In [1]:
```
%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]:
```
df_truck = pd.read_csv(INTERIM_DATA_DIR / 'estimated_registrations' / 'processed-truck_only.csv')
```

In [3]:
```
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]:
```
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]:
```
coordinates = {
    'tangensvingen': (60.89378600721336, 11.576611253561099),
    'sørbryn': (60.772323376282074, 11.308699373298074)
}
```

In [6]:
```
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]:
```
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]:
```
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]:
```
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

2 Premisser

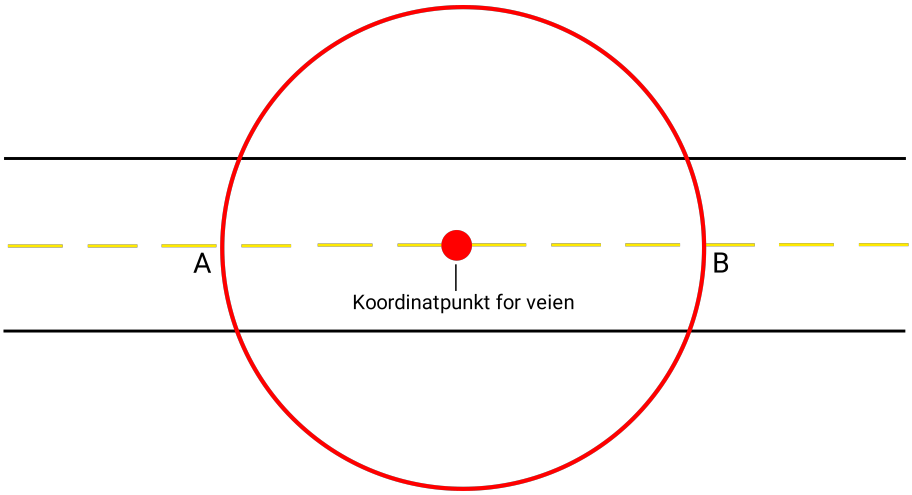
Datagrunnlag
Grunnlaget for denne besvarelsen er posisjonsdata innhentet i forbindelse med prøveordningen for økt tillatt totalvekt i tømmer/tungtransportindustrien. Denne dataen består av regelmessige logginger fra prøveordningens kjøretøy, hvor verdier for VIN, Organisasjonsnummer, Dato, Breddegrad, Lengdegrad, Høyde, Retning, Hastighet, Temperatur,Aktuell vekt, Akselvekt foran, Akselvekt bak, Drivstoffsnivå (%), Catalystnivå (%), Dekkmåling, Hard akselerasjon, Hard bremsing, og Hard sving er registrert. Til denne oppgaven er enkelte verdier mer relevante enn andre.

Begrensninger i dataen og tilnærminger for å overkomme disse
Intensjonen er som nevnt å sette opp en oversikt over antall passeringer prøveordningens kjøretøy har gjort på punktene som representerer veiene. Ettersom loggingen av data fra prøveordningen ikke har noen tilknytning til disse punktene og dermed ikke direkte er representativ for antallet passeringer, så er en mulig tilnærming å sette opp et område rundt vegstrekningen hvor punktene befinner seg, for deretter å sjekke hvor mange logginger som er foretatt innenfor disse områdene. Ved denne tilnærmingen står man imidlertid i fare for å telle den samme passeringen flere ganger; dersom kjøretøyet på den samme turen foretar flere logginger innen området, vil samtlige av disse kunne tatt med i tellingen, til tross for at det kun forekommer én passering. For å unngå dette, kan man basere seg på å kun telle logginger fra en gitt kombinasjon av kjøretøy og vei dersom det har gått en viss tid siden forrige logging for denne kombinasjonen. Effekten av denne tidsperioden er dermed at sekvensielle logginger ikke telles med flere ganger, og dermed at neste registrering trolig tilhører en annen passering. Intensjonen her er dermed å avgrense lengden på området til at et kjøretøy som passerer gjennom det etter all sannsynlighet har foretatt minst én logging i det. Et lengre område vil øke risikoen for at flere logginger innenfor området tilhørende samme passering kan bli telt som individuelle passeringer.

Dataen består av individuelle logginger på bilene og tilhengerne. Videre er det noe avvikk i loggingen fra bilene og deres respektive tilhengere, eksempelvis at enkelte biler har logget data i lengre perioder der deres tilhørende tilhenger ikke har noen logginger i det hele tatt. Dette skaper en komplikasjon rundt hvilke datapunkter man skal ta hensyn til: kun de som stammer fra bilene, kun tilhengerne, eller der det er registrert for både bil og tilhenger? Ettersom bilene har mulighet til å kjøre uten tilhengerne, men ikke motsatt, så er det her hovedsaklig tatt utgangspunkt i datapunktene tilhørende tilhengerne. Dermed er plots, statistikk, og argumentasjon for premisser basert på dataen fra tilhengerne. Resultat-seksjonen inneholder dog også en tabell basert på dataen logget utelukkende på bilene, som er beregnet på samme måte som for tilhengerne.

2.1 Lengde og bredde på området for registrering

Området rundt det konkrete koordinatpunktet en vei er definert på som benyttes til å registrere passeringer må være stor nok til at tidsintervallene mellom loggingene ikke fører til at et kjøretøy som kjører på en av veiene av interesse har mulighet til å passere gjennom hele dette området uten at det forekommer minst én logging. Figur 1 forsøker å illustrere dette: Et kjøretøy som passerer koordinatpunktet en vei er definert på ender med å logge data på punkter A og B, som et resultat av et gitt tidsintervall mellom loggingene og en gitt hastighet. Dersom området rundt koordinatpunktet til veien er for liten til å fange opp noen av disse registreringene, vil passeringen ikke bli registrert. Presisjonen til posisjonsloggingene er av såpass god kvalitet at bredde på området ikke er et problem, og bredde er dermed satt til 50 meter.

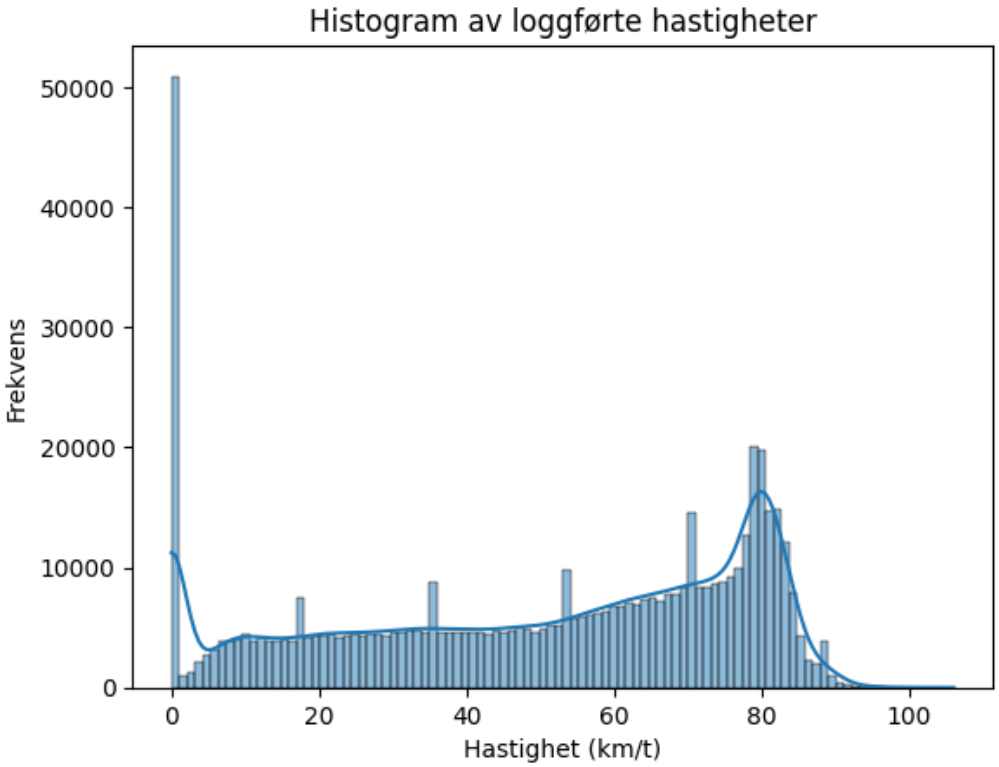


Figur 1: *Illustrasjon brukt for å begrunne valg av lengde av område for registreringer*

Forholdet mellom strekning s, hastighet v, og tid t, $s = v \times t$, tilsier dermed at lengden s som en funksjon av kjøretøyets hastighet v og tidsintervallet Δt mellom to påfølgende logginger må (eller hvertfall burde) være $s \geq (v \times \Delta t)/2$ i hver retning fra punktet.

Hastighet

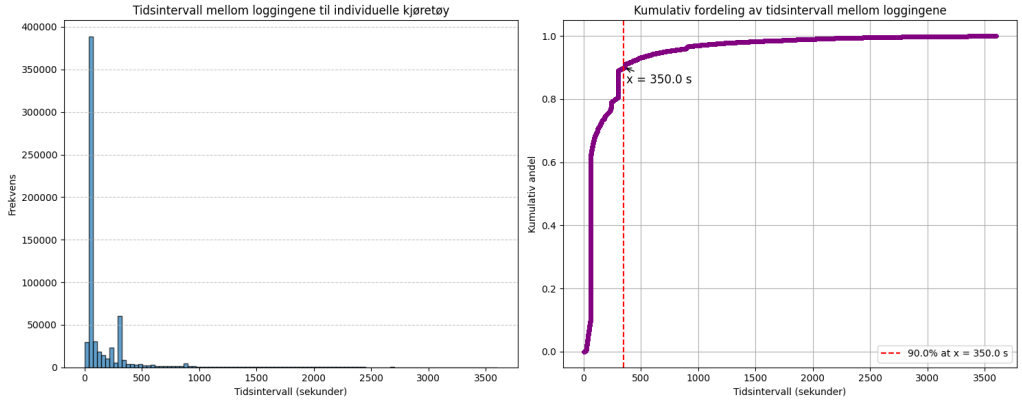
Desto større verdi for hastigheten v man plugger inn i formelen, desto lengre område får man, og desto flere logginger kan bli registrert som en passering av en vei. Ved å sette inn en hastighet som representerer det øvre sjiktet av hastighetene som er loggført, legger man dermed til rette for at de fleste reelle passeringer av koordinatpunktet til en vei kan bli registrert. Som man ser i Figur 2 (som viser fordelingen av alle loggførte hastigheter i datasettet) er det tilnærmet ingen registrerte hastigheter over **90 km/t**, så det er denne verdien som har blitt brukt for hastighet.



Figur 2: *Histogram over loggførte hastigheter*

Tidsintervall

Tidsintervallene mellom loggingene spiller også en sentral rolle i å bestemme lengden av området. Et større tidsintervall fører til en større distanse mellom loggingene, hvilket igjen forutsetter et lengre område for å fange opp passeringer. I likhet med verdien for hastighet, så vil en verdi fra det øvre sjiktet av tidsintervall mellom logginger føre til et område som fanger opp de fleste reelle passeringer. Figur 3 viser i venstre subplot et histogram over fordelingen av tidsintervallene mellom etterfølgende logginger innad i samme kjøretur for individuelle kjøretøy. Høyre subplot viser den samme dataen som en kumulativ fordeling, hvor det også kommer tydelig frem man kan ta hensyn til majoriteten av tidsintervallene (90%) ved å basere seg på å benytte en verdi av **rundt 6 minutter** (350 sekunder). Dermed har denne verdien blitt brukt som input for tidsintervallene.



Figur 3: *Histogram og kumulativ fordeling av tidsintervallene mellom etterfølgende logginger*

Med verdier tilhørende det øvre sjiktet av hastighet og tidsintervaller, så burde det resulterende området være romslig nok til å fange opp de aller fleste reelle passeringer. Plugger man inn verdiene for hastighet og tidsintervall, ender man dermed med en lengde på $s = (v \times \Delta t)/2 = (90km/t \times (350s/60^2))/2 = 4.375km$ i hver retning fra punktet. Imidlertid er det flere av punktene som ligger nærmere enn dette til vegkryss, hvor det følgelig ikke lenger er mulig å vite hvor kjøretøyet har kjørt. Dermed stopper området opp ved det første inntreffende av veikryss og 4.375km i hver retning av punktet.