Lecture 11

Fitting ARIMA Models

10/10/2018

Model Fitting

Fitting ARIMA

For an $ARIMA(p, d, q)$ model

- Requires that the data be 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 $ARMA(p, q)$ model to y'_t.
- To keep things simple we'll assume $w_t \overset{iid}{\sim} \mathcal{N}(0, \sigma_w^2)$

MLE - Stationarity & iid normal errors

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

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

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

The joint density of $\mathbf{y}$ is given by

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

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

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

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

$$\gamma_h = \frac{\sigma_w^2}{1-\phi^2}\phi^{|h|}$$

$$\underline{\mu} = \begin{pmatrix} \frac{\delta}{1-\phi} \\ \vdots \end{pmatrix} \quad n \times 1$$

$$\underline{\Sigma} = \begin{pmatrix} V & V\phi & V\phi^2 & & \\ V\phi & V & V\phi & & \\ V\phi^2 & V\phi & V & V\phi & \\ & & V\phi & V & \\ & & & & V \end{pmatrix}$$

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

Conditional Density

We can also rewrite the density as follows,

$$
\begin{aligned}
f_{\mathbf{y}} &= f_{y_t, y_{t-1}, \cdots, y_2, y_1} \\
&= f_{y_t | y_{t-1}, \cdots, y_2, y_1} f_{y_{t-1} | y_{t-2}, \cdots, y_2, y_1} \cdots f_{y_2 | y_1} f_{y_1} \\
&= f_{y_t | y_{t-1}} f_{y_{t-1} | y_{t-2}} \cdots f_{y_2 | y_1} f_{y_1}
\end{aligned}
$$

where,

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

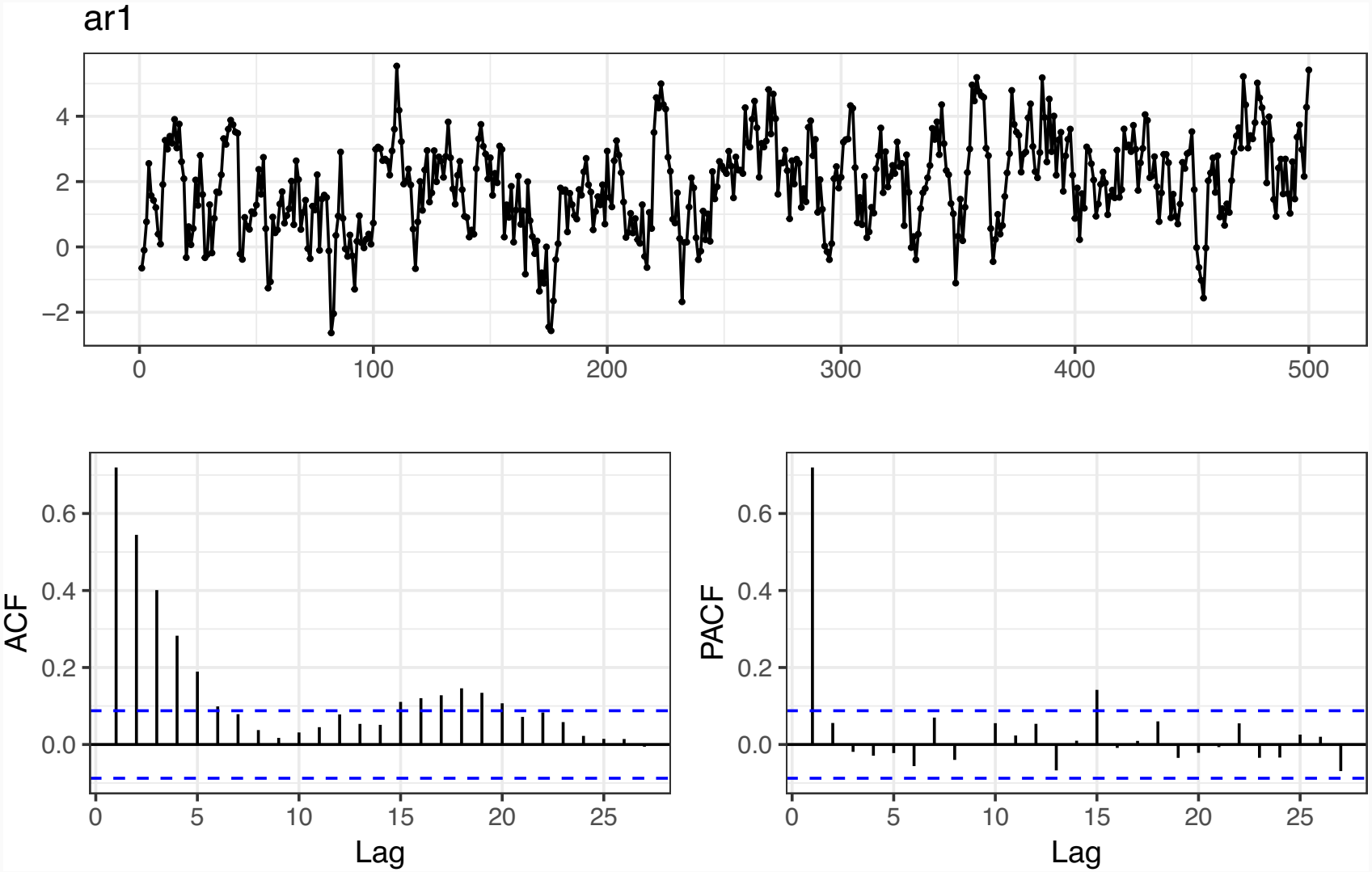
Log likelihood of AR(1)

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

$$\begin{aligned}
\ell(\delta, \phi, \sigma_w^2) = \log f_{\mathbf{y}} &= \log f_{y_1} + \sum_{i=2}^{t} \log f_{y_i|y_{i-1}} \\
&= -\frac{1}{2}\left(\log 2\pi + \log\sigma_w^2 - \log(1-\phi^2) + \frac{(1-\phi^2)}{\sigma_w^2}(y_1 - \delta)^2\right) \\
&\quad - \frac{1}{2}\left((n-1)\log 2\pi + (n-1)\log\sigma_w^2 + \frac{1}{\sigma_w^2}\sum_{i=2}^{n}(y_i - \delta + \phi\, y_{i-1})^2\right) \\
&= -\frac{1}{2}\left(n\log 2\pi + n\log\sigma_w^2 - \log(1-\phi^2)\right. \\
&\qquad \left. + \frac{1}{\sigma_w^2}\left((1-\phi^2)(y_1 - \delta)^2 + \sum_{i=2}^{n}(y_i - \delta + \phi\, y_{i-1})^2\right)\right)
\end{aligned}$$

AR(1) Example

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



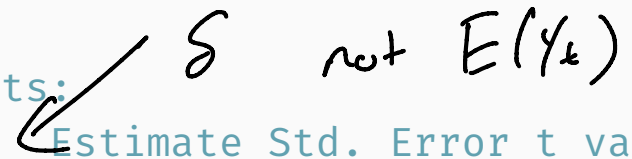
Arima

```
ar1_arima = forecast::Arima(ar1, order = c(1,0,0))
summary(ar1_arima)
## Series: ar1
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.7312  1.8934
## s.e.  0.0309  0.1646
##
## sigma^2 estimated as 0.994:  log likelihood=-707.35
## AIC=1420.71   AICc=1420.76   BIC=1433.35
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.005333274 0.9950158 0.7997576 -984.9413 1178.615 0.9246146
##                     ACF1
## Training set -0.04437489
```

$$\mu = \frac{\delta}{1-\phi} \qquad \delta = \mu(1-\phi)$$

lm

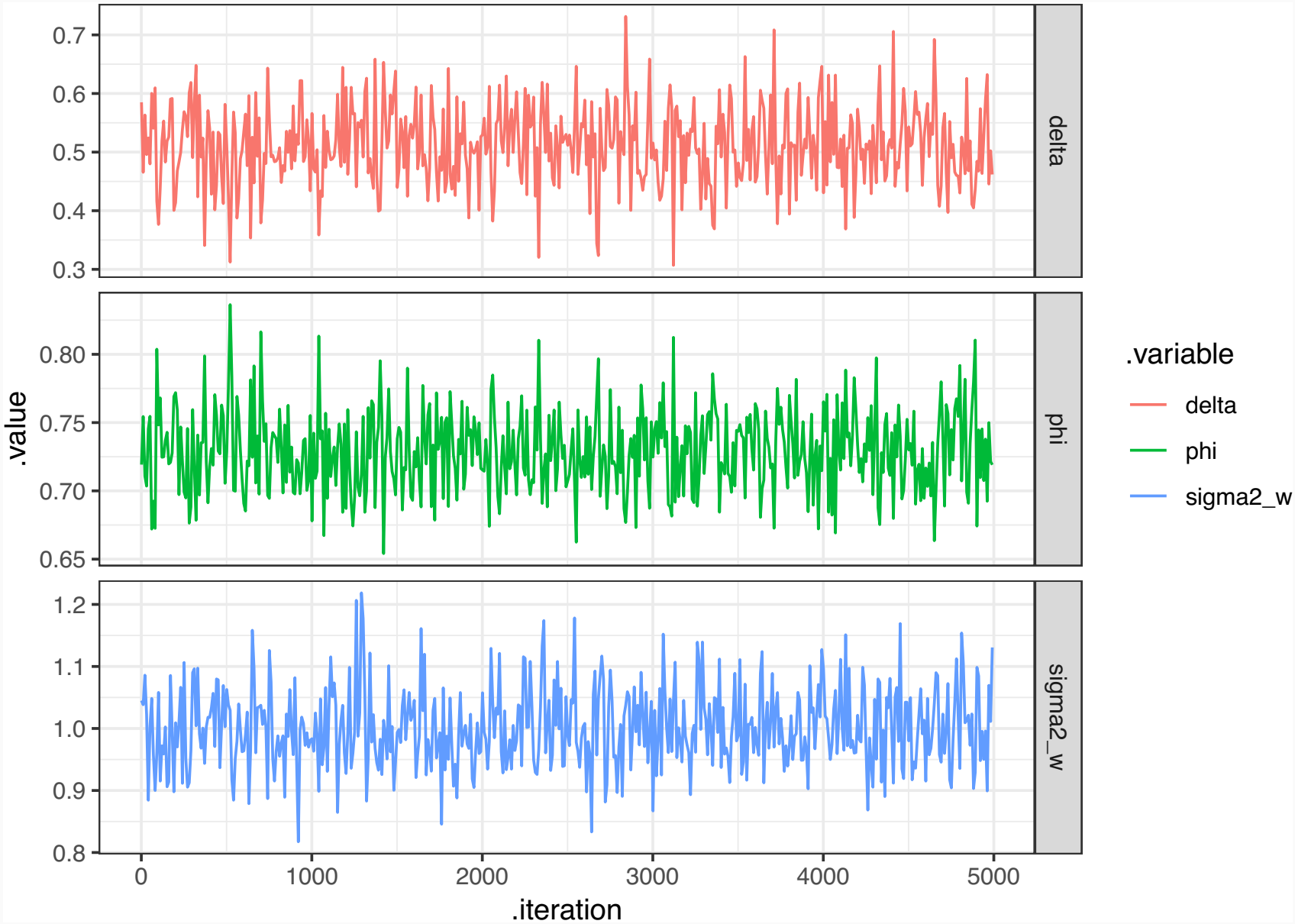
```
d = data_frame(y = ar1 %>% strip_attrs(), t=seq_along(ar1))
ar1_lm = lm(y~lag(y), data=d)
summary(ar1_lm)
##
## Call:
## lm(formula = y ~ lag(y), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2772 -0.6880  0.0785  0.6819  2.5704
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.52347    0.07328   7.144 3.25e-12 ***
## lag(y)       0.72817    0.03093  23.539  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9949 on 497 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.5272, Adjusted R-squared:  0.5262
## F-statistic: 554.1 on 1 and 497 DF,  p-value: < 2.2e-16
```



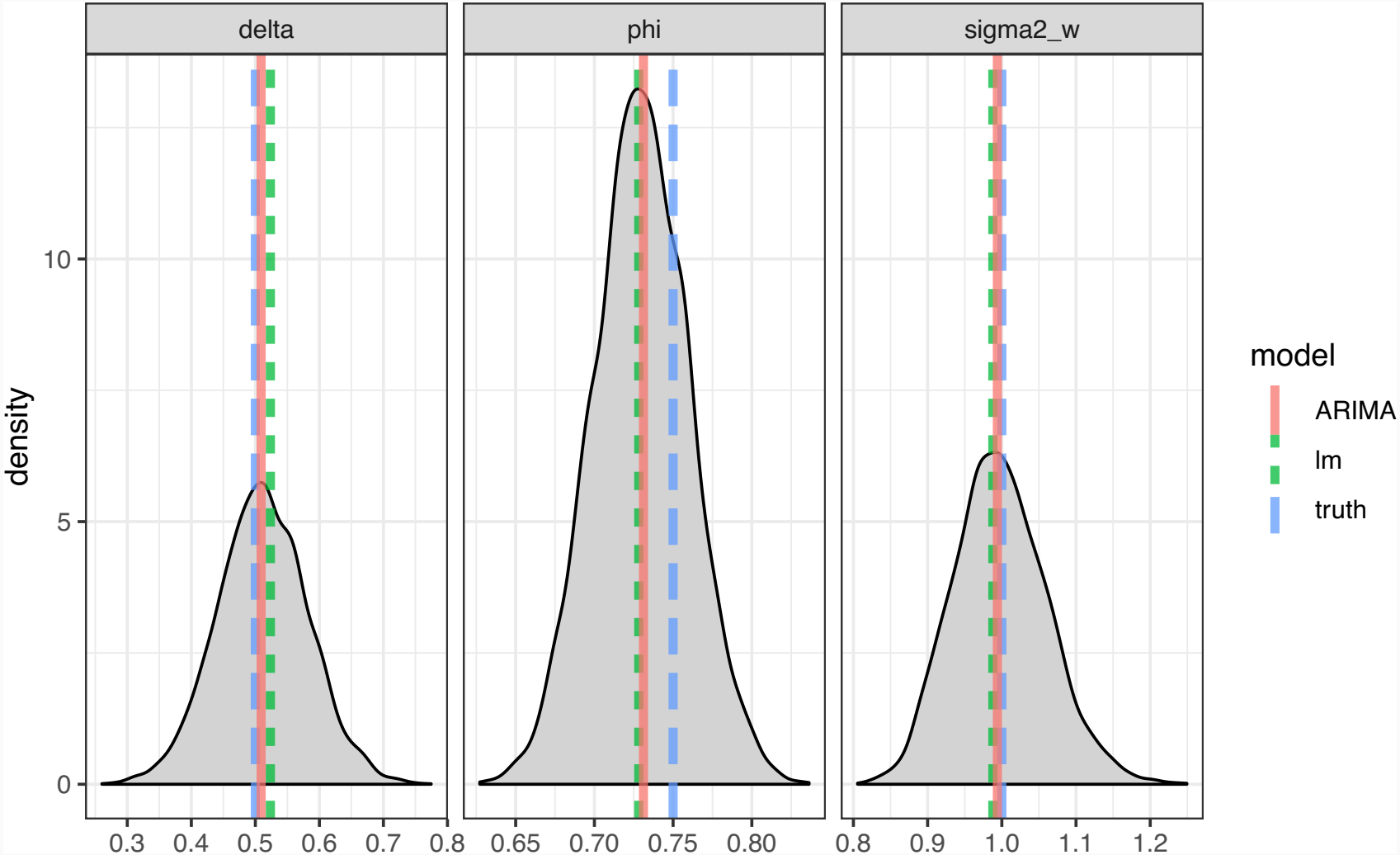
Bayesian AR(1) Model

```
ar1_model = "model{
# likelihood
  y[1] ~ dnorm(delta/(1-phi), (sigma2_w/(1-phi^2))^-1)
  y_hat[1] ~ dnorm(delta/(1-phi), (sigma2_w/(1-phi^2))^-1)

  for (t in 2:length(y)) {
    y[t] ~ dnorm(delta + phi*y[t-1], 1/sigma2_w)
    y_hat[t] ~ dnorm(delta + phi*y[t-1], 1/sigma2_w)
  }

  mu = delta/(1-phi)

# priors
  delta ~ dnorm(0,1/1000)
  phi ~ dnorm(0,1)
  tau ~ dgamma(0.001,0.001)
  sigma2_w <- 1/tau
}"
```

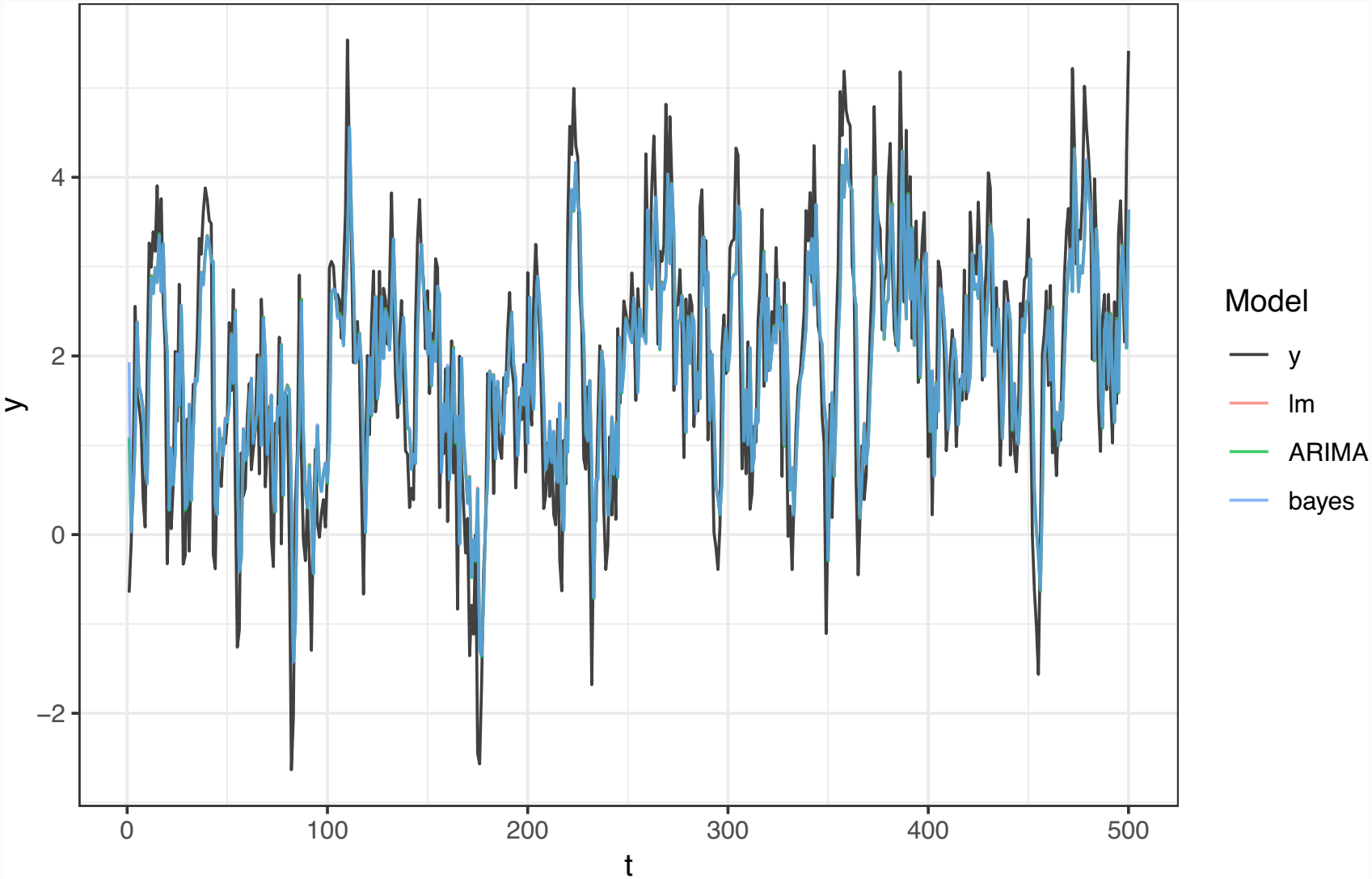
Chains


delta
phi
sigma2_w
.value
.iteration
0.7
0.6
0.5
0.4
0.3
0.80
0.75
0.70
0.65
1.2
1.1
1.0
0.9
0.8
0
1000
2000
3000
4000
5000
.variable
delta
phi
sigma2_w

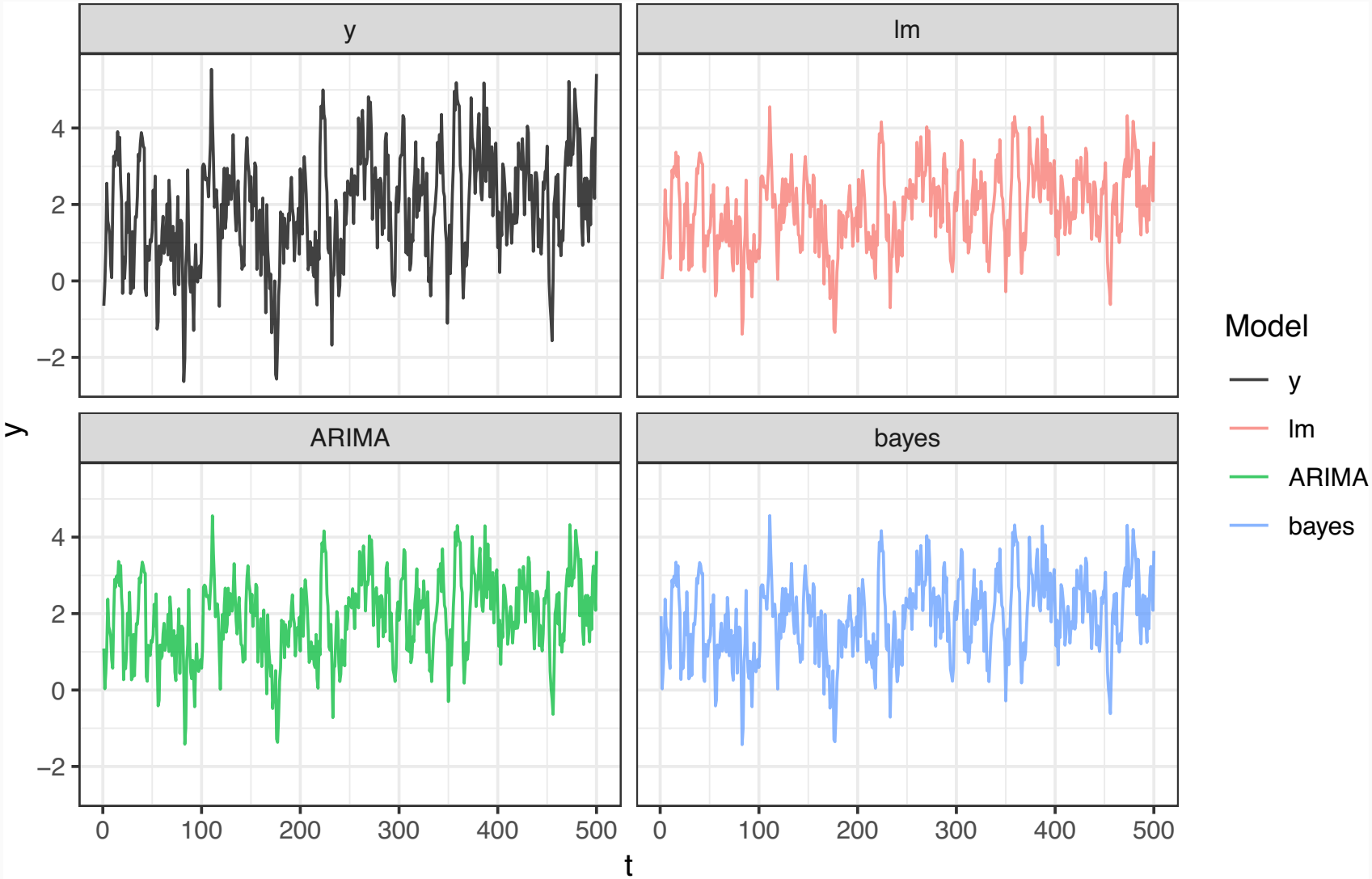
Posteriors


delta
phi
sigma2_w
density
10
5
0
0.3
0.4
0.5
0.6
0.7
0.8
0.65
0.70
0.75
0.80
0.8
0.9
1.0
1.1
1.2
model
ARIMA
lm
truth

Predictions



Faceted


y
lm
ARIMA
bayes
y
t
4
2
0
−2
0
100
200
300
400
500
Model
y
lm
ARIMA
bayes

Fitting AR(p) - Lagged Regression

We can rewrite the density as follows,

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

Fitting AR(p) - Lagged Regression

We can rewrite the density as follows,

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

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

Fitting AR(p) - Method of Moments

Recall for an AR(p) process,

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

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

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

these are called the Yule-Walker equations.

Yule-Walker

These equations can be rewritten into matrix notation as follows

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

where

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

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

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

Yule-Walker

These equations can be rewritten into matrix notation as follows

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

where

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

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

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

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

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

ARMA

Fitting $ARMA(2,2)$

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

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

Fitting $ARMA(2,2)$

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

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

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

Fitting $ARMA(2,2)$

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

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

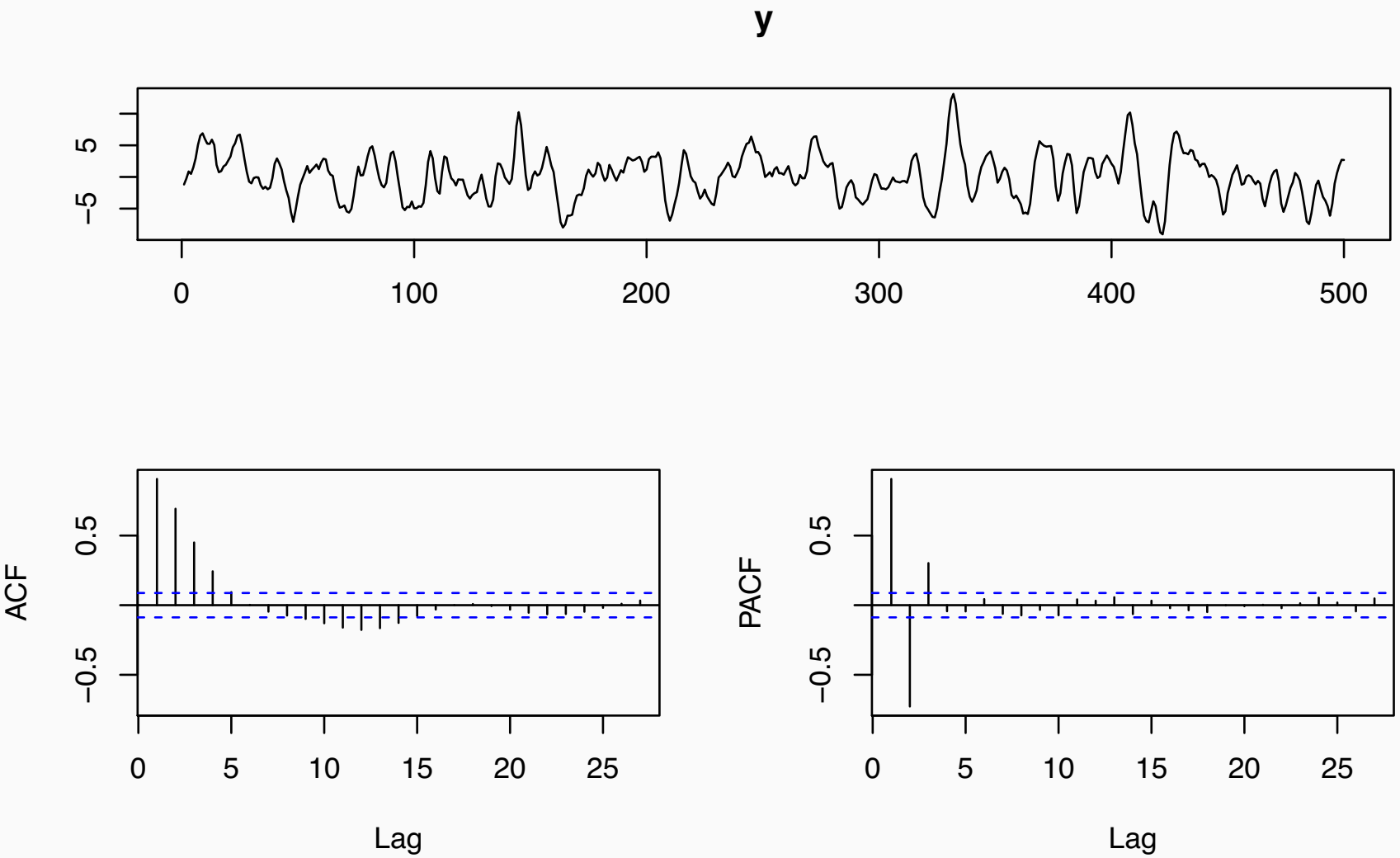
We could figure out $E(y_t)$, $Var(y_t)$, and $Cov(y_t, y_{t+h})$, but the last two are going to be pretty nasty and the full MVN likehood is similarly going to be unpleasant to work with.

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

$$\begin{aligned} y_t|\delta, & y_{t-1}, y_{t-2}, w_{t-1}, w_{t-2} \\ & \sim \mathcal{N}(\delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta_1 w_{t-1} + \theta_2 w_{t-2},\ \sigma_w^2) \end{aligned}$$

ARMA(2,2) Example

with $\phi = (1.3, -0.5)$, $\theta = (0.5, 0.2)$, $\delta = 0$, and $\sigma_w^2 = 1$ using the same models



ARIMA

```
forecast::Arima(y, order = c(2,0,2), include.mean = FALSE) %>% summary()
## Series: y
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2     ma1     ma2
##       1.2318  -0.4413  0.5888  0.2400
## s.e.  0.0843   0.0767  0.0900  0.0735
##
## sigma^2 estimated as 0.9401:  log likelihood=-693.82
## AIC=1397.64   AICc=1397.77   BIC=1418.72
##
## Training set error measures:
##                       ME     RMSE       MAE      MPE     MAPE      MASE
## Training set 0.00952552 0.965688 0.7680317 14.16358 142.9927 0.6321085
##                       ACF1
## Training set -0.0007167096
```

AR only lm

```
lm(y ~ lag(y,1) + lag(y,2)) %>% summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7407 -0.6299  0.0012  0.7195  3.2872
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01196    0.04600    0.26    0.795
## lag(y, 1)    1.57234    0.03057   51.43   <2e-16 ***
## lag(y, 2)   -0.73298    0.03059  -23.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.026 on 495 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.9181, Adjusted R-squared:  0.9178
## F-statistic:  2775 on 2 and 495 DF,  p-value: < 2.2e-16
```

1.3
-0.5

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

3. Regress y_t onto $y_{t-1}, \dots, y_{t-p}, \hat{w}_{t-1}, \dots \hat{w}_{t-q}$

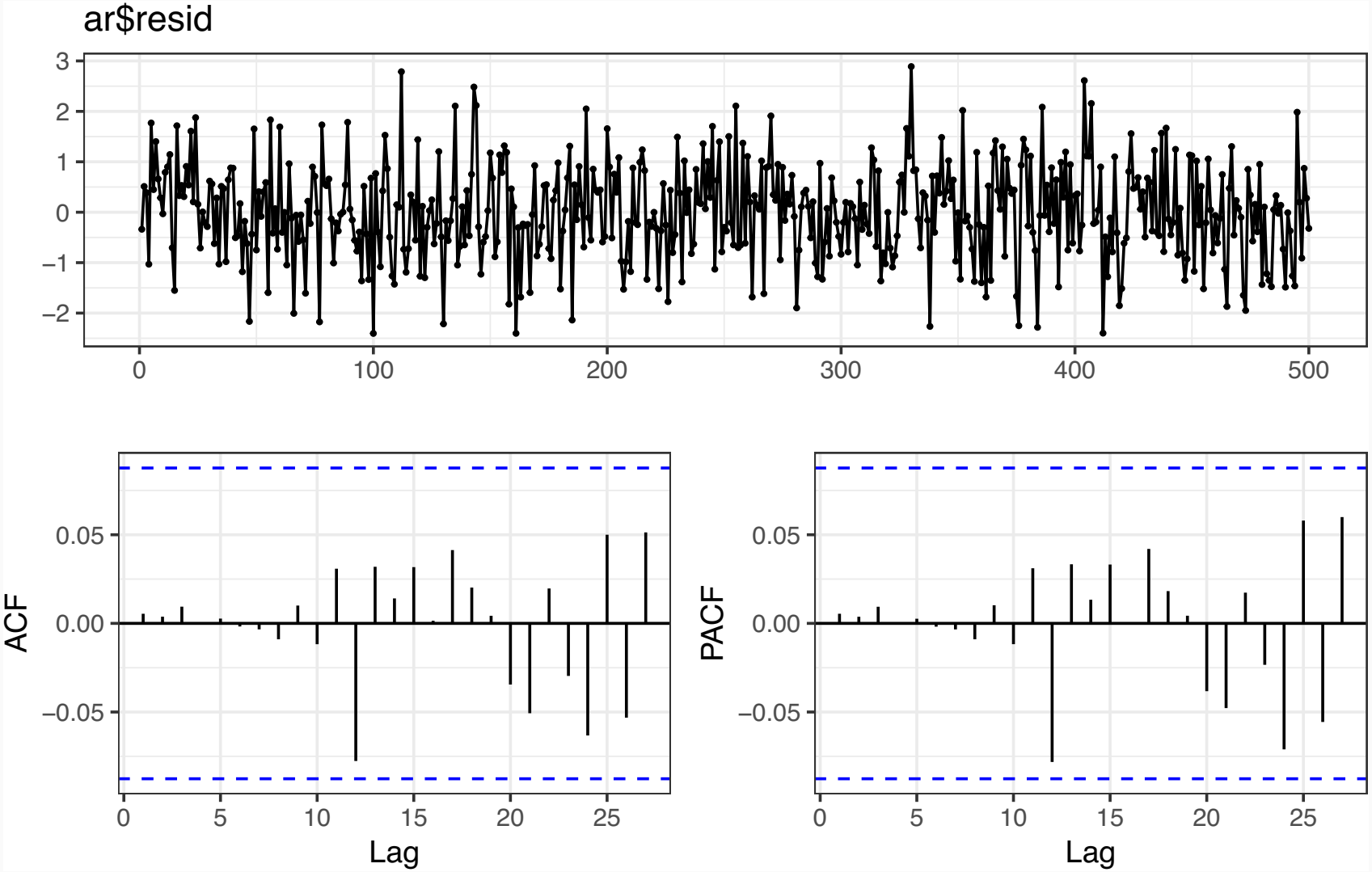
4. Update $\hat{w}_{t-1}, \dots \hat{w}_{t-q}$ based on current model, refit and then repeat until convergence

Hannan-Rissanen - Step 1 & 2

```
ar = ar.mle(y, order.max = 10)
ar = forecast::Arima(y, order = c(10,0,0))
ar
## Series: y
## ARIMA(10,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3     ar4      ar5     ar6     ar7      ar8
##       1.8069  -1.2555  0.3071  0.1379  -0.2025  0.0932  0.0266  -0.0665
## s.e.  0.0446   0.0924  0.1084  0.1097   0.1100  0.1099  0.1101   0.1096
##          ar9     ar10    mean
##       0.0687  -0.0634  0.0673
## s.e.  0.0935   0.0451  0.2910
##
## sigma^2 estimated as 0.9402:  log likelihood=-690.36
## AIC=1404.72   AICc=1405.36   BIC=1455.3
```

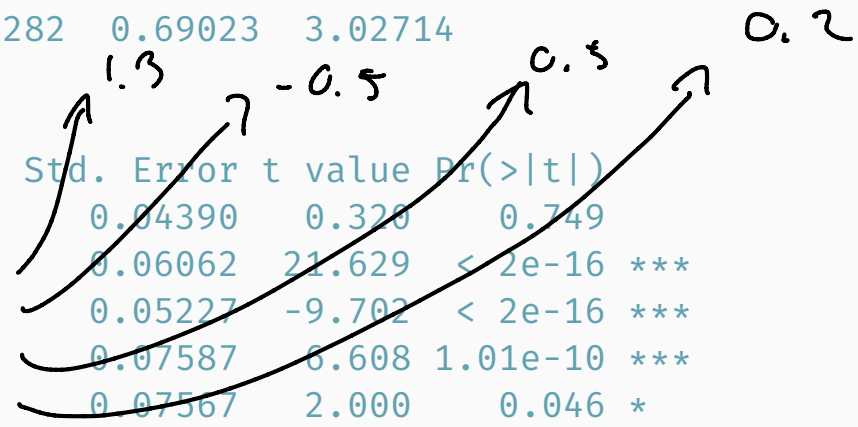
Residuals

```
forecast::ggtsdisplay(ar$resid)
```



Hannan-Rissanen - Step 3

```
d = data_frame(
  y = y %>% strip_attrs(),
  w_hat1 = ar$resid %>% strip_attrs()
)

(lm1 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat1,1) + lag(w_hat1,2), data=d)) %>%
  summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2) + lag(w_hat1, 1) + lag(w_hat1,
##     2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62536 -0.59631  0.00282  0.69023  3.02714
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.01405    0.04390   0.320    0.749
## lag(y, 1)       1.31110    0.06062  21.629  < 2e-16 ***
## lag(y, 2)      -0.50714    0.05227  -9.702  < 2e-16 ***
## lag(w_hat1, 1)  0.50136    0.07587   6.608 1.01e-10 ***
## lag(w_hat1, 2)  0.15136    0.07567   2.000    0.046 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9792 on 493 degrees of freedom
##   (2 observations deleted due to missingness)
```



Hannan-Rissanen - Step 4.1

```
d = modelr::add_residuals(d,lm1,"w_hat2")

(lm2 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat2,1) + lag(w_hat2,2), data=d)) %>%
  summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2) + lag(w_hat2, 1) + lag(w_hat2,
##     2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.48938 -0.63467 -0.02144  0.64875  3.06169
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.01732    0.04379   0.396   0.6926
## lag(y, 1)       1.28435    0.06181  20.780  < 2e-16 ***
## lag(y, 2)      -0.48458    0.05322  -9.106  < 2e-16 ***
## lag(w_hat2, 1)  0.52941    0.07545   7.017 7.58e-12 ***
## lag(w_hat2, 2)  0.17436    0.07553   2.308   0.0214 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9748 on 491 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.9267, Adjusted R-squared:  0.9261
## F-statistic:  1553 on 4 and 491 DF,  p-value: < 2.2e-16
```

Hannan-Rissanen - Step 4.2

```
d = modelr::add_residuals(d,lm2,"w_hat3")

(lm3 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat3,1) + lag(w_hat3,2), data=d)) %>%
  summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2) + lag(w_hat3, 1) + lag(w_hat3,
##     2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.58530 -0.61844 -0.03113  0.66855  2.94198
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.01411    0.04373   0.323  0.74712
## lag(y, 1)       1.26447    0.06200  20.395  < 2e-16 ***
## lag(y, 2)      -0.46888    0.05335  -8.788  < 2e-16 ***
## lag(w_hat3, 1)  0.55598    0.07631   7.285 1.29e-12 ***
## lag(w_hat3, 2)  0.20607    0.07650   2.694  0.00731 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9714 on 489 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.9274, Adjusted R-squared:  0.9268
## F-statistic:  1562 on 4 and 489 DF,  p-value: < 2.2e-16
```

Hannan-Rissanen - Step 4.3

```
d = modelr::add_residuals(d,lm3,"w_hat4")

(lm4 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat4,1) + lag(w_hat4,2), data=d)) %>%
  summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2) + lag(w_hat4, 1) + lag(w_hat4,
##     2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.47793 -0.62637 -0.02602  0.67771  3.01388
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.006615   0.043826   0.151   0.8801
## lag(y, 1)       1.270581   0.062110  20.457  < 2e-16 ***
## lag(y, 2)      -0.474331   0.053406  -8.882  < 2e-16 ***
## lag(w_hat4, 1)  0.546706   0.076724   7.126 3.75e-12 ***
## lag(w_hat4, 2)  0.195828   0.077098   2.540   0.0114 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9719 on 487 degrees of freedom
##   (8 observations deleted due to missingness)
## Multiple R-squared:  0.9269, Adjusted R-squared:  0.9263
## F-statistic:  1543 on 4 and 487 DF,  p-value: < 2.2e-16
```

Hannan-Rissanen - Step 4.4

```
d = modelr::add_residuals(d,lm4,"w_hat5")

(lm5 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat5,1) + lag(w_hat5,2), data=d)) %>%
  summary()
##
## Call:
## lm(formula = y ~ lag(y, 1) + lag(y, 2) + lag(w_hat5, 1) + lag(w_hat5,
##     2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.52482 -0.61986 -0.01755  0.65755  3.01411
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.002506   0.043950   0.057  0.95456
## lag(y, 1)       1.261588   0.062919  20.051  < 2e-16 ***
## lag(y, 2)      -0.468001   0.053869  -8.688  < 2e-16 ***
## lag(w_hat5, 1)  0.555958   0.077424   7.181 2.62e-12 ***
## lag(w_hat5, 2)  0.202817   0.077796   2.607  0.00941 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9728 on 485 degrees of freedom
##   (10 observations deleted due to missingness)
## Multiple R-squared:  0.9261, Adjusted R-squared:  0.9255
## F-statistic:  1519 on 4 and 485 DF,  p-value: < 2.2e-16
```

RMSEs

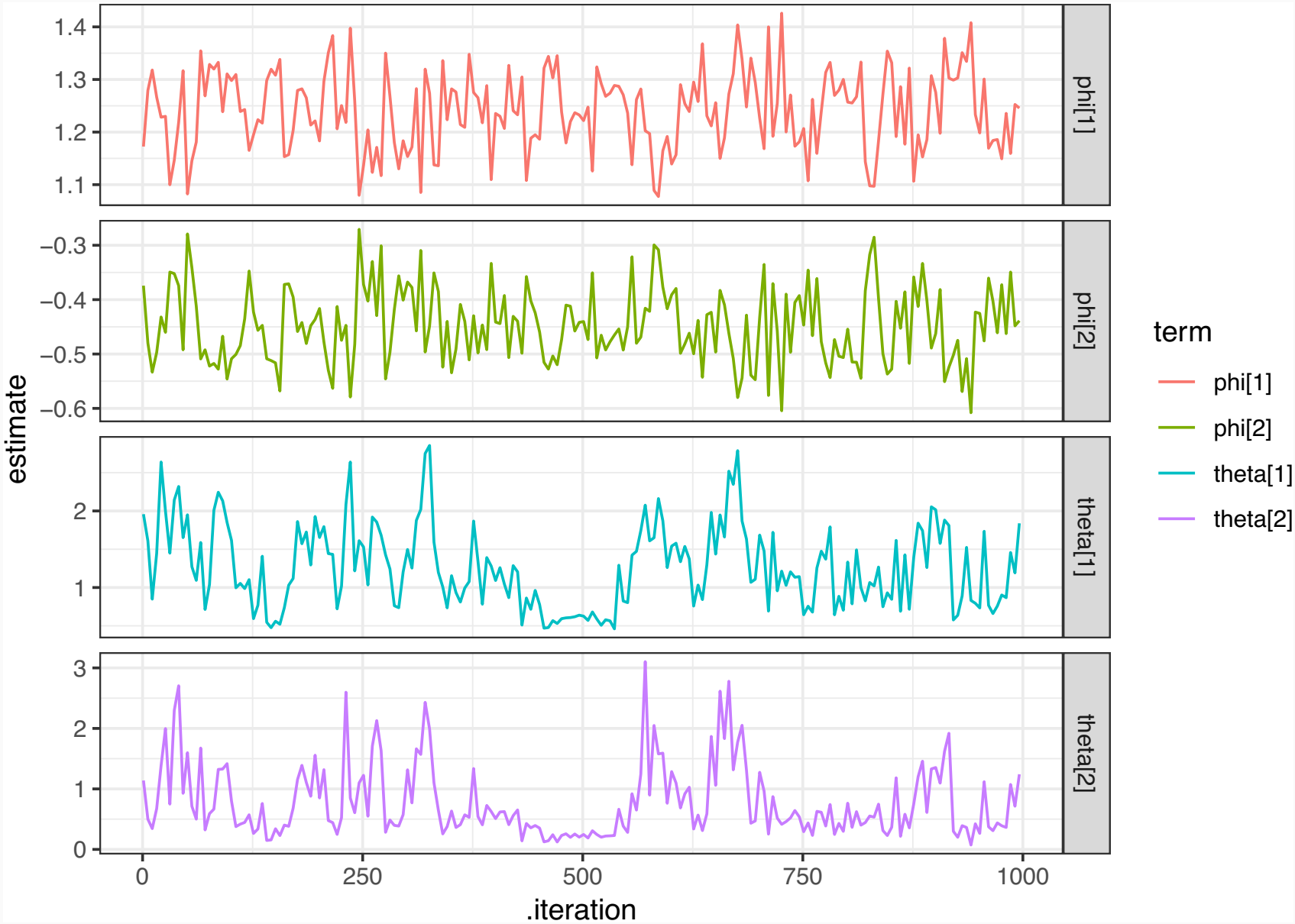
```
modelr::rmse(lm1, data = d)
## [1] 0.9743158

modelr::rmse(lm2, data = d)
## [1] 0.9698713

modelr::rmse(lm3, data = d)
## [1] 0.9665092

modelr::rmse(lm4, data = d)
## [1] 0.9669626

modelr::rmse(lm5, data = d)
## [1] 0.9678429
```

JAGS - Model

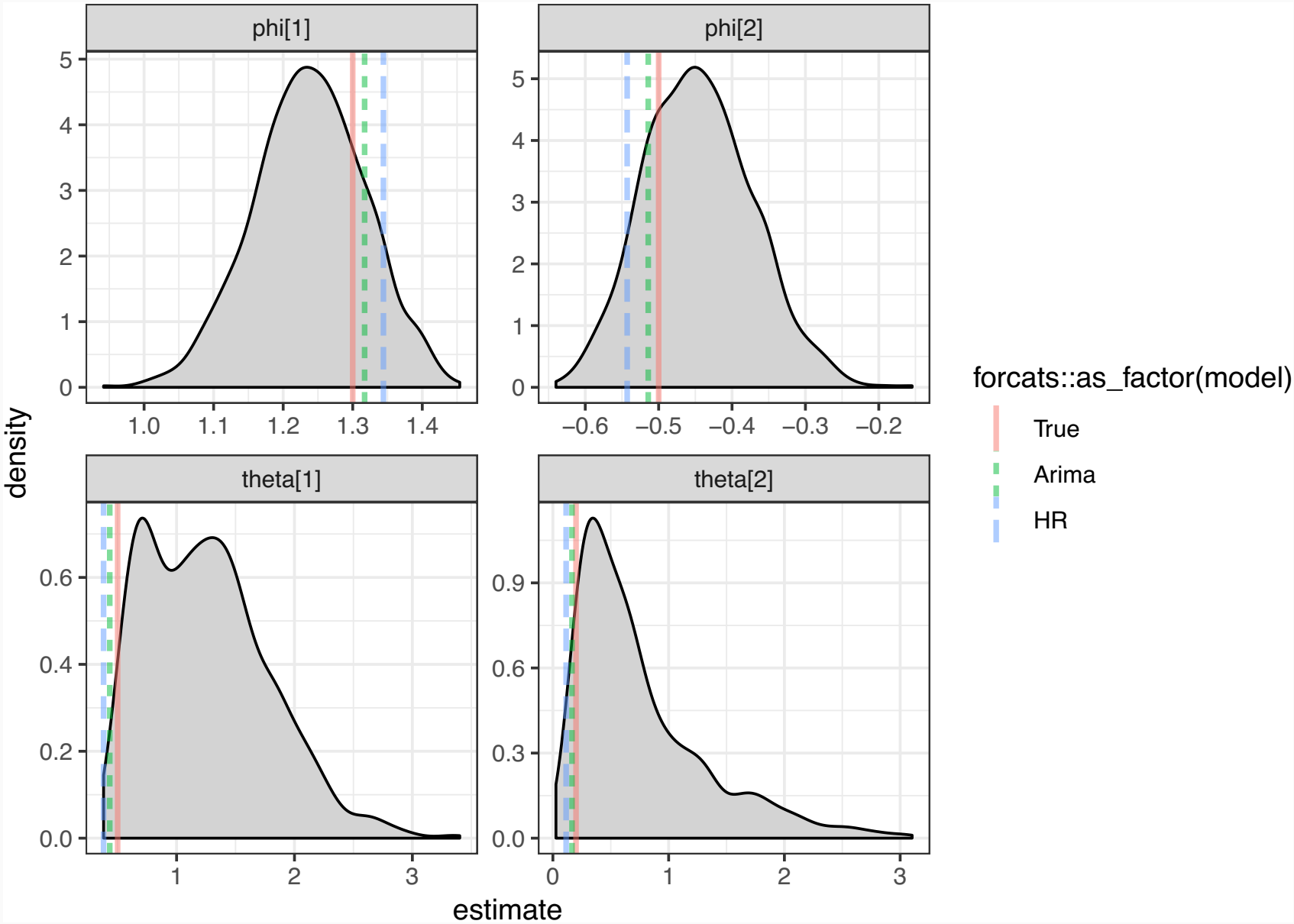
```
arma22_model = "model{
# Likelihood
  for (t in 1:length(y)) {
    y[t] ~ dnorm(mu[t], 1/sigma2_e)
  }

  mu[1] = phi[1] * y_0  + phi[2] * y_n1 + w[1] + theta[1]*w_0  + theta[2]*w_n1
  mu[2] = phi[1] * y[1] + phi[2] * y_0  + w[2] + theta[1]*w[1] + theta[2]*w_0

  for (t in 3:length(y)) {
    mu[t] = phi[1] * y[t-1] + phi[2] * y[t-2] + w[t] + theta[1] * w[t-1] + theta[2] * w[t-2]
  }

# Priors
  for(t in 1:length(y)){
    w[t] ~ dnorm(0,1/sigma2_w)
  }

  sigma2_w = 1/tau_w; tau_w ~ dgamma(0.001, 0.001)
  sigma2_e = 1/tau_e; tau_e ~ dgamma(0.001, 0.001)
  for(i in 1:2) {
    phi[i] ~ dnorm(0,1)
    theta[i] ~ dnorm(0,1)
  }

# Latent errors and series values
  w_0  ~ dnorm(0, 1/sigma2_w)
  w_n1 ~ dnorm(0, 1/sigma2_w)
  y_0  ~ dnorm(0, 1/4)
  y_n1 ~ dnorm(0, 1/4)
}"
```

$w[t] = y[t] - mu[t]$

JAGS - Fit


estimate
1.4
1.3
1.2
1.1
phi[1]
−0.3
−0.4
−0.5
−0.6
phi[2]
2
1
theta[1]
3
2
1
0
theta[2]
0
250
500
750
1000
.iteration
term
phi[1]
phi[2]
theta[1]
theta[2]

JAGS - Posteriors

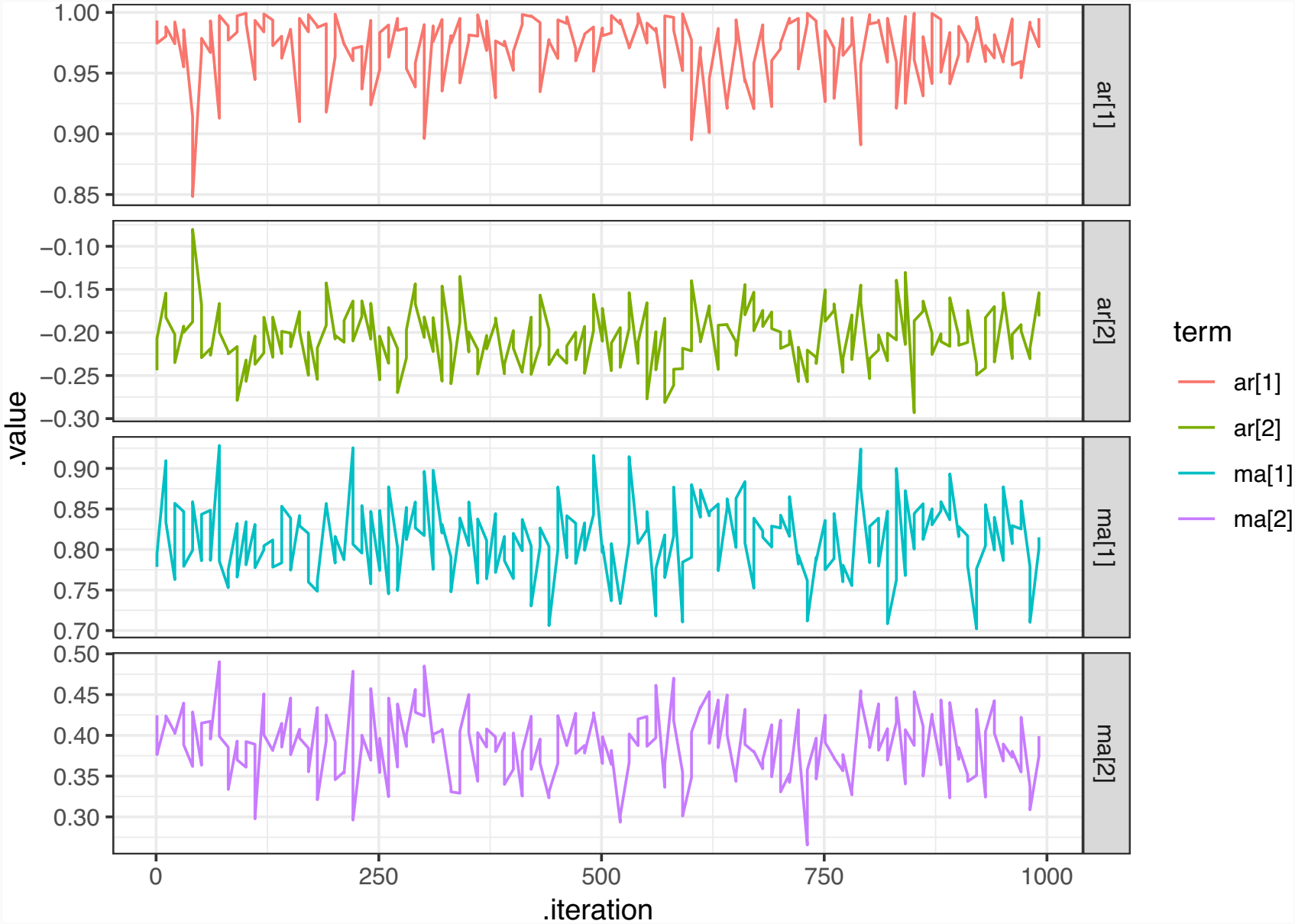


Using brms

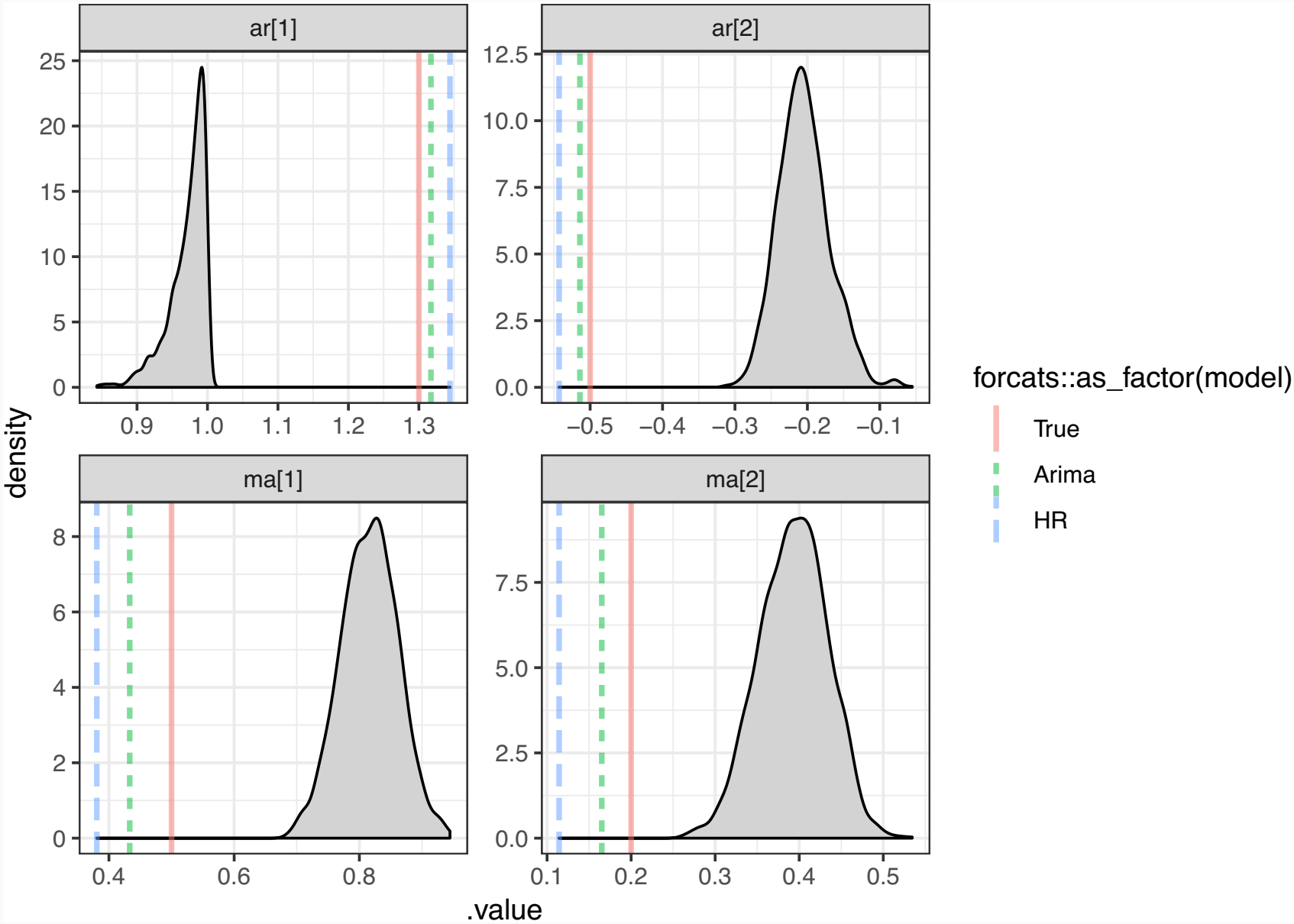
```
library(brms)
b = brm(y~1, data = d, autocor = cor_arma(p=2,q=2), chains = 2)
```

```
## Compiling the C++ model
## Start sampling
##
## SAMPLING FOR MODEL '90ac5aa650e5db28db8b9b83a5eb3c07' NOW (CHAIN 1).
##
## Gradient evaluation took 0.001742 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 17.42 seconds.
## Adjust your expectations accordingly!
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 13.2358 seconds (Warm-up)
##                9.79355 seconds (Sampling)
##                23.0293 seconds (Total)
```

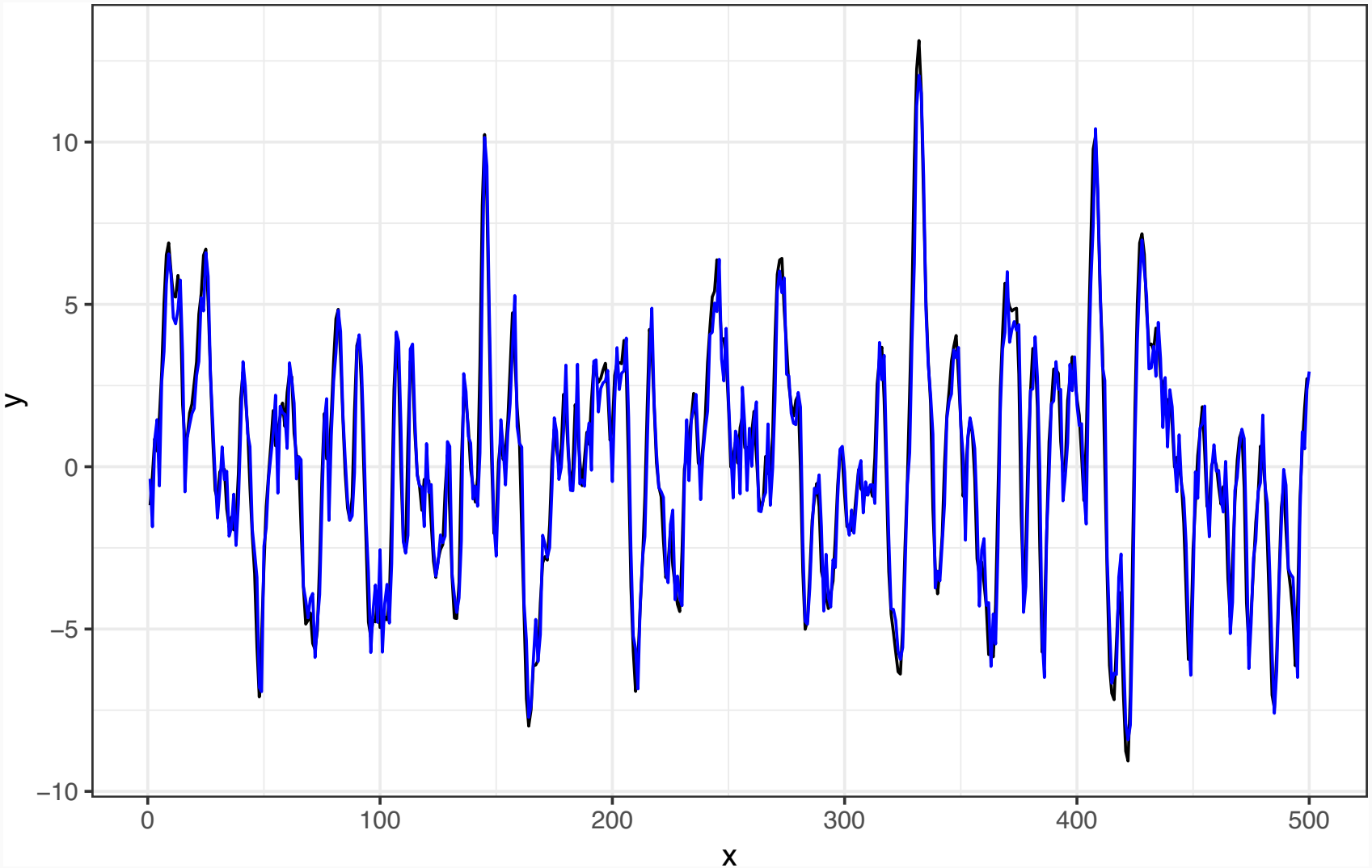
BRMS - Chains


1.00
0.95
0.90
0.85
ar[1]
−0.10
−0.15
−0.20
−0.25
−0.30
ar[2]
0.90
0.85
0.80
0.75
0.70
ma[1]
0.50
0.45
0.40
0.35
0.30
ma[2]
.value
0
250
500
750
1000
.iteration
term
ar[1]
ar[2]
ma[1]
ma[2]

BRMS - Posteriors



BRMS - Predictions



BRMS - Stancode

```
stancode(b)
## // generated with brms 2.5.0
## 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
##   int<lower=0> J_lag[N];
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   int max_lag = max(Kar, Kma);
## }
## parameters {
##   real temp_Intercept;  // temporary intercept
##   real<lower=0> sigma;  // residual SD
##   vector<lower=-1,upper=1>[Kar] ar;  // autoregressive effects
##   vector<lower=-1,upper=1>[Kma] ma;  // moving-average effects
## }
## transformed parameters {
## }
## model {
##   vector[N] mu = temp_Intercept + rep_vector(0, N);
##   // objects storing residuals
##   matrix[N, max_lag] E = rep_matrix(0, N, max_lag);
##   vector[N] e;
##   for (n in 1:N) {
##     mu[n] += head(E[n], Kma) * ma;
##     // computation of ARMA correlations
##     e[n] = Y[n] - mu[n];
##     for (i in 1:J_lag[n]) {
##       E[n + 1, i] = e[n + 1 - i];
##     }
##     mu[n] += head(E[n], Kar) * ar;
##   }
```