Machine Learning with Functional Data

Introduction and representation

Objectives

- Overview of statistical learning with functional data.
- Introduction of some important concepts in applied statistics.
- Introduction to the Python package scikit-fda.

Acknowledgments

Part of the material, figures, and slides in this course are derived from courses and talks given by:

Prof. Antonio Cuevas González (UAM)

Prof. Manuel Febrero Bande (USC)

Prof. José Ramón Berrendero Díaz (UAM)

Carlos Ramos Carreño (UAM)

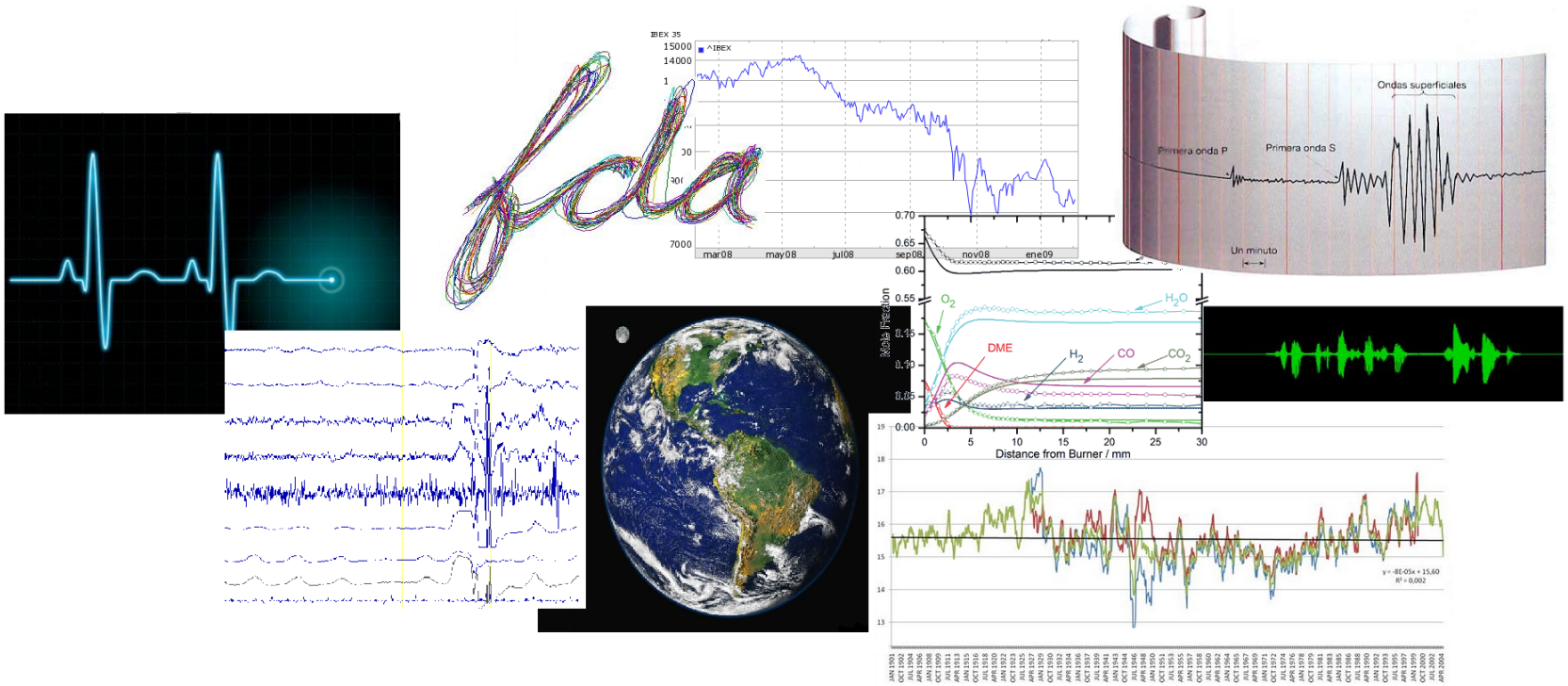
Historical evolution of statistical inference

Statistical theory	**Data**	**Parameters**	**Origin**
Classical inference	$\mathbb{R}$	$\mathbb{R}$	1920
Multivariate analysis	$\mathbb{R}^d (n >> d)$	$\mathbb{R}^k (n >> k)$	1940
Non parametrics	$\mathbb{R}^d (n >> d)$	$\mathcal{F}$	1960
High dimensionality	$\mathbb{R}^d (n < d)$	$\mathbb{R}^k$	2000
Functional data analysis	$\mathcal{F}$	$\mathbb{R}^k$ o $\mathcal{F}$	1990
Object oriented data analysis	Structures	$\mathbb{R}^k$ o $\mathcal{F}$	2010

n is the number of elements in the sample and d is the dimension of the observations (number of variables). Cuevas 2014

Functional data analysis

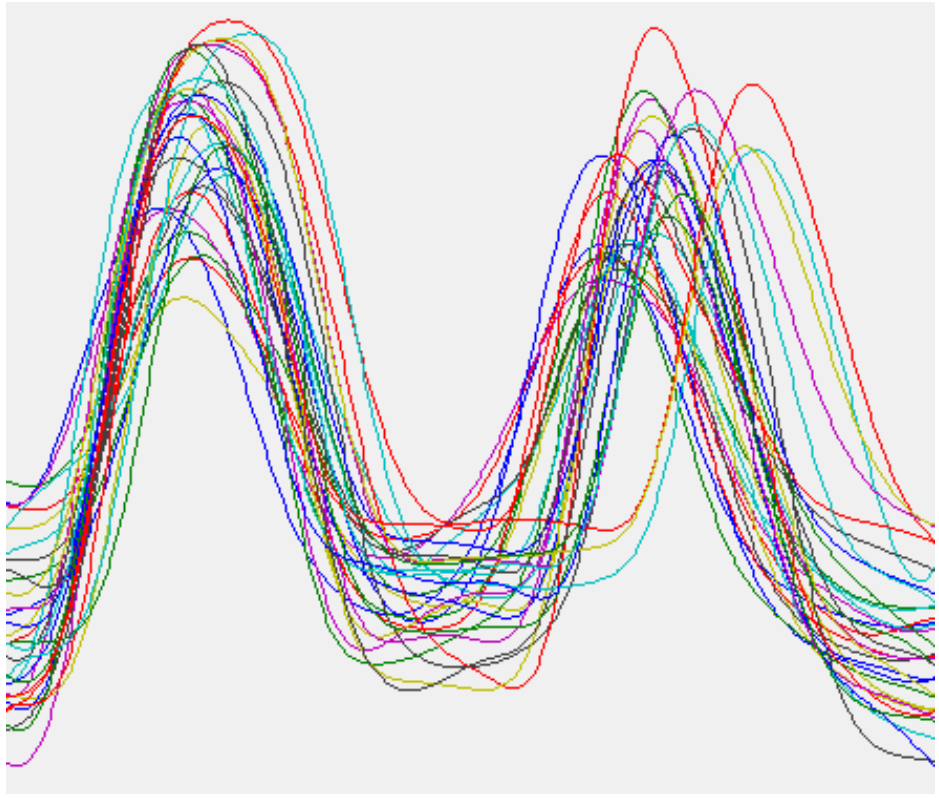
Functional data Analysis (FDA) (Ramsay, 1982) is the branch of statistics that studies those problems in which data vary over a continuum.
It usually appears related to continuous monitoring processes.



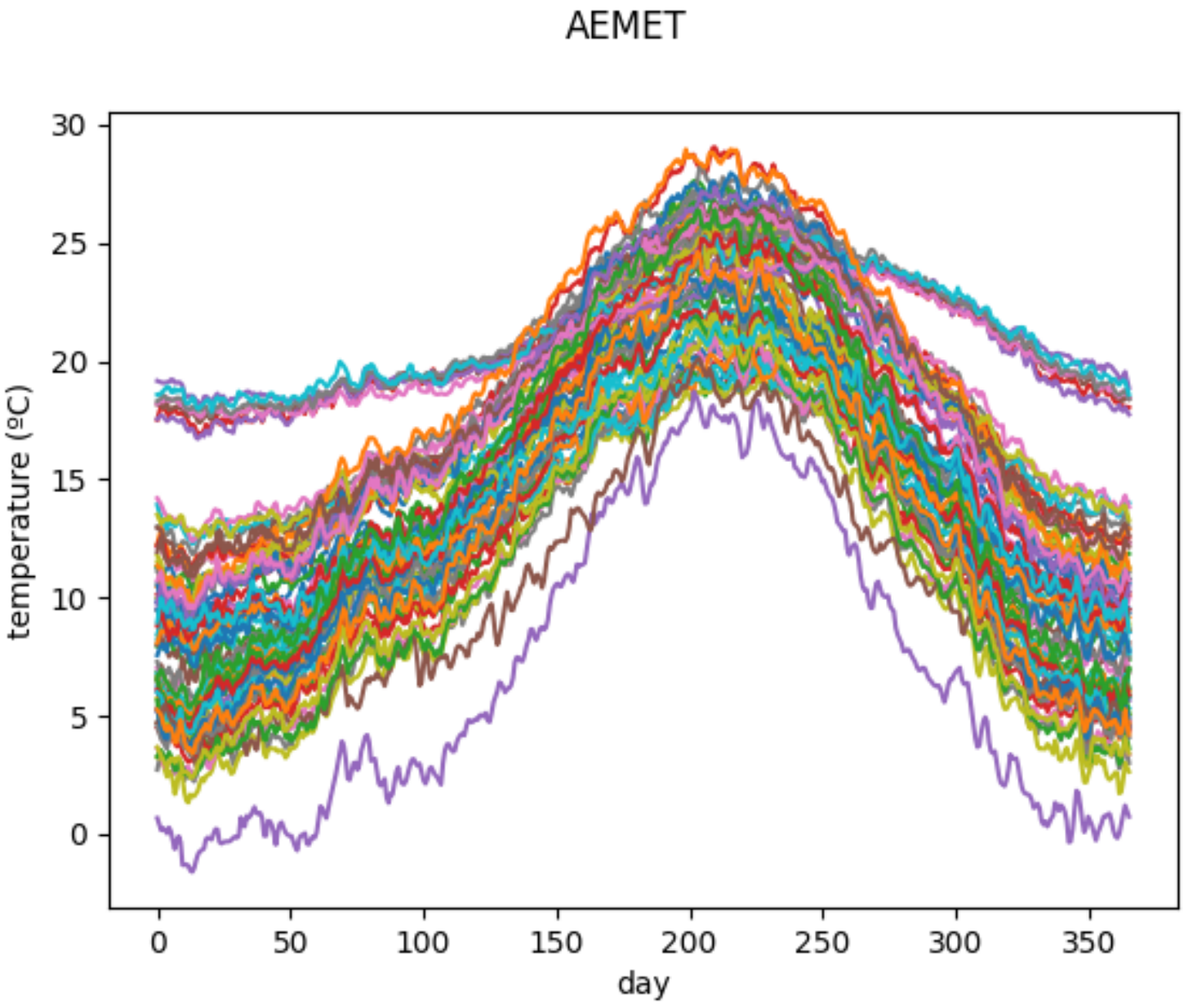
FDA: goals, problems...

but...

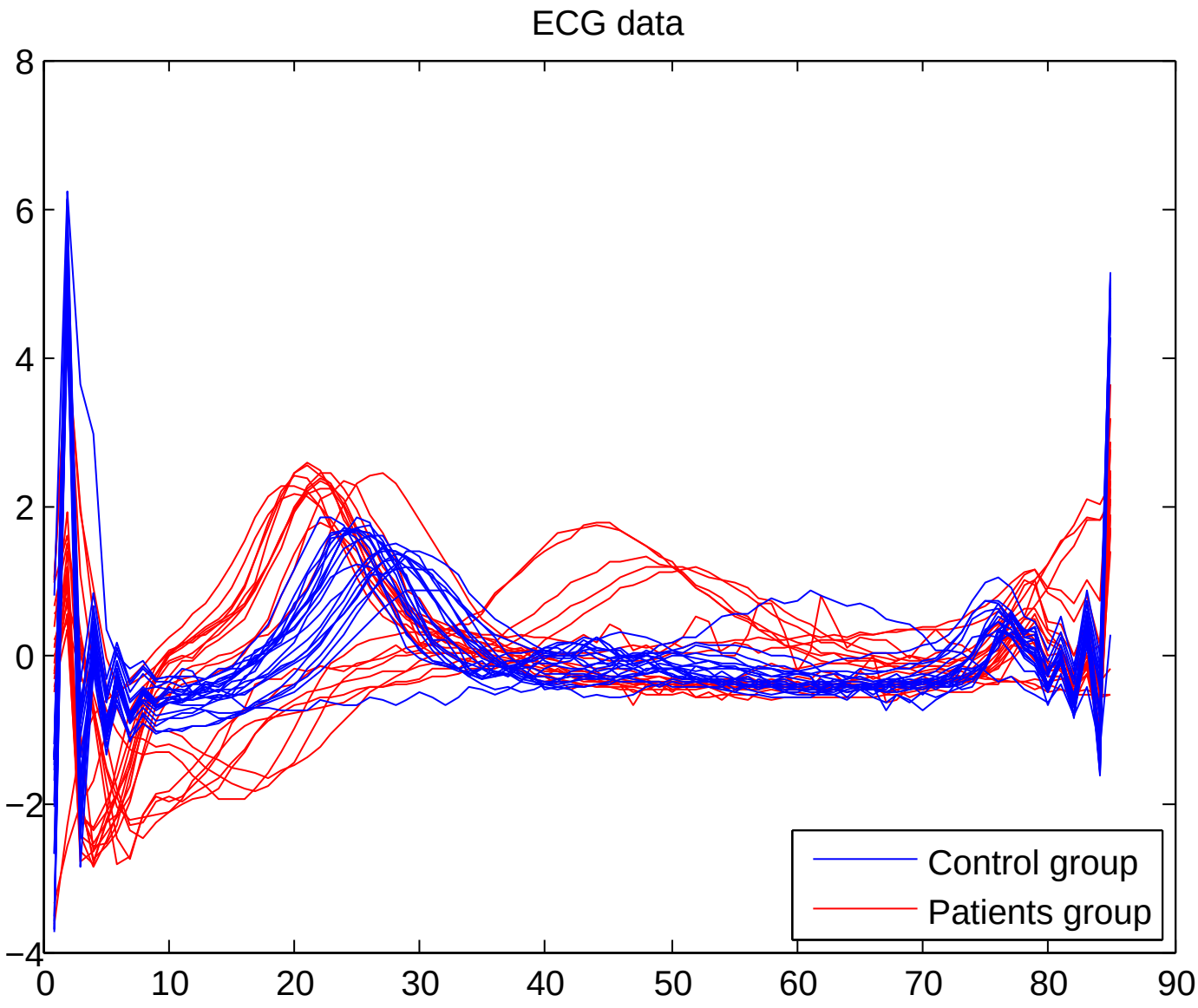
- Where is the information?
- Which is the similarity criterion?
- How can we deal with functional data?



FDA: Examples - Spanish Weather



FDA: Examples - Electrocardiograms



Related Concepts

- FDA: Regular and dense measures
- Longitudinal data: Irregular and sparse measures
- Spatial data: $X(s, t)$
- Multivariate FDA: Vectors of functions
- Dependent functional data: Functional time series, spatially distributed curves
- Time series: An observation evolving over time
- High dimension: $X \in \mathbb{R}^d$, $d > n$ (Magnitude vs shape)

References

- Ramsay and Silverman (2005, 1st ed. in 1997)
- Ramsay and Silverman (2002) Applications
- Graves et al. (2009) R, MATLAB
- Ferraty and Vieu (2006) Non parametric approach
- Bosq (2012, 1st ed. in 2000), Bosq and Blanke (2008) Mathematical tools
- Horvath and Kokozska (2012) Dependent data
- Hsing and Eubank (2015) Theoretical foundations
- Reimherr and Kokozska (2017)

- Cuevas (2014)
- Wang et al. (2015)

Where do the functional data live?

- $(\mathcal{C}[0,1], \|.\|)$: Banach space of continuous real functions in $[0,1]$ endowed with the supremum norm (Billingsley, 1968),

$$\|x\| = \sup_{t \in [0,1]} |x(t)|.$$

- $\left(L^2[0,1], \langle . \rangle\right)$: Hilbert space of square integrable real functions in $[0,1]$ with the usual inner product and the associated norm.

$$\langle x, y \rangle = \int_0^1 x(t) y(t) dt,$$

- Subspaces associated with semi-norms, for example

$$\|x\| = \langle x', x' \rangle^{1/2}$$

- Reproducing kernel Hilbert spaces (RKHS) associated with the covariance kernel of the process.

Functional spaces

- Closed and bounded sets are not necessarily compact.
- There is no natural ordering (no distribution functions). The notion of centrality is lost (outliers, depth).
- There is no natural measure (Lebesgue-like) that is translation-invariant for calculations (no density functions).
- Continuity. Linear operators have a much more complex structure (issues in linear regression).
- Probability measures are more difficult to handle. In particular, weak convergence of a sequence of random variables $\{X_n(t)\}$ to another $X(t)$ is not determined by the convergence of finite-dimensional marginals.

Functional spaces II

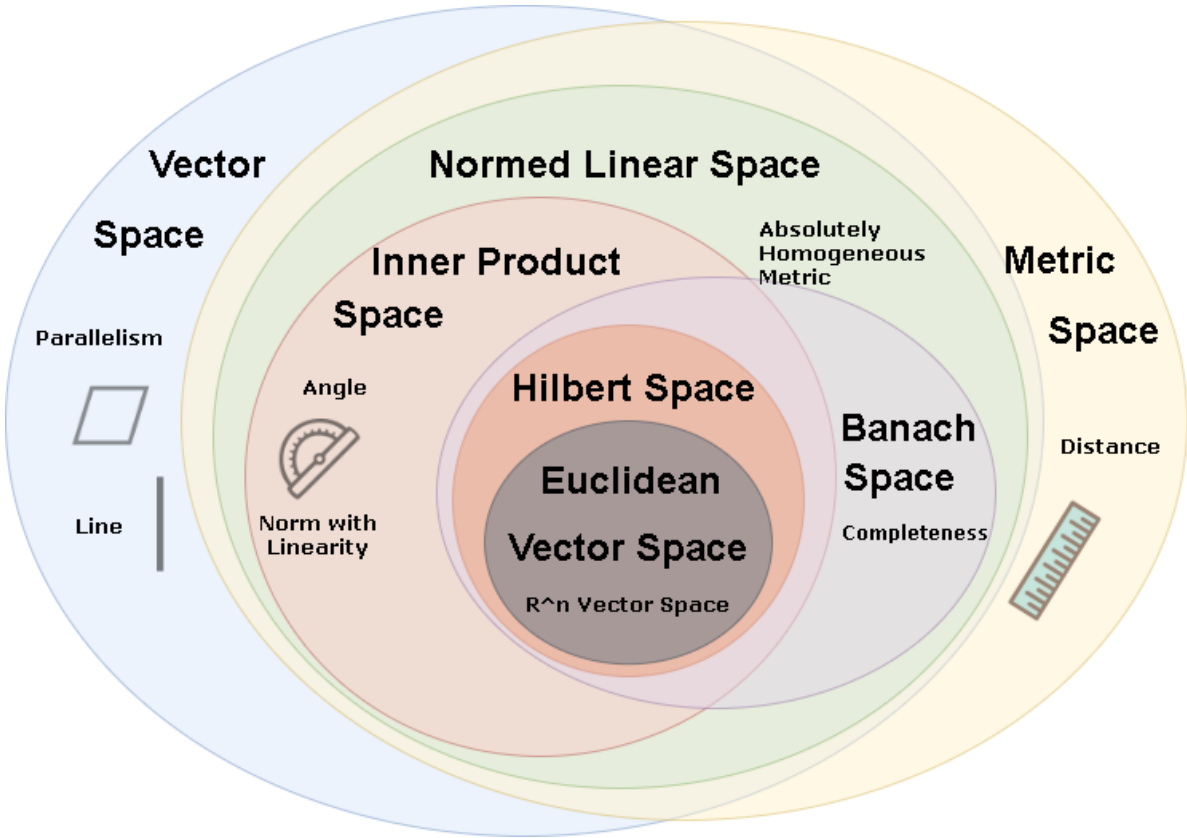
- The different norms are not equivalent and provide different information. It is not always easy to choose the most appropriate metric (closeness).
- In many cases, we may not have a Hilbert structure, making it impossible to measure angles and losing the notion of orthogonality.
- The balls $B(x, \epsilon)$ tend to have small probabilities. For example, in the case of Brownian motion $X = B(t)$, we have

$$\mathbb{P}(X \in B(0, \epsilon)) := P_X(B(0, \epsilon)) \sim \exp(-1/\epsilon^2), \text{ cuando } \epsilon \to 0.$$

As a result, more observations are needed to fill the space. This poses challenges in non-parametric methods, which often have slower convergence rates.

Functional spaces III

- High dimensionality and all related issues.
- Functional data are difficult to record.
- It is not clear what the best representation is.
- Graphical tools are designed for L^2.



scikit-fda

functional data analysis in python

Documentation: https://fda.readthedocs.io/en/stable/

Github: https://github.com/GAA-UAM/scikit-fda

PyPI: https://pypi.org/project/scikit-fda/

A joint work... in progress

- Alberto Suárez González - EPS-UAM
- Carlos Ramos Carreño - EPS UAM

- Álvaro Castillo
- Miguel Carbajo
- David García Fernandez
- Amanda Hernando
- Rafael Hidalgo
- Yujian Hong
- Pablo Marcos Manchón
- Pedro Martín
- Pedro M. Rodríguez-Ponga
- Pablo Pérez Manso
- Elena Petrunina
- Óscar Pinto Santamaría
- Luis A. Rodríguez Ramírez
- Álvaro Sánchez Romero
- Martín Sánchez Signorini
- Diego Serna
- David del Val

Ramos-Carreño et al. scikit-fda: A Python Package for Functional Data Analysis. *Journal of Statistical Software*. To appear

Some Python characteristics

- General purpose language.
- Robust standard library and many specific packages available.
- High-level language: easy to code and easy to read, no memory management or system architecture knowledge required.
- Free and open-source.
- Interpreted: the source code is executed line by line, not compiled.
- Portable: same code can be used on different machines and operative systems.
- Object-Oriented: design is focused on data and objects rather than functions and logic (also supports procedure-oriented programming).
- Dynamically typed: no need to declare the type of the variables. Can lead to runtime errors.
- Poor memory efficiency: in order to simplify programming, Python need a lot of memory space.
- Slow speed: Python cannot be used when speed is important.
- Version incompatibility and lack of support among libraries.
- Lack of statistical models and tests in core libraries, need of third-party packages.

Why Python?

- **FDA R community**: fda, fda.usc, tidyfun, refund, fdapace, funData, roahd, fds, Funclustering, fdasrvf, MFPCA, fdaPDE, sparseFLMM, FDBoost, rainbow, ftsa, funcy... and much more. https://cran.r-project.org/web/views/FunctionalData.html
- **FDA Python community**: fdasrsf (Tucker 2020) and FDApy (Golovkine 2021).
- Powerful, easy to use, generic purpose programming language.
- The SciPy environment: Numpy (arrays and linear algebra), SciPy (utilities for statistics, formats, integration...), Matplotlib, Jupyter (interactive Notebooks)...

Why scikit?

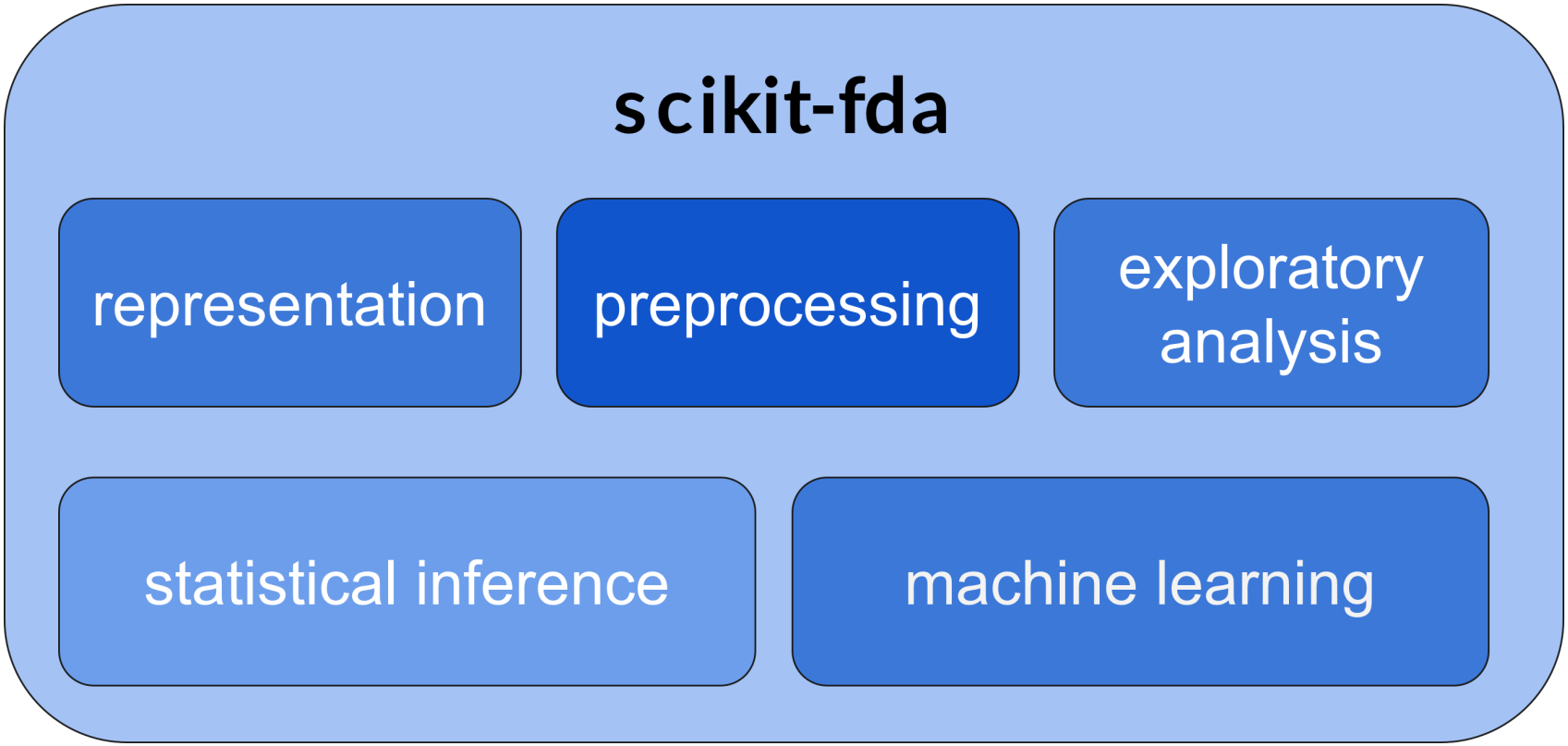
- Scipy Toolkits (SciKits)
- Specialized science packages

scikit-fda

functional data analysis in python

- Python package for Functional Data Analysis (FDA).
- General purpose: it provides a comprehensive set of tools for representation, preprocessing, exploratory analysis and statistical learning with functional data.
- Efficient, flexible and easy to use.
- Fully integrated in the Python scientific ecosystem (SciPy).
- Automatic tools for code and documentation quality.
- Free and open-source software under a BSD 3-Clause license.
- Open to contributions from the FDA community (not only code).

Library development



Currently, main active research lines are dimensionality reduction (PLS and variable selection), regression with functional response, sparse representation and linear discriminant.

The class FData

The *scikit-fda package provides tools for the representation of functional observations of the form* $x : \mathbb{R}^p \longrightarrow \mathbb{R}^q$, *with arbitrary* $p \geq 1$, *and* $q \geq 1$ *(although...).*

A functional dataset consists of a collection of N observations $\{x_n(t), t \in \mathcal{T}\}_{n=1}^{N}$

- `dataset_name`: *Name of the functional dataset.*
- `n_samples`: *Sample size, N.*
- `dim_domain`: *Dimension of the domain, p (inputs).*
- `domain_range`: *Limits of the intervals for each of the domain arguments. They are used as the default ranges for plotting and numerical integration.*
- `dim_codomain`: *Dimension of the codomain q (outputs).*
- `extrapolation`: *Default extrapolation strategy; for instance, constant or periodic.*

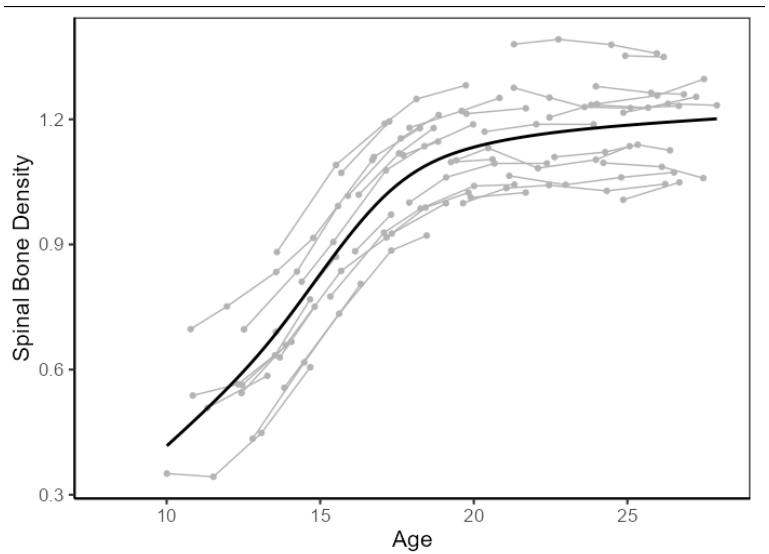
Does functional data really exist?

The point is that the data are always collected discretely,

$$x(t) = (x(t_1), \ldots, x(t_M))$$

(we are only able to observe discrete phenomena). There are two main cases:

- *Densely recorded data*: Discretization points $t_1 < \ldots < t_M$ can be arbitrarily close. Usually in a regular grid. For example, measurements with high-precision devices.
- *Sparsely recorded data*: There are limitations when choosing the discretization points, and they may be different in the different observations. For example, data form longitudinal studies in medicine.



In practice...

- In practice we have data discretized in a set of points $\{t_j\}_{j=1}^{M} \in \mathcal{T} = [0, T]$.
- Therefore, a functional sample $\{x_i(t)\}_{i=1}^{N}$ can be representd as a matrix with N rows (trajectories) and M columns (variables, $\{x_i(t_j)\}$).
- High dimensionality: typically $M > N$.
- Continuity: correlation between close variables grows with M.

However, functional data have an intrinsic structure that differentiates them from multivariate vectors. This makes it necessary to adopt functional approaches and models. Even if the resulting algorithm is going to be multivariate, it has to be done "right".

"The term functional in reference to observed data refers to the intrinsic structure of the data rather than to their explicit form."

Ramsay and Silverman (2005)

Objectives

- Recover the original function from the discretized data.
- Reduce dimension while preserving information.
- Remove noise from measurement errors.
- Obtain smoother versions (derivatives).

Methods

- Discretization.
- Coordinate expression in a basis.
- Smoothing.
- Dimensionality reduction (projection or variable selection).

Discrete representation

When dealing with dense functional data, the simplest representation is discretization.

Let $x(t)$ be a functional datum taking values in $[0, T]$. The easiest discretization consists in taking the partition $\{t_i\}_{i=1}^{M}$ equispaced, such that $0 \leq t_1 < t_2 < \cdots < t_d \leq T$, and considering the variables $\{x(t_i)\}_{i=0}^{M}$.

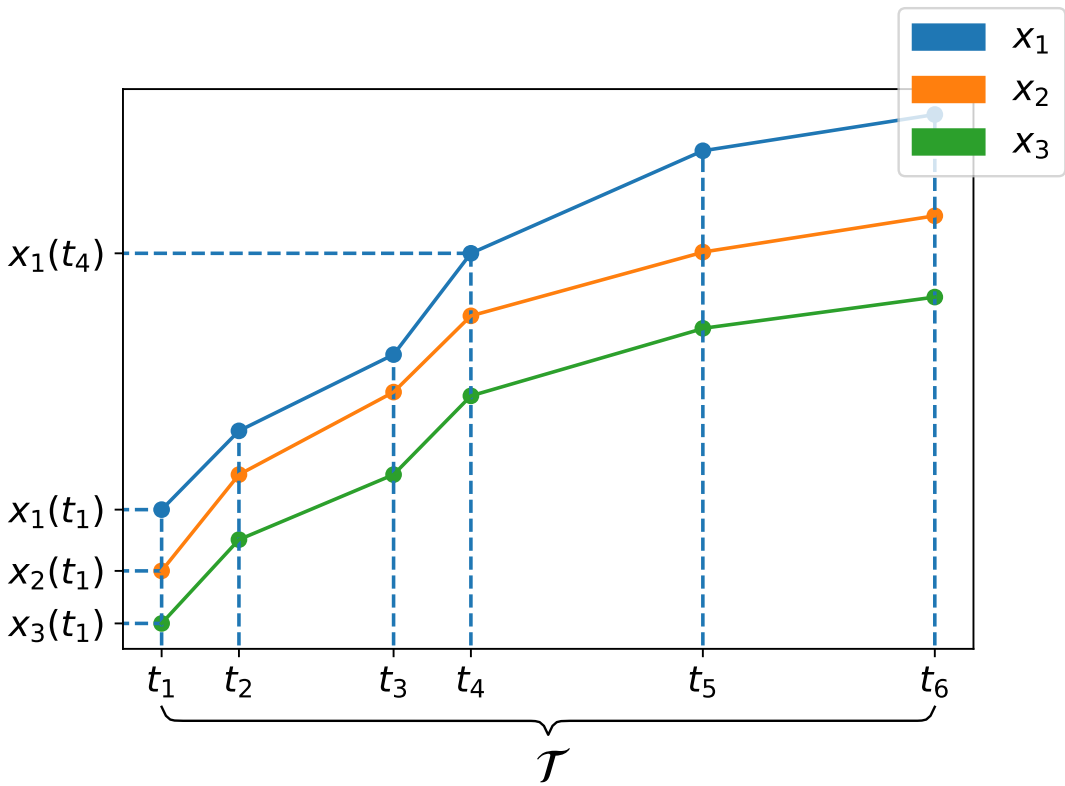
Difficulties

- Density of the partition (M).
- Distribution of the grid points.
- Values at discretization points.

The class FDataGrid

Functional data are often the result of monitoring a continuous process at a discrete set of points. For $p = q = 1$, $x(t) = (x(t_1), \ldots, x(t_M))$.

- `grid_points`: Sequence of discretization points $t_1, \ldots, t_M$
- `data_matrix`: *NumPy array of dimensions $N \times M$ in which the values of the N functional observations are stored.*
- `interpolation`: *Interpolation strategy for locations within the discretization grid. Linear by default.*

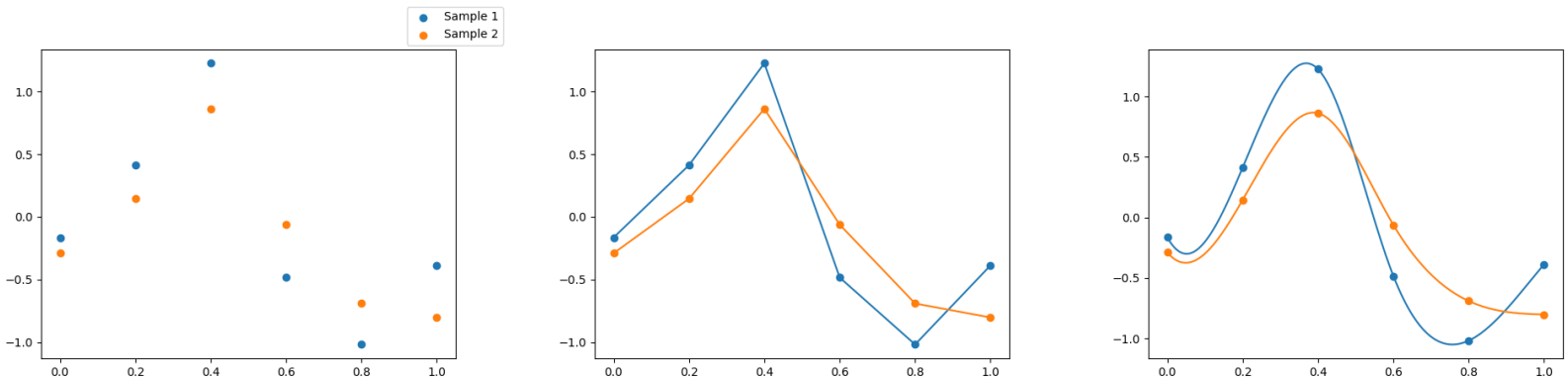


Interpolation

Interpolation allow us to evaluate the trajectories at points not included in the grid. In particular, it is necessary for plotting.

By default *sciki-fda* applies linear interpolation, simpler and computationally cheap, but limited (e.g. the result is non-differentiable).

The interpolation method of the *FDataGrid* could be changed setting the *attribute* *interpolation*.

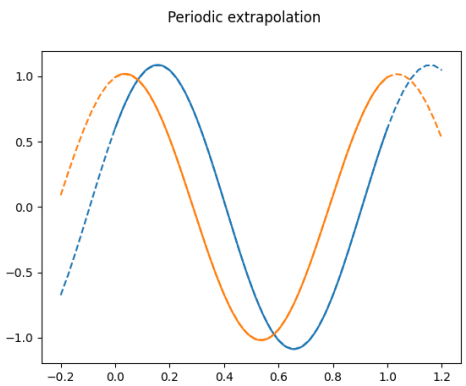


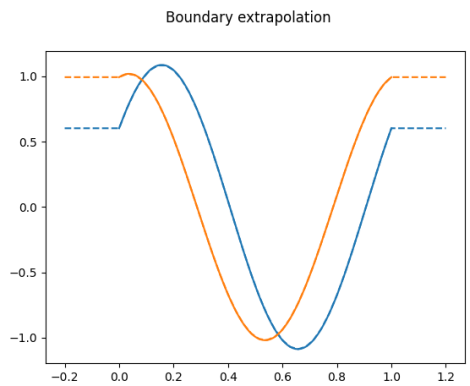
Extrapolation

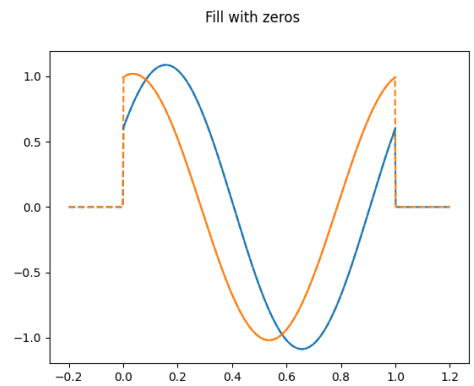
The extrapolation defines how to evaluate points that are outside the domain range of a *FData* object.

The extrapolation criterion of the *FData* could be changed in the *attribute* *extrapolation*.

If the it is not specified (type "*none*"), behavior outside the domain will change depending on the representation, obtaining a periodic behavior in the case of the Fourier basis and polynomial behaviors in the rest of the
cases.







Definition

A functional basis is a set of independent functions $\{\phi_k\}_{k\in\mathbf{N}}$ that allow us to express functions as series.

$$X(t) = \sum_{k\in\mathbb{N}} c_k\phi_k(t) \sim \tilde{X}(t) = \sum_{k=1}^{K} c_k\phi_k(t)$$

Coefficients

$$\begin{pmatrix} \langle X,\phi_1\rangle \\ \vdots \\ \langle X,\phi_i\rangle \\ \vdots \\ \langle X,\phi_K\rangle \end{pmatrix} = \begin{pmatrix} \langle \phi_1,\phi_1\rangle & \cdots & \langle \phi_i,\phi_1\rangle & \cdots & \langle \phi_K,\phi_1\rangle \\ \vdots & & \vdots & & \vdots \\ \langle \phi_1,\phi_i\rangle & \cdots & \langle \phi_i,\phi_i\rangle & \cdots & \langle \phi_K,\phi_i\rangle \\ \vdots & & \vdots & & \vdots \\ \langle \phi_1,\phi_K\rangle & \cdots & \langle \phi_i,\phi_K\rangle & \cdots & \langle \phi_K,\phi_K\rangle \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_K \end{pmatrix}$$

Orthonormal bases are preferable to simplify, $\langle \phi_i,\phi_j\rangle = \delta_{i,j}$

Advantages

- Dimensionality reduction keeping the "functionality".
- $\tilde{X}(t)$ is a smoothed version of X. Get rid of noise and numerical artifacts.
- Useful for derivatives

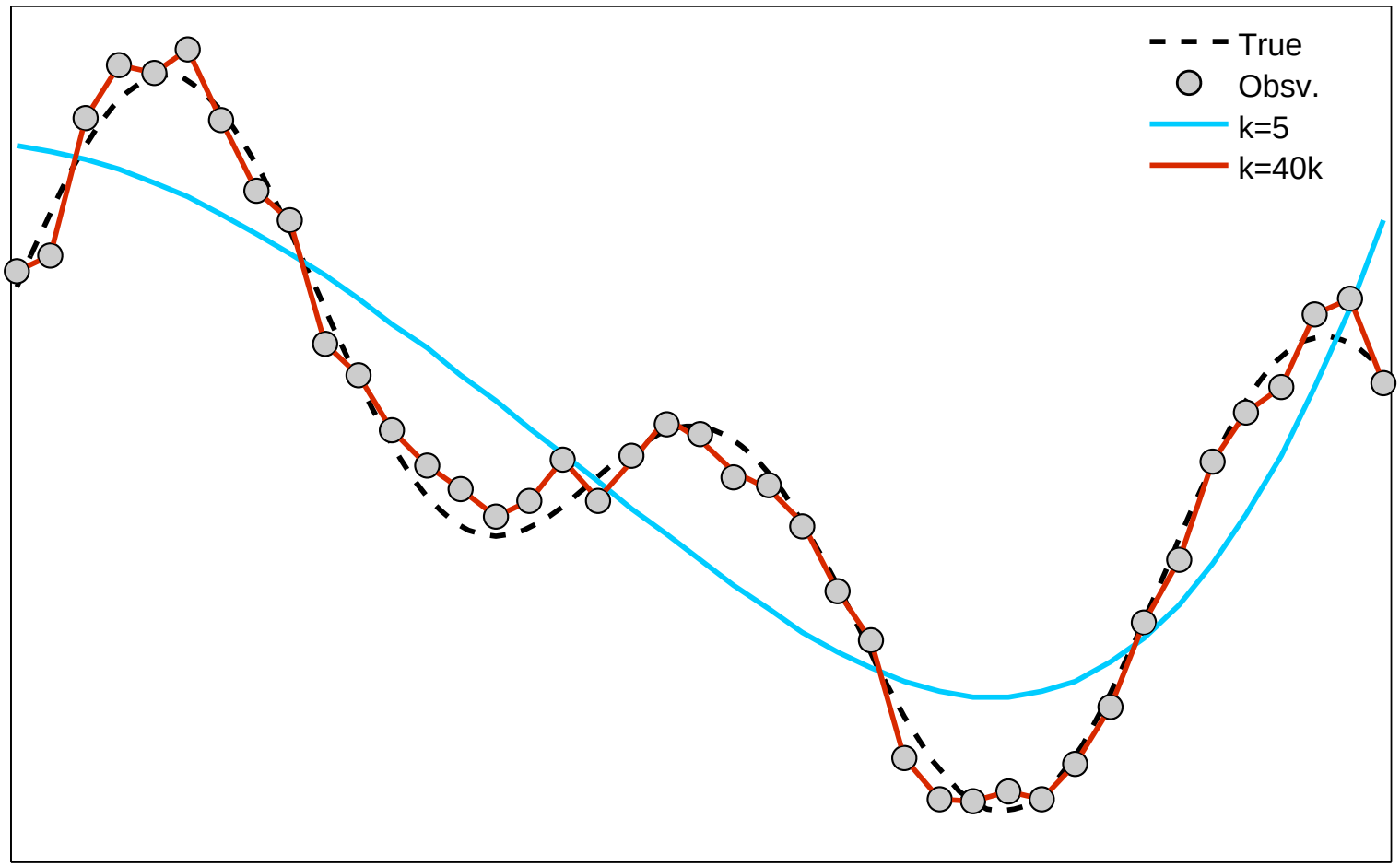
$$X^{(p)}(t) = \sum_{k \in \mathbb{N}} c_k \phi_k^{(p)}(t) \sim \sum_{k=1}^{K} c_k \phi_k^{(p)}(t)$$

Disdvantages

- It is an approximation, information loose.
- Estimation of K: undersmoothing and overfitting.
- Basis election.

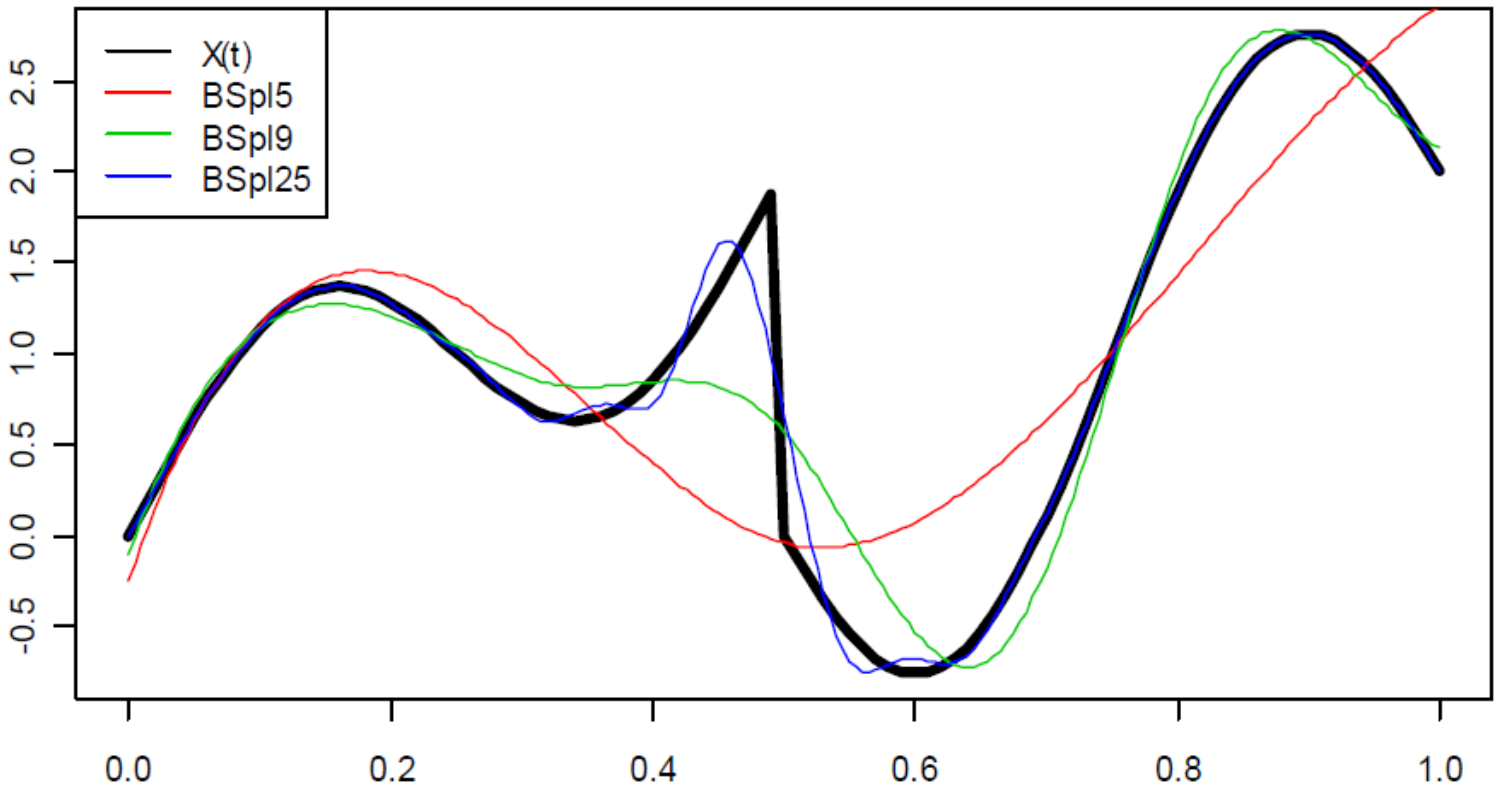
Election of K

K plays the role of a smoothing parameter.



Which basis to choose?

$\tilde{X}(t)$ inherits its properties from the base functions.



Pre-processing the data: basis representation

Basis representation Basis representation is a very usual way to transform the data: assume that $x \in L^2[0,1]$. Then, if $\{e_k(t)\}$ is a orthonormal basis of that space, we may think of fitting a function $\tilde{x}$ from the raw data in the following way: we fix a number J of basis functions, typically smaller than n, and we define

$$\tilde{x}(t) = \sum_{j=1}^{J} c_j e_j(t),$$

where the "Fourier coefficients" c_j are chosen in order to minimize

$$\sum_{k=1}^{N} \left(x(t_k) - \sum_{j=1}^{J} c_j e_j(t_k) \right)^2$$

Pre-processing the data: basis representation

Then, this representation process can be summarized in terms of two transformations,

$$(x(t_1), \ldots, x(t_N)) \mapsto (c_1, \ldots, c_J) \mapsto (\tilde{x}(t_1), \ldots, \tilde{x}(t_{N1})),$$

Hence, this procedure provides both a more compact representation of the data (as J is typically much smaller than N) and a "denoising" process, since $\tilde{x}$ can be considered as a smoothed version of the original data x.

Fourier

The extension of X in the Fourier basis has the form

$$\hat{X}(t) = c_0\phi_0 + \sum_r c_{2r-1}\phi_{2r-1}(t) + c_{2r}\phi_{2r}(t),$$

where $\phi_0(t) = \frac{1}{\sqrt{T}}$, $\phi_{2r-1}(t) = \frac{sen(r\omega t)}{\sqrt{T/2}}$, $\phi_{2r}(t) = \frac{cos(r\omega t)}{\sqrt{T/2}}$.

- Periodicity.
- Orthonormality when nodes are equispaced.
- Fast computation through the fast Fourier transform (FFT).
- Differentiability.
- Suitable for smooth and periodic functions.

Fourier basis

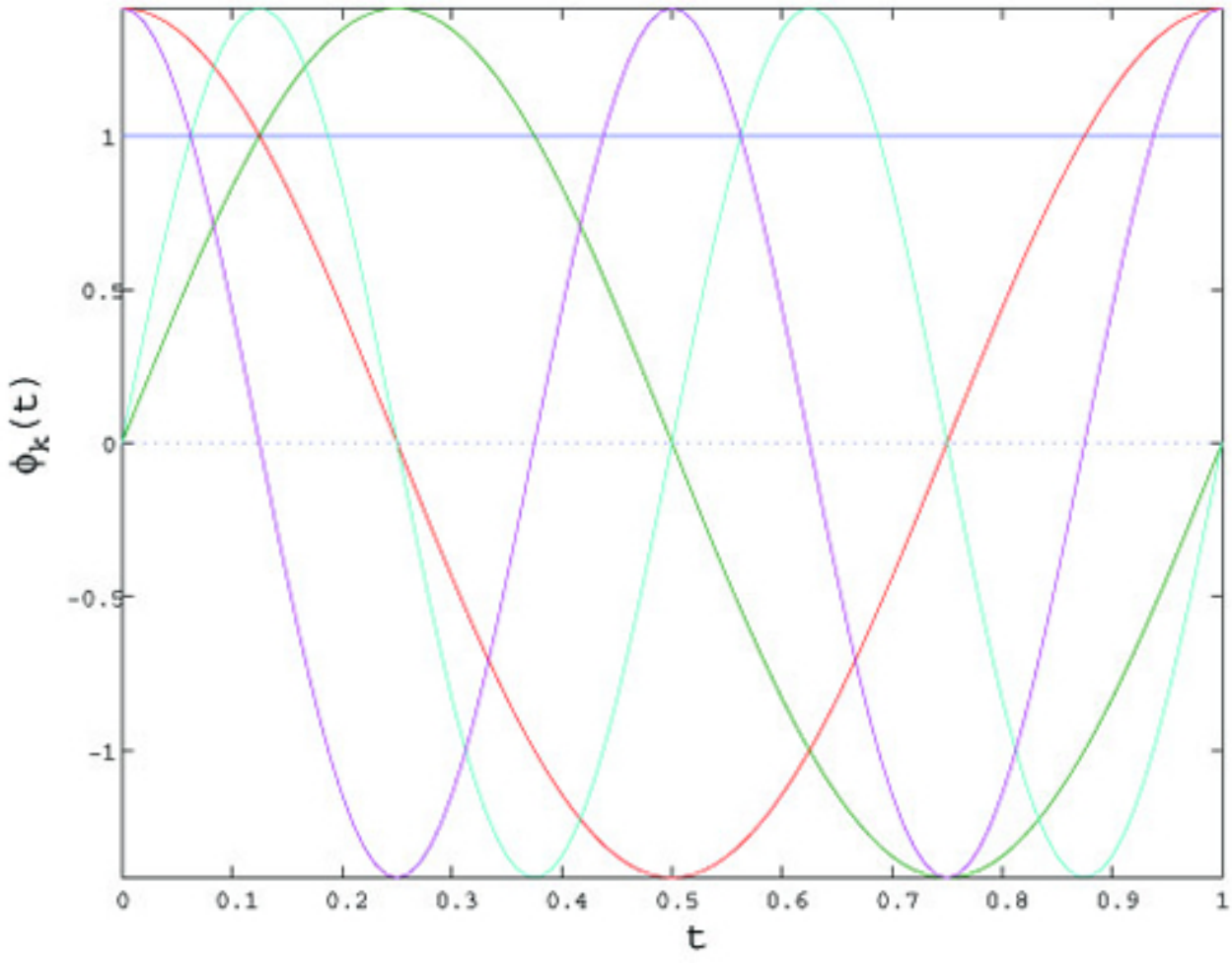
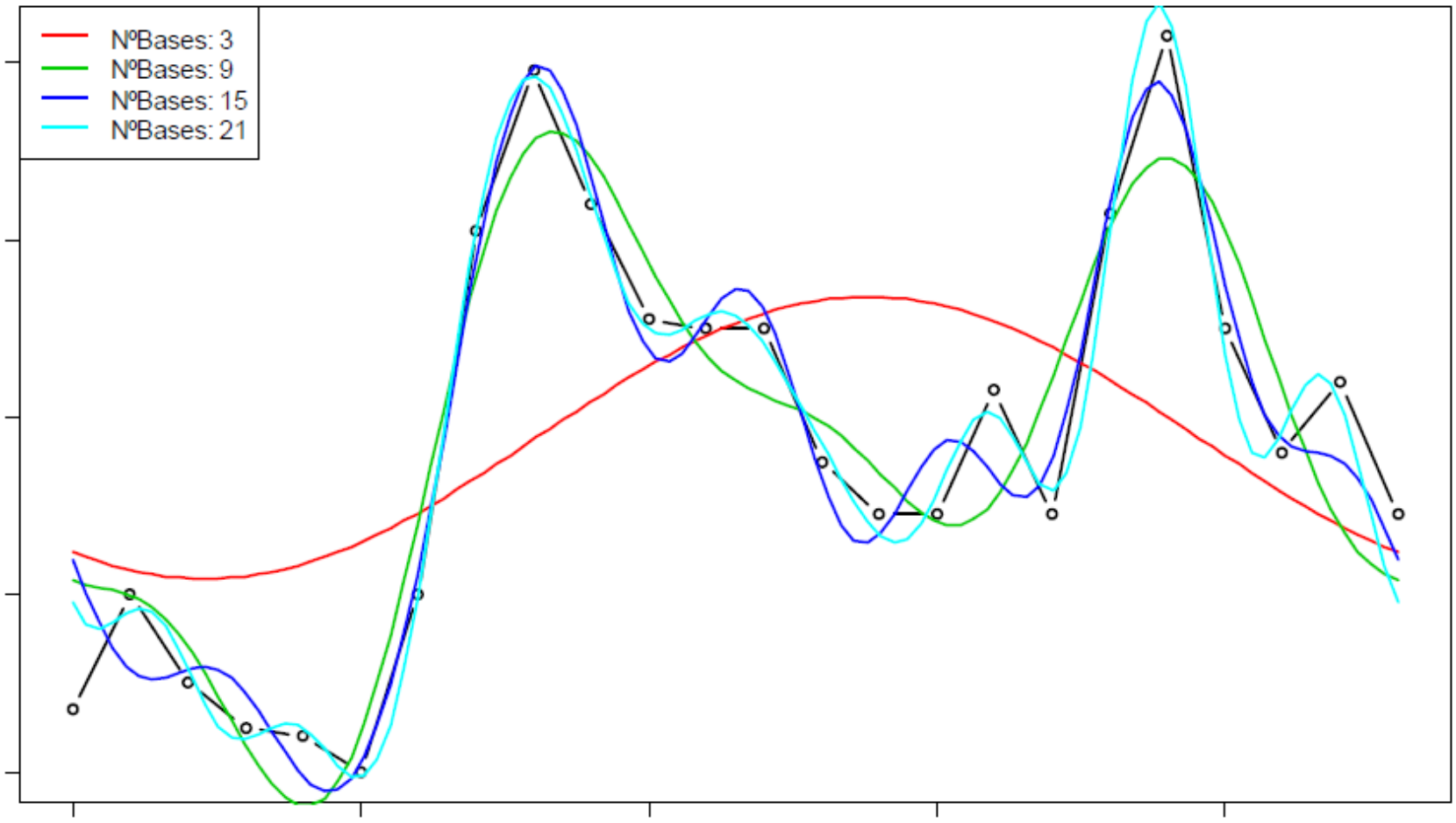


Figure: Base de Fourier de 5 elementos

Fourier basis



B-Spline De Boor (1978)

First, split the domain interval in L subintervals. Then a spline is a set of polynomials of order m defined in each subinterval so that they coincide in the nodes of the partition τ up to the derivative $m-2$.
B-splines (spline bases) approximate functions by splines as follows:

$$S(t) = \sum_{k=1}^{m+L-1} c_k B_k(t, \tau)$$

- Computationally efficient and flexible.
- The de Boor's algorithm calculates the coefficients easily.
- $m + L - 1$ parameters.
- Suitable for non periodic or irregular trajectories.

B-Spline basis

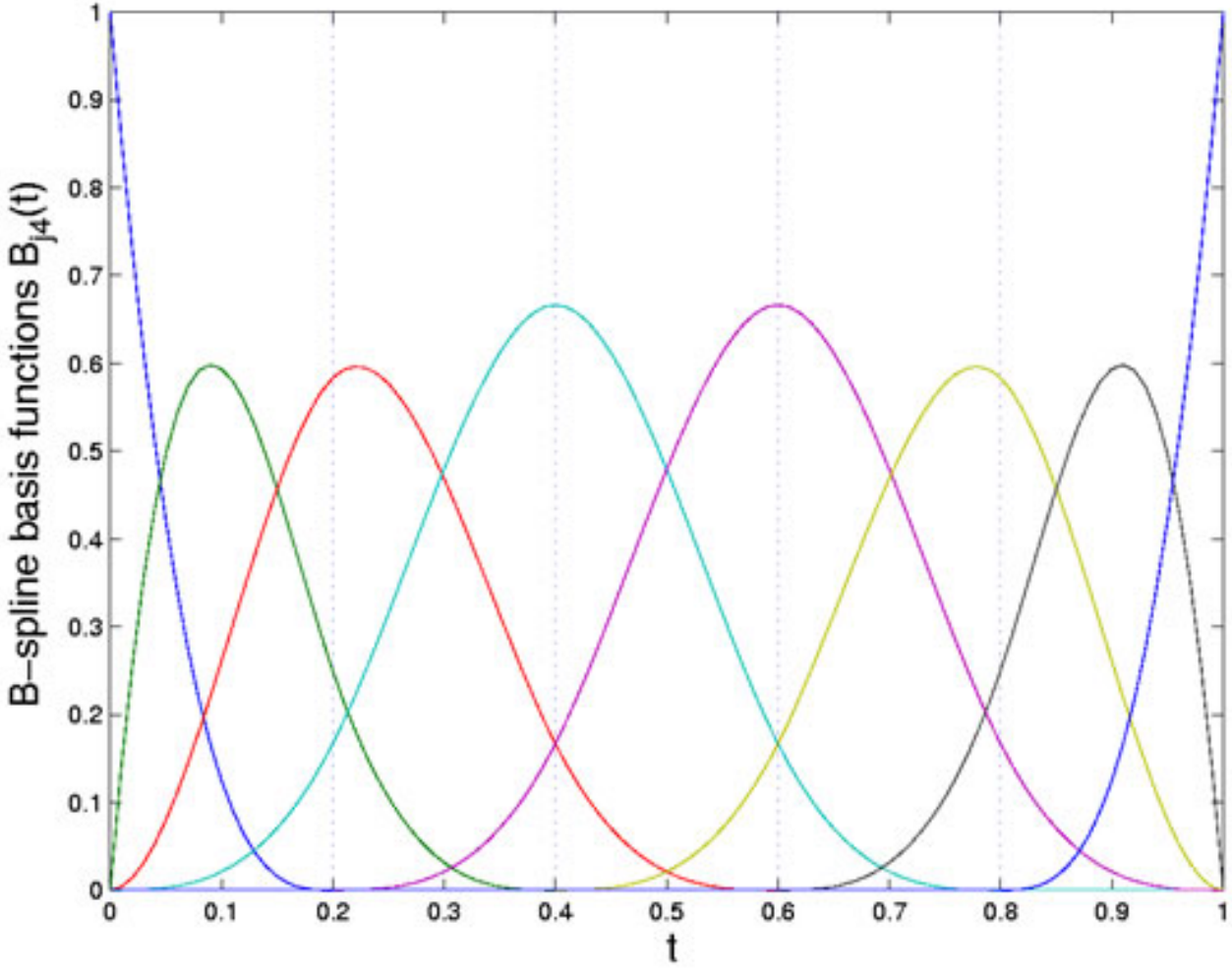
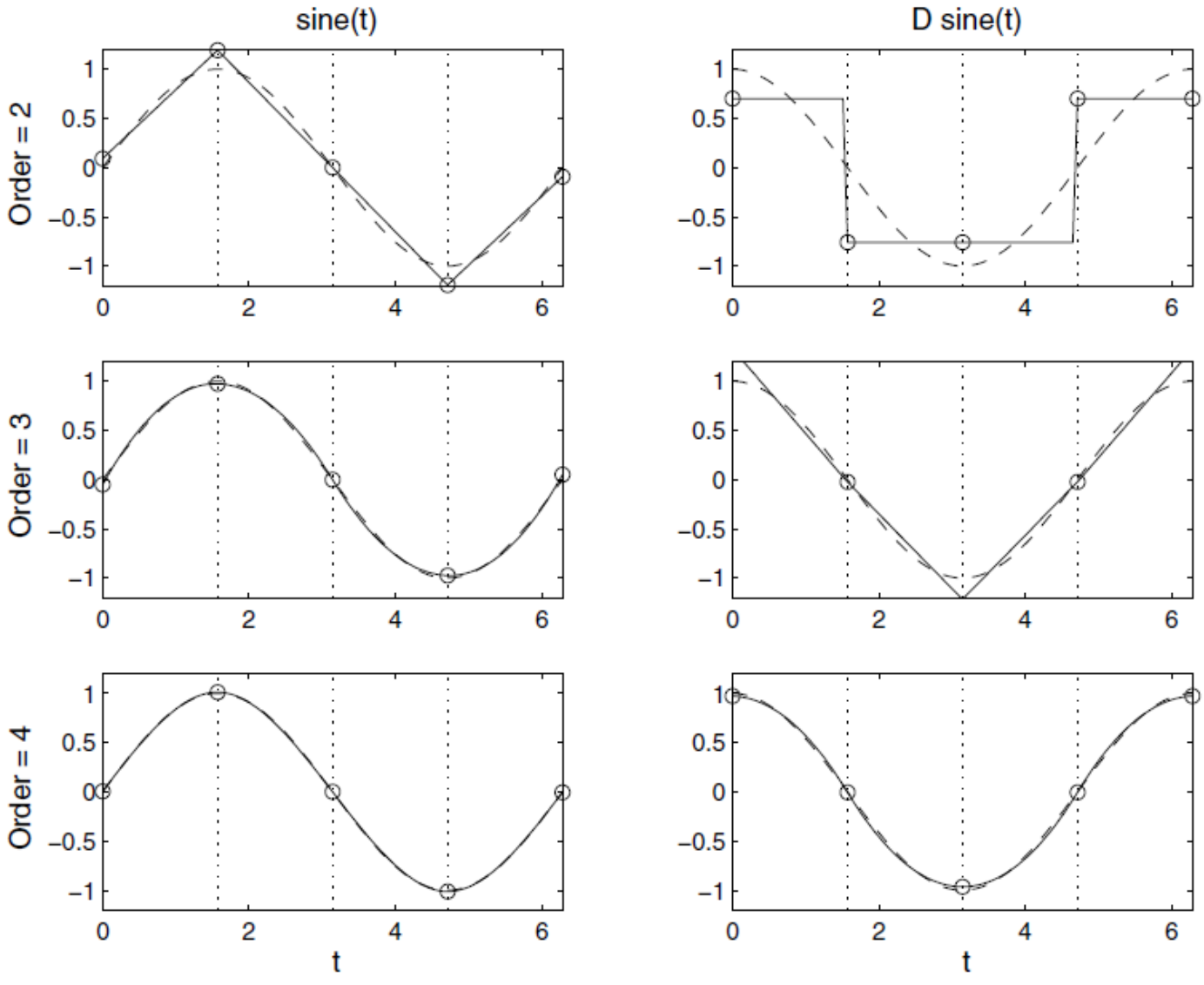
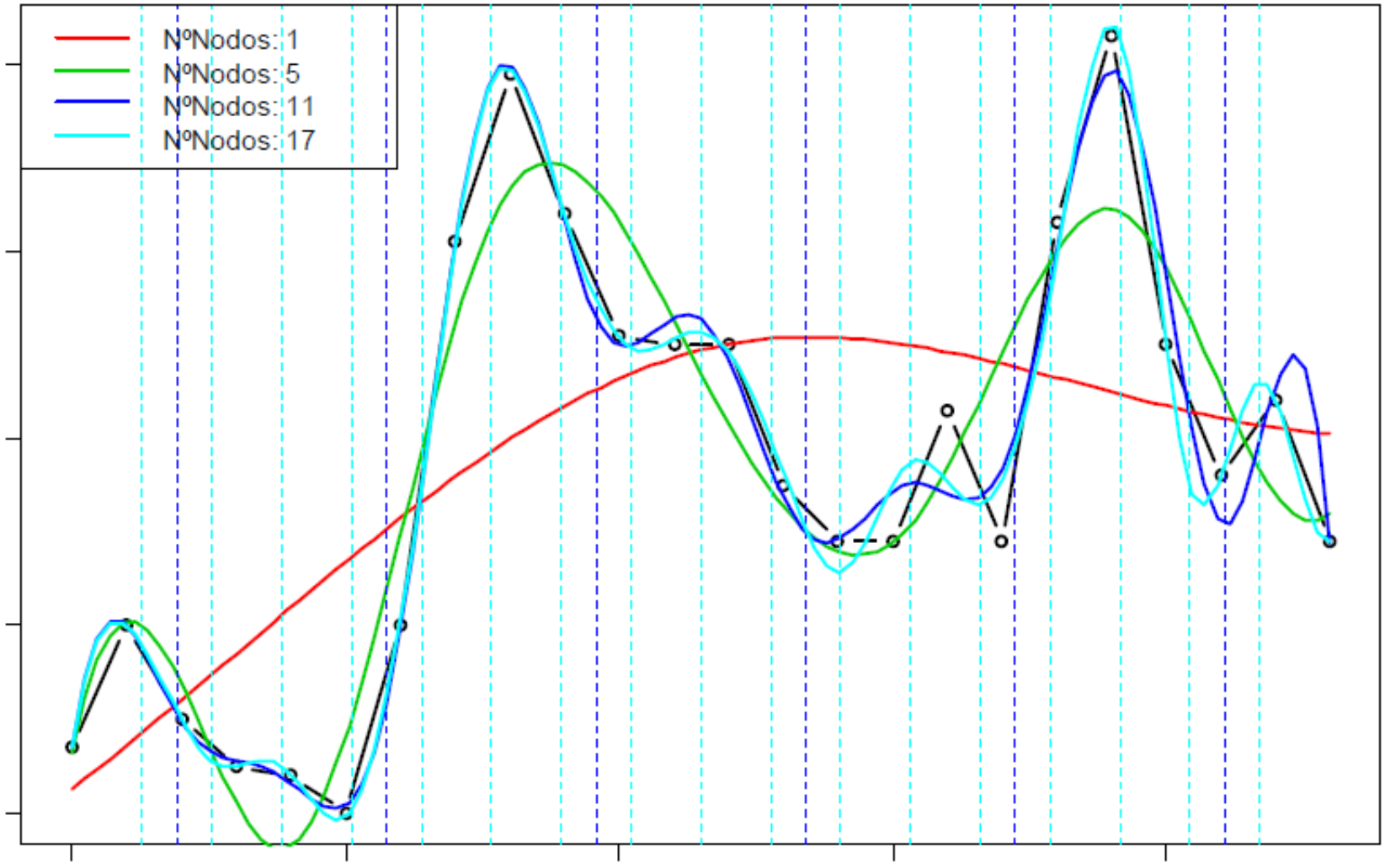


Figure: 8 elements of an order 4 spline basis.

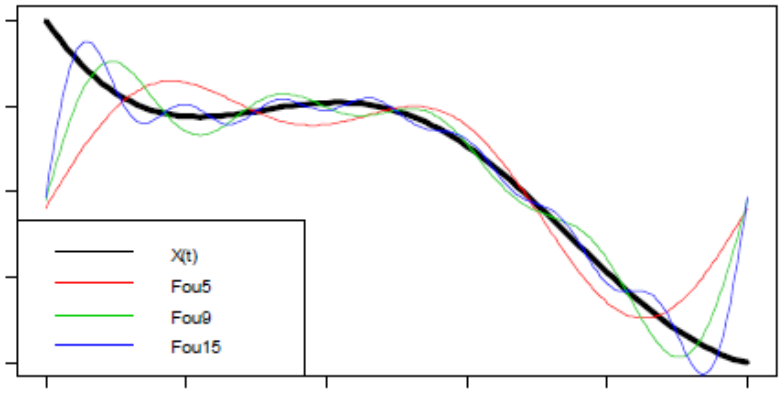
B-Spline basis

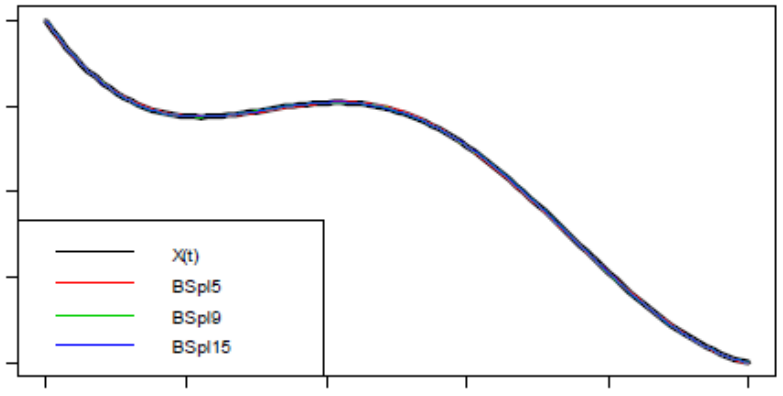


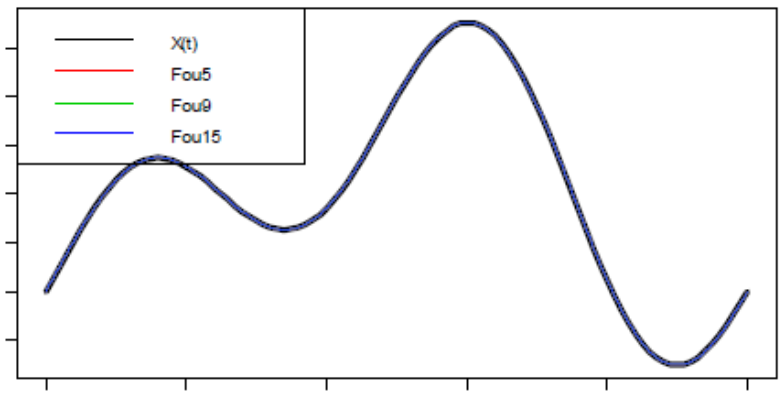
B-Spline basis

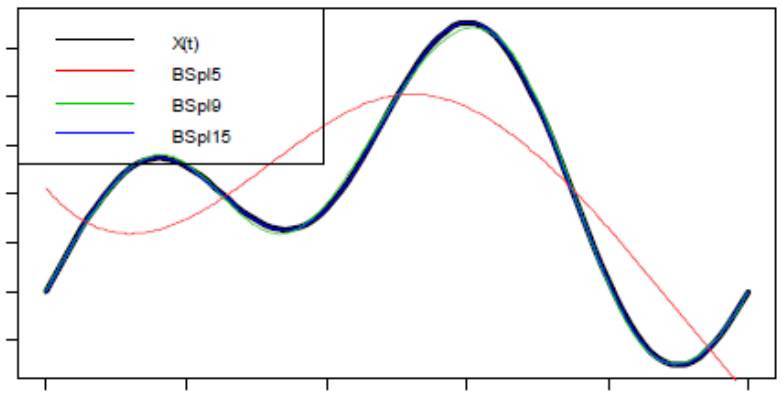


Fourier vs B-splines









The class FDataBasis

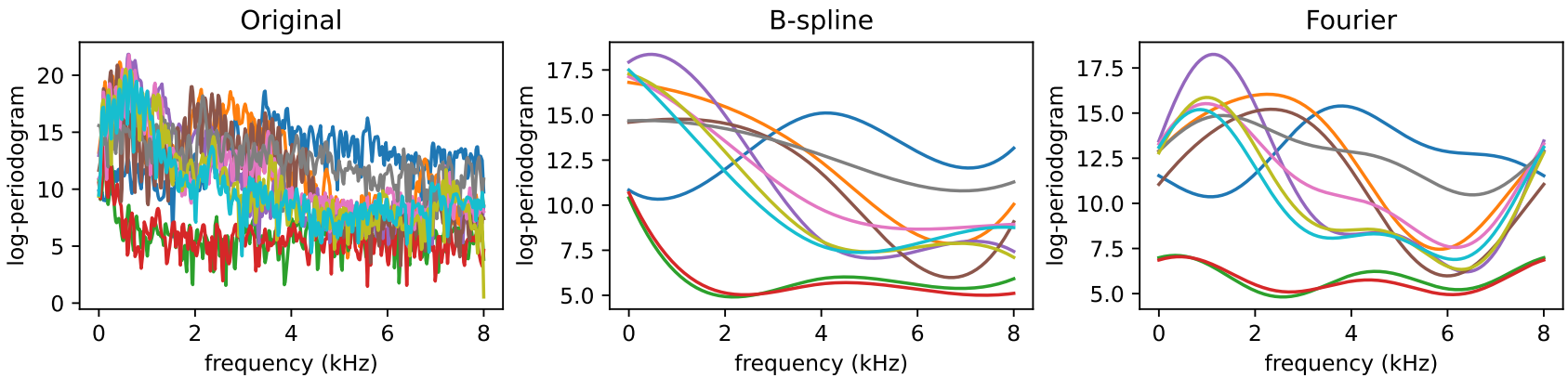
Given a countable basis $\{\phi_i(t)\}_{i\geq 1}$, $x(t) \approx \sum_{i=1}^{K} c_i \phi_i(t)$

- `basis`: The basis for the representation of the functional observations: *Constant, Monomial, BSpline and Fourier.*

  In addition, classes for other types of bases can be implemented by the user.

- `coefficients`: *The $N \times K$ matrix that contains the coefficients.*

```
X.to_basis(BSpline(n_basis=5))
X.to_basis(Fourier(n_basis=5))
```



Wavelets

The wavelets are built from two associated orthogonal functions, the father wavelet ϕ (scale factor) and the mother wavelet ψ. The elements of the basis are obtained from these two functions by translations and scale changes

$$\psi_{jk}(t) = 2^{-j/2}\psi(2^{-j}t - k), \quad \phi_{jk}(t) = 2^{-j/2}\phi(2^{-j}t - k).$$

Once the basis is chosen and $j = J$ fixed,

$$f(t) = \sum_k c_{J,k}\phi_{J,k}(t) + \sum_{j=1}^{J}\sum_k d_{j,k}\psi_{j,k}(t).$$

- Computationally efficient.
- Non-differentiable, in general.
- Suitable or discontinuous or highly irregular functions.

Wavelets

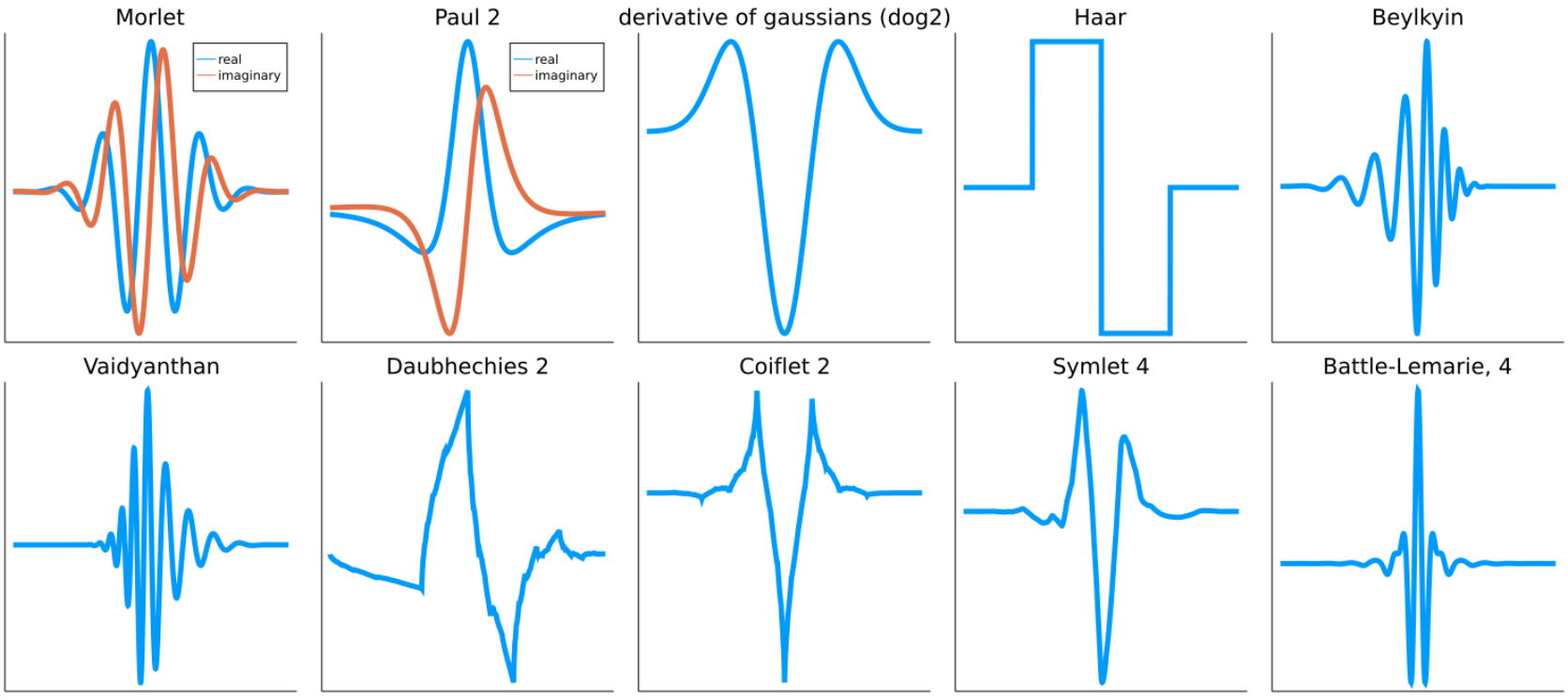
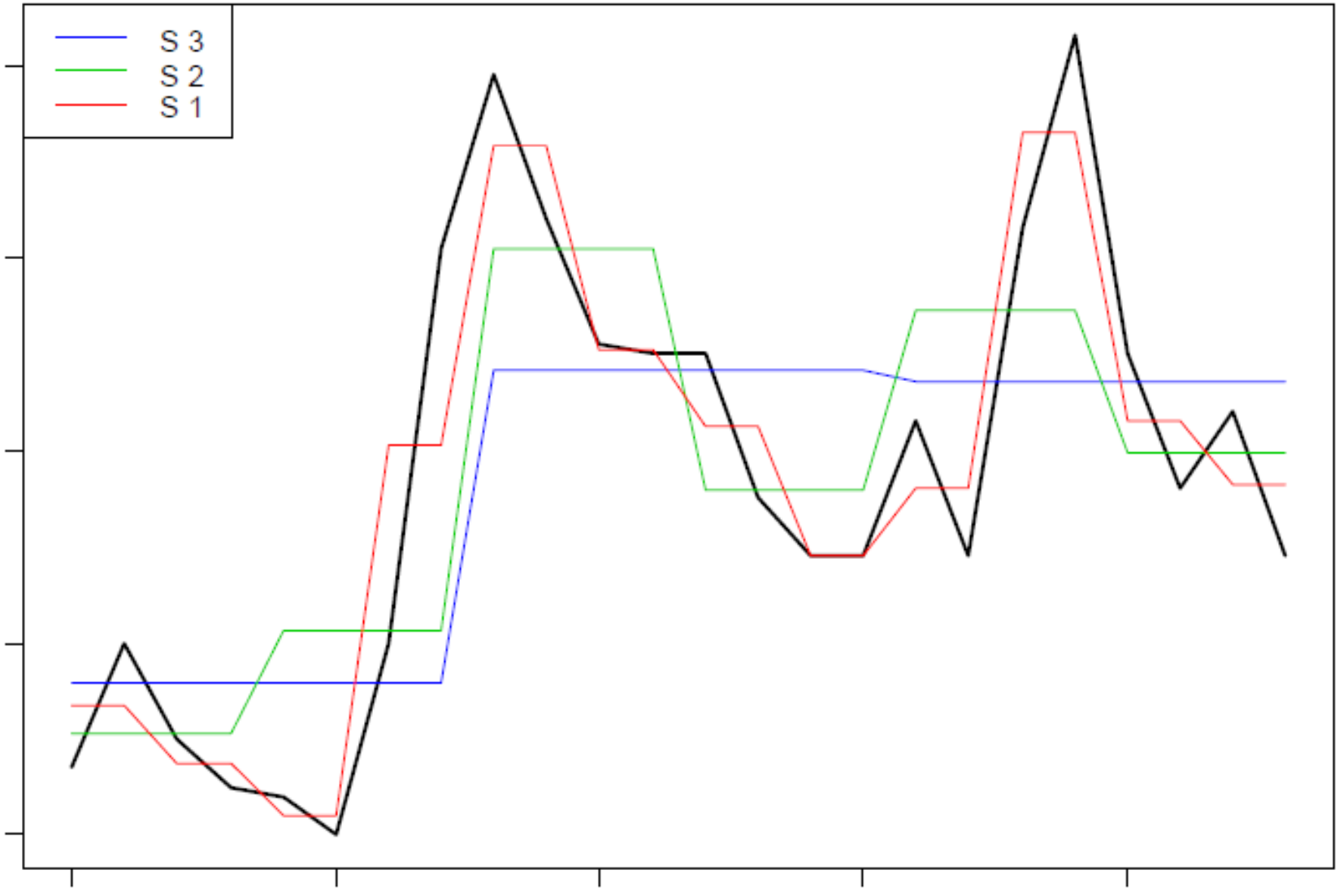
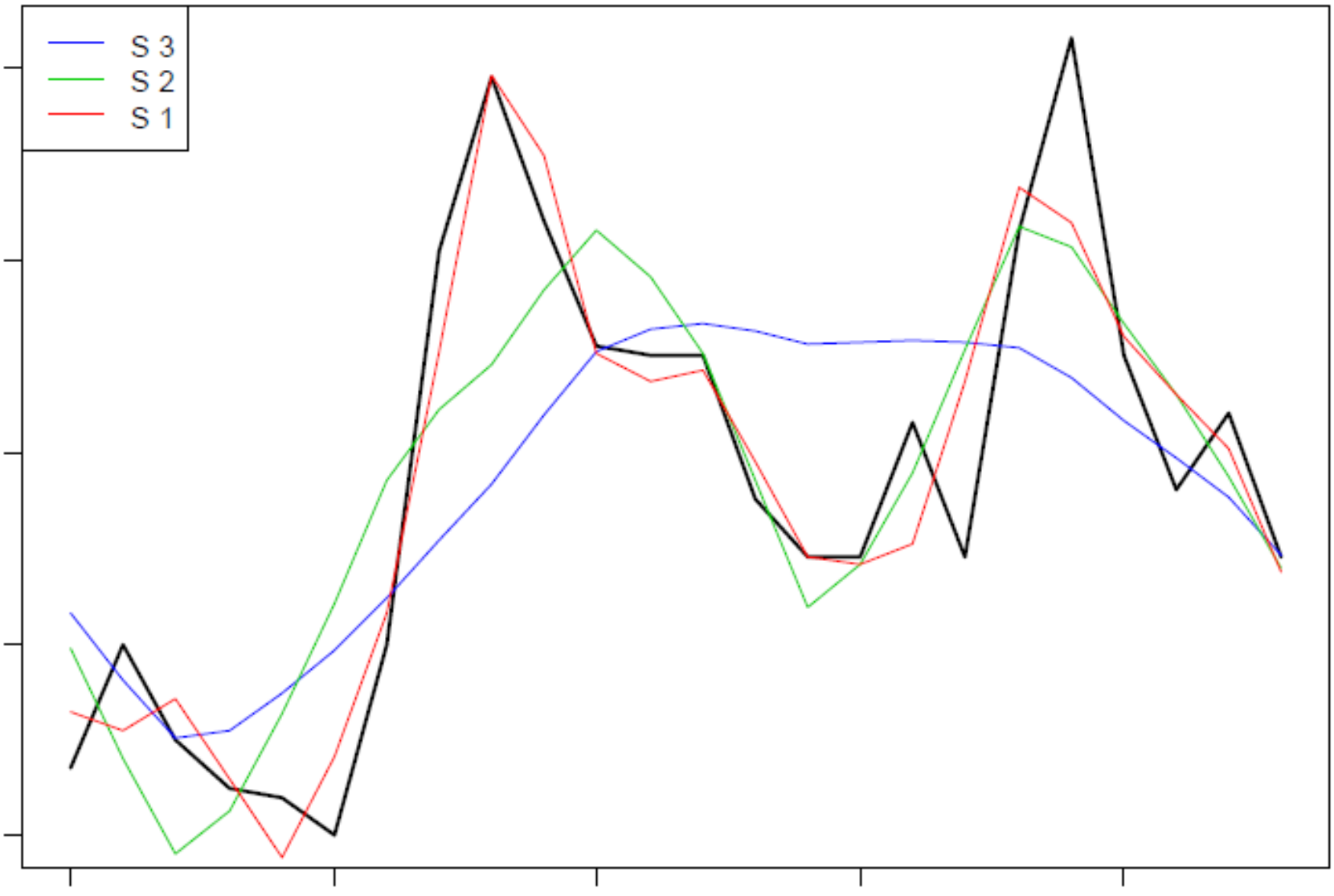


Figure: Different wavelets in *Wavelets.jl*.

Wavelets: the Haar basis



Wavelets: the Daubechies basis



Empirical / Data-driven bases

Those bases that are built from the data trying to optimize a specific objective are generally called empirical or data-driven bases.

- Maximizing Variance: Functional Principal Component analysis (FPCA) Ramsay and Silverman (2005).
- Maximizing covarianze with the response variable: Functional Partial Least Squares (PLS) Delaigle and Hall (2012).

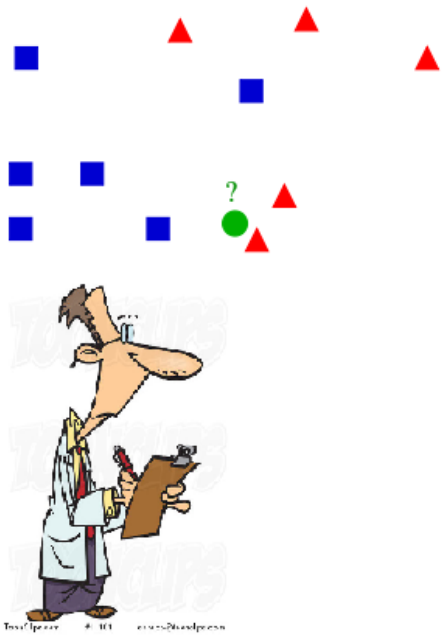
Currently, wavelets and empirical bases are not directly included in *scikit-fda*. However, these bases can be created by the user following the 'Creating a new basis' tutorial.

Learning types

Machines need information to learn

Unsupervised

Supervised



Semi-Supervised

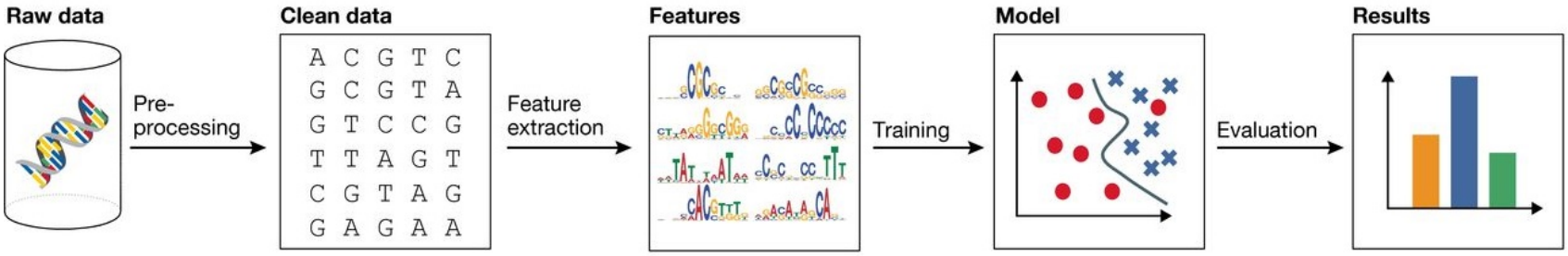
Classify

1.- Arrange (a group) in classes according to shared characteristics.

2.- Assign to a particular class or category.

Oxford English Dictionary

The classification process

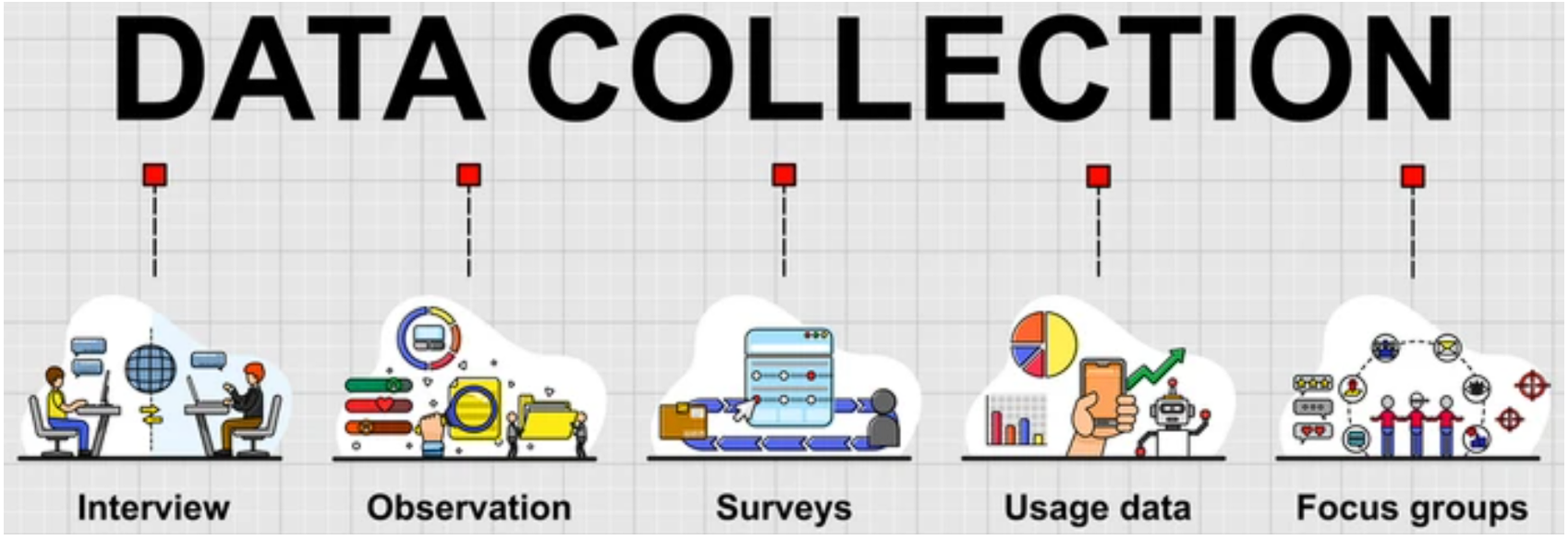


- Problem definition and Gathering data. Sampling, representativity, scope...
- Prepocessing - Data preparation. coding, cleaning, datos, limpieza, missing values, outliers...
- Feature extraction. dimensionality reduction, feature construction...
- Model/Classifier selection. (non-)parametric, (non-)linear, global or local, ensembles...
- Training. overfitting, algorithms, convergence, rates...
- Model evaluation. ¿Which is better?, evaluation metrics, generalization...

1. Representation of functional data
   - Basis representation

2. Data collection

3. Data cleaning and preprocessing

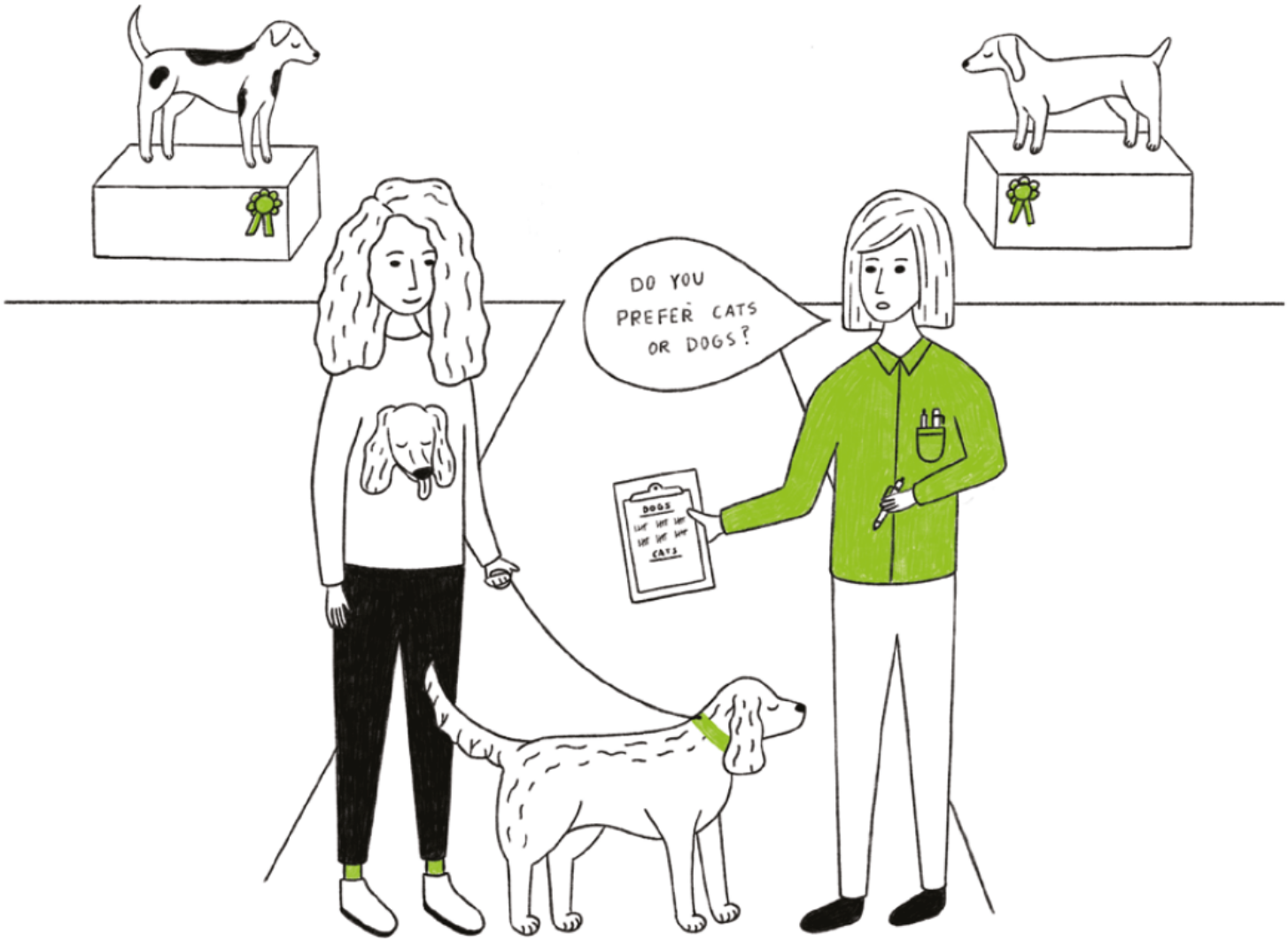


- **Input:** Real world. weather, people, animals...
- **Output:** Raw data. subset of measurements, images, discretized trajectories...

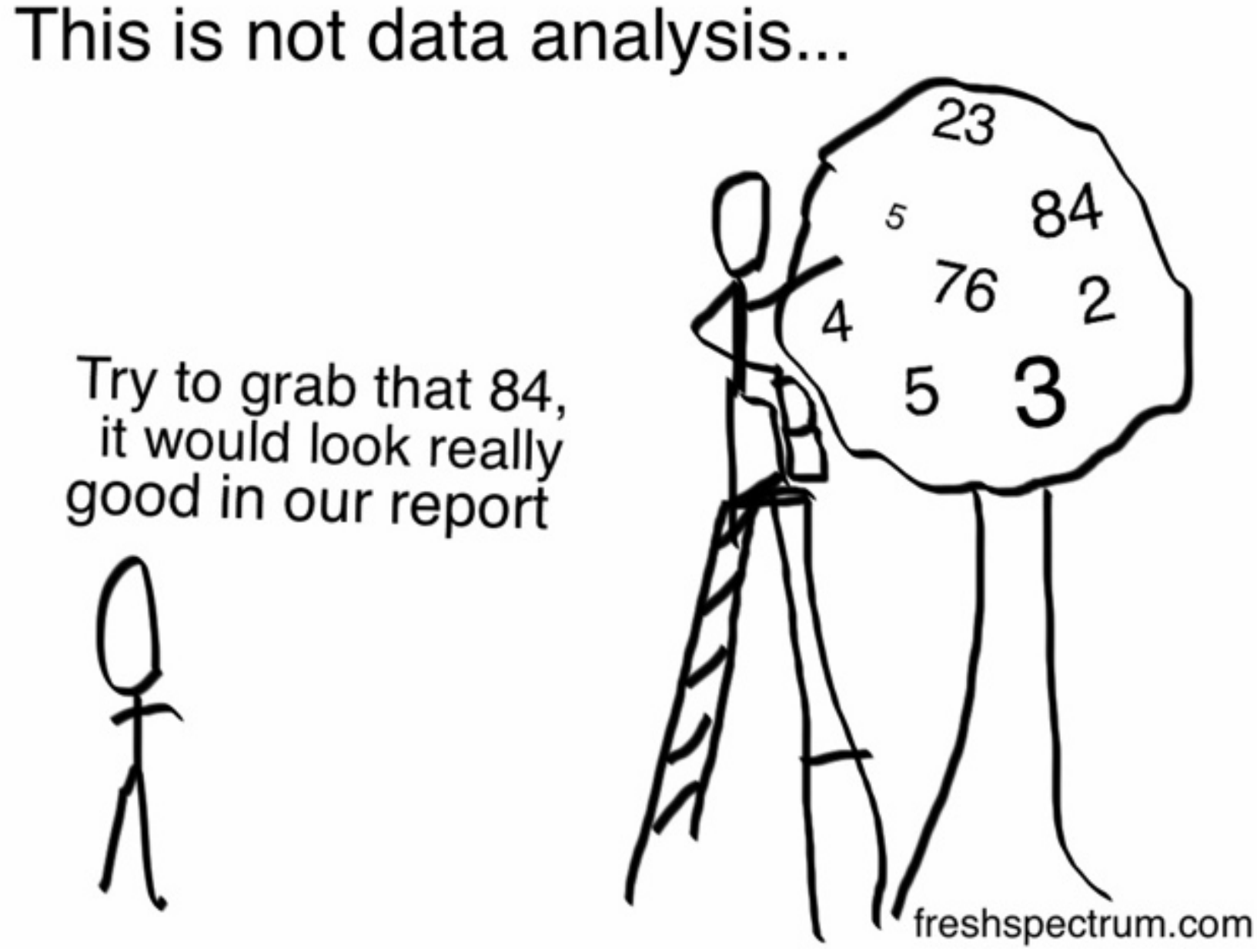
Data collection is a delicate process that often requires large amounts of time and effort.

- How to choose the examples?
- ¿How many examples are enough?
- What variables to measure? What questions to ask?
- What devices to use? How to integrate different sources?
- ...

Be careful!



Be careful!

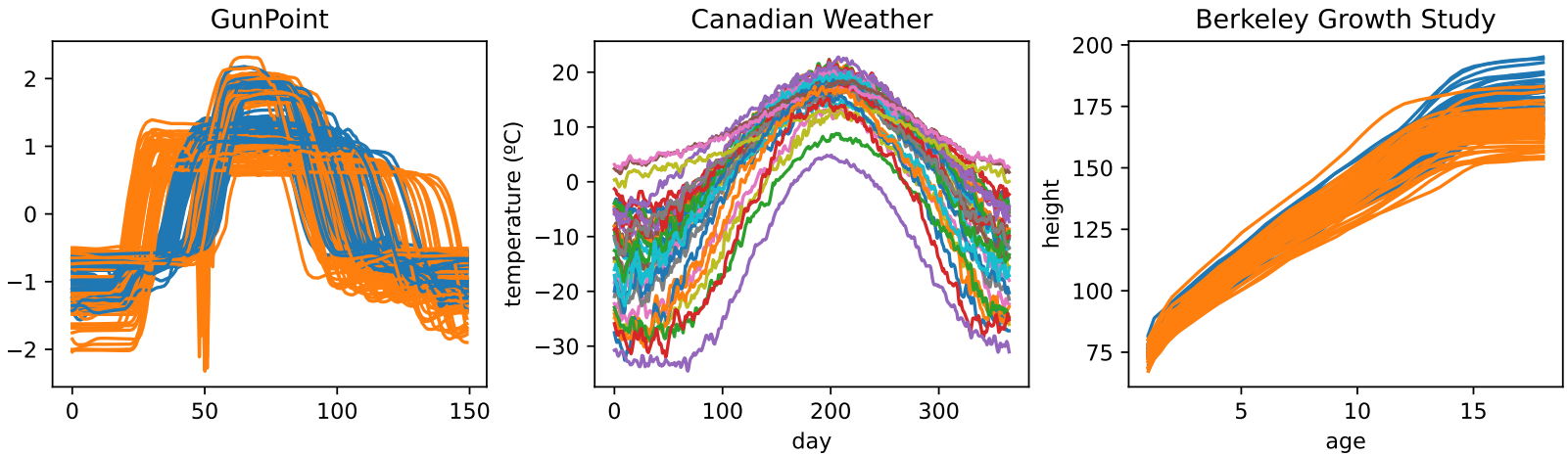

This is not data analysis...
Try to grab that 84, it would look really good in our report
23
5
84
76
2
4
5
3
freshspectrum.com

Loading datasets

The datasets are retrieved using the package *scikit-datasets* (Diaz-Vico and Ramos-Carreño, 2022) and converted to the *FData* format wit the correponding *fetch* function.

- Some datasets commonly used in the literature have their own functions, for example Tecator (*fetch_tecator()*) or the Canadian Weather (*fetch_weather()*)
- `fetch_cran(name, package_name, ...)` allow for retrieving datasets from `R` packages.
- `fetch_ucr(name, ...)` can be used to obtain datasets from the UEA and UCR Time Series Classification Repository

At this moment, *scikit-fda* does not offer input/output functions *CSV*, *EXCEL*, *MATLAB*, or other file formats. The user can parse the grid points and values of the functions to a `NumPy` array and construct the `FDataGrid`.



Functional Variable Ferraty and Vieu (2006)

A *Functional Variable* is a random variable X that takes values in a functional space (of infinite dimension). An observation x of X is called a *Functional Data*.

In particular, we will focus on the case where the data consists of real functions defined on a compact interval.

$$X_1 = X_1(t), \ldots, X_n = X_n(t), \ t \in [0, T].$$

FDA: Formalization II

Functional data can be also seen as realizations of an stochastic process. Hsing and Eubank (2015) Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $\mathcal{I} \subseteq \mathbb{R}$ be an index set. A stochastic process is a collection of random variables $\{X(\omega, t) : \omega \in \Omega, t \in \mathcal{I}\}$ where $X(\cdot, t)$ is $\mathcal{F}$-measurable function on Ω.

A functional data is simply a realization ("trajectory") of a stochastic process for all $t \in \mathcal{I}$.

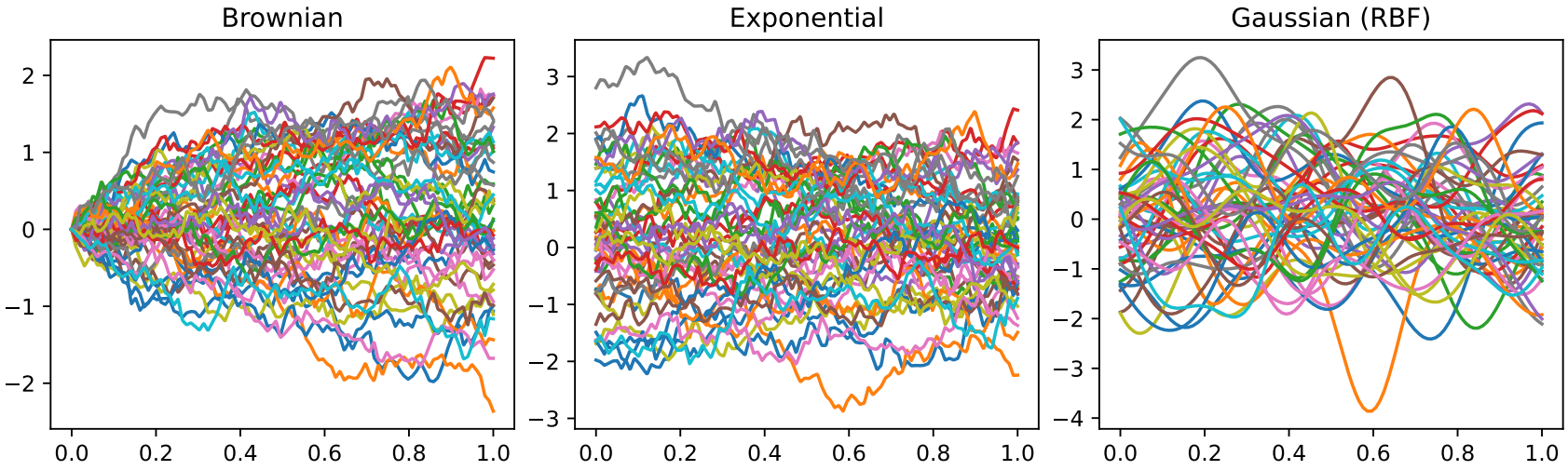
$\underline{\mathbb{R}^n}$		$\underline{\mathcal{F}}$
Random variables	$\rightleftarrows$	Stochastic process
Standard normal	$\rightleftarrows$	Gaussian process

Gaussian Process

A stochastic process is said to be Gaussian if and only if, for any $t_1, \ldots, t_k \in \mathcal{I}$, the k-dimensional random vector $(X(t_1), \ldots, X(t_k))$ follows a normal distribution. It is completely determined by $m(t)$ and $K(s,t)$. Doob (1953)

Standard Brownian Motion

A Gaussian process with stationary and independent increments such that $m(B(t)) = 0$ and $K(s,t) = Cov(B(s), B(t)) = \min(s,t)$. Mörters and Peres (2010)



make_gaussian_process

```
skfda.datasets.make_gaussian_process(n_samples=100, n_features=100, *,
start=0, stop=1, mean=0, cov=None, noise=0, random_state=None)   [source]
```

Generate Gaussian process trajectories.

Parameters:

- **n_samples** (*int*) – The total number of trajectories.
- **n_features** (*int*) – The total number of features (points of evaluation).
- **start** (*float*) – Starting point of the trajectories.
- **stop** (*float*) – Ending point of the trajectories.
- **mean** (*MeanLike*) – The mean function of the process. Can be a callable accepting a vector with the locations, or a vector with length `n_features`.
- **cov** (*CovarianceLike | None*) – The covariance function of the process. Can be a callable accepting two vectors with the locations, or a matrix with size `n_features` x `n_features`. By default, the Brownian covariance function is used.
- **noise** (*float*) – Standard deviation of Gaussian noise added to the data.
- **random_state** (*RandomStateLike*) – Random state.

Returns:

`FDataGrid` object comprising all the trajectories.

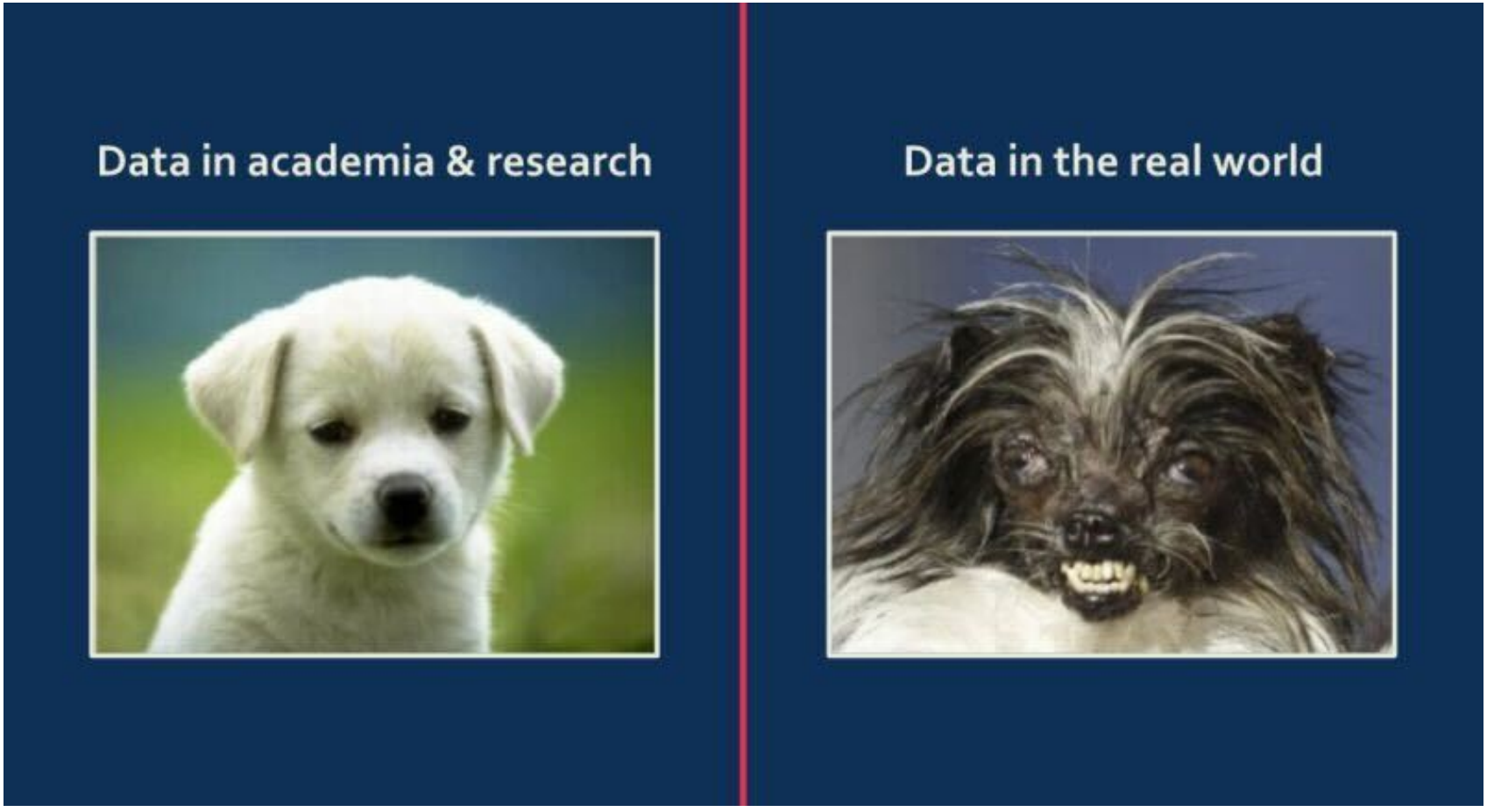
Covariance functions

This module contains several common covariance functions of Gaussian processes. These functions can be used as covariances in `make_gaussian_process()`.

`skfda.misc.covariances.Brownian`(*[, ...])	Brownian covariance function.
`skfda.misc.covariances.Covariance`()	Abstract class for covariance functions.
`skfda.misc.covariances.Exponential`(*[, ...])	Exponential covariance function.
`skfda.misc.covariances.Gaussian`(*[, ...])	Gaussian covariance function.
`skfda.misc.covariances.Linear`(*[, variance, ...])	Linear covariance function.
`skfda.misc.covariances.Polynomial`(*[, ...])	Polynomial covariance function.
`skfda.misc.covariances.Matern`(*[, variance, ...])	Matérn covariance function.
`skfda.misc.covariances.WhiteNoise`(*[, variance])	Gaussian covariance function.

1. Representation of functional data
   - Basis representation
2. Data collection
3. Data cleaning and preprocessing

Notice to users

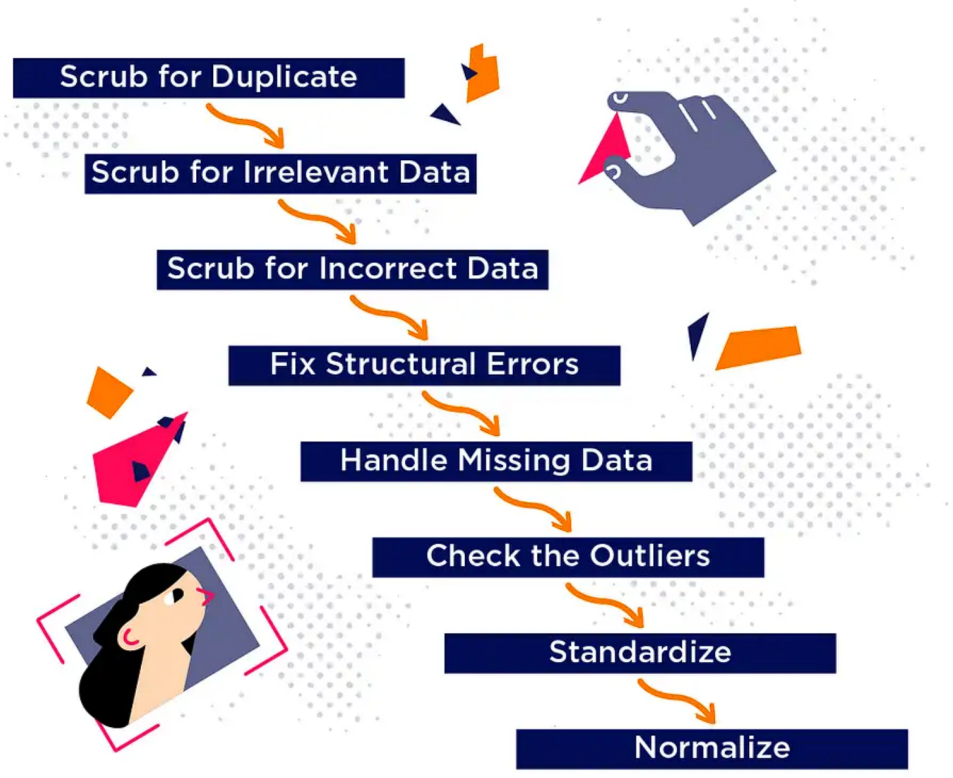


Data cleaning and preprocessing

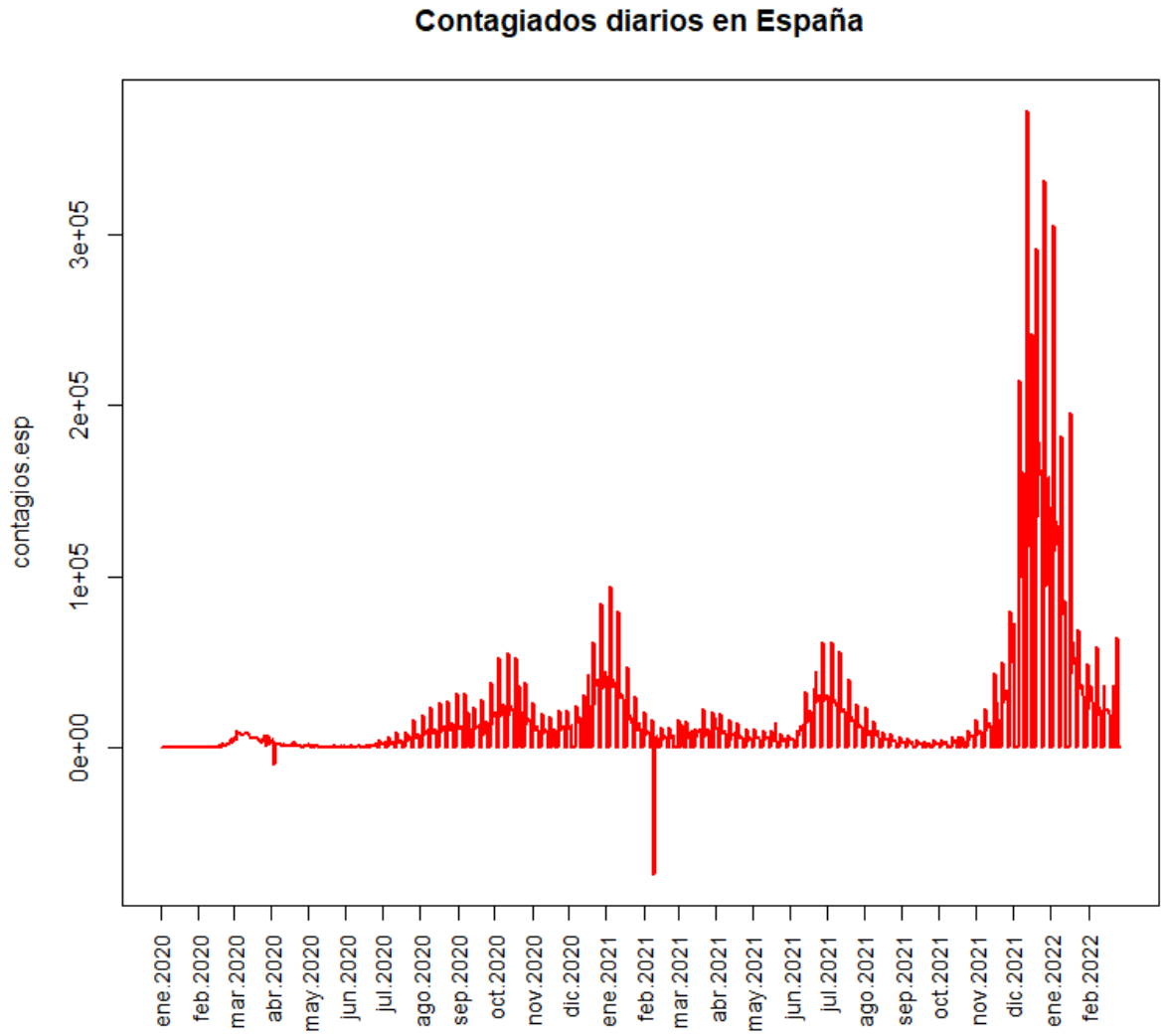
This process can be done both manually or automatically. Several issues related:

- Combination of data of different types from different sources in a coherent way.
- Define the (functional) framework and choose the data representation and the statistical tools.
- Dealing with missing values: removing, replacing, prediction, interpolation...
- Error detection and correction.
- Outlier identification.
- Filter noisy data, smoothing.
- Sparsity.
- Registration.
- Coding.
- Derivatives.
- Segmentation (images).
- ...

Quality of the data and data cleaning

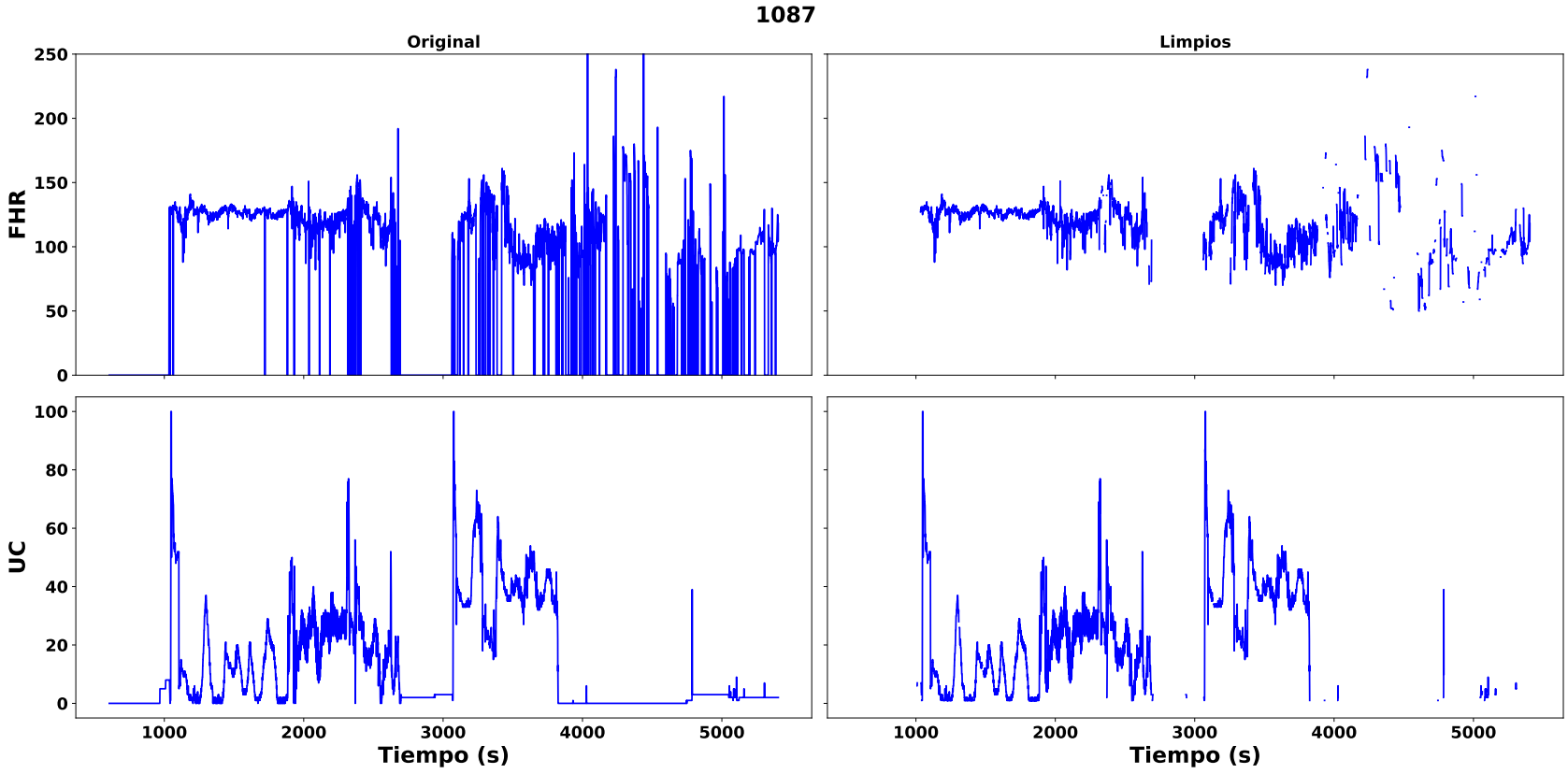


An example: COVID-19



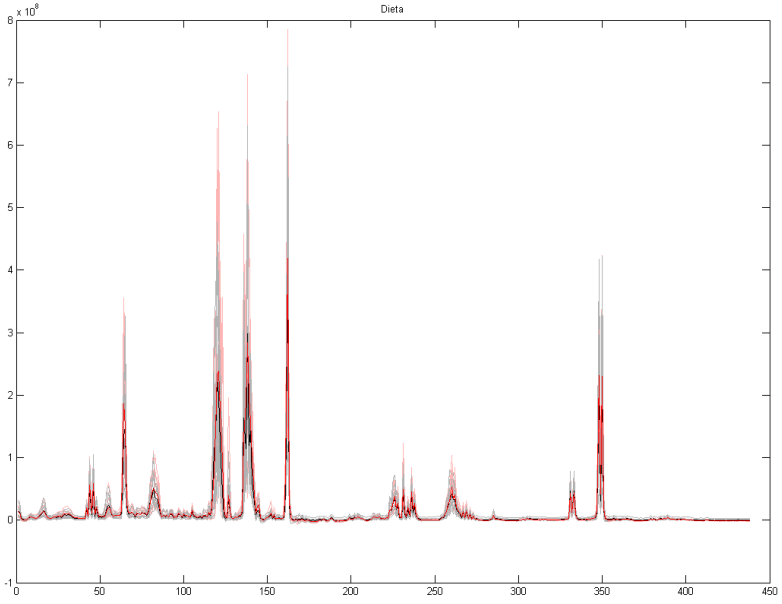
Data collection, errors, meaning of variables...

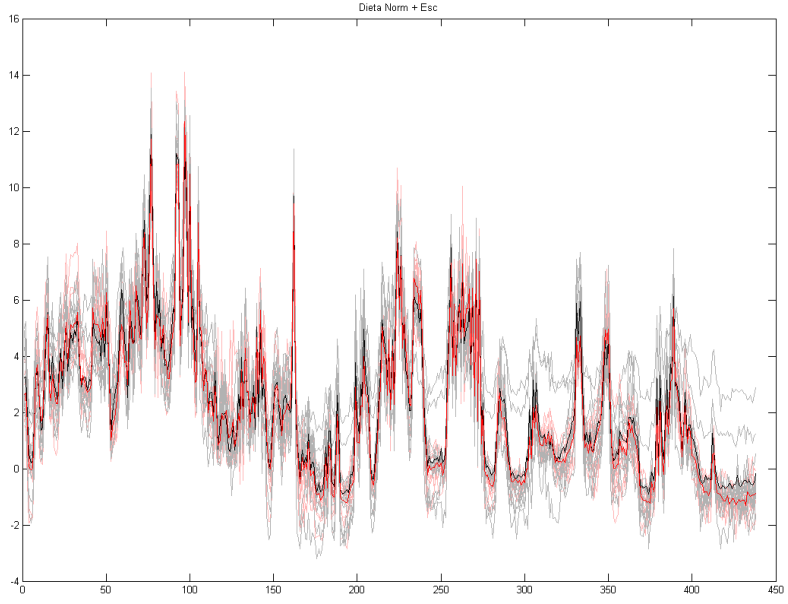
An example: DAHFI



Errors, missing values, atypical values, alignment...

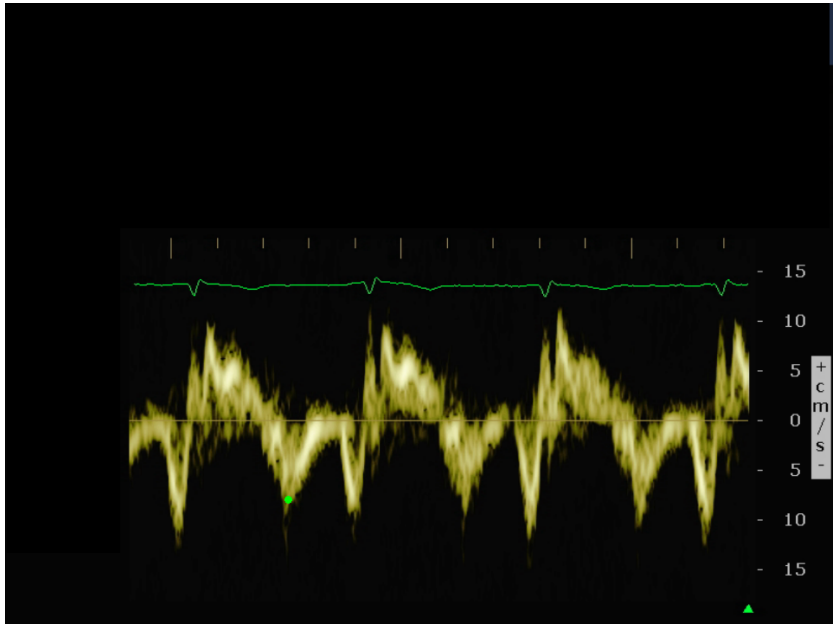
An example: NRM spectral fingerprints

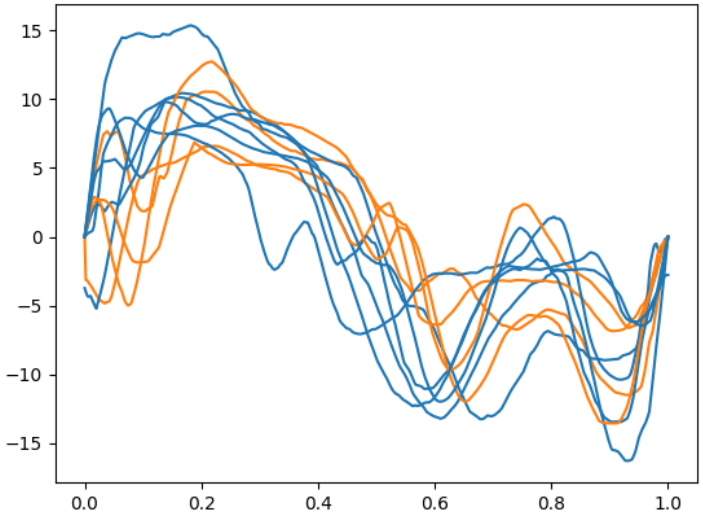




preprocessing, sample size, feature extraction...

An example: Doppler images

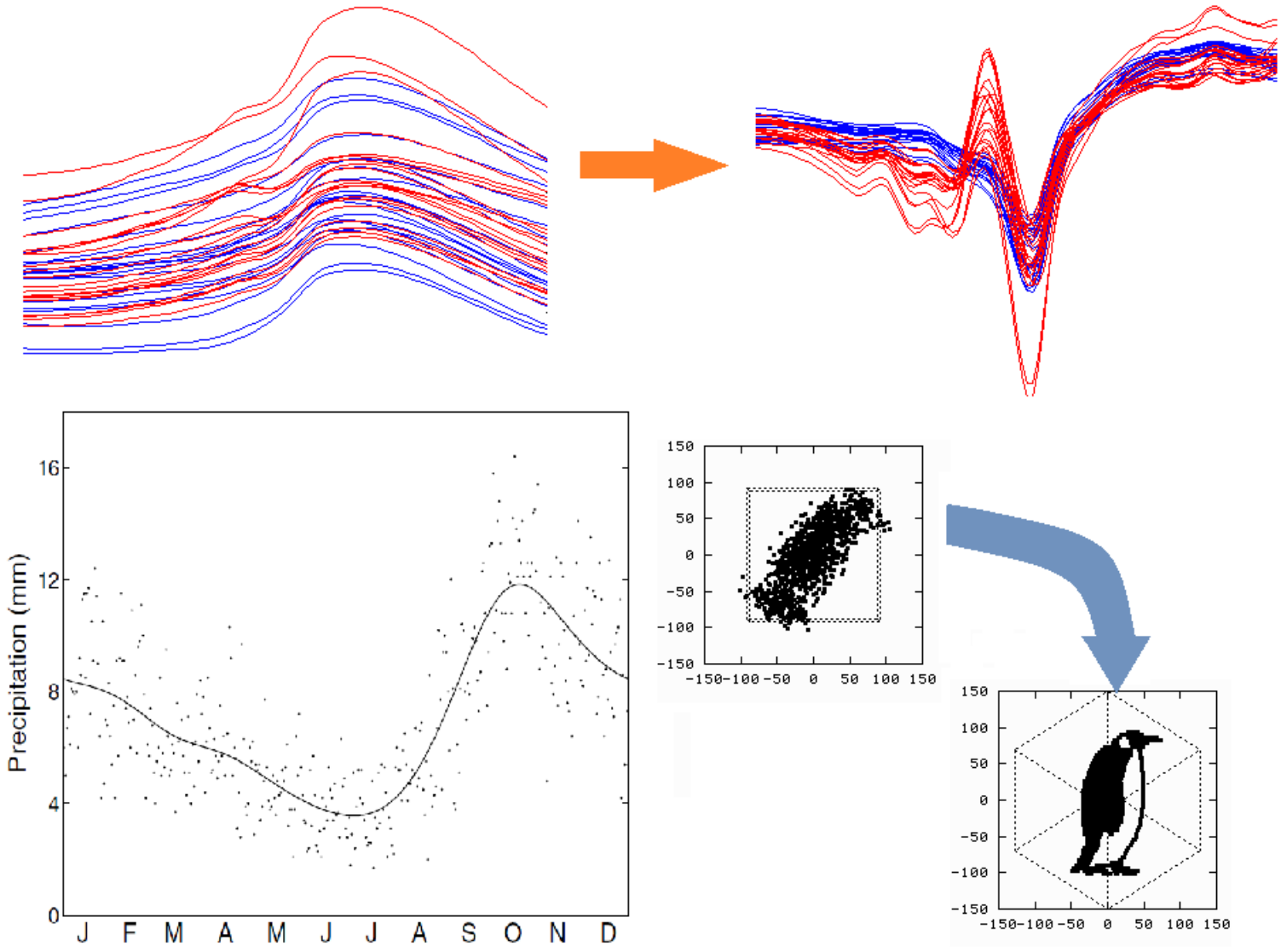




preprocessing - feature extraction, smoothing, registry...

Data available here

Preprocessing and data analysis



Usually, raw data $(x(t_1), \ldots, x(t_d))$ need to be processed before classification.

Some preprocessing tasks

- Representation.
- Filter/recover the signal: smoothing, interpolation... Recover $x(t)$ from $y(t) = x(t) + \epsilon(t)$.
- Solve registration problems.
- Outlier detection.
- Exploratory analysis.
- Feature extraction.
- To choose the functional framework and statistical tools.

"In our lust for measurement, we frequently measure that which we can rather than that which we wish to measure... and forget that there is a difference."

George Udny Yule

Another way to recover the original function is through convolutions with functional kernels. The result is a linear transformation.

$$x = x(t) \mapsto \tilde{x} = \tilde{x}(t) = \sum_{j=1}^{M} S_j(t) x(t_j),$$

where h denotes the bandwith and $S_j(t)$ can be, for example, the Nadaraya-Watson weights

$$S_j(t) = \frac{K\left(\frac{t-t_j}{h}\right)}{\sum_{j=1}^{M} K\left(\frac{t-t_j}{h}\right)} := W_{hM}(t - t_j), \tag{1}$$

associate to kernel K (usually Gaussian). Note that $S_j(t)$ provides in fact an approximation to the functional convolution transformation.Ramsay and Silverman (2005, ch. 4-6)

Kernel smoothing in skfda

class skfda.preprocessing.smoothing.**KernelSmoother**(*kernel_estimator=None*, ***, *weights=None*, *output_points=None*) [source]

Kernel smoothing method.

This module allows to perform functional data smoothing.

Let $t = (t_1, t_2, \ldots, t_n)$ be the points of discretisation and X the vector of observations at that points. Then, the smoothed values, $\hat{X}$, at the points $t' = (t'_1, t'_2, \ldots, t'_m)$ are obtained as

$$\hat{X} = \hat{H}X$$

where $\hat{H}$ is a matrix described in `HatMatrix`.

See also

`NadarayaWatsonHatMatrix` `LocalLinearRegressionHatMatrix` `KNeighborsHatMatrix`

Bandwith selection

class

skfda.preprocessing.smoothing.validation.LinearSmootherGeneralizedCVScorer(*penalization_function=None*)

[source]

Generalized cross validation scoring method for linear smoothers.

It calculates the general cross validation score for every sample in a FDataGrid object given a smoothing matrix $\mathbf{S}(h)$ calculated with a parameter h:

$$GCV(h) = \Xi(\mathbf{S}(h)) \frac{1}{M} \sum_{m=1}^{M} \left(x(t_m) - \hat{x}(t_m; h)\right)^2,$$

Where $\hat{x}(t_m; h)$ is the adjusted $x(t_m)$ and Ξ is a penalization function. By default the penalization function is:

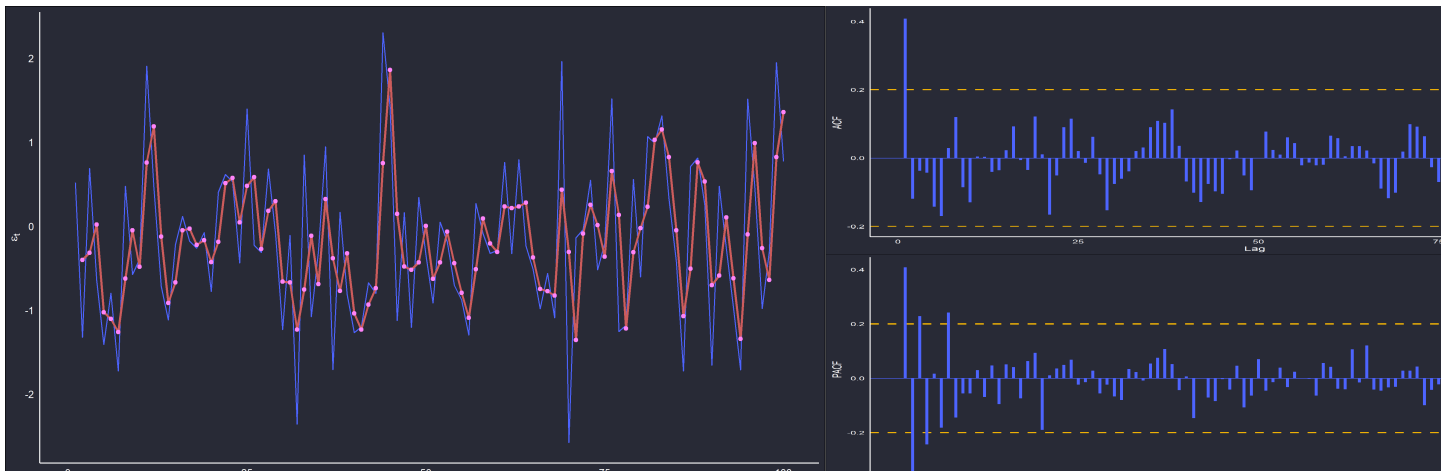
$$\Xi(\mathbf{S}(h)) = \frac{1}{(1 - \mathrm{tr}(\mathbf{S}(h))/M)^2}.$$

but others such as the Akaike's information criterion can be considered.

At the end of the daty... What is better?

It is a debated question. As usual, it depends on the problem and the data at hand:

- Smoothing is useful with noisy data or to take derivatives.
- In classification it is not clear that smoothing is an advantage, Carrol et al. (2013) show that the usual smoothing parameter does not work well in this context recommend undersmoothing.
- Slutzky-Yule effect (artificial correlation structure).



[1] https://www.patsprojects.org/post/slutzkyyule/the-slutzky-yule-effect-and-the-risks-of-smoothing-data/

The notion of closeness

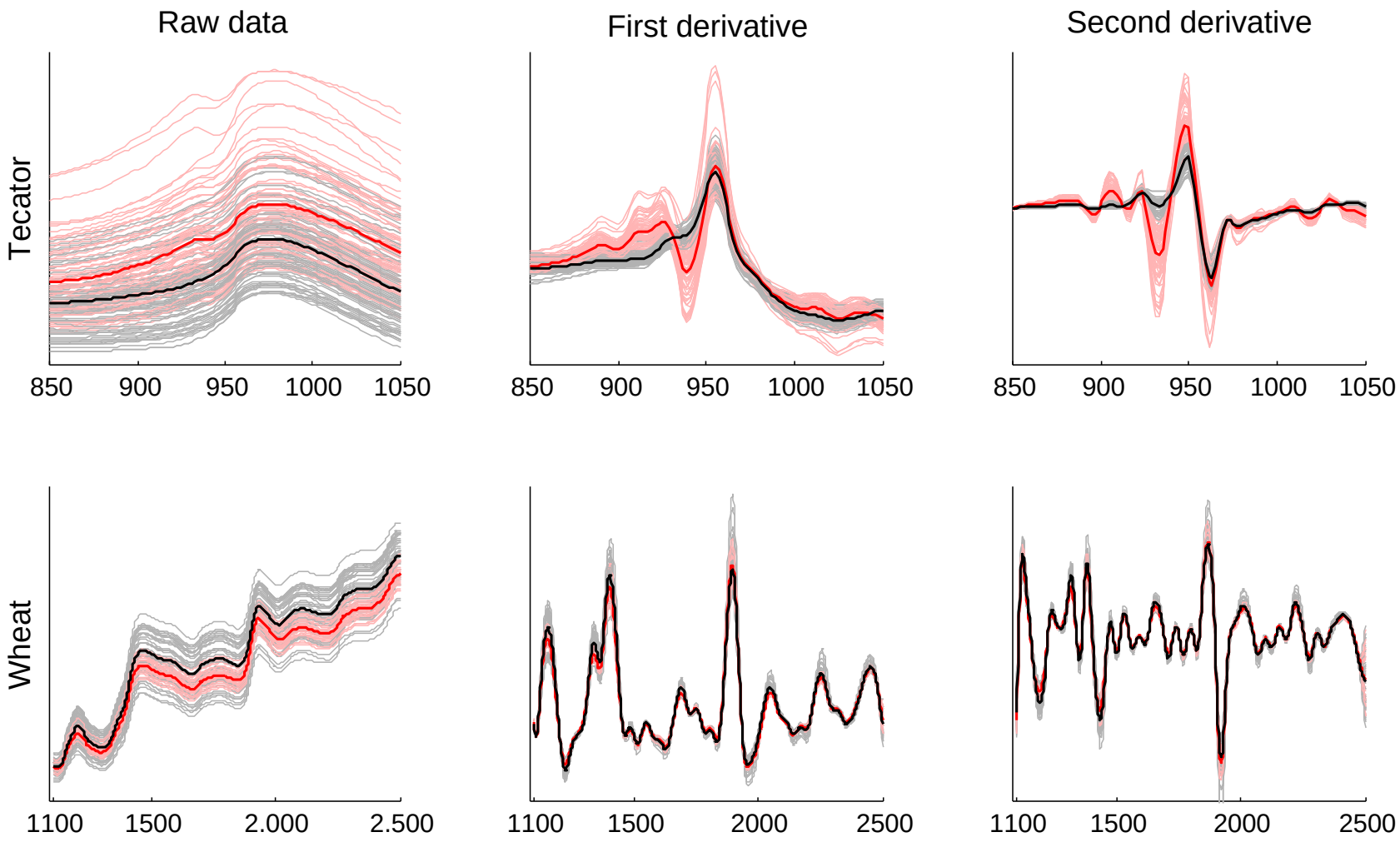
The closeness between two data is determined by the way of measuring the distance. The usefulness of a metric depends on where the relevant information for the problem is located. The usual metrics may be insufficient.

Definition: Let $\mathcal{F}$ be a functional space, it is a semi-metric space if exist a semi-metic $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ such that:

1. $\forall f \in \mathcal{F},\ d(f, f) = 0$ $(d(f, g) = 0 \nRightarrow f = g)$.
2. $\forall f, g \in \mathcal{F},\ d(f, g) = d(g, f)$
3. $\forall f, g, h \in \mathcal{F},\ d(f, g) \leq d(f, h) + d(h, g)$

Some examples of semi-metrics can be found in Ferraty and Vieu (2006).

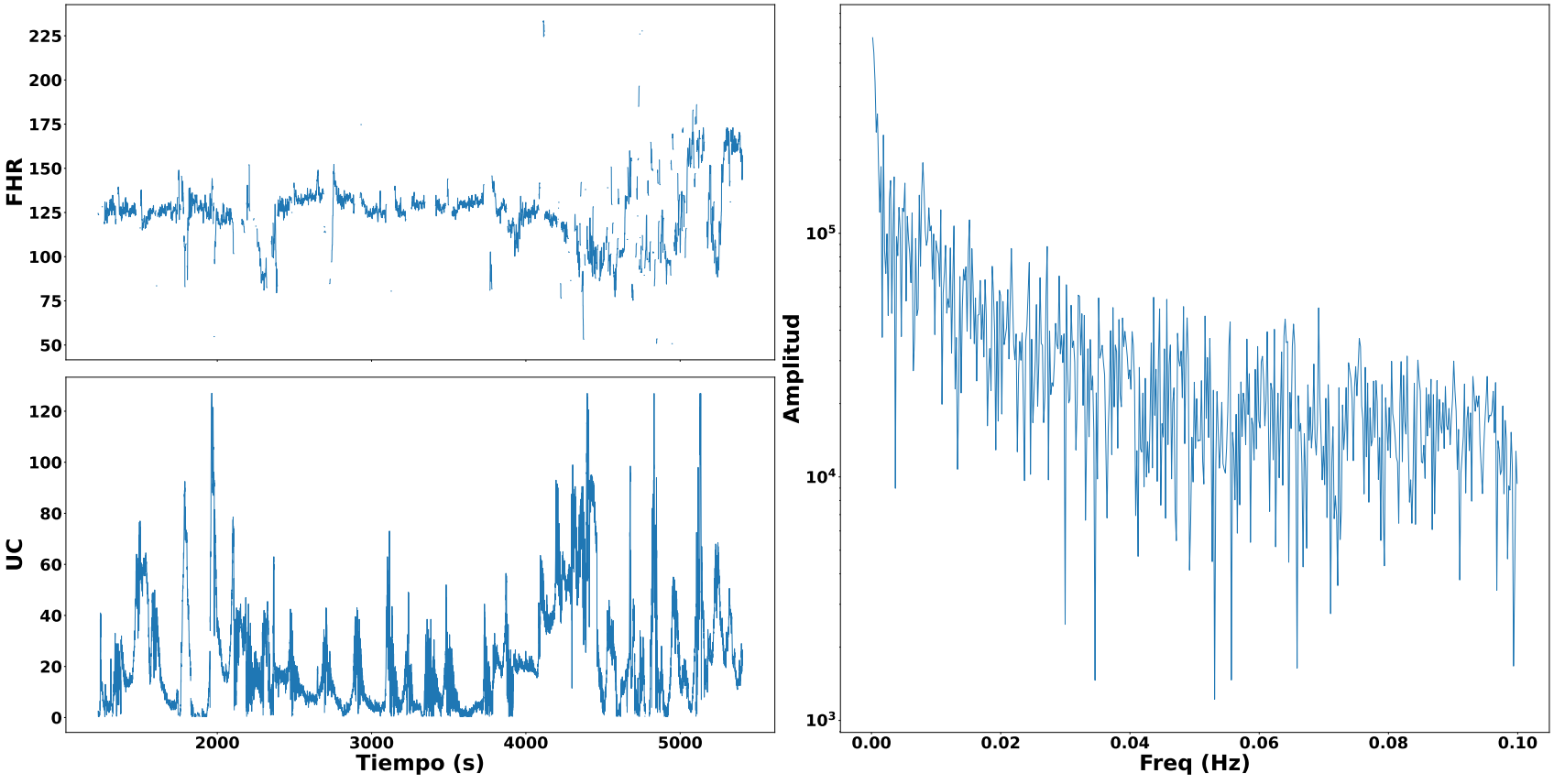
Other transformations



Derivatives and skfda

- Can reveal significant information that is not apparent in the original curves.
- Regularization: the norm of a derivative is a natural measure of the function's roughness Ramsay and Silverman 2005
- For *FDataGrid* objects, derivatives are approximated using finite differences.
- For FDataBasis objects, derivatives are computed exactly, by using the derivatives of the basis functions. Thus, if a new type of basis is designed, it is necessary to implement the derivatives of the basis functions in the corresponding class.

Fast Fourier Transform



‘Autocorrelations’