PV degradation rate estimation using seasonal-trend decomposition by locally estimated scatterpolot smoothing LOESS (STL) based on a direct translation from R

Marios Theristis and Joshua S. Stein, Sandia National Laboratories

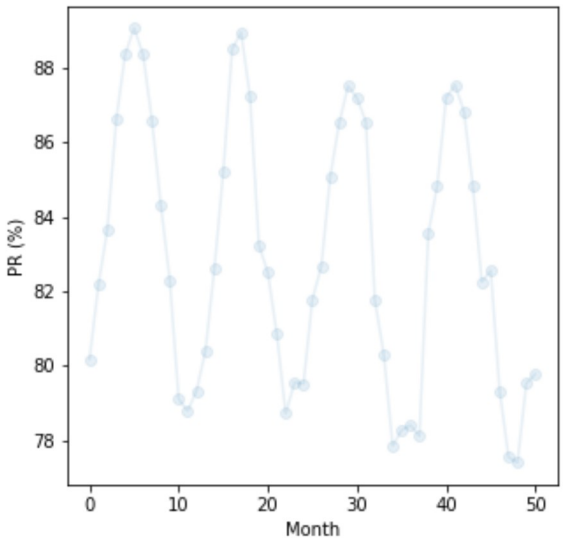
This notebook applies the non-parametric method of STL (seasonal-trend decomposition using locally estimated scatterplot smoothing, LOESS) to PV performance timeseries. Similar to classical decomposition, OLS is applied on the STL trend in order to calculate the degradation rate (DR). The method is applied using *rstl* which is a translation of R's STL function in Python.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from rstl import STL
import statsmodels.api as sm
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]:
```
#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('Month')
```

Out[3]:
```
Text(0.5, 0, 'Month')
```



Seasonal-trend decomposition using LOESS is applied with *rstl*. Similarly to seasonal decomposition, STL decomposes the timeseries to trend + seasonality + error, however, instead of using a centered moving average for extracting the trend, it uses LOESS. The STL function requires the frequency of the timeseries (e.g. freq = 12 when using monthly data) and information about the seasonal window; i.e. whether the data are periodic or the span (in lags) of the LOESS window for seasonal extraction. 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 [4]:
```
#daily 365, monthly 12 etc.
resolution = 12
```

In [5]:
```
stl = STL(df['PR'], resolution, "periodic")
```

In [6]:
```
trend = stl.trend
```

In [7]:
```
trend_df = pd.DataFrame(list(trend), columns = ['PR'])
```

In [8]:
```
trend_df.insert(loc = 0, column = 'Index', value = np.arange(len(trend_df.PR)))
```

Applying OLS on the trend:

In [9]:
```
y = trend_df.PR
x = trend_df.index
x, y = np.array(x), np.array(y)
```

In [10]:
```
x = sm.add_constant(x)
```

In [11]:
```
model = sm.OLS(y,x)
```

In [12]:
```
results = model.fit()
```

In [13]:
```
results.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.909
Model:	OLS	**Adj. R-squared:**	0.907
Method:	Least Squares	**F-statistic:**	488.2
Date:	Mon, 16 Dec 2019	**Prob (F-statistic):**	3.95e-27
Time:	09:55:21	**Log-Likelihood:**	7.9673
No. Observations:	51	**AIC:**	-11.93
Df Residuals:	49	**BIC:**	-8.071
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	84.2139	0.058	1445.078	0.000	84.097	84.331
x1	-0.0444	0.002	-22.095	0.000	-0.048	-0.040

Omnibus:	1.335	**Durbin-Watson:**	0.072
Prob(Omnibus):	0.513	**Jarque-Bera (JB):**	1.113
Skew:	0.148	**Prob(JB):**	0.573
Kurtosis:	2.340	**Cond. No.**	57.2

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

In [14]:
```
intercept, slope = results.params
```

In [15]:
```
#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[15]: (-0.48, -0.58)

In [16]:
```
DR_abs = round(resolution*slope, 2)
DR_abs
```

Out[16]: -0.53

In [17]:
```
#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[17]: (-0.57, -0.69)

In [18]:
```
DR_rel = round(100*resolution*slope/intercept,2)
DR_rel
```

Out[18]: -0.63

In [19]:
```
fig = sns.regplot(y=trend_df['PR'], x=trend_df['Index'], data=trend_df)
plt.xlabel("Month")
plt.ylabel("Trend of PR (%)")
plt.ylim(80, 85)
plt.title("Relative degradation rate of -0.63%/year on STL trend")
plt.show(fig)
```

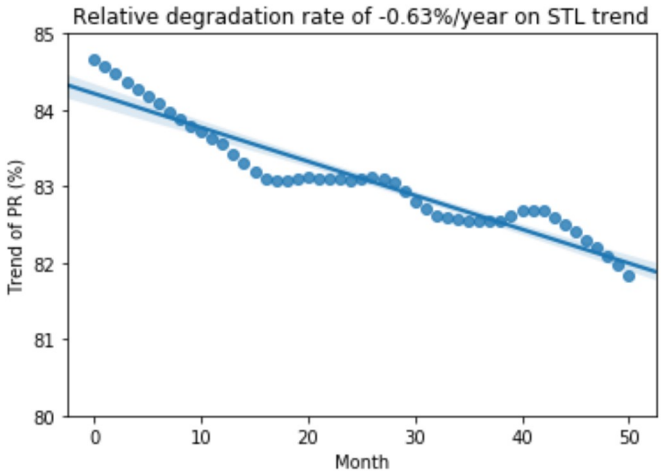


Table of results

In [20]:
```
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[20]:

	Parameter	Value (%/year)
0	Relative DR	-0.63
1	Relative CIL	-0.57
2	Relative CIH	-0.69
3	Absolute DR	-0.53
4	Absolute CIL	-0.48
5	Absolute CIH	-0.58

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)