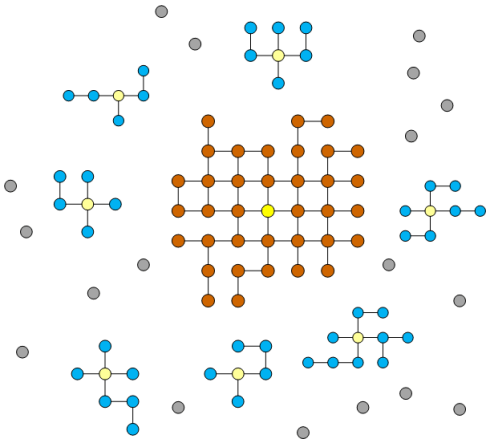
Podatki. Podatki so podani v obliki tabele, ki je vsebovala države, njihove pesmi in glasovanja na finalnem delu Evrovizije v razponu 11 let. Tabela je vsebovala nekaj atributov, ki niso relevantni in so bili zato odstranjeni (ime pesmi, izvajalec in drugo). Uporabljene so bile ocene ostalih držav za določeno pesem neke druge države. Število držav, torej atributov, je 47, število primerov pa je 291. Zaloga vrednosti teh ocen je med 0 in 12, povprečna ocena pa je 2,5. Vsako leto se niso vse države uvrstile v finalni del, zato manjka 34% podatkov. Za posamezno državo dobimo profil glasovanja kar kot posamezen atribut države. Ta vsebuje vse primere kako je država glasovala za ostale. Če izračunamo razdaljo med temi profili, dobimo blizu skupaj države, ki so podobno glasovale.

Računanje razdalj. Razdalje med profili so bile izračunane z Evklidsko razdaljo, ki računa razlike med glasovanji na podlagi posameznih primerov znotraj dveh profilov. Na podlagi teh razdalj je bila ustvarjena prva gruča. Za računanje razdale med njimi je bila uporabljena povprečna Evklidska razdalja med gručami. Torej se je izračunala razdalja med vsemi elementi obeh gruč in se nato delila s številom povezav.

V tabeli je primanjkovalo kar 34% podatkov. Če ne bi bili umetno dodani, bi izgubili velik del podatkov. Zato je bilo za manjkajoče primere uporabljeno povprečje glasovanj iz ostalih let. Torej če v določenem letu neka država ni glasovala za drugo drržavo, se je ta glas nadomestil s povprečjem iz glasovanj ostalih let za isto državo.



Slika 1: Vsako sliko opremi s podnapisom, ki pove, kaj slika prikazuje.

Dendogram V to poglavje lahko tudi vključiš kakšen metodološko zanimiv del kode. Primer vključitve kode oziroma implementirane funkcije v programskem jeziku Python je:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
```

```
    else:
        return fib(n-1) + fib(n-2)
```

Izris te kode je lahko sicer tudi lepši, poskušaš lahko najti še primernejši način vključevanja kode v Pythonu oziroma v tvojem izbranem programskem jeziku v okolje LaTeX.

Skupine in njihove preferenčne izbire. V tem poglavju podaš rezultate s kratkim (eno-odstavčnim) komentarjem. Rezultate lahko prikažeš tudi v tabeli (primer je tabela 1).

Odstavke pri pisanju poročila v LaTeX-u ločiš tako, da pred novim odstavkom pustiš prazno vrstico. Tudi, če pišeš poročilo v kakšnem drugem urejevalniku, morajo odstavki biti vidno ločeni. To narediš z zamikanjem ali pa z dodatnim presledkom.

Tabela 1: Atributi in njihove zaloge vrednosti.

ime spremenljivke	definicijsko območje	opis
cena	[0, 500]	cena izdelka v EUR
teža	[1, 1000]	teža izdelka v dag
kakovost	[slaba—srednja—dobra]	kakovost izdelka

Podajanje rezultati naj bo primerno strukturirano. Če ima naloga več podnalog, uporabi podpoglavja. Če bi želel poročati o rezultatih izčrpno in pri tem uporabiti vrsto tabel ali grafov, razmisli o varianti, kjer v tem poglavju prikažeš in komentiraš samo glavne rezultate, kakšne manj zanimive detajle pa vključite v prilogo (glej prilogi A in B).

Izjava o izdelavi domače naloge. Domačo nalogo in pripadajoče programe sem izdelal sam.

Priloge

Podrobni rezultati poskusov. Če je rezultatov v smislu tabel ali pa grafov v nalogi mnogo, predstavi v osnovnem besedilu samo glavne, podroben prikaz rezultatov pa lahko predstaviš v prilogi. V glavnem besedilu ne pozabi navesti, da so podrobni rezultati podani v prilogi.

Programska koda. Za domače naloge bo tipično potrebno kaj sprogramirati. Če ne bo od vas zahtevano, da kodo oddate posebej, to vključite v prilogo. Čisto za okus sem tu postavil nekaj kode, ki uporablja Orange (`http://www.biolab.si/orange`) in razvrščanje v skupine.

```
import random
import Orange

data_names = ["iris", "housing", "vehicle"]
data_sets = [Orange.data.Table(name) for name in data_names]

print "%10s␣%3s␣%3s␣%3s" % ("", "Rnd", "Div", "HC")
for data, name in zip(data_sets, data_names):
    random.seed(42)
```

```
    km_random = Orange.clustering.kmeans.Clustering(data, centroids = 3)
    km_diversity = Orange.clustering.kmeans.Clustering(data, centroids = 3,
        initialization=Orange.clustering.kmeans.init_diversity)
    km_hc = Orange.clustering.kmeans.Clustering(data, centroids = 3,
        initialization=Orange.clustering.kmeans.init_hclustering(n=100))
    print "%10s %3d %3d %3d" % (name, km_random.iteration, \
    km_diversity.iteration, km_hc.iteration)
```