Bioinformatics III

Eighth Assignment

Max Jakob (2549155)
Carolin Mayer (2552320)

June 19, 2018

Exercise 8.1: Data Preprocessing

This task has not been implemented.

Exercise 8.2: Correlation Measures

The implementation of the desired statistical measurements can be found in Listing 1.

Listing 1: Source code of the script `correlation.py`

```
from itertools import combinations
import math

def rank(x):
    """
    :param x: a list of values
    :return: ranking of the input list
    """
    xs = sorted(x)[::-1]
    x_indices = [0] * len(xs)
    for j in range(len(x)):
        indices = [a for a,b  in enumerate(xs) if b == x[j]]
        x_indices[j] = float(sum(indices))/len(indices)
    return x_indices

def pearson_correlation(x, y):
    """
    :param x: a list of values
    :param y: a list of values
    :return: Pearson correlation coefficient of X and Y
    """
    x_ = float(sum(x))/len(x)
    y_ = float(sum(y))/len(y)
    sum_num = 0
    sum_denom_x = 0
    sum_denom_y = 0
    for i in range(len(x)):
        sum_num += (x[i] - x_)* (y[i] - y_)
        sum_denom_x += (x[i] - x_)**2
        sum_denom_y += (y[i] - y_)**2

    return float(sum_num)/(math.sqrt(sum_denom_x) * math.sqrt(sum_denom_y))


def spearman_correlation(x, y):
    """
    :param x: a list of values
    :param y: a list of values
    :return: Spearman correlation coefficient of X and Y
    """
    return pearson_correlation(rank(x), rank(y))


def kendall_correlation(x, y):
    """
    :param x: a list of values
    :param y: a list of values
    :return: Kendall-B correlation coefficient of X and Y
    """
    xr = rank(x)
    yr = rank(y)
    nc = 0
    nd = 0
    ny = 0
    nx = 0
    pairs = zip(xr,yr)
    for i in range(len(pairs)):
        for j in range(i+1, len(pairs)):
            if (pairs[i][1] > pairs[j][1] and pairs[i][0] > pairs[j][0]) or (pairs[i][1] < pairs[j][
                nc += 1
            if (pairs[i][1] > pairs[j][1] and pairs[i][0] < pairs[j][0]) or (pairs[i][1] > pairs[j][
                nd += 1
            if pairs[i][1] == pairs[j][1] and pairs[i][0] != pairs[j][0]:
                ny += 1
            if pairs[i][0] == pairs[j][0] and pairs[i][1] != pairs[j][1]:
                nx += 1

    return float(nc - nd) / math.sqrt((nc+nd+nx)*(nc+nd+ny))

class CorrelationMatrix(dict):
    """
    This class behaves like a dictionary, where the correlation between two elements 1 and 2 is acc
    cor_matrix[(element_1, element_2)] or cor_matrix[(element_2, element_1)] since the matrix is sy
    It also stores the row (or column) names of the input DataMatrix.
    """
    def __init__(self, data_matrix, method, rows):
        """
        :param data_matrix: a DataMatrix (see data_matrix.py)
        :param method: string specifying the correlation method, must be 'Pearson', 'Spearman' or '
        :param rows: True if the correlation matrix should be constructed for the rows, False if fo
        """
        # initialise the dictionary
        super().__init__(self)

        # if rows = True, then compute the correlation matrix for the row data
        if rows:
            data = data_matrix.get_rows()
        # if rows = False, then compute the correlation matrix for the column data
        else:
            data = data_matrix.get_columns()

        # sorted list of row names (or column names) in the input data matrix
        self.names = list(sorted(data.keys()))

        # compute the correlation between all pairs of rows (or columns)
        for name_1, name_2 in combinations(data.keys(), 2):
            # use the specified correlation method
            if method == 'Pearson':
                correlation = pearson_correlation(data[name_1], data[name_2])
            elif method == 'Spearman':
                correlation = spearman_correlation(data[name_1], data[name_2])
            elif method == 'Kendall':
                correlation = kendall_correlation(data[name_1], data[name_2])
            else:
                raise ValueError('The␣correlation␣method␣not␣supported␣must␣be␣either␣Pearson,␣Spear

            # add the correlation symmetrically
        self[(name_1, name_2)] = correlation
        self[(name_2, name_1)] = correlation
```

Exercise 8.3: Gene Co–Expression Networks

This task has not been implemented.

Exercise 8.4: Hierarchical Clustering

This task has not been implemented.