fdaPDE 1.0

A. Colli, L. Colombo

January 21, 2020

Tutor: Prof. L. M. Sangalli, E. Arnone

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal Statistical Society Ser. B. Lila ERRORS!
- *Blood flow velocity field estimation via spatial regression with PDE penalization*: Azzimonti et al., 2015, JASA. NA
- *Spatial regression models over two-dimensional manifolds*: Ettinger et al., 2016, Biometrika. Cosmo-Beraha ERRORS!
- *Smooth Manifold-Functional Principal Component Analysis*: Lila et al., 2016, Annals of Applied Statistics.
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018. Negri ERRORS!

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal Statistical Society Ser. B. Lila ERRORS!
- *Blood flow velocity field estimation via spatial regression with PDE penalization*: Azzimonti et al., 2015, JASA. NA
- *Spatial regression models over two-dimensional manifolds*: Ettinger et al., 2016, Biometrika. Cosmo-Beraha ERRORS!
- *Smooth Manifold-Functional Principal Component Analysis*: Lila et al., 2016, Annals of Applied Statistics.
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018. Negri ERRORS!

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal Statistical Society Ser. B. Lila ERRORS!
- *Blood flow velocity field estimation via spatial regression with PDE penalization*: Azzimonti et al., 2015, JASA. NA
- *Spatial regression models over two-dimensional manifolds*: Ettinger et al., 2016, Biometrika. Cosmo-Beraha ERRORS!
- *Smooth Manifold-Functional Principal Component Analysis*: Lila et al., 2016, Annals of Applied Statistics.
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018. Negri ERRORS!

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal Statistical Society Ser. B. Lila ERRORS!
- *Blood flow velocity field estimation via spatial regression with PDE penalization*: Azzimonti et al., 2015, JASA. NA
- *Spatial regression models over two-dimensional manifolds*: Ettinger et al., 2016, Biometrika. Cosmo-Beraha ERRORS!
- *Smooth Manifold-Functional Principal Component Analysis*: Lila et al., 2016, Annals of Applied Statistics.
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018.
  Negri ERRORS!

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal Statistical Society Ser. B. fdaPDE 1.0
- *Blood flow velocity field estimation via spatial regression with PDE penalization*: Azzimonti et al., 2015, JASA. fdaPDE 1.0
- *Spatial regression models over two-dimensional manifolds*: Ettinger et al., 2016, Biometrika. fdaPDE 1.0
- *Smooth Manifold-Functional Principal Component Analysis*: Lila et al., 2016, Annals of Applied Statistics. fdaPDE 1.0
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018. fdaPDE 1.0

Statistical models with PDEs

- *Spatial Spline Regression Models*: Sangalli et al., 2013, Journal of the Royal S
- *Bl* *PDE*
- *Sm* Lila et al., 2016, Anna
- *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*: L. Negri, Thesis 2018. fdaPDE 1.0

Correct

New Functionalities

Stable

Regression model

- $\mathbf{p}_1, \ldots, \mathbf{p}_n$: n data locations over $\Omega \subset \mathbb{R}^2$,
- $z_i \in \mathbb{R}$ variable observed at $\mathbf{p}_i$
- $\mathbf{w}_i \in \mathbb{R}^q$ covariates associated with $z_i \in \mathbb{R}$ at $\mathbf{p}_i$
- $\epsilon_1, \ldots \epsilon_n$ random errors

$$\boxed{z_i = \mathbf{w}_i^t \boldsymbol{\beta} + f(\mathbf{p}_i) + \epsilon_i, \qquad i = 1, \ldots, n}$$

Estimate $\boldsymbol{\beta}$ and f minimizing

$$\sum_{i=1}^{n} \left(z_i - \mathbf{w}_i^t \boldsymbol{\beta} - f(\mathbf{p}_i)\right)^2 + \lambda \int_\Omega \left(Lf - u\right)^2 d\mathbf{p}$$

$K = \{K_{ij}\} \in \mathbb{R}^{2\times 2}$ spd, $\mathbf{b} = \{b_j\} \in \mathbb{R}^2$ transport, $c \in \mathbb{R}^+$ reaction.

$$Lf = -\mathrm{div}(K \nabla f) + \mathbf{b} \cdot \nabla f + cf.$$

Finite Element discretization

Estimate β and f:

$$\hat{\beta} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T(\mathbf{z} - \hat{\mathbf{f}}_n) \qquad \hat{f} = \hat{\mathbf{f}}^t\psi$$

The following holds:

$$\begin{bmatrix} -\mathbf{\Psi}^T\mathbf{Q}\mathbf{\Psi} & \lambda\mathbf{R}_1^T \\ \lambda\mathbf{R}_1 & \lambda\mathbf{R}_0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} -\mathbf{\Psi}^T\mathbf{Q}\,\mathbf{z} \\ \mathbf{u} \end{bmatrix}.$$

where

$$\mathbf{\Psi} = \begin{bmatrix} \psi_1(\mathbf{p}_1) & \dots & \psi_{N_\mathcal{T}}(\mathbf{p}_1) \\ \vdots & \vdots & \vdots \\ \psi_1(\mathbf{p}_n) & \dots & \psi_{N_\mathcal{T}}(\mathbf{p}_n) \end{bmatrix}$$

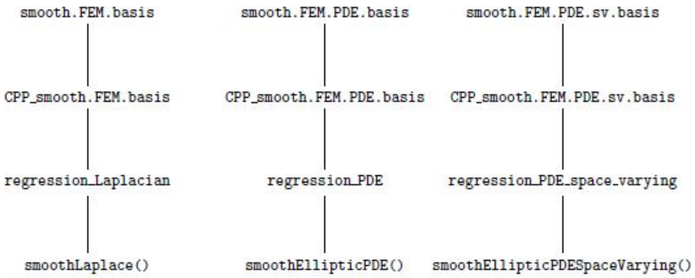
$$\mathbf{R}_0 = \int_{\Omega_\mathcal{T}} \boldsymbol{\psi}\boldsymbol{\psi}^T$$
$$\mathbf{R}_1 = \int_{\Omega_\mathcal{T}} \left(\nabla\boldsymbol{\psi}^T\mathbf{K}\nabla\boldsymbol{\psi} + \nabla\boldsymbol{\psi}^T\mathbf{b}\boldsymbol{\psi}^t + c\,\boldsymbol{\psi}\boldsymbol{\psi}^T\right)$$

$$\mathbf{Q} = \mathbf{I} - \mathbf{H} \qquad \mathbf{H} = \mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$$

History of the Code

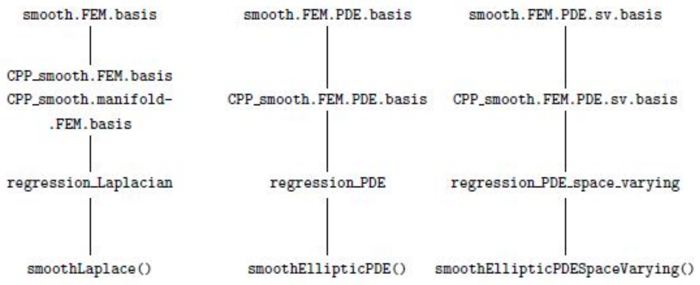
- Release Lila 2014: 2D smoothing Code
  - Bambini-Giussani 2017: Woodbury, GCV
  - Beraha-Cosmo 2017: 2.5D smoothing Code
  - Lila Github 2017: `C++` unification Code
  - Negri 2018: 3D smoothing, FPCA Code
- Release Colli-Colombo 2019: Areal smoothing Code

CRAN - 2014



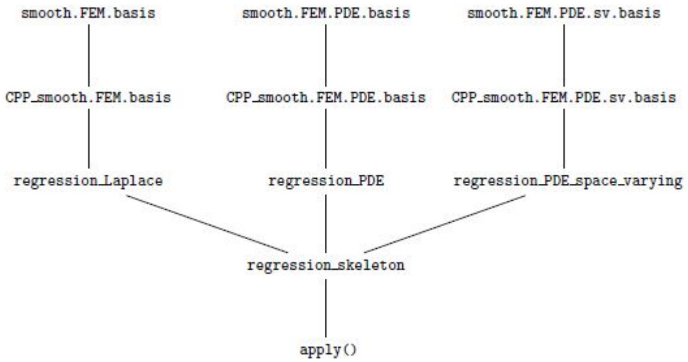
Go back

Cosmo-Beraha - 2017



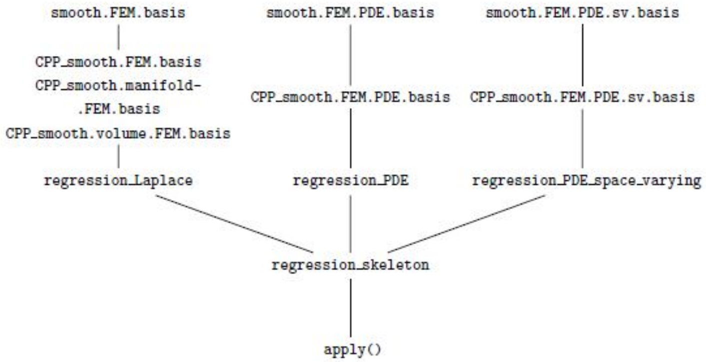
Go back

Lila GitHub - 2017



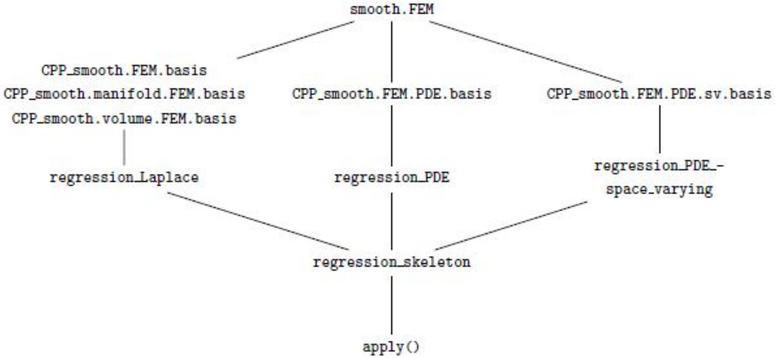
Go back

Negri - 2018



Go back

Colli-Colombo - 2019



Criticalities - Dirichlet BC

PROBLEM: CRASH Index shift overlap

```
R: BC$BC_indices<-as.vector(BC$BC_indices)-1
C++: std::for_each(bc_indices_.begin(),
bc_indices_.end(), [](int& i)i-=1;);
```

SOLUTION Do all index shifts in R.

Criticalities - Space-varying

PROBLEM 1 Wrong indexing of the mesh

`CPP_smooth.FEM.PDE.sv.basis` $\longrightarrow$ mesh indices shifted $\longrightarrow$ Call to `CPP_get_evaluations_points` $\longrightarrow$ mesh indices shifted $\longrightarrow$ ERROR

SOLUTION 1 Remove double index shift.

PROBLEM 2 Missing forcing term in the system

SOLUTION 2 Add boolean `isSpaceVarying` in class `MixedFERegression`

Modify the function `apply`:

```
_b.bottomRows(nnodes)=lambda*forcingTerm;
```

Criticalities - Barycentric coordinates 2.5D

PROBLEM In `getBaryCoordinates`:

```
lambda[k]=std::sqrt(detJ_point)/t.getArea();
```

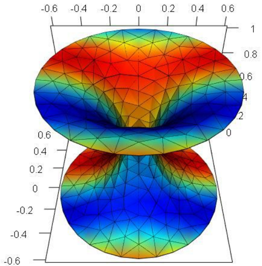
Missing 0.5 factor
For-loop logic does not depend on k

SOLUTION Replicate procedure of `isPointInside` function:

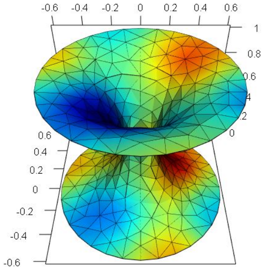
```
sol = A.colPivHouseholderQr().solve(b);
lambda(0)=1-sol(0)-sol(1);
lambda(1)=sol(0);
lambda(2)=sol(1);
```

where `A` 3x2 transformation to the reference triangle

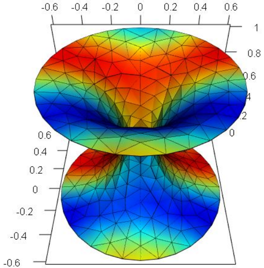
2.5D pointwise data test



(a) True field



(b) Pre-debug



(c) Post-debug

smooth.FEM.basis unification

R-level users can use three functions:

- smooth.FEM.basis: isotropic and stationary model for planar, manifold and volumetric domains;
- smooth.FEM.PDE.basis:anysotropic and stationary case (2D only);
- smooth.FEM.PDE.sv.basis: anysotropic and nonstationary case (2D only);

We unified these functions into a single one called smooth.FEM.

Parameters checking

Updated R helper function checkSmoothingParameters to recognize automatically the kind of PDE parameters:

```
...
space_varying=FALSE                # boolean: returned to smooth.FEM

if(!is.null(PDE_parameters$u)){
space_varying=TRUE
message("Smoothing: anysotropic and non-stationary case")

if(!is.function(PDE_parameters$K))
stop("'K' in 'PDE_parameters' is not a function")
...                                                   #other checks
}else if(!is.null(PDE_parameters)){
message("Smoothing: anysotropic and stationary case")
}
if(is.null(PDE_parameters))
message("Smoothing: isotropic and stationary case")
..
```

Redundant MixedFERegressionBase members

- `Real dirichlet_penalization` - used for BC in Lila;
- `Real pruning_coeff` - used for matrix assembling in Lila;
- `std::vector<coeff> tripletsData_` - Morally a vector to build the system matrix;
- `SpMat Amat` - Old SW block in CRAN-Giussani code;
- `SpMat Dmat` - Old NW block in CRAN-Giussani code;
- `SpMat Mmat` - Old SE block in CRAN-Giussani code;
- `SpMat coeffmatrix_` - The system matrix.

This class was never polished! These members were erased.

Extension to areal data

- $D_1, \ldots, D_n$: n disjoint subsets of $\Omega \subset \mathbb{R}^2$,
- $\bar{z}_i \in \mathbb{R}$ mean values of $z \in L^2(\Omega)$ over D_1, ..., D_n
- $\bar{\mathbf{w}}_i \in \mathbb{R}^q$ mean value of covariates over D_i

$$\boxed{\bar{\mathbf{z}}_i = \bar{\mathbf{w}}_i^T \boldsymbol{\beta} + \frac{1}{|D_i|} \int_{D_i} f + \bar{\epsilon}_i}$$

Estimate $\boldsymbol{\beta}$ and f by minimizing

$$\bar{J}_{\lambda,L,u}(\boldsymbol{\beta}, f) = \sum_{i=1}^{n} |D_i| \left(\bar{z}_i - \bar{\mathbf{w}}_i^T \boldsymbol{\beta} - \frac{1}{|D_i|} \int_{D_i} f \right)^2 + \lambda \int_{\Omega} (Lf - u)^2$$

Numerical solution

The following holds:

$$\begin{bmatrix} \bar{\boldsymbol{\Psi}}^T \mathbf{A}\bar{\mathbf{Q}}\bar{\boldsymbol{\Psi}} & \lambda \mathbf{R}_1^T \\ \mathbf{R}_1^T & -\mathbf{R}_0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}}_{\mathcal{T}}^* \\ \hat{\mathbf{g}}_{\mathcal{T}}^* \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\Psi}}^T \mathbf{A}\bar{\mathbf{Q}}\bar{\mathbf{z}} \\ \mathbf{u} \end{bmatrix} \tag{1}$$

where

$$\bar{\boldsymbol{\Psi}} = \begin{bmatrix} \frac{1}{|D_1|}\int_{D_1}\psi_1 & \cdots & \frac{1}{|D_1|}\int_{D_1}\psi_{N_{\mathcal{T}}} \\ \vdots & \ddots & \vdots \\ \frac{1}{|D_n|}\int_{D_n}\psi_1 & \cdots & \frac{1}{|D_n|}\int_{D_n}\psi_{N_{\mathcal{T}}} \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} |D_1| & & 0 \\ & \ddots & \\ 0 & & |D_n| \end{bmatrix}$$

$$\bar{\mathbf{z}} = (\bar{z}_1, \ldots, \bar{z}_n)^T$$

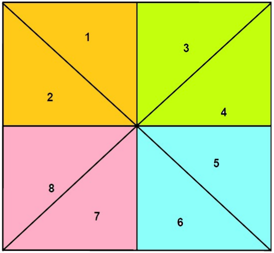
$$\bar{\mathbf{Q}} = \mathbf{I} - \bar{\mathbf{H}}, \qquad \bar{\mathbf{H}} = \bar{\mathbf{W}}(\bar{\mathbf{W}}^T\bar{\mathbf{W}})^{-1}\bar{\mathbf{W}}^T$$

The incidence matrix

$\mathcal{I}$ is a $n \times |\mathcal{T}|$ matrix such that $\mathcal{I}_{ij} = \begin{cases} 1 & \text{if element j is in region i} \\ 0 & \text{otherwise} \end{cases}$

Example:

$$\mathcal{I} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



Implementation

- User can provide the `incidence_matrix` to `smooth.FEM`:

```
smooth.FEM(locations=NULL, observations, FEMbasis,
           lambda, covariates = NULL, PDE_parameters,
           incidence_matrix, BC, GCV =TRUE,
           GCVmethod = "Stochastic")
```

- `matrixXi incidenceMatrix_` has been added in the `regressionData` object;
- matrix $\mathbf{A}$ has been added to the `MixedFERegressionBase`;
- Function `setPsi` is able to build $\bar{\boldsymbol{\Psi}}$;

System solving - Woodbury identity

AIM: invert the system matrix efficiently

Proposition

Woodbury matrix identity:

$$\mathbf{M}^{-1} = (\mathbf{E} + \mathbf{UCV})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VE}^{-1}\mathbf{U})^{-1}\mathbf{VE}^{-1}$$

where $\mathbf{E} \in \mathbb{R}^{l \times l}$, $\mathbf{U} \in \mathbb{R}^{l \times m}$, $\mathbf{C} \in \mathbb{R}^{m \times m}$, $\mathbf{V} \in \mathbb{R}^{m \times l}$.

- $\mathbf{M}$ full system matrix;
- $\mathbf{E}$ sparse symmetric;
- $\mathbf{C}$ full but very small.
- Need to write $\mathbf{M}$ as $\mathbf{E} + \mathbf{UCV}$

Matrix decomposition - pointwise

$$\mathbf{M} = \begin{bmatrix} \mathbf{\Psi}^T\mathbf{Q}\mathbf{\Psi} & \lambda\mathbf{R}_1^T \\ \lambda\mathbf{R}_1^T & -\lambda\mathbf{R}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{\Psi}^T\bar{\mathbf{\Psi}} & \lambda\mathbf{R}_1^T \\ \lambda\mathbf{R}_1^T & -\lambda\mathbf{R}_0 \end{bmatrix} + \begin{bmatrix} \mathbf{\Psi}^T(-\mathbf{H})\mathbf{\Psi} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{E} + \mathbf{B}$$

$$\mathbf{\Psi}^T(-\mathbf{H})\mathbf{\Psi} = (\mathbf{\Psi}^T\mathbf{W})(\mathbf{W}^T\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{\Psi}) = \bar{\mathbf{U}}\bar{\mathbf{C}}\bar{\mathbf{V}}$$

$$\left.\begin{aligned} \mathbf{U} &= \begin{bmatrix} \mathbf{\Psi}^T\mathbf{W} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2K\times q} \\ \mathbf{C} &= -(\mathbf{W}^T\mathbf{W})^{-1} = \bar{\mathbf{C}} \in \mathbb{R}^{q\times q} \\ \mathbf{V} &= \begin{bmatrix} \mathbf{W}^T\mathbf{\Psi} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{V}} & \mathbf{0} \end{bmatrix} = \mathbf{U}^T \ \in \mathbb{R}^{q\times 2K} \end{aligned}\right\} \longrightarrow \boxed{\mathbf{B} = \mathbf{UCV}}$$

Matrix decomposition - areal

$$\bar{\mathbf{M}} = \begin{bmatrix} \bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}\bar{\mathbf{Q}}\bar{\mathbf{\Psi}} & \lambda \mathbf{R}_1^T \\ \lambda \mathbf{R}_1^T & -\lambda \mathbf{R}_0 \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}\bar{\mathbf{\Psi}} & \lambda \mathbf{R}_1^T \\ \lambda \mathbf{R}_1^T & -\lambda \mathbf{R}_0 \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}(-\bar{\mathbf{H}})\bar{\mathbf{\Psi}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \bar{\mathbf{E}} + \bar{\mathbf{B}}$$

$$\bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}(-\bar{\mathbf{H}})\bar{\mathbf{\Psi}} = (\bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}\mathbf{W})(\mathbf{W}^T\mathbf{W})^{-1}(\mathbf{W}^T\bar{\mathbf{\Psi}}) = \bar{\mathbf{U}}\bar{\mathbf{C}}\bar{\mathbf{V}}$$

$$\left.\begin{aligned} &\mathbf{U} = \begin{bmatrix} \bar{\mathbf{\Psi}}^T \color{red}{\mathbf{A}}\mathbf{W} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2K\times q} \\ &\mathbf{C} = -(\mathbf{W}^T\mathbf{W})^{-1} = \bar{\mathbf{C}} \in \mathbb{R}^{q\times q} \\ &\mathbf{V} = \begin{bmatrix} \mathbf{W}^T\mathbf{\Psi} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{V}} & \mathbf{0} \end{bmatrix} \color{red}{\neq} \mathbf{U}^T \in \mathbb{R}^{q\times 2K} \end{aligned}\right\} \longrightarrow \boxed{\bar{\mathbf{B}} = \mathbf{UCV}}$$

Old `MixedFERegressionBase`

- System solved using Woodbury:

$$\mathbf{M} = \mathbf{E} + \mathbf{UCV}$$

- Matrices stored: `A_` (the $\mathbf{E}$), `U_` and `coeffmatrix_` (complete system matrix) ;
- `A_` coincides with the system matrix when there are no covariates;
- `coeffmatrix_` always built, but only used to compute stochastic GCV;
- **Inefficient!**

New MixedFERegressionBase

$$\mathbf{M} = \mathbf{E} + \mathbf{UCV}$$

`coeffmatrix_` = `matrixNoCov_` + `matrixOnlyCov_`

- Matrices stored: `matrixNoCov_` (old `A_`), `U_` and `V_`;
- No more `coeffmatrix_`;
- `matrixNoCov_` and `U_` adapted for areal data;
- Stochastic GCV uses Woodbury decomposition.

Areal system building

The functions `buildMatrixNoCov` and `getRightHandSide` use the areal matrix if needed:

buildMatrixNoCov:

```
 ...
SpMat DMat; //NW block of the matrix

if(regressionData_.getNumberOfRegions()==0) // pointwise data
 DMat=Psi.transpose()*Psi;
else                                // areal data: need to add the
                                       diag(|D_1|,...,|D_N|)
 DMat=Psi.transpose()*A_.asDiagonal()*Psi;
...
```

Areal Woodbury implementation

`system_factorize` adds **A** to the appropriate formulae when areal data is given:

system_factorize:

```
...
U_ = MatrixXr::Zero(2*nnodes, W.cols());
V_=MatrixXr::Zero(W.cols(),2*nnodes);
V_.leftCols(nnodes)=W.transpose()*psi_;

if(regressionData_.getNumberOfRegions()==0){ // pointwise data
 U_.topRows(nnodes) = psi_.transpose()*W;
}
else{                                           //areal data
 U_.topRows(nnodes) = psi_.transpose()*A_.asDiagonal()*W;
    }
 ...
```

Computing areas and integrals

- `MeshHandler` class is extended with the method

  ```
  Real elementMeasure(Id id)
  ```

  to compute the measure of elements of the mesh;
- `evaluator` object is extended with the method

  ```
  void integrate(UInt** incidenceMatrix, UInt nRegions,
  UInt nElements, const Real *coef, Real* result)
  ```

  to evaluate the $\frac{1}{|D_i|} \int_{D_i} f$

Performance comparison

- 2D unit square, divided in n=20 disjoint regions.
- Model

$$\bar{\mathbf{z}}_i = \bar{\mathbf{w}}_i^T \beta + \frac{1}{|D_i|} \int_{D_i} f + \bar{\epsilon}_i$$

$$f(x, y) = a_1 \sin(2\pi x) \cos(2\pi y) + a_2 \sin(3\pi x)$$

$$a_1, a_2 \sim \mathbb{U}_{\{-1.5, 1.5\}}.$$

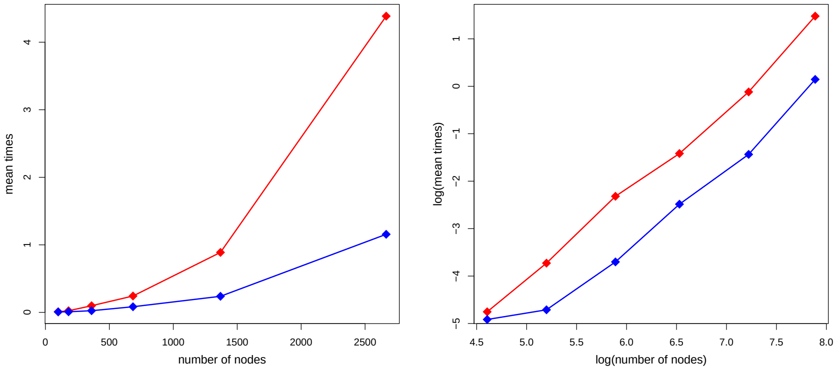
$$\bar{w_i} = \cos(3\pi y_i)$$

$$\beta = 1.2$$

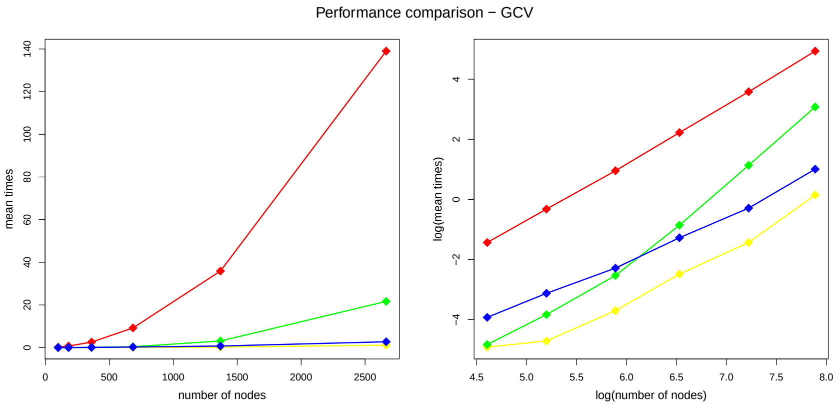
- $\bar{\epsilon}_1$,...,$\bar{\epsilon}_n$ random errors with variance $Var(\bar{\epsilon}_i) \propto \frac{1}{|D_i|}$.

1. **Standard resolution**, using the full system matrix
2. **Woodbury decomposition**

Performance comparison − System solving



1. **No GCV**
2. **Exact GCV**, computed via sequence of linear systems
3. **Stochastic GCV naive**
4. **Stochastic GCV Woodbury**



Testing areal data

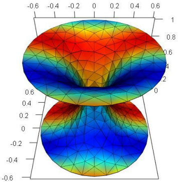
- Model for data generation:

$$\bar{\mathbf{z}}_i = \bar{\mathbf{w}}_i^T \beta + \frac{1}{|D_i|} \int_{D_i} f + \bar{\epsilon}_i$$

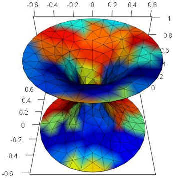
- N=50 repetitions of the simulation
- Evaluation of the estimate:

$$RMSE = \sum_{i=1}^{N} \frac{(f_i + \mathbf{W}\beta_i - \hat{f}_i - \mathbf{W}\hat{\beta}_i)^2}{\#nodes}$$

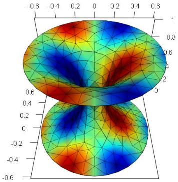
2.5D Hub (Areal)



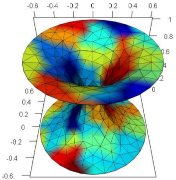
(a) True field



(b) True field - areal

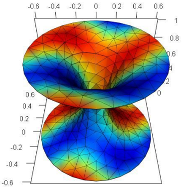


(c) True covariate

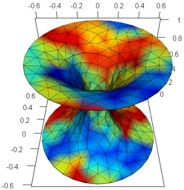


(d) True covariate - areal

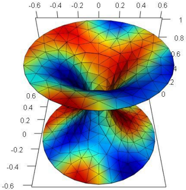
Areal isotropic smoothing: $\mathbf{W} \neq \mathbf{0}$



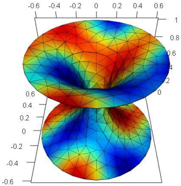
(a) Total field



(b) Areal data with noise

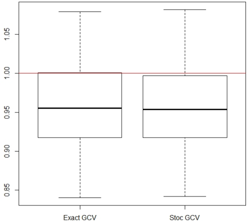


(c) Exact GCV

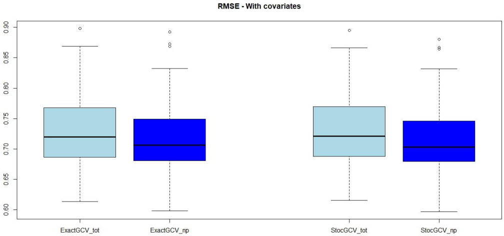


(d) Stochastic GCV

Estimates quality



(a) Betas



(b) RMSE

Steps for release

Requisites to release on CRAN: `R CMD check` giving no `ERRORS` nor `WARNINGS`.

Main changes:

- Removing MUMPS for size and portability;
- Revision of package manual;
- Correction of all warnings.

Checks performed on Windows and Linux.

Release code

Github:

`https://github.com/AlessandraColli/fdaPDE`

CRAN release:

`https://cran.r-project.org/web/packages/fdaPDE/index.html`

Main bibliographic references

- Eardi Lila, Laura M. Sangalli, Jim Ramsay and Luca Formaggia (2017). *fdaPDE: Functional Data Analysis and Partial Differential Equations; Statistical Analysis of Functional and Spatial Data, Based on Regression with Partial Differential Regularizations*. R package version 0.1-5.
- Sangalli L.M., Ramsay J.O., and Ramsay T.O. *Spatial spline regression models*. Journal of the Royal Statistical Society Ser. B, Statistical Methodology, 75, 4, 681-703, 2013.
- Ettinger B., Perotto S., Sangalli L.M. *Spatial regression models over two-dimensional manifolds*. Biometrika, 103 (1), 71-88, 2016.
- L. Azzimonti, L.M. Sangalli, P. Secchi, M. Domanin, F. Nobile, *Blood flow velocity field estimation via spatial regression with PDE penalization*, 2015, JASA.
- Lila, E., Aston, J. A. D., and Sangalli, L. M. (2016). *Smooth Principal Component Analysis over two-dimensional manifolds with an application to Neuroimaging*.

Main bibliographic references

A. Azzolin, E. Lila, *Regression models with differential regularization*, APSC report 2014

C. Bambini, L. Giussani, *Regression Models with Differential Regularization - Efficient computation of the Equivalent Degrees of Freedom*, APSC report 2017

A. Cosmo, M. Beraha, *Linear Models with Partial Differential Equations Regularization for data distributed over 2D manifolds*, APSC report 2017

L. Negri , *Smooth Functional Principal Component Analysis-Regularization technique for data distributed over planar and two-dimensional manifold domains*, APSC report 2018

L. Negri, *Functional Principal Component Analysis Over Volumetric Domains with Neuroimaging Applications*, Thesis 2018

R Core Team, *Writing R extensions - Version 3.6.1*, 2019.