Rating Shift Happens

Documentation

Felipe MENDEZ and Benjamin RENARD (INRAE, RiverLy and RECOVER).

December 2023

Introduction

The goal of `RatingShiftHappens` package is to create a tools package for detecting, visualizing and estimating rating shifts. This package was derived from BayDERS developed by M. Darienzo et al. (2021).

This documentation provides the description of several functions available to the segmentation process.

Three fundamental functions are available to segment a random variable :

1. Segmentation_Engine
2. Segmentation
3. Recursive_Segmentation

Installation

You can install the development version of RatingShiftHappens from GitHub using the following command. Please note that the RBaM package developed by Renard (2017) is also required to run this package.

```
# Before first use, install RatingShiftHappens and RBaM packages ones and for all, following these comm

# devtools::install_github("Felipemendezrios/RatingShiftHappens")
# devtools::install_github('BaM-tools/RBaM')

library(RatingShiftHappens)
```

Functions will be explained more precisely below along with an example.

Segmentation procedure for a *known* given number of segments

This basic example demonstrates the segmentation of annual maximum stages (H, m) for the Rhone River at Beaucaire, France, along with the associated uncertainties expressed as standard deviations (uH), divided into two groups. More information about the data set, please refer to the documentation available in `?RhoneRiverAMAX`.

In this package, all uncertainties are expressed as standard deviations rather than expanded uncertainties. Uncertainty in stage measurements will be in meters, while for discharge, it will be in cubic meters per second.

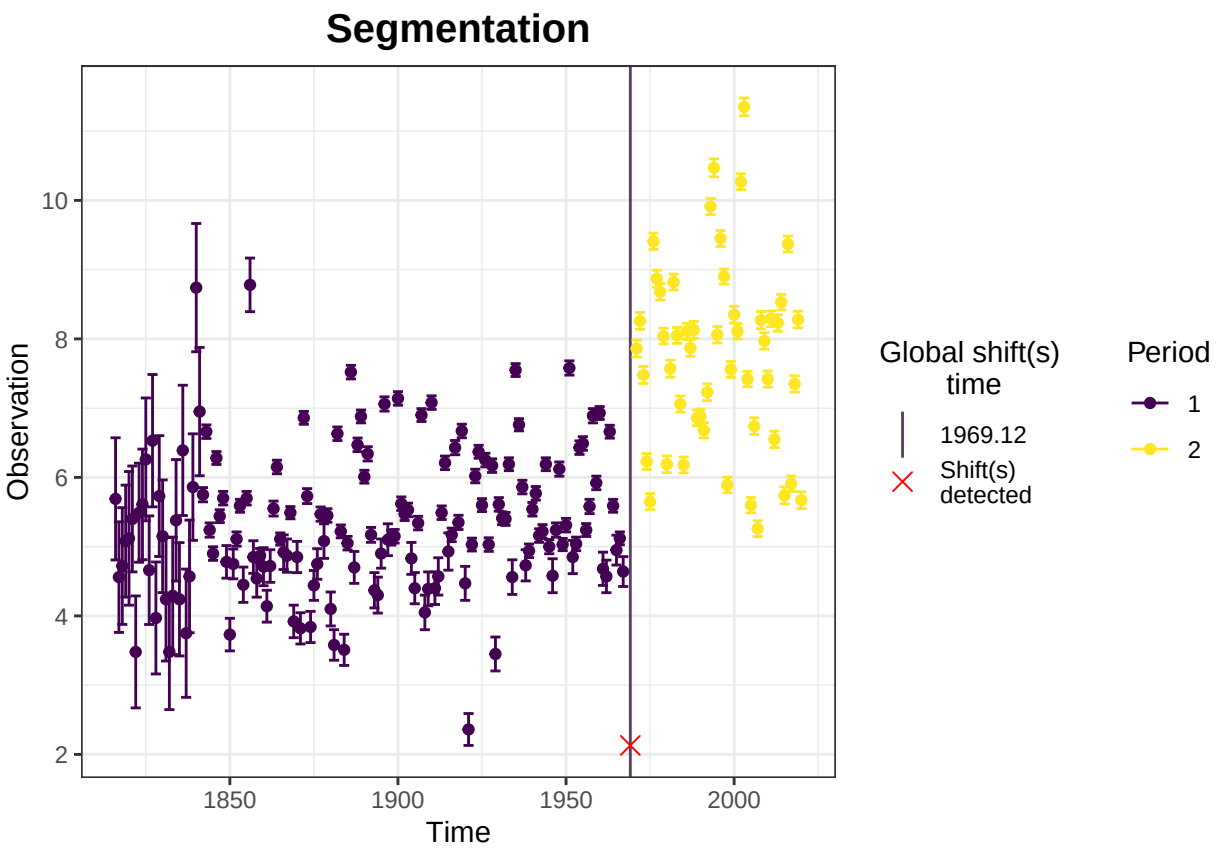
```
# Run segmentation engine function at two segments
res=Segmentation_Engine(obs=RhoneRiverAMAX$H,
                        time=RhoneRiverAMAX$Year,
                        u=RhoneRiverAMAX$uH,
                        nS=2)
# Data information
knitr::kable(head(res$summary$data),
             align = 'c',row.names = F)
```

time	obs	u	I95_lower	I95_upper	period
1816	5.69	0.4500	4.808016	6.571984	1
1817	4.56	0.4075	3.761315	5.358685	1
1818	4.72	0.4300	3.877216	5.562785	1
1819	5.08	0.4125	4.271515	5.888485	1
1820	5.12	0.4925	4.154718	6.085282	1
1821	5.40	0.3900	4.635614	6.164386	1

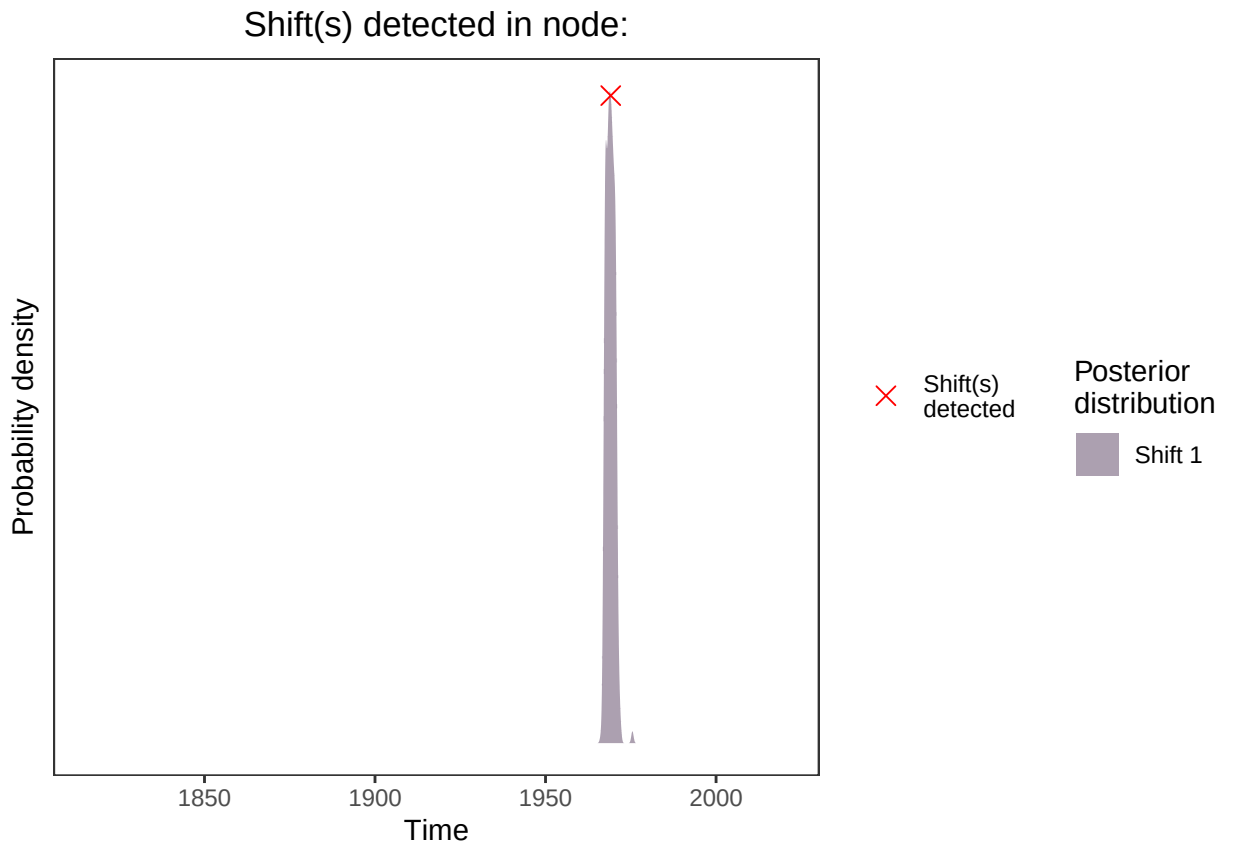
```
# Shift information
knitr::kable(head(res$summary$shift),
             align = 'c',row.names = F)
```

tau	I95_lower	I95_upper
1969.12	1967.09	1971.35

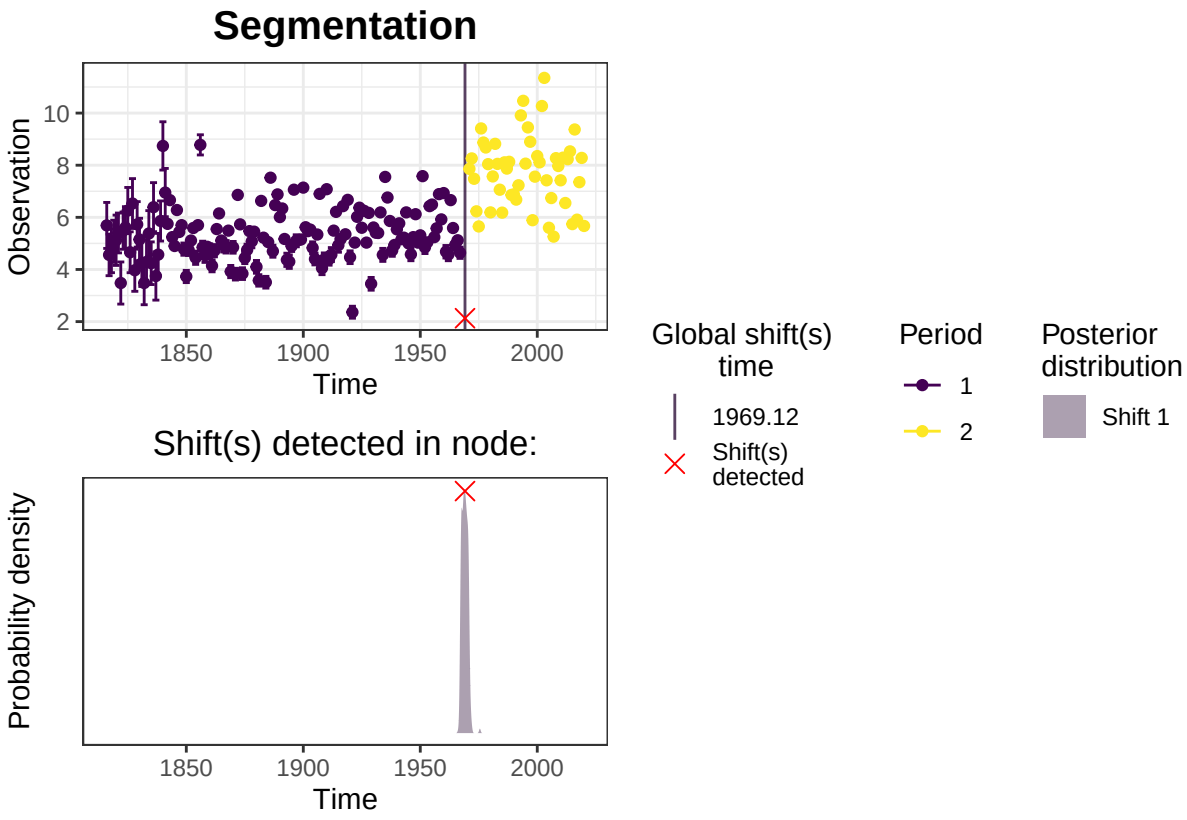
```
# Plot segmentation
Plots=PlotSegmentation(summary=res$summary,
                       plot_summary=res$plot)

# Observations of the random variable and shift time estimated
Plots$observation_and_shift
```



```
# Probability distribution function for detecting shift time with a 95% credibility interval
Plots$shift_time_density
```




```
# Final plot segmentation
Plots$final_plot
```



For more advanced details:

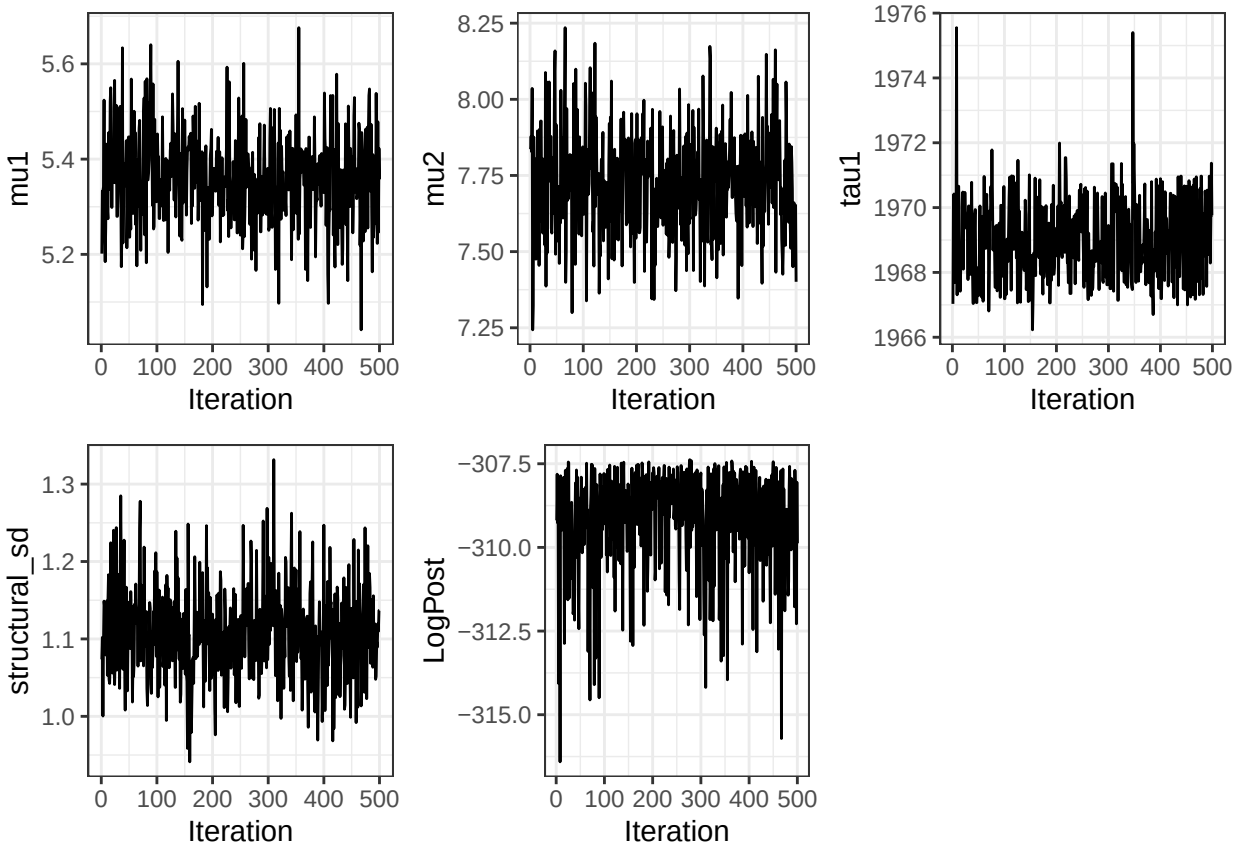
MCMC sampling demonstrate all combinations of parameters estimated.

```
knitr::kable(head(res$mcmc),align = 'c')
```

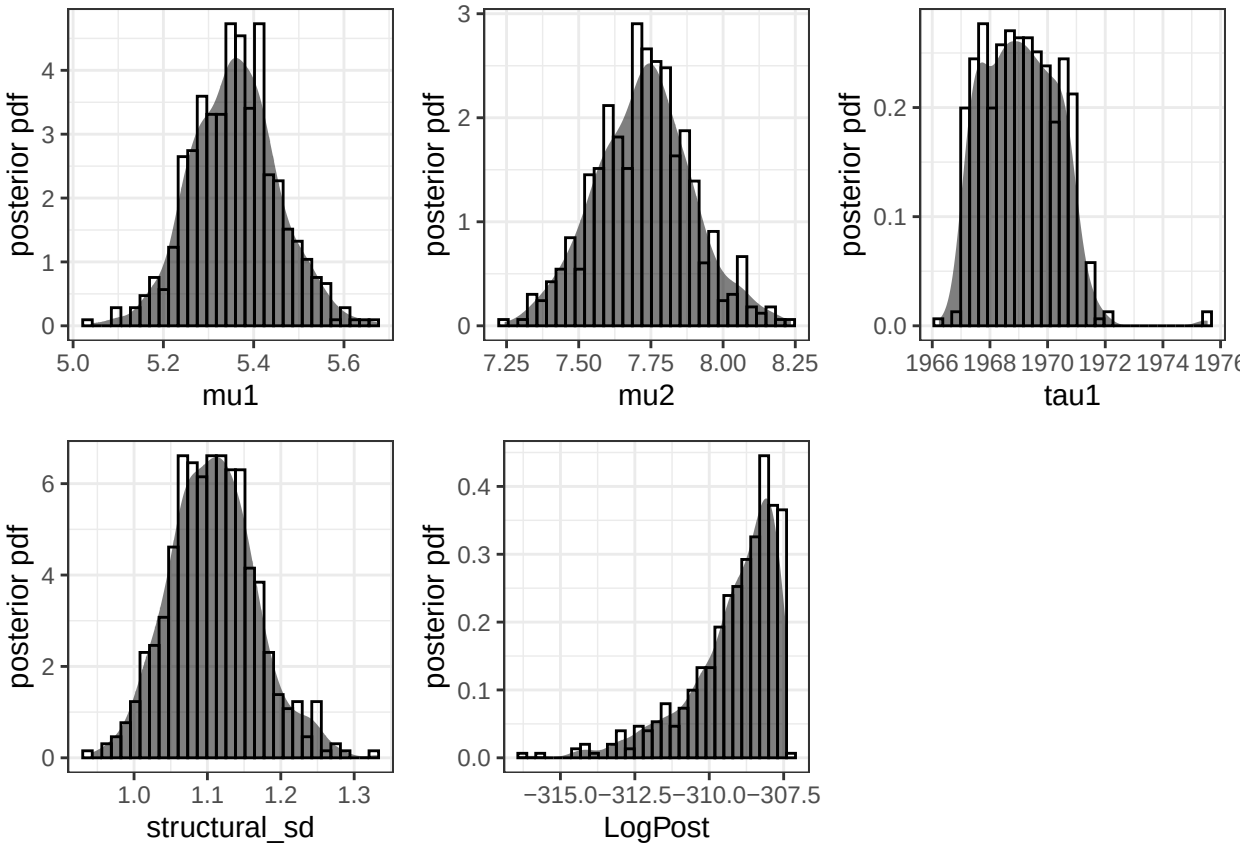
mu1	mu2	tau1	structural_sd	LogPost
5.20155	7.83626	1967.03	1.07363	-309.191
5.33280	7.87847	1970.18	1.10196	-307.813
5.33280	7.82841	1970.42	1.00044	-309.262
5.34227	8.03573	1967.69	1.09610	-309.200
5.52388	7.24338	1967.76	1.14877	-314.065
5.24600	7.32321	1969.19	1.14107	-311.731

The `RBaM` package provides several functions to explore MCMC samples.

```
# Trace plot for each parameter, useful to assess convergence.
plots=RBaM::tracePlot(res$mcmc)
gridExtra::grid.arrange(grobs=plots,ncol=3)
```

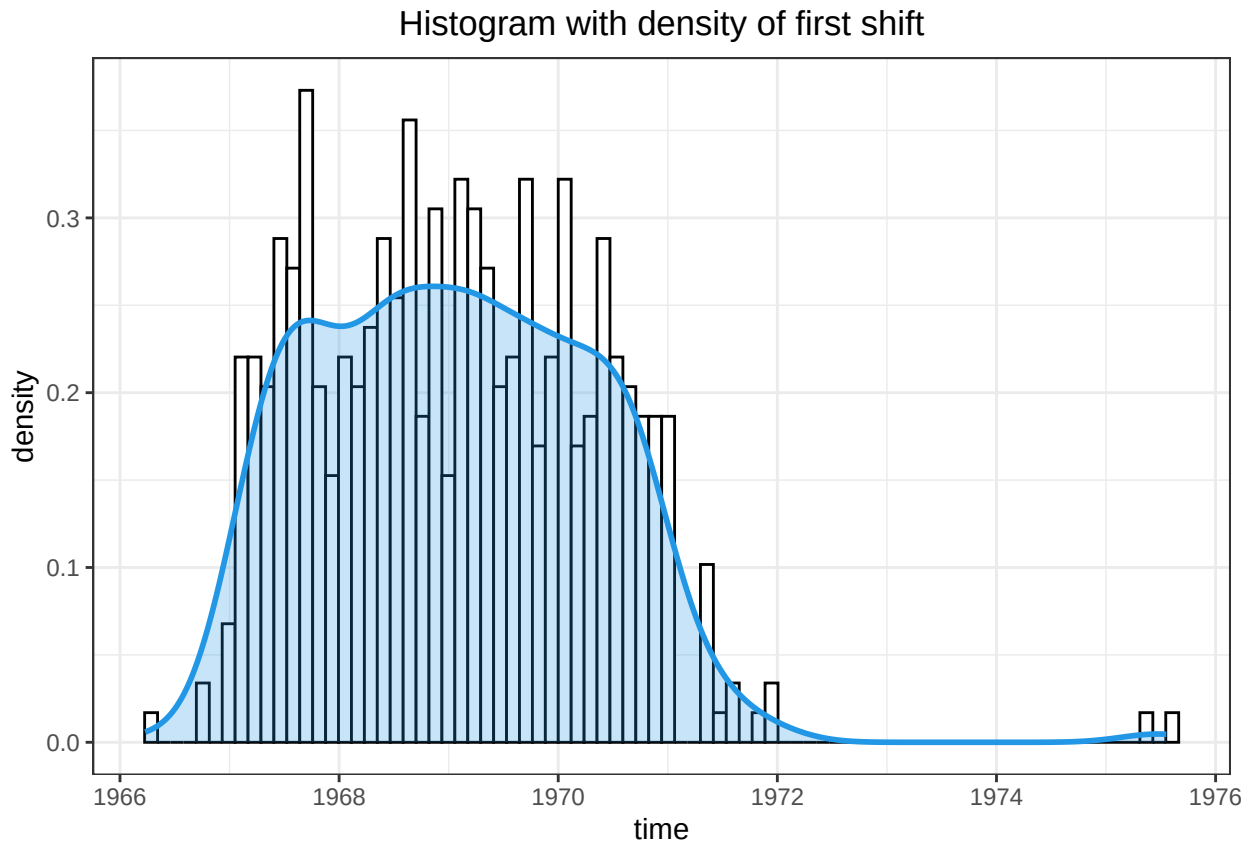



```
# Density plot for each parameter
plots=RBaM::densityPlot(res$mcmc)
gridExtra::grid.arrange(grobs=plots,ncol=3)
```



In this example, the focus will be on exploring the uncertainty associated with the first shift time.

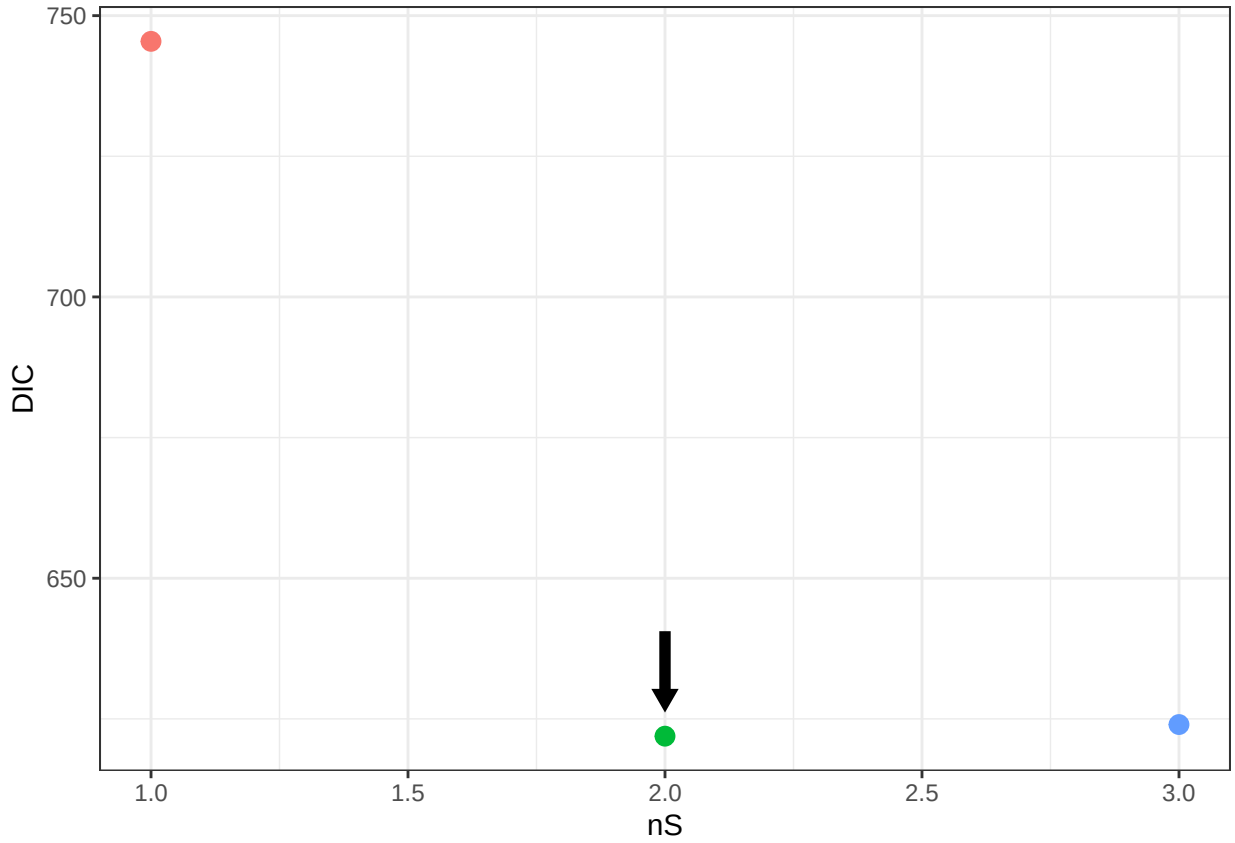
```
Shift=data.frame(time=res$mcmc$tau1)
ggplot2::ggplot(Shift,ggplot2::aes(x=time))+
  ggplot2::geom_histogram(ggplot2::aes(y=..density..),col=1,fill='white',bins=80)+
  ggplot2::labs(title='Histogram with density of first shift')+
  ggplot2::theme_bw()+
  ggplot2::theme(plot.title = ggplot2::element_text(hjust = 0.5))+
  ggplot2::geom_density(col=4,lwd=1,fill=4,alpha=0.25)
```



Segmentation procedure for an *unknown* number of segments

This is a basic example, which shows you how to segment the same dataset with an **unknown** number of segments :

```
# Run segmentation engine function at two segments
res=Segmentation(obs=RhoneRiverAMAX$H,
                 time=RhoneRiverAMAX$Year,
                 u=RhoneRiverAMAX$uH,
                 nSmax=3)

# Get lower DIC value and optimal number of segments (to define optimal solution)
DIC.df = data.frame(nS=c(1:3),DIC=c(res$results[[1]]$DIC,res$results[[2]]$DIC,res$results[[3]]$DIC))
nSopt=res$nS

ggplot2::ggplot(DIC.df,ggplot2::aes(x=nS,y=DIC,col=factor(nS)))+
  ggplot2::geom_point(size=3,show.legend = F)+
  ggplot2::annotate('segment',
                    x=nSopt,y=min(DIC.df$DIC)*1.03,xend=nSopt,yend=min(DIC.df$DIC)*1.01,
                    linewidth=2,linejoin = "mitre",
                    arrow=ggplot2::arrow(type='closed',length=ggplot2::unit(0.01,'npc')))+
  ggplot2::theme_bw()
```



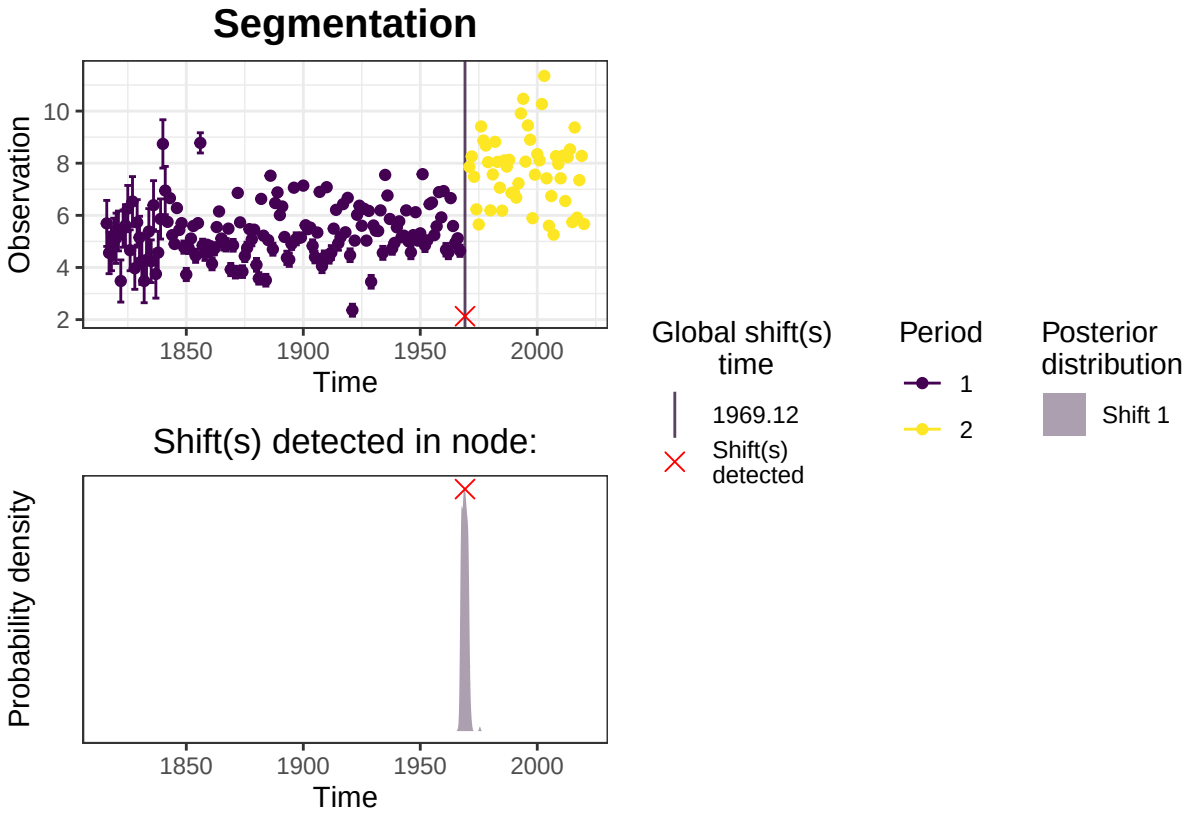

```
# Data information
knitr::kable(head(res$results[[nSopt]]$summary$data),
             align = 'c',row.names = F)
```

time	obs	u	I95_lower	I95_upper	period
1816	5.69	0.4500	4.808016	6.571984	1
1817	4.56	0.4075	3.761315	5.358685	1
1818	4.72	0.4300	3.877216	5.562785	1
1819	5.08	0.4125	4.271515	5.888485	1
1820	5.12	0.4925	4.154718	6.085282	1
1821	5.40	0.3900	4.635614	6.164386	1

```
# Shift information
knitr::kable(head(res$results[[nSopt]]$summary$shift),
             align = 'c',row.names = F)
```

tau	I95_lower	I95_upper
1969.12	1967.09	1971.35

```
# Final plot segmentation
PlotSegmentation(summary=res$summary,
                 plot_summary = res$plot)$final_plot
```



Recursive segmentation procedure for an *unknown* number of segments

This is a basic example, which shows you how to segment the data set with an **unknown** number of segments using a recursive process:

```
# Apply recursive segmentation
results=Recursive_Segmentation(obs=RhoneRiverAMAX$H,
                               time=RhoneRiverAMAX$Year,
                               u=RhoneRiverAMAX$uH,
                               nSmax=3)

# Data information
knitr::kable(head(results$summary$data),
             align = 'c',row.names = F)
```

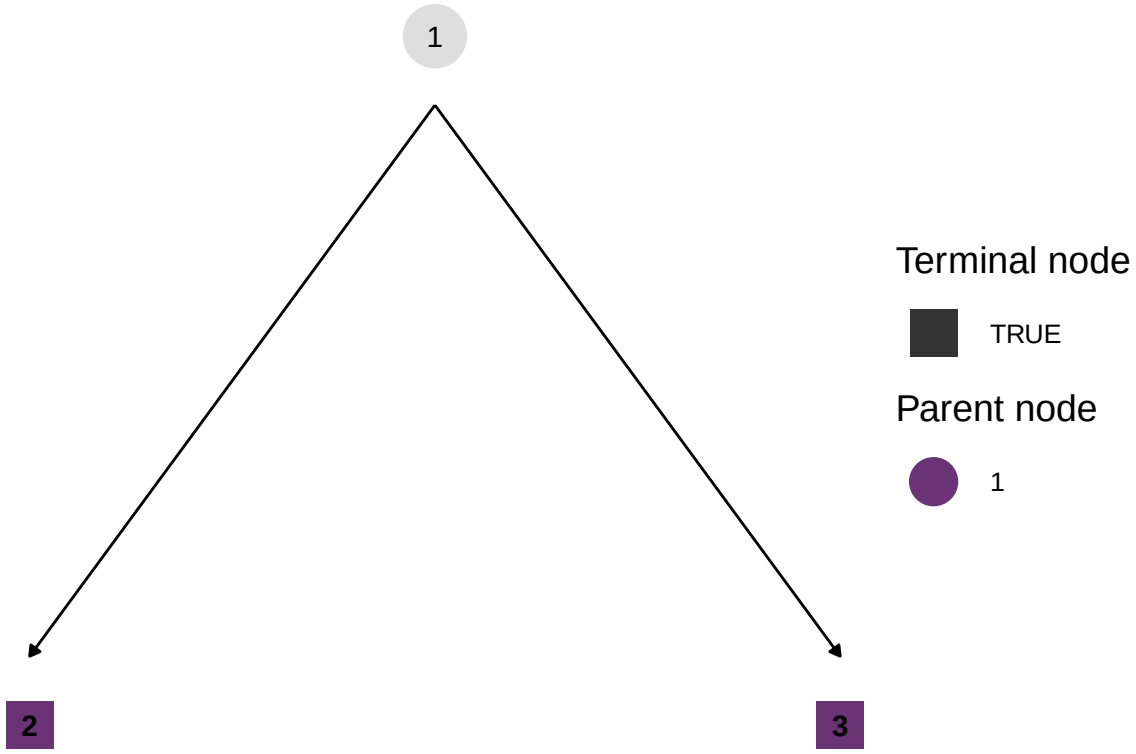
time	obs	u	I95_lower	I95_upper	period
1816	5.69	0.4500	4.808016	6.571984	1
1817	4.56	0.4075	3.761315	5.358685	1
1818	4.72	0.4300	3.877216	5.562785	1
1819	5.08	0.4125	4.271515	5.888485	1
1820	5.12	0.4925	4.154718	6.085282	1
1821	5.40	0.3900	4.635614	6.164386	1

```
# Shift information
knitr::kable(head(results$summary$shift),
```

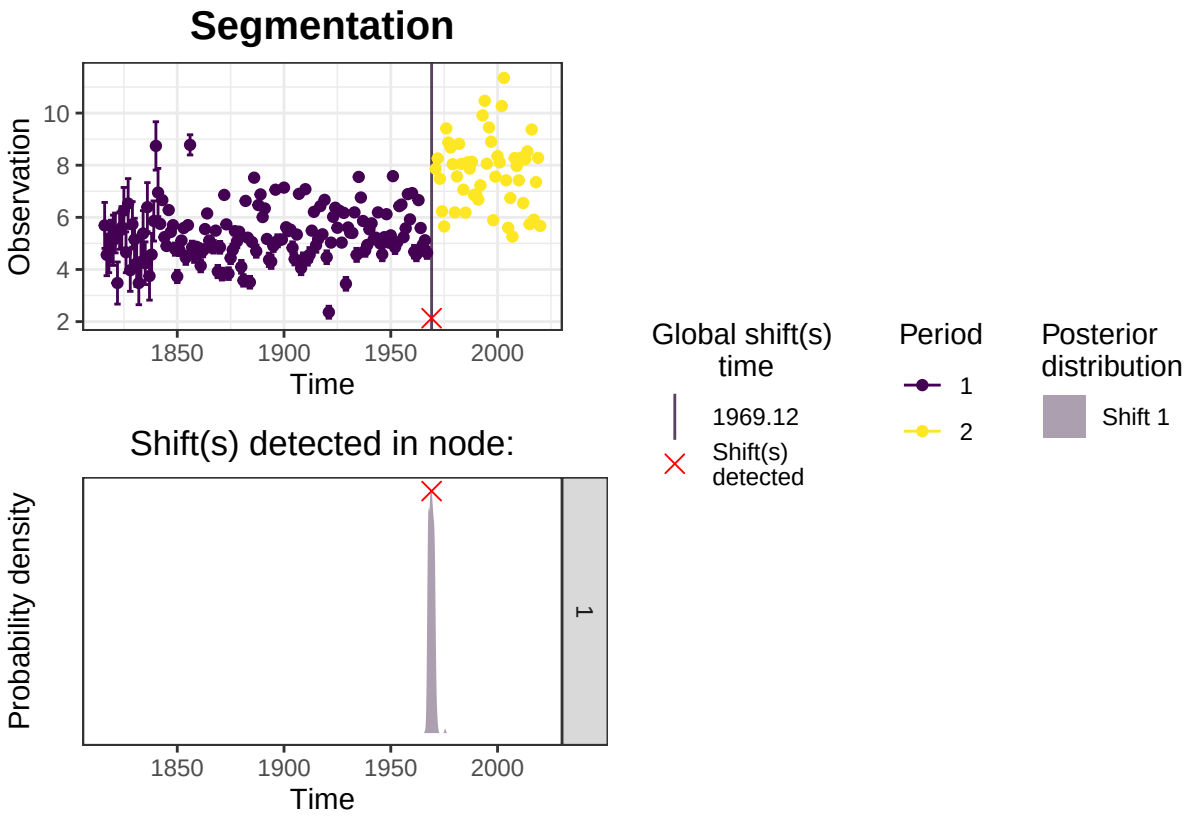
```
              align = 'c',row.names = F)
```

tau	I95_lower	I95_upper	id_iteration
1969.12	1967.09	1971.35	1

```
 # Have a look at recursion tree
 results$tree
#>   indx level parent nS
#> 1    1     1      0  2
#> 2    2     2      1  1
#> 3    3     2      1  1

 # Visualize tree structure
 PlotTree(tree=results$tree)
```




```
 # Final plot segmentation
 PlotSegmentation(summary=results$summary,
                  plot_summary = results$plot)$final_plot
```



Hydrometry field

Detection and segmentation have so far been applied only to the residual of a random variable. However, in hydrometry, a key objective is to predict discharge from stage using a rating curve that may vary over time (shift time).

First, the rating curve approach will be discussed; then, a method for detecting shift times on the rating curve based on the gaugings will be considered; and finally, a method for detecting shift times based on the flood recession will be explained.

Model fitting for the rating curve

Many models are available to describe the rating curve. All fitting models with their equations supported by the package are listed below. For more details, refer to `GetCatalog_RC()` to determine which model could be used to estimate the rating curve.

```
# Get model available to estimate the rating curve
GetCatalog_RC()$models
#> [1] "FitRC_LOESS"                       "FitRC_Exponential"
#> [3] "FitRC_LinearRegression"            "FitRC_SimplifiedBaRatin"
#> [5] "FitRC_SimplifiedBaRatinWithPrior" "FitRC_BaRatinBAC"
#> [7] "FitRC_BaRatinKAC"

# Get equation of each model
GetCatalog_RC()$equations
#> [1] "EquationRC_LOESS"            "EquationRC_BaRatinBAC"
#> [3] "EquationRC_BaRatinKAC"       "EquationRC_Exponential"
```

```
#> [5] "EquationRC_LinearRegression"
```

All these equations Q(h) allow for the proper transformation of stage to discharge, following the specified assumption for each fitting model.

Models can either be non-parametric, such as as `FitRC_LOESS`, which relies solely on data for calculation, or parametric, like `FitRC_BaRatinKAC` with three parameters (a,b,c) per hydraulic control, integrating physics and geometry properties of the river in the estimation process.

Hereafter, the employed model will be an exponential regression (`FitRC_Exponential`) and the BaRatin model for estimating discharge.

The exponential regression needs two parameters, denoted as a and b, following the equation :

$$Q(h) = a \cdot e^{(b \cdot h)}$$

The BaRatin model needs the parameters a, b and c per hydraulic control, following the equation:

$$Q(h) = a \cdot (h - b)^c \quad \text{for } (h > k) \quad (\text{and } Q = 0 \quad \text{if } h \leq b)$$

Dataset

The Ardèche hydrometric station at Meyras is introduced as a new dataset, further information in `?ArdecheRiverGaugings`. The dataset includes stages (H, in meters) and discharge ADCP measurements (Q, in cubic meters per second) all accompanied by uncertainties.

```
knitr::kable(head(ArdecheRiverGaugings),
             align = 'c',row.names = F)
```

Day	Month	Year	Hour	Minute	Second	Date	H	Q	uQ
7	11	2001	16	30	0	2001-11-07 16:30:00	0.17	1.520	0.0532
4	12	2001	14	45	0	2001-12-04 14:45:00	0.10	0.727	0.0254
10	1	2002	14	0	0	2002-01-10 14:00:00	0.06	0.500	0.0175
13	2	2002	16	45	0	2002-02-13 16:45:00	0.08	1.110	0.0389
23	4	2002	17	45	0	2002-04-23 17:45:00	0.17	1.740	0.0609
2	5	2002	13	40	0	2002-05-02 13:40:00	0.22	2.370	0.0829

Recursive model and segmentation procedure for an *unknown* number of segments

This function enables the modeling of the rating curve and ensures its continual update at each segmentation for an **unknown** number of segments. This approach leads to a better fit for the model as it is consistently updated with gauging data from the current period.

Rating curve using exponential regression An exponential regression model is employed to construct the rating curve using observed data point represented by stage and discharge information. This regression estimates the relationships between a dependent variable and one independent variables from a statistical perspective.

```
# Apply recursive model and segmentation
results=ModelAndSegmentation_Gaugings(H=ArdecheRiverGaugings$H,Q=ArdecheRiverGaugings$Q,
                                      time=ArdecheRiverGaugings$Date,
                                      uQ=ArdecheRiverGaugings$uQ,
                                      nSmax=2,
                                      nMin=2,
```

```
                                    funk=FitRC_Exponential)

# Data information
knitr::kable(head(results$summary$data),
             align = 'c',row.names = FALSE)
```

time	H	uH	Q	uQ	Q_I95_lower	Q_I95_upper	Qsim	uQ_simQsim_I95_lower	Qsim_I95_upper	period
2001-11-07 16:30:00	0.17	0	1.520	0.0532	1.41572991.62427013.927984			3.281194 - 2.503038	10.359006 - 2.407984	1
2001-12-04 14:45:00	0.10	0	0.727	0.0254	0.67721690.77678313.462623			3.281194 - 2.968400	9.893644 - 2.735622	1
2002-01-10 14:00:00	0.06	0	0.500	0.0175	0.46570060.53429943.221893			3.281194 - 3.209129	9.652915 - 2.721893	1
2002-02-13 16:45:00	0.08	0	1.110	0.0389	1.03375741.18624263.340089			3.281194 - 3.090933	9.771111 - 2.230090	1
2002-04-23 17:45:00	0.17	0	1.740	0.0609	1.62063821.85936183.927984			3.281194 - 2.503038	10.359006 - 2.187984	1
2002-05-02 13:40:00	0.22	0	2.370	0.0829	2.20751902.53248104.298207			3.281194 - 2.132815	10.729229 - 1.928207	1

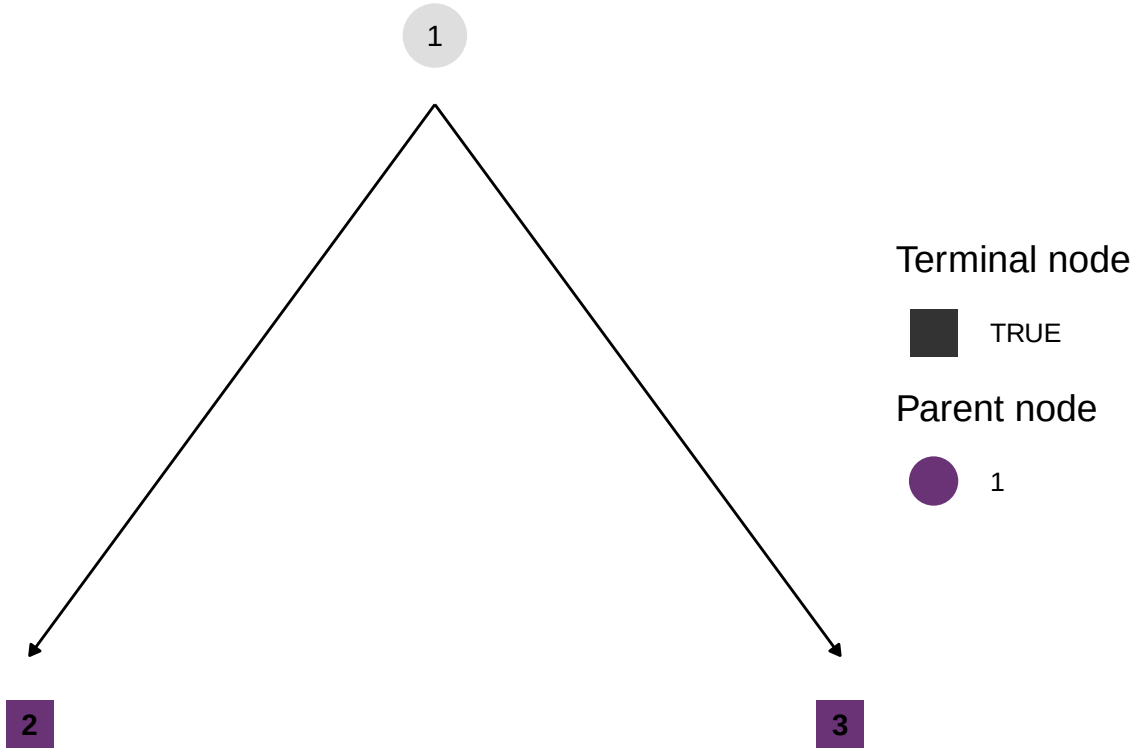
```
# Shift information
knitr::kable(head(results$summary$shift),
            align = 'c',row.names = FALSE)
```

tau	I95_lower	I95_upper	id_iteration
2008-09-29 22:44:24	2006-11-17 21:03:57	2009-10-01 22:51:14	1

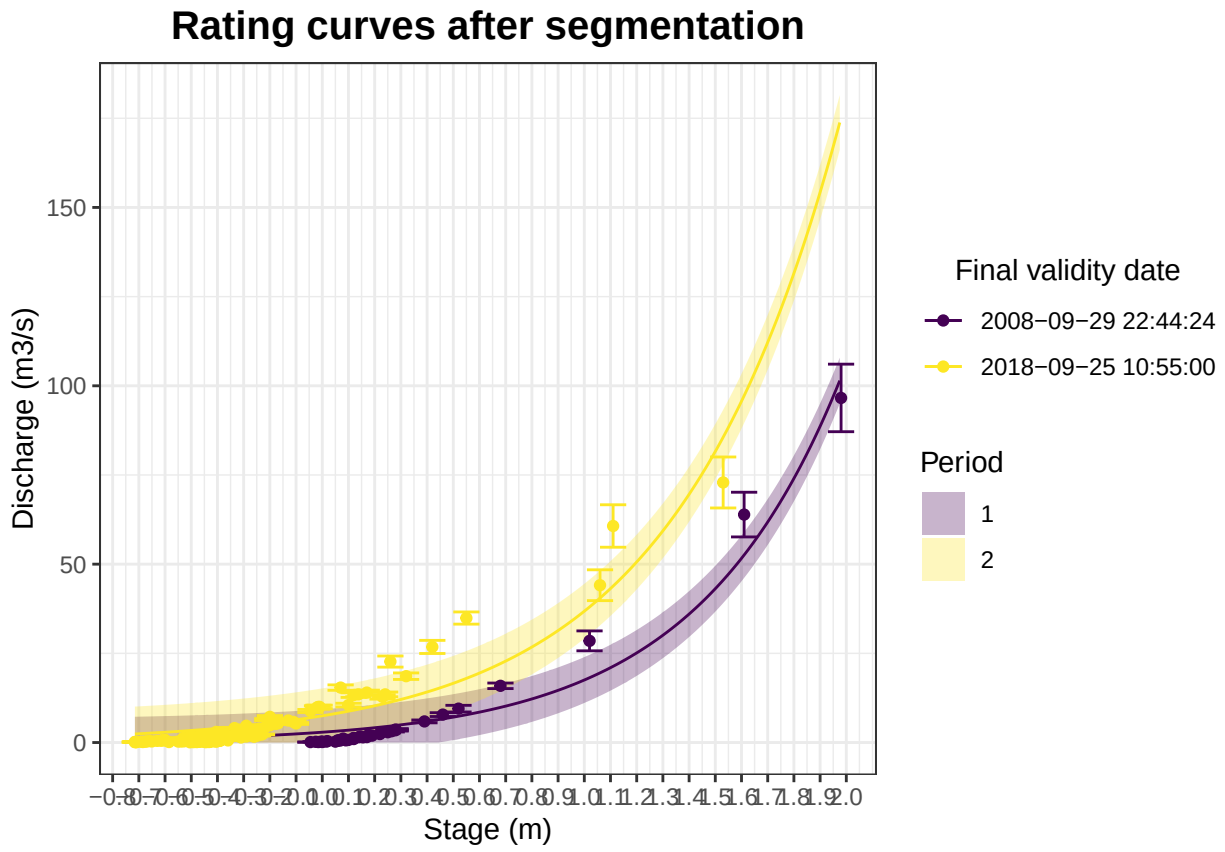
```
# Parameters estimation of the rating curve
results$summary$param.equation
#>          a        b
#> 1 2.891811 1.801431
#> 2 7.491781 1.591981

# Have a look at recursion tree
results$tree
#>   indx level parent nS
#> 1    1     1      0  2
#> 2    2     2      1  1
#> 3    3     2      1  1
```

Several plot functions are available to simplify graphical representation, aligning with the structure of specific functions integrated in the package:

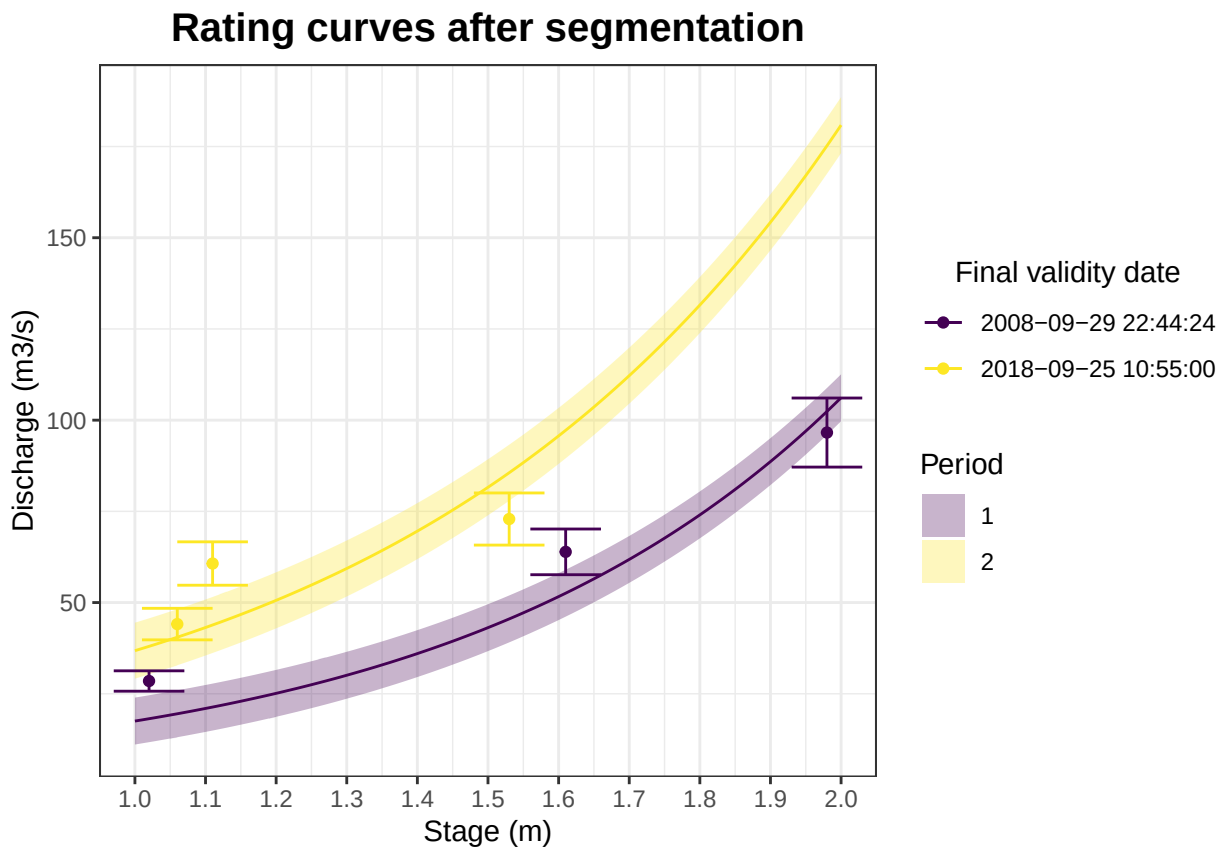
```
# Visualize tree structure
PlotTree(results$tree)
```



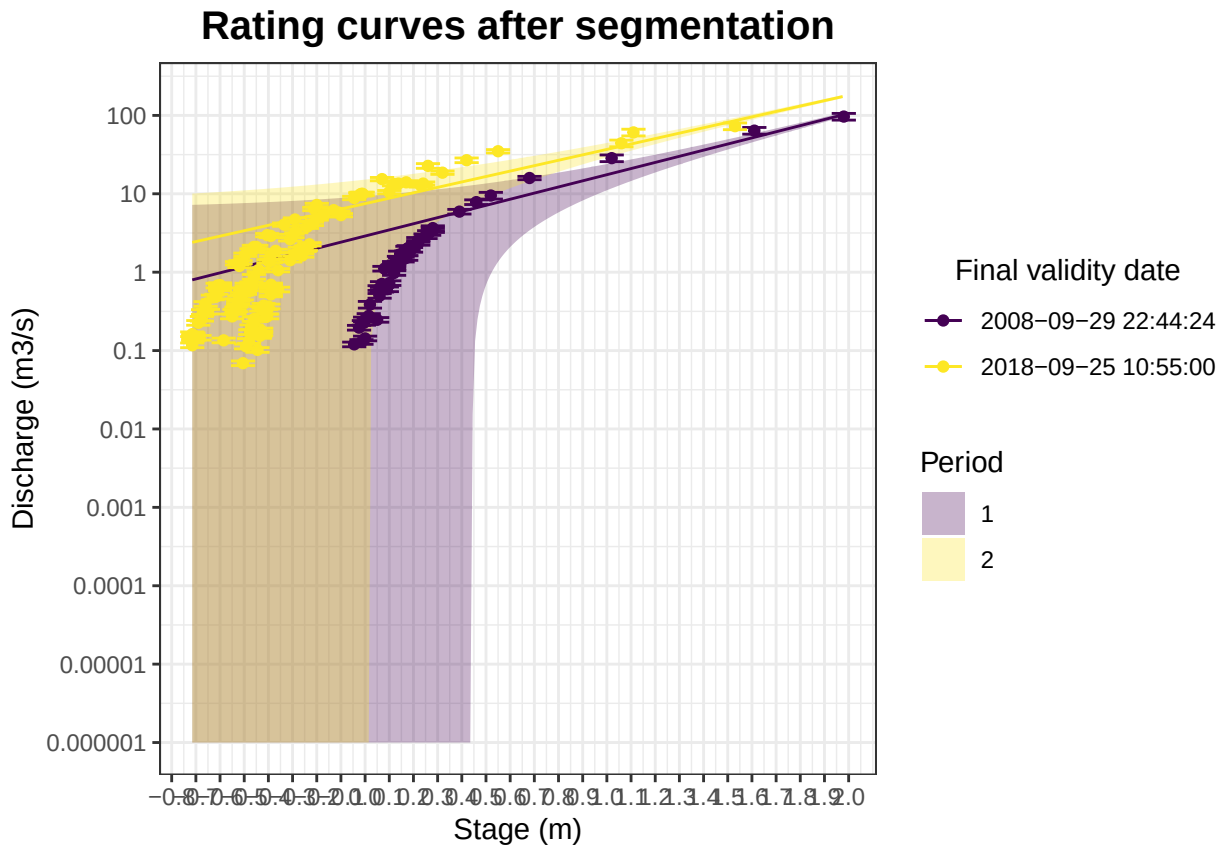

```
# parameter of the rating curve
a=results$summary$param.equation$a
b=results$summary$param.equation$b

# Plot the rating curve after segmentation following a regression exponential
PlotRCSegmentation_Gaugings(summary=results$summary,
                            equation = EquationRC_Exponential,
                            a=a,
                            b=b)
```



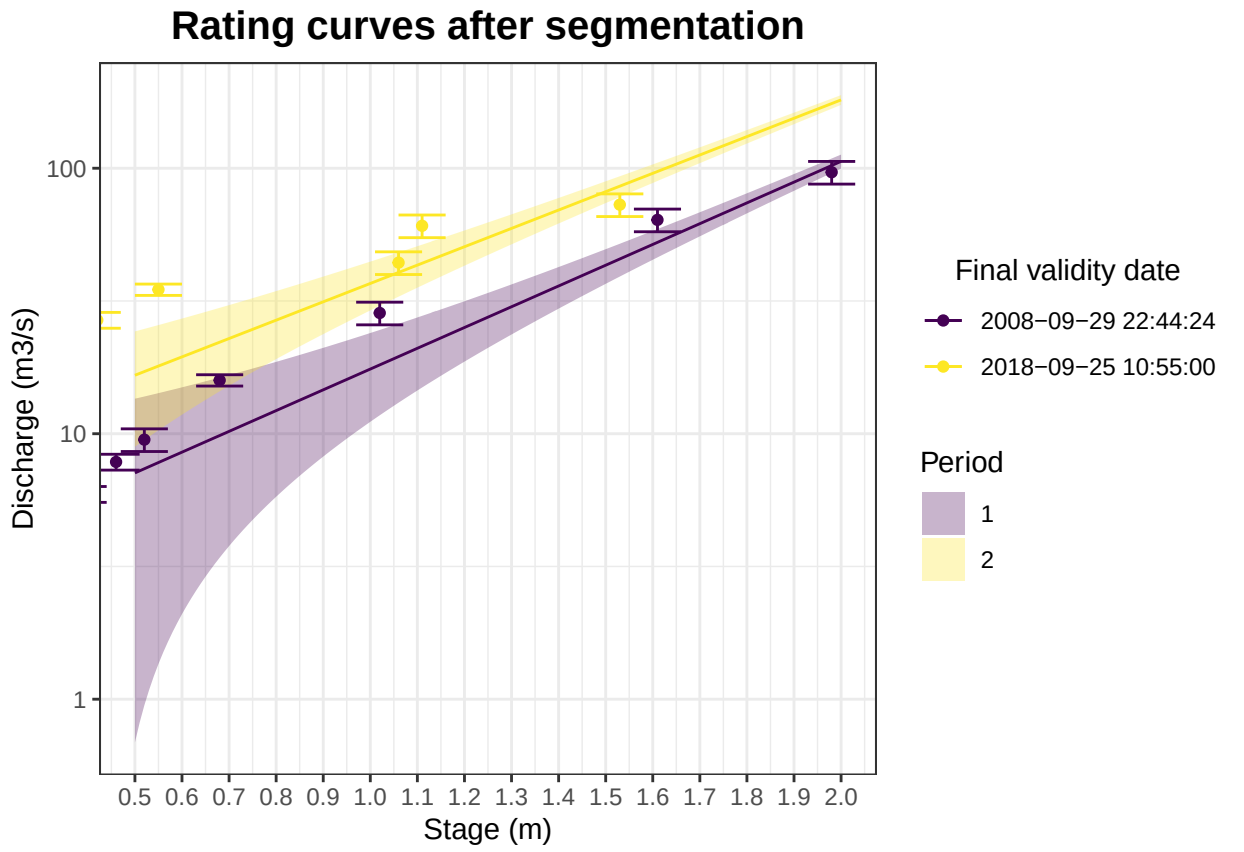

```
# Plot the rating curves after segmentation with zoom user-defined
PlotRCSegmentation_Gaugings(summary=results$summary,
                            equation = EquationRC_Exponential,
                            a=a,
                            b=b,
                            autoscale = FALSE,
                            Hmin_user = 1,
                            Hmax_user = 2,
                            H_step_discretization = 0.01)
```



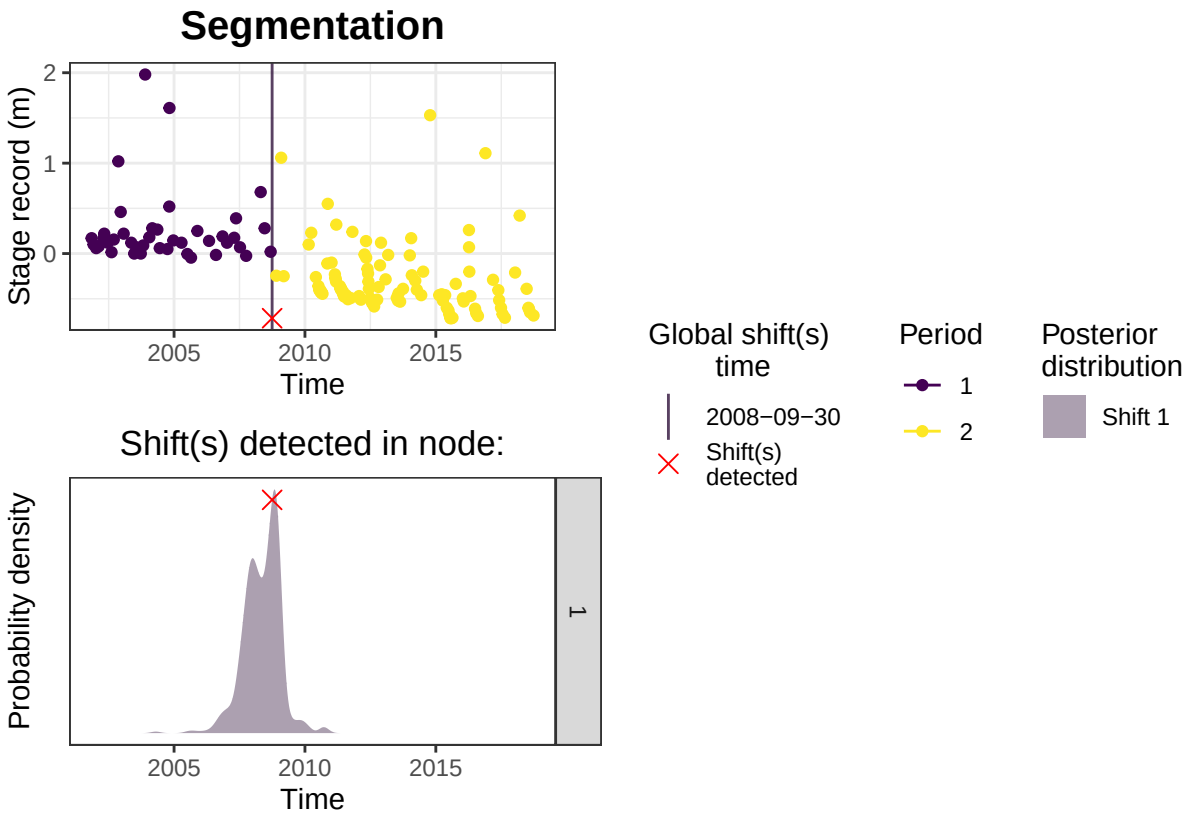

```
# Plot the rating curves after segmentation in log scale
PlotRCSegmentation_Gaugings(summary=results$summary,
                            logscale=TRUE,
                            equation = EquationRC_Exponential,
                            a=a,
                            b=b)
```



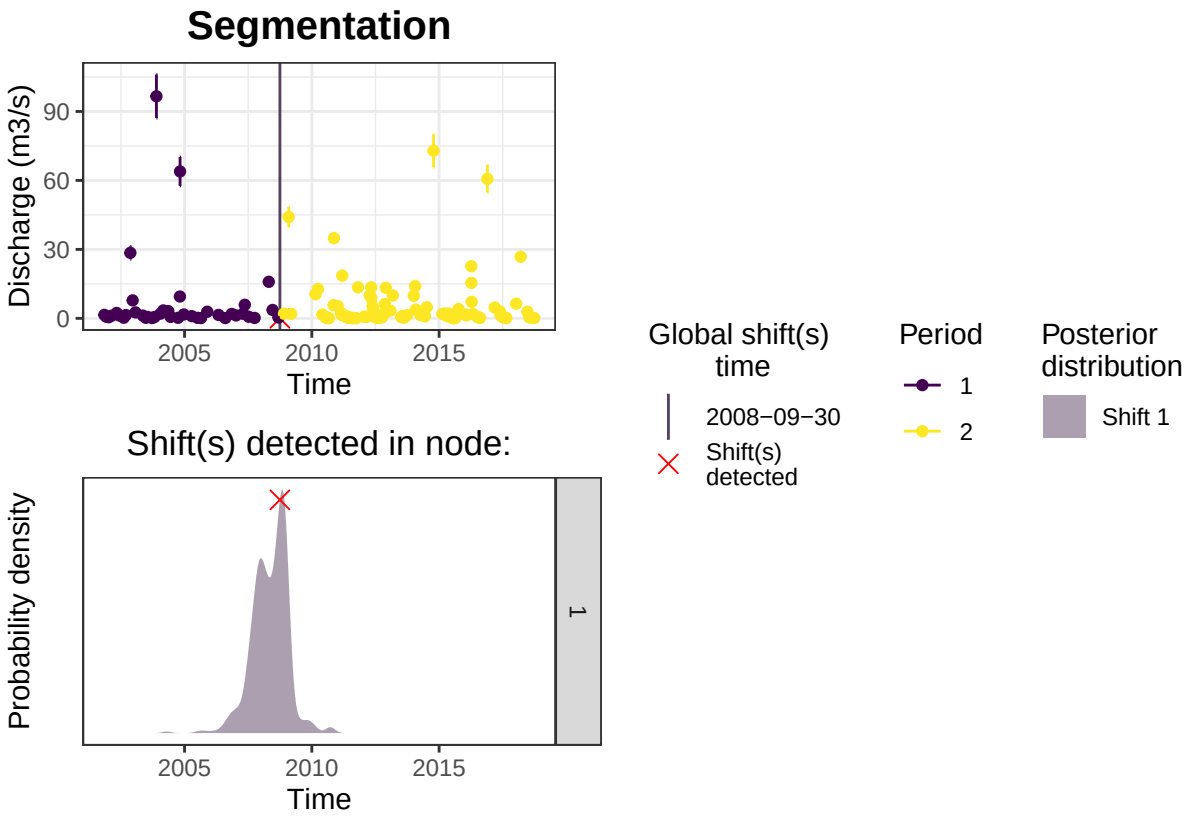

```
# Plot the rating curves after segmentation in log scale with zoom
PlotRCSegmentation_Gaugings(summary=results$summary,
                            logscale=TRUE,
                            equation = EquationRC_Exponential,
                            a=a,
                            b=b,
                            autoscale = FALSE,
                            Hmin_user = 0.5,
                            Hmax_user = 2,
                            H_step_discretization = 0.01)
```



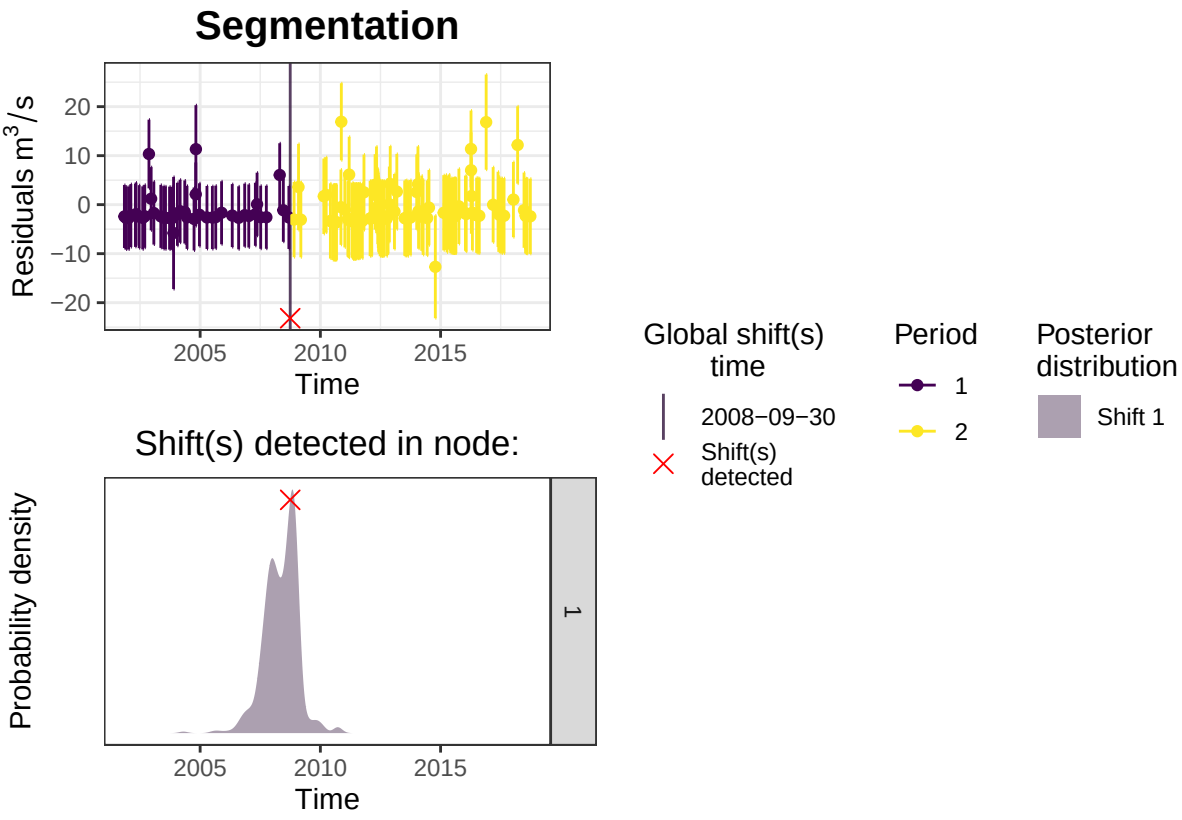

```
# Plot shift times in stage record
PlotSegmentation_Gaugings_Hdt(summary=results$summary,
                              plot_summary=results$plot)$final_plot
```




```
# Plot shift times in discharge observations
PlotSegmentation_Gaugings_Qdt(summary=results$summary,
                              plot_summary=results$plot)$final_plot
```




```
# Plot residual
PlotSegmentation_Gaugings_Residual(summary=results$summary,
                                   plot_summary=results$plot)$final_plot
```



Rating curve using BaRatin method The Bayesian BaRatin method (Bayesian Rating curve, Le Coz et al. (2014); Horner et al. (2018)) has been developed at INRAE.

The method combines the strength of a probabilistic approach (parameter estimation and uncertainty quantification) and a physically-based approach (physical interpretation of parameters and of their change when a rating shift occurs, more reliable extrapolation). BaRatin provides a way for field hydrologists to simply formalize and make use of the expertise they do have on flows at their gauging stations, along with gaugings and their uncertainties.

In this package, the methodology of BaRatin will not been explained in detail. Instead, hydraulic control matrix and prior information on parameters will be used to run the segmentation and estimation of the rating curve.

Hydraulic analysis Firstly, a hydraulic analysis of the gauging station is required to set the hydraulic control matrix. The Ardèche River at Meyras station is of interest because it illustrates a frequently-encountered 3-control configuration (riffle, main channel, floodway) and it has been already studied by Le Coz et al. (2014) and Mansanarez et al. (2016) .

At low flows, the stage-discharge relation is controlled by the geometry of a critical section induced by a natural riffle.

As stage increases, the riffle becomes drowned and the stage-discharge relation is controlled by the geometry and roughness of the main channel.

At high flows, part of the water flows into two floodways located on the right and left banks. Since the two floodways get activated at roughly the same stage, they are combined into a single control.

To assist the user in entering the hydraulic control matrix, the `Builder_ControlMatrix` function was

developed. This function interacts with the user to facilitate the creation of the matrix.

```
# Hydraulic control matrix
controlMatrix=matrix(c(1,0,0,0,1,1,0,0,1),ncol=3,nrow=3)
```

Prior information The method required prior information on the parameters a, k and c per hydraulic control following this equation:

$$Q(h) = a * (h - b)^c \quad \text{for } (h > k) \quad (\text{and } Q = 0 \quad \text{if } h \leq b)$$

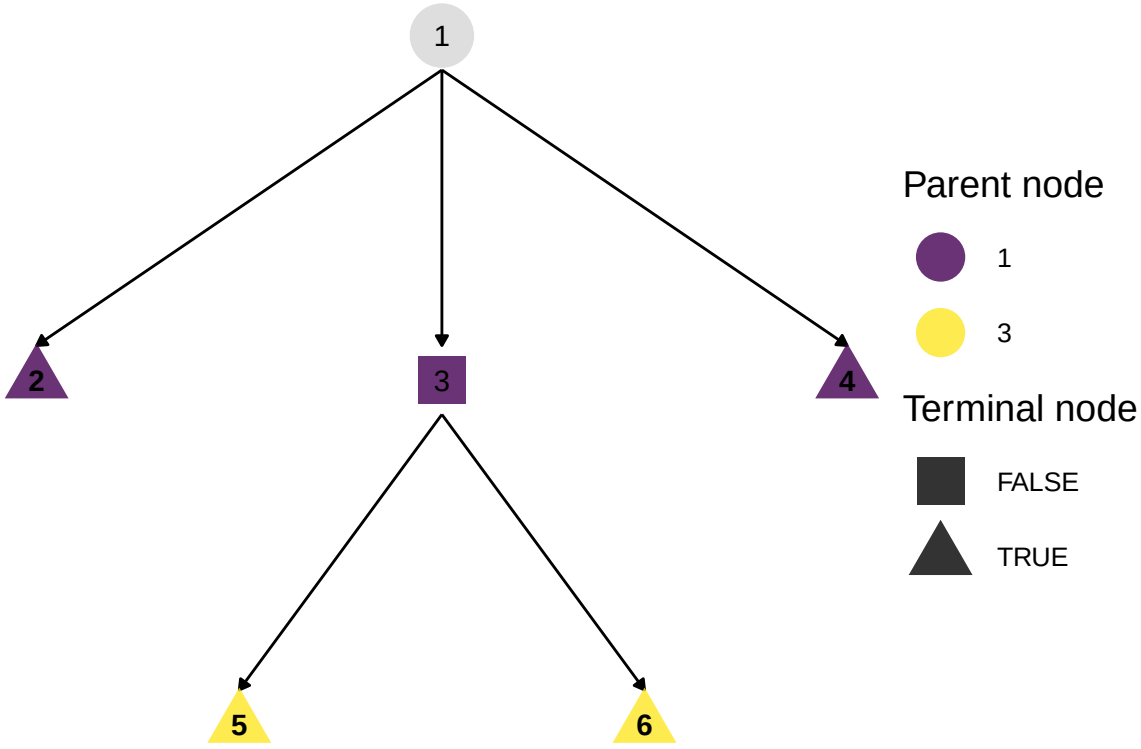
- Parameter a is the coefficient representing the geometry and physical properties of the control. It will be estimated differently depending on the type of control.
- Parameter b is the offset; when stage falls below the value b, discharge is zero.
- Parameter c is the exponent, which depends solely on the type of control.
- Parameter k is the activation stage; when the water level falls below the value k, the control becomes inactive.

See the details of the values entered here: prior specification for the case of study of Ardeche at Meyras gauging station.

```
# Set prior information to each hydraulic control
a1=RBaM::parameter(name='a1',init=14.17,prior.dist='Gaussian',prior.par=c(14.17,3.65))
k1=RBaM::parameter(name='k1',init=-0.6,prior.dist='Gaussian',prior.par=c(-0.6,0.5))
c1=RBaM::parameter(name='c1',init=1.5,prior.dist='Gaussian',prior.par=c(1.5,0.025))
a2=RBaM::parameter(name='a2',init=26.5,prior.dist='Gaussian',prior.par=c(26.5,8.4))
k2=RBaM::parameter(name='k2',init=0,prior.dist='Gaussian',prior.par=c(0,0.5))
c2=RBaM::parameter(name='c2',init=1.67,prior.dist='Gaussian',prior.par=c(1.67,0.025))
a3=RBaM::parameter(name='a3',init=31.82,prior.dist='Gaussian',prior.par=c(31.8,10.9))
k3=RBaM::parameter(name='k3',init=1.2,prior.dist='Gaussian',prior.par=c(1.2,0.2))
c3=RBaM::parameter(name='c3',init=1.67,prior.dist='Gaussian',prior.par=c(1.67,0.025))

# Set a list of the same parameters for all controls
a.object=list(a1,a2,a3)
k.object=list(k1,k2,k3)
c.object=list(c1,c2,c3)
```

The `Builder_Prior_Knowledge` function was developed and available to help the user to create these objects in a interactive way.

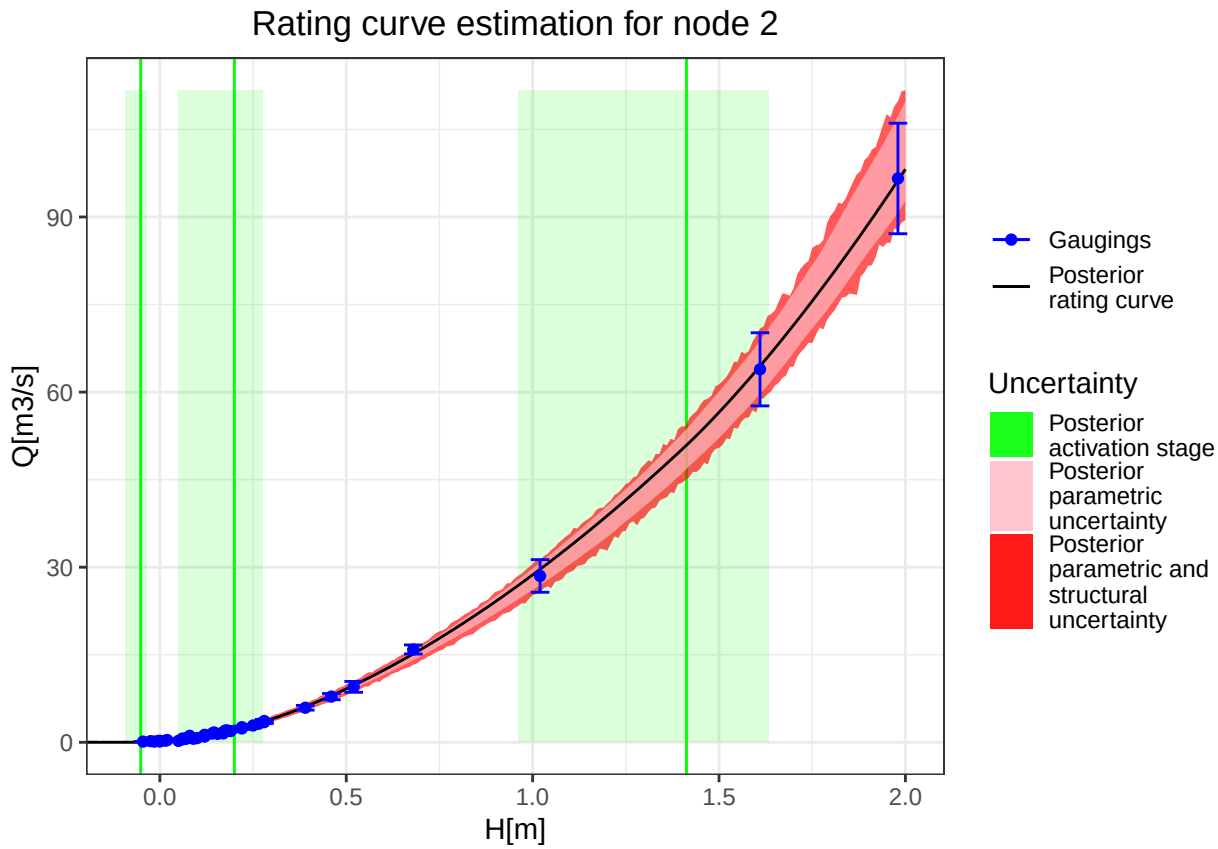
```
# Apply recursive model and segmentation with BaRatin multi-control method
resultsBaRatin=ModelAndSegmentation_Gaugings(H=ArdecheRiverGaugings$H,
                                             Q=ArdecheRiverGaugings$Q,
                                             time=ArdecheRiverGaugings$Date,
                                             uQ=ArdecheRiverGaugings$uQ,
                                             nSmax=3,
                                             nMin=2,
                                             funk=FitRC_BaRatinKAC,
                                             a.object=a.object,
                                             k.object=k.object,
                                             c.object=c.object,
                                             controlMatrix=controlMatrix)

 # Visualize tree structure
 PlotTree(resultsBaRatin$tree)
```



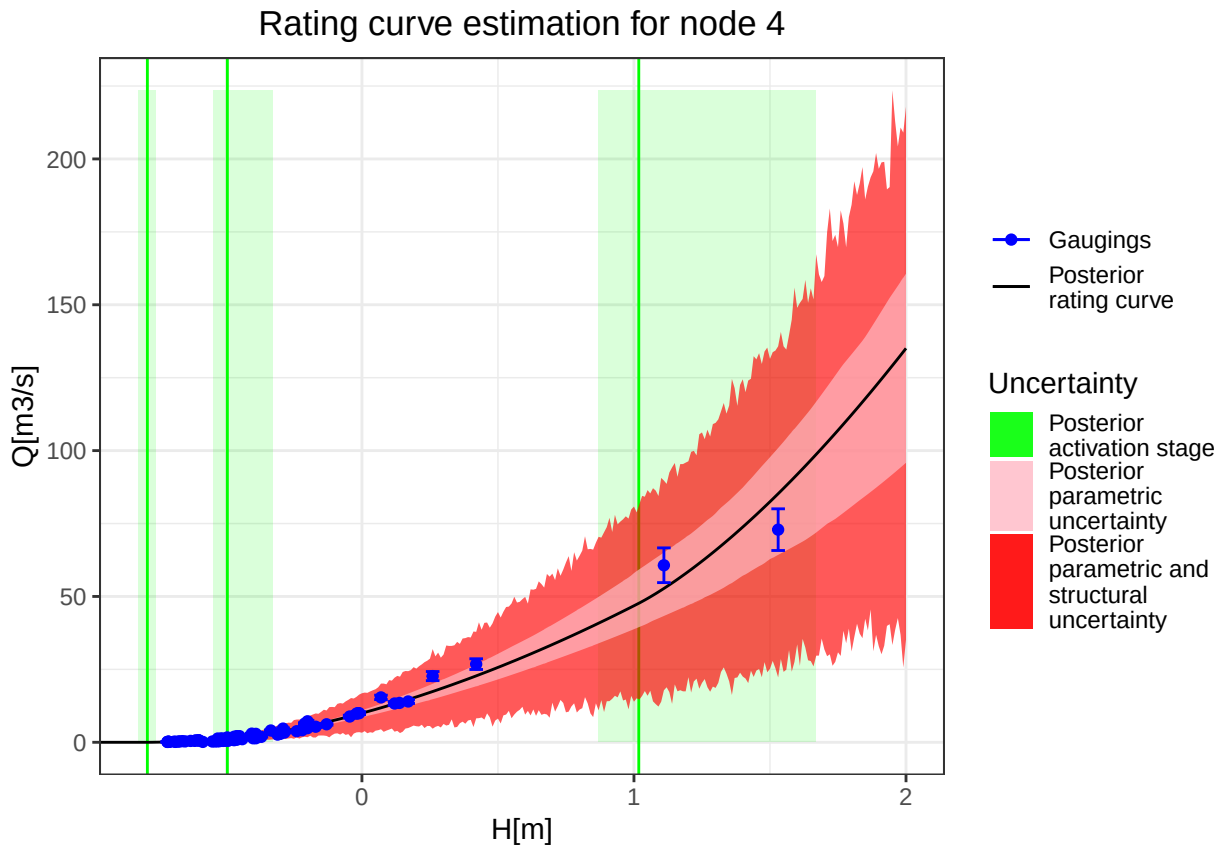

```
 # Terminal nodes
 terminal = resultsBaRatin$tree$indx[which(resultsBaRatin$tree$nS==1)]
 terminal
#> [1] 2 4 5 6
```

Plot the rating curves after using BaRatin method. It is possible to plot all nodes in the tree structure by setting `allnodes=TRUE`. To reduce calculation time, it is advisable to specify the vector of nodes for plotting the rating curve.

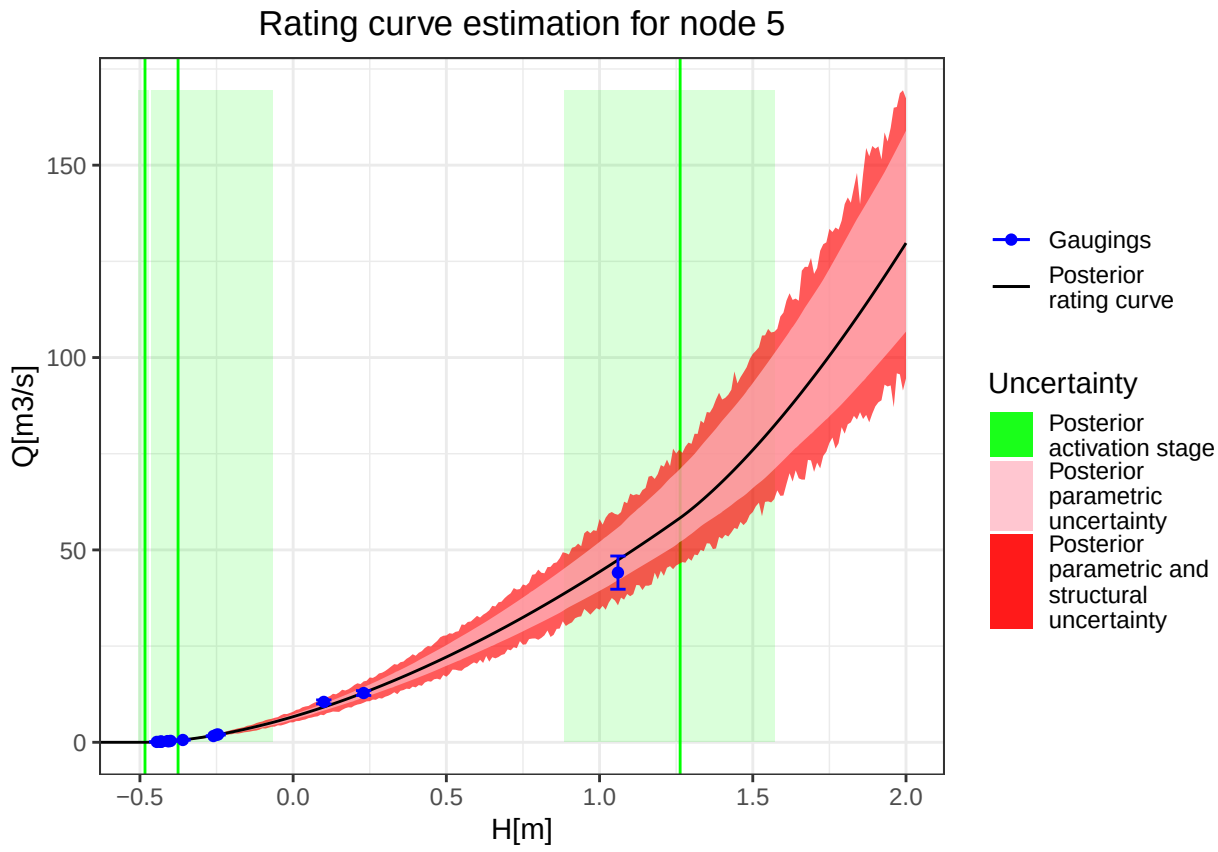
```
 # Plot the rating curves after using BaRatin method
 plotsRC=PlotRCSegmentation_Gaugings_Tree(Hgrid=data.frame(seq(-1,2,by=0.01)),
                                          autoscale=FALSE,
                                          temp.folder=file.path(tempdir(),'BaM'),
                                          CalibrationData='CalibrationData.txt',
                                          allnodes=FALSE,
                                          nodes=terminal)

 plotsRC$PlotRCPred
#> [[1]]
```


Rating curve estimation for node 2
Q[m3/s]
H[m]
0
30
60
90
0.0
0.5
1.0
1.5
2.0
Gaugings
Posterior rating curve
Uncertainty
Posterior activation stage
Posterior parametric uncertainty
Posterior parametric and structural uncertainty

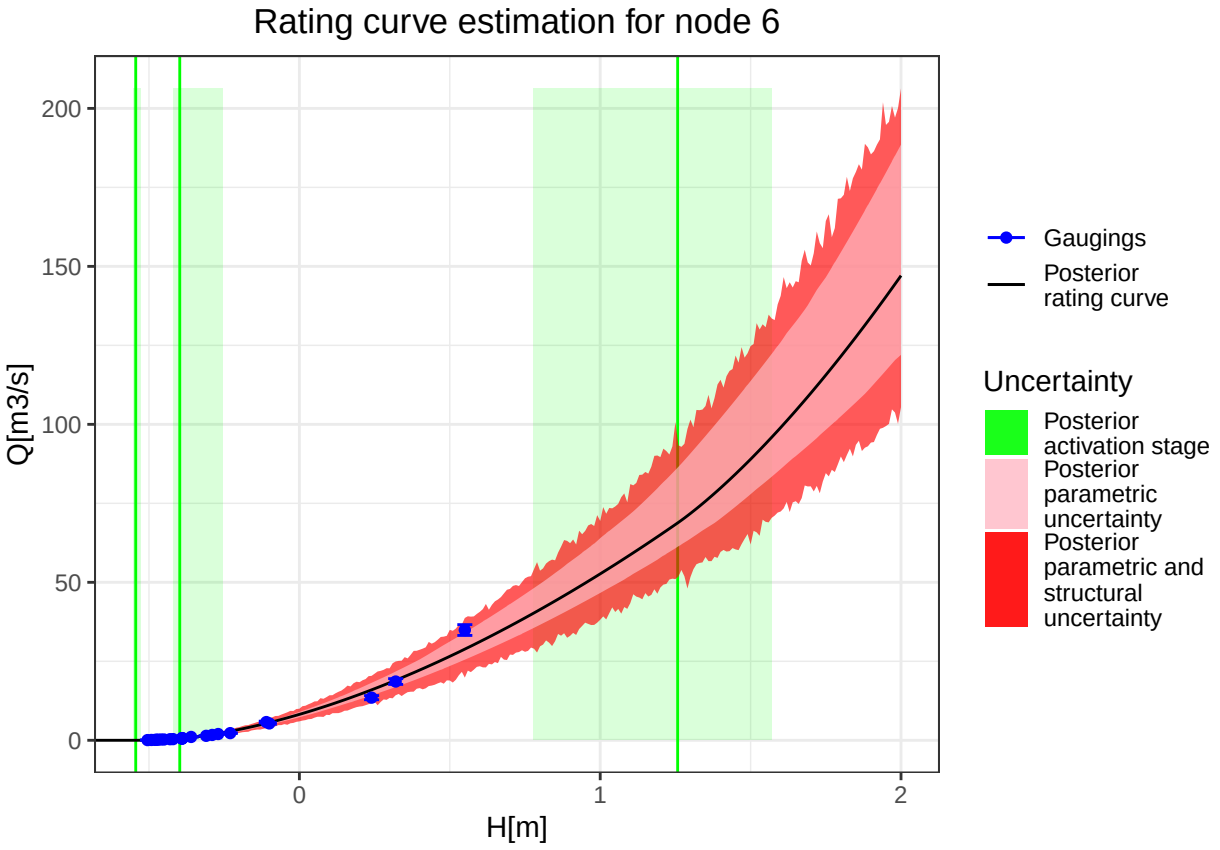
```
#>
#> [[2]]
```


Rating curve estimation for node 4
Q[m3/s]
H[m]
200
150
100
50
0
0
1
2
Gaugings
Posterior rating curve
Uncertainty
Posterior activation stage
Posterior parametric uncertainty
Posterior parametric and structural uncertainty

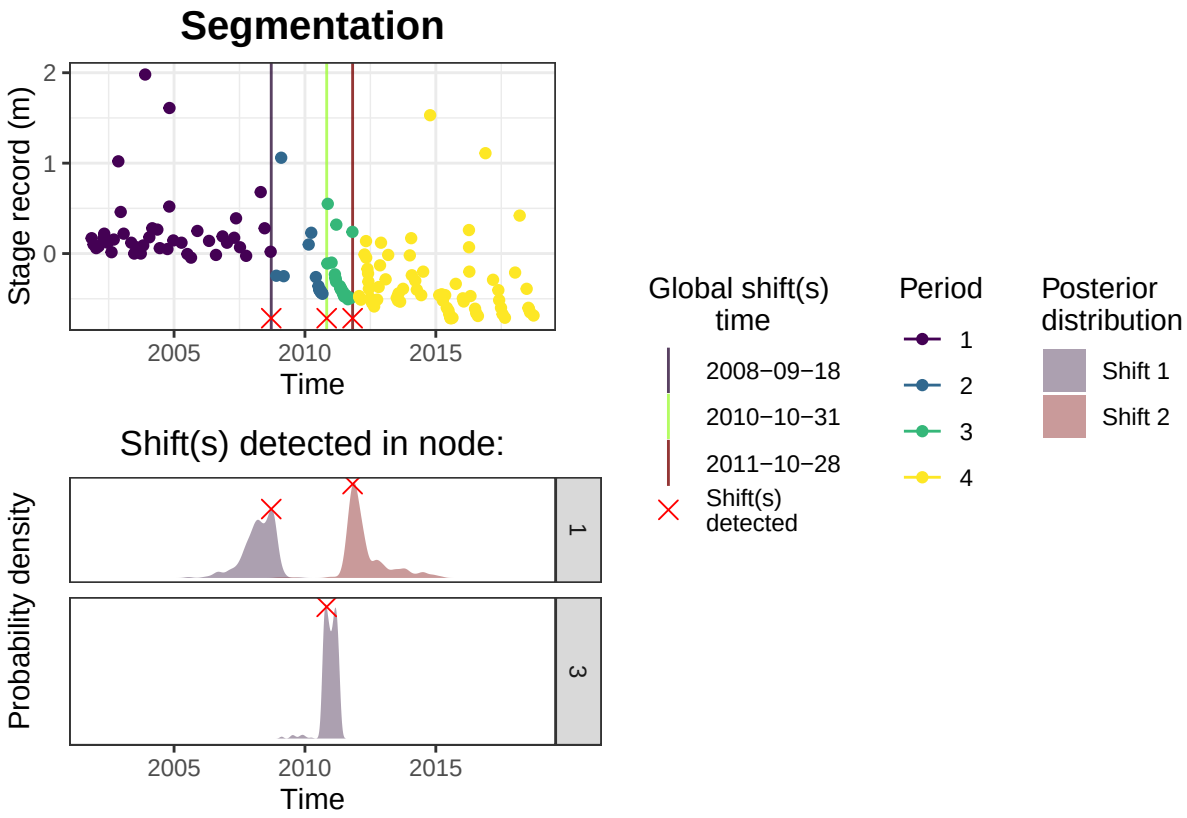
```
#>
#> [[3]]
```


Rating curve estimation for node 5
Q[m3/s]
H[m]
150
100
50
0
−0.5
0.0
0.5
1.0
1.5
2.0
Gaugings
Posterior rating curve
Uncertainty
Posterior activation stage
Posterior parametric uncertainty
Posterior parametric and structural uncertainty

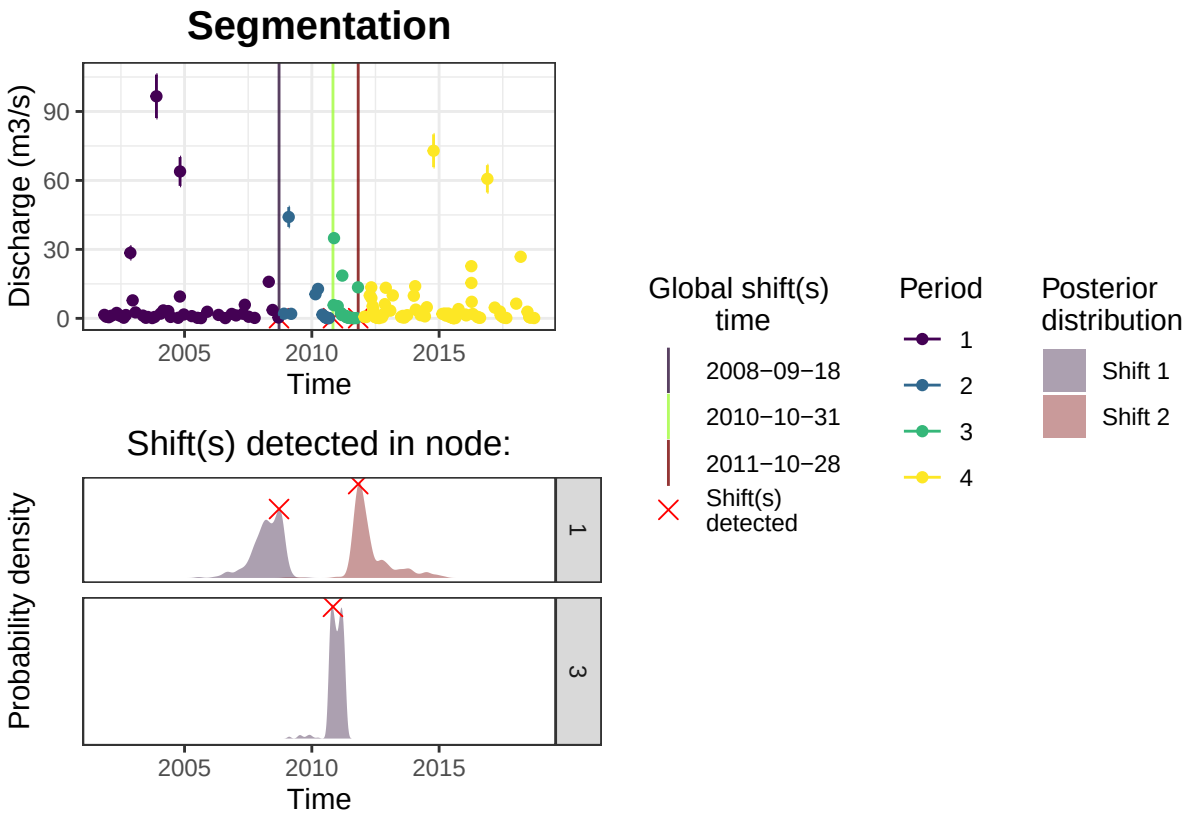
```
#>
#> [[4]]
```



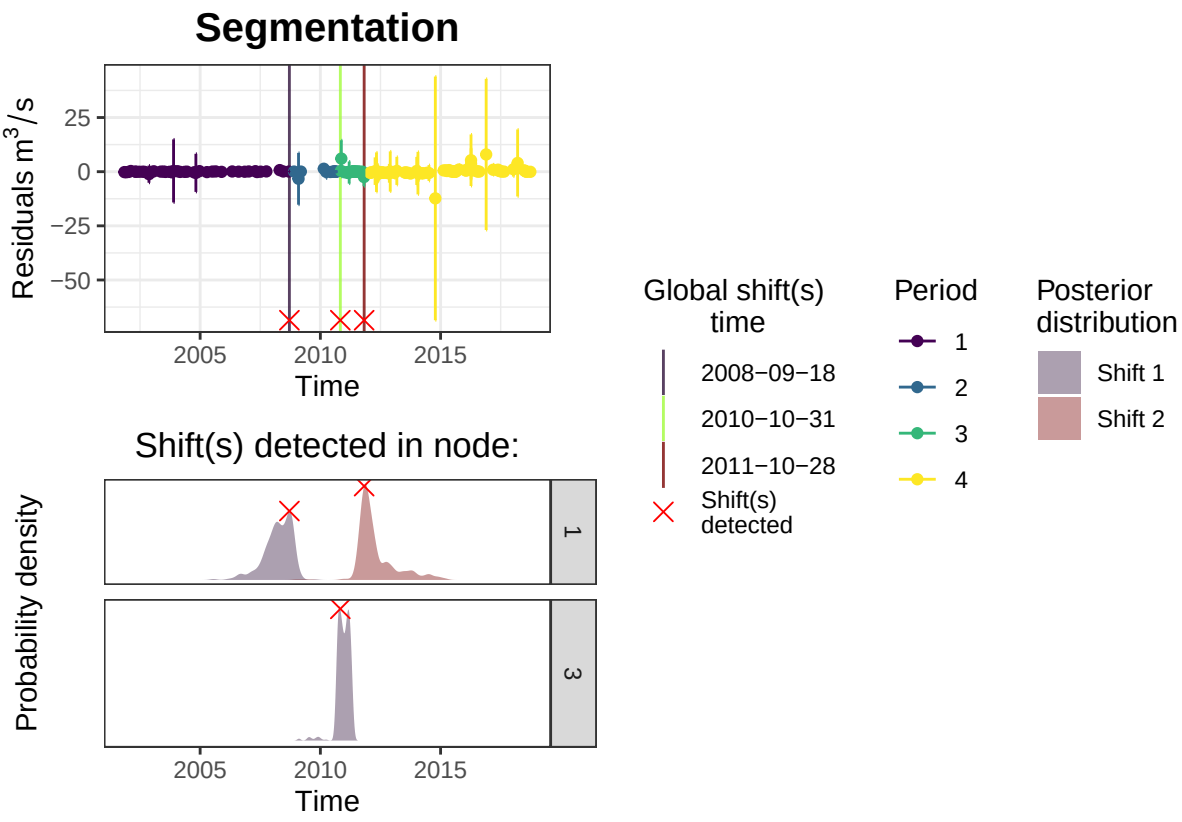

```
# Plot shift times in stage record
PlotSegmentation_Gaugings_Hdt(summary=resultsBaRatin$summary,
                              plot_summary=resultsBaRatin$plot)$final_plot
```




```
# Plot shift times in discharge observations
PlotSegmentation_Gaugings_Qdt(summary=resultsBaRatin$summary,
                              plot_summary=resultsBaRatin$plot)$final_plot
```

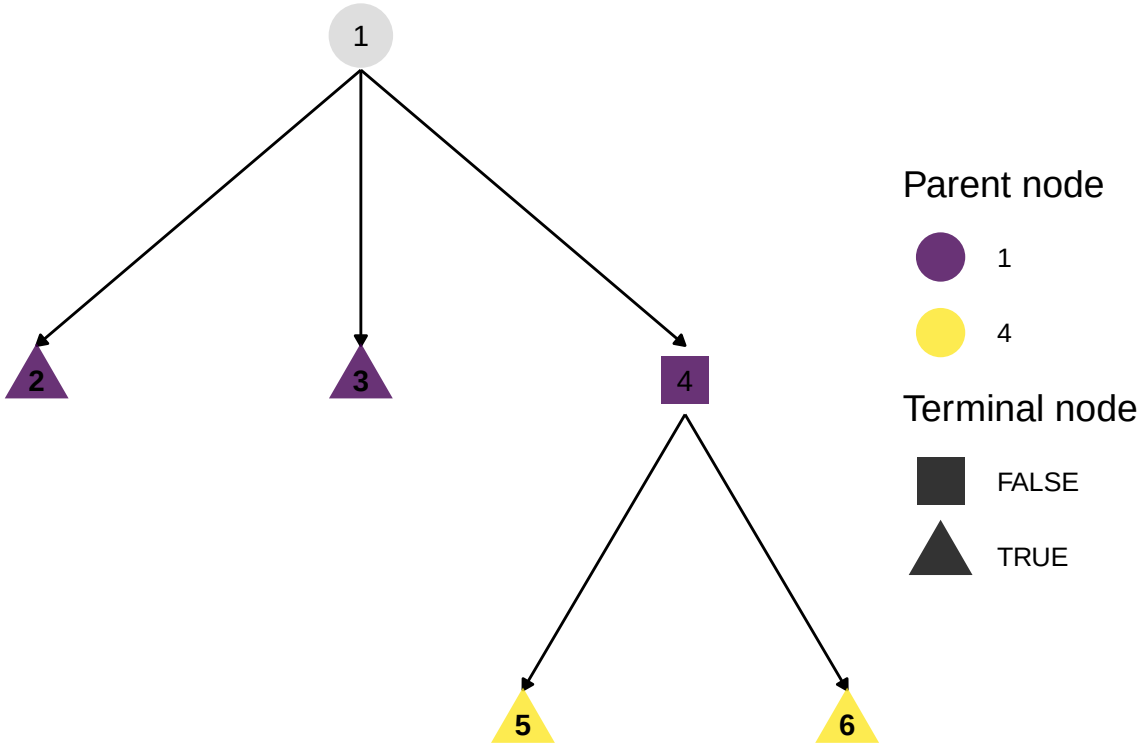



```
# Plot residual
PlotSegmentation_Gaugings_Residual(summary=resultsBaRatin$summary,
                                   plot_summary=resultsBaRatin$plot)$final_plot
```

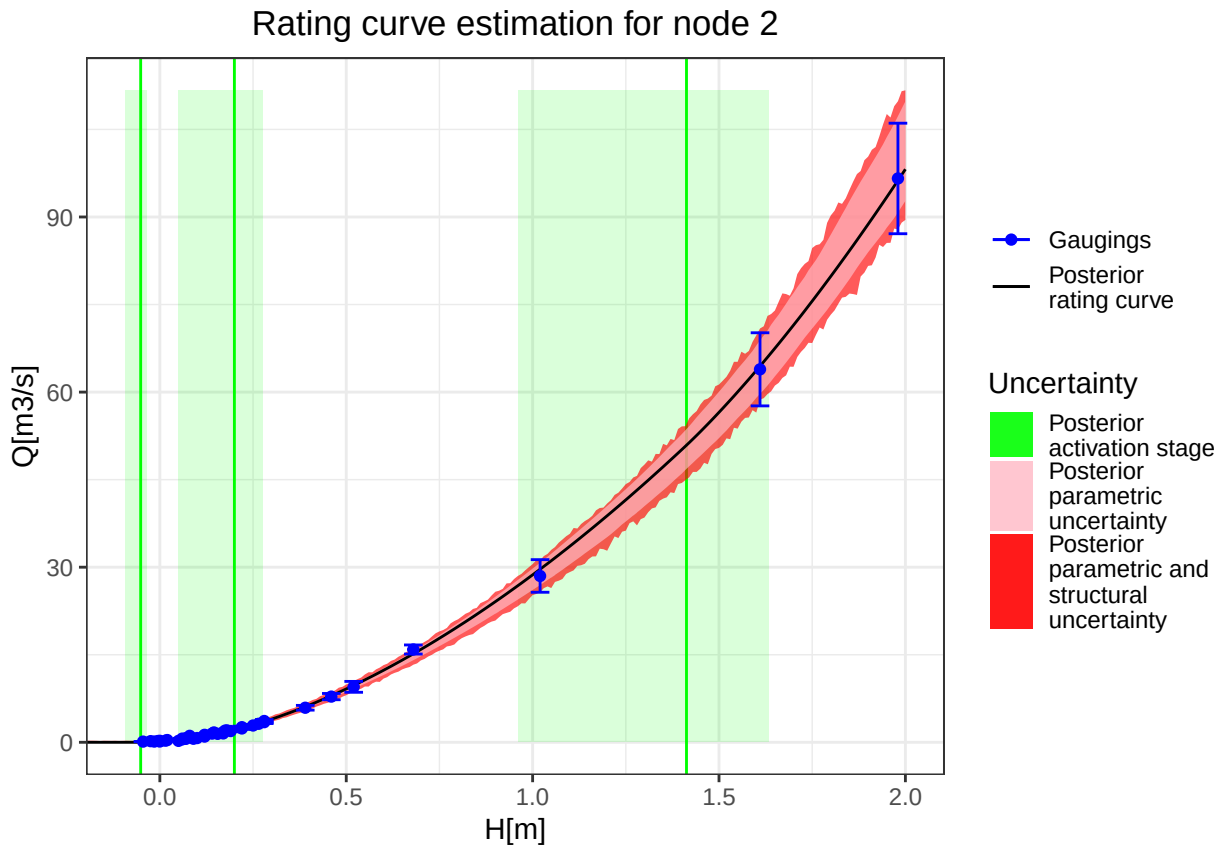


A particular case: the impact of vegetation Uncertainty associated to water level measurements during low flows could lead to a very high uncertainty when comparing with the rating curve, due to the low sensitivity of control during low flows. As a result, it was often necessary to be even more cautious when deciding to change the curve based on these measurements. That's why, it is possible to account for the uncertainty in the gauged height, which increases the overall uncertainty when estimating the uncertainty of the residuals to compensate for the issue identified.

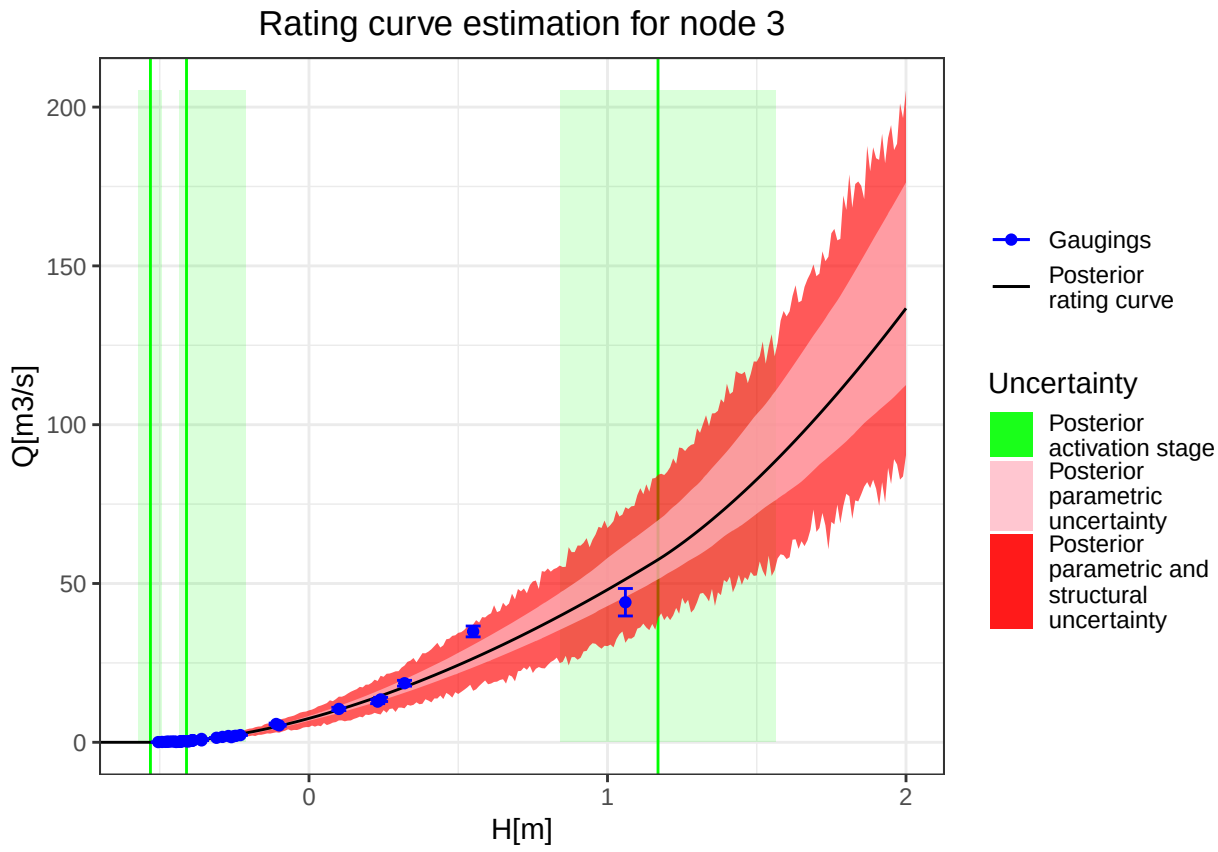
```
# Apply recursive model and segmentation with BaRatin multi-control method. Assumption, 0.05 m for unce
resultsBaRatinWithuH=ModelAndSegmentation_Gaugings(H=ArdecheRiverGaugings$H,
                                                   Q=ArdecheRiverGaugings$Q,
                                                   time=ArdecheRiverGaugings$Date,
                                                   uQ=ArdecheRiverGaugings$uQ,
                                                   uH=rep(0.05,length(ArdecheRiverGaugings$H)),
                                                   nSmax=3,
                                                   nMin=2,
                                                   funk=FitRC_BaRatinKAC,
                                                   a.object=a.object,
                                                   k.object=k.object,
                                                   c.object=c.object,
                                                   controlMatrix=controlMatrix)

 # Visualize tree structure
 PlotTree(resultsBaRatinWithuH$tree)
```



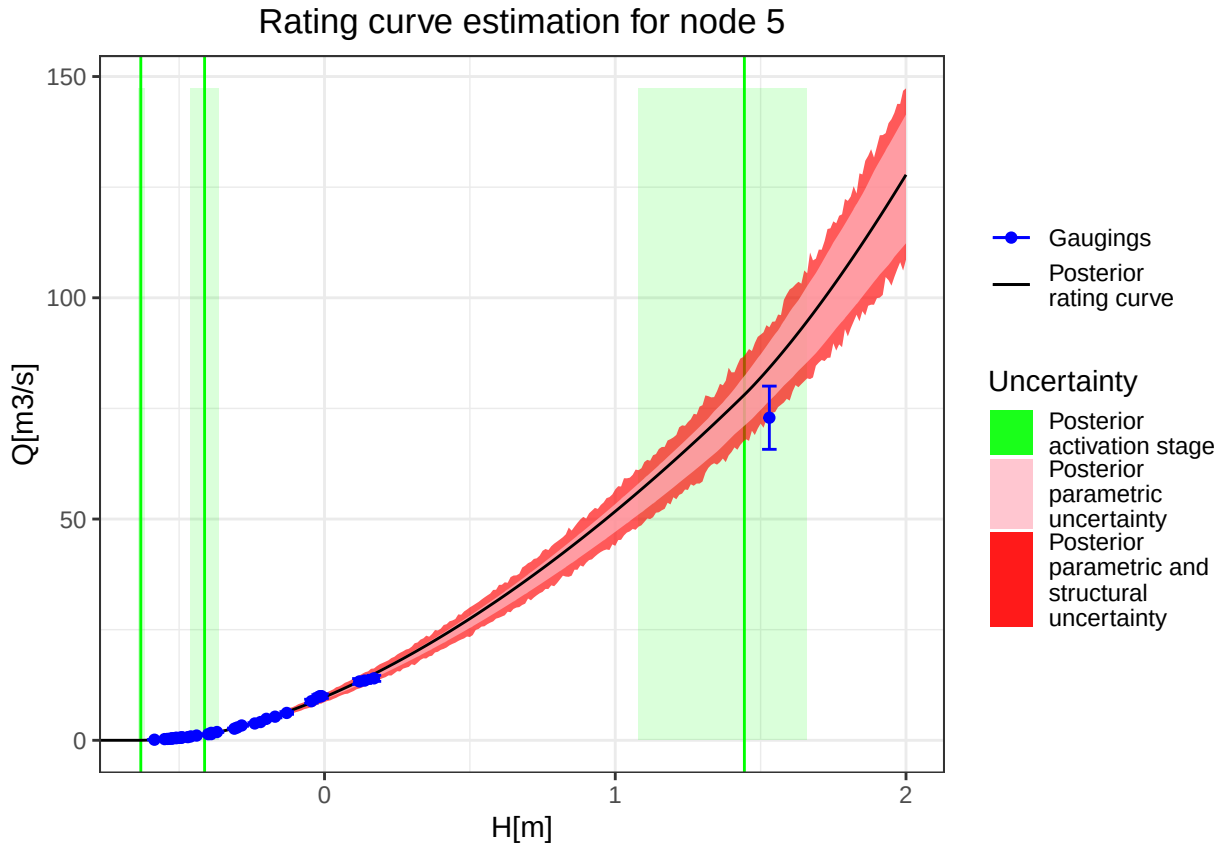

```
 # Terminal nodes
 terminal = resultsBaRatinWithuH$tree$indx[which(resultsBaRatinWithuH$tree$nS==1)]
 terminal
#> [1] 2 3 5 6

 plotsRC= PlotRCSegmentation_Gaugings_Tree(Hgrid=data.frame(seq(-1,2,by=0.01)),
                                           autoscale=FALSE,
                                           temp.folder=file.path(tempdir(),'BaM'),
                                           CalibrationData='CalibrationData.txt',
                                           allnodes=FALSE,
                                           nodes=terminal)

 plotsRC$PlotRCPred
#> [[1]]
```


Rating curve estimation for node 2
Q[m3/s]
H[m]
0
30
60
90
0.0
0.5
1.0
1.5
2.0
Gaugings
Posterior rating curve
Uncertainty
Posterior activation stage
Posterior parametric uncertainty
Posterior parametric and structural uncertainty

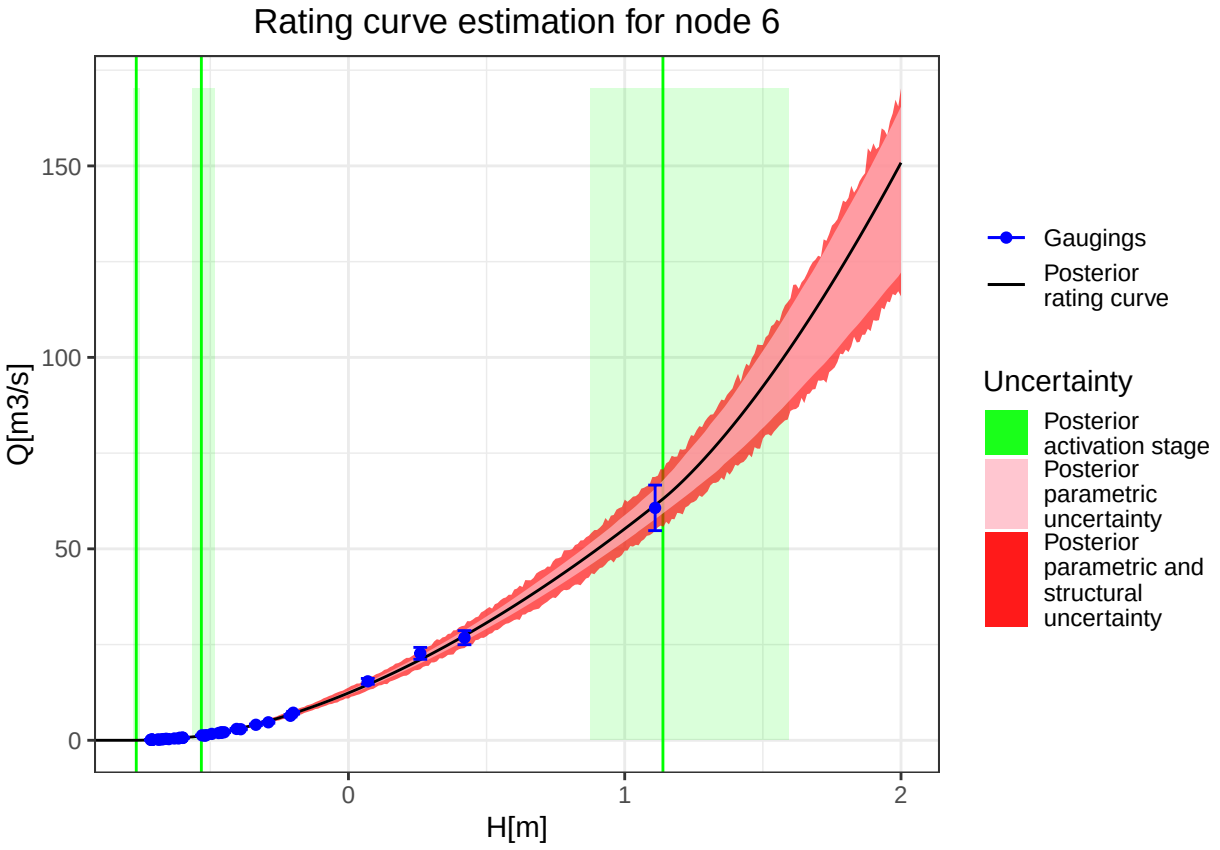
```
#>
#> [[2]]
```


Rating curve estimation for node 3
200
150
100
50
0
Q[m3/s]
0
1
2
H[m]
Gaugings
Posterior rating curve
Uncertainty
Posterior activation stage
Posterior parametric uncertainty
Posterior parametric and structural uncertainty

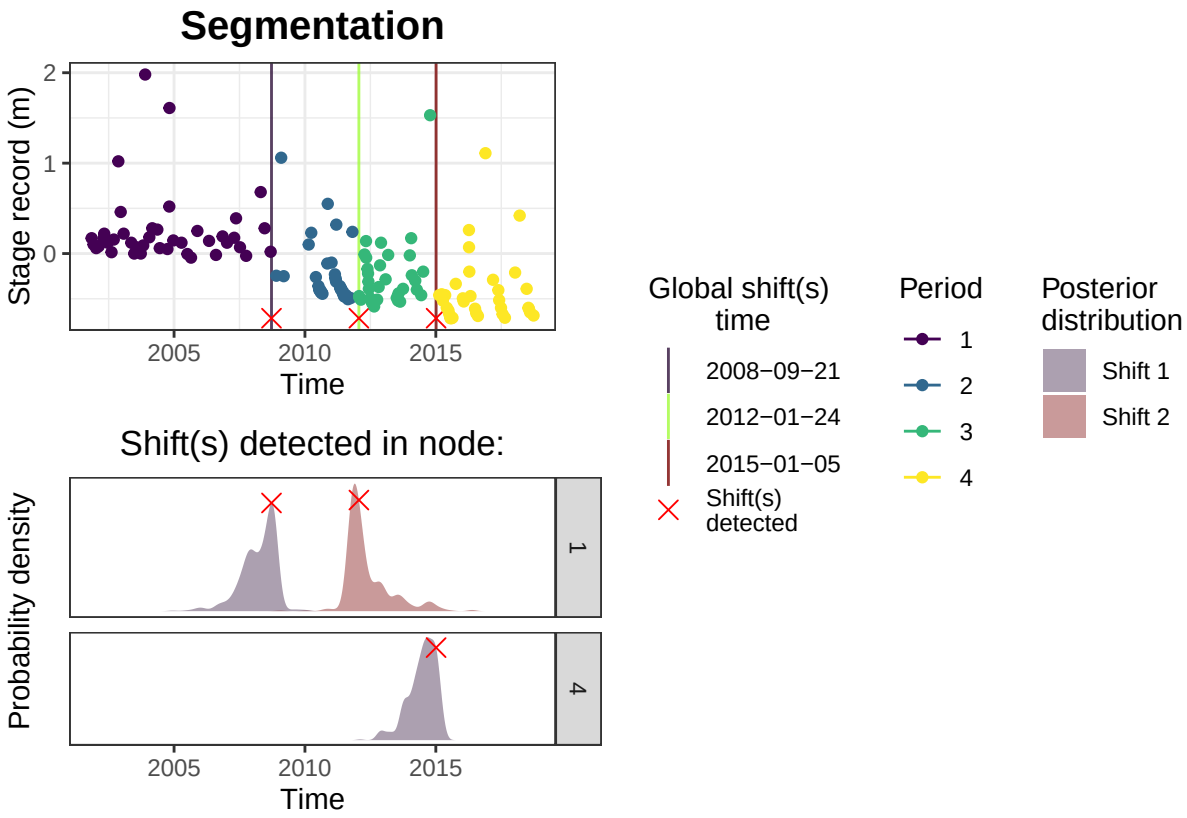
```
#>
#> [[3]]
```


Rating curve estimation for node 5
150
100
50
0
Q[m3/s]
0
1
2
H[m]
Gaugings
Posterior
rating curve
Uncertainty
Posterior
activation stage
Posterior
parametric
uncertainty
Posterior
parametric and
structural
uncertainty

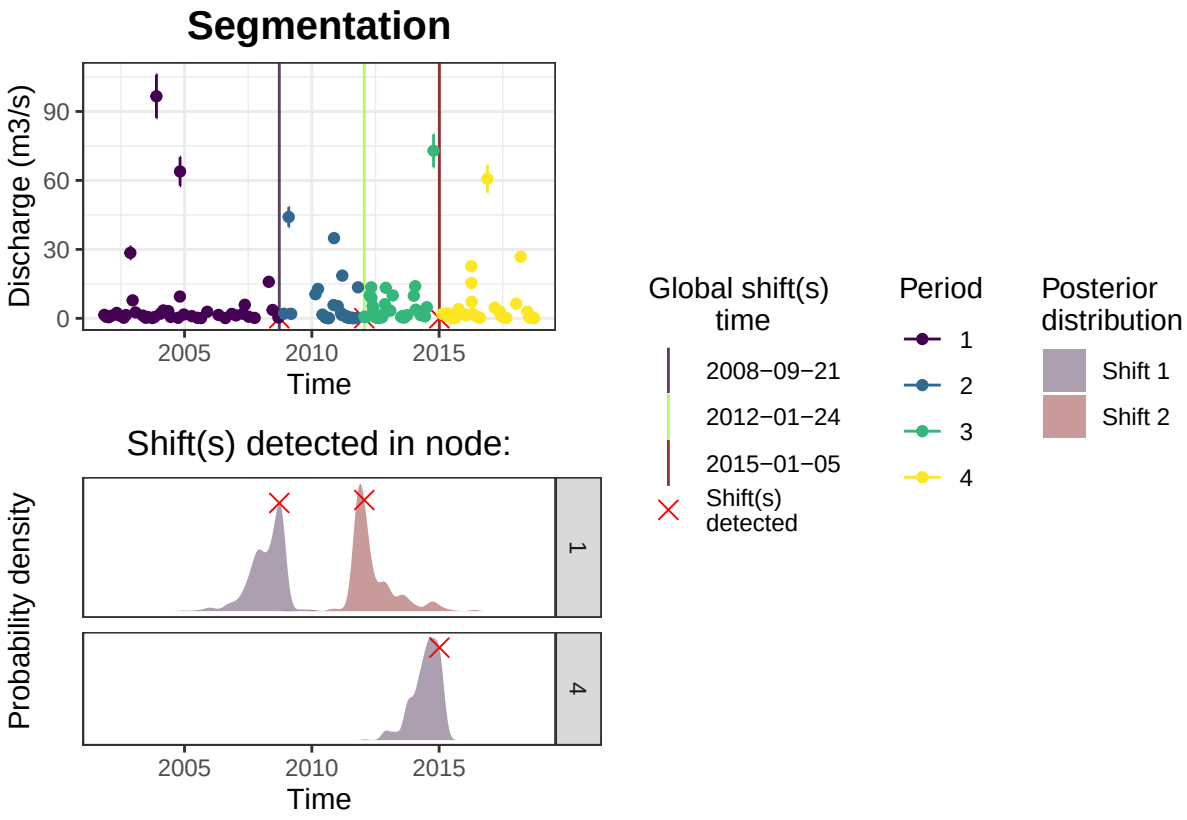
```
#>
#> [[4]]
```



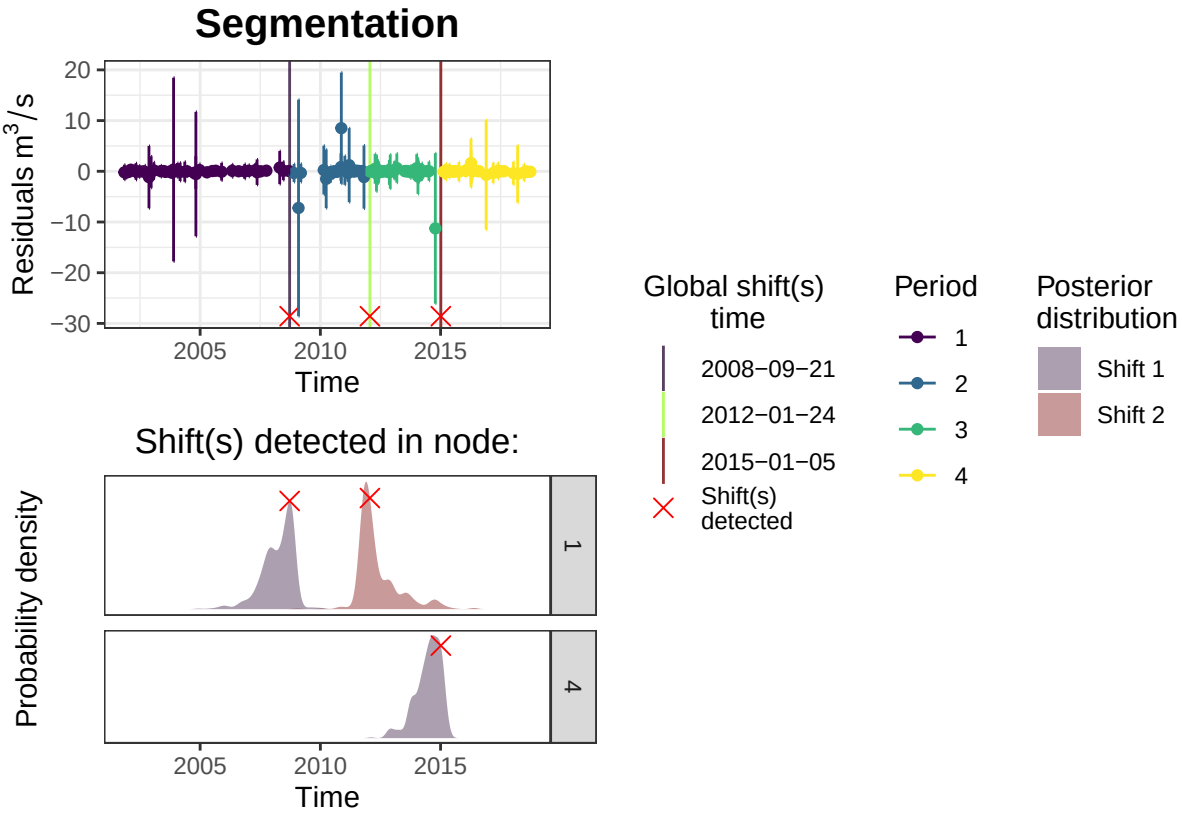

```
# Plot shift times in stage record
PlotSegmentation_Gaugings_Hdt(summary=resultsBaRatinWithuH$summary,
                              plot_summary=resultsBaRatinWithuH$plot)$final_plot
```




```
# Plot shift times in discharge observations
PlotSegmentation_Gaugings_Qdt(summary=resultsBaRatinWithuH$summary,
                              plot_summary=resultsBaRatinWithuH$plot)$final_plot
```




```
# Plot residual
PlotSegmentation_Gaugings_Residual(summary=resultsBaRatinWithuH$summary,
                                   plot_summary=resultsBaRatinWithuH$plot)$final_plot
```



Stage-recession analysis

Until now, shift time detection has relied on gauging information. Nevertheless, such data are not available for all hydrometric stations. Therefore, an alternative method has been developed to detect shift times using a variable common across the entire hydrometric network: the stage record. This approach relies solely on water level data over time to estimate shift times.

The main objective of the method is to estimate riverbed evolution at hydrometric stations. The approach primarily focuses on analyzing stage values during the recession phase following a flood, extracted from the available stage records.

In theory, following a flood, if the recession is long enough, the water level will eventually reach the riverbed (asymptotic height). By comparing consecutive recession phases, changes in the riverbed can be detected, indicating whether a shift has occurred.

To accomplish this, the method is divided into three steps:

1. Extraction of the stage-recessions
2. Bayesian estimation of the stage-recessions
3. Recessions segmentation

Extraction of the stage-recessions The first step consists in extracting all stage-recession data. However, not all existing recessions are extracted, but only those that meet the user-defined criteria.

A stage-recession period is a period of the stage record characterized by only decreasing stage values. A parameter is used to separate the recessions: χ [m]. According to this parameter a recession period k is ended when the rise or fall of stage (e.g. due to incoming rainfall event) between two consequent data exceeds χ (hence, when [$h_{rec}(t_i)$ - $h_{rec}(t_{i-1})$] > χ).

Then, stage-recessions are selected based on the following user-defined conditions:

- A minimum number of stage data for each recession, e.g., $n_{min} = 10$.
- A minimum duration (in days) for the recession, e.g., $t_{min} = 10$ days.
- A minimum and maximum distance between recession data points (minimum distance to handle an excessive number of data points and maximum distance to avoid extended periods without data, which may create discontinuous recessions)

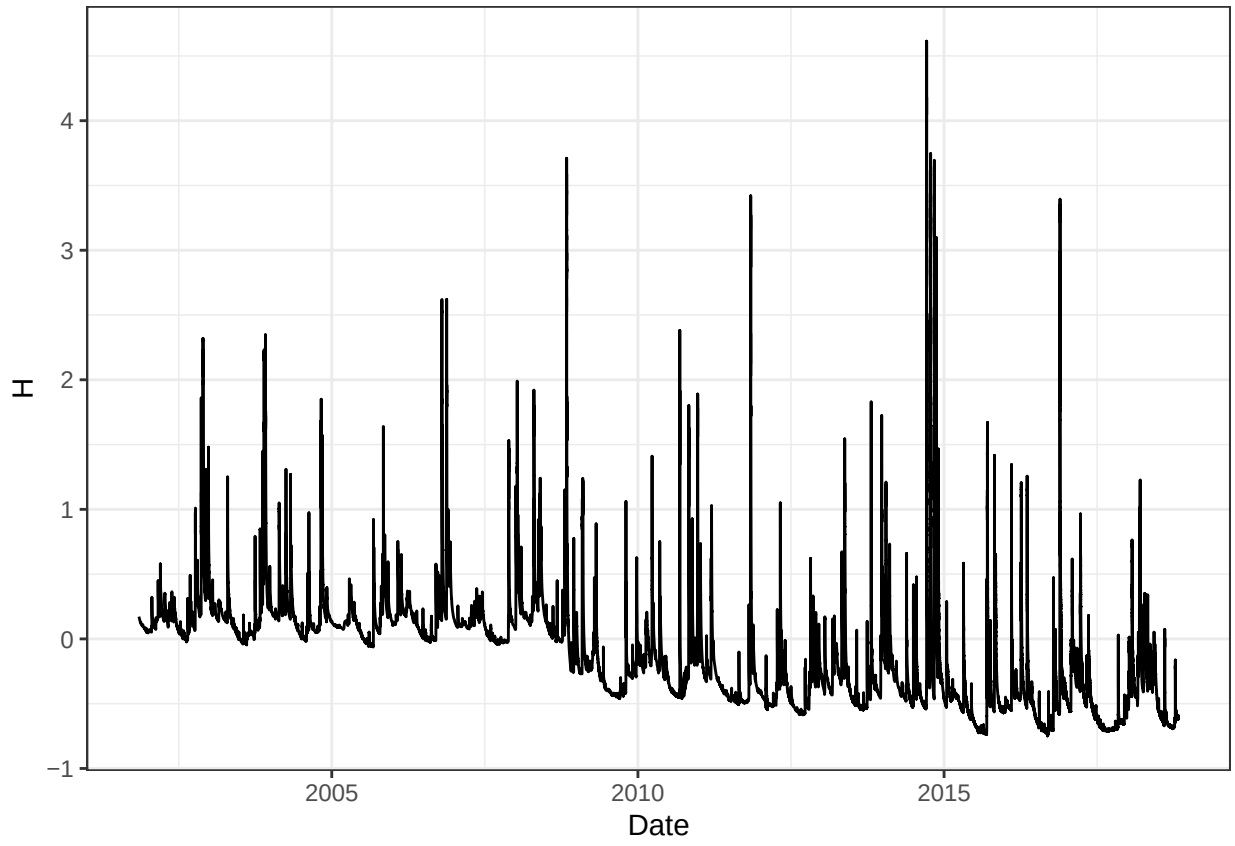
Let N_{rec} denote the number of extracted recessions, and N_k the number of stage values in the k-th recession.

The total number of stage values across all recessions is therefore: $N_{tot} = \sum_{k=1}^{N_{rec}} N_k$

For more details, please refer to the documentation `?Extract_Recessions`.

Let's examine the stage record for the Ardèche at the Meyras hydrometric station (`?ArdecheRiverStage`):

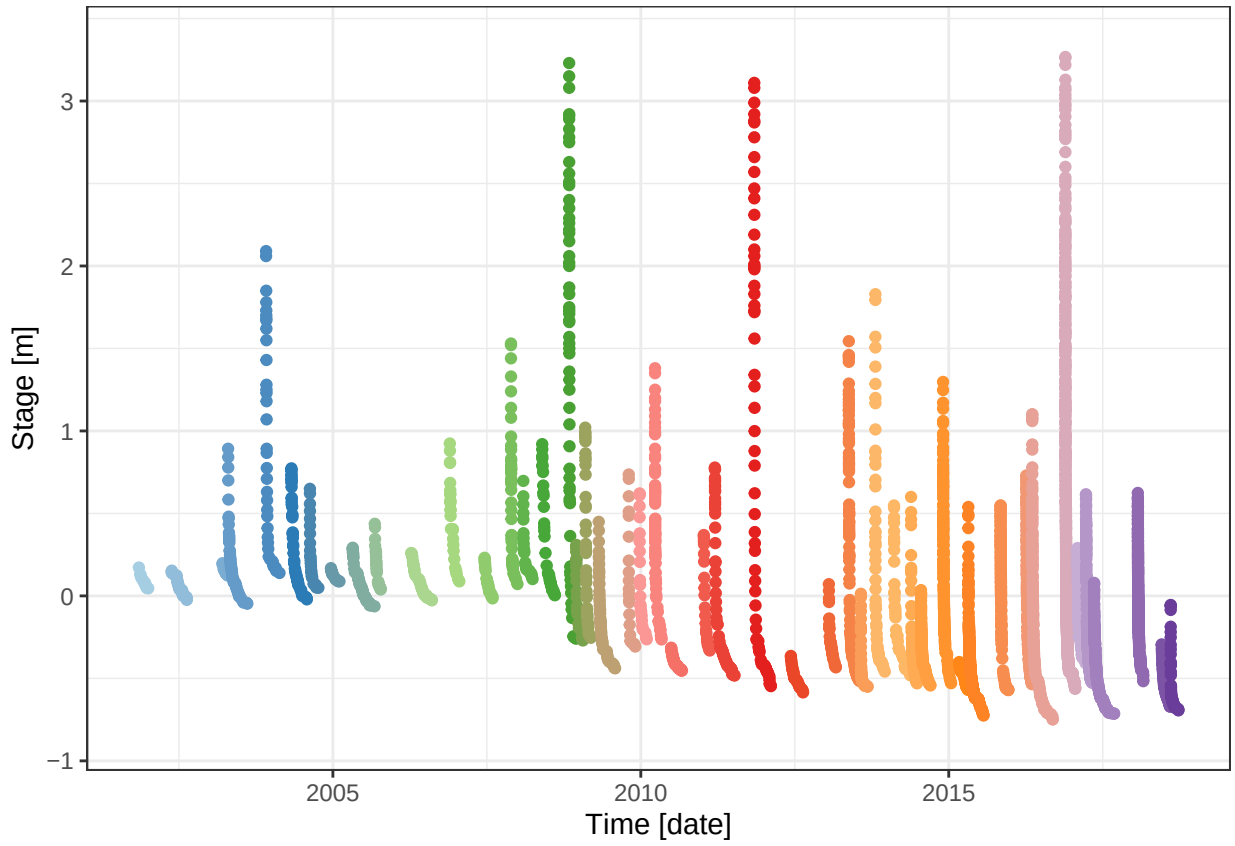
```
ggplot2::ggplot(ArdecheRiverStage,ggplot2::aes(x=Date,y=H))+
  ggplot2::geom_line()+
  ggplot2::theme_bw()
```



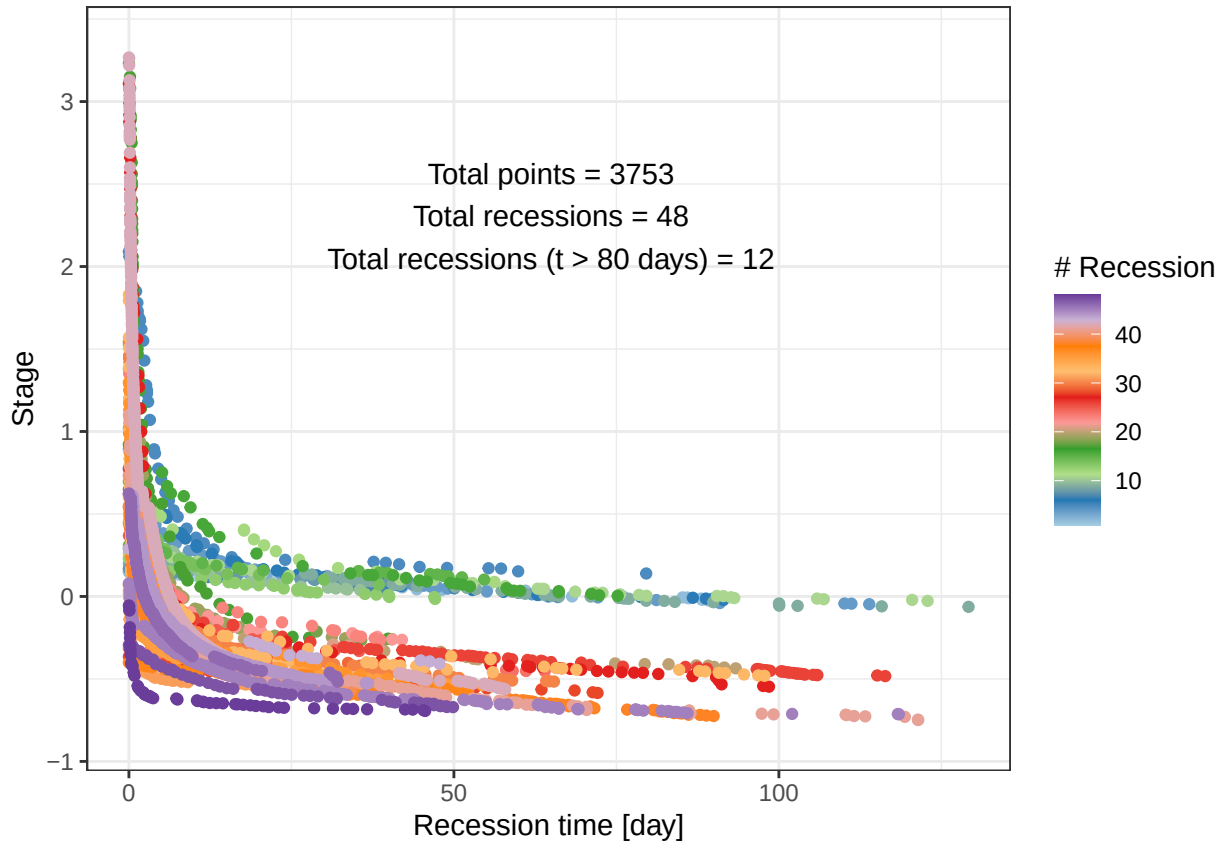
The previously shown stage record does not have any associated uncertainty. Therefore, an uncertainty of 0.05 meters will be applied to the entire stage record. This will then be used to extract all stage-recession periods based on user-defined criteria, as follows:

```
recessions_extracted=Extract_Recessions(H=ArdecheRiverStage$H,
                                        uH=0.05,
                                        time=ArdecheRiverStage$Date,
                                        chi=0.5,
                                        tgood=30,
                                        delta.t.min = 0,
```

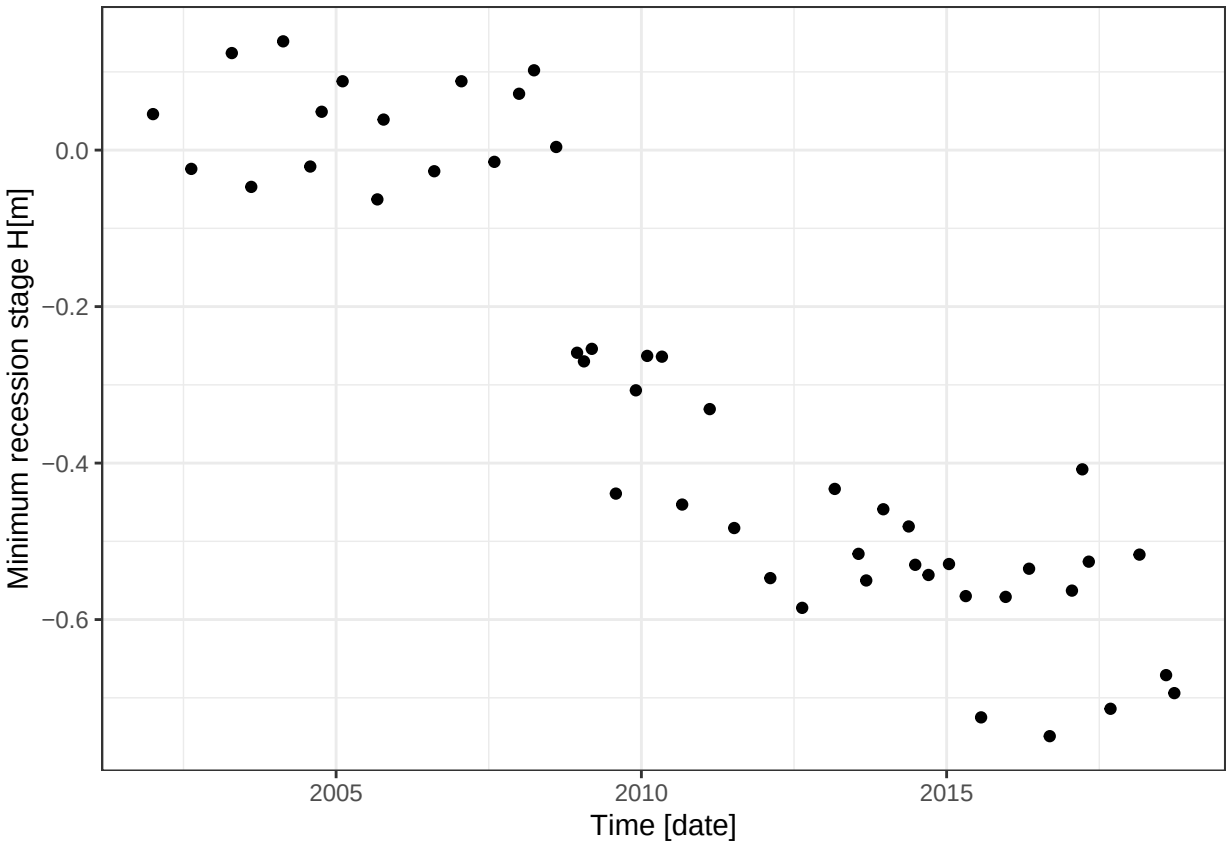
```
                                        delta.t.max = 20,
                                        Nmin.rec = 10)

recessions = recessions_extracted$Rec_extracted

# Plot information of recession extracted
PlotExtract_Recessions(Rec_extracted = recessions)
#> [[1]]
```




```
#>
#> [[2]]
```


Total points = 3753
Total recessions = 48
Total recessions (t > 80 days) = 12
Stage
Recession time [day]
Recession
40
30
20
10
3
2
1
0
−1
0
50
100

```
#>
#> [[3]]
```




```
# Check summary table of recession extracted
knitr::kable(head(recessions_extracted$summary_rec),
             align = 'c',row.names = FALSE)
```

indx	diff_time_extremes	count_data
1	55.15000	14
2	89.13958	31
3	30.47986	11
4	113.90000	53
5	79.59028	39
6	91.47014	58

Bayesian estimation of the stage-recessions Once the stage-recession data have been extracted, the second step of the proposed method is based on the estimation of all recessions through a probabilistic Bayesian pooling approach.

Using this approach all recessions are not estimated singularly but together in a unique model where some recession parameters are constant to all recessions and other parameters are specific to the each recession. We refer the reader to a similar 'pooling' approach described in Mansanarez et al. (2019).

Models fitting available in the package may be consulted here:

```
# Get model available to estimate the recession curve
names(GetCatalog_Recession())
#> [1] "M3"  "BR1"
```

```
GetCatalog_Recession()$M3$Equation()
#> [1] "alpha_1_k*exp(-lambda_1*t)+alpha_2_k*exp(-lambda_2*t)+beta_k"
```

For instance, the recession model with two exponential terms and asymptotic (`EquationRec_M3()`, proposed by Matteo Darienzo (2021)) will be considered for the calculation, and it is described below:

$$\alpha_{1_k} \cdot \exp(-\lambda_1 \cdot t) + \alpha_{2_k} \cdot \exp(-\lambda_2 \cdot t) + \beta_k$$

- Stable parameters: λ_1 , λ_2
- Recession-specific parameters: α_{1_k}, α_{2_k}, β_k

The estimation of the stable parameters will remain constant across all periods, in contrast to the recession-specific parameters, which will be estimated independently for each period.

Recessions segmentation Once the recession model has been estimated, the third step of the method consists in the segmentation of the estimated recessions. In particular we analyse the temporal evolution of the asymptotic stage parameter β_k, which corresponds to the elevation b of the lowest control.

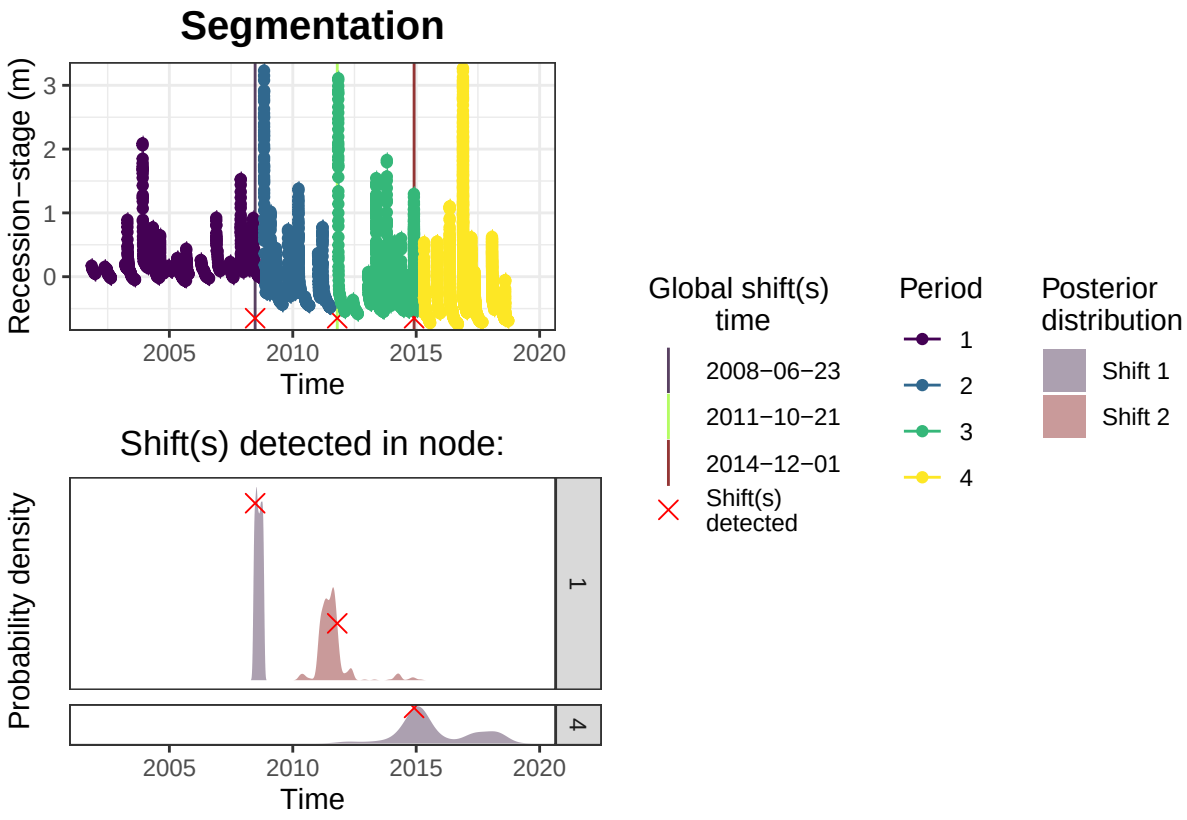
This information may be interpreted as a random variable that needs to be segmented using a recursive segmentation procedure, specifically the `Recursive_Segmentation()`function. The time associated to each parameter β_k is the time at which the corresponding recession begins.

Both Bayesian estimation of the stage-recession and the recession segmentation are grouped in the same function `ModelAndSegmentation_Recessions` as shown here:

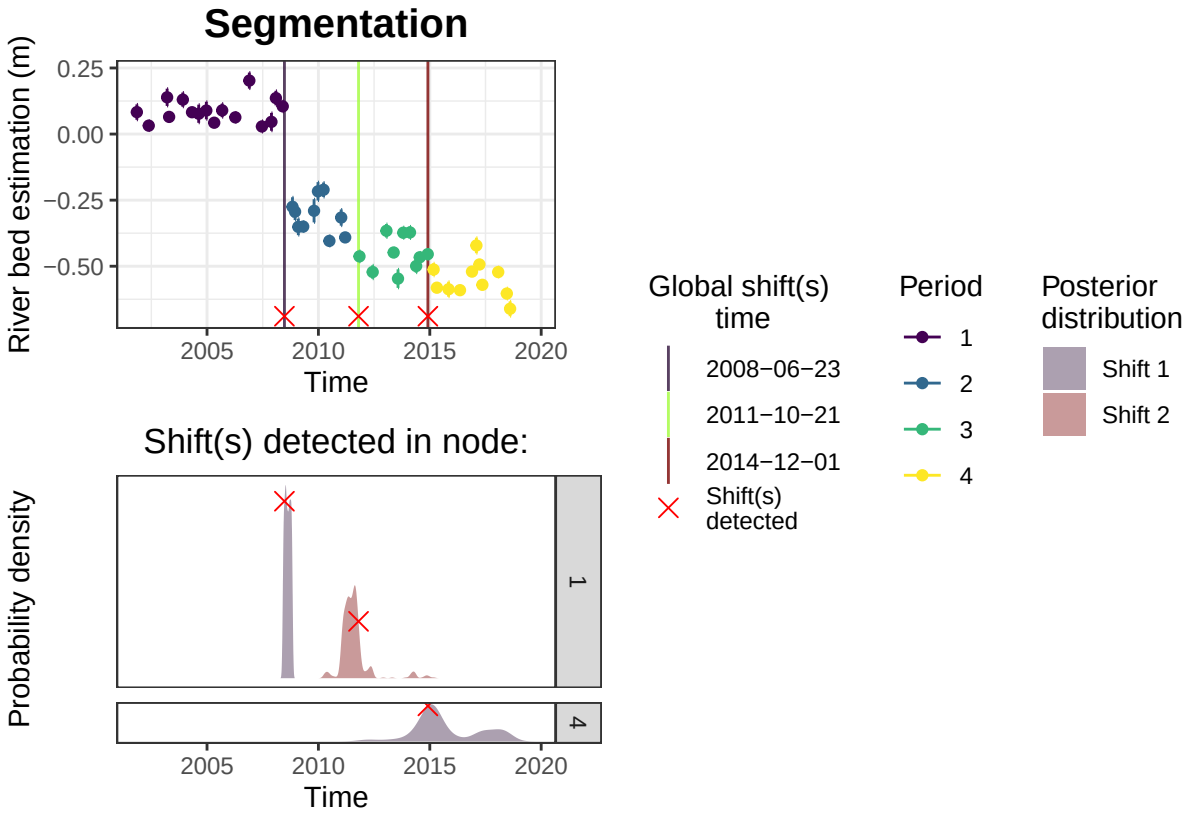
```
model_rec=ModelAndSegmentation_Recessions(time_rec=recessions$time_rec,
                                          daterec=recessions$date,
                                          hrec=recessions$hrec,
                                          uHrec=recessions$uHrec,
                                          indx=recessions$indx,
                                          nSmax = 3,
                                          nMin = 1,
                                          funk='M3')
```

The estimation and segmentation have been completed. Next, plot functions have been developed to generate figures for interpreting the obtained results.

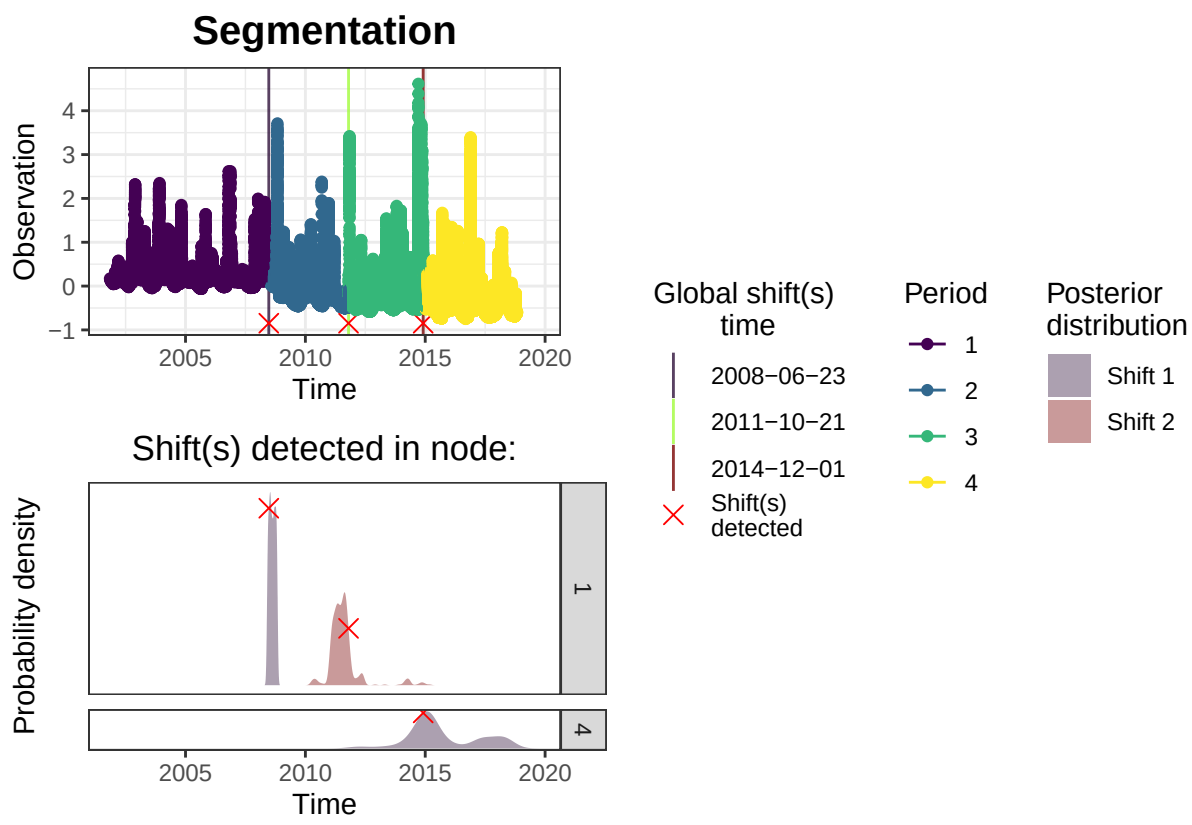
```
# Plot recession segmentation
 PlotSegmentation_Recessions(model_rec=model_rec)
#> $plot.rec.segmented
```




```
#>
#> $plot.b.segmented
```




```
# Plot using all data of stage record
PlotSegmentation_Recessions_Hdt(time=ArdecheRiverStage$Date,
                                obs=ArdecheRiverStage$H,
                                u=0.05,
                                plot_summary = model_rec$plot)
```



Références

Darienzo, Matteo. 2021. "Detection and Estimation of Stage-Discharge Rating Shifts for Retrospective and Real-Time Streamflow Quantification." PhD thesis. https://theses.hal.science/tel-03211343v1.

Darienzo, M., B. Renard, J. Le Coz, and M. Lang. 2021. "Detection of Stage-Discharge Rating Shifts Using Gaugings: A Recursive Segmentation Procedure Accounting for Observational and Model Uncertainties." *Water Resources Research* 57 (4): e2020WR028607. https://doi.org/10.1029/2020WR028607.

Horner, I., B. Renard, J. Le Coz, F. Branger, H. K. McMillan, and G. Pierrefeu. 2018. "Impact of Stage Measurement Errors on Streamflow Uncertainty." *Water Resources Research* 54 (3): 1952–76. https://doi.org/10.1002/2017WR022039.

Le Coz, J., B. Renard, L. Bonnifait, F. Branger, and R. Le Boursicaud. 2014. "Combining Hydraulic Knowledge and Uncertain Gaugings in the Estimation of Hydrometric Rating Curves: A Bayesian Approach." *Journal of Hydrology* 509 (February): 573–87. https://doi.org/10.1016/j.jhydrol.2013.11.016.

Mansanarez, V., J. Le Coz, B. Renard, M. Lang, G. Pierrefeu, and P. Vauchel. 2016. "Bayesian Analysis of Stage-Fall-Discharge Rating Curves and Their Uncertainties." *Water Resources Research* 52 (9): 7424–43. https://doi.org/10.1002/2016WR018916.

Mansanarez, V., B. Renard, J. Le Coz, M. Lang, and M. Darienzo. 2019. "Shift Happens! Adjusting Stage-Discharge Rating Curves to Morphological Changes at Known Times." *Water Resources Research* 55 (4): 2876–99. https://doi.org/10.1029/2018WR023389.

Renard, Benjamin. 2017. "BaM ! (Bayesian Modeling): Un code de calcul pour l'estimation d'un modèle quelconque et son utilisation en prédiction." {Report}. irstea. https://hal.inrae.fr/hal-02606929.