03_importer_data

October 12, 2025

[1]:
```
# ============================================================================
# GOOGLE COLAB SETUP / GOOGLE COLAB SETUP
# ============================================================================

# Sjekk om vi kjører i Google Colab
try:
    import google.colab
    IN_COLAB = True
    print("  Kjører i Google Colab - installerer avhengigheter...")
    print("  Running in Google Colab - installing dependencies...")

    # Installer nødvendige pakker
    import subprocess
    import sys
    try:
        subprocess.check_call([sys.executable, "-m", "pip", "install", "-q",
                               "networkx", "matplotlib", "plotly", "pydantic",
                               "pyyaml", "pandas", "ipywidgets", "pillow",␣
 ↪"kaleido"])
        print("  Pakker installert")
    except Exception as e:
        print(f"  Pip install feilet: {e}")

    # Fjern eksisterende slektstre-mappe hvis den finnes
    import shutil
    import os
    if os.path.exists('/content/slektstre'):
        shutil.rmtree('/content/slektstre')
        print("  Fjernet eksisterende slektstre-mappe")

    # Klon repository
    try:
        subprocess.check_call(['git', 'clone', 'https://github.com/arvidl/
 ↪slektstre.git'])
        print("  Repository klonet")
    except Exception as e:
        print(f"  Git clone feilet: {e}")
```

```
# Legg til src-mappen til Python path og importer direkte
sys.path.insert(0, '/content/slektstre/src')
print("  Path lagt til")

# Importer slektstre-modulene direkte for å unngå navnekonflikt
import importlib.util
import types

# Først, fjern konfliktende moduler fra sys.modules
modules_to_remove = ['tree', 'models', 'localization']
for module_name in modules_to_remove:
    if module_name in sys.modules:
        del sys.modules[module_name]

# Last inn models.py først
try:
    spec = importlib.util.spec_from_file_location("slektstre_models", "/
↪content/slektstre/src/models.py")
    slektstre_models = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(slektstre_models)

    # Opprett midlertidig models modul
    temp_models_module = types.ModuleType('models')
    temp_models_module.Person = slektstre_models.Person
    temp_models_module.Gender = slektstre_models.Gender
    temp_models_module.Ekteskap = slektstre_models.Ekteskap
    temp_models_module.FamilieData = slektstre_models.FamilieData
    sys.modules['models'] = temp_models_module

    print("  models.py lastet")
except Exception as e:
    print(f"  models.py feilet: {e}")

# Last inn localization.py
try:
    spec = importlib.util.spec_from_file_location("slektstre_localization",␣
↪"/content/slektstre/src/localization.py")
    slektstre_localization = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(slektstre_localization)

    # Opprett midlertidig localization modul
    temp_localization_module = types.ModuleType('localization')
    temp_localization_module.t = slektstre_localization.t
    sys.modules['localization'] = temp_localization_module

    print("  localization.py lastet")
```

```
    except Exception as e:
        print(f"  localization.py feilet: {e}")

    # Last inn tree.py som slektstre_tree
    try:
        spec = importlib.util.spec_from_file_location("slektstre_tree", "/
↪content/slektstre/src/tree.py")
        slektstre_tree = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(slektstre_tree)

        # Opprett midlertidig tree modul
        temp_tree_module = types.ModuleType('tree')
        temp_tree_module.Slektstre = slektstre_tree.Slektstre
        sys.modules['tree'] = temp_tree_module

        print("  tree.py lastet")
    except Exception as e:
        print(f"  tree.py feilet: {e}")

    # Last inn family_io.py
    try:
        spec = importlib.util.spec_from_file_location("slektstre_io", "/content/
↪slektstre/src/family_io.py")
        slektstre_io = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(slektstre_io)
        print("  family_io.py lastet")
    except Exception as e:
        print(f"  family_io.py feilet: {e}")

    # Last inn visualization.py
    try:
        spec = importlib.util.spec_from_file_location("slektstre_viz", "/
↪content/slektstre/src/visualization.py")
        slektstre_viz = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(slektstre_viz)
        print("  visualization.py lastet")
    except Exception as e:
        print(f"  visualization.py feilet: {e}")

    print("  Slektstre-moduler lastet inn i Colab")

except ImportError:
    IN_COLAB = False
    print("  Kjører lokalt / Running locally")
    import sys
    sys.path.append('../src')
except Exception as e:
```

```
    print(f"  Colab setup feilet: {e}")
    IN_COLAB = False
    print("  Fallback til lokal modus / Fallback to local mode")
    import sys
    sys.path.append('../src')

print(f"  Miljø: {'Google Colab' if IN_COLAB else 'Lokal'}")
print(f"  Environment: {'Google Colab' if IN_COLAB else 'Local'}")
```

```
Kjører lokalt / Running locally
Miljø: Lokal
Environment: Local
```

1 Import/eksport av data

I denne notebooken lærer du hvordan du importerer og eksporterer familie-data i forskjellige formater.

1.1 Støttede formater

- **YAML** (anbefalt) - Lesbar struktur
- **JSON** - Universell kompatibilitet

- **CSV** - Enkel tabellstruktur
- **GEDCOM** - Genealogi-standard

[2]:
```
# Importer nødvendige biblioteker
import matplotlib.pyplot as plt
import pandas as pd
from datetime import date

# Importer slektstre-moduler (fungerer både lokalt og i Colab)
if IN_COLAB:
    # Bruk de modulene vi lastet inn i Colab-setup
    Person = slektstre_models.Person
    Gender = slektstre_models.Gender
    Ekteskap = slektstre_models.Ekteskap
    FamilieData = slektstre_models.FamilieData
    Slektstre = slektstre_tree.Slektstre
    load_from_yaml = slektstre_io.load_from_yaml
    save_to_yaml = slektstre_io.save_to_yaml
    load_from_json = slektstre_io.load_from_json
    save_to_json = slektstre_io.save_to_json
    load_from_csv = slektstre_io.load_from_csv
    save_to_csv = slektstre_io.save_to_csv
    export_to_gedcom = slektstre_io.export_to_gedcom
else:
    # Lokale imports
```

```
    import sys
    sys.path.append('../src')
    from models import Person, Ekteskap, FamilieData, Gender
    from tree import Slektstre
    from family_io import (
        load_from_yaml, save_to_yaml,
        load_from_json, save_to_json,
        load_from_csv, save_to_csv,
        export_to_gedcom
    )

print("  Alle biblioteker importert!")
```

```
 Alle biblioteker importert!
```

1.2 1. YAML Format (Anbefalt)

YAML er det mest lesbare formatet for familie-data:

[3]:
```
# Last eksempel-familie fra YAML
if IN_COLAB:
    familie_data = load_from_yaml('/content/slektstre/data/eksempel_familie.
 ↪yaml')
else:
    familie_data = load_from_yaml('../data/eksempel_familie.yaml')

slektstre = Slektstre(familie_data)

print(f"Familie lastet med {len(familie_data.personer)} personer og␣
 ↪{len(familie_data.ekteskap)} ekteskap")
print(f"Beskrivelse: {familie_data.beskrivelse}")

# Vis første person som eksempel
if familie_data.personer:
    første_person = familie_data.personer[0]
    print(f"\nEksempel person: {første_person.fullt_navn} (ID: {første_person.
 ↪id})")
```

```
Familie lastet med 17 personer og 5 ekteskap
Beskrivelse: Eksempel familie med 4 generasjoner - Lundervold familien

Eksempel person: Erik Lundervold (ID: p1)
```

[4]:
```
# Lagre til YAML
save_to_yaml(familie_data, "eksport_familie.yaml")
print("  Familie-data eksportert til eksport_familie.yaml")
```

```
 Familie-data eksportert til eksport_familie.yaml
```

1.3 2. JSON Format

JSON er godt for programmatisk bruk og kompatibilitet:

[5]:
```
# Eksporter til JSON
save_to_json(familie_data, "eksport_familie.json")
print(" Familie-data eksportert til eksport_familie.json")

# Last fra JSON
familie_data_json = load_from_json("eksport_familie.json")
slektstre_json = Slektstre(familie_data_json)

print(f"JSON-fil lastet med {len(familie_data_json.personer)} personer")
```

```
 Familie-data eksportert til eksport_familie.json
JSON-fil lastet med 17 personer
```

1.4 3. CSV Format

CSV er enkelt for tabellbasert data. La oss lage et eksempel:

[6]:
```
# Eksporter til CSV
save_to_csv(familie_data, "eksport_familie.csv")
print(" Familie-data eksportert til eksport_familie.csv")

# Vis CSV-innhold
df = pd.read_csv("eksport_familie.csv")
print(f"\nCSV-fil inneholder {len(df)} rader")
print("\nFørste 5 rader:")
print(df.head())
```

```
 Familie-data eksportert til eksport_familie.csv

CSV-fil inneholder 17 rader

Første 5 rader:
   id fornavn mellomnavn   etternavn   kjønn fødselsdato    dødsdato  \
0  p1    Erik        NaN  Lundervold    male  1920-03-15  1995-08-22
1  p2  Ingrid      Marie      Hansen  female  1925-07-10  2010-12-03
2  p3   Arvid        NaN  Lundervold    male  1950-05-20  2022-11-15
3  p4  Helena      Sofia  Lundervold  female  1952-09-12         NaN
4  p5   Bjørn        NaN  Lundervold    male  1955-01-08         NaN

    fødested dødssted  bilde_sti                       notater  \
0     Bergen     Oslo        NaN  Arbeidet som ingeniør på NSB
1  Trondheim     Oslo        NaN       Lærer og mor til 4 barn
2       Oslo   Bergen        NaN       Professor i informatikk
3       Oslo      NaN        NaN          Arkitekt og kunstner
4       Oslo      NaN        NaN        Lærer og fotballtrener
```

```
                                                historier foreldre  barn  partnere
0  Flyktet fra Norge under krigen|Bygde sitt eget…      NaN   NaN       NaN
1  Møtte Erik på dans i 1947|Spilte piano og sang…      NaN   NaN       NaN
2  Doktorgrad fra MIT|Grunnla flere teknologisels…    p1|p2   NaN       NaN
3  Designet flere kjente bygninger i Bergen|Malte…    p1|p2   NaN       NaN
4  Spilte fotball på høyt nivå i ungdommen|Trente…    p1|p2   NaN       NaN
```

[7]:
```python
# Last fra CSV
familie_data_csv = load_from_csv("eksport_familie.csv")
slektstre_csv = Slektstre(familie_data_csv)

print(f"CSV-fil lastet med {len(familie_data_csv.personer)} personer")
print(f"Antall ekteskap: {len(familie_data_csv.ekteskap)}")
```

```
CSV-fil lastet med 17 personer
Antall ekteskap: 5
```

1.5 4. GEDCOM Format

GEDCOM er standarden for genealogi-programmer:

[8]:
```python
# Eksporter til GEDCOM
export_to_gedcom(familie_data, "eksport_familie.ged")
print("  Familie-data eksportert til eksport_familie.ged")

# Vis første linjer av GEDCOM-filen
with open("eksport_familie.ged", "r", encoding="utf-8") as f:
    linjer = f.readlines()[:20]
    print("\nFørste 20 linjer av GEDCOM-filen:")
    for i, linje in enumerate(linjer, 1):
        print(f"{i:2d}: {linje.rstrip()}")
```

```
 Familie-data eksportert til eksport_familie.ged

Første 20 linjer av GEDCOM-filen:
 1: 0 HEAD
 2: 1 SOUR SLEKTSTRE
 3: 1 VERS 1.0
 4: 1 DATE 11 Oct 2025
 5: 1 CHAR UTF8
 6: 0 @FAM@ FAM
 7:
 8: 0 @p1@ INDI
 9: 1 NAME Erik /Lundervold/
10: 1 SEX M
11: 1 BIRT
12: 2 DATE 15 Mar 1920
13: 2 PLAC Bergen
```

```
14: 1 DEAT
15: 2 DATE 22 Aug 1995
16: 2 PLAC Oslo
17: 1 NOTE Arbeidet som ingeniør på NSB
18:
19: 0 @p2@ INDI
20: 1 NAME Ingrid /Hansen/
```

1.6 5. Sammenligning av formater

La oss sammenligne størrelsen og kompleksiteten:

[9]:
```
import os

# Sammenlign filstørrelser
filer = ["eksport_familie.yaml", "eksport_familie.json", "eksport_familie.csv",␣
 ↪"eksport_familie.ged"]

print("  Filstørrelser:")
for fil in filer:
    if os.path.exists(fil):
        størrelse = os.path.getsize(fil)
        print(f"{fil:25s}: {størrelse:6d} bytes")
    else:
        print(f"{fil:25s}: Ikke funnet")
```

```
  Filstørrelser:
eksport_familie.yaml     :   7532 bytes
eksport_familie.json     :  10974 bytes
eksport_familie.csv      :   2633 bytes
eksport_familie.ged      :   2654 bytes
```

1.7 6. Validering av importerte data

La oss sjekke at alle formater gir samme resultat:

1.8 GEDCOM-format

GEDCOM (GEnealogical Data COMmunication) er en internasjonal standard for utveksling av genealogiske data. Det er et tekstbasert format som brukes av de fleste genealogi-programmer.

1.8.1 GEDCOM-struktur

GEDCOM-filer består av hierarkiske linjer med følgende struktur:

`NIVÅ TAG [VERDI]`

Eksempler:

```
0 HEAD
1 SOUR SLEKTSTRE
```

```
1 VERS 1.0
1 DATE 15 DEC 2024

0 @p1@ INDI
1 NAME Erik /Lundervold/
2 GIVN Erik
2 SURN Lundervold
1 SEX M
1 BIRT
2 DATE 15 MAY 1920
2 PLAC Bergen, Norge
1 DEAT
2 DATE 10 JAN 1995
2 PLAC Oslo, Norge

0 @e1@ FAM
1 HUSB @p1@
1 WIFE @p2@
1 MARR
2 DATE 20 AUG 1978
2 PLAC Bergen, Norge
```

1.8.2 GEDCOM-tagger

Person-tagger: - `INDI` - Individ (person) - `NAME` - Navn - `GIVN` - Fornavn - `SURN` - Etternavn - `SEX` - Kjønn (M/F) - `BIRT` - Fødsel - `DEAT` - Død - `MARR` - Ekteskap - `DIV` - Skilsmisse

Familie-tagger: - `FAM` - Familie - `HUSB` - Ektemann - `WIFE` - Ektefelle - `CHIL` - Barn

Metadata-tagger: - `DATE` - Dato - `PLAC` - Sted - `NOTE` - Notater - `SOUR` - Kilde

1.8.3 Fordeler med GEDCOM

1. **Standardisert** - Fungerer med alle genealogi-programmer
2. **Portabel** - Enkelt å dele mellom systemer
3. **Komplett** - Støtter alle typer genealogiske data
4. **Lesbar** - Menneske-lesbart tekstformat

1.8.4 Eksport til GEDCOM

Vårt slektstre-program kan eksportere data til GEDCOM-format for kompatibilitet med andre genealogi-programmer som: - Ancestry.com - FamilySearch - MyHeritage - Gramps - Family Tree Maker

[10]:
```python
# Eksporter til GEDCOM-format
export_to_gedcom(familie_data, "eksport_familie.ged")
print("  Familie-data eksportert til eksport_familie.ged")

# Vis første del av GEDCOM-filen
print("\n  Første del av GEDCOM-filen:")
```

```
with open("eksport_familie.ged", "r", encoding="utf-8") as f:
    lines = f.readlines()
    for i, line in enumerate(lines[:20]):  # Vis første 20 linjer
        print(f"{i+1:2d}: {line.rstrip()}")

    if len(lines) > 20:
        print(f"... og {len(lines) - 20} linjer til")

print(f"\n GEDCOM-fil statistikk:")
print(f"Totalt antall linjer: {len(lines)}")
print(f"Fil størrelse: {os.path.getsize('eksport_familie.ged')} bytes")
```

```
 Familie-data eksportert til eksport_familie.ged

 Første del av GEDCOM-filen:
 1: 0 HEAD
 2: 1 SOUR SLEKTSTRE
 3: 1 VERS 1.0
 4: 1 DATE 11 Oct 2025
 5: 1 CHAR UTF8
 6: 0 @FAM@ FAM
 7:
 8: 0 @p1@ INDI
 9: 1 NAME Erik /Lundervold/
10: 1 SEX M
11: 1 BIRT
12: 2 DATE 15 Mar 1920
13: 2 PLAC Bergen
14: 1 DEAT
15: 2 DATE 22 Aug 1995
16: 2 PLAC Oslo
17: 1 NOTE Arbeidet som ingeniør på NSB
18:
19: 0 @p2@ INDI
20: 1 NAME Ingrid /Hansen/
… og 170 linjer til

 GEDCOM-fil statistikk:
Totalt antall linjer: 190
Fil størrelse: 2654 bytes
```

[11]:
```
# Sammenlign alle eksporterte formater
print(" Sammenligning av alle eksporterte formater:")
print("Format     Fil størrelse    Personer   Ekteskap")
print("=" * 50)

# YAML
```

```
yaml_size = os.path.getsize("eksport_familie.yaml")
print(f"YAML       {yaml_size:8d} bytes    {len(familie_data.personer):8d}  ␣
 ↪{len(familie_data.ekteskap):8d}")

# JSON
json_size = os.path.getsize("eksport_familie.json")
print(f"JSON       {json_size:8d} bytes    {len(familie_data.personer):8d}  ␣
 ↪{len(familie_data.ekteskap):8d}")

# CSV
csv_size = os.path.getsize("eksport_familie.csv")
csv_ekteskap_size = os.path.getsize("eksport_familie_ekteskap.csv")
print(f"CSV        {csv_size + csv_ekteskap_size:8d} bytes    {len(familie_data.
 ↪personer):8d}   {len(familie_data.ekteskap):8d}")

# GEDCOM
gedcom_size = os.path.getsize("eksport_familie.ged")
print(f"GEDCOM     {gedcom_size:8d} bytes    {len(familie_data.personer):8d}  ␣
 ↪{len(familie_data.ekteskap):8d}")

print("\n Tips:")
print("- YAML: Best for menneske-lesbarhet og redigering")
print("- JSON: Best for programmatisk bruk og API-er")
print("- CSV: Best for Excel og enkle dataanalyser")
print("- GEDCOM: Best for kompatibilitet med genealogi-programmer")
```

```
 Sammenligning av alle eksporterte formater:
Format     Fil størrelse    Personer   Ekteskap
================================================
YAML           7532 bytes          17          5
JSON          10974 bytes          17          5
CSV            3041 bytes          17          5
GEDCOM         2654 bytes          17          5

 Tips:
- YAML: Best for menneske-lesbarhet og redigering
- JSON: Best for programmatisk bruk og API-er
- CSV: Best for Excel og enkle dataanalyser
- GEDCOM: Best for kompatibilitet med genealogi-programmer
```

[12]:
```
# Sammenlign antall personer og ekteskap
formater = {
    "YAML": familie_data,
    "JSON": familie_data_json,
    "CSV": familie_data_csv
}
```

```
print("  Sammenligning av importerte data:")
print(f"{'Format':<8} {'Personer':<10} {'Ekteskap':<10}")
print("-" * 30)

for format_navn, data in formater.items():
    print(f"{format_navn:<8} {len(data.personer):<10} {len(data.ekteskap):<10}")

# Sjekk at alle har samme antall personer
antall_personer = [len(data.personer) for data in formater.values()]
if len(set(antall_personer)) == 1:
    print("\n  Alle formater har samme antall personer!")
else:
    print("\n  Formater har forskjellig antall personer")
```

```
  Sammenligning av importerte data:
Format   Personer   Ekteskap
------------------------------
YAML     17         5
JSON     17         5
CSV      17         5

  Alle formater har samme antall personer!
```

1.9 7. Rydde opp

La oss slette de midlertidige filene:

[13]:
```
# Slett midlertidige filer
import os

filer_til_sletting = [
    "eksport_familie.yaml",
    "eksport_familie.json",
    "eksport_familie.csv",
    "eksport_familie.ged"
]

for fil in filer_til_sletting:
    if os.path.exists(fil):
        os.remove(fil)
        print(f"  Slettet {fil}")

print("\n  Opprydding fullført!")
```

```
  Slettet eksport_familie.yaml
  Slettet eksport_familie.json
  Slettet eksport_familie.csv
  Slettet eksport_familie.ged
```

```
Opprydding fullført!
```

1.10 Oppsummering

I denne notebooken har du lært:

1. **YAML** - Lesbar struktur, anbefalt format
2. **JSON** - Programmatisk kompatibilitet
3. **CSV** - Enkel tabellstruktur
4. **GEDCOM** - Genealogi-standard
5. Sammenligning av formater
6. Validering av importerte data
7. Opprydding av midlertidige filer

Anbefalinger: - Bruk **YAML** for manuell redigering - Bruk **JSON** for programmatisk bruk - Bruk **CSV** for enkel dataoverføring - Bruk **GEDCOM** for kompatibilitet med andre genealogi-programmer

Neste steg: Gå til `04_visualisering.ipynb` for å utforske alle visualiseringsalternativer.