%This code was used for the voxelwise analysis (Freud, Culham, Plaut and Behrmann). It calls text files created by BV and utilizes the SLM toolbox. 8/3/2017.For any questions please write to erezfreud@gmail.com

%voxel data from BV. The code correlates and calculate reg. coff between different

 %mni cord (y/z) and the linear slope.

clear all;

clear all;

path = 'define the path to main folder';

no\_cat\_flag=0 % enables to analyze the data based on pictures of object and tools (averaged) (0) or based on picture of objects (excluding all tools) (1)

Experiment 1 subjects:

%subjects={'160224104136','160224122535','160301110546','160307093730','160307111751','160307130827','160310120847','160420131627','160421091747','160427140455','160510164658'}; %

Experiment 2 subjects

subjects={'512\_exp2','024\_exp2','036\_exp2','314\_exp2','351\_exp2','408\_exp2','355\_exp2','236\_exp2','622\_exp2','100\_exp2','133\_exp2'}; % exp2

roi = {'ventral\_r','ventral\_l','dorsal\_r','dorsal\_l'}; %don’t change the order of rois

x=1:5;

% The first loop defines all the visually responsive voxels. It calls a txt file created in BV that includes that beta weights and the t value of all voxels that were included in the initial mask (based on a group-averaged RFX analysis).

group\_voxels=cell(397456,9,length(roi));

for r=1:4

    A = importdata([path 'group\voxel analysis\' roi{r} '\_exp1and2\_MNI\_revision.txt'],' ',24);

    temp=A.data;

    temp1=sortrows(temp, [4,6,8,10]);

    temp\_final=num2cell(temp1);

    group\_t=temp\_final(:,[1 2 3 4 6 8 10 12]);

    clear temp\_final

    clear temp1

    clear temp

    clear A

    for v=2:length(group\_t)

    if cell2mat(group\_t(v-1,6))==cell2mat(group\_t(v,6)) %reduce resampled voxels

    group\_voxels(v,4:9,r)=num2cell(NaN);

    group\_voxels(v,1:3,r)=group\_t(v,1:3);

    else

        if r>2

            if ((cell2mat(group\_t(v,4))>2.6 || cell2mat(group\_t(v,5))>2.6  || cell2mat(group\_t(v,6))>2.6 || cell2mat(group\_t(v,7))>2.6)  || cell2mat(group\_t(v,8))>2.6)

            group\_voxels(v,1:8,r)=group\_t(v,:);

            group\_voxels(v,9,r)=num2cell(v);

            else

            group\_voxels(v,4:9,r)=num2cell(NaN);

            group\_voxels(v,1:3,r)=group\_t(v,1:3);

            end

        else

            if (cell2mat(group\_t(v,4))>2.6 || cell2mat(group\_t(v,5))>2.6  || cell2mat(group\_t(v,6))>2.6  || cell2mat(group\_t(v,7))>2.6)  || cell2mat(group\_t(v,8))>2.6

            group\_voxels(v,1:8,r)=group\_t(v,:);

            group\_voxels(v,9,r)=num2cell(v);

            else

            group\_voxels(v,4:9,r)=num2cell(NaN);

            group\_voxels(v,1:3,r)=group\_t(v,1:3);

            end

        end

    end

            group\_voxels(1,4:9,r)=num2cell(NaN);

            group\_voxels(1,1:3,r)=group\_t(v,1:3);

    end

end

 ventral\_right\_voxels=sortrows( cell2mat(group\_voxels(:,:,1)),[9,1 -2 3]);

 ventral\_left\_voxels=sortrows( cell2mat(group\_voxels(:,:,2)),[9,-1,-2,3]);

 dorsal\_right\_voxels=sortrows( cell2mat(group\_voxels(:,:,3)),[9,1,-2,3]);

 dorsal\_left\_voxels=sortrows( cell2mat(group\_voxels(:,:,4)),[9,-1,-2,3]);

  clear group\_voxels group\_t

 ventral\_right= nan(sum(~isnan(ventral\_right\_voxels(:,9))),9,length(subjects));

 ventral\_left=nan(sum(~isnan(ventral\_left\_voxels(:,9))),9,length(subjects));

 dorsal\_right=nan(sum(~isnan(dorsal\_right\_voxels(:,9))),9,length(subjects));

 dorsal\_left=nan(sum(~isnan(dorsal\_left\_voxels(:,9))),9,length(subjects));

% The result of the first loop is four matrices (ventral right, ventral left, dorsal right and dorsal left) each of them includes all the voxels that were found to be visually responsive (and their coordinates). The txt files were created in BV (voxel-wise GLM analysis)

% The second loop computes the slope (shape sensitivity) for each participant and each voxel.

for s= 1:length(subjects) % all subjects

    for r=1:length(roi) % all rois

    if no\_cat\_flag==1

    A = importdata([path subjects{s} '\voxel analysis\' roi{r} '\_mni\_revision.txt'],' ',24);

    [path subjects{s} '\voxel analysis\' roi{r} '\_mni\_revision.txt']

    else

    A = importdata([path subjects{s} '\voxel analysis\' roi{r} '\_mni\_revision\_onlyobjects.txt'],' ',24);

    end

    temp=A.data;

    if r==1

        for v=1:length(ventral\_right)

        temp\_intersect=intersect(intersect(find((temp(:,1))==ventral\_right\_voxels(v,1)),find((temp(:,2))==ventral\_right\_voxels(v,2))),find((temp(:,3))==ventral\_right\_voxels(v,3))); % detects the voxel

        ventral\_right(v,1:8,s)=(temp(temp\_intersect,[1 2 3 4 6 8 10 12])); % imports the coordinates (1,2,3) and the beta weights 4-intact 6-s4 8-s16 10 s-64 12-s256

        slopeO=polyfit(x,ventral\_right(v,4:8,s),1); % computes the linear fit of the beta weights

        ventral\_right(v,9,s)=slopeO(1); %objects

        end

    elseif r==2

       for v=1:length(ventral\_left)

        temp\_intersect=intersect(intersect(find((temp(:,1))==ventral\_left\_voxels(v,1)),find((temp(:,2))==ventral\_left\_voxels(v,2))),find((temp(:,3))==ventral\_left\_voxels(v,3)));

        ventral\_left(v,1:8,s)=(temp(temp\_intersect,[1 2 3 4 6 8 10 12]));

        slopeO=polyfit(x,ventral\_left(v,4:8,s),1);

        ventral\_left(v,9,s)=slopeO(1);

       end

    elseif r==3

       for v=1:length(dorsal\_right)

        temp\_intersect=intersect(intersect(find((temp(:,1))==dorsal\_right\_voxels(v,1)),find((temp(:,2))==dorsal\_right\_voxels(v,2))),find((temp(:,3))==dorsal\_right\_voxels(v,3)));

        dorsal\_right(v,1:8,s)=(temp(temp\_intersect,[1 2 3 4 6 8 10 12]));

        slopeO=polyfit(x,dorsal\_right(v,4:8,s),1);

        dorsal\_right(v,9,s)=slopeO(1);

       end

    else

        for v=1:length(dorsal\_left)

        temp\_intersect=intersect(intersect(find((temp(:,1))==dorsal\_left\_voxels(v,1)),find((temp(:,2))==dorsal\_left\_voxels(v,2))),find((temp(:,3))==dorsal\_left\_voxels(v,3)));

        dorsal\_left(v,1:8,s)=(temp(temp\_intersect,[1 2 3 4 6 8 10 12]));

        slopeO=polyfit(x,dorsal\_left(v,4:8,s),1);

        dorsal\_left(v,9,s)=slopeO(1);

        end

    end

    end

             a=s

end

%reverse slopes (negative to positive, such that positive slope equals to greater beta weights for the less distorted images

ventral\_right\_avg(:,9)=-ventral\_right\_avg(:,9);

ventral\_left\_avg(:,9)=-ventral\_left\_avg(:,9);

dorsal\_right\_avg(:,9)=-dorsal\_right\_avg(:,9);

dorsal\_left\_avg(:,9)=-dorsal\_left\_avg(:,9);

%reverse slopes (negative to positive)

ventral\_right(:,9,:)=-ventral\_right(:,9,:);

ventral\_left(:,9,:)=-ventral\_left(:,9,:);

dorsal\_right(:,9,:)=-dorsal\_right(:,9,:);

dorsal\_left(:,9,:)=-dorsal\_left(:,9,:);

 %

 %This loop measures the distance from posterior- inferior voxel based on the z and y coordinates

for i=1:length(ventral\_left)

    xy=[min(ventral\_left(:,2,1)),max(ventral\_left(:,3,1));ventral\_left(i,2,1),ventral\_left(i,3,1)];

    ventral\_left(i,10,:)=pdist(xy,'euclidean');

end

ventral\_left\_avg(:,10)=ventral\_left(:,10,1)

for i=1:length(ventral\_right)

    xy=[min(ventral\_right(:,2,1)),max(ventral\_right(:,3,1));ventral\_right(i,2,1),ventral\_right(i,3,1)];

    ventral\_right(i,10,:)=pdist(xy,'euclidean');

end

ventral\_right\_avg(:,10)=ventral\_right(:,10,1)

for i=1:length(dorsal\_right)

    xy=[min(dorsal\_right(:,2,1)),min(dorsal\_right(:,3,1));dorsal\_right(i,2,1),dorsal\_right(i,3,1)];

    dorsal\_right(i,10,:)=pdist(xy,'euclidean');

end

dorsal\_right\_avg(:,10)=dorsal\_right(:,10,1)

for i=1:length(dorsal\_left)

    xy=[min(dorsal\_left(:,2,1)),min(dorsal\_left(:,3,1));dorsal\_left(i,2,1),dorsal\_left(i,3,1)];

    dorsal\_left(i,10,:)=pdist(xy,'euclidean');

end

dorsal\_left\_avg(:,10)=dorsal\_left(:,10,1)

%%

%%%%

%This part of the code utilizes the SLM toolbox to computes the piecewise regression for each participant. The function slm\_optimize (see at the bottom of this code) is used to optimize the R2. The SLM toolbox is an external toolbox that needs to be downloaded.

g=0;

h=0;

gg=0

%to mark where if I didn't find a good match with the SLM method

noSLM\_vr=NaN(length(subjects),12);noSLM\_vl=NaN(length(subjects),12);noSLM\_dr=NaN(length(subjects),12);noSLM\_dl=NaN(length(subjects),12);

 for s=1:length(subjects)

     for r=1:4

        if r==1

        %ventral right – distance (y and z coordinate) – simple correlations

spatial\_correlation\_vr(s,1)=corr(ventral\_right(:,9,s),ventral\_right(:,10,s),'rows','complete'); %simple correlation y axis

        spatial\_correlation\_vr(s,2)=corr(ventral\_right(:,9,s),ventral\_right(:,2,s),'rows','complete');

%piecewise regression:

        points\_x=slm\_optimize(ventral\_right(:,10,s),ventral\_right(:,9,s));

        slm\_second=slmengine(ventral\_right(:,10,s),ventral\_right(:,9,s),'degree','linear','knots',[points\_x]);

        if or(points\_x(2)-points\_x(1)<3,points\_x(3)-points\_x(2)<3)

        points\_x=NaN;

        noSLM\_vr(s,1)=1;

        end

        compoenet\_vr(s,1)=corr(ventral\_right(find((ventral\_right(:,10,s))<points\_x(2)),10,s),ventral\_right([find((ventral\_right(:,10,s))<points\_x(2))],9,s));

        compoenet\_vr(s,2)=corr(ventral\_right([find((ventral\_right(:,10,s))>=points\_x(2))],10,s),ventral\_right([find((ventral\_right(:,10,s))>=points\_x(2))],9 ,s));

        vr\_sq(s,1)=slm\_second.stats.R2; % %shows the R2

        %computes the location of the regression inflection point

        elbow\_loc\_vr(s,1)=points\_x(2);

%y axis

        points\_y=slm\_optimize(ventral\_right(:,2,s),ventral\_right(:,9,s));

slm\_second=slmengine(ventral\_right(:,2,s),ventral\_right(:,9,s),'degree','linear','knots',[point s\_y]);

        if or(points\_y(2)-points\_y(1)<3,points\_y(3)-points\_y(2)<3)

            points\_y=NaN;

            noSLM\_vr(s,2)=1;

        end

        points\_y\_group(s,1:2)=points\_y(1:2);

compoenet\_vr(s,3)=corr(ventral\_right([find((ventral\_right(:,2,s))<points\_y(2))],2,s),ventral\_right([find((ventral\_right(:,2,s))<points\_y(2))],9,s));

compoenet\_vr(s,4)=corr(ventral\_right([find((ventral\_right(:,2,s))>=points\_y(2))],2,s),ventral\_right([find((ventral\_right(:,2,s))>=points\_y(2))],9,s));

        vr\_sq(s,2)=slm\_second.stats.R2;

elbow\_sensitivity\_vr(s,2)=mean(ventral\_right(find(ventral\_right(:,2,s)<points\_y(2)+4 & ventral\_right(:,2,s)>points\_y(2)-4 ),9,s));

        elbow\_loc\_vr(s,2)=points\_y(2);

        elseif r==2 %ventral left

        spatial\_correlation\_vl(s,1)=corr(ventral\_left(:,9,s),ventral\_left(:,10,s),'rows','complete');

        spatial\_correlation\_vl(s,2)=corr(ventral\_left(:,9,s),ventral\_left(:,2,s),'rows','complete');

        points\_x=slm\_optimize(ventral\_left(:,10,s),ventral\_left(:,9,s));

        slm\_second=slmengine(ventral\_left(:,10,s),ventral\_left(:,9,s),'degree','linear','knots',[points\_x]);

        if or(points\_x(2)-points\_x(1)<3,points\_x(3)-points\_x(2)<3)

        noSLM\_vl(s,1)=1;

        points\_x=NaN;

        end

        vl\_sq(s,1)=slm\_second.stats.R2;

        compoenet\_vl(s,1)=corr(ventral\_left(find((ventral\_left(:,10,s))<points\_x(2)),10,s),ventral\_left([find((ventral\_left(:,10,s))<points\_x(2))],9,s));

        compoenet\_vl(s,2)=corr(ventral\_left([find((ventral\_left(:,10,s))>=points\_x(2))],10,s),ventral\_left([find((ventral\_left(:,10,s))>=points\_x(2))],9,s));

        elbow\_loc\_vl(s,1)=points\_x(2);

        points\_y=slm\_optimize(ventral\_left(:,2,s),ventral\_left(:,9,s));

        slm\_second=slmengine(ventral\_left(:,2,s),ventral\_left(:,9,s),'degree','linear','knots',[points\_y]);

        if or(points\_y(2)-points\_y(1)<3,points\_y(3)-points\_y(2)<3)

            noSLM\_vl(s,2)=1;

            points\_y=NaN;

        end

        compoenet\_vl(s,3)=corr(ventral\_left([find((ventral\_left(:,2,s))<points\_y(2))],2,s),ventral\_left([find((ventral\_left(:,2,s))<points\_y(2))],9,s));

        compoenet\_vl(s,4)=corr(ventral\_left([find((ventral\_left(:,2,s))>=points\_y(2))],2,s),ventral\_left([find((ventral\_left(:,2,s))>=points\_y(2))],9,s));

        vl\_sq(s,2)=slm\_second.stats.R2;

        elbow\_loc\_vl(s,2)=points\_y(2);

        elseif r==3 % dorsal right

        spatial\_correlation\_dr(s,1)=corr(dorsal\_right(:,9,s),dorsal\_right(:,10,s),'rows','complete');

        spatial\_correlation\_dr(s,2)=corr(dorsal\_right(:,9,s),dorsal\_right(:,2,s),'rows','complete');

        points\_x=slm\_optimize(dorsal\_right(:,10,s),dorsal\_right(:,9,s));

        slm\_second=slmengine(dorsal\_right(:,10,s),dorsal\_right(:,9,s),'degree','linear','knots',[points\_x]);

            if or(points\_x(2)-points\_x(1)<3,points\_x(3)-points\_x(2)<3)

            noSLM\_dr(s,1)=1;

            points\_x=NaN;

            end

        compoenet\_dr(s,1)=corr(dorsal\_right(find((dorsal\_right(:,10,s))<points\_x(2)),10,s),dorsal\_right([find((dorsal\_right(:,10,s))<points\_x(2))],9,s));

        compoenet\_dr(s,2)=corr(dorsal\_right([find((dorsal\_right(:,10,s))>=points\_x(2))],10,s),dorsal\_right([find((dorsal\_right(:,10,s))>=points\_x(2))],9,s));

        elbow\_loc\_dr(s,1)=points\_x(2);

        dr\_sq(s,1)=slm\_second.stats.R2;

        points\_y=slm\_optimize(dorsal\_right(:,2,s),dorsal\_right(:,9,s));

        slm\_second=slmengine(dorsal\_right(:,2,s),dorsal\_right(:,9,s),'degree','linear','knots',[points\_y]);

        if or(points\_y(2)-points\_y(1)<3,points\_y(3)-points\_y(2)<3)

            noSLM\_dr(s,2)=1;

            points\_y=NaN;

        end

        points\_y\_group(s,5:6)=points\_y(1:2);

        compoenet\_dr(s,3)=corr(dorsal\_right([find((dorsal\_right(:,2,s))<points\_y(2))],2,s),dorsal\_right([find((dorsal\_right(:,2,s))<points\_y(2))],9,s));

        compoenet\_dr(s,4)=corr(dorsal\_right([find((dorsal\_right(:,2,s))>=points\_y(2))],2,s),dorsal\_right([find((dorsal\_right(:,2,s))>=points\_y(2))],9,s));

        dr\_sq(s,2)=slm\_second.stats.R2;

        elbow\_loc\_dr(s,2)=points\_y(2);

       % dorsal left

        else

        spatial\_correlation\_dl(s,1)=corr(dorsal\_left(:,9,s),dorsal\_left(:,10,s),'rows','complete');

        spatial\_correlation\_dl(s,2)=corr(dorsal\_left(:,9,s),dorsal\_left(:,2,s),'rows','complete');

        points\_x=slm\_optimize(dorsal\_left(:,10,s),dorsal\_left(:,9,s));

        slm\_second=slmengine(dorsal\_left(:,10,s),dorsal\_left(:,9,s),'degree','linear','knots',[points\_x]);

        if or(points\_x(2)-points\_x(1)<3,points\_x(3)-points\_x(2)<3)

            noSLM\_dl(s,1)=1;

            points\_x=NaN;

        end

        compoenet\_dl(s,1)=corr(dorsal\_left(find((dorsal\_left(:,10,s))<points\_x(2)),10,s),dorsal\_left([find((dorsal\_left(:,10,s))<points\_x(2))],9,s));

        compoenet\_dl(s,2)=corr(dorsal\_left([find((dorsal\_left(:,10,s))>=points\_x(2))],10,s),dorsal\_left([find((dorsal\_left(:,10,s))>=points\_x(2))],9,s));

        dl\_sq(s,1)=slm\_second.stats.R2;

        elbow\_loc\_dl(s,1)=points\_x(2);

        points\_y=slm\_optimize(dorsal\_left(:,2,s),dorsal\_left(:,9,s));

        slm\_second=slmengine(dorsal\_left(:,2,s),dorsal\_left(:,9,s),'degree','linear','knots',[points\_y]);

        if or(points\_y(2)-points\_y(1)<3,points\_y(3)-points\_y(2)<3) % if there are not two component

            noSLM\_dl(s,2)=1;

            points\_y=NaN;

        end

        compoenet\_dl(s,3)=corr(dorsal\_left([find((dorsal\_left(:,2,s))<points\_y(2))],2,s),dorsal\_left([find((dorsal\_left(:,2,s))<points\_y(2))],9,s));

        compoenet\_dl(s,4)=corr(dorsal\_left([find((dorsal\_left(:,2,s))>=points\_y(2))],2,s),dorsal\_left([find((dorsal\_left(:,2,s))>=points\_y(2))],9,s));

        dl\_sq(s,2)=slm\_second.stats.R2;

        elbow\_loc\_dl(s,2)=points\_y(2);

        end

    end

 g=g+3;

 s

 gg=gg+3;

 end

 %fisher transformation

compoenet\_dl(:)=0.5\*log((1+compoenet\_dl(:,:))./(1-compoenet\_dl(:,:)));

compoenet\_dr(:)=0.5\*log((1+compoenet\_dr(:,:))./(1-compoenet\_dr(:,:)));

compoenet\_vr(:)=0.5\*log((1+compoenet\_vr(:,:))./(1-compoenet\_vr(:,:)));

compoenet\_vl(:)=0.5\*log((1+compoenet\_vl(:,:))./(1-compoenet\_vl(:,:)));

spatial\_correlation\_dl(:,:)=0.5\*log((1+spatial\_correlation\_dl(:,:))./(1-spatial\_correlation\_dl(:,:)));

spatial\_correlation\_dr(:,:)=0.5\*log((1+spatial\_correlation\_dr(:,:))./(1-spatial\_correlation\_dr(:,:)));

spatial\_correlation\_vl(:,:)=0.5\*log((1+spatial\_correlation\_vl(:,:))./(1-spatial\_correlation\_vl(:,:)));

spatial\_correlation\_vr(:,:)=0.5\*log((1+spatial\_correlation\_vr(:,:))./(1-spatial\_correlation\_vr(:,:)));

%%

%count number of shape selective voxels in each pathway

for s = 1:length(subjects)

    num\_of\_vox(s,1)= sum(ventral\_right(:,9,s)>0)

    num\_of\_vox(s,2)= sum(ventral\_left(:,9,s)>0)

    num\_of\_vox(s,3)= sum(dorsal\_right(:,9,s)>0)

    num\_of\_vox(s,4)= sum(dorsal\_left(:,9,s)>0)

    avg\_slope(s,1)=nanmean(ventral\_right(ventral\_right(:,9,s)>0,9,s))

    avg\_slope(s,2)=nanmean(ventral\_left(ventral\_left(:,9,s)>0,9,s))

    avg\_slope(s,3)=nanmean(dorsal\_right(dorsal\_right(:,9,s)>0,9,s))

    avg\_slope(s,4)=nanmean(dorsal\_left(dorsal\_left(:,9,s)>0,9,s))

end

% slm\_optimize function looks for the highest R2 value and determines the inflection point of the piecewise regression

function [knot2]=slm\_optimize (x,y)

        clear a

        a=1;

        clear slmtry

        for i=min(x)+1:max(x)-1

        slm\_second=slmengine(x,y,'degree','linear','knots',[min(x),i,max(x)]);

        slmtry(a,:)=[i,slm\_second.stats.R2] ;

        a=a+1;

        end

        knot2=[min(x),slmtry(find(slmtry(:,2)==max(slmtry(:,2))),1),max(x)];

       % r2=slmtry(find(slmtry(:,2)==max(slmtry(:,2))),2);

end