R-package pendensity - Density Estimation with a Penalized Mixture Approach

Christian Schellhase
Centre for Statistics, Bielefeld University
Department for Business Administration and Economics

Version 0.2.2

June 22, 2010

Abstract

We give an overview about our R-package pendensity. Therefore, we briefly discuss the several options and advantages of our approach. First of all, the univariate density estimation without any covariate is discussed, followed by the extension to covariate dependencies. Furthermore, the variety of possibilities of the plot function are discussed. Thereby, we focus on examples of stock prices.

1 Simple Example

For a short introduction, we show two simple examples.

1.1 Example without any covariate

We start directly and load the corresponding R-package

```
> library(pendensity)
```

We consider a random sample of size $N = 100$ from the standard normal distribution. To estimate the corresponding density, we can easily run the estimation with `pendensity()`.

```
> set.seed(27)
> y <- rnorm(100)
> test <- pendensity(y ~ 1)
```

We can plot the estimated density using the corresponding function `plot()`, see Figure 1.

1.2 Example with covariate

We show a simple example, with sample size $N = 400$, separated in two groups of equal size $N = 200$. In between these groups of normal distributed values, we shift the mean by 0.5 for a constant variance.

```
> set.seed(27)
> x <- rep(c(0, 1), 200)
> y <- rnorm(400, x * 0.5, 1)
> test2 <- pendensity(y ~ as.factor(x))
```

Again, we can plot the estimated density using the corresponding function `plot()`, see Figure 2.

Once we have estimated a density with covariates, we can test for equality of the corresponding densities. As we can see, the p-value indicates inequality.

```
> test.equal(test2)

        [,1]
2 vs. 1    0
```

2 pendensity

We contribute an `R-package pendensity` for density estimation based on the idea of penalized splines. Therefore, the main program contains several parameters, which can be varied by user. First, we speak about the possible global options of `pendensity()`.

2.1 global options of pendensity()

Most of the options are equal, independent if covariates are considered or not.

2.1.1 Density estimation without covariate

Density estimation without covariate(s) is easily done by using `pendensity()`. We need a formula to describe the desired results. For the following we define y as the interesting variable. If we want to estimate the density of y without any covariate, we type `pendensity(y~1)`. We can change several parameters for the estimate. Here we mention the important parameter:

no.base: We can change the used number of basis functions, by default this parameter is 'NULL'. Without any other changes, this results in a base with 41 B-spline bases. We can change this parameter, but we should not choose it too small, e.g. no.base=10 results in $2 * 10 + 1 = 21$ B-spline bases for B-splines of degree two.

max.iter: The maximal number of iterations should be selected too large or even too small. Depending on the selected penalty parameter λ, `pendensity()` needs more or less iterations. The default is 20.

λ_0: The default for the penalty parameter is 50000. Sometimes, one may change this value, resulting in fewer iterations.

q: The order of the used B-splines, default is $q = 3$

m: the used difference order of the penalty. The default is m=q, that is the order of the used B-splines.

with.border: We run several simulations and sometimes add additional bases at the end of the support of the observed value. This may improve the fit, in particular at the boundary. Usually we set `with.border=1` or `with.border=2`.

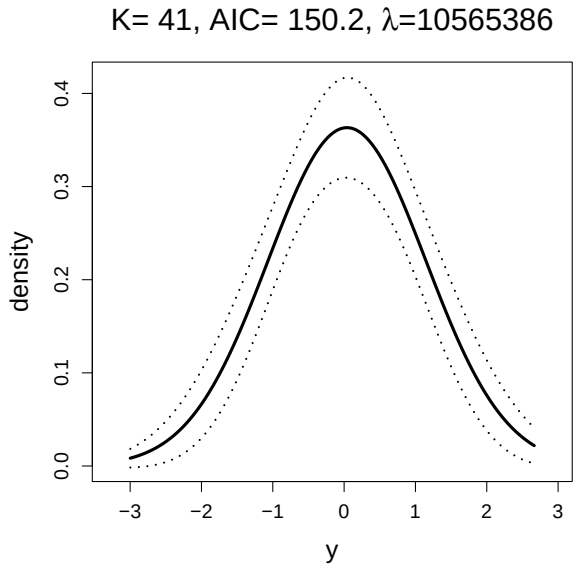

Figure 1: Plot of the estimated density.

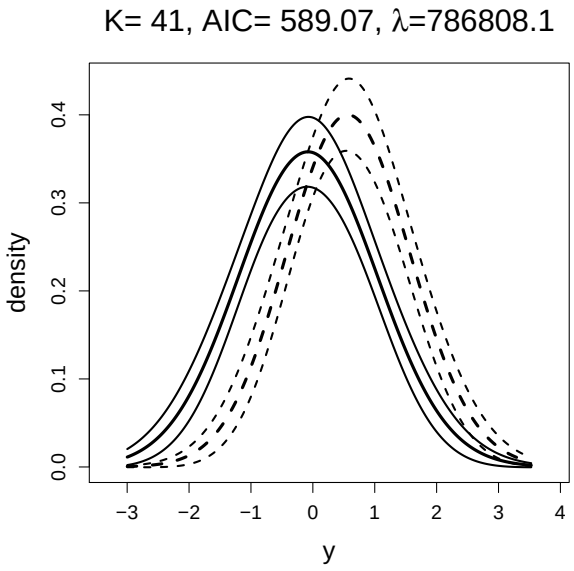

Figure 2: Plot of the estimated densities in Section 1.2.

2.1.2 Density estimation with covariate

If we want to estimate a density with covariate, only the formula in `pendensity()` changes. Keep in mind, only factorial covariates are used, indicating the corresponding groups. First we consider an example with one covariate x for the response y. So, we type `pendensity(y∼x)` (x has to be a factor, otherwise the call has to be `pendensity(y as.factor(x))`. Of course, we can also estimate densities with two or more covariates. Therefore we add the different factorial covariates with '+' in the call, e.g. `pendensity(y∼x+z)`.

2.2 plot.pendensity()

Having done an estimation without covariate, we can easily plot the density with `plot()`. Using this function of plot, we can determine several parameters. At first, we can choose between the usual plot and lattice. Secondly, we can get three different type of plots. Now, we highlight the main options.

latt: TRUE/FALSE indicates if lattice should be used.

plot.val: 1 indicates the normal density plot, 2 indicates the distribution function of the observation values and 3 indicates the analytic distribution function.

val: We can calculate the corresponding density at unobserved points. Therefore, we set a vector of y, at which the estimated density is calculated

confi: TRUE/FALSE, if the confidence intervals should be plotted or not

By default, the title of each plot contains the final number of knots 'K', the corresponding Akaike Information Criteria (AIC) and the optimal λ. We can change this with the known option of the generic function `plot()`, using `main=...` . Additionally, we can add text below the plot, using `sub=...` . Moreover, we can use `xlab` and `ylab`.

2.3 test.equal()

For each estimated pendensity object dependent on some covariate we can test for equality of the densities. Therefore, we can easily use the `test.equal()` function and get an output with the corresponding p-value(s). See the corresponding example in Section 1.2.

If one estimates a density with covariate indicating more than two groups, we compare the densities pairwise. Therefore, we present the following example

```
> x <- c(rep(0, 50), rep(1, 100), rep(2, 100))
> y <- rnorm(250, x, 1)
> x <- as.factor(x)
> test3 <- pendensity(y ~ x)

> test.equal(test3)

        [,1]
3 vs. 2    0
3 vs. 1    0
2 vs. 1    0
```

2.4 Calculate density values

In many applications one is interested to calculate the density in unobserved values. To do this, we can use the `plot()` function using the argument `val`. We consider the example from Section 1.1 and Section 1.2. If we want to get the density value in a given set of points, we type just `plot(obj,val=...)`. This function is only available for densities `plot.val=1`, not for the distribution. We get the values for each group separately. Additionally, the corresponding values of the standard deviation are listed.

For the example in Section 1.1 we get

```
> points <- c(1, -0.5, 0, 0.5, 1)
> plot(test, val = points)

$y
[1]  1.0 -0.5  0.0  0.5  1.0

$fy
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.2494086 0.3225361 0.3629941 0.3331572 0.2494086

$sd.down.y.val
[1] 0.2034252 0.2754246 0.3094312 0.2828216 0.2034252
```

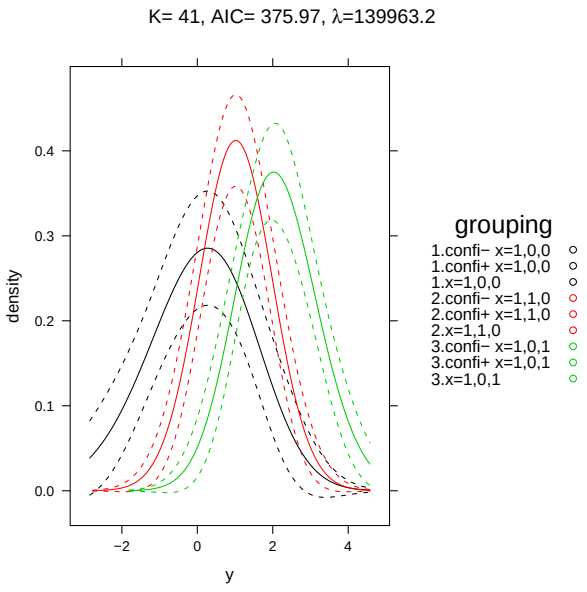

Figure 3: Plot of the estimated densities of example in Section 2.3.

```
$sd.up.y.val
[1] 0.2953921 0.3696476 0.4165570 0.3834927 0.2953921
```

For the example in Section 1.2 we get

```
> points <- c(1, -0.5, 0, 0.5, 1)
> plot(test2, val = points)

$y
[1]  1.0 -0.5  0.0  0.5  1.0

$fy
$fy$baseline
[1] 0.2254301 0.3336584 0.3571322 0.3134388 0.2254301

$fy$`as.factor(x)=1`
[1] 0.3640085 0.2244045 0.3404441 0.3992445 0.3640085


$sd.down.y.val
$sd.down.y.val$baseline
[1] 0.1939494 0.2978627 0.3173781 0.2778318 0.1939494

$sd.down.y.val$`as.factor(x)=1`
[1] 0.3234061 0.1883559 0.3063653 0.3588498 0.3234061


$sd.up.y.val
$sd.up.y.val$baseline
[1] 0.2569108 0.3694542 0.3968862 0.3490457 0.2569108

$sd.up.y.val$`as.factor(x)=1`
[1] 0.4046109 0.2604532 0.3745229 0.4396391 0.4046109
```

3 Stock Examples

For a more detailed overview of our package `pendensity`, we consider the attached data of the daily final values of German stocks Deutsche Bank AG und Lufthansa AG from 2000 to 2008. We give some density examples and corresponding comments.

3.1 Allianz 2006

We are interested to look at the density of the German stock *A*llianz in 2006. Therefore, we build differences of first order and estimate the density.

```
> data(Allianz)
> form <- "%d.%m.%y"
> time.Allianz <- strptime(Allianz[, 1], form)
> data.Allianz <- Allianz[which(time.Allianz$year == 106), 2]
> d.Allianz <- diff(data.Allianz)
> density.Allianz <- pendensity(d.Allianz ~ 1)
```

This density is estimated with the default settings of `pendensity()`. Now, we like to highlight the different options to plot densities. We can also use `lattice` to create plots, see Figure 4.

3.2 Allianz 2006 and 2007

We are interested to look at the density of the german stock *A*llianz in 2006 and 2007. Therefore, we build differences of first order of the stock values and estimate the density depending on covariate. Again, we can plot this estimate. We show the result in Figure 5, also for the lattice output.

```
> data.Allianz <- Allianz[which(time.Allianz$year == 106 | time.Allianz$year ==
+     107), 2]
> d.Allianz <- diff(data.Allianz)
> density.Allianz2 <- pendensity(d.Allianz ~ as.factor(time.Allianz$year))
```

Checking for equality of the estimated densities is done with

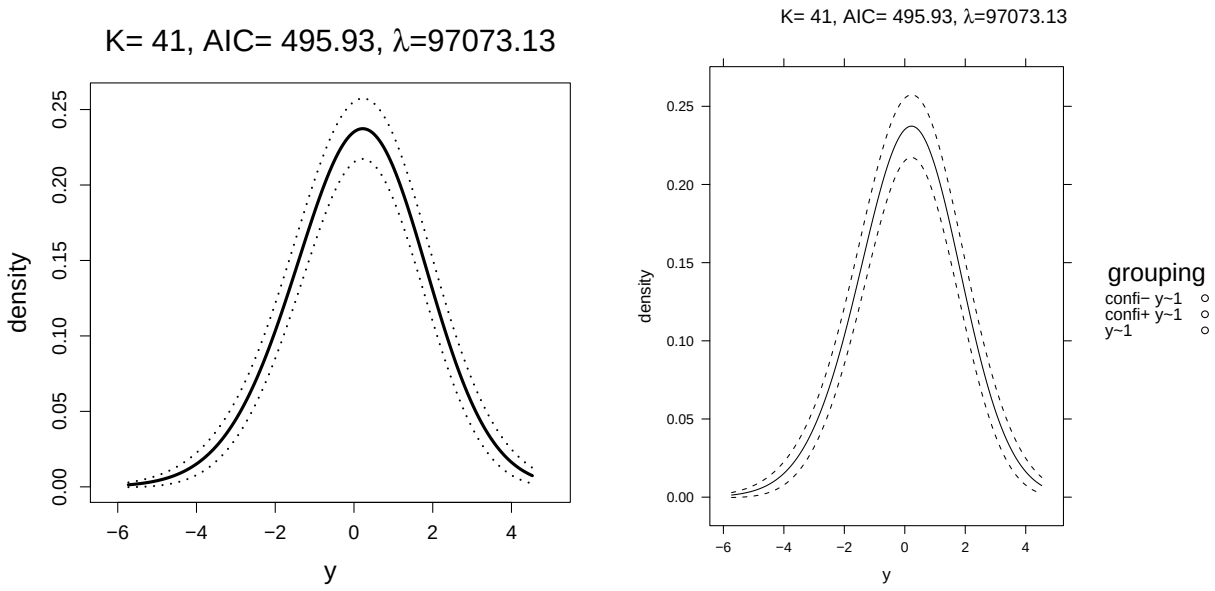

Figure 4: Plot of the estimated density of Allianz 2006.

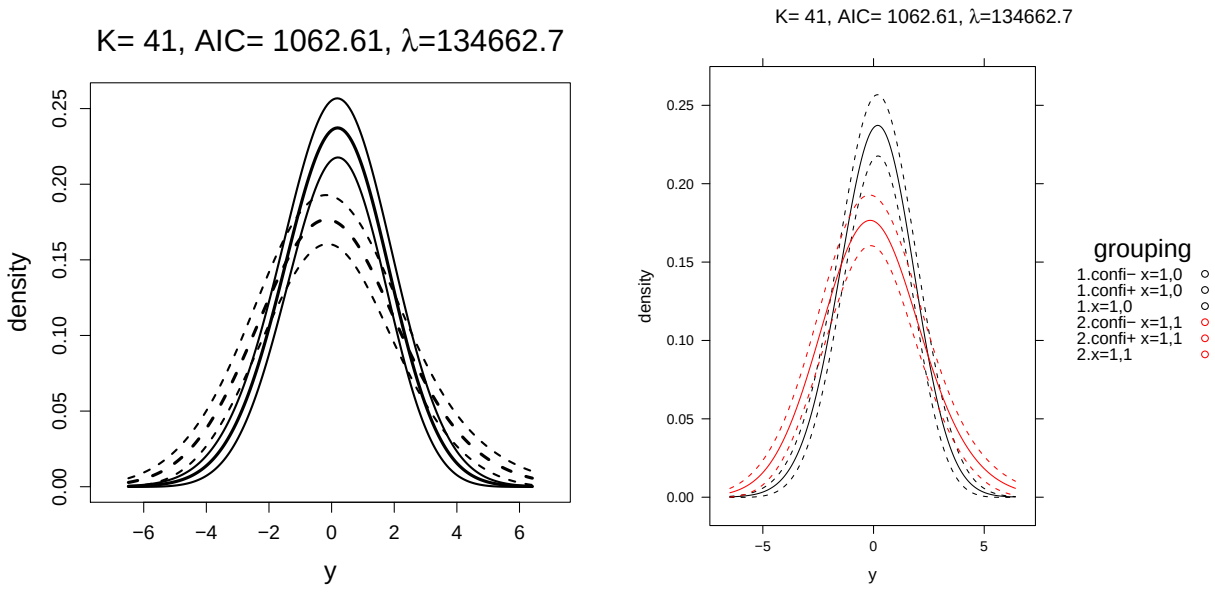

Figure 5: Plot of the estimated densities of Allianz in 2006 and 2007.

```
> test.equal(density.Allianz2)

         [,1]
2 vs. 1 0.012
```

4 Number of knots

As mentioned in many famous papers, the number of knots has not to be too small for penalized spline smoothing. A good rule is to select 20 up to 40 knots.If one chooses more knots, the results do not change, due to the used penalization concept. Here, we show a short example for a sample y from standard normal distribution of size $N = 400$.

```
> set.seed(27)
> y <- rnorm(400)
> density1 <- pendensity(y ~ 1, no.base = 10)
> density2 <- pendensity(y ~ 1, no.base = 15)
> density3 <- pendensity(y ~ 1, no.base = 20)
> density4 <- pendensity(y ~ 1, no.base = 25)
```

For comparison, we plot these densities in Figure 6. We see easily, that there is no change in between these figures, while the penalty increases with the size of the base. The AIC values are more or less equal, the very small differences are numerical noises. By default, `pendensity()` estimates the densities with 41 B-spline bases.

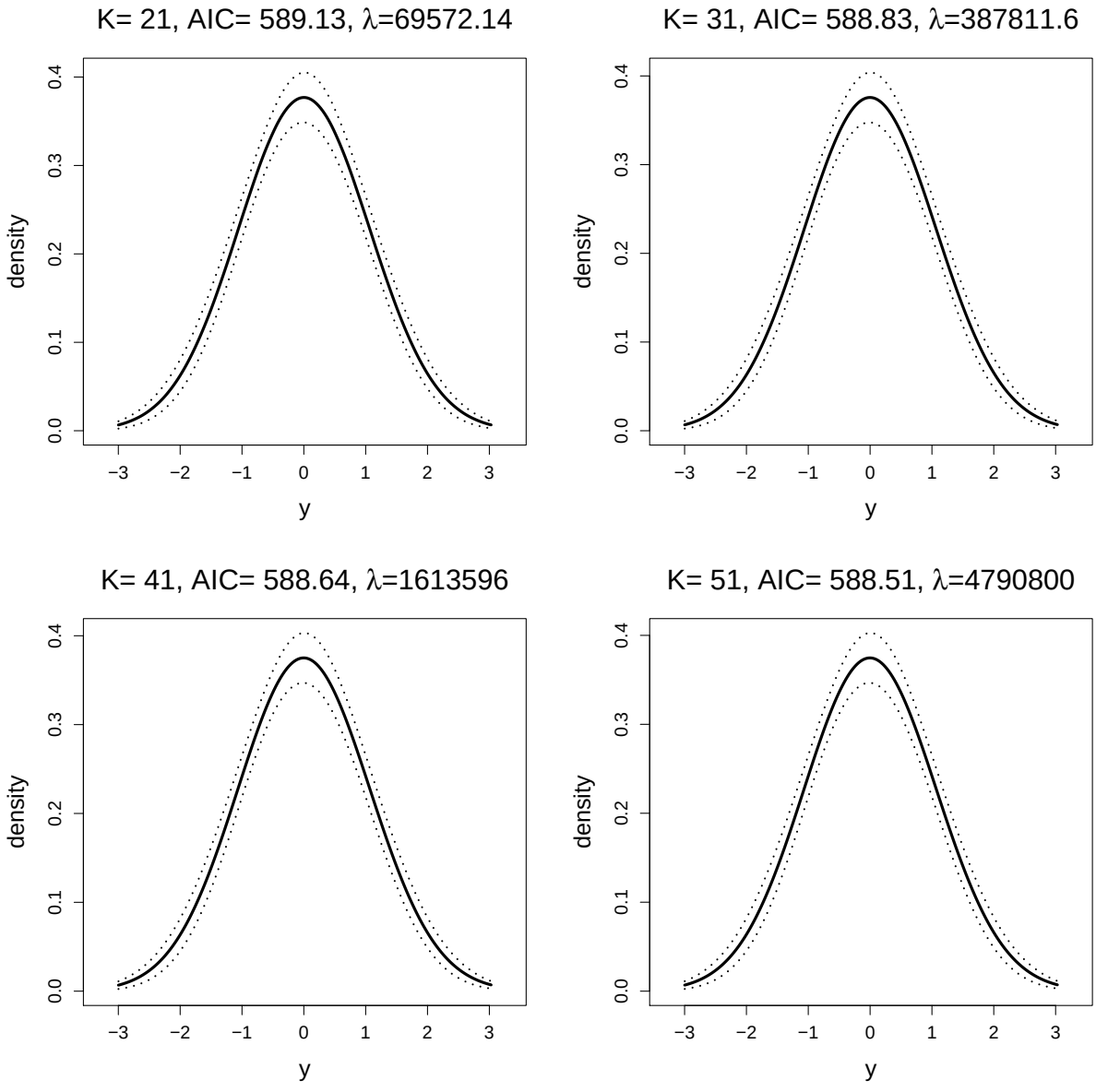

Figure 6: Plot of a sample y for different number of knots.