In [1]:
```python
%load_ext autoreload
%autoreload 2
import pandas as pd
import polars as pl
import os
import numpy as np
from source.config import INTERIM_DATA_DIR, RAW_DATA_DIR
from source.features_dir.estimated_registrations import table
from source.features_dir.estimated_registrations import THRESHOLD_KM_REGISTRATION_RADIUS_FROM_COORDINATE_POINT
from source.features_dir.estimated_registrations import THRESHOLD_HOUR_AVOID_COUNTING_DUPLICATE_REGISTRATIONS
```

```
2025-02-17 23:56:50.922 | INFO     | source.config:<module>:13 - PROJ_ROOT path is: /home/anders/engasjement_svv
```

In [2]:
```python
df_truck = pd.read_csv(INTERIM_DATA_DIR / 'estimated_registrations' / 'processed-truck_only.csv')
```

In [3]:
```python
INDEX_DATE = 1
INDEX_N_AXLES = 7

def load_dfs(location: str) -> pd.DataFrame:
    df_bwim =  pl.concat([
        pl.read_csv(
            RAW_DATA_DIR / 'BWIM' / location / f,
            has_header=False,
            truncate_ragged_lines=True,
            ignore_errors=True,
            separator=';',
            decimal_comma=True
        ) for f in os.listdir(RAW_DATA_DIR / 'BWIM' / location) if f.endswith('.csv')
    ], how='diagonal_relaxed').to_pandas()
    df_bwim['datetime'] = pd.to_datetime(df_bwim.iloc[:, INDEX_DATE], format='%Y-%m-%d-%H-%M-%S-%f', errors='coerce')
    df_bwim['date'] = df_bwim['datetime'].dt.date
    df_bwim['n_axles'] = df_bwim.iloc[:, INDEX_N_AXLES]
    df_bwim['vehicle_length'] = df_bwim.apply(
        lambda row: float(str(row[10 + row['n_axles']]).replace(',', '.')) if row[10 + row['n_axles']] is not None else np.nan,
        axis=1
    )
    df_bwim = df_bwim[df_bwim['vehicle_length'] >= 16]
    return df_bwim
```

In [4]:
```python
df_bwim_tangensvingen = load_dfs('tangensvingen')
valid_dates_tangensvingen = df_bwim_tangensvingen['date'].unique()

df_bwim_sørbryn = load_dfs('sørbryn')
valid_dates_sørbryn = df_bwim_sørbryn['date'].unique()
```

```
/tmp/ipykernel_42946/2245160203.py:19: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent w
ith DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  lambda row: float(str(row[10 + row['n_axles']]).replace(',', '.')) if row[10 + row['n_axles']] is not None else np.nan,
/tmp/ipykernel_42946/2245160203.py:19: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent w
ith DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  lambda row: float(str(row[10 + row['n_axles']]).replace(',', '.')) if row[10 + row['n_axles']] is not None else np.nan,
```

In [5]:
```python
coordinates = {
    'tangensvingen': (60.89378600721336, 11.576611253561099),
    'sørbryn': (60.772323376282074, 11.308699373298074)
}
```

In [6]:
```python
table_tangensvingen = table(
    df=df_truck,
    road_coordinates=coordinates,
    threshold_radius_km=THRESHOLD_KM_REGISTRATION_RADIUS_FROM_COORDINATE_POINT,
    threshold_time_hours=THRESHOLD_HOUR_AVOID_COUNTING_DUPLICATE_REGISTRATIONS,
    subpath='bwim74t',
    valid_dates=valid_dates_tangensvingen
).query('Vei == "tangensvingen"')

table_sørbryn = table(
    df=df_truck,
    road_coordinates=coordinates,
    threshold_radius_km=THRESHOLD_KM_REGISTRATION_RADIUS_FROM_COORDINATE_POINT,
    threshold_time_hours=THRESHOLD_HOUR_AVOID_COUNTING_DUPLICATE_REGISTRATIONS,
    subpath='bwim74t',
    valid_dates=valid_dates_sørbryn
).query('Vei == "sørbryn"')
```

```
Processing roads:   0%|
| 0/2 [00:00<?, ?it/s]
10480
Processing roads:  50%|██████████████████████████████████████████████████████████████████████████                                                                         | 1/2 [0
0:01<00:01,  1.68s/it]
10704
Processing roads:   0%|
| 0/2 [00:00<?, ?it/s]
10480
Processing roads:  50%|██████████████████████████████████████████████████████████████████████████                                                                         | 1/2 [0
0:01<00:01,  1.39s/it]
10704
rocessing roads: 100%|████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 2/2 [0
0:02<00:00,  1.08s/it]
```

In [7]:
```python
def create_road_registrations(df_bk74: pd.DataFrame, df_bwim) -> pd.DataFrame:
    # 74 er filtrert til å kun ta dagene BWIM sensorene var i drift
    data, columns = [], ['År', 'Registreringer BK74', 'Registreringer BWIM', 'Prosent BK74 av BWIM']
    for year in [2022, 2023, 2024]:
        registrations_year_bk74 = sum([int(df_bk74[f'{year} {tonnage}t']) for tonnage in [60, 65, 68, 74]])
        registrations_year_bwim = int(len(df_bwim[df_bwim['datetime'].dt.year == year]))
        percentage_bk74_bwim = registrations_year_bk74 / registrations_year_bwim * 100 if registrations_year_bwim > 0 else 0
        data.append([year, registrations_year_bk74, registrations_year_bwim, percentage_bk74_bwim])
    return pd.DataFrame(data=data, columns=columns)
```

In [8]:
```python
create_road_registrations(table_tangensvingen, df_bwim_tangensvingen)
```

```
/tmp/ipykernel_42946/1388232334.py:5: FutureWarning: Calling int on a single element Series is deprecated and will raise a TypeError in the future. Use int(ser.iloc[0]) instead
  registrations_year_bk74 = sum([int(df_bk74[f'{year} {tonnage}t']) for tonnage in [60, 65, 68, 74]])
```

Out[8]:

	År	Registreringer BK74	Registreringer BWIM	Prosent BK74 av BWIM
0	2022	18	508	3.543307
1	2023	137	1391	9.849029
2	2024	48	1152	4.166667

In [9]:
```python
create_road_registrations(table_sørbryn, df_bwim_sørbryn)
```

```
/tmp/ipykernel_42946/1388232334.py:5: FutureWarning: Calling int on a single element Series is deprecated and will raise a TypeError in the future. Use int(ser.iloc[0]) instead
  registrations_year_bk74 = sum([int(df_bk74[f'{year} {tonnage}t']) for tonnage in [60, 65, 68, 74]])
```

Out[9]:

	År	Registreringer BK74	Registreringer BWIM	Prosent BK74 av BWIM
0	2022	20	529	3.780718
1	2023	125	2629	4.754660
2	2024	111	1814	6.119074