WIRTSCHAFTS
UNIVERSITÄT
WIEN

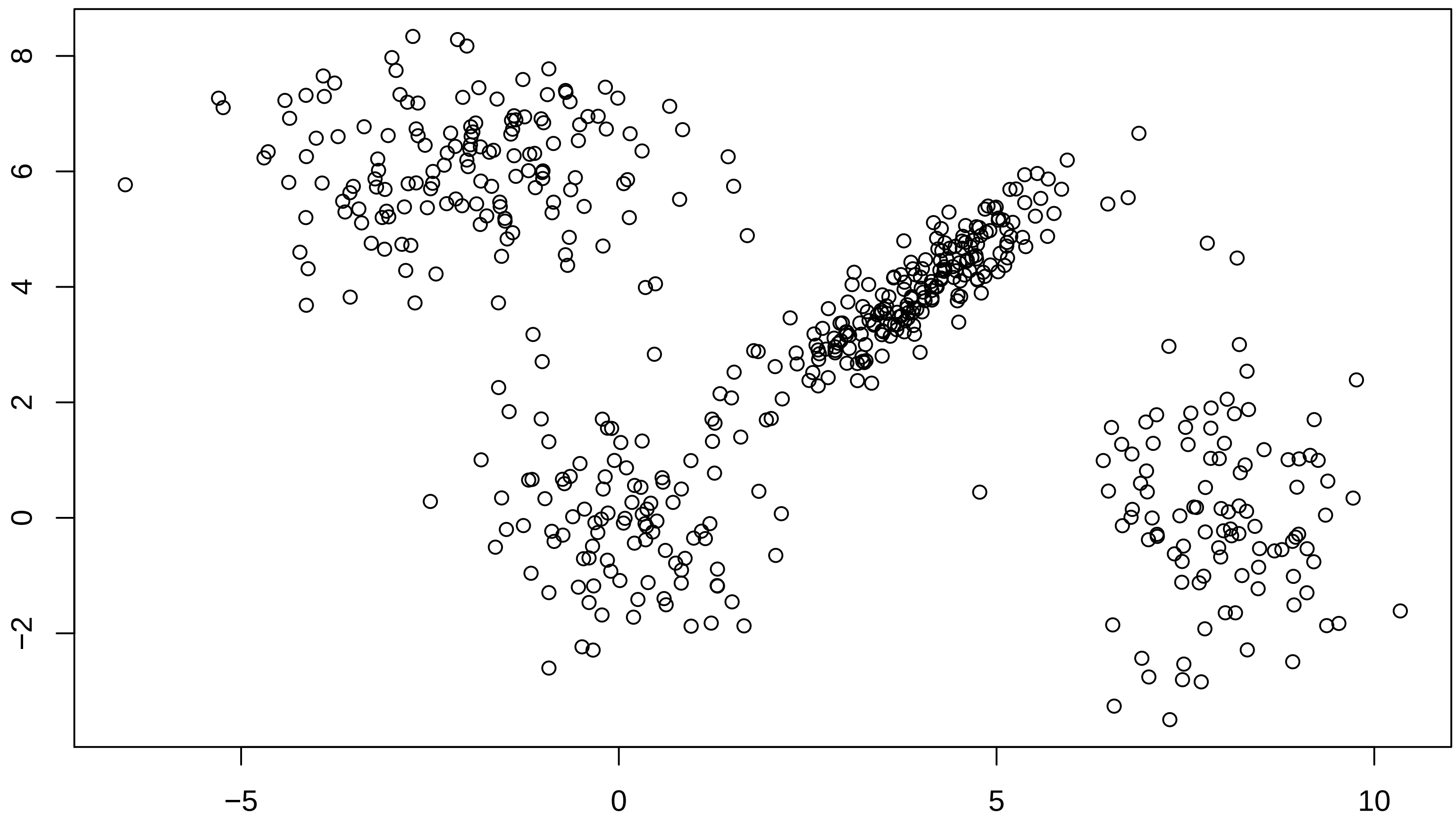
FlexMix: Flexible fitting of finite mixtures with the EM algorithm

Bettina Grün
WU Wien

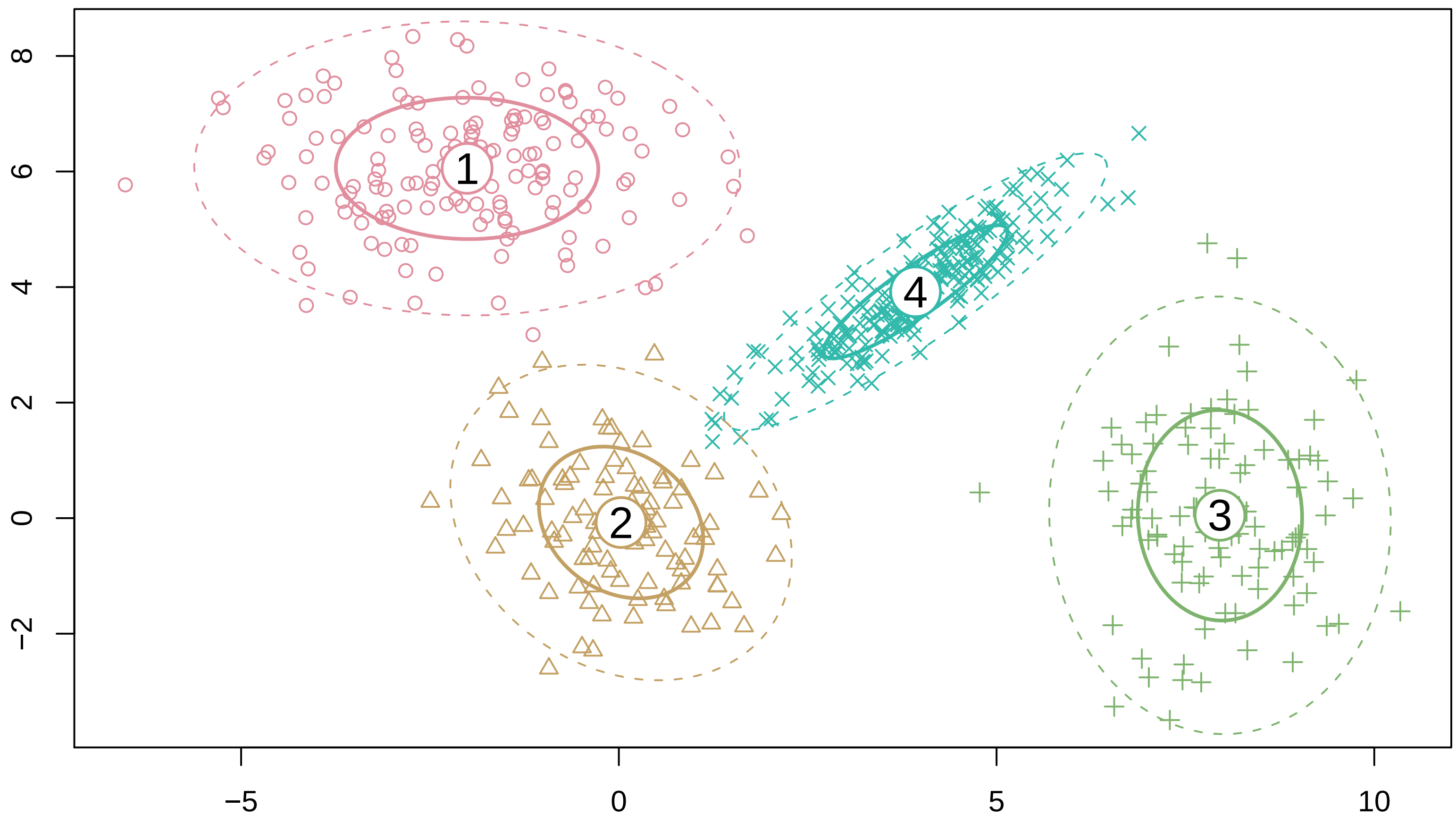
Friedrich Leisch
LMU München

useR! 2008, August 12–14 2008

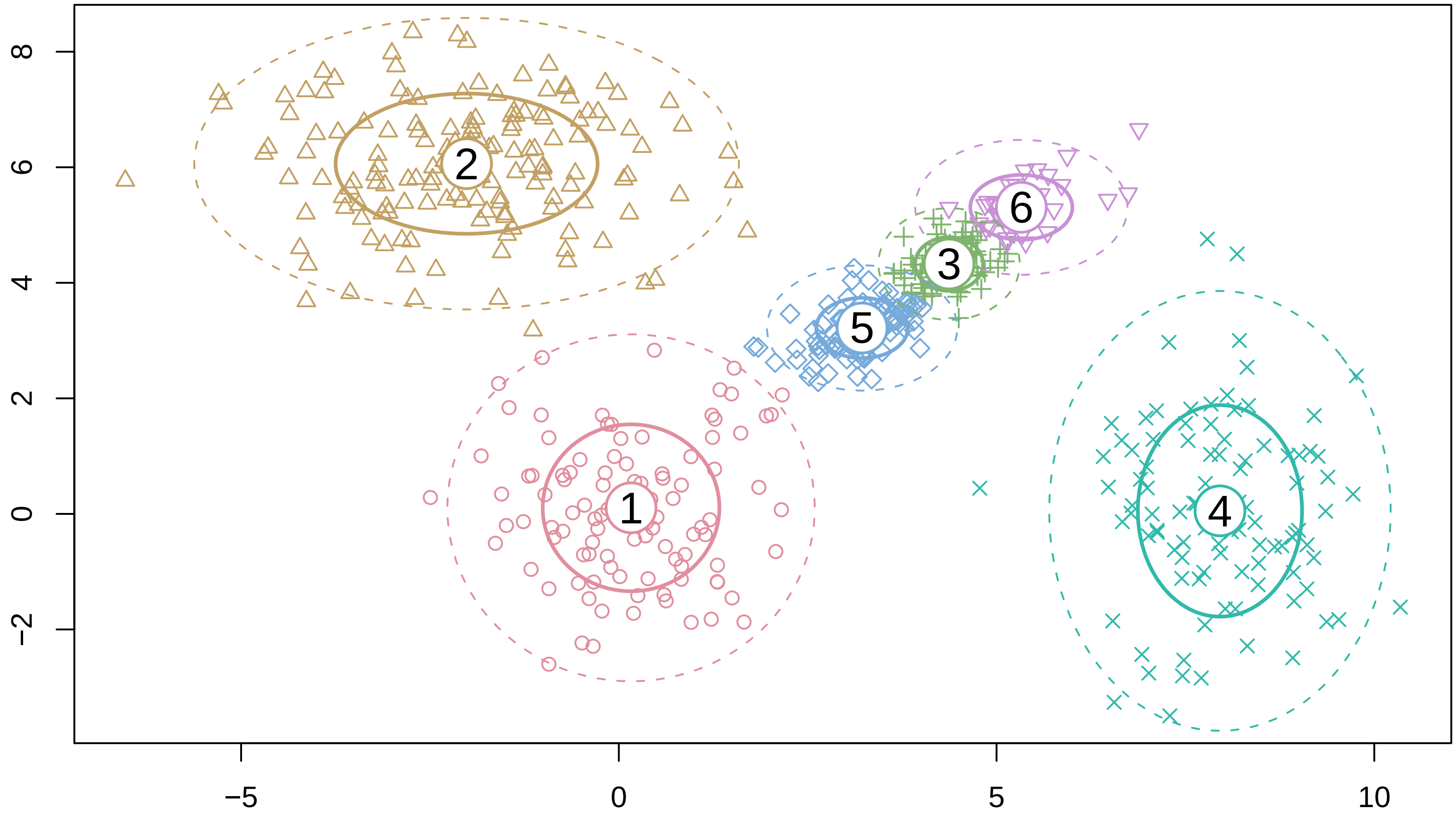
Finite mixture models



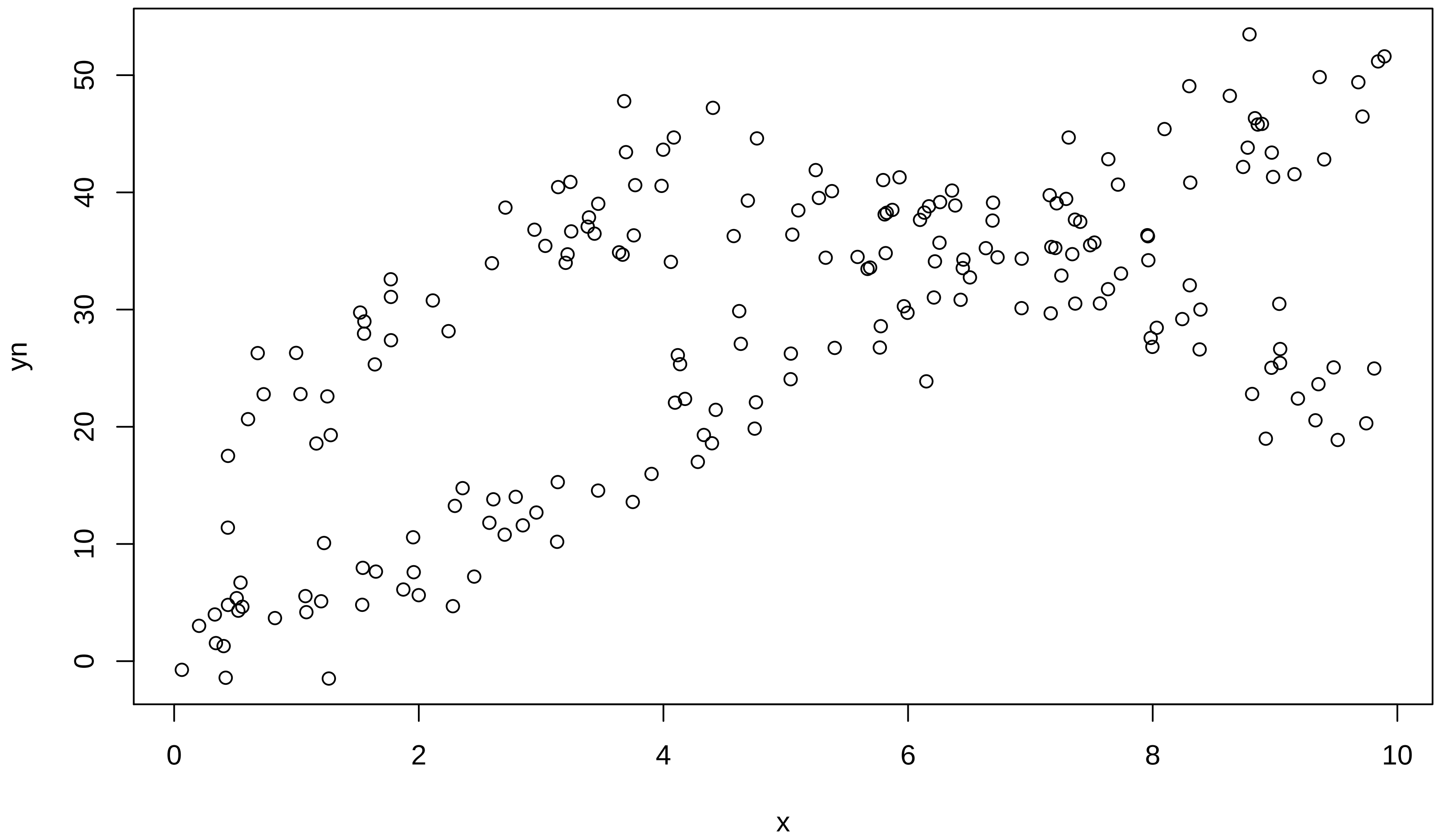
Finite mixture models



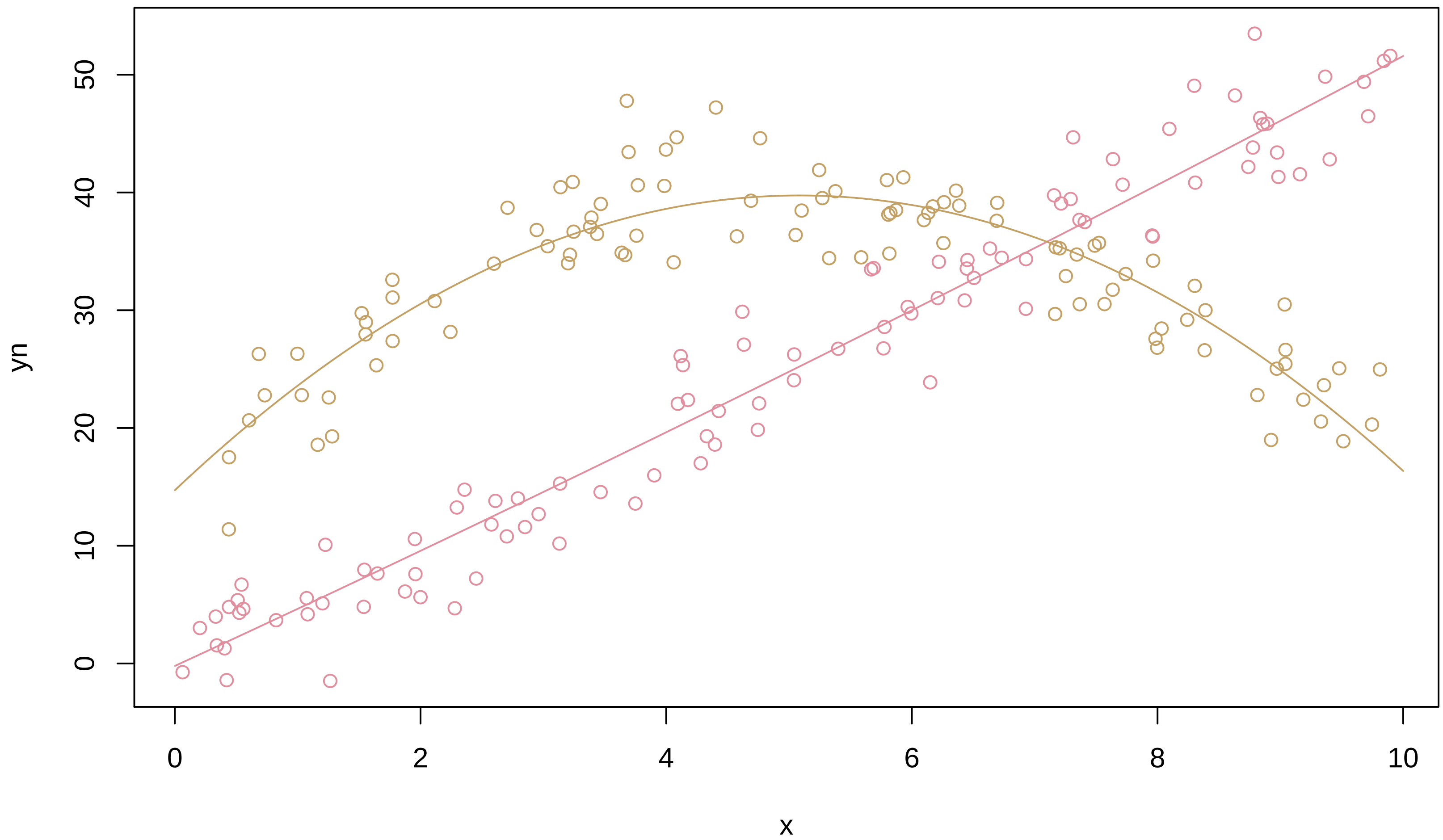
Finite mixture models



Finite mixture models



Finite mixture models



Finite mixture models

The finite mixture density is given by

$$
\begin{aligned}
h(y|x, w, \psi) &= \sum_{k=1}^{K} \pi_k(w, \alpha) f_k(y|x, \theta_k) \\
&= \sum_{k=1}^{K} \pi_k(w, \alpha) \prod_{d=1}^{D} f_{kd}(y_d|x_d, \theta_{kd}),
\end{aligned}
$$

with

$$
\forall w : \sum_{k=1}^{K} \pi_k(w, \alpha) = 1 \quad \wedge \quad \pi_k(w, \alpha) > 0 \, \forall k.
$$

The posterior probabilities are given by

$$
\tau_k(y|x, \psi) = \frac{\pi_k(w, \alpha) f_k(y|x, \theta_k)}{\sum_{l=1}^{K} \pi_l(w, \alpha) f_l(y|x, \theta_l)}.
$$

EM algorithm

- General method for ML estimation in a missing data setting
  $\longrightarrow$ component membership
- Iterates between

  E-step: determines the a-posteriori probabilities

  M-step: maximizes the complete likelihood where the missing component memberships are replaced
  $\longrightarrow$ weighted ML problem of the component specific model and the concomitant variable model
- Likelihood is increased in each step
  $\longrightarrow$ converges to a local optimum if the likelihood is bounded
- Variants: additional step between E- and M-step
  - **Stochastic EM (SEM):** assigns each observation to one component by drawing from the multinomial distribution induced by the a-posteriori probabilities
  - **Classification EM (CEM):** assigns each observation to the component with the maximum a-posteriori probability

FlexMix Design

- Primary goal is extensibility: ideal for trying out new mixture models
- No replacement of specialized mixture packages like **mclust**, but complement
- Usage of S4 classes and methods
- Formula-based interface
- Multivariate responses:
  - **Combination of univariate families:** assumption of independence (given x), each response may have its own model formula, i.e., a different set of regressors
  - **multivariate families:** if family handles multivariate response directly, then arbitrary multivariate response distributions are possible

Fit function `flexmix()`

- `flexmix()` takes the following arguments:

  formula: A symbolic description of the model to be fit. The general form is `y~x|g` where `y` is the response, `x` the set of predictors and `g` an optional grouping factor for repeated measurements.

  data: An optional data frame containing the variables in the model.

  k: Number of clusters (not needed if `cluster` is specified).

  cluster: Either a matrix with `k` columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations.

  model: Object of class `"FLXM"` or list of these objects.

  concomitant: Object of class `"FLXP"`.

  control: Object of class `"FLXcontrol"` or a named list.

  - repeated calls of `flexmix()` with `stepFlexmix()`
  - returns an object of class `"flexmix"`

Controlling the EM algorithm

- `"FLXcontrol"`: for the overall behaviour of the EM algorithm:

  iter.max: Maximum number of iterations

  minprior: Minimum prior probability for components

  verbose: If larger than zero, then `flexmix()` gives status messages each verbose iterations.

  classify: One of “auto”, “weighted”, “CEM” (or “hard”), “SEM” (or “random”).

  For convenience `flexmix()` also accepts a named list of control parameters with argument name completion, e.g.

  ```
  flexmix(..., control=list(class="r"))
  ```

Variants of mixture models

Component specific models: `FLXMxxx()`

- Model-based clustering: `FLXMCxxx()`
  - `FLXMCmvnorm()`
  - `FLXMCmvbinary()`
  - `FLXMCmvpois()`
  - ...
- Clusterwise regression: `FLXMRxxx()`
  - `FLXMRglm()`
  - `FLXMRglmfix()`
  - `FLXMRziglm()`
  - ...

Concomitant variable models: `FLXPxxx()`

- `FLXPconstant()`
- `FLXPmultinom()`

Methods for `"flexmix"` objects

- `show()`, `summary()`: some information on the fitted model
- `plot()`: rootogram of posterior probabilities
- `refit()`: refits an estimated mixture model to obtain other additional information, such as for example the variance-covariance matrix
- `logLik()`, `BIC()`, `...`: obtain log-likelihood and model fit criteria
- `parameters()`, `priors()`: obtain component specific or concomitant variable model parameters and prior class probabilities/component weights
- `posteriors()`, `clusters()`: obtain a-posteriori probabilities and assignments to the maximum a-posteriori probability
- `fitted()`, `predict()`: fitted and predicted (component-specific) values

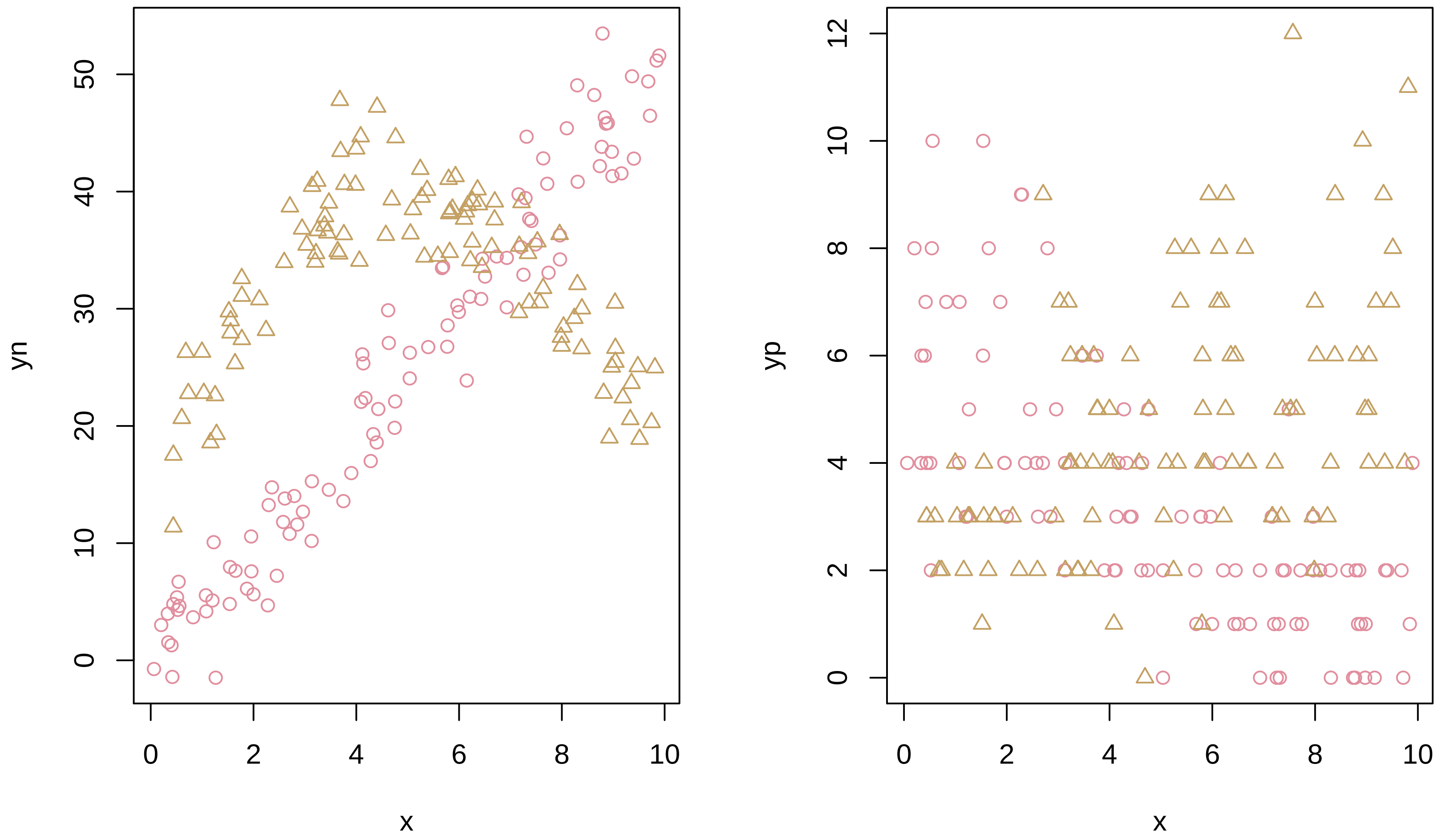
Example: artificial data

- 200 observations from a mixture given by

$$h(y|x,\psi) = \frac{1}{2}\text{Normal}(yn|15 + 10x - x^2, 9)\text{Poi}(yp|e^{1+0.1x}) + \\ + \frac{1}{2}\text{Normal}(yn|5x, 9)\text{Poi}(yp|e^{2-0.2x})$$

where Normal$(y|\mu, \sigma^2)$ is the Gaussian distribution and Poi$(y|\lambda)$ the Poisson distribution.

Example: artificial data



Example: artificial data

```
> set.seed(1802)
> library("flexmix")
> data("NPreg")
> Model_n <- FLXMRglm(yn ~ . + I(x^2))
> Model_p <- FLXMRglm(yp ~ ., family = "poisson")
> m1 <- flexmix(. ~ x, data = NPreg, k = 2, model = list(Model_n, Model_p),
+               control = list(verbose = 10))
Classification: weighted
  10 Log-likelihood :   -1044.7688
  11 Log-likelihood :   -1044.7678
converged
> m1
Call:
flexmix(formula = . ~ x, data = NPreg, k = 2, model = list(Model_n,
    Model_p), control = list(verbose = 10))

Cluster sizes:
  1   2
 96 104

convergence after 11 iterations
```

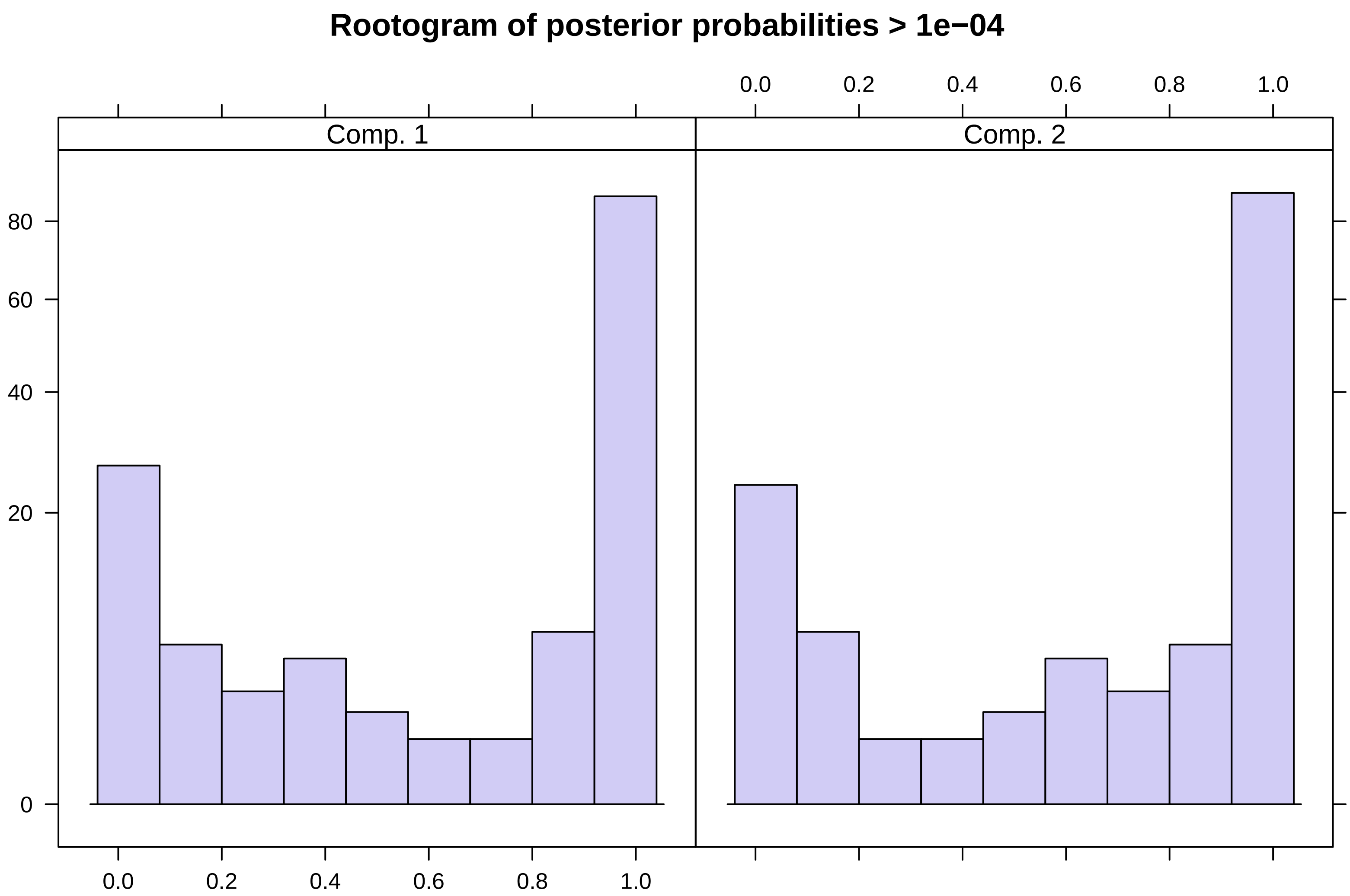
Example: artificial data

```
> summary(m1)
Call:
flexmix(formula = . ~ x, data = NPreg, k = 2, model = list(Model_n,
    Model_p), control = list(verbose = 10))

       prior size post>0 ratio
Comp.1 0.493   96    139 0.691
Comp.2 0.507  104    137 0.759

'log Lik.' -1044.768 (df=13)
AIC: 2115.536   BIC: 2158.414
> plot(m1)
```

Example: artificial data

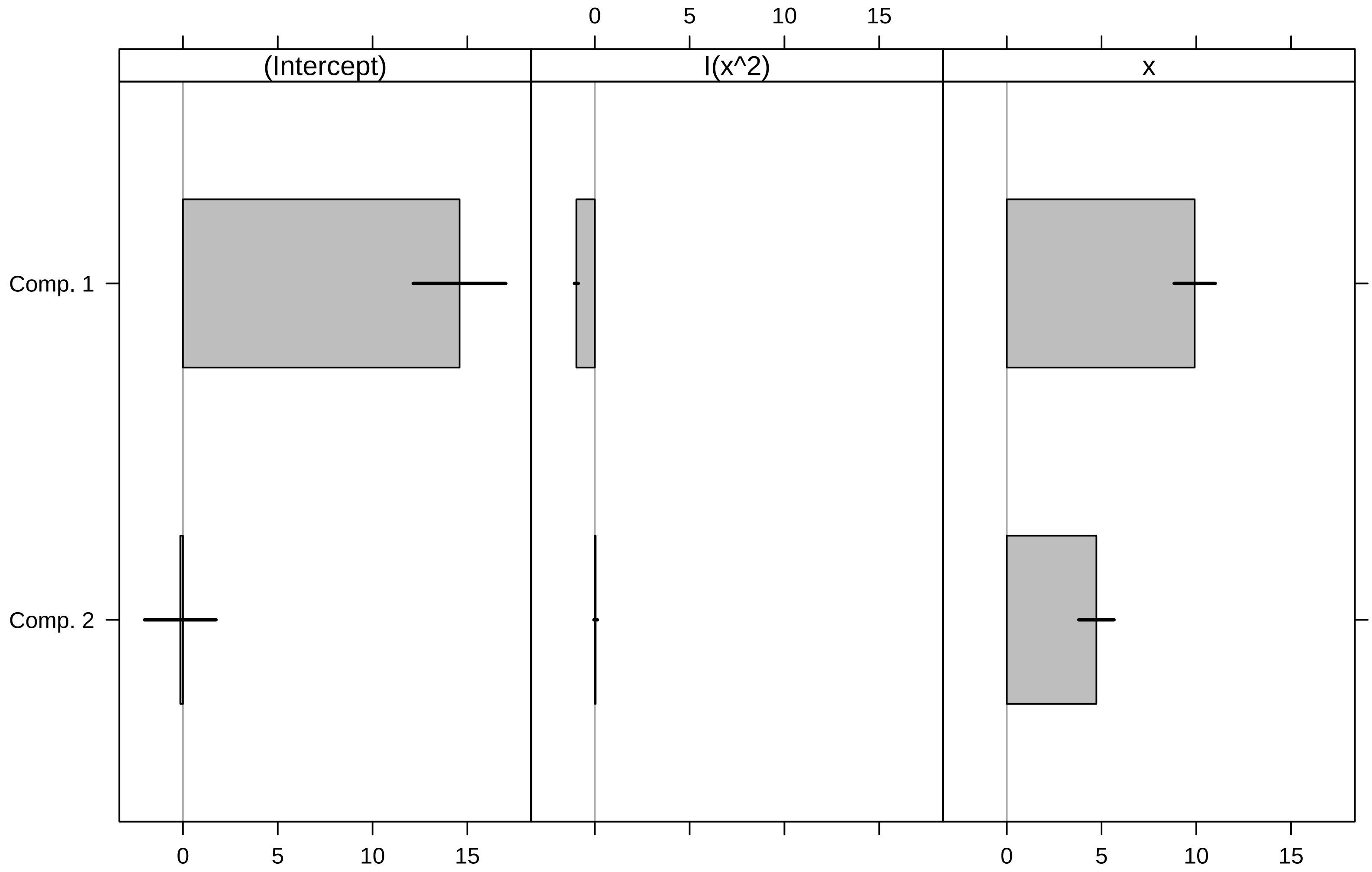


Example: artificial data

```
> m1_refit <- refit(m1)
> summary(m1_refit, which = "model", model = 1)
$Comp.1
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) 14.58965    1.24635  11.706 < 2.2e-16 ***
x            9.91572    0.55294  17.933 < 2.2e-16 ***
I(x^2)      -0.97578    0.05201 -18.762 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

$Comp.2
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.140549   0.961868 -0.1461   0.8838
x            4.732610   0.474428  9.9754   <2e-16 ***
I(x^2)       0.042722   0.046890  0.9111   0.3622
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> plot(m1_refit, bycluster = FALSE)
```

Example: artificial data

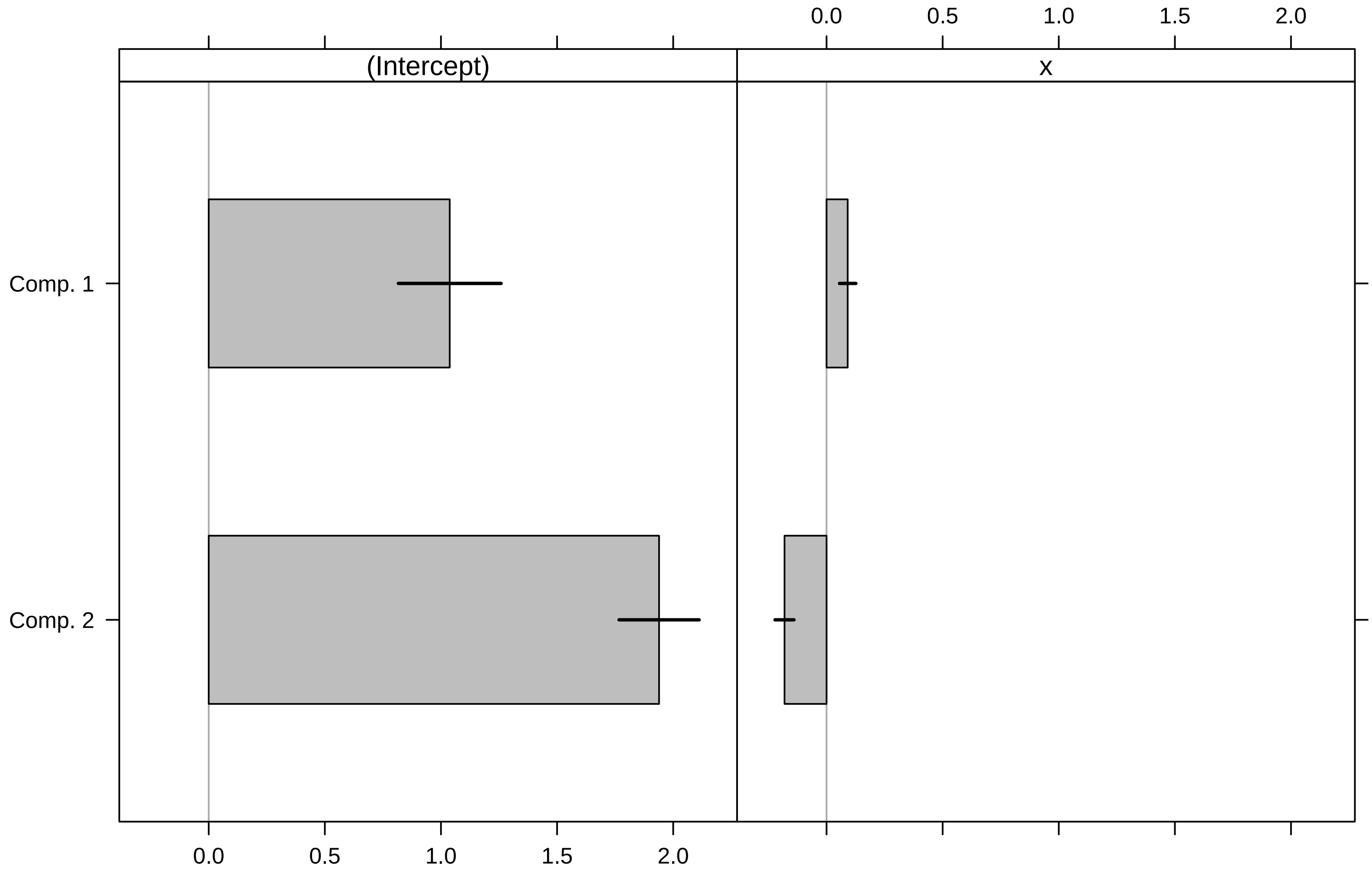


Example: artificial data

```
> summary(m1_refit, which = "model", model = 2)
$Comp.1
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) 1.037805   0.113005  9.1837 < 2.2e-16 ***
x           0.091034   0.017994  5.0592  4.21e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

$Comp.2
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  1.939213   0.088046 22.0249 < 2.2e-16 ***
x           -0.180959   0.020856 -8.6767 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> plot(m1_refit, model = 2, bycluster = FALSE)
```

Example: artificial data



Example: artificial data

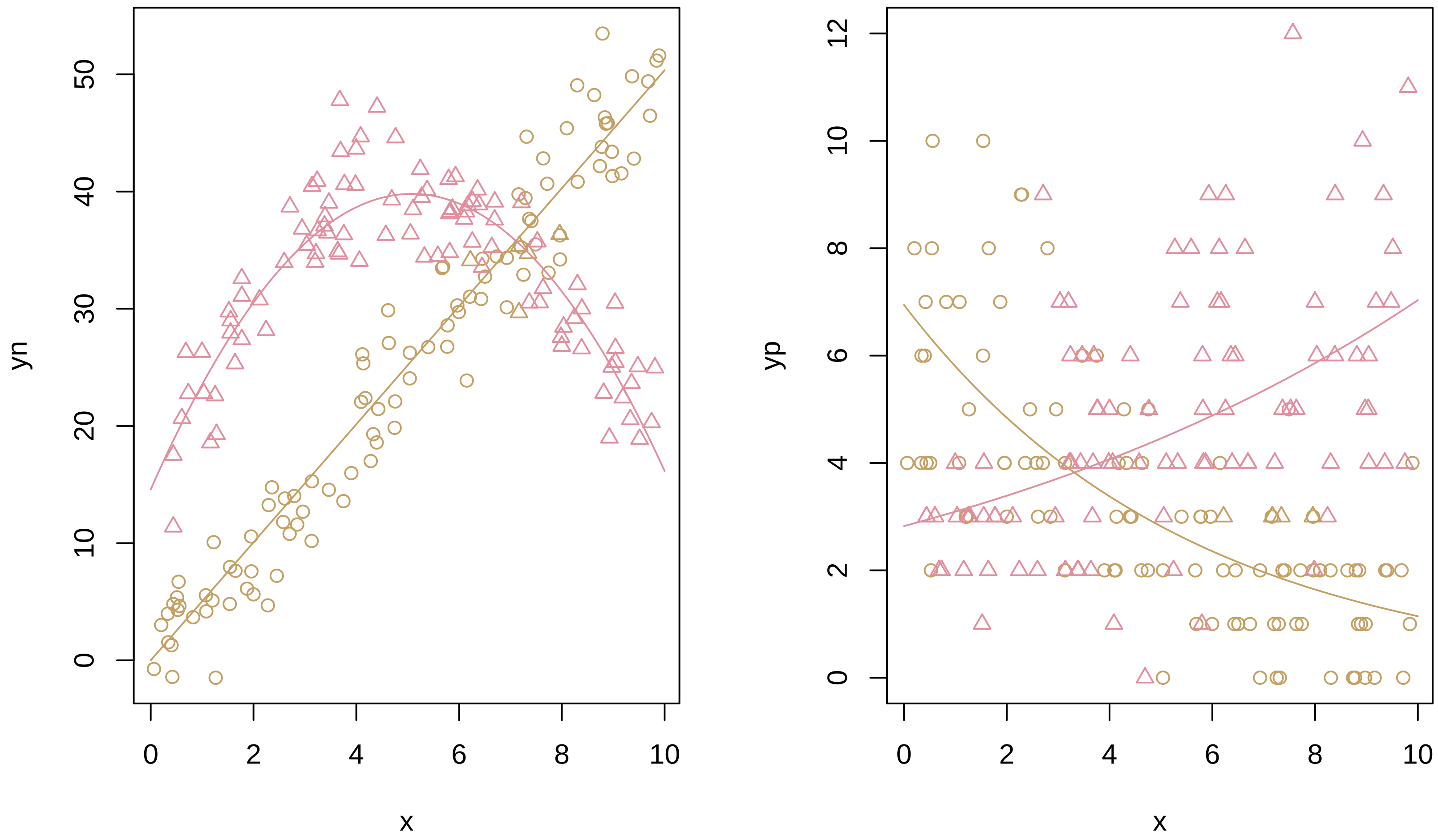
```
> Model_n2 <- FLXMRglmfix(yn ~ . + 0, nested = list(k = c(1, 1),
+                                        formula = c(~ 1 + I(x^2), ~ 0)))
> m2 <- flexmix(. ~ x, data = NPreg, cluster = posterior(m1),
+               model = list(Model_n2, Model_p))
> m2
Call:
flexmix(formula = . ~ x, data = NPreg, cluster = posterior(m1),
    model = list(Model_n2, Model_p))

Cluster sizes:
  1   2
 96 104

convergence after 3 iterations

> c(BIC(m1), BIC(m2))
[1] 2158.414 2149.956
```

Example: artificial data



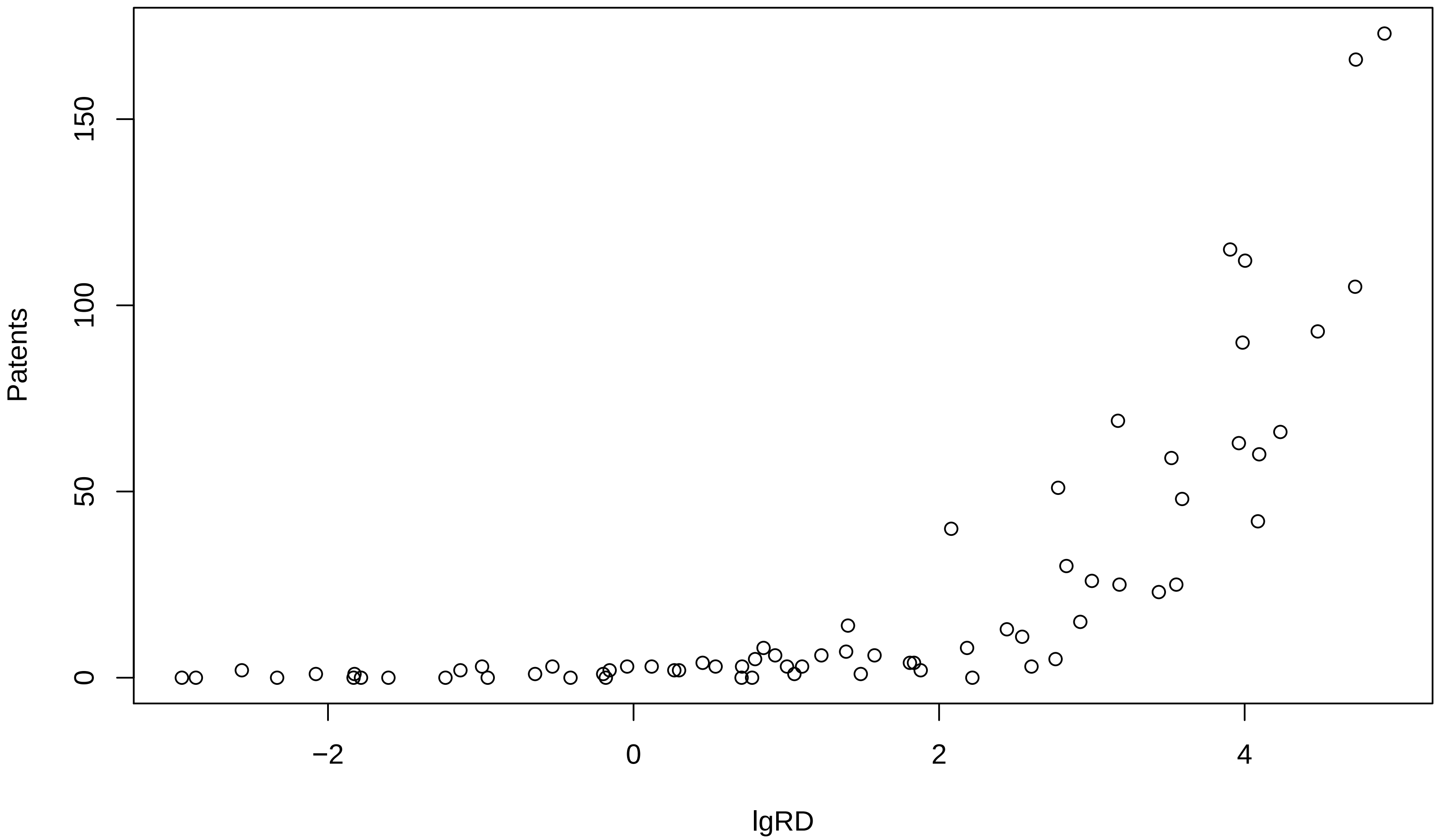
Example: `patent` **data**

given in Wang, Cockburn and Puterman (1998)

- 70 observations from pharmaceutical and biomedical companies in 1976 taken from the National Bureau of Economic Research R&D Masterfile
- Variables:
  - number of patent applications
  - R&D spending
  - sales in millions

$$
\begin{aligned}
h(\text{Patents} \mid \text{lgRD}, \text{RDS}, \psi) &= \sum_{s=1}^{S} \pi_s(\text{RDS}, \alpha) \text{Poi}(\text{Patents} \mid \lambda_s) \\
\log(\lambda_s) &= \beta_1^s + \text{lgRD} \cdot \beta_2^s
\end{aligned}
$$

Example: `patent` data



Example: `patent` data

```
> data("patent")
> Conc <- FLXPmultinom(~ RDS)
> (m_step <- stepFlexmix(Patents ~ lgRD, k = 2:5, nrep = 5,
+                        concomitant = Conc, data = patent,
+                        model = FLXMRglm(family = "poisson")))
2 : * * * * *
3 : * * * * *
4 : * * * * *
5 : * * * * *

Call:
stepFlexmix(Patents ~ lgRD, concomitant = Conc, data = patent,
    model = FLXMRglm(family = "poisson"), k = 2:5, nrep = 5)

  iter converged k k0    logLik      AIC      BIC      ICL
2   26      TRUE 2  2 -218.4911 448.9822 462.4731 473.6855
3   29      TRUE 3  3 -197.6752 415.3504 437.8354 453.5647
4   39      TRUE 4  4 -193.8785 415.7571 447.2360 471.2140
5   37      TRUE 5  5 -192.6904 421.3808 461.8537 512.0378
```

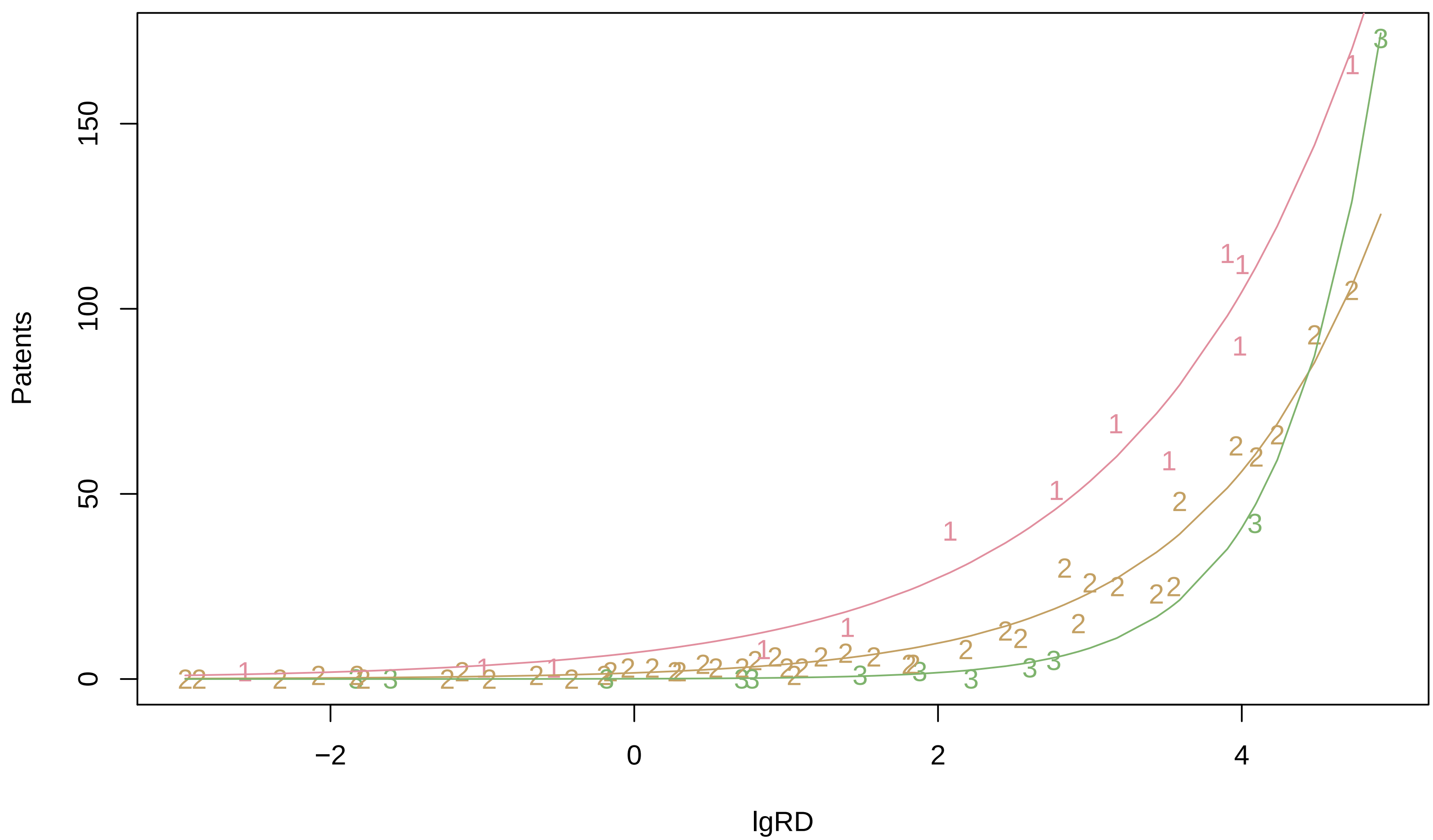
Example: `patent` data

```
> (m1 <- getModel(m_step, "BIC"))
Call:
stepFlexmix(Patents ~ lgRD, concomitant = Conc, data = patent,
    model = FLXMRglm(family = "poisson"), k = 3, nrep = 5)

Cluster sizes:
 1  2  3
13 45 12

convergence after 29 iterations
```

Example: `patent` data

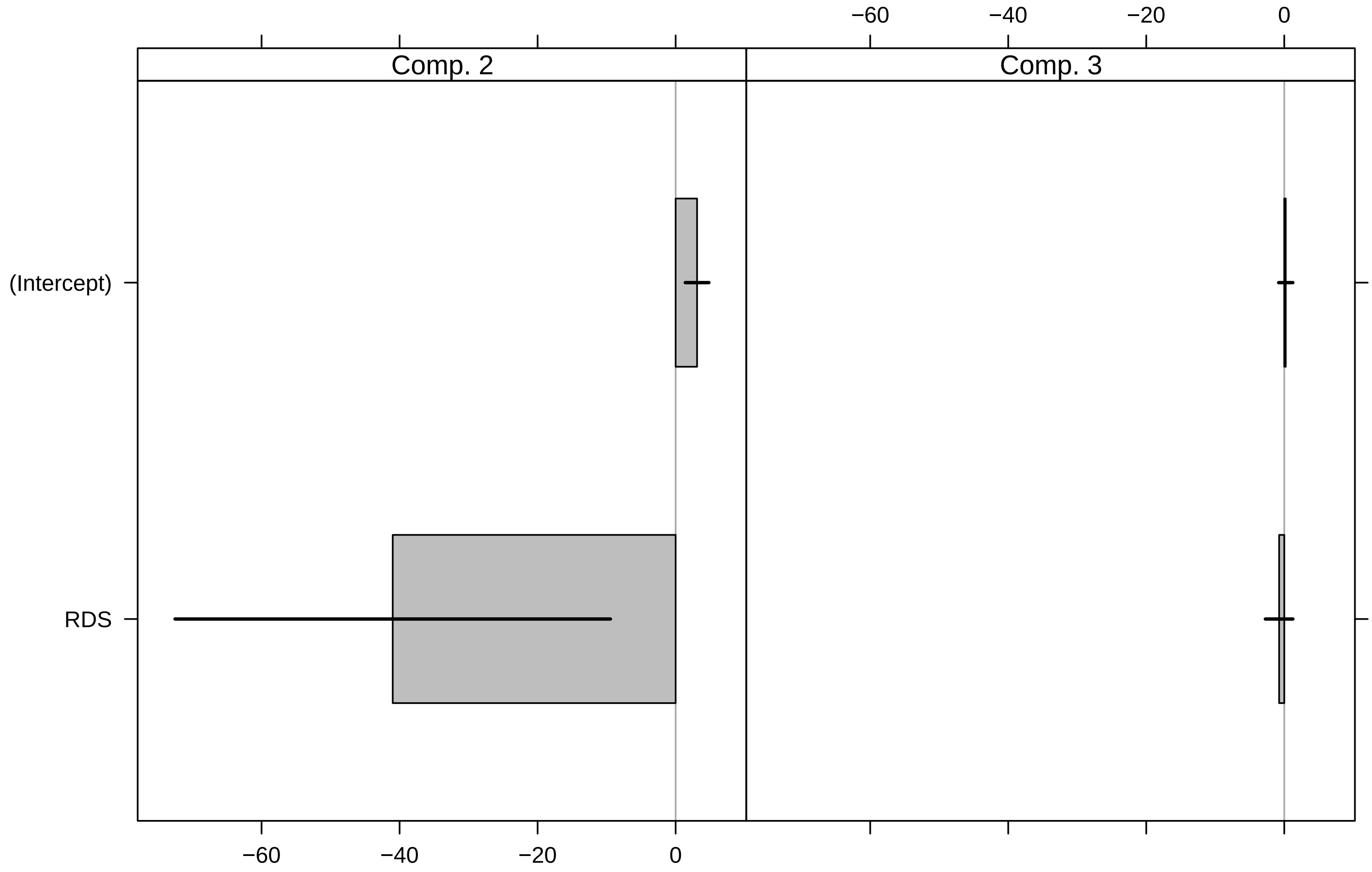


Example: patent data

```
> m1_refit <- refit(m1)
> summary(m1_refit, which = "concomitant")
$Comp.2
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)   3.10653    0.87491  3.5507 0.0003842 ***
RDS         -40.99625   16.09568 -2.5470 0.0108642 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

$Comp.3
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.21385    0.52411  0.4080   0.6833
RDS         -0.74566    1.01832 -0.7322   0.4640
> plot(m1_refit, which = "concomitant")
```

Example: `patent` data



Summary

- **FlexMix** offers an easy and extensible way of EM-based estimation of finite mixture models in R.
  $\Rightarrow$ Users are able to write their own model drivers to fit new variants of mixture models.
- **FlexMix** currently contains only interpreted code.
  $\Rightarrow$ An efficient M-step is crucial to fit large models in reasonable time.
  $\Rightarrow$ Popular models are re-implemented in C by Arijit Das as a "Google Summer of Code 2008" project.

For more information see

`http://cran.r-project.org/package=flexmix`.