PV degradation rate estimation using seasonal-trend decomposition by locally weighted scatterpolot smoothing LOWESS (STL)

Marios Theristis and Joshua S. Stein, Sandia National Laboratories

This notebook applies the non-parametric method of STL (seasonal-trend decomposition using locally weighted scatterplot smoothing, LOWESS) to PV performance timeseries. LOWESS basically fits a non-parametric line to the PR timeseries plot and is usually treated the same as LOESS, in literature. While LOESS can be used with multiple predictors, LOWESS would only work with univariate datasets (e. g. timestamp Vs PR) and therefore, excess columns should be dropped. Since LOWESS-based STL and LOESS-based STL yield different results in Python, both methods are presented in separate jupyter notebooks. Again, the degradation rate (DR) is estimated by applying OLS on the decomposed trend which is extracted using LOWESS.

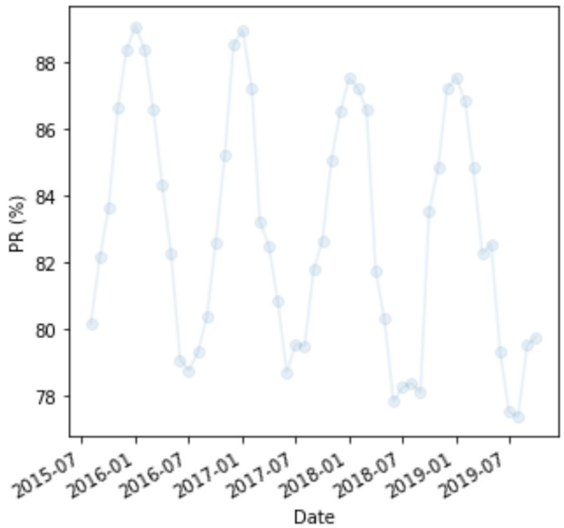
In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm
from stldecompose import decompose
from matplotlib import pyplot as plt
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
```

In [2]:
```
#reading the csv file that contains timeseries with operational and irradiance dat
a. In this example, we import the monthly performance ratio.
df = pd.read_csv(r'C:\...\Sample_data.csv', delimiter = ',' , parse_dates= ['Timest
amp'], dayfirst = False)
```

In [3]:
```
df.index = df['Timestamp']
```

In [4]:
```
#dropping columns since this method works only with univariate datasets
df.drop(columns=['Timestamp', 'Month'], inplace=True)
```

In [5]:
```
#plot the monthly PRs
fig, axs = plt.subplots(figsize=(5,5))
axs.plot(df.index, df.PR, 'o-', alpha = 0.1)
axs.set_ylabel('PR (%)');
axs.set_xlabel('Date')
fig.autofmt_xdate()
```



The *stldecompose* model is similar to the _statsmodels.tsa.seasonal*decompose* method but substitutes the centered moving average with a LOWESS regression using _statsmodels.nonparametric.smoothers*lowess.lowess* for a convolution in its trend estimation. The STL function requires the period of the timeseries (e.g. period = 12 when using monthly data); the fraction of the data used when estimating each y-value is set to 0.6. Once the timeseries decomposition is done, a new dataframe with the trend values is created. Finally, OLS is applied on the trend in order to calculate the absolute and relative DR as follows:

DR_abs = resolution * slope

DR_rel = 100 *resolution* slope/intercept

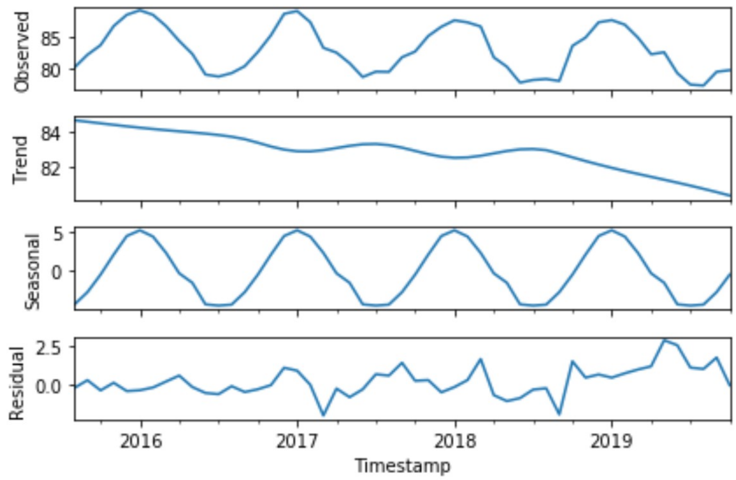
Lower and upper confidence intervals are calculated for a confidence level of 95% (i.e. significance level, alpha = 0.05)

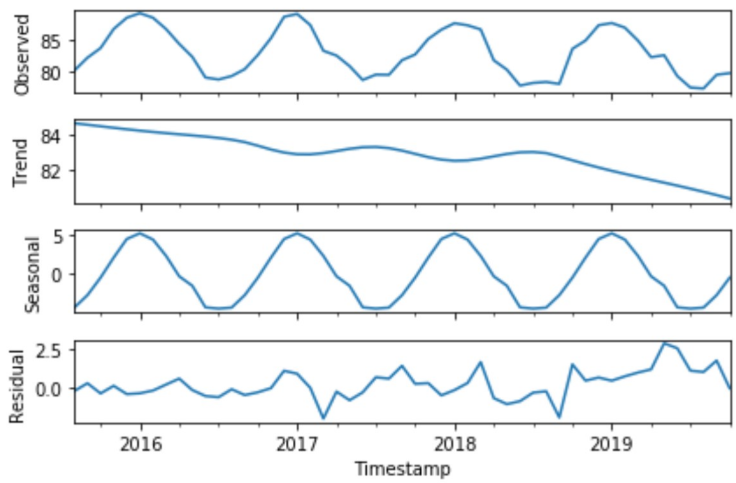
In [6]:
```
#daily 365, monthly 12 etc.
resolution = 12
```

In [7]:
```
stl = decompose(df, period = resolution)
```

In [8]:
```
stl.plot()
```

Out[8]:





In [9]:
```
stl.trend.insert(loc=0, column = 'Index', value=np.arange(len(stl.trend)))
```

Applying OLS on the trend:

In [10]:
```
y = stl.trend.PR
x = stl.trend.Index
x, y = np.array(x), np.array(y)
```

In [11]:
```
x = sm.add_constant(x)
```

In [12]:
```
model = sm.OLS(y,x)
```

In [13]:
```
results = model.fit()
```

In [14]:
```
results.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.883
Model:	OLS	**Adj. R-squared:**	0.881
Method:	Least Squares	**F-statistic:**	371.3
Date:	Mon, 16 Dec 2019	**Prob (F-statistic):**	1.64e-24
Time:	09:54:01	**Log-Likelihood:**	-21.853
No. Observations:	51	**AIC:**	47.71
Df Residuals:	49	**BIC:**	51.57
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	84.6283	0.105	809.260	0.000	84.418	84.838
x1	-0.0695	0.004	-19.268	0.000	-0.077	-0.062

Omnibus:	0.651	**Durbin-Watson:**	0.085
Prob(Omnibus):	0.722	**Jarque-Bera (JB):**	0.288
Skew:	0.178	**Prob(JB):**	0.866
Kurtosis:	3.092	**Cond. No.**	57.2

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [15]:
```
intercept, slope = results.params
```

In [16]:
```
#confidence level 95%
CI_abs = resolution*results.conf_int(alpha  = 0.05)[1]
CIL_abs = CI_abs[1]
CIH_abs = CI_abs[0]
round(CIL_abs,2),round(CIH_abs,2)
```

Out[16]: (-0.75, -0.92)

In [17]:
```
DR_abs = round(resolution*slope, 2)
DR_abs
```

Out[17]: -0.83

In [18]:
```
#confidence level 95%
CI_rel = 100*resolution*results.conf_int(alpha  = 0.05)[1]/intercept
CIL_rel = CI_rel[1]
CIH_rel = CI_rel[0]
round(CIL_rel,2),round(CIH_rel,2)
```

Out[18]: (-0.88, -1.09)

In [19]:
```
DR_rel = round(100*resolution*slope/intercept,2)
DR_rel
```

Out[19]: -0.98

In [20]:
```
fig = sns.regplot(y=stl.trend['PR'], x=stl.trend['Index'], data=stl.trend)
plt.xlabel("Month")
plt.ylabel("Trend of PR (%)")
plt.title("Relative degradation rate of -0.98%/year on STL trend")
plt.ylim(75, 90)
plt.xlim(0, 55)
plt.show(fig)
```

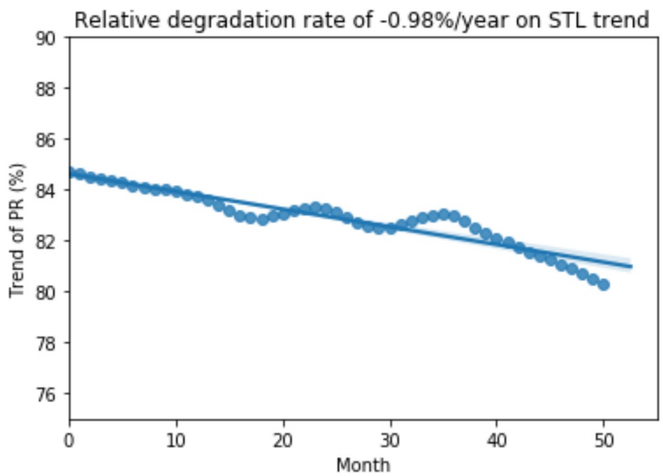


Table of results

In [21]:
```
Results = [['Relative DR', round(DR_rel,2) ], ['Relative CIL', round(CIL_rel,2)],
['Relative CIH', round(CIH_rel,2)], ['Absolute DR', round(DR_abs,2)], ['Absolute CI
L', round(CIL_abs,2)], ['Absolute CIH', round(CIH_abs,2)]]

# Create the pandas DataFrame
Results = pd.DataFrame(Results, columns = ['Parameter', 'Value (%/year)'])

Results
```

Out[21]:

	Parameter	Value (%/year)
0	Relative DR	-0.98
1	Relative CIL	-0.88
2	Relative CIH	-1.09
3	Absolute DR	-0.83
4	Absolute CIL	-0.75
5	Absolute CIH	-0.92

More information about the STL decomposition method can be found in:

Hyndman, Rob J., and George Athanasopoulos. Forecasting: principles and practice. OTexts, 2014.

https://otexts.com/fpp2/stl.html (https://otexts.com/fpp2/stl.html)