Table of Contents


```
clear;
home;
close all;
```

Load Toolboxes

the code requires MTEX version 5.11.2: https://mtex-toolbox.github.io/download (also works with 5.10.2)

```
%Toolboxes - folder locations
mtex_location='C:\Users\ruthb\OneDrive\Documents\MTEX\mtex-5.11.2'; % update
 this

%start mtex if needed
try EBSD;
catch
    run(fullfile(mtex_location,"startup.m"));
end
```

h5oina file location

```
% folder
file1_folder='C:\Users\ruthb\DocumentsOnLaptop\GitHub\MTEX_Workshop
\Data'; %location where the data is stored
% file
file1_name='Ti6246 Post_PECS Site 2 Map Data 4'; %should be a h5oina file, do
 not add in the .h5oina file extension
```

make a results folder if needed

```
%create a results folder
```

```
file1_name_us=file1_name;
file1_name_us(strfind(file1_name_us,' '))='_';

resultsdir=fullfile(cd,'results',file1_name_us);
if isdir(resultsdir) == 0; mkdir(resultsdir); end
```

load the h5oina file

```
% Make a full file name
file1_full=fullfile(file1_folder,[file1_name '.h5oina']);

% load one file
% h5oina_file=file1_full; % file name
warningOn=0; % turn on/off warnings during loading
[ebsd_original,dataset_header,ebsd_patternmatched,h5_original,h5_patternmatch]
 = load_h5oina_pm2(file1_name,file1_folder,warningOn);
ebsd=ebsd_patternmatched; % if not pattern matched it will default to original
 ebsd data
```

Ti6246 Post_PECS Site 2 Map Data 4 - ebsd data loaded

sort the phase names if needed

```
% updates
phase_names={'Ti Alpha','Ti Beta'}; %overwrite the phase names
phase_colors={'R','B'}; %overwrite the colors

% define the crystal symmetry manually
% to see the current values, look at e.g. ebsd(ebsd.phase==1).CS
CS1_new = [crystalSymmetry('6/mmm', [3.5, 3.5, 5.5],[90,90,120]*degree, 'X||
a*', 'Y||b', 'Z||c*', 'mineral', phase_names{1}, 'color', phase_colors{1})];
CS2_new = [crystalSymmetry('m-3m', [3.9, 3.9, 3.9],
[90,90,90]*degree, 'mineral', phase_names{2}, 'color', phase_colors{2})];

% make a new list of crystal symmetries
CS_updated={'notIndexed',CS1_new,CS2_new};

%update the crystal symmetry info
ebsd.CS=CS_updated;
```

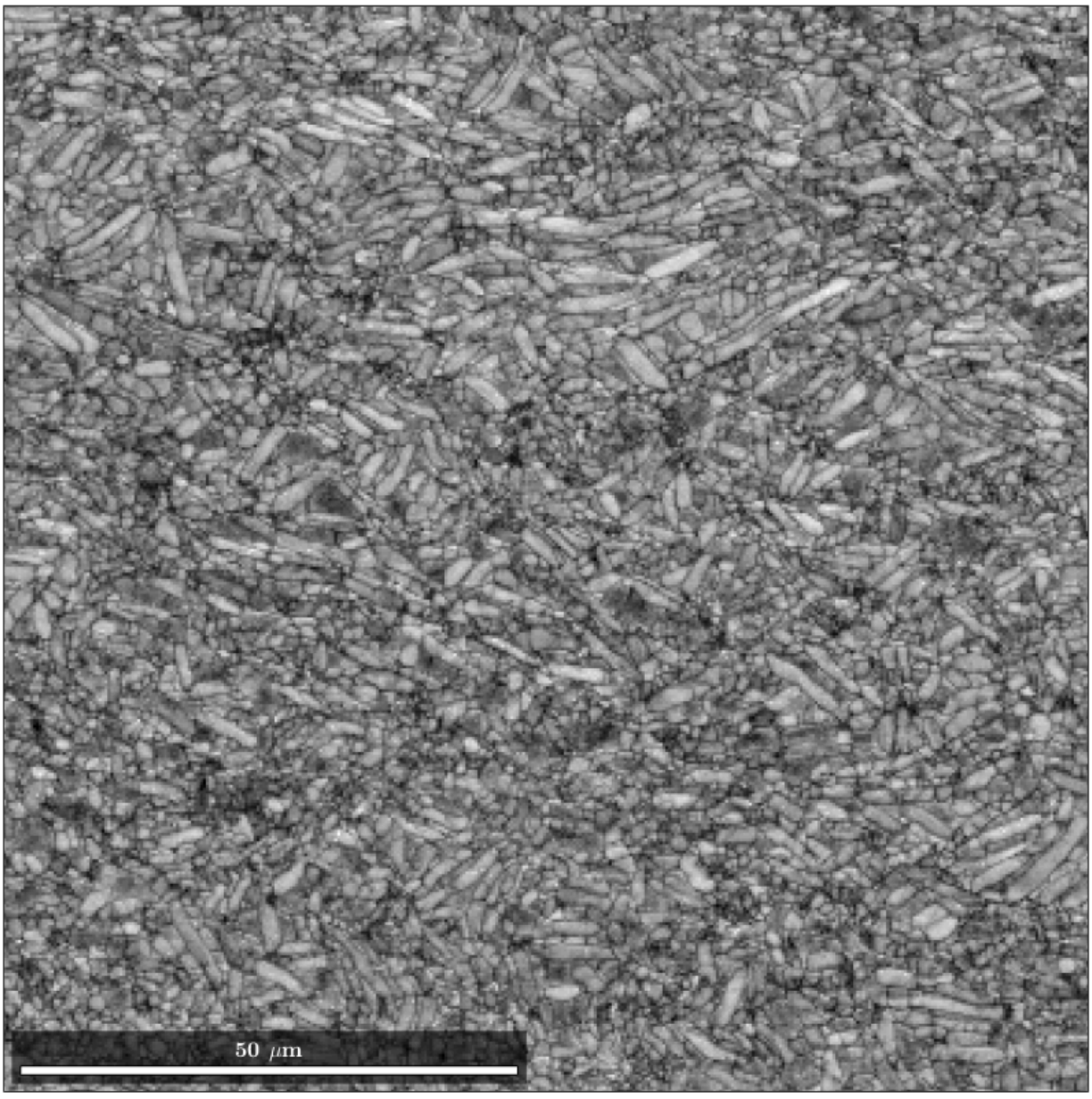
Set the plotting preferences - this has to be validated for your instrument

```
%this is set up for the pFIB and Oxford Insturments systems at UBC
setMTEXpref('xAxisDirection','east');          %aztec
setMTEXpref('zAxisDirection','outofPlane');    %aztec

%font size and overwrite the scale bar if you want
setMTEXpref('FontSize',12)
```

```
%extra (workshop)
file_dset='1'; %data set number of interest in the h5 file
mb_length = 50; %micro bar length for plots - if you want to override, you can
 comment this out/clear this variable and the override will not happen
```

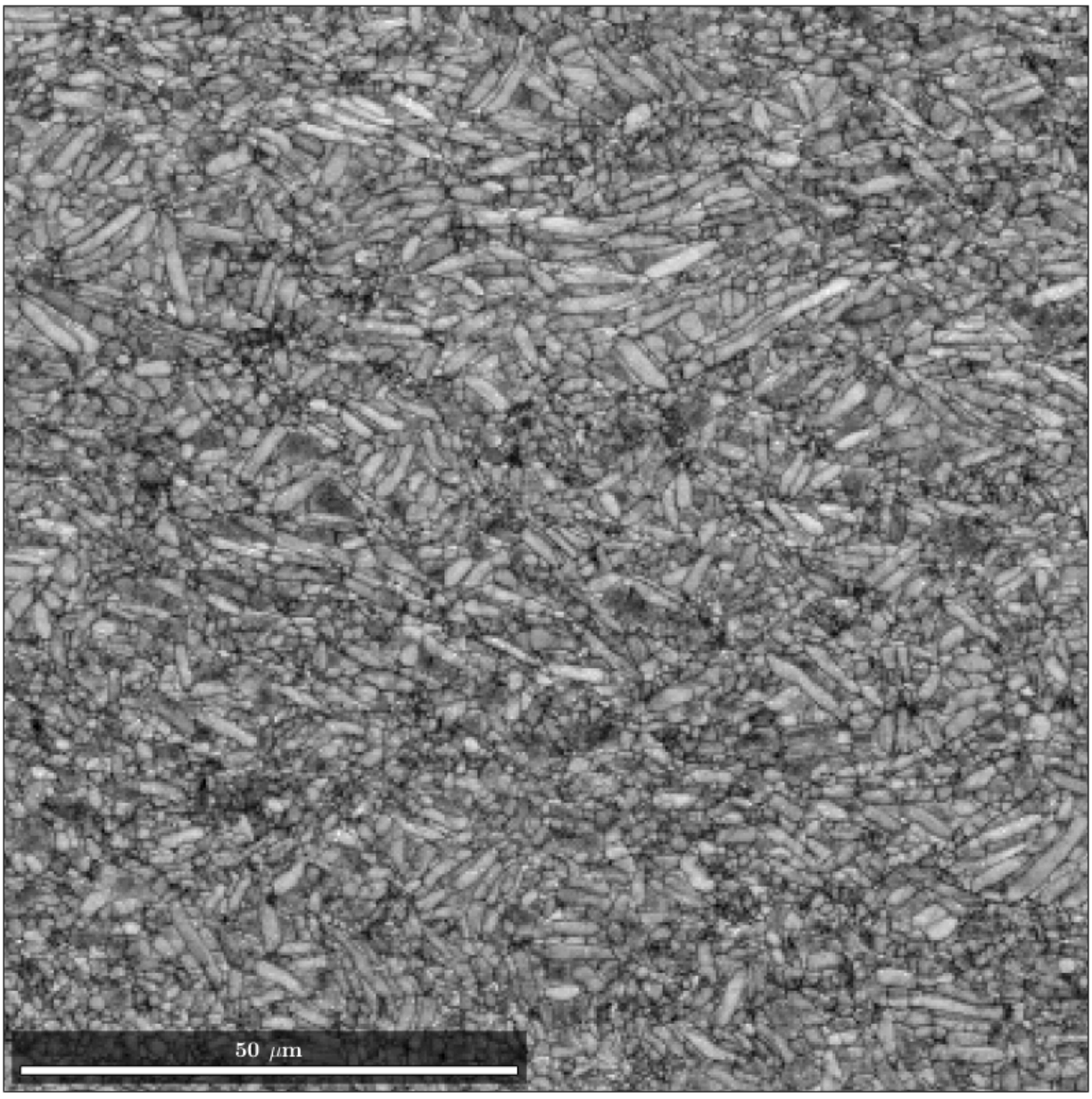
Now do some plots

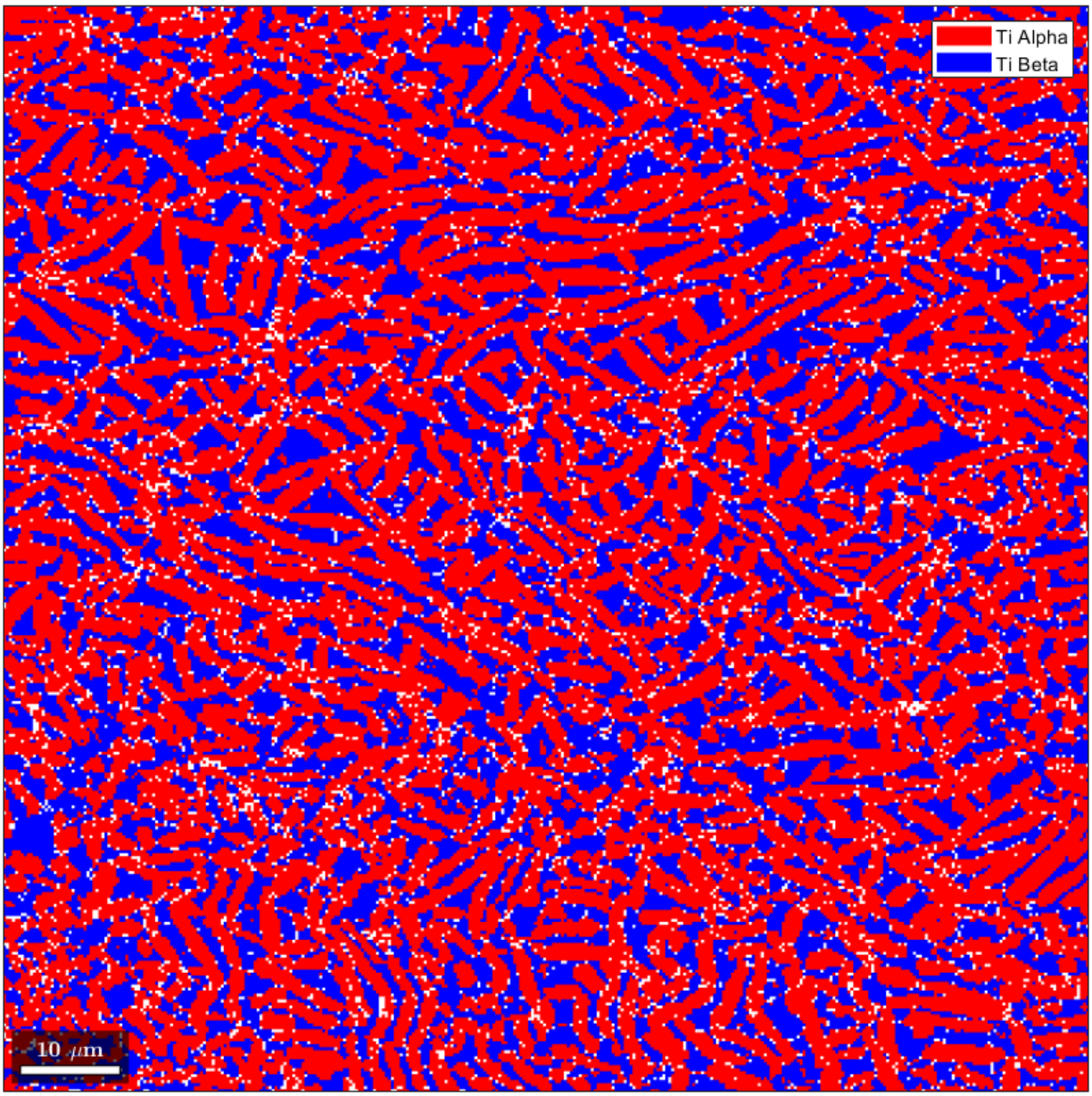
```
%band contrast
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');

if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
 micronbar length to make it prettier/easier to read
```

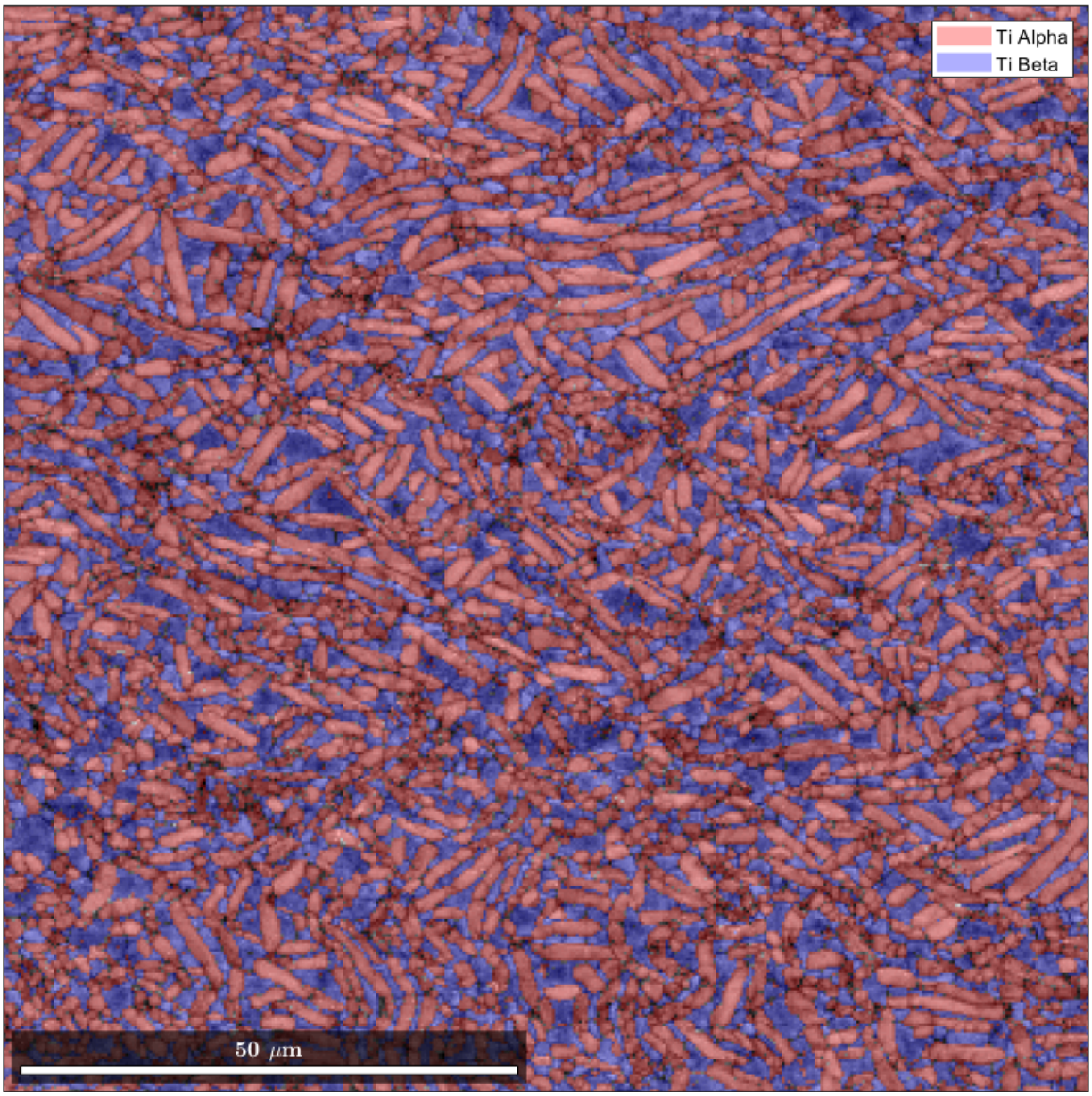


Plot the map of phases/indexed data

```
figure;
plot(ebsd');

%overlay with Band Contrast
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
hold on;
plot(ebsd,'FaceAlpha','0.3');
hold on;
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
 micronbar length to make it prettier/easier to read
```




Ti Alpha
Ti Beta
10 μm

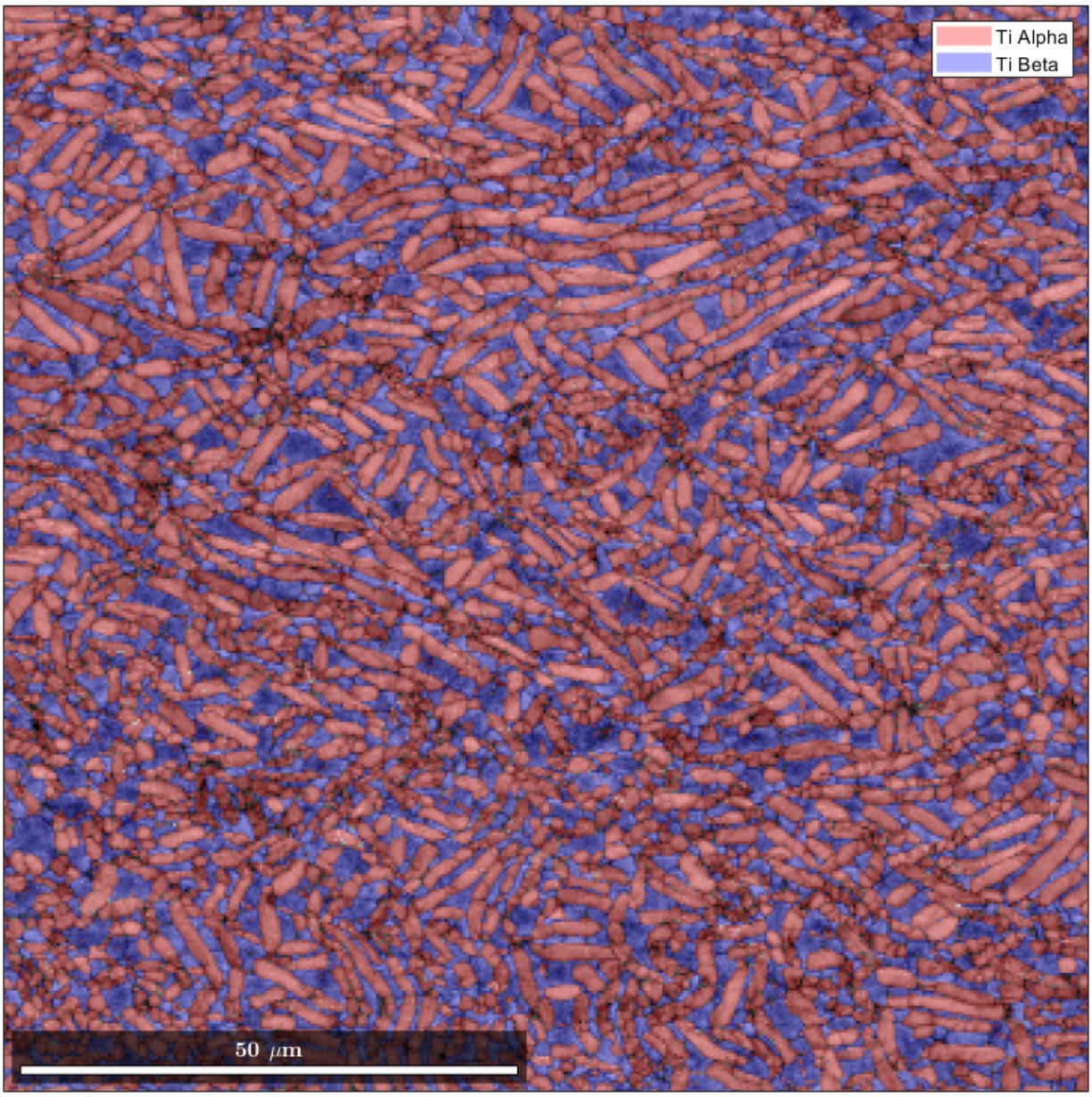


IPF-x, y and z

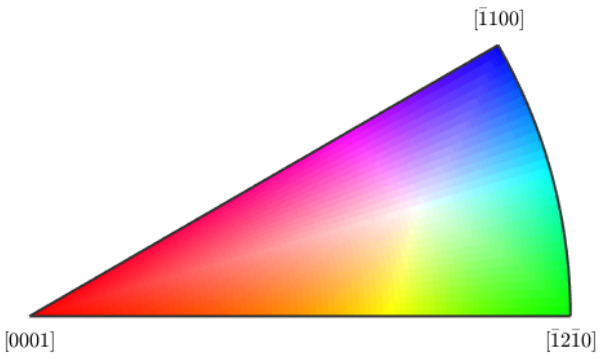
compute the colors

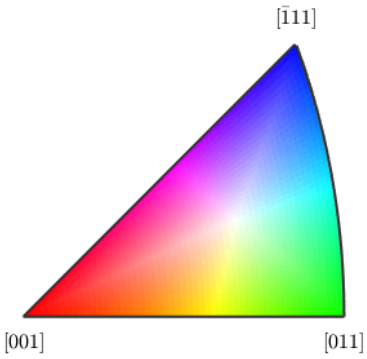
```
ipfKey1 = ipfTSLKey(ebsd(phase_names{1}));
ipfKey2 = ipfTSLKey(ebsd(phase_names{2}).CS);

%plot the key
f1=figure;
plot(ipfKey1)
nextAxis
plot(ipfKey2)
```

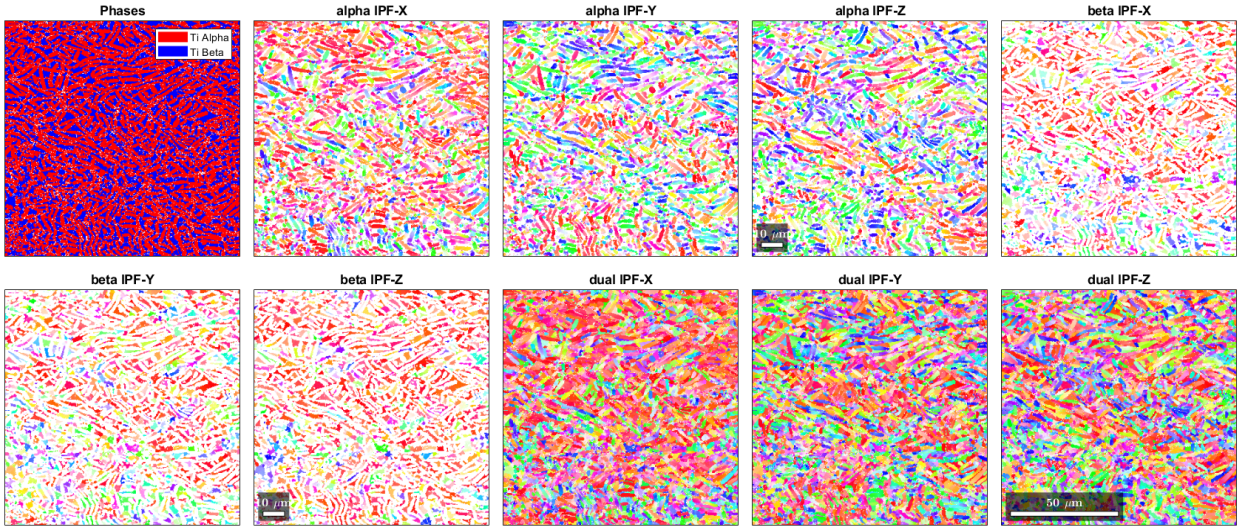
```
% now generate the IPF colour maps
ipfKey1.inversePoleFigureDirection = vector3d.X;
colors_X1 = ipfKey1.orientation2color(ebsd(phase_names{1}).orientations);
ipfKey1.inversePoleFigureDirection = vector3d.Y;
colors_Y1 = ipfKey1.orientation2color(ebsd(phase_names{1}).orientations);
ipfKey1.inversePoleFigureDirection = vector3d.Z;
colors_Z1 = ipfKey1.orientation2color(ebsd(phase_names{1}).orientations);

ipfKey2.inversePoleFigureDirection = vector3d.X;
colors_X2 = ipfKey2.orientation2color(ebsd(phase_names{2}).orientations);
ipfKey2.inversePoleFigureDirection = vector3d.Y;
colors_Y2 = ipfKey2.orientation2color(ebsd(phase_names{2}).orientations);
ipfKey2.inversePoleFigureDirection = vector3d.Z;
colors_Z2 = ipfKey2.orientation2color(ebsd(phase_names{2}).orientations);


%now plot the three IPF coloured maps
f2=figure;
plot(ebsd,'micronbar','off');
title('Phases')
nextAxis
plot(ebsd(phase_names{1}),colors_X1,'micronbar','off'); title('alpha IPF-X')
nextAxis
plot(ebsd(phase_names{1}),colors_Y1,'micronbar','off'); title('alpha IPF-Y')
nextAxis
plot(ebsd(phase_names{1}),colors_Z1,'micronbar','on'); title('alpha IPF-Z')

nextAxis
plot(ebsd(phase_names{2}),colors_X2,'micronbar','off'); title('beta IPF-X')
nextAxis
plot(ebsd(phase_names{2}),colors_Y2,'micronbar','off'); title('beta IPF-Y')
nextAxis
plot(ebsd(phase_names{2}),colors_Z2,'micronbar','on'); title('beta IPF-Z')

nextAxis
plot(ebsd(phase_names{1}),colors_X1,'micronbar','off'); hold on
plot(ebsd(phase_names{2}),colors_X2,'micronbar','off');  title('dual IPF-X')
nextAxis
plot(ebsd(phase_names{1}),colors_Y1,'micronbar','off'); hold on
plot(ebsd(phase_names{2}),colors_Y2,'micronbar','off'); title('dual IPF-Y')
nextAxis
plot(ebsd(phase_names{1}),colors_Z1,'micronbar','on'); hold on
plot(ebsd(phase_names{2}),colors_Y2,'micronbar','off'); title('dual IPF-Z')

if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
 micronbar length to make it prettier/easier to read

%save the figures;
drawnow(); %updates the graphics engine to make sure it saves things properly
% print(f1,fullfile(resultsdir,'Map_IPFkey.png'),'-dpng','-r600');
% print(f2,fullfile(resultsdir,'Map_IPF.png'),'-dpng','-r600');
```


Ti Alpha
Ti Beta
50 μm

m-3m


[$\bar{1}$100]
[0001]
[$\bar{1}2\bar{1}0$]

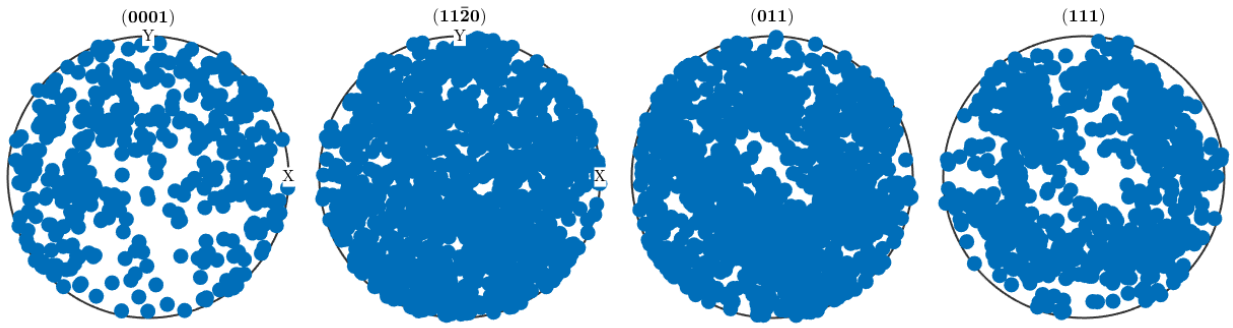

[$\bar{1}$11]
[001]
[011]



Plot the PF - scatter

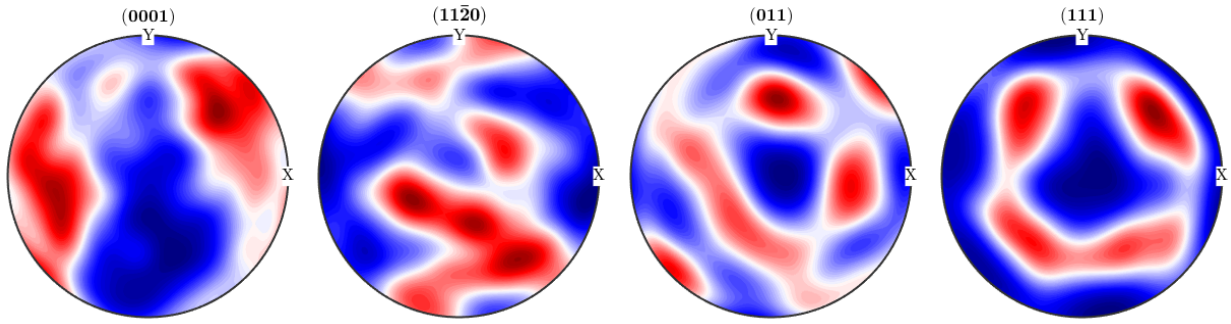
```
%plot planes as spots, and use a subset
figure
plotPDF(ebsd(phase_names{1}).orientations,
[Miller(0,0,1,ebsd(phase_names{1}).CS) Miller(1,1,0,ebsd(phase_names{1}).CS)])
nextAxis
plotPDF(ebsd(phase_names{2}).orientations,
[Miller(0,1,1,ebsd(phase_names{2}).CS) Miller(1,1,1,ebsd(phase_names{2}).CS)])
```

```
  I'm plotting 416 random orientations out of 80328 given orientations
  You can specify the the number points by the option "points".
  The option "all" ensures that all data are plotted
  I'm plotting 208 random orientations out of 44937 given orientations
  You can specify the the number points by the option "points".
  The option "all" ensures that all data are plotted
```



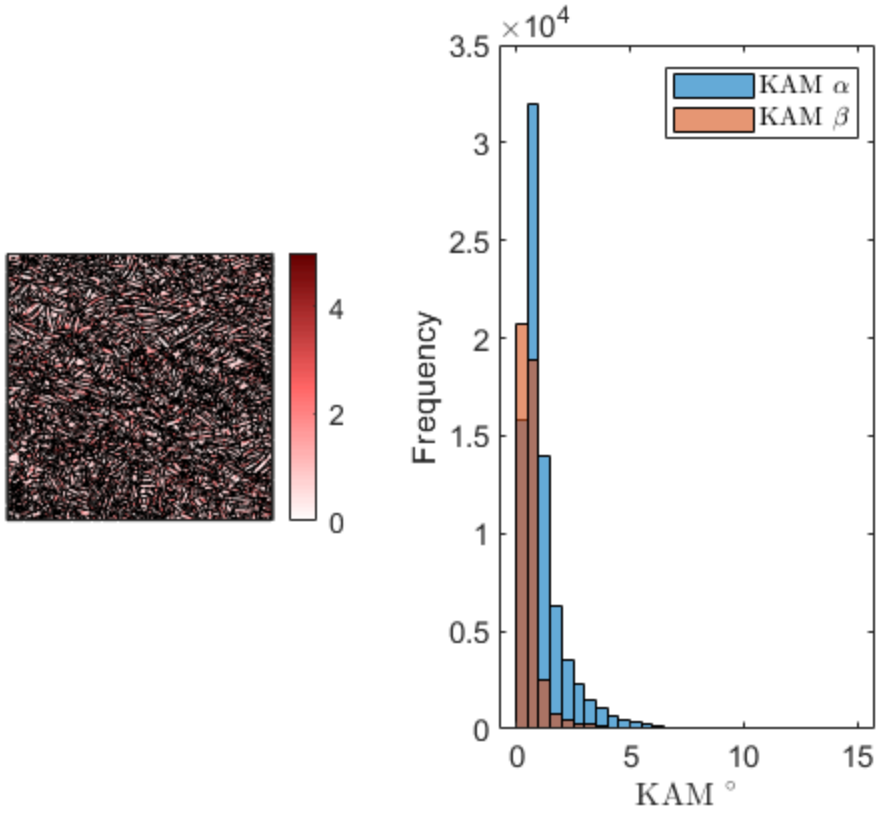
Construct the ODF

```
%suggested reading:
%https://mtex-toolbox.github.io/ODFTutorial.html
%https://mtex-toolbox.github.io/PoleFigure2ODF.html

% compute the ODF
odf1 = calcDensity(ebsd(phase_names{1}).orientations);
odf2 = calcDensity(ebsd(phase_names{2}).orientations);
% You could also change/fix the half width - compare this different ODF
% odf = calcDensity(ebsd(mineral_name).orientations,'halfwidth',2*degree);

figure
plotPDF(odf1,[Miller(0,0,1,ebsd(phase_names{1}).CS)
 Miller(1,1,0,ebsd(phase_names{1}).CS)])
mtexColorbar('location','southoutside')
mtexColorMap blue2red
nextAxis
plotPDF(odf2,[Miller(0,1,1,ebsd(phase_names{2}).CS)
 Miller(1,1,1,ebsd(phase_names{2}).CS)])
mtexColorbar('location','southoutside')
mtexColorMap blue2red
```



Plot the KAM

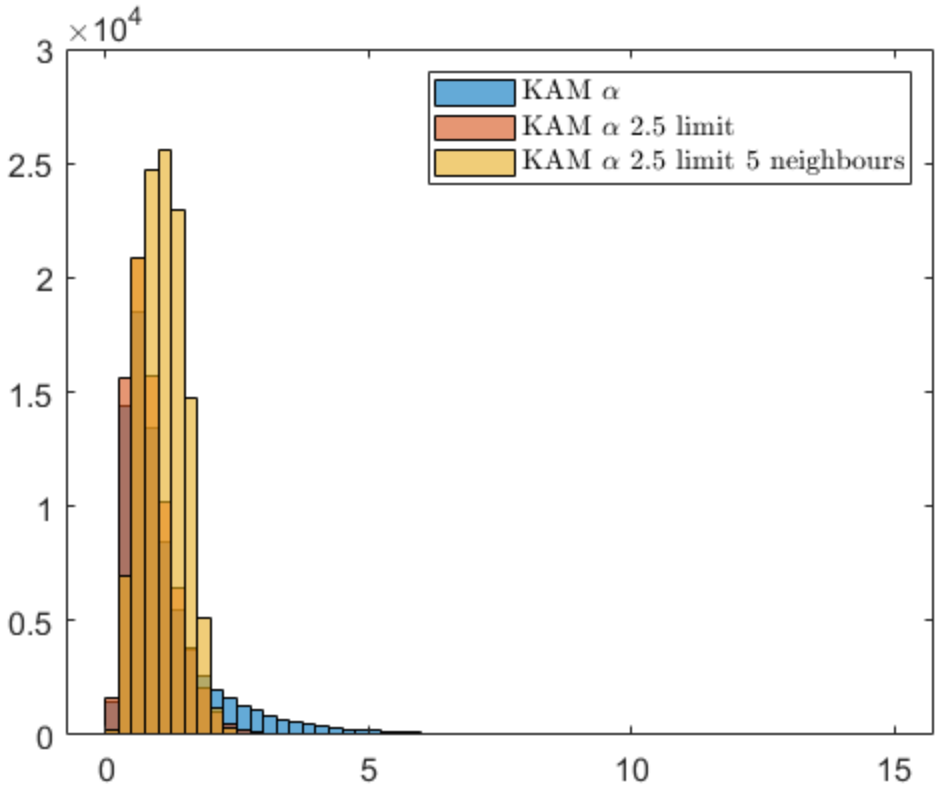
```
%see https://mtex-toolbox.github.io/EBSDKAM.html for more detail
% it is important to read this carefully, the KAM can be calculated in lots
% of different ways
ebsd=gridify(ebsd); %gridify the data
[grains,ebsd.grainId] = calcGrains(ebsd,'angle',10*degree);
%smooth the grains to be prettier
grains_pretty = smooth(grains,5);

kam_alpha = ebsd(phase_names{1}).KAM / degree;
kam_beta = ebsd(phase_names{2}).KAM / degree;

%plot the spatial map
f1=figure;
s1=subplot(1,2,1); %create a subplot axis
plot(ebsd(phase_names{1}),kam_alpha,'parent',s1);
hold on;
plot(ebsd(phase_names{2}),kam_beta,'parent',s1);
clim([0,15])
mtexColorbar
mtexColorMap LaboTeX
plot(grains_pretty.boundary,'lineWidth',0.5,'parent',s1)
hold off
%note that using subplot removes the micronbar!!!


KAM_bins=linspace(0,15,31); %use half degree bins - 31 between 0 and 15
s2=subplot(1,2,2);
histogram(kam_alpha,KAM_bins,'Parent',s2);
hold on
histogram(kam_beta,KAM_bins,'Parent',s2);

legend({'KAM $$\alpha$$','KAM $$\beta$$'},'interpreter','latex')
ylabel('Frequency')
xlabel('KAM $$^{\circ}$$','interpreter','latex')

% as we can see that this graph has a different color range - rescale
% figure 1 color axis
s1.CLim=[0 5];
```



Now we can play with KAM this metric has some nuance

see https://mtex-toolbox.github.io/EBSDKAM.html for a full explore

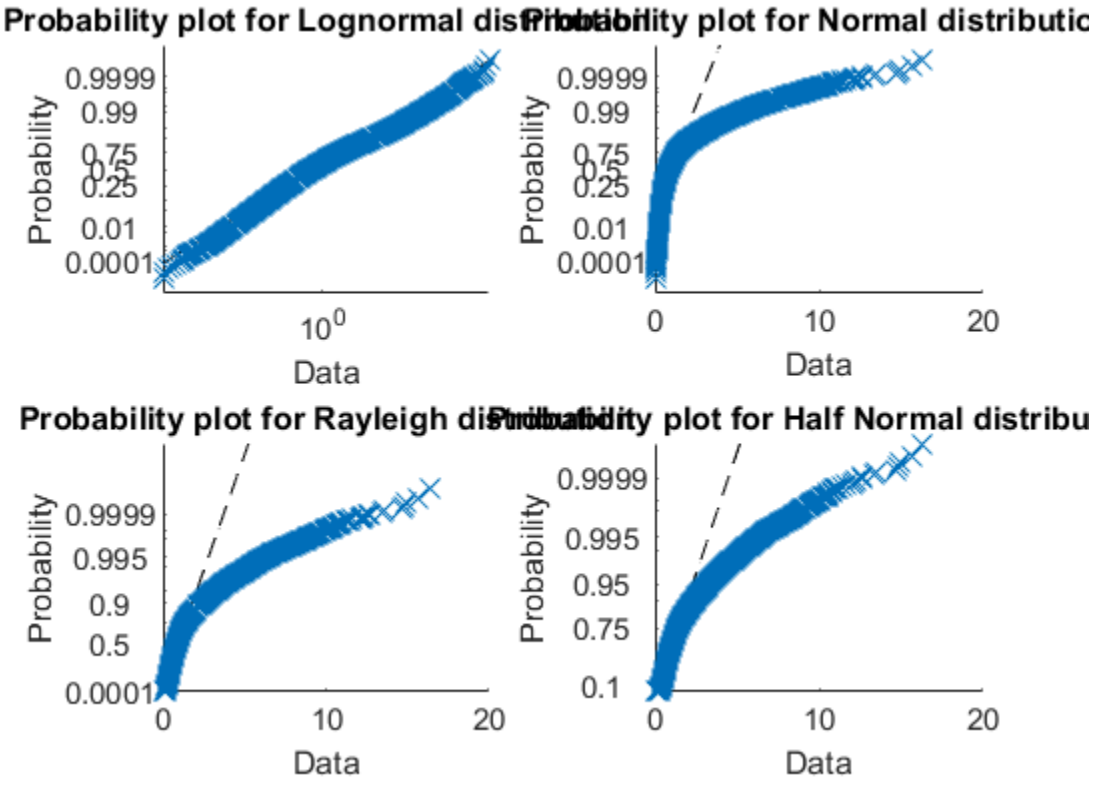
```
kam_alpha = ebsd(phase_names{1}).KAM / degree;
kam_alpha_3 = ebsd(phase_names{1}).KAM('threshold',3*degree) / degree;
kam_alpha_3_5neigh=ebsd.KAM('threshold',2.5*degree,'order',5) / degree;

KAM_bins=linspace(0,15,61); %use 0.25 degree bins - 61 between 0 and 15

figure;
histogram(kam_alpha,KAM_bins);
hold on
histogram(kam_alpha_3,KAM_bins);
histogram(kam_alpha_3_5neigh,KAM_bins);

legend({'KAM $$\alpha$$','KAM $$\alpha$$ 2.5 limit','KAM $$\alpha$$ 2.5 limit
 5 neighbours'},'interpreter','latex')
```



We can also test if this data fits a distribution

```
%use prob plot to test a few potential distribution types
%
% look at prob plot for what you can try

figure;
subplot(2,2,1);
probplot('lognormal',kam_alpha);

subplot(2,2,2);
probplot('normal',kam_alpha);

subplot(2,2,3);
probplot('rayleigh',kam_alpha);

subplot(2,2,4);
probplot('half normal',kam_alpha);

% if the measured data (blue) fits on the model data (dashed line), then
% this distribution could be a reasonable approximation for the data type
%
% there are quite a few theories about what this distribution *should* be
%
% see work by Wolfgang Pantleon, as well as work by Jiang, Britton &
```

```
% Wilkinson
```



Further ideas

```
%you could extract one or more grains from the alpha phase and look at the
%orientations of the nearby beta

%you could measure the grain size or aspect ratio
%you could estimate the volume fraction
%
%you could estimate the volume fraction and consider that unindexed points
%are sysemtically more likley to be one phase or another
%
%you could estimate volume fraction after a clean up routine
```

copy this m file over, for archival purposes

```
mf_long=mfilename('fullpath');
[f1,f2,f3]=fileparts(mf_long);
mf_start=[mf_long '.m'];
mf_end=fullfile(resultsdir,[f2 '.m']);
try
copyfile(mf_start,mf_end)
catch
    warning('m file not saved, likely due to spaces in the file name');
```

```
end
```

Published with MATLAB® R2022b