depmixS4

A flexible package to estimate mixture and hidden Markov models

Maarten Speekenbrink (m.speekenbrink@ucl.ac.uk) and Ingmar Visser
LondonR, 10 September 2013

DepmixS4

- An R package to estimate **dep**endent **mix**ture models
- Written by Ingmar Visser (University of Amsterdam) and Maarten Speekenbrink (University College London)
- First version (0.1.0) was released on CRAN on 26-Mar-2008. The current development version is 1.3.0 (available on RForge and soon on CRAN).

The objective of the package is to provide a flexible implementation of mixture and hidden Markov models. Mixture components are (mostly) implemented as generalized linear models. Using S4, users can easily define their own observation models.

Mixture models

In a *mixture model*, each observation is assumed to be drawn from one of a number of distinct subpopulations (component distributions). Which subpopulation an observation is drawn from is not directly observable and represented by a latent state.

A mixture distribution over observations $Y_t, t = 1, \dots, T$, is defined as

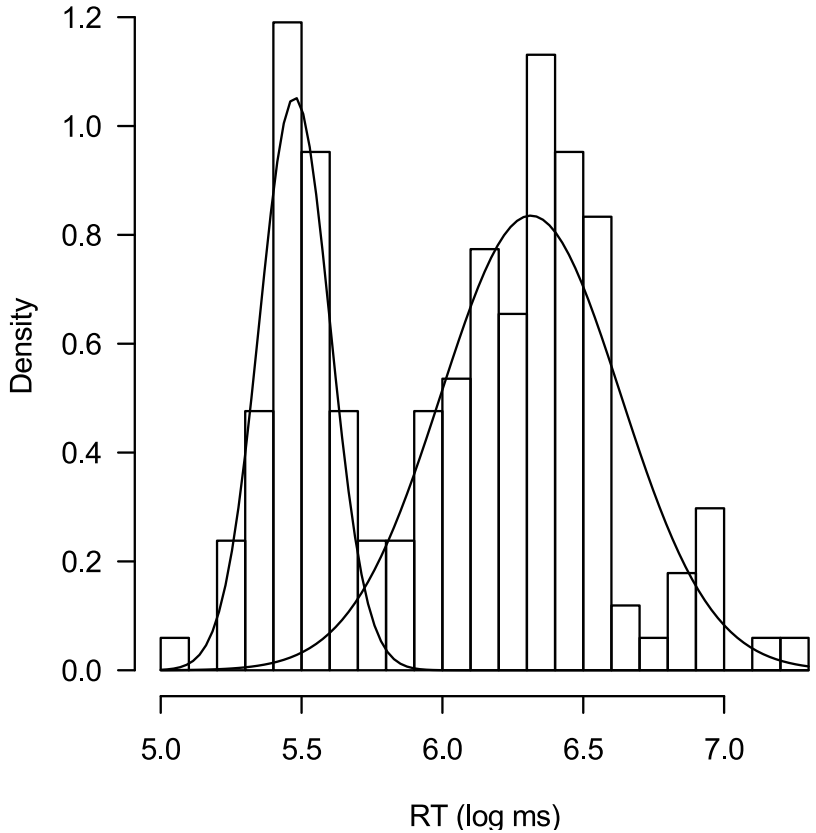
$$p(Y_t = y) = \sum_{i=1}^{N} p(Y_t = y|S_t = i)P(S_t = i)$$

where

- $S_t \in \{1, \dots, N\}$ denotes the latent state (a.k.a. "class", "component") of observation t
- $P(S_t = i)$ denotes the probability that the latent state at t equals i
- $p(Y_t = y|S_t = i)$ denotes the density of observation Y_t (evaluated at y), conditional upon the latent state being $S_t = i$; i.e., it is the value of the i-th component density (evaluated at y).

Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

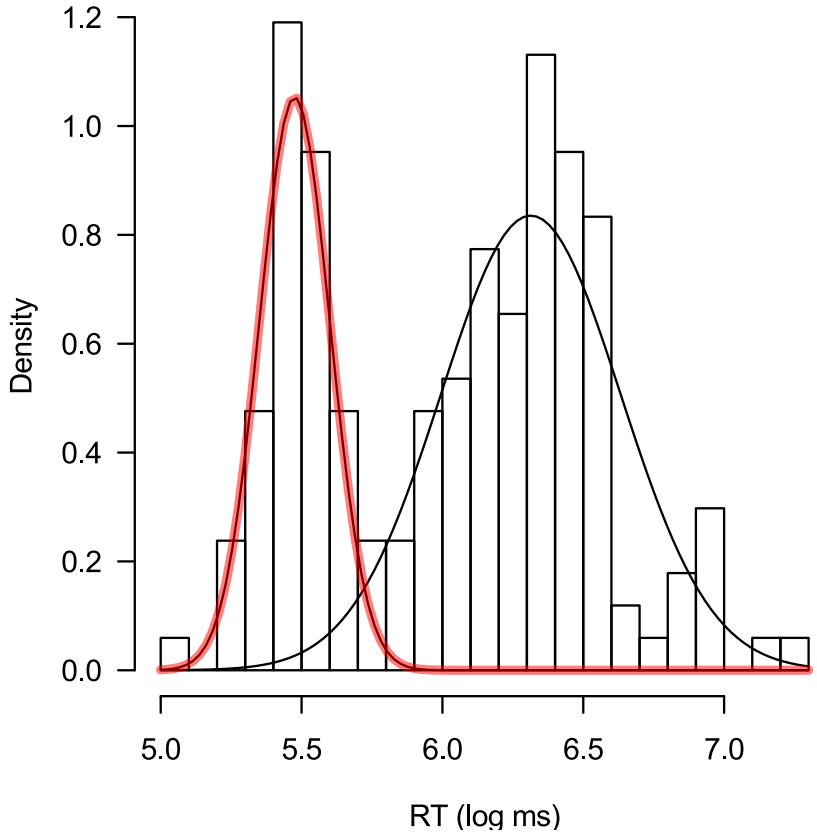


Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

Model components:

- $p(Y_t|S_t = 1) = N(5.48, 0.13)$

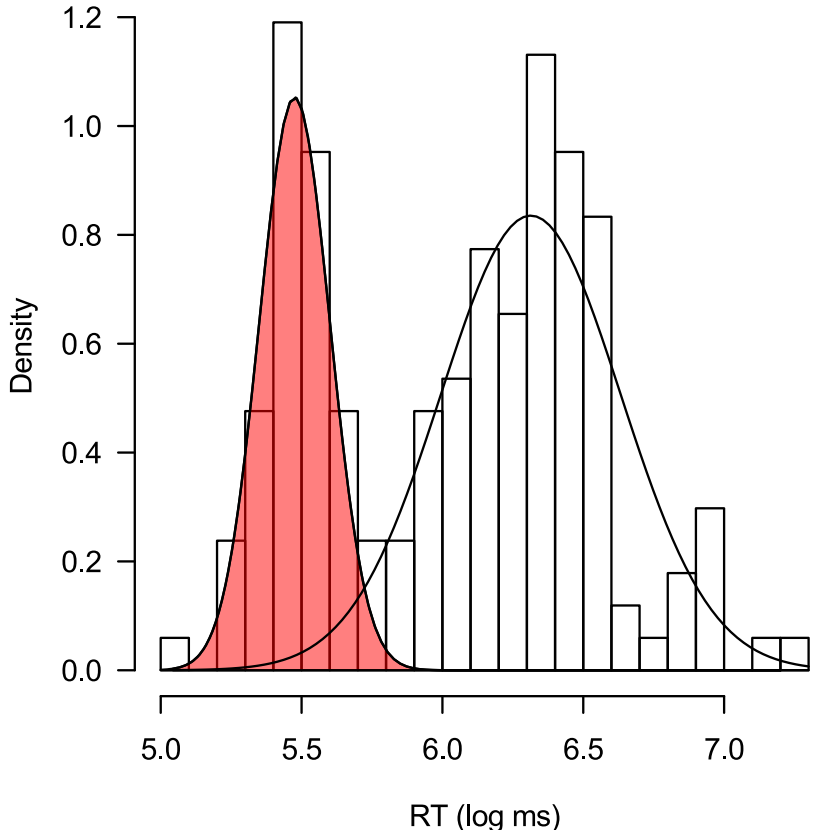


Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

Model components:

- $p(Y_t|S_t = 1) = N(5.48, 0.13)$
- $P(S_t = 1) = 0.33$

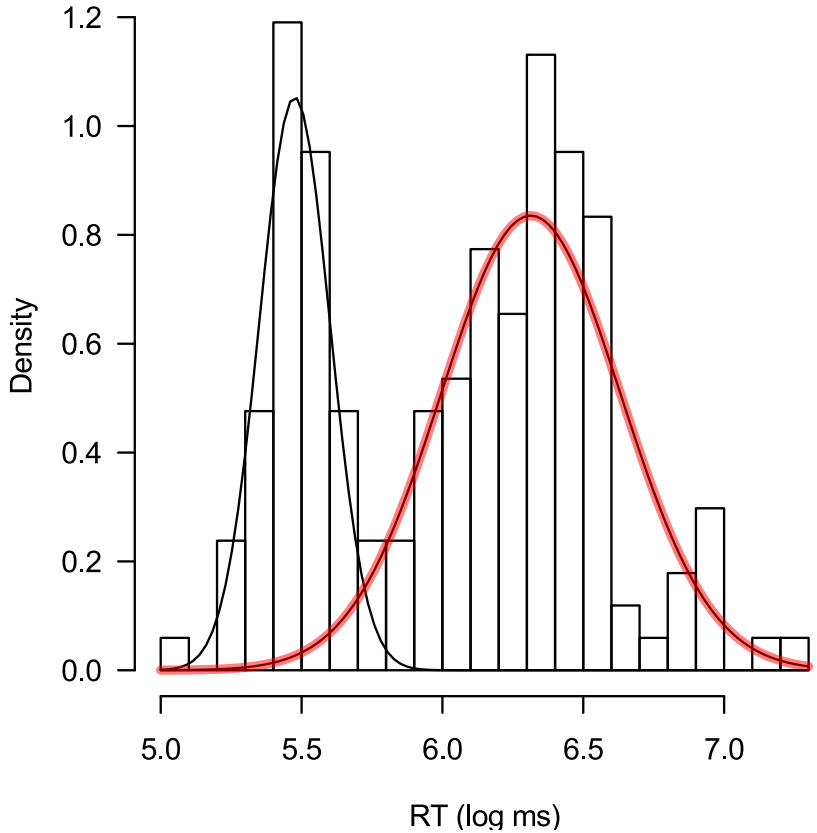


Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

Model components:

- $p(Y_t|S_t = 1) = N(5.48, 0.13)$
- $P(S_t = 1) = 0.33$
- $p(Y_t|S_t = 2) = N(6.31, 0.32)$

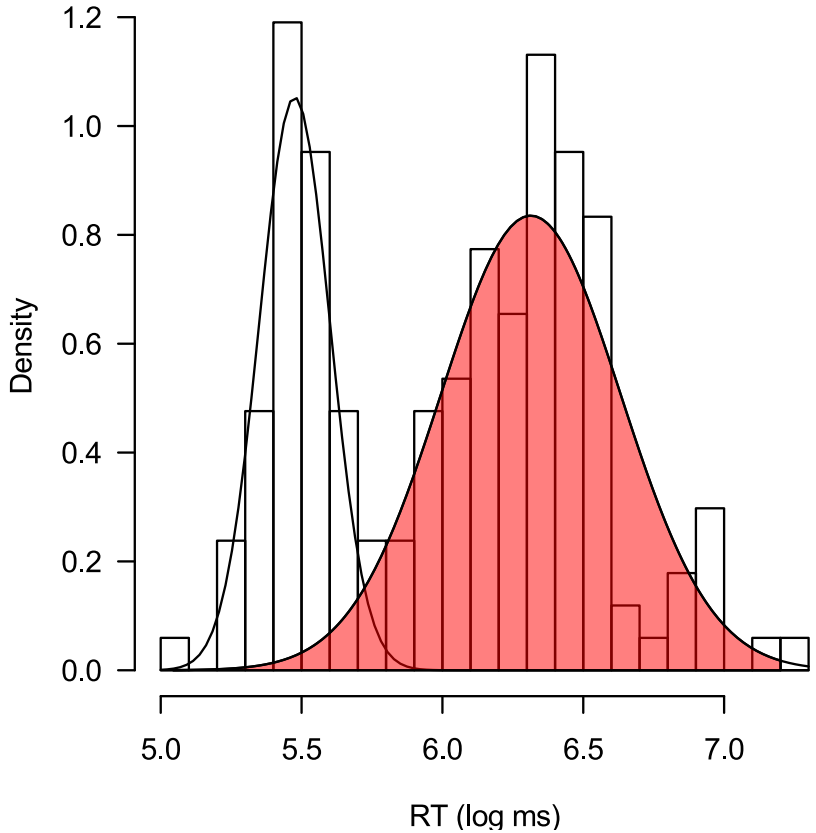


Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

Model components:

- $p(Y_t|S_t = 1) = N(5.48, 0.13)$
- $P(S_t = 1) = 0.33$
- $p(Y_t|S_t = 2) = N(6.31, 0.32)$
- $P(S_t = 2) = 0.67$

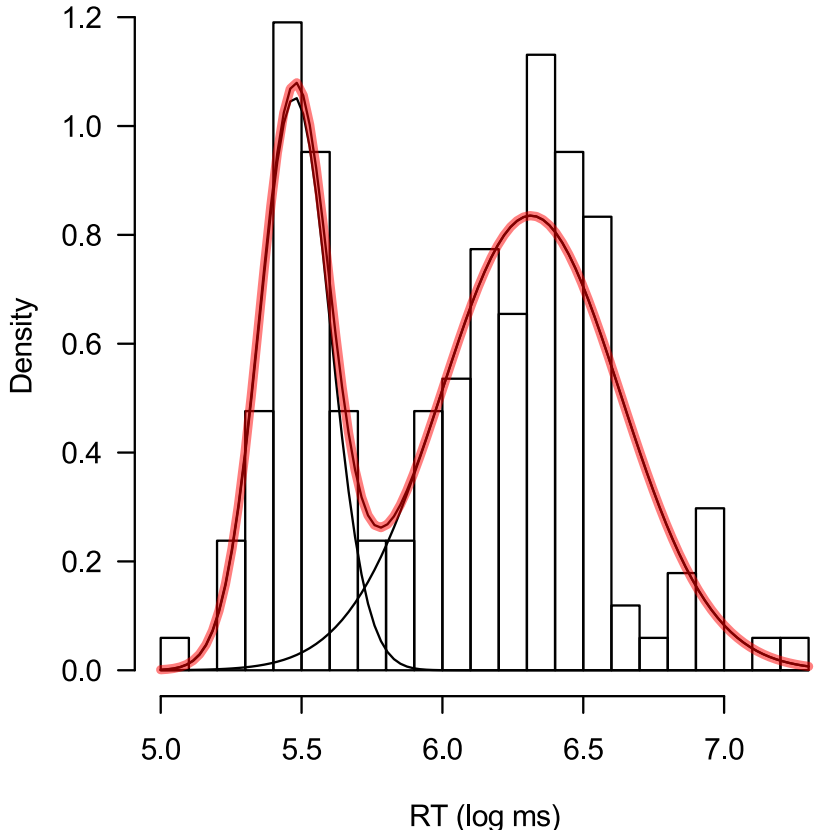


Mixture distribution:

$$p(Y_t) = \sum_{i=1}^{N} p(Y_t|S_t = i)P(S_t = i)$$

Model components:

- $p(Y_t|S_t = 1) = N(5.48, 0.13)$
- $P(S_t = 1) = 0.33$
- $p(Y_t|S_t = 2) = N(6.31, 0.32)$
- $P(S_t = 2) = 0.67$
- $p(Y_t)$



Dependent mixture models

In a *dependent mixture model*, states are assumed to be statistically dependent. The process underlying state transitions is a homogenous first-order Markov process. This process is completely defined by the initial state probabilities

$$P(S_1 = 1), \ldots, P(S_1 = N)$$

and the state transition matrix

$$\begin{pmatrix} P(S_t = 1|S_{t-1} = 1) & P(S_t = 2|S_{t-1} = 1) & \cdots & P(S_t = N|S_{t-1} = 1) \\ P(S_t = 1|S_{t-1} = 2) & P(S_t = 2|S_{t-1} = 2) & \cdots & P(S_t = N|S_{t-1} = 2) \\ \vdots & \vdots & \ddots & \vdots \\ P(S_t = 1|S_{t-1} = N) & P(S_t = 2|S_{t-1} = N) & \cdots & P(S_t = N|S_{t-1} = N) \end{pmatrix}$$

Implementation in `depmixS4`

Models are estimated using Expectation-Maximization (EM) or numerical optimization (when parameters are constrained). The structure of a dependent mixture model allows it to be divided into three submodels:

- The prior model: $P(S_1|x, \theta_{\text{prior}})$
- The transition model: $P(S_t|x, S_{t-1}, \theta_{\text{trans}})$
- The response model: $P(Y_t|S_t, x, \theta_{\text{resp}})$

The prior and transition models are implemented as multinomial regression models, and the response models as generalized linear models. Within the EM algorithm, the maximization-step can be performed by weighted maximum likelihood (as in e.g., `glm.fit()`).

Using depmixS4

Mixture models can be constructed using the `mix()` function, and hidden Markov models with the `depmix()` function. The `depmix()` function takes the following arguments:

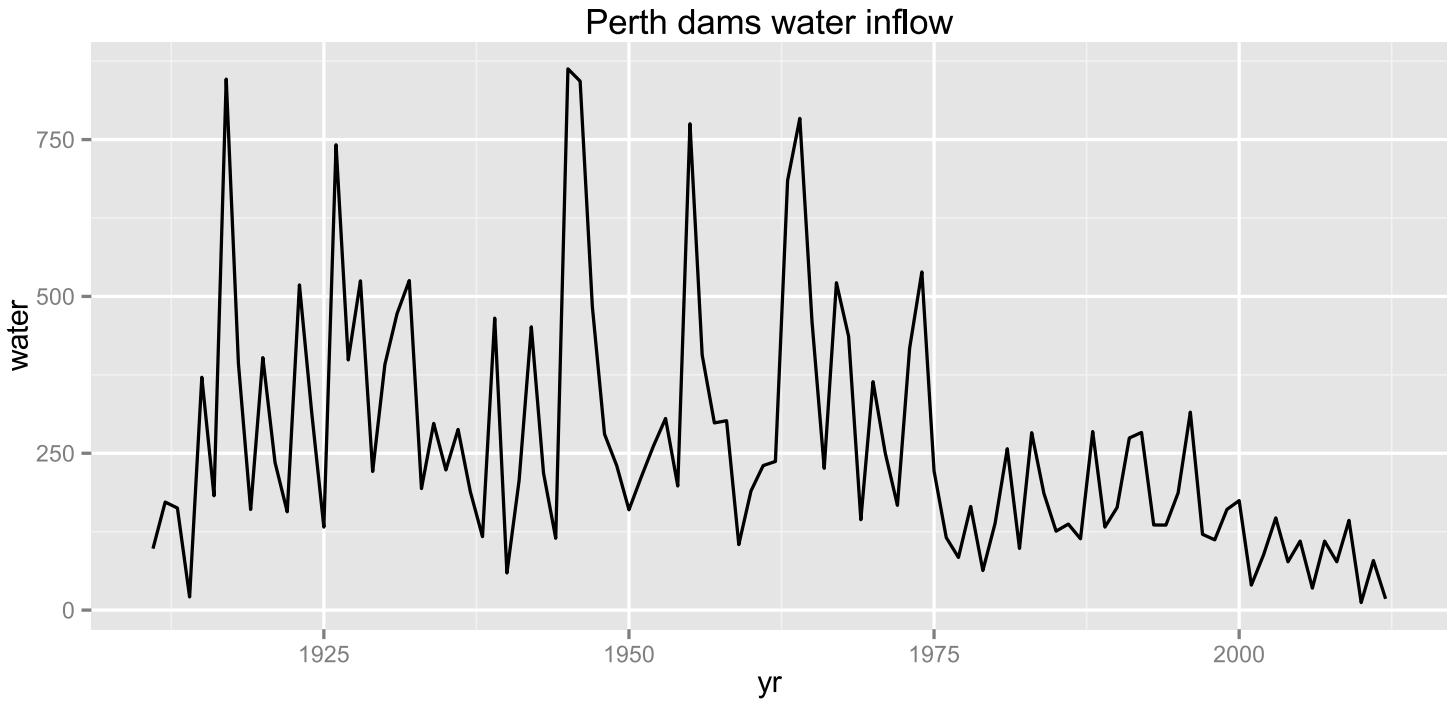
1. `response`: a formula specifying the response models (univariate) or a list with formulae (multivariate)
2. `nstates`: the number of states/components
3. `data`: the data frame containing the variables in the response models
4. `family`: the family of the response models (as in the `glm` function) or a list with families (multivariate)
5. `ntimes`: a vector with the length of each time-series in the data.

More general models can be constructed with the `makeDepmix()` function. Models are estimated by calling the `fit()` function on a constructed model.

Example 1: Climate change

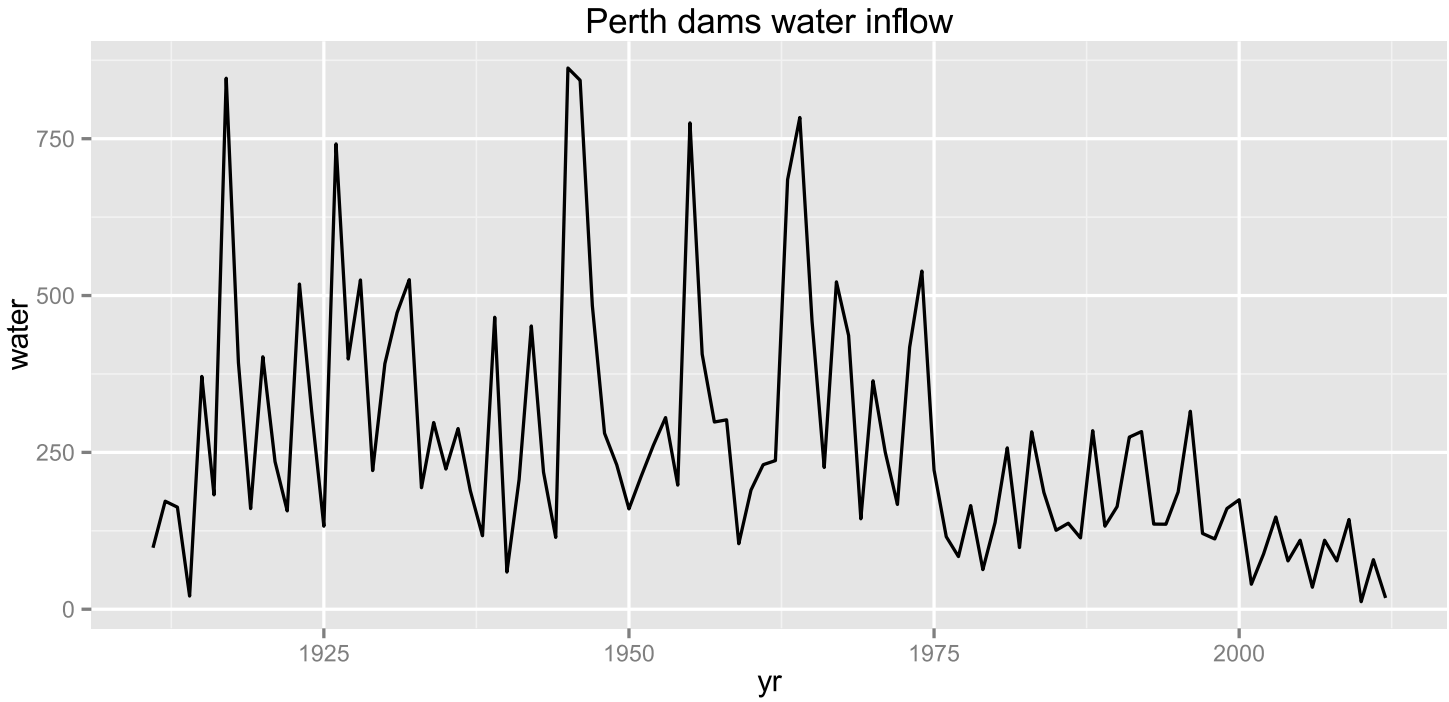
Climate change

Data from the Water Corporation of Western Australia containing the yearly inflow in water catchment dams around Perth, South-west Australia from 1911 through to 2012.



Climate change

1. Is there a trend in these data?
2. Are there sudden shifts in the catchment amounts?



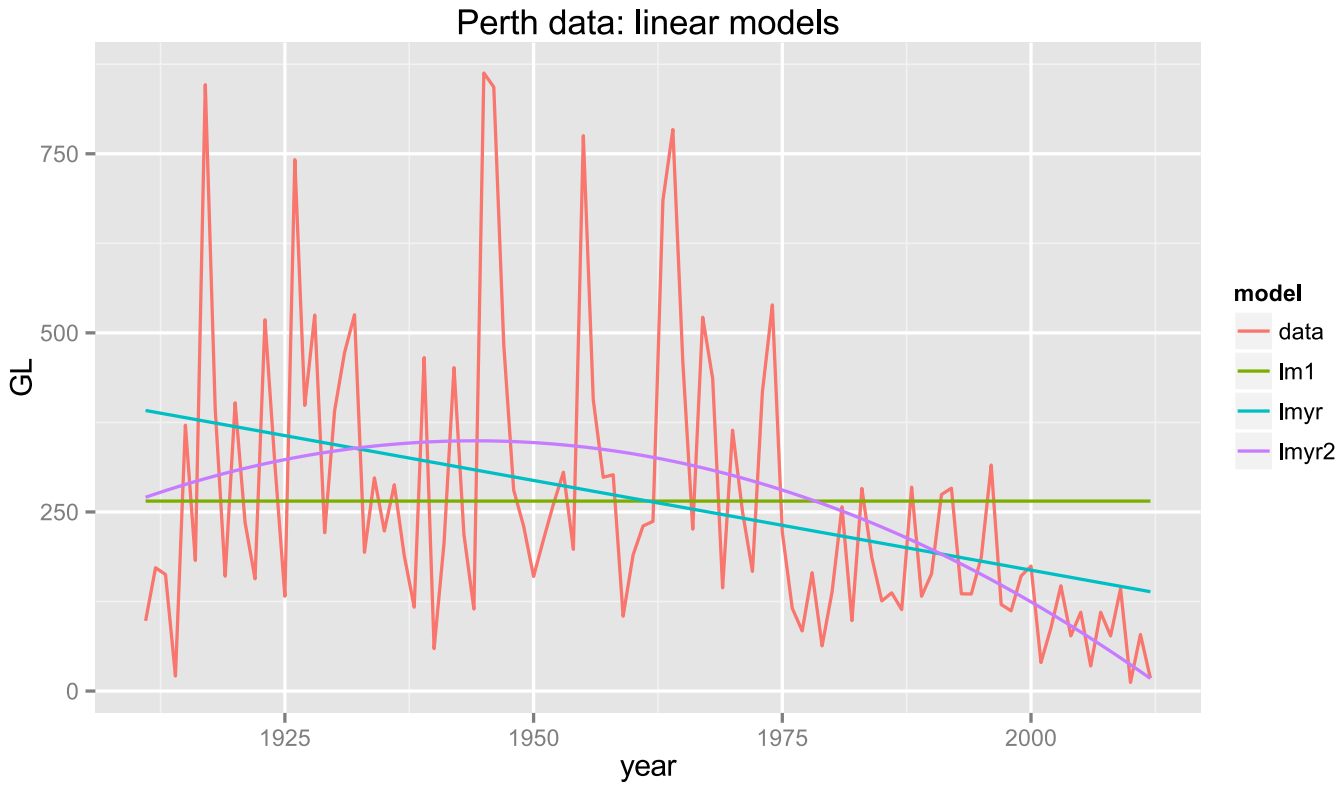
Typical analyses: linear models

```
lm1 <- lm(water ~ 1, data = perth)
lmyr <- lm(water ~ yr, data = perth)
lmyr2 <- lm(water ~ yr + I(yr^2), data = perth)
anova(lm1, lmyr, lmyr2)
```

```
Analysis of Variance Table

Model 1: water ~ 1
Model 2: water ~ yr
Model 3: water ~ yr + I(yr^2)
  Res.Df     RSS Df Sum of Sq    F  Pr(>F)
1    101 3811695
2    100 3257024  1    554671 18.7 3.7e-05 ***
3     99 2939677  1    317347 10.7  0.0015 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(both linear and quadratic trends are significant)

Typical analyses: linear models



Typical analyses: AR models

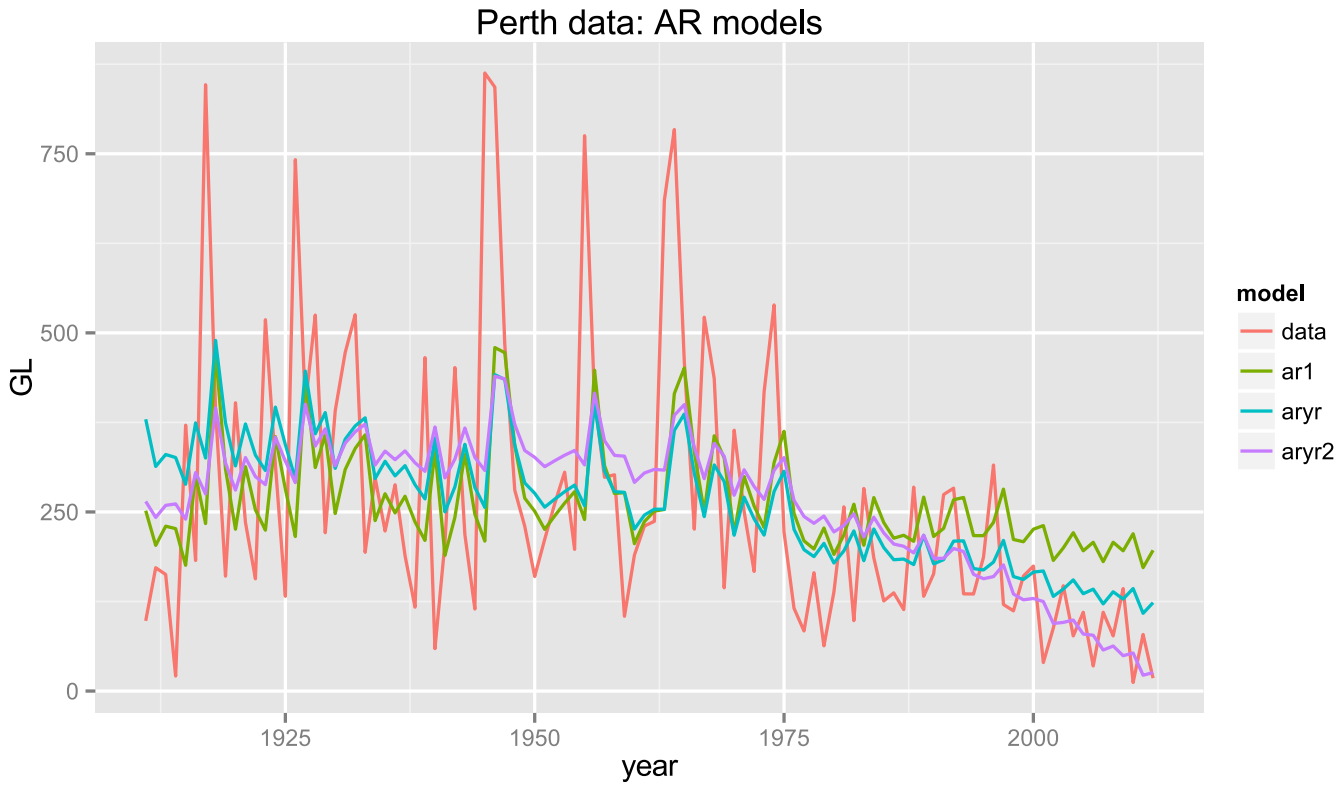
```
arOrder <- ar(perth$water)$order
ar1 <- arima(perth$water, c(arOrder, 0, 0))
aryr <- arima(perth$water, c(arOrder, 0, 0), xreg = perth$yr)
aryr2 <- arima(perth$water, c(arOrder, 0, 0), xreg = cbind(yr = scale(perth$yr),
    yr2 = scale(perth$yr)^2))
print(c(ar1 = AIC(ar1), aryr = AIC(aryr), aryr2 = AIC(aryr2)))
```

```
  ar1  aryr aryr2
 1355  1349  1344
```

```
# Compare to linear models:
innerCode
print(c(lm1 = AIC(lm1), lmyr = AIC(lmyr), lmyr2 = AIC(lmyr2)))
```

```
  lm1  lmyr lmyr2
 1367  1353  1345
```

Typical analyses: AR models



Change point models

1. Assume one or more discrete change points in the data
2. Mean, trend, and/or other parameters (eg ar parameters) of the process may change

Can be estimated with hidden Markov models by restricting the transition matrix

Transition matrix for 1 change point

$$\begin{pmatrix} p_1 & 1-p_1 \\ 0 & 1 \end{pmatrix}$$

Transition matrix for 2 changepoints

$$\begin{pmatrix} p_1 & 1-p_1 & 0 \\ 0 & p_2 & 1-p_2 \\ 0 & 0 & 1 \end{pmatrix}$$

Transition matrix for 3 changepoints

$$\begin{pmatrix} p_1 & 1-p_1 & 0 & 0 \\ 0 & p_2 & 1-p_2 & 0 \\ 0 & 0 & p_3 & 1-p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Et cetera

In HMM literature these models are also called left-right models or Bakis models

Model with 1 changepoint

In the EM algorithm, probabilities initialized as 0 or 1 remain there, so provide starting values according to change point models and all will be fine...

```
mod2 <- depmix(water ~ 1, data = perth, ns = 2, trst = c(0.9, 0.1, 0, 1), inst = c(1,
    0))
set.seed(1)
fm2 <- fit(mod2, verbose = FALSE)
```

```
converged at iteration 3 with logLik: -654.5
```

```
depmixS4::summary(fm2)
```

```
Initial state probabilties model
pr1 pr2
  1   0


Transition matrix
         toS1   toS2
fromS1 0.9845 0.0155
fromS2 0.0000 1.0000

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1           337.0 204.08
St2           141.4  76.16
```

Predicting states

Use the `posterior()` function to obtain the maximum a posteriori state sequence (column 1) and the posterior state probabilities (remaining columns).

```
pst <- posterior(fm2)
head(pst)
```

```
  state     X1      X2
1     1 1.0000 0.00000
2     1 0.9489 0.05110
3     1 0.8312 0.16875
4     1 0.6558 0.34420
5     1 0.9849 0.01513
6     1 0.9536 0.04642
```

```
pst[, 1]
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
 [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Model with 2 changepoints

```
# 2-state models
mod3 <- depmix(water ~ 1, data = perth, ns = 3, trst = c(0.9, 0.1, 0, 0, 0.9,
    0.1, 0, 0, 1), inst = c(1, 0, 0))
set.seed(1)
fm3 <- fit(mod3, verbose = FALSE)
## EM converges to a local maximum, so it is advisable to fit a model
## repeatedly with different random starting values
for (i in 1:100) {
    try(tfm3 <- fit(mod3, verbose = FALSE))
    if (logLik(tfm3) > logLik(fm3))
        fm3 <- tfm3
}
```

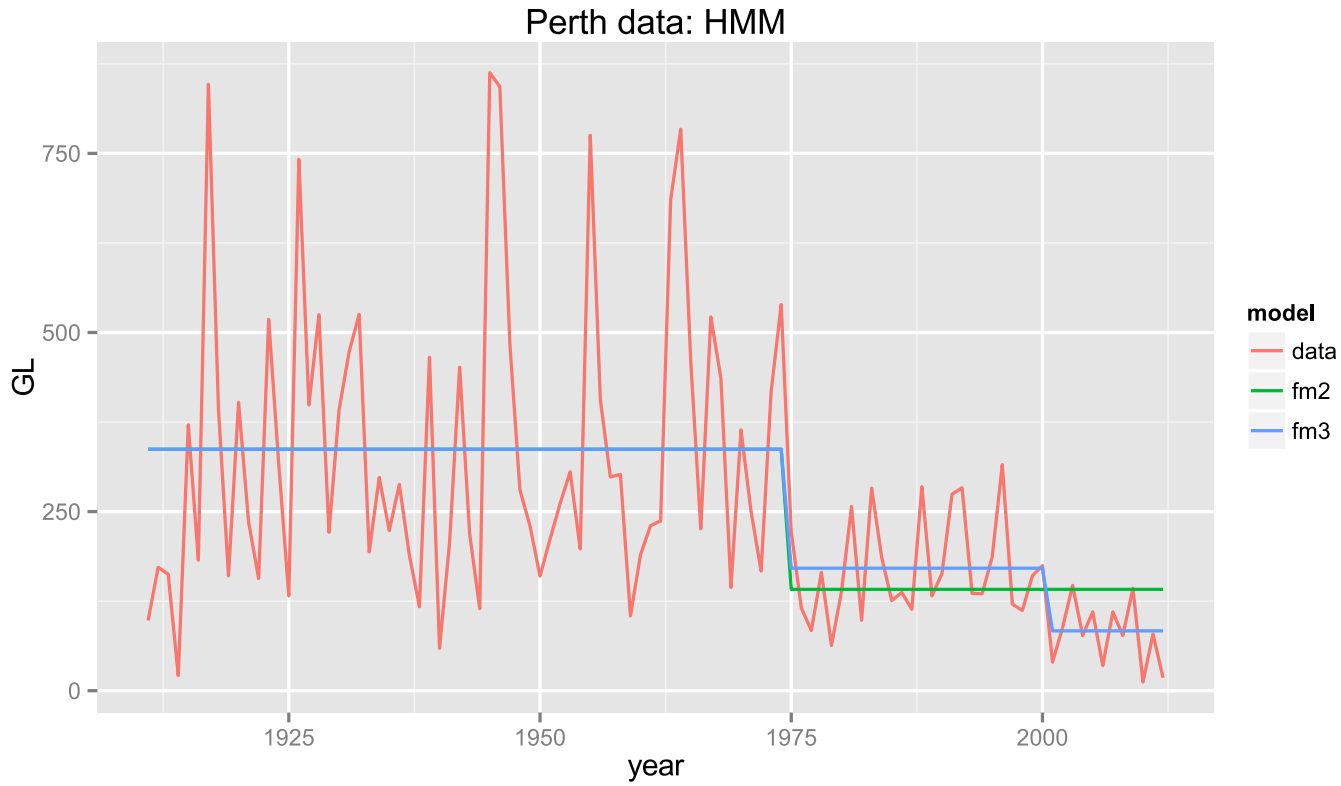
Model with 2 changepoints

```
depmixS4::summary(fm3)
```

```
Initial state probabilties model
pr1 pr2 pr3
  1   0   0


Transition matrix
         toS1   toS2    toS3
fromS1 0.9845 0.0155 0.00000
fromS2 0.0000 0.9598 0.04018
fromS3 0.0000 0.0000 1.00000

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1          336.99  204.2
St2          171.00   71.3
St3           83.49   46.3
```

Changepoint models

The relative drought that started after 1976 is possibly related to the then occurring El Nina/El Nino event.

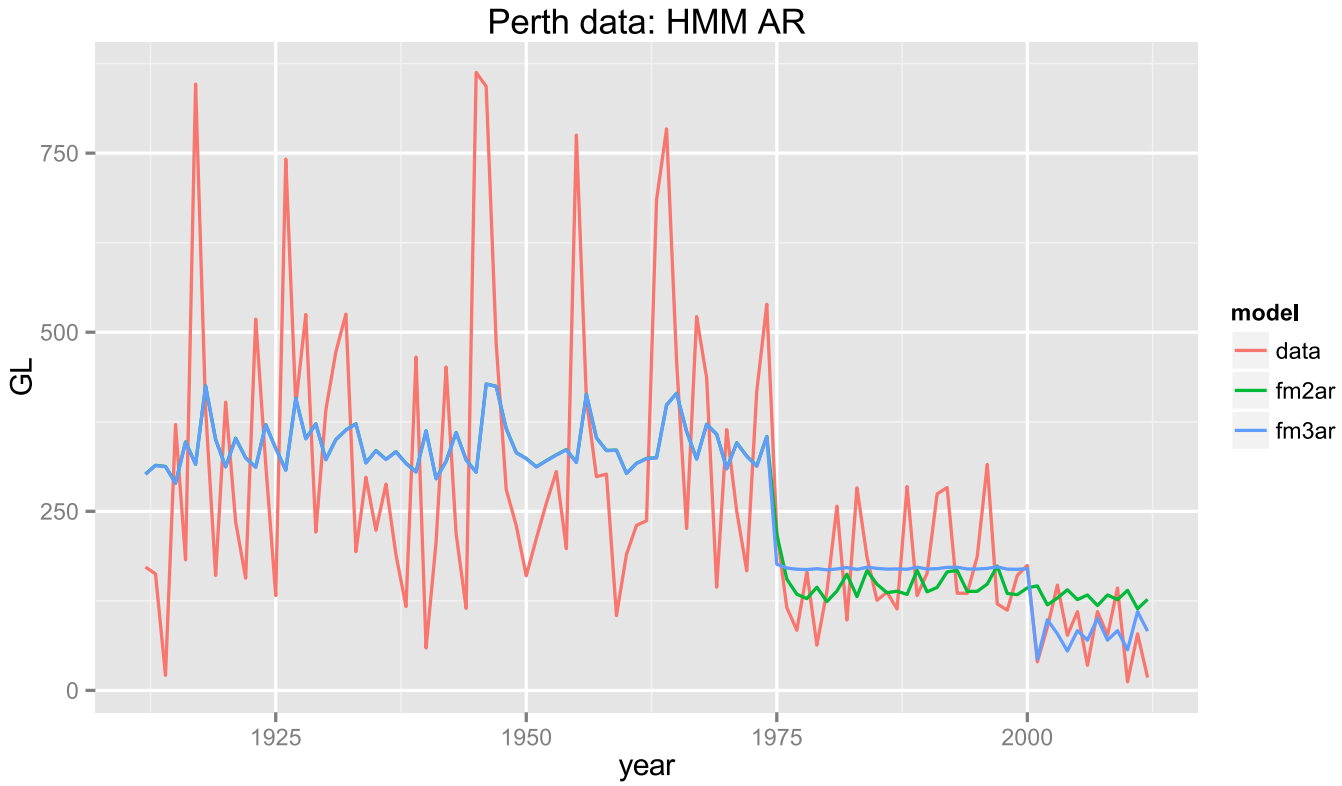


AR-like models with changepoints

```
mod2 <- depmix(water ~ wtmin1, data = perth[-1, ], ns = 2, trst = c(0.9, 0.1,
    0, 1), inst = c(1, 0))
set.seed(123)
fm2ar <- fit(mod2, verbose = FALSE)
for (i in 1:100) {
    tfm2ar <- fit(mod2, verbose = FALSE)
    if (logLik(tfm2ar) > logLik(fm2ar))
        fm2ar <- tfm2ar
}
mod3 <- depmix(water ~ wtmin1, data = perth[-1, ], ns = 3, trst = c(0.9, 0.1,
    0, 0, 0.9, 0.1, 0, 0, 1), inst = c(1, 0, 0))
set.seed(123)
fm3ar <- fit(mod3, verbose = FALSE)
for (i in 1:100) {
    try(tfm3ar <- fit(mod3, verbose = FALSE))
    if (logLik(tfm3ar) > logLik(fm3ar))
        fm3ar <- tfm3ar
}
```

AR-like models with changepoints

Change point in the same year as the 'ordinary' changepoint model



Model selection

		AIC	BIC
linear	lm1	1367.3819	1372.6318
	lmyr	1353.3413	1361.2163
	lmyr2	1344.8849	1355.3848
arima	ar1	1355.3371	1363.2121
	aryr	1348.9058	1359.4057
	aryr2	1343.6796	1356.8044
hmm linear	fm2	1332.82	1345.9449
	fm3	1327.9301	1348.9299
hmm ar	fm2ar	1318.7176	1337.0924
	fm3ar	1316.0646	1344.9393

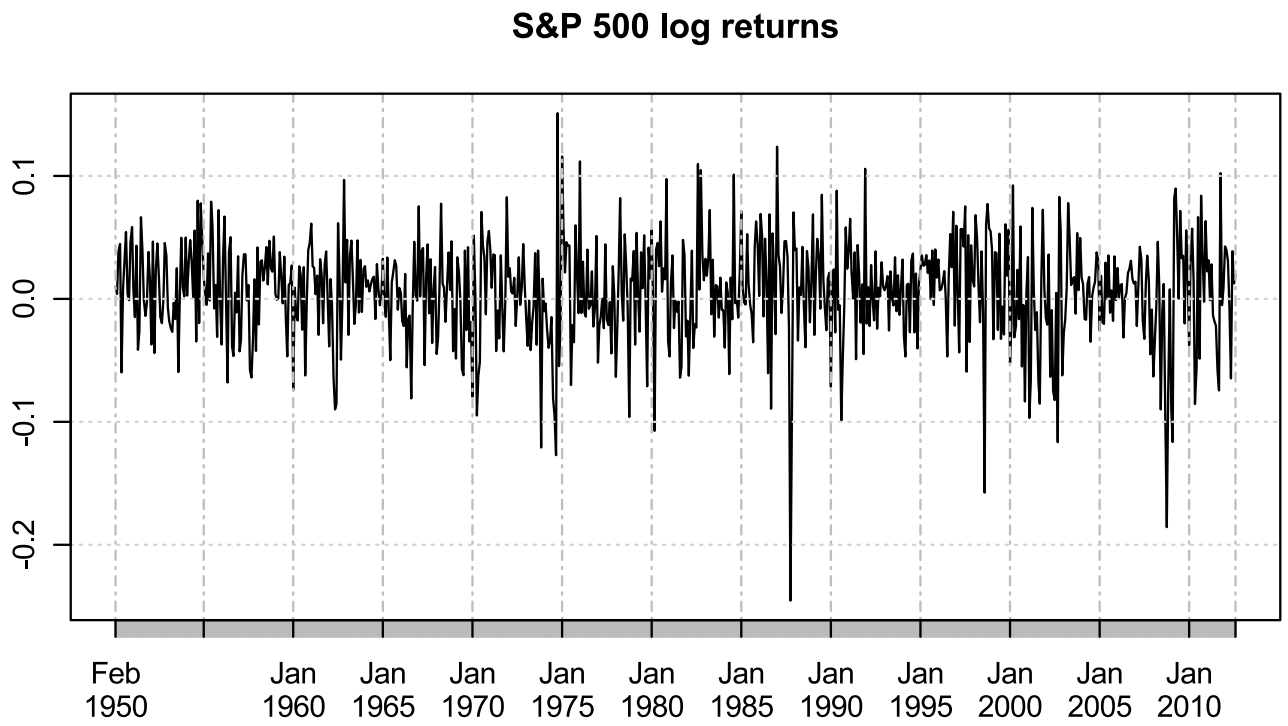
Example 2: Bull and bear markets

S & P 500 returns

```
library(TTR)
# load SP500 returns
Sys.setenv(tz = "UTC")
sp500 <- getYahooData("^GSPC", start = 19500101, end = 20120909, freq = "daily")
ep <- endpoints(sp500, on = "months", k = 1)
sp500 <- sp500[ep[2:(length(ep) - 1)]]
sp500$logret <- log(sp500$Close) - lag(log(sp500$Close))
sp500 <- na.exclude(sp500)
```

S & P 500 returns

```
plot(sp500$logret, main = "S&P 500 log returns")
```



Bull and bear markets?

```
mod <- depmix(logret ~ 1, nstates = 2, data = sp500)
set.seed(1)
fm2 <- fit(mod, verbose = FALSE)
```

```
converged at iteration 88 with logLik: 1348
```

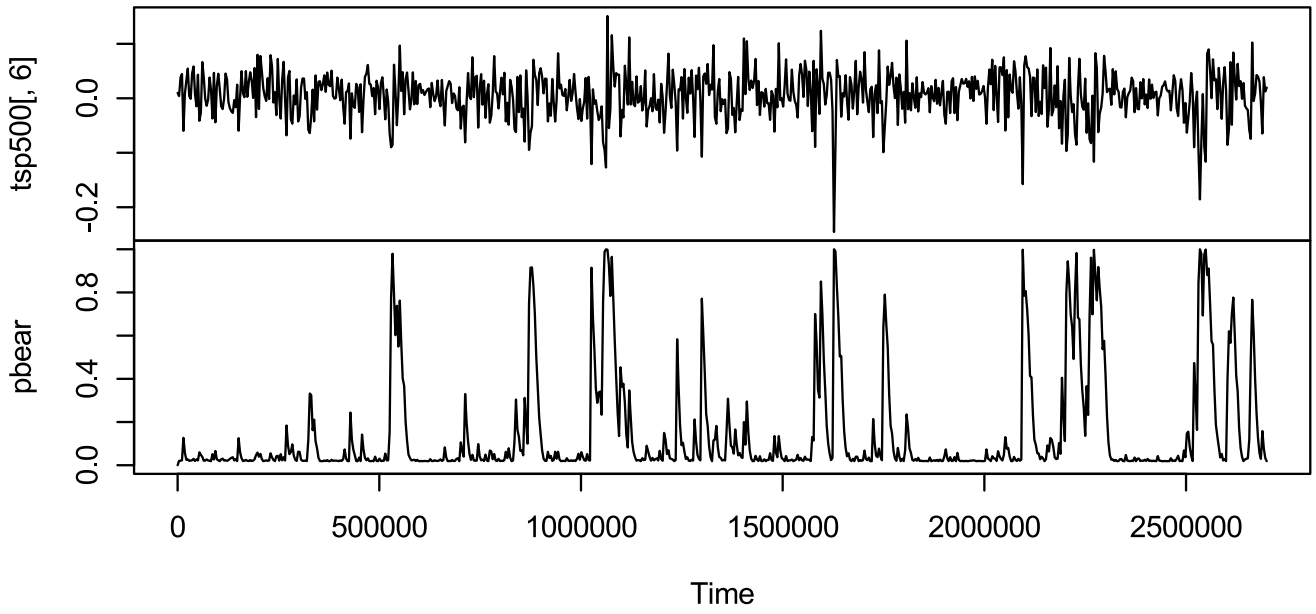
```
depmixS4::summary(fm2)
```

```
Initial state probabilties model
pr1 pr2
  0   1


Transition matrix
          toS1   toS2
fromS1 0.82151 0.1785
fromS2 0.03914 0.9609

Response parameters
Resp 1 : gaussian
    Re1.(Intercept)  Re1.sd
St1        -0.01505 0.06484
St2         0.01045 0.03378
```

Classification (1)

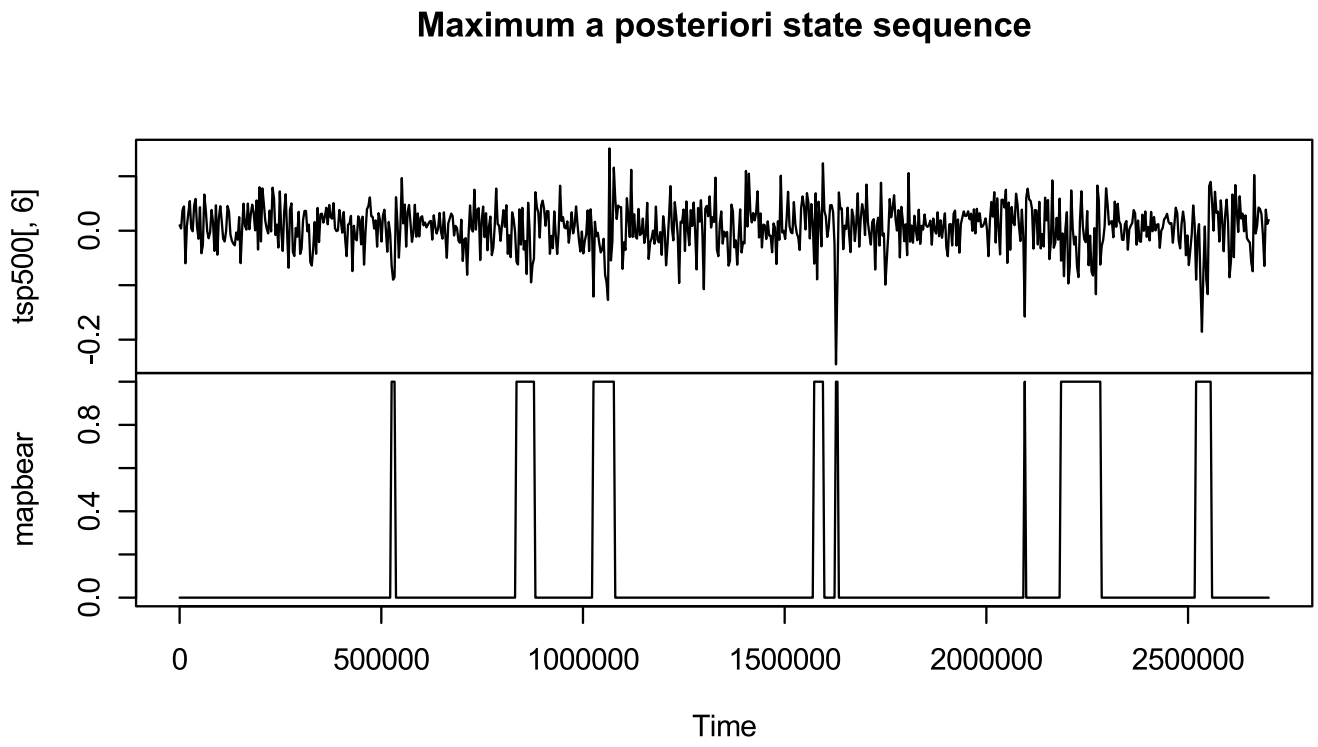
```
tsp500 <- as.ts(sp500)
pbear <- as.ts(posterior(fm2)[, 2])
tsp(pbear) <- tsp(tsp500)
plot(cbind(tsp500[, 6], pbear), main = "Posterior probability of state 1 (volatile, negative market
```

Posterior probability of state 1 (volatile, negative markets).



Classification (2)

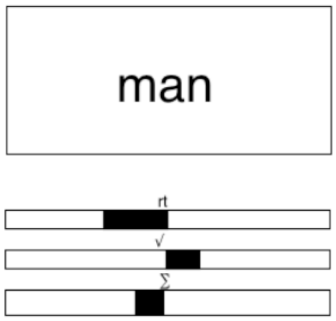
```
mapbear <- as.ts(posterior(fm2)[, 1] == 1)
tsp(mapbear) <- tsp(tsp500)
plot(cbind(tsp500[, 6], mapbear), main = "Maximum a posteriori state sequence")
```



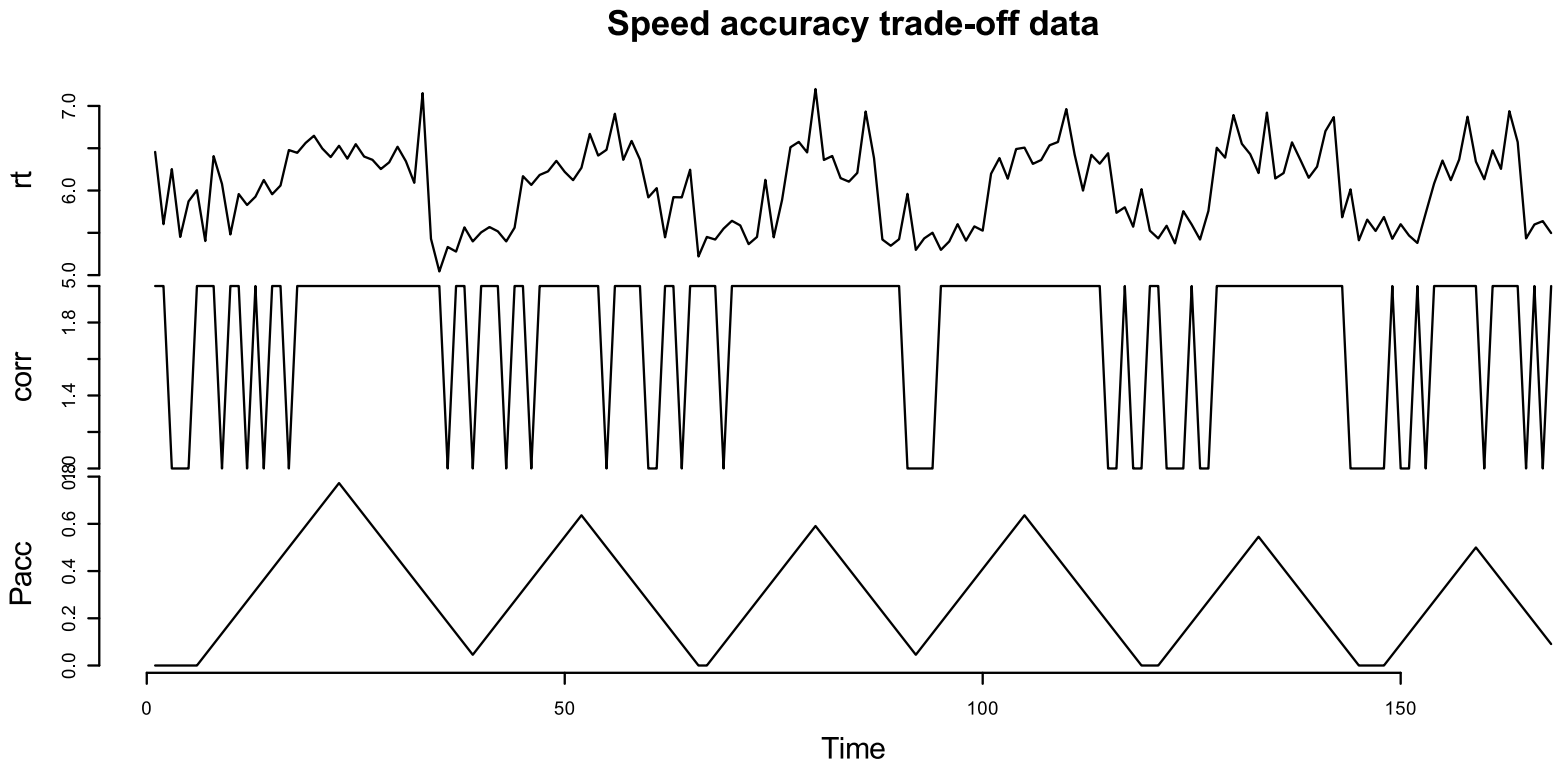
Example 3: Speed-accuracy trade-off

Speed accuracy trade-off task

1. lexical decision: word or non-word
2. reward for speed versus accuracy changes on a trial-by-trial basis
3. is the trade-off discrete of gradual?



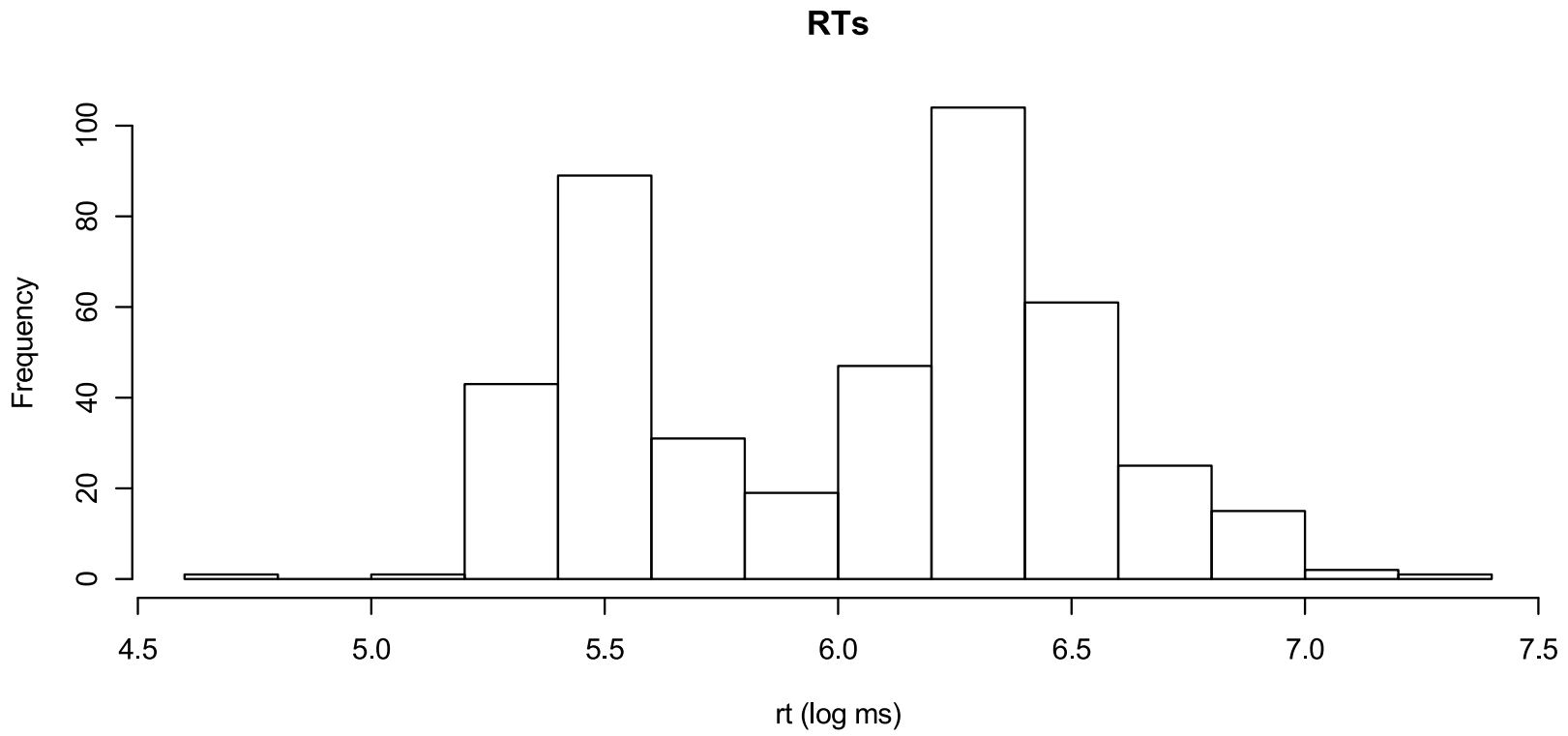
Speed accuracy trade-off data



1. What is the relationship between Pacc, RT and accuracy?
2. Is the trade-off continuous or discontinuous?

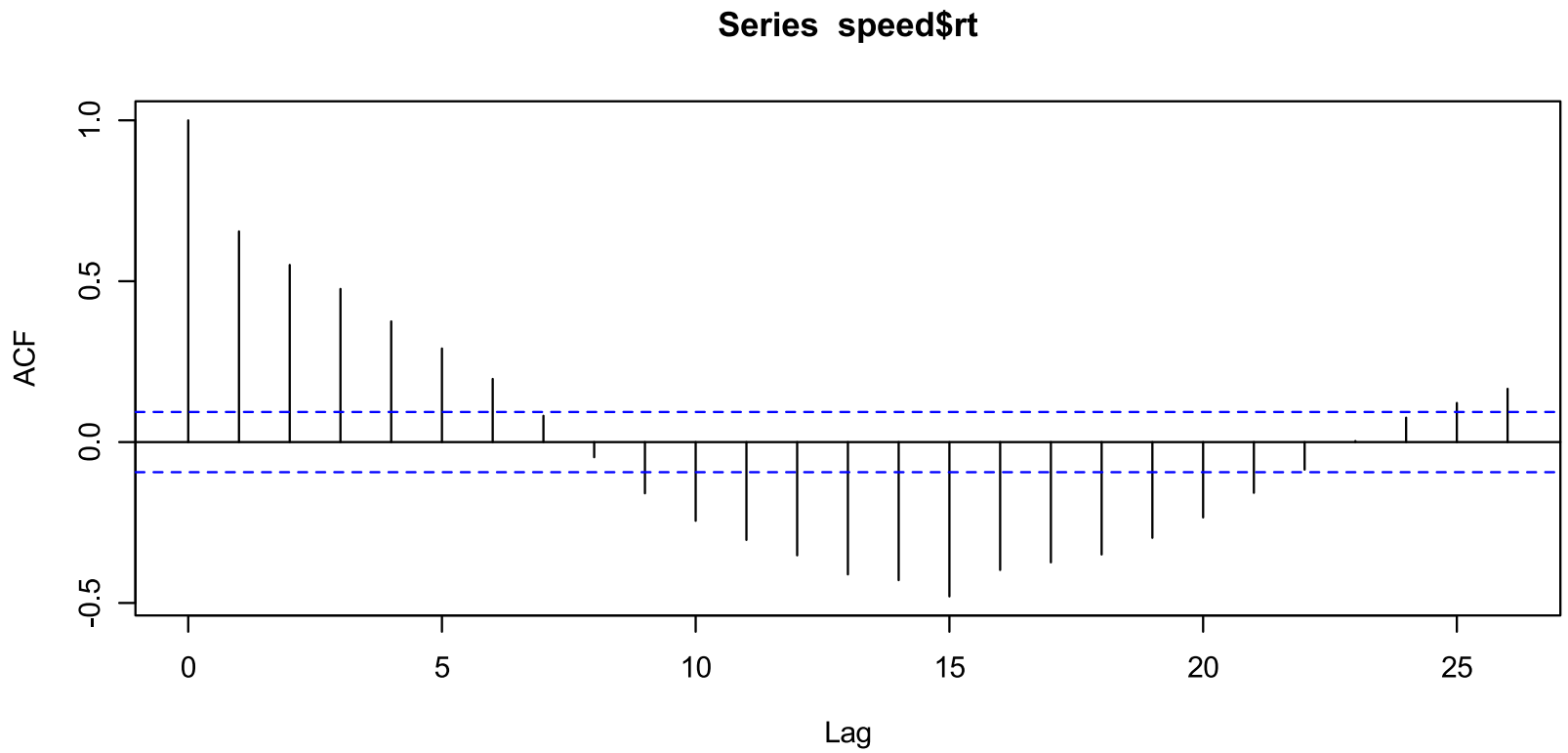
Speed accuracy trade-off

Preliminaries: data bi- or multi-modal?



Speed accuracy trade-off

Preliminaries: dependence in the data?



A simple hidden Markov model

```
data(speed) # included in depmixS4
mod1 <- depmix(list(rt~1,corr~1),
               data=speed,
               nstates=2,
               family=list(gaussian(),multinomial("identity")),
               ntimes=c(168,134,137))
# fit the model
set.seed(1234)
fmod1 <- fit(mod1, verbose=FALSE)
```

```
converged at iteration 24 with logLik: -296.1
```

Parameter estimates

```
depmixS4::summary(fmod1)
```

```
Initial state probabilties model
pr1 pr2
  0   1


Transition matrix
          toS1   toS2
fromS1 0.89886 0.1011
fromS2 0.08359 0.9164

Response parameters
Resp 1 : gaussian
Resp 2 : multinomial
    Re1.(Intercept) Re1.sd Re2.pr1 Re2.pr2
St1           5.521 0.2023 0.47207  0.5279
St2           6.392 0.2396 0.09851  0.9015
```

Accounting for the effect of Pacc

Using `Pacc` as a covariate on the transition probabilities

```
data(speed)
mod2 <- depmix(list(rt~1,corr~1),
               data=speed,
               transition=~Pacc,
               nstates=2,
               family=list(gaussian(),multinomial("identity")),
               ntimes=c(168,134,137))
# fit the model
set.seed(1234)
fmod2 <- fit(mod2, verbose=FALSE)
```

```
converged at iteration 24 with logLik: -249
```

Model parameters

```
depmixS4::summary(fmod2, which = "response")
```

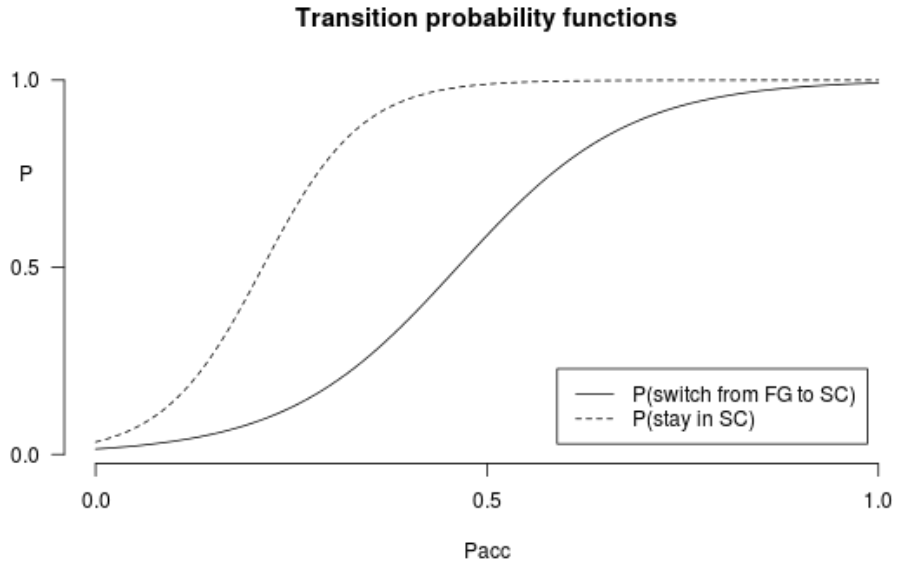
```
Response parameters
Resp 1 : gaussian
Resp 2 : multinomial
    Re1.(Intercept) Re1.sd Re2.pr1 Re2.pr2
St1           5.522 0.2029 0.47426  0.5257
St2           6.394 0.2374 0.09572  0.9043
```

Model parameters

```
depmixS4::summary(fmod2, which = "transition")
```

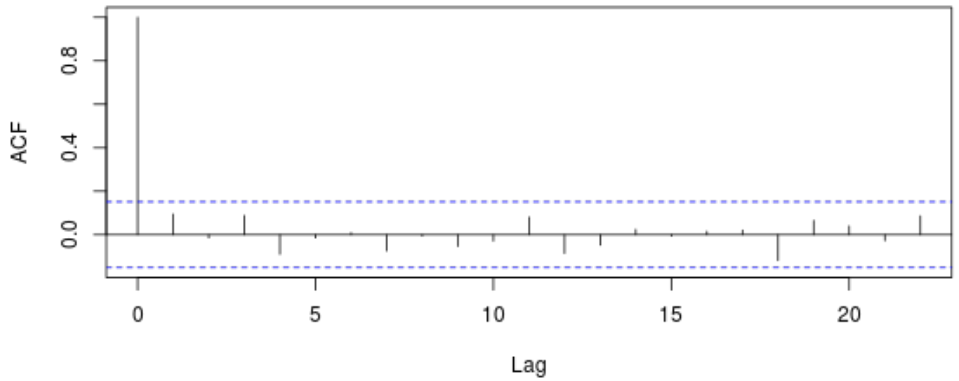
```
Transition model for state (component) 1
Model of type multinomial (mlogit), formula: ~Pacc
<environment: 0x1af6c00>
Coefficients:
            St1    St2
(Intercept)   0 -4.223
Pacc          0  9.134
Probalities at zero values of the covariates.
0.9856 0.01445

Transition model for state (component) 2
Model of type multinomial (mlogit), formula: ~Pacc
<environment: 0x1af6c00>
Coefficients:
            St1    St2
(Intercept)   0 -3.373
Pacc          0 15.804
Probalities at zero values of the covariates.
0.9669 0.03314
```

Transition probability function



Model checking

Autocorrelation function of the residuals of the RTs



Parameters constraints and inference

```
pars <- c(unlist(getpars(fmod2)))

# constrain the initial state probs to be 0 and 1 also constrain the
# guessing probs to be 0.5 and 0.5 (ie the probabilities of corr in state 1)
pars[1] = 0
pars[2] = 1  # this means the process will always start in state 2
pars[13] = 0.5
pars[14] = 0.5  # the corr parameters in state 1 are now both 0, corresponding the 0.5 prob
mod3 <- setpars(mod2, pars)
```

Parameters constraints and inference

```
# fix the parameters by setting:
free <- c(0, 0, rep(c(0, 1), 4), 1, 1, 0, 0, 1, 1, 1, 1)
# fit the model
fmod3 <- fit(mod3, fixed = !free)
```

```
Iter: 1 fn: 249.2129     Pars:  -4.22283  9.13384 -3.37335 15.80431  5.52172  0.20292  6.39370  0.2
solnp--> Completed in 1 iterations
```

Parameters constraints and inference

Likelihood ratio test on the fitted models (note: as some of the parameters in mod3 are on the bound of the parameter space, the likelihood-ratio test is not exactly valid...)

```
# likelihood ratio not significant, hence fmod3 may be preferred to fmod2
llratio(fmod2, fmod3)
```

```
log Likelihood ratio (chi^2): 0.481 (df=2), p=0.786.
```