ARIMA models: Further analysis

EC 361–001

Prof. Santetti
Spring 2024

Materials

Required readings:

- `Hyndman & Athanasopoulos, ch. 9`
    - sections 9.5—9.6.

Motivation

Motivation

Our last lecture covered **three** main concepts:

1. *Differencing*;
2. *Autoregressive* models;
3. *Moving average* models.

If we put these pieces together, we come up with **A**uto**R**egressive **I**ntegrated **M**oving **A**verage models.

- Also known as **ARIMA** models.

Therefore, we will now focus on forecast methods for **stationary** (I) data, where we can use **lagged values** as predictors, be that from the variable **itself** (AR), or from **random** (MA) components.

Non-seasonal ARIMA models

Non-seasonal ARIMA models

A full **ARIMA(*p*,*d*,*q*)** model can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

where y_t is a *stationary* time series.

The ***(p,d,q)*** specification denotes the following:

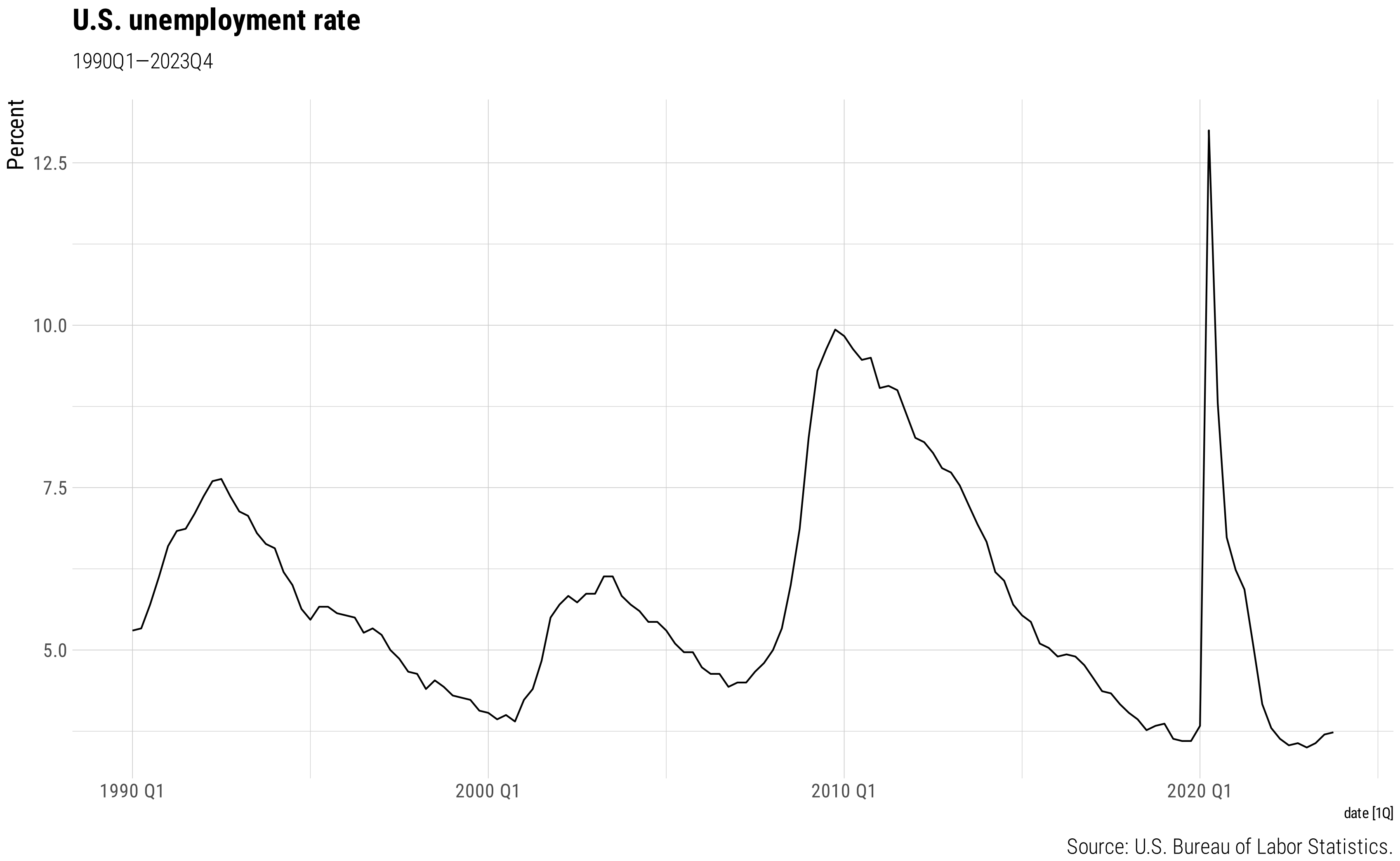
- ***p***: order of the **autoregressive** part;
- ***d***: degree of **differencing** in order to achieve *stationarity*;
- ***q***: order of the **moving average** part.

Non-seasonal ARIMA models

Given the **ARIMA** notation, how do your define the following:

- A *white noise* process?
- A *random walk* process?
- An *AR(2)* model?
- An *MA(4)* model?

Non-seasonal ARIMA models

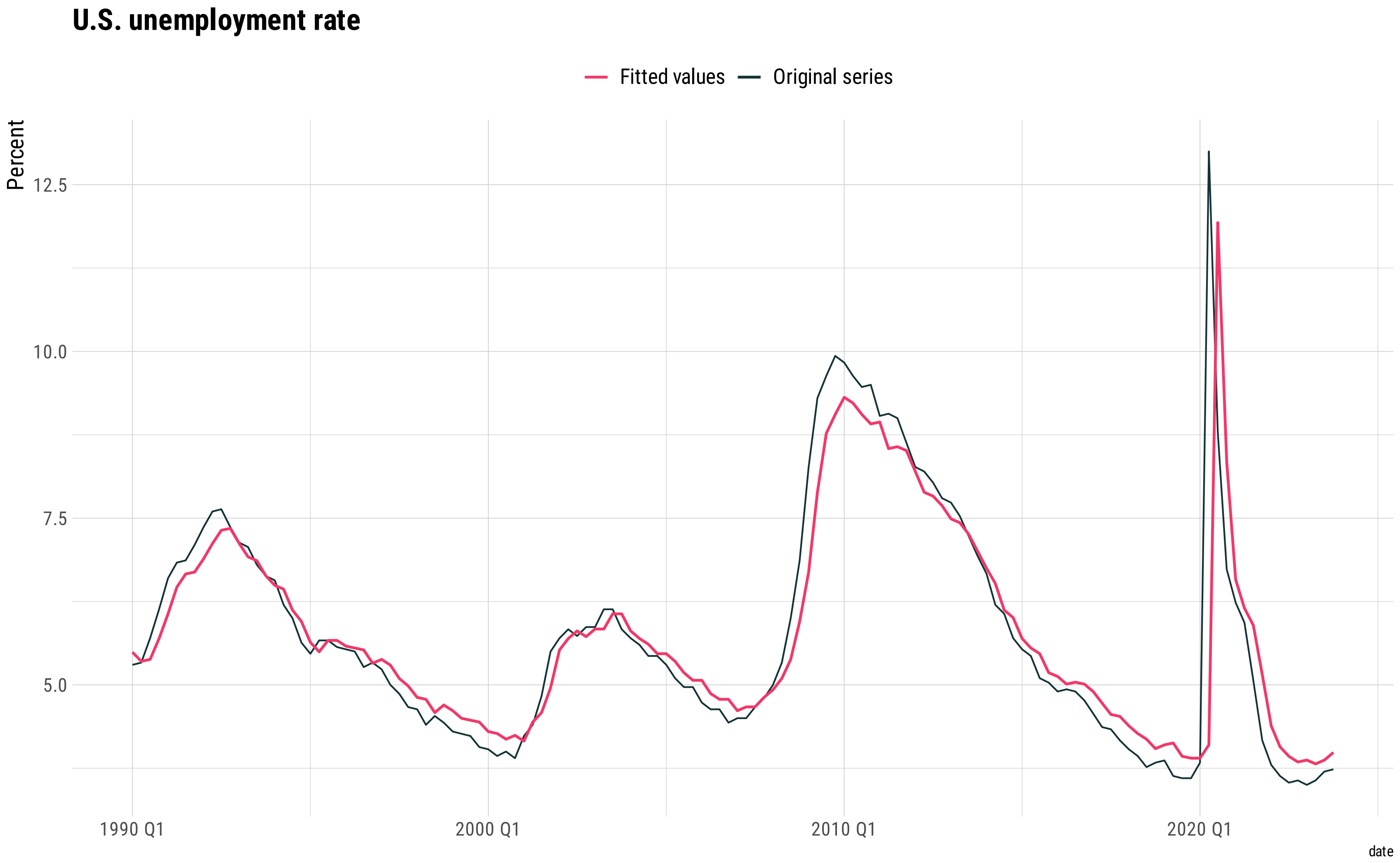


Non-seasonal ARIMA models

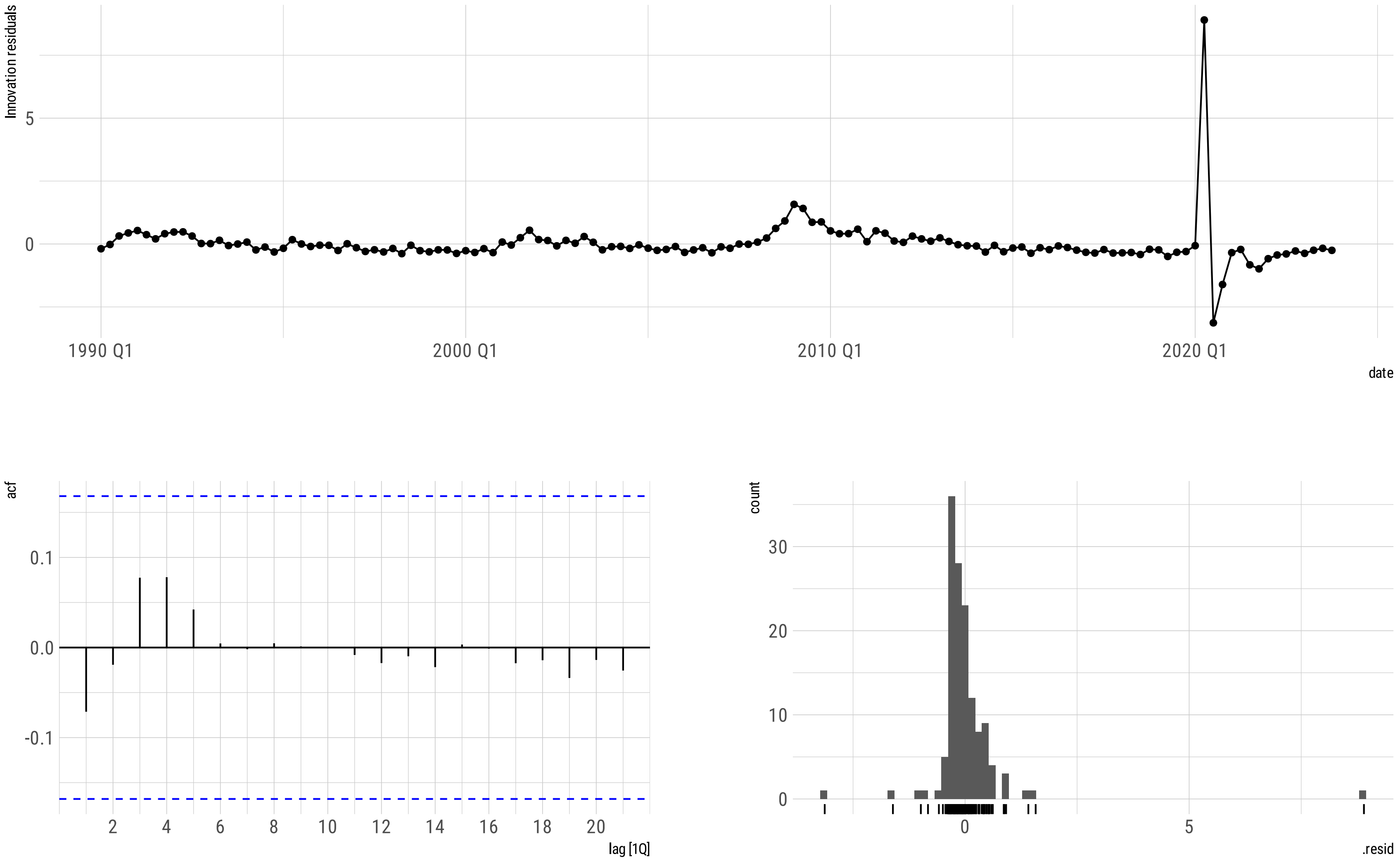
```
unemp_fit <- unemp_ts |>
  model(unemp_arima = ARIMA(unrate))

unemp_fit |>
  report()
```

```
#> Series: unrate
#> Model: ARIMA(1,0,0) w/ mean
#>
#> Coefficients:
#>          ar1  constant
#>       0.8545    0.8244
#> s.e.  0.0435    0.0740
#>
#> sigma^2 estimated as 0.8176:  log likelihood=-178.93
#> AIC=363.86   AICc=364.04   BIC=372.6
```

Non-seasonal ARIMA models



Non-seasonal ARIMA models



Non-seasonal ARIMA models

The **Portmanteau tests** are still valid for ARIMA model residuals:

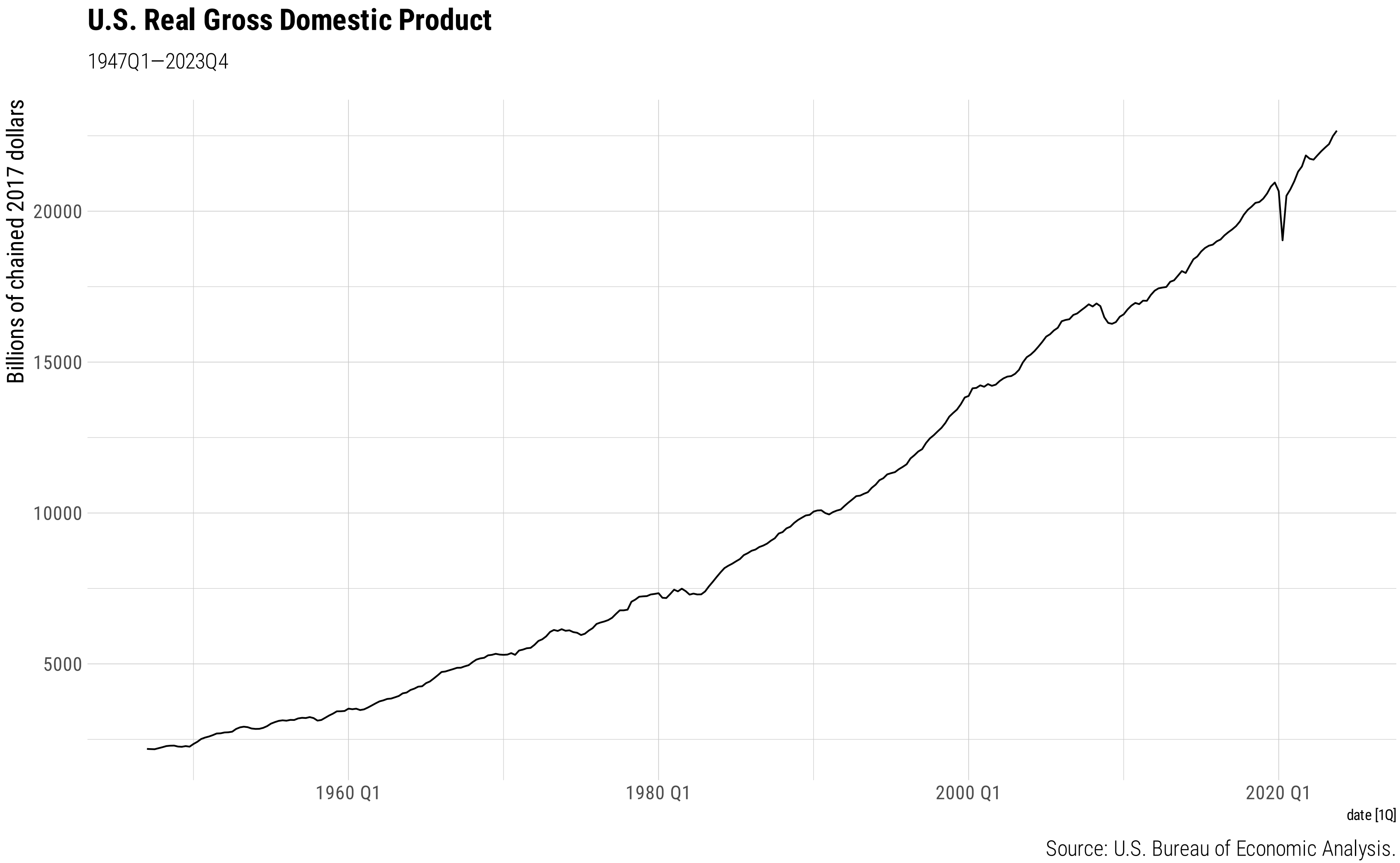
```
unemp_fit ▷
  augment() ▷
  features(.innov, ljung_box, lag = 10, dof = 1)
```

```
#> # A tibble: 1 × 3
#>   .model      lb_stat lb_pvalue
#>   <chr>         <dbl>     <dbl>
#> 1 unemp_arima    2.73     0.974
```

A more **precise** estimation of Portmanteau tests for ARIMA models can be obtained by including **degrees-of-freedom**:

- given by $p + q$.

Non-seasonal ARIMA models

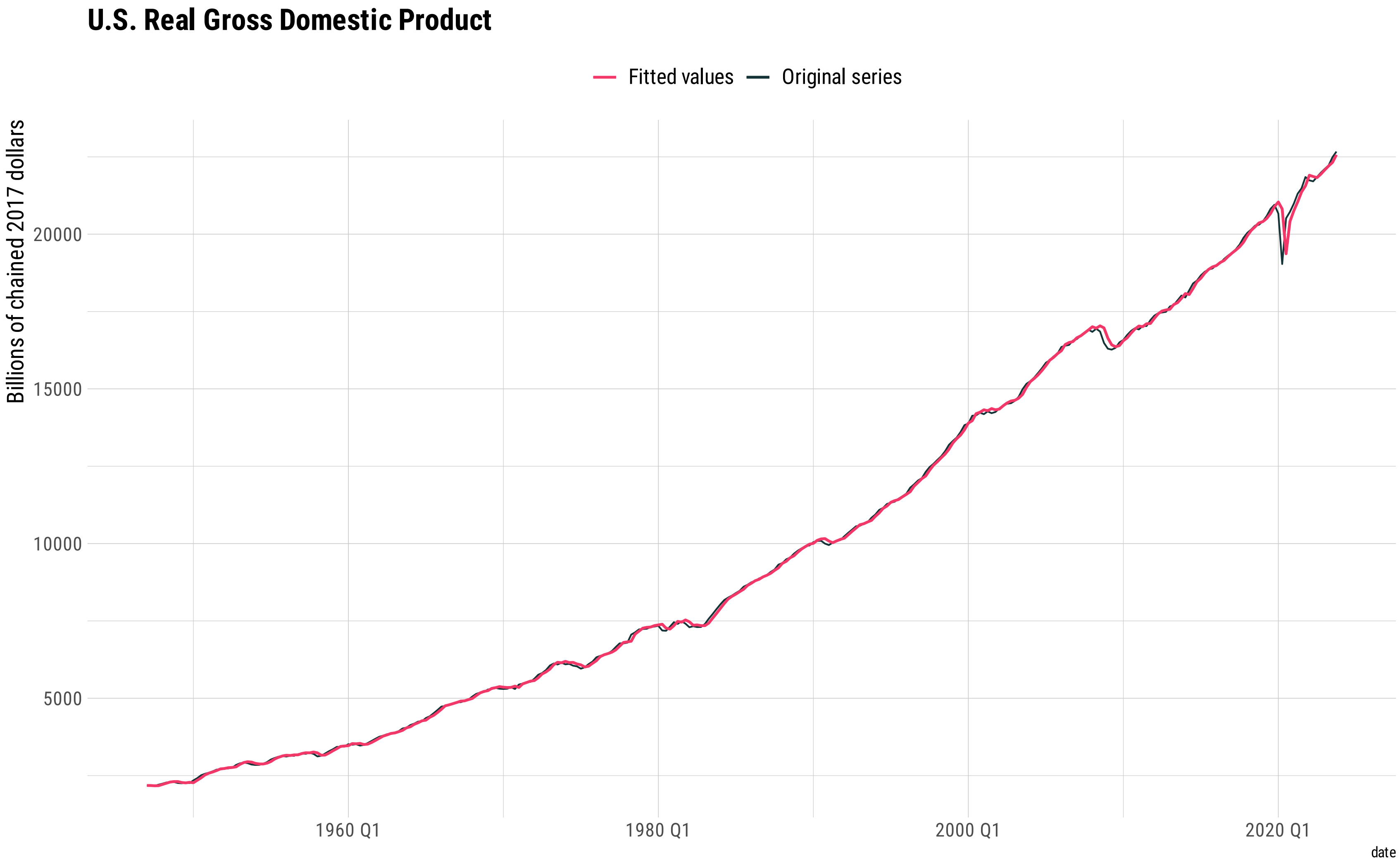


Non-seasonal ARIMA models

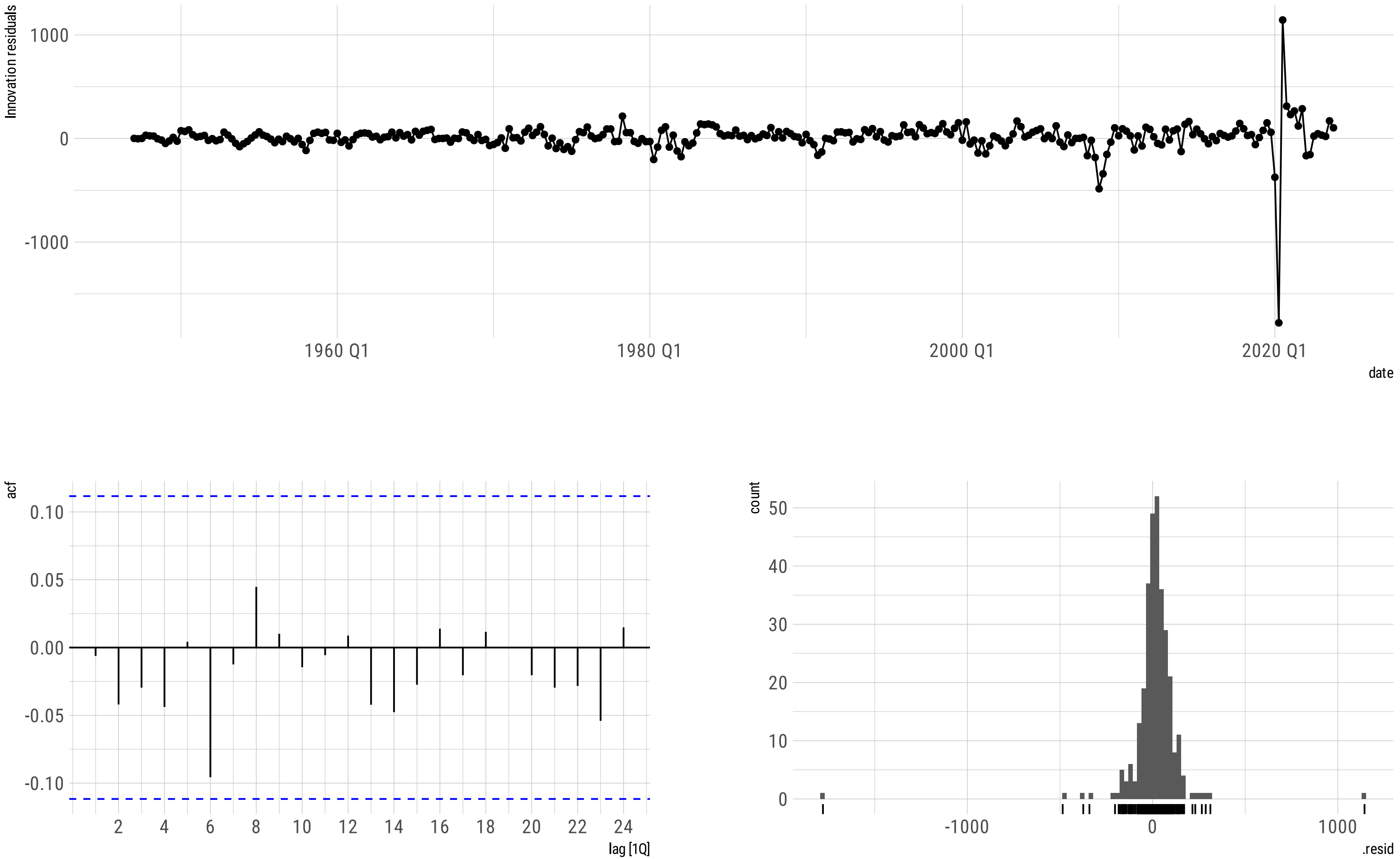
```
gdp_fit ← gdp_ts ▷
  model(gdp_arima = ARIMA(gdp))

gdp_fit ▷
  report()
```

```
#> Series: gdp
#> Model: ARIMA(0,2,2)
#>
#> Coefficients:
#>           ma1     ma2
#>       -1.1359  0.1556
#> s.e.   0.0589  0.0584
#>
#> sigma^2 estimated as 22105:  log likelihood=-1965.46
#> AIC=3936.91   AICc=3936.99   BIC=3948.08
```

Non-seasonal ARIMA models



Non-seasonal ARIMA models

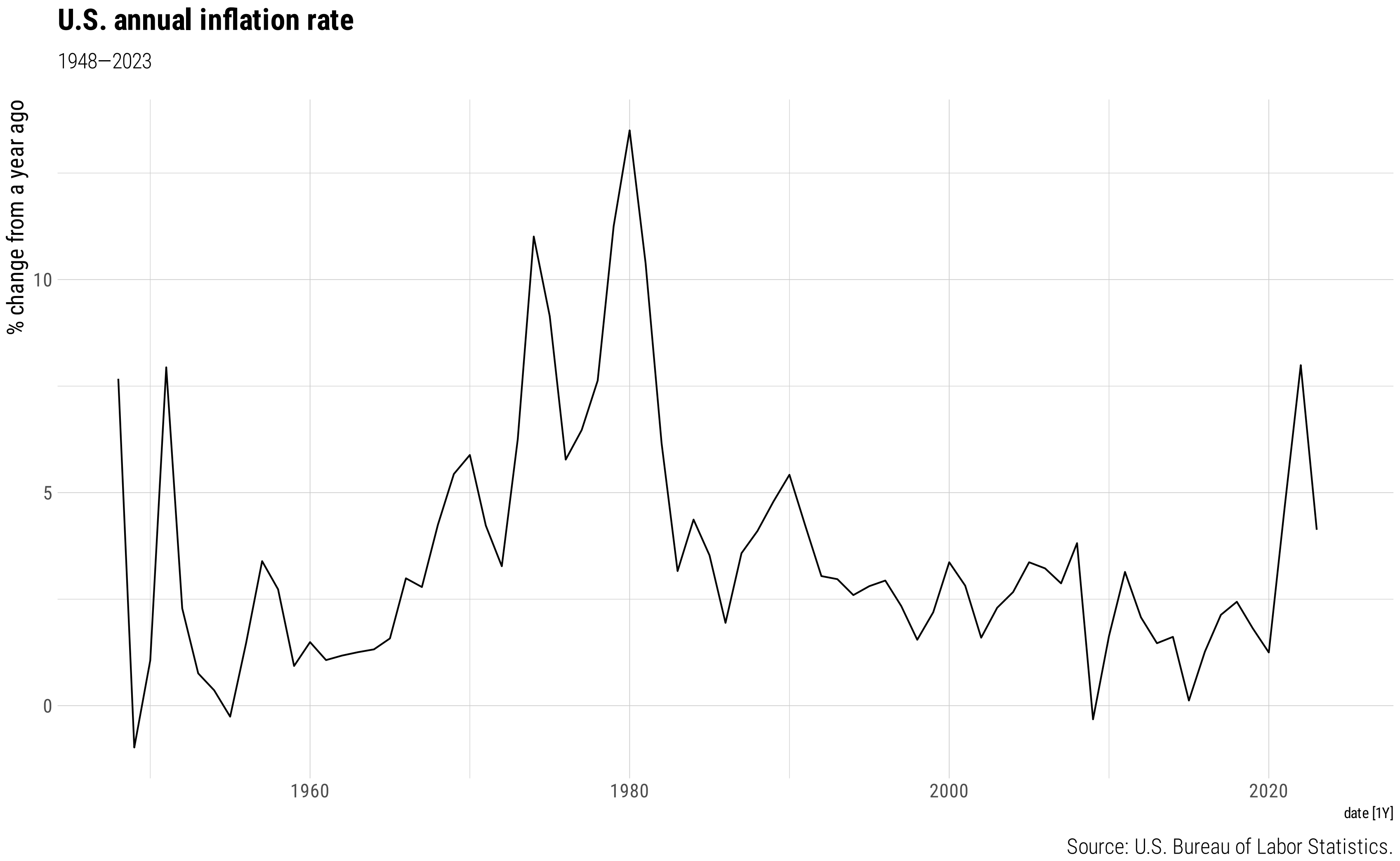


Non-seasonal ARIMA models

```
gdp_fit |>
  augment() |>
  features(.innov, ljung_box, lag = 10, dof = 2)
```

```
#> # A tibble: 1 × 3
#>   .model    lb_stat lb_pvalue
#>   <chr>       <dbl>     <dbl>
#> 1 gdp_arima    5.13     0.744
```

Non-seasonal ARIMA models

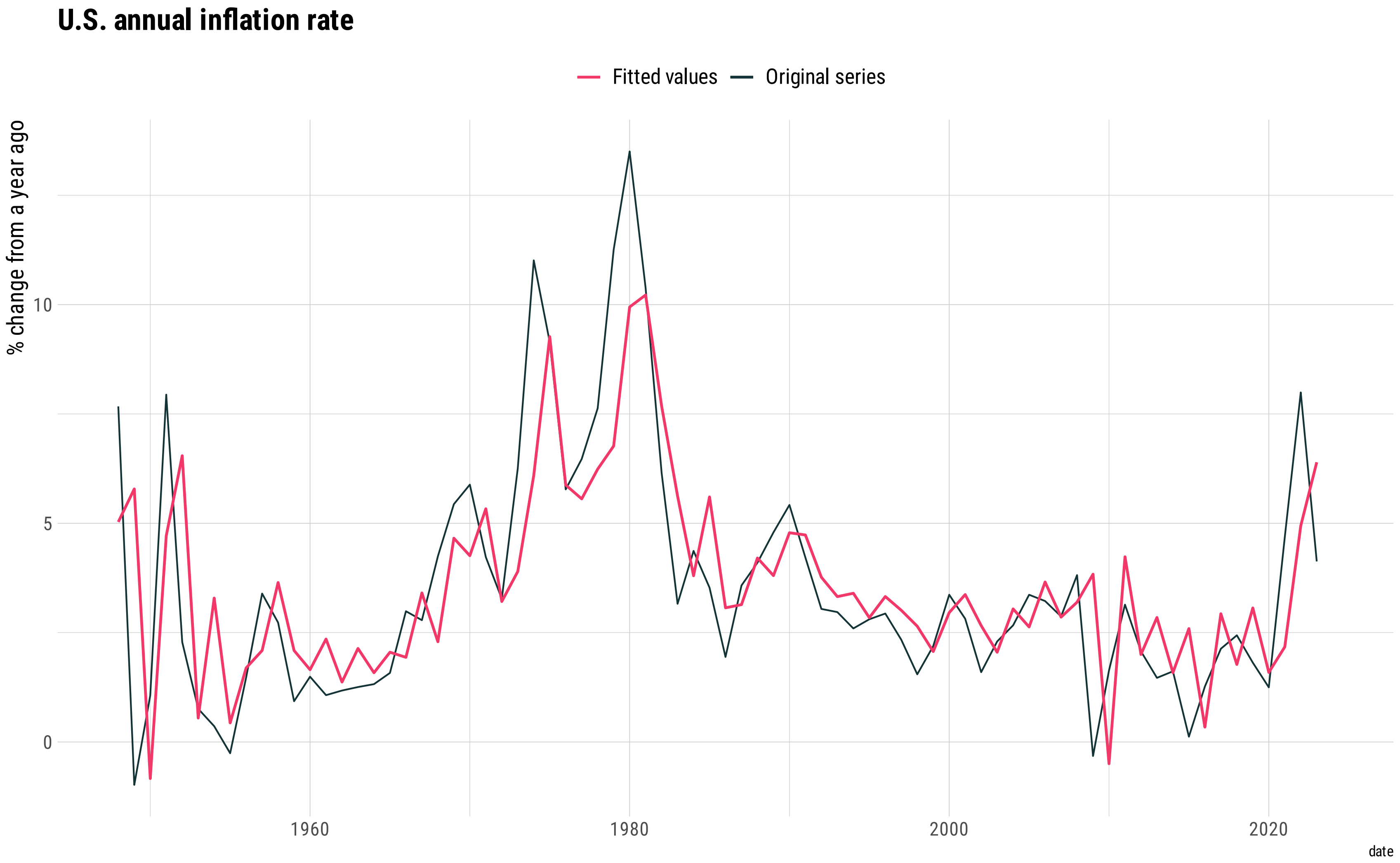


Non-seasonal ARIMA models

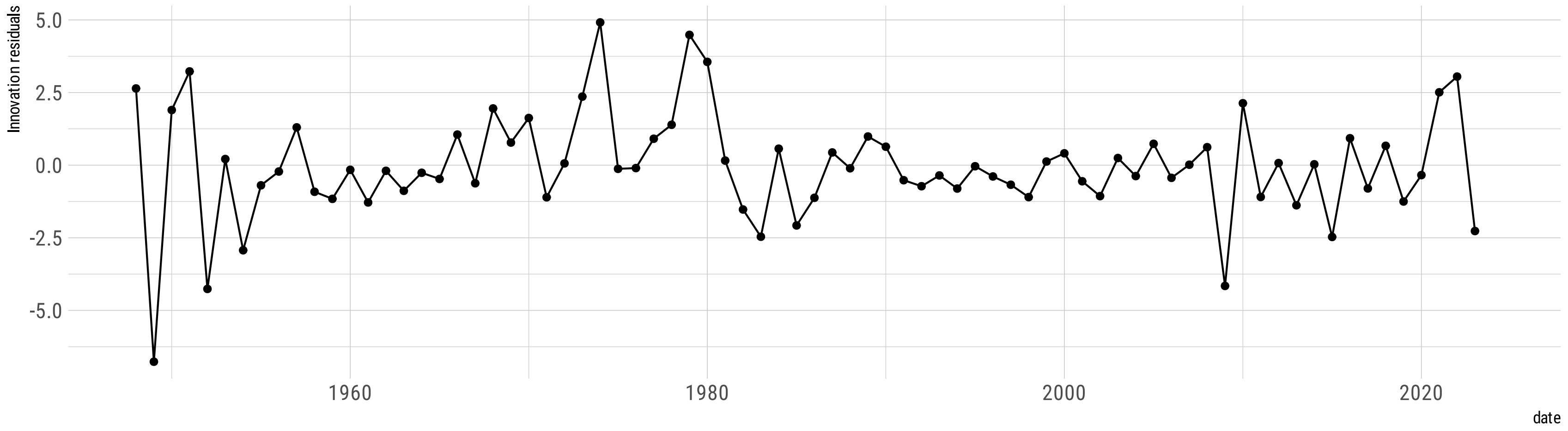
```
infrate_ts ▷
  model(infrate_arima = ARIMA(infrate)) ▷
  report()
```

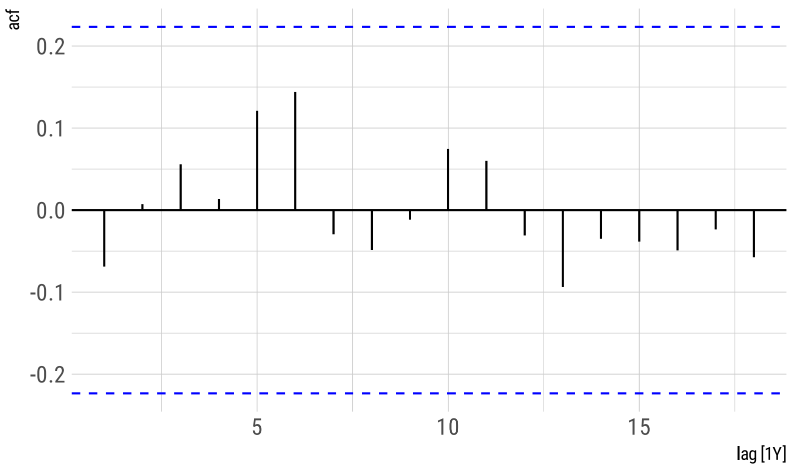
```
#> Series: infrate
#> Model: ARIMA(1,0,2) w/ mean
#>
#> Coefficients:
#>          ar1     ma1      ma2  constant
#>       0.8726  0.0861  -0.5195    0.4627
#> s.e.  0.0870  0.1321   0.1183    0.1116
#>
#> sigma^2 estimated as 3.472:  log likelihood=-154.27
#> AIC=318.54   AICc=319.39   BIC=330.26
```

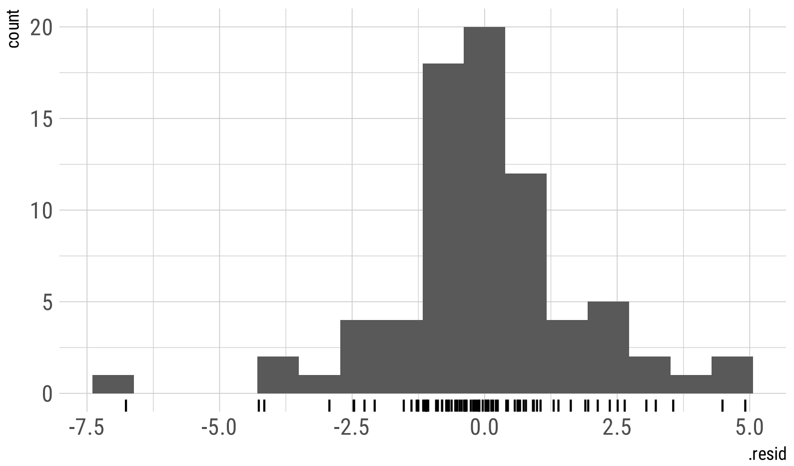
Non-seasonal ARIMA models



Non-seasonal ARIMA models







Non-seasonal ARIMA models

```
infrate_fit ▷
  augment() ▷
  features(.innov, ljung_box, lag = 10, dof = 3)
```

```
#> # A tibble: 1 × 3
#>   .model        lb_stat lb_pvalue
#>   <chr>           <dbl>     <dbl>
#> 1 infrate_arima    4.42     0.730
```

Order selection

Order selection

A **time plot** is not **sufficient** to determine the order of an ARIMA model.

- So how to determine the values of p, q, and d?

Since a key feature of ARIMA models concerns modeling the **autocorrelations** in the data, we can appeal to the **autocorrelation function (ACF)** plot we have studied before.

Recall that an ACF plot shows the **autocorrelations** which measure the relationship between y_t and $y_{t\text{-}k}$ for different values of k.

Order selection

But *think about the following*:

- If y_t and $y_{t\text{-}1}$ are **correlated**, then $y_{t\text{-}1}$ and $y_{t\text{-}2}$ must also be correlated.
- This implies that y_t and $y_{t\text{-}2}$ might **also** be correlated.
- But is this latter correlation due to their **connection** to $y_{t\text{-}1}$ or because of any **new information** contained in $y_{t\text{-}2}$ that could be used in forecasting y_t?

Order selection

With these issues in mind, we can also use the **Partial Autocorrelation Coefficient Function (PACF)**.

> The **partial autocorrelation** at lag k is the correlation that results after **removing the effect** of *any* correlations due to the terms at shorter lags.

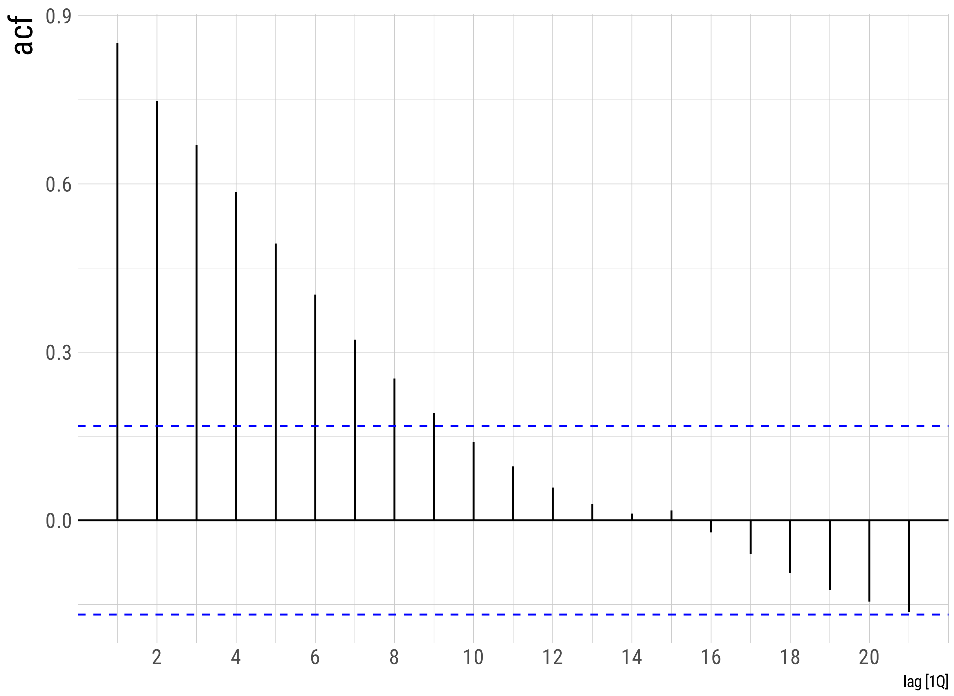
In other words, the partial autocorrelation coefficient measures the relationship between y_t and $y_{t\text{-}k}$ **after removing** the effects of lags 1, 2, 3,..., k-1.

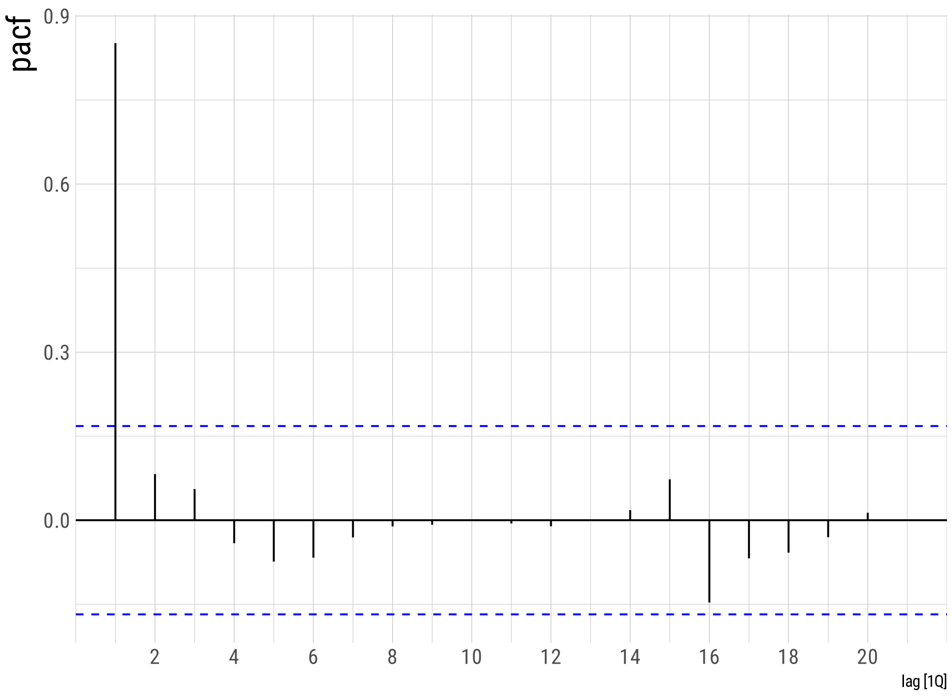
- This way, the *first* partial autocorrelation is always **equal** to the *first* autocorrelation coefficient, since there is no observation in between to be removed.

Order selection

```r
unemp_ts |>
  ACF(unrate) |>
  autoplot() +
  easy_y_axis_title_size(18)
```

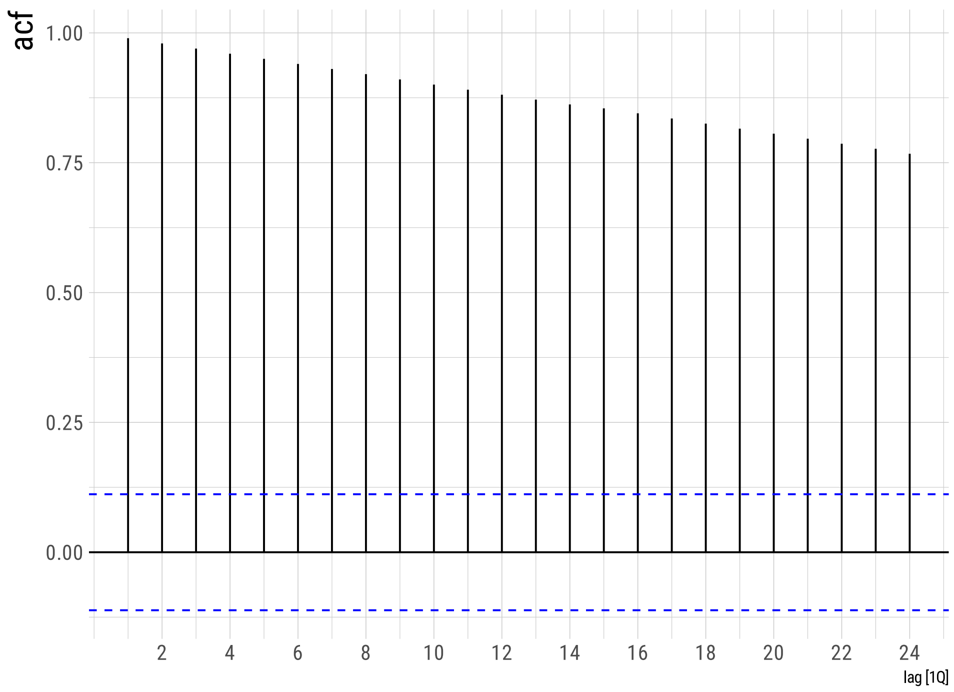
```r
unemp_ts |>
  PACF(unrate) |>
  autoplot() +
  easy_y_axis_title_size(18)
```



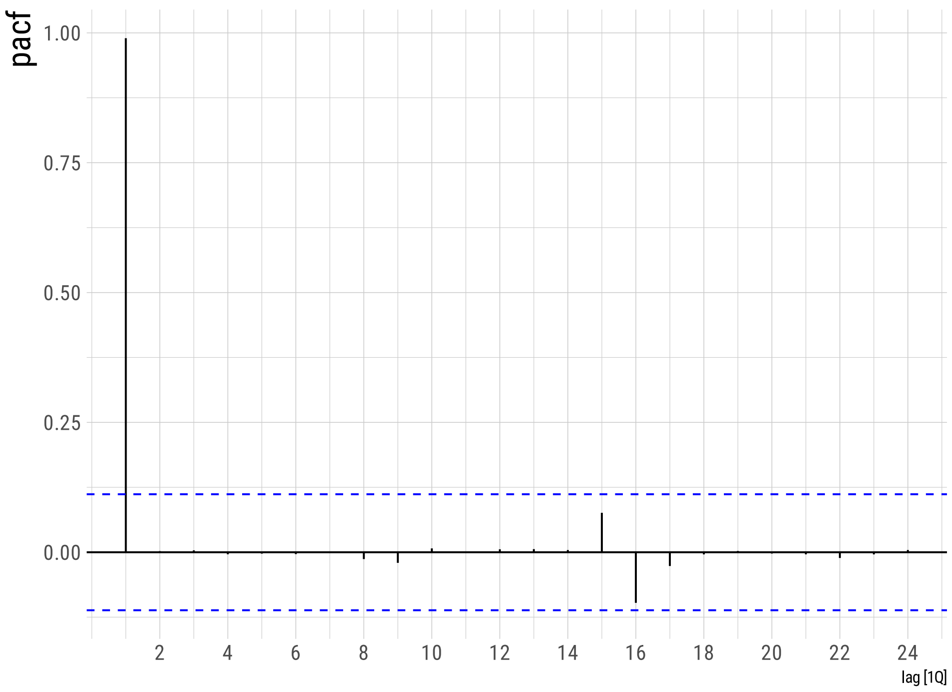


Order selection

```
gdp_ts ▷
  ACF(gdp) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```

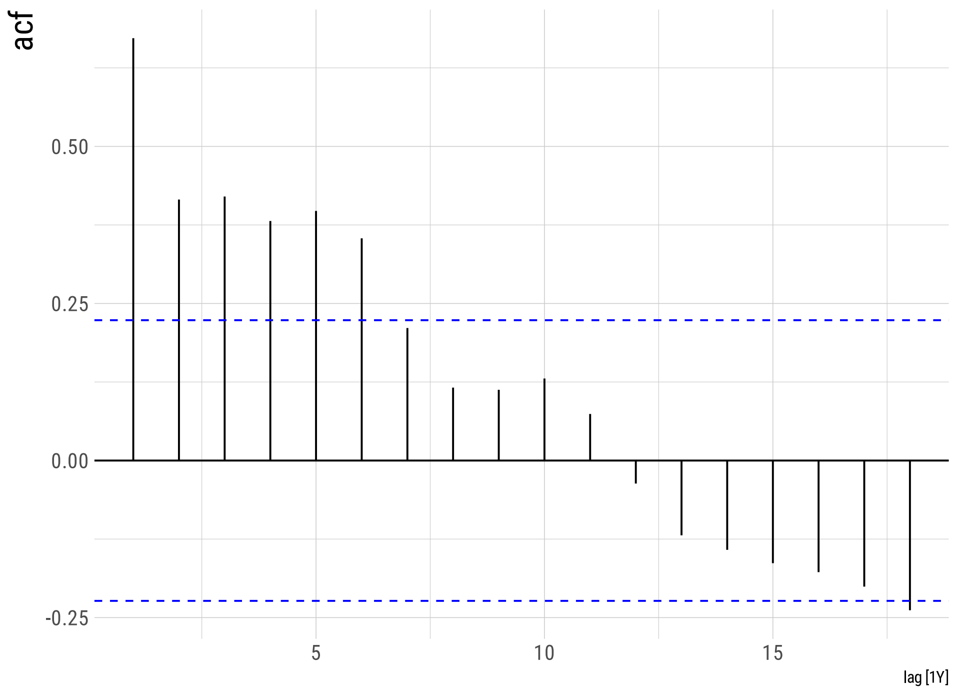



```
gdp_ts ▷
  PACF(gdp) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```

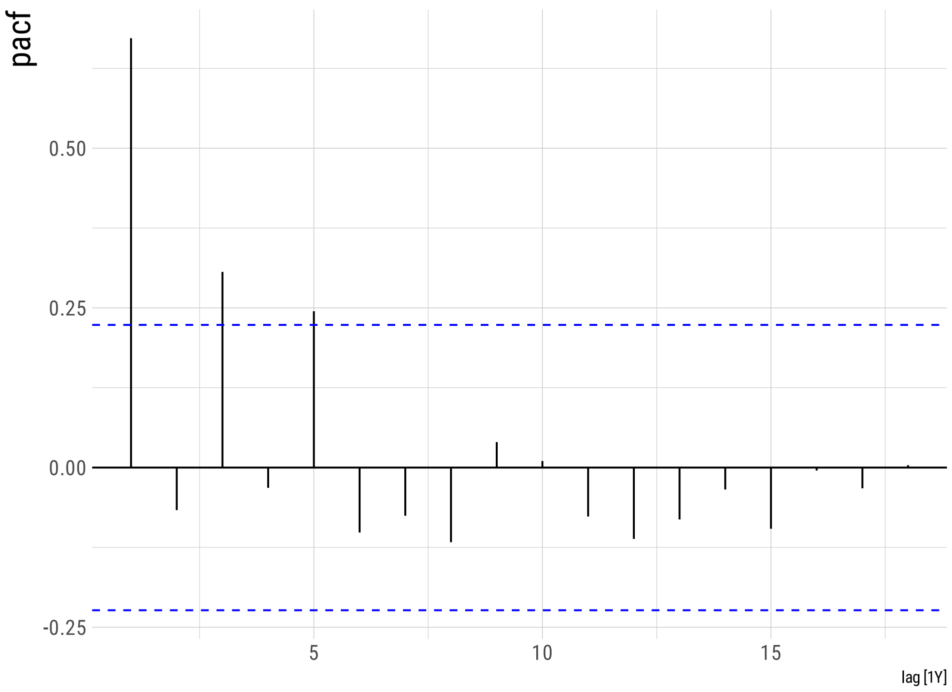


Order selection

```r
infrate_ts ▷
  ACF(infrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```




```r
infrate_ts ▷
  PACF(infrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```



Order selection

In the cases of **ARIMA($0, d, q$)** or **ARIMA($p, d, 0$)** models, then ACF and PACF plots can be very helpful to determine the ordering of the ARIMA model.

- If p and q are positive, then these plots are not that helpful.

In case of an **ARIMA($p, d, 0$)** model:

- the ACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag p in the PACF, but none beyond lag p.

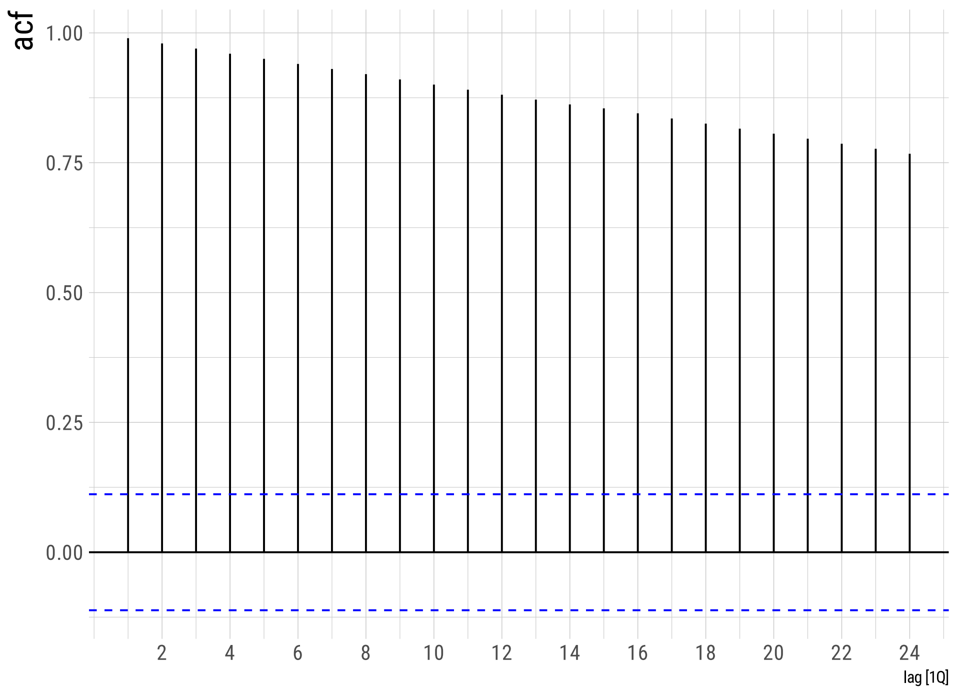
In case of an **ARIMA($0, d, q$)** model:

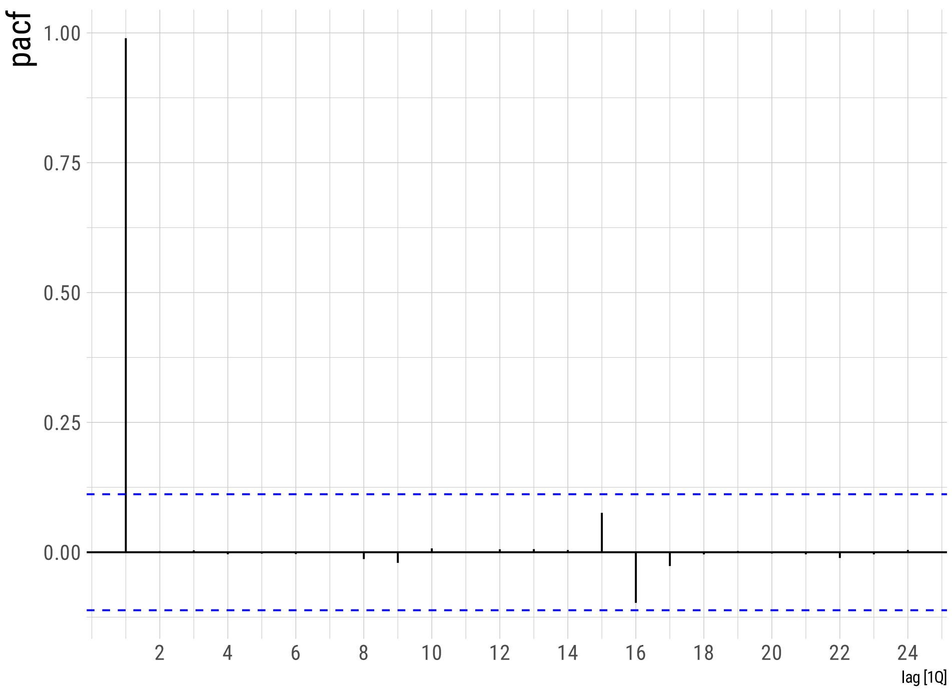
- the PACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag q in the ACF, but none beyond lag q.

Order selection

```r
gdp_ts ▷
  ACF(gdp) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```

```r
gdp_ts ▷
  PACF(gdp) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```





Order selection

```
gdp_ts ▷
  model(arima110 = ARIMA(gdp ~ pdq(1, 1, 0)), # ARIMA(1, 1, 0)
        arima211 = ARIMA(gdp ~ pdq(2, 1, 1)), # ARIMA(2, 1, 1)
        arima_auto = ARIMA(gdp)) # letting the fable package choose.
```

```
#> # A mable: 1 x 3
#>                  arima110                 arima211     arima_auto
#>                   <model>                  <model>        <model>
#> 1 <ARIMA(1,1,0) w/ drift> <ARIMA(2,1,1) w/ drift> <ARIMA(0,2,2)>
```

Order selection

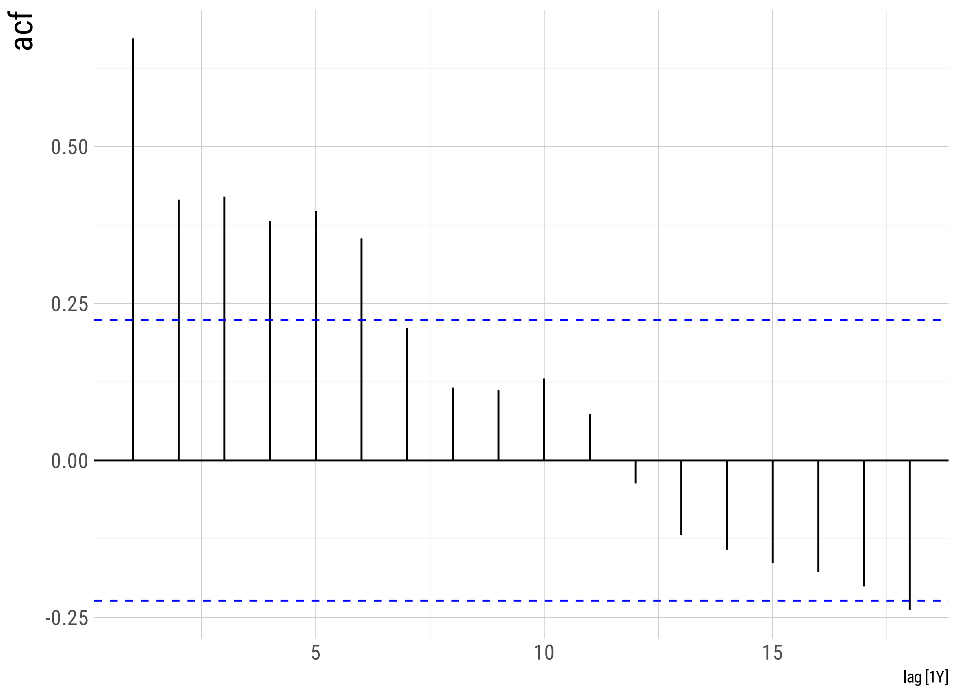
```r
gdp_ts ▷
  model(arima110 = ARIMA(gdp ~ pdq(1, 1, 0)),
        arima211 = ARIMA(gdp ~ pdq(2, 1, 1)),
        arima_auto = ARIMA(gdp)) ▷
  glance() ▷
  arrange(AICc) ▷
  select(.model, AIC, AICc) # Two selection criteria.
```

```
#> # A tibble: 3 × 3
#>   .model       AIC  AICc
#>   <chr>      <dbl> <dbl>
#> 1 arima_auto 3937. 3937.
#> 2 arima110   3951. 3951.
#> 3 arima211   3955. 3955.
```

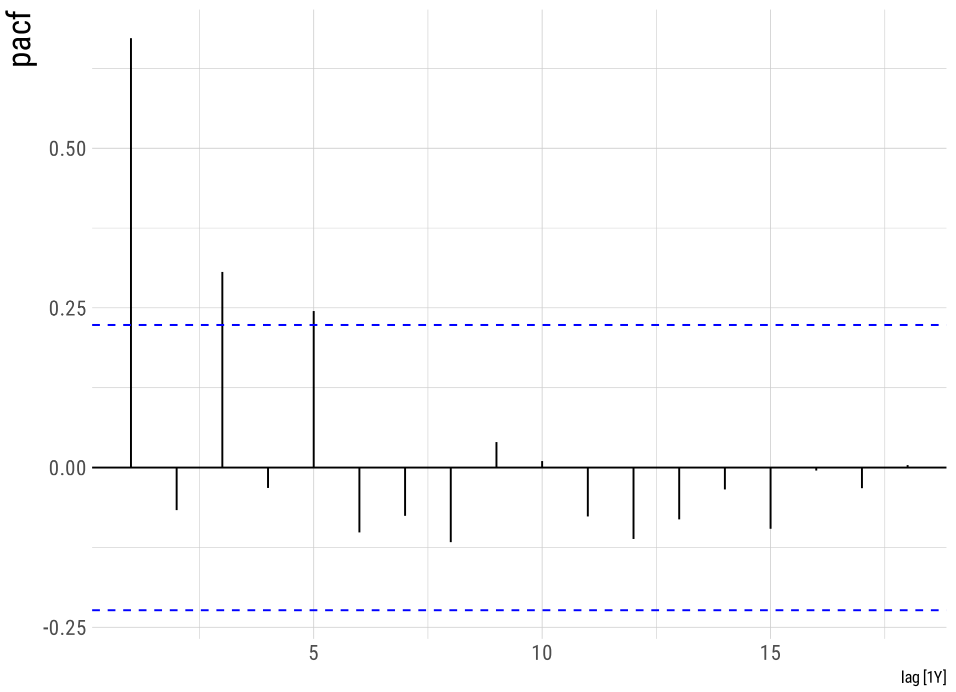
The `{fable}` package estimates ARIMA models using **Maximum Likelihood (ML)** estimation, and its **automatic** choice is for the model that **minimizes** the corrected *Akaike Information Criterion* (*AICc*).

Order selection

```
infrate_ts ▷
  ACF(infrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```




```
infrate_ts ▷
  PACF(infrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```



Order selection

```
infrate_ts ▷
  model(arima500 = ARIMA(infrate ~ pdq(5, 0, 0)),
        arima301 = ARIMA(infrate ~ pdq(3, 0, 1)),
        arima_auto = ARIMA(infrate))
```

```
#> # A mable: 1 x 3
#>                 arima500                 arima301               arima_auto
#>                  <model>                  <model>                  <model>
#> 1 <ARIMA(5,0,0) w/ mean> <ARIMA(3,0,1) w/ mean> <ARIMA(1,0,2) w/ mean>
```

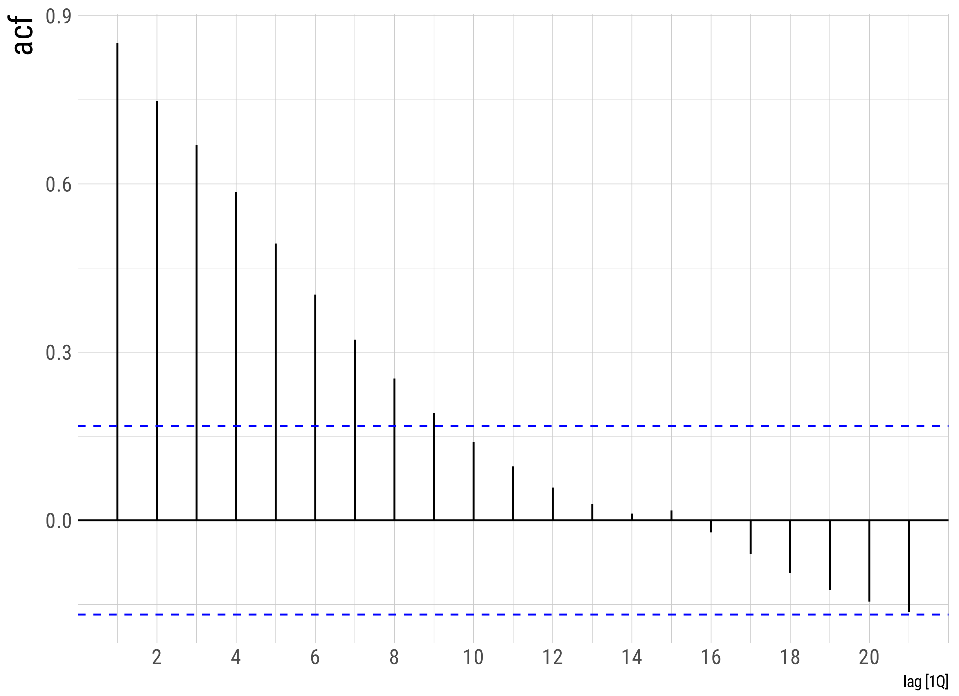
Order selection

```
infrate_ts ▷
  model(arima500 = ARIMA(infrate ~ pdq(5, 0, 0)),
        arima301 = ARIMA(infrate ~ pdq(3, 0, 1)),
        arima_auto = ARIMA(infrate)) ▷
  glance() ▷
  arrange(AICc) ▷
  select(.model, AIC, AICc)
```

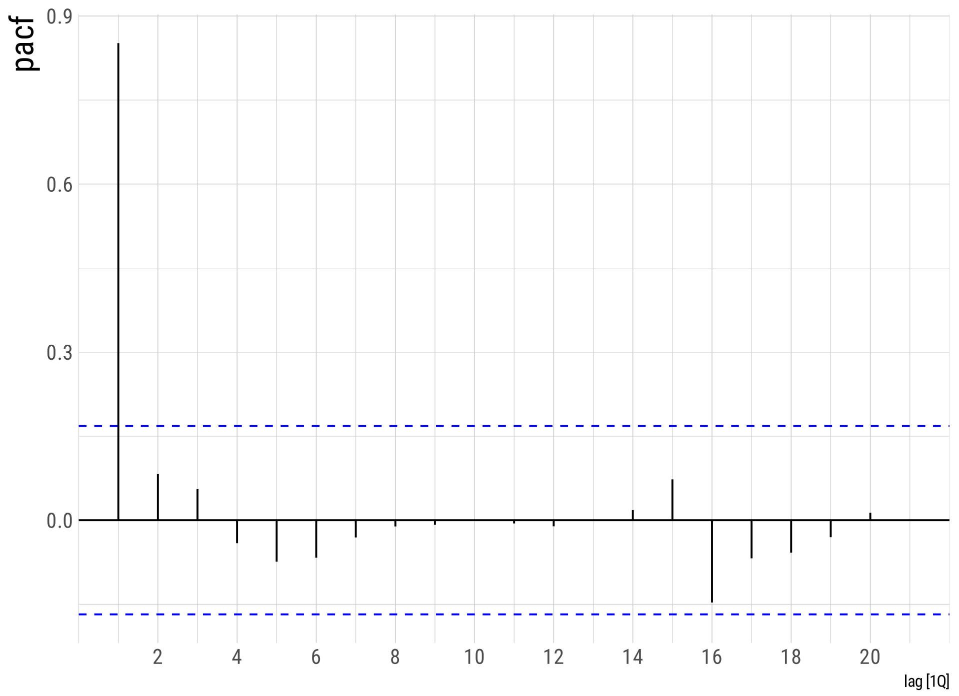
```
#> # A tibble: 3 × 3
#>   .model       AIC  AICc
#>   <chr>      <dbl> <dbl>
#> 1 arima_auto  319.  319.
#> 2 arima301    319.  320.
#> 3 arima500    319.  321.
```

Order selection

```r
unemp_ts ▷
  ACF(unrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```




```r
unemp_ts ▷
  PACF(unrate) ▷
  autoplot() +
  easy_y_axis_title_size(18)
```



Order selection

```r
unemp_ts |>
  model(arima100 = ARIMA(unrate ~ pdq(1, 0, 0)),
        arima101 = ARIMA(unrate ~ pdq(1, 0, 1)),
        arima_auto = ARIMA(unrate))
```

```
#> # A mable: 1 x 3
#>                  arima100                 arima101               arima_auto
#>                   <model>                  <model>                  <model>
#> 1 <ARIMA(1,0,0) w/ mean> <ARIMA(1,0,1) w/ mean> <ARIMA(1,0,0) w/ mean>
```

Order selection

```
unemp_ts ▷
  model(arima100 = ARIMA(unrate ~ pdq(1, 0, 0)),
        arima101 = ARIMA(unrate ~ pdq(1, 0, 1)),
        arima_auto = ARIMA(unrate)) ▷
  glance() ▷
  arrange(AICc) ▷
  select(.model, AIC, AICc)
```

```
#> # A tibble: 3 × 3
#>   .model       AIC  AICc
#>   <chr>      <dbl> <dbl>
#> 1 arima100    364.  364.
#> 2 arima_auto  364.  364.
#> 3 arima101    365.  365.
```

Next time: ARIMA modeling and forecasting