fdasrsf Documentation

Release 1.1.0

J. Derek Tucker

September 30, 2013

CONTENTS

A python package for functional data analysis using the square root slope framework which performs pair-wise and group-wise alignment as well as modeling using functional component analysis.

CHAPTER

ONE

FUNCTIONAL ALIGNMENT

Group-wise function alignment using SRSF framework and Dynamic Programming

moduleauthor:: Derek Tucker <dtucker@stat.fsu.edu>

time_warping.**align_fPCA**(*f*, *time*, *num_comp=3*, *showplot=True*, *smoothdata=False*)

aligns a collection of functions while extracting principal components. The functions are aligned to the principal components

Parameters

- **f** (*np.ndarray*) – numpy ndarray of shape (M,N) of M functions with N samples
- **time** (*np.ndarray*) – vector of size N describing the sample points
- **num_comp** – number of fPCA components
- **showplot** – Shows plots of results using matplotlib (default = T)
- **smooth_data** (*bool*) – Smooth the data using a box filter (default = F)
- **sparam** (*double*) – Number of times to run box filter (default = 25)

Return type tuple of numpy array

Return fn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return qn aligned srvfs - similar structure to fn

Return q0 original srvf - similar structure to fn

Return mqn srvf mean or median - vector of length N

Return gam warping functions - similar structure to fn

Return q_pca srsf principal directions

Return f_pca functional principal directions

Return latent latent values

Return coef coefficients

Return U eigenvectors

time_warping.**align_fPLS**(*f*, *g*, *time*, *comps=3*, *showplot=True*, *smoothdata=False*, *max_itr=100*)

This function aligns a collection of functions while performing principal least squares

Parameters

- **f** (*np.ndarray*) – numpy ndarray of shape (M,N) of M functions with N samples
- **g** (*np.ndarray*) – numpy ndarray of shape (M,N) of M functions with N samples

- **time** (*np.ndarray*) – vector of size N describing the sample points
- **comps** – number of fPLS components
- **showplot** – Shows plots of results using matplotlib (default = T)
- **smooth_data** (*bool*) – Smooth the data using a box filter (default = F)

Return type tuple of numpy array

Return fn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return gn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return qfn aligned srvfs - similar structure to fn

Return qgn aligned srvfs - similar structure to fn

Return qf0 original srvf - similar structure to fn

Return qg0 original srvf - similar structure to fn

Return gam warping functions - similar structure to fn

Return wqf srsf principal weight functions

Return wqg srsf principal weight functions

Return wf srsf principal weight functions

Return wg srsf principal weight functions

Return cost cost function value

time_warping.**srsf_align**(*f*, *time*, *method='mean'*, *showplot=True*, *smoothdata=False*, *lam=0.0*)

This function aligns a collection of functions using the elastic square-root slope (srsf) framework.

Parameters

- **f** (*np.ndarray*) – numpy ndarray of shape (M,N) of M functions with N samples
- **time** (*np.ndarray*) – vector of size N describing the sample points
- **method** – (string) warp calculate Karcher Mean or Median (options = "mean" or "median") (default="mean")
- **showplot** – Shows plots of results using matplotlib (default = T)
- **smoothdata** (*bool*) – Smooth the data using a box filter (default = F)
- **lam** (*double*) – controls the elasticity (default = 0)

Return type tuple of numpy array

Return fn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return qn aligned srvfs - similar structure to fn

Return q0 original srvf - similar structure to fn

Return fmean function mean or median - vector of length N

Return mqn srvf mean or median - vector of length N

Return gam warping functions - similar structure to fn

Return orig_var Original Variance of Functions

Return amp_var Amplitude Variance

Return phase_var Phase Variance

Examples >>> import tables >>> fun=tables.open_file("../Data/simu_data.h5") >>> f = fun.root.f[:] >>> f = f.transpose() >>> time = fun.root.time[:] >>> out = srsf_align(f,time)

time_warping.**srsf_align_pair**(*f*, *g*, *time*, *method='mean'*, *showplot=True*, *smoothdata=False*, *lam=0.0*)

This function aligns a collection of functions using the elastic square-root slope (srsf) framework.

Parameters

- **f** (*np.ndarray*) – numpy ndarray of shape (M,N) of M functions with N samples
- **g** – numpy ndarray of shape (M,N) of M functions with N samples
- **time** (*np.ndarray*) – vector of size N describing the sample points
- **method** – (string) warp calculate Karcher Mean or Median (options = "mean" or "median") (default="mean")
- **showplot** – Shows plots of results using matplotlib (default = T)
- **smoothdata** (*bool*) – Smooth the data using a box filter (default = F)
- **lam** (*double*) – controls the elasticity (default = 0)

Return type tuple of numpy array

Return fn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return gn aligned functions - numpy ndarray of shape (M,N) of M functions with N samples

Return qfn aligned srvfs - similar structure to fn

Return qgn aligned srvfs - similar structure to fn

Return qf0 original srvf - similar structure to fn

Return qg0 original srvf - similar structure to fn

Return fmean f function mean or median - vector of length N

Return gmean g function mean or median - vector of length N

Return mqfn srvf mean or median - vector of length N

Return mqgn srvf mean or median - vector of length N

Return gam warping functions - similar structure to fn

CHAPTER

TWO

FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS

Vertical and Horizontal Functional Principal Component Analysis using SRSF

moduleauthor:: Derek Tucker <dtucker@stat.fsu.edu>

fPCA.**horizfPCA**(*gam*, *time*, *no*, *showplot=True*)

This function calculates horizontal functional principal component analysis on aligned data

Parameters

- **gam** – numpy ndarray of shape (M,N) of M warping functions
- **time** – vector of size N describing the sample points
- **no** (*int*) – number of components to extract (default = 1)
- **showplot** (*bool*) – Shows plots of results using matplotlib (default = T)

Return type tuple of numpy ndarray

Return q_pca srsf principal directions

Return f_pca functional principal directions

Return latent latent values

Return coef coefficients

Return U eigenvectors

fPCA.**vertfPCA**(*fn*, *time*, *qn*, *no=1*, *showplot=True*)

This function calculates vertical functional principal component analysis on aligned data

Parameters

- **fn** – numpy ndarray of shape (M,N) of M aligned functions with N samples
- **time** – vector of size N describing the sample points
- **qn** – numpy ndarray of shape (M,N) of M aligned SRSF with N samples
- **no** (*int*) – number of components to extract (default = 1)
- **showplot** (*bool*) – Shows plots of results using matplotlib (default = T)

Return type tuple of numpy ndarray

Return q_pca srsf principal directions

Return f_pca functional principal directions

Return latent latent values

Return coef coefficients

Return U eigenvectors

CHAPTER

THREE

FUNCTIONAL PRINCIPAL LEAST SQUARES

Partial Least Squares using SVD

moduleauthor:: Derek Tucker <dtucker@stat.fsu.edu>

fPLS.**pls_svd**(*time*, *qf*, *qg*, *no*, *alpha=0.0*)

This function computes the partial least squares using SVD

Parameters

- **time** – vector describing time samples
- **qf** – numpy ndarray of shape (M,N) of M functions with N samples
- **qg** – numpy ndarray of shape (M,N) of M functions with N samples
- **no** – number of components
- **alpha** – amount of smoothing (Default = 0.0 i.e., none)

Return type numpy ndarray

Return wqf f weight function

Return wqg g weight function

Return alpha smoothing value

Return values singular values

CHAPTER

FOUR

UTILITY FUNCTIONS

Utility functions for SRSF Manipulations

moduleauthor:: Derek Tucker <dtucker@stat.fsu.edu>

utility_functions.**SqrtMean**(*gam*)
calculates the srsf of warping functions with corresponding shooting vectors

Parameters **gam** – numpy ndarray of shape (M,N) of M warping functions with N samples

Return type 2 numpy ndarray and vector

Return mu Karcher mean psi function

Return gam_mu vector of dim N which is the Karcher mean warping function

Return psi numpy ndarray of shape (M,N) of M SRSF of the warping functions

Return vec numpy ndarray of shape (M,N) of M shooting vectors

utility_functions.**SqrtMeanInverse**(*gam*)
finds the inverse of the mean of the set of the diffeomorphisms gamma

Parameters **gam** – numpy ndarray of shape (M,N) of M warping functions with N samples

Return type vector

Return gamI inverse of gam

utility_functions.**cumtrapzmid**(*x*, *y*, *c*)
cumulative trapezoidal numerical integration taken from midpoint

Parameters

- **x** – vector of size N describing the time samples
- **y** – vector of size N describing the function
- **c** – midpoint

Return type vector

Return fa cumulative integration

utility_functions.**diffop**(*n*, *binsize=1*)
Creates a second order differential operator

Parameters

- **n** – dimension
- **binsize** – dx (default = 1)

Return type numpy ndarray

Return m matrix describing differential operator

utility_functions.**elastic_distance**(*f1*, *f2*, *time*, *lam=0.0*)

" calculates the distances between function, where f1 is aligned to f2. In other words calculates the elastic distances

Parameters

- **f1** – vector of size N
- **f2** – vector of size N
- **time** – vector of size N describing the sample points
- **lam** – controls the elasticity (default = 0.0)

Return type scalar

Return Dy amplitude distance

Return Dx phase distance

utility_functions.**f_to_srsf**(*f*, *time*)

converts f to a square-root slope function (SRSF)

Parameters

- **f** – vector of size N samples
- **time** – vector of size N describing the sample points

Return type vector

Return q srsf of f

utility_functions.**geigen**(*Amat*, *Bmat*, *Cmat*)

generalized eigenvalue problem of the form

max tr L'AM / sqrt(tr L'BL tr M'CM) w.r.t. L and M

:param Amat numpy ndarray of shape (M,N) :param Bmat numpy ndarray of shape (M,N) :param Bmat numpy ndarray of shape (M,N)

Return type numpy ndarray

Return values eigenvalues

Return Lmat left eigenvectors

Return Mmat right eigenvectors

utility_functions.**gradient_spline**(*time*, *f*, *smooth=False*)

This function takes the gradient of f using b-spline smoothing

Parameters

- **time** – vector of size N describing the sample points
- **f** – numpy ndarray of shape (M,N) of M functions with N samples
- **smooth** – smooth data (default = F)

Return type tuple of numpy ndarray

Return f0 smoothed functions functions

Return g first derivative of each function

Return g2 second derivative of each function

utility_functions.**innerprod_q**(*time*, *q1*, *q2*)
calculates the innerproduct between two srsfs

:param time vector descrbing time samples :param q1 vector of srsf 1 :param q2 vector of srsf 2

Return type scalar

Return val inner product value

utility_functions.**invertGamma**(*gam*)
finds the inverse of the diffeomorphism gamma

Parameters **gam** – vector describing the warping function

Return type vector

Return gamI inverse of gam

utility_functions.**optimum_reparam**(*q1*, *time*, *q2*, *lam=0.0*)
calculates the warping to align srsf q2 to q1

Parameters

- **q1** – vector of size N or array of NxM samples of first SRSF
- **time** – vector of size N describing the sample points
- **q2** – vector of size N or array of NxM samples samples of second SRSF
- **lam** – controls the amount of elasticity (default = 0.0)

Return type vector

Return gam describing the warping function used to align q2 with q1

utility_functions.**optimum_reparam_pair**(*q*, *time*, *q1*, *q2*, *lam=0.0*)
calculates the warping to align srsf pair q1 and q2 to q

Parameters

- **q** – vector of size N or array of NxM samples of first SRSF
- **time** – vector of size N describing the sample points
- **q1** – vector of size N or array of NxM samples samples of second SRSF
- **q2** – vector of size N or array of NxM samples samples of second SRSF
- **lam** – controls the amount of elasticity (default = 0.0)

Return type vector

Return gam describing the warping function used to align q2 with q1

utility_functions.**outlier_detection**(*q*, *time*, *mq*, *k=1.5*)
calculates outlier's using geodesic distances of the SRSFs from the median

Parameters

- **q** – numpy ndarray of N x M of M SRS functions with N samples
- **time** – vector of size N describing the sample points
- **mq** – median calculated using time_warping.srsf_align()
- **k** – cutoff threshold (default = 1.5)

Returns q_outlier: outlier functions

utility_functions.**randomGamma**(*gam*, *num*)
generates random warping functions

Parameters

- **gam** – numpy ndarray of N x M of M of warping functions
- **num** – number of random functions

Returns rgam: random warping functions

utility_functions.**rgam**(*N*, *sigma*, *num*)
Generates random warping functions

Parameters

- **N** – length of warping function
- **sigma** – variance of warping functions
- **num** – number of warping functions

Returns gam: numpy ndarray of warping functions

utility_functions.**smooth_data**(*f*, *sparam*)
This function smooths a collection of functions using a box filter

Parameters

- **f** – numpy ndarray of shape (M,N) of M functions with N samples
- **sparam** – Number of times to run box filter (default = 25)

Return type numpy ndarray

Return f smoothed functions functions

utility_functions.**update_progress**(*progress*)
This function creates a progress bar

Parameters **progress** – fraction of progress

utility_functions.**warp_q_gamma**(*time*, *q*, *gam*)
warps a srsf q by gam

:param time vector describing time samples :param q vector describing srsf :param gam vector describing warping function

Return type numpy ndarray

Return q_temp warped srsf

References:

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2 [math.ST].

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

f

t

u