skfda.preprocessing.registration.landmark_registration_war

skfda.preprocessing.registration.landmark_registration_warping(*fd*, *landmarks*, ***, *location=None*, *eval_points=None*) [source]

Calculate the transformation used in landmark registration.

Let t_{ij} the time where the sample i has the feature j and t_j^* the new time for the feature. The warping function will transform the new time in the old time, i.e., $h_i(t_j^*) = t_{ij}$. The registered samples can be obtained as $x_i^*(t) = x_i(h_i(t))$.

See [RS05-7-3-1] for a detailed explanation.

Parameters:
- **fd** (`FData`) – Functional data object.
- **landmarks** (*array_like*) – List containing landmarks for each samples.
- **location** (*array_like*, *optional*) – Defines where the landmarks will be alligned. By default it will be used as location the mean of the landmarks.
- **eval_points** (*array_like*, *optional*) – Set of points where the functions are evaluated to obtain a discrete representation of the object.

Returns: FDataGrid with the warpings function needed to register the functional data object.

Return type: `FDataGrid`

Raises: `ValueError` – If the object to be registered has domain dimension greater than 1 or the list of landmarks or locations does not match with the number of samples.

References:

[RS05-7-3-1] Ramsay, J., Silverman, B. W. (2005). Feature or landmark registration. In *Functional Data Analysis* (pp. 132-136). Springer.

Examples

```
>>> from skfda.datasets import make_multimodal_landmarks
>>> from skfda.datasets import make_multimodal_samples
>>> from skfda.preprocessing.registration import landmark_registration_warping
```

We will create a data with landmarks as example

```
>>> fd = make_multimodal_samples(n_samples=3, n_modes=2, random_state=9)
>>> landmarks = make_multimodal_landmarks(n_samples=3, n_modes=2,
...                                       random_state=9)
>>> landmarks = landmarks.squeeze()
```

The function will return the corresponding warping function

```
>>> warping = landmark_registration_warping(fd, landmarks)
>>> warping
FDataGrid(...)
```

The registered function can be obtained using function composition

```
>>> fd.compose(warping)
FDataGrid(...)
```