Time Series Analysis
Lecture 2

Regression With Time Series, An Introduction to Exploratory Time Series Data Analysis and Time Series Smoothing

datascience@berkeley

Time-Series Smoothing Techniques: Examples Using a Real-World Time Series

Initial Unemployment Insurance Claim

- This example make uses of the initial unemployment insurance claim (or initial claims) data collected by the Bureau of Labor Statistics (BLS) of the U.S. Department of Labor.

- This is one of the most watched economic measures by policy makers, business leaders, economists, professionals in many fields, and consumers. It tracks the number of people who have **filed jobless claims for the first time during the specified period** with the appropriate government labor office. This number represents an inflow of people receiving unemployment benefits.

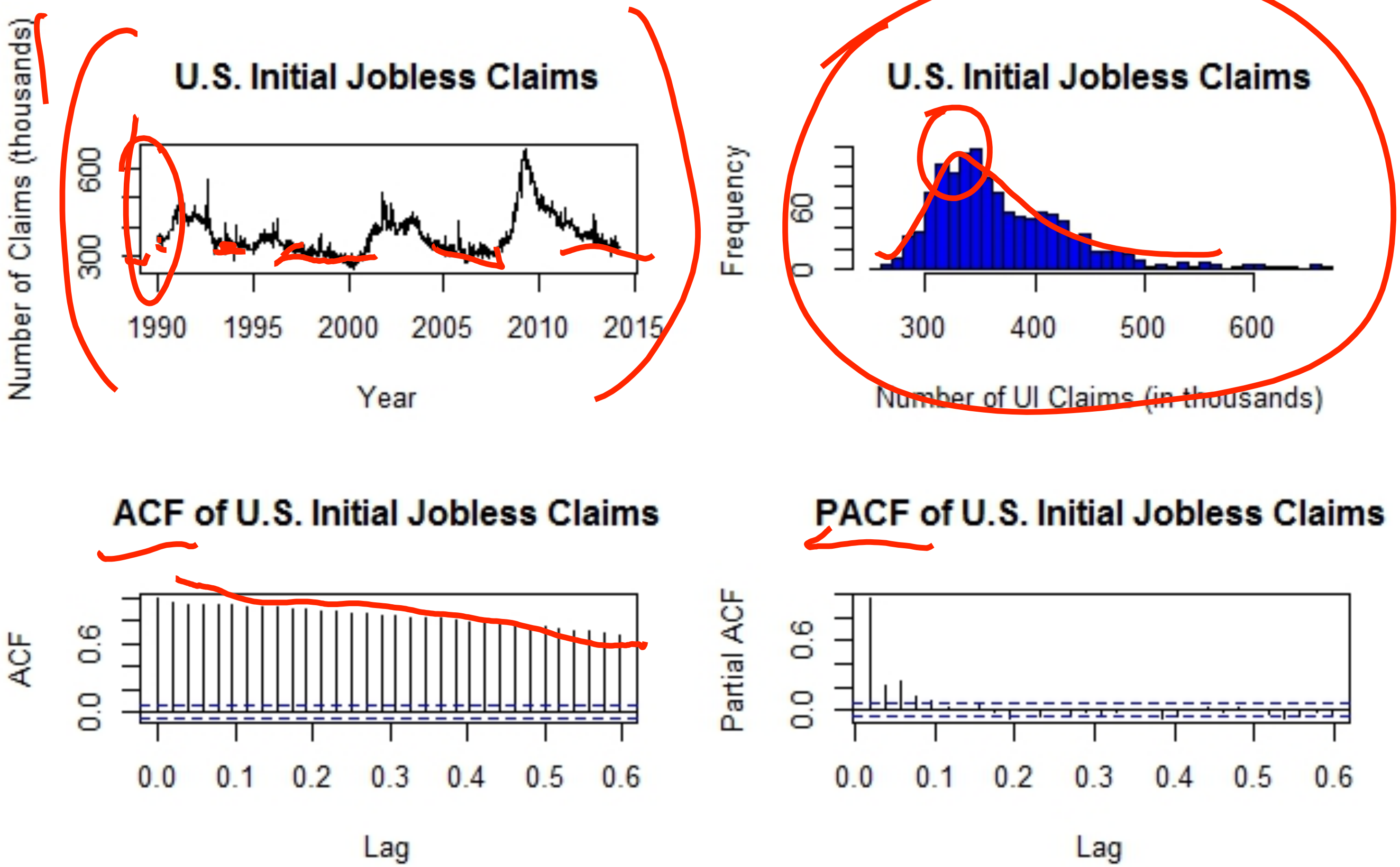
- **Definition of initial UI claims:** "The Department of Labor's Unemployment Insurance (UI) programs provide unemployment benefits to eligible workers who become unemployed through no fault of their own, and meet certain other eligibility requirements." (http://www.dol.gov/dol/topic/unemployment-insurance/)

- The **UI weekly claims data** are used in current economic analysis of unemployment trends in the nation and in each state. Initial claims measure emerging unemployment, and continued weeks claimed measure the number of persons claiming unemployment benefits.

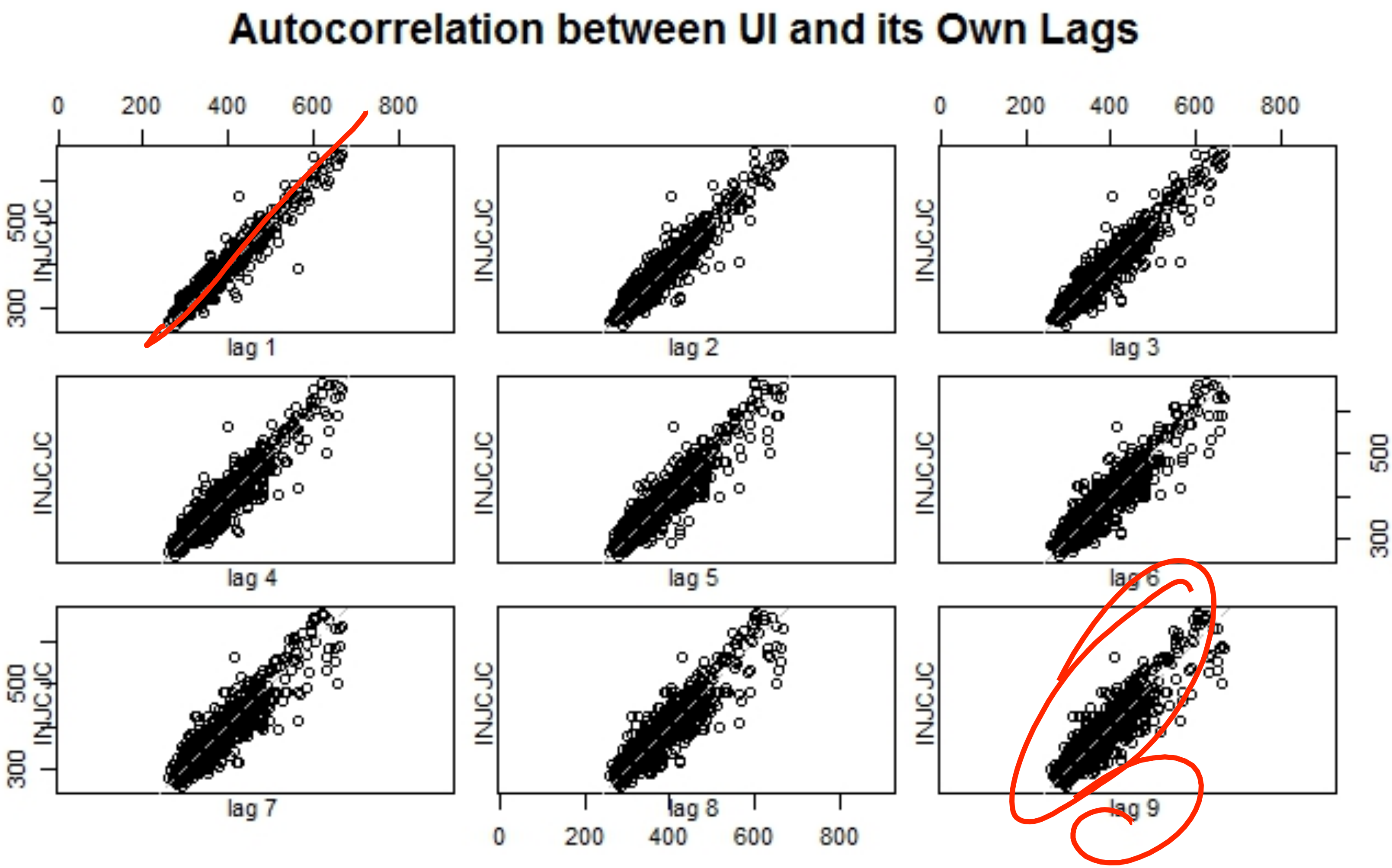
- After importing the data to R, we first look at the basic structure of the data, such as number of variables, number of observations, time intervals, and total time period it is covered.
- This is an important practice and can be used to check if the dataset loaded to R is the same as the original data. It is particular important if the dataset is large.

```
'data.frame': 1300 obs. of  3 variables:
 $ Date   : chr  "5-Jan-90" "12-Jan-90" "19-Jan-90" "26-Jan-90" ...
 $ INJCJC : int  355 369 375 345 368 367 348 350 351 349 ...
 $ INJCJC4: num  362 366 364 361 364 ...
> dim(data1)
[1] 1300    3
> head(data1)
       Date INJCJC  INJCJC4
1  5-Jan-90    355   362.25
2 12-Jan-90    369   365.75
3 19-Jan-90    375   364.25
4 26-Jan-90    345   361.00
5  2-Feb-90    368   364.25
6  9-Feb-90    367   363.75
```

- Next we will look at the time plot, density plot, and scatter plot matrix of the current vs. the lag values, ACF, and PACF.
- Since we have already introduced the initial UI data series earlier in the lecture, I will not spend time on these graphs.



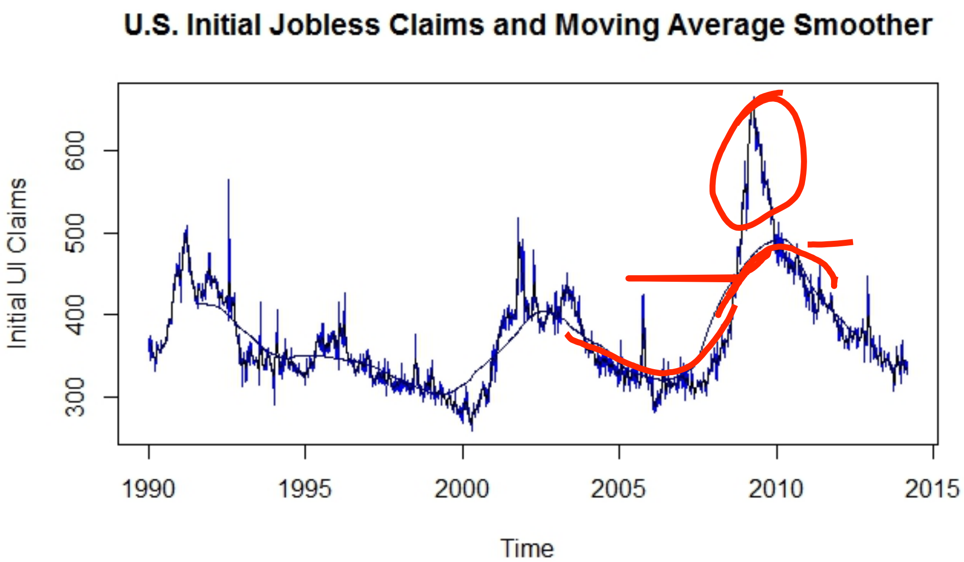
- Next we will look at the time plot, density plot, and scatter plot matrix of the current vs. the lag values, ACF, and PACF.



Symmetric Moving Average Smoother

- Recall that when applying a smoothing technique, we need to pay attention to the smoother parameter. Some smoothing techniques require an explicit specification of the smoothing parameter.
- In the case of moving average smoother, the "implicit" smoothing parameter is the number of values used in calculating each of the moving average value.

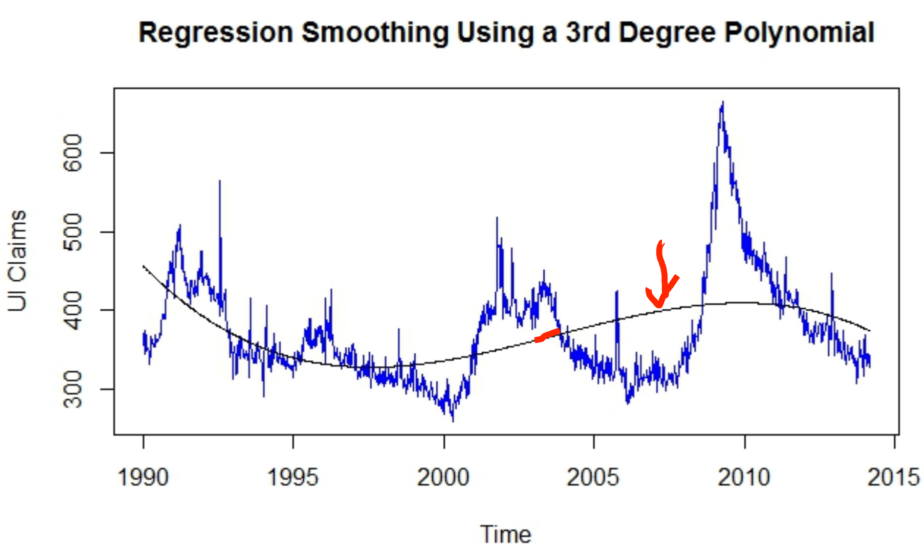
```
ma5 = filter(INJCJC, sides=2, rep(1,5)/5)
ma157 = filter(INJCJC, sides=2, rep(1,157)/157)
```



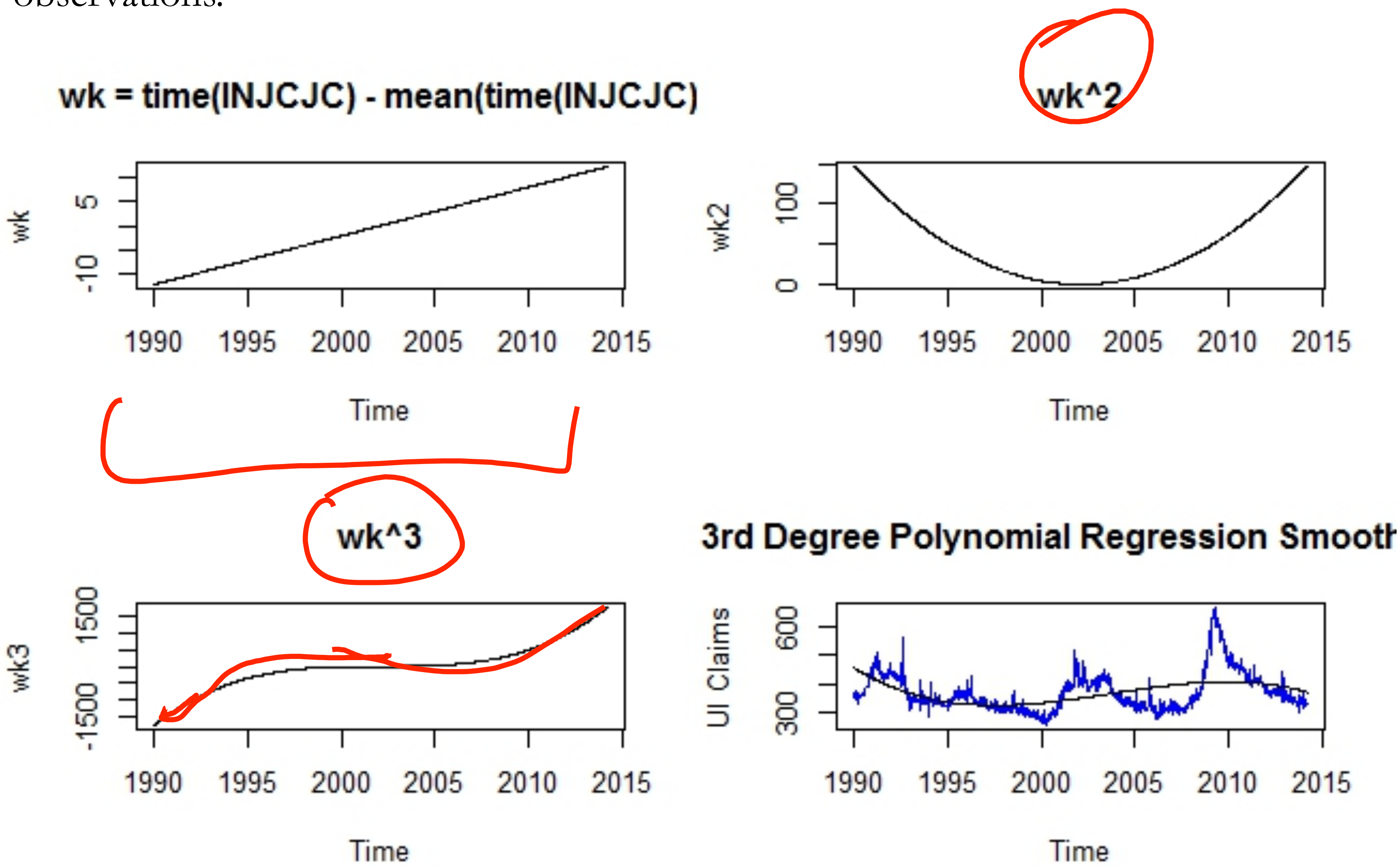
Regression Smoother

- A regression smoothing using a third-degree polynomial regression.
- It could capture the general shape of the series at the first 10 years of the historical observations, but it does not capture well the trend dynamics in the last 15 years of the historical observation period.

```
wk = time(INJCJC) - mean(time(INJCJC))
wk2 = wk^2
wk3 = wk^3
reg1 = lm(INJCJC~wk + wk2 + wk3, na.action=NULL)
plot(INJCJC, type="l", col="blue",
     main="Regression Smoothing Using a 3rd Degree Polynomial",
     ylab="UI Claims")
lines(fitted(reg1), col="black")
```



- A regression smoothing using a third-degree polynomial regression.
- It could capture the general shape of the series at the first 10 years of the historical observations.



Spline Smoothers

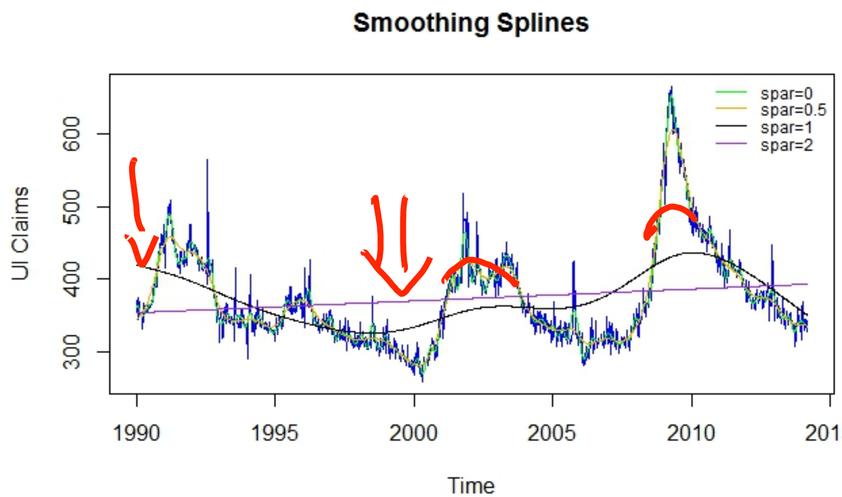
- Analogous to the role bandwidth plays in a kernel smoother, lambda is the smoothing parameter in the spline smoothers.
- We estimate four spline smoothers using different smoother parameter values.

```
plot(INJCJC, type="l", col="blue",
     main="Smoothing Splines",
     ylab="UI Claims")
lines(smooth.spline(time(INJCJC), INJCJC, spar=0)  , col="green")
lines(smooth.spline(time(INJCJC), INJCJC, spar=0.5), col="orange")
lines(smooth.spline(time(INJCJC), INJCJC, spar=1)  , col="black")
lines(smooth.spline(time(INJCJC), INJCJC, spar=2)  , col="purple")
```

$$\sum_{t=1}^{n} [x_t - f_t]^2 + \lambda \int \left(f_t''\right)^2 dt$$

f_t is a cubic spline with a knot at each t

λ is the smoothing parameter

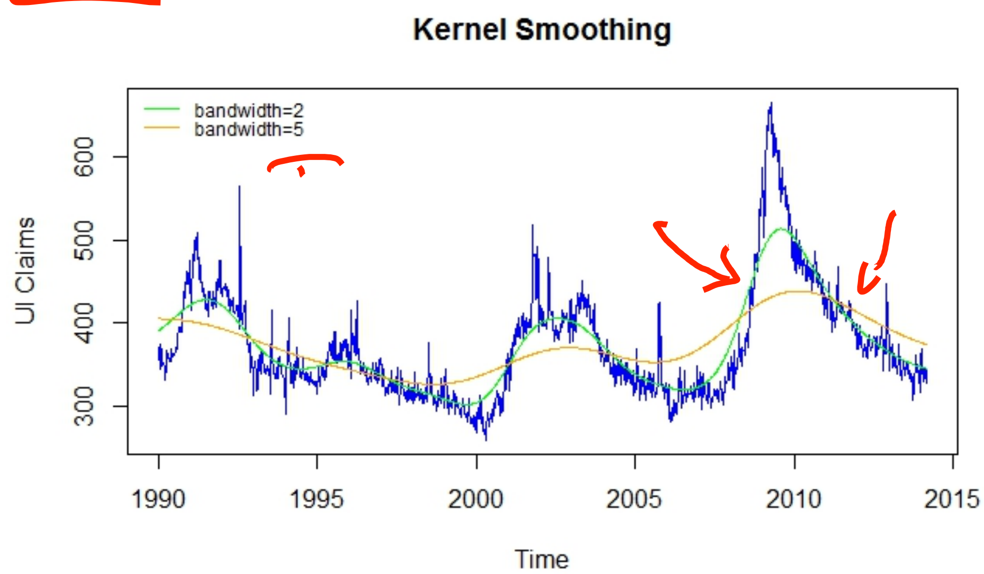


Kernel Smoothers

- For kernel smoother, the choice of bandwidth is critical, as it has a great influence on the smoothness of the kernel estimates.
- The choice of the kernel function, however, is less important.

$$\hat{f}_t = \sum_{i=1}^{n} w_i(t) x_i \qquad w_i(t) = \frac{K\left(\frac{t-i}{b}\right)}{\sum_{j=1}^{n} K\left(\frac{t-i}{b}\right)}$$

```
lines(ksmooth(time(INJCJC), INJCJC, "normal", bandwidth=2))
lines(ksmooth(time(INJCJC), INJCJC, "normal", bandwidth=5))
```



Berkeley SCHOOL OF INFORMATION