Time Series Regression and Forecasting

Juergen Meinecke

Roadmap

Practical Guidelines and Example

Practical Guidelines for Time Series Analysis

Suppose I give you a time series Y_t and ask you to produce forecasts, what should you do?

Let's try to tie up all loose ends and come up with some practical advice

Given the time series Y_t, here's what you should do to produce the best forecast...

- Graph it

  Sometimes you can spot deterministic trends in graphs (e.g., GDP)

  A graph will definitely not help you spot a stochastic trend
- Run an ADF unit root test on Y_t

- If you reject the null hypothesis of a unit root:
  Estimate the AR(p) model for Y_t:

$$Y_t = \beta_0 + \alpha t + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + u_t$$

  (It's always safe to include a deterministic time trend)
  You can easily estimate this in `Python`
  Use the AR(p) model for Y_t to produce your forecasts

- If you cannot reject the null hypothesis of a unit root:
  Generate first differences ΔY_t of the data and run a unit root test on ΔY_t
  In 90% of practical situations, you will reject the null hypothesis of a unit root in ΔY_t (in other words, first differencing effectively removes the unit root)
  This suggests that you should use an AR(p) model for ΔY_t:

$$\Delta Y_t = \beta_0 + \alpha t + \beta_1 \Delta Y_{t-1} + \beta_2 \Delta Y_{t-2} + \ldots + \beta_p \Delta Y_{t-p} + u_t$$

  Since ΔY_t is stationary, you can easily estimate this in `Python`
  Use the AR(p) model for ΔY_t to produce your forecasts

Time Series Regression and Forecasting

Juergen Meinecke

Roadmap

Practical Guidelines and Example

Example: CPI and Inflation

During the last two weeks I have used inflation as the running example to illustrate important concepts

Recall that we defined inflation to be

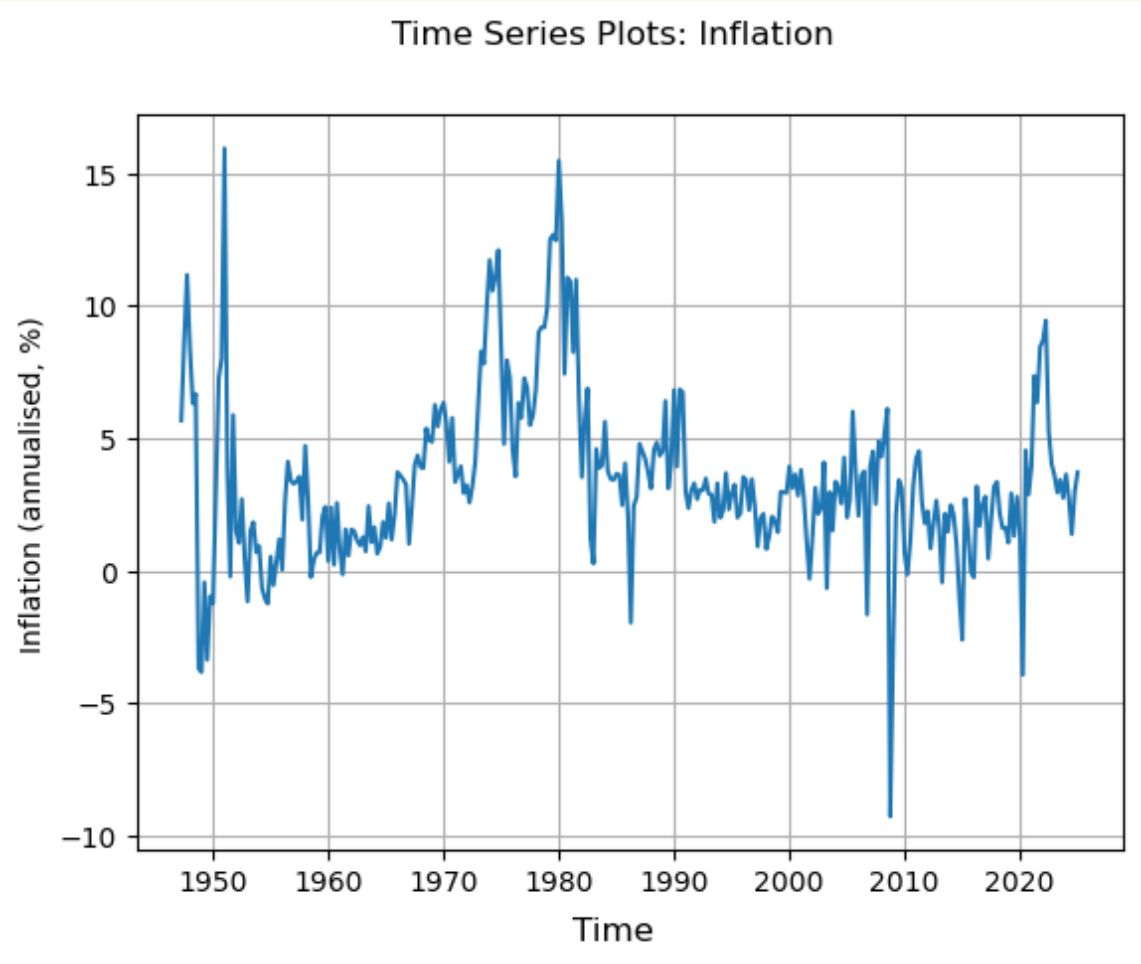
$$\texttt{infl}_t := 400 \cdot \left(\ln(\texttt{cpi}_t) - \ln(\texttt{cpi}_{t-1})\right)$$

(this is the *annualized* version of quarter-to-quarter inflation; but it would work just the same if we hadn't used the annualized version)

Our goal:
Come up with the best AR(p) model to forecast inflation

Let's go through the steps

Graphing the data



Deterministic trend: not obvious

Stochastic trend: never obvious

Test for unit root

A few slides ago...

Python Code

```
> from statsmodels.tsa.stattools import adfuller
> adfuller(df.infl.dropna(), maxlag = 4, regression='ct')

(-4.425190419784897,
0.001995606026544201,
4,
307,
'1%': -3.9885651881396162,    '5%': -3.424888419060633,    '10%': -3.135514568313964,
1330.8744210409302)
```

Dickey Fuller critical value is -3.41

Reject the null hypothesis of unit root here

Therefore, our best time series model is this one:

$$\texttt{infl}_t = \beta_0 + \alpha t + \beta_1 \texttt{infl}_{t-1} + \beta_2 \texttt{infl}_{t-2} + \beta_3 \texttt{infl}_{t-3} + \beta_4 \texttt{infl}_{t-4} + u_t$$

Let's estimate it...

Python Code (output edited)

```
> ar4trend = smf.ols('infl ~ l1infl + l2infl + l3infl + l4infl + trend',
                     data=df, missing='drop').fit(use_t=False)
> print(ar4trend.summary())

OLS Regression Results
==============================================================================
Dep. Variable:                   infl   R-squared:                       0.559
Model:                            OLS   Adj. R-squared:                  0.551
Method:                 Least Squares   F-statistic:                     76.48
No. Observations:                 308
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.7484      0.296      2.525      0.012       0.167       1.329
l1infl         0.6176      0.057     10.886      0.000       0.506       0.729
l2infl         0.0073      0.064      0.114      0.909      -0.118       0.133
l3infl         0.3138      0.064      4.887      0.000       0.188       0.440
l4infl        -0.1671      0.056     -2.976      0.003      -0.277      -0.057
trend       4.205e-05      0.001      0.031      0.975      -0.003       0.003
==============================================================================
```

What should we do with these estimates now?

Reminder:
the time series `CPIAUCSL` ran from 1947:Q1 to 2025:Q1

That's 313 observations on `CPIAUCSL`

That's 312 observations on `infl`

Remember that our main goal is to produce forecasts

Let's produce an inflation forecast for 2025:Q2

It's very easy given our AR(4) estimates

Recall the observed data on inflation

Python Code

```
> df.infl.tail(5)
quarter
2024Q1    3.638668
2024Q2    2.720218
2024Q3    1.386306
2024Q4    2.988335
2025Q1    3.714311
```

Obtain forecast by plugging into the estimated model:

$$\begin{aligned}\widehat{\texttt{infl}}_{2025{:}Q2|2025{:}Q1} &= 0.7484 + 0.00004205 \cdot 312 \\ &\quad + 0.6176 \cdot (3.7143) + 0.0073 \cdot (2.9883) \\ &\quad + 0.3138 \cdot (1.3863) - 0.1671 \cdot (2.7202) \\ &= 3.0578\end{aligned}$$

Time Series Regression and Forecasting

Juergen Meinecke

Roadmap

Practical Guidelines and Example

- Pseudo Out-of Sample Forecasts

The `CPIAUCSL` time series runs from 1947:Q1 to 2025:Q1

To measure forecast performance, we could follow this little algorithm:

1. pretend like your data already ends in s=2005:Q1
2. estimate an AR(4) model for $\texttt{infl}_t$ for the time frame 1947:Q1 to s
3. calculate a forecast for period $s+1$: $\widehat{\texttt{infl}}_{s+1|s}$
4. compare the forecast to the actual realization $\texttt{infl}_{s+1}$; difference is the forecast error: $\texttt{infl}_{s+1} - \widehat{\texttt{infl}}_{s+1|s}$
5. shift s up by one period an jump back to step 2

Repeat until you reach end of your data set ($s \leqslant T$), with T :=2025:Q1

This algorithm creates a shifting or rolling one-quarter pseudo out-of-sample forecast

So we obtain lots of one-quarter forecasts of inflation for the time period 2005:Q2 to 2025:Q1

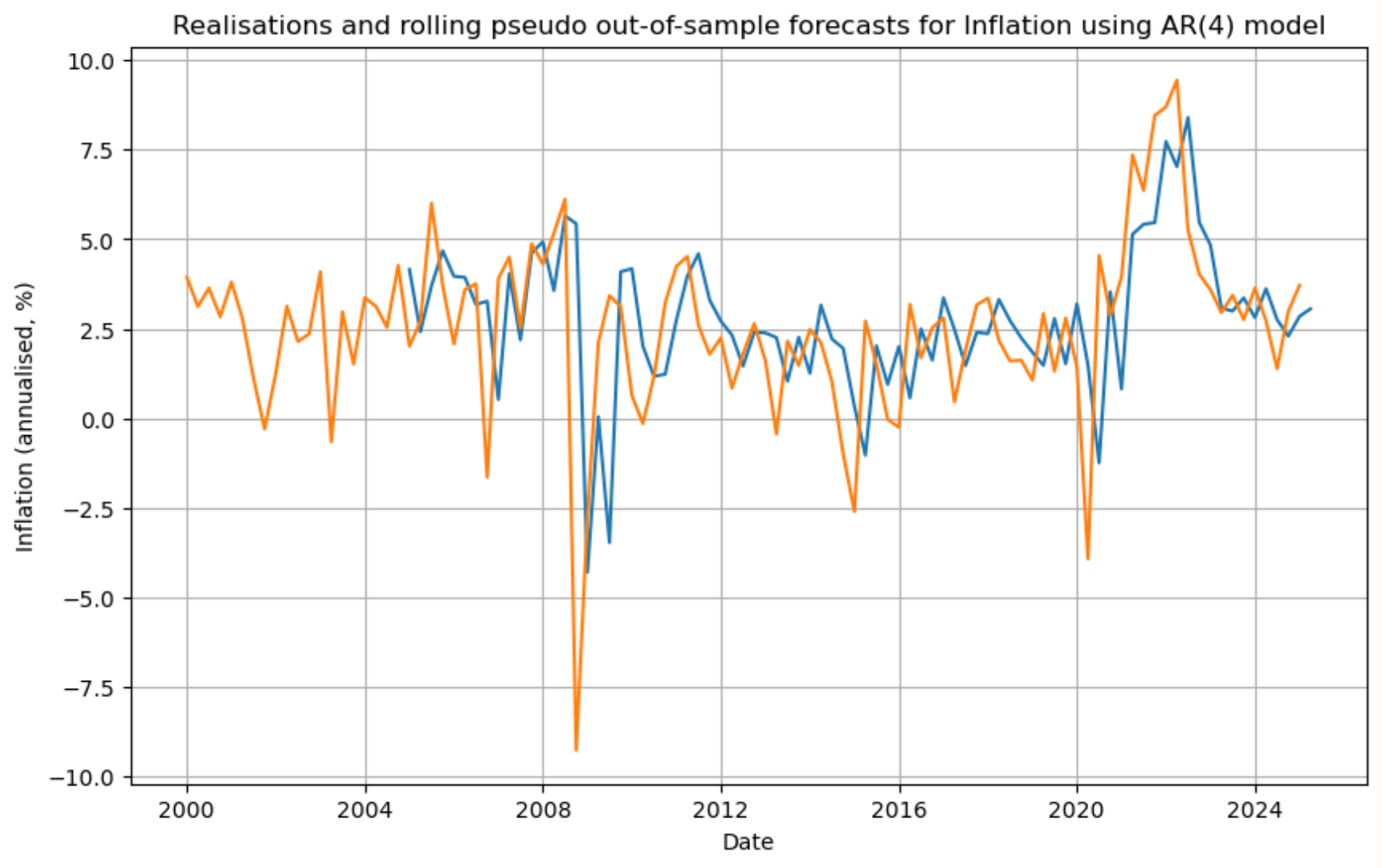
This puts us in a position to compare our one-quarter forecasts to the *actual* inflation numbers

This is a bit complicated to code in `Python`, so I won't ask you to do it yourselves

Nevertheless, I have implemented this algorithm in `Python` and ran it over the inflation data

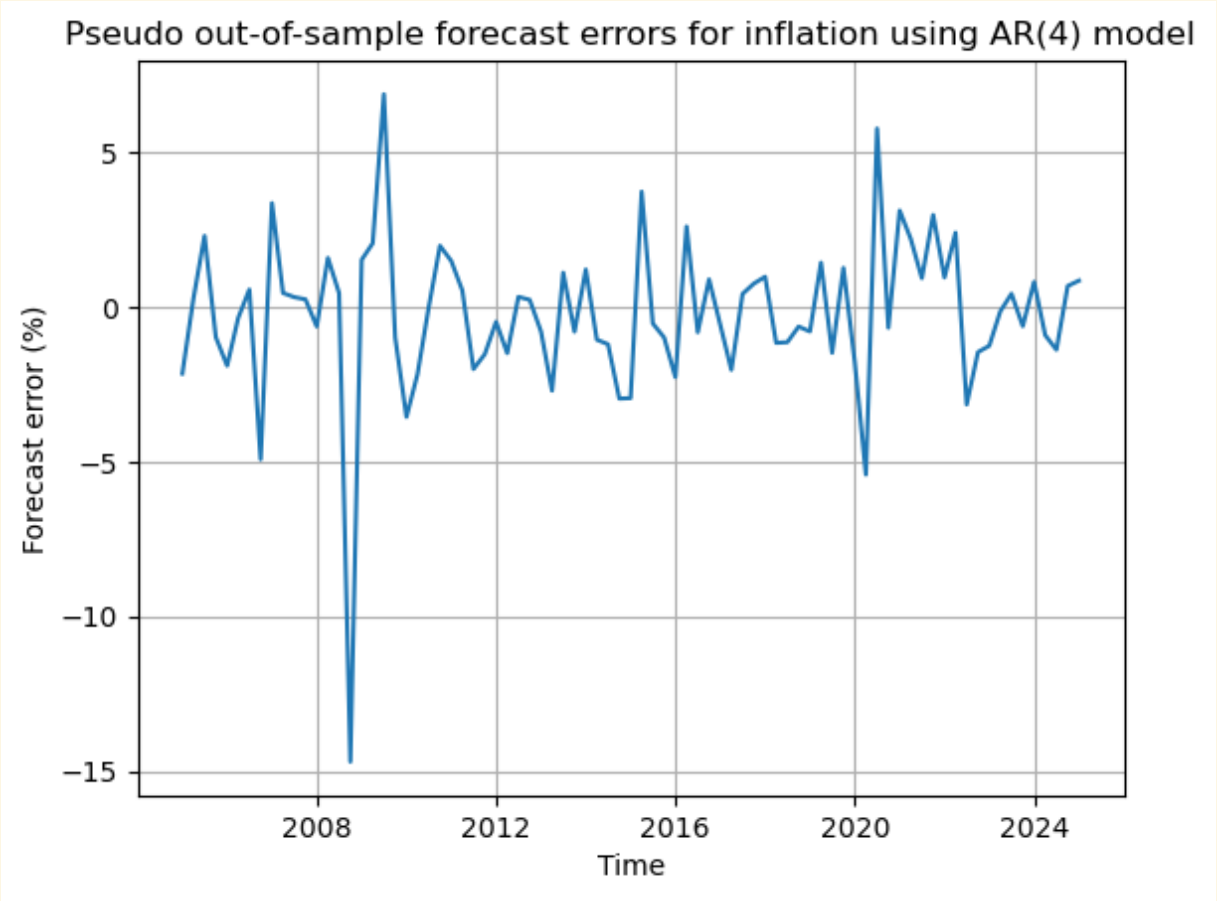
Here's the output in two pictures...

Inflation: actual versus forecast



Forecast error

(vertical difference between two lines from previous slide)



Is this a good model or a bad model?

This simple AR(4) model already does a reasonable job

And it took you only two and a half weeks of time series econometrics to be able to do this!

Where from here?

You could try different AR(p) models
(What's the optimal 'choice' of p?)

We could use an estimate of the so-called mean square forecast error (MSFE) to judge which model produces the best forecasts

You will learn much more on time series regression and forecasting in courses like EMET3007 or EMET3008

If you care more about the micro-econometric applications that we covered in weeks 1-9 of the semester, then EMET3004 or EMET3006 are right for you