University of Southern Denmark
IMADA
Ricardo Campello

DM583: Data Mining

Exercise 2: Data Representation

Exercise 2-1 Variable Types

a) Identify the types of variables in the above example assuming that Exam X is measuring the amount of certain cells per unit of volume and cannot be fractional. NOTE: Consider variable types according to the data mining taxonomy discussed in the course, NOT according to variable types in any particular programming language.

Name	Temp.	Nausea	Rash Area	Pain	Exam X	Result
Chloe	38.3	yes	large	no	300	Negative
Mary	37.8	no	large	yes	450	Positive
Anne	37.4	no	large	yes	420	Negative
Leah	37.0	no	small	no	550	Negative
James	37.7	yes	small	yes	500	Positive
John	39.1	no	medium	yes	1000	Positive

Proposed Solution:

Name: it is clearly a **categorical** variable. And since in principle (in most applications) there is no reason to impose any order between names, this categorical variable is **nominal**.
Temperature: since temperature can be any real-valued number within a certain interval, this is a **numerical continuous variable**.
Nausea: Given the values of this variable, which indicate whether a patient has (yes) or not (no) nausea, this is a binary variable. Although these two values could be represented differently (e.g. 1 and 0 instead of yes and no), as they stand now, they are nominal values, and therefore this variable is a **binary nominal variable**.
Rash Area: The categorical values clearly suggest an order small < medium < large, case in which this variable is then a **categorical ordinal variable**.
Pain: Same as nausea.
Exam X: Since the variable is measuring an amount of certain cells per unit of volume this is certainly a numerical variable, and given that its values cannot be fractional, it is a **numerical discrete variable**.
Result: Same as pain and nausea.

b) Identify appropriate types for the following variables:

- Number of certain blood cells in a blood sample
- The relative rank of each student of a class with respect to their final grades
- Length of an object as measured to any desired precision
- The hair color of an individual
- The postal code of an individual

Hint: think not only of the values that they possibly take, but also the operations that make sense for each of them.

Proposed Solution:

- Number of certain blood cells in a blood sample:
  - Since this is a count of cells, this variable is clearly a **numeric discrete variable**.
- The relative rank of each student of a class with respect to their final grades
  - Relative ranks are an ordering (1^{st}, 2^{nd}, 3^{rd}, …). In principle, they could be represented as a numerical variable (with values 1, 2, 3, …). However, we would unlikely want/need to perform arithmetic operations with ranks – e.g., what would be the meaning of adding the relative positions of the 1^{st} and 5^{th} best students, 1 + 5 = 6? Rather, we will most likely just want to compare students (e.g. "is rank of student A < rank of student B?"), case in which this variable is better represented as a **categorical ordinal variable**.
- Length of an object as measured to any desired precision
  - The length measured to any desired precision can be any non-negative real-valued number, therefore this is a **numeric continuous variable**.
- The hair color of an individual
  - Hair color takes on a finite set of nominal values (red, black, etc.) which in principle do not follow any order, therefore this is a **categorical nominal variable**.
- The postal code of an individual
  - In many countries, postal codes can take on alphanumeric characters (both digits and letters), i.e., they are values of a **categorical nominal variable**.
    - **Note**: In countries where postal codes contain numerical digits only, these codes can be represented as a numerical variable internally in a computer. Even though *arithmetic operations on post codes would unlikely make any sense*, representing these codes as numbers may be more computationally efficient than as categorical values, depending on the application scenario in practice. Conceptually, however, they are still categorical.

Exercise 2-2 Distances for Numerical Data

(a) Manually compute the Euclidean distance between every pair of data observations, $\mathbf{x}_i = [x_{i1}\ x_{i2}\ ...\ x_{i4}]$ and $\mathbf{x}_j = [x_{j1}\ x_{j2}\ ...\ x_{j4}]$, each of which is described by 4 numerical variables, as shown in the dataset below, then check your results by recomputing the distances using function `dist()` with argument `method = "euclidean"` in R:

	Variable 1	Variable 2	Variable 3	Variable 4
x_1	2	-1	1	0
x_2	7	0	-4	8
x_3	4	3	5	2
x_4	5	10	-1	5

Proposed Solution:

Manually:
$d(x_1, x_2) = \sqrt{(2-7)^2 + (-1-0)^2 + (1-(-4))^2 + (0-8)^2} = \sqrt{115} = 10.72$
$d(x_1, x_3) = \sqrt{(2-4)^2 + (-1-3)^2 + (1-5)^2 + (0-2)^2} = \sqrt{40} = 6.32$
… (analogous procedure for all other pairs – see results from the R solution below)

In R:

```
x1 <- c(2,-1,1,0)
x2 <- c(7,0,-4,8)
x3 <- c(4,3,5,2)
x4 <- c(5,10,-1,5)
X <- rbind(x1,x2,x3,x4)
D <- dist(X, method = "euclidean", diag = TRUE, upper = TRUE)
> D
          x1        x2        x3        x4
x1  0.000000 10.723805  6.324555 12.609520
x2 10.723805  0.000000 11.618950 11.045361
x3  6.324555 11.618950  0.000000  9.746794
x4 12.609520 11.045361  9.746794  0.000000
```

(b) Repeat item (a) above now using Manhattan rather than Euclidean distance.

Proposed Solution:

Manually:
$d(x_1, x_2) = |2 - 7| + |-1 - 0| + |1 - (-4)| + |0 - 8| = 19$
$d(x_2, x_4) = |7 - 5| + |0 - 10| + |-4 - (-1)| + |8 - 5| = 18$
… (analogous procedure for all other pairs – see results from the R solution below)

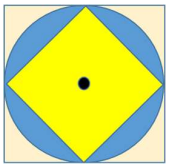
In R:

```
x1 <- c(2,-1,1,0)
x2 <- c(7,0,-4,8)
x3 <- c(4,3,5,2)
x4 <- c(5,10,-1,5)
X <- rbind(x1,x2,x3,x4)
D <- dist(X, method = "manhattan", diag = TRUE, upper = TRUE)
> D
    x1 x2 x3 x4
x1  0 19 12 21
x2 19  0 21 18
x3 12 21  0 17
x4 21 18 17  0
```

(c) Given a fixed reference point $\mathbf{x}_1$ on the plane, and another point $\mathbf{x}_2$ at a distance $d(\mathbf{x}_1,\mathbf{x}_2)$ from $\mathbf{x}_1$, it is easy to see that the set of all points that are at this very same distance from $\mathbf{x}_1$ form a circle with radius $d(\mathbf{x}_1,\mathbf{x}_2)$ centred at $\mathbf{x}_1$ if d is the **Euclidean** distance. What are the shapes formed if d is: **Manhattan**? **Suprema**?

Proposed Solution:

Euclidean distance (equidistant points form a circle)
Manhattan distance (equidistant points form a diamond)
Suprema distance (equidistant points form a square)



(d) Given two records (i.e., data objects or observations) **p** and **q**, each of which is described by n=6

numerical variables, as follows:

$$\mathbf{p} = [1 \quad 2 \quad -3 \quad 2 \quad 0 \quad 8]$$
$$\mathbf{q} = [0 \quad 6 \quad 2 \quad -1 \quad 2 \quad 5]$$

Compute the Euclidean, Manhattan, and Suprema distances: (i) manually; (ii) using R

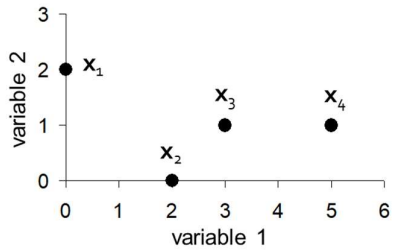
Proposed Solution:

Manually:
$d_E(p, q) = \sqrt{(1-0)^2 + (2-6)^2 + (-3-2)^2 + (2-(-1))^2 + (0-2)^2 + (8-5)^2} = \sqrt{64} = 8$
$d_M(p, q) = |1-0| + |2-6| + |-3-2| + |2-(-1)| + |0-2| + |8-5| = 18$
$d_S(p, q) = \max\{ |1-0| , |2-6| , |-3-2| , |2-(-1)| , |0-2| , |8-5| \} = 5$

In R:

```
p <- c(1, 2, -3, 2, 0, 8)
q <- c(0, 6, 2, -1, 2, 5)

Euclidean <- sqrt(sum((p-q)^2)) # or Euclidean <- dist(rbind(p,q),method="euclidean")
# Result Euclidean = 8

Manhattan <- sum(abs(p-q)) # or Manhattan <- dist(rbind(p,q),method="manhattan")
# Result Manhattan = 18

Suprema <- max(abs(p-q)) # or Suprema <- dist(rbind(p,q),method="maximum")
# Result Manhattan = 5
```

(e) Given a dataset with four data points (i.e., data objects or observations), each of which is described by n=2 numerical variables, as follows:



Compute the distance matrices for this dataset using Euclidean, Manhattan, and Suprema distances: (i) Manually (at least one non-diagonal entry of each matrix); and (b) Using R (the whole matrices)

Proposed Solution:

Manually (one entry for each distance):
$d_E(x_4, x_2) = \sqrt{(5-2)^2 + (1-0)^2} = \sqrt{10} = 3.1622$
$d_M(x_4, x_2) = |5-2| + |1-0| = 4$
$d_S(x_4, x_2) = \max\{ |5-2| , |1-0| \} = 3$

In R:

```
> X <- data.frame(variable1 = c(0,2,3,5), variable2 = c(2,0,1,1))
> D_Euclidean <- dist(X, method = "euclidean", diag = TRUE, upper = TRUE)
> D_Euclidean
         1        2        3        4
1 0.000000 2.828427 3.162278 5.099020
2 2.828427 0.000000 1.414214 3.162278
3 3.162278 1.414214 0.000000 2.000000
4 5.099020 3.162278 2.000000 0.000000

> D_Manhattan <- dist(X, method = "manhattan", diag = TRUE, upper = TRUE)
> D_Manhattan
  1 2 3 4
1 0 4 4 6
2 4 0 2 4
3 4 2 0 2
4 6 4 2 0

> D_Suprema <- dist(X, method = "maximum", diag = TRUE, upper = TRUE)
> D_Suprema
  1 2 3 4
1 0 2 3 5
2 2 0 1 3
3 3 1 0 2
4 5 3 2 0
```

(f) Consider the *Ionosphere Radar Data*, which can be found online, e.g., from the archived UCI repository:

- Dataset description at: https://archive.ics.uci.edu/ml/datasets/Ionosphere
- Dataset as a CSV file (no variable names/header, "." as decimal point, and "," as separator) at: https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data

This dataset contains 351 records (rows) described by 34 numeric variables (columns). The 35th column is a categorical class label ("good" or "bad"). You are asked the following: read the file into a data frame in R, convert it to a numeric matrix getting rid of the class label (last column), then compute the Minkowski distances corresponding to the first 10 records (rows) of the remaining data matrix (with 34 columns), using the Minkowski parameter set to $p = 1$, $p = 2$, and $p = \infty$.

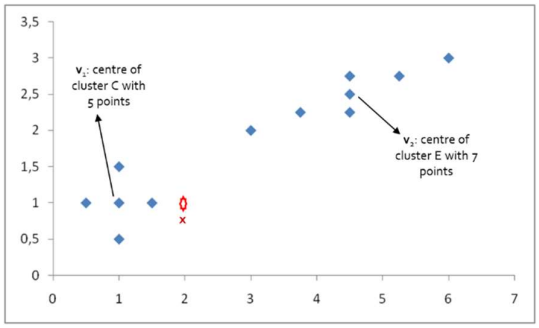
Proposed Solution:

Note that Minkowski with $p = 1$, $p = 2$ and $p = \infty$ correspond to the Manhattan, Euclidean and Suprema distances, respectively. Then, the solution can be as follows:

```
> Ionosphere <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data",
                           header = FALSE, sep = ",", dec = ".")

> dim(Ionosphere)
[1] 351  35

> Ionosphere <- as.matrix(Ionosphere[,-35])

> dist(Ionosphere[1:10,], method = "euclidean")
         1        2        3        4        5        6        7        8        9
2  2.776359
3  1.169728 3.380033
4  4.772563 4.454509 4.671267
5  1.377347 2.540531 1.838508 4.728259
6  3.049072 2.783883 3.129187 3.853289 2.721250
7  1.530471 2.753930 2.399696 5.509146 1.433354 3.562901
8  4.803640 4.859196 4.679506 3.754628 4.871717 3.820612 5.447144
9  1.205018 3.472495 1.695460 5.587566 2.013416 3.923487 1.491085 5.317335
10 3.564314 3.271473 3.452336 4.107967 3.156991 1.413085 4.095763 4.031295 4.330577

> dist(Ionosphere[1:10,], method = "manhattan")
         1        2        3        4        5        6        7        8        9
2  13.08095
3   5.35971 15.80038
4  21.05729 20.76960 20.30478
5   6.21387 11.52616  8.50424 20.75218
6  15.16631 12.58796 14.24844 17.81826 11.78266
7   7.57704 12.28109 11.57535 24.99635  6.42595 17.61223
8  22.28699 22.63900 21.16584 15.73696 22.27638 15.66824 26.06633
9   5.81361 16.54506  7.82230 25.08336  9.06074 20.15200  6.56417 25.00768
10 17.20169 14.54198 16.08502 19.09482 14.07940  5.57928 19.53089 16.48354 21.27012
```

```
> dist(Ionosphere[1:10,], method = "maximum")

        1       2       3       4       5       6       7       8       9
2  1.12221
3  0.45772 1.10868
4  1.60536 1.45300 1.75535
5  0.53525 1.02974 0.66143 1.65697
6  0.97202 1.02959 1.09924 1.11949 1.04064
7  0.48305 1.26313 0.69877 1.82510 0.55828 1.04525
8  1.85243 1.62237 1.75535 1.39330 1.65158 1.10196 1.82510
9  0.39934 1.42077 0.57480 1.90392 0.81820 1.09924 0.62314 1.92570
10 1.30906 1.36117 1.20908 1.63728 1.03762 0.75303 1.25429 1.63728 1.38233
```

(g) Consider the following toy example seen in our lecture slides, involving a 2-dimensional dataset with two natural clusters of points (in blue diamonds), namely, cluster *C* (with 5 points at the bottom left) and cluster *E* (with 7 points, at the top right), as well as an arbitrary reference point **x** (red circle):




```
C <- matrix(c(0.5, 1, 1, 0.5, 1, 1, 1, 1.5, 1.5, 1), 5, 2, byrow = TRUE) # Cluster C
v1 <- c(mean(C[,1]),mean(C[,2])) # Compute Centre of Cluster C (2-dimensional mean)
cov_C <- cov(C) # Compute Covariance Matrix of Cluster C
x <- c(2,1) # Point x
dxv1_sq <- mahalanobis(x, v1, cov_C) # [SQUARED] Mahalanobis dist. from x to v1
dxv1 <- sqrt(dxv1_sq) # Mahalanobis dist. from x to v1
```

With the R code above, we can compute the Mahalanobis distance from **x** = [2 1] to $\mathbf{v}_1$ = [1 1] as the centre of cluster *C*. The resulting distance is $d(\mathbf{x},\mathbf{v}_1) = 2.82$, or $d(\mathbf{x},\mathbf{v}_1)^2 = 8$. If the covariance matrix of cluster *E* is:

$$\text{Cov\_E} = \begin{bmatrix} 0.80 & 0.27 \\ 0.27 & 0.11 \end{bmatrix}$$

and the centre is $\mathbf{v}_2$ = [4.5 2.5], compute de distance from **x** to $\mathbf{v}_2$, $d(\mathbf{x},\mathbf{v}_2)$, with the aid of function `mahalanobis()` in R. The result of the exercise above should be $d(\mathbf{x},\mathbf{v}_2) = 5.53$, or $d(\mathbf{x},\mathbf{v}_2)^2 = 30.62$. Notice that, in this case, **x** is much closer to Cluster *C* than it is to Cluster *E*. Repeat the computations using the formula directly, rather than the `mahalanobis()` function. **Note**: This function in R returns the *squared* Mahalanobis distance.

Now, assume we moved point x slightly to position [2 1.5]. Re-compute distances $d(\mathbf{x},\mathbf{v}_1)$ and $d(\mathbf{x},\mathbf{v}_2)$, noticing that only point **x** and the Mahalanobis distances themselves need to be re-computed, nothing else. The result should be $d(\mathbf{x},\mathbf{v}_1) = 3.16$ ($d(\mathbf{x},\mathbf{v}_1)^2 = 10$) and $d(\mathbf{x},\mathbf{v}_2) = 3.01$ ($d(\mathbf{x},\mathbf{v}_2)^2 = 9.10$). That is, **x** is now closer to $\mathbf{v}_2$ (Ellipsoidal Cluster *E*) than to $\mathbf{v}_1$ (Spherical Cluster *C*).

Notes and Hints:

When using the formula, you can still possibly have the aid of R to perform algebraic operations:

- $d(\mathbf{x},\mathbf{v}) = \sqrt{(\mathbf{x}-\mathbf{v})^T Cov(\mathbf{X})^{-1}(\mathbf{x}-\mathbf{v})}$

Note that **X** in the equation above should be the subset (cluster) of interest, whereas Cov(**X**) is its covariance matrix, with inverse $\text{Cov}(\mathbf{X})^{-1}$. The inverse of a matrix, when it exists, can be computed in R, e.g., using

command `solve()`.You can create arrays of any arbitrary dimensions, including 1-dimensional *row vectors* or *column vectors* (matrices with a single column or row) using the command `array()` in R. You can transpose a matrix using the `t()` command.

Proposed Solution:

In R using the built-in function `mahalanobis()`:

```
> C <- matrix(c(0.5, 1, 1, 0.5, 1, 1, 1, 1.5, 1.5, 1), 5, 2, byrow = TRUE) # Cluster C
> v1 <- c(mean(C[,1]),mean(C[,2])) # Compute Centre of Cluster C (2-dimensional mean)
> Cov_C <- cov(C) # Compute Covariance Matrix of Cluster C
> v2 <- c(4.5, 2.5) # Centre of Cluster E (given)
> Cov_E <- matrix(c(0.8, 0.27, 0.27, 0.11), 2, 2, byrow = TRUE) # Cov. Matrix of Cluster E (given)

> Cov_E
     [,1] [,2]
[1,] 0.80 0.27
[2,] 0.27 0.11

> x <- c(2,1) # Point x
>
> dxv1_sq <- mahalanobis(x, v1, Cov_C) # [Squared] Mahalanobis dist. from x to v1
> dxv1 <- sqrt(dxv1_sq) # Mahalanobis dist. from x to v1

> dxv1
[1] 2.828427

> dxv2_sq <- mahalanobis(x, v2, Cov_E) # [Squared] Mahalanobis dist. from x to v2
> dxv2 <- sqrt(dxv2_sq) # Mahalanobis dist. from x to v2

> dxv2
[1] 5.53436

> x <- c(2,1.5) # New point x
>
> dxv1_sq <- mahalanobis(x, v1, Cov_C) # [Squared] Mahalanobis dist. from x to v1
> dxv1 <- sqrt(dxv1_sq) # Mahalanobis dist. from x to v1

> dxv1
[1] 3.162278

> dxv2_sq <- mahalanobis(x, v2, Cov_E) # [Squared] Mahalanobis dist. from x to v2
> dxv2 <- sqrt(dxv2_sq) # Mahalanobis dist. from x to v2

> dxv2
[1] 3.017608
```

Using the Formula (with the aid of R and only for the second point):

```
> x <- c(2,1.5) # Point x
> x_dif_v1 = array(x-v1, dim=c(2,1)) # 2 x 1 column matrix (vector x - v1)
> dxv1 = sqrt(t(x_dif_v1)%*%solve(Cov_C)%*%x_dif_v1)
> dxv1
         [,1]
[1,] 3.162278

> x_dif_v2 = array(x-v2, dim=c(2,1)) # 2 x 1 column matrix (vector x - v2)
> dxv2 = sqrt(t(x_dif_v2)%*%solve(Cov_E)%*%x_dif_v2)
> dxv2
         [,1]
[1,] 3.017608
```

(h) Compute the Pearson and Spearman correlations between the following two (7-dimensional) numeric observations: **p** = [10 -3 0 4 1 0 3] and **q** = [1 1 4 -2 3 -1 4]. Do it both manually as well as in R.

Proposed Solution:

In R:

```
> p <- c(10, -3,  0,  4,  1,  0,  3)
> q <- c(10,  1,  4, -2,  3, -1,  4)
> cor(p, q, method = "pearson") # Same as cov(p,q)/(sd(p)*sd(q))
[1] 0.638777
> cor(p, q, method = "spearman")
[1] 0.3
```

Manually: Starting with Pearson, we can easily compute the mean values of **p** and **q** as mean_p = 2.142857 and mean_q = 2.714286, respectively. Similarly, we can compute their standard deviations as sd_p = 4.140393 and sd_q = 3.988077, respectively. We can then use the formula to compute Pearson as:

```
(1/6)*( (10 - 2.142857)*(10 - 2.714286) + (-3 - 2.142857)*(1 - 2.714286) + (0
- 2.142857)*(4 - 2.714286) + (4 - 2.142857)*(-2 - 2.714286) + (1 -
2.142857)*(3 - 2.714286) + (0 - 2.142857)*(-1 - 2.714286) + (3 - 2.142857)*(4
- 2.714286) ) / (4.140393*3.988077) = 0.6387
```

Spearman can be seen as Pearson on the rankings of the original vectors, i.e., rank(p) = [7 1 2.5 6 4 2.5 5] and rank(q) = [7 3 5.5 1 4 2 5.5], both with mean value of 4 and standard deviation of 2.140872. We can then compute the Spearman correlation of the original observations as:

```
(1/6)*( (7 - 4)*(7 - 4) + (1 - 4)*(3 - 4) + (2.5 - 4)*(5.5 - 4) + (6 - 4)*(1
- 4) + (4 - 4)*(4 - 4) + (2.5 - 4)*(2 - 4) + (5 - 4)*(5.5 - 4) ) /
(2.140872*2.140872) = 0.3
```

(i) Compute the Cosine similarity between the following records, each of which is described by 8 numeric variables: **p** = [1 0 0 4 1 0 0 3] and **q** = [0 5 0 2 3 1 0 4]. Do it in R but explicitly applying the formula, rather than using any built-in function.

Proposed Solution:

```
> # Computing Cosine Similarity in R:
> p <- c(1,  0,  0,  4,  1,  0,  0,  3)
> q <- c(0,  5,  0,  2,  3,  1,  0,  4)
> cos_pq <- (p%*%q)/sqrt((p%*%p)*(q%*%q))
>
> cos_pq
          [,1]
[1,] 0.5968492
>
> # Or Alternatively:
> cos_pq <- sum(p*q)/sqrt(sum(p^2)*sum(q^2))
>
> cos_pq
          [,1]
[1,] 0.5968492
```

Exercise 2-3 Proximity Measures for Categorical Data

(a) Consider the following dataset with 4 data objects described by 10 binary *categorical* variables each.

$$\mathbf{X} = \begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Compute the Simple Matching Coefficient (SMC) and the Jaccard coefficient for pairs of observations in the dataset above (manually and in R).

Proposed Solution:

Manually: Let's illustrate the calculations using the pair $\mathbf{x}_2$ and $\mathbf{x}_3$; clearly, we have $n_{01} = 4$, $n_{10} = 0$, $n_{00} = 4$, and $n_{11} = 2$, so SMC = (2+4) / (4+0+4+2) = 6/10 = 0.6, whereas Jaccard = 2 / (4+0+2) = 2/6 = 0.333.

In R:

```
> x2 <- c("0","0","0","0","0","0","1","0","0","1")
> x3 <- c("1","0","0","1","1","0","1","0","1","1")
> CT <- table(x2,x3) # Contingency table
> SSC <- (CT[1,1]+CT[2,2])/sum(CT)
> SSC
[1] 0.6
> Jaccard <- CT[2,2]/(CT[2,2]+CT[1,2]+CT[2,1])
> Jaccard
[1] 0.3333333
```

(b) Discuss what conditions are required for SMC to reach its extreme values (0 and 1). Repeat for Jaccard.

Proposed Solution:

For SMC = $(n_{00} + n_{11}) / (n_{01} + n_{10} + n_{00} + n_{11})$ n_{01} to become zero (0), we necessarily need to have a complete absence of any agreement, i.e., $n_{00} = n_{11} = 0$, which necessarily implies $n_{10} + n_{01} = n$ (complete disagreement across all variables), where n is the total number of variables. For SMC to become one (1), we necessarily need to have a complete absence of any disagreement, i.e., $n_{01} = n_{10} = 0$, which necessarily implies $n_{00} + n_{11} = n$ (complete agreement across all variables). For Jaccard = $n_{11} / (n_{01} + n_{10} + n_{11})$ to become zero (0), we need to have an absence of any agreements other than 0-0 matches, i.e., we need $n_{11} = 0$. For Jaccard to become one (1), as for SMC, we again necessarily need to have a complete absence of any disagreement, i.e., $n_{01} = n_{10} = 0$.

(c) Suppose that observations are students, and each binary categorical variable indicates that a students is ("1") or not ("0") enrolled in a subject (course). There are n=20 variables (courses). You want to assess similarity between students based on the courses that they choose. Should you use SMC or Jaccard?

Proposed Solution:

Since we want to assess similarity between students based on the courses that they take, rather than on all the courses that they do not take, we should use Jaccard and not SMC, because the former will ignore all 0-0 matches (courses that neither student chooses), whereas the latter will consider those as indicative of a similarity as much as the 1-1 matches (courses that both students choose).

(d) Repeat item (c) above now assuming that the 20 variables are questions of a questionnaire that the students answer as TRUE ("1") or FALSE ("0"), and we want to assess similarity of students based on their answers to the same questions.

Proposed Solution:

Based on the question statement, there is no reason to believe that FALSE or TRUE answers should be treated differently. Therefore, we should use SMC instead of Jaccard.

(e) Suppose we computed the *similarity* between two observations described by categorical variables and obtained SMC = 6/10 and Jaccard = 1/3. Convert these proximity values from similarity to *dissimilarity*.

Proposed Solution:

Since both SMC and Jaccard range within the [0,1] interval, it is trivial to convert them into dissimilarity by

taking their complement, i.e., SMC_dis = 1 – SMC = 1 – 6/10 = 4/10 and Jaccard_dis = 1 – Jaccard = 1 – 1/3 = 2/3.

Exercise 2-4 Categorical to Numeric Conversion

(a) Consider a variable Size that takes on categorical ordinal values {Very Small, Small, Medium, Large, Very Large, Extra Large}, where we know that their semantic respect their intended relative orders. Without any further information about the application domain or context, propose a simple yet suitable conversion scheme to represent this ordinal variable as a numeric variable instead.

Proposed Solution:

Since the variable is ordinal, we want to keep information about the relative order, but without further information the simplest scheme is one with equally spaced numerical values, e.g., Size $\in$ {0, 0.2, 0.4, 0.6, 0.8, 1}, where 0 represents "Very Small", 0.2 represents "Small", and so forth.

(b) Now consider a nominal categorical value Political_Affiliation that can take on values {Party A, Party B, Party C}. Show how this variable can be replaced by a numerical representation using a one-hot encoding scheme.

Proposed Solution:

We can replace Political_Affiliation with three (mutually exclusive) binary variables, Party A, Party B, and Party C, each of which will be 1 if and only if Political_Affiliation takes on the corresponding categorical value for an observation, 0 otherwise. In other words, Political_Affiliation = Party A is replaced with [Party A, Party B, Party C] = [1 0 0], Political_Affiliation = Party B is replaced with [Party A, Party B, Party C] = [0 1 0], and Political_Affiliation = Party C is replaced with [Party A, Party B, Party C] = [0 0 1].