Fitting ARIMA Models

Lecture 12

Dr. Colin Rundel

Assessing Predictive Performance

Test train split

The general approach is to keep the data ordered and split the first `prop`% into the training data and the remainder as testing data.


```
co2 = as_tsibble(co2)
co2_split = rsample::initial_time_split(co2, prop=0.9)
```

```
1 rsample::training(co2_split)
```

```
# A tsibble: 421 x 2 [1M]
      index value
      <mth> <dbl>
 1 1959 Jan  315.
 2 1959 Feb  316.
 3 1959 Mar  316.
 4 1959 Apr  318.
 5 1959 May  318.
 6 1959 Jun  318
 7 1959 Jul  316.
 8 1959 Aug  315.
 9 1959 Sep  314.
10 1959 Oct  313.
# ℹ 411 more rows
```

```
1 rsample::testing(co2_split)
```

```
# A tsibble: 47 x 2 [1M]
      index value
      <mth> <dbl>
 1 1994 Feb  359.
 2 1994 Mar  360.
 3 1994 Apr  361.
 4 1994 May  362.
 5 1994 Jun  361.
 6 1994 Jul  360.
 7 1994 Aug  357.
 8 1994 Sep  356.
 9 1994 Oct  356
10 1994 Nov  358.
# ℹ 37 more rows
```

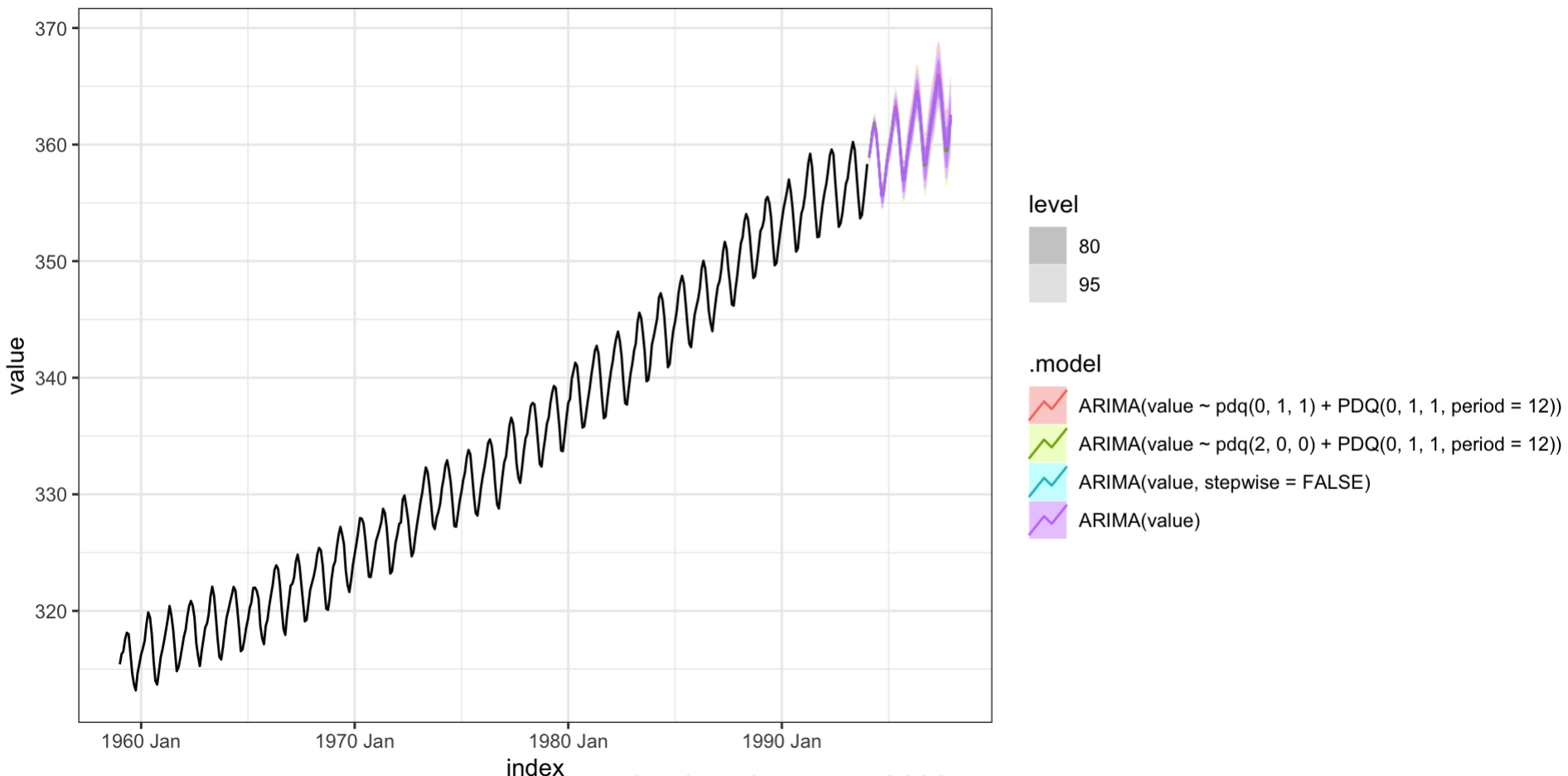
Model fit (training)

```
mm = rsample::training(co2_split) |>
  model(
    ARIMA(value~pdq(2,0,0) + PDQ(0,1,1, period=12)),
    ARIMA(value~pdq(0,1,1) + PDQ(0,1,1, period=12)),
    ARIMA(value),
    ARIMA(value, stepwise = FALSE)
  )
glance(mm)
```

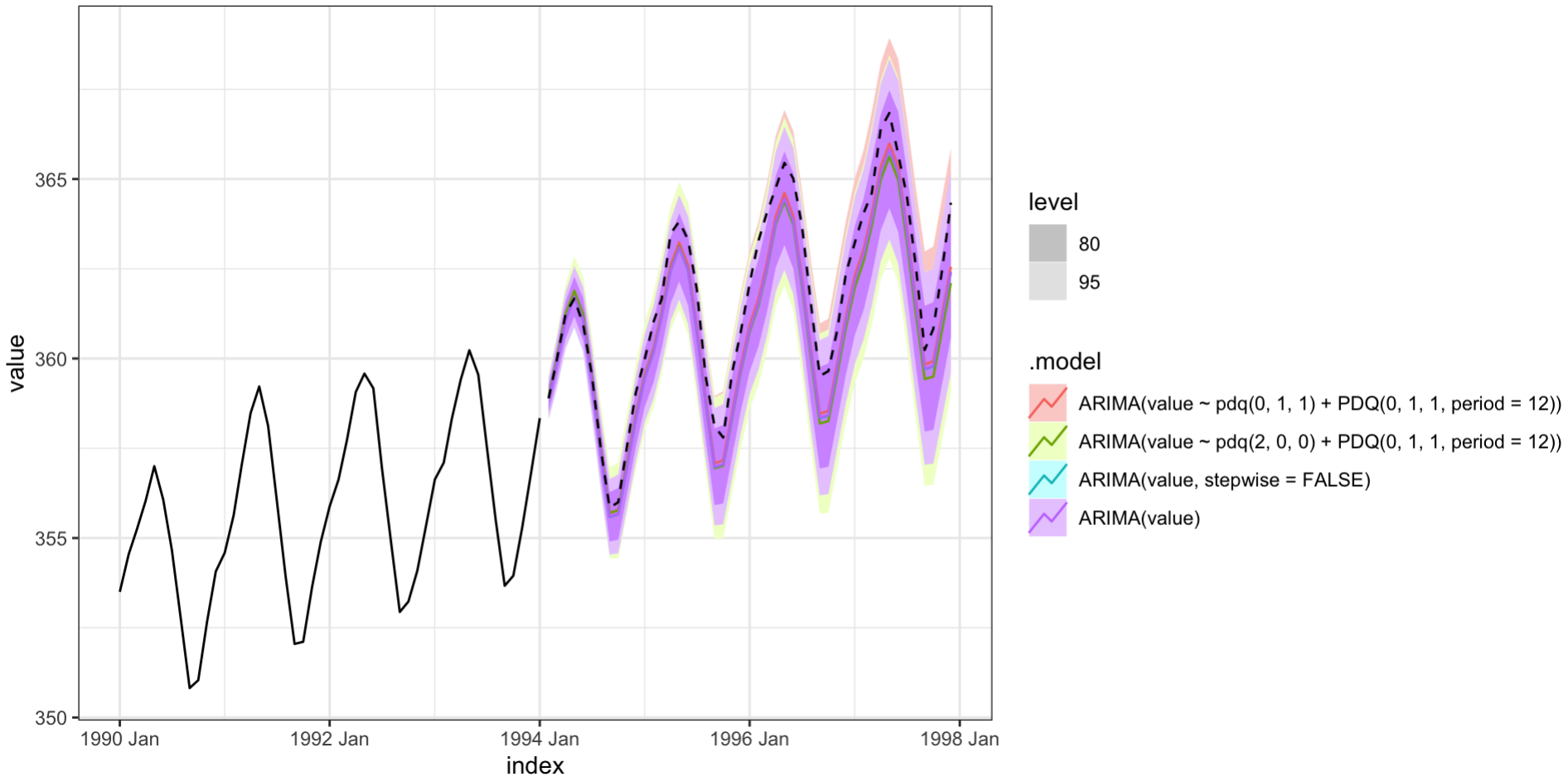
```
# A tibble: 4 × 8
  .model                      sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots
  <chr>                        <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>
1 ARIMA(value ~ pdq(2, 0, 0)… 0.0865   -78.7  167.  167.  187. <cpl>    <cpl>
2 ARIMA(value ~ pdq(0, 1, 1)… 0.0858   -76.8  160.  160.  172. <cpl>    <cpl>
3 ARIMA(value)                0.0852   -73.8  160.  160.  184. <cpl>    <cpl>
4 ARIMA(value, stepwise = FA… 0.0853   -73.7  161.  162.  189. <cpl>    <cpl>
```

Forecasting

```
mm |>
  forecast(new_data = rsample::testing(co2_split)) |>
  autoplot(
    rsample::training(co2_split)
  )
```



Forecasting (1990-1996)



Accuracy

Out-of-sample:

```
mm |>
  forecast(new_data = rsample::testing(co2_sp
  accuracy(rsample::testing(co2_split))
```

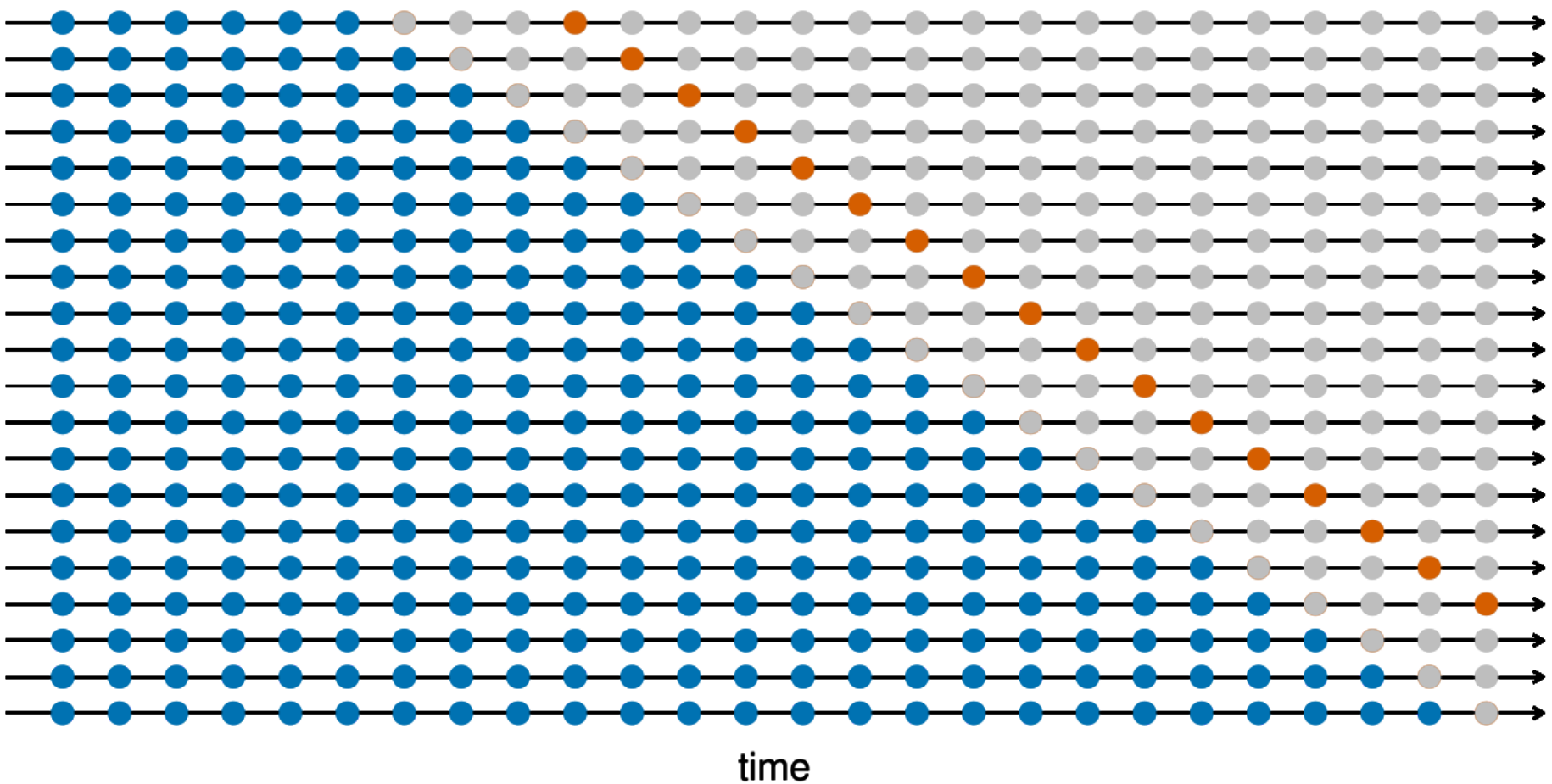
```
# A tibble: 4 × 10
  .model                    .type     ME  RMSE
MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>                     <chr>  <dbl> <dbl>
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ARIMA(value ~ pdq(0, 1,… Test   0.742 0.881
0.764 0.205 0.211   NaN   NaN 0.684
2 ARIMA(value ~ pdq(2, 0,… Test   0.929 1.10
0.956 0.256 0.264   NaN   NaN 0.761
3 ARIMA(value)             Test   0.862 0.987
0.871 0.238 0.240   NaN   NaN 0.686
4 ARIMA(value, stepwise =… Test   0.875 1.00
0.883 0.241 0.244   NaN   NaN 0.689
```

Within-sample:

```
mm |>
  accuracy()
```

```
# A tibble: 4 × 10
  .model                .type     ME  RMSE   MAE
MPE   MAPE  MASE RMSSE   ACF1
  <chr>                 <chr>  <dbl> <dbl> <dbl>
<dbl>  <dbl> <dbl> <dbl>  <dbl>
1 ARIMA(value ~ pdq(… Trai… 0.0118 0.288 0.234
0.00340 0.0701 0.192 0.212 0.0122
2 ARIMA(value ~ pdq(… Trai… 0.0118 0.288 0.229
0.00349 0.0685 0.188 0.212 0.0288
3 ARIMA(value)        Trai… 0.0144 0.286 0.227
0.00426 0.0680 0.187 0.210 0.0128
4 ARIMA(value, stepw… Trai… 0.0149 0.285 0.227
0.00440 0.0680 0.187 0.210 0.0113
```

Rolling forecasting origin



::: {.aside} From Hyndman FPP3 - Chp 5.10 :::

One-step forecasts on test data

```
mm |>
  refit(rsample::testing(co2_split)) |>
  accuracy()
```

```
# A tibble: 4 × 10
  .model              .type      ME  RMSE   MAE      MPE   MAPE  MASE RMSSE   ACF1
  <chr>               <chr>   <dbl> <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl>  <dbl>
1 ARIMA(value ~ pdq… Trai…  0.135   0.322 0.294  0.0376 0.0815 0.179 0.186 0.0796
2 ARIMA(value ~ pdq… Trai… -0.0401  0.318 0.220 -0.0111 0.0608 0.134 0.184 0.0383
3 ARIMA(value)       Trai… -0.0479  0.319 0.219 -0.0133 0.0606 0.133 0.185 0.0250
4 ARIMA(value, step… Trai… -0.0487  0.320 0.219 -0.0135 0.0606 0.133 0.185 0.0252
```

Model Fitting

Fitting ARIMA

For an $\mathrm{ARIMA}(p, d, q)$ model,

- Assumes that the data is stationary after differencing
- Handling d is straight forward, just difference the original data d times (leaving $n - d$ observations)

$$y'_t = \Delta^d y_t$$

- After differencing, fit an $\mathrm{ARMA}(p, q)$ model to y'_t.
- To keep things simple we'll assume $w_t \overset{iid}{\sim} N(0, \sigma_w^2)$

MLE - Stationarity & iid normal errors

If both of these assumptions are met, then the time series y_t will also be normal.

In general, the vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_t)'$ will have a multivariate normal distribution with mean $\{\boldsymbol{\mu}\}_i = E(y_i) = E(y_t)$ and covariance $\boldsymbol{\Sigma}$ where $\{\boldsymbol{\Sigma}\}_{ij} = \gamma(i-j)$.

and therefore the joint density of **y** is given by

$$f_{\boldsymbol{y}}(\boldsymbol{y}) = (2\pi)^{-t/2}\ \det(\boldsymbol{\Sigma})^{-1/2}\ \times \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right)$$

AR

Fitting AR(1)

$$y_t = \delta + \phi\, y_{t-1} + w_t$$

We need to estimate three parameters: δ, ϕ, and σ_w^2, we know

$$E(y_t) = \frac{\delta}{1-\phi} \qquad Var(y_t) = \frac{\sigma_w^2}{1-\phi^2}$$

$$\gamma(h) = \frac{\sigma_w^2}{1-\phi^2}\phi^{|h|}$$

Using these properties it is possible to write the distribution of y as a MVN but that does not make it easy to write down a (simplified) closed form for the MLE estimate for δ, θ, and σ_w^2.

Conditional Density

We can also rewrite the density as follows,

$$\begin{aligned} f(\boldsymbol{y}) &= f(y_t, y_{t-1}, \ldots, y_2, y_1) \\ &= f(y_t|y_{t-1}, \ldots, y_2, y_1) f(y_{t-1}|y_{t-2}, \ldots, y_2, y_1) \cdots f(y_2|y_1) f(y_1) \\ &= f(y_t|y_{t-1}) f(y_{t-1}|y_{t-2}) \cdots f(y_2|y_1) f(y_1) \end{aligned}$$

where,

$$\begin{aligned} y_1 &\sim N\left(\delta, \frac{\sigma_w^2}{1-\phi^2}\right) \\ y_t|y_{t-1} &\sim N\left(\delta + \phi y_{t-1}, \sigma_w^2\right) \\ f(y_t|y_{t-1}) &= \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{1}{2}\frac{(y_t - \delta + \phi y_{t-1})^2}{\sigma_w^2}\right) \end{aligned}$$

Log likelihood of AR(1)

$$\log f(y_t|y_{t-1}) = -\frac{1}{2}\left(\log 2\pi + \log \sigma_w^2 + \frac{1}{\sigma_w^2}(y_t - \delta + \phi y_{t-1})^2\right)$$

$$\ell(\delta, \phi, \sigma_w^2) = \log f(\boldsymbol{y}) = \log f(y_1) + \sum_{i=2}^{t} \log f(y_i|y_{i-1})$$

$$= -\frac{1}{2}\left(\log 2\pi + \log \sigma_w^2 - \log(1-\phi^2) + \frac{(1-\phi^2)}{\sigma_w^2}(y_1 - \delta)^2\right)$$

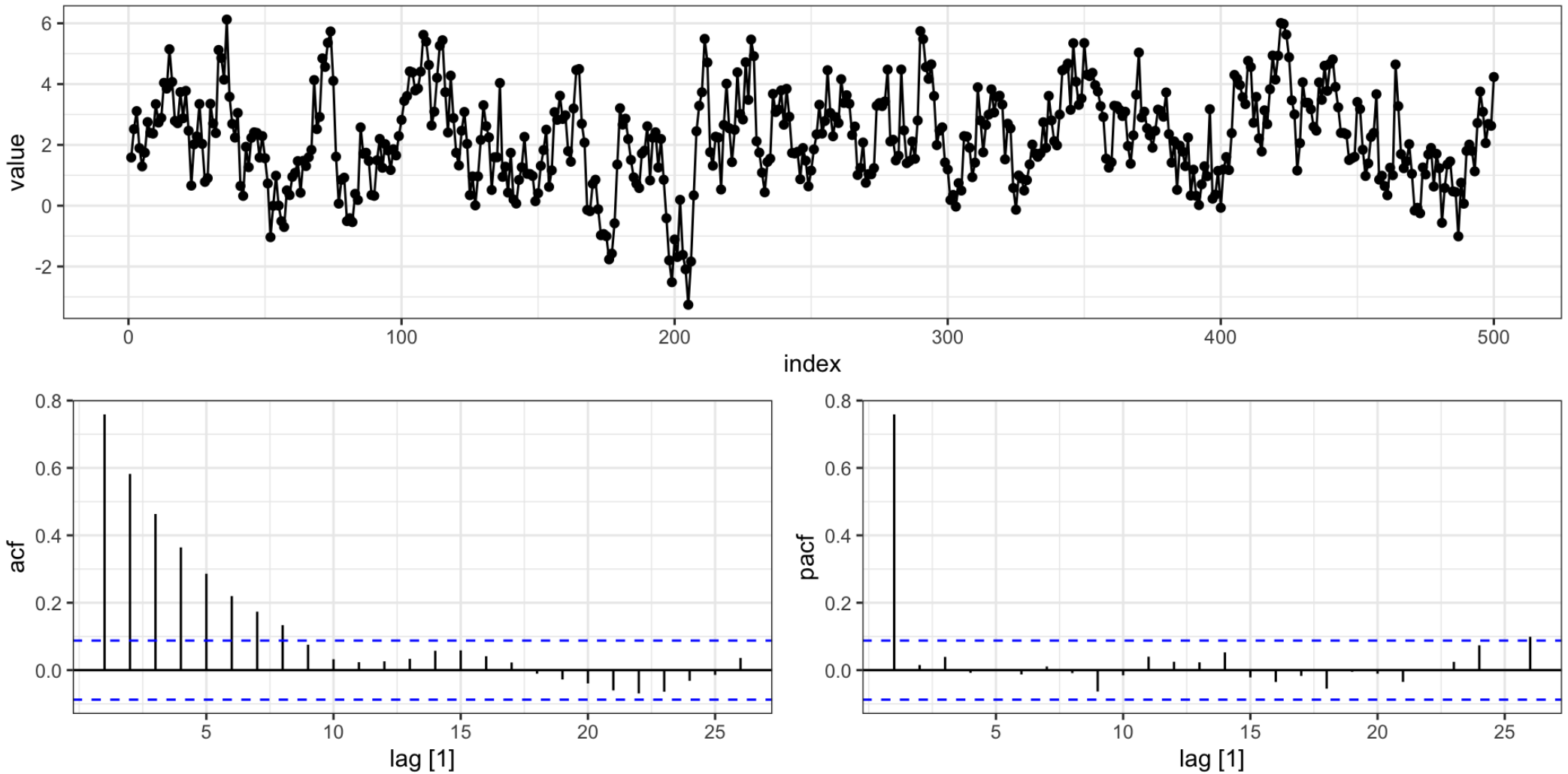
$$-\frac{1}{2}\left((n-1)\log 2\pi + (n-1)\log \sigma_w^2 + \frac{1}{\sigma_w^2}\sum_{i=2}^{n}(y_i - \delta + \phi y_{i-1})^2\right)$$

$$= -\frac{1}{2}\left(n\log 2\pi + n\log \sigma_w^2 - \log(1-\phi^2)\right.$$

$$\left. + \frac{1}{\sigma_w^2}\left((1-\phi^2)(y_1 - \delta)^2 + \sum_{i=2}^{n}(y_i - \delta + \phi y_{i-1})^2\right)\right)$$

AR(1) Example

with $\phi = 0.75$, $\delta = 0.5$, and $\sigma_w^2 = 1$,



ARIMA

```
1 ar1_arima = model(ar1, ARIMA(value~pdq(1,0,0)))
2 report(ar1_arima)
```

```
Series: value
Model: ARIMA(1,0,0) w/ mean

Coefficients:
         ar1  constant
      0.7601    0.5320
s.e.  0.0290    0.0453

sigma^2 estimated as 1.045:  log likelihood=-719.84
AIC=1445.67   AICc=1445.72   BIC=1458.32
```

lm

```r
ar1_lm = lm(value ~ lag(value), data=ar1)
summary(ar1_lm)
```

```
Call:
lm(formula = value ~ lag(value), data = ar1)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.7194 -0.6991 -0.0139  0.6323  3.3518 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.53138    0.07898   6.728 4.74e-11 ***
lag(value)   0.76141    0.02918  26.090  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.023 on 497 degrees of freedom
```

Bayesian AR(1) Model

```
1 library(brms) # must be loaded for arma to work
2 ( ar1_brms = brm(value ~ arma(p = 1, q = 0), data=ar1, refresh=0) )
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: value ~ arma(p = 1, q = 0)
   Data: ar1 (Number of observations: 500)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Correlation Structures:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
ar[1]     0.76      0.03     0.70     0.82 1.00     4257     3077

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     2.20      0.20     1.81     2.59 1.00     3474     2626

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     1.02      0.03     0.96     1.09 1.00     3445     2855

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
```

::: {.small}

```
1 brms::variables(ar1_brms)
```

```
[1] "b_Intercept" "ar[1]"       "sigma"       "lprior"      "lp__"
```

brms Intercept vs δ?

The reported Intercept from the brms model is $E(y_t)$ and not δ - for an ARIMA(1,0,0)

$$E(y_t) = \frac{\delta}{1-\phi} \quad \Rightarrow \quad \delta = E(y_t)*(1-\phi)$$

True $E(y_t)$:

```
0.5 / (1-0.75)
```

```
[1] 2
```

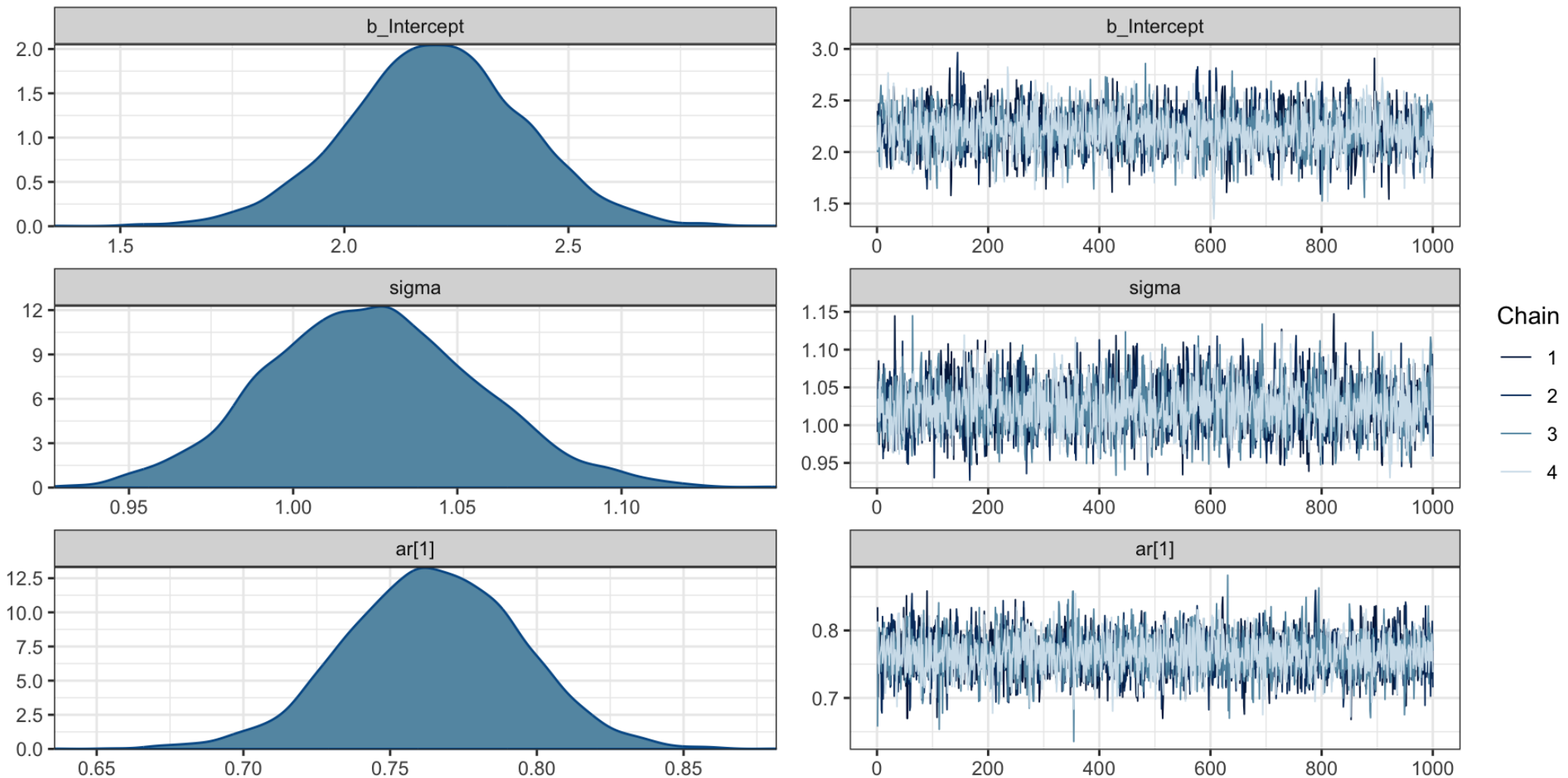
Posterior mean δ:

```
summary(ar1_brms)$fixed$Estimate
  (1 - summary(ar1_brms)$cor_pars
```

```
[1] 0.5186563
```

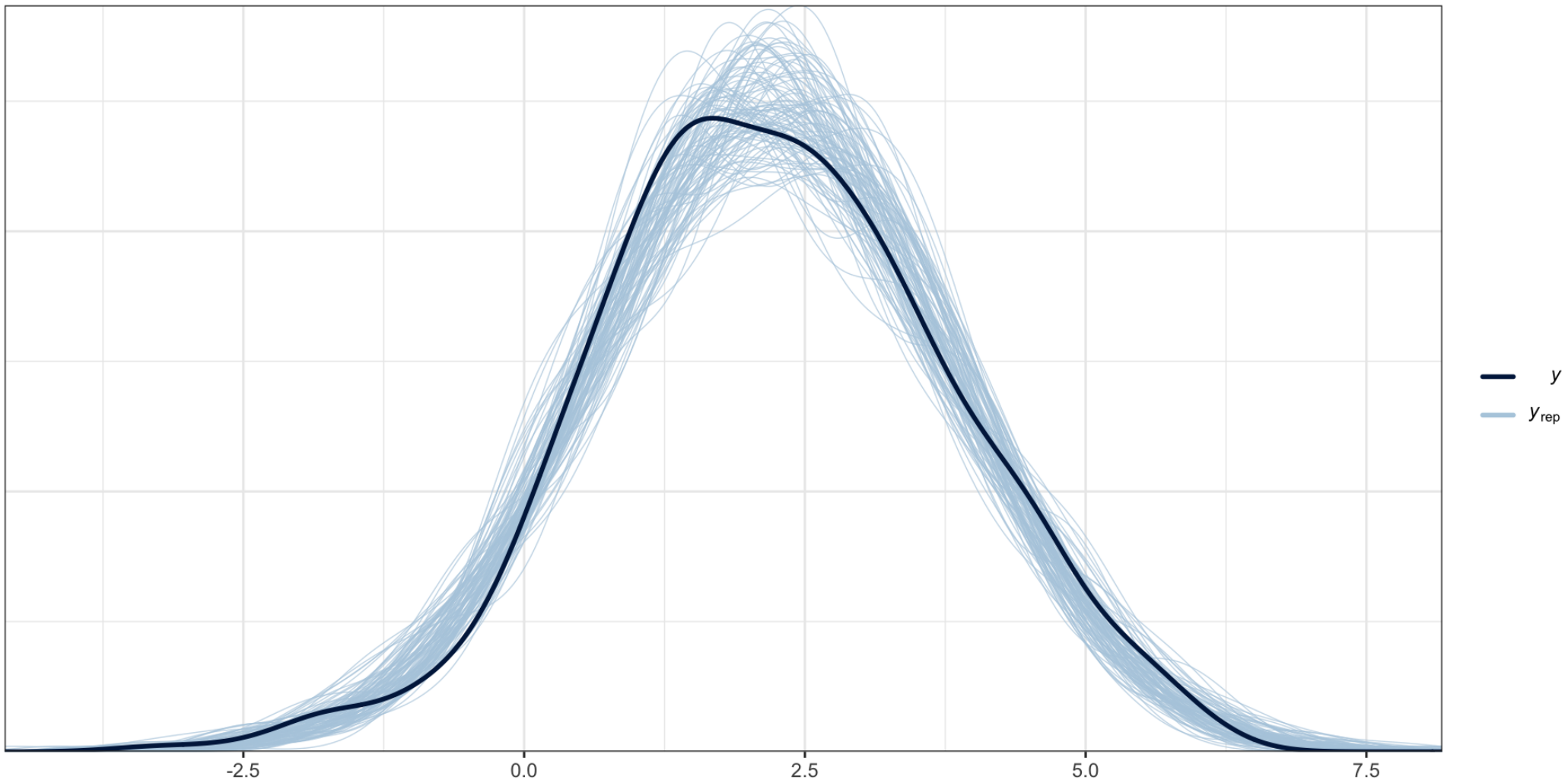
Chains

```
1 plot(ar1_brms)
```

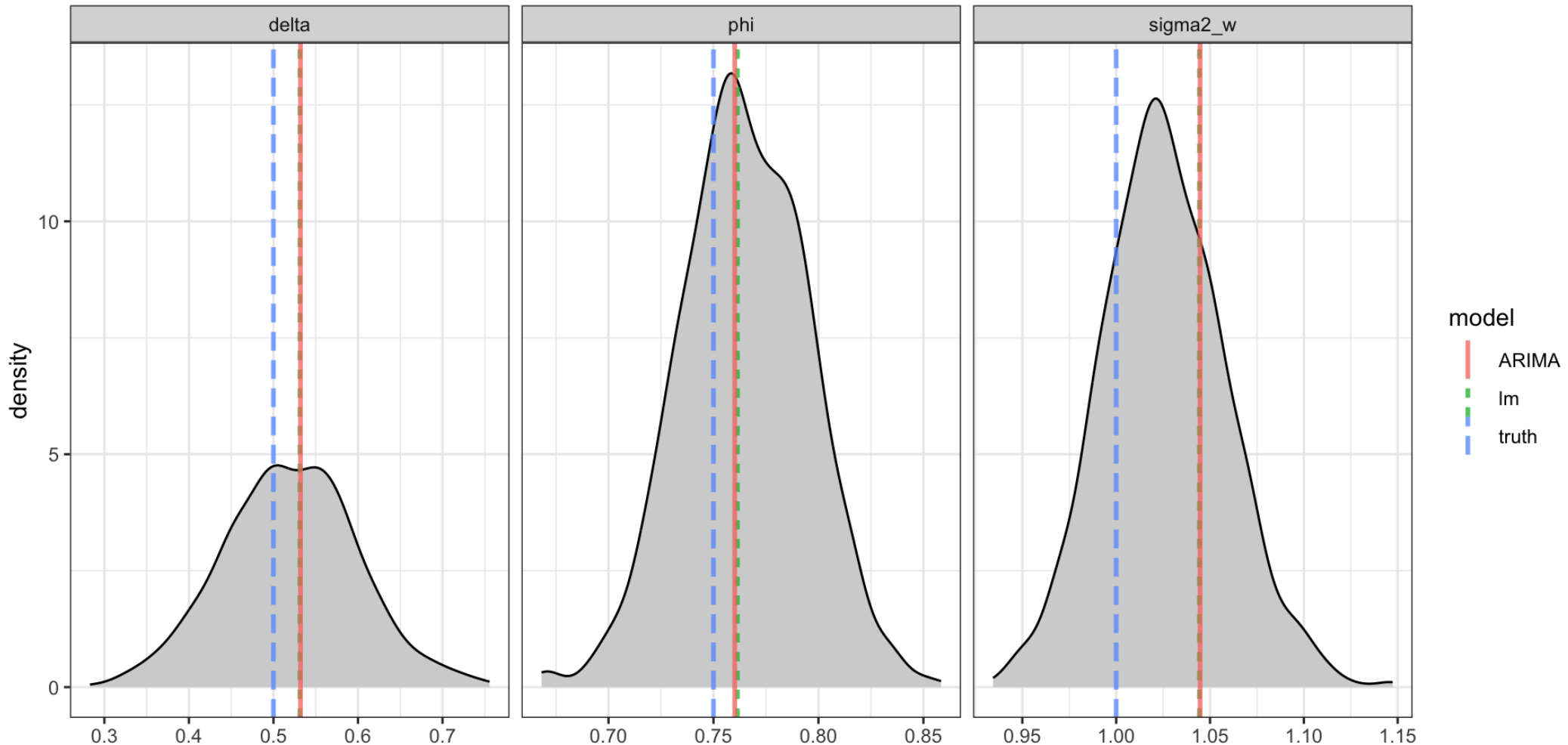


PP Checks

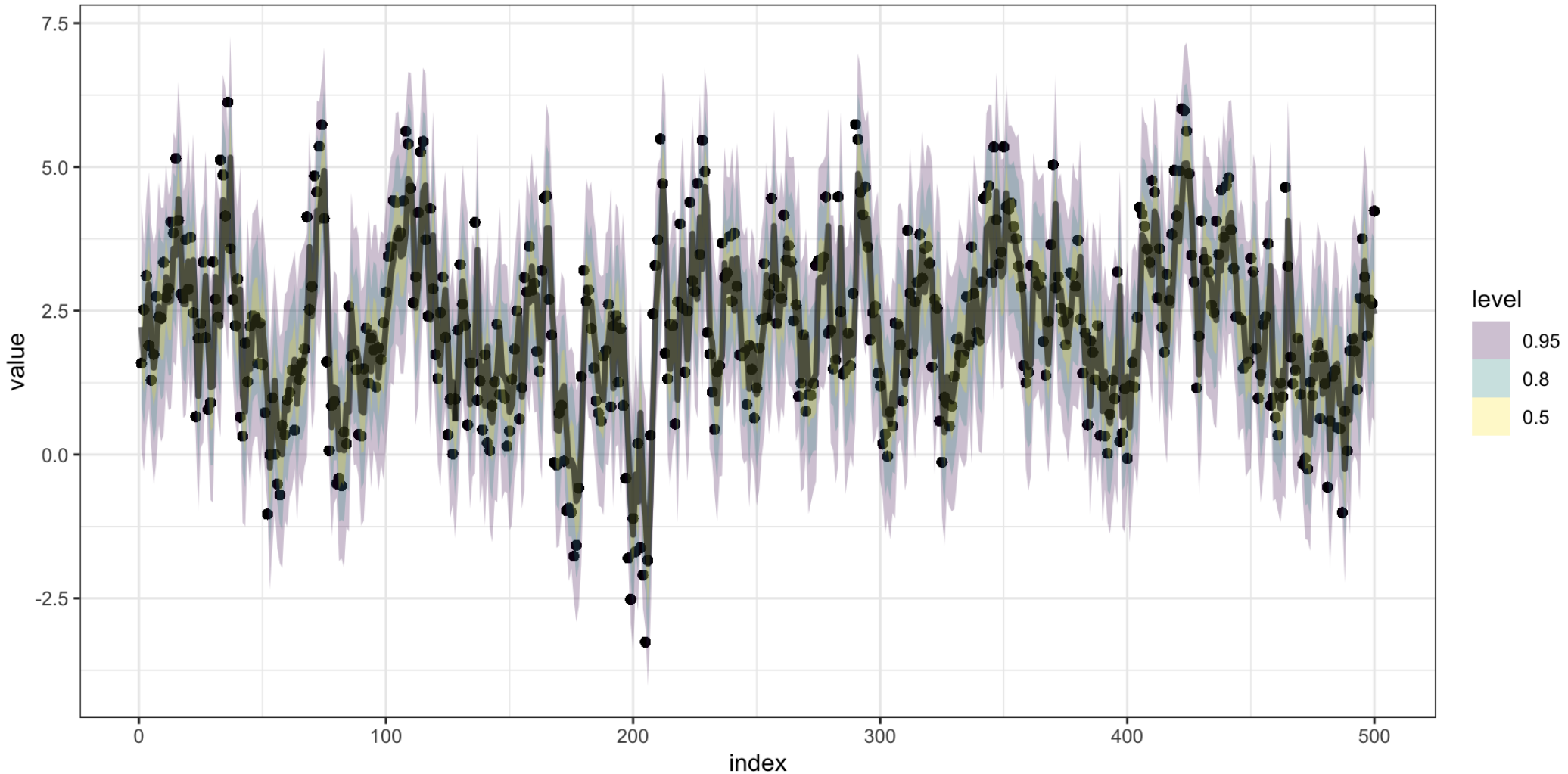
```
1  pp_check(ar1_brms, ndraws=100)
```



Posteriors


delta
phi
sigma2_w
density
10
5
0
0.3
0.4
0.5
0.6
0.7
0.70
0.75
0.80
0.85
0.95
1.00
1.05
1.10
1.15
model
ARIMA
lm
truth

Predictions



Forecasting

```
ar1_brms_fc = ar1_brms |>
  predicted_draws_fix(
    newdata = tibble(index=501:550, value=NA)
  ) |>
  filter(.chain == 1)
```

Fitting AR(p)

Lagged Regression

As with the AR(1), we can rewrite the density using conditioning,

$$\begin{aligned} f(\boldsymbol{y}) &= f(y_t, y_{t-1}, \ldots, y_2, y_1) \\ &= f(y_n | y_{n-1}, \ldots, y_{n-p}) \cdots f(y_{p+1} | y_p, \ldots, y_1) f(y_p, \ldots, y_1) \end{aligned}$$

Regressing y_t on $y_{t-1}, \ldots, y_{t-p}$ gets us an approximate solution, but it ignores the $f(y_1, y_2, \ldots, y_p)$ part of the likelihood.

How much does this matter (vs. using the full likelihood)?

- If p is near to n then probably a lot
- If $p << n$ then probably not much

Method of Moments

Recall for an AR(p) process,

$$\gamma(0) = \sigma_w^2 + \phi_1\gamma(1) + \phi_2\gamma(2) + \ldots + \phi_p\gamma(p)$$
$$\gamma(h) = \phi_1\gamma(h-1) + \phi_2\gamma(h-2) + \ldots \phi_p\gamma(h-p)$$

We can rewrite the first equation in terms of σ_w^2,

$$\sigma_w^2 = \gamma(0) - \phi_1\gamma(1) - \phi_2\gamma(2) - \ldots - \phi_p\gamma(p)$$

these are called the Yule-Walker equations.

Yule-Walker

These equations can be rewritten into matrix notation as follows

$$\underset{p\times p}{\boldsymbol{\Gamma}_p}\underset{p\times 1}{\boldsymbol{\phi}} = \underset{p\times 1}{\boldsymbol{\gamma}_p} \qquad \underset{1\times 1}{\sigma_w^2} = \underset{1\times 1}{\gamma(0)} - \underset{1\times p}{\boldsymbol{\phi}'}\underset{p\times 1}{\boldsymbol{\gamma}_p}$$

where

$$\underset{p\times p}{\boldsymbol{\Gamma}_p} = \gamma(j-k)_{j,k}$$

$$\underset{p\times 1}{\boldsymbol{\phi}} = (\phi_1, \phi_2, \ldots, \phi_p)'$$

$$\underset{p\times 1}{\boldsymbol{\gamma}_p} = (\gamma(1), \gamma(2), \ldots, \gamma(p))'$$

If we estimate the covariance structure from the data we obtain $\hat{\boldsymbol{\gamma}}_p$ and $\hat{\boldsymbol{\Gamma}}_p$ which we can plug in and solve for $\boldsymbol{\phi}$ and σ_w^2,

$$\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\Gamma}}_p^{-1}\hat{\boldsymbol{\gamma}}_p \qquad \hat{\sigma}_w^2 = \gamma(0) - \hat{\boldsymbol{\gamma}}_p'\hat{\boldsymbol{\Gamma}}_p^{-1}\hat{\boldsymbol{\gamma}}_p$$

Stan Code

```
1 ar1_brms |> stancode()
```

```
// generated with brms 2.20.1
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  // data needed for ARMA correlations
  int<lower=0> Kar;  // AR order
  int<lower=0> Kma;  // MA order
  // number of lags per observation
  int<lower=0> J_lag[N];
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int max_lag = max(Kar, Kma);
}
parameters {
  real Intercept;  // temporary intercept for centered predictors
  vector[Kar] ar;  // autoregressive coefficients
  real<lower=0> sigma;  // dispersion parameter
}
transformed parameters {
```

ARMA

Fitting $\mathrm{ARMA}(2,2)$

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta_1 w_{t-1} + \theta_2 w_{t-2} + w_t$$

We now need to estimate six parameters: δ, ϕ_1, ϕ_2, θ_1, θ_2 and σ_w^2.

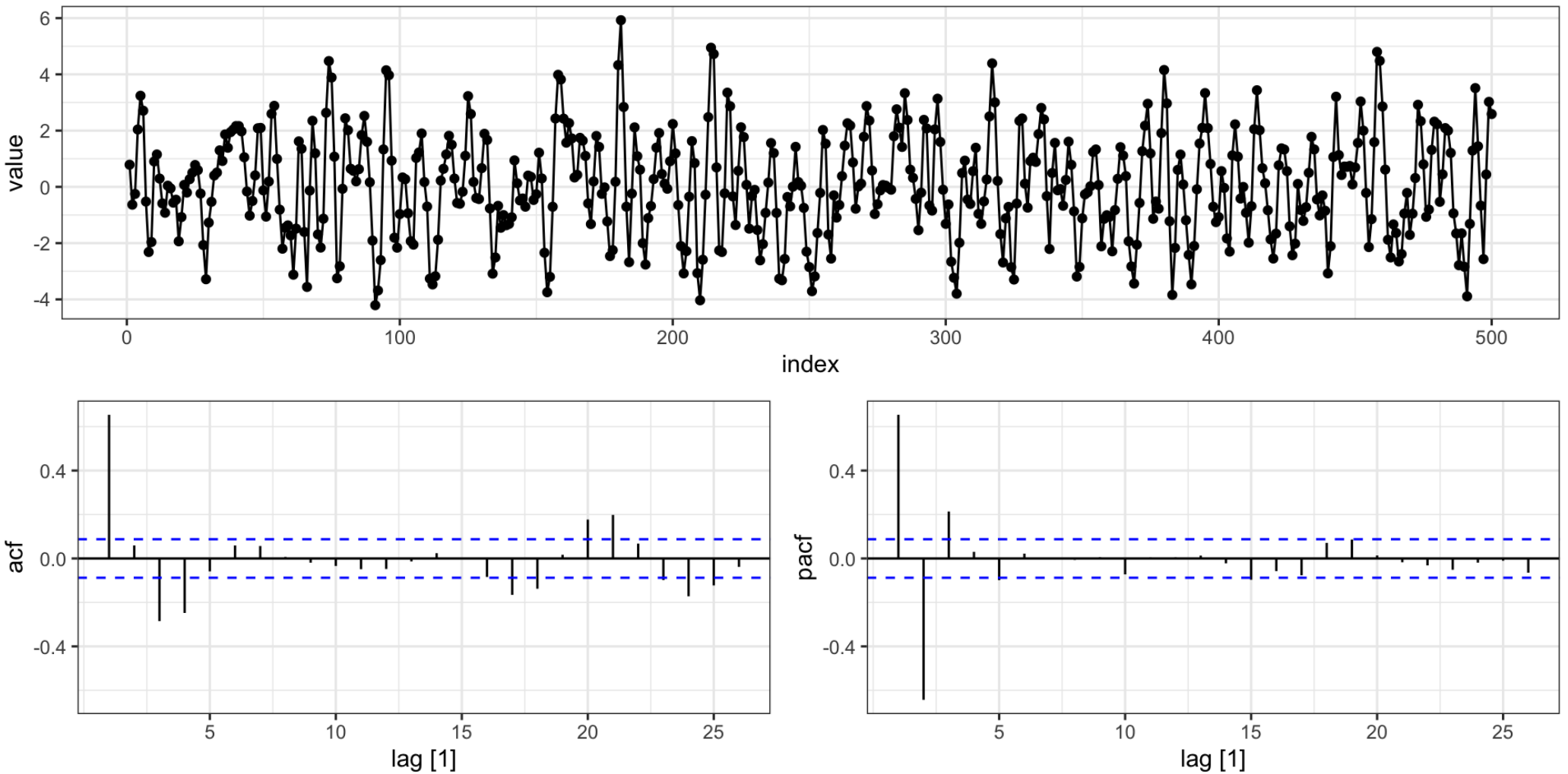
We could figure out $\mathrm{E}(y_t)$, $\mathrm{Var}(y_t)$, and $\mathrm{Cov}(y_t, y_{t+h})$, but the last two are going to be pretty nasty and the full MVN likehood is similarly going to be unpleasant to work with.

Like the AR(1) and AR(p) processes we want to use conditioning to simplify things.

$$y_t | \delta, y_{t-1}, y_{t-2}, w_{t-1}, w_{t-2} \sim \mathrm{N}(\delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta_1 w_{t-1} + \theta_2 w_{t-2}$$

ARMA(2,2) Example

with $\phi = (0.75, -0.5)$, $\theta = (0.5, 0.2)$, $\delta = 0$, and $\sigma_w^2 = 1$ using the same models



ARIMA

```
1 model(d, ARIMA(value ~ 1+pdq(2,0,2))) |> report()
```

```
Series: value
Model: ARIMA(2,0,2) w/ mean

Coefficients:
         ar1      ar2     ma1     ma2  constant
      0.7294  -0.4972  0.4888  0.2538    0.0400
s.e.  0.0868   0.0586  0.0941  0.0727    0.0807

sigma^2 estimated as 1.084:  log likelihood=-728.01
AIC=1468.01   AICc=1468.18   BIC=1493.3
```

AR only lm

```
1 lm(value ~ lag(value,1) + lag(value,2), data=d) |> summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.95562 -0.69955  0.00587  0.77063  3.13283 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.02892    0.04802   0.602    0.547    
lag(value, 1)  1.07883    0.03430  31.455   <2e-16 ***
lag(value, 2) -0.64708    0.03438 -18.820   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.071 on 495 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.6677,    Adjusted R-squared:  0.6664 
F-statistic: 497.4 on 2 and 495 DF,  p-value: < 2.2e-16
```

Hannan-Rissanen Algorithm

1. Estimate a high order AR (remember AR $\Leftrightarrow$ MA when stationary + invertible)
2. Use AR to estimate values for unobserved w_t via `lm` with `lag`s
3. Regress y_t onto $y_{t-1}, \ldots, y_{t-p}, \hat{w}_{t-1}, \ldots \hat{w}_{t-q}$
4. Update $\hat{w}_{t-1}, \ldots \hat{w}_{t-q}$ based on current model,
5. Goto step 3, repeat until convergence

Hannan-Rissanen - Step 1 & 2

```r
(ar.mle(d$value, order.max = 10))
```

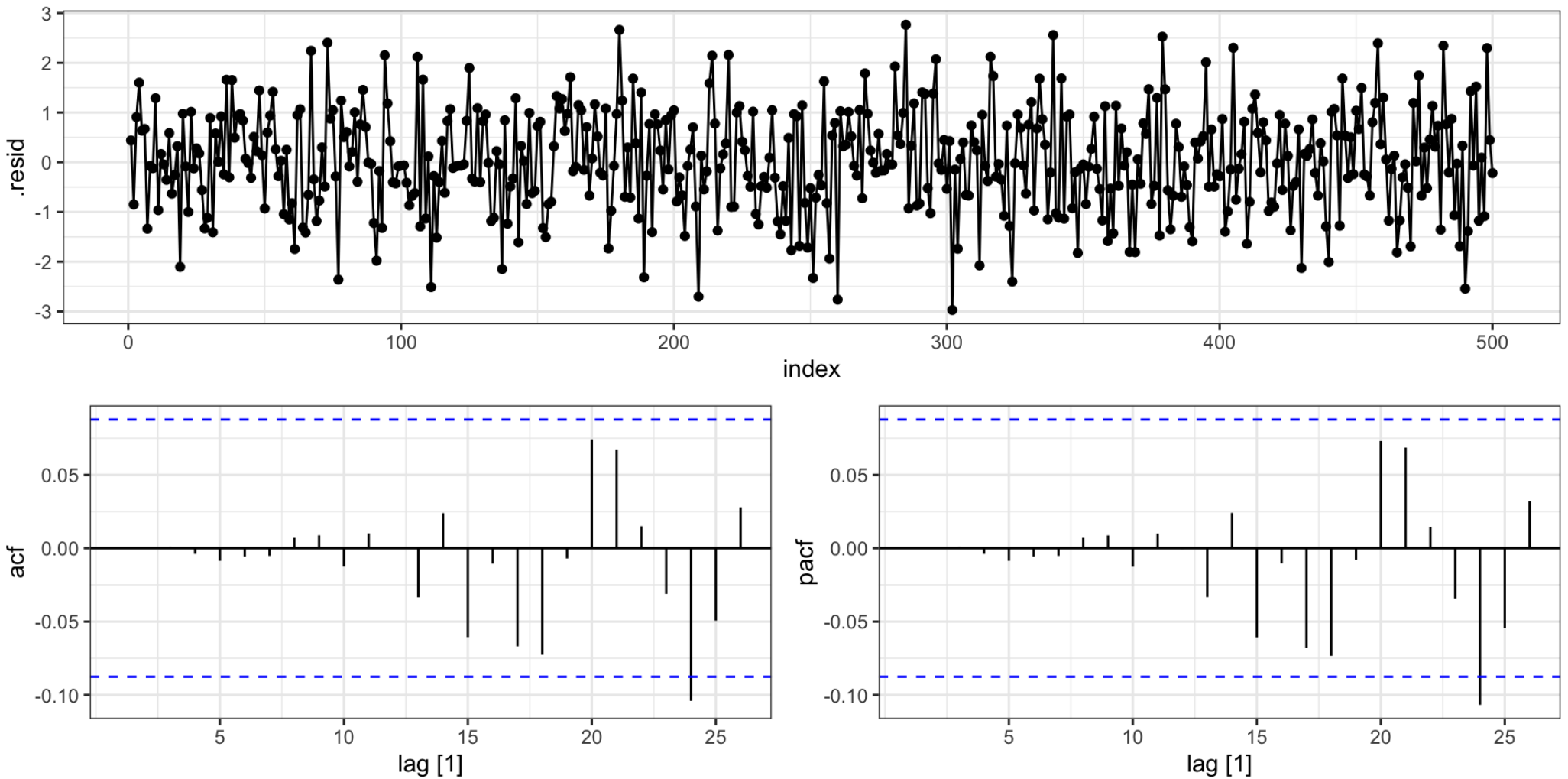
```
Call:
ar.mle(x = d$value, order.max = 10)

Coefficients:
      1         2         3         4         5
 1.2135   -0.8334    0.0921    0.1544   -0.0989

Order selected 5  sigma^2 estimated as  1.072
```

```r
ar = model(d, ARIMA(value ~ 0 + pdq(10,0,0)))
report(ar)
```

```
Series: value
Model: ARIMA(10,0,0)

Coefficients:
         ar1      ar2     ar3     ar4      ar5
      1.2167  -0.8380  0.0929  0.1696  -0.1273
s.e.  0.0446   0.0702  0.0796  0.0800   0.0807
         ar6     ar7      ar8     ar9     ar10
      0.0199  0.0275  -0.0839  0.1000  -0.0732
s.e.  0.0809  0.0809   0.0808  0.0713   0.0453

sigma^2 estimated as 1.089:  log
likelihood=-726.61
AIC=1475.22   AICc=1475.76   BIC=1521.58
```

Residuals

```
1 residuals(ar) |>
2  gg_tsdisplay(y=.resid, plot_type="partial")
```



Hannan-Rissanen - Step 3

```
d = mutate(d, w_hat = residuals(ar)$.resid)

(lm2 = lm(value ~ lag(value,1) + lag(value,2) + lag(w_hat,1) + lag(w_hat,2), data=d)) |>
  summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2) + lag(w_hat,
    1) + lag(w_hat, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-2.95027 -0.67742 -0.06244  0.71858  2.75976

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.02114    0.04679   0.452  0.65165
lag(value, 1)  0.75166    0.07469  10.064  < 2e-16 ***
lag(value, 2) -0.50657    0.04264 -11.881  < 2e-16 ***
lag(w_hat, 1)  0.46618    0.08744   5.332 1.49e-07 ***
lag(w_hat, 2)  0.23440    0.08200   2.859  0.00443 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.043 on 493 degrees of freedom
```

Hannan-Rissanen - Step 4

```r
d = mutate(d, w_hat = augment(lm2, newdata = d)$.resid)

(lm3 = lm(value ~ lag(value,1) + lag(value,2) + lag(w_hat,1) + lag(w_hat,2), data=d)) |>
  summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2) + lag(w_hat,
    1) + lag(w_hat, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-2.94447 -0.66913 -0.05082  0.74799  2.82135

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.03373    0.04685   0.720  0.47181
lag(value, 1) 0.75139    0.07502  10.016  < 2e-16 ***
lag(value, 2) -0.50684   0.04272 -11.866  < 2e-16 ***
lag(w_hat, 1) 0.46361    0.08739   5.305 1.71e-07 ***
lag(w_hat, 2) 0.23576    0.08184   2.881  0.00414 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.042 on 491 degrees of freedom
```

Hannan-Rissanen - Step 3.2 + 4.2

```r
d = mutate(d, w_hat = augment(lm3, newdata = d)$.resid)

(lm4 = lm(value ~ lag(value,1) + lag(value,2) + lag(w_hat,1) + lag(w_hat,2), data=d)) |>
  summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2) + lag(w_hat,
    1) + lag(w_hat, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-2.97496 -0.67003 -0.04571  0.76249  2.78324

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.02735    0.04700   0.582  0.56093
lag(value, 1)  0.75600    0.07540  10.026  < 2e-16 ***
lag(value, 2) -0.50689    0.04287 -11.825  < 2e-16 ***
lag(w_hat, 1)  0.45663    0.08792   5.194 3.03e-07 ***
lag(w_hat, 2)  0.22510    0.08229   2.736  0.00645 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.044 on 489 degrees of freedom
```

Hannan-Rissanen - Step 3.3 + 4.3

```r
d = mutate(d, w_hat = augment(lm4, newdata = d)$.resid)

(lm5 = lm(value ~ lag(value,1) + lag(value,2) + lag(w_hat,1) + lag(w_hat,2), data=d)) |>
  summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2) + lag(w_hat,
    1) + lag(w_hat, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-2.95849 -0.66820 -0.03764  0.74071  2.77925

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.02909    0.04699   0.619  0.53617
lag(value, 1)  0.73934    0.07568   9.770  < 2e-16 ***
lag(value, 2) -0.50027    0.04291 -11.657  < 2e-16 ***
lag(w_hat, 1)  0.47620    0.08792   5.416 9.57e-08 ***
lag(w_hat, 2)  0.24336    0.08249   2.950  0.00333 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.042 on 487 degrees of freedom
```

Hannan-Rissanen - Step 3.4 + 4.4

```r
d = mutate(d, w_hat = augment(lm5, newdata = d)$.resid)

(lm6 = lm(value ~ lag(value,1) + lag(value,2) + lag(w_hat,1) + lag(w_hat,2), data=d)) |>
  summary()
```

```
Call:
lm(formula = value ~ lag(value, 1) + lag(value, 2) + lag(w_hat,
    1) + lag(w_hat, 2), data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-2.96225 -0.68420 -0.04689  0.75628  2.77699

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.02844    0.04721   0.602  0.54718
lag(value, 1)  0.75148    0.07681   9.784  < 2e-16 ***
lag(value, 2) -0.50068    0.04335 -11.550  < 2e-16 ***
lag(w_hat, 1)  0.46357    0.08938   5.186 3.16e-07 ***
lag(w_hat, 2)  0.22960    0.08355   2.748  0.00622 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.044 on 485 degrees of freedom
```

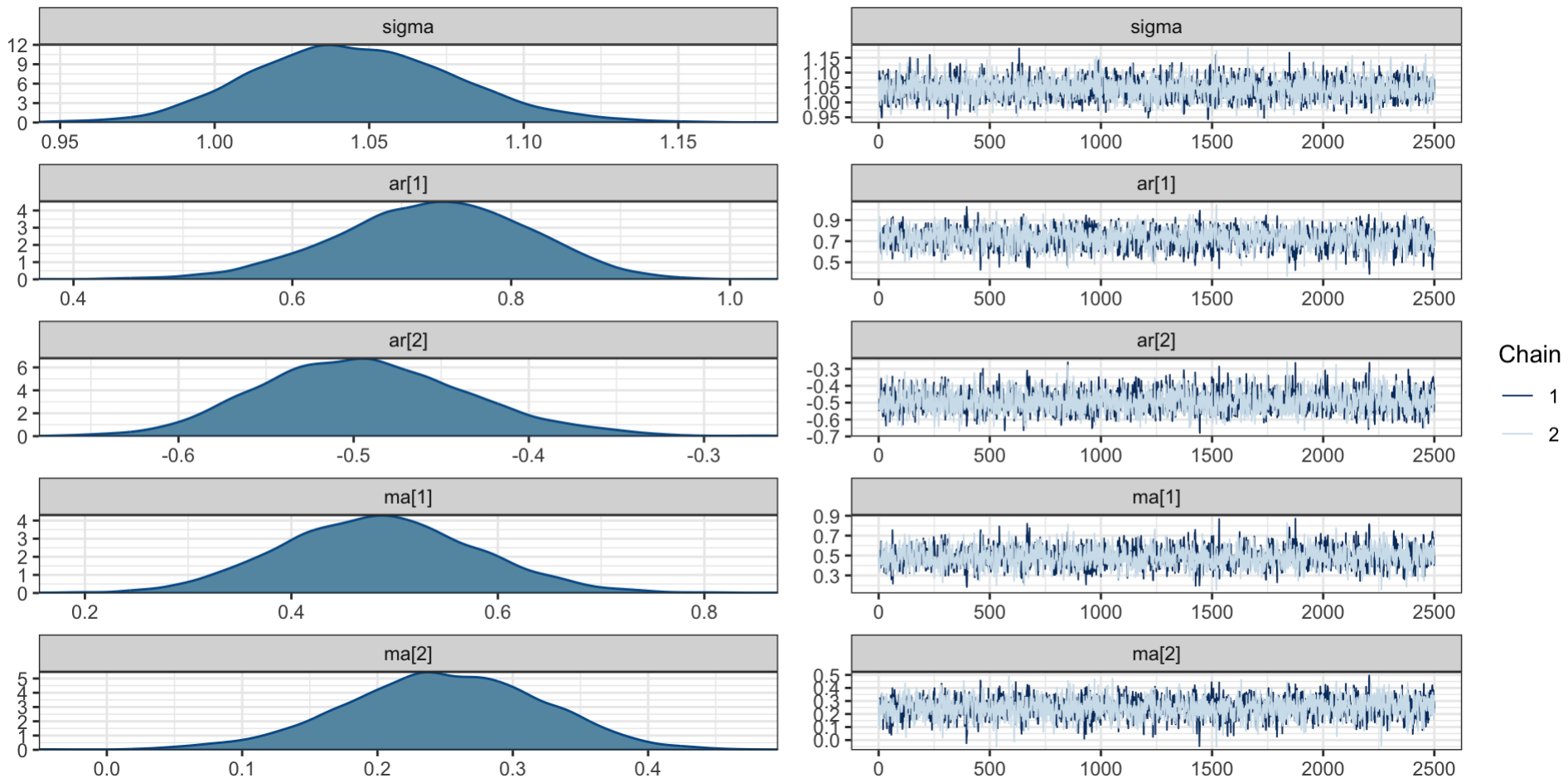
BRMS

```
( arma22_brms = brm(
    value~arma(p=2,q=2)-1, data=d,
    chains=2, refresh=0, iter = 5000, cores = 4
) )
```

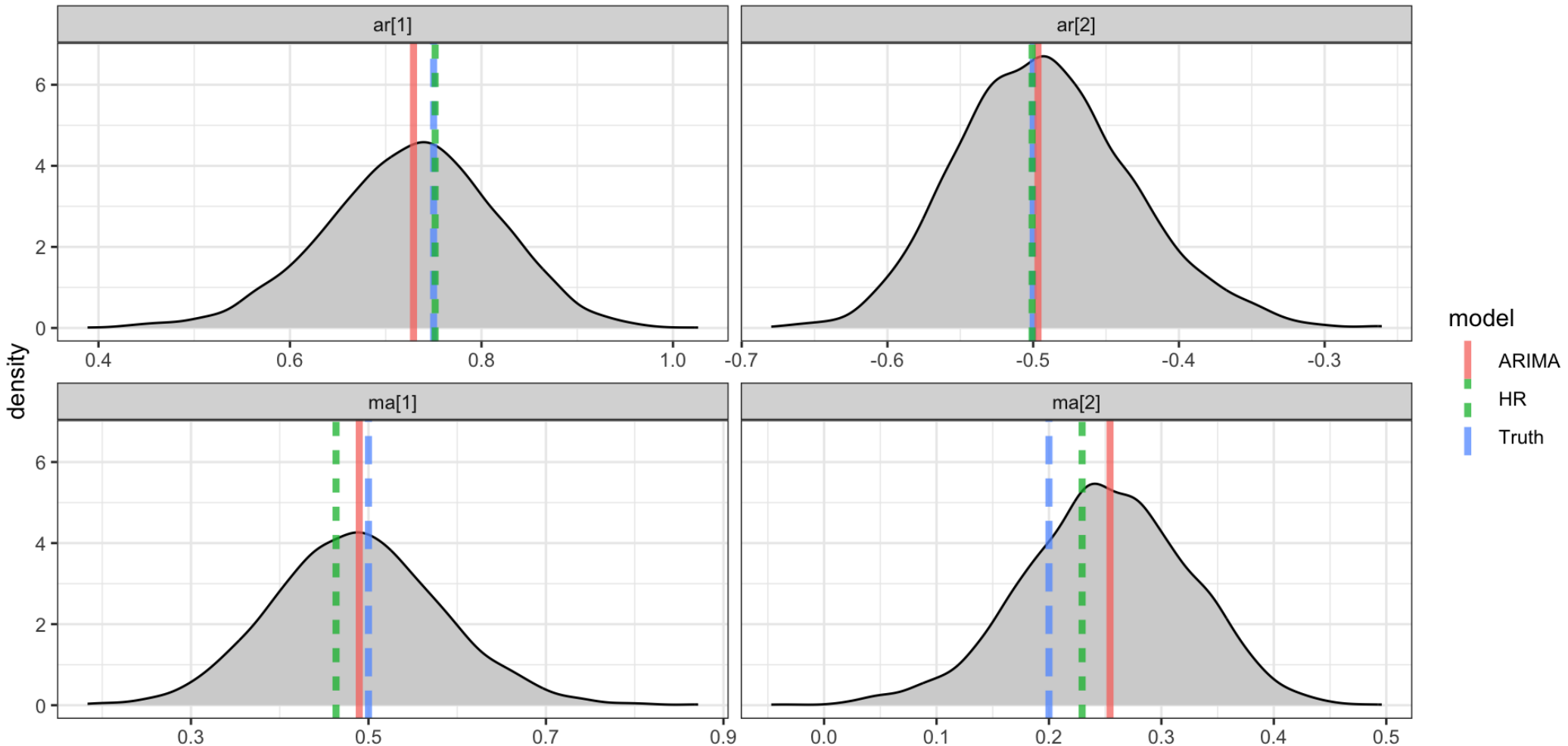
```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: value ~ arma(p = 2, q = 2) - 1
   Data: d (Number of observations: 500)
  Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
         total post-warmup draws = 5000

Correlation Structures:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
ar[1]     0.73      0.09     0.55     0.89 1.00     1864     2605
ar[2]    -0.49      0.06    -0.60    -0.37 1.00     2205     3109
ma[1]     0.49      0.09     0.31     0.67 1.00     1884     2629
ma[2]     0.25      0.07     0.10     0.38 1.00     2150     2858

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     1.05      0.03     0.98     1.11 1.00     2811     2968

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
```

Chains

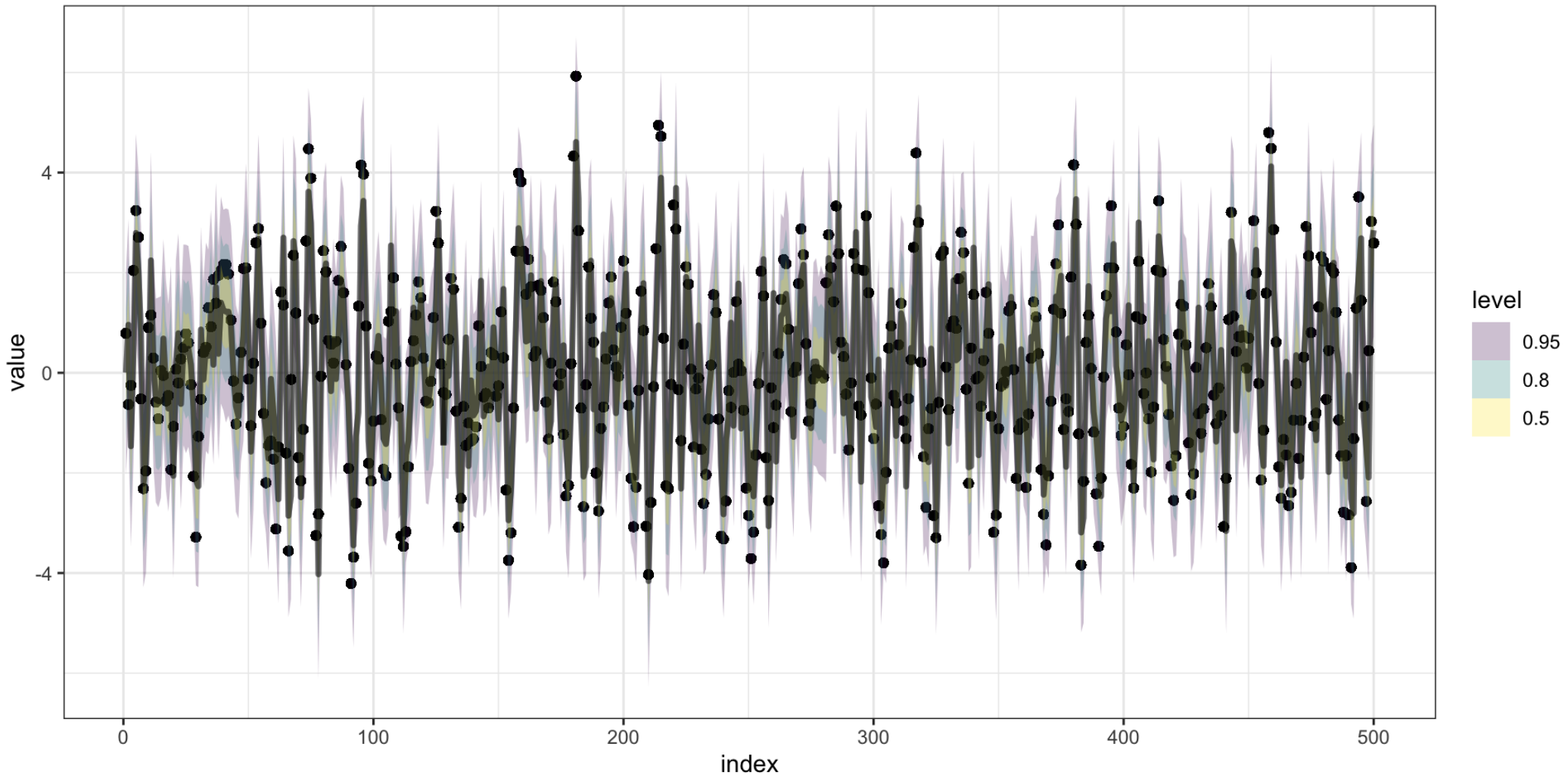
```
1 plot(arma22_brms)
```



Comparison



Predictions



Forecasting

```
arma22_brms_fc = arma22_brms |>
  predicted_draws_fix(
    newdata = tibble(index=501:550)
  ) |>
  filter(.chain == 1)
```

Stan Code

```
1 arma22_brms |> stancode()
```

```
// generated with brms 2.20.1
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  // data needed for ARMA correlations
  int<lower=0> Kar;  // AR order
  int<lower=0> Kma;  // MA order
  // number of lags per observation
  int<lower=0> J_lag[N];
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int max_lag = max(Kar, Kma);
}
parameters {
  vector[Kar] ar;  // autoregressive coefficients
  vector[Kma] ma;  // moving-average coefficients
  real<lower=0> sigma;  // dispersion parameter
}
transformed parameters {
```