Cursus Robbe Magerman Data Science and AI

Robbe Magerman

28/05/2025

Inhoudstafel

1 Hoofdstuk 0: Basisgegevens

- Toevoegen van alle mogelijke package imports die we nodig gaan hebben

```
# Importing the necessary packages
import numpy as np                                    # "Scientific computing"
import scipy.stats as stats                           # Statistical tests

import pandas as pd                                   # Data Frame
from pandas.api.types import CategoricalDtype

import matplotlib.pyplot as plt                       # Basic visualisation
from statsmodels.graphics.mosaicplot import mosaic    # Mosaic diagram
import seaborn as sns                                 # Advanced data
    visualisation
```

2 Hoofdstuk 1: Samples

2.1 Theorie

2.2 Measurement levels (NORI):

2.2.1 Kwalitatief

- Beschrijven eigenschappen of categorieën.
- Niet numeriek (of als het wel cijfers zijn, betekenen ze niks wiskundig).
- Voorbeelden:
  - Geslacht (man/vrouw)
  - Land (België, Frankrijk)
  - Kleur (rood, blauw)

2.2.2 Quantitatief

- Zijn echte getallen met een betekenis.
- Je kunt ermee rekenen (optellen, gemiddelde nemen, verhoudingen, enz.).
- Voorbeelden:
  - Leeftijd in jaren
  - Temperatuur in graden
  - Lengte in meter

2.2.3 Kwalitatieve measurements

- Nominaal (geen volgorde):
  - Categorieën zonder rangorde.
  - Voorbeelden: geslacht, kleur, land.
- Ordinaal (wel volgorde):
  - Categorieën met rangorde.
  - Maar: de afstand tussen rangen is niet bekend of niet gelijk.
  - Voorbeelden:
    - Opleidingsniveau (middelbaar < bachelor < master)
    - Militaire rang

2.2.4 Quantitatieve measurements

- Interval:
  - Gelijke verschillen tussen waarden.
  - Geen absoluut nulpunt: 0 heeft een betekenis (niet “niets”, maar juist wel).
  - Je kan geen verhoudingen maken (20°C is niet twee keer zo warm als 10°C).
  - Voorbeelden: temperatuur in °C of °F
- Ratio (verhouding):
  - Gelijke verschillen + absoluut nulpunt (0 betekent “niets”).
  - Je kan verhoudingen berekenen.
  - Voorbeelden: lengte in meter, gewicht in kg, energie in joule.
- Voorbeeld voor de duidelijkheid:
  - 20 meter is wel 1/3 langer dan 15 meter => ratio
  - 20°C is NIET 1/3 warmer dan 15°C => interval (omdat 0°C niet “geen warmte” is)

2.3 Populatie

- Populatie = iedereen of alles dat je wil onderzoeken.
- **Voorbeeld**: alle studenten aan HoGent, alle auto’s in België, alle klanten van een winkel.
- Het is vaak te veel om allemaal tegelijk te onderzoeken, daarom pikken we een sample er uit

2.4 Sample

- Steekproef = een klein groepje uit de populatie.

- Je kiest een deel van de populatie om gegevens van te verzamelen.
- **Voorbeeld**: je wil weten hoeveel uur studenten studeren per week. Je vraagt het aan 100 studenten (dat is je steekproef) in plaats van alle 10.000.

2.5 Hoe kies je een goede steekproef

- **Random steekproef (willekeurig)**
  - **Iedereen uit de populatie heeft evenveel kans om gekozen te worden.**
  - Zo vermijd je vooroordelen of vertekeningen.
  - Voorbeeld: je trekt 100 studenten at random uit de hele studentenlijst van HoGent.
- **Niet-random steekproef (niet willekeurig)**
  - **Je kiest mensen die gemakkelijk bereikbaar zijn.**
  - Dit heet ook wel een gemakssteekproef.
  - Voorbeeld: je vraagt gewoon aan je klasgenoten of vrienden.
  - Probleem: deze groep is misschien niet representatief, dus je resultaten kunnen vertekend zijn.

2.6 Sampling errors:

- **Sampling errors:**
  - Toevallige steekproeffouten (Accidental sampling errors)
    - Fouten door **puur toeval**.
    - Je hebt niemand iets verkeerds gedaan, maar het resultaat is toch niet representatief.
    - Voorbeeld: je kiest willekeurig 10 studenten, maar het zijn toevallig allemaal jongens → je steekproef is toevallig scheef.
  - Systematische steekproeffouten (Systemmatic sampling errors)
    - Fouten die ontstaan omdat de manier waarop je mensen kiest vooroordeel inbouwt (per ongeluk mensen uitsluiten).
    - Je sluit onbewust een deel van de populatie uit.
    - Voorbeelden:
      - Online enquête → mensen zonder internet kunnen niet meedoen.
      - Straatinterview → alleen mensen die toevallig op straat zijn worden bevraagd.
      - Vrijwillige enquête → alleen mensen die zin hebben om mee te doen reageren (die zijn vaak meer betrokken).
- **Systeematische errors:**
  - Perongeluk (toevallig) (Accidental systematic errors)
    - Fouten door vergissingen tijdens het invullen of verwerken van data.
    - Voorbeeld: iemand kruist per ongeluk het verkeerde vakje aan in de enquête.
  - Systematisch (structureel) (Systematic system errors)
    - Fouten die altijd opnieuw gebeuren omdat er iets fundamenteel mis is met de meetmethode.
    - Voorbeelden:
      - Je gebruikt een verkeerd afgestelde weegschaal → alle gewichten zijn fout.
      - Mensen liegen bewust (bv. zeggen dat ze minder roken dan echt).
      - Het feit dat je iets meet, verandert het gedrag van mensen (bv. iemand gedraagt zich anders omdat ze weten dat ze geobserveerd worden).

2.7 Een dataset inlezen met basismethoden

- Data inlezen van github:

```
titanic = pd.read_csv('https://raw.githubusercontent.com/DataRepo2019/Data-
    files/master/titanic.csv')
titanic.head() # of titanic.tail(10)
```

- Properties aflezen van je dataset (aantal rijen, aantal kolommen, datatypes ...)

```
# How many rows does the DataFrame have?
print(f"Number of rows: {len(titanic)}")
```

```
# How many columns?
print(f"Number of columns: {len(titanic.columns)}")

# How many rows and columns, i.e. the shape
print(f"The shape of the Data Frame is: {titanic.shape}")

# General information about the DataFrame
print("*"*50)
titanic.info()

# Give the data type of each column.
print("*"*50)
print(titanic.dtypes)

# How many columns of each data type are there?
print("*"*50)
print(titanic.dtypes.value_counts())
```

- Sommige datasets hebben geen index, daarom kunnen we dat gemakkelijk zetten met: **set_index()**:

```
titanic.set_index(['PassengerId'])
```

2.8 Qualitatieve en ordinale variabelen

- Een kolom omzetten naar een qualitieve variabele:

```
# Describe the variable Survived -> is considered to be quantitative
print(titanic.Survived.describe())

# Convert to a categorical variable
titanic.Survived = titanic.Survived.astype('category')

# Ask to describe once more -> now it is considered to be qualitative
print(titanic.Survived.describe())
```

- Een kolom omzetten naar een ordinale variabele:

```
print(titanic.Embarked.unique())

embarked_type = CategoricalDtype(categories=['S', 'C', 'Q'], ordered=True)
titanic.Embarked = titanic.Embarked.astype(embarked_type)
titanic.Embarked.describe()

sns.countplot(data=titanic, x='Embarked')
```

2.9 Data selecteren

2.9.1 Tonen van specifieke kolommen:

```
titanic.Age # of titanic['Age']

# Select adjacent columns
titanic.iloc[:, 2:4]

# meerdere kolommen tegelijk
titanic[['Name', 'Age', 'Cabin']]
```

```
# Observation with row number 5 (counting from zero)
print(titanic.iloc[5])

# De eerste 4 kolommen
titanic.iloc[0:4]
```

2.9.2 Filteren van kolommen

```
# Volledige dataset tonen met deze volwaarde
titanic[titanic.Age < 18]

# Volledige dataset tonen met deze volwaarde maar enkel Embarked
titanic[titanic.Age < 18].Embarked

# Volledige dataset tonen met deze volwaarde maar enkel Age en Embarked
titanic[titanic['Age'] < 18][['Age', 'Embarked']]

# Filteren via een query
titanic.query("(Sex=='male') and (Age < 18)")
```

2.9.3 Kolommen verwijderen

```
titanic = titanic.drop("PassengerId", axis="columns")
```

2.9.4 Andere nuttige functies

```
# Toont de datatypes van alle kolommen
titanic.info()

# Verwijdert elke rij met minstens een Null waarde
titanic.dropna() # how="all" verwijdert enkel met rijen met alleen maar Null
    waarden

# Toont hoeveel rijen elke kolom ingevuld heeft staan
print(titanic.count())

# Rijen vervangen door nieuwe rijen
avg_age = titanic['Age'].mean()
titanic = titanic.fillna(value={'Age' : avg_age})

# Een nieuwe kolommen maken van een al-bestaande kolom
titanic['Age_Times_Two'] = titanic['Age'] * 2

# Unieke waarde tonen
titanic['Age'].unique()

# Waardes vervangen via een set
mapping_dict = {'male' : 0, 'female' : 1}
titanic['Sex'] = titanic['Sex'].map(mapping_dict)

# Waardes vervangen via een functie
def age_to_category(age):
    if age < 12:
        return "child"
    if age < 18:
```

```
        return "teen"
    return "adult"

titanic['AgeCategory'] = titanic['Age'].map(age_to_category)
titanic.head()
```

3 Hoofdstuk 2: Analysis 1 var

3.1 Measure of Central Tendency

- Mean/avarage:
  - Som van alle waarden delen door aantal waarden
- Median
  - Middelste waarde van alle waarden
  - Even waarden = middelste twee waarden / 2
- Mode
  - De waarde die het meeste voorkomt

3.2 Measures of Dispersion

- Range
  - De absolute waarde van het verschil tussen de hoogste en kleinste waarde
- Quartilen (Q1, Q2 en Q3)
  - Bereken eerst de mediaan (Q2)
  - Bereken dan tussen het begin van de waarden en Q2 ook de mediaan
  - Hetzelfde voor Q2 tot het einde
- Interkwartiele afstand
  - Het verschil tussen Q3 en Q1
- Variantie
  - We gaan de array [2, 4, 8, 16, 30]
  - Eerst berekenen we het gemiddelde = 12
  - Nu berekenen hoe ver elke waarde ligt van het gemiddelde (Xi - gemiddelde)
    - 2 - 12 = -10
    - 4 - 12 = -8
    - 8 - 12 = -4
    - 16 - 12 = 4
    - 30 - 12 = 16
    - Dan nemen we de som van al die waarden en nemen we daar de macht van
- Standaarddeviatie
  - Dit is simpelweg de vierkantswortel van de variantie

3.3 Kenmerken Standaardeviatie

- Kan de standaarddeviatie negatief zijn? → Nee, standaarddeviatie is altijd 0 of positief.
- Wat is de kleinste mogelijke waarde? → 0, wat betekent dat alle waarden exact gelijk zijn.
- Wat doen outliers met de standaarddeviatie? → Ze vergroten de standaarddeviatie aanzienlijk.
- Wat is de eenheid van standaarddeviatie? → Dezelfde als die van de originele variabele.
- Hoe interpreteer je de standaarddeviatie samen met het gemiddelde? → Ze tonen samen het centrum en de spreiding van je data.

Measurement Level	Center	Spread Distribution
Qualitative	Mode	—
Quantitative	Average/Mean Median	Variance, Standard Deviation Range, Interquartile Range

Figure 1: alt text

Measurement level	Chart type
Qualitative	Bar chart
Quantitative	Boxplot Histogram Density plot

Figure 2: alt text

3.4 Qualitatieve variabelen

- Data visualiseren van kwalitatieve variabelen tonen we dat via catplots:

```
# Bar chart in Seaborn: catplot() with 'kind = "count"''
sns.catplot(data=tips, kind='count', y='day');
```

- De 'mode' kan je weer weergeven via deze deze codes:

```
tips.mode() # or tips.nameField.mode()
```

- 'Describe' zal van alle kwalitatieve variabelen enkele statistieken weergeven:

```
tips.describe()
```

3.5 Quantitative variables

- Data visualiseren van kwalitatieve variabelen tonen we dat via boxplots:

```
sns.boxplot(data=tips, x='tip');
```

- Dit kan ook met een staafdiagram (histogram):

```
sns.displot(data=tips, x='tip'); # je kan ook bins aanpassen hier
```

- Alsook met een kdeplot:

```
sns.kdeplot(data=tips, x='tip');
```

- We kunnen ook de vorige twee plotten combineren in één:

```
sns.displot(data=tips, x='tip', kde=True);
```

- En uiteindelijk een vioolplot, dit is een combinatie van de box en density plot.

```
sns.violinplot(data=tips, x='tip');
```

3.6 Centrality and Dispersion Measures

```
# Centrality and dispersion deasures
# Mean, standard deviation & friends
print(f"Mean:                {tips.tip.mean()}")
print(f"Standard deviation:  {tips.tip.std()}") # Pay attention: n-1 in the
    denominator
print(f"Variance:            {tips.tip.var()}") # Pay attention: n-1 in the
    denominator
print(f"Skewness:            {tips.tip.skew()}")
print(f"Kurtosis:            {tips.tip.kurtosis()}")

# Median & co
print(f"Minimum:             {tips.tip.min()}")
print(f"Median:              {tips.tip.median()}")
print(f"Maximum:             {tips.tip.max()}")
percentiles = [0.0, 0.25, 0.5, 0.75, 1.0]
print(f"Percentiles          {percentiles}\n{tips.tip.quantile(percentiles)}"
    )
print(f"Inter Quartile Range: {stats.iqr(tips.tip)}")
print(f"Range :              {tips.tip.max() - tips.tip.min()}")
```

3.7 Formula for Standard Deviation

- Dit zijn alle hulpmethoden die worden gebruikt voor het berekenen hiervan:

```
# Variance of the population
def pop_var(x):
    n = len(x)
    mean = sum(x) / n
    return 1/n * sum((x - mean) ** 2)

# Standard deviation of the population
def pop_sd(x):
    return np.sqrt(pop_var(x))

# Variance of a sample
def sample_var(x):
    n = len(x)
    mean = sum(x) / n
    return 1/(n-1) * sum((x - mean) ** 2)

# Standard deviation of the sample
def sample_sd(x):
    return np.sqrt(sample_var(x))

# If we calculate the variance in both ways, we see a difference:
a = np.array([4, 8, 6, 5, 3, 2, 8, 9, 2, 5])

print(f"Population variance: {pop_var(a)}")
print(f"Sample variance:     {sample_var(a)}")
print()
print(f"Population standard deviation: {pop_sd(a)}")
print(f"Sample standard deviation:     {sample_sd(a)}")
```

- Wat doet dit:

```
population = np.arange(0, 101)
```

```
population_mean = sum(population) / len(population)
print(f'Population mean: {population_mean}')

population_variance = pop_var(population)
print(f'Population variance: {population_variance}')

population_stdev = pop_sd(population)
print(f'Population standard deviation: {population_stdev}')
```

4 Hoofdstuk 3: Central Limit Testing

```
mu = 63    # Mean score History
sigma = 8  # Standard deviation History
x = np.linspace(mu - 4 * sigma, mu + 4 * sigma, num=201)
y = stats.norm.pdf(x, mu, sigma)
plt.plot(x, y)
plt.axvline(75, color="green"); # Show a green line for Alexandra's score for
    History
```

5 Hoofdstuk 4: Bivariate Qual

5.1 Crosstab

- Hier is hoe je een crosstab kan maken met een legende:

```
df.Gender = df.Gender.astype('category')
age_scale = CategoricalDtype(categories=['Less then 20', 'Between 20 and 40', '
    Between 40 and 60', 'Over 60'], ordered=True)
df.Age = df.Age.astype(age_scale)
observed = pd.crosstab(df.Preference, df.Age)
observed.plot(kind='barh', stacked=True)
```

5.2 Chi-kwadraad toets

5.3 Goodsness of fit test

- Een Goodness-of-Fit test kijkt of de waargenomen data overeenkomt met wat je zou verwachten op basis van een kansverdeling.
- Bijvoorbeeld:
  - Stel je gooit een dobbelsteen 60 keer. Je verwacht dat elk getal (1 t/m 6) ongeveer 10 keer voorkomt. Maar je telt: 8, 12, 9, 11, 7, 13. Is dat verschil normaal toeval, of past het niet bij een eerlijke dobbelsteen?

```
types =                ['mutant', 'human', 'alien', 'god', 'demon']
observed =    np.array([   127,       75,       98,      27,      73])
expected_p = np.array([   .35,      .17,      .23,     .08,     .17])
```

- Om aan het resultaat te komen moeten we dit allemaal gaan uitvoeren:

```
alpha = 0.05                # Significance level
n = sum(observed)           # Sample size
k = len(observed)           # Number of categories
dof = k - 1                 # Degrees of freedom
expected = expected_p * n   # Expected absolute frequencies in the sample
g = stats.chi2.isf(alpha, df=dof)  # Critical value

# Goodness-of-fit-test in Python:
chi2, p = stats.chisquare(f_obs=observed, f_exp=expected)

print("Significance level  alpha = %.2f" % alpha)
print("Sample size         n = %d" % n)
print("k = %d; df = %d" % (k, dof))
print("Chi-squared        x squared = %.4f" % chi2)
print("Critical value      g = %.4f" % g)
print("p-value             p = %.4f" % p)
```

- Hier plotten we alles even

```
# Plot of the case:
# x-values:
x = np.linspace(0, 15, num=100)
# probability density of the chi-squared distribution with 4 degrees of freedom
y = stats.chi2.pdf(x, df=dof)
# the number q for which the right tail probability is exactly 5%:
q = stats.chi2.isf(alpha, df=dof)

fig, tplot = plt.subplots(1, 1)
tplot.plot(x, y)                    # probability density
```

```
tplot.fill_between(x, y, where=x>=q, # critical area
    color='lightblue')
tplot.axvline(q)                     # critical value
tplot.axvline(chi2, color='orange'); # chi-squared
```

5.4 Chi-squared toets voor onafhankelijkheid

- Hiermee test je of twee categorische variabelen onafhankelijk zijn.
- Voorbeeld: Is er een verband tussen geslacht en voorkeur?

```
# Maak een kruistabel (contingentietabel)
contingency = pd.crosstab(df['Gender'], df['Preference'])

# Chi-kwadraattoets
chi2, p, dof, expected = stats.chi2_contingency(contingency)

print("x squared = %.4f" % chi2)
print("df = %d" % dof)
print("p = %.4f" % p)
```

6 Hoofdstuk 5: Bivariate Qual Quant

- Als je wilt weten of een numeriek verschil bestaat tussen twee of meer categorieën gebruik je dit.
- Bijvoorbeeld: “Geven mannen gemiddeld meer fooi dan vrouwen?

6.1 Boxplot

- Toont spreiding, mediaan en outliers per groep.
- Prima manier om verschil tussen groepen visueel te vergelijken.

```
tips = sns.load_dataset("tips")
sns.boxplot(data=tips, x='tip', y='sex');
```

6.2 Violinplot

- Combineert boxplot met een density plot.
- Je ziet ook hoe de data verdeeld is.

```
sns.violinplot(data=tips, x='tip', y='sex');
```

6.3 Density/kdeplot

- Laat de kansverdeling van de variabele tip zien, gescheiden per geslacht.
- Handig om verschillen in vorm, spreiding of centrum te ontdekken.

```
sns.kdeplot(data=tips, hue='sex', x='tip');
```

6.4 Barplot met errors

- Toont het gemiddelde per groep + standaarddeviatie als foutmarge.
- Goed voor een snelle visuele indruk.

```
sns.barplot(data=tips, x='sex', y='tip', errorbar='sd');
```

6.5 2-sample t-test for independent samples

- Wordt gebruikt om te testen of er een significant verschil is tussen de gemiddelden van twee onafhankelijke groepen.
- Dit gebruik je wanneer je twee niet-gerelateerde groepen vergelijkt, dus twee totaal verschillende steekproeven.
- Voorbeeld:
  - Je test een medicijn op 10 mensen en vergelijkt hun reactietijd met 10 andere mensen die géén medicijn kregen.
  - Dit zijn onafhankelijke (non-related) steekproeven.

```
stats.ttest_ind(a=control, b=treatment, alternative='less', equal_var=False
    )
```

- Dit kan ook

```
stats.ttest_ind(alternative='two-sided', a=tips.tip[tips.sex == 'Male'], b=tips
    .tip[tips.sex == 'Female']);
```

Waarde	Betekenis
`"two-sided"`	Er is een verschil (je weet niet of het groter of kleiner is)

Waarde	Betekenis
`"less"`	De eerste groep (`a`) heeft een **lager gemiddelde** dan groep `b`
`"greater"`	De eerste groep (`a`) heeft een **hoger gemiddelde** dan groep `b`

6.6 The t-test for paired samples

- Gebruik je wanneer je twee metingen hebt van dezelfde groep personen/objecten.
- Dit gebruik je wanneer je dezelfde mensen/objecten twee keer meet, onder verschillende condities.
- Voorbeeld:
  - Je laat dezelfde 10 mensen eerst een medicijn nemen en meet hun reactietijd. Daarna geef je ze geen medicijn en meet je opnieuw.
  - Dit zijn afhankelijke (paired) metingen.
- Uitgewerkt voorbeeld: Zelfde auto's getest met gewone benzine vs. additieven.

```
# Measurements
regular =   np.array([16, 20, 21, 22, 23, 22, 27, 25, 27, 28])
additives = np.array([19, 22, 24, 24, 25, 25, 26, 26, 28, 32])
# Visualization
sns.boxplot(
    data={'Regular': regular, 'Additives': additives},
    orient='h');
```

```
# Paired t-test with ttest_rel()
stats.ttest_rel(regular, additives, alternative='less')
```

6.7 Effect Size

- Een test kan statistisch significant zijn maar het verschil kan praktisch onbelangrijk zijn.
- Daarom gebruiken we een effectgrootte zoals Cohen's

6.8 Cohens D

- Maat voor hoe groot het verschil tussen twee gemiddelden is, in standaarddeviaties.

```
def cohen_d(a, b):
    na = len(a)
    nb = len(b)
    pooled_sd = np.sqrt( ((na-1) * np.var(a, ddof=1) +
                          (nb-1) * np.var(b, ddof=1)) / (na + nb - 2) )
    return (np.mean(b) - np.mean(a)) / pooled_sd
```

```
# Effect size of additives in gasoline:
cohen_d(regular, additives)
```

- Betekenis resultaten | Cohen's d | Betekenis | | ——— | ————- | | 0.2 | Klein effect | | 0.5 | Gemiddeld effect | | 0.8+ | Groot effect |

7 Hoofdstuk 6: Regession Analysis

7.1 Visualisation

- Voor we een regressiemodel toepassen, is het belangrijk om eerst visueel te kijken of er een (lineair) verband is tussen twee variabelen.
- Hiervoor gebruiken we een scatterplot met relplot() uit Seaborn.

```
sns.relplot(data=penguins,  x='flipper_length_mm', y='body_mass_g');
```

- Door extra variabelen zoals species (soort) en sex (geslacht) toe te voegen, kan je groepen visueel onderscheiden:

```
sns.relplot(data=penguins,
            x='flipper_length_mm', y='body_mass_g',
            hue='species', style='sex');
```

- Ook extra dimensies zoals grootte kan je tonen:

```
sns.relplot(data=penguins,
            x='bill_depth_mm', y='bill_length_mm',
            size='body_mass_g',
            hue='species', style='sex');
```

7.2 Method of least squares introduction

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

$$\beta_0 = \overline{y} - \beta_1 \overline{x}$$

- Dit zijn de formules die we hiervoor moeten bekomen
- Dit wordt gedaan met de volgende code:

```
least_squares = pd.DataFrame({
        'x': male_chinstrap.flipper_length_mm,
        'y': male_chinstrap.body_mass_g
    })
mx = least_squares.x.mean()
my = least_squares.y.mean()


least_squares['(x-x mean)'] = least_squares['x'] - mx
least_squares['(y-y mean)'] = least_squares['y'] - my

least_squares['(x-x mean)(y-y mean)'] = least_squares['(x-x mean)'] *
   least_squares['(y-y mean)']
least_squares['(x-x mean) to the power of 2'] = least_squares['(x-x mean)'] **
   2
least_squares
```

- Hier worden de regressielijn berekent

```
male_chinstrap_x = male_chinstrap['flipper_length_mm']
male_chinstrap_y = male_chinstrap['body_mass_g']

# b en richtingscoefficient
a, b = np.polyfit(male_chinstrap_x, male_chinstrap_y, 1)
print(f'Regression line: y = {a} * x + {b}')
```

- Visulatie van de lijn

```
x_values = [xmin, xmax]
y_values = [beta1 * x_values[0] + beta0, beta1 * x_values[1] + beta0]
sns.lineplot(x=x_values, y=y_values);
sns.scatterplot(x=male_chinstrap.flipper_length_mm, y=male_chinstrap.
    body_mass_g);
```

- De grafiek kan ook zo:

```
# sns.lmplot(data=male_chinstrap, x='flipper_length_mm', y='body_mass_g');
sns.regplot(data=male_chinstrap, x='flipper_length_mm', y='body_mass_g');
```

7.3 Covariantie

- Covariantie meet de samenhang tussen twee variabelen. Het zegt iets over de richting van het verband (positief/negatief), maar niet over de sterkte.
- Covariantie berekenen:

```
np.cov(
    male_chinstrap.flipper_length_mm,
    male_chinstrap.body_mass_g,
    ddof=1)[0][1]
```

*De correlatie (Pearson's R) is een gestandaardiseerde maat voor samenhang (tussen -1 en +1):

```
cor = np.corrcoef(
    male_chinstrap.flipper_length_mm,
    male_chinstrap.body_mass_g)[0][1]
print(f"R is approximicly {cor:.4f}")
```

7.4 Coeffecient of determination

- R^2 vertelt hoeveel procent van de variantie in y verklaard wordt door x (verklaarde variantie).
- Je berekent het eenvoudig met:

```
cor ** 2
```

- Bijvoorbeeld: Bijvoorbeeld als R = 0.87 R tot de tweede is ongeveer 0.76, wat betekent dat 76% van de spreiding in y verklaard wordt door x.

$abs(R)$	R^2	**Explained variance**	**Linear relation**
< .3	< .1	< 10%	very weak
.3 - .5	.1 - .25	10% - 25%	weak
.5 - .7	.25 - .5	25% - 50%	moderate
.7 - .85	.5 - .75	50% - 75%	strong
.85 - .95	.75 - .9	75% - 90%	very strong
> .95	> .9	> 90%	exceptionally strong

Figure 3: alt text

8 Hoofdstuk 7: Time Series

8.1 Componenten van Tijdreeksgegevens

- Een tijdreeks is een reeks waarnemingen die op opeenvolgende tijdstippen zijn verzameld (bijvoorbeeld maandelijks of jaarlijks). Een tijdreeks bestaat meestal uit vier componenten:
  - **Trendcomponent**: De algemene richting van de data op lange termijn (bijv. een stijgende of dalende lijn).
  - **Seizoenscomponent**: Veel tijdreeksen vertonen een seizoenspatroon, zoals jaarlijkse herhalingen in verkoopcijfers. Dit patroon is doorgaans voorspelbaar van jaar tot jaar. Regelmatige patronen die zich voordoen op vaste tijdsintervallen, zoals jaarlijks of maandelijks. Voorbeeld: hogere ijsverkoop in de zomer
  - **Cyclische component**: Dit verwijst naar stijgingen en dalingen die geen vaste periode hebben, zoals economische recessies en herstelperiodes. Het verschil met seizoenspatronen is dat cyclische patronen onregelmatig zijn qua duur, terwijl seizoenspatronen gekoppeld zijn aan de kalender en een vaste periode hebben.
  - **Willekeurige component (of ruis)**: Dit is het onvoorspelbare deel dat de onregelmatige, zigzag-uitstraling van de meeste tijdreeksgrafieken veroorzaakt. Het omvat inherente willekeur, onvoorspelbare "schokken" en menselijk gedrag. Onverklaarbare of toevallige variaties. Dit is het "chaotische" deel dat overblijft als de andere drie componenten zijn verwijderd.

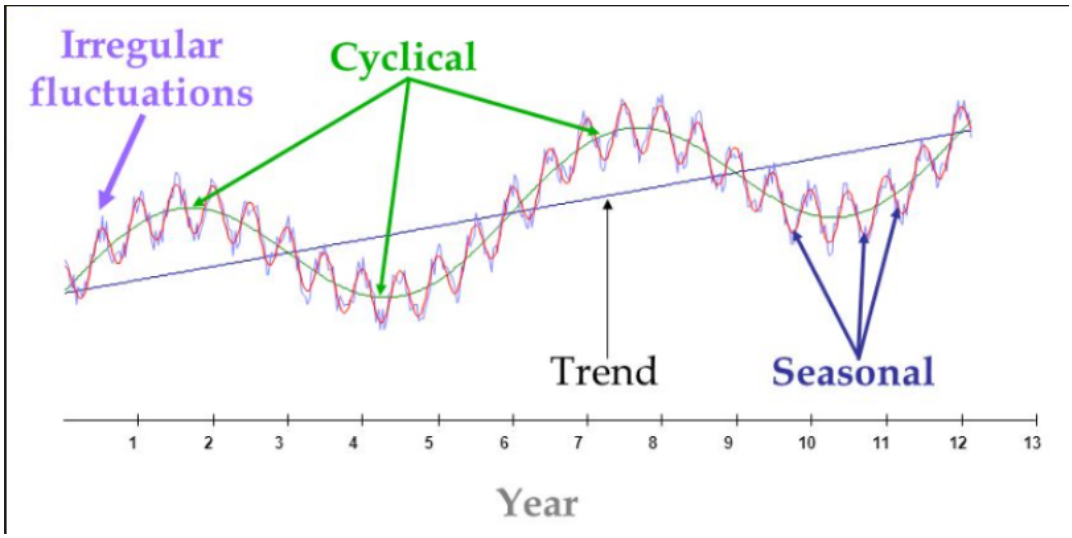

Figure 4: alt text

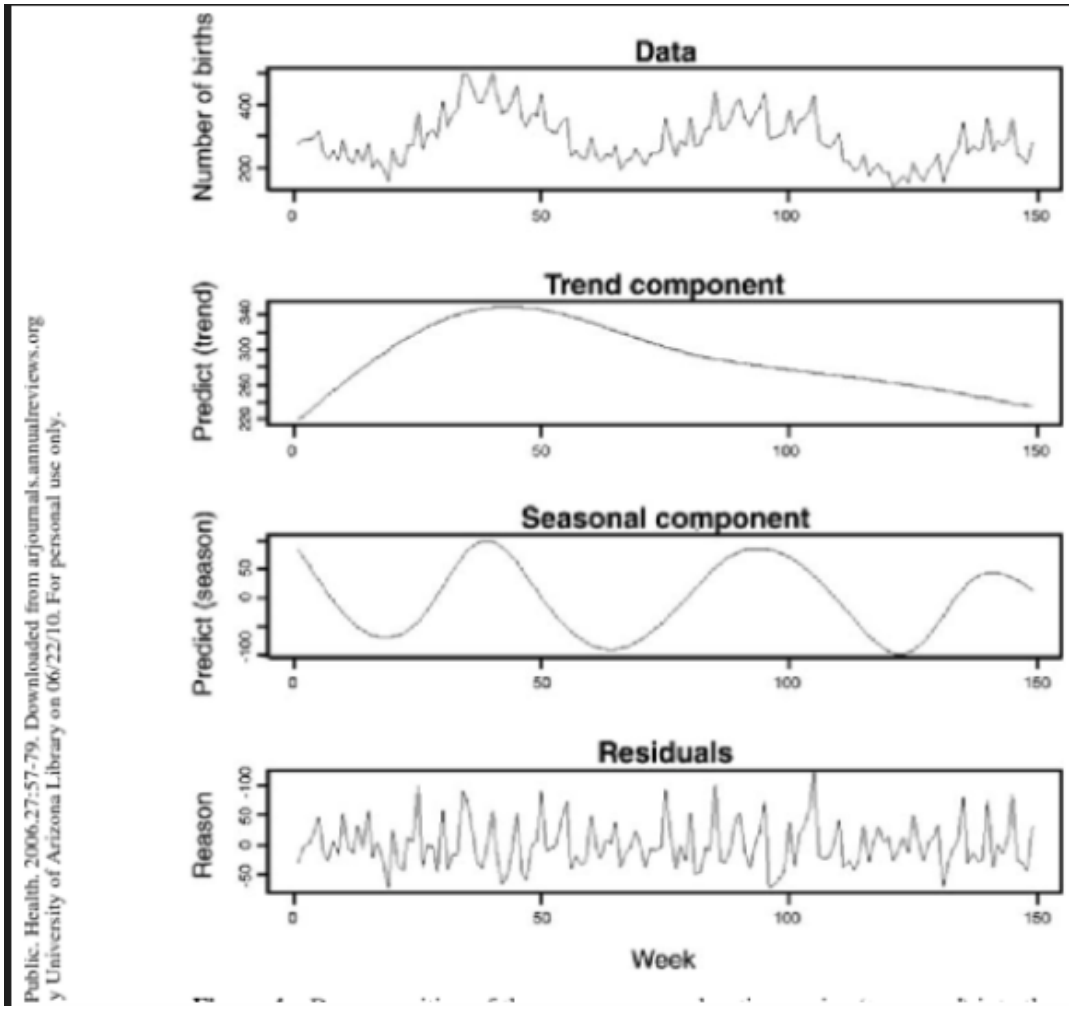

Figure 5: alt text

8.2 Werken met een date als index

- Bij tijdreeksen is het cruciaal om een datumkolom als index te gebruiken. Zo kan pandas automatisch met tijdstructuren werken.

```
# Read the data from CSV, parse dates and set the index
wounded_data = pd.read_csv(
    'https://raw.githubusercontent.com/HoGentTIN/dsai-en-labs/main/data/
       number_of_heavily_wounded_car_accidents_BE.txt',
    delimiter = "\t",
    parse_dates=['date']).set_index(['date'])

# Geef expliciet aan dat het maandelijks is (MS = Month Start)
wounded_data.index = pd.DatetimeIndex(wounded_data.index, freq='MS')

wounded_data.head()
```

8.3 Time Series Models

8.3.1 Tabellen

- Om te bepalen welk model het is, moeten we de data even plotten
- Dit kan je op twee manieren doen:

```
wounded_ts = pd.Series(
    data=np.array(wounded_data['number_of_heavily_wounded']),
    index=wounded_data.index
)
wounded_ts.plot(marker='o');
```

```
wounded_data.plot( y='number_of_heavily_wounded', figsize=[10,5]);
```

8.3.2 Maken van voorspellingen

- We plotten eerst de volledige data is

```
number_of_samples = len(wounded_data['number_of_heavily_wounded'])

a, b = np.polyfit(np.arange(0, number_of_samples), wounded_data['
    number_of_heavily_wounded'].values, 1)

# plot the time series and the regression line
plt.plot(np.arange(0, number_of_samples), wounded_data['
    number_of_heavily_wounded'])
plt.axline((0, b), slope=a)
```

- Voorspellingen maken met deze regressielijn:

```
# voorspelling januari 2024
print(f'voorspelling aantal zwaargewonden in januari 2024 = {a *
    number_of_samples + b}')

# februari 2024 (+1 maand)
print(f'voorspelling aantal zwaargewonden in februari 2024 = {a * (
    number_of_samples + 1) + b}')

# maand 2024 (+2 maand)
print(f'voorspelling aantal zwaargewonden in maart 2024 = {a * (
    number_of_samples + 2) + b}')
```

- Regressielijn met seaborn:

```
sns.regplot(
    x=np.arange(0,number_of_samples),
    y=wounded_data['number_of_heavily_wounded'],
    line_kws={'color': 'b'},
    scatter_kws={'color':'g'});
```

8.4 Moving average

8.4.1 Simple Moving Average

- Met deze functies kijken we zoveel waarden in de toekomst (SMA = Simple Moving Average)
- Indien je None waarden hebt voeg dan de parameter `min_periods=1` toe in `rolling`.
- Voortschrijdend gemiddelde over een vast aantal vorige maanden.

```
wounded_data['SMA3'] = wounded_data['number_of_heavily_wounded'].rolling(3).
    mean()
wounded_data['SMA5'] = wounded_data['number_of_heavily_wounded'].rolling(5).
    mean()
```

```
wounded_data['SMA10'] = wounded_data['number_of_heavily_wounded'].rolling(10).
    mean()
wounded_data
```

- En dit plotten we zo:

```
wounded_data.plot(
    y=['number_of_heavily_wounded', 'SMA3', 'SMA5', 'SMA10'],
    figsize=[10,5]);
```

8.4.2 Exponentional Moving Average (EMA) of Simple Exponential Smoothing (SES)

- Pandas heeft een ingebouwde functie om dit te berekenen
- Hier krijgt de meest recente waarneming meer gewicht dan oudere.

```
wounded_data['EMA_0.1'] = wounded_data['number_of_heavily_wounded'].ewm(alpha
    =.1, adjust=False).mean()
wounded_data['EMA_0.5'] = wounded_data['number_of_heavily_wounded'].ewm(alpha
    =.5, adjust=False).mean()

wounded_data.plot(y=['number_of_heavily_wounded',  'EMA_0.1', 'EMA_0.5'],
    figsize=[10,5]);
```

- SES berekenen met Holt Winters (zelfde effect, meer controle)

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing

# Train the model
data_ses = SimpleExpSmoothing(wounded_data['number_of_heavily_wounded']).fit(
    smoothing_level=0.1, optimized=False)

# Add fitted values to the dataframe
wounded_data['SES_predicted_level'] = data_ses.level
wounded_data.head(20)
```

- En dan platten we weer:

```
wounded_data.plot(y=['number_of_heavily_wounded',  'SES'], figsize=[10,5]);
```

- Hierdoor kunnen we voorspellingen doen via `forecast`:

```
data_ses_fcast = data_ses.forecast(12)

wounded_data.plot(y=['number_of_heavily_wounded',  'SES'], figsize=[10,5])
data_ses_fcast.plot(marker='.', legend=True, label='Forecast');
```

8.4.3 Double exponetional Smoothing (Holt's methode)

- Voegt een trendcomponent toe aan SES:n

```
from statsmodels.tsa.api import Holt

data_des = Holt(wounded_data['number_of_heavily_wounded']).fit(
    smoothing_level=.1,
    smoothing_trend=.2,
    optimized=False)
```

```
wounded_data['DES'] = data_des.level
wounded_data.tail()
```

- Dit plotten we weer

```
wounded_data.plot(y=['number_of_heavily_wounded',  'DES'], figsize=[10,5]);
```

- Dit kan wederom worden geforcasted:

```
data_des_fcast = data_des.forecast(12)

# Plot observations, fitted values and forecast
wounded_data['number_of_heavily_wounded'].plot(marker='o', legend=True)
wounded_data['DES'].plot(legend=True, label='DES fitted values', figsize
   =[10,5]);
data_ses_fcast.plot(marker='.', legend=True, label='Forecast SES');
data_des_fcast.plot(marker='.', legend=True, label='Forecast DES');

wounded_data['DES_prediction_next_month'] = data_des.level + data_des.trend #
   predicted value for next month => m = 1
wounded_data['DES_fittedvalues'] = data_des.fittedvalues
wounded_data.tail()
```

8.4.4 Triple exponentional Smoothing (Holt-Winters methode)

- Voegt ook seizoenspatronen toe:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

train = wounded_data.number_of_heavily_wounded[:-12]
test = wounded_data.number_of_heavily_wounded[-12:]

wounded_hw = ExponentialSmoothing(train, trend='add', seasonal='add',
   seasonal_periods=12, freq='MS').fit()

train.plot(legend=True, label='train')
test.plot(legend=True, label='test')
(wounded_hw.level + wounded_hw.season).plot(legend=True, label='fitted');
```

```
wounded_predicted = wounded_hw.forecast(12)

train.plot(legend=True, label='train')
wounded_hw.fittedvalues.plot(legend=True, label='fitted')

test.plot(legend=True, label='test')
wounded_predicted.plot(legend=True, label='predicted')

plt.title('Train, test, fitted & predicted values using Holt-Winters');
```

```
test.plot(legend=True, label='test');
wounded_predicted.plot(legend=True, label='predicted');
plt.title('Test data vs. predicted values');
```

```
# Predicted value by forecast():
print(wounded_predicted.iloc[0])
```

```
# Computed manually from model internals:
(wounded_hw.level.iloc[-1] + wounded_hw.trend.iloc[-1]) + wounded_hw.season.
    iloc[-12]
```

- Evaluatie:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

print(f'MAE = {mean_absolute_error(test,wounded_predicted)}')
print(f'MSE = {mean_squared_error(test,wounded_predicted)}')
print(f'Square root MSE  = {np.sqrt(mean_squared_error(test,wounded_predicted))
    }')
print(f'stdev = {wounded_data.number_of_heavily_wounded.std()}')
```

8.5 Decomposing a Time Series

- Om elke component (trend, seizoen, residu) apart te analyseren:

```
from statsmodels.tsa.seasonal import seasonal_decompose

# remove other columns (e.g. SES, DES, ...)
data = wounded_data[['number_of_heavily_wounded']]

wounded_decomposed = seasonal_decompose(data, model='additive')
wounded_decomposed.plot();
```

8.6 Samenvatting

Methode	Wat het doet	Sterkte
SMA	Gemiddelde over vaste venster	Simpel, maar gevoelig voor outliers
EMA / SES	Meer gewicht aan recente waarden	Sneller reagerend op veranderingen
Holt (DES)	Trend erbij	Voorspelt stijging/daling
Holt-Winters	Trend + seizoen	Beste voor complexe patronen

Vraag	Antwoord	Model(s) die passen
Is er een **duidelijke trend**?	Ja	Holt (Double Exponential Smoothing), Regressie
Is er een **seizoenspatroon**?	Ja	Holt-Winters (Triple Exponential Smoothing)
Zijn er **geen duidelijke patronen**?	Ja	Simple/Exponential Moving Average, SES
Is de data **sterk fluctuerend**?	Ja	EMA of SES (geeft gewicht aan recente waarden)
Is het **ruis met kleine schommelingen**?	Ja	SMA of SES (voor gladstrijken)