Explore_Data_EH

May 18, 2024

[1]:
```
import pandas as pd
import altair as alt
import matplotlib.pyplot as plt
import datetime
from pathlib import Path

TABLE_ID = '80072ned'
TABLE_PATH = Path(f'./data/{TABLE_ID}')
GRAPH_PATH = Path(f'./plots/{TABLE_ID}_{datetime.datetime.now().
 ↪strftime("%Y%m%dT%H%M%S")}')

GRAPH_PATH.mkdir(exist_ok=True, parents=True)

def get_figure_number():
    figure_number = 0
    while True:
        yield figure_number
        figure_number += 1
figure_number_generator = get_figure_number()

def get_graph_name(title: str) -> str:
    return f'{next(figure_number_generator):0=3d}_{title}'

def save_graph(graph: alt.Chart, title: str):
    graph_name = get_graph_name(title)
    graph.save(GRAPH_PATH / f'{graph_name}.svg')
    graph.save(GRAPH_PATH / f'{graph_name}.png')

def save_fig(figure, title: str):
    graph_name = get_graph_name(title)
    figure.savefig(GRAPH_PATH / f'{graph_name}.svg')
    figure.savefig(GRAPH_PATH / f'{graph_name}.png')

# required for export to pdf with images?
alt.renderers.enable('png')
```

[1]:
```
RendererRegistry.enable('png')
```

1 UWV Exploratory Analysis

[2]:
```
slp: pd.DataFrame = pd.read_parquet(TABLE_PATH / f'{TABLE_ID}.parquet')
slp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5460 entries, 0 to 5459
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     5460 non-null   int64
 1   sbi                    5460 non-null   category
 2   period                 5460 non-null   category
 3   sick_leave_percentage  5150 non-null   float64
 4   period_title           5460 non-null   category
 5   period_status          5460 non-null   category
 6   period_year            5460 non-null   int64
 7   period_type            5460 non-null   category
 8   period_quarter_number  5460 non-null   int64
 9   period_quarter         5460 non-null   int64
 10  sbi_title              5460 non-null   category
 11  sbi_description        5460 non-null   category
 12  category_group_id      5460 non-null   int64
 13  category_group_title   5460 non-null   category
dtypes: category(8), float64(1), int64(5)
memory usage: 324.2 KB
```

[3]:
```
slp
```

[3]:

	id	sbi	period	sick_leave_percentage	period_title	\
0	0	T001081	1996KW01	5.5	1996 1e kwartaal	
1	1	T001081	1996KW02	4.6	1996 2e kwartaal	
2	2	T001081	1996KW03	4.0	1996 3e kwartaal	
3	3	T001081	1996KW04	4.7	1996 4e kwartaal	
4	4	T001081	1996JJ00	4.7	1996	
…	…	…	…	…	…	
5455	5455	WP19098	2023KW01	6.5	2023 1e kwartaal	
5456	5456	WP19098	2023KW02	5.7	2023 2e kwartaal	
5457	5457	WP19098	2023KW03	5.5	2023 3e kwartaal	
5458	5458	WP19098	2023KW04	6.4	2023 4e kwartaal	
5459	5459	WP19098	2023JJ00	6.0	2023	

	period_status	period_year	period_type	period_quarter_number	\
0	Definitief	1996	KW	1	
1	Definitief	1996	KW	2	
2	Definitief	1996	KW	3	
3	Definitief	1996	KW	4	
4	Definitief	1996	JJ	0	

```
…            …          …          …                    …
5455     Voorlopig         2023          KW                    1
5456     Voorlopig         2023          KW                    2
5457     Voorlopig         2023          KW                    3
5458     Voorlopig         2023          KW                    4
5459     Voorlopig         2023          JJ                    0

      period_quarter                          sbi_title  \
0              19961  A-U Alle economische activiteiten
1              19962  A-U Alle economische activiteiten
2              19963  A-U Alle economische activiteiten
3              19964  A-U Alle economische activiteiten
4              19960  A-U Alle economische activiteiten
…              …                                  …
5455           20231      100 of meer werkzame personen
5456           20232      100 of meer werkzame personen
5457           20233      100 of meer werkzame personen
5458           20234      100 of meer werkzame personen
5459           20230      100 of meer werkzame personen

                                       sbi_description  category_group_id  \
0     Alle economische activiteiten \r\nDeze categor…                  1
1     Alle economische activiteiten \r\nDeze categor…                  1
2     Alle economische activiteiten \r\nDeze categor…                  1
3     Alle economische activiteiten \r\nDeze categor…                  1
4     Alle economische activiteiten \r\nDeze categor…                  1
…                                                  …                 …
5455  Het aantal "werkzame personen" bestaat uit: \r…                  5
5456  Het aantal "werkzame personen" bestaat uit: \r…                  5
5457  Het aantal "werkzame personen" bestaat uit: \r…                  5
5458  Het aantal "werkzame personen" bestaat uit: \r…                  5
5459  Het aantal "werkzame personen" bestaat uit: \r…                  5

     category_group_title
0                  Totaal
1                  Totaal
2                  Totaal
3                  Totaal
4                  Totaal
…                   …
5455      Bedrijfsgrootte
5456      Bedrijfsgrootte
5457      Bedrijfsgrootte
5458      Bedrijfsgrootte
5459      Bedrijfsgrootte

[5460 rows x 14 columns]
```

[4]:
```
slp.category_group_title.value_counts()
```

[4]:
```
category_group_title
Bedrijfstak         2520
Bedrijfsklasse      1820
Bedrijfssector       560
Bedrijfsgrootte      420
Totaal               140
Name: count, dtype: int64
```

1.1 Get train and test

We will use 2022 and up as the final test data. All prior tot 2022 will be training data. To test the trained model, we wil use 2021. So we get three splits:

- All data from 2013 onward and prior to 2021 is the real train data. This is the data to perform exploratory data analysis on.
- All data from 2021 wil be the test set to test our trained models on.
- When we are really done, 2022 and onwards will be the final test set.

Additionally, we will only use the quarterly numbers (period_type = 'KW')

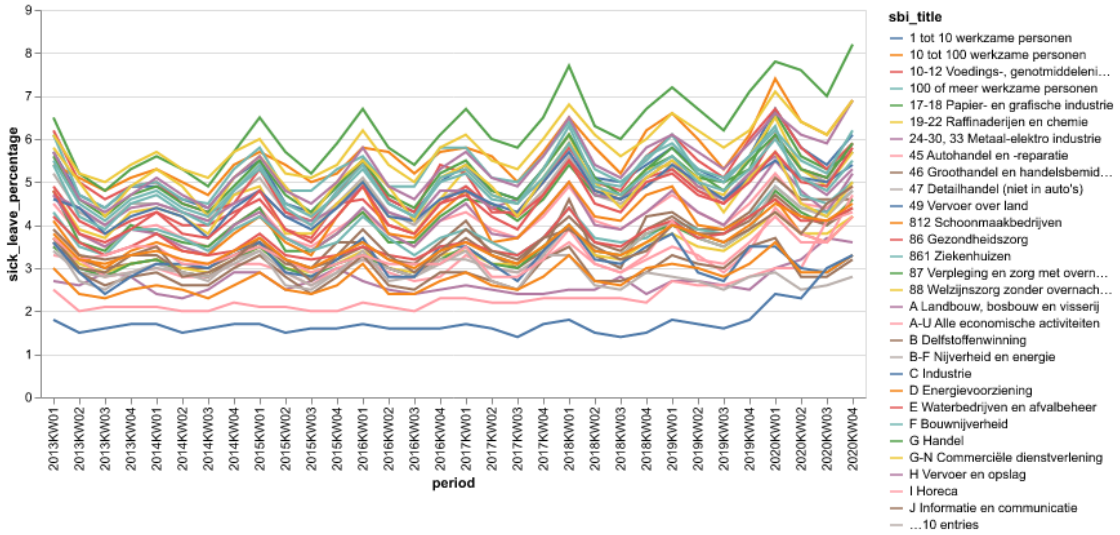
[5]:
```
slp_test = slp[(slp.period_year >= 2022) & (slp.period_type == 'KW')]
slp_train = slp[(slp.period_year > 2012) & (slp.period_year < 2021) & (slp.
 ↪period_type == 'KW')]
slp_train_test = slp[(slp.period_year == 2021) & (slp.period_type == 'KW')]
```

[6]:
```
slp_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1248 entries, 85 to 5443
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     1248 non-null   int64
 1   sbi                    1248 non-null   category
 2   period                 1248 non-null   category
 3   sick_leave_percentage  1248 non-null   float64
 4   period_title           1248 non-null   category
 5   period_status          1248 non-null   category
 6   period_year            1248 non-null   int64
 7   period_type            1248 non-null   category
 8   period_quarter_number  1248 non-null   int64
 9   period_quarter         1248 non-null   int64
 10  sbi_title              1248 non-null   category
 11  sbi_description        1248 non-null   category
 12  category_group_id      1248 non-null   int64
 13  category_group_title   1248 non-null   category
dtypes: category(8), float64(1), int64(5)
```

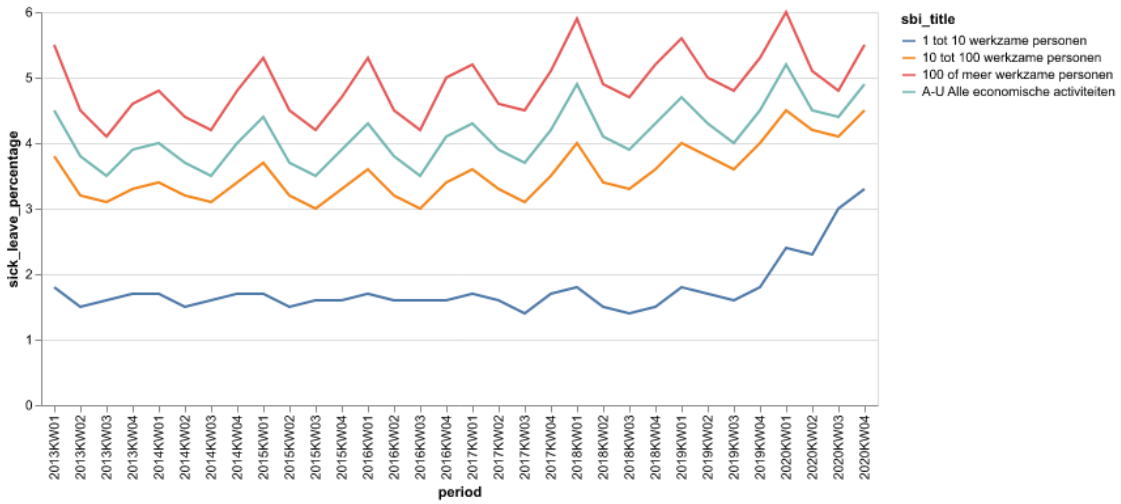
```
memory usage: 95.3 KB
```

[7]:
```
c = alt.Chart(slp_train).mark_line().encode(
    x='period',
    y='sick_leave_percentage',
    color='sbi_title'
)

save_graph(c, 'slp_train_all_sbi')
c
```

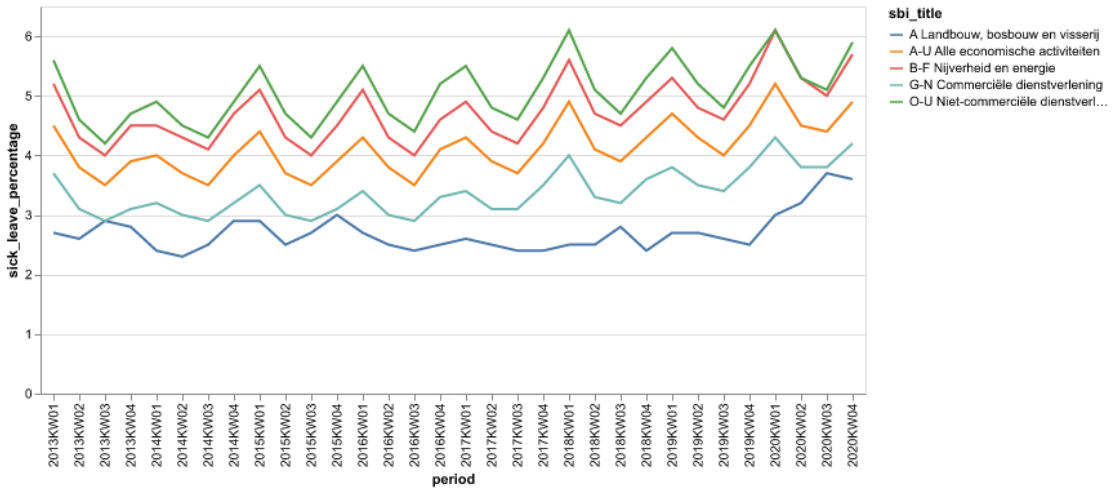
[7]:


[8]:
```
c = alt.Chart(slp_train[slp_train.category_group_title.isin(['Bedrijfsgrootte',␣
↪'Totaal'])]).mark_line().encode(
    x='period',
    y='sick_leave_percentage',
    color='sbi_title'
)

save_graph(c, 'slp_train_sbi_size')
c
```

[8]:



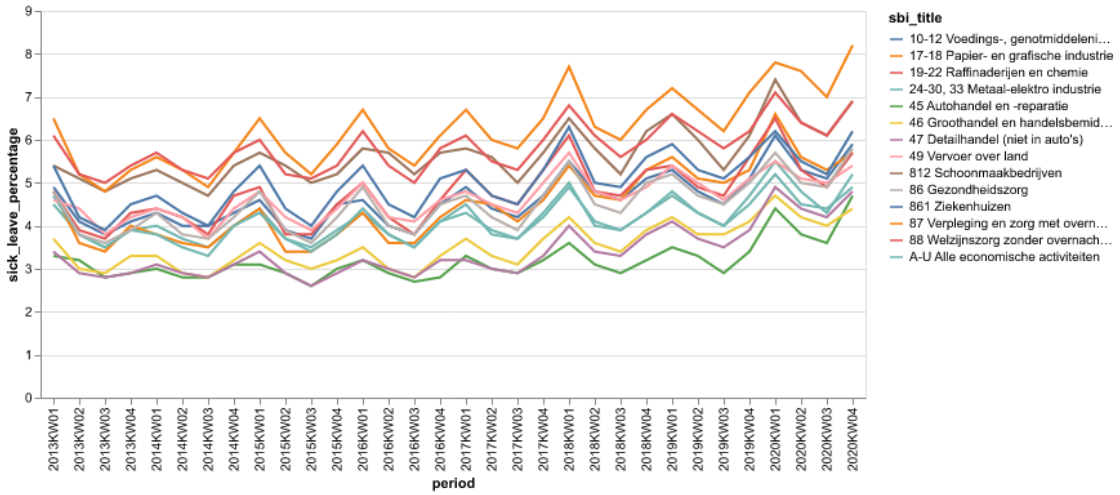
[9]:
```
c = alt.Chart(slp_train[slp_train.category_group_title.isin(['Bedrijfssector',␣
↪'Totaal'])]).mark_line().encode(
    x='period',
    y='sick_leave_percentage',
    color='sbi_title'
)

save_graph(c, 'slp_train_sbi_sector')
c
```

[9]:


[10]:
```
c = alt.Chart(slp_train[slp_train.category_group_title.isin([ 'Bedrijfsklasse',␣
↪'Totaal'])]).mark_line().encode(
    x='period',
```

```
    y='sick_leave_percentage',
    color='sbi_title'
)

save_graph(c, 'slp_train_sbi_klasse')
c
```

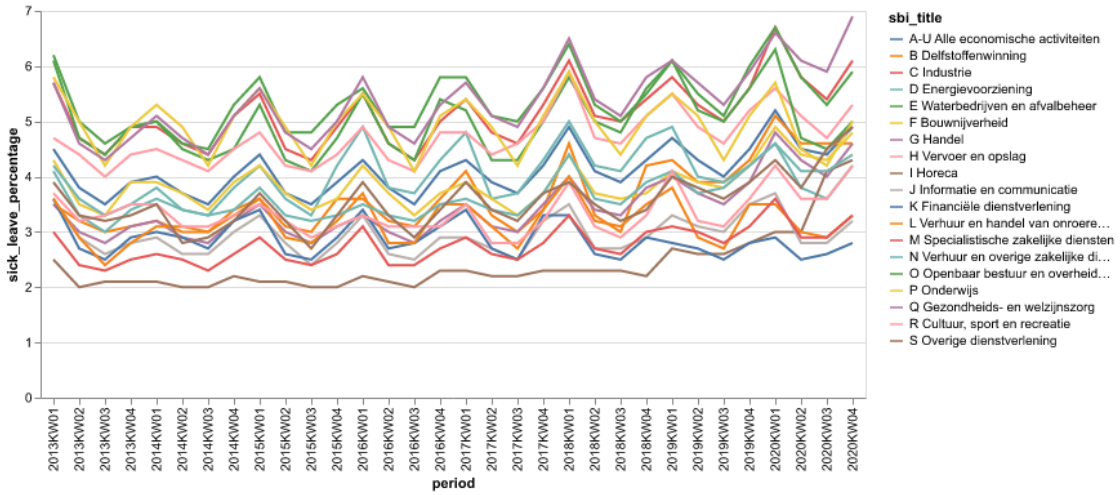
[10]:



[11]:

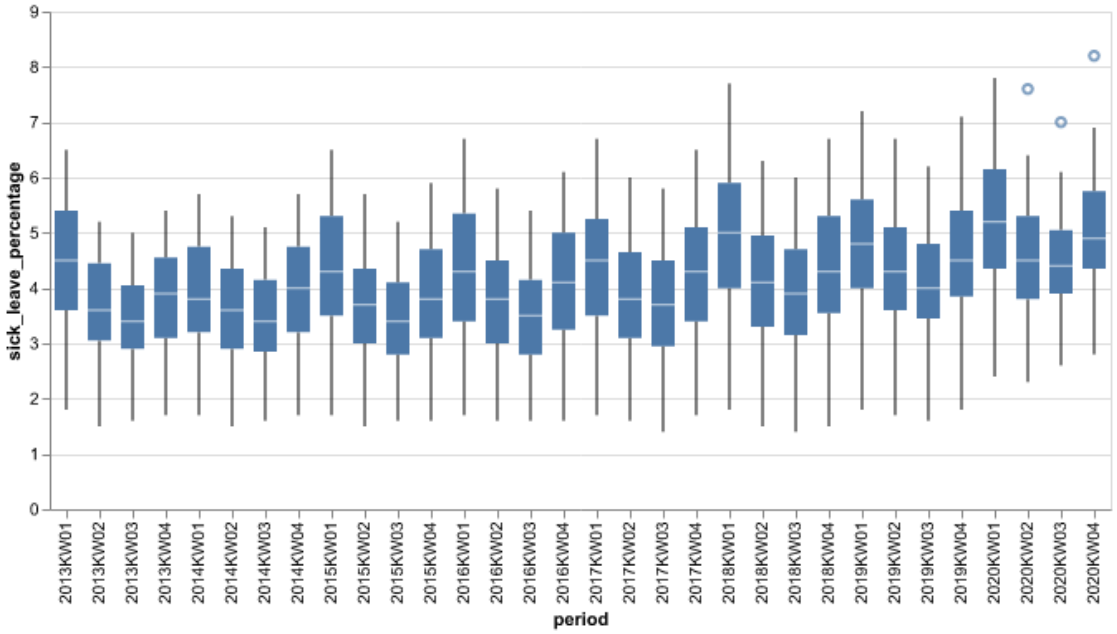
```
c = alt.Chart(slp_train[slp_train.category_group_title.isin(['Bedrijfstak',␣
↪'Totaal'])]).mark_line().encode(
    x='period',
    y='sick_leave_percentage',
    color='sbi_title'
)

save_graph(c, 'slp_train_sbi_tak')
c
```

[11]:



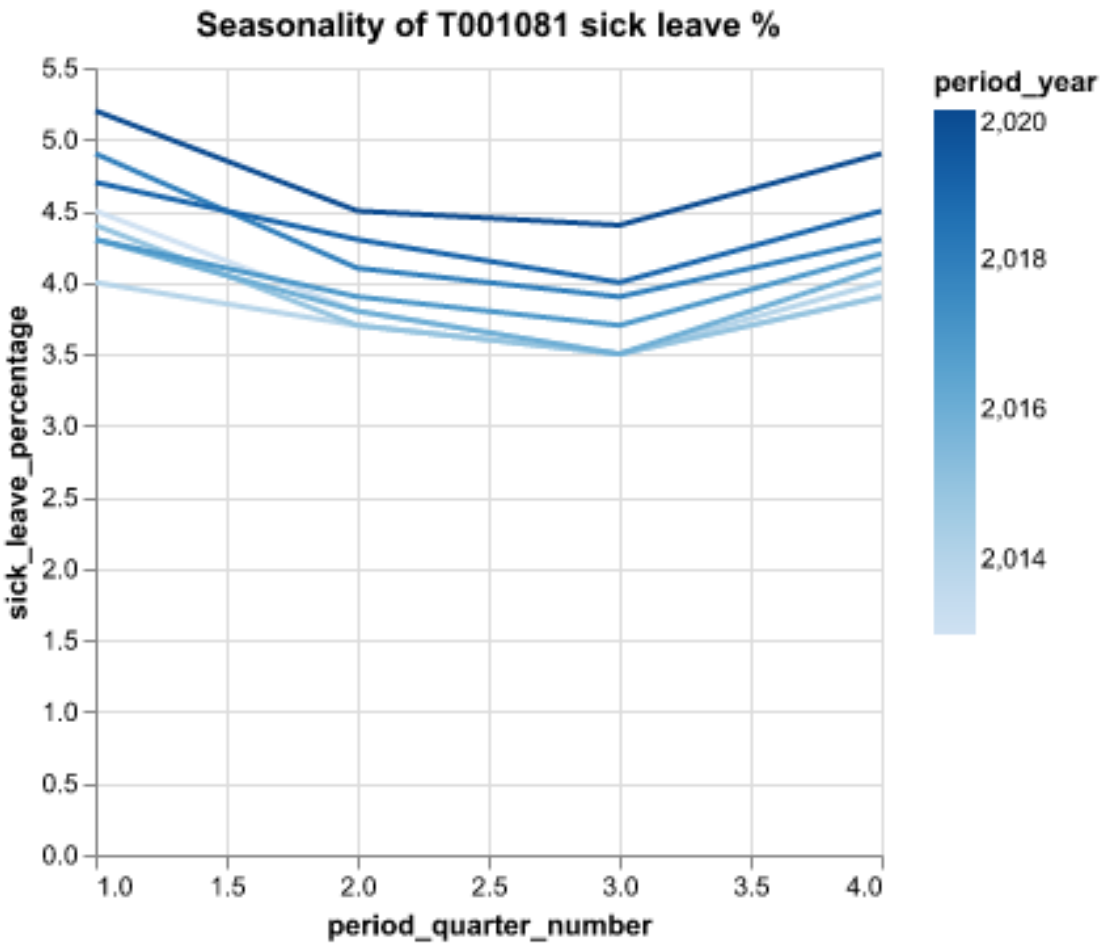
[12]:
```
c = alt.Chart(slp_train).mark_boxplot().encode(
    x='period',
    y='sick_leave_percentage',
)

save_graph(c, 'slp_train_boxplot')
c
```

[12]:



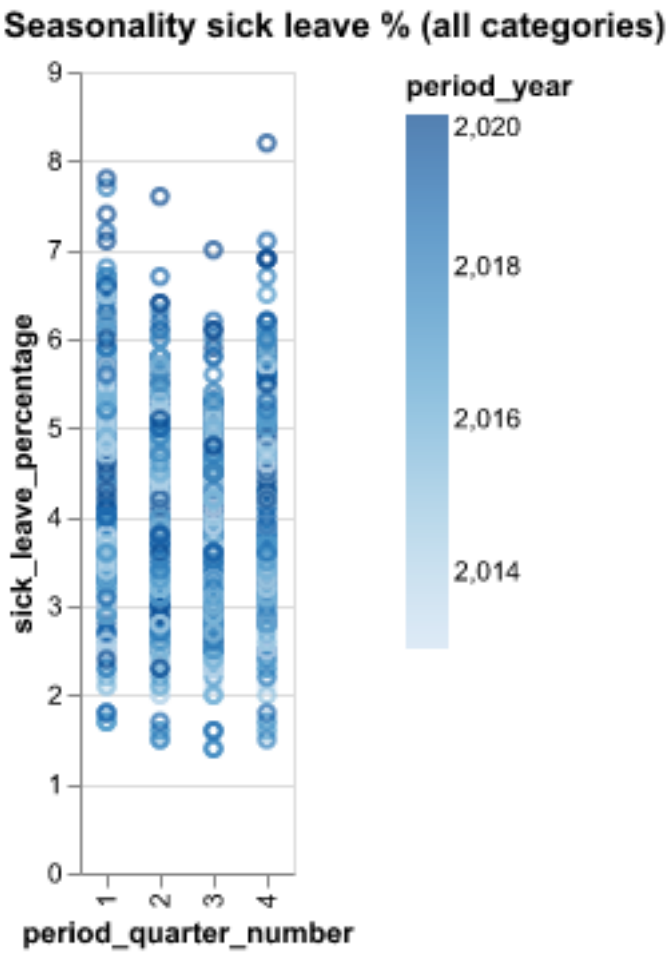
[13]:
```
slp_train_total = slp_train[slp_train.sbi == 'T001081']

c = alt.Chart(slp_train_total).mark_line().encode(
    x='period_quarter_number',
    y='sick_leave_percentage',
    color='period_year'
).properties(title='Seasonality of T001081 sick leave %')

save_graph(c, 'slp_train_total_seasonality')
c
```

[13]:



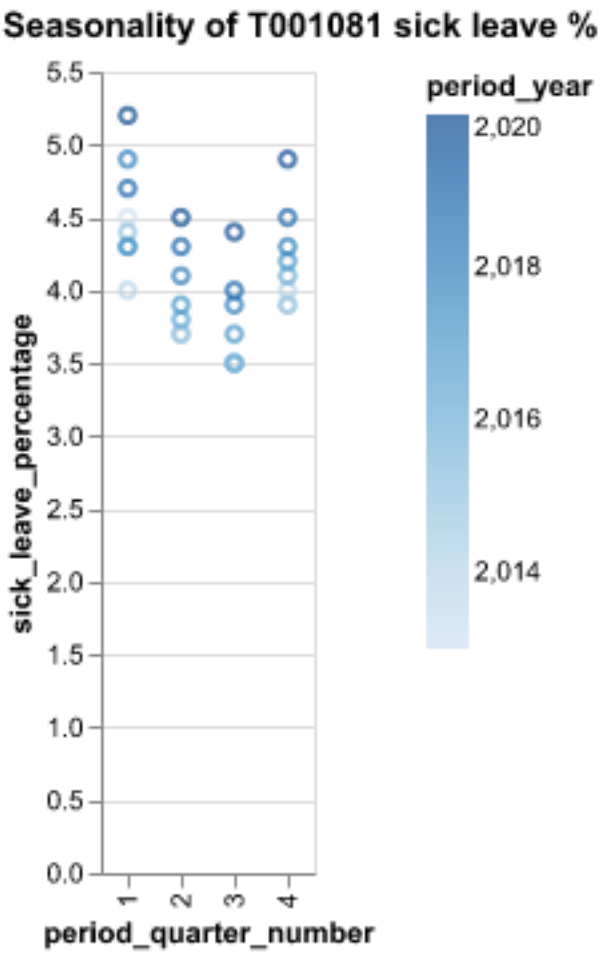
[14]:
```
c = alt.Chart(slp_train).mark_point().encode(
     x=alt.X(alt.repeat("column"), type='ordinal'),
    y='sick_leave_percentage',
    color='period_year'
).repeat(
    column=['period_quarter_number']
).properties(
    title='Seasonality sick leave % (all categories)'
)

save_graph(c, 'slp_train_seasonality_all_sbi')
c
```

[14]:



[15]:
```
c = alt.Chart(slp_train_total).mark_point().encode(
    x=alt.X(alt.repeat("column"), type='ordinal'),
    y='sick_leave_percentage',
    color='period_year'
).repeat(
    column=['period_quarter_number']
).properties(
    title='Seasonality of T001081 sick leave %'
)

save_graph(c, 'slp_train_total_seasonality_point')
c
```

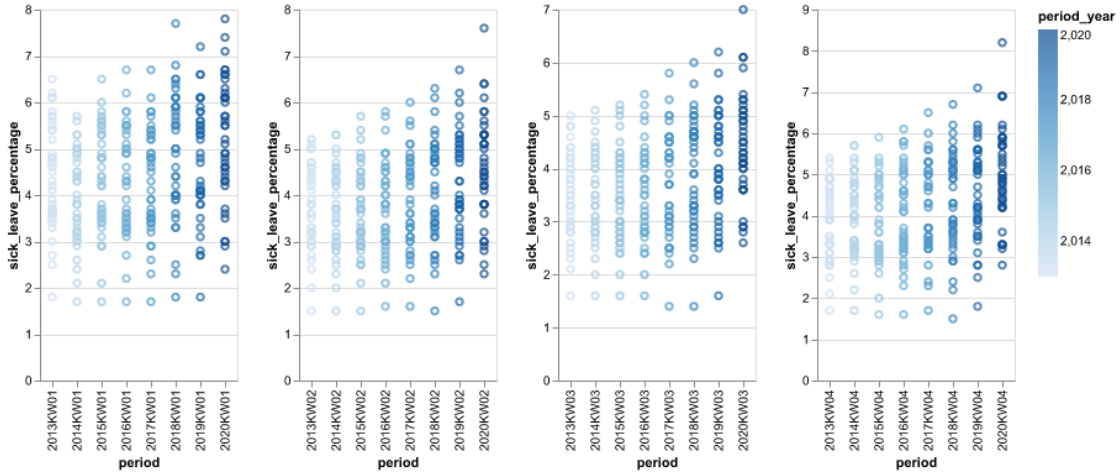
[15]:



[16]:
```
c = alt.Chart(slp_train[slp_train.period_quarter_number == 1]).mark_point().
 ↪encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train[slp_train.period_quarter_number  == 2]).mark_point().
 ↪encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train[slp_train.period_quarter_number  == 3]).mark_point().
 ↪encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train[slp_train.period_quarter_number  == 4]).mark_point().
 ↪encode(
```

```
    x='period',
    y='sick_leave_percentage',
    color='period_year'
)

save_graph(c, 'slp_train_seasonality_all_sbi_point_per_quarter')
c
```

[16]:

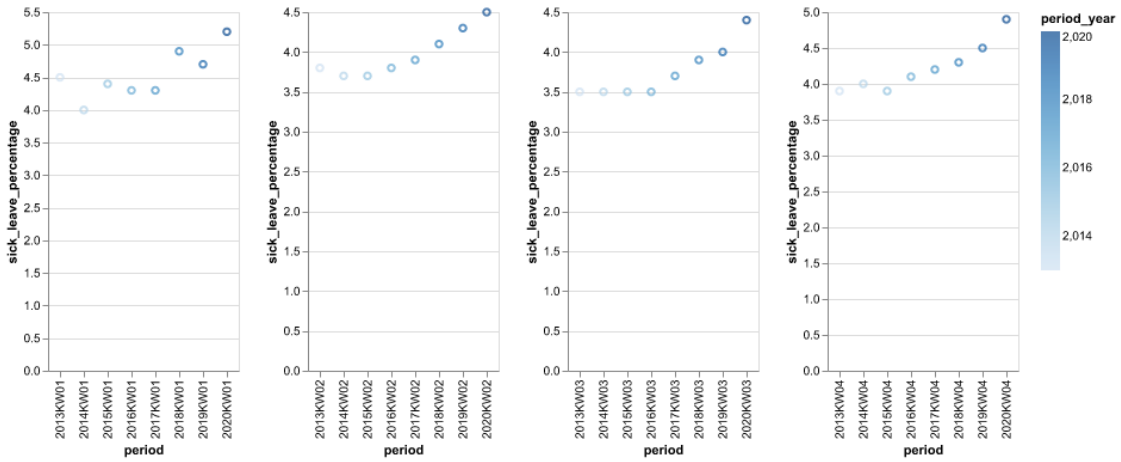


[17]:

```
c = alt.Chart(slp_train_total[slp_train_total.period_quarter_number ==  1]).
 ↪mark_point().encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train_total[slp_train_total.period_quarter_number ==  2]).
 ↪mark_point().encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train_total[slp_train_total.period_quarter_number ==  3]).
 ↪mark_point().encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
) | alt.Chart(slp_train_total[slp_train_total.period_quarter_number ==  4]).
 ↪mark_point().encode(
    x='period',
    y='sick_leave_percentage',
    color='period_year'
)
```

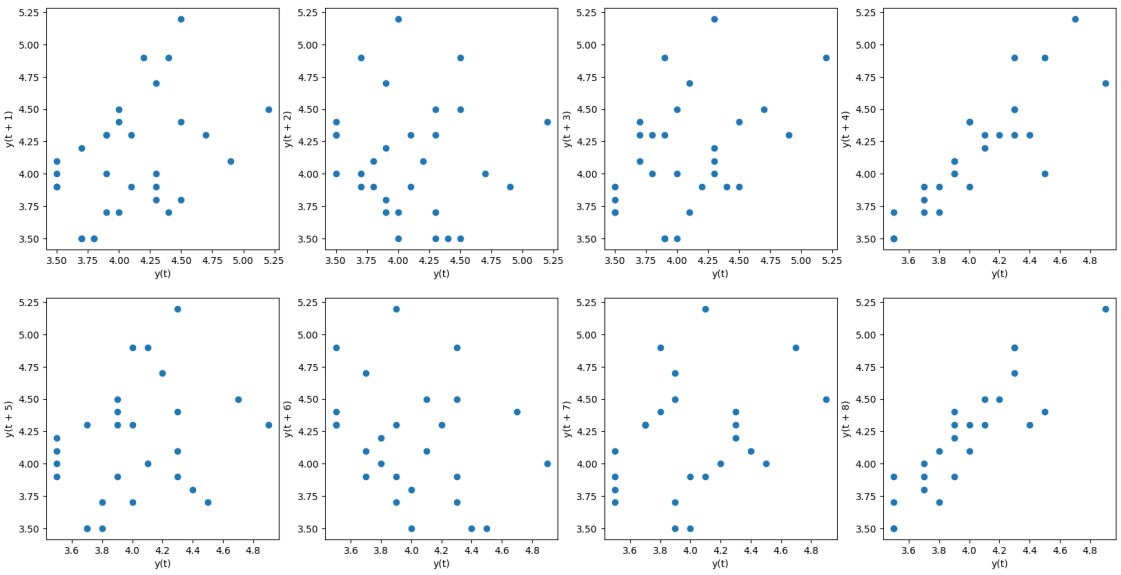
```
save_graph(c, 'slp_train_total_seasonality_point_per_quarter')
c
```

[17]:



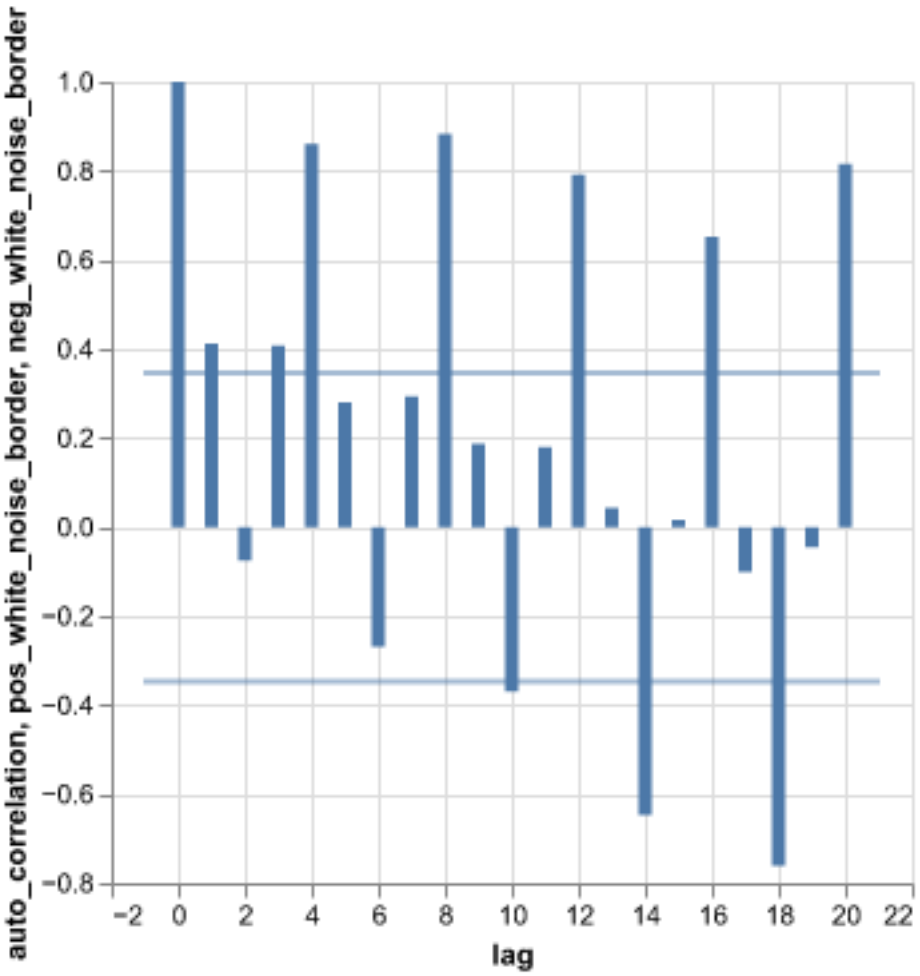
[18]:

```
from pandas.plotting import lag_plot

fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(20,10))

for i, lag in enumerate(range(1,9)):
    lag_plot(slp_train_total.sick_leave_percentage, lag=lag, ax=axes[i // 4, i␣
 ↪% 4])

save_fig(plt, 'slp_train_total_lag_plot')
plt.show()
```



[19]:
```
import math

start_lag = 0
lag_length = 21

lagged_auto_correlation = pd.DataFrame()
lagged_auto_correlation['lag'] = range(start_lag, lag_length)

white_noise_border = 1.96 / math.sqrt(len(slp_train_total.
 ↪sick_leave_percentage))

wn_border = pd.DataFrame()
wn_border['lag'] = range(start_lag - 1, lag_length + 1)
wn_border['pos_white_noise_border'] = [white_noise_border for _ in␣
 ↪range(start_lag - 1, lag_length + 1)]
wn_border['neg_white_noise_border'] = [-white_noise_border for _ in␣
 ↪range(start_lag - 1, lag_length + 1)]

lagged_auto_correlation['auto_correlation'] = [slp_train_total.
 ↪sick_leave_percentage.autocorr(lag=lag) for lag in␣
 ↪lagged_auto_correlation['lag']]

c = alt.Chart(lagged_auto_correlation).mark_bar().encode(
    x='lag',
    y='auto_correlation',
) + alt.Chart(wn_border).mark_line(strokeDash=[1,1]).encode(
    x='lag',
    y='pos_white_noise_border',
) + alt.Chart(wn_border).mark_line(strokeDash=[1,1]).encode(
    x='lag',
    y='neg_white_noise_border'
)

save_graph(c, 'slp_train_total_auto_correlation')
c
```

[19]:



[20]:
```
moving_average = pd.DataFrame()

moving_average['quarter'] = slp_train_total.period
moving_average['sick'] = slp_train_total.sick_leave_percentage

for window in range(3, 16, 2):
    moving_average[f'{window}-MA'] = slp_train_total.sick_leave_percentage.
 ↪rolling(window, center=True).mean()

moving_average
```

[20]:

	quarter	sick	3-MA	5-MA	7-MA	9-MA	11-MA	13-MA	\
85	2013KW01	4.5	NaN	NaN	NaN	NaN	NaN	NaN	
86	2013KW02	3.8	3.933333	NaN	NaN	NaN	NaN	NaN	
87	2013KW03	3.5	3.733333	3.94	NaN	NaN	NaN	NaN	
88	2013KW04	3.9	3.800000	3.78	3.842857	NaN	NaN	NaN	

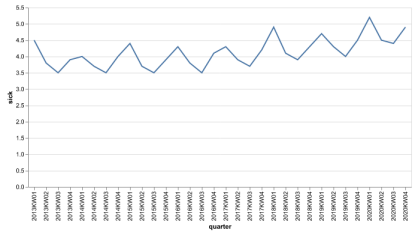
90	2014KW01	4.0	3.866667	3.72	3.771429	3.922222	NaN	NaN
91	2014KW02	3.7	3.733333	3.82	3.857143	3.833333	3.863636	NaN
92	2014KW03	3.5	3.733333	3.92	3.885714	3.800000	3.809091	3.900000
93	2014KW04	4.0	3.966667	3.86	3.828571	3.844444	3.854545	3.846154
95	2015KW01	4.4	4.033333	3.82	3.814286	3.888889	3.881818	3.823077
96	2015KW02	3.7	3.866667	3.90	3.900000	3.866667	3.845455	3.869231
97	2015KW03	3.5	3.700000	3.96	3.942857	3.844444	3.854545	3.900000
98	2015KW04	3.9	3.900000	3.84	3.871429	3.911111	3.909091	3.892308
100	2016KW01	4.3	4.000000	3.80	3.828571	3.944444	3.945455	3.892308
101	2016KW02	3.8	3.866667	3.92	3.914286	3.888889	3.918182	3.946154
102	2016KW03	3.5	3.800000	4.00	3.971429	3.888889	3.900000	4.015385
103	2016KW04	4.1	3.966667	3.92	3.942857	3.966667	4.009091	3.992308
105	2017KW01	4.3	4.100000	3.90	3.928571	4.077778	4.063636	4.007692
106	2017KW02	3.9	3.966667	4.04	4.085714	4.055556	4.063636	4.069231
107	2017KW03	3.7	3.933333	4.20	4.171429	4.066667	4.063636	4.130769
108	2017KW04	4.2	4.266667	4.16	4.142857	4.155556	4.145455	4.130769
110	2018KW01	4.9	4.400000	4.16	4.142857	4.222222	4.218182	4.146154
111	2018KW02	4.1	4.300000	4.28	4.257143	4.222222	4.209091	4.223077
112	2018KW03	3.9	4.100000	4.38	4.342857	4.233333	4.227273	4.307692
113	2018KW04	4.3	4.300000	4.26	4.314286	4.322222	4.345455	4.323077
115	2019KW01	4.7	4.433333	4.24	4.257143	4.433333	4.418182	4.361538
116	2019KW02	4.3	4.333333	4.36	4.414286	4.388889	4.436364	4.453846
117	2019KW03	4.0	4.266667	4.54	4.500000	4.422222	4.436364	NaN
118	2019KW04	4.5	4.566667	4.50	4.514286	4.533333	NaN	NaN
120	2020KW01	5.2	4.733333	4.52	4.542857	NaN	NaN	NaN
121	2020KW02	4.5	4.700000	4.70	NaN	NaN	NaN	NaN
122	2020KW03	4.4	4.600000	NaN	NaN	NaN	NaN	NaN
123	2020KW04	4.9	NaN	NaN	NaN	NaN	NaN	NaN

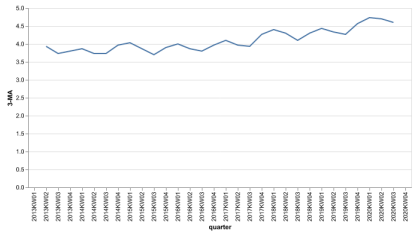
	15-MA
85	NaN
86	NaN
87	NaN
88	NaN
90	NaN
91	NaN
92	NaN
93	3.866667
95	3.840000
96	3.873333
97	3.900000
98	3.886667
100	3.900000
101	3.980000
102	4.020000
103	4.013333
105	4.006667

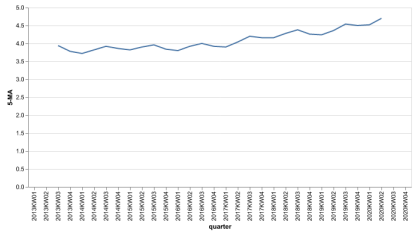
```
106  4.073333
107  4.126667
108  4.133333
110  4.146667
111  4.240000
112  4.306667
113  4.326667
115  4.366667
116       NaN
117       NaN
118       NaN
120       NaN
121       NaN
122       NaN
123       NaN
```

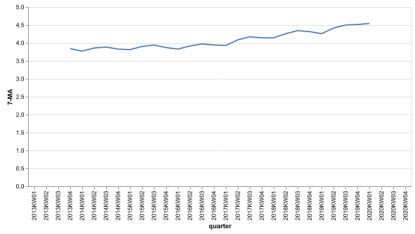
[21]:
```
charts = [alt.Chart(moving_average).mark_line().encode(x='quarter', y='sick')]

for window in range(3, 16, 2):
    charts.append(alt.Chart(moving_average).mark_line().encode(x='quarter',␣
 ↪y=f'{window}-MA'))

c = alt.vconcat(*charts)
save_graph(c, 'slp_train_total_moving_average')
c
```

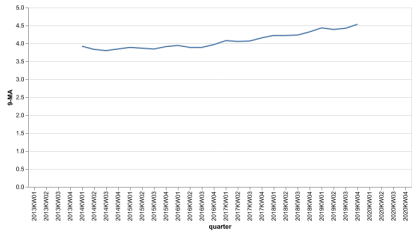
[21]:

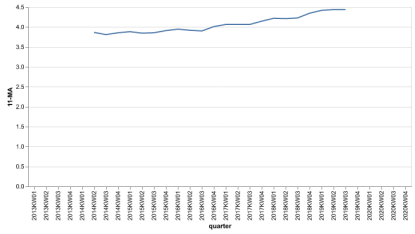

tick
quarter

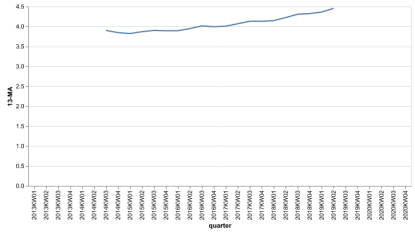

3-MA
quarter

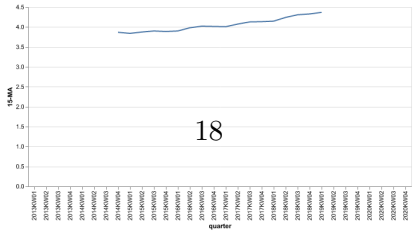

5-MA
quarter


7-MA
quarter


9-MA
quarter


11-MA
quarter


13-MA
quarter


15-MA
quarter

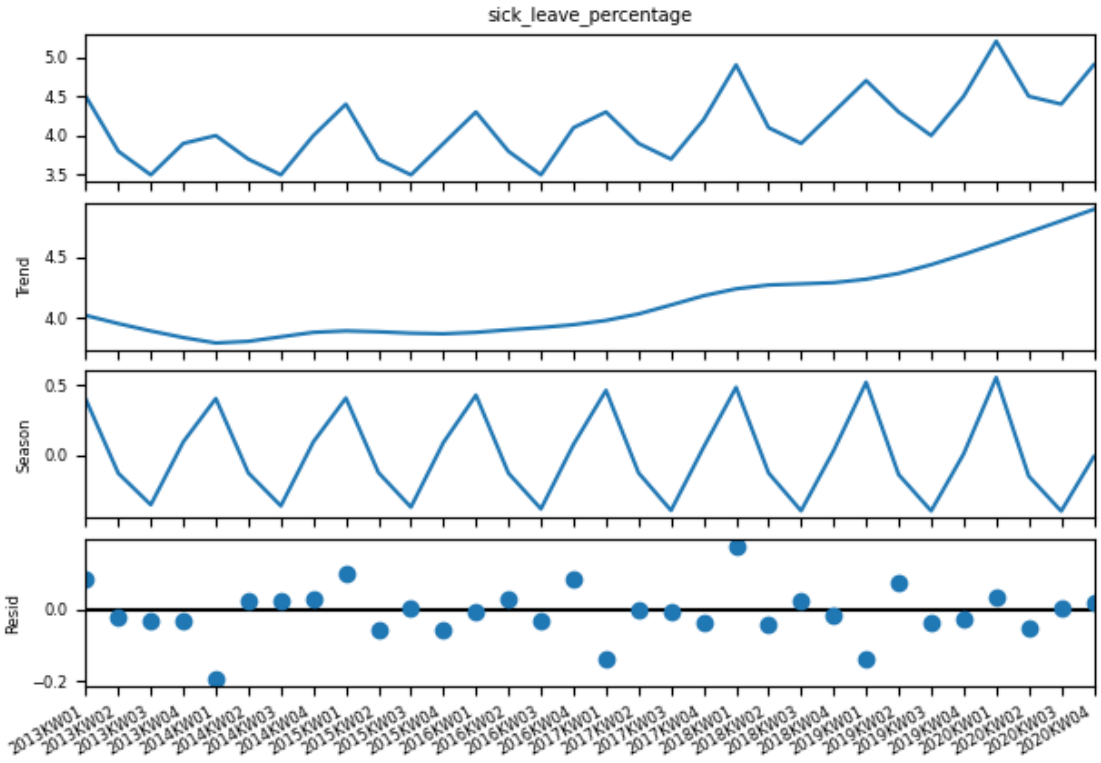
[22]:
```
from statsmodels.tsa.seasonal import STL

slp_series = slp_train_total.sick_leave_percentage
slp_series.index = slp_train_total.period
slp_series
```

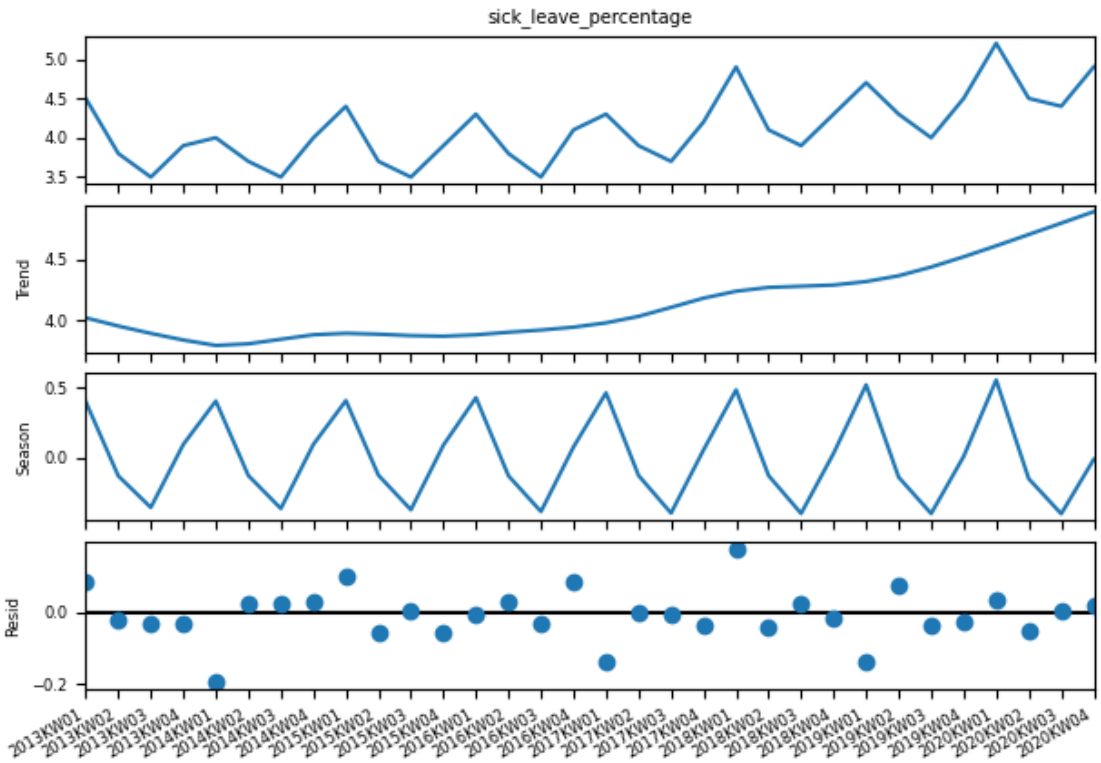
[22]:
```
period
2013KW01    4.5
2013KW02    3.8
2013KW03    3.5
2013KW04    3.9
2014KW01    4.0
2014KW02    3.7
2014KW03    3.5
2014KW04    4.0
2015KW01    4.4
2015KW02    3.7
2015KW03    3.5
2015KW04    3.9
2016KW01    4.3
2016KW02    3.8
2016KW03    3.5
2016KW04    4.1
2017KW01    4.3
2017KW02    3.9
2017KW03    3.7
2017KW04    4.2
2018KW01    4.9
2018KW02    4.1
2018KW03    3.9
2018KW04    4.3
2019KW01    4.7
2019KW02    4.3
2019KW03    4.0
2019KW04    4.5
2020KW01    5.2
2020KW02    4.5
2020KW03    4.4
2020KW04    4.9
Name: sick_leave_percentage, dtype: float64
```

[23]:
```
plt.rc("font", size=6)
stl = STL(slp_series, period=4)
res = stl.fit()
```

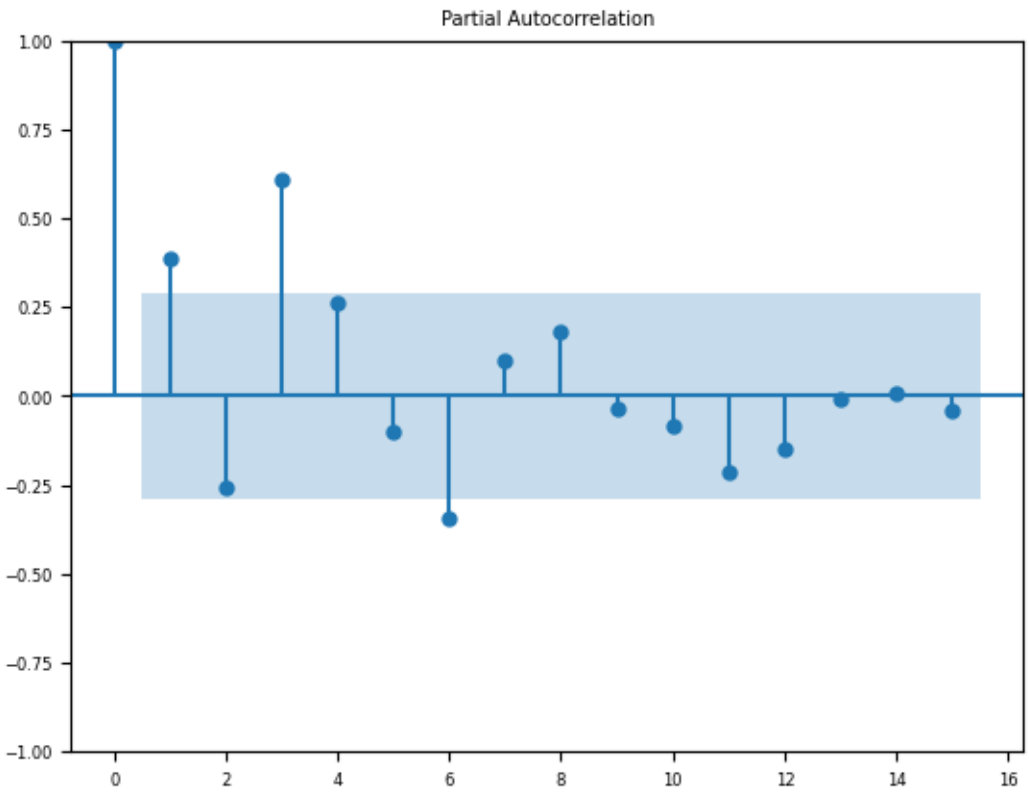
```
fig = res.plot()
fig.autofmt_xdate()
save_fig(fig, 'slp_train_total_STL_plot')
fig
```

[23]:





[24]:
```
from statsmodels.graphics.tsaplots import plot_pacf

plot_pacf(slp_series, lags=15, alpha=0.1)
save_fig(plt, 'slp_train_total_pacf')
plt.show()
```


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
12
14
16