

Time series: a Phillips curve on Swedish data

Programming & Data Management, Part II · Further topic F3
(optional, not examined)

Magnus Lodefalk
Örebro University

NA446A, Örebro University
Autumn 2026



Today's goals

By the end of today you can:

1. Rebuild Session 4's **quarterly series** from saved answers, and plot them before any regression
2. Compare **default, robust and HAC (Newey–West)** standard errors, and read the residuals' **autocorrelation** (Durbin–Watson, ACF)
3. Measure **persistence** with an AR(1), and say what a time-series regression **cannot** tell you

Six terms for this module

autocorrelation	the correlation of a series with itself k periods earlier
ACF	the autocorrelation at every lag k , drawn as bars
HAC (Newey–West)	standard errors that allow unequal variance and correlation over time, up to a chosen number of lags
Durbin–Watson	a quick test of lag-1 residual correlation: 2 means none, near 0 means strong
AR(1)	$u_t = \alpha + \rho u_{t-1} + e_t$: this quarter explained by the last
half-life	quarters until half of a shock has gone: $\ln 0.5 / \ln \rho$

- Nothing to install, no network: SCB's answers of 29 September are saved in `data/raw/scb/`.

The data and one regression

Step 1: recap, from saved answers to quarters

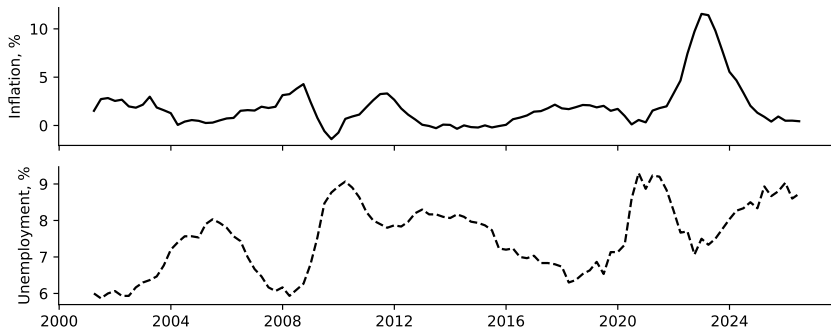
```
for path in sorted(RAW.glob("*.json")):
    monthly[path.stem] = read_scb(path)
q = pd.DataFrame({"cpi": to_quarters(monthly["cpi"]),
                  "unemp": to_quarters(monthly["unemp"])}))
q["infl"] = q["cpi"].pct_change(4) * 100
```

```
102 2001Q1 2026Q2
      infl  unemp
2026-03-31  0.50   8.60
2026-06-30  0.44   8.73
```

`read_scb` and `to_quarters` are Session 4's functions (mean of three months, complete quarters only); the loop also asserts that every month arrived.

- 102 quarters after `dropna()`: unemployment starts in 2001.

Step 2: look at both series before any regression



- Both move slowly: a high quarter is followed by another high quarter.

Inflation: CPI, same quarter a year earlier. Unemployment: ages 15–74, seasonally adjusted (SCB). The 2022–23 peak followed the energy-price shock.

Step 3: one coefficient, four standard errors

```
model = smf.ols("infl ~ unemp", data=q)
ses = {"default": model.fit().bse["unemp"]}
for L in LAG_GRID:
    ses[f"HAC {L}"] = model.fit(cov_type="HAC",
                                cov_kwds={"maxlags": L}).bse["unemp"]
```

```
coef: -0.605
se: default 0.243, HAC 0 0.114, HAC 4 0.178, HAC 12 0.174
Newey-West rule of thumb, floor(4(T/100)^(2/9)): 4
```

HAC 0 is the robust (White) error; 4 lags is the rule of thumb for $T = 102$.

- HAC is *smaller* than the default here, not always larger as the HT25 script said: it re-estimates the uncertainty, in either direction.

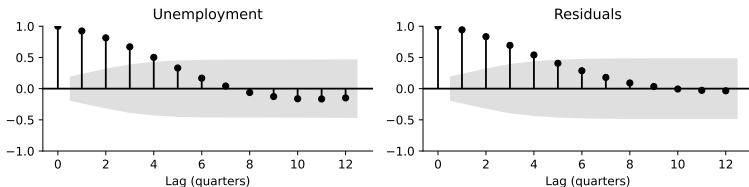
Stata: `reg infl unemp; newey infl unemp, lag(4): -0.605 (0.179)`; Stata's small-sample factor moves the third decimal.

Autocorrelation and persistence

Step 4: how correlated are the residuals over time?

```
rho_u = ols.resid.autocorr(1)  
durbin_watson(ols.resid)
```

Durbin-Watson: 0.11 | lag-1 autocorrelation: residuals 0.94 unemp 0.94



- Bars outside the grey band differ from zero: the residuals stay correlated for about five quarters.

Stata: `estat dwatson after reg: 0.111; ac unemp` for the plot.

HAC is a correction, not a cure

```
factor = np.sqrt((1 + rho_x * rho_u)
                  / (1 - rho_x * rho_u))
```

```
|implied se: 1.02 (default x 4.2)
```

A back-of-the-envelope rule: if both the regressor and the error follow an AR(1), the default variance is too small by $(1 + \rho_x \rho_u) / (1 - \rho_x \rho_u)$. With both ρ at 0.94, the standard error should be about four times the default.

- With a standard error near 1.0, a slope of -0.6 is indistinguishable from zero. Four lags of HAC do not capture correlation this persistent: 102 quarters hold far less independent information than 102 observations.

A rough rule, not a test: it assumes both series are AR(1). Its point is the order of magnitude.

Step 5: how persistent is unemployment? An AR(1)

```
ar1 = AutoReg(q["unemp"].to_numpy(),  
              lags=1).fit()  
rho = ar1.params[1]
```

```
| rho: 0.94 | se: 0.032 | after 4 quarters: 0.78 | half-life, quarters: 11.2
```

`params[1]` is positional, which works because `to_numpy()` hands `AutoReg` a plain array; on a pandas Series, write the label. The HT25 script's `ols.params[0]` on a labelled Series stops with a `KeyError` in pandas 3.

- $\rho = 0.94$: this quarter's unemployment is mostly last quarter's.

Stata: `reg unemp L.unemp` 0.940 (0.033).

Check your understanding

Answer on your own first, then compare with a neighbour

Unemployment is one point above its average. With $\rho = 0.94$, predict: how far above is it four quarters later, and after how many quarters is half the gap gone?

Check your understanding

Answer on your own first, then compare with a neighbour

Unemployment is one point above its average. With $\rho = 0.94$, predict: how far above is it four quarters later, and after how many quarters is half the gap gone?

Answer: $0.94^4 = 0.78$ points after a year; half is gone after $\ln 0.5 / \ln 0.94 \approx 11$ quarters, almost three years. A shock to Swedish unemployment is slow to fade, which is why the regression's residuals are so persistent too.

Step 6: in changes, and what it cannot say

```
dq = q[["infl", "unemp"]].diff().dropna()
fd = smf.ols("infl ~ unemp", data=dq).fit(
    cov_type="HAC", cov_kwds={"maxlags": MAXLAGS})
```

```
| coef: -1.037 | HAC se: 0.301 | N: 101 | lag-1 autocorrelation of residuals: 0.38
```

- In changes, the residuals are far less persistent (0.38), and quarters when unemployment rose were quarters when inflation fell
- Still a correlation: the Riksbank sets its rate in response to both series, and a price shock (2022) moves inflation and unemployment for its own reasons
- A time-series regression describes co-movement. A causal claim needs a design.

Stata: newey D.infl D.unemp, lag(4): -1.037 (0.304).

Summary, and where to go next

1. Saved answers, one function, `resample("QE"), pct_change(4)`; plot first
2. `fit(cov_type="HAC", cov_kwds={"maxlags": 4})`; read Durbin-Watson and the ACF; with residual autocorrelation of 0.94, treat every standard error here with caution
3. `AutoReg(..., lags=1)` for persistence; differences as a check; correlations, not effects

Read next (open, checked 30 Sep 2026):

- Turrell et al., *Coding for Economists*, "Time series":
aeturrell.github.io/coding-for-economists/time-series.html
- statsmodels, *Autoregressions* (AR models, lag choice, forecasts):
statsmodels.org/stable/examples/notebooks/generated/autoregressions.html