22	Young	T19	33	Middle aged
T2	15	Young	T11	25	Young	T20	35	Middle aged
T3	16	Young	T12	25	Young	T21	35	Middle aged
T4	16	Young	T13	25	Young	T22	36	Middle aged
T5	19	Young	T14	25	Young	T23	40	Middle aged
T6	20	Young	T15	30	Middle aged	T24	45	Middle aged
T7	20	Young	T16	33	Middle aged	T25	46	Middle aged
T8	21	Young	T17	33	Middle aged	T26	52	Middle aged
T9	22	Young	T18	33	Middle aged	T27	70	Senior

Stratified Sampling (according to age)

T1	13	Young
T2	15	Young
T15	30	Middle aged
T16	33	Middle aged
T27	70	Senior

3.13
code is in 2nd attachment Homework 2.ipynb
A:

```
# 3.13 a
# Calculate the count number of distinct value in determined attributes
# Create a list that contains attributes' name
att = att list
#iterate each attribute
for i in range(att):
    count_num = value_counts()
# create a dictionary of attributes and their count number of distinct values
hierarchy = dict(att=attribute name; value =count number of each attributes)

# Sort the dictionary by the value
hierarchy = sorted dict by value

# Print the result
for i in hierarchy:
   print every strata
```

B:

```
# 3.13 b
#equal width
def equiwidth(da, m): # da is data, m is number of bin
    a = len(da)
    w = int((max(da) - min(da)) / m) # calculate the bin width
    min1 = min(da)
    dt = []
    for i in range(0, m + 1):
        dt = dt + [min1 + w * i] # iterate bin
    dti=[]

    for i in range(0, m):
        temp = []
        for j in arr1:
            if j > dt[i] and j < dt[i+1]:
                temp += [j]
        dti += [temp]
    print(dti)

# load data
# implement
equiwidth(data, number)
```

C:

```
# 3.13 c
#equal depth(frequency)
def equidepth(da, m): # da is data, m is number of bin

    a = len(da)
    n = int(a / m) # calculate the bin width
    for i in range(0, m): # iterate bin
        dt = []
        for j in range(i * n, (i + 1) * n):
            if j >= a:
                break
            dt = dt + [arr1[j]]
        print(dt)

# load data
# implement
equidepth(data, number)
```