

# obligatorisk oppgave statistikk

Kristian Skibrek

all kode for oppgaven er tilgjengelig på [\[github\]](#)(legg til url her)

## Oppgave 1

```
import pandas as pd

varme = pd.read_csv("./varme.csv")
kulde = pd.read_csv("./kulde.csv")
```

### forklaring av kode:

Jeg lager dataframes av csv filene med pandas read\_csv metode

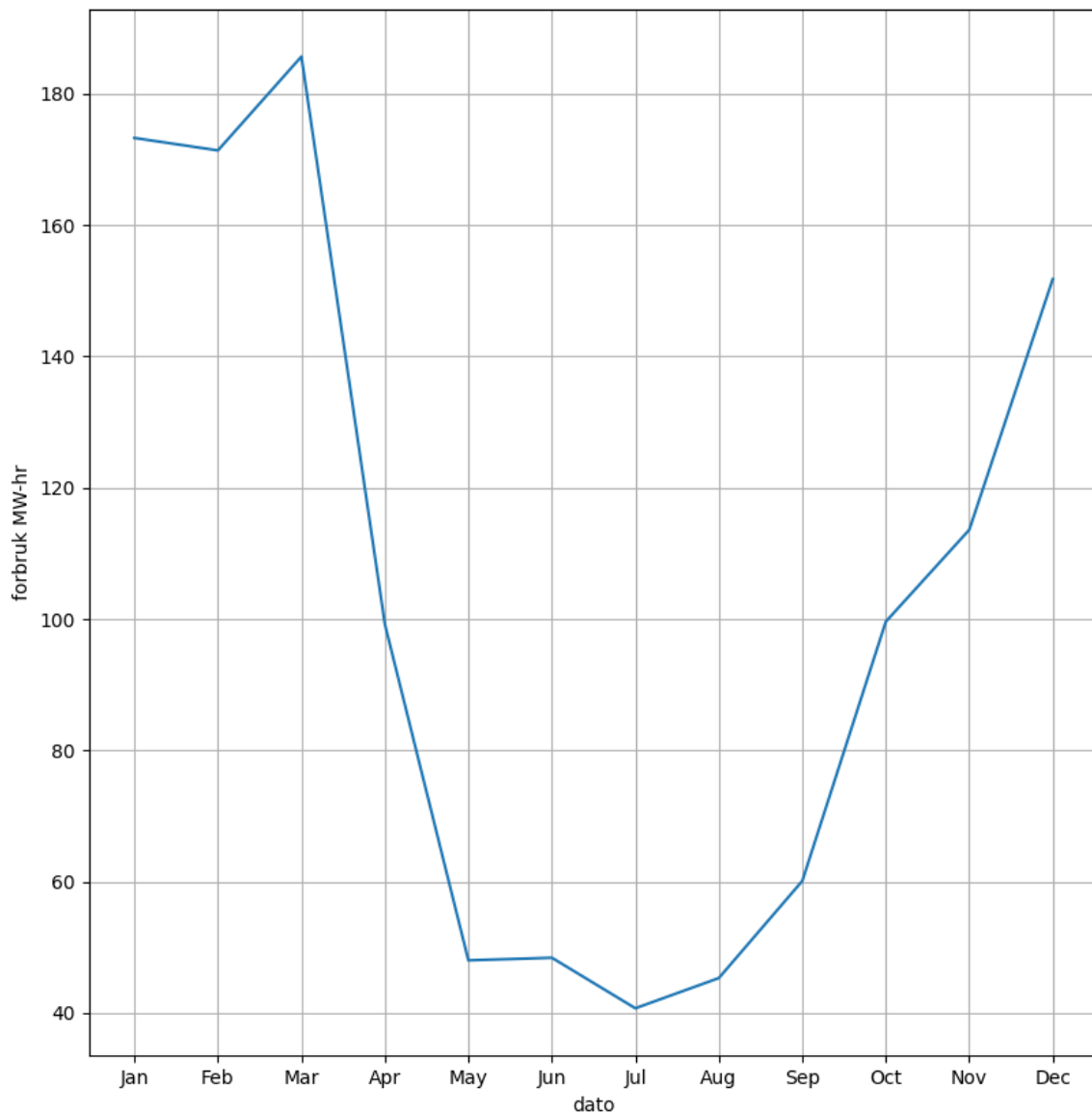
## Oppgave 2

```
varme["Timestamp"] = pd.to_datetime(varme["Timestamp"], format="%d-%b-%y
%H:%M:%S")
varme.at[0, "Timestamp"] = varme.at[0, "Timestamp"] -
pd.DateOffset(months=1)
varme['maaned'] = varme['Timestamp'].dt.to_period('M')
maaneder_v = varme.groupby('maaned').tail(1)
maaneder_v.loc[:, 'Value (MW-hr)'] = maaneder_v['Value (MW-hr)'].diff()
maaneder_v = maaneder_v.drop(0)

plt.plot(list(range(0, len(maaneder_v))), maaneder_v["Value (MW-hr)"])

plt.xlabel("dato")
plt.ylabel("forbruk MW-hr")
plt.xticks(list(range(0, 12)), ["Jan", "Feb", "Mar", "Apr", "May",
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])
plt.grid()
plt.show()

plt.show
```



## forklaring av kode

jeg finner forbruket per måned ved å ta forbruket på slutten av måneden og trekke fra forbruket på slutten av forrige måned. Jeg finner forbruket på slutten av hver måned ved å gi hver rad en kollonne for måneden den tilhører med `to_period('M')` og velger så den siste av hver måned med `.groupby('maaned').tail(1)`. Jeg finner forskjellen mellom forbruket i hver rad med `maaneder['Value (MW-hr)'].diff()`. For å få en gyldig verdi for

januar sier jeg at den hører til måneden før og fjerner den igjen før jeg plotter (fordi første raden i dataframen til .diff blir NaN).

kilder jeg brukte for å komme fram til denne koden:

[finn differanse mellom hver linje i pandas dataframe](#)

[gruppere rader i dataframe](#)

[offset dato for rad i dataframe](#)

[finn periode for rad i dataframe](#)

## Oppgave 3

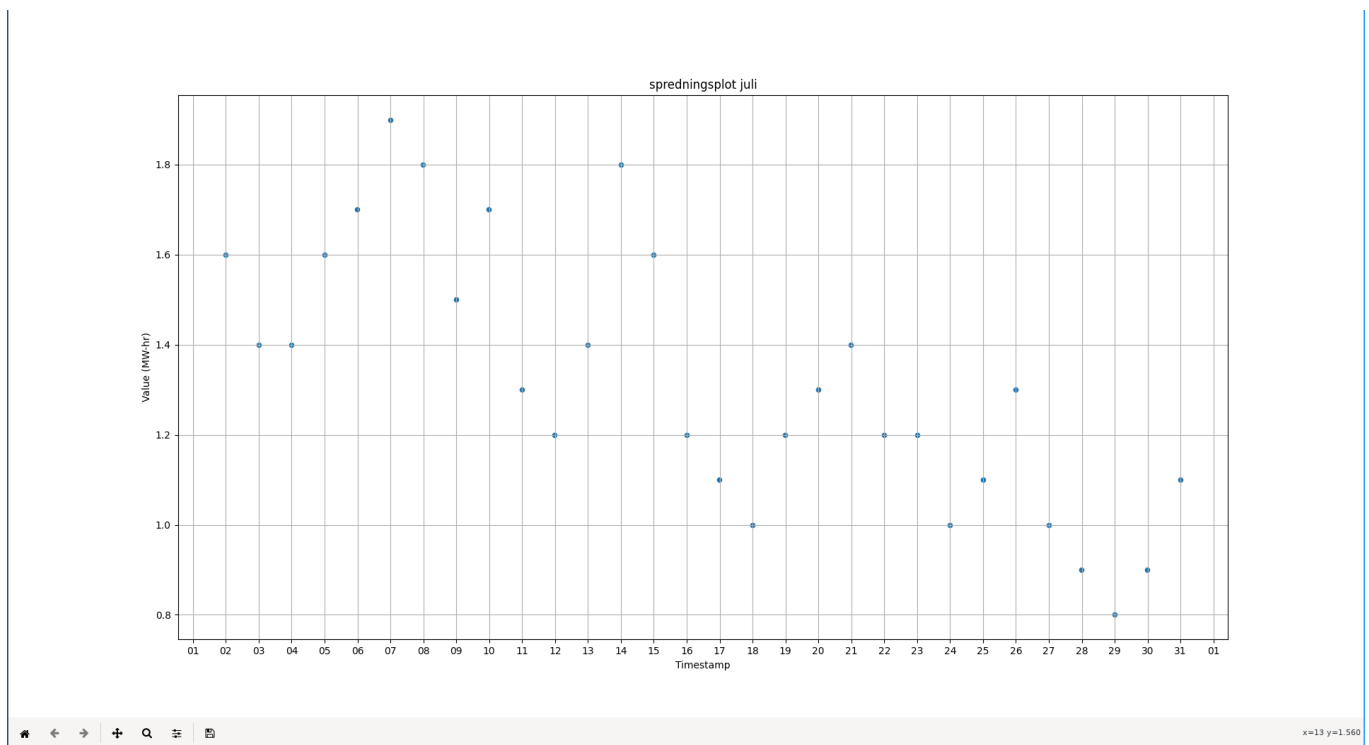
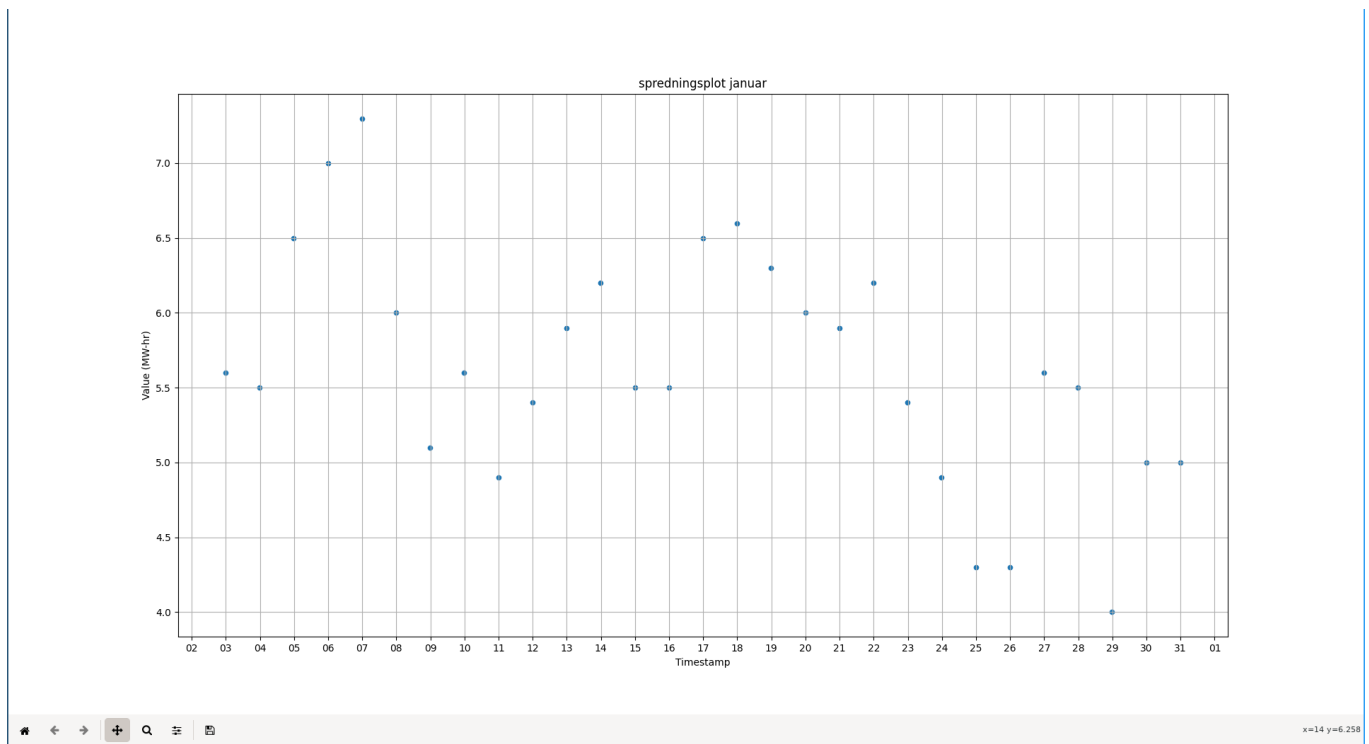
```
varme['dag'] = varme['Timestamp'].dt.to_period('D')
dager_v = varme.groupby("dag").tail(1)
dager_v.loc[:, "Value (MW-hr)"] = dager_v['Value (MW-hr)'].diff()

dager_jan = dager_v.loc[
    (dager_v['Timestamp'] > "2018-01-01") &
    (dager_v['Timestamp'] < "2018-01-31")]
dager_jul = dager_v.loc[
    (dager_v['Timestamp'] > "2018-07-01") &
    (dager_v['Timestamp'] < "2018-07-31")]

dager_jan.plot.scatter("Timestamp", "Value (MW-hr)")
plt.gca().xaxis.set_major_locator(DayLocator())
plt.gca().xaxis.set_major_formatter(DateFormatter("%d"))
plt.grid()

dager_jul.plot.scatter("Timestamp", "Value (MW-hr)")
plt.gca().xaxis.set_major_locator(DayLocator())
plt.gca().xaxis.set_major_formatter(DateFormatter("%d"))
plt.grid()

plt.show()
```



## forklaring av kode

jeg bruker samme teknikker som i forrige oppgave, men gir varme dataframen en kolonne for dag også. Jeg lager så en ny dataframe hvor jeg filtrerer ut kun timestamps om hører til januar og juli.

kilder jeg brukte for å komme fram til denne koden:

[boolean masking for å filtrere dataframes](#)

[datodager på x akse](#)

## Oppgave 4

```
print(np.mean(dager_jan["Value (MW-hr)"]))
print(np.mean(dager_jul["Value (MW-hr)"]))
print(np.std(dager_jan["Value (MW-hr)"], ddof=1))
print(np.std(dager_jul["Value (MW-hr)"], ddof=1))
```

output:

```
5.603333333333333
1.3200000000000012
0.7919784624959691
0.2952497481677353
```

## forklaring av kode

jeg bruker funksjoner fra numpy for å regne ut gjennomsnitt og standardavvik.

[numpy.mean](#)

## Oppgave 5

ddof står for "*delta degrees of freedom*". Bruken forklarest lettest ved å vise hvor den kommer inn i formelen for standardavviket:

$$\sqrt{\sum \frac{\bar{x} - x_i)^2}{n - \text{ddof}}}$$

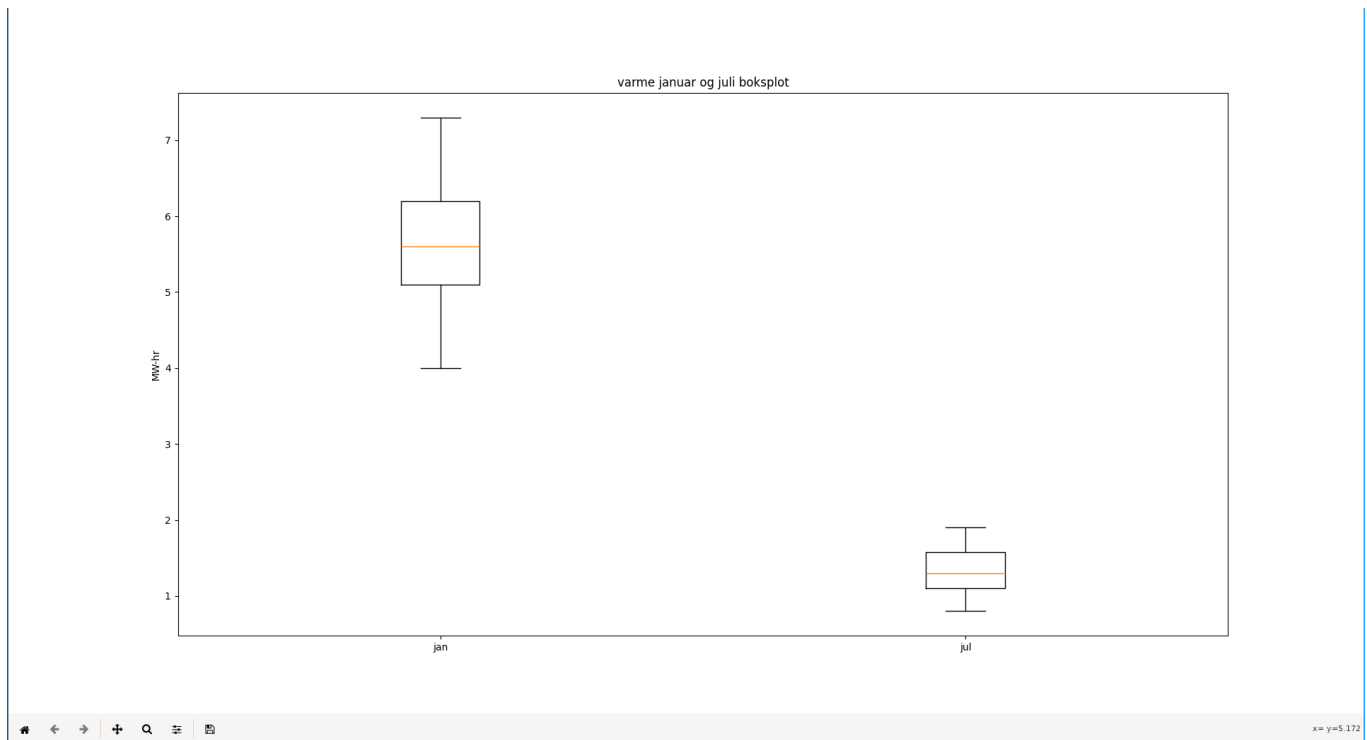
ved å sette ddof til 1 får vi divisjon med  $n - 1$  istedenfor  $n$ . i vårt tilfelle er det  $n-1$  vi vil ha fordi på grunn av antallet uavhengige differanser. Dette kalles presis utvalgsvarians.

[numpy standard deviation](#)

[wiki standard deviation](#)

## Oppgave 6

```
plt.boxplot([dager_jan["Value (MW-hr)"], dager_jul["Value (MW-hr)"]])
plt.xticks([1, 2], ["jan", "jul"])
plt.ylabel("MW-hr")
```

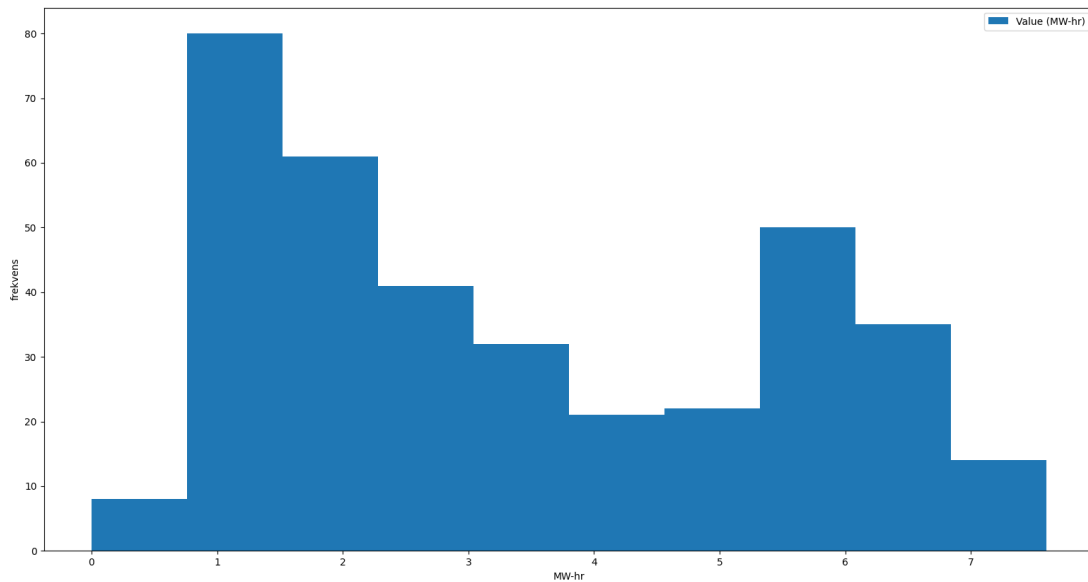


## forklaring av kode

for å plote dagene som boksplot bruker jeg [matplotlib.pyplot.boxplot](#)

## Oppgave 7

```
dager_v.drop(0).plot.hist()
plt.ylabel("frekvens")
plt.xlabel("MW-hr")
```



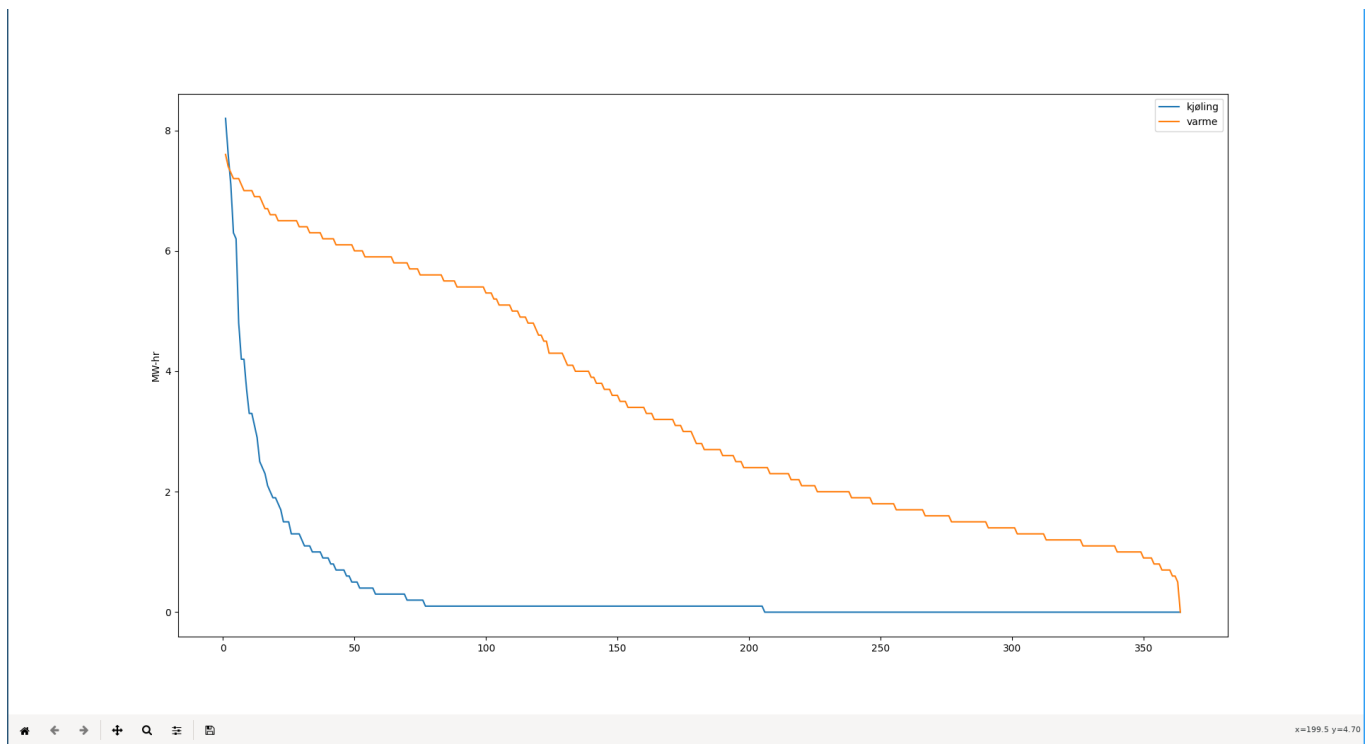
## forklaring av kode:

Jeg bruker dataframes innebygde abstraksjon av matplotlib for å plote funksjonen. jeg dropper første kolumnen fordi det er en kopi av den første målingen satt til dagen før (se tidligere oppgave)

## Oppgave 8

```
# lage dataframe for dagsforbruk av kjøling
kulde["Timestamp"] = pd.to_datetime(kulde["Timestamp"], format="%d-%b-%y
%H:%M:%S")
kulde['dag'] = kulde['Timestamp'].dt.to_period('D')
dager_k = kulde.groupby("dag").tail(1)
dager_k.loc[:, "Value (MW-hr)"] = dager_k['Value (MW-hr)'].diff()

plt.plot(sorted(dager_k["Value (MW-hr)"], reverse=True),
label="kjøling")
plt.plot(sorted(dager_v["Value (MW-hr)"], reverse=True), label="varme")
plt.ylabel("MW-hr")
plt.legend()
plt.show()
```



## forklaring av kode:

jeg lager først en dataframe for kjøling, så plotter jeg dem og sorterer etter bruken av mw i baklengs ordre, så det største kommer først.

## Oppgave 9

jeg fant ingen gode kilder på hva et persentilplott er, googling ga ingen gode resultater, og ordet ser ikke ut til å være nevnt i boken heller, jeg måtte derfor (i tillegg til kildene nevnt i teksten under) ta i bruk løsningsforslaget for å se om jeg var på rett spor med svaret mitt.

jeg går utifra at persentil p[ engelsk er persetnile og bruker [wikipedias definisjon av persentiler](#) og [How to Construct a Percentile Graph](#) for å komme til konklusjonen at persentiler måler hvor stor del av det totale uvalget hver verdi er. Det er altså dette som plottes på y-aksen og på x-aksen har man selve verdien. Dette er altså i motsetning til varighetskurven som har mw på y-aksen og dag på x-aksen.

## Oppgave 10



Varighetskurven sier meg at det brukes mer strøm på varme enn kjøling. Histogrammet har to topper på rundt 6 og 1 mw-hr, noe som også reflekteres i varighetskurven, siden det er flatepartier her. Dette betyr at det har vært mange dager med 1 og 6 mw-hr forbruk.