Psychometrics of the continuous mind: Time continuous multiple regression as a method to exploit the dynamics of computer mouse movements

Stefan Scherbaum & Maja Dshemuchadse

Supplement

A tutorial to the TCMR toolbox

This tutorial guides you through the basic steps of mouse movement analysis in Matlab using the TCMR toolbox.

In the tutorial data set (10 subjects from the data of Scherbaum et al, 2010, Cognition, study 1), you find the following data:

For each participant, you have a struct of trials. Each trial consists of fields for

- response code (the final response left/right): response
- congruency (1=congruent, 2=incongruent)
- time series of x coordinates
- time series of y coordinates
- time series of sample timings

For reasons of simplicity, invalid trials (erroneous response, missed deadlines) have been removed already.

Before starting the tutorial, make sure that you have the **Matlab Statistics Toolbox** installed, as well as **fminsearchbnd**: (https://de.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd--fminsearchcon, John D'Errico, 2012) and **geom2d** (https://de.mathworks.com/matlabcentral/fileexchange/7844-geom2d, David Legland, 2017).

First load the data into the workspace and create a struct for each participant with the basic properties of the trials (response, congruency) as vectors. This is our usual way of organizing the data: one struct per participant. However, feel free to use any form of organization (tables, CSV files,…) you are familiar with.

```
clear
for p=1:10
    out('Loading log ',p);
    load(out('./log',p,'.mat'));
    pstruct=catTrialLog(trials,{'response','congruency','rt'});%extract basic data from each trial struct
    pstruct.trials=trials;%add trial struct for further processing
    pdata(p)=pstruct;
end
```

Then you will have to perform the following pre-processing steps for each participant:

1. Correct potential sample timing errors: since the mouse is not recorded by a hardware device (as for example in EEG recordings), samples might have been recorded irregularly in time.

2. Align start point of x coordinates to 0, then mirror responses so that all trials are aligned similarly (this step could be omitted if appropriate).

3. Extract the continuous signal properties (angle, velocity) from the raw data, in stimulus-locked and time-normalized format.

4. Extract the static signal properties, i.e. mean deviation and maximum deviation.

```
%% perform preprocessing
warp_samples=100;%100 time slices for time normalized trajectories;
stimlock_samples=200;%200 samples are planned for stimulus locked trajetories - rest is filled with nan
screen_width=1280;%width of screen in the experiment
sf=95;%sampling was at 95 Hz

for p=1:length(pdata)
    out('Preprocessing dataset ',p,'...');
    %initialize matrices for stimulus locked raw data
    pdata(p).x_stimlock=nan(length(pdata(p).trials),stimlock_samples);
    pdata(p).t_stimlock=nan(length(pdata(p).trials),stimlock_samples);
    pdata(p).y_stimlock=nan(length(pdata(p).trials),stimlock_samples);

    %perform preprocessing on each trials
    for tr=1:length(pdata(p).trials)
        %correct sample timing errors
        [x,y,t]=correctSampleTiming(pdata(p).trials(tr).x,pdata(p).trials(tr).y,pdata(p).trials(tr).t);
        %mirror right responses to left
        if(pdata(p).response(tr)==2)
            x=screen_width-x;
        end
        %align x coordinates to zero as start point
        x=x-x(1);

        %add data to raw data matrices
        pdata(p).x_stimlock(tr,:)=normLength(x,stimlock_samples,0);
        pdata(p).y_stimlock(tr,:)=normLength(y,stimlock_samples,0);
        pdata(p).t_stimlock(tr,:)=normLength(t,stimlock_samples,0);
    end

    %extract all dynamic data from stimulus locked raw data
    [pdata(p).x_warp,pdata(p).y_warp,pdata(p).angle_warp,pdata(p).velocity_warp,...
     pdata(p).dev_warp,pdata(p).angle_stimlock,pdata(p).velocity_stimlock,pdata(p).dev_stimlock]...
        =calcTrajectories(pdata(p).x_stimlock,pdata(p).y_stimlock,warp_samples,sf);
    [pdata(p).meandev,pdata(p).maxdev]=calcStatic(pdata(p).x_stimlock,pdata(p).y_stimlock);
end
```

After these preprocessing steps, first check the data quality.

1. Draw the time course of average movements on X-axis for both conditions

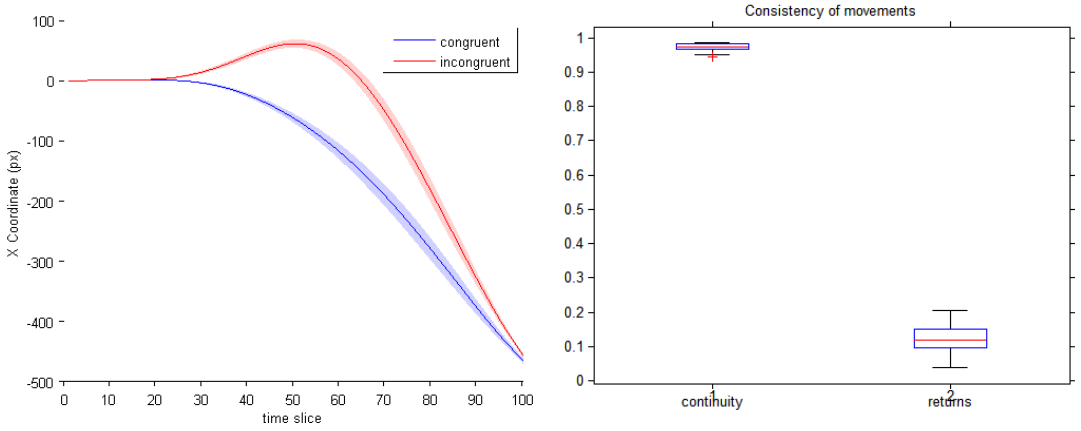
2. Check movement consistency by calculating the movement index (how consistent/straight was the upwards movement, how many backwards movements occurred)

```
%% inspect data

% plot data by condition
%aggregate data
clear cx ix
for p=1:length(pdata)
    cx(p,:)=nanmean(pdata(p).x_warp(pdata(p).congruency==1,:));
    ix(p,:)=nanmean(pdata(p).x_warp(pdata(p).congruency==2,:));
end

%plot mean movements
figure;hold on
errorArea(mean(cx),ste(cx),'b');errorArea(mean(ix),ste(ix),'r');
legend('congruent','incongruent');xlabel('time_slice');ylabel('X Coordinate (px)');

%check movement quality: how straight is each movement and how many movements show returns
for p=1:length(pdata)
    [cont(p,:),returns(p,:)]=calcMovementContinuity(pdata(p).y_warp,1);
end
figure
boxplot([cont,returns])
set(gca,'XTick',1:2);set(gca,'XTickLabels','continuity|returns')
title('Consistency of movements')
```



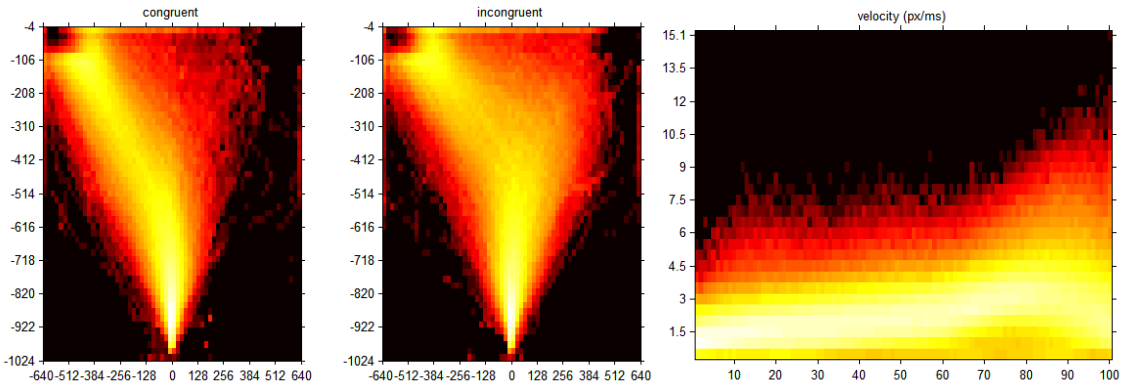
3. Draw X/Y heatmaps for both conditions to judge movement distribution.

4. Draw a heatmap of movement velocities to check the regularity of movements.

```
% plot heatmap of pooled trial per condition
%pool all trials
allx=vertcat(pdata.x_warp);
ally=vertcat(pdata.y_warp);
allvel=vertcat(pdata.velocity_warp);
allcong=vertcat(pdata.congruency);

%plot heatmaps of conditions
figure;colormap('hot')
subplot(1,2,1);
imagep2d(allx(allcong==1,:),-ally(allcong==1,:),-640:20:640,-1024:20:0,[],[],1);
title('congruent')
subplot(1,2,2);
imagep2d(allx(allcong==2,:),-ally(allcong==2,:),-640:20:640,-1024:20:0,[],[],1);
title('incongruent')

% plot heatmap of velocity (=consistency of movement)
figure;colormap('hot')
imagep(allvel,[],[],[],1)
title('velocity (px/ms)')
```



After checking the preconditions, you can now apply TCMR to each participants dataset in the following steps:

1. Code the regressors for the response repetition bias and for congruency.

2. Normalize the regressors to [-1,1].

3. Perform TCMR.

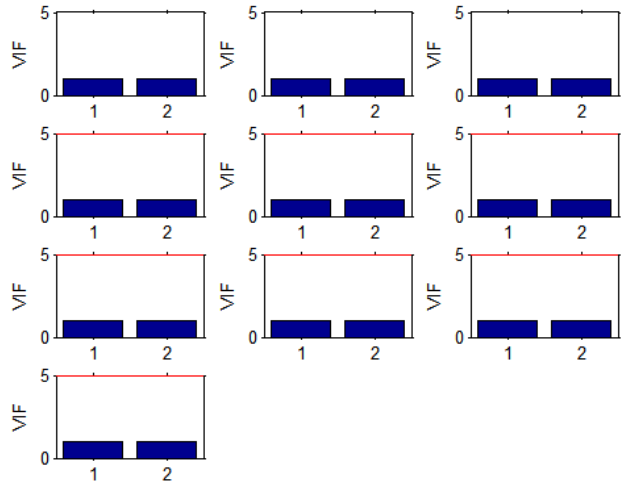
```
clear betas
figure
for p=1:length(pdata)

    %calc regressors of each participant
    congs=pdata(p).congruency;%congruency
    n1response=pdata(p).response~=[0;pdata(p).response(1:end-1)];%response repetition bias

    %concatenate regressors and normalize each to [-1,1]
    regressors=normalizeRegressors([n1response,congs]);
    %define data for TCMR
    regdata=pdata(p).angle_warp;

    %catch plotted information from TCMR
    subplots(length(pdata),p);

    betas(:,:,p)=TCMRegression(regdata,regressors,10);
end
```



Variance inflation factors will be shown for each participant's two regressors. As long as they stay below 5, multicollinearity is no issue.

After performing TCMR per participant, you can extract group-level properties of each beta, e.g. significant segments and peaks.

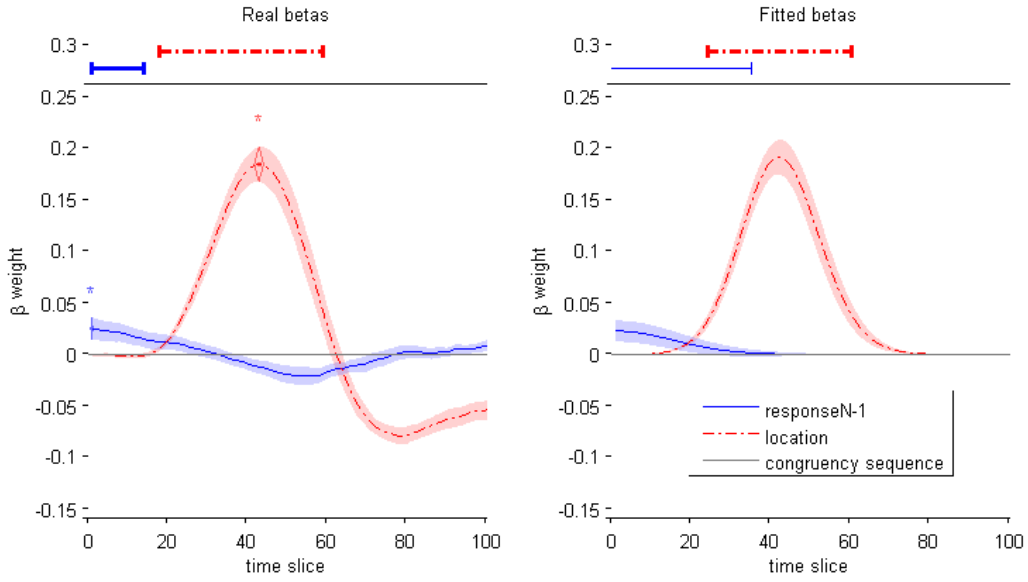
```
%% determine sime process parameters diretly from data
peaks=findStatPeaks(betas,'jackknife');
segments=findStatSegments(betas,0.05);
```

To extract individual parameters, apply the gauss fit procedure to the complete set of beta weights extracted by TCMR: run the gauss fit function.

```
%% perform fitting
figure
[fitbetas,fitparams,fitvalues]=fitRegression(betas,'estimate',1);
```

Afterwards, plot a summary of the results. The parameters can be analysed or exported for further statistical analysis.

```
%% plot results summary for TCMR and fitting results
figure;clear s
s(1)=subplot(1,2,1);
plotRegression(betas);
plotSegmentLines(segments,10);
plotPeaks(peaks,0.05);
xlabel('time slice');ylabel('\beta weight')
title('Real betas')
s(2)=subplot(1,2,2);
plotRegression(fitbetas);
plotModelLines(fitparams,0.1);
xlabel('time slice');ylabel('\beta weight')
title('Fitted betas')
legend({'responseN-1','location','congruency sequence'})
linkaxes(s)
```




```
%% save parameters for statistical analyis in external software (e.g. JASP)
saveCSV('fit_parameters.csv',{'n1response_time','n1response_width',...
                              'n1response_strength','cong_time','cong_width','cong_strength'},...
        [squeeze(fitparams(1,:,:))',squeeze(fitparams(2,:,:))']);
```

```
%% write table with data to commandline
writeParameterTable(fitparams,fitvalues);
writePeakTable(peaks);
writeSegmentTable(segments);
```