skfda.preprocessing.registration.landmark_shift

skfda.preprocessing.registration.landmark_shift(*fd*, *landmarks*, *location=None*, ***, *restrict_domain=False*, *extrapolation=None*, *eval_points=None*, ***kwargs*) [source]

Perform a shift of the curves to align the landmarks.

Let t^* the time where the landmarks of the curves will be aligned, t_i the location of the landmarks for each curve and $\delta_i = t_i - t^*$.

The registered samples will have their feature aligned.

$$x_i^*(t^*) = x_i(t^* + \delta_i) = x_i(t_i)$$

Parameters:

- **fd** (`FData`) – Functional data object.
- **landmarks** (*array_like*) – List with the landmarks of the samples.
- **location** (*numeric or callable*, *optional*) – Defines where the landmarks will be alligned. If a numeric value is passed the landmarks will be alligned to it. In case of a callable is passed the location will be the result of the the call, the function should be accept as an unique parameter a numpy array with the list of landmarks. By default it will be used as location $\frac{1}{2}(max(\text{landmarks}) + min(\text{landmarks}))$ wich minimizes the max shift.
- **restrict_domain** (*bool*, *optional*) – If True restricts the domain to avoid evaluate points outside the domain using extrapolation. Defaults uses extrapolation.
- **extrapolation** (*str or Extrapolation*, *optional*) – Controls the extrapolation mode for elements outside the domain range. By default uses the method defined in fd. See extrapolation to more information.
- **eval_points** (*array_like*, *optional*) – Set of points where the functions are evaluated in `shift()` .
- ****kwargs** – Keyword arguments to be passed to `shift()` .

Returns: Functional data object with the registered samples.

Return type: `FData`

Examples

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

We will create a data with landmarks as example

```
>>> fd = make_multimodal_samples(n_samples=3, random_state=1)
>>> landmarks = make_multimodal_landmarks(n_samples=3, random_state=1)
>>> landmarks = landmarks.squeeze()
```

The function will return the sample registered

```
>>> landmark_shift(fd, landmarks)
FDataGrid(...)
```