

# **tidyverts & prophet**

**Lecture 12**

Dr. Colin Rundel

# Tidy time series

# ts objects

In base R, time series are usually encoded using the `ts` S3 class,

```
1 co2
```

	Jan	Feb	Mar	Apr	May	Jun
1959	315.42	316.31	316.50	317.56	318.13	318.00
1960	316.27	316.81	317.42	318.87	319.87	319.43
1961	316.73	317.54	318.38	319.31	320.42	319.61
1962	317.78	318.40	319.53	320.42	320.85	320.45
1963	318.58	318.92	319.70	321.22	322.08	321.31
1964	319.41	320.07	320.74	321.40	322.06	321.73
1965	319.27	320.28	320.73	321.97	322.00	321.71
1966	320.46	321.43	322.23	323.54	323.91	323.59
1967	322.17	322.34	322.88	324.25	324.83	323.93
1968	322.40	322.99	323.73	324.86	325.40	325.20
1969	323.83	324.26	325.47	326.50	327.21	326.54
1970	324.89	325.82	326.77	327.97	327.91	327.50
1971	326.01	326.51	327.01	327.62	328.76	328.40
1972	326.60	327.47	327.58	329.56	329.90	328.92
1973	328.37	329.40	330.14	331.33	332.31	331.90

```
1 typeof(co2)
```

```
[1] "double"
```

```
1 class(co2)
```

```
[1] "ts"
```

```
1 attributes(co2)
```

```
$tsp
```

```
[1] 1959.000 1997.917 12.000
```

```
$class
```

```
[1] "ts"
```

# tidyverts

This is an effort headed by Rob Hyndman (of forecast fame) and others to provide a consistent tidy data based framework for working with time series data and models.

Core packages:

- `tsibble` - temporal data frames and related tools
- `fable` - tidy forecasting (modelling)
- `feasts` - feature extraction and statistics
- `tsibbldata` - sample tsibble data sets

# tsibble

A tsibble is a tibble with additional infrastructure for encoding temporal data - specifically a tsibble is a tidy data frame with an *index* and *key* where

- the *index* is the variable that describes the inherent ordering of the data (from past to present)
- and the *key* is one or more variables that define the unit of observation over time
- each observation should be uniquely identified by the *index* and *key*

# global\_economy

```
1 tsibbledata::global_economy
```

```
# A tsibble: 15,150 x 9 [1Y]
# Key:      Country [263]
  Country Code    Year     GDP Growth     CPI Imports
  <fct>   <fct> <dbl>   <dbl> <dbl> <dbl>   <dbl>
1 Afghan... AFG    1960 5.38e8      NA     NA    7.02
2 Afghan... AFG    1961 5.49e8      NA     NA    8.10
3 Afghan... AFG    1962 5.47e8      NA     NA    9.35
4 Afghan... AFG    1963 7.51e8      NA     NA   16.9
5 Afghan... AFG    1964 8.00e8      NA     NA   18.1
6 Afghan... AFG    1965 1.01e9      NA     NA   21.4
7 Afghan... AFG    1966 1.40e9      NA     NA   18.6
8 Afghan... AFG    1967 1.67e9      NA     NA   14.2
```

# vic\_elec

```
1 tsibbledata::vic_elec

# A tsibble: 52,608 x 5 [30m]
#   <Australia/Melbourne>

  Time           Demand Tempera...¹ Date
  <dttm>        <dbl>    <dbl> <date>
1 2012-01-01 00:00:00  4383.    21.4 2012-01-01
2 2012-01-01 00:30:00  4263.    21.0 2012-01-01
3 2012-01-01 01:00:00  4049.    20.7 2012-01-01
4 2012-01-01 01:30:00  3878.    20.6 2012-01-01
5 2012-01-01 02:00:00  4036.    20.4 2012-01-01
6 2012-01-01 02:30:00  3866.    20.2 2012-01-01
7 2012-01-01 03:00:00  3694.    20.1 2012-01-01
8 2012-01-01 03:30:00  3562.    19.6 2012-01-01
```

# aus\_retail

```
1 tsibbledata::aus_retail

# A tsibble: 64,532 x 5 [1M]
# Key:      State, Industry [152]
#       State     Indus...¹ Serie...² Month Turno...³
#       <chr>     <chr>    <chr>    <mth>    <dbl>
# 1 Australian Cafes,... A33498... 1982 Apr      4.4
# 2 Australian Cafes,... A33498... 1982 May      3.4
# 3 Australian Cafes,... A33498... 1982 Jun      3.6
# 4 Australian Cafes,... A33498... 1982 Jul      4
# 5 Australian Cafes,... A33498... 1982 Aug      3.6
# 6 Australian Cafes,... A33498... 1982 Sep      4.2
# 7 Australian Cafes,... A33498... 1982 Oct      4.8
# 8 Australian Cafes,... A33498... 1982 Nov      5.4
```

# as\_tsibble()

Existing ts objects can be converted to a tsibble easily,

```
1 tsibble::as_tsibble(co2)
```

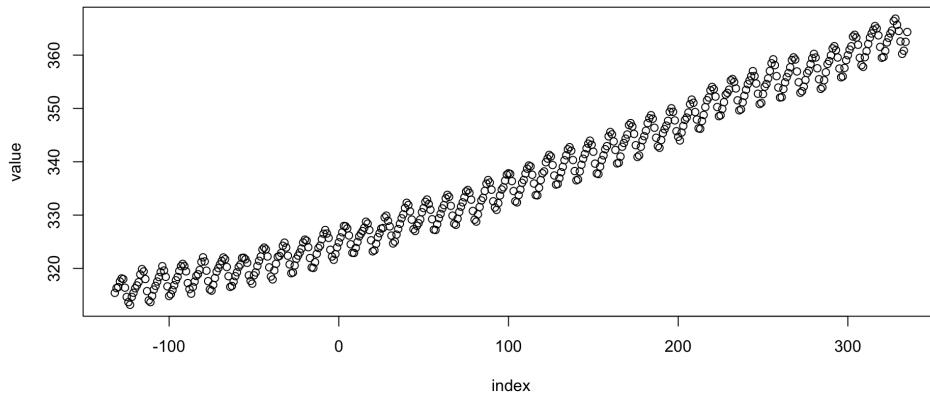
```
# A tsibble: 468 x 2 [1M]
```

	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.

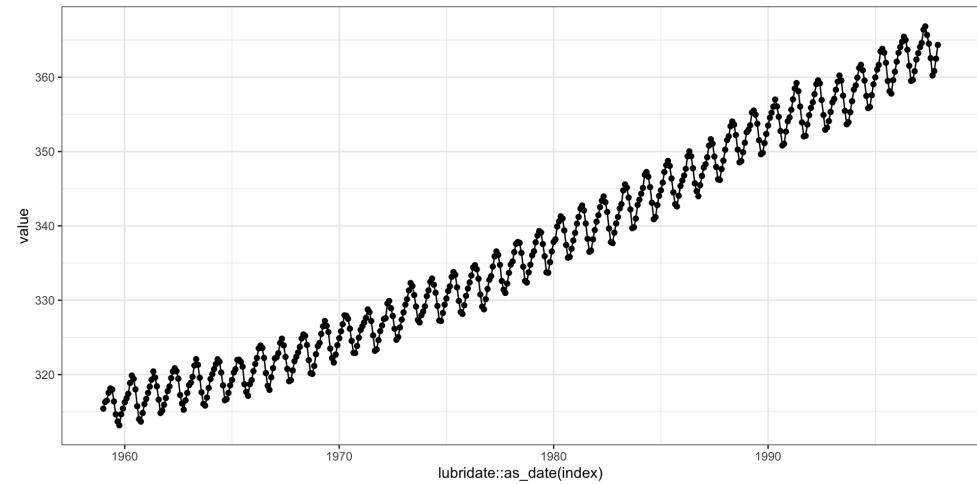
# plotting tsibbles

As the tsibble is basically just a tibble which is basically just a data frame both base and ggplot plotting methods will work.

```
1 tsibble::as_tsibble(co2) %>%  
2   plot()
```

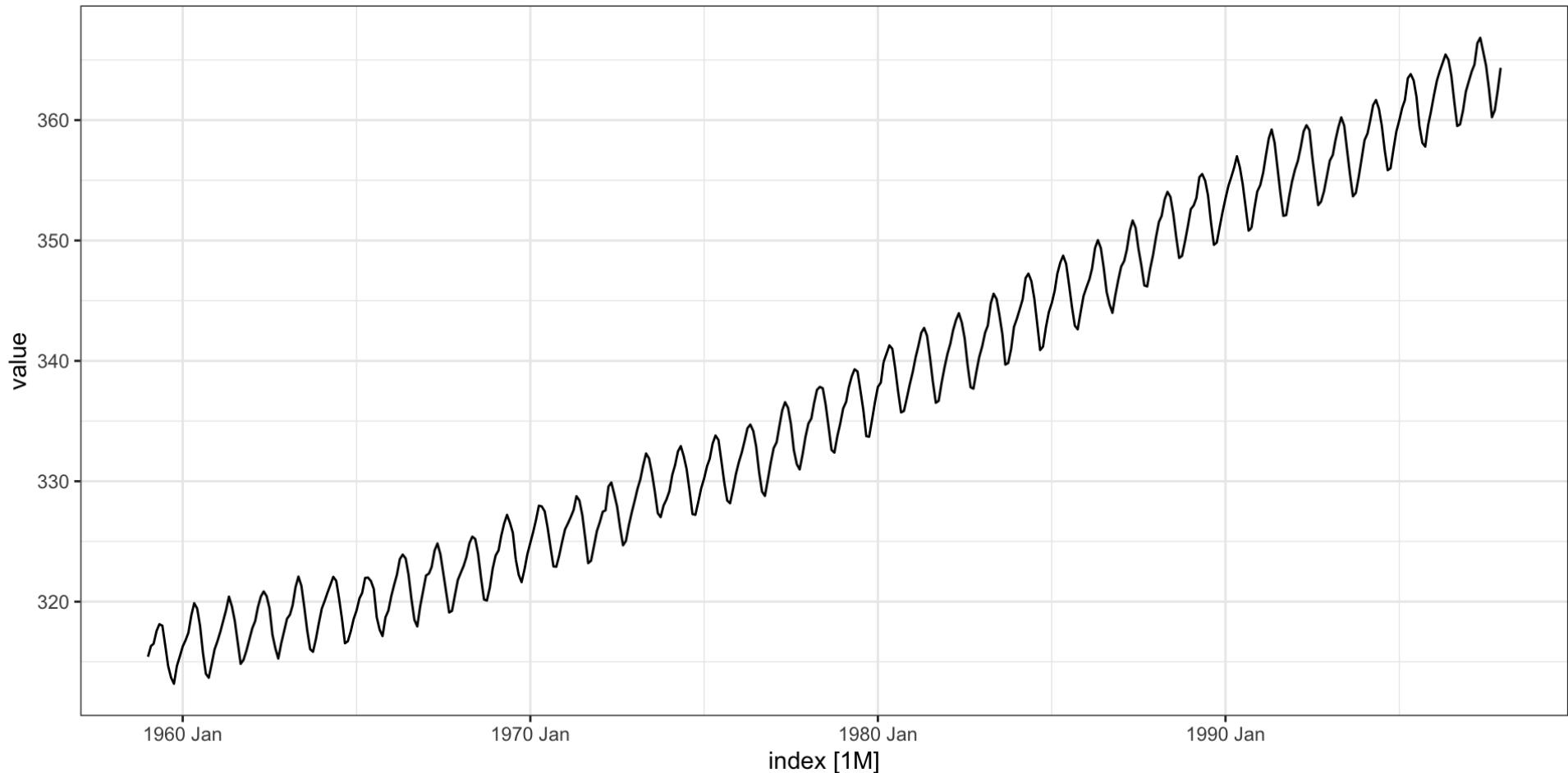


```
1 tsibble::as_tsibble(co2) %>%  
2   ggplot(  
3     aes(x=lubridate::as_date(index), y=value)  
4   ) +  
5     geom_point() +  
6     geom_line()
```



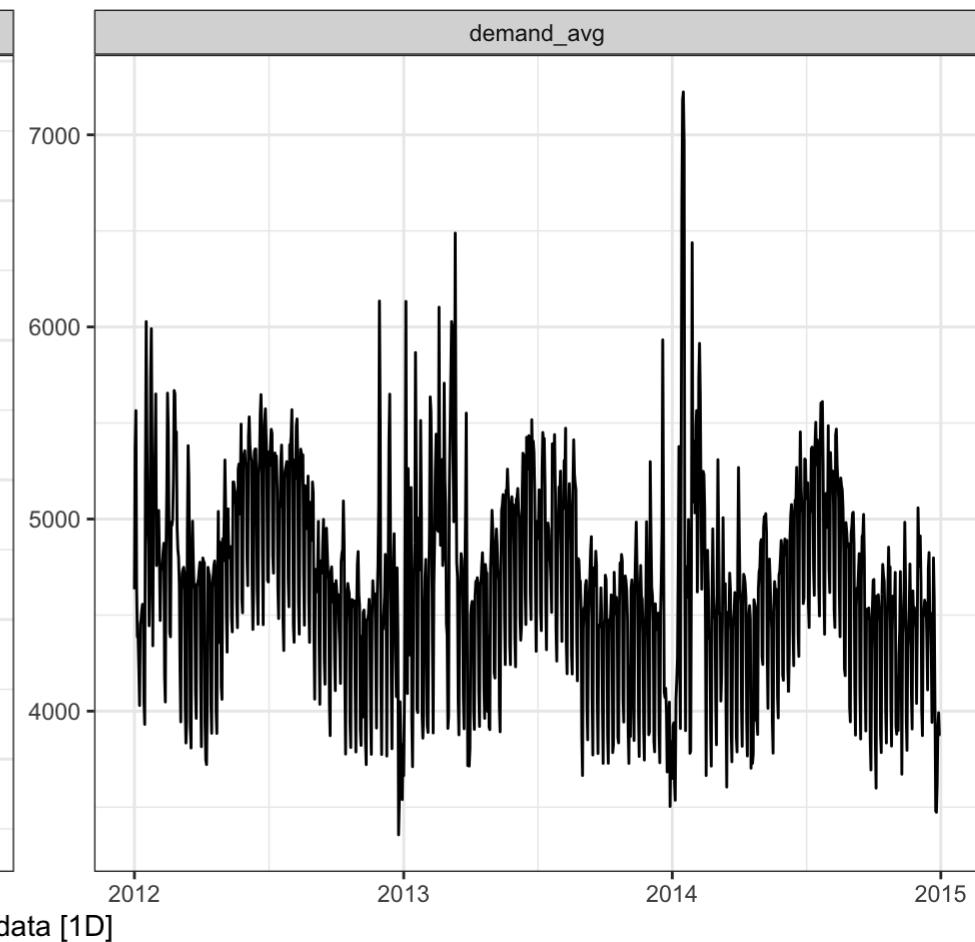
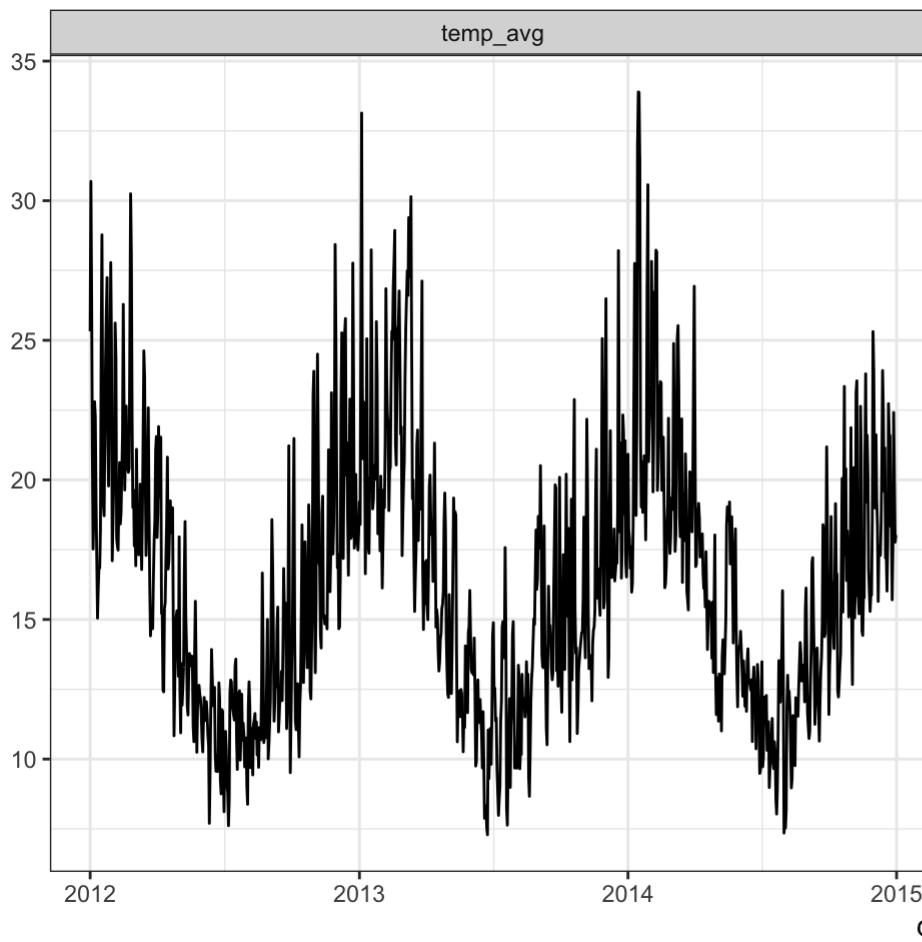
# autoplot

```
1 library(fabletools) # needed to support autoplot  
2 tsibble::as_tsibble(co2) %>%  
3   autoplot(.vars = vars(value))
```



# Multiple variables

```
1 tsibbledata::vic_elec %>%
2   tsibble::index_by(data = ~ lubridate::as_date(.)) %>%
3   summarize(
4     demand_avg = mean(Demand, na.rm=TRUE),
5     temp_avg = mean(Temperature, na.rm=TRUE)
6   ) %>%
7   autoplot(.vars = vars(temp_avg, demand_avg))
```

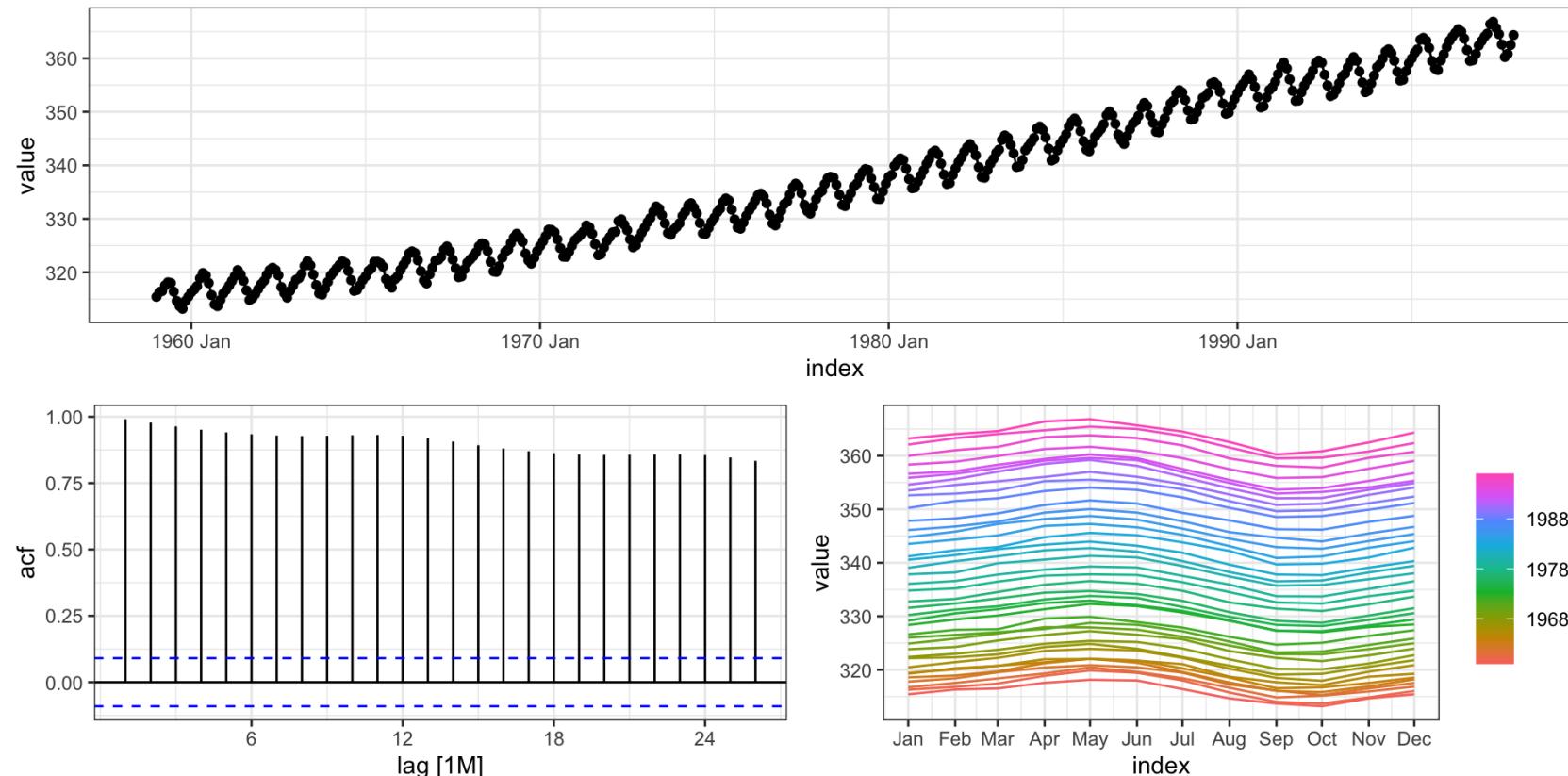


data [1D]

# ggtsdisplay() replacement

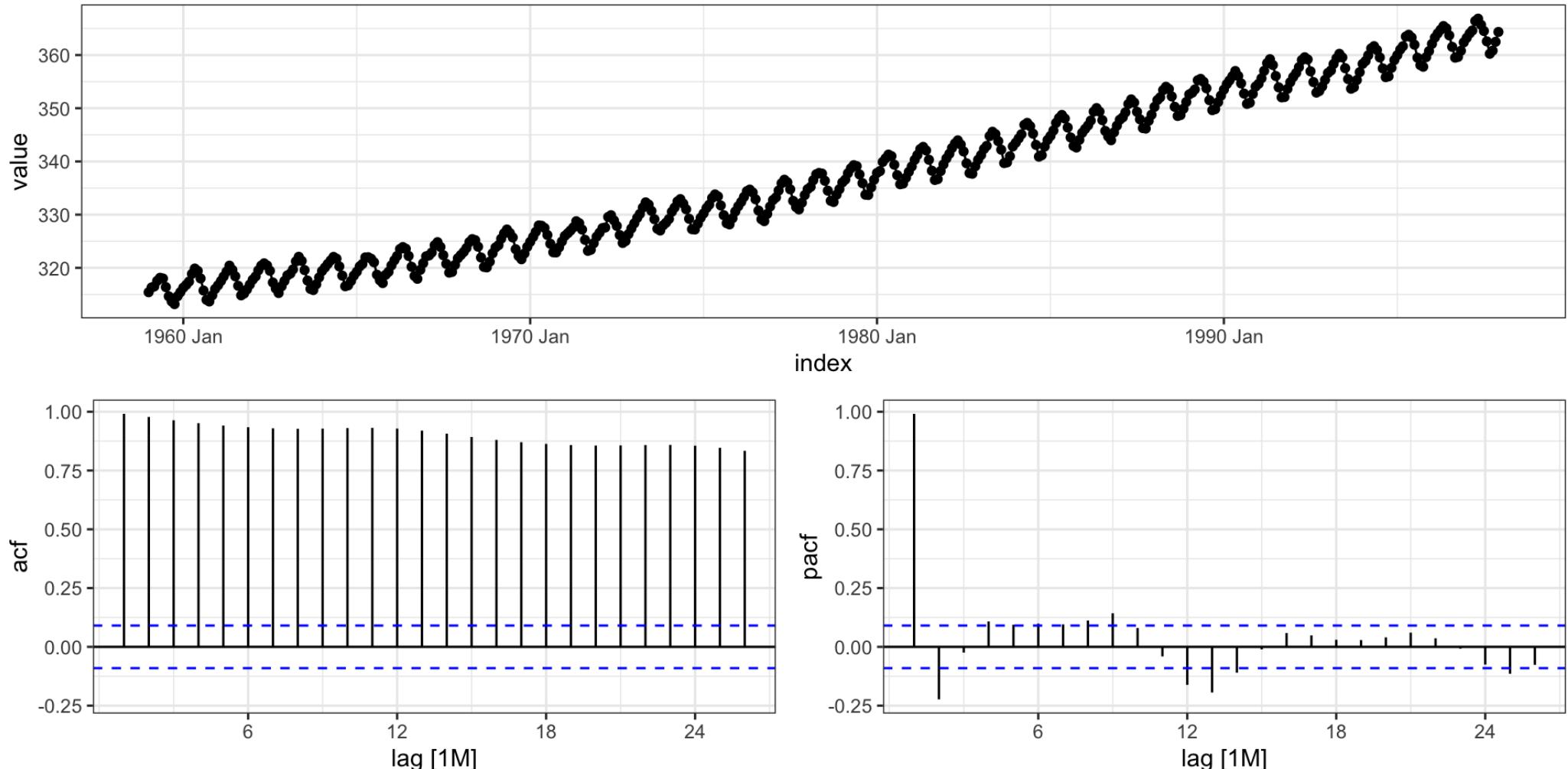
The equivalent to the `ggtsdisplay()` plot is provided by `feasts` with `gg_tsdisplay()`,

```
1 tsibble::as_tsibble(co2) %>%
 2   feasts::gg_tsdisplay(y = value)
```



# ggtsdisplay() - pACF

```
1 tsibble::as_tsibble(co2) %>%
2   feasts::gg_tsdisplay(y = value, plot_type = "partial")
```



# Modeling with fable

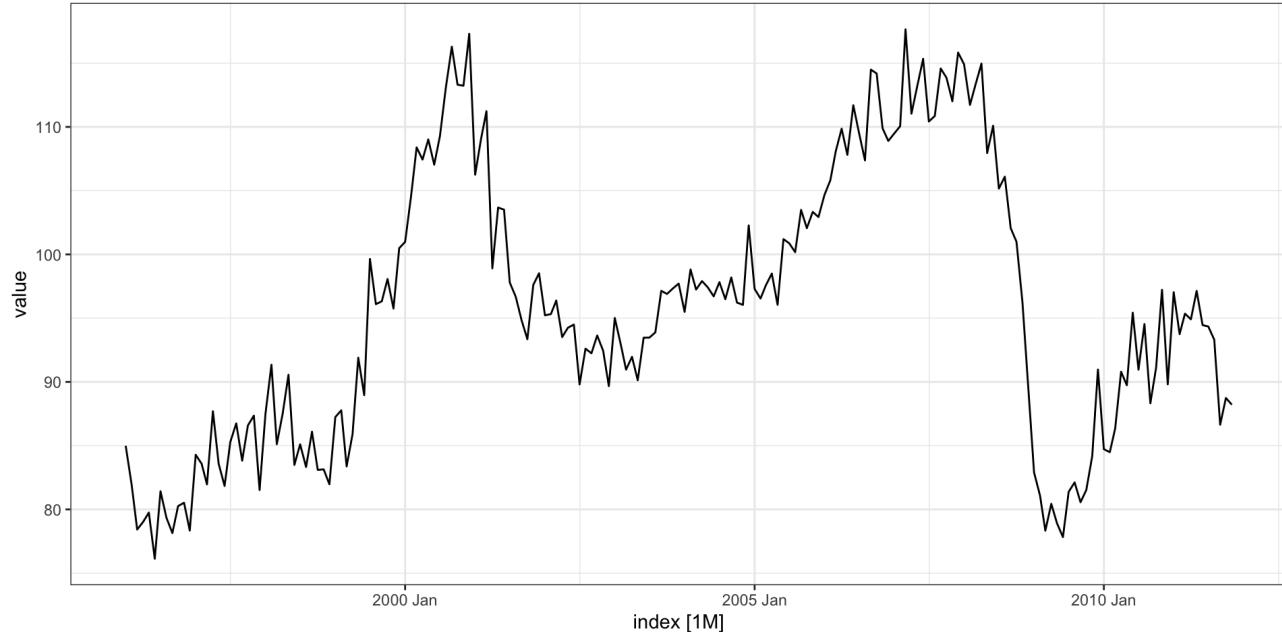
# elec\_sales

```
1 ( elec_sales = as_tsibble(  
2   elec_sales  
3 ) )
```

```
# A tsibble: 191 x 2 [1M]
```

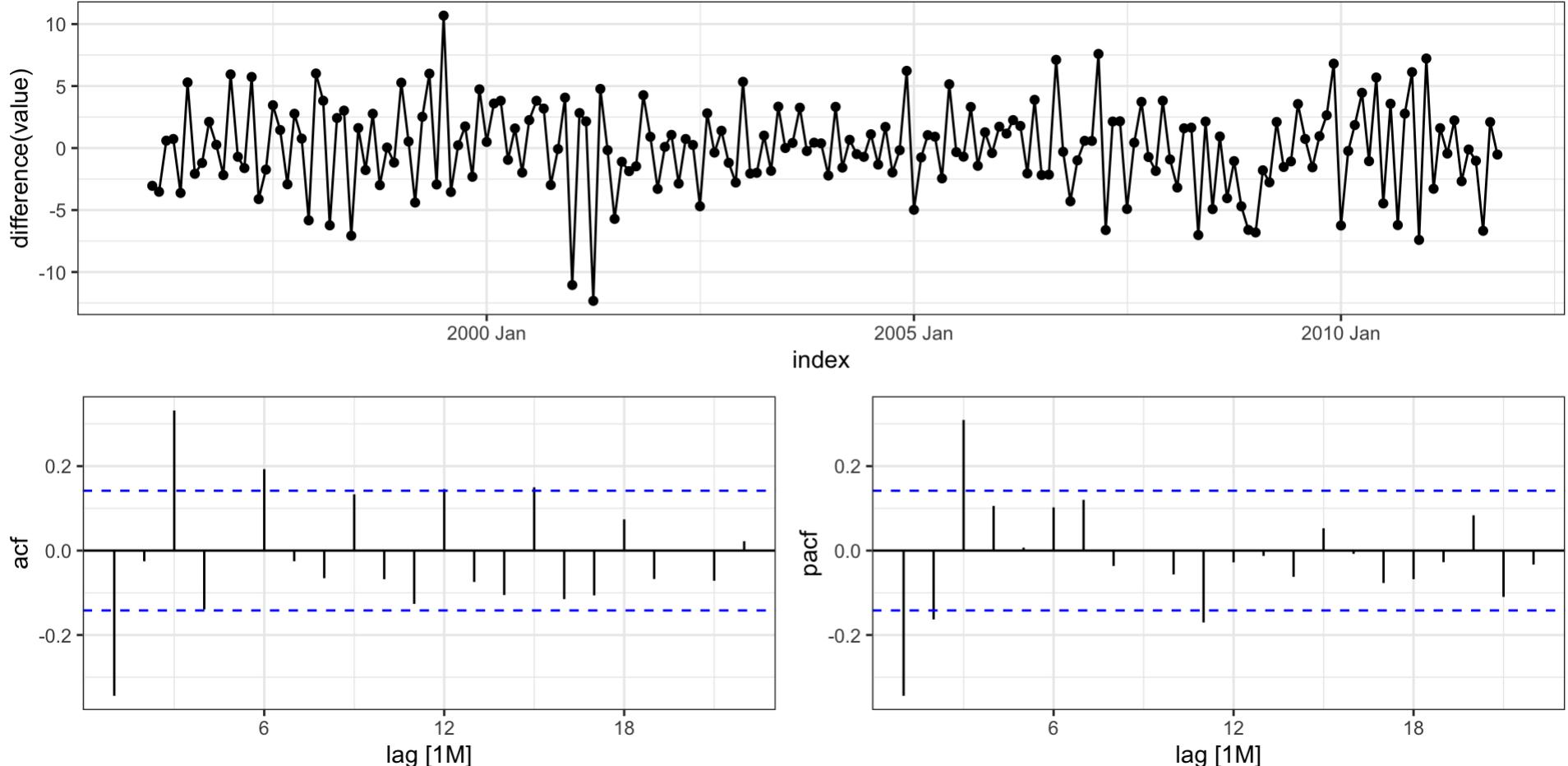
	index	value
1	1996 Jan	85.0
2	1996 Feb	81.9
3	1996 Mar	78.4
4	1996 Apr	79.0
5	1996 May	79.7
6	1996 Jun	76.1
7	1996 Jul	81.4
8	1996 Aug	79.3
9	1996 Sep	78.1
10	1996 Oct	80.3
# ... with 181 more rows		

```
1 elec_sales %>%  
2   autoplot(value)
```



# Differencing

```
1 elec_sales %>%
2   feasts::gg_tsdisplay(difference(value), plot_type="partial")
```



# Modeling

To fit a model with fable we use the `model()` function along with a specific function for the model we are trying to fit (`ARIMA()` here).

As with the rest of tidyverts - `fable` using a tidy approach for modeling which means that the model results are stored in a tibble (called a `mable`).

```
1 library(fable)
2 ( m = elec_sales %>%
3     model(
4         ARIMA(value ~ pdq(3,1,0))
5     )
6 )
```

```
# A mable: 1 x 1
`ARIMA(value ~ pdq(3, 1, 0))` 
<model>
1 <ARIMA(3,1,0)>
```

# Model summary

```
1 m %>%
2   report()
```

Series: value

Model: ARIMA(3,1,0)

Coefficients:

	ar1	ar2	ar3
-	-0.3488	-0.0386	0.3139
s.e.	0.0690	0.0736	0.0694

sigma^2 estimated as 9.853: log likelihood=-485.67

AIC=979.33 AICc=979.55 BIC=992.32

# Model details (broom + yardstick)

```
1 m %>% glance()

# A tibble: 1 × 8
  .model          sigma2 log_lik    AIC   AICc    BIC ar_roots ma_roots
  <chr>        <dbl>    <dbl> <dbl> <dbl> <dbl> <list>    <list>
1 ARIMA(value ~ pdq(3, 1, 0))  9.85 -486.  979.  980.  992. <cpl [3]> <cpl [0]>
```

...

```
1 m %>% accuracy()

# A tibble: 1 × 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(3, 1, 0)) Training 0.0117  3.11  2.43 -0.0435  2.56  0.296  0.281 -0.0346
```

...

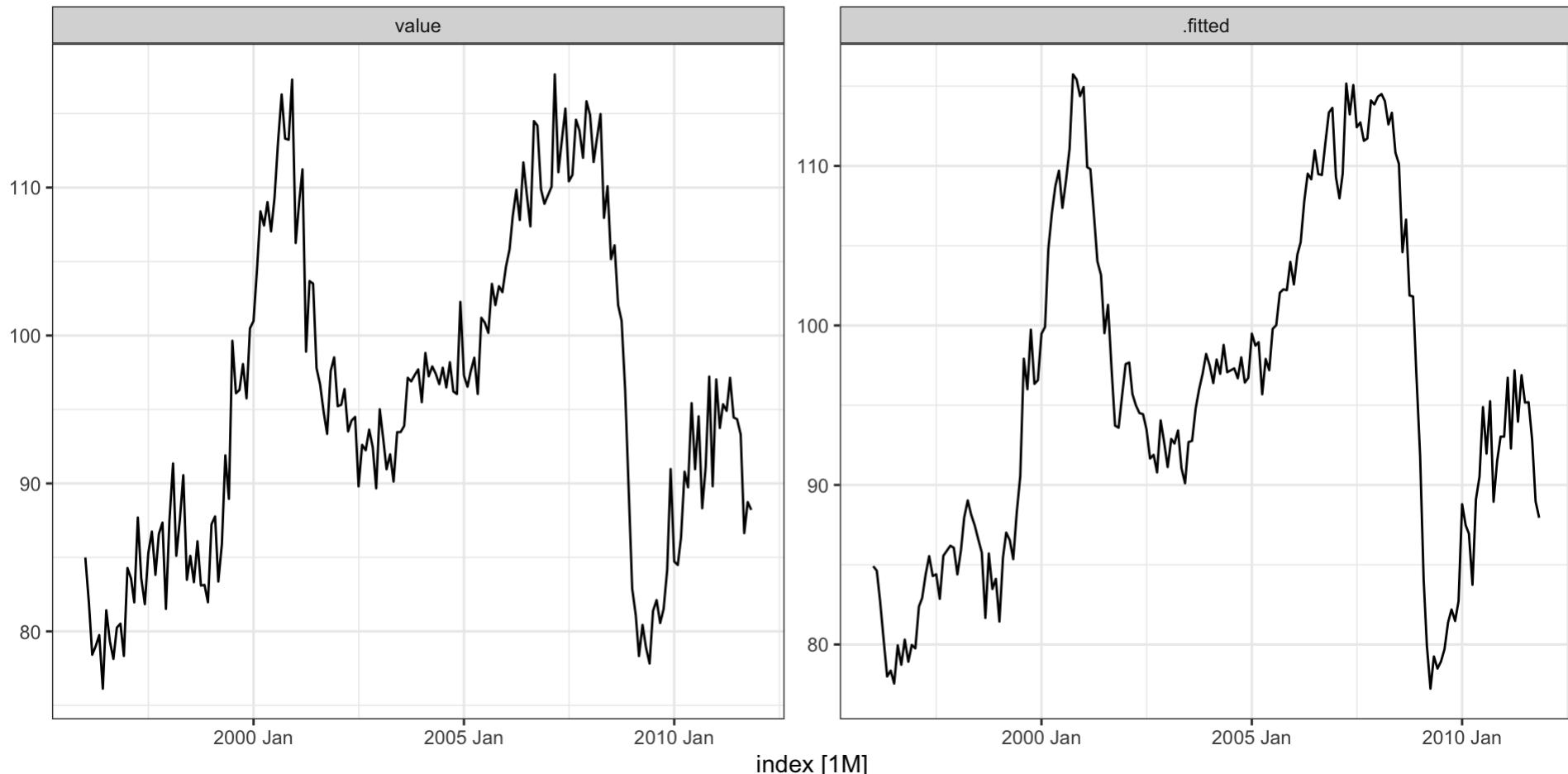
```
1 m %>% augment()

# A tsibble: 191 x 6 [1M]
# Key:   .model [1]
  .model          index value .fitted .resid .innov
  <chr>        <mth> <dbl>    <dbl>    <dbl>    <dbl>
1 ARIMA(value ~ pdq(3, 1, 0)) 1996 Jan  85.0    84.9  0.0850  0.0850
2 ARIMA(value ~ pdq(3, 1, 0)) 1996 Feb  81.9    84.6 -2.68   -2.68
3 ARIMA(value ~ pdq(3, 1, 0)) 1996 Mar  78.4    82.7 -4.28   -4.28
4 ARIMA(value ~ pdq(3, 1, 0)) 1996 Apr  79.0    80.3 -1.25   -1.25
5 ARIMA(value ~ pdq(3, 1, 0)) 1996 May  79.7    78.0  1.76   1.76
6 ARIMA(value ~ pdq(3, 1, 0)) 1996 Jun  76.1    78.4 -2.24   -2.24
7 ARIMA(value ~ pdq(3, 1, 0)) 1996 Jul  81.4    77.5  3.87  3.87
```

```
8 ARIMA(value ~ pdq(3, 1, 0)) 1996 Aug 79.3    79.9 -0.599 -0.599
9 ARIMA(value ~ pdq(3, 1, 0)) 1996 Sep 78.1    78.7 -0.588 -0.588
10 ARIMA(value ~ pdq(3, 1, 0)) 1996 Oct 80.3    80.3 -0.0448 -0.0448
# ... with 181 more rows
```

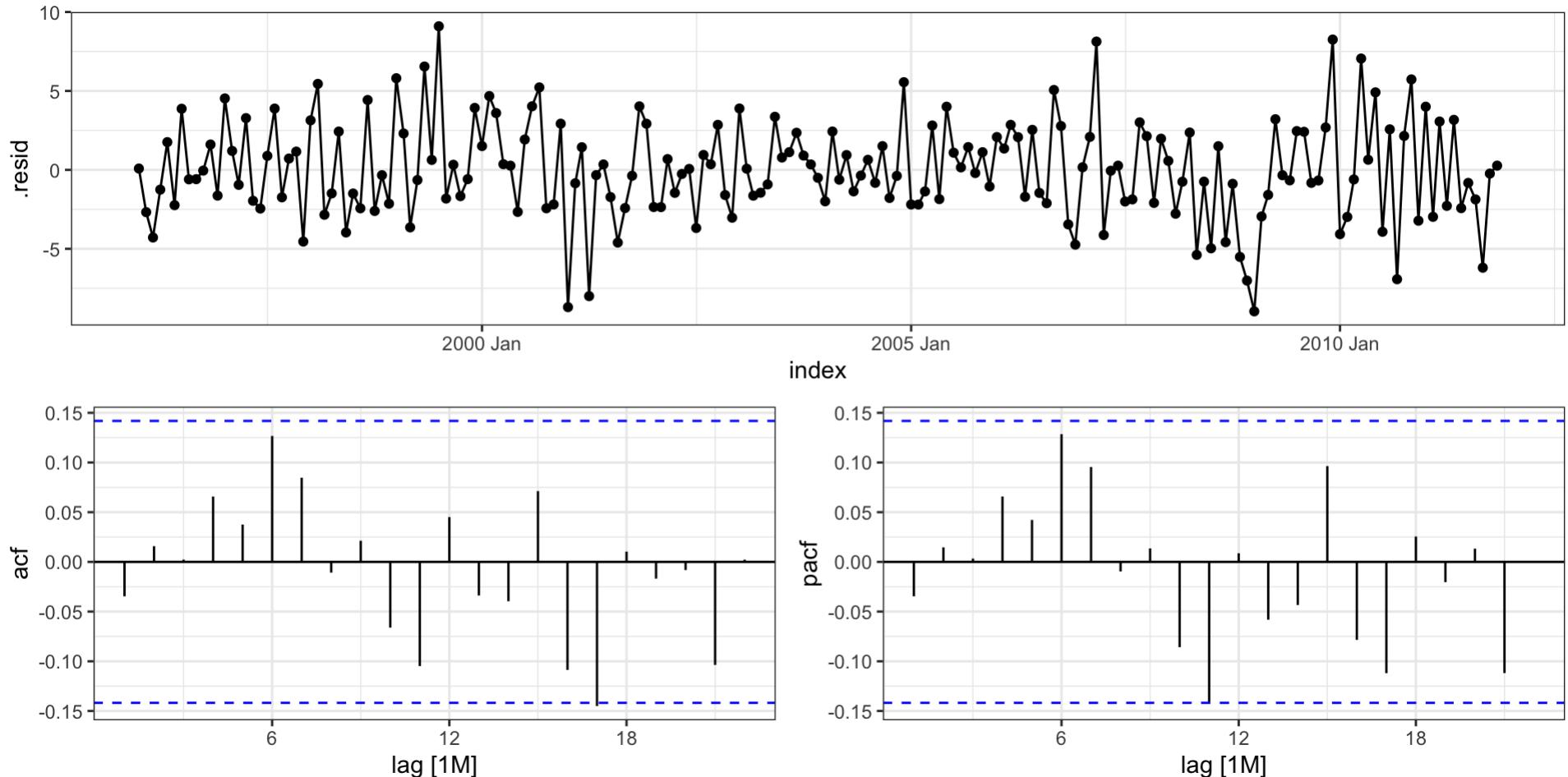
# Observed vs predicted

```
1 m %>%
2   augment() %>%
3   autoplot(vars(value, .fitted))
```



# Residuals

```
1 m %>%
2   augment() %>%
3   feasts::gg_tsdisplay(.resid, plot_type="partial")
```



# Forecasting

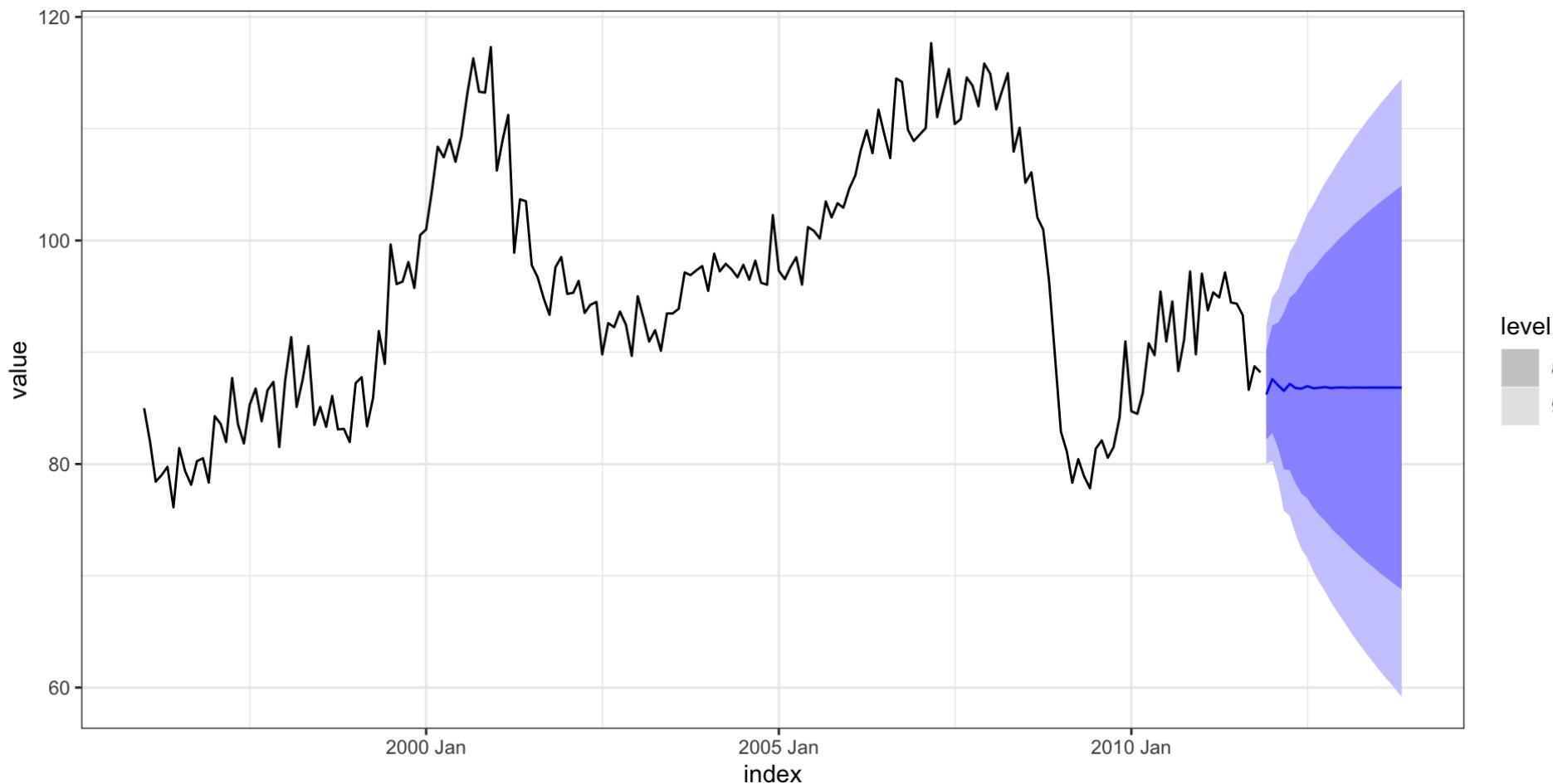
```
1 m %>%
2   forecast()
```

# A fable: 24 x 4 [1M]  
# Key: .model [1]

	.model	index	value	.mean
	<chr>	<mth>	<dist>	<dbl>
1	ARIMA(value ~ pdq(3, 1, 0))	2011 Dec	N(86, 9.9)	86.2
2	ARIMA(value ~ pdq(3, 1, 0))	2012 Jan	N(88, 14)	87.6
3	ARIMA(value ~ pdq(3, 1, 0))	2012 Feb	N(87, 19)	87.0
4	ARIMA(value ~ pdq(3, 1, 0))	2012 Mar	N(87, 30)	86.5
5	ARIMA(value ~ pdq(3, 1, 0))	2012 Apr	N(87, 36)	87.2
6	ARIMA(value ~ pdq(3, 1, 0))	2012 May	N(87, 44)	86.8
7	ARIMA(value ~ pdq(3, 1, 0))	2012 Jun	N(87, 54)	86.7
8	ARIMA(value ~ pdq(3, 1, 0))	2012 Jul	N(87, 62)	87.0

# Forecasting - autoplot

```
1 m %>%
2   forecast() %>%
3   autoplot(elec_sales)
```



# Comparing models

The `fable model()` function also has the ability to fit multiple models at the same time which then makes comparison more straight forward.

```
1 ( mm = elec_sales %>%
2   model(
3     arima310 = ARIMA(value ~ pdq(3,1,0)),
4     arima013 = ARIMA(value ~ pdq(0,1,3)),
5     autoarima = ARIMA(value),
6     autoarima_bf = ARIMA(value, stepwise = FALSE)
7   )
8 )
```

```
# A mable: 1 x 4
      arima310      arima013      autoarima    autoarima_bf
      <model>       <model>       <model>        <model>
1 <ARIMA(3,1,0)> <ARIMA(0,1,3)> <ARIMA(3,1,1)> <ARIMA(1,1,5)>
```

# broom + multiple models

```
1 mm %>% glance() %>% arrange(AICc)
```

# A tibble: 4 × 8

	.model	sigma2	log_lik	AIC	AICc	BIC	ar_roots	ma_roots
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<list>	<list>	
1	autoarima	9.74	-484.	978.	978.	994.	<cpl [3]>	<cpl [1]>
2	autoarima_bf	9.63	-482.	978.	979.	1001.	<cpl [1]>	<cpl [5]>
3	arima310	9.85	-486.	979.	980.	992.	<cpl [3]>	<cpl [0]>
4	arima013	10.2	-489.	986.	987.	999.	<cpl [0]>	<cpl [3]>

```
1 mm %>% accuracy() %>% arrange(RMSE)
```

# 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	autoarima_bf	Training	-0.00719	3.05	2.41	-0.0458	2.55	0.294	0.275	0.00916
2	autoarima	Training	-0.00123	3.08	2.39	-0.0429	2.52	0.291	0.278	0.00893
3	arima310	Training	0.0117	3.11	2.43	-0.0435	2.56	0.296	0.281	-0.0346
4	arima013	Training	0.0105	3.17	2.40	-0.0486	2.53	0.292	0.286	-0.0210

# Forecasting - autoplot

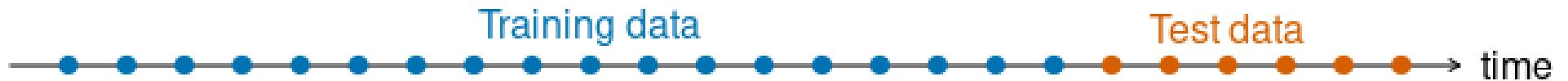
```
1 mm %>%
2   forecast() %>%
3   autoplot(elec_sales)
```



# Cross validation

# 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.



```
1 elec_sales_split = rsample::initial_time_split(elec_sales, prop=0.9)
```

```
1 rsample:::training(elec_sales_split)
```

```
# A tsibble: 171 x 2 [1M]
  index value
  <mth> <dbl>
1 1996 Jan  85.0
2 1996 Feb  81.9
3 1996 Mar  78.4
4 1996 Apr  79.0
5 1996 May  79.7
6 1996 Jun  76.1
7 1996 Jul  81.4
8 1996 Aug  79.3
9 1996 Sep  78.1
10 1996 Oct  80.3
# ... with 161 more rows
```

```
1 rsample:::testing(elec_sales_split)
```

```
# A tsibble: 20 x 2 [1M]
  index value
  <mth> <dbl>
1 2010 Apr  90.8
2 2010 May  89.7
3 2010 Jun  95.4
4 2010 Jul  91.0
5 2010 Aug  94.5
6 2010 Sep  88.3
7 2010 Oct  91.1
8 2010 Nov  97.2
9 2010 Dec  89.8
10 2011 Jan  97.0
11 2011 Feb  93.7
12 2011 Mar  95.4
13 2011 Apr  94.9
```

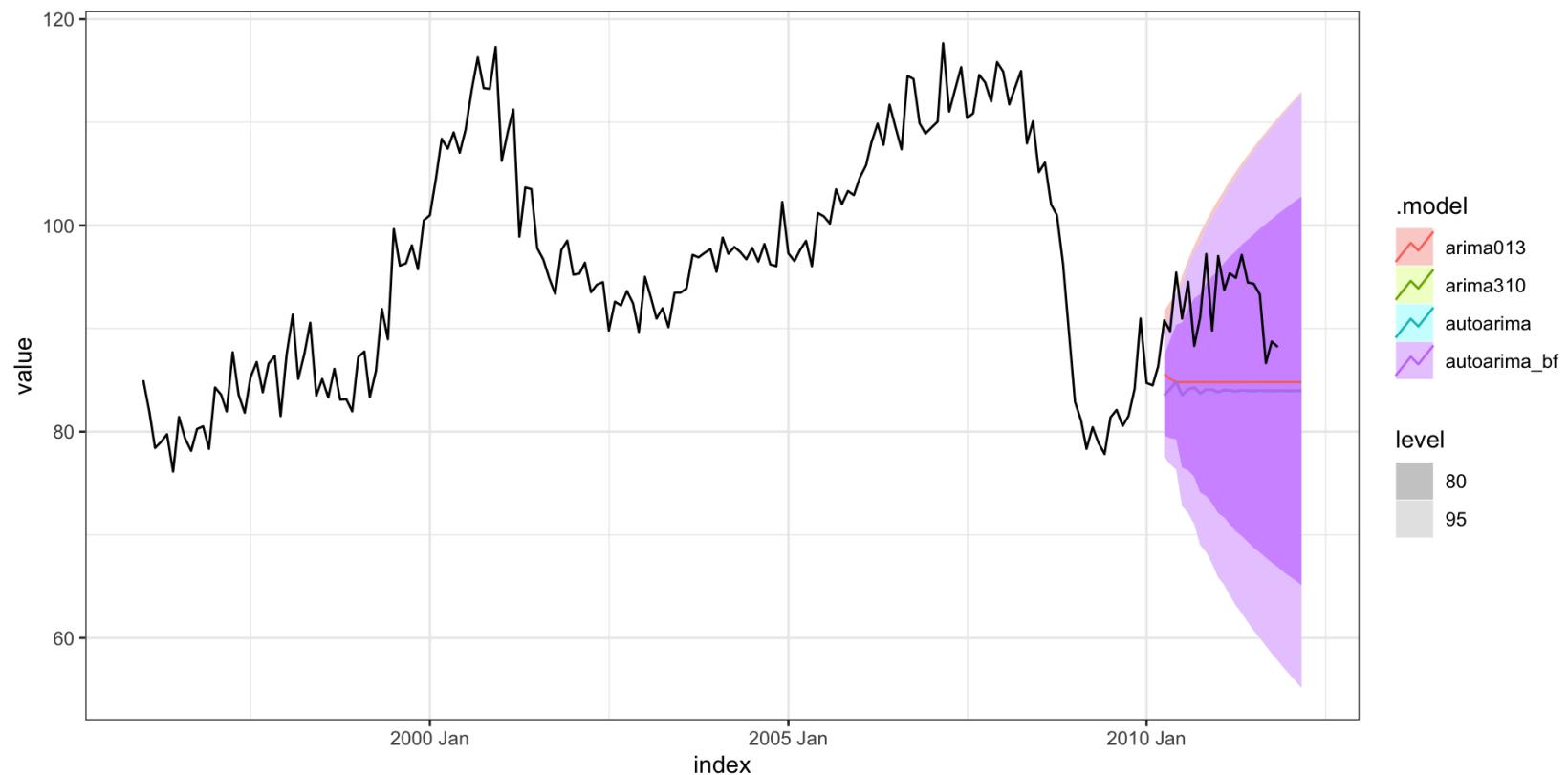
# Model fit (training)

```
1 ( mm = rsample::training(elec_sales_split) %>%
2   model(
3     arima310 = ARIMA(value ~ pdq(3,1,0)),
4     arima013 = ARIMA(value ~ pdq(0,1,3)),
5     autoarima = ARIMA(value),
6     autoarima_bf = ARIMA(value, stepwise = FALSE)
7   )
8 )
```

```
# A mable: 1 x 4
      arima310      arima013      autoarima    autoarima_bf
      <model>       <model>       <model>        <model>
1 <ARIMA(3,1,0)> <ARIMA(0,1,3)> <ARIMA(3,1,0)> <ARIMA(3,1,0)>
```

# Forecasting

```
1 mm %>%
2   forecast() %>%
3   autoplot(
4     elec_sales
5   )
```



# Accuracy

## Out-of-sample:

```
1 mm %>%
2   forecast() %>%
3   accuracy(elec_sales)
```

# A tibble: 4 × 10

	.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
	<chr>	<chr>	<dbl>							
1	arima013	Test	7.73	8.39	7.73	8.24	8.24	0.935	0.742	0.148
2	arima310	Test	8.60	9.17	8.60	9.18	9.18	1.04	0.811	0.162
3	autoarima	Test	8.60	9.17	8.60	9.18	9.18	1.04	0.811	0.162
4	autoarima_bf	Test	8.60	9.17	8.60	9.18	9.18	1.04	0.811	0.162

## Within-sample:

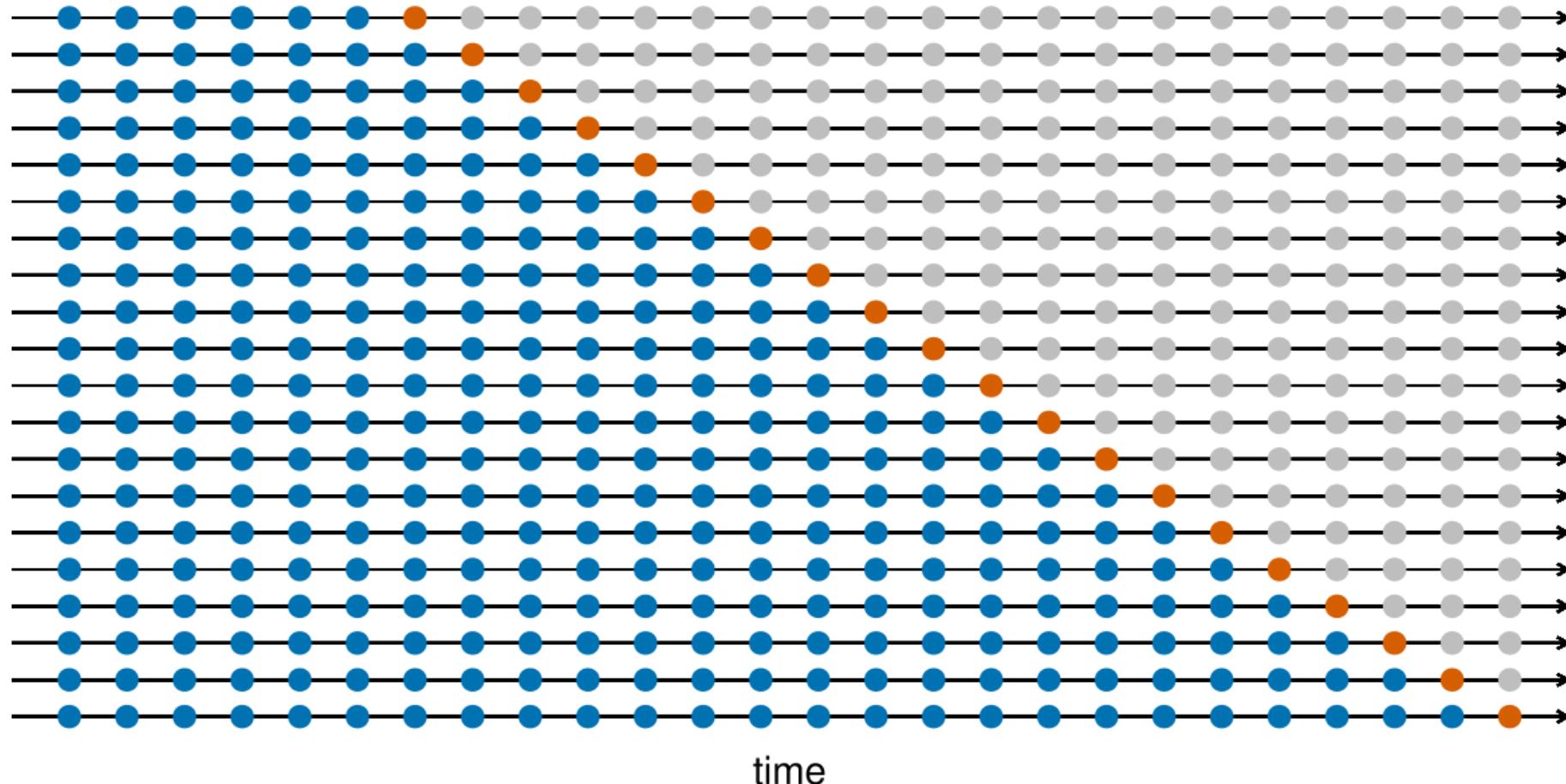
```
1 mm %>%
2   accuracy()
```

# A tibble: 4 × 10

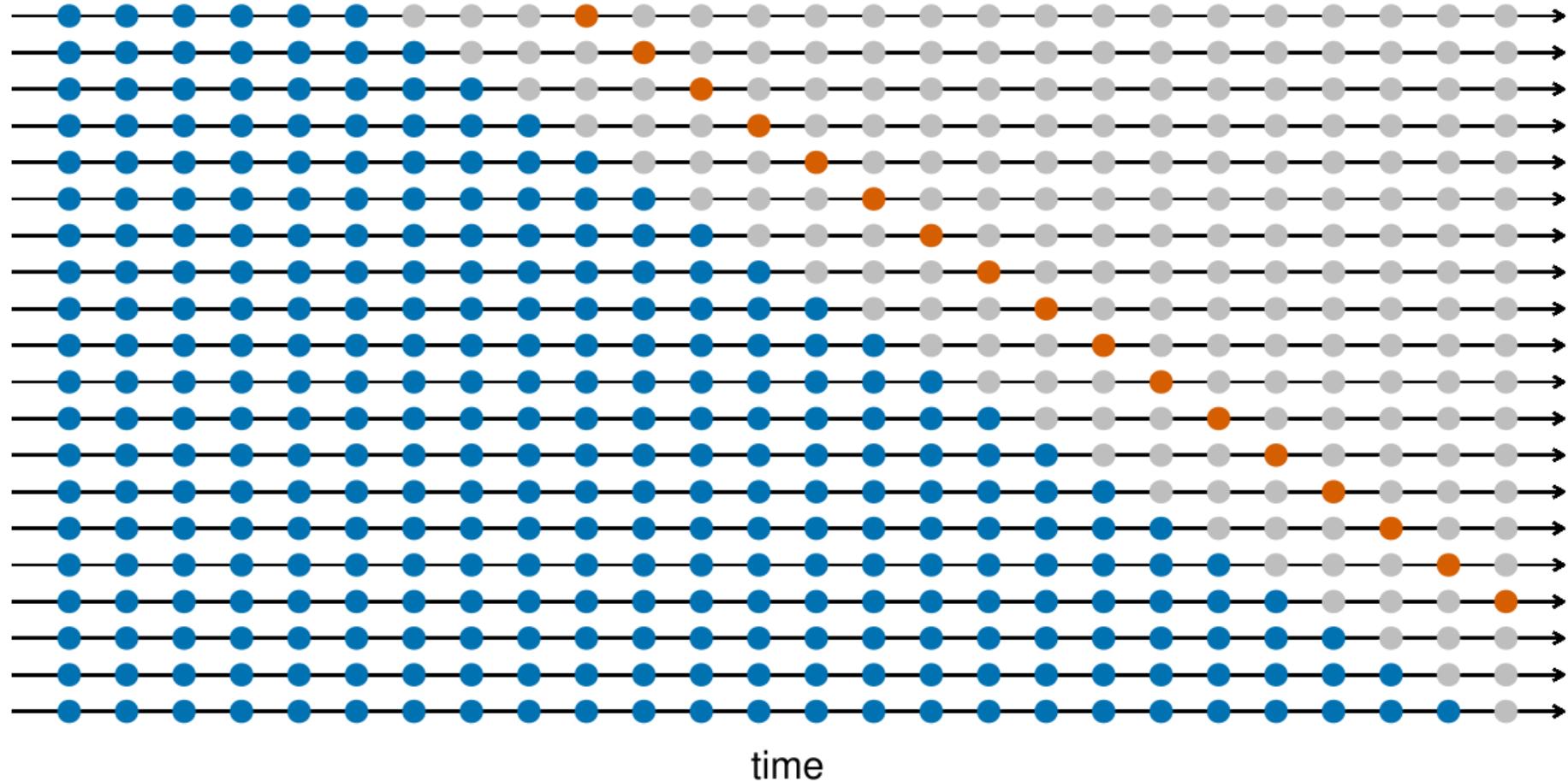
	.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	arima310	Training	-0.00435	3.00	2.31	-0.0535	2.42	0.279	0.265	-0.0317
2	arima013	Training	-0.000151	3.08	2.32	-0.0541	2.44	0.281	0.272	-0.0318
3	autoarima	Training	-0.00435	3.00	2.31	-0.0535	2.42	0.279	0.265	-0.0317
4	autoarima_bf	Training	-0.00435	3.00	2.31	-0.0535	2.42	0.279	0.265	-0.0317

# Rolling forecasting origin

One-step ahead predictive performance



## Four-step ahead predictive performance



# Prophet

# Prophet model

Prophet uses a modeling framework that looks a lot like traditional GAM approaches. Specifically the time series is modeled as

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where

- $g(t)$  is a piecewise linear trend component
- $s(t)$  is a seasonal component based on Fourier terms with specified period(s) and order
- $h(t)$  are holiday effects (specific dummy coded variables for important dates / times)
- $\epsilon_t$  white noise error

# Implementation

Prophet is implemented in its own R package and Python packages ([prophet](#)) which provide all of the basic functionality.

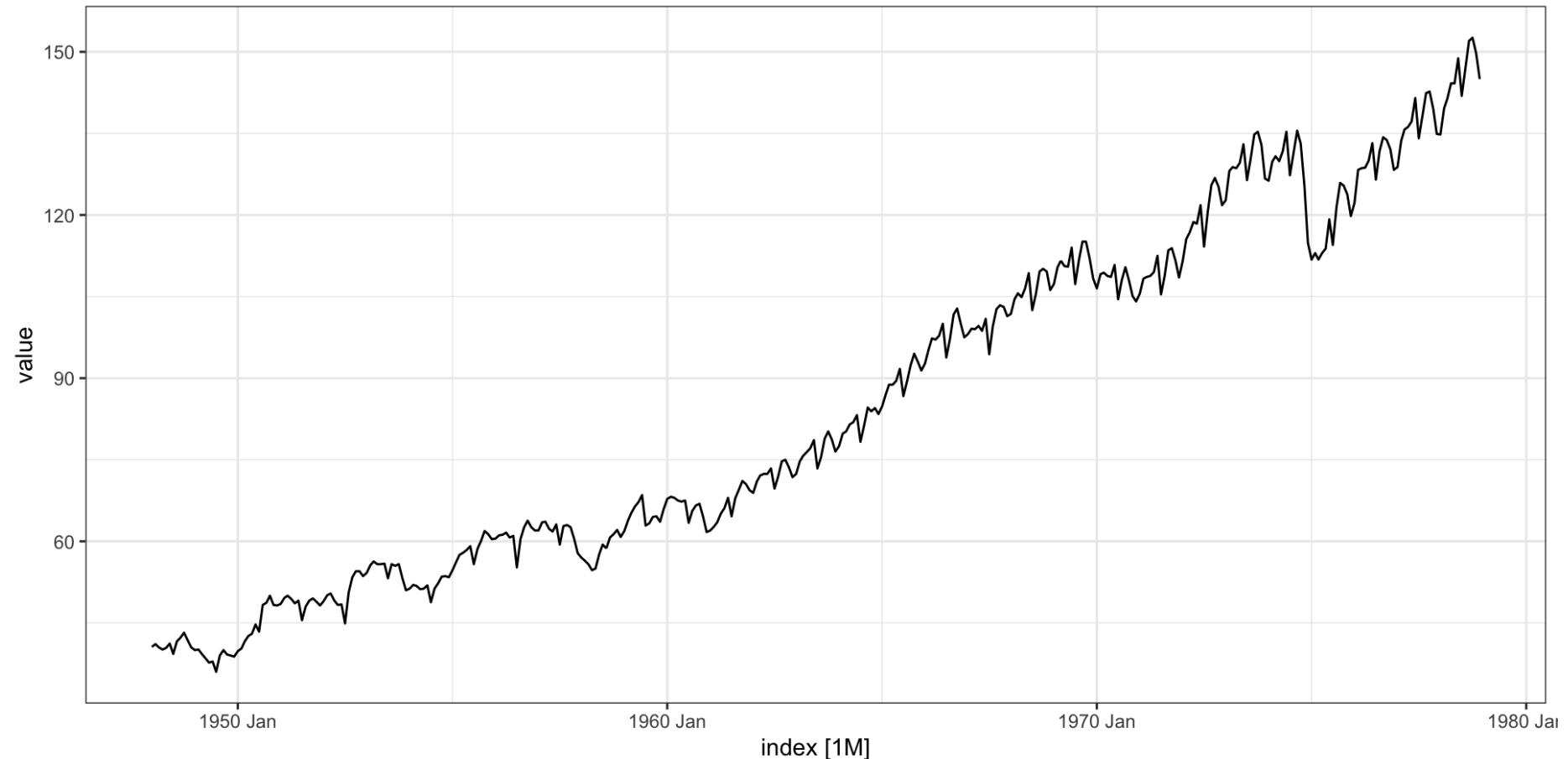
The model fitting is done using a Bayesian approach with the specific implementation using Stan.

Other frameworks like [fable](#) (or [modeltime](#)) provide higher level interfaces to this package.

# prodn from astsa

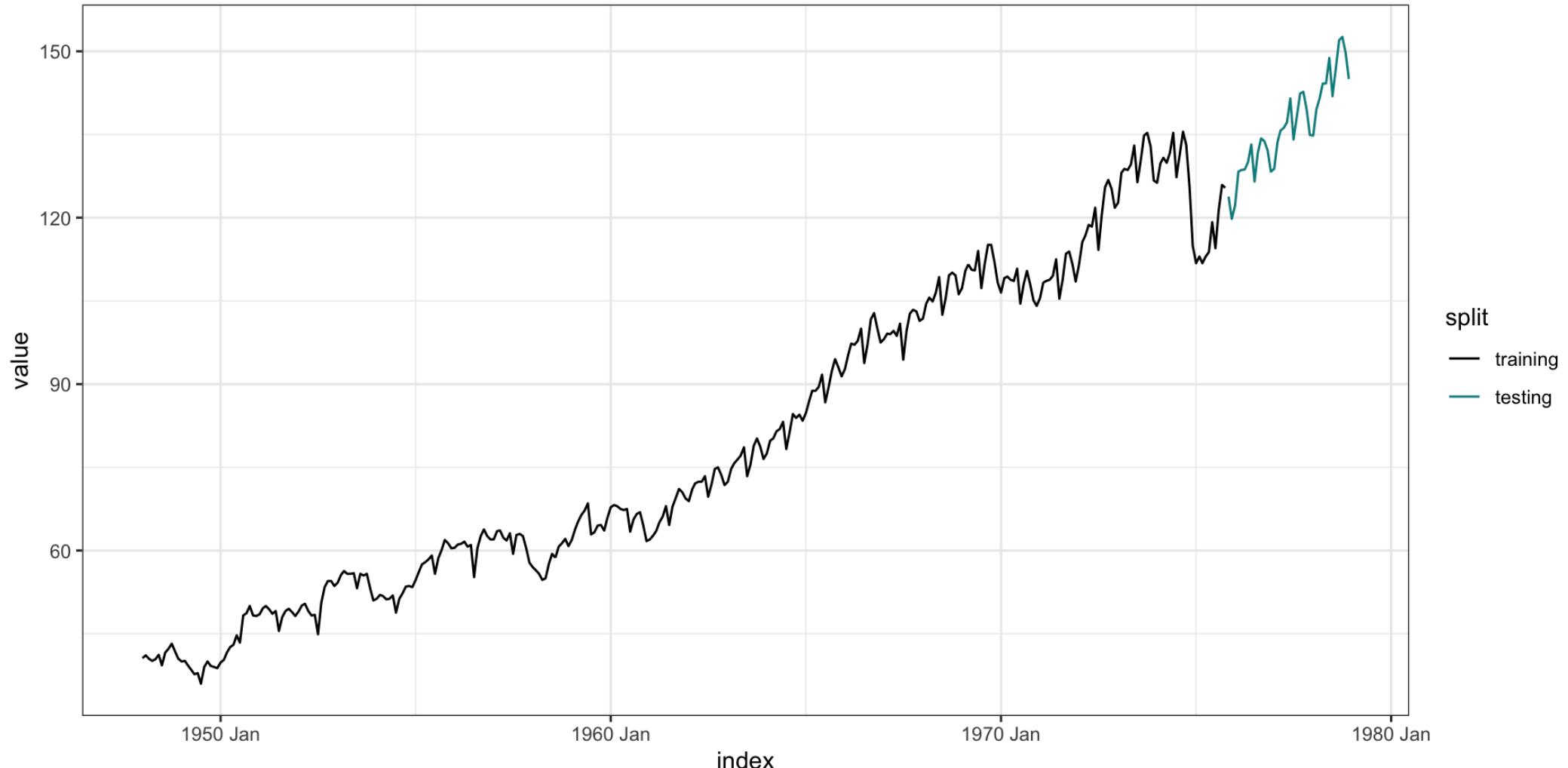
Monthly Federal Reserve Board Production Index (1948-1978)

```
1 prodn = tsibble::as_tsibble(astsa::prodn)
2 autoplot(prodn, value)
```



# Test train split

```
1 prodn_split = rsample::initial_time_split(prodn, prop=0.90)
```



# Models

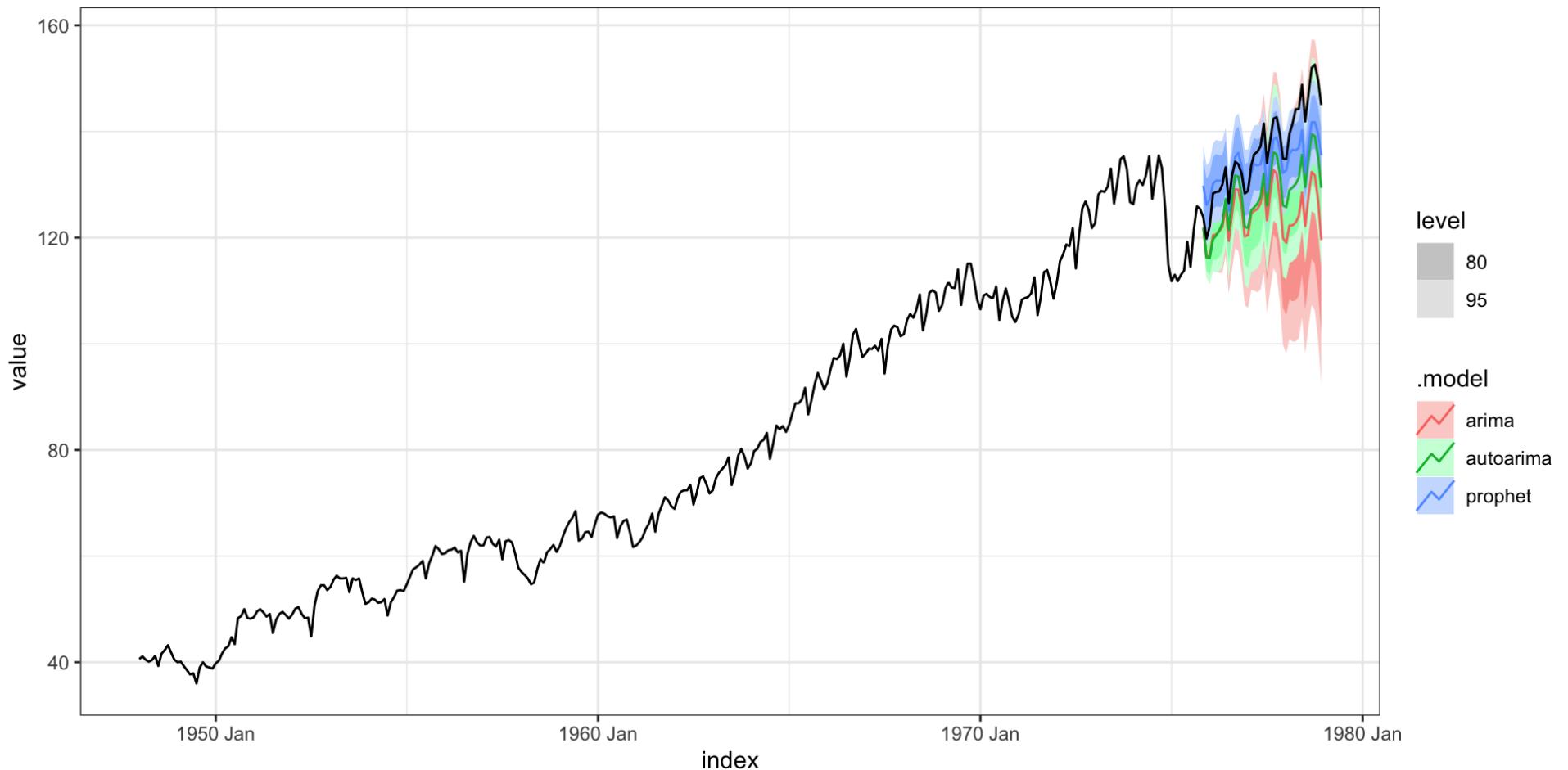
```
1 library(fable.prophet)
2
3 ( prodn_fit = rsample:::training(prodn_split) %>%
4   model(
5     arima = ARIMA(value ~ pdq(1,1,0) + PDQ(0,1,3)),
6     autoarima = ARIMA(value),
7     prophet = prophet(value ~ season(period = 12,
8                               type = "multiplicative")))
9   )
10 )
```

# A mable: 1 x 3

	arima	autoarima	prophet
	<model>	<model>	<model>
1	<ARIMA(1,1,0)(0,1,3)[12]>	<ARIMA(2,0,1)(0,1,1)[12]	w/ drift
			<prophet>

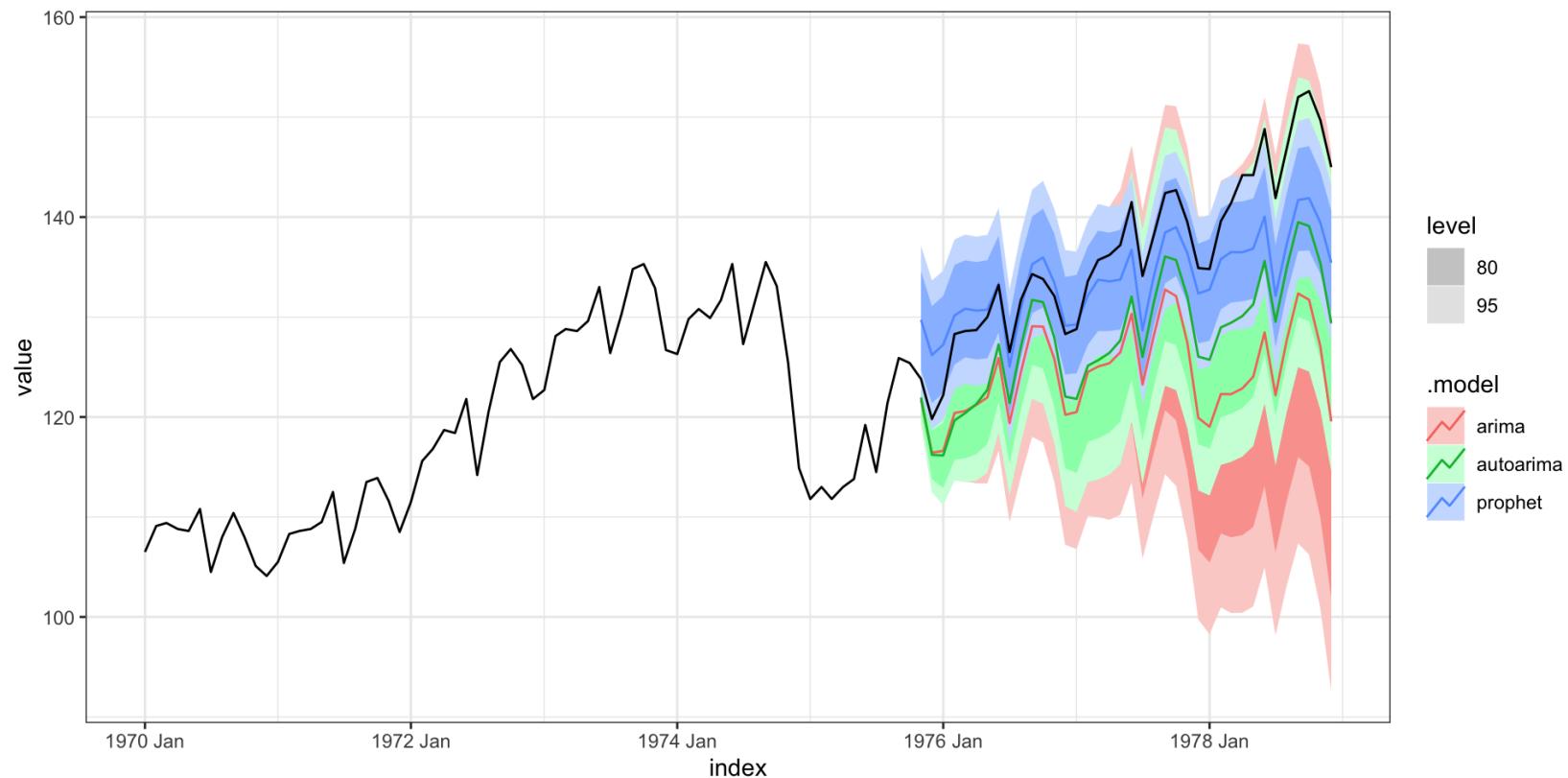
# Forecast

```
1 prodn_fc = forecast(prodn_fit, h=38)
2 prodn_fc %>%
3   autoplot(prodn)
```



# Forecast - zoom

```
1 prodn_fc = forecast(prodn_fit, h=38)
2 prodn_fc %>%
3   autoplot(
4     prodn %>% filter(index >= make_yearmonth(1970L, 1L))
5   )
```



# Accuracy

## Out-of-sample:

```
1 prodn_fit %>%
2   forecast() %>%
3   accuracy(prodn)

# A tibble: 3 × 10
  .model    .type     ME   RMSE    MAE     MPE    MAPE    MASE   RMSSE   ACF1
  <chr>    <chr>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 arima     Test    7.93  8.29  7.93  5.95  5.95  1.56  1.34  0.755
2 autoarima Test    6.52  6.93  6.52  4.91  4.91  1.28  1.12  0.759
3 prophet   Test    0.145 3.22  2.65  0.0162 2.01  0.522 0.521  0.833
```

## Within-sample:

```
1 prodn_fit %>%
2   accuracy()

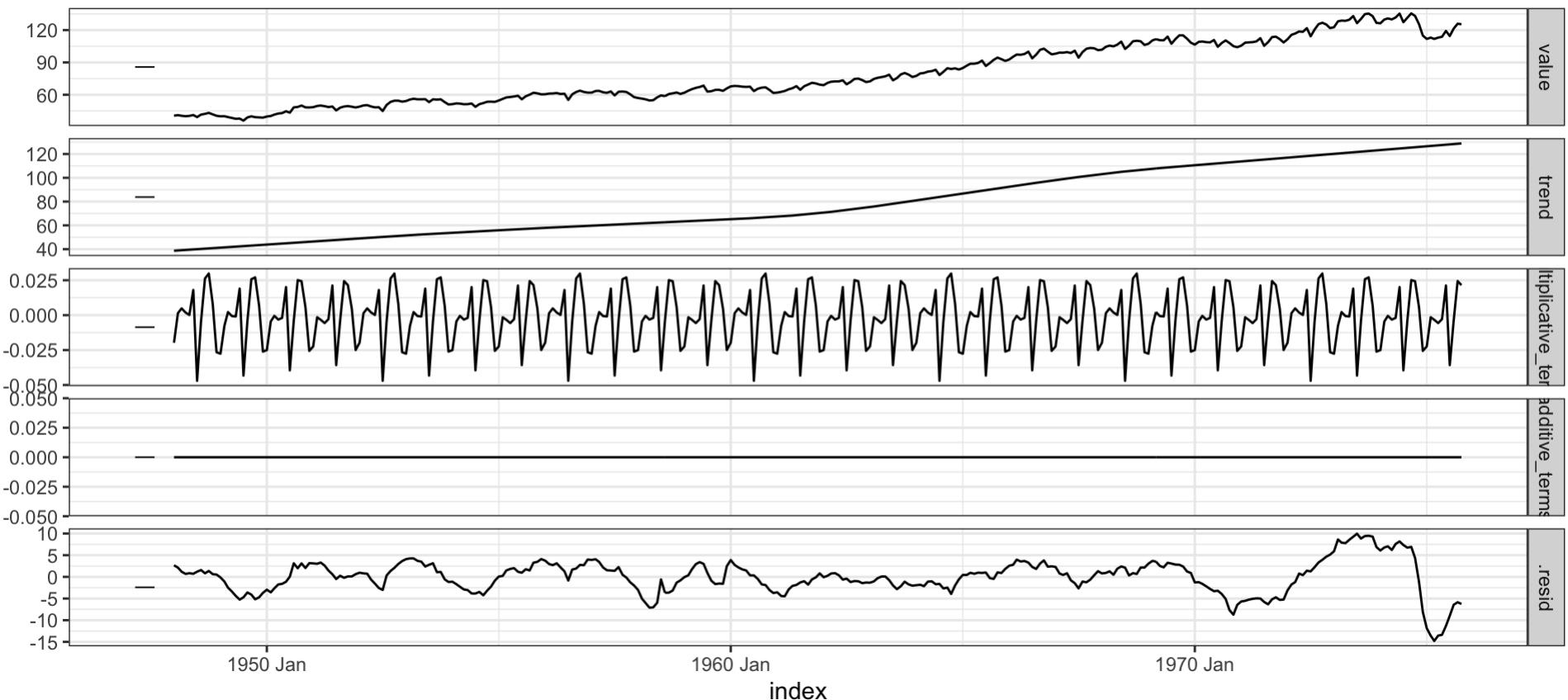
# A tibble: 3 × 10
  .model    .type      ME   RMSE    MAE     MPE    MAPE    MASE   RMSSE   ACF1
  <chr>    <chr>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 arima    Training -0.00242  1.14  0.821  0.0129  1.11  0.162  0.185 -0.0348
2 autoarima Training  0.000504  1.17  0.831 -0.0402  1.11  0.164  0.189  0.00432
3 prophet   Training  0.000883  3.83  2.87  -0.234   3.70  0.565  0.620  0.950
```

# Components

```
1 prodn_fit %>%
2   select(prophet) %>%
3   components() %>%
4   autoplot()
```

Prophet decomposition

value = trend \* (1 + multiplicative\_terms) + additive\_terms + .resid



# Complex seasonality

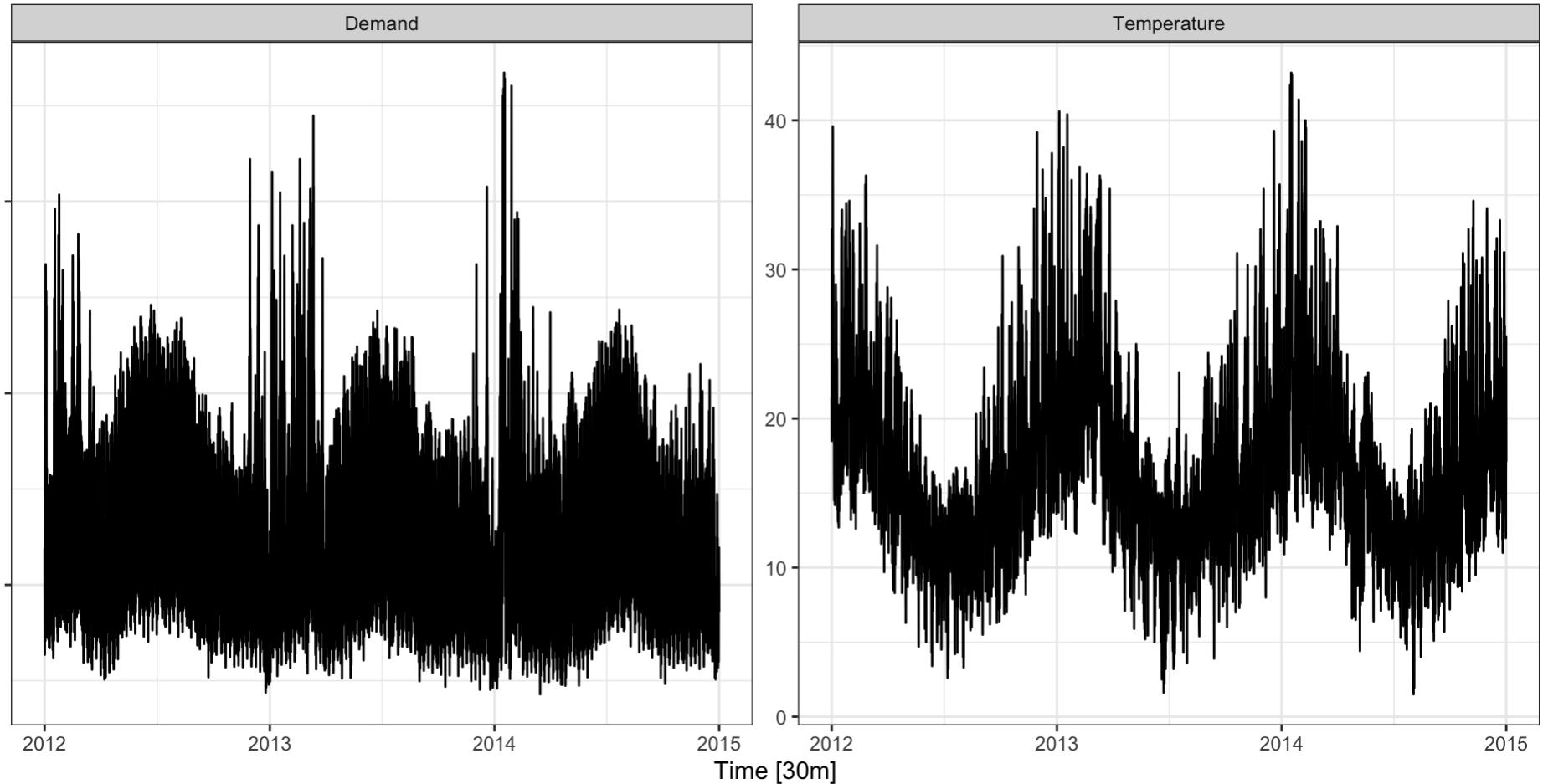
# Half-hourly electricity demand - vic\_elec

```
1 tsibbledata::vic_elec

# A tsibble: 52,608 x 5 [30m] <Australia/Melbourne>
  Time           Demand Temperature Date   Holiday
  <dttm>        <dbl>      <dbl> <date>  <lgl>
1 2012-01-01 00:00:00 4383.     21.4 2012-01-01 TRUE
2 2012-01-01 00:30:00 4263.     21.0 2012-01-01 TRUE
3 2012-01-01 01:00:00 4049.     20.7 2012-01-01 TRUE
4 2012-01-01 01:30:00 3878.     20.6 2012-01-01 TRUE
5 2012-01-01 02:00:00 4036.     20.4 2012-01-01 TRUE
6 2012-01-01 02:30:00 3866.     20.2 2012-01-01 TRUE
7 2012-01-01 03:00:00 3694.     20.1 2012-01-01 TRUE
8 2012-01-01 03:30:00 3562.     19.6 2012-01-01 TRUE
9 2012-01-01 04:00:00 3433.     19.1 2012-01-01 TRUE
```

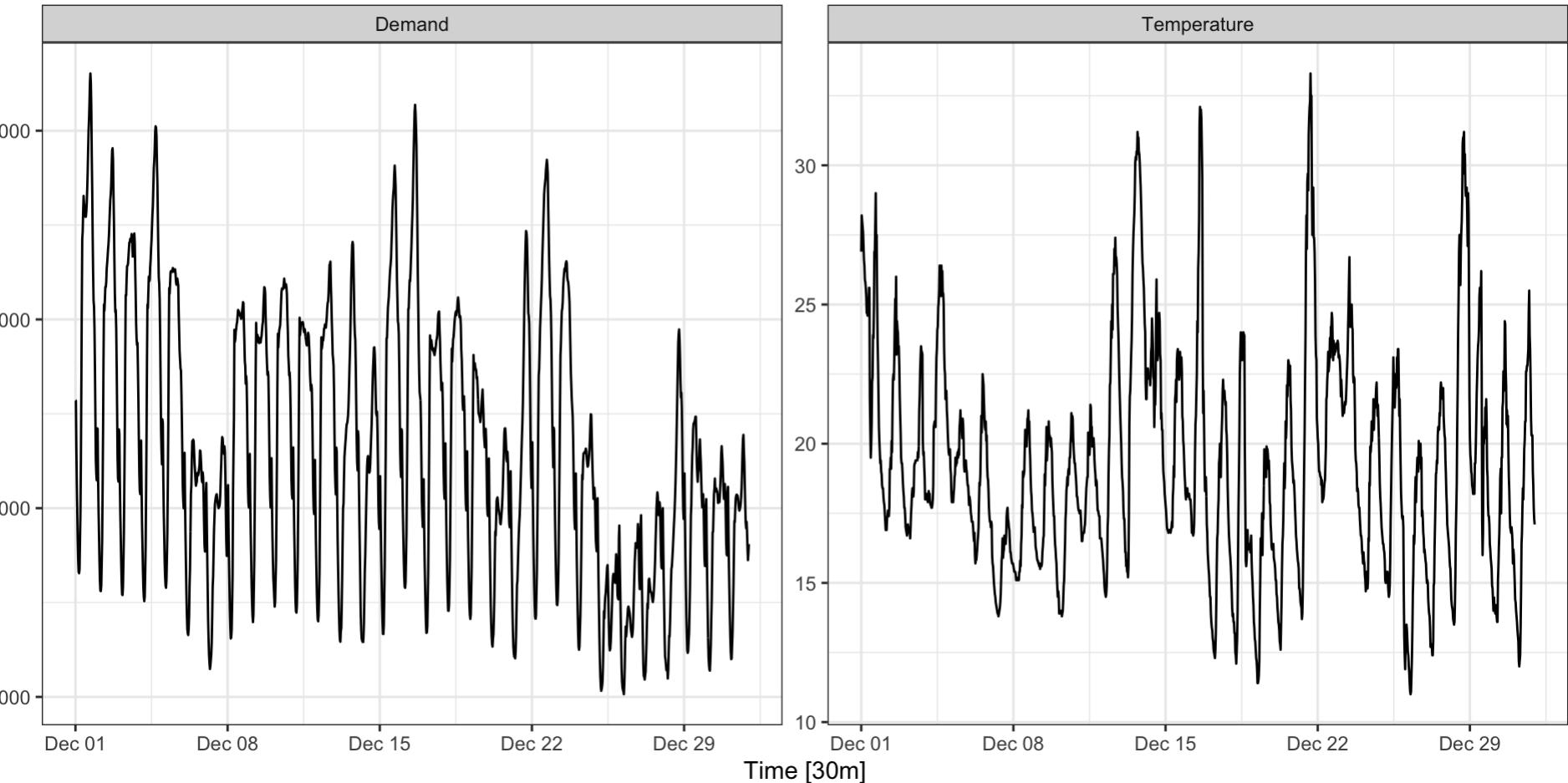
# Full data

```
1 tsibbledata:::vic_elec %>%
2   autoplot(vars(Demand, Temperature))
```



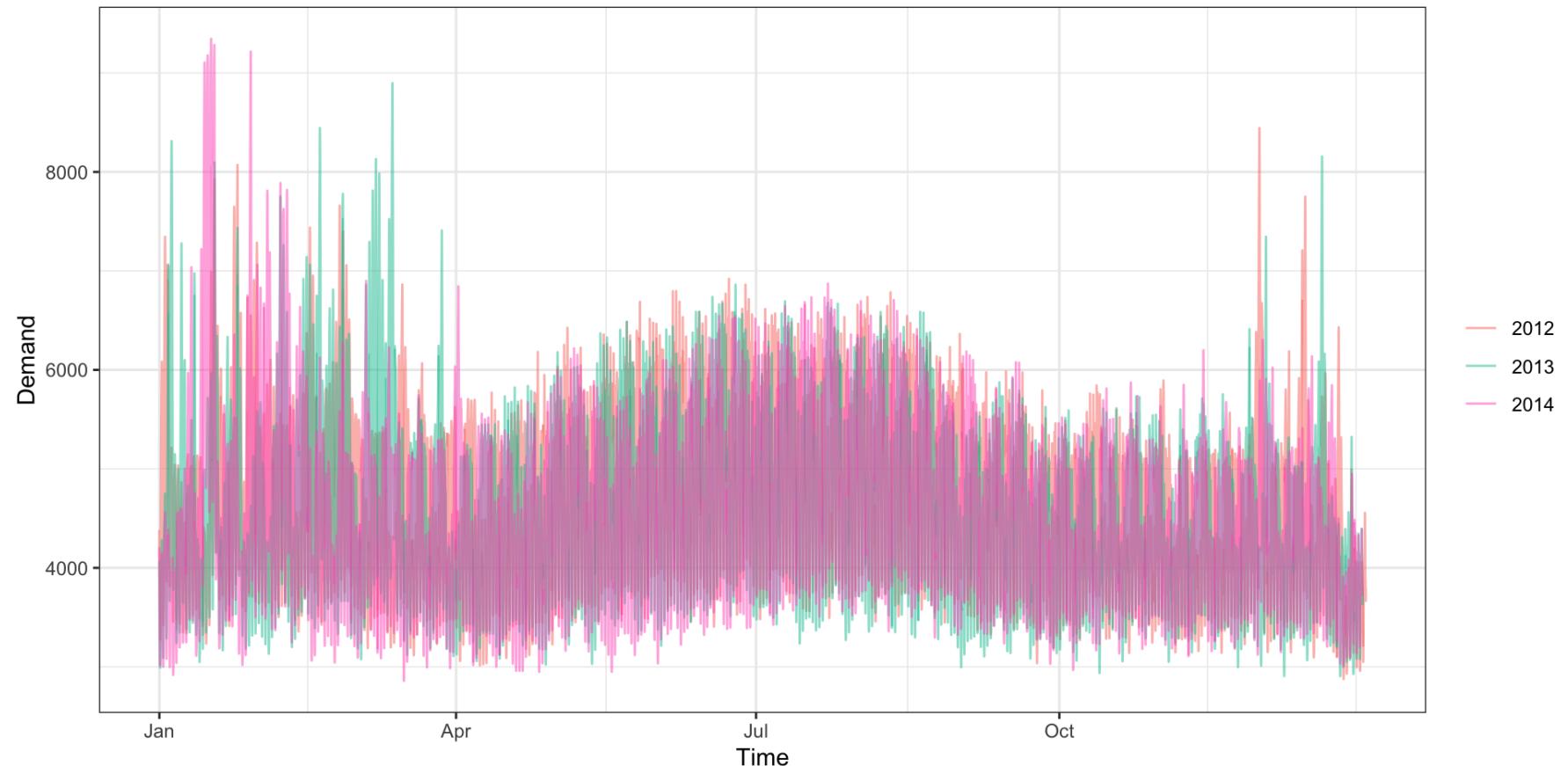
# Data - December 2014

```
1 tsibbledata::vic_elec %>%
2   filter(Time >= lubridate::ymd("2014/12/01")) %>%
3   autoplot(vars(Demand, Temperature))
```



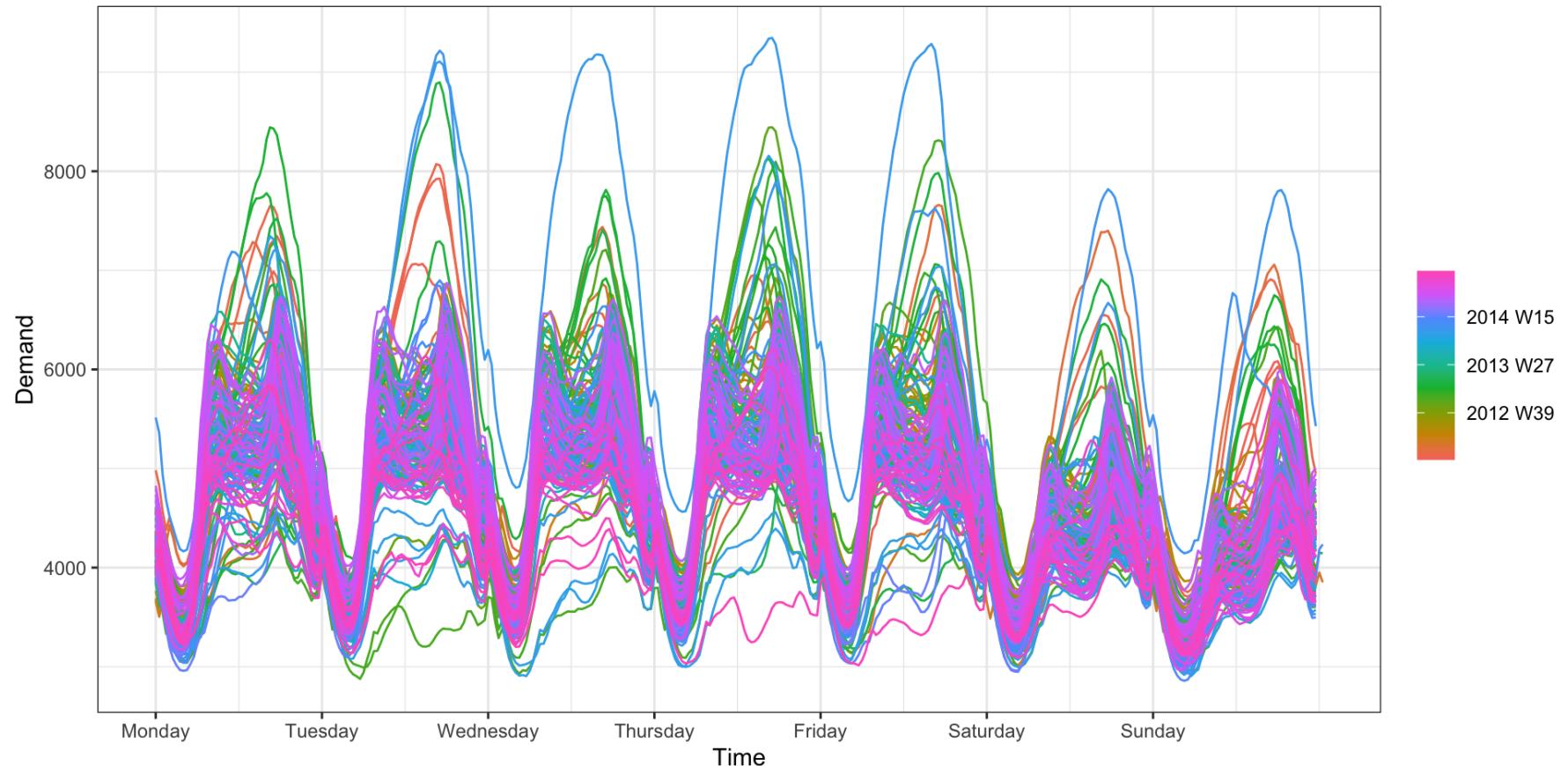
# Seasonality - Yearly

```
1 tsibbledata::vic_elec %>%
2   feasts::gg_season(
3     y=Demand, period = "year", alpha=0.5
4   )
```



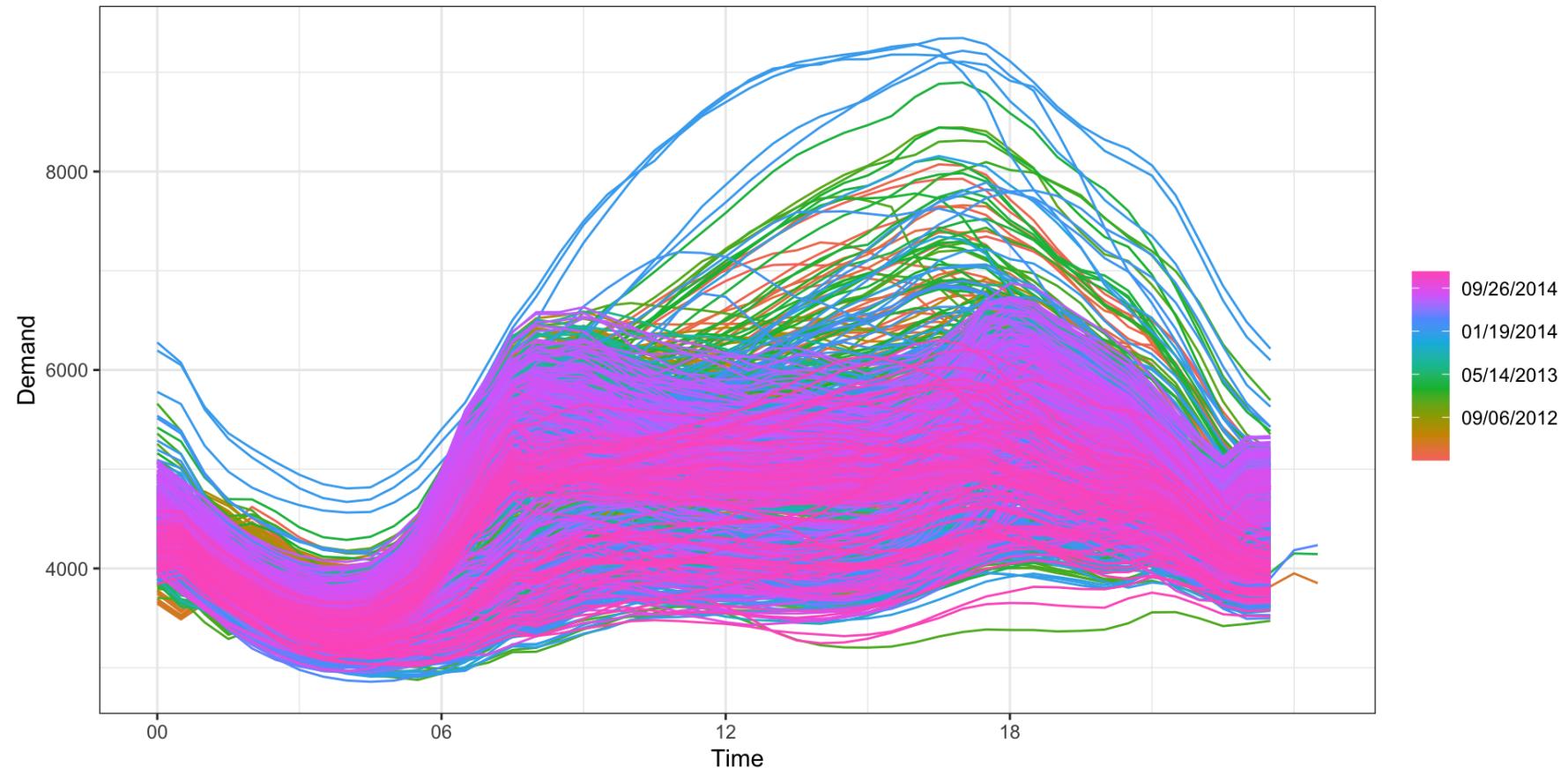
# Seasonality - Weekly

```
1 tsibbledata::vic_elec %>%
2   feasts::gg_season(
3     y=Demand, period = "week"
4   )
```



# Seasonality - Daily

```
1 tsibbledata::vic_elec %>%
2   feasts::gg_season(
3     y = Demand, period = "day"
4   )
```



# Model

```
1 ( vic_elec_fit = tsibbledata::vic_elec %>%
 2   model(
 3     p = prophet(Demand ~ Temperature + Holiday +
 4                 season(period = "day", order = 10) +
 5                 season(period = "week", order = 5) +
 6                 season(period = "year", order = 3)))
 7   )
 8 )
```

```
# A mable: 1 x 1
```

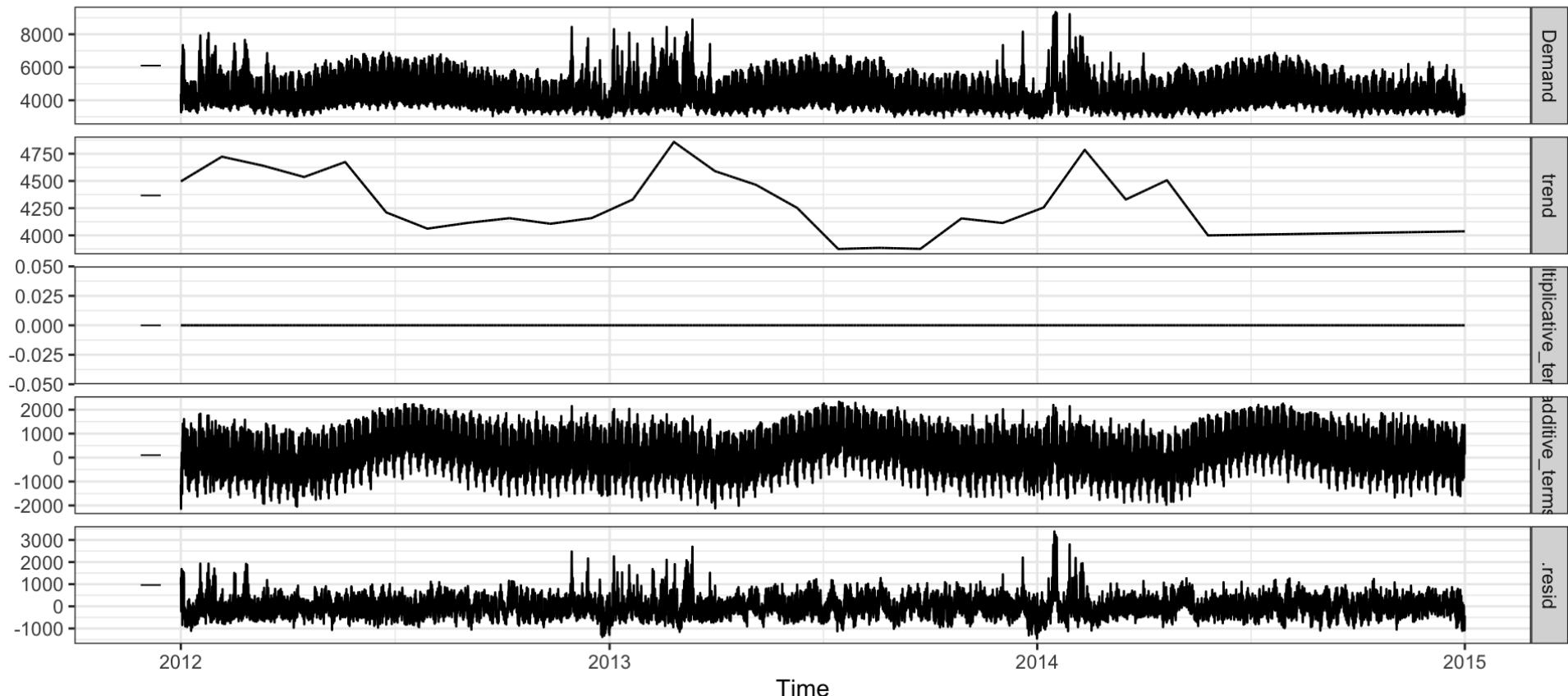
```
  p
<model>
1 <prophet>
```

# Components

```
1 vic_elec_fit %>%
2   components() %>%
3   autoplot()
```

Prophet decomposition

Demand = trend \* (1 + multiplicative\_terms) + additive\_terms + .resid



# Residuals

```
1 vic_elec_fit %>%
2   feasts::gg_tsresiduals()
```

