Modeling and Analysis of Time Series Data
Chapter 8: Smoothing in the time and frequency domains

STATS 531, Winter 2026

Edward Ionides

Introduction to smoothing in time series analysis

- Estimating a nonparametric trend from a time series is known as smoothing. We will review some standard smoothing methods.
- We also smooth the periodogram to estimate a spectral density.
- Smoothers have convenient interpretation in the frequency domain. A smoother typically shrinks high frequency components and preserves low frequency components.

A motivating example

- The economy fluctuates between periods of rapid expansion and periods of slower growth or contraction.
- High unemployment is one of the most visible signs of a dysfunctional economy, in which labor is under-utilized, leading to hardships for many individuals and communities.
- Economists, politicians, businesspeople and the general public have an interest in understanding fluctuations in unemployment.
- Economists try to distinguish between fundamental structural changes in the economy and shorter-term boom/bust cycles that appear to be a natural part of capitalist business activity.
- Monthly US unemployment figures are published by the Bureau of Labor Statistics (BLS).
- Measuring unemployment has subtleties, but these are not our immediate focus.

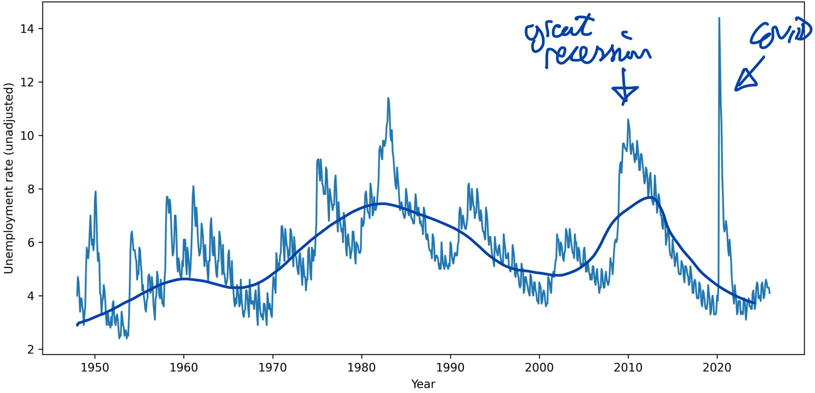
```python
import subprocess
result = subprocess.run(['head', 'unadjusted_unemployment.csv'],
    capture_output=True, text=True)
print(result.stdout)
```

```
# Data extracted on: February 7, 2024
# Updated on: February 7, 2025 and January 25, 2026
# from http://data.bls.gov/timeseries/LNU04000000
# Labor Force Statistics from the Current Population Survey
# Data unavailable Oct 2025 due to the lapse in appropriations.
# Not Seasonally Adjusted
# Series title:        (Unadj) Unemployment Rate
# Series Id:           LNU04000000
# Labor force status:  Unemployment rate
# Type of data:        Percent or rate
```

```python
U1 = pd.read_csv("unadjusted_unemployment.csv", comment='#')
U1.iloc[:3,:10]
```

	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
0	1948	4.0	4.7	4.5	4.0	3.4	3.9	3.9	3.6	3.4
1	1949	5.0	5.8	5.6	5.4	5.7	6.4	7.0	6.3	5.9
2	1950	7.6	7.9	7.1	6.0	5.3	5.6	5.3	4.1	4.0

```
u1 = U1.iloc[:, 1:13].values.flatten()
date = np.arange(1948, 1948 + len(u1)/12, 1/12)
u1[933] = (u1[932]+u1[934])/2 # interpolate NaN
plt.figure(figsize=(10, 5))
plt.plot(date, u1, '-')
plt.ylabel("Unemployment rate (unadjusted)")
plt.xlabel("Year")
plt.tight_layout(); plt.show()
```

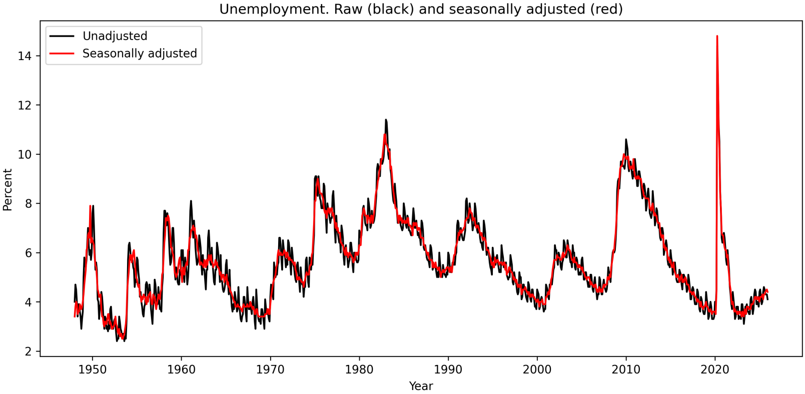


- We see seasonal variation and economic cycles on top of a trend.

- The seasonal variation looks like an additive effect, say an annual fluctuation with amplitude around 1 percentage point.

- Sometimes, we may prefer to look at monthly seasonally adjusted unemployment, also provided by BLS.

```
U2 = pd.read_csv("adjusted_unemployment.csv",
    comment='#')
u2 = U2.iloc[:, 1:13].values.flatten()
u2[933] = (u2[932]+u2[934])/2 # interpolate NaN
```

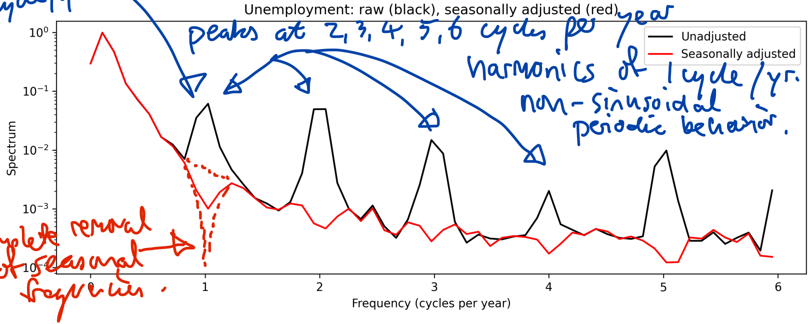


- We can wonder how the BLS adjusts the data, and if this introduces any artifacts that a careful statistician should be aware of.

Seasonal adjustment in the frequency domain

To help understand the seasonal adjustment, we look at what it does to the smoothed periodogram. We compute smoothed periodograms for both time series and compare them.

```
freqs1, psd1 = signal.welch(u1, fs=12,
    nperseg=len(u1)//8, scaling='spectrum')
freqs2, psd2 = signal.welch(u2, fs=12,
    nperseg=len(u2)//8, scaling='spectrum')
```



sin/cos at 1 cycle/yr

peaks at 2, 3, 4, 5, 6 cycles per year

harmonics of 1 cycle/yr.

non-sinusoidal periodic behavior.

Complete removal of seasonal frequencies.

Comments on the smoothed periodogram

Question. Why does the unadjusted spectrum have peaks at 2,3,4,5,6 cycles per year as well as 1 cycle per year?

Question. Comment on what you learn from comparing these smoothed periodograms.

see previous slide.

Definition: The ratio of the periodograms of the smoothed and unsmoothed time series is the **frequency response function** of the smoother.

- The frequency response function tells us how much the smoother contracts (or inflates) the sine and cosine components at each frequency ω.
- A frequency response may involve change in phase as well as magnitude, but here we consider only magnitude.
- Linear, time invariant transformations do not move power between frequencies, so they are characterized by their frequency response function.

If we scale or shift the data, the smoothed estimate should have the same scale or shift. A smooth approximation to the sum of two time series should be approximately the sum of the two smoothed series. So, smoothers are approximately **linear and time invariant**.

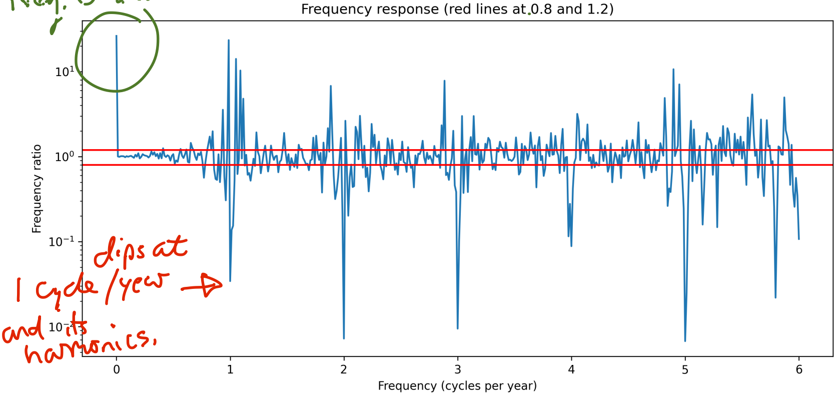
If a system moves power between frequencies, it must be nonlinear,

Calculating a frequency response function

We investigate the frequency response of the smoother used by Bureau of Labor Statistics to deseasonalize unemployment data.

```
# Compute unsmoothed periodograms and their ratio
freqs_u1, psd_u1 = signal.periodogram(u1,fs=12,scaling='spectrum')
freqs_u2, psd_u2 = signal.periodogram(u2,fs=12,scaling='spectrum')
freq_response = psd_u2 / psd_u1
```

Freq. 0 should be removed



dips at 1 cycle/year →
and its harmonics.

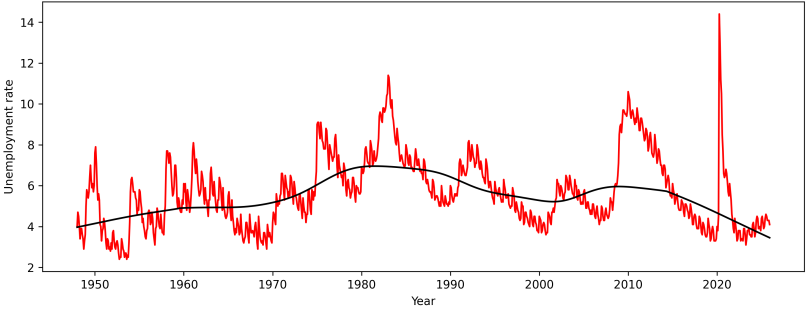
Question. What do you learn from this frequency response plot?

The BLS deseasonalization shrinks seasonal frequencies by a factor of 10^{-1}–10^{-2}, and leaves all other frequencies almost unchanged.

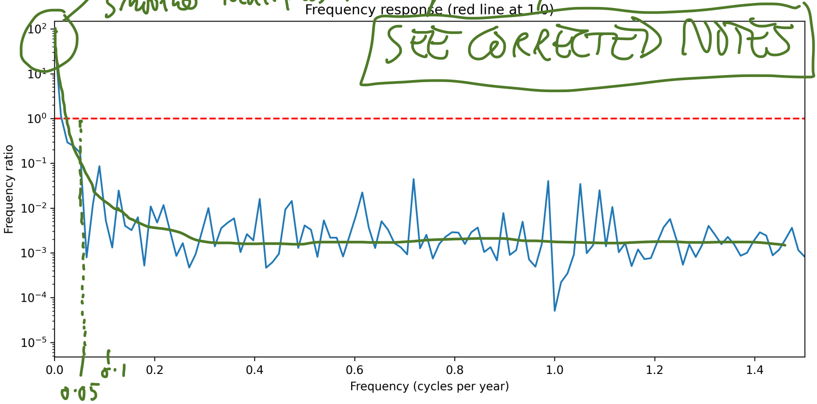
Estimating trend by Loess smoothing

- Loess is a *Local linear regression* approach (perhaps an acronym for LOcal Estimation by Smoothing) also known as *Lowess* (perhaps LOcally WEighted Sum of Squares).
- At each point in time, Loess computes a linear regression (a constant, linear or quadratic trend estimate) using only neighboring times.
- We can imagine a moving window of points included in the regression.
- `lowess` is a Python implementation from statsmodels, with the fraction of points included in the moving window being controlled by the `frac` parameter.
- We can choose a value of the span that visually separates long term trend from business cycle.

A Loess smooth of unemployment



We compute the frequency response function for the Lowess smooth: we don't really think the smoother multiplies low frequencies by 100. This is an artifact.



SEE CORRECTED NOTES

Question. Describe the frequency domain behavior of this filter.

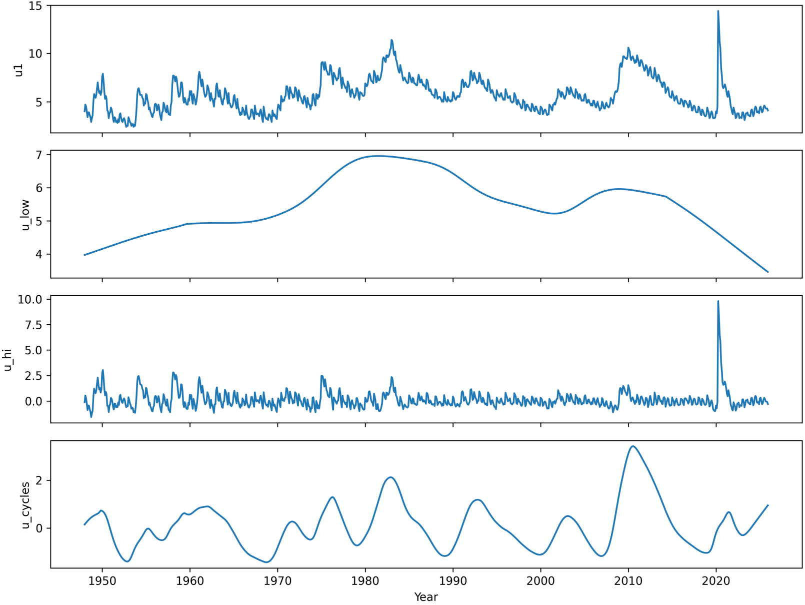
This smoother is preserving ~~for~~ cycles of 20 yr or longer (frequencies of 1/20 yr^{-1} or smaller). Other cycles are shrunk by a factor of $\approx 10^{-3}$.

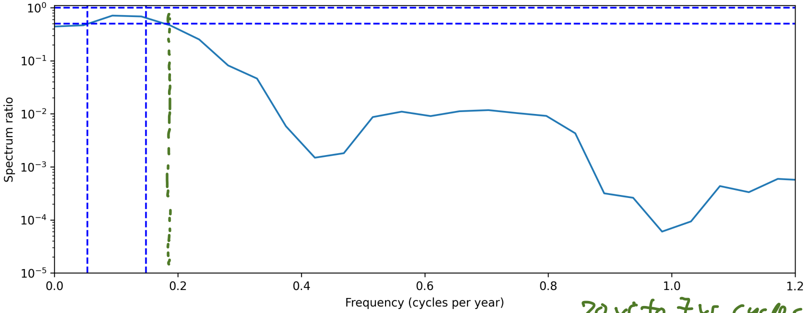
Extracting business cycles: A band pass filter

- For the unemployment data, high frequency variation might be considered “noise” and low frequency variation might be considered trend.
- A band of mid-range frequencies might be considered to correspond to the business cycle.
- We build a smoothing operation in the time domain to extract business cycles, and then look at its frequency response function.

```
u_low = lowess(u1, date, frac=0.3)[:, 1]
u_hi = u1 - lowess(u1, date, frac=0.05)[:, 1]
u_cycles = u1 - u_hi - u_low
```

Decomposition of unemployment as trend + noise + cycles





← 20 yr to 7 yr cycles, or 6 for .....

Frequency range (ratio > 0.5): [0.053, 0.148]

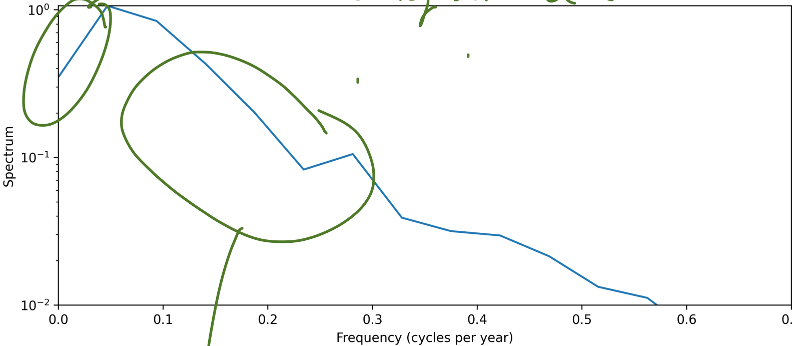
Question. Describe the frequencies (and corresponding periods) that this decomposition identifies as business cycles.

This decomposition proposes that cycles of 6-20 yr count as "business cycles".

Economists might want 4 yr cycles to count as business cycles.

Below is a smoothed periodogram for the raw unemployment data, plotted up to 0.7 cycles per year to focus on relevant frequencies.

0 freq should be removed



Question. Comment on the evidence for and against the concept of a business cycle in the above figure.

No clear peaks in "business cycle frequencies", so the economy does not resonate at clear, specific frequencies.

Common smoothers in Python

- Above, we have used the *local regression smoother* `lowess` from statsmodels, but there are other similar options.
- `scipy.ndimage.gaussian_filter1d` is a *Gaussian kernel smoother*. The default periodogram smoother in `signal.welch` is also a form of kernel smoothing.
- `scipy.interpolate.UnivariateSpline` is a *spline smoother*.
- You can learn about alternative smoothers, and try them out if you like, but `lowess` is a good practical choice for many smoothing applications.

Bandwidth for a smoother

- All these smoothers have some concept of a *bandwidth*, which is a measure of the size of the neighborhood of time points in which data affect the smoothed value at a particular time point.

- The concept of bandwidth is most obvious for kernel smoothers, but exists for other smoothers.

- We usually only interpret bandwidth up to a constant. For a particular smoothing algorithm and software implementation, you learn by experience to interpret the comparative value. Smaller bandwidth means less smoothing.

- Typically, when writing reports, it makes sense to focus on the tuning parameter for the smoother in question, which is not the bandwidth unless you are doing kernel smoothing.

Further reading

- Section 2.3 of Shumway and Stoffer [1] discusses smoothing of time series, in the time domain.
- Section 4.2 of Shumway and Stoffer [1] presents a frequency response function for linear filters, related to this chapter but in a different context.

References and Acknowledgements

[1] Robert H Shumway and David S Stoffer. *Time Series Analysis and its Applications: With R Examples*. 4th. Springer, 2017.

- Compiled on February 9, 2026using Python.
-
- We acknowledge previous versions of this course.

[1] Robert H Shumway and David S Stoffer. *Time Series Analysis and its Applications: With R Examples*. 4th. Springer, 2017.