Clustering

JG|U
JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Manuel Mauch, Marius Köppel

DM 17/18

12. 12. 2017

JG|U

Inhaltsverzeichnis

Agglomerative and Divisive Clustering

The goal of this task was to create three dentograms for each dataset with R and the provided algorithmus Agnes and Diana. First we will present this algorithmus and then we will discuss the dentograms of each dataset.

Agnes

- Blabal
- Blabal

Diana

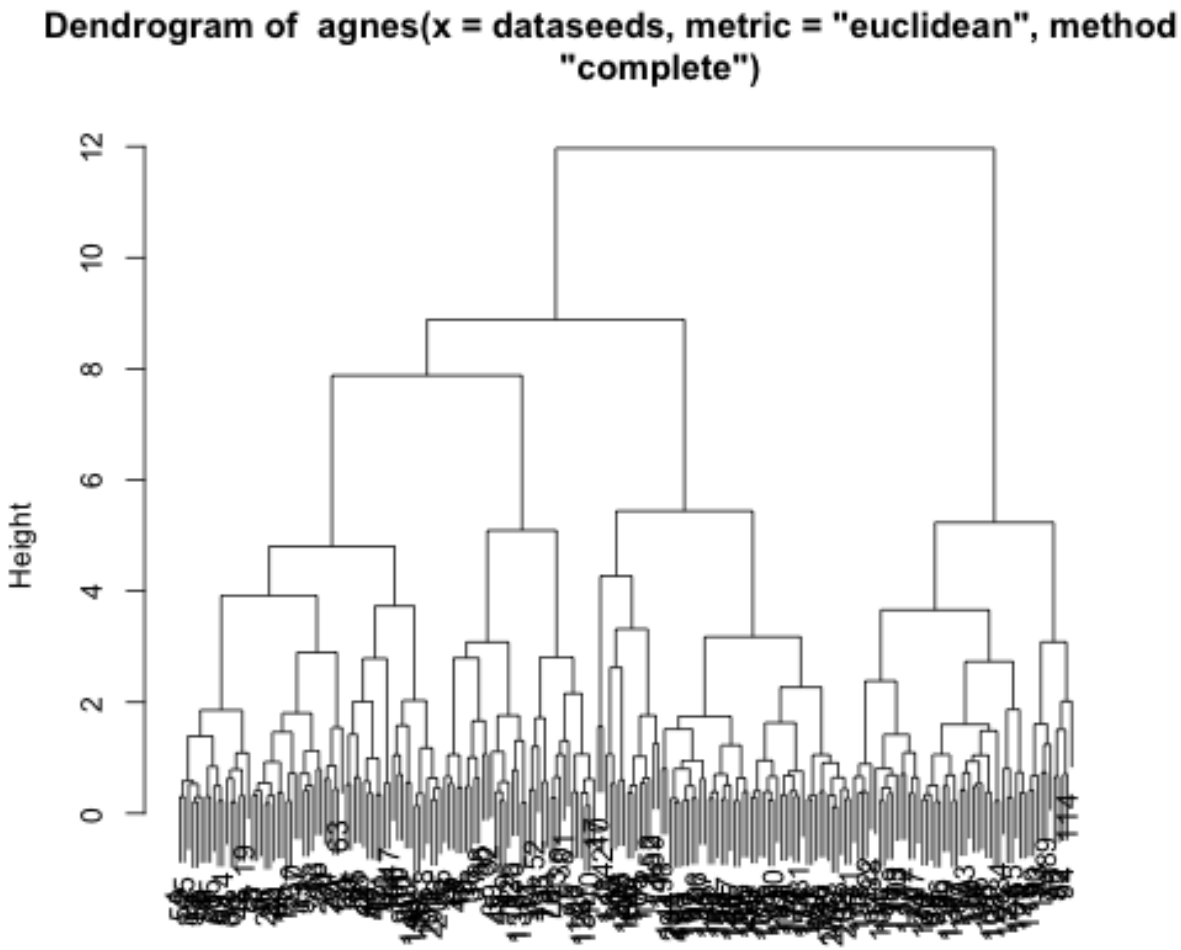
- Blabal
- Blabal

S2 from S-sets

For this dataset we were not able to create to plots, because it took to much time for proceed.

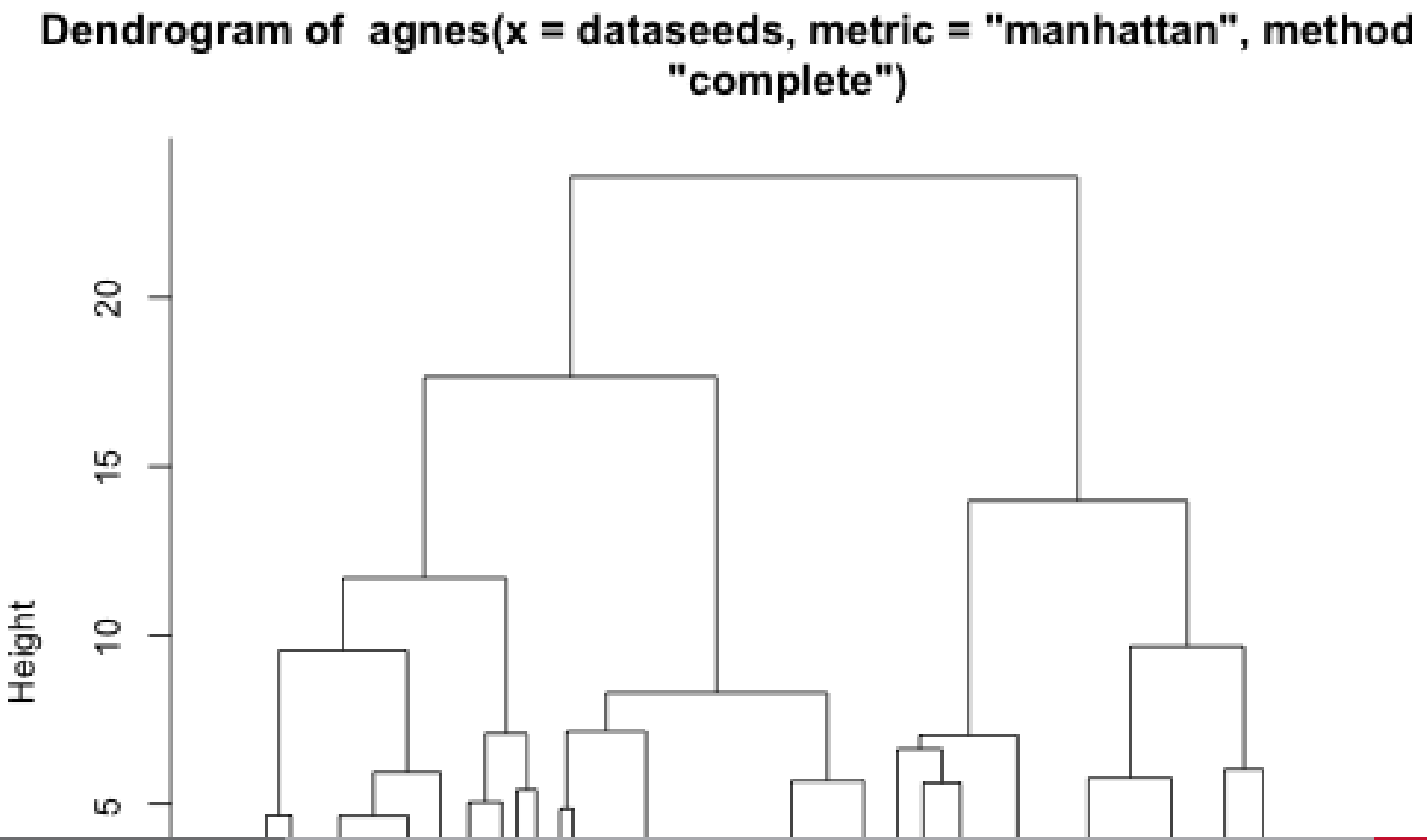
Seeds

Detaset Seeds



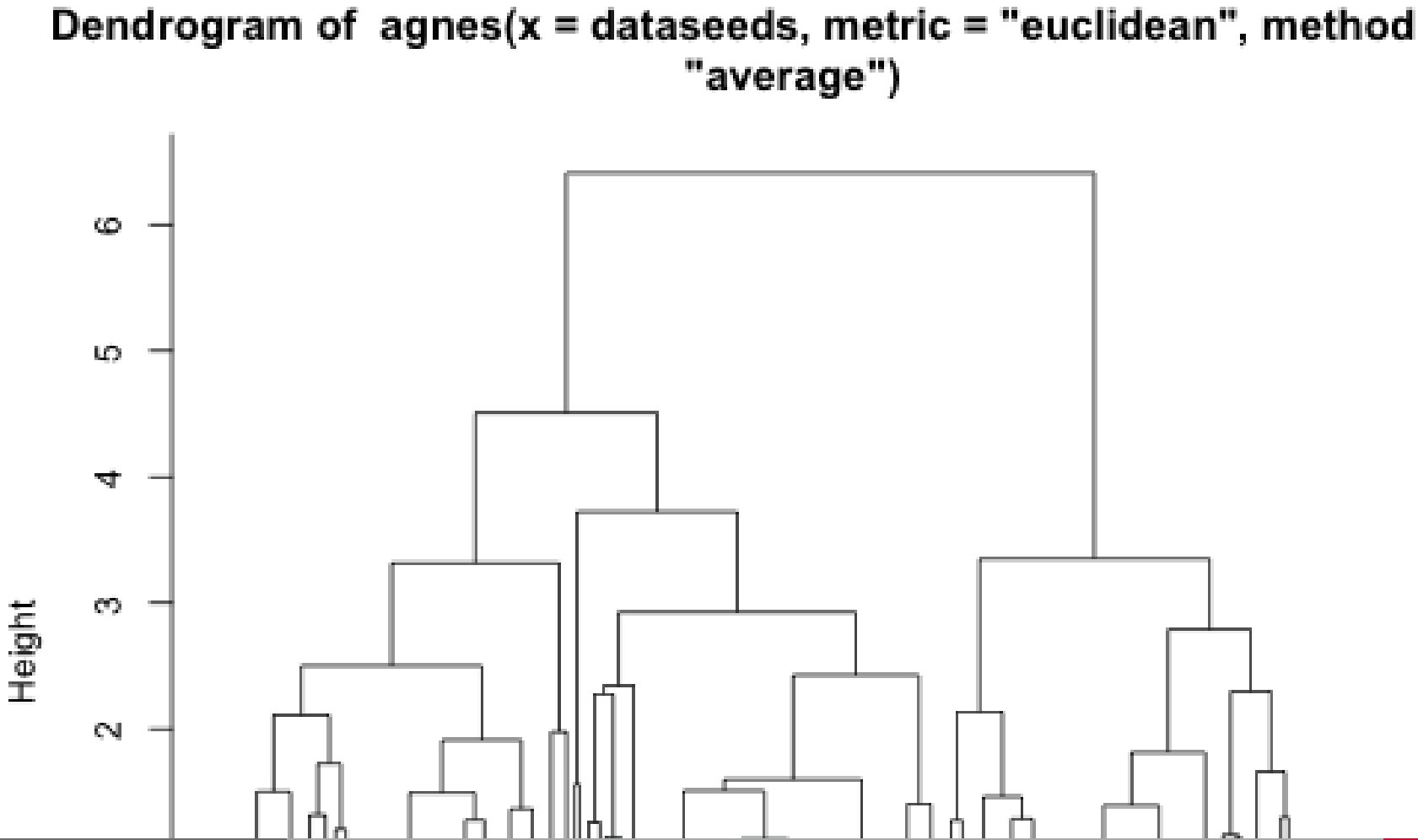
Seeds

Detaset Seeds



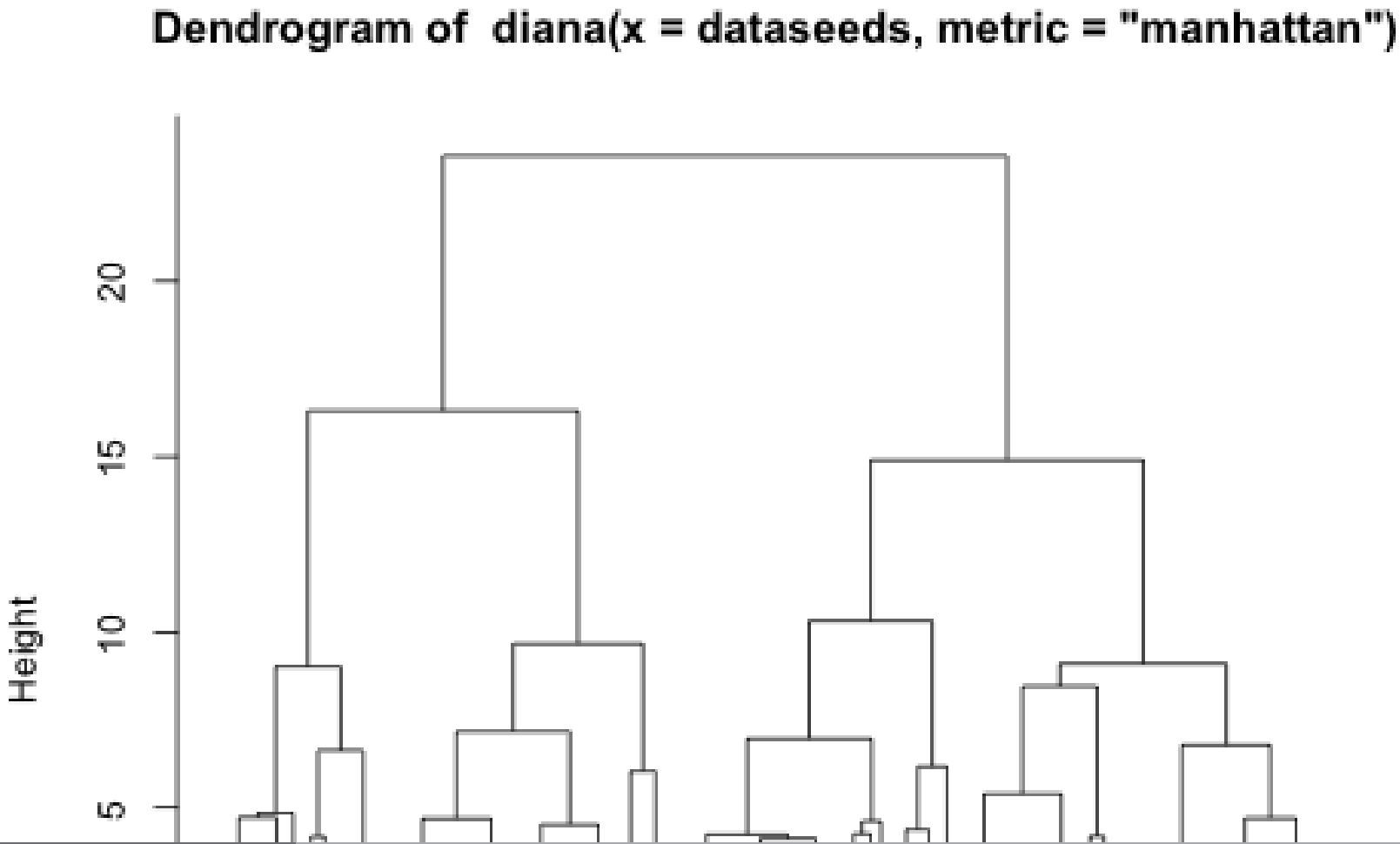
Seeds

Detaset Seeds



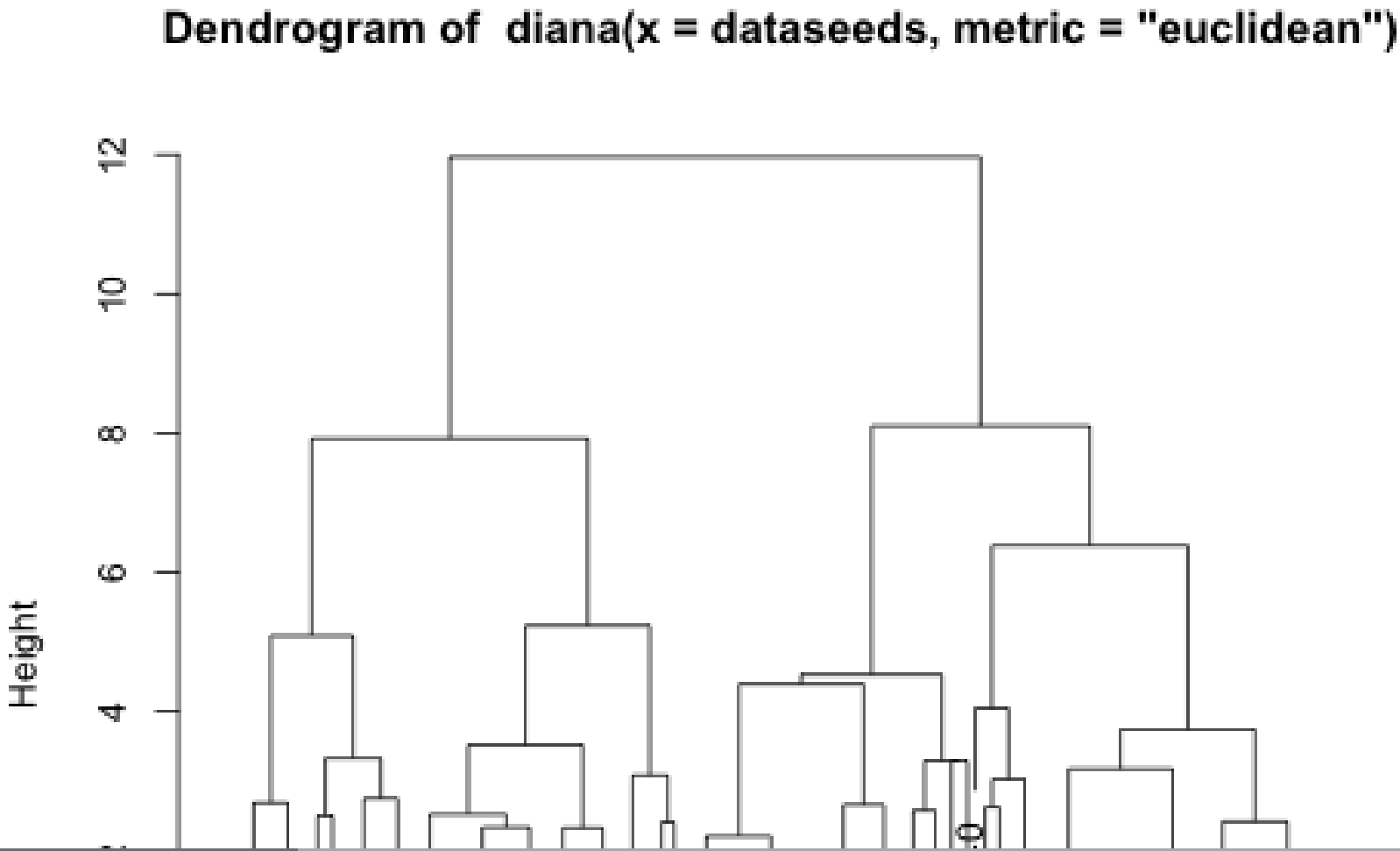
Seeds

Detaset Seeds



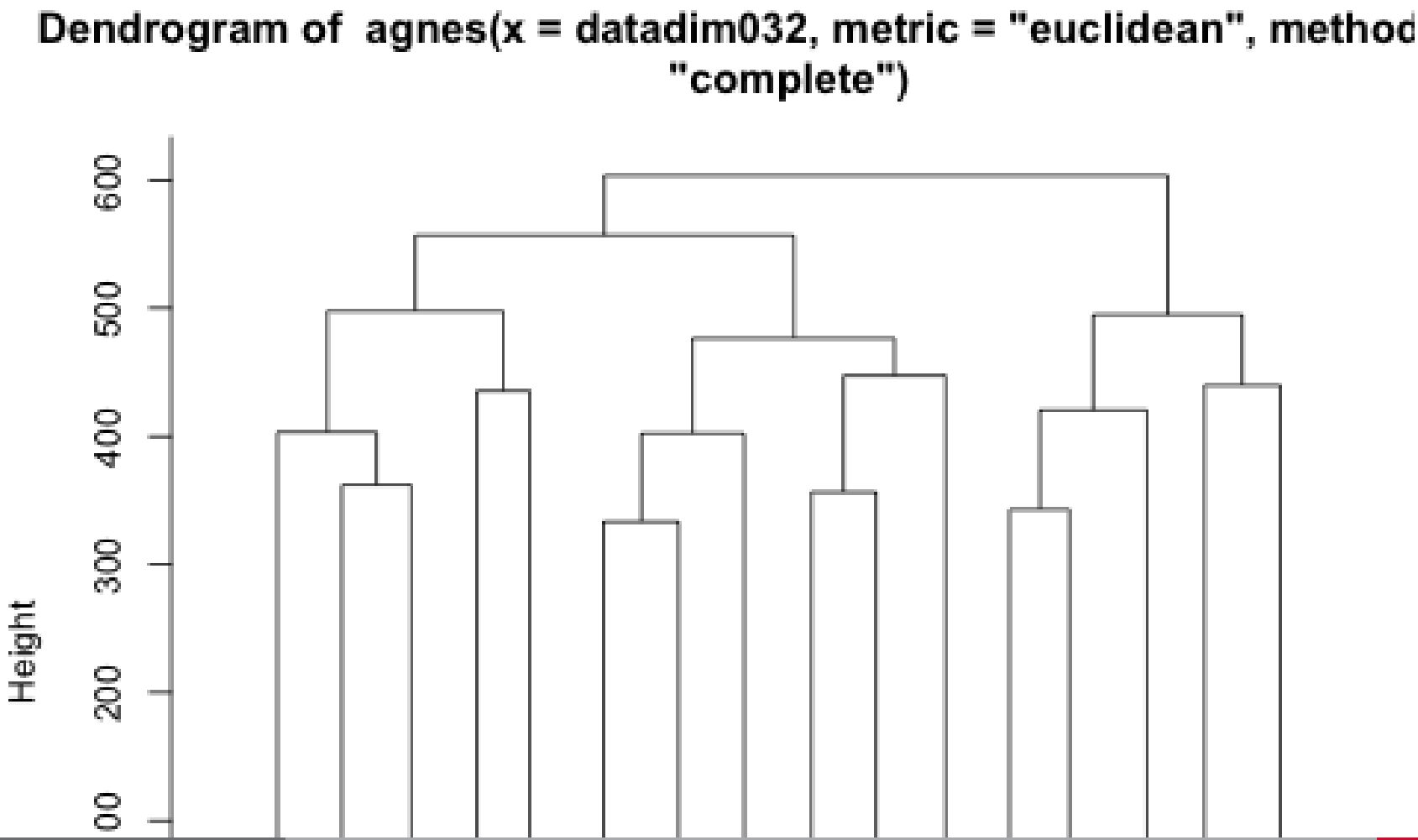
Seeds

Detaset Seeds



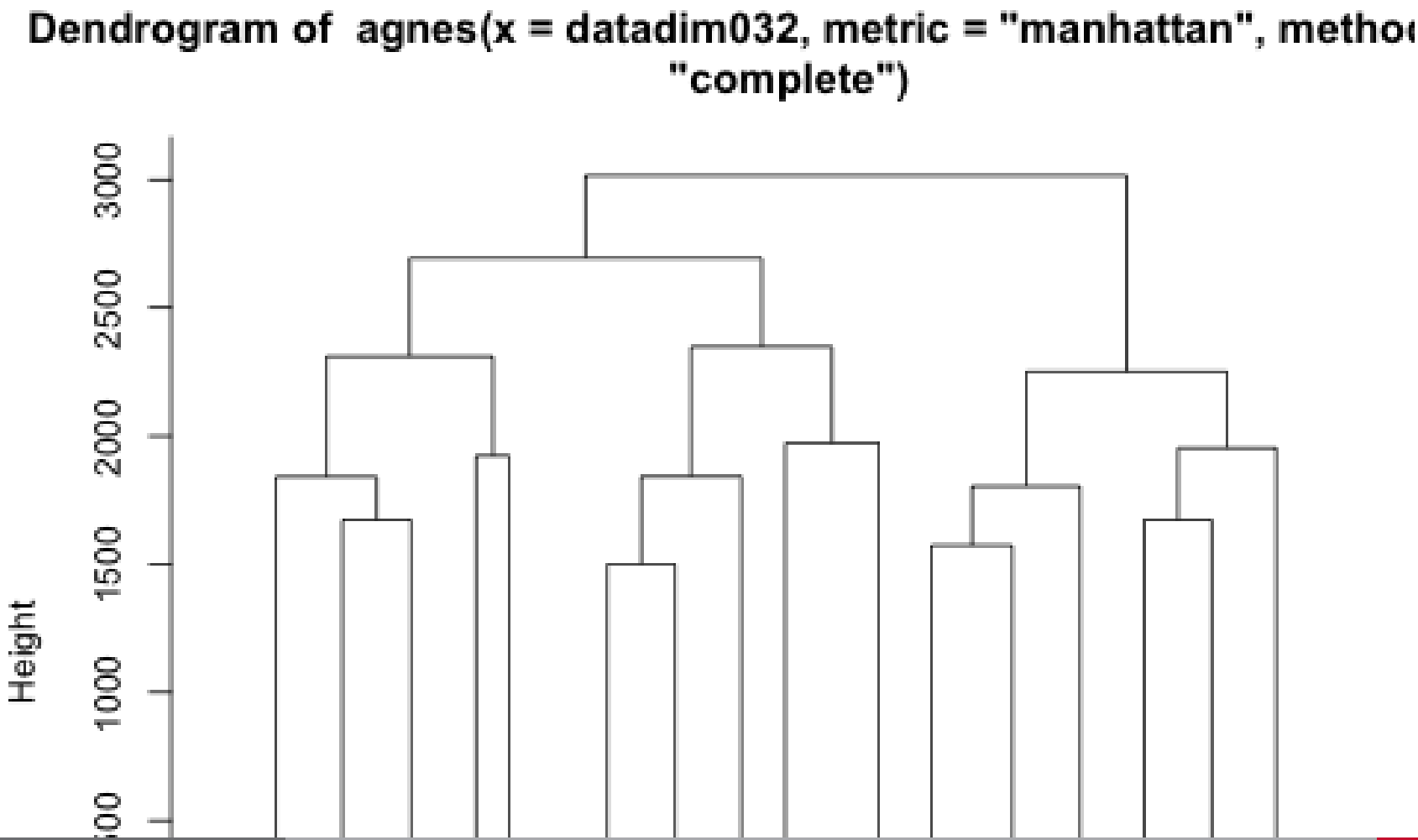
Dim032 from DIM-sets (high)

Detaset Dim032



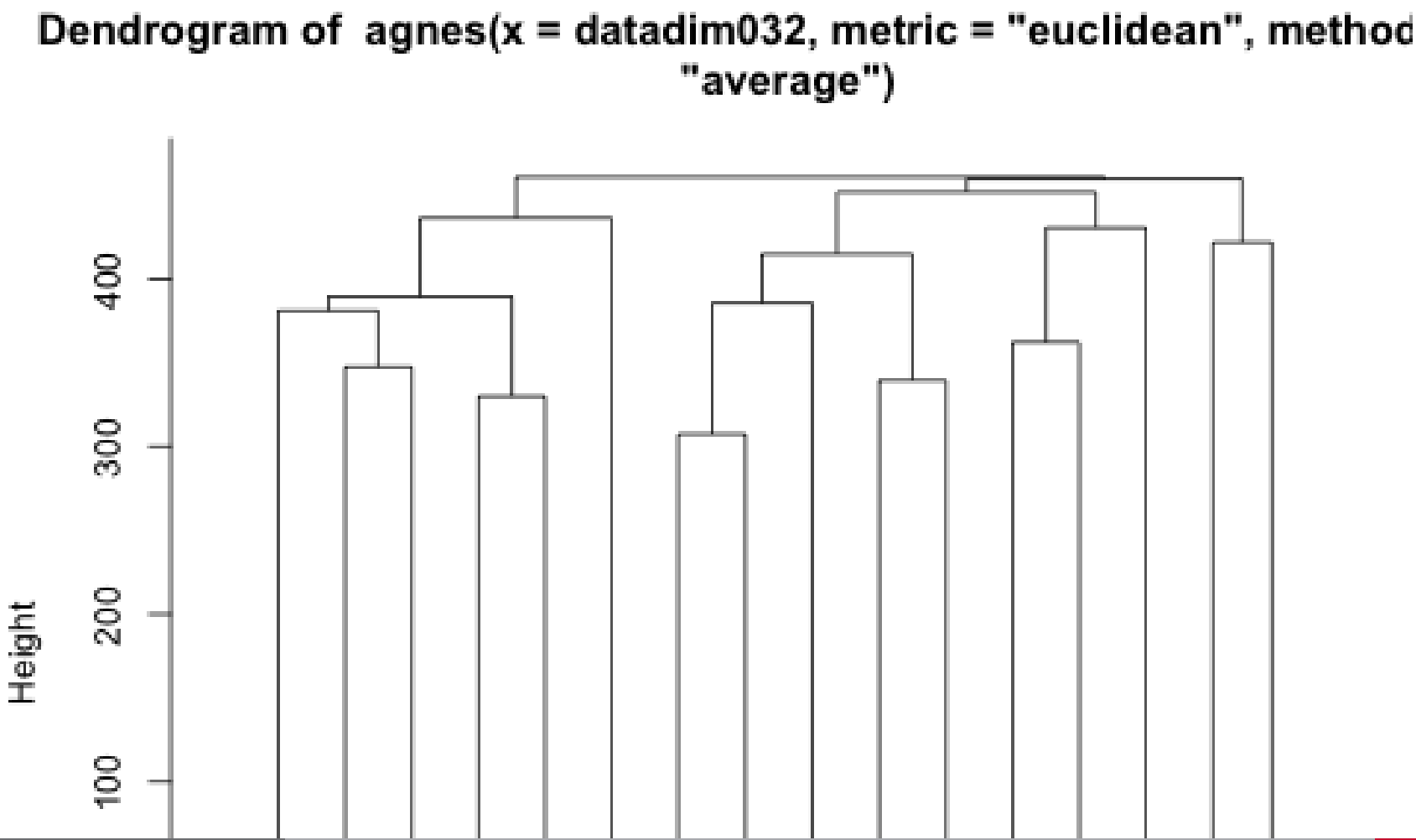
Dim032 from DIM-sets (high)

Detaset Dim032



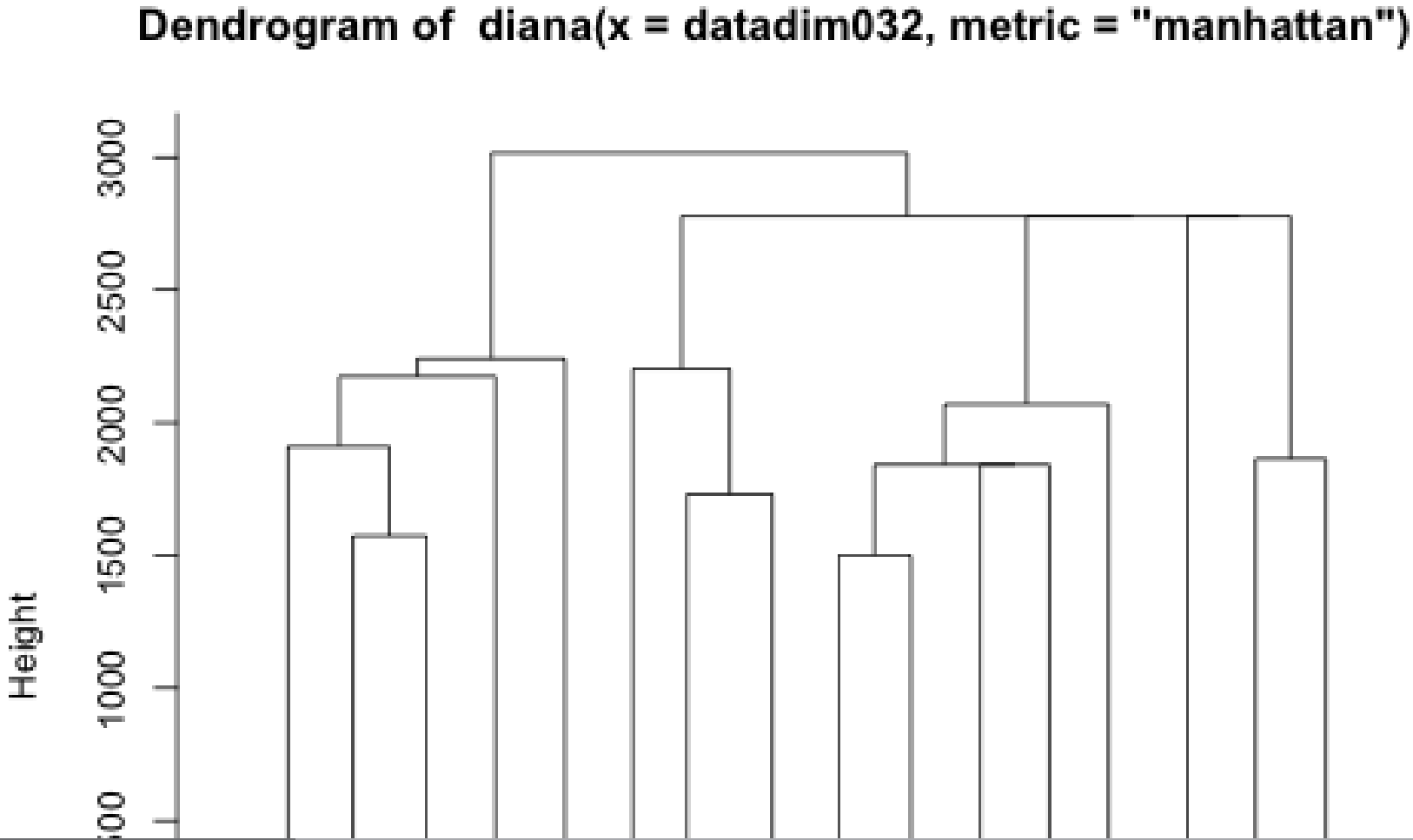
Dim032 from DIM-sets (high)

Detaset Dim032



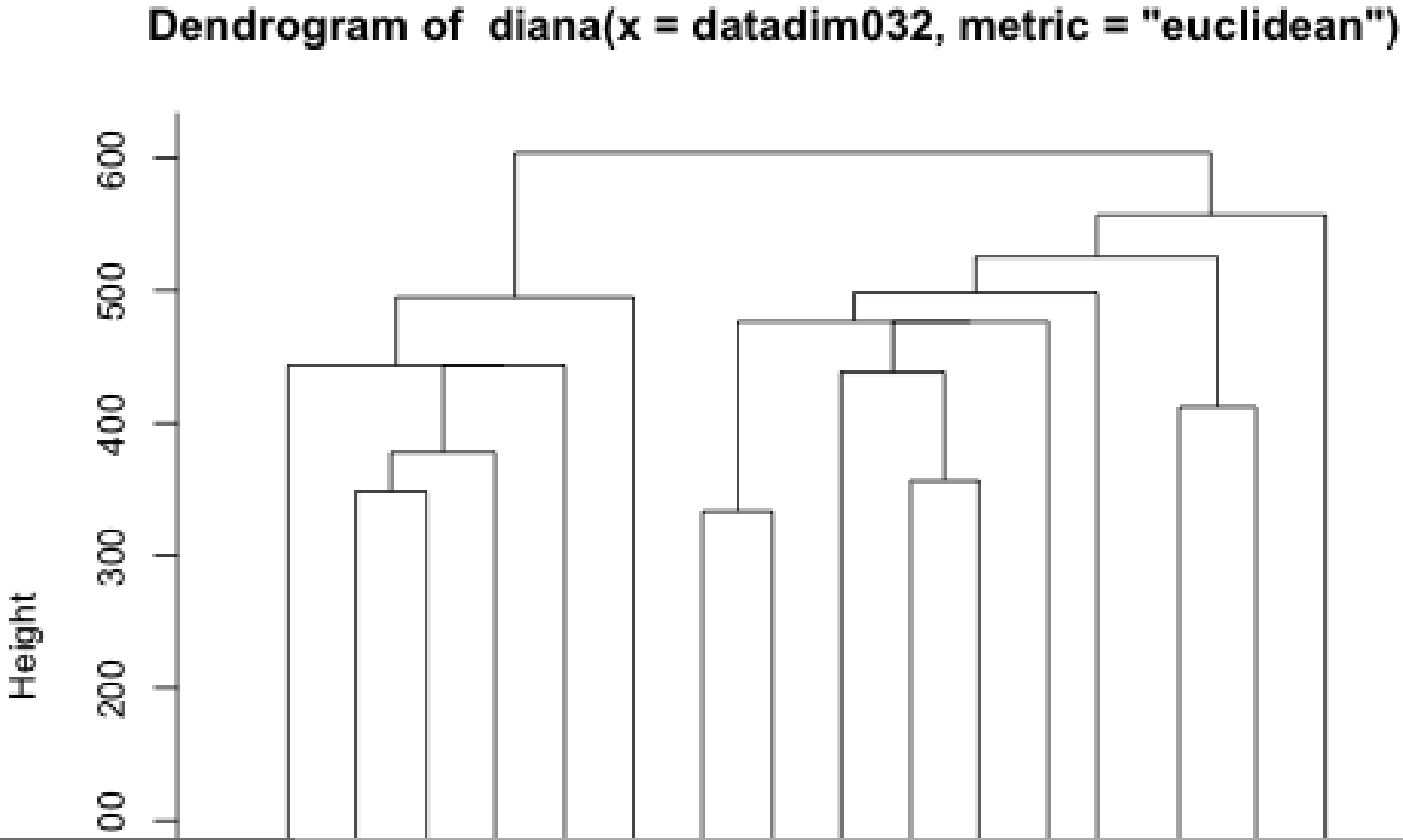
Dim032 from DIM-sets (high)

Detaset Dim032



Dim032 from DIM-sets (high)

Detaset Dim032



Conclusion

TODO (MANUEL): Was ist hight in dentogram?, Wie gehen die Algorithmen und Interpretation der unterschiedlichen Teile.

K-means

```
import sys
import random
from math import sqrt
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import accuracy_score
import numpy as np

# Idea from http://www.caner.io/purity-in-python.html
def purityScore(clusters, classes):
    """
    Calculate the purity score for the given cluster
        assignments and ground truth classes

    :param clusters: the cluster assignments array
    :type clusters: numpy.array

    :param classes: the ground truth classes
```

K-means

Data	RAND	normalized mutual information	purity of clus
jain.txt	2	6	1
compound.txt	3	3	1

Set S2 had no lable so it was not possible to calculate the best k

K-means

ADD EXAMPLE PLOTS

K-means

ADD PROBLEM WITH LINEAR CLUSTERING ADD IMAGE FROM ML U9

EM

```
import numpy as np
import math

def read(path):
    data = []
    lable = []
    with open(path) as csv:
        for line in csv:
            data.append(float(line.split("\n")[0]))
            lable.append(0)
    return lable, data

def logLikelihood(data, k, parameters):
    logLikeli = 0
    for x in data:
        logLikeli += np.log(parameters["p"] *
            gaus(parameters["sig1"], parameters["mu1"], x)
                        + parameters["p"] *
```