

```
function [outputArg1,outputArg2] = runSPM(folderin,folderout)
%Runs processJumps using file list from getSFIfiles
%then runs SPM stats from spm1d package
%
%
% param.SPM = true;
% setParameters = param;
% group = 1 (benchmarking) or 2 (aclr)
```

IMPORT DATA

```
[reslts] = processJumps(folderin,[],[],[],folderout);
SFIbench = readmatrix(fullfile("Z:\Private Student Folders\ShadanA", "wrangled_benchmark_SFI.csv"), 'Range',2, 'OutputType','string');
SFIaclr = readmatrix(fullfile("Z:\Private Student Folders\ShadanA", "wrangled_ACLRmatched_SFI.csv"), 'Range',2, 'OutputType','string');
```

MAIN

extract all force data for each jump

```
alljumps = [];
Dir = struct; % Dir (Directory) will summarize
col = 0;

for iFile = 1:length(reslts)
    for iJump = 1:length(reslts(iFile).jumpStats)
        if ~isempty(reslts(iFile).jumpStats(iJump).jumpData)
            col = col+1;
            jumpdata =
reslts(iFile).jumpStats(iJump).jumpData(reslts(iFile).jumpStats(iJump).phases(1,1):reslts(iFile).jumpStats(iJump).phases(4,1),2:3); % L
and R force for each jump

            normfactor = length(jumpdata)/1001;
            jumpdata = interp1([1:length(jumpdata)],jumpdata,linspace(1,length(jumpdata),101)); % normalize to 101 points
            jumpdata = sum(jumpdata,2); % sum by row to create total force

            Dir(col).Athlete = reslts(iFile).Athlete;
```

```

Dir(col).Sex = reslts(iFile).Sex;
Dir(col).JumpNum = reslts(iFile).jumpStats(iJump).JumpNumb; % each athlete has 3 jumps on each leg
Dir(col).Legs = reslts(iFile).jumpStats(iJump).SL; % jumping leg
Dir(col).JumpType = reslts(iFile).JumpType; % CMJ (countermovement jump) or RHT (repeated hop test)
Dir(col).extraload = reslts(iFile).Extra_Load; % jumping with load
Dir(col).SystemMass = reslts(iFile).System_Mass; % athlete mass
Dir(col).phases = round(reslts(iFile).jumpStats(iJump).velImpulseTimes/normfactor);

% assign limb function catagory
if SFIbench(find(contains(SFIbench(:,2),reslts(iFile).Athlete)),34)=="Good"
    Dir(col).LimbFunction = 'Good';
elseif SFIbench(find(contains(SFIbench(:,2),reslts(iFile).Athlete)),34)=="Poor"
    Dir(col).LimbFunction = 'Poor';
elseif SFIaclr(find(contains(SFIaclr(:,2),reslts(iFile).Athlete)),34)=="Poor" % different csv file for aclr athletes
    Dir(col).LimbFunction = 'Poor';
elseif SFIaclr(find(contains(SFIaclr(:,2),reslts(iFile).Athlete)),34)=="Fair" % different csv file for aclr athletes
    Dir(col).LimbFunction = 'Fair';
end

% assign ACLR status
if SFIaclr(find(contains(SFIaclr(:,2),reslts(iFile).Athlete)),24)=="Yes"
    Dir(col).ACLstatus = 'ACLR'; % ACL injured athlete
    Dir(col).ACLside = SFIaclr(find(contains(SFIaclr(:,2),reslts(iFile).Athlete)),26); % injured limb
elseif SFIaclr(find(contains(SFIaclr(:,2),reslts(iFile).Athlete)),24)=="No"
    Dir(col).ACLstatus = 'Match'; % matched control
    Dir(col).ACLside = '';
else
    Dir(col).ACLstatus = 'Control'; % limb function control
    Dir(col).ACLside = '';
end

alljumps(:,col) = jumpdata./(Dir(col).SystemMass*9.81); % normalize force to body mass

end
end
end

```

```
alljumps = alljumps'; % transpose data. SPM wants ROWS as subjects and COLS at timepoints
```

CREATE LIMB FUNCTION GROUPS FOR LEFT, RIGHT, INJURED LEG

```
% Good perceived limb function
Left_good_sfi = find(arrayfun(@(x) strcmp(x.Legs, 'L') & strcmp(x.LimbFunction, 'Good'), Dir));
Right_good_sfi = find(arrayfun(@(x) strcmp(x.Legs, 'R') & strcmp(x.LimbFunction, 'Good'), Dir));
Left_good_sfi = alljumps(Left_good_sfi, :);
Right_good_sfi = alljumps(Right_good_sfi, :);

% Poor perceived limb function
Left_poor_sfi = find(arrayfun(@(x) strcmp(x.Legs, 'L') & strcmp(x.LimbFunction, 'Poor') & ~strcmp(x.ACLstatus, 'ACLR'), Dir));
Right_poor_sfi = find(arrayfun(@(x) strcmp(x.Legs, 'R') & strcmp(x.LimbFunction, 'Poor') & ~strcmp(x.ACLstatus, 'ACLR'), Dir));
Left_poor_sfi = alljumps(Left_poor_sfi, :);
Right_poor_sfi = alljumps(Right_poor_sfi, :);

% ACLR between injured limbs
acl_inj = find(arrayfun(@(x) strcmp(x.ACLstatus, 'ACLR') & ...
    (strcmp(x.ACLside, 'L') & strcmp(x.Legs, 'L')) | ...
    (strcmp(x.ACLside, 'R') & strcmp(x.Legs, 'R')), Dir)); % find indicies where the jumping leg = ACLR leg
acl_uninj = find(arrayfun(@(x) strcmp(x.ACLstatus, 'ACLR') & ...
    (strcmp(x.ACLside, 'L') & strcmp(x.Legs, 'R')) | ...
    (strcmp(x.ACLside, 'R') & strcmp(x.Legs, 'L')), Dir)); % find indicies where the jumping leg is not ACLR leg
acl_inj = alljumps(acl_inj, :);
acl_uninj = alljumps(acl_uninj, :);

% ACLR matched controls
control = find(arrayfun(@(x) strcmp(x.ACLstatus, 'Match') & strcmp(x.Legs, 'R'), Dir));
control = alljumps(control, :);
```

RUN SPM ANALYSIS - spm1d program

```

% left to right
LR_good_spm = spm1d.stats.ttest_paired(Left_good_sfi, Right_good_sfi);
LR_good_spmi = LR_good_spm.inference(0.05, 'two_tailed', true);

LR_poor_spm = spm1d.stats.ttest_paired(Left_poor_sfi, Right_poor_sfi);
LR_poor_spmi = LR_poor_spm.inference(0.05, 'two_tailed', true);

% ACLR between injured limb groups
aclr_spm = spm1d.stats.ttest_paired(acl_inj, acl_uninj);
aclr_spmi = aclr_spm.inference(0.05, 'two_tailed', true);

```

CREATE FIGURE

```

fig12 = tiledlayout(2,3);
fig12.Title.String = 'Between Limb Comparison of Single Leg CMJ';
fig12.Title.FontWeight = 'bold';
fig12.Title.FontSize = 14;
fig12.TileSpacing = 'compact';
% fig12.Position = [1000 300 650 1000];

poorspm = nexttile; % SPM - Poor SFI
LR_poor_spmi.plot()
LR_poor_spmi.plot_p_values()
title('Poor SFI Group', 'FontWeight','normal')

goodspm = nexttile; % SPM - Good SFI
LR_good_spmi.plot()
LR_good_spmi.plot_p_values()
title('Good SFI Group', 'FontWeight','normal')

aclrspm = nexttile; % SPM - ACLR SFI
aclr_spmi.plot()
aclr_spmi.plot_p_values()
title('ACLR Group', 'FontWeight','normal')

poorfz = nexttile; % Fz-time curve - Poor SFI (Left is blue, Right is grey)

```

```

spm1d.plot.plot_meanSD(Left_poor_sfi, 'color', '#457DEE')
hold on
spm1d.plot.plot_meanSD(Right_poor_sfi, 'color', '#8D8E90')
hold off

goodfz = nexttile; % Fz-time curve - Good SFI (Left is blue, Right is grey)
spm1d.plot.plot_meanSD(Left_good_sfi, 'color', '#457DEE')
hold on
spm1d.plot.plot_meanSD(Right_good_sfi, 'color', '#8D8E90')
hold off

aclrfz = nexttile; % Fz-time curve - injured is blue, uninjured is grey
spm1d.plot.plot_meanSD(acl_inj, 'color', '#457DEE')
hold on
spm1d.plot.plot_meanSD(acl_uninj, 'color', '#8D8E90')
hold off

linkaxes ([poorspm, poorfz], 'x');
linkaxes ([goodspm, goodfz], 'x');
linkaxes ([aclrspm, aclrfz], 'x');
linkaxes ([poorspm, goodspm, aclrspm], 'y');
xlabel ([poorfz, goodfz, aclrfz], 'Time (%)')
ylabel ([goodspm, aclrspm], '')
ylabel (poorfz, 'Force (N/BW)')
xticks ([poorspm, poorfz, goodspm, goodfz, aclrspm, aclrfz], [25 50 75 100])
xticklabels ([poorfz, goodfz, aclrfz], {'25' '50' '75' '100'})
xticklabels ([poorspm, goodspm, aclrspm], {})
legend(poorfz, {'Left Limb', '', 'Right Limb'});
legend(goodfz, {'Left Limb', '', 'Right Limb'});
legend(aclrfz, {'Injured Limb', '', 'Uninjured Limb'});
saveas(fig12, 'Z:\Private Student Folders\ShadanA\Figures\BetweenLimbs_ACL_Good_Poor.jpg')

outputArg1 = fig12;
outputArg2 = fig13;
end

```