IST[A/G/T]1003: Statistisk læring og data science

Klyngeanalyse

Mette Langaas IMF/NTNU

19 October, 2020

Contents

Læringsmål

Etter at du har lest dette kompendiet, sett de tre videoene som er laget og deltatt på zoom-forelesningen skal du

- kunne forstå hvorfor det er interessant å gjøre klyngeanalyse
- kjenne igjen situasjoner der klyngeanalyse vil være en aktuell metode å bruke
- kjenne begrepene avstandsmål, koblingstype, dendrogram
- forstå algoritmen for å utføre hierarkisk klyngeanalyse og K-gjennomsnitt-klyngeanalyse
- kunne utføre klyngeanalyse i Python
- kunne svare godt på klyngeanalyseoppgaven i den tellende prosjektoppgaven

Introduksjon

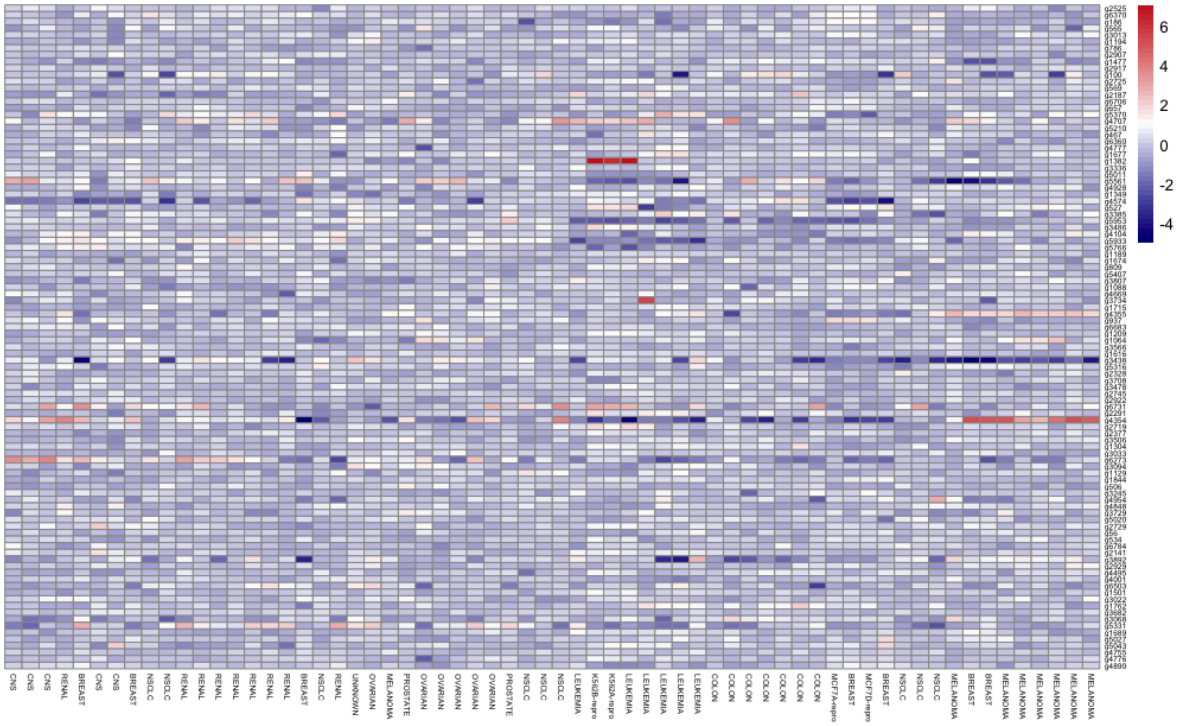
Videoer som diskuterer innholdet av dette kompendiet:

- Klyngeanalyse (8:43 min)
- Hierarkisk klyngeanalyse (11:26 min)
- K-gjennomsnitt-klyngeanalyse (8:38 min)

Eksempel: genaktivitet

Datasett: Vi skal se på et datasett som består av celleprøver fra kreftsvulster fra ulike pasienter. Det er totalt 64 celleprøver (disse er navngitt i datasettet som breast, renal, melanoma, colon, ...), og mange prøver er fra samme type vev. For hver celleprøve har man målt hvor aktivt hvert av 6830 gener er. At et gen er aktivt betyr at det produserer et stoff som heter mRNA - det er det som er målt her - og mRNA kan videre omdannes til protein - og det er proteiner som er arbeidshestene i cellene våre.

Figuren viser celleprøvene langs x-aksen og 100 tilfeldig valgte gener langs y-aksen (det blir litt mye informasjon å se på hvis hvis vi plotter alle 6830 genene). Et piksel i bildet viser for en gitt celleprøve aktiviteten til et gitt gen, og er denne pikselen blå er aktiviteten til genet lavt og er den rød er aktiviteten til genet høyt.



En av grunnene til at dette datasettet ble samlet inn var at man ønsket å finne hvilke celleprøver som hadde lik aktivitetsprofil for genene. Det kan være flere celleprøver som ligner på hverandre. Kan vi dermed lage grupper (kalt klynger) av celleprøver som er lik hverandre? Hvis vi gjør det kan det kanskje bidra til at vi forstår hvilke celletyper som har samme underliggende biologiske mekanismer, og lære mer om hvordan man finne behandling for sykdom.

Hva er klyngeanalyse?

Vi har: en datamatrise X med n rader (observasjoner) og p kolonner (variabler), men vi har ingen respons Y (slik som vi hadde i veiledet (supervised) læring med regresjon og klassifikasjon). I eksemplet med genaktivitet var observasjonene de $n = 64$ celleprøvene og variablene de $p = 6830$ genene som vi måte aktiviteten til.

Mål: Målet med klyngeanalyse er å finne ukjente klynger i data. Kan vi finne ut noe interessant om dataene? Spesielt ønsker vi å lage klyngene slik at observasjonene innen hver klynge er mer lik hverandre enn observasjonene fra ulike klynger. Finner vi slike klynger forstår vi kanskje mer av disse dataene?

Plass i verden: Klyngeanalyse er en metode innen såkalt "ikke-veiledet" læring (unsupervised). En annen viktig metode innen ikke-veiledet læring er dimensjonsreduksjon. Der er målet å visualisere data av høy dimensjon (oftest høy p) i to dimensjoner via transformasjoner (populære metoder er prinsipalkomponenetanalyse og t-SNE).

Avstandsmål

Når vi skal lage klynger av observasjoner, må vi ha et mål på hvor like to observasjoner er. Til det bruker vi et avstandsmål.

Notasjon: La x_{ij} være målingen for observasjon i av variabel j , og x_{kj} være målingen for observasjon k for variabel j.

Som et eksempel kan vi tenke på at i og k kan være to personer, og $j = 1$ kan være måling av vekt, $j = 2$ kan være måling av høyde, $j = 3$ kan være måling av (systolisk) blodtrykk, $j = 4$ kan være måling av oksygenopptak, og så videre. Vi har totalt p variabler vi har målt.

Egenskaper vi ønsker når vi ser på avstand

Formelt kan vi si at D er en avstandsfunksjon hvis den kan ta ikke-negative reelle verdier, og vi kan kalle det et avstandsmål hvis vi for tre observasjoner x, y og z har at

- $D(x, y) = 0$ hvis $x = y$ (identitet)
- $D(x, y) + D(y, z) \geq D(x, z)$ (trekantulikhet)
- $D(x, y) = D(y, x)$ (symmetri)

Euklidsk avstand

Et populært avstandsmål for klyngeanalyse er euklidsk avstand. Den euklidske avstanden mellom observasjon i og observasjon k er

$$D_E(i, k) = \sqrt{\sum_{j=1}^{p}(x_{ij} - x_{kj})^2}$$

Både euklidsk avstand og kvadrert euklidsk avstand D_E^2 er brukt som avstandsmål innen klyngeanalyse.

Hvis $p = 2$ eller $p = 3$ kan vi se på observasjonene som punkter henholdsvis i planet eller rommet. Euklidsk avstand er da en vanlige avstanden mellom punkter - som vi er vant med. Vi kan tenke på euklidsk avstand som lengden av den korteste veien mellom to punkter i rommet.

City Block-avstand

Et annet avstandsmål heter Manhattan eller City Block - og har fått navnet sitt fra å tenke på hvordan man beveger seg i en by med kvartaler - da kan man ikke gå korteste veien gjennom en kvartal med bygninger men må gå rundt.

$$D_M(i,k) = \sum_{j=1}^{p} |x_{ij} - x_{kj}|$$

Hvis $p = 2$, slik at vi kan se på de to observasjonene som punkter i planet, blir dette forskjell i x-koordinat pluss forskjell i y-koordinat - derav navnet på avstandsmålet.

Hvis vi har binære variabler (variabler som bare tar to verdier, mest vanlig å kode dette som 0 og 1), kalles denne avstanden også Hammingavstand og brukes innen informasjonsteori.

Korrelasjonsavstand

Vi har tidligere i emnet (kapittel 4) snakket om korrelasjon, og empirisk korrelasjon (kapittel 7). Da var vi interessert i korrelasjonen mellom to variabler. Nå er vi heller interessert i korrelasjonen mellom to observasjoner.

Husk at korrelasjonen går fra -1 til 1, og måler lineær sammenheng. En høy positiv korrelasjon betyr at to observasjoner er (lineært) lik hverandre (høye verdier opptrer sammen) og en høy negativ korrelasjon (her betyr høy et stort tall i absoluttverdi) viser en nær lineær sammenheng der høye verdier for for en observasjon sees sammen med lave verdier for en annen observasjon.

La $\bar{x}_i$ være gjennomsnittet over alle variabler målt for observasjon i, altså $\bar{x}_i = \frac{1}{p}\sum_{j=1}^{p} x_{ij}$, og tilsvarende for observasjon k, $\bar{x}_k = \frac{1}{p}\sum_{j=1}^{p} x_{kj}$

Hvis vi ikke skal ha fokus på likhet, men heller på avstand, er det naturlig å la avstandsmålet defineres som 1 minus korrelasjonen:

$$D_R(i,k) = 1 - \frac{\sum_{j=1}^{p}(x_{ij} - \bar{x}_i)(x_{kj} - \bar{x}_k)}{\sqrt{\sum_{j=1}^{p}(x_{ij} - \bar{x}_i)^2}\sqrt{\sum_{j=1}^{p}(x_{kj} - \bar{x}_k)^2}}$$

Hva er verdiområdet for korrelasjonsavstanden? Korrelasjonsavstanden kan være mellom 0 (hvis korrelasjonen er 1) og 2 (hvis korrelasjonen er -1), og en korrelasjon på 0 gir en avstand på 1.

Det finnes varianter av korrelasjonsavstanden. Grunnet koding av variabler så kan det være at en korrelasjon på -1 betyr at to observasjoner er like, og da brukes heller en minus kvadratet av korrelasjonskoeffisienten.

Korrelasjonsavstanden måler ikke direkte likheten mellom to observasjoner, men heller hvordan høye og lave verdier (i forhold til observasjonens gjennomsnittsverdi) opptrer sammen. I eksemplet med genuttrykk er dette det mest populærer avstandsmålet - fordi da vil absoluttverdien til aktiviteten til et gen ikke ha så mye å si, men hvordan høye og laver verdier for flere gener opptrer sammen. Det er også vanlig å sentrere og standardisere data for genuttrykk for hvert gen i et preprosesseringssteg.

Det finnes veldig mange andre avstandsmål, men vi vil fokusere på disse tre. Valg av avstandsmål er koblet til hvilken type data vi har, og hva vi regner som interessante forskjeller mellom to observasjoner.

For to personer har vi målt vekt, høyde og (systolisk) blodtrykk. Vi vil se hvor like de to personene er hvis vi både tar hensyn til vekt, høyde og blodtrykk ved å se på euklidsk og city block-avstand.

- La $x_{11} = 60$ være person 1 sin vekt, $x_{12} = 170$ person 1 sin høyde, $x_{13} = 130$ person 1 sitt blodtrykk
- La $x_{21} = 80$ være person 2 sin vekt, $x_{22} = 175$ person 2 sin høyde, $x_{23} = 120$ person 2 sitt blodtrykk

Euklidsk avstand: $\sqrt{(60-80)^2+(170-175)^2+(130-120)^2} = 22.9$

City block-avstand: $|60-80|+|170-175|+|130-120| = 35$

Korrelasjonsavstand: Her trenger vi først å regne ut at $\bar{x}_1 = (60+170+130)/3 = 120$ og $\bar{x}_2 = (80+175+120)/3 = 125$, og deretter for nevneren at det to faktorene blir $\sqrt{\sum_{j=1}^{p}(x_{1j}-\bar{x}_1)^2} = \sqrt{(60-120)^2+(170-120)^2+(130-120)^2} = 78.74$ og $\sqrt{\sum_{j=1}^{p}(x_{2j}-\bar{x}_2)^2} =$

$\sqrt{(80-125)^2+(175-125)^2+(120-125)^2} = 67.45$. Da kan vi regne avstanden: $1-\frac{(60-120)\cdot(80-125)+(170-120)\cdot(175-125)+(130-}{78.74\cdot 67.45}$ $1-\frac{5150}{5311.01} = 1-0.97 = 0.03$.

Legg merke til at avstandene er på veldig ulike skalaer.

Vi kan også regne ut avstander i Python, for eksempel med funksjoner i modulen *spatial* i pakken *scipy*. Men, for metodene vi skal se på videre er dette "innebygget" i metoden.

Metoder for klyngeanalyse

Vi skal i dette emnet studere to metoder for klyngeanalyse:

- Hierarkisk klyngeanalyse: vi skal se på det som heter agglomerativ klyngeanalyse, som går ut på at vi for alle par av observasjoner finner avstanden mellom de to observasjonene. Deretter lager vi en klynge av de to som har lavest verdi av avstandsmålet. og erstatter de to observasjonene med denne klyngen. Vi gjentar denne operasjonen, og fortsetter slik til alle observasjonene er i samme klynge. En fin ting med denne typen klyngeanalyse er at vi underveis kan presentere resultatene i en slags trestruktur (et såkalt dendrogram), og trenger ikke på forhånd spesifisere hvor mange klynger vi ser etter i dataene.
- K-gjennomsnitt klyngeanalyse: Her bestemmer vi antall klynger K på forhånd. Algoritmen går ut på å fordele de n observasjonene tilfeldig i K klynger, regne ut et midtpunkt (sentroide) i klyngen og så flytte observasjoner til den klyngen de ligger nærmest. Deretter beregnes nye midtpunkt og observasjoner flyttes - og dette gjentas til det ikke trengs flere omorganiseringer av observasjonene.

Vi skal nå se på disse to metodene i mer detalj.

Hierarkisk klyngeanalyse

Vi skal kun se på det som heter agglomerativ (eller bottom-up) hierarkisk klyngeanalyse. Det er den mest populære versjonen av hierarkisk klyngeanalyse.

Vi har tidligere sagt at målet med klyngeanalyse er å finne ukjente klynger i data, og at vi ønsker å lage klyngene slik at observasjonene innen hver klynge er mer lik hverandre enn observasjonene fra ulike klynger.

For å kunne utføre en hierarkisk klyngeanalyse må vi først lære to begreper:

- avstandsmatrise
- kobling mellom klynger (linkage)

Avstandsmatrise

I hierarkisk klyngeanalyse trenger vi en avstandsfunksjon eller et avstandsmål, for eksempel euklidsk avstand, city block-avstand eller korrelasjonsavstand.

Anta at vi har n observasjoner. Vi kan da regne ut avstanden mellom alle par av observasjoner. Det blir $\binom{n}{2} = \frac{n\cdot(n-1)}{2}$ mulige par av observasjoner. Hvis $n = 5$ blir dette tallet 10 og hvis $n = 6830$ blir det 23 millioner.

Disse avstandene putter vi inn i en $n \times n$ *avstandsmatrise*.

- På diagonalen setter vi inn tallet 0 fordi dette er avstanden fra en observasjon til seg selv.
- I posisjon $(1, 2)$ i matrisen har vi avstanden mellom observasjon 1 og 2.
- Siden avstandsmålet vårt er symmetrisk blir avstanden mellom observasjon 1 og 2 lik avstanden mellom observasjon 2 og 1 - og matrisen vår blir også symmetrisk.
- Avstanden mellom observasjon i og k finner vi da i posisjon (i, k) og (k, i) i matrisen.

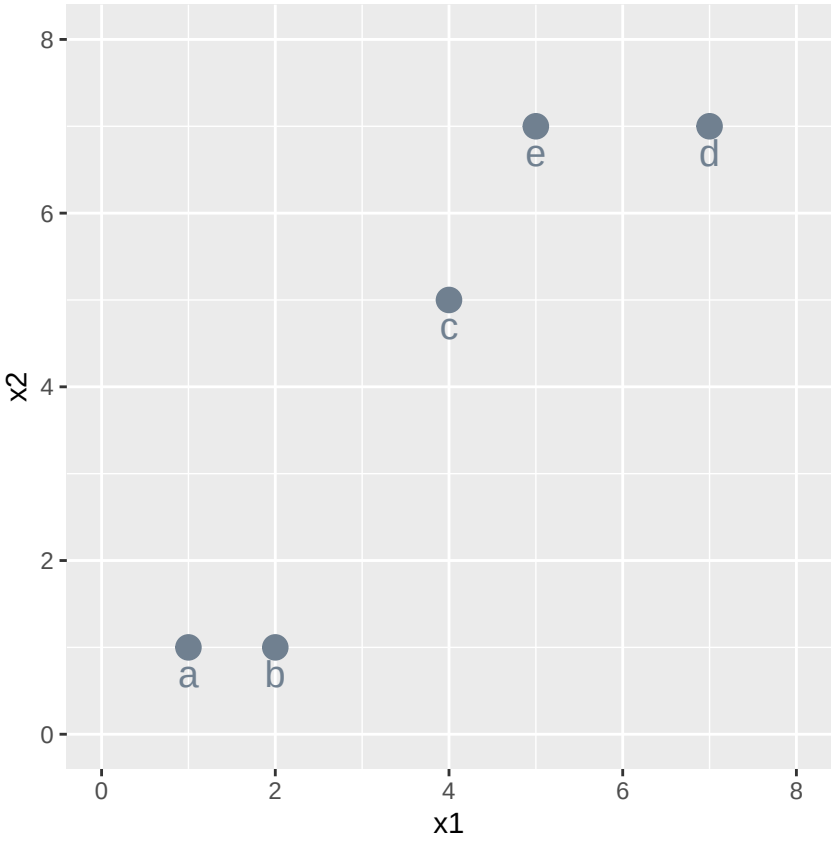
Merk: hvis n er stor vil det kreve mye datakraft og lagringsplass å lage og ta vare på denne matrisen. Derfor er ikke hierarkisk klyngeanalyse egnet for veldig store datasett.

Illustrasjonseksempel

Vi har et datasett med $n = 5$ observasjoner med $p = 2$ variabler, og vil lage en avstandsmatrise basert på euklidsk avstand.

Observasjoner

```
##   x1 x2
## a  1  1
## b  2  1
## c  4  5
## d  7  7
## e  5  7
```



Avstandsmatrise

```
##     a   b   c   d   e
## a 0.0 1.0 5.0 8.5 7.2
## b 1.0 0.0 4.5 7.8 6.7
## c 5.0 4.5 0.0 3.6 2.2
## d 8.5 7.8 3.6 0.0 2.0
```

```
## e 7.2 6.7 2.2 2.0 0.0
```

Kobling (linkage)

Hvis vi har to observasjoner vet vi hvordan vi skal regne avstanden mellom dem (med vårt valgte avstandsmål). Men, hvordan skal vi regne avstanden mellom to klynger av observasjoner? Jo, vi bruker ulike typer *kobling* (engelsk: linkage) for å generalisere avstand mellom observasjoner til avstand mellom klynger.

Anta at vi har to klynger med observasjoner, kall dem klynge 1 og klynge 2. I illustrasjonen består klynge 1 av observasjon a og b, og klynge 2 av observasjon c, d og e.

Det finnes mange ulike såkalte koblingstyper, og noen av de mest populære er som følger (navnet i hakeparentes bruker vi når vi gjør analysene i Python). Dette kan illustreres grafisk hvis vi har observasjoner av $p = 2$ variabler.

- Singel kobling (minimal avstand)[single]: Beregn alle parvise avstander mellom observasjonene i klynge 1 til observasjonene i klynge 2, og registrer den *minste* av disse avstandene.
- Komplett kobling (maksimal avstand)[complete]: Beregn alle parvise avstander mellom observasjonene i klynge 1 til observasjonene i klynge 2, og registrer den *største* av disse avstandene.
- Gjennomsnittskobling (gjennomsnittlig avstand)[average]: Beregn alle parvise avstander mellom observasjonene i klynge 1 til observasjonene i klynge 2, og registrer *gjennomsnittet* av disse avstandene.
- Sentroide-kobling [centroid]: Beregn sentroiden til klynge 1 (gjennomsnittsvektor av dimensjon p) og sentroiden til klynge 2. Se på de to sentroidene som to observasjoner og regn så avstanden mellom dem.

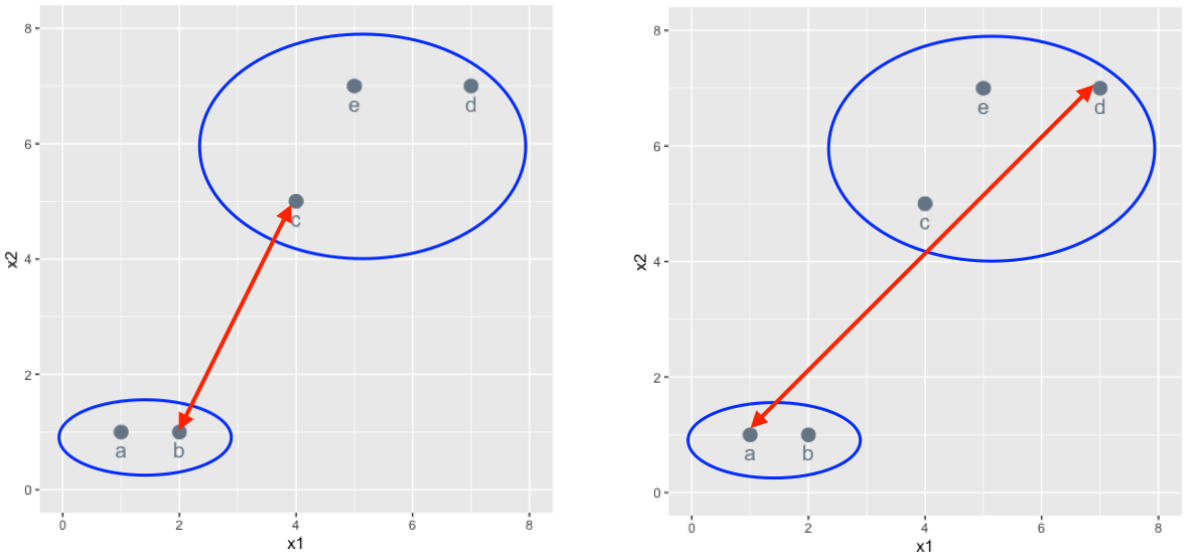


Figure 1: Singel (venstre) og komplett (høyre) kobling

Algoritme for hierarkisk klyngeanalyse

Før algoritmen starter må man bestemme seg for

- hvilket avstandsmål skal brukes (f.eks: euklidsk, city block, korrelasjon, ...)
- hvilken koblingstype skal brukes (f.eks: singel, komplett, gjennomsnitt, sentroide, ...)

Og så regner man ut avstandsmatrisen for alle n observasjoner.

Først, behandles hver observasjon som om den er sin egen klynge - så nå er det n klynger til å begynne med.

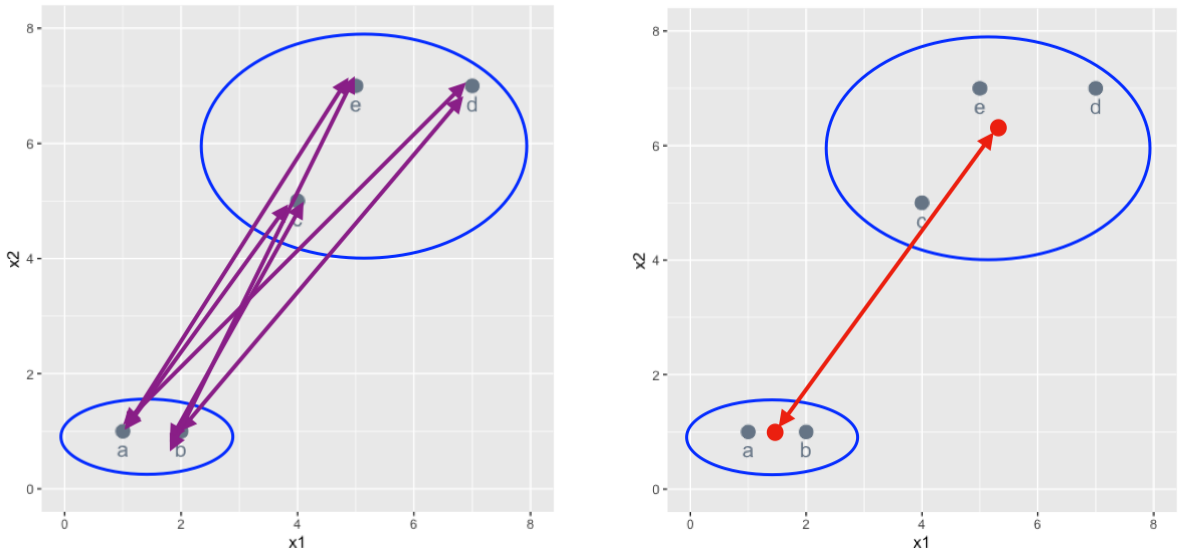


Figure 2: Gjennomsnitt (venstre) og sentroide (høyre) kobling

Så starter hovedløkken i algoritmen – som vi repeterer til det alle observasjonene er i samme klynge. For $i = n, n-1, ..., 2$ utføres dette:

1. **Slå sammen:** Det er nå i klynger. For hver klynge regn ut avstanden til alle andre klynger. Finn de to klyngene som er nærmest hverandre og slå dem sammen til en klynge.
2. **Beregne avstander:** Beregn nye parvise avstander mellom alle klynger. Det er nå $i-1$ klynger, så regn ut $\frac{(i-1)\cdot(i-2)}{2}$ avstander (ved bruk av valgt avstandsmål og linkage-type).

Illustrasjon av algoritmen

Se videoen Hierarkisk klyngeanalyse for illustrasjon av algoritmen med euklidsk avstand og komplett kobling.

Dendrogram

(I videoen over vil du naturlig få introdusert begrepet dendrogram koblet til algoritmen til hierarkisk klyngeanalyse.)

Vi vil forklare et dendrogram ved å se på hvordan et dendrogram ser ut for illustrasjonseksemplet når euklidsk avstand og komplett kobling er bruk. Dette er dendrogrammet:

Et dendrogram er et slags opp-ned tre, der observasjonene sees om løvnoder (a-e i figuren) og gir en slag horisontal akse. Rekkefølgen til observasjonene har med hvilken rekkefølge vi har laget klyngene på, men vi kan ikke bruke det til å si hvilke observasjoner som er nærmest hverandre. På den vertikale aksen ser vi ved hvilken verdi av kombinasjonen av avstandsmål og kobling, at klynger er slått sammen. Vi bruker den vertikale aksen til å se hvor like observasjonene er.

Vi kan lese følgende ut av dendrogrammet:

1) Vi startet med å slå sammen observasjon a og b, som har avstand 1, til en klynge.

De vertikale strekene leder opp fra hver observasjon, og møtes av en horisontal strek med y-koordinat lik avstanden mellom observasjonene som slås sammen til en klynge (her var avstanden 1)

2) Det andre som skjer er at vi slår sammen observasjon d og e, som har avstand 2,til en klynge.

Den horisontale streken over klynge d og e har y-koordinat 2. Dette var nå den minste avstanden.

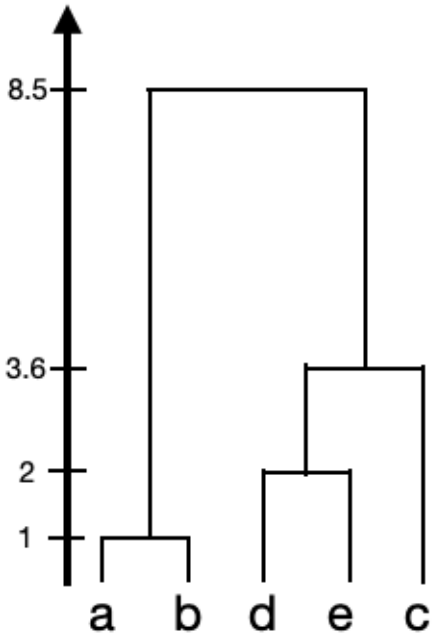


Figure 3: Dendrogram

3) Det tredje som skjer er at vi slår sammen observasjon c og klyngen til d og e, som har avstand 3.6.

Den horisontale streken over c, d og e har y-koordinat 3.6. Dette var nå den minste avstanden.

4) Det fjerde og siste som skjer i klyngeanalyssen er at klyngen med a og b slås sammen med klyngen med c, d og e Avstanden mellom disse er 8.5.

Hvor mange klynger skal vi velge?

Vi kan bruke dendrogrammet til å bestemme hvor mange klynger vi skal velge.

Noen ganger har vi en klar mening om hvor mange klynger vi leter etter i data, og hvis vi har det kan vi bare gå inn i dendrogrammet på en avstand der det er akkurat så mange klynger – og tilordne klyngemedlemskap basert på dette.

- I avstand mellom 3.6 og 8.5 (vertikal akse), er det 2 vertikal linjer – som svarer til 2 klynger – og hvilke observasjoner som er med i klyngene leser vi av ved å følge de vertikale linjene ned til løvnodene – så da har vi ab i den ene klyngen og cde i den andre.
- I en lavere avstand, mellom 2 og 3.6, er det tre vertikale linjer som svarer til tre klynger, av observasjon ab, c og de.
- I avstand mellom 1 og 2 er det fire vertikale linjer, altså fire klynger, det er ab, c, d og e.
- I avstand som er lavere enn en som definerte den første klyngen som ble dannet (lavere enn 1) har vi bare enkeltobservasjonene – så da har vi de 5 opprinnelige observasjonene som klynger.

På denne måten kan vi bruke dendrogrammet til å bestemme hvor mange klynger vi skal velge og samtidig tilordne klyngemerkelapper til observasjonene.

Hierarkisk klyngeanalyse i Python

Her er hvordan vi kan utføre klyngeanalyse med illustrasjonseksemplet vårt. I eksemplet over hadde vi brukt euklidsk avstand og komplett kobling. Under har jeg valgt singel kobling. Avstandene i dendrogrammet blir ikke de samme med singel og komplett kobling.

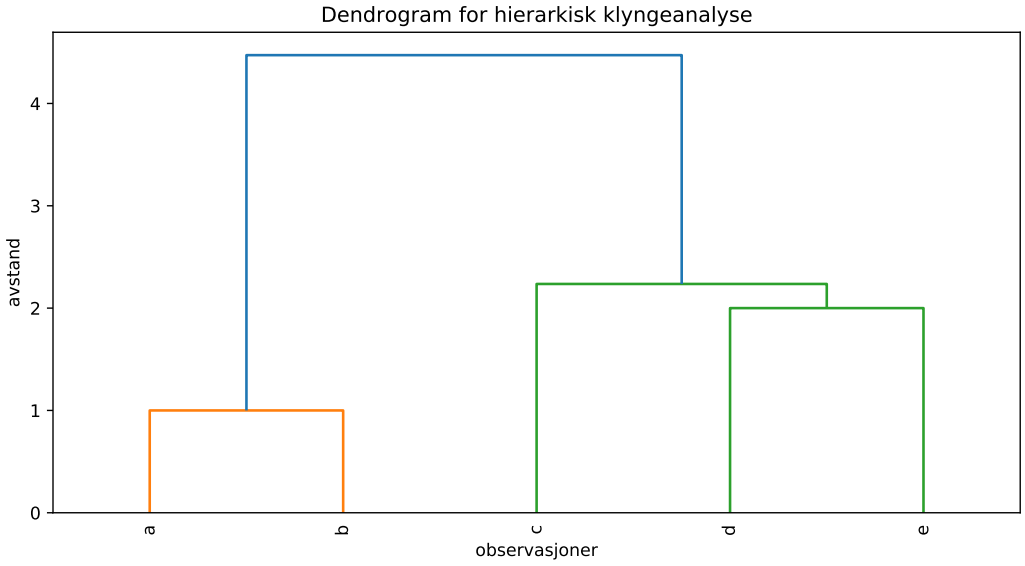
```
import pandas as pd
import numpy as np
import string
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
# datasettet
column_names = ['x1', 'x2']
row_names    = list(string.ascii_lowercase)[0:5]
matrix = np.reshape((1,1,2,1,4,5,7,7,5,7),(5,2))
df = pd.DataFrame(matrix, columns=column_names, index=row_names)

# utføre selve analysen med data, kobling (method) og avstandsmål (metric)
Z = linkage(y=df, method='single',metric='euclidean')

# lage dendrogram
plt.figure(figsize=(10, 5))
plt.title('Dendrogram for hierarkisk klyngeanalyse')
plt.xlabel('observasjoner')
plt.ylabel('avstand')
dendrogram(Z,labels=row_names,
    leaf_rotation=90.,  # roterer navn på observasjoner på x-aksen
    leaf_font_size=10.,  # setter fontstørrelse på teksten på x-aksen
)
```

```
## {'icoord': [[5.0, 5.0, 15.0, 15.0], [35.0, 35.0, 45.0, 45.0], [25.0, 25.0, 40.0, 40.0], [10.0, 10.0,
```

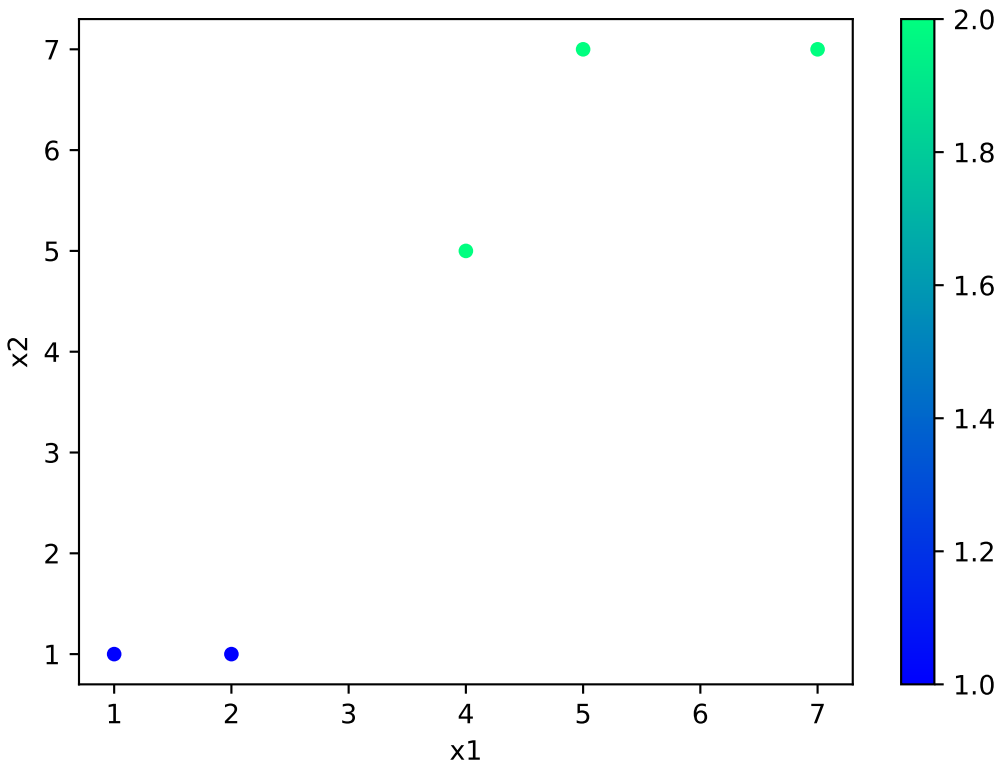
```
plt.show()
```




```
klyngelab=fcluster(Z, t=2, criterion='maxclust')
print(klyngelab)
```

```
## [1 1 2 2 2]
```

```
df.plot(kind='scatter',x='x1',y='x2',c=klyngelab,cmap='winter')
```



Medisinsk eksempel

Vi skal tilbake til datasettet med celleprøver fra ulike pasienter, der $n = 64$ og $p = 6830$, og på nytt se på figuren med celleprøvene langs x-aksen og 100 tilfeldig valgte gener langs y-aksen. Et piksel i bildet viser for en gitt celleprøve aktiviteten til et gitt gen, og er denne pikselen blå er aktiviteten til genet lavt og er den rød er aktiviteten til genet høyt.

Følgende er resultat av hierarkisk klyngeanalyse basert på de $p = 6830$, med gjennomsnitt-kobling og euklidsk avstand.

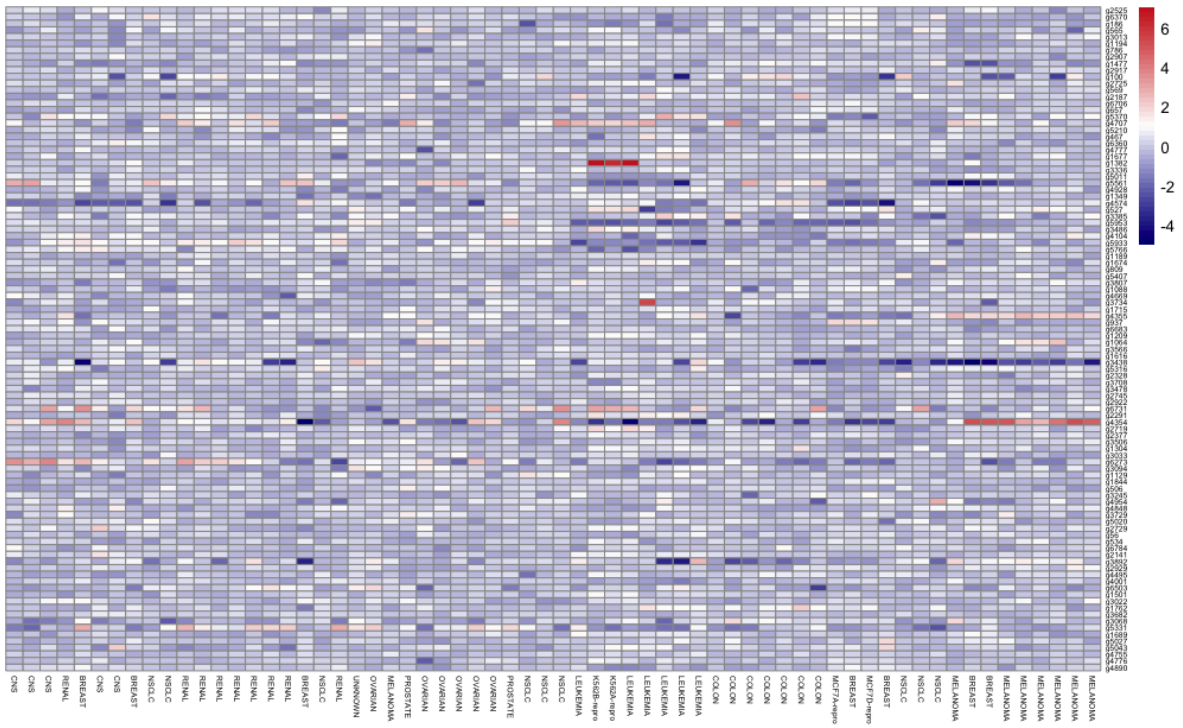
Hvis vi velger å dele data inn i tre klynger, gjør vi det rundt euklidsk avstand rundt 118. Det spesielle med dette datasettet er at vi vet hvilke celleprøver vi har observasjonene fra. Med tre klynger: Vi ser at prøver av type leukemi og tykktarm dominerer den venstre klyngen, den midterste klyngen er dominert av melanom (føflekk-kreft) og den høyre av eggstokk- og nyre-kreft-prøver.

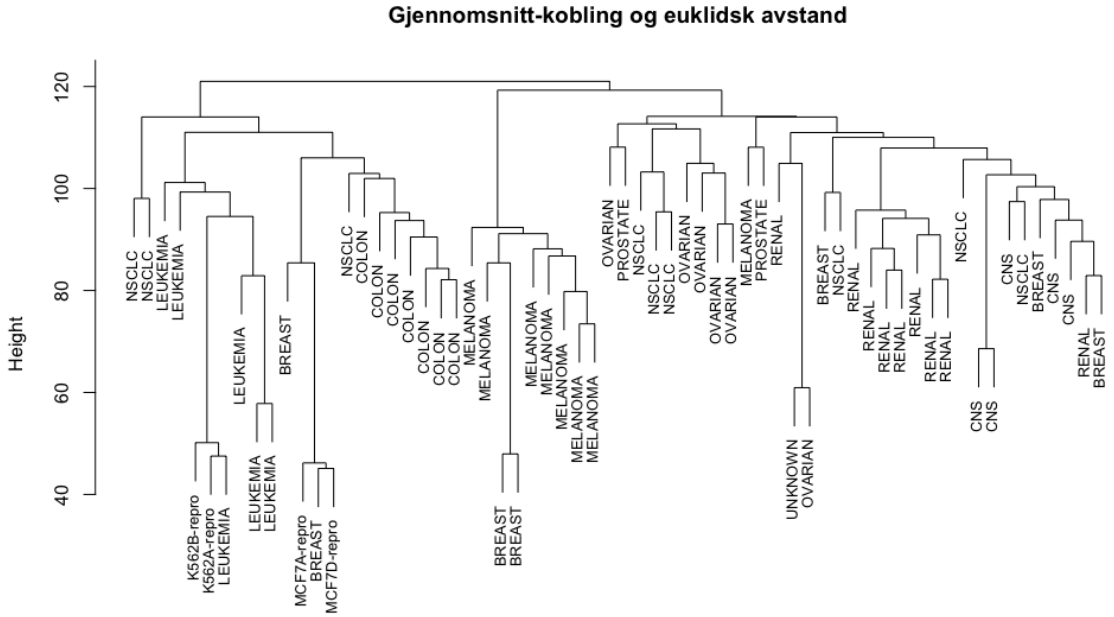
Den ukjente prøven - med merkelapp "UNKNOWN" - er i følge klyngeanalysen mest lik en eggstokk-kreft-prøve.

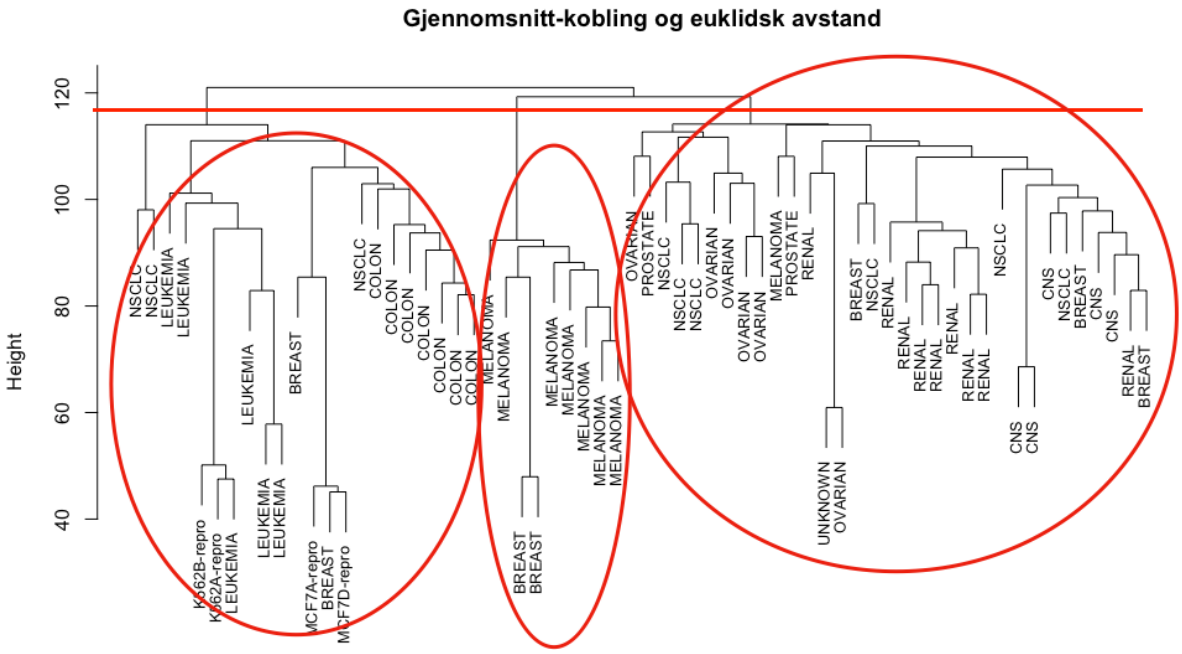
K-gjennomsnitt-klyngeanalyse

Vi skal se på den mest populære metoden for ikke-hierarkisk klyngeanalyse - K-gjennomsnitt klyngeanalyse (eng: K-means).

For denne metoden så må man bestemme antall klynger K av observasjoner på forhånd.


6
4
2
0
-2
-4


Gjennomsnitt-kobling og euklidsk avstand
Height
120
100
80
60
40
NSCLC
NSCLC
LEUKEMIA
LEUKEMIA
K562B-repro
K562A-repro
LEUKEMIA
LEUKEMIA
LEUKEMIA
BREAST
MCF7A-repro
BREAST
MCF7D-repro
NSCLC
COLON
COLON
COLON
COLON
COLON
COLON
COLON
MELANOMA
MELANOMA
BREAST
BREAST
MELANOMA
MELANOMA
MELANOMA
MELANOMA
MELANOMA
OVARIAN
PROSTATE
NSCLC
NSCLC
OVARIAN
OVARIAN
OVARIAN
OVARIAN
MELANOMA
PROSTATE
RENAL
UNKNOWN
OVARIAN
BREAST
NSCLC
RENAL
RENAL
RENAL
RENAL
RENAL
RENAL
RENAL
NSCLC
CNS
CNS
CNS
NSCLC
BREAST
CNS
CNS
RENAL
BREAST



Målet med metoden er at

- *alle* observasjoner skal være medlem i en klynge, men
- hver observasjon skal bare være medlem i *en* klynge, og
- variasjonen innen hver klynge skal være så liten som mulig

Hvordan skal vi definere variasjon innen en klynge? Jo, vi bruker et avstandsmål, og for K-gjennomsnitt-klyngeanalyse er det kvadrert euklidsk avstand D_E^2. (Andre avstandsmål gir andre navn på metoden.)

Variasjon innen en klynge

Litt notasjon:

- For klynge k har vi en mengde C_k som angir hvilke observasjoner som tilhører klyngen. Eksempel: hvis observasjon 1, 3 og 6 er med i klynge k, da er $C_k = \{1, 3, 6\}$.
- Antall observasjoner i klynge k skrives $|C_k|$. I eksemplet vil da $|C_k| = 3$.

Variasjonen innen klynge k er summen av alle parvise kvadrerte euklidske avstander mellom observasjonene i klyngen, delt på antall observasjoner i klyngen. Dette kan skrives

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

Her er da i og i' to observasjoner som er med i klynge k. Videre er x_{ij} er den målte verdien til variabel j for observasjon i og $x_{i'j}$ er den målte verdien til variabel j for observasjon i'.

Et optimeringsproblem

Hvis vi ønsker å fordele observasjonene i K klynger så variasjonen inne hver klynge er så liten som mulig, kan vi skrive dette som et optimeringsproblem.

Minimer over alle $C_1, \ldots, C_K$

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

Er det mulig å lage en algoritme for å løse dette problemet?

Det er et vanskelig problem. Hvor mange mulige løsninger finnes det? Hvis vi forenkler litt så kan vi tenke at for den første observasjonen er det K klynger som er mulig, det er det også for den andre, og så videre, så K^n mulige løsninger (med en liten korreksjon slik at vi har minst en observasjon i hver av de K-klyngene og at ikke rekkefølgen på klyngene har noe å si). Med $K = 3$ og $n = 100$ blir det i størrelsesorden $5 \cdot 10^{47}$. Det er mange mulige løsninger!

Det finnes heldigvis en enkel algoritme som har vist seg å gi en god løsning til problemet og er rask, men den kan ikke garantere noe annet enn et lokalt minimum.

Kjernen i algoritmen er at optimeringsproblemet vårt kan omskrives – Istedenfor å se på summen av kvadrerte avstander mellom alle par av observasjoner i en klynge ser vi heller på summen av kvadrert avstand fra hver observasjon i klyngen til gjennomsnittet av alle observasjoner i klyngen.

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \sum_{k=1}^{K} 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

hvor $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$.

Den p-dimensjonale vektoren med klyngegjennomsnitt kalles klyngesentroiden, $(\bar{x}_{k1}, \bar{x}_{k2}, \cdots, \bar{x}_{kp})$.

K-gjennomsnittalgoritmen

Før algoritmen starter må man velge verdi for K.

Deretter må hver av de n observasjonene tilfeldig tilordnes til en av de K klyngene. Det er flere muligheter her, og en er å tilfeldig velge ut K observasjoner og sette disse som klyngesentroider. De resterende observasjonene tilordnes klyngen med nærmeste klyngesentroide

Så starter hovedløkken i algoritmen – som vi gjentar til ingen observasjoner endrer klyngemedlemskap.

1. For hver klynge regn ut klyngesentroiden.
2. Hver observasjon tilordnes klyngen med nærmeste klyngesentroide.

Hvor god er denne algoritmen?

I hvert steg av algoritmen over så er man garantert at kriteriet man forsøker å minimere (se ligningen over) vil minke i verdi. Det betyr at man vil komme til et lokalt - eller et globalt minimum.

Dette betyr at resultatet av algoritmen vil avhenge av hvordan du tilordent hver observasjon til en klynge i steg 1 av algoritmen. For å bøte på dette er det anbefalt å kjøre algoritmen flere ganger, med ulike startsituasjoner, og så velge den beste av løsningene.

Illustrasjon av algoritmen på et syntetisk datasett

Se videoen K-gjennomsnitt-klyngeanalyse for illustrasjon av algoritmen.

K-gjennomsnitt-algoritmen i Python

Vi utfører klyngeanalysen i tre steg:

1. Angi antall klynger du ønsker
2. Initialiser K-gjennomsnitt-modellen
3. Tilpass K-gjennomsnitt-modellen

Resultatene fra metoden er klyngemedlemskap (eller *klyngemerkelapper*) og klyngesentere (sentroidene).

```
import pandas as pd # lese data
import matplotlib.pyplot as plt # plotting
from sklearn.cluster import KMeans #k-gjennomsnitt-klyngeanalyse

df = pd.read_csv('../Rwork/simkmdata.csv')

from sklearn.cluster import KMeans
modell = KMeans(n_clusters=3, random_state=0,n_init=10,init='k-means++')
tilpass=modell.fit(df)
print(tilpass.labels_)
```

```
## [0 1 0 0 0 2 2 2 1 1 1 1]
```

```
print(tilpass.cluster_centers_)
```

```
## [[ 3.41191917 -0.34986976]
##  [ 0.36475588  3.72672333]
##  [ 5.32276491  4.55513792]]
```

Hvordan skal man velge K?

Noen ganger har man en god formening om hva man at K skal være, hvis man vil dele pikslene i et bilde inn i klynger for å lage et to-farget bilde, så er det klart at K må være 2.

Andre ganger vet man ikke helt og må prøve seg frem. Da er det vanlig å velge et intervall for K, og så studere resultatene eller basere seg på å sammenligne godhetsmål for løsningene. Optimeringskriteriet vårt vil alltid minke når K øker, så vi kan ikke bare velge K så stor som mulig. En ad hoc metode heter albu-metoden og går ut på å velge K der det er en knekk på plottet for K mot optimeringskriteriet. Det finnes også andre løsninger, men dette et vanskelig problem.

Referanser

Presentasjonen av hierarkisk klyngeanalyse er basert på kapittel 10.4, og K-gjennomsnitt-klyngeanalyse på kapittel 10.3 i boka "Introduction to statistical learning with applications in R" av James, Witten, Hastie og Tibshirani, og brukes som pensumlitteratur i emnet TMA4268 Statistisk læring. Boka er tilgjengelig som pdf, https://www-bcf.usc.edu/~gareth/ISL/, eller som ebok fra Springer (er du på vpn ntnu så er den tilgjengelig): https://www.springer.com/gp/book/9781461471370

Datasettet med genuttrykk i celleprøver er beskrevet her: http://genome-www.stanford.edu/nci60/, og finnes i R-pakken ISLR https://cran.r-project.org/web/packages/ISLR/index.html. Figurene er laget i R.

De andre to datasettene er oppkonstruert for å enkelt forklare algoritmene for klyngeanalyse.

Hierarkisk klyngeanalyse kan gjøre i Python ved å bruke funksjonene linkage, dendrogram og fcluster i scipy.cluster.hierarchy https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html og K-gjennomsnitt-klyngeanalyse ved funksjonen KMeans i sklearn.cluster https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

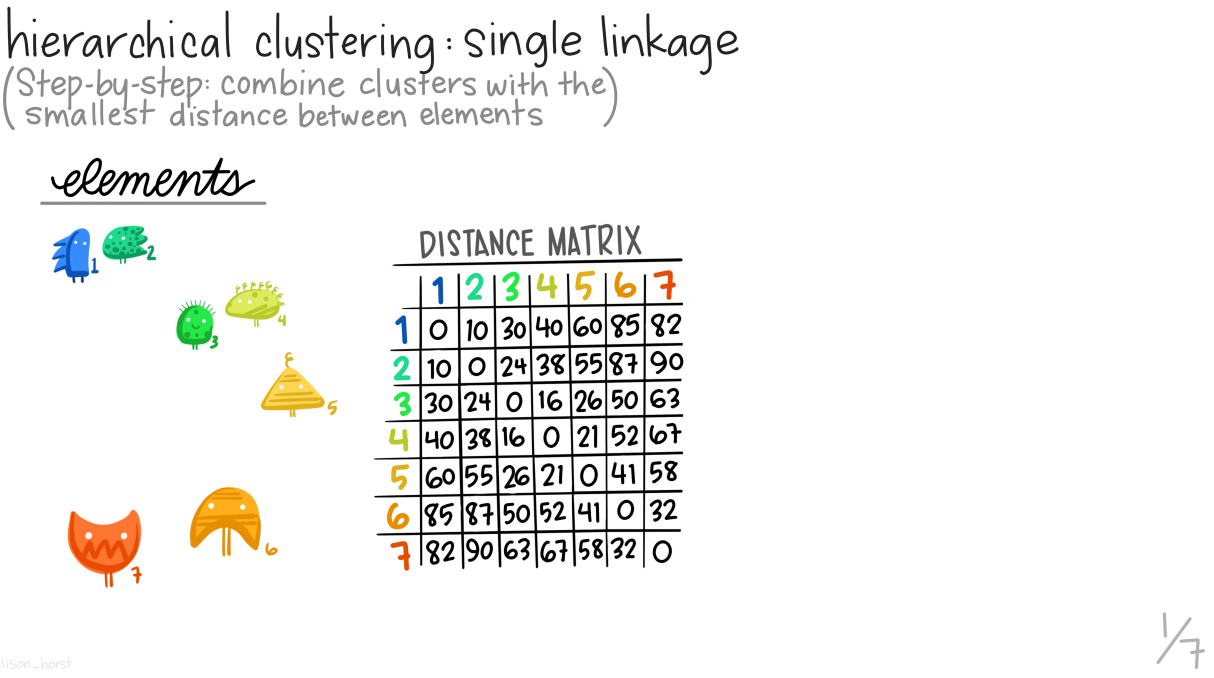
Vedlegg: klyngeanalyse som tegneserie

Hierarkisk klyngeanalyse som en tegneserie

Credit: Artwork by @allison_horst

Vi har 7 observasjoner og vil gjøre en hierkisk klyngeanalyse med single linkage og et (uspesifisert) avstandsmål.

1/7) Vi starter med å beregne en avstandsmatrise mellom alle 7 observasjoner.



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

2/7) observasjon 1 og 2 var nærmest hverandre og de blir en klynge.

3/7) Siden vi bruker single linkage (minste avstand mellom ethvert observasjon i en klynge mot og alle observasjonene i en annen klynge) trenger vi egentlig ikke lage en ny avstandsmatrise, bare bruke den ved å se på minste avstand til observasjoner i en klynge (etter vi har strøket over avstander mellom observasjoner som er i samme klynge). Vi kikker etter minste avstand i avstandsmatrisen og ser at det er mellom observasjon 3 og 4. De blir nå en klynge.

4/7) Nå er observasjon 5 som er nærmest hverandre klyngen med 3 og 4. Vi lager en ny klynge med 3 og 4 og 5.

5/7) Nå er det korteste avstanden mellom observasjon 2 og 3, og siden observasjon 2 er i klynge med 1 og observasjon 3 er i klynge med 4 og 5 så får vi nå en stor klynge med observasjon 1, 2, 3, 4 og 5.

6/7) observasjon 6 og 7 er nå nærmest hverandre og de blir en klynge.

7/7) Da er det to klynger, og de vil til slutt settes sammen til en stor klynge.

K-gjennomsnitt-klyngeanalyse som en tegneserie

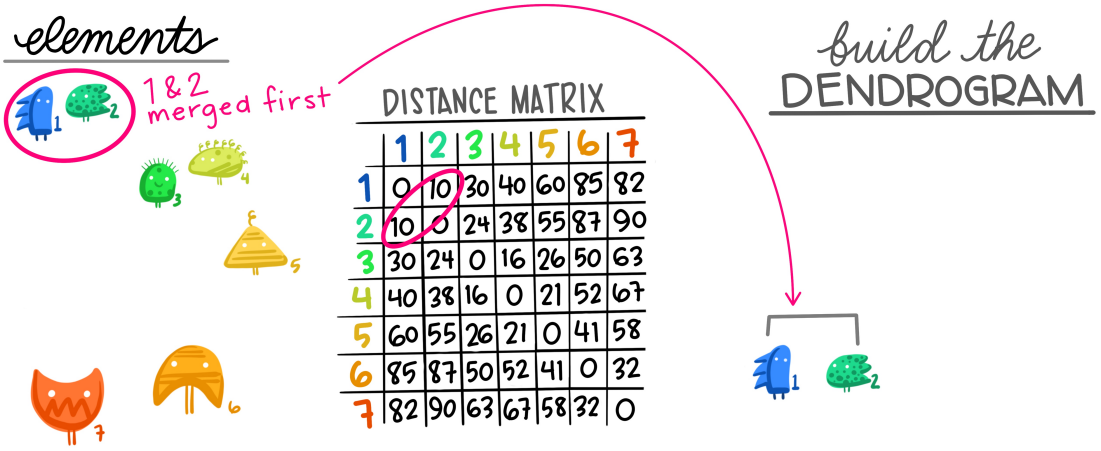
Credit: Artwork by @allison_horst

1) Vi har bestemt at vi leter etter $K = 3$ klynger, og velger tilfeldig posisjonen til tre såkalte sentroider i dataene.

2) Det kan godt være å tilveldig velge ut tre observasjoner i datasettet til å være posisjonen til tre sentroider.

3) Bestem hvilken klynge hver av de n observasjonene tilhører - ved å se hvilken sentroide som er nærmest. Her inngår valg av avstandsmål. Default er kvadratisk euklidsk avstand.

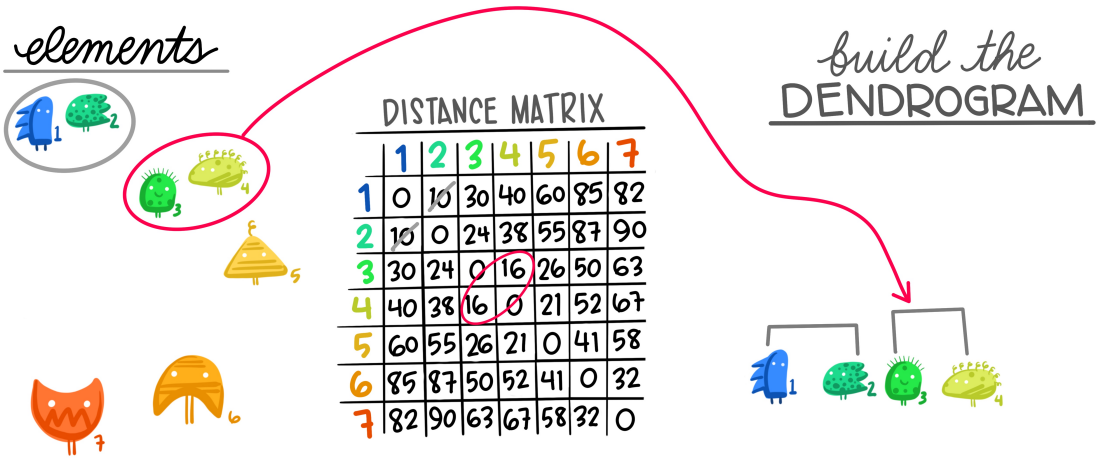
Treat each element as a cluster
- Find smallest distance between elements in 2 clusters
- Those clusters get merged.



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

2/7

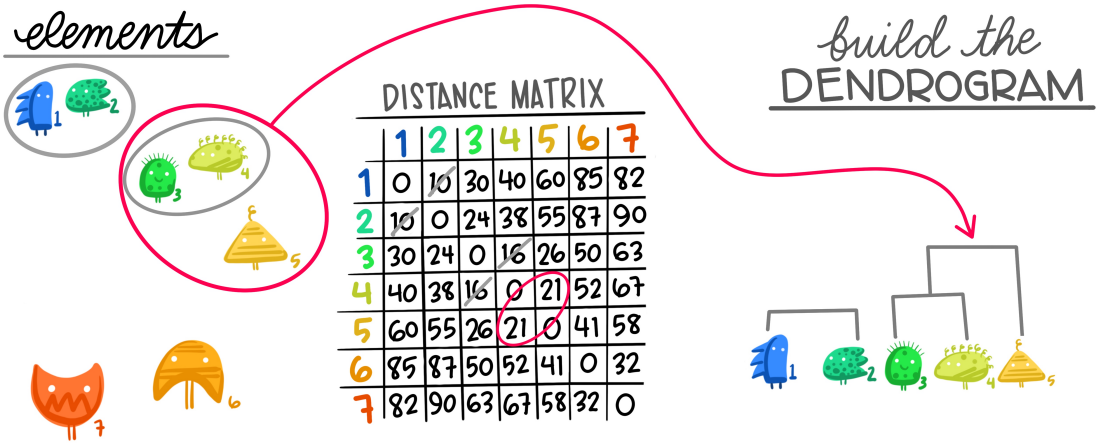
Now 1 & 2 are a single cluster.
Find the 2 clusters with smallest distance between elements, then merge them.



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

3/7

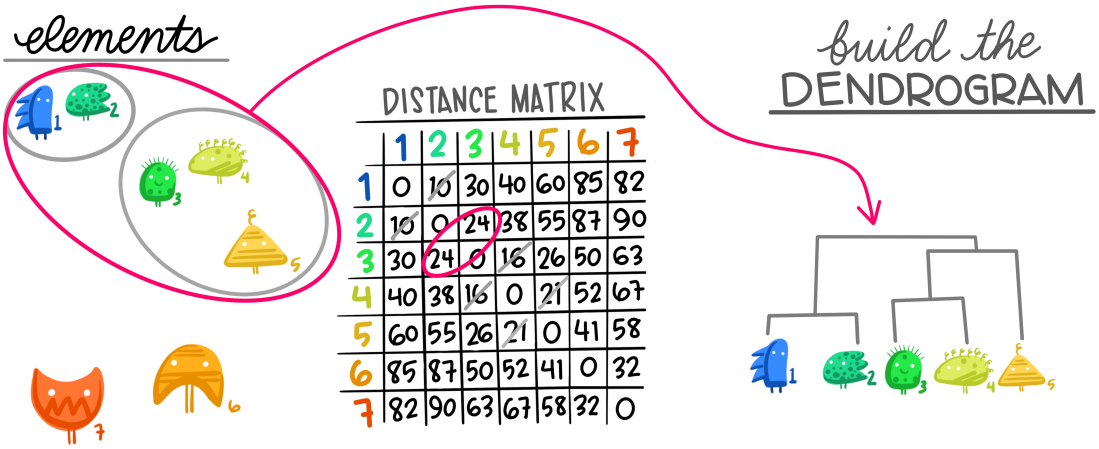
Repeat! Now the 2 clusters with the smallest distance between elements are the (3,4) and 5 clusters, so we merge them!



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

4/7

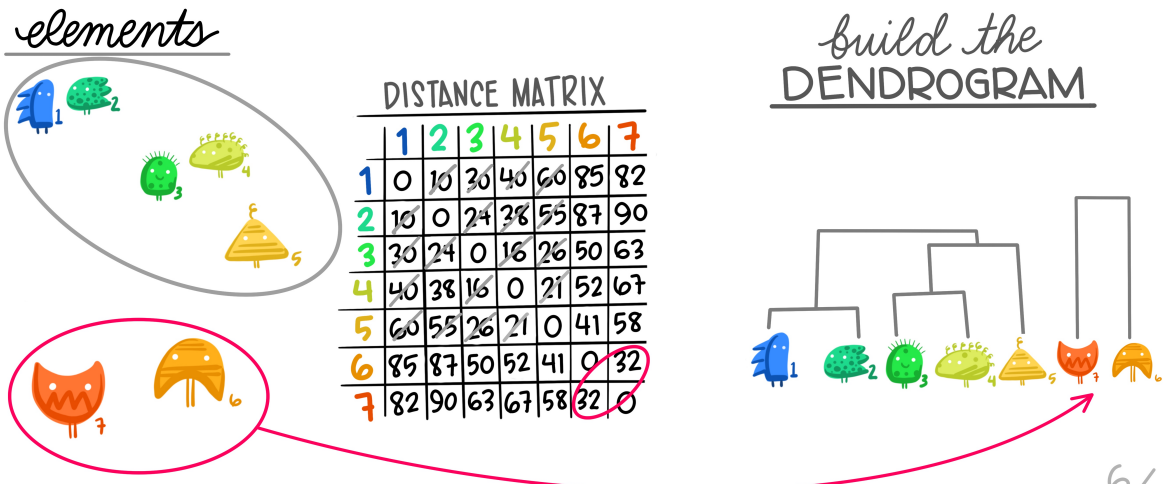
Yep, do it again! Now, the smallest distance between elements in two clusters is between 2 & 3, so we merge the clusters they're in!



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

5/7

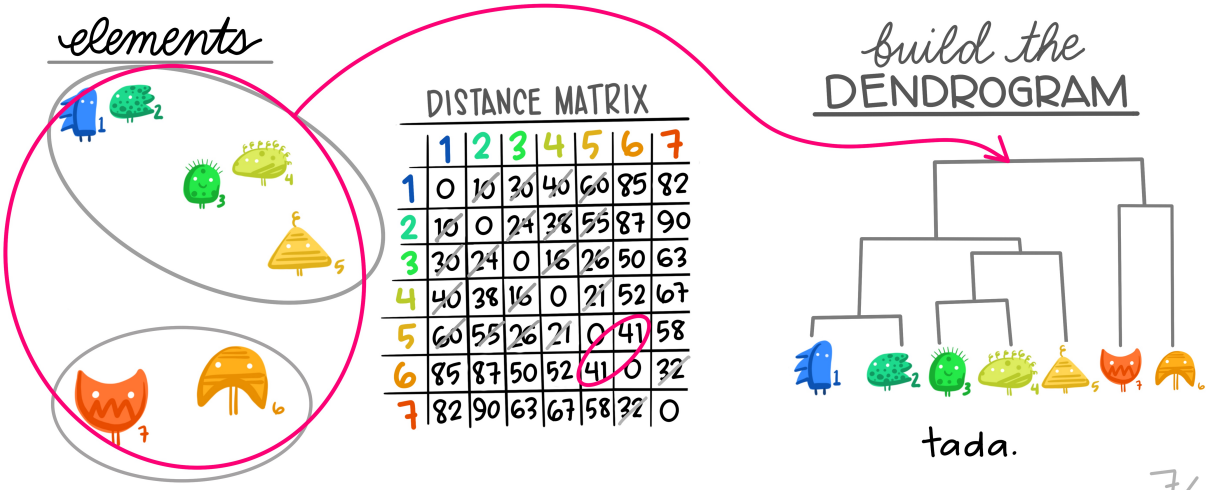
The next smallest distance between elements in separate clusters is between 6 & 7, so we merge them...



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

6/7

Now we only have two clusters, so they get merged!



	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

7/7

4) Nå er alle observasjoner tilordenet et klynge, og nå beregnes nye sentroider basert på hvilke observasjoner som tilhører hver klynge. Det betyr at plasseringen til sentroidene vi satt i steg 2 nå endrer seg.

. . .) Men, nå som sentroidene har flyttet på seg så er det ikke sikkert at alle observasjoner er nærmest sin sentroide.

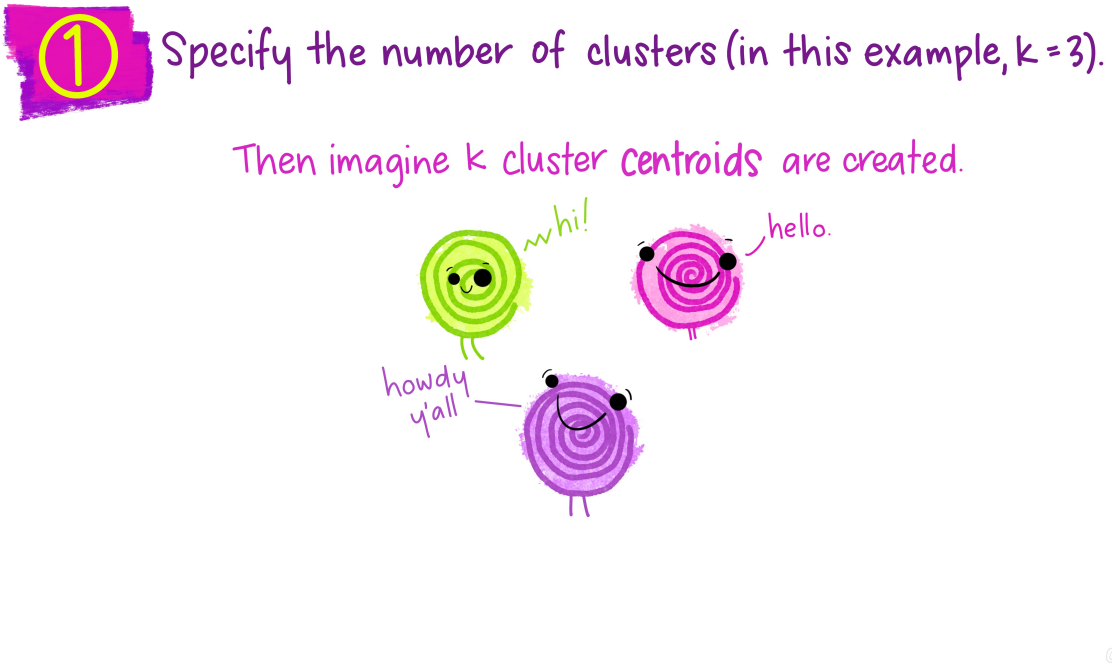
5) Bestem nytt klyngemedlemskap slik at alle observasjoner er med i klyngen til sentroiden de er nærmest. å som observasjonene har endret klyngemedlemskap må vi regne på nytt hva som er sentroidene i de ulike klyngene.

6) Ny klyngetilhørighet for noen observasjoner.

7) Nye klyngesentroider igjen.

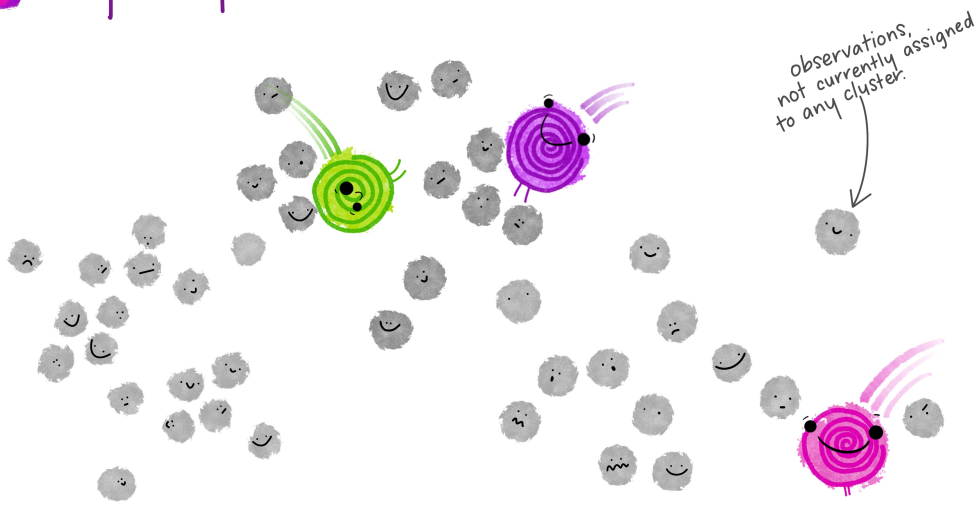
. . .) Dette er en iterativ prosess og vi fortsetter til ingen observasjoner endrer klyngemedlemskapet sitt.

fin) Ferdig - alle observasjoner er i sin endelige klynge!



N

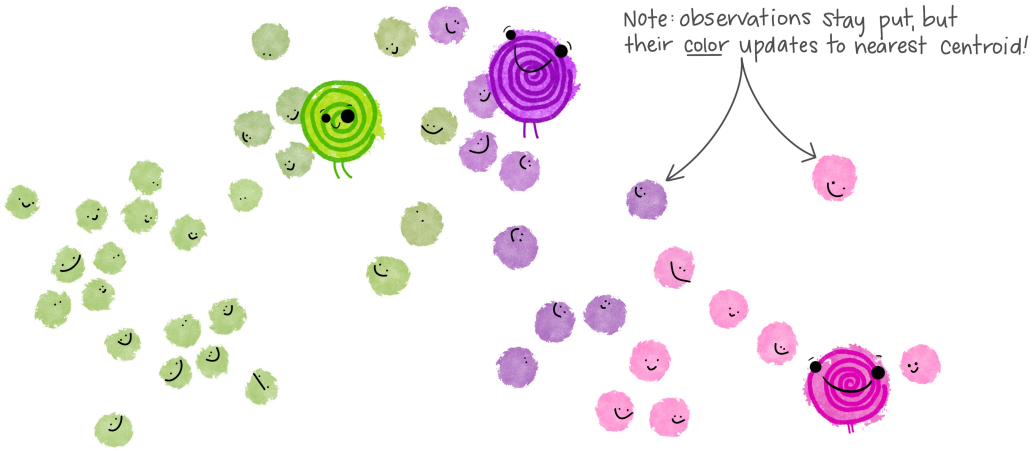
2 Those k centroids get randomly placed in your space.



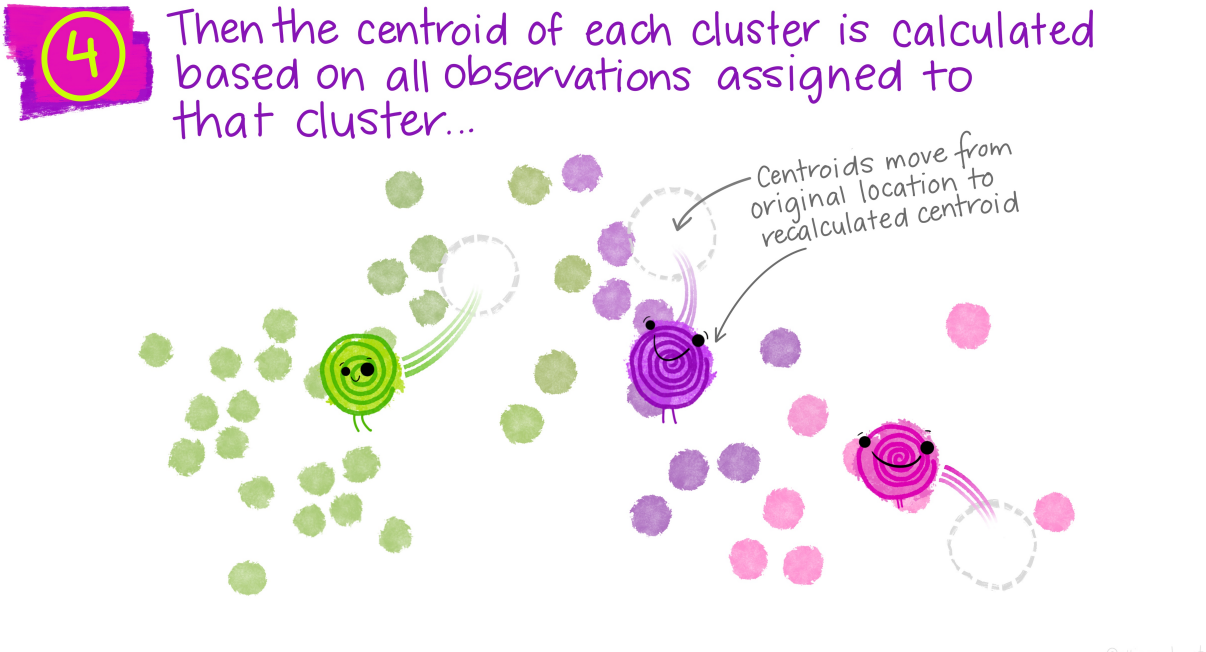
@allison_horst

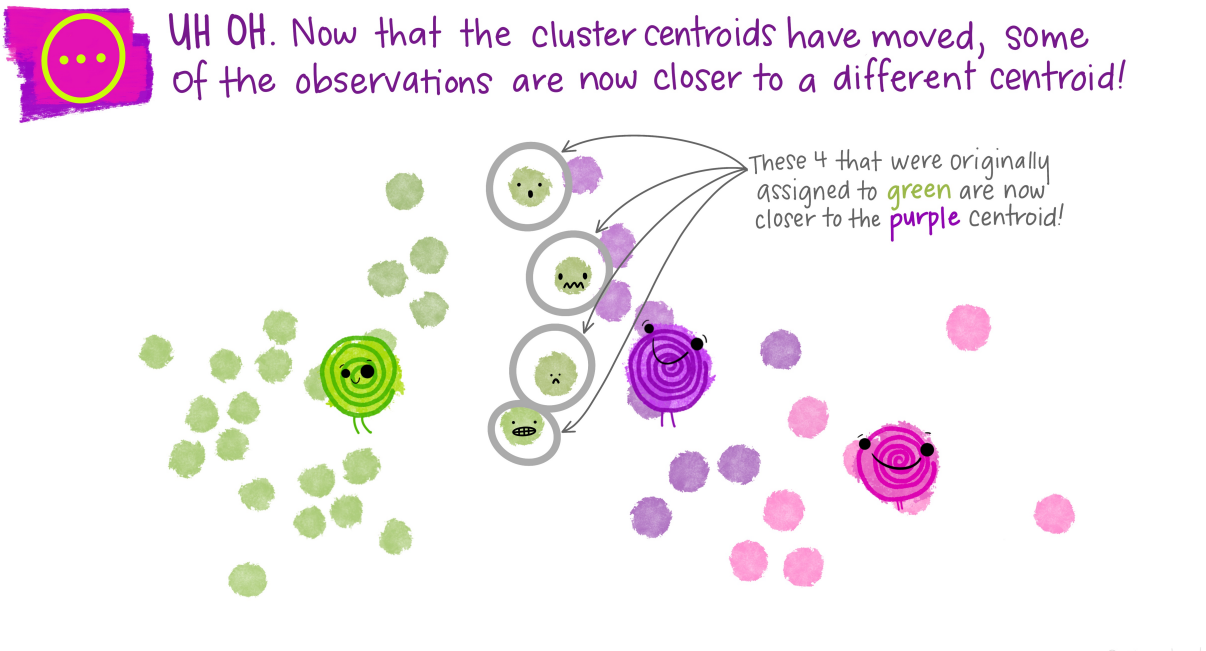
3 Each observation gets temporarily "assigned" to its closest centroid.

(e.g. by Euclidean distance)



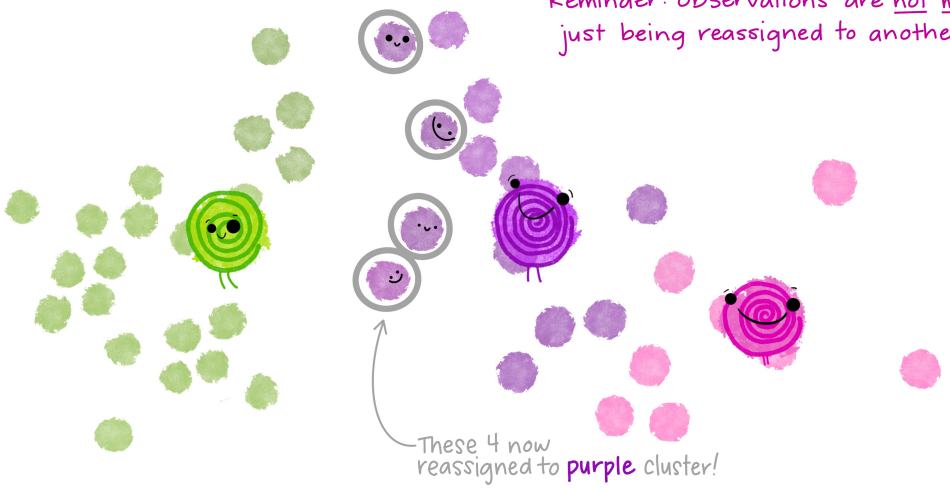
@allison_horst


4
Then the centroid of each cluster is calculated based on all observations assigned to that cluster...
Centroids move from original location to recalculated centroid
@allison_horst


...
UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!
These 4 that were originally assigned to green are now closer to the purple centroid!
@allison_horst

5 NO PROBLEM!
Observations get reassigned* to a different cluster based on the recalculated centroid.

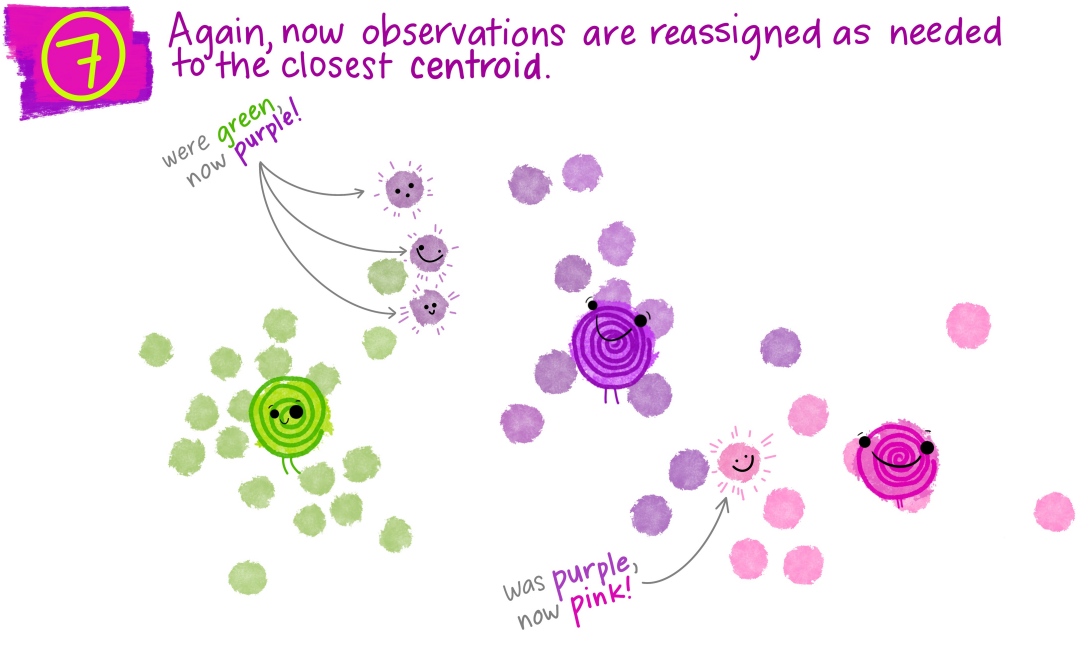
*Reminder: observations are not moving, just being reassigned to another cluster.

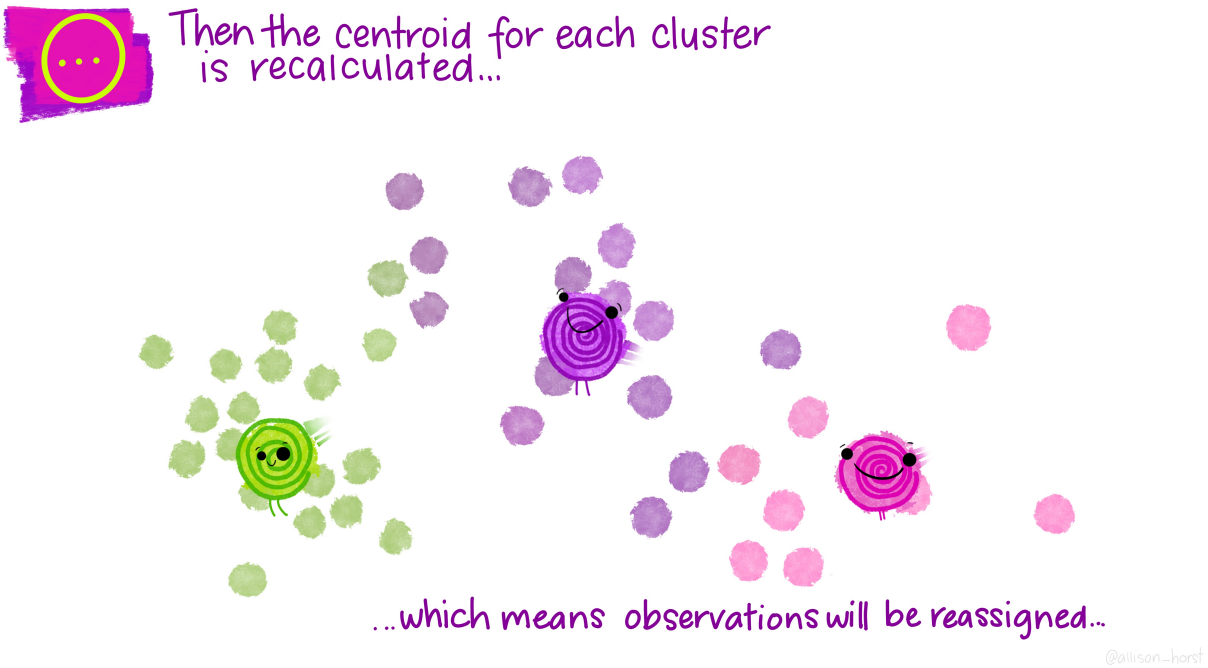


@allison_horst

6 But now that observations have been reassigned, the centroids need to move again [recalculate centroids from updated clusters]

@allison_horst


7
Again, now observations are reassigned as needed to the closest centroid.
were green, now purple!
was purple, now pink!
@allison_horst


...
Then the centroid for each cluster is recalculated...
...which means observations will be reassigned...
@allison_horst

That iterative process of

Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid

↓

Continues until nothing is moving or being reassigned anymore!

@allison_horst

Which means the iteration is done and each observation is assigned to its final cluster.

@allison_horst