

Fitting ARIMA Models

Lecture 12

Dr. Colin Rundel

Model Fitting

Fitting ARIMA

For an 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 ARMA(p, q) model to y'_t .
- To keep things simple we'll assume $w_t \stackrel{\text{iid}}{\sim} (0, \sigma_w^2)$

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 $\mathbf{y} = (y_1, y_2, \dots, y_t)'$ 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

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

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

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

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

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 , ϕ , and σ_w^2 .

Conditional Density

We can also rewrite the density as follows,

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

where,

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

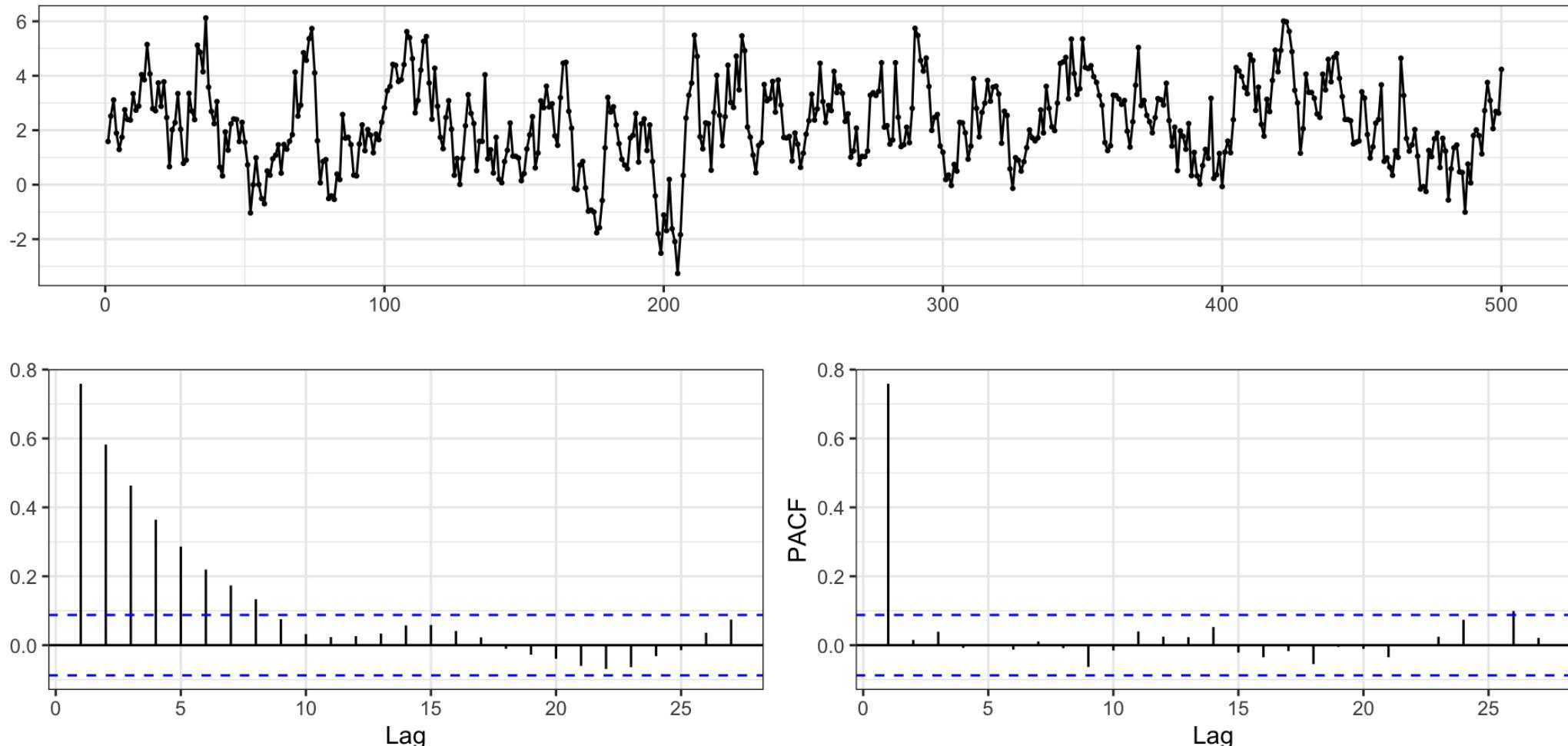
Log likelihood of AR(1)

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

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

AR(1) Example

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



ARIMA

```
1 ( ar1_arima = forecast::Arima(ar1, order = c(1,0,0)) )
```

Series: ar1

ARIMA(1,0,0) with non-zero mean

Coefficients:

	ar1	mean
	0.7601	2.2178
s.e.	0.0290	0.1890

$\sigma^2 = 1.045$: log likelihood = -719.84

AIC=1445.67 AICc=1445.72 BIC=1458.32

mean vs δ ?

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

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

True $E(y_t)$:

```
1 0.5 / (1-0.75)  
[1] 2
```

Sample δ :

```
1 ar1_arima$coef[2] *  
2 (1 - ar1_arima$model$phi)
```

intercept
0.5319962

lm

```
1 d = tsibble::as_tsibble(ar1) %>%
2   as_tibble() %>%
3   rename(y = value)
4 summary({ ar1_lm = lm(y~lag(y), data=d) })
```

Call:

```
lm(formula = y ~ lag(y), data = d)
```

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(y)	0.76141	0.02918	26.090	< 2e-16 ***

Bayesian AR(1) Model

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

Family: gaussian

Links: mu = identity; sigma = identity

Formula: y ~ arma(p = 1, q = 0)

Data: d (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	0.71	0.82	1.00		3820	2893	

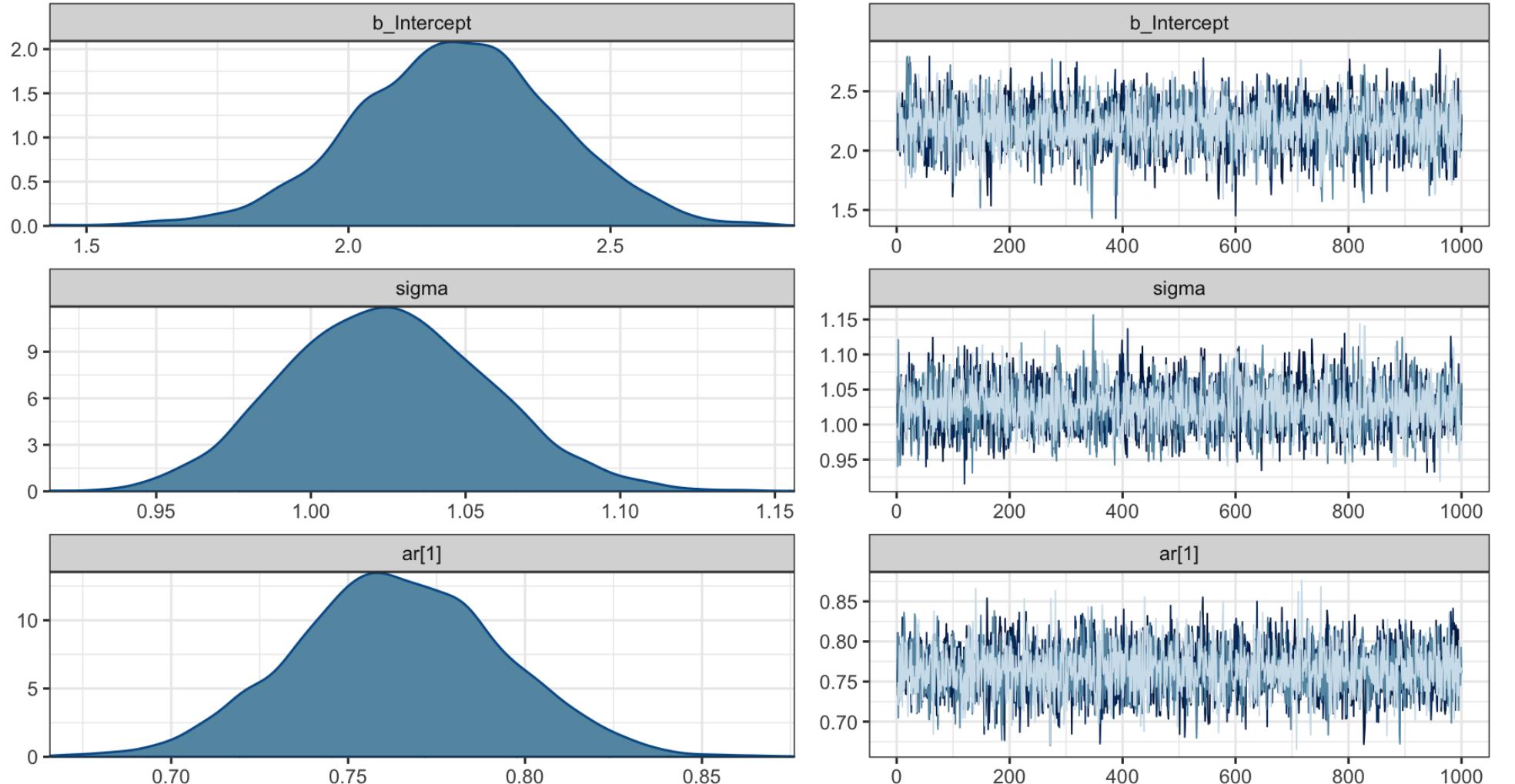
Population-Level Effects:

	Estimate	Est.Error	l-95%	CI	u-95%	CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	2.20	0.19	1.82	2.58	1.00		3728	2811	

Family Specific Parameters:

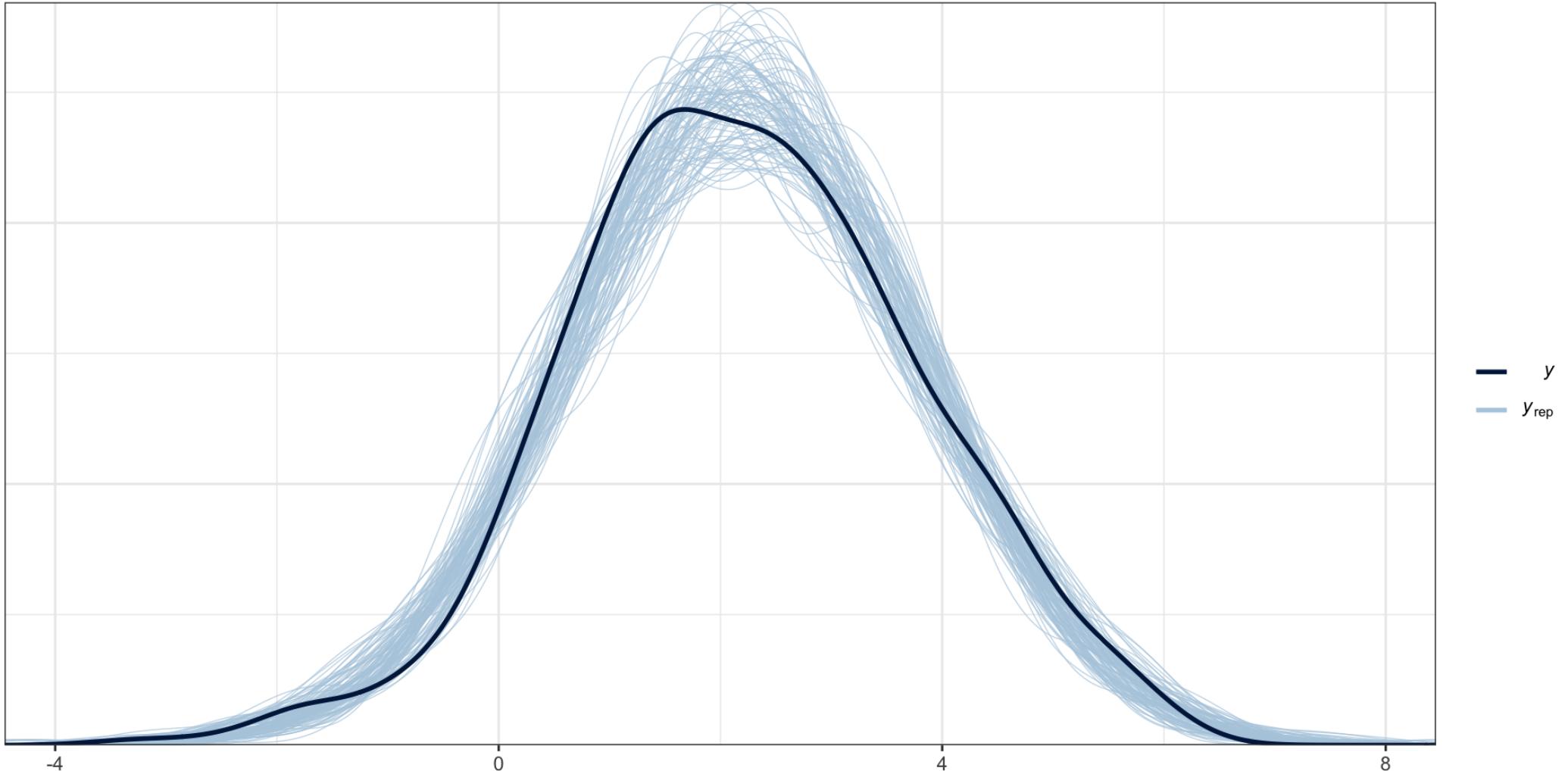
Chains

```
1 plot(ar1_brms)
```

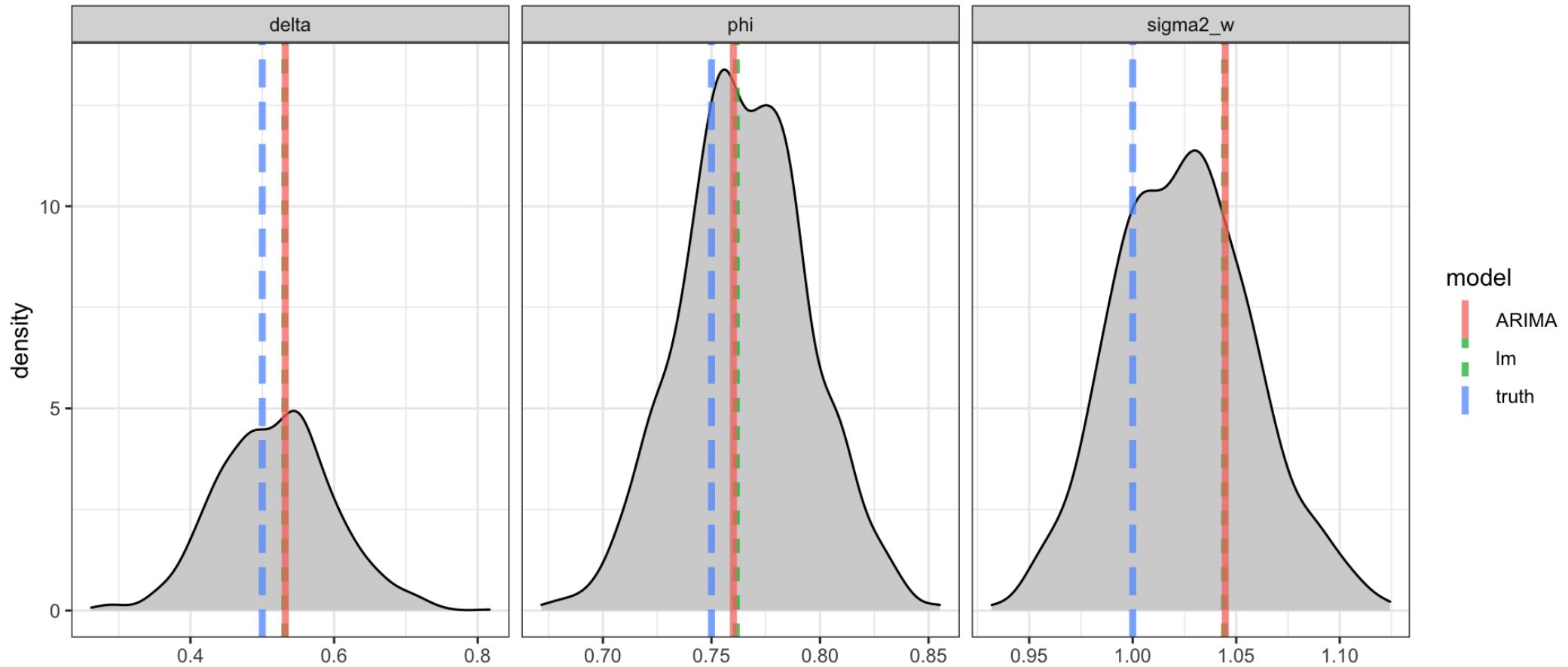


PP Checks

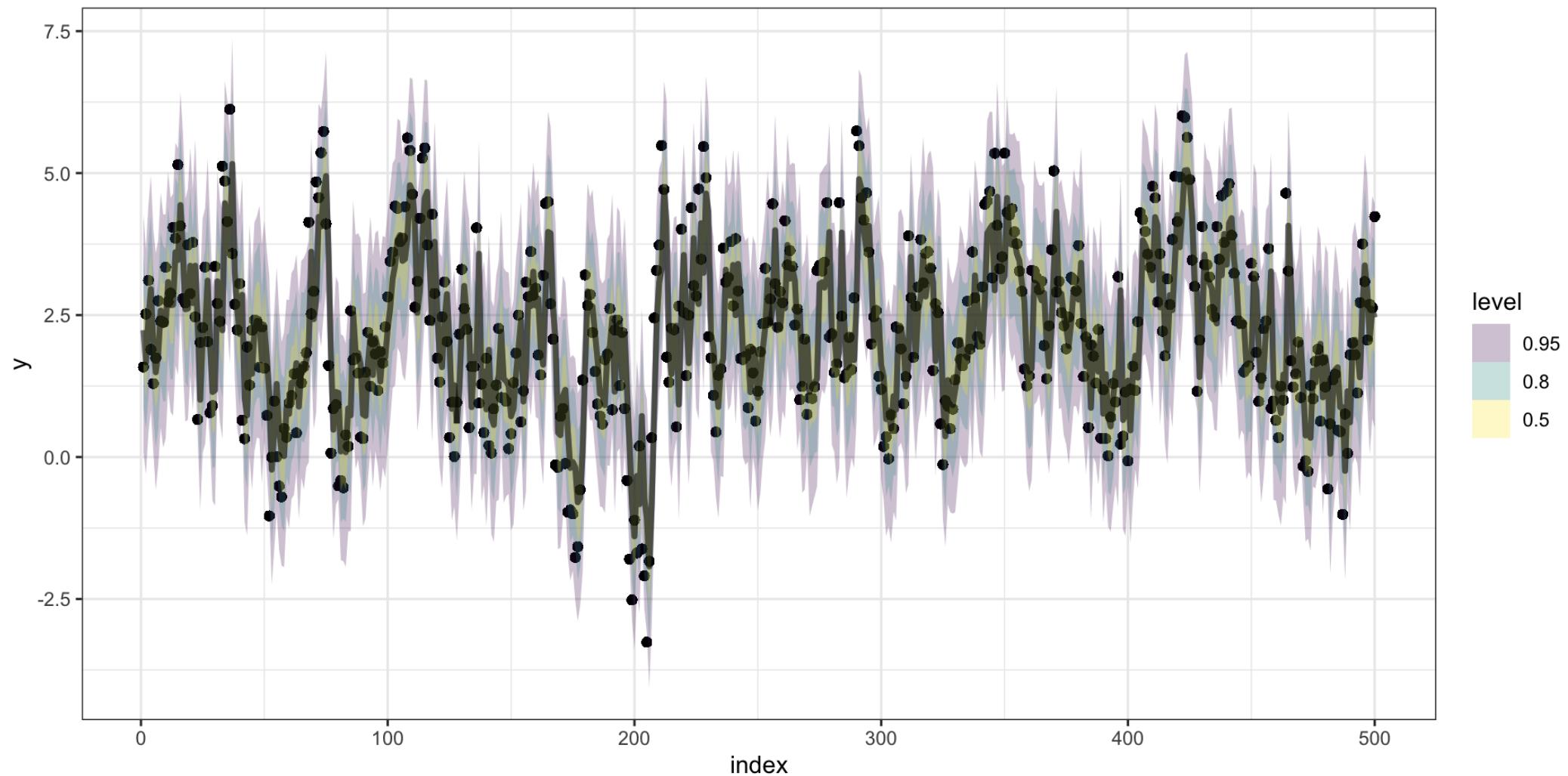
```
1 pp_check(ar1_brms, ndraws=100)
```



Posteriors

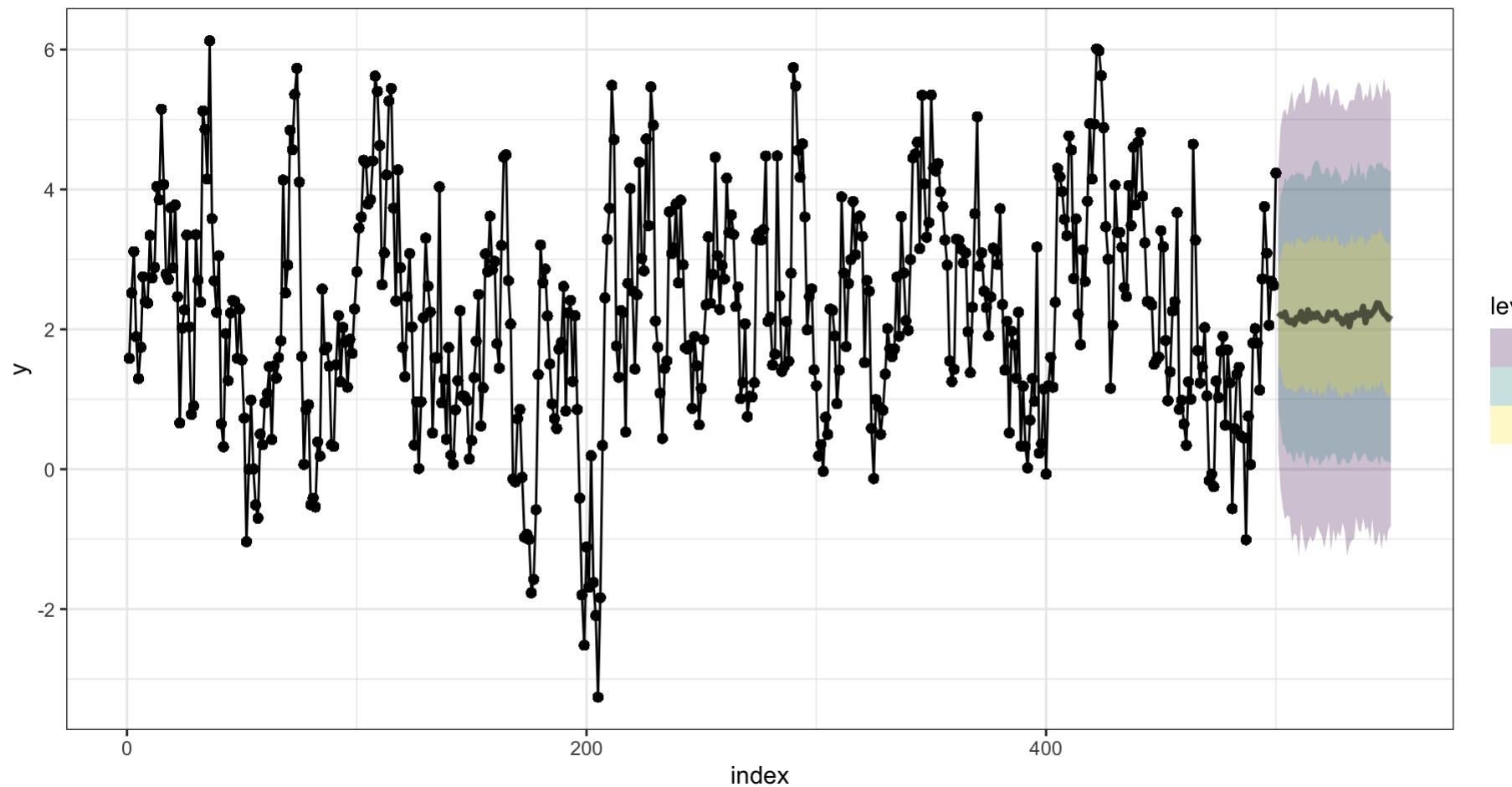


Predictions



Forecasting

```
1 ar1_brms_fc = ar1_brms %>%
2   predicted_draws_fix(
3     newdata = tibble(index=501:550, y=NA)
4   ) %>%
5   filter(.chain == 1)
```



⋮
⋮

Fitting AR(p)

Lagged Regression

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

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

Regressing y_t on y_{t-1}, \dots, y_{t-p} gets us an approximate solution, but it ignores the $f(y_1, y_2, \dots, y_p)$ 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 $p \ll n$ then probably not much

Method of Moments

Recall for an AR(p) process,

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

$$\gamma(h) = \phi_1\gamma(h-1) + \phi_2\gamma(h-2) + \dots + \phi_p\gamma(h-p)$$

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

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

these are called the Yule-Walker equations.

Yule-Walker

These equations can be rewritten into matrix notation as follows

$$\begin{matrix} \boldsymbol{\Gamma}_p & \boldsymbol{\phi} \\ p \times p & p \times 1 \end{matrix} = \begin{matrix} \boldsymbol{\gamma}_p \\ p \times 1 \end{matrix} \quad \sigma_w^2 = \gamma(0) - \boldsymbol{\phi}' \boldsymbol{\gamma}_p$$

where

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

$$\boldsymbol{\phi} = (\phi_1, \phi_2, \dots, \phi_p)'$$

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

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

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

ARMA

Fitting ARMA(2, 2)

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

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

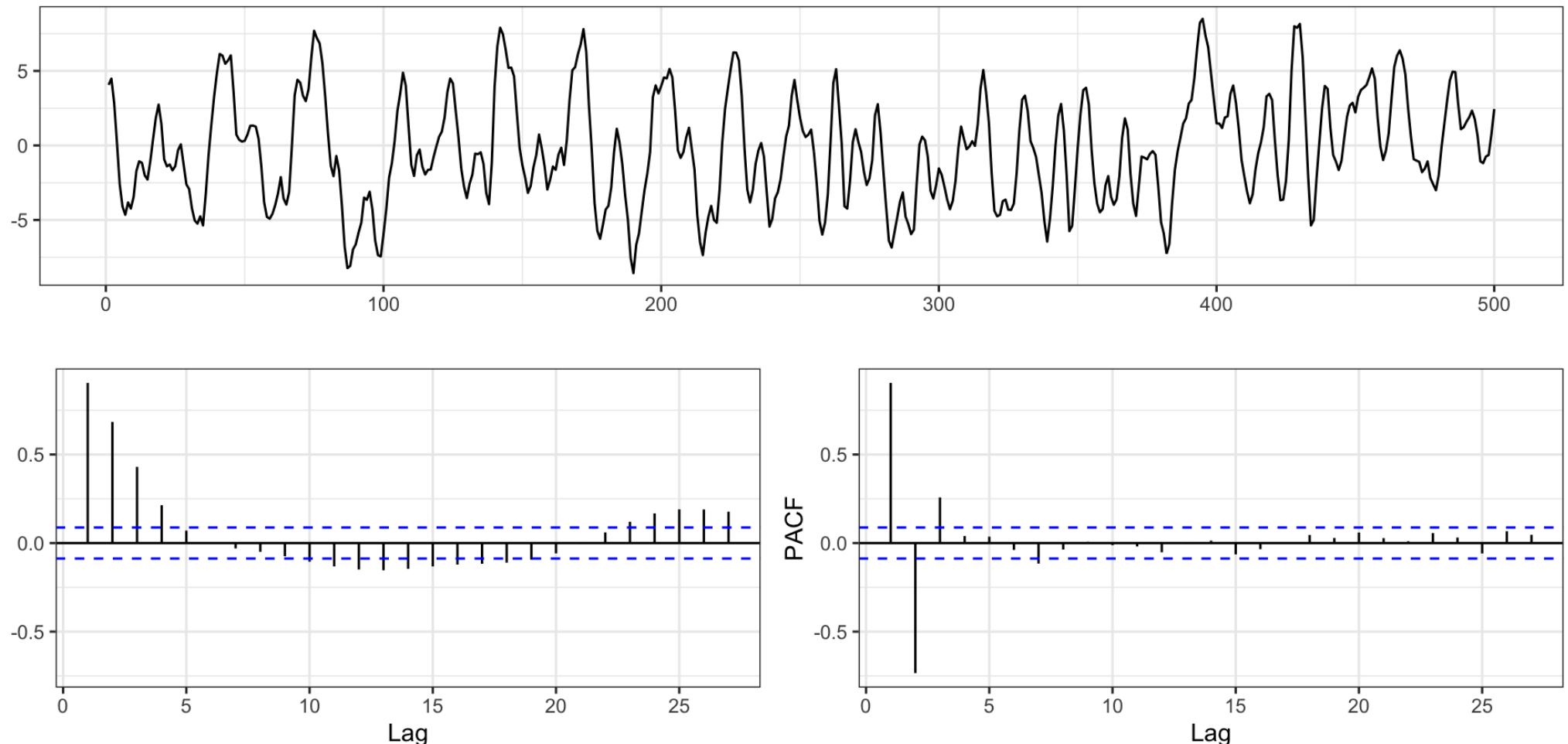
We could figure out $E(y_t)$, $\text{Var}(y_t)$, and $\text{Cov}(y_t, y_{t+h})$, but the last two are going to be pretty nasty and the full MVN likelihood is similarly going to be unpleasant to work with.

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

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

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

```
1 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
	1.3702	-0.5634	0.4324	0.1274
s.e.	0.0652	0.0597	0.0740	0.0593

sigma^2 = 0.928: log likelihood = -690.59

AIC=1391.18 AICc=1391.3 BIC=1412.25

Training set error measures:

AR only lm

```
1 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
-2.5507	-0.6394	0.0231	0.6421	2.8795

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.04934	0.04521	-1.091	0.276
lag(y, 1)	1.58595	0.02972	53.360	<2e-16 ***
lag(y, 2)	-0.75056	0.02968	-25.288	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Hannan-Rissanen Algorithm

1. Estimate a high order AR (remember $\text{AR} \Leftrightarrow \text{MA}$ when stationary + invertible)
2. Use AR to estimate values for unobserved w_t via `lm` with `lags`
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,
5. Goto 3, repeat until convergence

Hannan-Rissanen - Step 1 & 2

```
1 (ar = ar.mle(y, order.max = 10))
```

Call:

```
ar.mle(x = y, order.max = 10)
```

Coefficients:

1	2	3	4	5
1.7977	-1.1882	0.2859	-0.0282	-0.0585
6	7			
0.1857	-0.1212			

Order selected 7 sigma^2 estimated as 0.8989

```
1 (ar = forecast::Arima(y, order = c(10,0,0)))
```

Series: y

ARIMA(10,0,0) with non-zero mean

Coefficients:

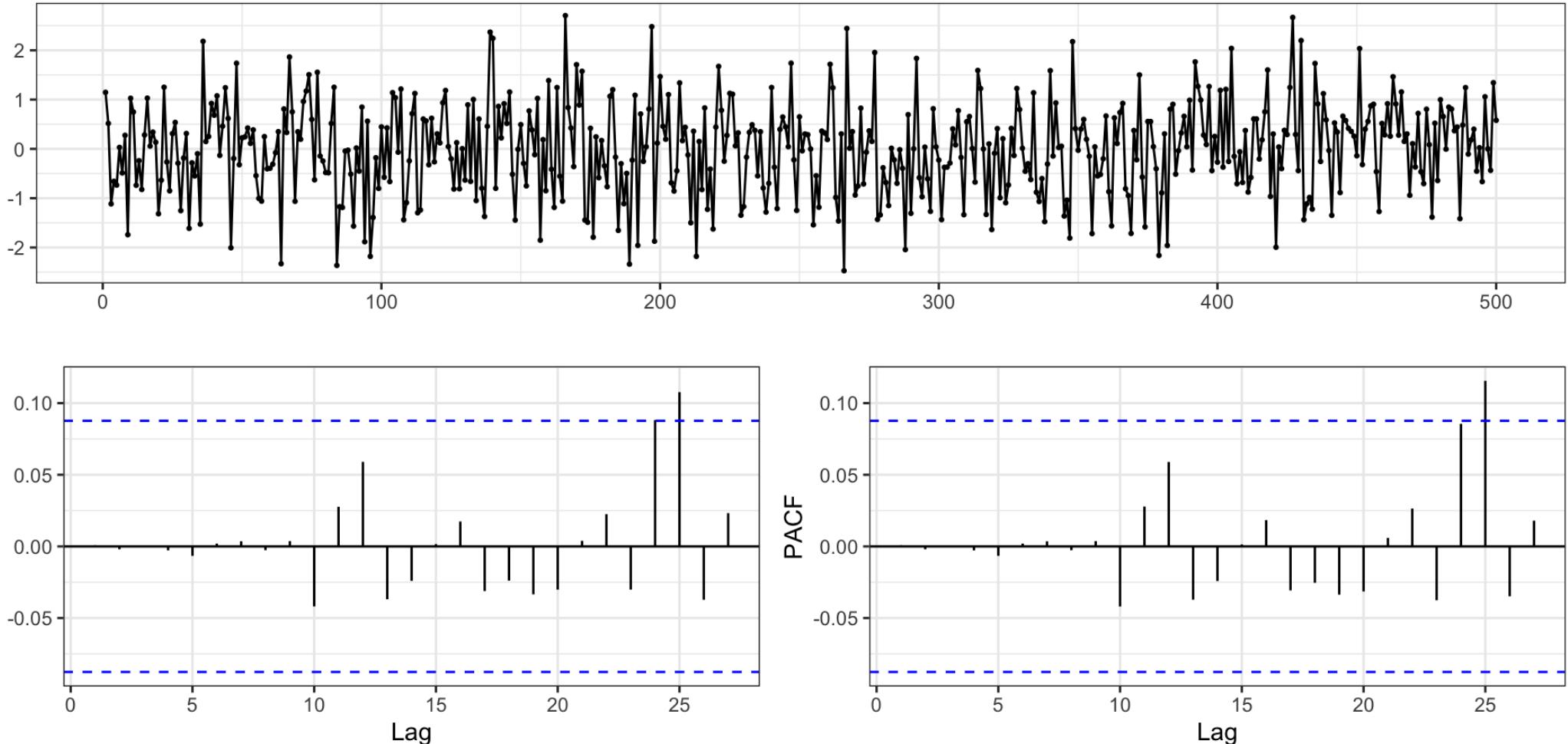
	ar1	ar2	ar3	ar4	ar5
1.7977	1.7940	-1.1824	0.2840	-0.0291	-0.0497
s.e.	0.0447	0.0919	0.1061	0.1069	0.1066
	ar6	ar7	ar8	ar9	ar10
0.1857	0.1492	-0.0657	-0.0298	-0.0028	0.0020
	0.1067	0.1072	0.1068	0.0925	0.0449
	mean				
	-0.2661				
s.e.	0.3251				

sigma^2 = 0.9183: log likelihood = -684.48

ATC=1392.97 ATC α =1393.61 BTC=1443.54

Residuals

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



Hannan-Rissanen - Step 3

```
1 d = tibble(  
2   y = y %>% strip_attr(),  
3   index = seq_along(y),  
4   w_hat1 = ar$resid %>% strip_attr()  
5 )  
6  
7 (lm1 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat1,1) + lag(w_hat1,2), data=d)) %>%  
8   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
-2.60703	-0.64222	0.05338	0.61697	2.65558

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.05660	0.04374	-1.324	Sta 344 Fall 2022

Hannan-Rissanen - Step 4

```
1 d = modelr::add_residuals(d, lm1, "w_hat2")
2
3 (lm2 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat2,1) + lag(w_hat2,2), data=d)) %>%
4   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
-2.56613	-0.62523	0.04716	0.60059	2.75031

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.05595	0.04367	-1.281	0.2007
lag(y, 1)	1.35622	0.05582	24.296	< 2e-16 ***

Hannan-Rissanen - Step 3.2 + 4.2

```
1 d = modelr::add_residuals(d, lm2, "w_hat3")
2
3 (lm3 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat3,1) + lag(w_hat3,2), data=d)) %>%
4   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
-2.65130	-0.61871	0.05274	0.59498	2.79882

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.05273	0.04369	-1.207	0.228
lag(y, 1)	1.33721	0.05702	23.453	< 2e-16 ***

Hannan-Rissanen - Step 3.3 + 4.3

```
1 d = modelr::add_residuals(d, lm3, "w_hat4")
2
3 (lm4 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat4,1) + lag(w_hat4,2), data=d)) %>%
4   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
-2.63962	-0.61993	0.05707	0.60544	2.76777

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.05192	0.04385	-1.184	0.2370
lag(y, 1)	1.33603	0.05786	23.089	< 2e-16 ***

Hannan-Rissanen - Step 3.4 + 4.4

```
1 d = modelr::add_residuals(d, lm4, "w_hat5")
2
3 (lm5 = lm(y ~ lag(y,1) + lag(y,2) + lag(w_hat5,1) + lag(w_hat5,2), data=d)) %>%
4   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
-2.64145	-0.61486	0.04973	0.60549	2.78622

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.05057	0.04383	-1.154	0.249
lag(y, 1)	1.33475	0.05789	23.056	< 2e-16 ***

BRMS

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

Family: gaussian

Links: mu = identity; sigma = identity

Formula: y ~ arma(p = 2, q = 2) - 1

Data: d (Number of observations: 500)

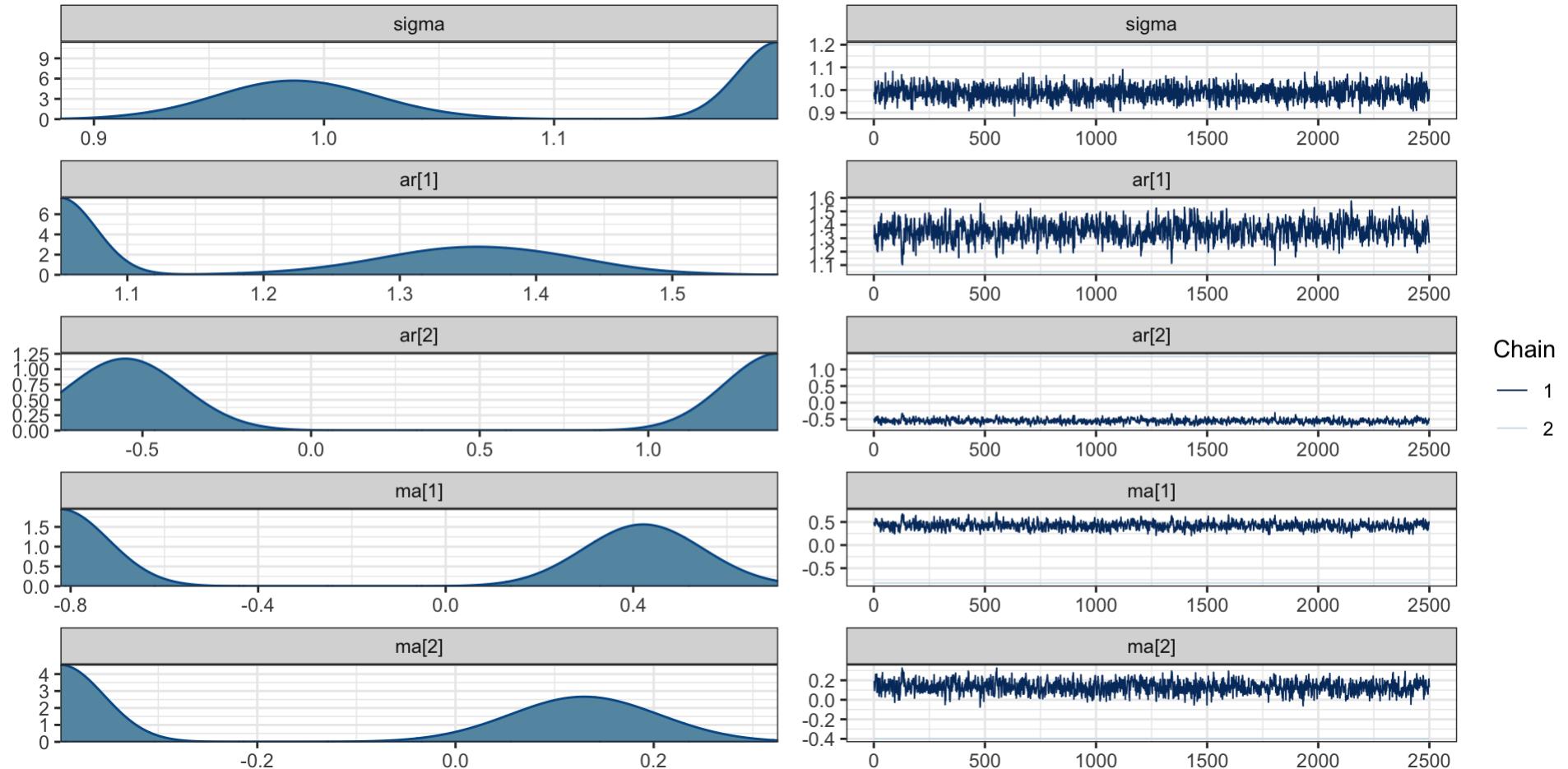
Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
total post-warmup draws = 5000

Correlation Structures:

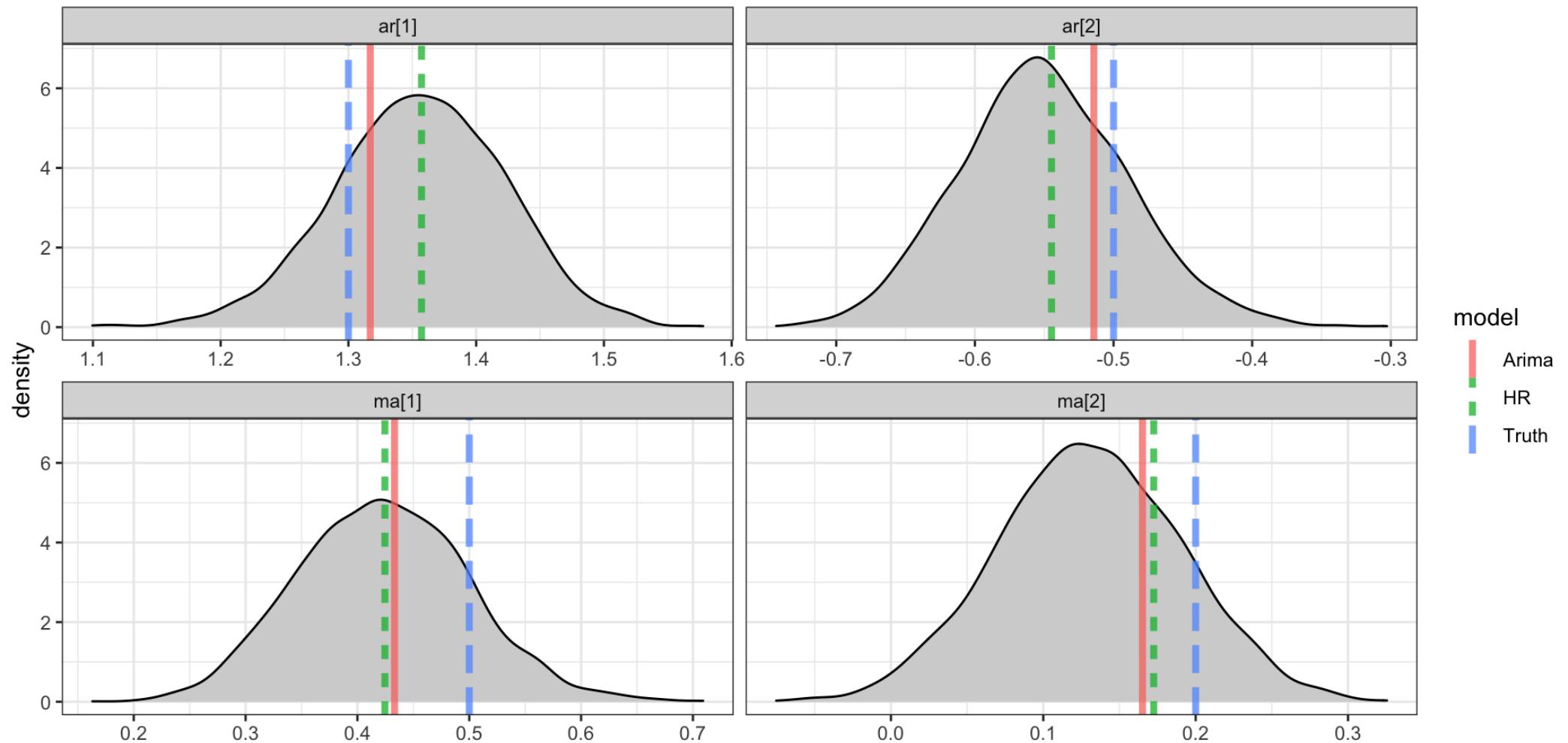
	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
ar[1]	1.20	0.16	1.05	1.46	2.24	3	2
ar[2]	0.42	0.97	-0.65	1.38	2.24	3	NA
ma[1]	-0.20	0.62	-0.82	0.55	2.24	3	2
ma[2]	-0.13	0.27	-0.40	0.23	2.24	3	2

Chains

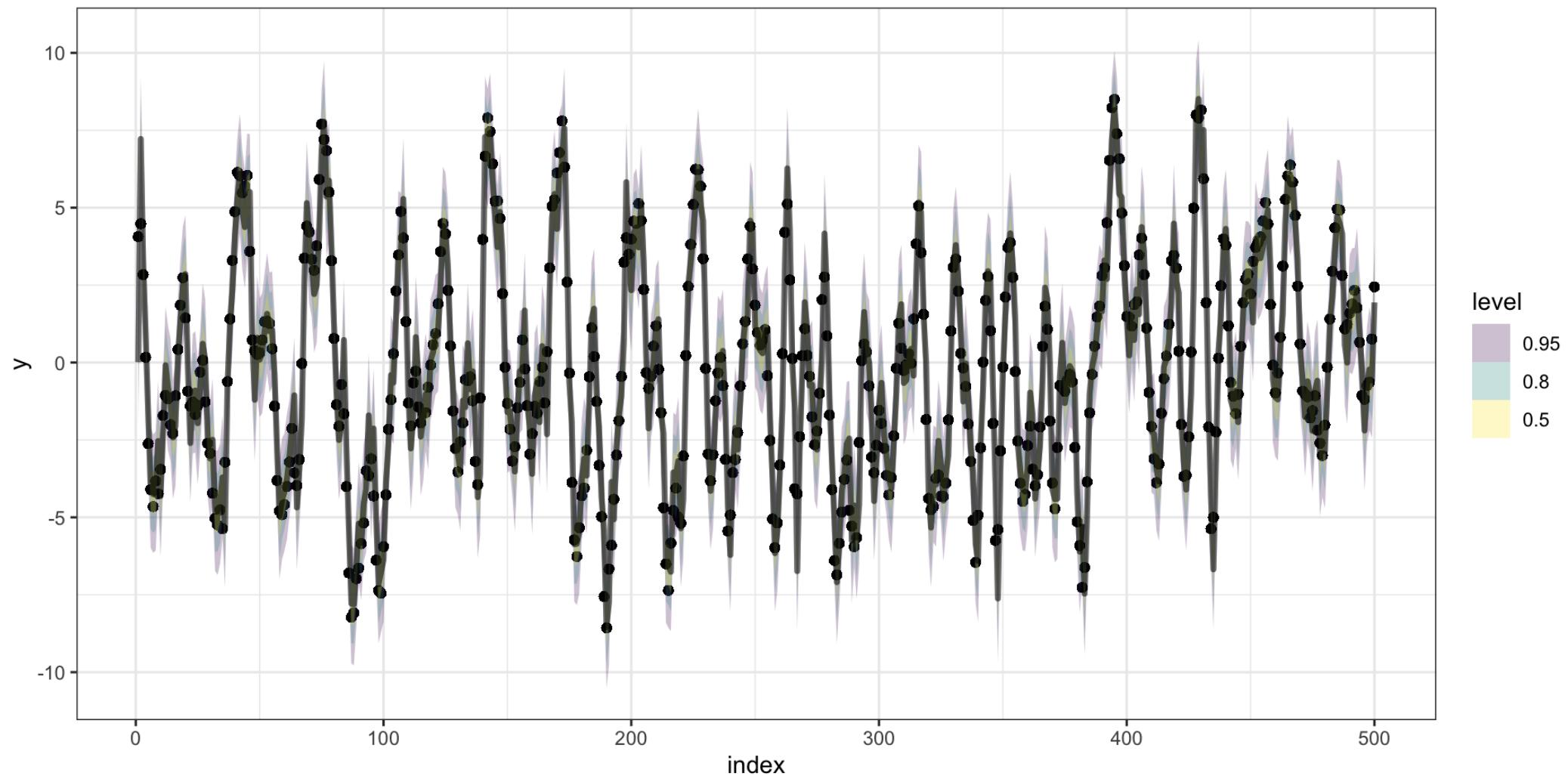
```
1 plot(arma22_brms)
```



Comparison



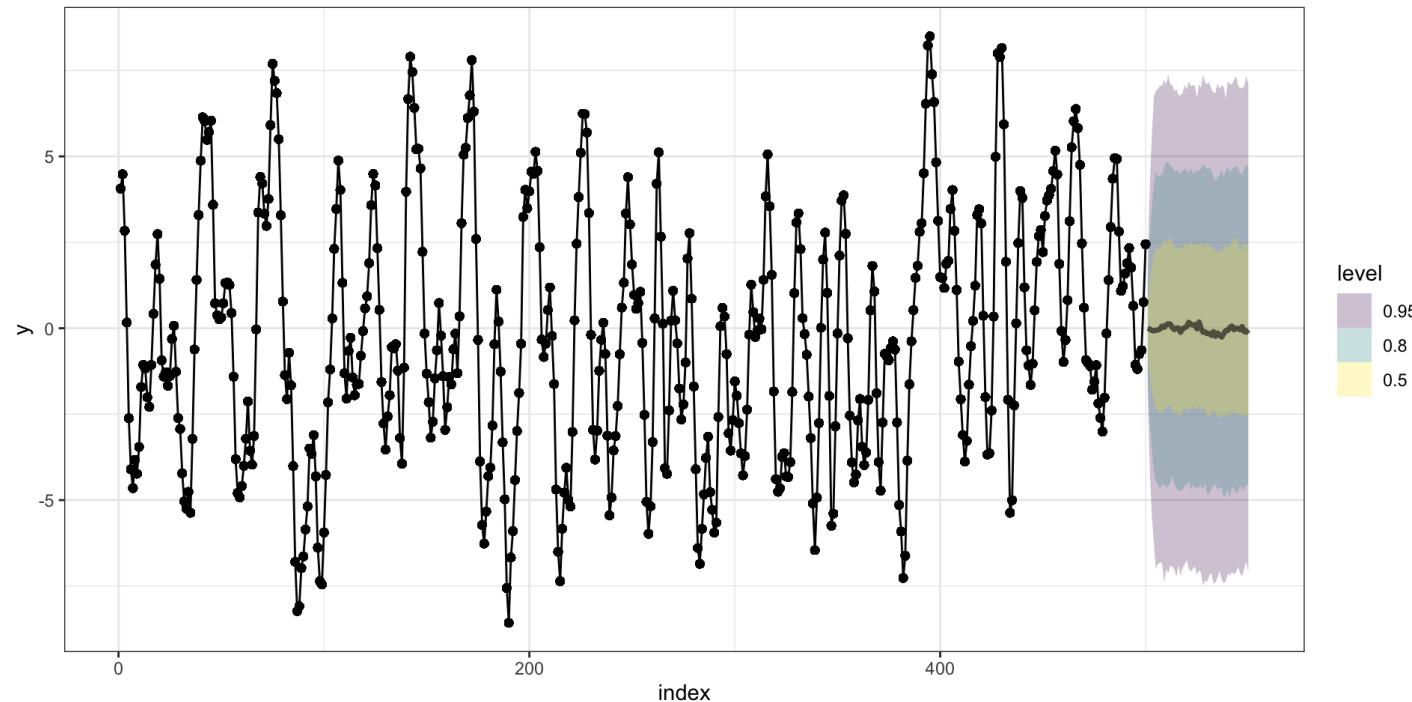
Predictions



Forecasting

:::{.small}

```
1 arma22_brms_fc = arma22_brms %>%
2   predicted_draws_fix(
3     newdata = tibble(index=501:550, y=NA)
4   ) %>%
5   filter(.chain == 1)
```



Stan Code

```
1  arma22_brms %>% stancode()

// generated with brms 2.18.0
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
  // 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);
}
```