# Package ‘svcm’

April 19, 2007

**Type** Package

**Title** 2d and 3d Space-Varying Coefficient Models

**Version** 0.1.1

**Date** 2007-04-19

**Author** Susanne Heim, with support from Paul Eilers, Thomas Kneib, and Michael Kobl

**Maintainer** Susanne Heim <susanneheim@gmx.net>

**Description** 2d and 3d space-varying coefficient models are fitted to regular grid data using either a full B-spline tensor product approach or a sequential approximation. The latter one is computationally more efficient. Resolution increment is enabled.

**License** GPL version 2 or newer

**Depends** R (>= 2.4.0), Matrix, splines

**LazyLoad** yes

**LazyData** yes

**URL** http://www.statistik.lmu.de/~heim

## R topics documented:

`brain2d` *Two-dimensional Diffusion Weighted Dataset*

## Description

The data set consists of six transformed diffusion weighted images (DWI) showing a representative axial slice of the human brain. The stored values can directly be passed to estimate the diffusion tensor elements (regression coefficients) using a transform of the applied gradients as regressors.

## Usage

```
data(brain2d)
```

## Format

The first two dimensions provide the transformed signal intensities of one brain slice sized $90 \times 75$ voxels. The third dimension encodes for the direction of the six applied diffusion weighting gradients.

## Details

The present DTI data set was acquired at 1.5 T (Signa Echospeed; GE Medical Systems) using a spin-echo echo-planar sequence with TR/TE = 4200ms/120ms and diffusion gradients in a six noncollinear directions with a b-value of 880 s/mm^2. One axial slice was selected from a volume of six DWI (b = 880 s/mm^2) and one reference image (b = 0 s/mm^2). In-plane resolution amounts to $1.875 \times 1.875$ mm^2.

The transformation of the raw signal intensities,

$$y = -\frac{1}{b}\log\left(\frac{S_i}{S_0}\right), i = 1, \ldots, 6$$

is derived from the Stejskal-Tanner equation and is proposed by Papadakis et al.

## Source

Diffusion Tensor Imaging was performed at the Max-Planck-Institute of Psychiatry, Munich, Germany.

## References

Basser P. J. and Jones D. K. (2002) Diffusion-tensor MRI: Theory, experimental design and data analysis – a technical review. *NMR in Biomedicine* **15**, 456-467.

Mori S. and Barker P. B. (1999) Diffusion magnetic resonance imaging: Its principle and applications. *The Anatomical Record* **257**, 102-109.

Papadakis N. G., Xing D., Huang C. L.-H., Hall L. and Carpenter T. A. (1999). A comparative study of acquisition schemes for diffusion tensor imaging using MRI. Journal of Magnetic Resonance 137, 67-82.

Stejskal E. O. and Tanner J. E. (1965) Spin diffusion measurements: Spin echoes in the presence of time-dependent field gradient. *The Journal of Chemical Physics* **42**, 288-292.

### Examples

```
data(brain2d)
dim(brain2d)
old.par <- par(no.readonly=TRUE)
par(pin=c(6, 1.2))
image(t(matrix(brain2d, dim(brain2d)[1], dim(brain2d)[2]*6)), axes=FALSE,
      col=grey.colors(256))
title("Six Diffusion Weighted Images")
par(old.par)
```

---

## `brain3d` *Three-dimensional Diffusion Weighted Dataset*

---

### Description

To keep the computational effort low a volume of $15 \times 30 \times 6$ voxels was chosen from the original whole brain volume. The extract depicts the posterior part of the lateral ventricles and the corpus callosum so both areas with low and high signal intensities are contained. The six transformed diffusion weighted images can directly be passed to estimate the diffusion tensor elements (regression coefficients) using a transform of the applied gradients as regressors.

### Usage

```
data(brain3d)
```

### Format

The first three dimensions of the data array contain the number of voxels in x-, y- and z-direction. The fourth dimension encodes for the direction of the six applied diffusion weighting gradients.

### Details

The present DTI data set was acquired at 1.5 T (Signa Echospeed; GE Medical Systems) using a spin-echo echo-planar sequence with TR/TE = 4200ms/120ms and diffusion gradients in a six noncollinear directions with a b-value of 880 s/mm^2. The extracted volume originates from six DWI (b = 880 s/mm^2) and one reference image (b = 0 s/mm^2). In-plane resolution amounts to $1.875 \times 1.875$ mm^2, slice thickness is 4.0 mm.

The transformation of the raw signal intensities,

$$y = -\frac{1}{b} \log \left( \frac{S_i}{S_0} \right), i = 1, \ldots, 6$$

is derived from the Stejskal-Tanner equation and is proposed by Papadakis et al.

### Source

Diffusion Tensor Imaging was performed at the Max-Planck-Institute of Psychiatry, Munich, Germany.

### References


Basser P. J. and Jones D. K. (2002) Diffusion-tensor MRI: Theory, experimental design and data analysis – a technical review. *NMR in Biomedicine* **15**, 456-467.

Mori S. and Barker P. B. (1999) Diffusion magnetic resonance imaging: Its principle and applications. *The Anatomical Record* **257**, 102-109.

Papadakis N. G., Xing D., Huang C. L.-H., Hall L. and Carpenter T. A. (1999). A comparative study of acquisition schemes for diffusion tensor imaging using MRI. Journal of Magnetic Resonance 137, 67-82.

Stejskal E. O. and Tanner J. E. (1965) Spin diffusion measurements: Spin echoes in the presence of time-dependent field gradient. *The Journal of Chemical Physics* **42**, 288-292.


### Examples

```
data(brain3d)
dim(brain3d)
old.par <- par(no.readonly = TRUE)
par(pin=c(1.1, 3.4), mfrow=c(1, 6))
for (i in 1:dim(brain3d)[4])
  image(matrix(aperm(brain3d[,,,i], c(2,1,3)), nrow=dim(brain3d)[2]),
        axes=FALSE, col=grey.colors(256), main=paste("DWI", i))
title("6 DWIs of a (15 x 30 x 6) human brain extract in axial view",
      outer=TRUE, line=-10)
par(old.par)
```

---

## cleversearch *Optimization over a parameter grid*

---

### Description

This function allows greedy/full grid search in any dimension.

### Usage

```
cleversearch(fn, lower, upper, ngrid, startvalue, logscale = TRUE,
             clever = TRUE, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `fn` | a function to be minimized (or maximized), with the only argument being the parameter over which minimization is to take place. The return value must be scalar. |
| `lower` | numeric vector containing the lower bounds on the parameter grid |
| `upper` | numeric vector containing the upper bounds on the parameter grid |
| `ngrid` | integer number determining the grid length in every dimension |
| `startvalue` | optional initial value for the parameter to be optimized over |
| `logscale` | logical, whether to construct the grid on logarithmic scale |
| `clever` | logical, whether to perform a greedy grid search with lookup-table or a full grid evaluation. The latter is only available up to 3d. |
| `verbose` | logical. Should the search process be monitored? |

### Details

Unless `startvalue` is specified, the search starts at the lower bound of the 1d parameter space or at the middle of the 2d/3d grid.

### Value

A list with components

| | |
|---|---|
| `par` | optimal parameter value that was found. |
| `value` | `fn` value corresponding to `par`. |
| `counts` | number of calls to 'fn'. |

### See Also

`optim`

### Examples

```
simplefun <- function(vec) { return(sum(vec^2)) }
opt <- cleversearch(simplefun, c(-1, -1), c(1, 1), 51, logscale=FALSE)
```

---

## `resolution` *Re-scaling resolution of SVCM predictors and effects*

---

### Description

This routine serves post-hoc adjustment of the resolution of a space-varying coefficient model estimated by `svcm`.

### Usage

```
resolution(X, svcmlist, fac)
```

### Arguments

| | |
|---|---|
| `X` | (r x p)-array of covariates |
| `svcmlist` | return list of function `svcm` |
| `fac` | 2d or 3d vector of scaling factors |

### Details

The basis function approach underlying `svcm` allows to rescale the original resolution by evaluating the basis functions at additional points. Assuming that the voxel center is most representative for the whole voxel, `fac`-times resolution of 1d data with $n$ voxels sized $vsize$ bases on coordinates

$$\left(i - \frac{1}{2}\right) \cdot \frac{vsize}{fac}, \quad i = 1, \ldots, n \cdot fac.$$

See also doc file resolution_scheme.pdf.
The formula is applied into x-, y- and z-direction and results in a refined equidistant 2d resp. 3d grid. It also means that, in general, the resized arrays of predictors and effects do no longer contain the values at the former coordinates.

Note that memory requirements can be enormous depending on object size and the intended resolution.

## Value

A list with components

| | |
|---|---|
| `fitted` | fitted values at `fac`-times rescaled resolution. |
| `effects` | estimated effects at `fac`-times rescaled resolution. |

## Examples

```
##DTI data; regressors are given by the diffusion weigthing gradients
data(brain2d)
X <- matrix(c(0.5, 0.5,   0,   0, 0.5, 0.5,
                0,   0, 0.5, 0.5, 0.5, 0.5,
              0.5, 0.5, 0.5, 0.5,   0,   0,
                0,   0,   0,   0,   1,  -1,
                1,  -1,   0,   0,   0,   0,
                0,   0,   1,  -1,   0,   0), nrow = 6)
M <- svcm(brain2d, X, knots=c(60, 50), deg=c(1, 1), vsize=c(1.875,
          1.875), search=TRUE, type="SEQ", lambda.init=rep(0.1, 2),
          lower=rep(-5, 2), upper=rep(0, 2), ngrid=10)
M2 <- resolution(X, M, fac=c(2, 2))

##show data extract at original and double resolution
extract <- list(M$fit[21:40, 21:60, 1],
                M2$fit[(21*2):(40*2), (21*2):(60*2), 1],
                M$eff[21:40, 21:60, 1],
                M2$eff[(21*2):(40*2), (21*2):(60*2), 1])
zlim1 <- range(extract[[1]], extract[[2]])
zlim2 <- range(extract[[3]], extract[[4]])
old.par <- par(no.readonly = TRUE)
par(pin=c(3*1, 3*0.67), mfrow=c(2, 2))
image(t(extract[[1]]), axes=FALSE, zlim=zlim1, col=grey.colors(256))
title("Fitted Values")
image(t(extract[[2]]), axes=FALSE, zlim=zlim1, col=grey.colors(256))
title("Fitted Values at Double Resolution")
image(t(extract[[3]]), axes=FALSE, zlim=zlim2, col=grey.colors(256))
title("Estimated VC Surface (1st DT element)")
image(t(extract[[4]]), axes=FALSE, zlim=zlim2, col=grey.colors(256))
title("VC Surface at Double Resolution")
par(old.par)
```

---

`svcm-package` *2d and 3d Space-Varying Coefficient Models*

---

## Description

2d and 3d space-varying coefficient models are fitted to regular grid data using either a full B-spline tensor product approach or a sequential approximation. The latter one is computationally more efficient. Resolution increment is enabled.

## Details

| | |
|---|---|
| Package: | svcm |
| Type: | Package |
| Version: | 0.1.1 |
| Date: | 2007-04-19 |
| License: | GPL version 2 or newer |
| Depends: | R (>= 2.4.0), Matrix, splines |
| LazyLoad: | yes |
| LazyData: | yes |
| URL: | http://www.statistik.lmu.de/ heim |

Originally, VCMs have been suggested by Hastie and Tibshirani (1993) for regressions with coefficients varying smoothly over a one-dimensional continuous variable such as time-varying effects. This package provides extensions to two- and three-dimensional space-varying coefficients surfaces for regularly gridded data without missings. Such a SVCM takes into account spatial correlation. The use of spline-basis functions serves to model the spatial coefficient field. As a consequence, estimates are accessible at any arbitrary position, not only on the original grid of voxels. Resolution can be easily increased and moreover penalized for possible initial anisotropy of the voxel size.

Two techniques are implemented. The multidimensional smoothing approach takes advantage of the sparsity of the spatial arrays involved. The second sequential one basically adapts the 'new smoothing spline' in Dierckx (1982), thus reducing the 3d (or higher-dimensional) problem to a sequence of one-dimensional smoothers.

The 2d and 3d examples have been chosen from the field of human brain imaging.

## Author(s)

Susanne Heim, with support from Paul Eilers, Thomas Kneib, and Michael Kobl

Maintainer: Susanne Heim <susanneheim@gmx.net>

## References

Dierckx P. (1982) A fast algorithm for smoothing data on a rectangular grid while using spline functions. *SIAM Journal on Numerical Analysis* **19(6)**, 1286-1304.

Hastie T. and Tibshirani R. (1993) Varying-Coefficient Models (with discussion). *Journal of the Royal Statistical Society* B **55**, 757-796.

Heim S., Fahrmeir L., Eilers P. H. C. and Marx B. D. (2006) Space-Varying Coefficient Models for Brain Imaging. *Ludwig-Maximilians-University, SFB 386*, Discussion Paper **455**.

---

`svcm` *Fitting space-varying coefficient models*

---

## Description

'svcm' is used to fit a 2d or 3d space-varying coefficient model or to merely smooth the input data using penalized B-splines. The smoothing works either sequentially or multidimensionally involving tensor products. So far, only space-invariant regressors are allowed. Data must be on a regular grid without missings.

## Usage

```
svcm(Y, X, vsize = c(1, 1, 1), knots = c(10, 10, 10),
     deg = c(1, 1, 1), opd = c(1, 1, 1), search = TRUE,
     lambda.init = rep(0.001, 3), method = "grid", type = "SEQ", ...)
```

## Arguments

| | |
|---|---|
| `Y` | array of observational data. Last dimension must correspond to the number of rows of `X`. |
| `X` | (r x p)-design matrix |
| `vsize` | numeric vector of the voxel size |
| `knots` | vector of the numbers of inner knots in x-, y- (and z-) direction |
| `deg` | vector of degrees of the basis functions in x-, y- (and z-) direction |
| `opd` | vector of the order of the difference penalties in x-, y- (and z-) direction |
| `search` | logical. If TRUE, the smoothing parameter will be optimized using `method` and GCV. If FALSE, `lambda.init` specifies the fixed smoothing parameter. |
| `lambda.init` | compulsory; initial value of global or dimension-specific smoothing parameter. See **Details**. |
| `method` | optimization method to be used. See **Details**. |
| `type` | character. `"SEQ"` (sequential) or `"TP"` (tensor product). |
| `...` | parameters to be passed to the optimization algorithm specified by `method`. |

## Details

The purpose of `lambda.init` is three-fold: First, the length determines the use of either global or dimension-specific penalties. Second, it serves as fixed smoothing parameter if search is deactivated. Third, it is used as initial value from the `optim` algorithm which runs in case of a multidimensional tuning parameter when no grid search is desired.

Unless method equals `"grid"`, `optimize` is called in the case of a global tuning parameter requiring a specified `interval` to be passed. While `optimize` does not take a starting value explicitly, a `startvalue` can be passed to `cleversearch`, e.g. `startvalue = lambda.init`.

In the case of a dimension-specific tuning parameter, method `"grid"` evokes a full or greedy grid search (see `cleversearch`). Amongst others, simplex method `"Nelder-Mead"` or quasi-Newton `"L-BFGS-B"` with positivity constraint for the smoothing parameter are conceivable, too. For further specification see `optim`.

For simple smoothing of `Y` set `X = matrix(1, 1, 1)` and ascertain that the last dimension of `Y` matches `dim(X)[1]`.

## Value

A list with components:

| | |
|---|---|
| `fitted` | fitted values as array of the same dimension as `Y` |
| `effects` | effects of dimension (n.x, n.y, p) resp. (n.x, n.y, n.z, p). |
| `coeff` | coefficients (amplitudes of the basis functions) of dimension (p, r.x, r.y) resp. (p, r.x, r.y, r.z) with r.x number of basis functions in x-direction. |
| `knots` | see `knots`. |
| `deg` | see `deg`. |

`opd` see `opd`.

`vsize` see `vsize`.

`type` character describing the SVCM variant used. See `type`.

`call` the matched call.

`opt` a list with components depending on `search`, i.e. on whether optimization was performed or not:

**time** calculation/optimization time

**par** initial value `lambda.init` or the best parameter found.

**value** GCV value corresponding to `par`.

**GCVtab** matrix of the search process with values of lambda and corresponding GCV value.

**...** further values returned by optim(), optimize()

## Warnings

This model assumes the regressors to be space-invariant. Data must be on a regular grid without missings.

## Background

In the general case of 2d mesh data, Dierckx (1982, 1993) demonstrates the equivalence of successive univariate smoothing with smoothing based on a full bivariate B-spline matrix. However, the equivalence does no longer hold if penalties are introduced. Dierckx proposes the so-called 'new smoothing spline' as approximation to the multidimensional penalized smoothing (`type = "TP"`). While Dierckx determines the penalty structure through the spline degree and the equidistance between adjacent knots, the present implementation (`type = "SEQ"`) uses penalties of simple differences.

The calculation of GCV involves an inversion which is achieved using the recursion formula for band matrices in Hutchinson/de Hoog (1985). My collegue Thomas Kneib not only recommended this paper but also provided us with the basic.

## Note

The observations in `Y` are assigned to the center of the respective grid unit sized `vsize`. Hence the basis functions are evaluated at these coordinates.

## References

Dierckx P. (1982) A fast algorithm for smoothing data on a rectangular grid while using spline functions. *SIAM Journal on Numerical Analysis* **19(6)**, 1286-1304.

Dierckx P. (1993) *Curve and surface fitting with splines.* Oxford: Monographs on Numerical Analysis, Oxford University Press.

Heim S., Fahrmeir L., Eilers P. H. C. and Marx B. D. (2006) Space-Varying Coefficient Models for Brain Imaging. *Ludwig-Maximilians-University, SFB 386*, Discussion Paper **455**.

Hutchinson M. F. and de Hoog F. R. (1985) Smoothing noisy data with spline functions. *Journal of Numerical Mathematics* **47**, 99-106.

### See Also

`optimize` for one-dimensional minimization,
`optim` here explicitly method "L-BGFS-B",
`cleversearch` for clever or full grid search.

### Examples

```
## 2d DT-MRI data
data(brain2d)
X <- matrix(c(0.5, 0.5,   0,   0, 0.5, 0.5,
                0,   0, 0.5, 0.5, 0.5, 0.5,
              0.5, 0.5, 0.5, 0.5,   0,   0,
                0,   0,   0,   0,   1,  -1,
                1,  -1,   0,   0,   0,   0,
                0,   0,   1,  -1,   0,   0), nrow = 6)
##2d grid search for lambda; 60*50*6=18000 parameters (amplitudes) in total
M <- svcm(brain2d, X, knots=c(60, 50), deg=c(1, 1), vsize=c(1.875, 1.875),
          type="SEQ", lambda.init=rep(0.1, 2), search=TRUE,
          method="grid", lower=rep(-5, 2), upper=rep(0, 2), ngrid=10)
str(M$opt)

## show results
zlim <- range(brain2d, M$fit)
old.par <- par(no.readonly=TRUE)
par(pin=c(6, 1.2), mfrow=c(3, 1))
image(t(matrix(brain2d, nrow=dim(brain2d)[1])), axes=FALSE, zlim=zlim,
      col=grey.colors(256))
title("Observations: Six Diffusion Weighted Images")
image(t(matrix(M$fitted, nrow=dim(M$fit)[1])), axes=FALSE, zlim=zlim,
      col=grey.colors(256))
title("Fitted Values")
image(t(matrix(M$effects, nrow=dim(M$eff)[1])), axes=FALSE,
      col=grey.colors(256))
title("Estimated Coefficient Surfaces: Six Elements of the Diffusion Tensor")
par(old.par)

## 3d DT-MRI data; same regressors as in 2d; fixed global smoothing parameter
data(brain3d)
M3d <- svcm(brain3d, X, knots=c(5, 10, 5), deg=c(1, 1, 1), search=FALSE,
            vsize=c(1.875, 1.875, 4.0), type="TP", lambda.init=1)

## visualize results
zlim <- range(brain3d[,,,1], M3d$fit[,,,1])
old.par <- par(no.readonly = TRUE)
par(pin=c(1.8, 5), mfrow=c(1, 3))
image(matrix(aperm(brain3d[,,,1], c(2,1,3)), nrow=dim(brain3d)[2]),
      axes=FALSE, zlim=zlim, col=grey.colors(256))
title("(a) Obs: 1st DWI")
image(matrix(aperm(M3d$fit[,,,1], c(2,1,3)), nrow=dim(brain3d)[2]),
      axes=FALSE, zlim=zlim, col=grey.colors(256))
title("(b) Fits of 1st DWI")
image(matrix(aperm(M3d$eff[,,,1], c(2,1,3)), nrow=dim(brain3d)[2]),
             axes=FALSE, col=grey.colors(256))
title("(c) Effects: 1st DT element")
title("Six axial slices of the 1st DWI-transform (a) and its fit (b);
      \n\n(c) corresponds to the first diffusion tensor component.",
```

```
         outer=TRUE, line=-5)
par(old.par)
```

# Index