Functional data analysis (FDA)

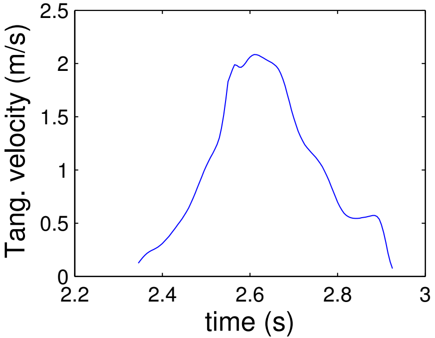
Jason Friedman

MACCS
Macquarie University

19th November 2010

What is functional data analysis (FDA)

- Functional data is made up of repeated measurements, taken as a function of something (e.g., time)
- For example, a trajectory is an example of functional data - we have the position or velocity sampled at many time points



What is functional data analysis

- The classic way to study functional data is to use a parametric model $p(x|\theta)$, where we describe the data using a small number of parameters
- The nonparametric approach is where we assume only smoothness, and fit a function $p(x)$ to the data.
- We can look at many types of data with this method.
- Today I will focus on movement data

What is functional data analysis

- The main benefit of FDA is that we look at functional data (trajectories) as a whole
- It does not require us to select a single dependent variable to study
- It is particularly useful at the "exploration" stage, to see what the variation in the data is
- We can use functional versions of common analysis methods (e.g. mean, std, PCA, ANOVA)
- We can take advantage of the extra information we get from looking at all the data rather than selecting a dependent variable, which necessarily means losing information

Representing functions by basis functions

- In FDA, we want to represent our experimental data with a series of basis functions
- In this way, we can deal with the parameters of the basis functions and not the actual data
- We want to select enough parameters to represent well the data, but not too many parameters

Representing functions by basis functions

- We can use any type of function (in theory) as a basis function
- For periodic data, we usually use a Fourier basis
- In practice, we usually use splines for trajectory data.
- Note that this is the same type of function we used for interpolation, and we use it here for similar reasons. Again, we use a series of splines joined together
- We then select the parameters of these functions to give the best fit to the data

$$x = \sum_{k=1}^{N} c_k \Phi_k \approx y$$

Matlab: Representing functions by basis functions

- All the examples today use the FDA matlab toolkit (available online from `http://functionaldata.org`)
- We create a basis function by defining the range of x values it should span (in our case time), the number of basis functions (50), and the order of the spline (typically $4 = $ cubic).

```
t = (0:1/200:maxtime)';
pbasis = create_bspline_basis([0 maxtime],50,4);
fdnames = {'time','repetition','tang. vel'};
[fdobj,df,gcv] = smooth_basis(...
  t,tangvel_array,pbasis,[],fdnames);
```

- We can also do a similar procedure with time-normalised data

Summary statistics for functional data

- We can do summary statistics (e.g. mean, standard deviation) with functional data
- It is a good way to make sure that everything is working properly

Matlab: Summary statistics for functional data

- Plot the mean $\pm$ standard deviation

```
figure;
plot(mean(fdobj));
hold on;
plot(mean(fdobj)-std(fdobj));
plot(mean(fdobj)+std(fdobj));
ylim([-0.2 1.8]);
```

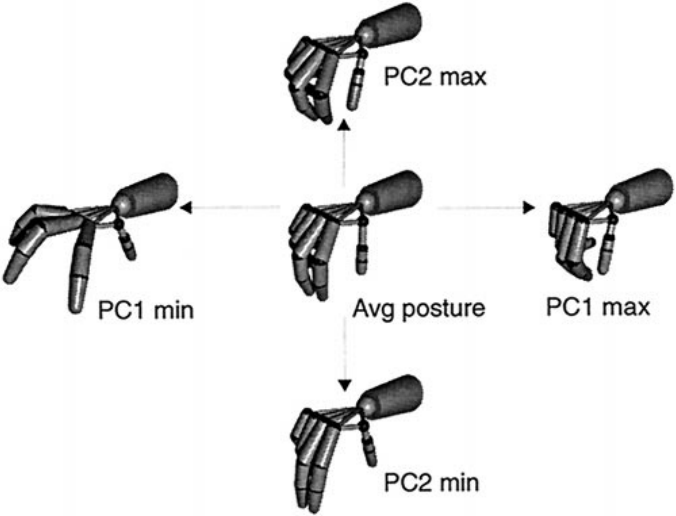
- Note that this is using the overloaded functions for addition and plot provided in the FDA toolkit
- If you want to plot these values in some other way, you need to sample the functional data object

"Regular" PCA

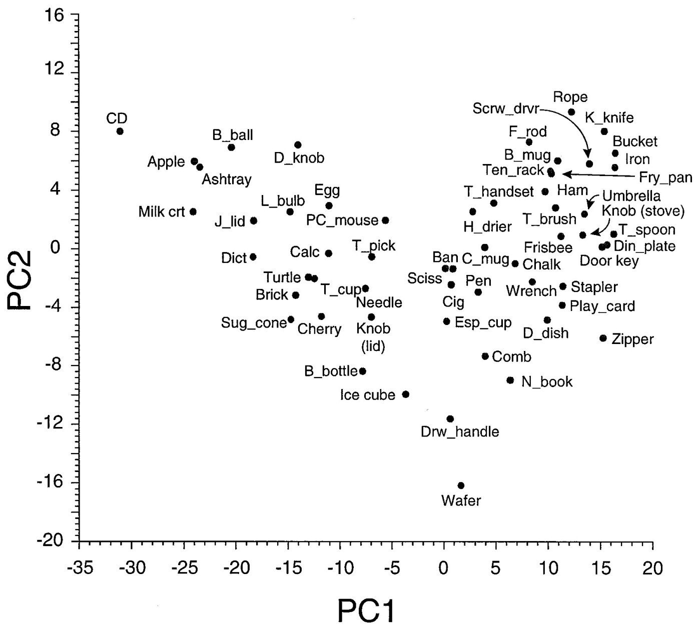
- Principal Component Analysis (PCA) is a technique that identifies the sources of variance in data
- The first principal component is a new axis that has as high a variance along it as possible
- Subsequent principal components have as high a variance as possible (with the remaining variane), and are orthogonal to all previous principal components

“Regular” PCA - example with grasping

- PCA allows us to reconstruct (approximately) the original data using a lower dimensional representation



“Regular” PCA - example with grasping



Functional PCA

- Regular PCA is not very useful for data with 1 or 2 dimensions
- In functional PCA, the "principal component" consists of the entire trajectory
- As with regular PCA, it is typical to first subtract the mean trajectory before performing the analysis
- Again, the first principal component explains most of the variance, and the other PCs are orthogonal (but this time in a space of coefficients of the basis functions)
- So the first PC is a function that is the major source of variation from the mean

Matlab: Functional PCA

- We need to specify how many PCs we want (here we pick 4)
- For this type of data 2-3 PCs is usually sufficient (use a scree plot to decide)

```
nharm = 4;
pcastr = pca_fd(fdobj,nharm);
plot_pca_fd(pcastr,1,0,0,[],1);
```

- The toolkit provides a function to plot the principal components
- Green indicates mean + PC1, red indicates mean -PC1.

Functional PCA

- We can quantify the differences in conditions by looking at the PC weights (i.e. how much they vary from the mean in the direction of the PC)
- We can use the PC weights as a DV in a standard statistic analysis

```
for PC=1:nharm
    for condition=1:numconditions
        thesescores = pcastr.harmscr...
          (c==condition,PC);
        meanscores(condition,PC) = ...
          mean(thesescores);
        stdscores(condition,PC) = ...
          std(thesescores);
    end
    subplot(2,2,PC);
    errorbar([3 6 12 24 48],meanscores(:,PC),...
      stdscores(:,PC));
    xlabel('coherence');
    ylabel('PC weight');
    title(['PC' num2str(PC)]);
end
```

Functional ANOVA

- We can also do analogues of the usual statistical tests
- Rather than performing the tests on a particular measure, we can perform the tests as a function of something (e.g. time)
- In order to perform an ANOVA, we need to define a linear model
- In this case, we hypothesise that the tangential velocity is a function of the coherence

Functional ANOVA

- We define a linear model, then find the best parameters (using regression):

$$y(t) = \beta_0(t) + \sum_{j=1}^{5} x_{ij}\beta_j(t) + \epsilon_i(t)$$

  - $y(t)$ is the functional response
  - β are the weights
  - x_{ij} is 1 or 0, corresponding to the coherence
  - The first column is all ones (for the mean)
- We need to include the constraint

$$\sum_{j=1}^{5} \beta_j(t) = 0$$

for all t to identify the separate effects.

Functional ANOVA

- We can then calculate a continuous version of an F score:

$$F(t) = \frac{Var[\hat{y}(t)]}{\frac{1}{n}\sum(y_i(t) - \hat{y}(t))^2}$$

- where $\hat{y}$ is the predicted vales from the regression, y the actual values
- However, we don't know what the critical F value is
- So instead we use a permutation test
- The null hypothesis is that there is no effect of coherence, so we can switch the coherence labels.
- We repeat this many times, then see if F is greater than 0.95 of the F values from the permutations.

Matlab: Functional ANOVA

- Include the constraint by adding a column of zeros to the coefficients

```
coef = getcoef(fdobj_timenormalised);
coef_augmented = [coef zeros(50,1)];
fd_obj_tn_aug = fd(coef_augmented,pbasis_norm,...
fdnames);
```

Matlab: Functional ANOVA

- Set up the the covariates, according to the coherence levels

```
coherenceCell = cell(numconditions+1,1);
coherenceCell{1} = ones(151,1);
for j=2:numconditions+1;
    xj = zeros(151,1);
    xj(c == j-1) = 1;
    xj(151) = 1;
    coherenceCell{j} = xj;
end
```

Matlab: Functional ANOVA

- Set up the regression coefficients

```
betabasis = create_bspline_basis([0 1],50,4);
betafdPar = fdPar(betabasis);
betaCell  = cell(numconditions+1,1);
for j = 1:numconditions+1
    betaCell{j} = betafdPar;
end
```

- Perform the regression

```
fRegressStr = fRegress(fd_obj_tn_aug,...
 coherenceCell, betaCell);
```

Matlab: Functional ANOVA

- Perform the permutation test (slow!!!)

```
F_res = Fperm_fd(fd_obj_tn_aug, coherenceCell, ...
    betaCell);
```

- Compare the F values to the 0.95 quantile from permuted data

```
 argvals = linspace(0,1,101);
 phdl = plot(argvals, F_res.Fvals, 'b-', ...
argvals, F_res.qvals_pts, 'b--', ...
   [min(argvals), max(argvals)], ...
   [F_res.qval,F_res.qval], 'b:');
   xlabel('normalised time');
   ylabel('F score');
   legend('Observed statistic',...
       'pointwise','maximum');
```

Summary

- FDA is a technique for analysing functional data (e.g. trajectories) as a whole
- It avoids the need to extract scalar measures, and can help identify subtle differences that may be lost otherwise
- Analysis techniques analogous to classical statistics are available to use with FDA (t-tests, ANOVAs, etc)