SUPPORTING INFORMATION

S1. *VSI* and related MATLAB codes including sensitivity investigation

S1.1 *VSI* function

```
function [ result ] = fn_vsi(inside)
% This function calculates the VSI of the object represented as logical
% array named 'inside'

sizeInside=size(inside);

%Find the smallest xy-plane index
for i1=1:sizeInside(1,3)
    Px=(inside(:,:,i1)==1);%crosssection plane
    Np=sum(Px(:));
    if Np~=0
        zmin=i1;
        break
    end
end

%Find the largest xy-plane index
for i1=sizeInside(1,3):-1:1
    Px=(inside(:,:,i1)==1);%crosssection plane
    Np=sum(Px(:));
    if Np~=0
        zmax=i1;
        break
    end
end

num=0;
for z1=zmin+1:zmax
    Px=(inside(:,:,z1)==1);%crosssection plane
    Npxy=sum(Px(:));%number of voxels in crosssection plane
    num=num+Npxy*(z1-zmin);
end

%N: Total number of voxels
N=sum(sum(sum(inside)));
if N~=1
result=exp(-2*num/(N*(N^(1/3)-1)));
else
    result=1;
end
end
```

S1.2 Sensitivity of *VSI* on discretization and size of cubes and spheres

S1.2.1 Sensitivity of *VSI* on discretization

Cubes

```
clear all
clc
close all

%% Construct a 64x64x64 cube; Unit voxel length=8


[X,Y,Z]=meshgrid(8:8:64);
inside=(X>0)&(X<65)&(Y>0)&(Y<65)&(Z>0)&(Z<65);
C=zeros(size(X));
x=4:8:68;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=64 unit; unit voxel length=8 unit');

% Calculate VSI
VSI8=fn_vsi(inside)


%% Construct a 64x64x64 cube; Unit voxel length=4

[X,Y,Z]=meshgrid(4:4:64);
inside=(X>0)&(X<65)&(Y>0)&(Y<65)&(Z>0)&(Z<65);
C=zeros(size(X));
x=2:4:66;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=64 unit; unit voxel length=4 unit');

% Calculate VSI
VSI16=fn_vsi(inside)


%% Construct a 64x64x64 cube; Unit voxel length=2

[X,Y,Z]=meshgrid(2:2:64);
inside=(X>0)&(X<65)&(Y>0)&(Y<65)&(Z>0)&(Z<65);
```

```
C=zeros(size(X));
x=1:2:65;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=64 unit; unit voxel length=2 unit');

% Calculate VSI
VSI32=fn_vsi(inside)



%% Construct a 64x64x64 cube; Unit voxel length=1

[X,Y,Z]=meshgrid(1:64);
inside=(X>0)&(X<65)&(Y>0)&(Y<65)&(Z>0)&(Z<65);
C=zeros(size(X));
x=0.5:1:64.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=64 unit; unit voxel length=1 unit');

% Calculate VSI
VSI64=fn_vsi(inside)
```

Spheres

```
clear
close all
clc

%% Construct a sphere with Radius=32 and Unit voxel=8
R1=32;
uvl=8;%Unit voxel length
[X,Y,Z]=meshgrid(-40:uvl:40);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-44:uvl:44;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=8 units; Diameter=7 voxels');

% Calculate VSI
VSI1=fn_vsi(inside)

%% Construct a sphere with Radius=32 and Unit voxel length=4
% Use interpolation to construct the new sphere
clear x* y* z*
[xi,yi,zi] = meshgrid(-40:4:40,-40:4:40,-40:4:40);
vi1 = interp3(X,Y,Z,inside,xi,yi,zi,'linear');
vi1=(vi1>0);
x1=-42:4:42;
y1=x1;
z1=x1;
C1=zeros(size(xi));
plotCubes(x1,y1,z1,vi1,C1);

% Calculate VSI
VSI2=fn_vsi(vi1)

%% Construct a sphere with Radius=32 and Unit voxel length=2
% Use interpolation to construct the new sphere
clear x* y* z*
[xi,yi,zi] = meshgrid(-40:2:40,-40:2:40,-40:2:40);
vi2 = interp3(X,Y,Z,inside,xi,yi,zi,'linear');
vi2=(vi2>0);
x1=-41:2:41;
y1=x1;
```

```
z1=x1;
C1=zeros(size(xi));
plotCubes(x1,y1,z1,vi2,C1);

% Calculate VSI
VSI3=fn_vsi(vi2)
%% Construct a sphere with Radius=32 and Unit voxel length=1
% Use interpolation to construct the new sphere
clear x* y* z*
[xi,yi,zi] = meshgrid(-40:40,-40:40,-40:40);
vi3 = interp3(X,Y,Z,inside,xi,yi,zi,'linear');
vi3=(vi3>0);
x1=-40.5:40.5;
y1=x1;
z1=x1;
C1=zeros(size(xi));
plotCubes(x1,y1,z1,vi3,C1);

% Calculate VSI
VSI4=fn_vsi(vi3)
```

S1.2.1 Sensitivity of *VSI* on size

Cubes

```
clear
clc
close all

%% Construct a cube with side=4 unit; Unit voxel length=1
[X,Y,Z]=meshgrid(1:4);
inside=(X>0)&(X<5)&(Y>0)&(Y<5)&(Z>0)&(Z<5);
C=zeros(size(X));
x=0.5:1:4.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=4 unit; unit voxel length=1 unit');

% Calculate VSI
VSI4=fn_vsi(inside)

%% Construct a cube with a=8 and Unit voxel length=1

[X,Y,Z]=meshgrid(1:8);
inside=(X>0)&(X<9)&(Y>0)&(Y<9)&(Z>0)&(Z<9);
C=zeros(size(X));
x=0.5:1:8.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=8 unit; unit voxel length=1 unit');

% Calculate VSI
VSI8=fn_vsi(inside)


%% Construct a cube with a=16 and Unit voxel length=1

[X,Y,Z]=meshgrid(1:16);
inside=(X>0)&(X<17)&(Y>0)&(Y<17)&(Z>0)&(Z<17);
C=zeros(size(X));
x=0.5:1:16.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
```

```
title('Cube edge length=16 unit; unit voxel length=1 unit');

% Calculate VSI
VSI16=fn_vsi(inside)



%% Construct a 32x32x32 cube and Unit voxel length=1

[X,Y,Z]=meshgrid(1:32);
inside=(X>0)&(X<33)&(Y>0)&(Y<33)&(Z>0)&(Z<33);
C=zeros(size(X));
x=0.5:1:32.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=32 unit; unit voxel length=1 unit');

% Calculate VSI
VSI32=fn_vsi(inside)

%% Construct a 64x64x64 cube and Unit voxel length=1

[X,Y,Z]=meshgrid(1:64);
inside=(X>0)&(X<65)&(Y>0)&(Y<65)&(Z>0)&(Z<65);
C=zeros(size(X));
x=0.5:1:64.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Cube edge length=64 unit; unit voxel length=1 unit');

% Calculate VSI
VSI64=fn_vsi(inside)
```

Spheres

```
clear
clc
close all

%% Construct a sphere with R=4 and Unit voxel=1
R1=4;
[X,Y,Z]=meshgrid(-20:20);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=1 units; Radius=4 voxels');

% Calculate VSI
VSI4=fn_vsi(inside)

%% Construct a sphere with R=8 and Unit voxel=1
clear

R1=8;
[X,Y,Z]=meshgrid(-20:20);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=1 units; Radius=8 voxels');

% Calculate VSI
VSI8=fn_vsi(inside)


%% Construct a sphere with R=16 and Unit voxel=1
clear

R1=16;
[X,Y,Z]=meshgrid(-20:20);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=1 units; R=16 voxels');
```

```
% Calculate VSI
VSI16=fn_vsi(inside)

%% Construct a sphere with R=32 and Unit voxel=1
clear

R1=32;
[X,Y,Z]=meshgrid(-40:40);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-40.5:40.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=1 units; Radius=32 voxels');

% Calculate VSI
VSI32=fn_vsi(inside)


%% Construct a sphere with R=64 and Unit voxel=1

R1=64;
[X,Y,Z]=meshgrid(-100:100);
R=sqrt(X.^2+Y.^2+Z.^2);
inside=R<R1;
C=zeros(size(X));
x=-100.5:100.5;
y=x;
z=x;
plotCubes(x,y,z,inside,C);
title('Unit voxel length=1 units; Radius=64 voxels');

% Calculate VSI
VSI64=fn_vsi(inside)
```

S1.3 *VSI* Codes for spreading of ellipsoids

```
clear
clc
close all


%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(4 4 32)
xr=4;yr=4;zr=32; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;%Semi-ellipsoid
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))%Show total number of voxels

% Calculate VSI
VSI1=fn_vsi(inside)

%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(5 5 17)
xr=5;yr=5;zr=17; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))
% Calculate VSI
VSI2=fn_vsi(inside)


%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(6 6 12)
xr=6;yr=6;zr=12; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))

% Calculate VSI
VSI3=fn_vsi(inside)

%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(7 7 9)
xr=7;yr=7;zr=9; %semi-axis lengths
```

```
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))

% Calculate VSI
VSI4=fn_vsi(inside)

%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(8 8 7)
xr=8;yr=8;zr=7; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))

% Calculate VSI
VSI5=fn_vsi(inside)

%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(9 9 6)
xr=9;yr=9;zr=6; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))

% Calculate VSI
VSI6=fn_vsi(inside)

%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(10 10 5)
xr=10;yr=10;zr=5; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))
```

```
% Calculate VSI
VSI7=fn_vsi(inside)
%% Construct an elipsoid with (xc,yc,zc)=(0 0 0) (xr yr zr)=(11 11 4)
xr=11;yr=11;zr=4; %semi-axis lengths
[X,Y,Z]=meshgrid(-20:20,-20:20,1:20);
inside=(((X.^2)/(xr^2))+((Y.^2)/(yr^2))+((Z.^2)/(zr^2)))<1;
C=zeros(size(X));
x=-20.5:20.5;
y=x;
z=0.5:20.5;
plotCubes(x,y,z,inside,C);
sum(sum(sum(inside)))

% Calculate VSI
VSI8=fn_vsi(inside)
```

S1.4 *VSI* cell spreading codes

```
clear
close all;
clc

load cellInfoQacl13apr17 %Includes filenames, file paths, and cell masks
cellInfo50(:,4:9)=[];

for ic1=1:50   %Cell counter

% [fname, fld] =  uigetfile({'*.*'},'multiselect','on');
% fpath = [fld,fname]
fld=cellInfo50{ic1,1};
fname=cellInfo50{ic1,2};
BWc=cellInfo50{ic1,3};%Cell mask

% Load GFAP images and segment by percentage thresholding
% N is the number of selected files
n=size(fname);
Ns=n(1,2);
%Construct GFAP MIP image of cell of interest
for ij=1:Ns
        fnamen=fname(1,ij);
        fn=char(fnamen);
        II=importdata([fld,fn]);
        I(:,:,ij)=rgb2gray(II);
end
MIP=max(I,[],3);%Maximum intensity projection image. 3 is the DIM z.

%% Find the bounding box of the cell
IMIP=BWc.*(MIP>0); %IMIP is the MIP thresholded image
%figure;imshow(IMIP);
[y,x]=find(IMIP==1);%Cell is in [min(x) min(y) max(x) max(y)]
%Find the max(cell_width, cell_length)
Imw=max((max(x)-min(x)),(max(y)-min(y)));
Imwx=max(x)-min(x);
Imwy=max(y)-min(y);

% Slice thickness 1.13um : Olympus Fluoviewer Software
% File>Properties>Z dimension
sliceThickness=1.13; %unit is um
% Field of view File>Properties>X&Y dimension
FOV=317.13*317.13; %unit is um^2
szBWc=size(BWc);%# of pixels of the source image
unitXYDim=317.13/1024;%=0.3097 um
unitZ2XY=round(sliceThickness/unitXYDim);%The ratio of Z to XY dimension
unitZDim=1.13/unitZ2XY;%=0.2825 um

cellMaxXArea=0;%Maximum cross sectional area of the cell [um^2]
cellVolume=0;%Cell volume [um^3]
```

```
sliceCounter=0;%Cell slice number which has the highest Xsection area
for ij=1:Ns
  Im = I(:,:,ij);
  BWsr(:,:,ij)=(Im>0).*BWc;
  % Discard small objects containing fewer than %0.5 pixels
  numPixDiscard=round(sum(BWc(:))*0.005);
  BWsr(:,:,ij) = bwareaopen(BWsr(:,:,ij),numPixDiscard);
   %Measure the cross sectional area of the cell
   cellXArea=FOV*sum(sum(BWsr(:,:,ij)))/(szBWc(1,1)*szBWc(1,2));
     if cellXArea>cellMaxXArea
      cellMaxXArea=cellXArea;
    end
   %Measure cell volume
   cellVolume=cellVolume+sliceThickness*cellXArea;
end
%% Discard slices which do not have cell pixels
j1=1;
for i1=1:Ns
 if sum(sum(BWsr(:,:,i1)))~=0 %if there is cell in the i1th slice
 %figure, imshow(BWsr(:,:,i1));
  BWcell3d(:,:,j1)=logical(BWsr(:,:,i1));
  j1=j1+1;
 end
end

%% Discard the background pixels in BWcell3d
d1=size(BWcell3d);
zIMcell3d=d1(1,3);
exIm=5;%Pad 5 pixels to the corners
IMcell3d=false(Imw+2*exIm,Imw+2*exIm,zIMcell3d);
for i1=1:zIMcell3d
IMcell3d(exIm+1:Imwy+exIm,exIm+1:Imwx+exIm,i1)=BWcell3d(min(y):min(y)+Imwy-
1,min(x):min(x)+Imwx-1,i1);
end

%% Upsample using interpolation
[dx,dy,dz]=size(IMcell3d);
xr=floor(dx/2); yr=floor(dy/2); zr=floor(dz/2);
%Construct the current grid
if mod(dx,2)==0
[X,Y,Z]=meshgrid(-xr:xr-1,-yr:yr-1,-zr:zr);%Cube centers
else
[X,Y,Z]=meshgrid(-xr:xr,-yr:yr,-zr:zr);
end

%Check if size(X)==size(IMcell3Dbin) else pad zero plane
szx=size(X);
if szx(1,3)>dz
    IMcell3d(:,:,dz+1)=zeros(dx,dy);
end
[dx,dy,dz]=size(IMcell3d);
```

```
%Construct the interpolation grid
[xi,yi,zi] = meshgrid(-xr:0.5:xr,-yr:0.5:yr,-zr:0.125:zr);
vi = interp3(X,Y,Z,IMcell3d,xi,yi,zi,'cubic');
vi=(vi>0.2);
% Following lines skipped to reduce VSI calculation time
% x1=-xr-0.25:0.5:xr+0.25;%Cube corners in x
% y1=x1;
% z1=-zr-0.0625:0.125:zr+0.0625;%Cube corners in z
% C1=zeros(size(xi));
% plotCubes(x1,y1,z1,vi,C1);
%
% szVi=size(vi);
% BPIxt=max(vi,[],1);
% BPIxz=reshape(BPIxt,[szVi(1,1),szVi(1,3)]);
% BPIyt=max(vi,[],2);
% BPIyz=reshape(BPIyt,[szVi(1,1),szVi(1,3)]);
% %Put a black frame
% szBPIxz=size(BPIxz);
% BPIxzf=zeros(szBPIxz(1,1),szBPIxz(1,2)+2);
% BPIxzf(:,2:end-1)=BPIxz;
% BPIxzf=imrotate(BPIxzf,90);
% %figure;imshow(BPIxzf);title('BPIxz');
%
% szBPIyz=size(BPIyz);
% BPIyzf=zeros(szBPIyz(1,1),szBPIyz(1,2)+2);
% BPIyzf(:,2:end-1)=BPIyz;
% BPIyzf=imrotate(BPIyzf,90);
% %figure;imshow(BPIyzf);title('BPIyz');
%
% %% Show 2D cell image
% BPIxy=max(vi,[],3);
% %figure;imshow(BPIxy);title('BPIxy');

%% Calculate cell VSI
VSI=fn_vsi(vi);

%% Calculate CSI-V
CSI=fn_csi_v(vi);

%% Write data to a cell
szVi=size(vi);
Nz=szVi(1,3);
cellInfo50(ic1,1:9)={fld fname BWc cellVolume cellMaxXArea VSI CSI Nz Ns};
ic1
%close
%% Clear variables
clear A* B* C* I* M* X Y Z Px d* vi x* y* z*;
end

%% Save input&output data to a an excel file
```

```
header={'Folder_name','file_names','cellMask','cellVolume
[um^3]','cellMaxXsArea[um^2]','VSI3','CSI','#cell_z_series','#source_z_series'};
xlswrite('vsi3Meas8jun17',header,'rPLG24h');
xlswrite('vsi3Meas8jun17',cellInfo50,'rPLG24h','A2');
```

S1.5 Plot cubes function

```
function plotCubes(x,y,z,V,C)
%makes 3D plot of shape defined by logical 3d-matrix V, with the color
%the small cubes given by the similar shaped matrix C
%x, y and z are vectors indicating the edge locations, so their lengths
%should be one larger than the corresponding dimension of the matrices

% Reference:
% http://www.mathworks.com/matlabcentral/newsreader/view_thread/236226
% Subject: 3D plot of solid object built from tiny cubes
% Accessed on: 03.05.2016
% Author: Bas


[nx,ny,nz] = size(V);

if length(x) ~= nx + 1 | length(y) ~= ny + 1 | length(z) ~= nz + 1
   error('Length of x, y and z must be one larger than size of V and C')
end

dc = 0.02 * (max(C(V)) - min(C(V))); %small color difference to create 3D effect

%-X
ilin = find(cat(1, V(1,:,:), ~V(1:end-1,:,:) & V(2:end,:,:)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = repmat(x(ix), 4, 1);
py = [y(iy); y(iy); y(iy+1); y(iy+1)];
pz = [z(iz); z(iz+1); z(iz+1); z(iz)];
c = C(ilin);
%-Y
ilin = find(cat(2, V(:,1,:), ~V(:,1:end-1,:) & V(:,2:end,:)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = [px, [x(ix); x(ix); x(ix+1); x(ix+1)]];
py = [py, repmat(y(iy), 4, 1)];
pz = [pz, [z(iz); z(iz+1); z(iz+1); z(iz)]];
c = [c, C(ilin) + dc];
%-Z
ilin = find(cat(3, V(:,:,1), ~V(:,:,1:end-1) & V(:,:,2:end)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = [px, [x(ix); x(ix); x(ix+1); x(ix+1)]];
py = [py, [y(iy); y(iy+1); y(iy+1); y(iy)]];
pz = [pz, repmat(z(iz), 4, 1)];
c = [c, C(ilin) - dc];
%+X
ilin = find(cat(1, V(1:end-1,:,:) & ~V(2:end,:,:), V(end,:,:)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = [px, repmat(x(ix+1), 4, 1)];
py = [py, [y(iy); y(iy); y(iy+1); y(iy+1)]];
pz = [pz, [z(iz); z(iz+1); z(iz+1); z(iz)]];
c = [c, C(ilin)];
```

```
%+Y
ilin = find(cat(2, V(:,1:end-1,:) & ~V(:,2:end,:), V(:,end,:)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = [px, [x(ix); x(ix); x(ix+1); x(ix+1)]];
py = [py, repmat(y(iy+1), 4, 1)];
pz = [pz, [z(iz); z(iz+1); z(iz+1); z(iz)]];
c = [c, C(ilin) + dc];
%+Z
ilin = find(cat(3, V(:,:,1:end-1) & ~V(:,:,2:end), V(:,:,end)))';
[ix,iy,iz] = ind2sub([nx,ny,nz], ilin);
px = [px, [x(ix); x(ix); x(ix+1); x(ix+1)]];
py = [py, [y(iy); y(iy+1); y(iy+1); y(iy)]];
pz = [pz, repmat(z(iz+1), 4, 1)];
c = [c, C(ilin) - dc];

figure;
patch(px,py,pz,c)
%patch(px,py,pz)
axis equal vis3d
%shading flat;
shading faceted; %dc can be set to zero when using faceted

end
```

S2. Two-way ANOVA

Dependent variable: *VSI*

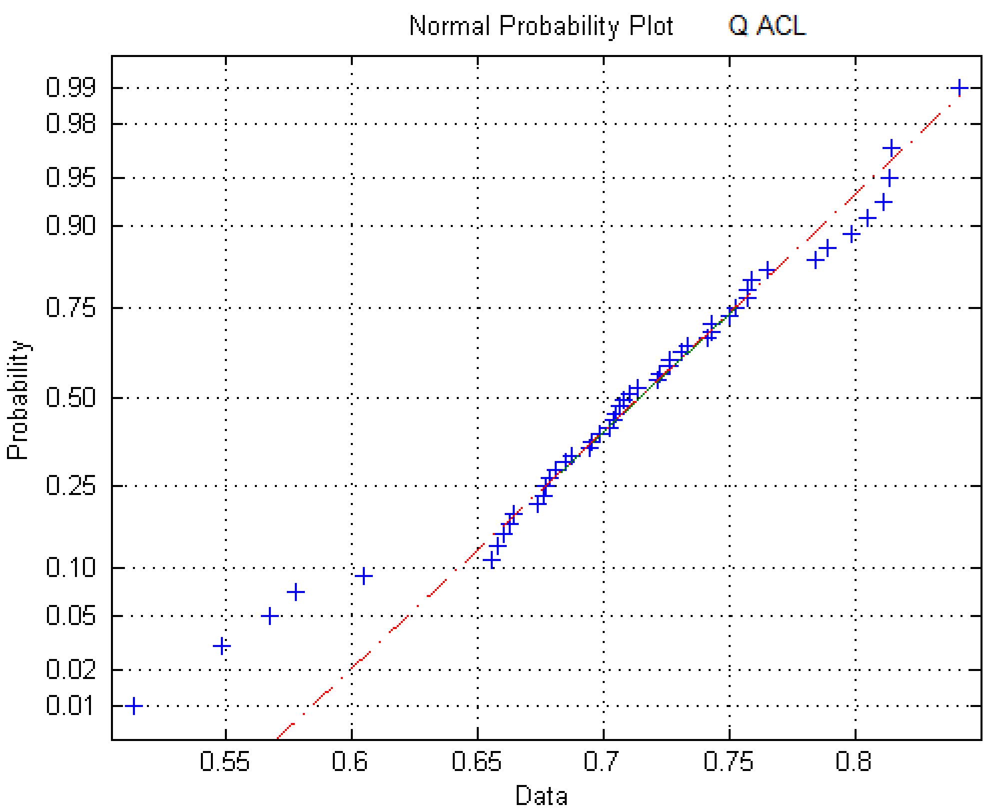
Two independent variables: Cell culture material type and cell immunoreactivity level

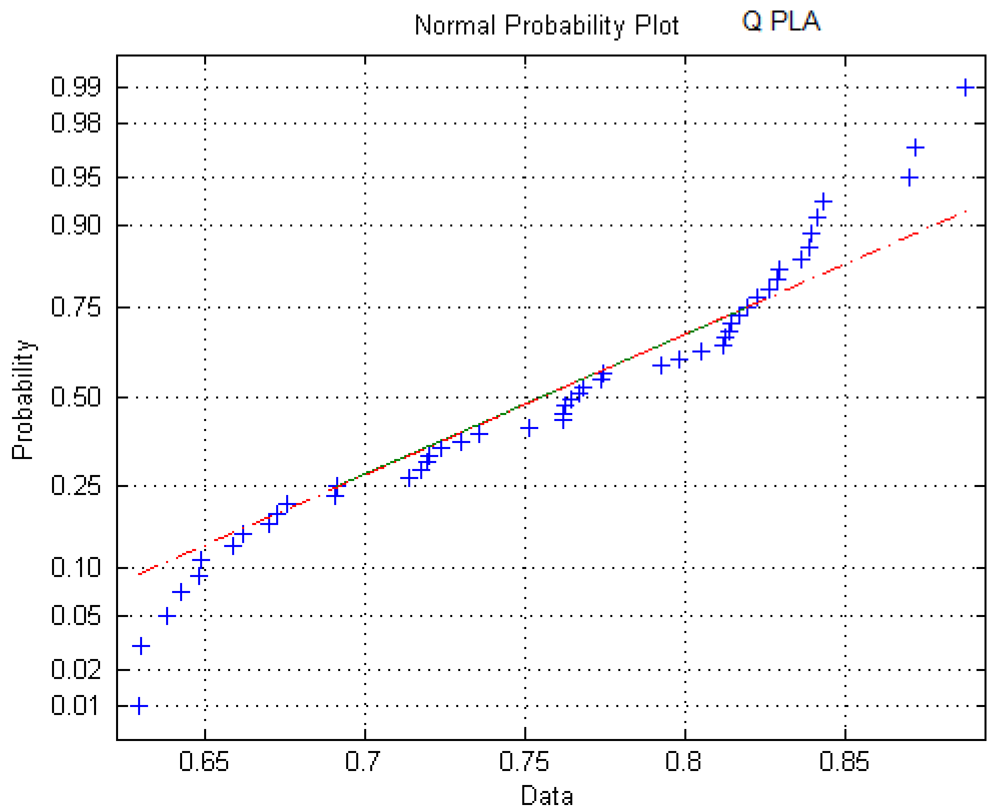
Replication:50

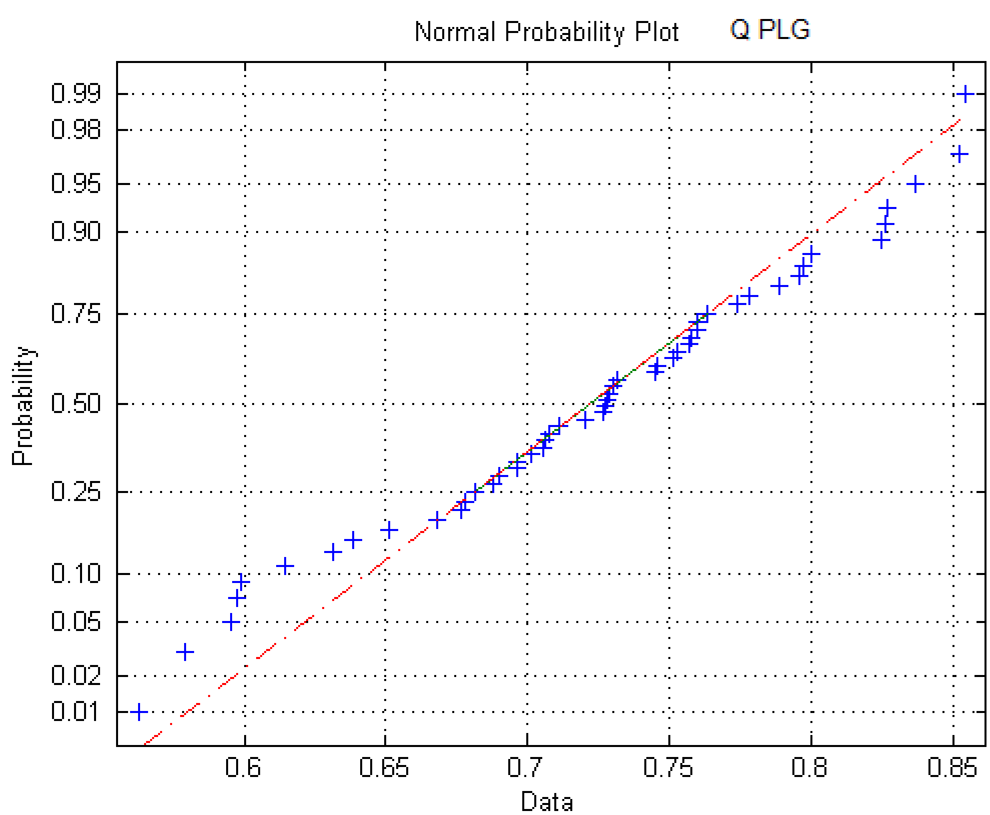
Number of groups: 8

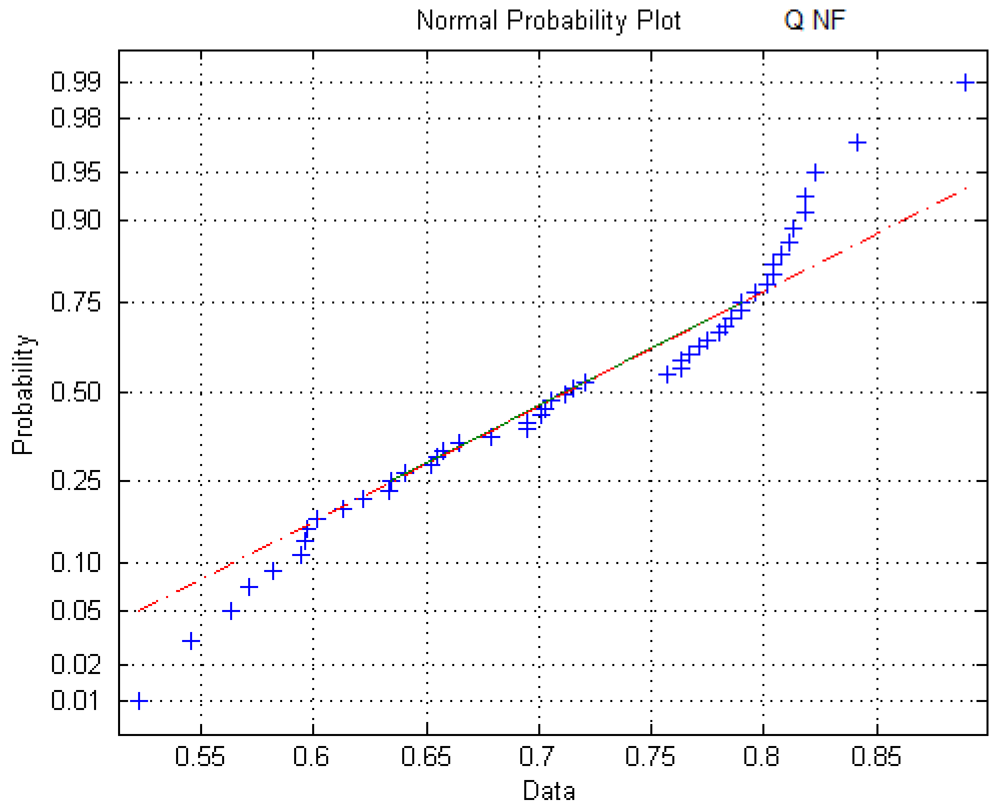
1. Each sample is an independent random sample.

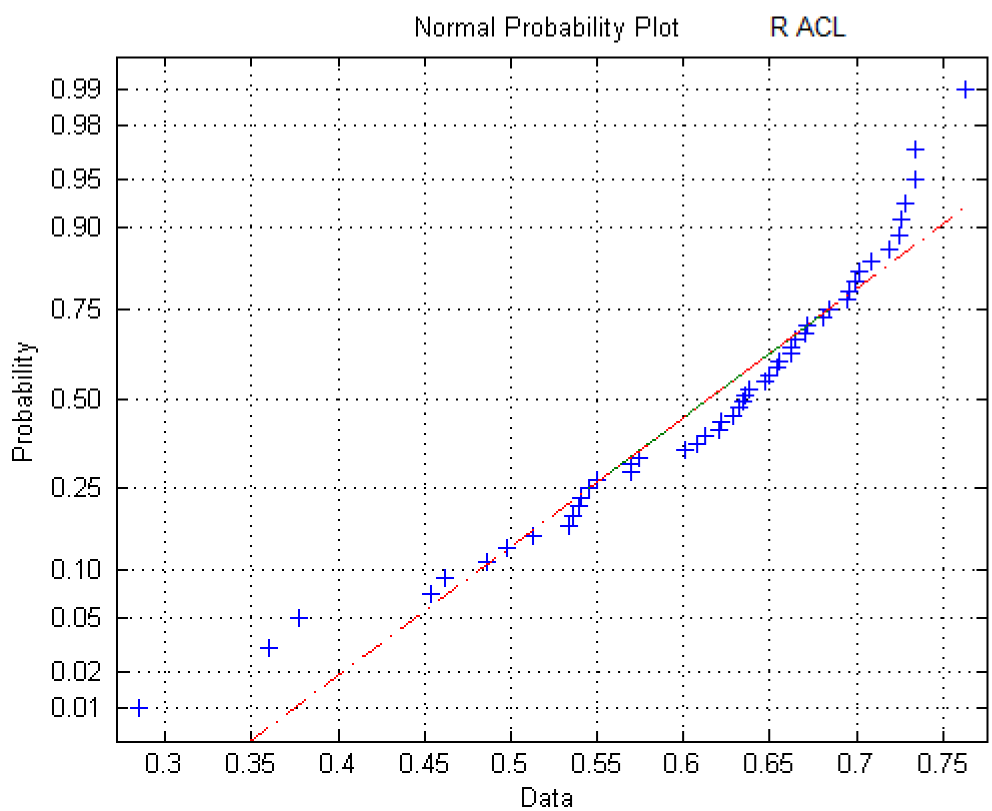
2. The distribution of the response variable follows a normal distribution. The normal probability plots of *VSI*s of quiescent- and reactive-like astrocytes (Q and R) cultured on Aclar (ACL), PLL Aclar (PLA), PLL glass(PLG) and nanofibrillar scaffolds(NF) are:

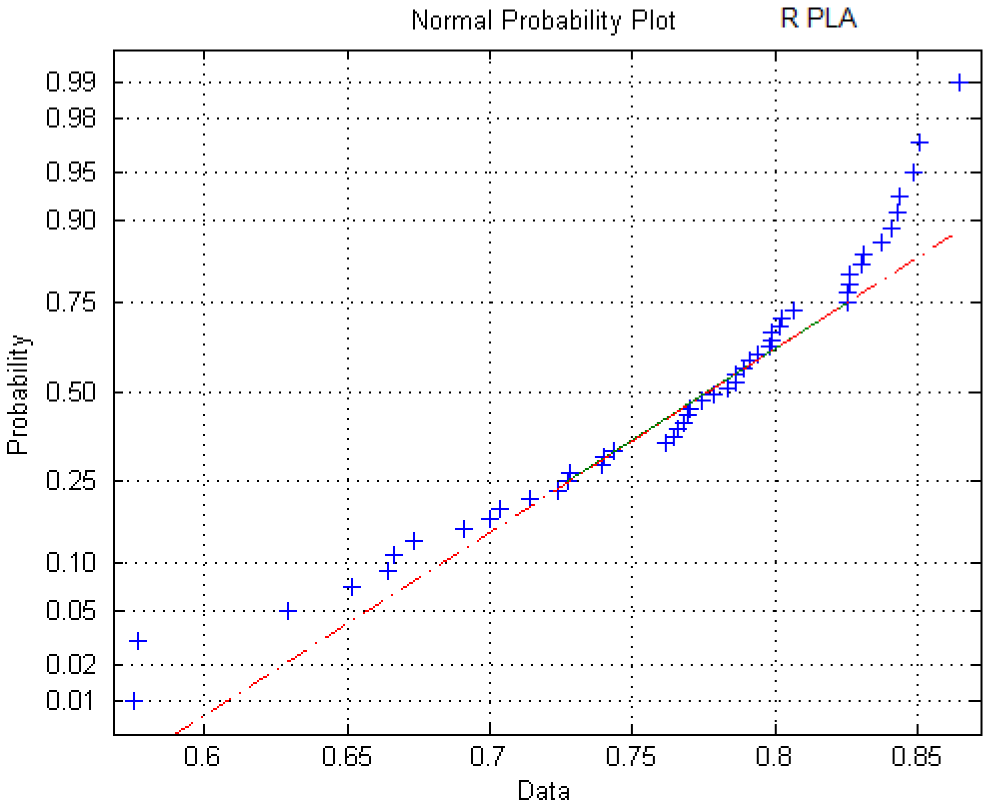


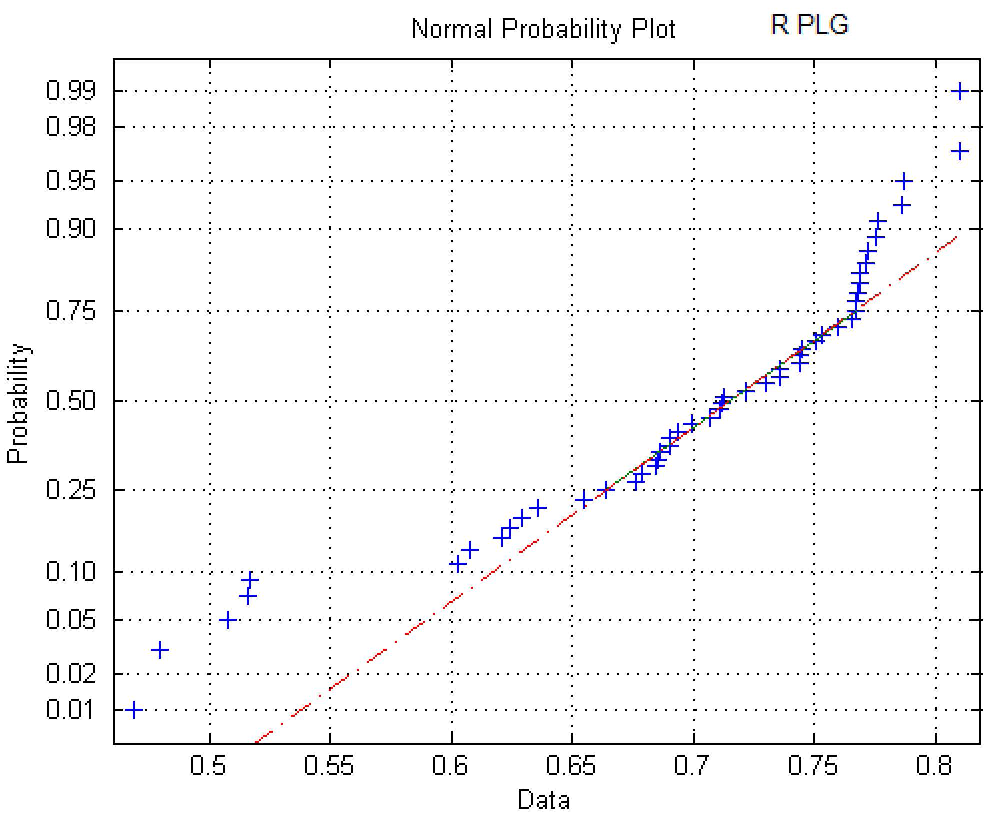

Normal Probability Plot Q PLA
Probability
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.65
0.7
0.75
0.8
0.85
Data

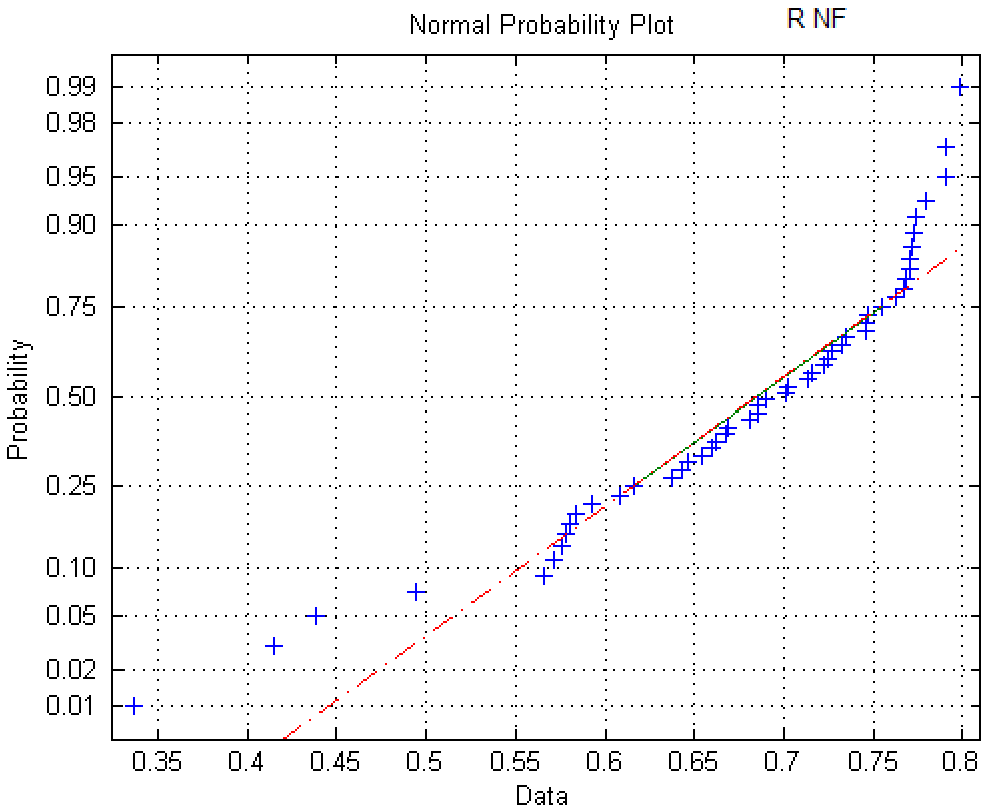

Normal Probability Plot Q PLG
Probability
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.6
0.65
0.7
0.75
0.8
0.85
Data


Normal Probability Plot Q NF
Probability
Data
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.55
0.6
0.65
0.7
0.75
0.8
0.85


Normal Probability Plot R ACL
Probability
Data
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
0.7
0.75


Normal Probability Plot
R PLA
Probability
Data
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.6
0.65
0.7
0.75
0.8
0.85


Normal Probability Plot
R PLG
Probability
Data
0.99
0.98
0.95
0.90
0.75
0.50
0.25
0.10
0.05
0.02
0.01
0.5
0.55
0.6
0.65
0.7
0.75
0.8



3. The largest sample standard deviation divided by the smallest sample standard deviation is 1.512, so the population variances are equal across responses for the group levels.

A 4(Cell culture material type: PLL glass, Aclar, PLL Aclar, nanofibrillar scaffolds) x 2(immunoreactivity level: quiescent-like vs. reactive-like) between subjects ANOVA was conducted to study *VSI* differences between material type as a function of immunoreactivity level.

Two-way ANOVA Hypotheses

Hypothesis 1

H_0: Astrocyte reactivity level will have no significant effect on *VSI*.

Hypothesis 2

H_0: Cell culture material type will have no significant effect on *VSI*.

Hypothesis 3

H_0: Material type and astrocyte reactivity level interaction will have no significant effect on *VSI*.

Descriptive Statistics

Dependent Variable: VSI

Material type	Reactivity	Mean	Std. Deviation	N
Aclar	Quiescent-like	.7100	.06837	50
	Reactive-like	.6110	.10335	50
	Total	.6605	.10036	100
PLL Aclar	Quiescent-like	.7583	.07332	50
	Reactive-like	.7647	.06942	50
	Total	.7615	.07111	100
PLL Glass	Quiescent-like	.7224	.07223	50
	Reactive-like	.6971	.08548	50
	Total	.7098	.07975	100
Nanofibrillar scaffolds	Quiescent-like	.7126	.09191	50
	Reactive-like	.6745	.10242	50
	Total	.6935	.09869	100
Total	Quiescent-like	.7258	.07883	200
	Reactive-like	.6868	.10597	200
	Total	.7063	.09530	400

Tests of Between-Subjects Effects

Dependent Variable: VSI

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	.830[a]	7	.119	16.635	.000
Intercept	199.558	1	199.558	28002.142	.000
Material_type	.532	3	.177	24.874	.000
Reactivity	.152	1	.152	21.322	.000
Material_type * Reactivity	.146	3	.049	6.834	.000
Error	2.794	392	.007		
Total	203.182	400			
Corrected Total	3.623	399			

All null hypotheses were rejected because all *P* values are less than 0.05. Both reactivity level (Q/R) and material type (acl/nf/pla/plg) have significant effects on *VSI*. Material type and reactivity level interaction has a significant effect on *VSI*.

Pairwise Comparisons

Dependent Variable: VSI

(I) Material type	(J) Material type	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]
					Lower Bound
Aclar	PLL Aclar	-.101*	.012	.000	-.124
	PLL Glass	-.049*	.012	.000	-.073
	Nanofibrillar scaffolds	-.033*	.012	.006	-.056
PLL Aclar	Aclar	.101*	.012	.000	.078
	PLL Glass	.052*	.012	.000	.028
	Nanofibrillar scaffolds	.068*	.012	.000	.045
PLL Glass	Aclar	.049*	.012	.000	.026
	PLL Aclar	-.052*	.012	.000	-.075
	Nanofibrillar scaffolds	.016	.012	.174	-.007
Nanofibrillar scaffolds	Aclar	.033*	.012	.006	.010
	PLL Aclar	-.068*	.012	.000	-.091
	PLL Glass	-.016	.012	.174	-.040

Pairwise Comparisons

Dependent Variable: VSI

(I) Reactivity	(J) Reactivity	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Quiescent-like	Reactive-like	.039*	.008	.000	.022	.056
Reactive-like	Quiescent-like	-.039*	.008	.000	-.056	-.022

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Pairwise Comparisons

Dependent Variable: VSI

Reactivity	(I) Material type	(J) Material type	Mean Difference (I-J)	Std. Error	Sig.[b]
Quiescent-like	Aclar	PLL Aclar	-.048*	.017	.004
		PLL Glass	-.012	.017	.462
		Nanofibrillar scaffolds	-.003	.017	.880
	PLL Aclar	Aclar	.048*	.017	.004
		PLL Glass	.036*	.017	.034
		Nanofibrillar scaffolds	.046*	.017	.007
	PLL Glass	Aclar	.012	.017	.462
		PLL Aclar	-.036*	.017	.034
		Nanofibrillar scaffolds	.010	.017	.559
	Nanofibrillar scaffolds	Aclar	.003	.017	.880
		PLL Aclar	-.046*	.017	.007
		PLL Glass	-.010	.017	.559
Reactive-like	Aclar	PLL Aclar	-.154*	.017	.000
		PLL Glass	-.086*	.017	.000
		Nanofibrillar scaffolds	-.063*	.017	.000
	PLL Aclar	Aclar	.154*	.017	.000
		PLL Glass	.068*	.017	.000
		Nanofibrillar scaffolds	.090*	.017	.000
	PLL Glass	Aclar	.086*	.017	.000
		PLL Aclar	-.068*	.017	.000
		Nanofibrillar scaffolds	.023	.017	.181
	Nanofibrillar scaffolds	Aclar	.063*	.017	.000
		PLL Aclar	-.090*	.017	.000
		PLL Glass	-.023	.017	.181

Pairwise Comparisons

Dependent Variable: VSI

Material type	(I) Reactivity	(J) Reactivity	Mean Difference (I-J)	Std. Error	Sig.[b]
Aclar	Quiescent-like	Reactive-like	.099*	.017	.000
	Reactive-like	Quiescent-like	-.099*	.017	.000
PLL Aclar	Quiescent-like	Reactive-like	-.006	.017	.705
	Reactive-like	Quiescent-like	.006	.017	.705
PLL Glass	Quiescent-like	Reactive-like	.025	.017	.135
	Reactive-like	Quiescent-like	-.025	.017	.135
Nanofibrillar scaffolds	Quiescent-like	Reactive-like	.038*	.017	.025
	Reactive-like	Quiescent-like	-.038*	.017	.025

S3. *VSI* and *CSI-V* comparison of astrocytes using representative binary projection images

S3.1 Quiescent-like versus reactive-like astrocytes on Aclar

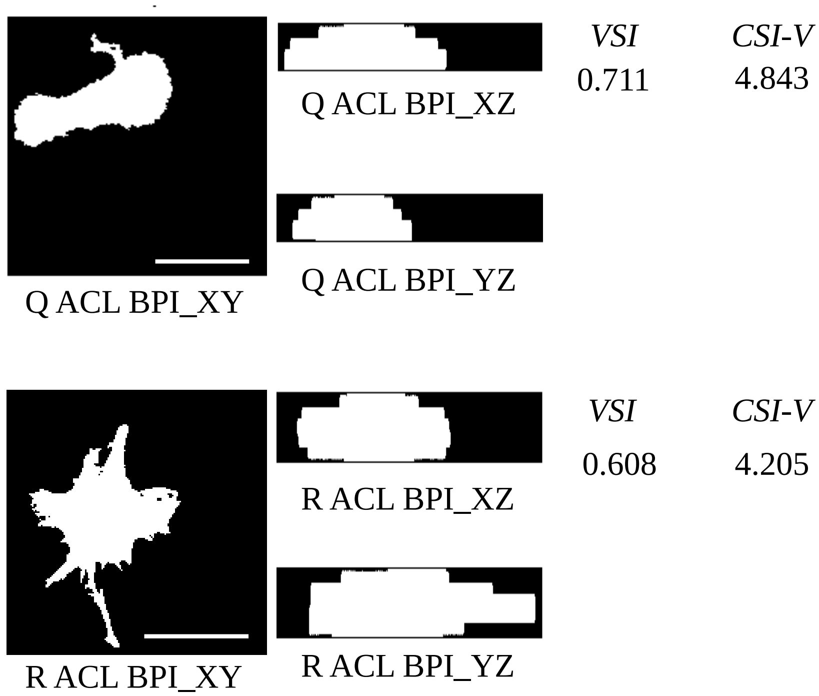


Figure 1. *VSI* and *CSI-V* comparison of quiescent- and reactive-like astrocytes cultured on Aclar. Quiescent-like astrocytes on Aclar are more spread than the reactive-like astrocytes on Aclar. *VSI* successfully quantifies the cell spreading. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; ACL stands for Aclar; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 μm.

S3.2 Quiescent-like versus reactive-like astrocytes on nanofibrillar scaffolds

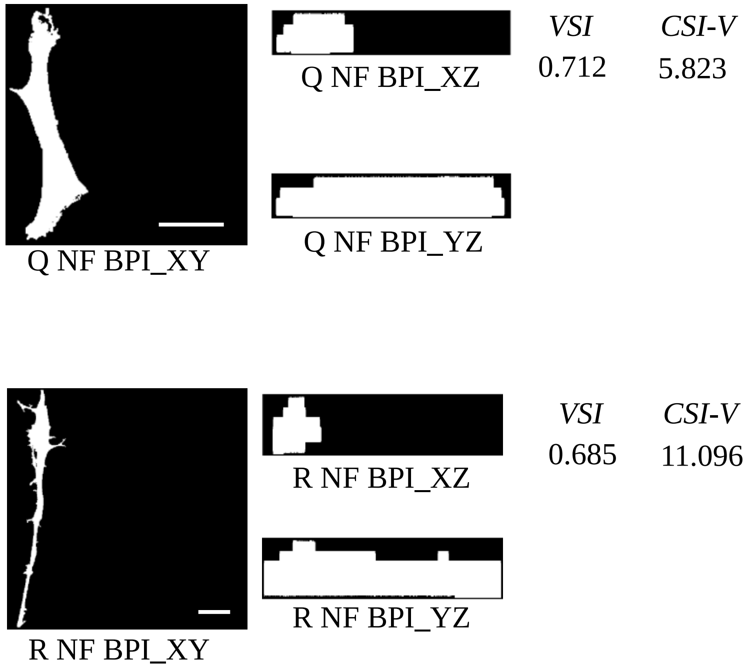
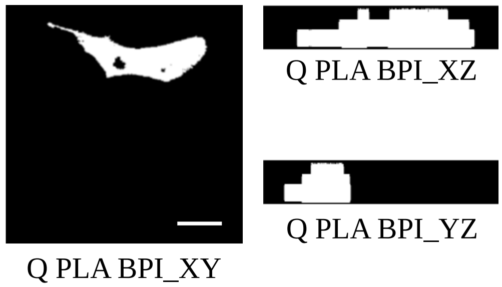
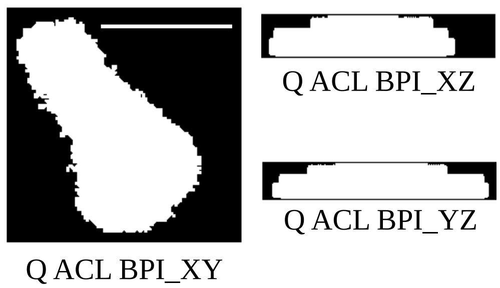


Figure 2. *VSI* and *CSI-V* comparison of quiescent- and reactive-like astrocytes cultured on nanofibrillar scaffolds. Quiescent-like astrocytes on nanofibrillar scaffolds are more spread than the reactive-like astrocytes. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; ACL stands for Aclar; PLA stands for PLL Aclar; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 μm.

S3.3 Quiescent-like astrocytes on PLL Aclar versus ones on Aclar



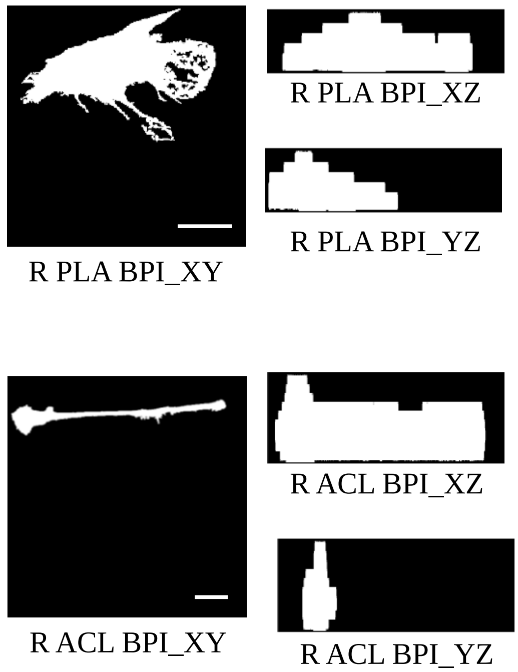
VSI	*CSI-V*
0.767	10.767



VSI	*CSI-V*
0.707	4.478

Figure 3. *VSI* and *CSI-V* comparison of quiescent-like astrocytes cultured on PLL Aclar and Aclar. Quiescent-like astrocytes on PLL Aclar are more spread than the ones on Aclar. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; ACL stands for Aclar; PLA stands for PLL Aclar; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 μm.

S3.4 Reactive-like astrocytes on PLL Aclar versus reactive-like astrocytes on Aclar



VSI	*CSI-V*
0.762	17.418

VSI	*CSI-V*
0.601	8.875

Figure 4. *VSI* and *CSI-V* comparison of reactive-like astrocytes cultured on PLL Aclar and Aclar. Reactive-like astrocytes on PLL Aclar are more spread than the ones on Aclar. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; ACL stands for Aclar; PLA stands for PLL Aclar; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 μm.

S3.5 Reactive-like astrocytes on PLL Glass versus reactive-like astrocytes on Aclar

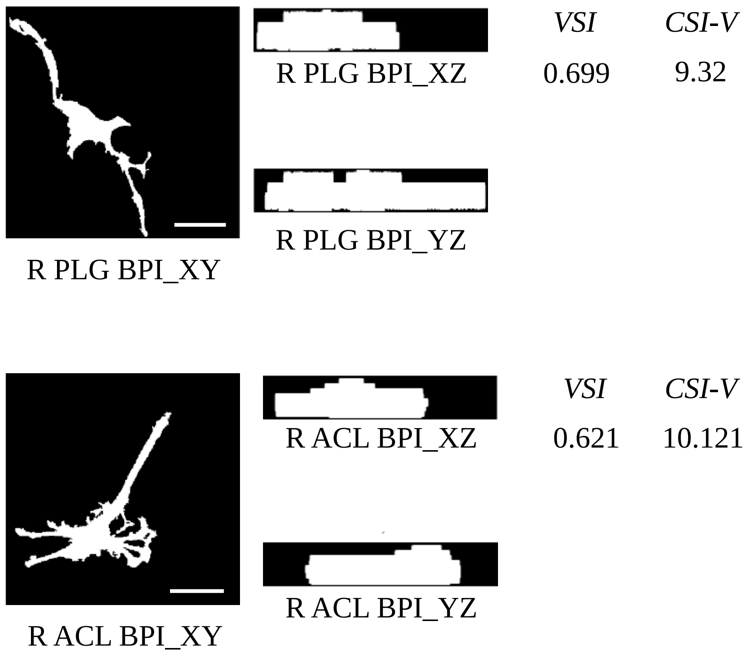


Figure 5. *VSI* and *CSI-V* comparison of reactive-like astrocytes cultured on PLL Glass and Aclar. Reactive-like astrocytes on PLL Glass are more spread than the ones on Aclar. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; ACL stands for Aclar; PLG stands for PLL Glass; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 μm.

S3.6 Reactive-like astrocytes on nanofibrillar scaffolds versus ones on PLL Aclar

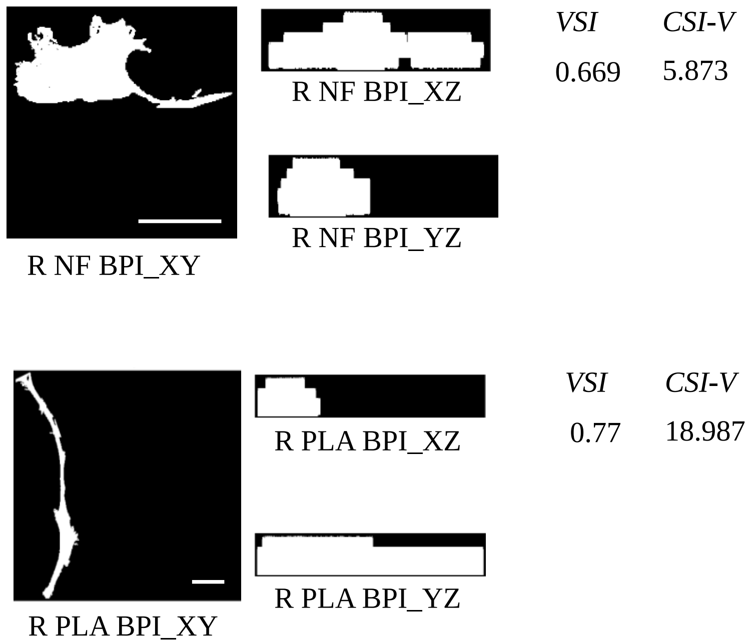


Figure 6. *VSI* and *CSI-V* comparison of reactive-like astrocytes cultured on nanofibrillar scaffolds and reactive-like astrocytes on Aclar. Reactive-like astrocytes on nanofibrillar scaffolds are less spread than the ones on PLL Aclar. R stands for reactive-like astrocyte; Q stands for quiescent-like astrocyte; PLA stands for PLL Aclar; NF stands for nanofibrillar scaffolds; BPI stands for binary projection image; XY, YZ, and XY stand for the image projection planes. Scale bars show 20 µm.