Normal Dynamic Linear Models

André F. B. Menezes

Last compiled on May 31, 2023

Contents

1 Introduction

This vignette will show examples of univariate Normal Dynamic Linear Models (DLMs) in real application using the `RBATS` package.

```
library(RBATS)
```

2 Bayesian Normal Dynamic Linear Models

Let the information set be represented as $D_t = \{D_0, \ldots, D_{t-1}\}$ at any time, with D_0 referring to the initial prior information at $t = 0$. Assuming that $(\boldsymbol{\theta}_0 \mid D_0) \sim N[\mathbf{m}_0, \mathbf{C}_0]$, for some known moments $\mathbf{m}_0$ and $\mathbf{C}_0$, the general normal DLM is defined by:

$$Y_t = \mathbf{F}_t^\top \boldsymbol{\theta}_t + \nu_t, \qquad \nu_t \sim N[0, V_t] \tag{1}$$

$$\boldsymbol{\theta}_t = \mathbf{G}_t \boldsymbol{\theta}_{t-1} + \boldsymbol{\omega}_t, \quad \boldsymbol{\omega}_t \sim N[\mathbf{0}, \mathbf{W}_t] \tag{2}$$

where Y_t is the observation at time t, $\mathbf{F}_t$ is a $p \times 1$ regression vector with known constants at time t, ν_t is the observation error at time t, $\boldsymbol{\theta}_t$ is a $p \times 1$ state vector at time t, $\boldsymbol{\omega}_t$ is the state evolution error, $\mathbf{G}_t$ is a $p \times p$ known matrix describing the state evolution, V_t is the observation variance, and $\mathbf{W}_t$ is the evolution variance-covariance matrix. This model is describe by the quadruple $\{\mathbf{F}_t, \mathbf{G}_t, V_t, \mathbf{W}_t\}$.

Two major model types can be distinguished: time series models where $\mathbf{F}_t = \mathbf{F}$ and $\mathbf{G}_t = \mathbf{G}$; and dynamic regression models where $\mathbf{F}_t = (X_{t,1}, \ldots, X_{t,p})'_t$ and $\mathbf{G} = \mathbf{I}_p$, where $X_{t,i}$ denotes the i covariate at time t and $\mathbf{I}_p$ indicates the identity matrix of order p.

2.1 Model components

DLMs are defined as the combination of simpler components based on the superposition principle to make it simple to identify process features. The superposition of $r \geq 1$ sub-models represented by $\{\mathbf{F}_i, \mathbf{G}_i, V_i, \mathbf{W}_i\}_t$, for $i = 1, \cdots, r$, can be used to specify a DLM. By defining $\mathbf{F}'_t = (\mathbf{F}'_1, \cdots, \mathbf{F}'_r), \mathbf{G}_t = \text{diag}(\mathbf{G}_1, \cdots, \mathbf{G}_r)_t$ and $\mathbf{W}_t = (\mathbf{W}_1, \cdots, \mathbf{W}_r)_t$, we can obtain the model described by the r components.

In fact, the current version of `RBATS` package implements three main model components: polynomial, seasonal and regressors. In the sequel, we will show how to define those models throughout the `dlm` function.

2.1.1 Nonstationary polynomial trend models

These models are special case of time series model, where $\mathbf{F}_t = \mathbf{F}$ and $\mathbf{G}_t = \mathbf{G}$. The simplest model is the local level, where $\mathbf{F} = \mathbf{F} = 1$ and the scalar vector θ_t represents the expected level of the series at time t, that is

$$Y_t = \theta_t + \nu_t \quad \text{and} \quad \theta_t = \theta_{t-1} + \omega_t.$$

The class of second order polynomial models has $\mathbf{F} = (1, 0)^\top$ and $\mathbf{G} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and the state vector $\boldsymbol{\theta}_t = (\mu_t, \beta_t)$ has two elements, the first representing the level and the second the growth.

The general p-order polynomial models have a p-dimensional state vector, $\mathbf{F} = (1, 0, \ldots, 0)^\top$, and $\mathbf{G}$ matrix $p \times p$ of Jordan form, where the diagonal and superdiagonal elements are all equal to one, that is

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Then, the element $\theta_{t,r}$ in the state vector represents the r-th derivative or difference in the series trend at time t. It undergoes random fluctuations over time, influenced by the corresponding elements of the innovations vector $\boldsymbol{\omega}_t$.

2.1.2 Regressions

$\mathbf{F}_t = (X_{t,1}, \ldots, X_{t,p})^\top$ and $\mathbf{G} = \mathbf{I}_p$, where $X_{t,i}$ denotes the i covariate at time t and $\mathbf{I}_p$ indicates the identity matrix of order p.

2.1.2.1 Autoregressions The basic representation of autoregression models of order p in state space form consider $\mathbf{F}_t = (y_{t-1}, \ldots, y_{t-p})^\top$ and $\mathbf{G} = \mathbf{I}_p$.

Although not available in the current version of the `RBATS` package another common representation has $\mathbf{F}_t = (1, 0, \ldots, 0)^\top$ and

$$\mathbf{G} = \begin{pmatrix} \phi_1 & 1 & 0 & \ldots & 0 \\ \phi_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{p-1} & 0 & 0 & \ldots & 1 \\ \phi_p & 0 & 0 & \ldots & 0 \end{pmatrix}$$

with $V_t = 0$ for all t and $\mathbf{W}_t$ having zero entries except the $\mathbf{W}_{1,1} = w_t > 0$.

2.1.3 Seasonal

There are two main approaches to specify the seasonality component in the `RBATS` package. Also, the current version only supports one component of seasonality, that is, it is not possible to include multiple seasonality such as weekly and annual.

2.1.3.1 Free form

2.1.3.2 Fourier form

2.2 Sequential Inference

One main feature of Bayesian DLMs is the sequential inference based on upon the Bayes' theorem. If the observation and evolution equation are known the results coincide with the Kalman filter. Because of that, the the sequential operation to obtain the prior, predictive and posterior distribution is known as forward filter. In `RBATS` the method `forward_filter` for objects of class `dlm` perform this operation sequentially on time, including the estimation of the variances, V_t and $\mathbf{W}_t$.

The results presented below, which assume that V_t and $\mathbf{W}_t$ are known, are established in Chapter 4 of West and Harrison (1997).

Evolution – prior distribution

$$(\boldsymbol{\theta}_t \mid D_{t-1}) \sim N[\mathbf{a}_t, \mathbf{R}_t]$$

with $\mathbf{a}_t = \mathbf{G}\,\mathbf{m}_{t-1}$ and $\mathbf{R}_t = \mathbf{G}'\mathbf{C}_{t-1}\mathbf{G}_t + \mathbf{W}_t$.

Forecast – predictive distribution

$$(Y_t \mid D_{t-1}) \sim N[f_t, Q_t]$$

where $f_t = \mathbf{F}_t'\,\mathbf{a}_t$ and $Q_t = \mathbf{F}_t'\,\mathbf{R}_t\,\mathbf{F}_t + V_t$.

Updating – posterior distribution

$$(\boldsymbol{\theta}_t \mid D_t) \sim N[\mathbf{m}_t, \mathbf{C}_t]$$

with $\mathbf{m}_t = \mathbf{a}_t + \mathrm{A}_t\,e_t$ and $\mathbf{C}_t = \mathbf{R}_t - \mathrm{A}_t\,\mathrm{A}_t'\,Q_t$, where $\mathrm{A}_t = \mathbf{R}_t\,\mathbf{F}_t/Q_t$ and $e_t = y_t - f_t$.

2.2.1 Estimation of unknown time-varying observation variance

In order to estimate the observation variance the normal-gamma distribution is used to obtain a conjugate analysis. Also, to introduce a dynamic in the observation variance we assume the following model for the $\phi_t = 1/V_t$, the observation precision

$$\phi_t = \gamma_t\,\phi_{t-1}/\beta, \quad \text{with} \quad \gamma_t \sim \mathrm{Beta}[\beta n_{t-1}/2, (1-\beta)\,n_{t-1}/2] \tag{3}$$

$$(\phi_0 \mid D_0) \sim \mathrm{Gamma}(n_0/2, d_0/2) \tag{4}$$

where the preceding degrees of freedom (n_0) and the sum of squares (d_0), respectively are prior parameters that should be specified.

The variance discount factor, $\beta \in (0, 1]$, enables the variance to evolve over time, and if it is equal to one, it leads to the constant variance model, $V_t = V$, for all t. Typically, β values are between 0.95 and 0.99 (see Chapter 10 of West and Harrison (1997)).

Unconditional to V_t, all of the previously presented distributions transform into t-Student distributions (see Chapter 10 of West and Harrison (1997)).

2.2.2 Specification of evolution variance matrix

To maintain the sequential inference feature, $\mathbf{W}_t$ is specified using the **discount factor** technique. A single discount factor denoted by $\delta \in (0, 1]$ is a multiplicative factor useful to **increase the uncertainty** in the evolution step of the sequential inference in DLMs.

Note that the prior variance is defined as $\mathbf{R}_t = \mathbf{G}_t'\mathbf{C}_{t-1}\mathbf{G}_t + \mathbf{W}_t = \mathbf{P}_t + \mathbf{W}_t$, where $\mathbf{P}_t = \mathbf{G}_t'\mathbf{C}_{t-1}\mathbf{G}_t$. For the static model $\mathbf{W}_t = 0$, so that $\mathbf{R}_t = \mathbf{G}_t'\mathbf{C}_{t-1}\mathbf{G}_t$. On the other hand, if the model is dynamic the precision $\mathbf{R}_t^{-1}$ is reduced relative to the posterior at time $t-1$, i.e., $\mathbf{P}_t^{-1}$. Then the intuitive way to express this relation is by

$$\mathbf{R}_t^{-1} = \delta \mathbf{P}_t^{-1} \rightarrow \mathbf{R}_t = \delta^{-1}\mathbf{P}_t$$

where $\delta \in (0, 1]$ is a discount factor. This implies that the system variance can be specified as

$$\mathbf{W}_t = \frac{1-\delta}{\delta}\,\mathbf{P}_t = \frac{1-\delta}{\delta}\,\mathbf{G}_t'\mathbf{C}_{t-1}\mathbf{G}_t.$$

In practice, a single discount factor is usually not appropriate, so different discounts are used for each model component. That is, the evolution matrices, $\mathbf{W}_{it}$, are determined by $\mathbf{W}_{it} = (1/\delta_i - 1)\,\mathbf{P}_{it}$, where δ_i is the discount factor for the i-th component, for $i = 1, \ldots, r$. In addition, δ_i lies in $(0, 1]$ and its typical range for practical analysis is $[0.9, 0.99]$. Clearly, if $\delta_i = 1$, then $\mathbf{W}_{it} = 0$ which leads to the static model for the component i. Conversely, if $\delta_i \approx 0$ the elements of $\mathbf{W}_{it}$ will be very large, resulting in an unstable model component.

The automatic monitoring method also available in RBATS throughout the `dlm` object is closely related to the specification of discount factor values. In particular, model adaptation is carried out by raising the uncertainty of the state parameters when a model breakdown is observed. We will discuss this below.

2.2.3 Smoothing distribution

2.3 Automatic Monitoring

3 Examples

3.1 Flow of the River Nile

To analyse this data set we will consider the local level model and compare the results with the `StructTS` function included in the `stats` package. The `StructTS` function is simple to use and performs the maximum likelihood estimation to obtain the observation and evolution variance,

which are assumed to be constant in time. However, this function does not provide standard errors of the estimates, which are required to compute the prediction interval.

```
fit_Nile <- StructTS(Nile, "level")
fit_Nile
#>
#> Call:
#> StructTS(x = Nile, type = "level")
#>
#> Variances:
#>   level  epsilon
#>    1469    15099
```

We can obtain the posterior level (filtered) component with the `fitted` function and the smoothing level using the `tsSmooth`, both methods should be applied to the `fit_Nile` object. We will keep those information in a `data.frame` to compare with the results from `RBATS`.

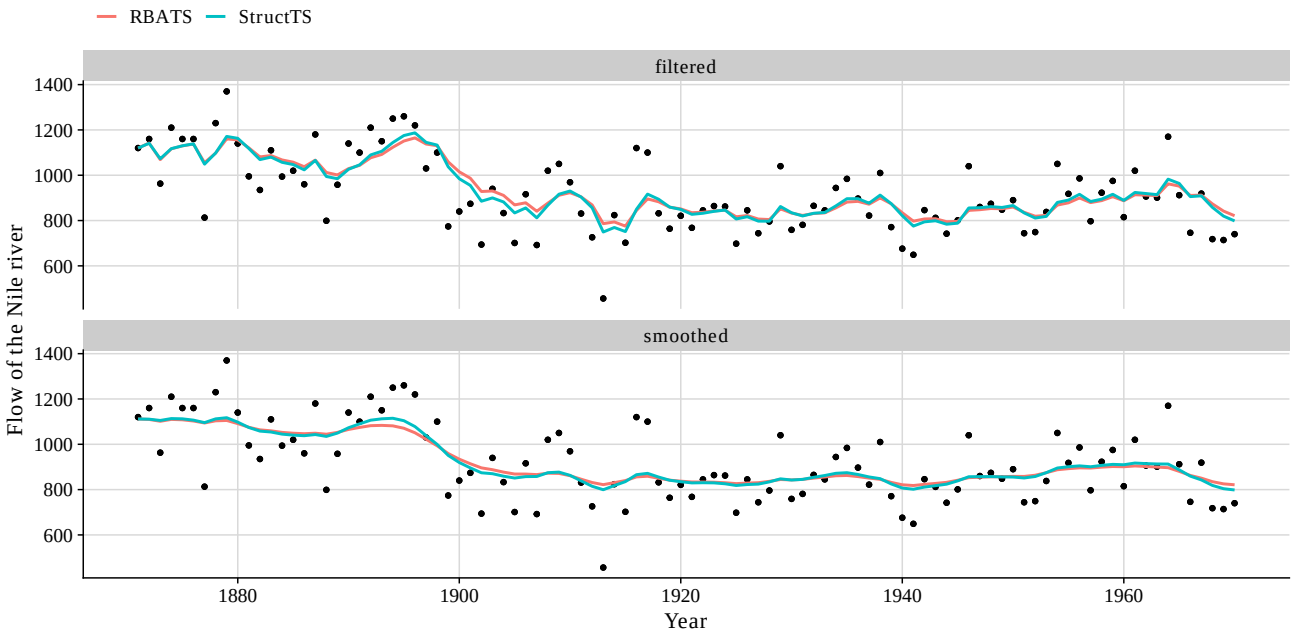
```
data_Nile <- data.frame(
  t = seq_along(Nile),
  time = seq.Date(as.Date("1871-01-01"), as.Date("1970-01-01"), by = "year"),
  y = c(Nile),
  filtered__StructTS = c(fitted(fit_Nile)),
  smoothed__StructTS = c(tsSmooth(fit_Nile))
)
```

The local level model in `RBATS` is defined considering the polynomial argument in `dlm` with `order = 1`. Also, I will consider a discount factor of `0.80` for the level and equal one for the variance, meaning that the observation variance is constant at time. The `fit` method for the `dlm` object performs the filtering and smoothing using the sequential Bayesian inference describe above. Under the Bayes approach we need to specify prior moments for the level component. In this case, a vague prior with mean 1000 and variance 1000 is specified.

```
model <- dlm(polynomial = list(order = 1, discount_factor = 0.80),
             df_variance = 1)
fit_model <- fit(model = model, y = c(Nile), a = matrix(1000, ncol = 1),
                 R = diag(1000, 1))
fit_model
#>
#> Bayesian Dynamic Linear Model
#>
```

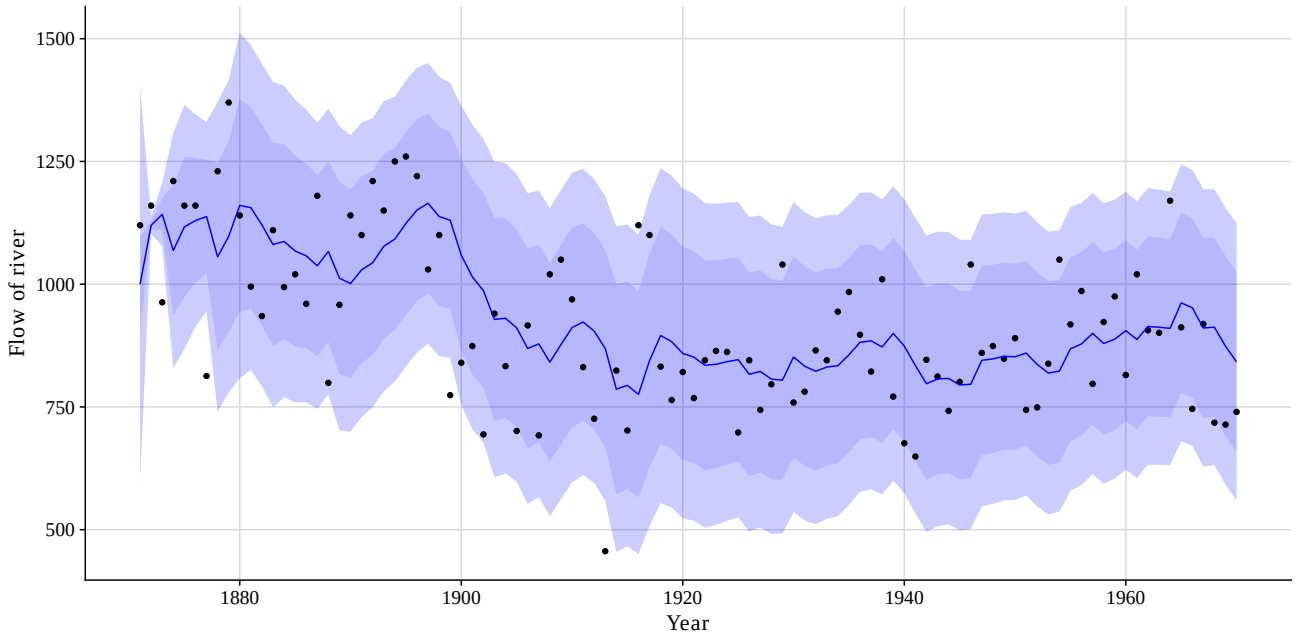
```
#> Call:
#> dlm(polynomial = list(order = 1, discount_factor = 0.8), df_variance = 1)
#>
#> Model specfication:
#>   parameter discount_factors
#> 1     level              0.8
#>
#> Variance law: identity
#>
#> Posterior parameters at time 0:
#>   parameter    mean variance
#> 1     level 821.317 3229.909
#>
#> Predictive log-likelihood: -648.9846
```

```
data_Nile$filtered__RBATS <- fit_model$filtered$m[1, ]
data_Nile$smoothed__RBATS <- fit_model$smoothed$ak[1, ]
data_Nile_pivotted <- tidyr::pivot_longer(data_Nile, cols = -c(t, time, y)) |>
  tidyr::separate(col = "name", into = c("distribution", "package"))
ggplot(data_Nile_pivotted, aes(x = time, y = y)) +
  facet_wrap(~distribution, ncol = 1) +
  geom_point(size = 1.5) +
  geom_line(aes(y = value, col = package), linewidth = 1) +
  labs(x = "Year", y = "Flow of the Nile river", col = "")
```



In fact, the Bayesian approach provides not only a point estimate, but a entire predictive or smoothed distribution. We can explore this by using the method `extract` applied to the `dlm.fit` objects.

```
data_predictive <- extract(x = fit_model, prob_interval = c(0.05, 0.20),
                           distribution = "filter", component = "response")
dplyr::glimpse(data_predictive)
#> Rows: 100
#> Columns: 10
#> $ t               <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
#> $ y               <dbl> 1120, 1160, 963, 1210, 1160, 1160, 813, 1230, 1370, 1140, 9
#> $ distribution    <chr> "filter", "filter", "filter", "filter", "filter", "filter",
#> $ mean            <dbl> 1000.0000, 1119.8801, 1142.1590, 1068.7525, 1116.5922, 1129
#> $ variance        <dbl> 1001.00000, 17.29921, 412.89638, 7438.95069, 9357.58979, 78
#> $ degrees_freedom <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
#> $ ci_lower__95    <dbl> 597.9937, 1101.9844, 1077.4922, 829.2859, 867.9279, 913.146
#> $ ci_upper__95    <dbl> 1402.006, 1137.776, 1206.826, 1308.219, 1365.257, 1345.860,
#> $ ci_lower__80    <dbl> 902.6265, 1112.0374, 1108.8803, 936.5144, 973.8231, 1002.19
#> $ ci_upper__80    <dbl> 1097.374, 1127.723, 1175.438, 1200.991, 1259.361, 1256.807,
data_predictive$time <- seq.Date(as.Date("1871-01-01"), as.Date("1970-01-01"),
                                 by = "year")
ggplot(data_predictive, aes(x = time, y = y)) +
  geom_ribbon(aes(ymin = ci_lower__95, ymax = ci_upper__95),
              fill = "blue", alpha = 0.2) +
  geom_ribbon(aes(ymin = ci_lower__80, ymax = ci_upper__80),
              fill = "blue", alpha = 0.1) +
  geom_point(size = 1.5) +
  geom_line(aes(y = mean), col = "blue") +
  labs(x = "Year", y = "Flow of river")
```



3.2 Telephone calls

```
model <- dlm(polynomial = list(order = 2, discount_factor = 0.80),
             df_variance = 1)
fit_model <- fit(model = model, y = c(telephone_calls$average_daily_calls),
                 a = matrix(c(300, 0), ncol = 1),
                 R = diag(1000, 2))
fit_model
#>
#> Bayesian Dynamic Linear Model
#>
#> Call:
#> dlm(polynomial = list(order = 2, discount_factor = 0.8), df_variance = 1)
#>
#> Model specfication:
#>   parameter discount_factors
#> 1     level              0.8
#> 2    growth              0.8
#>
#> Variance law: identity
#>
#> Posterior parameters at time 0:
#>   parameter       mean variance
#> 1     level 230.306993 673.8733
```

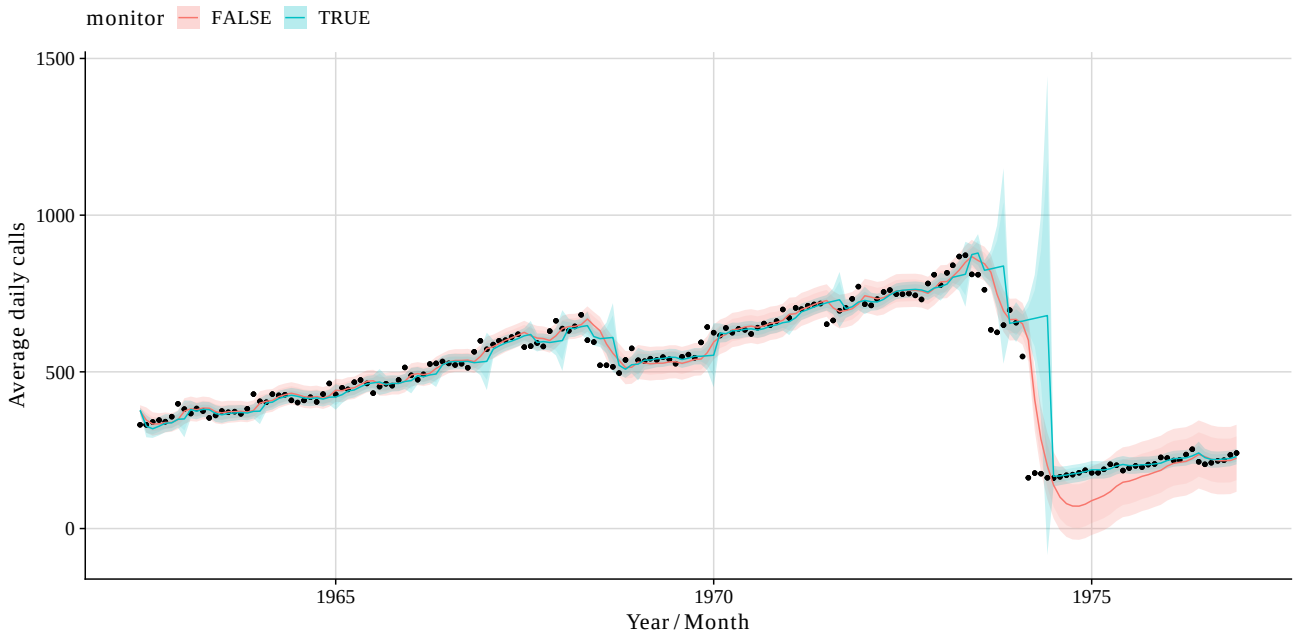
```
#> 2    growth   1.393191  18.7187
#>
#> Predictive log-likelihood: -990.0082

model_monitor <- dlm(polynomial = list(order = 2, discount_factor = c(0.90, 0.95)),
                     df_variance = 1,
                     monitor = list(
                       execute = TRUE, verbose = TRUE, start_time = 10L,
                       bilateral = TRUE,
                       bf_threshold = 0.135,
                       location_shift = 4, scale_shift = 1,
                       discount_factors = list(polynomial = c(0.2, 0.95))))
fit_model_monitor <- fit(model = model_monitor,
                         y = c(telephone_calls$average_daily_calls),
                         a = matrix(c(300, 0), ncol = 1),
                         R = diag(1000, 2))
fit_model_monitor
#>
#> Bayesian Dynamic Linear Model
#>
#> Call:
#> dlm(polynomial = list(order = 2, discount_factor = c(0.9, 0.95)), df_variance = 1,
#>     monitor = list(execute = TRUE, verbose = TRUE, start_time = 10L, bilateral = TR
#>         bf_threshold = 0.135, location_shift = 4, scale_shift = 1, discount_factors
#>             0.95))))
#>
#> Model specfication:
#>   parameter discount_factors
#> 1     level             0.90
#> 2    growth             0.95
#>
#> Variance law: identity
#>
#> Posterior parameters at time 0:
#>   parameter       mean  variance
#> 1     level 232.643792 30.056820
#> 2    growth   3.395046  1.028914
#>
```

```
#> Predictive log-likelihood: -634.0194
```

```
data_predictive <- extract(x = fit_model, prob_interval = c(0.05, 0.20),
                           distribution = "filter", component = "response")
data_predictive$time <- as.Date(telephone_calls$time)
data_predictive$monitor <- FALSE
str(data_predictive)
#> 'data.frame':    180 obs. of  12 variables:
#>  $ t              : int  1 2 3 4 5 6 7 8 9 10 ...
#>  $ y              : int  350 339 351 364 369 331 331 340 346 341 ...
#>  $ distribution   : chr  "filter" "filter" "filter" "filter" ...
#>  $ mean           : num  300 350 328 349 367 ...
#>  $ variance       : num  1001 2189.87 9.04 77.09 78.77 ...
#>  $ degrees_freedom: num  1 2 3 4 5 6 7 8 9 10 ...
#>  $ ci_lower__95   : num  -102 149 319 325 344 ...
#>  $ ci_upper__95   : num  702 551 338 374 390 ...
#>  $ ci_lower__80   : num  203 262 323 336 354 ...
#>  $ ci_upper__80   : num  397 438 333 363 380 ...
#>  $ time           : Date, format: "1962-01-01" "1962-02-01" "1962-03-01" ...
#>  $ monitor        : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
data_predictive_monitor <- extract(x = fit_model_monitor,
                                   prob_interval = c(0.05, 0.20),
                                   distribution = "filter",
                                   component = "response")
data_predictive_monitor$time <- as.Date(telephone_calls$time)
data_predictive_monitor$monitor <- TRUE
colnames(data_predictive) == colnames(data_predictive_monitor)
#>  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
data_predictive <- dplyr::bind_rows(data_predictive, data_predictive_monitor)
dplyr::glimpse(data_predictive)
#> Rows: 360
#> Columns: 12
#> $ t              <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
#> $ y              <int> 350, 339, 351, 364, 369, 331, 331, 340, 346, 341, 357, 398,
#> $ distribution   <chr> "filter", "filter", "filter", "filter", "filter", "filter",
#> $ mean           <dbl> 300.0000, 349.9500, 328.0784, 349.3399, 366.9695, 375.3114,
#> $ variance       <dbl> 1001.000000, 2189.871567, 9.043504, 77.087156, 78.769575, 5
#> $ degrees_freedom <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
#> $ ci_lower__95   <dbl> -102.0063, 148.6028, 318.5080, 324.9629, 344.1550, 357.0156
```

```
#> $ ci_upper__95    <dbl> 702.0063, 551.2973, 337.6487, 373.7169, 389.7839, 393.6072,
#> $ ci_lower__80    <dbl> 202.6265, 261.7105, 323.1533, 335.8785, 353.8707, 364.5462,
#> $ ci_upper__80    <dbl> 397.3735, 438.1896, 333.0035, 362.8014, 380.0683, 386.0766,
#> $ time            <date> 1962-01-01, 1962-02-01, 1962-03-01, 1962-04-01, 1962-05-01
#> $ monitor         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL
```

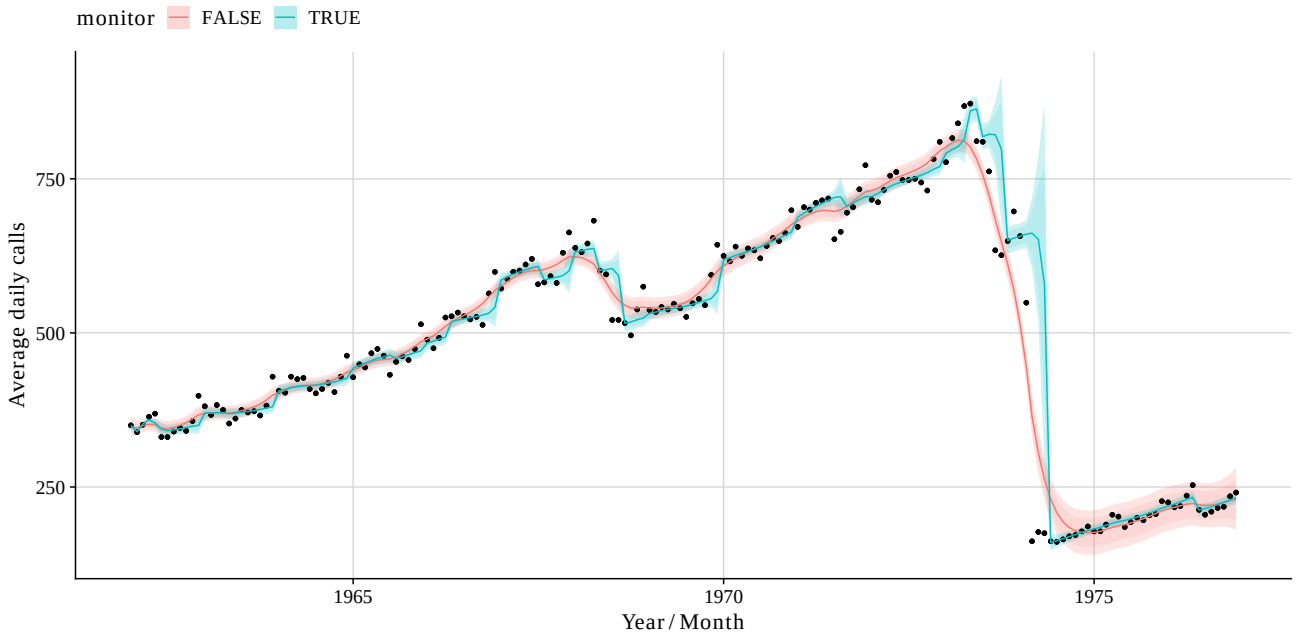
```
ggplot(data_predictive[data_predictive$t > 5, ], aes(x = time, y = y)) +
  geom_ribbon(aes(ymin = ci_lower__95, ymax = ci_upper__95, fill = monitor),
              alpha = 0.2) +
  geom_ribbon(aes(ymin = ci_lower__80, ymax = ci_upper__80, fill = monitor),
              alpha = 0.1) +
  geom_point(size = 1.2) +
  geom_line(aes(y = mean, col = monitor)) +
  labs(x = "Year/Month", y = "Average daily calls")
```




```
data_state <- extract(x = fit_model, prob_interval = c(0.05, 0.20),
                      distribution = "smooth", component = "state")
data_state$time <- rep(as.Date(telephone_calls$time), times = 2)
data_state$monitor <- FALSE
str(data_state)
#> 'data.frame':    360 obs. of  13 variables:
#>  $ t              : int  1 2 3 4 5 6 7 8 9 10 ...
#>  $ y              : int  350 339 351 364 369 331 331 340 346 341 ...
#>  $ distribution   : chr  "smooth" "smooth" "smooth" "smooth" ...
```

```
#>  $ parameter      : chr  "level" "level" "level" "level" ...
#>  $ mean           : num  347 346 350 352 351 ...
#>  $ variance       : num  15 13.6 15.7 16.3 18.7 ...
#>  $ degrees_freedom: num  2 3 4 5 6 7 8 9 10 11 ...
#>  $ ci_lower__95   : num  331 334 339 342 340 ...
#>  $ ci_upper__95   : num  364 358 361 362 361 ...
#>  $ ci_lower__80   : num  340 340 344 346 344 ...
#>  $ ci_upper__80   : num  355 352 356 358 357 ...
#>  $ time           : Date, format: "1962-01-01" "1962-02-01" "1962-03-01" ...
#>  $ monitor        : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
data_state_monitor <- extract(x = fit_model_monitor,
                              prob_interval = c(0.05, 0.20),
                              distribution = "smooth", component = "state")
data_state_monitor$time <- rep(as.Date(telephone_calls$time), times = 2)
data_state_monitor$monitor <- TRUE

data_state <- rbind(data_state, data_state_monitor)

ggplot(data_state[data_state$parameter == "level", ],
       aes(x = time, y = y)) +
  geom_ribbon(aes(ymin = ci_lower__95, ymax = ci_upper__95, fill = monitor),
              alpha = 0.2) +
  geom_ribbon(aes(ymin = ci_lower__80, ymax = ci_upper__80, fill = monitor),
              alpha = 0.1) +
  geom_point(size = 1.2) +
  geom_line(aes(y = mean, col = monitor)) +
  labs(x = "Year/Month", y = "Average daily calls")
```



3.3 Monthly Airline Passenger Numbers