

CHAPTER 1
THE SECOND

1.1 Electron Microscopy Preparation

1.1.1 Tissue Processing

Tissue Processing Dehydration to Plastic

Christopher Creveling, Graduate Student *

June 6th, 2018

Abstract

Tissue preparation in the electron microscopy lab to use TEM to look at the vitreo-retinal interface in eyes.

§1 Introduction

This document is intended to be used to process tissue from formalin to embedded plastic to be used on the transmission electron microscope (TEM) to identify the orientation of collagen fibers.

§1.1 Sorting

Begin first by sorting the tissue in two piles of tissue that was peeled and tissue that was adjacent to the peeled region. Then write down the identification ID # on the paper to keep the proper vial straight during the tissue process.

§1.1.1 Identification ID

Sheep #, L/R, E/P, P/A

For example, *UL-15A-B Left Equator Peel* can be reduced to *UL15LEP*

§2 Dehydration

First place samples in glass vials. Use forceps if it is required to remove excess waste from the container. Properly label the samples from before section 1.1.1 and place the label on the vial. Before adhering the label to the vial, write down the number of specimens in the vial to ensure that the specimens don't get lost during the process. Use tape to ensure that the label will not be removed from the vial during the process.

*N. Chandler was with the Electron Microscopy Facility, University of Utah, Salt Lake City, UT, 84112 USA e-mail: (see <http://www.bioscience.utah.edu/molecular-biology/core-facilities.php>).

§2.1 Buffer Rinse

Remove the fixative from the existing vial using the micropipette. Be sure not to suck out the tissue. Then fill the vial with buffer - 0.1M Sodium Cacodylate buffer.

§2.1.1 Agitation

Put the sample vials in the rotating agitator for 5 minutes.

§2.2 Buffer Rinse

Remove the buffer from section 2.1 and replace with new buffer - 0.1M Sodium Cacodylate buffer.

§2.2.1 Agitation

Put the sample vials in the rotating agitator again for 5 minutes.

§2.3 Osmium dilution

During the previous agitation step in section 2.2.1 dilute the osmium tetroxide OsO_4 (4% in dH_2O) with 0.2 M Sodium Cacodylate buffer in a 1:1 mixture. Be sure to filter the Osmium tetroxide with a millipore filter to remove any excess particulate that would otherwise result in artifacts inside the tissue.

§2.4 Osmium rinse

Remove the 0.1M Sodium cacodylate buffer from the vials and replace with the diluted Osmium from section 2.3. Use just enough diluted Osmium to cover the tissue.

§2.4.1 Agitation

Put the sample vials back in the rotating agitator again for one hour.

§2.5 DI water rinse

Remove the diluted Osmium tetroxide from the vials and rinse with DI water. The DI water will be filtered¹. This step is done to remove excess osmium.

§2.5.1 Agitation

Put the sample vials back in the rotating agitator again for 5 minutes.

§2.6 Uranyl Acetate rinse

Remove the DI water from the vials and replace with Saturated 4% Aqueous Uranyl Acetate. The Uranyl Acetate also needs to be filtered using a millipore filternote1 on a 10 ml syringe.

§2.6.1 Agitation

Put the sample vials back in the rotating agitator again for one hour.

¹The millipore filter is used to remove any excess particulate that would otherwise result in artifacts inside the tissue.

§3 Final acetone dehydration step

The final step of the dehydration process is to replace all of the moisture in the tissue from H_2O to pure acetone. This is done with a series of rinses in various percentages of alcohol with the last set of rinses in acetone. **Note - if there is not enough alcohol mixtures in the hood then you will need to make more. When making the dilutions, use the graduated cylinder that is in the sink and mix the highest concentrations first to ensure that the percentages of alcohol is correct. Start with 95 then 70 then 50 etc. Also be sure that the ethanol containers are covered to prevent evaporation during each step of the dehydration process.

§3.1 50% Ethanol Alcohol

Remove the urinal acetate from the vial in section 2.6 to the appropriate container. Next use the micropipette and fill the vial with 50% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.1.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.2 70% Ethanol Alcohol

Remove the 50% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 70% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.2.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.3 95% Ethanol Alcohol

Remove the 70% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 95% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.3.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.4 95% Ethanol Alcohol

Remove the 95% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 95% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.4.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.5 100% Ethanol Alcohol

Remove the 95% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 100% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.5.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.6 100% Ethanol Alcohol

Remove the 100% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 100% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.6.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.7 100% Ethanol Alcohol

Remove the 100% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 100% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.7.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.8 100% Ethanol Alcohol

Remove the 100% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with 100% Ethanol Alcohol; ensure that the tissue specimen is well covered.

§3.8.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.9 Acetone

Remove the 100% Ethanol Alcohol from the vial. Next use the micropipette and fill the vial with acetone; ensure that the tissue specimen is well covered.

§3.9.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.10 Acetone

Remove the acetone from the vial. Next use the micropipette and fill the vial with acetone; ensure that the tissue specimen is well covered.

§3.10.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.11 Acetone

Remove the acetone from the vial. Next use the micropipette and fill the vial with acetone; ensure that the tissue specimen is well covered.

§3.11.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§3.12 Acetone

Remove the acetone from the vial. Next use the micropipette and fill the vial with acetone; ensure that the tissue specimen is well covered.

§3.12.1 Agitation

Put the sample vials in the rotating agitator again for 10 minutes.

§4 Infiltration

Once the tissue samples have been completely dehydrated and all moisture in the sample has been replaced with acetone, the next step is to infiltrate with plastic. This will allow the tissue to be embedded and then cut using the Ultramicrotomes. This will also take a few steps that still incorporate various mixtures of acetone and plastic.

§4.1 Acetone & Plastic

The first step is to remove the acetone from the vial using a micropipette and replacing it with a 1:1 mixture of acetone and plastic. Again, as mentioned before, the vial does not need to be filled up to the brim, just enough to thoroughly allow plastic to infiltrate the tissue.

§4.1.1 Agitation

Put the sample vials in the rotating agitator again for one hour.

§4.2 Acetone & Plastic Overnight Option**

If you are to finish the process for the day and return the next, then perform the following option, if not skip to section 4.3. First remove the 1:1 mixture from section 4.1 and replace with a 3:1 mixture of plastic to acetone and let it sit overnight.

§4.3 Acetone & Plastic

If you are to finish the process the same day then skip section 4.2. First remove the 1:1 mixture from section 4.1 and replace with a 3:1 mixture of plastic to acetone.

§4.3.1 Agitation

Put the sample vials in the rotating agitator again for one hour.

§4.4 Pure Plastic

First remove the 3:1 mixture from either section 4.2 or 4.3 and replace with pure plastic.

§4.4.1 Agitation

Put the sample vials in the rotating agitator again for one hour.

§4.4.2 Vacuum

Place all of the vials with the lids removed inside the vacuum chamber. Turn the pump on to remove air from the chamber. This will remove all air from the samples that has been embedded inside the tissue and will allow the infiltration of plastic to fully take affect. Let the samples sit inside the vacuum chamber for one hour.

§4.5 Pure Plastic

Remove the pure plastic from section 4.4 and replace with pure plastic again.

§4.5.1 Agitation

Put the sample vials in the rotating agitator again for one hour.

§4.5.2 Vacuum

Place all of the vials with the lids removed inside the vacuum chamber. Turn the pump on to remove air from the chamber. This will remove all air from the samples that has been embedded inside the tissue and will allow the infiltration of plastic to fully take affect. Let the samples sit inside the vacuum chamber for one hour.

§5 Embedding

The next step is to embed the plasticized tissue into the mold. Before forgoing with this process, a list of all of the specimens will need to be created on Excel to print and cut out. For example if there are five specimens in the same vial, make a list of sample names with the specimen ID (A), specimen ID (B), ... specimen ID (E). Next, grab a razor blade and a wooden stir stick. Simply use the razor blade to shave away wood from the stir stick to make a flat surface. The flat surface will be used to transfer specimens from the vials to the mold. Place the printed out label inside the mold and set the mold inside the oven to let it bake the specimens to cure the plastic.

§6 Cutting

After the plastic has cured, remove the specimen to be cut and use the microtome to shave away thin layers to be used for TEM.

§7 Grid Staining

Once thin sections have been placed on grids from section 6 the grids will need to be stained to increase the contrast for TEM. Two chemicals will be Uranyl Acetate and Lead Citrate.

§7.1 Preparation

Using the square petri-dish and wax from the cupboard cut the wax to fit the inside the petri-dish. Clean the wax with alcohol and DI water to remove any impurities on the wax that would alter the grid samples. This will also prevent the drops from coagulating together on the wax. Simply rinse the wax to clean it off.

§7.2 Chemical Prep

After the wax has been cleaned and cut remove the saturated Uranyl Acetate and Reynold's Lead Citrate from the refrigerator. Grab two small 1 ml syringes from the drawer and fill up each syringe with either UA or Lead Citrate. Then place one small filter on the end of the syringe filled with UA and two filters on the syringe filled with Lead Citrate.

§7.3 UA Stain

Using the 1 ml syringe with a single filter place a droplet of UA for each grid that you need to stain evenly spaced on the wax pad. Use the forceps and remove the grids from the grid holder and place on top of the UA droplet. Be sure to place the grid shiny side down to allow the UA to stain the specimen.

§7.4 Timer - 18 minutes

Set the timer for 18 minutes. During this time fill up enough 30 ml syringes with DI water for rinsing both UA and Lead Citrate. You will need approximately 10 ml per sample per rinse. Place a large filter on the end of the syringe.

§7.5 Staging Area

Grab a small round petri dish and insert two filter papers to absorb the water following the rinse. Use a pen or pencil to mark the paper to help organize the order of specimens to prevent a mix up.

§7.6 First Rinse

After 18 minutes, pick up the grid with forceps and rinse with 10 ml of DI water. Hold the forceps at a 60° angle from the horizontal and drip the water down the curved section of the forceps. After the rinse, place the specimens inside the round petri dish to remove excess DI water. Once all of the specimens have been placed on the filter paper, a few sodium hydroxide crystals will need to be placed inside the square petri dish. The NaOH will help prevent any sort of moisture from interfering with the grid during the staining process. Next, use the other 1 ml syringe with Lead Citrate and place drops on the wax pad following the same procedure mentioned before in section 7.3.

§7.7 Lead Citrate Rinse

Using the forceps, grip the grid and place it on top of the Lead Citrate droplet with the shiny side down which allows the grid to be stained. Set the timer for eight minutes.

§7.8 Second Rinse

After eight minutes have passed, repeat the same step as in 7.6. Once the grids have completely dried, place them back in the grid holder and they are ready for the TEM.

§7.9 Cleanup

Dispose of the petri dish in the unwanted UA container.

§8 Transmission Electron Microscopy

Transmission electron microscopy (TEM) is a microscopy technique in which a beam of electrons is transmitted through an ultra-thin specimen, interacting with the specimen as it passes through it.

§8.1 TEM

Head over to the TEM and begin imaging!

#	Step	Instruction	Time	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Dehydration	0.1 M Sodium Cacodylate buffer	A* 5 minutes								
2	Dehydration	0.1 M Sodium Cacodylate buffer	A* 5 minutes								
3	Fix	4% OsO_4 with 0.2 M Sodium Cacodylate buffer (1:1 filtered)	A* 60 minutes								
4	Rinse	DI water rinse	A* 5 minutes								
5	Stain	Saturated 4% Aqueous Uranyl Acetate (filtered)	A* 60 minutes								
6	Dehydration	50% Ethanol	A* 10 minutes								
7	Dehydration	70% Ethanol	A* 10 minutes								
8	Dehydration	95% Ethanol	A* 10 minutes								
9	Dehydration	95% Ethanol	A* 10 minutes								
10	Dehydration	100% Ethanol	A* 10 minutes								
11	Dehydration	100% Ethanol	A* 10 minutes								
12	Dehydration	100% Ethanol	A* 10 minutes								
13	Dehydration	100% Ethanol	A* 10 minutes								
14	Dehydration	Acetone	A* 10 minutes								
15	Dehydration	Acetone	A* 10 minutes								
16	Dehydration	Acetone	A* 10 minutes								
17	Dehydration	Acetone	A* 10 minutes								
18	Infiltration	1:1 Plastic to Acetone	A* 60 minutes								
19	Infiltration	3:1 Plastic to Acetone	A* 60 minutes								
20	Infiltration	Pure Plastic	A* 60 minutes								
21	Vacuum	Vacuum	V* 60 minutes								
22	Infiltration	Pure Plastic	A* 60 minutes								
23	Vacuum	Vacuum	V* 60 minutes								
24	Embedding	Embedding	Limitless								

Table 1: Simplified instructions to check off the steps during the tissue processing by hand. A* indicates Agitation, and V* indicates Vacuum.

Station #	Step	Instruction	Time	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1	-	-	-	-	-	-	-
2	Dehydration	0.1 M Sodium Cacodylate buffer	A* 10 minutes				
3	Fix	4% OsO_4 with 0.2 M Sodium Cacodylate buffer (1:1 filtered)	A* 60 minutes				
4	Rinse	DI water rinse	A* 10 minutes				
5	Stain	Saturated 4% Aqueous Uranyl Acetate (filtered)	A* 60 minutes				
6	Dehydration	50% Ethanol	A* 10 minutes				
7	Dehydration	70% Ethanol	A* 10 minutes				
8	Dehydration	95% Ethanol	A* 10 minutes				
9	Dehydration	95% Ethanol	A* 10 minutes				
10	Dehydration	100% Ethanol	A* 10 minutes				
11	Dehydration	100% Ethanol	A* 10 minutes				
12	Dehydration	100% Ethanol	A* 10 minutes				
13	Dehydration	100% Ethanol	A* 10 minutes				
14	Dehydration	Acetone	A* 10 minutes				
15	Dehydration	Acetone	A* 10 minutes				
16	Dehydration	Acetone	A* 10 minutes				
17	Dehydration	Acetone	A* 10 minutes				
18	Infiltration	1:1 Plastic to Acetone	A* 60 minutes				
19	Infiltration	3:1 Plastic to Acetone	A* 60 minutes				
20	Infiltration	Pure Plastic	A* & V* 60 minutes				
21	Infiltration	Pure Plastic	A* & V* 60 minutes				
22	-	-	-	-	-	-	-
23	-	-	-	-	-	-	-
24	-	-	-	-	-	-	-

Table 2: Simplified instructions to check and make sure the automatic tissue processor is set up at the correct stations. Each vial should be filled with 20 ml when processing. Be sure to check the program on the automatic tissue processor; it should be marked by program #2. A* indicates Agitation, and V* indicates Vacuum.

1.1.2 Tissue Processing

Tissue Processing Dehydration to Plastic

Christopher Creveling, Graduate Student *

June 8th, 2018

Abstract

Tissue preparation in the electron microscopy lab to prepare the EMbed 812 for tissue processing.

§1 Introduction

This document is intended to be used to process tissue from formalin to embedded plastic to be used on the transmission electron microscope (TEM) to identify the orientation of collagen fibers.

§2 EMbed 812

§2.1 Personal Protective Equipment

Begin first by grabbing a lab coat and then use paper towels and acetone to clean off the scale used for measuring out the mass of various resin mixtures.

§2.2 Recipe for EMbed 812

Remove the four chemicals for the EMbed 812 resin from the cabinet by using the WPE-147. Where W.P.E. is the Weight per Epoxide Equivalent).

Ingredient	Unit
EMbed 812 Resin	51.80 g
DDSA	26.68 g
NMA	21.67 g
BDMA	2.5 ml

Table 1: Simplified instructions to check and make sure the automatic tissue processor is set up at the correct stations. Each vial should be filled with 20 ml when processing. Be sure to check the program on the automatic tissue processor; it should be marked by program #2.

*N. Chandler was with the Electron Microscopy Facility, University of Utah, Salt Lake City, UT, 84112 USA e-mail: (see <http://www.bioscience.utah.edu/molecular-biology/core-facilities.php>).

§3 Lab Equipment

Grab one (400 ml) Tripore container along with four clean pipettes.

§4 EMbed 812

Balance the scale with the Tripore container and pour in 51.80 g of EMbed 812 Resin. Clean the bottle and cap with Kimwipes and throw out the pipette. Grab a strip of Parafilm to stretch over the cap to ensure an air-tight seal.

§5 DDSA

Balance the scale after 51.80 g of EMbed 812 Resin has been added to the Tripore container. Add 26.68 g of DDSA to the container by first underpouring and then using a pipette to add the rest of the DDSA. Clean the bottle and cap with Kimwipes and throw out the pipette. Grab a strip of Parafilm to stretch over the cap to ensure an air-tight seal.

§6 NMA

Balance the scale after 26.68 g of DDSA has been added to the Tripore container. Add 21.67 g of NMA to the container by first underpouring and then using a pipette to add the rest of the NMA. Clean the bottle and cap with Kimwipes and throw out the pipette. Grab a strip of Parafilm to stretch over the cap to ensure an air-tight seal.

§7 Stir

Move the Tripore container inside the hood. Add a stirbar to the Tripore container containing EMbed 812 Resin, DDSA, and NMA to stir the mixture for 10 minutes in the hood.

§8 BDMA

In the hood, while the Tripore container is being stirred, use a graduated micropipette and obtain 2.5 ml of BDMA. The BDMA is used as the accelerant for polymerization.

§9 Stir

Stir the Tripore container containing EMbed 812 Resin, DDSA, NMA, and BDMA mixture for 10 minutes in the hood. The mixture should turn orange after the BDMA has been added.

§10 Parafilm

Using Parafilm wrap, ensure that each chemical lid has been wrapped to keep an air-tight seal.

§11 Syringe

Grab eight syringes and caps from the cupboards and prepare them for filling up with the resin. Place the newly filled resin syringes in the freezer.

§12 Clean-up

Clean the magnetic stirbars with acetone. Be sure to put vinyl liners inside the gloves.

§13 Embed

Embed the tissue samples!

1.2 MatLab Least Squares

</> Script 1: Matlab script that performs a least squares regression calculation. </>

```
1 function [A] = Least_Squares(A)
2 % Calculate the slope and y-intercept using matrix math
3 % x & y are the coordinates of points
4 x = A(:,1);
5 y = A(:,2);
6 Z = ones(length(x),2);
7 Z(:,2) = x;
8 % Calculate the matrix inverse for the constants of the regression
9 A = inv(Z'*Z)*(Z'*y);
10 return
11 end
```

1.3 Ridge Detection Input Parameters

</> Script 2: Matlab script that determine ridge detection parameters using TEM images. </>

```
1 % Sigma selection parameter
2 % Christopher Creveling
3
4 close all
5 clear
6 clc
7
8 [file_name_root, dirname] = uigetfile('*.tif');
9 info = imfinfo(file_name_root);
10 % Gathers the resolution from the image data
11 resolution = info.XResolution;
12
13 line_width = 0.026; % Micron length
14 U = 204; % Image upper intensity value (background)
15 P = 160; % Pixel intensity for the contrast value
16
17 % line_width = input('Max of four line width measurements (Microns)\n');
18 fprintf('Resolution %f (pixels/micron)\n', resolution);
19
20 fprintf('Line width %f (microns)\n', line_width);
21 % resolution = 623.1429; % conversion between length and pixels
22
23 L = line_width*resolution; %Line width in pixels
24 fprintf('Line width %f (pixels)\n', L);
25 w = L/2; % width of a line in pixels
26 sigma = w/sqrt(3) + 0.4; % calculated sigma value
27 % sigma = 3.1
28 fprintf('Sigma = %f\n', sigma)
29
30 % sigma = 3.8; % approximate value
31
32 fprintf('U --- %d\n', U);
33 fprintf('P --- %d\n', P);
```

```

34 % Contrast (difference between upper and selected pixel intensity values)
35 h = U - P;
36 %h = 42;
37 fprintf('h = %d\n', h)
38
39 % First derivative of the gaussian kernel [Equation 4] - 1D
40 g_p1Dx = @(x, sigma)-x/(sqrt(2*pi)*sigma^3)*exp(-(x^2)/(2*sigma^2));
41 % Second directional derivative approximation [Equation 8]
42 rb_pp1Dx = @(x) h*(g_p1Dx(x + w, sigma) - g_p1Dx(x - w, sigma));
43 % Evaluate the second order approximation at zero to find out the upper
44 % threshold value 1D
45 fprintf('1D upper threshold approximation is %f\n', abs(rb_pp1Dx(0)))
46
47 % First derivative of the 2D gaussian kernel [Equation 4]
48 g_p2Dx = @(x, y, sigma)-x/(2*pi*sigma^4)*exp(-(x^2 + y^2)/(2*sigma^2));
49 % First derivative of the 2D gaussian kernel [Equation 4]
50 g_p2Dy = @(x, y, sigma)-y/(2*pi*sigma^4)*exp(-(x^2 + y^2)/(2*sigma^2));
51 % Second directional derivative approximation [Equation 8]
52 % rb_pp2D = @(x, y) h*(g_p2Dx(x + w, y, sigma) - ...
53 %     g_p2Dx(x - w, y, sigma) + g_p2Dy(x, y + w, sigma) - ...
54 %     g_p2Dy(x, y - w, sigma));
55 % Evaluate the second order approximation at zero to find out the upper
56 % threshold value 1D
57 % fprintf('2D upper threshold approximation is %f\n', abs(rb_pp2D(0, 0)))
58 % s = 0.006:0.001:0.03; % Range of sigma values
59 %
60 % for i = 1:length(s)
61 %     H(i) = abs(h*(g_p1Dx(0 + w, s(i)) - g_p1Dx(0 - w, s(i))));
62 % end
63 % H';

```

1.4 Analyze Ridge Detection Output

</> Script 3: Matlab script that analyzes ridge detection output from TEM images. </>

```

1 % Name: Christopher Creveling
2 % Date: 11/13/18
3 % Title: Image analysis Ridge Detection Interpretation
4
5 % Description: After running a non-local means filter and further running
6 % a Ridge-Detection algorithm through Fiji I am trying to learn to extract
7 % what the output is giving me
8
9 %{
10 Output from Ridge-Detection
11 /** This class holds one extracted line. The field num contains the number of
12 points in the line. The coordinates of the line points are given in the
13 arrays row and col. The array angle contains the direction of the normal
14 to each line point, as measured from the row-axis. Some people like to
15 call the col-axis the x-axis and the row-axis the y-axis, and measure the
16 angle from the x-axis. To convert the angle into this convention, subtract
17 PI/2 from the angle and normalize it to be in the interval [0, 2*PI). The
18 array response contains the response of the operator, i.e., the second
19 directional derivative in the direction of angle, at each line point. The
20 arrays width_l and width_r contain the width information for each line point

```



```

21 if the algorithm was requested to extract it; otherwise they are NULL. If
22 the line position and width correction was applied the contents of width_l
23 and width_r will be identical. The arrays asymmetry and contrast contain
24 the true asymmetry and contrast of each line point if the algorithm was
25 instructed to apply the width and position correction. Otherwise, they are
26 set to NULL. If the asymmetry, i.e., the weaker gradient, is on the right
27 side of the line, the asymmetry is set to a positive value, while if it is
28 on the left side it is set to a negative value. */
29 %}
30
31 clear all;
32 close all force; % Force the message boxes to close
33 clear;
34 clc;
35
36 cd 'Z:\students\Yousef\TEM\Ridge detection\Fiji Output'
37
38 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39 % Real TEM Image Data
40 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41 % Import the data from the CSV file
42 synthetic = false;
43
44 % file root name _crop
45 file_name_root = 'H160993LPA-3_12_L4';
46 % File name extension _h85_H191_L06_S44
47 file_name_extension = '_C41_U351_L02_S220_W0010';
48 % file_name_extension = '';
49
50 % Ridge Detection Results
51 table_1 = readtable(strcat(file_name_root, file_name_extension, '_RD.csv'));
52 % Ridge Detection Junction Results
53 % table_2 = readtable(strcat(file_name_root, file_name_extension, ...
54 % '_RD_J.csv'));
55 % Ridge Detection Summary Results
56 table_3 = readtable(strcat(file_name_root, file_name_extension, '_RD_S.csv'));
57 % Extract information from the original image
58 img = imread(strcat(file_name_root, '.tif'));
59 info = imfinfo(strcat(file_name_root, '.tif'));
60 x_scale = info.XResolution;
61 y_scale = info.YResolution;
62 val = 1;%input(prompt);
63
64 fiber_color_num = 11; % the number of fiber divisions for the visual output
65
66 height = size(img, 2);
67 width = size(img, 1);
68
69 % Set up the file for outputting data
70 fileID = fopen(strcat(file_name_root, file_name_extension, '.txt'), 'w');
71
72 %%
73 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74 % Identify how to properly shift the TEM image
75 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 prompt = ['Are the collagen fibers on the top (1), right (2), ' ...
77 'bottom (3), or left (4)? \n'];
78 % val = 2;%input(prompt);

```

```

79 if (val == 1)
80     % No need to shift pixels
81     shift_x = 0;
82     shift_y = 0;
83 elseif (val == 2)
84     % shift pixels to the right
85     shift_x = max(width) - max(table_1.X*x_scale);
86     shift_y = 0;
87 elseif (val == 3)
88     % shift pixels down
89     shift_x = 0;
90     shift_y = max(width) - max(table_1.X*y_scale);
91 elseif (val == 4)
92     % No need to shift pixels
93     shift_x = 0;
94     shift_y = 0;
95 else
96     err = 'Invalid input';
97     error(err);
98 end
99
100
101 %%
102 % Ask for collagen
103 % Ask for collagen
104 figure
105 imshow(img);
106
107
108
109 answer = questdlg('Do Collagen fibers exist?');
110 switch answer
111     case 'Yes'
112         close
113
114         %%
115         % Plot the RD classification color for all of the fiber segments detected
116         % by the algorithm
117         % figure
118         % imshow(img);
119         % hold on
120         % RD_classification = unique(table_1.Class);
121         % RD_class_vals = []; % Empty array
122         % C = hsv(length(RD_classification));
123         % for i = 1:length(RD_classification)
124             % Ridge_Detection_Class{i} = RD_classification(i);
125             % RD_class_vals(i).XY = [table_1.X(categorical(table_1.Class) ==
126     RD_classification{i}), ...
127             % table_1.Y(categorical(table_1.Class) ==
128     RD_classification{i})*x_scale;
129             % plot(RD_class_vals(i).XY(:, 1) + shift_x, RD_class_vals(i).XY(:, 2) +
130     shift_y, '.', 'Color', C(i, :), 'markersize', 5, 'linewidth', 3);
131             % Legend(i) = num2str(RD_classification(i));
132             % end
133             % legend(RD_classification, 'location', 'best');

```

```

134 Length_segment = table_1.Length; % extract line length
135 Contour_ID = table_1.ContourID;
136
137 %%
138 % Ridge-Detection results
139
140 figure;
141 imshow(img);
142 title('\bf Original Image')
143
144 msgStr = ['Select two points that define the ILM (Right to Left' ...
145         ' if Collagen Fibrils are above, Left to Right if' ...
146         ' Collagen Fibrils are below'];
147 % Indicate the ILM used for angle calculations
148 f = msgbox(msgStr, 'ILM');
149 pause(3);
150 [ILM.x, ILM.y] = ginput(2);
151 % delete(f); % Delete the message box
152 hold on
153 plot(ILM.x, ILM.y, 'bo', 'linewidth', 2);
154 % Sorts rows of the input to maintain correct order (ascending)
155 % ILM.x = sortrows(ILM.x);
156 ILM_slope = [];
157 ILM_angle = [];
158 for i = 1:length(ILM.x)-1
159     numerator = (ILM.y(i+1) - ILM.y(i));
160     denominator = (ILM.x(i+1) - ILM.x(i));
161     % slope of the line
162     ILM_slope(i) = numerator/denominator;
163     % Angle of the ILM relative to the x-axis
164     ILM_angle(i) = -atan(numerator/denominator)*180/pi;
165 end
166 fprintf('ILM slope = %f\n', ILM_slope);
167
168 slope = mean(ILM_slope); % Mean slope between the points
169 y_int = ILM.y(1) - slope*ILM.x(1); % Solve for the y-intercept
170
171
172
173 %% Create Rectangle
174
175 x1 = linspace(ILM.x(1), ILM.x(2));
176 y1 = linspace(ILM.y(1), ILM.y(2));
177 d = 1 * x_scale; %distance in microns
178
179 height = size(img, 2);
180 width = size(img, 1);
181 aLine = [-ILM_slope, 1, -y_int];
182
183 fcn = @(x)ILM_slope*x + y_int; % Function handle
184 fplot(fcn, [0, width], 'r');
185
186 start_ = [ILM.x(1) ILM.y(1)];
187 goal_ = [ILM.x(2) ILM.y(2)];
188
189 n = 2;
190 t = linspace(0, 1, n);
191 v = goal_ - start_;

```

```

192 x3 = start_(1) + t*v(1);
193 y3 = start_(2) + t*v(2);
194 v = d* v / norm(v);
195
196 for i=1:n
197     line([x3(i) - v(2)], [y3(i) + v(1)]);
198     plot([x3(i) - v(2)], [y3(i) + v(1)], 'ro', 'linewidth', 2);
199 end
200
201 x3f = x3 - v(2);
202 y3f = y3 + v(1);
203
204 % Coordinates of the region of interest within the 1 micron rectangle
205 xv = [ILM.x(1), x3f(1), x3f(2), ILM.x(2)];
206 yv = [ILM.y(1), y3f(1), y3f(2), ILM.y(2)];
207
208 % Plots the 1 micron rectangle
209 plot(xv, yv, 'r--', 'LineWidth', 1.5)
210
211 Answer = questdlg('Is this correct?');
212
213 switch Answer
214     case 'Yes'
215         In = inpolygon(table_1.X*x_scale, table_1.Y*y_scale, xv, yv);
216
217         table_1.X = In .* table_1.X;
218         table_1.Y = In .* table_1.Y;
219
220         table_1(~table_1.X, :) = [];
221
222     case 'No'
223         fprintf('Please run code again')
224         msgbox('Please run code again');
225
226     return
227 end
228
229
230 %%%%% End of create Rectangle
231
232 %%
233 % Define the input parameters for the line to border points (Ax+By+C=0)
234 % A = slope
235 % B = integer in front of y
236 % C = y-intercept
237 aLine = [-slope, 1, -y_int];
238
239
240 % extrapolate the ILM line on the image as well as calculate the distance
241 ILM_x_pts = linspace(0, width, 100);
242 for i = 1:length(ILM_x_pts)
243     ILM_line(i) = slope*ILM_x_pts(i) + y_int; % + ILM.x(end)
244 end
245 ILM_length = sqrt((ILM.x(2)-ILM.x(1))^2 + (ILM.y(2) - ILM.y(1))^2);
246 ILM_length = ILM_length/x_scale;
247 fprintf('ILM length = %f microns\n', ILM_length);
248 ILM_angle = (mean(ILM_angle));
249 fprintf(['ILM angle is %f degrees relative to the x-axis ' ...

```

```

250         '(Unit circle)\n'], ILM_angle);
251
252     fiber_min_length = 0.044962164;
253
254     % Indicate the five points on the ILM used for thickness measurements
255     figure
256     imshow(img)
257     for i = 1:5
258         f = msgbox(['Select the first two points that define the ' ...
259             'ILM thickness'], 'ILM');
260         % pause(1);
261         [ILM_thick(i).x, ILM_thick(i).y] = ginput(2);
262         hold on
263         plot(ILM_thick(i).x, ILM_thick(i).y, 'g-o', 'linewidth', 1);
264         % Pythagorean theorem
265         ILM_thick(i).measurement = sqrt((ILM_thick(i).x(1) - ILM_thick(i).x(2))^2
↪ + ...
266             (ILM_thick(i).y(1) - ILM_thick(i).y(2))^2);
267         delete(f); % Delete the message box
268     end
269     for i = 1:5
270         ILM_measurement(i) = ILM_thick(i).measurement;
271     end
272     L{4} = 'ILM thickness measurements';
273     %legend(L, 'location', 'best');
274     axis image;
275
276     ILM_thickness = mean(ILM_measurement)/x_scale*1000;
277     fprintf('Average ILM thickness is %f nanometers \n', ILM_thickness);
278
279
280
281
282
283     %%
284     % Loop over all of the unique Contour ID's and identify the length of each
285     % one
286     ID_num = unique(Contour_ID);
287     for i = 1:length(ID_num)
288         unique_ID_lengths(i) = mean(table_1.Length(table_1.ContourID ==
↪ ID_num(i)));
289         unique_ID_widths(i) = mean(table_1.LineWidth(table_1.ContourID ==
↪ ID_num(i)));
290         unique_ID_ang_of_norm(i) = mean(table_1.AngleOfNormal(table_1.ContourID
↪ == ID_num(i)));
291     end
292
293
294     %%
295     % figure;
296     % imshow(img);
297     % hold on
298
299     % fiber_color_num = 12; % the number of fiber divisions for the visual
300     % output (chosen from up above)
301
302     % Properly match the associated ContourID with the unique_ID number and the
303     % specified fiber length

```

```

304
305     fiber_length = linspace(min(Length_segment), ...
306         max(Length_segment)*0.8, fiber_color_num); %
307     C = hsv(length(fiber_length)); % Splits up the colormap into 11 unique values
308     m_size = 5;
309
310     % Loop over the unique fiber segment lengths to break them apart by lengths
311     for i = 1:length(fiber_length)
312         % if the length of the fibers is longer than the specified bin put them
    ↪ here
313         if i == length(fiber_length)
314             % extract X & Y coordinates of each point based on the criteria
315             fiber(i).x = table_1.X(table_1.Length > fiber_length(i));
316             % extract X & Y coordinates of each point based on the criteria
317             fiber(i).y = table_1.Y(table_1.Length > fiber_length(i));
318             % Calculate fiber area (LineLength *LineWidth)
319             % fiber(i).area = datatbl.Length(datatbl.Length >
    ↪ fiber_length(i)).*datatbl.LineWidth(datatbl.Length > fiber_length(i));
320             fiber(i).len = table_1.Length(table_1.Length > fiber_length(i));
321             fiber(i).wid = table_1.LineWidth(table_1.Length > fiber_length(i));
322             % Fiber area = length * width (pixels)
323             fiber(i).area = fiber(i).len.*fiber(i).wid;
324             % Calculates the angle of the fiber
325             % fiber(i).angle = atan2(max(fiber(i).y) - min(fiber(i).y),
    ↪ max(fiber(i).x) - min(fiber(i).x))*180/pi;
326         else
327             % extract X & Y coordinates of each point based on the criteria
328             fiber(i).x = table_1.X(table_1.Length > fiber_length(i) & ...
329                 table_1.Length <= fiber_length(i+1));
330             % extract X & Y coordinates of each point based on the criteria
331             fiber(i).y = table_1.Y(table_1.Length > fiber_length(i) & ...
332                 table_1.Length <= fiber_length(i+1));
333             % Calculate fiber area (LineLength *LineWidth)
334             % fiber(i).area = datatbl.Length(datatbl.Length > fiber_length(i) &
    ↪ ...
335                 % datatbl.Length <=
    ↪ fiber_length(i+1)).*datatbl.LineWidth(datatbl.Length > fiber_length(i) & ...
336                 % datatbl.Length <= fiber_length(i+1));
337             fiber(i).len = table_1.Length(table_1.Length > fiber_length(i) & ...
338                 table_1.Length <= fiber_length(i+1));
339             fiber(i).wid = table_1.LineWidth(table_1.Length > fiber_length(i) &
    ↪ ...
340                 table_1.Length <= fiber_length(i+1));
341             fiber(i).area = fiber(i).len.*fiber(i).wid; % Fiber area = length *
    ↪ width (pixels)
342             % Calculates the angle of the fiber
343             % fiber(i).angle = atan2(max(fiber(i).y) - min(fiber(i).y), ...
344                 % max(fiber(i).x) - min(fiber(i).x))*180/pi;
345         end
346         tot_fiber_area(i) = sum(fiber(i).area); % sum up fiber area
347     end
348
349     % fiber_area = sum(tot_fiber_area); % fiber area
350     % fprintf('Area of fiber segments [pixels] %f\n', fiber_area);
351     % fprintf(['Collagen fiber segment density (Area of fibers ' ...
352         % '[pixels]/ILM length (nanometers)) %f\n'], ...
353     % fiber_area/ILM_length);
354

```

```

355     % Plot the fibers
356     % for i = 1:length(fiber_length)
357     %     plot(fiber(i).x*x_scale + shift_x, fiber(i).y*y_scale + shift_y, '.',
↪ 'color', C(i, :), 'markersize', m_size);
358     % end
359     %
360     % title('\bf Scatter Plot of Collagen fiber segments with corresponding
↪ lengths');
361     %
362     % Create the legend based upon the length in the fiber array
363     % for i = 1:length(fiber_length)
364     %     if i == length(fiber_length)
365     %         Legend{i} = strcat('L \geq', num2str(fiber_length(i)), '\mu', 'm');
366     %     else
367     %         Legend{i} = strcat(num2str(fiber_length(i)), ...
368     %             ' < L \leq', num2str(fiber_length(i+1)), '\mu', 'm');
369     %     end
370     % end
371     %
372     % [h, ~] = legend(Legend);
373     % %% children of legend of type line
374     % ch = findobj(get(h, 'children'), 'type', 'line');
375     % set(ch, 'Markersize', 24); %% set value as desired
376     % set(h, 'Interpreter', 'latex', 'location', 'best');
377     % axis image;
378     % set(gca, 'DataAspectRatio', [1 1 1]) % Adjust the aspect ratio for printing
379
380
381     %%
382
383     % %%
384     % XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
385     % % Data from the Ridge Detection Junction Results CSV file
386     % XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
387     % T2_x = table_2.X;
388     % T2_y = table_2.Y;
389     % T2_ID1 = table_2.ContourID1;
390     % T2_ID2 = table_2.ContourID2;
391     %
392     % figure
393     % imshow(img);
394     % hold on;
395     % C = hsv(length(T2_x));
396     % for i = 1:length(T2_x)
397     %     %     plot(All_Fibers(i).XYRes(:, 1), All_Fibers(i).XYRes(:, 2), '.',
↪ 'color', C(i, :), 'linewidth', 2);
398     %     %     plot(T2_x(i)*x_scale + shift_x, T2_y(i)*y_scale + shift_y, 'o',
↪ 'linewidth', 3, 'markersize', 8, 'color', C(i, :));
399     %     %     hold on;
400     % end
401     % axis image
402
403
404     %%
405     % XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
406     % Data from the Ridge Detection Summary Results CSV file
407     % XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
408     % Length_T3 = table_3.Length;

```

```

409     % Width_T3 = table_3.MeanLineWidth;
410     % ContourID_T3 = table_3.ContourID;
411     %
412     % figure
413     % subplot(1, 2, 1);
414     % hist(Length_T3, fiber_color_num); % histogram of the lengths from the
→ summary results file
415     % xlabel('\bf Lengths from summary results file');
416     %
417     % subplot(1, 2, 2);
418     % hist(Width_T3, fiber_color_num); % histogram of the lengths from the
→ summary results file
419     % xlabel('\bf Mean line width from summary results file');
420
421     % % Plots all of the junction points from the Fiji Output
422     % figure;
423     % imshow(img);
424     % hold on
425     % plot(c_X + shift_x, c_Y + shift_y, '.', bj_X + shift_x, bj_Y + shift_y, '.',
→ ej_X + shift_x, ej_Y + shift_y, '.', sj_X + shift_x, sj_Y + shift_y, '.', nj_X +
→ shift_x, nj_Y + shift_y, '.', 'markersize', 5)
426     % title('\bf Ridge-Detection Results')
427     % Legend_1 = legend({'Closed Points', 'Both Junction', 'End Junction', 'Start
→ Junction', 'No Junction'}, 'location', 'best');
428     % axis image
429     % [h, ~] = legend(Legend_1);
430     % ch = findobj(get(h, 'children'), 'type', 'line'); %// children of legend of
→ type line
431     % set(ch, 'Markersize', 24); %// set value as desired
432     % set(h, 'Interpreter', 'latex', 'location', 'best');
433     % axis image;
434     % set(gca, 'DataAspectRatio', [1 1 1]) % Adjust the aspect ratio for printing
435
436     %%
437
438     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
439     % Identify the fiber segments that are greater than the threshold and
440     % identify whether or not they overlap and combine them into a single fiber
441     % if they do
442     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
443
444     % close all force;
445     % clc;
446
447     % extract the ContourID & Length in an array
448     ID_Length = unique([table_1.ContourID, table_1.Length], 'rows');
449     % Identify fiber segments that are greater than the minimum length
450     segments = ID_Length(ID_Length(:, 2) >= fiber_min_length);
451     % Identify fiber segments that are less than the minimum length
452     short_segments = ID_Length(ID_Length(:, 2) < fiber_min_length);
453
454
455     % Loop over all of the unique segments to identify which ones are contained
456     % in the longer fibers by looking at all combinations. i.e. if two fiber
457     % segments have matching coordinates/slope they would be combined into a
458     % single fiber and the list of potential fibers would decrease
459
460     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

461 % Go over the matching fibers and further eliminate duplicates
462 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
463
464 % Initialize the arrays
465 tic
466 atol = 0.02; % relative tolerance
467 rtol = 0.01; % absolute tolerance
468
469 c1 = 1; % while loop 1 counter
470 c3 = 1; % fiber match counter
471 count = 1; % iteration counter
472 Lib = [];
473 fiber_union = []; % Fiber unions
474 fiber_segment = []; % initialize the array to be zero
475 condition_segment = []; % Initialize the array to be zero
476 lone_fibers = [];
477 check_1 = false; % Initialize the while loop statements
478
479 while (check_1 == false)
480     check_2 = false; % Initialize the while loop statements
481     c2 = 2; % while loop 2 counter
482     while (check_2 == false)
483
484         % X-coordinates
485         A1 = table_1.X(table_1.ContourID == segments(c1));
486         % Y-coordinates
487         A2 = table_1.Y(table_1.ContourID == segments(c1));
488
489         % X-coordinates
490         B1 = table_1.X(table_1.ContourID == segments(c2));
491         % Y-coordinates
492         B2 = table_1.Y(table_1.ContourID == segments(c2));
493
494         A = [A1, A2]; % [X, Y] coordinates from contour ID (i)
495         B = [B1, B2]; % [X, Y] coordinates from contour ID (j)
496
497         % Find the number of matches between array A and B and store them
498         % every iteration
499         % compares the two arrays to find matches (:, 1:2)(:, 1:2)
500         Lib.logical = double(ismember(A, B, 'rows'));
501         % finds the mean value of the comparison array
502         Lib.mean = mean(Lib.logical);
503         % finds the mode value of the comparison array
504         Lib.mode = mode(Lib.logical);
505         % Sums the zeros
506         Lib.num_zero = sum(Lib.logical == 0);
507         % sums the ones
508         Lib.num_one = sum(Lib.logical == 1);
509         % Identifies the combination of contour ID#s
510         Lib.IDs = [segments(c1), segments(c2)];
511
512         % Consider looking at the slope of each line segment
513         % Pass in an array of coordinates to find the slope & y-intercept [
↪ a_0 + a_1*x]
514         MA = Least_Squares(A);
515         % Pass in an array of coordinates to find the slope & y-intercept [
↪ a_0 + a_1*x]
516         MB = Least_Squares(B);

```

```

517
518 Ax = A(:, 1);
519 Ay = A(:, 2);
520 Bx = B(:, 1);
521 By = B(:, 2);
522
523 % Find the distance between the segments
524 C_A = [mean(Ax), mean(Ay)]; % Center of mass for A
525 C_B = [mean(Bx), mean(By)]; % Center of mass for B
526
527 % Distance between fiber centers
528 D_AB = sqrt((C_A(2) - C_B(2))^2 + (C_A(1) - C_B(1))^2);
529
530 % Local extrema of each fiber segment
531 A_E(1) = min(Ax);
532 A_E(2) = max(Ax);
533 A_E(3) = min(Ay);
534 A_E(4) = max(Ay);
535 B_E(1) = min(Bx);
536 B_E(2) = max(Bx);
537 B_E(3) = min(By);
538 B_E(4) = max(By);
539
540 % Distance between local extrema for each fiber segment assuming
541 % they are linear
542 % Distance between fiber centers
543 L_A = sqrt((A_E(2) - A_E(1))^2 + (A_E(4) - A_E(3))^2);
544 % Distance between fiber centers
545 L_B = sqrt((B_E(2) - B_E(1))^2 + (B_E(4) - B_E(3))^2);
546
547 % Three conditions need to be satisfied
548 % Looks at the mode of the overlap values if there are any
549 condition_1 = (Lib.mode == 1);
550 % Compares how close the two slopes of similar segments are
551 condition_2 = (all(abs(MA(2) - MB(2)) <= atol + rtol*abs(MB(2))));
552 % Compares how close the two y-intercepts are
553 condition_3 = (all(abs(MA(1) - MB(1)) <= atol + rtol*abs(MB(1))));
554 % Is the distance between the fiber centers less than the length of
555 → the fiber segment
556 condition_4 = ((D_AB < L_A) || (D_AB < L_B));
557 condition_5 = (c1 ~= c2); % checks to see if A & B are duplicates
558
559 % Used for debugging
560 [condition_1, condition_2, condition_3, condition_4, ...
561 condition_5, segments(c1), segments(c2), count, ...
562 (max(table_1.ContourID) + 1)];
563
564 % Five conditions need to be satisfied
565 if [condition_1 && condition_5 || condition_2 && ...
566 condition_3 && condition_4 && condition_5]
567
568 A3 = table_1.Length(table_1.ContourID == segments(c1));
569 A4 = table_1.Contrast(table_1.ContourID == segments(c1));
570 A5 = table_1.Asymmetry(table_1.ContourID == segments(c1));
571 A6 = table_1.LineWidth(table_1.ContourID == segments(c1));
572 A7 = table_1.AngleOfNormal(table_1.ContourID == segments(c1));
573
574 B3 = table_1.Length(table_1.ContourID == segments(c2));

```

```

574 B4 = table_1.Contrast(table_1.ContourID == segments(c2));
575 B5 = table_1.Asymmetry(table_1.ContourID == segments(c2));
576 B6 = table_1.LineWidth(table_1.ContourID == segments(c2));
577 B7 = table_1.AngleOfNormal(table_1.ContourID == segments(c2));
578
579 A = [A, A3, A4, A5, A6, A7]; % Combine A with A3:A7
580 B = [B, B3, B4, B5, B6, B7]; % Combine B with B3:B7
581
582 fiber_pair = [segments(c1), segments(c2)];
583
584 % Update the vertical array of matching fiber segment overlaps
585 fiber_segment = vertcat(fiber_segment, fiber_pair);
586
587 % write down which conditions were satisfied per segment
588 condition_quad = [condition_1, condition_2, ...
589                  condition_3, condition_4, condition_5];
590 condition_segment = vertcat(condition_segment, ...
591                             condition_quad);
592
593 % merge the two contourID's (XY) coordinates together without
↪ duplicating points
594 fiber_union(c3).XY = [union(A, B, 'rows', 'stable')];
595 f_len = length(fiber_union(c3).XY); % length of the matched fiber
↪ segment
596
597 % Length of the new segments is going to be a mixture of the
598 % two fiber segments
599 L_A = unique(table_1.Length(table_1.ContourID == segments(c1)));
600 L_B = unique(table_1.Length(table_1.ContourID == segments(c2)));
601
602 % Fiber A contains all of fiber B
603 case_1 = (Lib.mode == 1) && (Lib.num_zero == 0);
604 % Fiber A contains the majority of fiber B
605 case_2 = (Lib.mode == 1) && (condition_2 == 1) && ...
606         (condition_3 == 1) && (condition_4 == 1);
607 % Fiber A contains the minority of fiber B
608 case_3 = (Lib.mode == 0) && (condition_2 == 1) && ...
609         (condition_3 == 1) && (condition_4 == 1);
610 % Fiber A does not contain fiber B
611 case_4 = (Lib.num_one == 0) && (condition_2 == 1) && ...
612         (condition_3 == 1) && (condition_4 == 1);
613
614 if case_1 == 1
615     % Max of the two fiber segments length
616     new_fiber_len = max([A3;B3]);
617 elseif case_2 == 1
618     overlap = Lib.num_one;
619     % If the majority of the points overlap, find the percentage
620     new_fiber_len = (L_A + L_B - ...
621                     (overlap/length(A)*L_A + ...
622                      overlap/length(B)*L_B)/2);
623 elseif case_3 == 1
624     overlap = Lib.num_one;
625     % If the majority of the points overlap, find the percentage
626     new_fiber_len = L_A + L_B - ...
627                     (overlap/length(A)*L_A + ...
628                      overlap/length(B)*L_B)/2;
629 elseif case_4 == 1

```

```

630         % if the two fibers don't overlap
631         new_fiber_len = L_A + L_B;
632     else
633         % Average the two lengths
634         new_fiber_len = 0.5*(L_A + L_B);
635     end
636
637     % store the matching contourID with the coordinates
638     fiber_union(c3).segment_match = fiber_pair;
639     % adds a new ContourID number (max(ContourID) + 1)
640     fiber_union(c3).New_ContourID = ones(f_len, 1) *
↪ (max(table_1.ContourID) + 1);
641     if strcmp(table_1.Properties.VariableNames{1}, 'Var1')
642         % update the number Var1 number. Some of the outputs have
↪ this. If not, comment out
643         fiber_union(c3).Var1 = ones(f_len,
↪ 1).*table_1.Var1(end):table_1.Var1(end) + f_len - 1;
644     end
645     fiber_union(c3).Frame = ones(f_len, 1);
646     fiber_union(c3).Pos_ = 1:f_len;
647     fiber_union(c3).X = fiber_union(c3).XY(:, 1);
648     fiber_union(c3).Y = fiber_union(c3).XY(:, 2);
649     %; % Update new fiber length
650     fiber_union(c3).Length = ones(f_len, 1)*new_fiber_len;
651     fiber_union(c3).Contrast = fiber_union(c3).XY(:, 4);
652     fiber_union(c3).Asymmetry = fiber_union(c3).XY(:, 5);
653     fiber_union(c3).LineWidth = fiber_union(c3).XY(:, 6);
654     fiber_union(c3).AngleOfNormal = fiber_union(c3).XY(:, 7);
655     fiber_union(c3).Class(1:f_len) = {'new_fiber'};
656     fiber_union(c3).Class = fiber_union(c3).Class(1:f_len)';
657
658     % create a shortcut for the list
659     fu = fiber_union(c3);
660     % transpose the position
661     fu.Pos_ = fu.Pos_';
662     % If the attribute is in the CSV file add the info
663     if strcmp(table_1.Properties.VariableNames{1}, 'Var1')
664         fu.Var1 = fu.Var1'; % Transpose the column
665         % new matching segment info
666         table_1_new_fiber_segment = table(fu.Var1, ...
667             fu.Frame, fu.New_ContourID, fu.Pos_, fu.X, ...
668             fu.Y, fu.Length, fu.Contrast, fu.Asymmetry, ...
669             fu.LineWidth, fu.AngleOfNormal, fu.Class);
670     else
671         % If the attribute is not in the CSV file, move on without it
672         % new matching segment info
673         table_1_new_fiber_segment = table(fu.Frame, ...
674             fu.New_ContourID, fu.Pos_, fu.X, fu.Y, ...
675             fu.Length, fu.Contrast, fu.Asymmetry, ...
676             fu.LineWidth, fu.AngleOfNormal, fu.Class);
677     end
678     % stores the variable names to the new table for merging
679     table_1_new_fiber_segment.Properties.VariableNames =
↪ table_1.Properties.VariableNames;
680     % append new matching segment info to table1
681     table_1 = [table_1;table_1_new_fiber_segment];
682
683     % % Plot both segments that are being eliminated

```

```

684         % figure;
685         % imshow(img);
686         % hold on
687         % plot(Ax*x_scale + shift_x, Ay*y_scale + shift_y, 'r.',
↪ 'markersize', 5);
688         % plot(Bx*x_scale + shift_x, By*y_scale + shift_y, 'bo',
↪ 'markersize', 5);
689         % txt = {'\leftarrow A -s#', num2str(segments(c1)), '\leftarrow B
↪ -s#', num2str(segments(c2))};
690         % text(mean(Ax*x_scale) + shift_x, mean(Ay*y_scale) + shift_y,
↪ strcat(txt{1}, txt{2}));
691         % text(mean(Bx*x_scale) + shift_x, mean(By*y_scale) + shift_y,
↪ strcat(txt{3}, txt{4}));
692         %
693         % % Used for debugging
694         % fprintf('A ----- %f, B ----- %f, New Fiber #%d ----- %f\n', ...
695         % L_A, L_B, unique(fiber_union(c3).New_ContourID), ...
696         % new_fiber_len);
697
698
699         % If the length of segment_A is longer than segment_B get rid
700         % of the smaller segment (segment_B)
701         if(length(A) > length(B))
702             % Update the table with the new ContourID #
703             segments(c1) = max(table_1.ContourID);
704             % Delete the ID number from list 'B'
705             segments(c2) = [];
706             % start from the top of the list
707             % c2 = 1;
708
709         % If the two segments are identical
710         elseif (segments(c1) ~= segments(c2))
711             % Update the table with the new ContourID #
712             segments(c2) = max(table_1.ContourID);
713             % Delete the ID number from list 'A'
714             segments(c1) = [];
715             % start from the top of the list
716             % c1 = 1;
717         end
718         % restart from the top of the list
719         c1 = 1;
720         % c2 = 2;
721         % Update the matched pairs counter
722         c3 = c3 + 1;
723     end
724
725     % If the length of segments is 1 or 0, or the last iteration of the
↪ loop
726     if (length(segments) <= 1) || (length(segments) == c2)
727         % If there are no more matches after the end of looping through
↪ the
728         % it is considered a 'lone fiber'
729         lone_fibers = [lone_fibers; segments(c1)];
730         % Delete the ID number from list 'A'
731         segments(c1) = [];
732         % restart from the top of the list
733         c1 = 1;
734         fprintf(['Segment # %.0f removed from the list ' ...

```

```

735         'of potential segments (%d)\n'], ...
736         lone_fibers(end), length(segments));
737     % If there are no more combinations that can be ...
738     % checked then all unique fibers have been ...
739     % identified and concatenated
740     check_2 = true;
741     if (length(segments) == 0) || (length(segments) == 1) % (c1 ==
↪ length(segments)) || (c1 > length(segments))
742         % If there are no more combinations that can
743         % be checked then all unique fibers have been
744         % identified and concatenated
745         check_1 = true;
746     end
747 end
748
749 if (condition_1 && condition_5 || ...
750     condition_2 && condition_3 && ...
751     condition_4 && condition_5)
752     % restart from the top of the list if a segment was removed
753     c2 = 2;
754 else
755     % Update the iteration for while loop #2
756     c2 = c2 + 1;
757 end
758 count = count + 1; % Update the number of iterations
759 end
760 % c1 = c1 + 1; % Update the iteration for while loop #1 % We don't need
761 % to update this because we are eliminating the c1 point if there are
762 % not matches after each c2 iteration through all of the segments. We
763 % should probably eliminate the first while loop because it is
764 % unnecessary to increment now in this 2.0 version of the code by
765 % eliminating the c1 point.
766 end
767 toc
768
769 %%
770 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
771 % Plot the fiber segments that matched from the previous step
772 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
773
774 % sort the fibers from the previous loop to color code by length
775 combined_and_lone_fibers = [segments;lone_fibers];
776 fiber_len_array = []; % zero array
777 for i = 1:length(combined_and_lone_fibers)
778     fiber_len = [combined_and_lone_fibers(i), ...
779         mean(table_1.Length(table_1.ContourID == ...
780             combined_and_lone_fibers(i)))];
781     fiber_len_array = vertcat(fiber_len_array, fiber_len);
782 end
783
784 % Sort the fibers based on their length
785 combined_and_lone_fibers = sortrows(fiber_len_array, 2);
786 C = parula(length(combined_and_lone_fibers));
787 % Overlay of the fibers and the original image
788 h = figure;
789 imshow(img);
790 hold on
791 for i = 1:length(combined_and_lone_fibers)

```

```

792         % figure;
793         % imshow(img);
794         % hold on
795         x1 = table_1.X(table_1.ContourID == combined_and_lone_fibers(i));
796         y1 = table_1.Y(table_1.ContourID == combined_and_lone_fibers(i));
797         % Plot dots instead of connected lines
798         plot(x1*x_scale + shift_x, y1*y_scale + shift_y, '.', 'markersize', 5,
↪ 'color', C(i, :));
799         % i % Plot the ID # i
800         % txt = {'\leftarrow #', num2str(combined_and_lone_fibers(i))};
801         % text(mean(x1*x_scale) + shift_x, mean(y1*y_scale) + shift_y,
↪ strcat(txt{1}, txt{2}));
802         title('\bf True Fibers');
803     end
804     plot(xv, yv, 'r--', 'LineWidth', 1.5)
805     plot(ILM.x, ILM.y, 'r--', 'LineWidth', 1.5)
806     title('\bf True Fibers!');
807     % Saves the figure as a Tif
808     saveas(h, strcat(file_name_root, file_name_extension, '.tif'));
809
810     %% Look at the matching fibers that were used to construct the complete
811     %% fiber
812     % for i = 1:length(fiber_segment)
813     %     figure;
814     %     imshow(img);
815     %     hold on
816     %     x1 = table_1.X(ContourID == fiber_segment(i, 1));
817     %     y1 = table_1.Y(ContourID == fiber_segment(i, 1));
818     %     x2 = table_1.X(ContourID == fiber_segment(i, 2));
819     %     y2 = table_1.Y(ContourID == fiber_segment(i, 2));
820     %     plot(x1*x_scale, y1*y_scale, 'r.', 'markersize', 5);
821     %     plot(x2*x_scale, y2*y_scale, 'bo', 'markersize', 10);
822     % end
823
824     % filtered out contour ID's that were too small
825     % ID_eliminated = unique(table_1.ContourID(table_1.Length <
↪ length_threshold));
826     % filtered out contour ID's that were too small
827     % ID_eliminated = unique(table_1.ContourID((table_1.Length <
↪ fiber_min_length)));
828
829     %%
830     filtered_fibers = length(short_segments);
831     fprintf('Filtered out %d fiber segments\n', filtered_fibers);
832     fprintf('Remaining eligible fibers = %d fibers\n', ...
833         length(segments));
834     fprintf('Total unique fibers = %d fibers\n', ...
835         length(combined_and_lone_fibers));
836
837
838
839     % Loop over all the current IDs that satisfy the criteria
840     for i = 1:length(combined_and_lone_fibers)
841         cur_x = table_1.X(find(table_1.ContourID == ...
842             combined_and_lone_fibers(i)));
843         cur_y = table_1.Y(find(table_1.ContourID == ...
844             combined_and_lone_fibers(i)));
845         %     cur_xRes = cur_x*x_scale + shift_x;

```

```

846 %      cur_yRes = cur_y*y_scale + shift_y;
847 Filt_Fibers_XY = [cur_x, cur_y];
848 %      Filt_Fibers_XYRes = [cur_xRes, cur_yRes];
849 Filt_Fibers(i).Length = unique(table_1.Length(table_1.ContourID == ...
850     combined_and_lone_fibers(i)));
851 Filt_Fibers(i).Width = table_1.LineWidth(table_1.ContourID == ...
852     combined_and_lone_fibers(i));
853 Filt_Fibers(i).ID = combined_and_lone_fibers(i);
854 Filt_Fibers(i).Area = Filt_Fibers(i).Length.*Filt_Fibers(i).Width; % Area
    ↪ of fibers
855
856 %      sort_cur_x = sort(table_1.X(combined_and_lone_fibers(i)));
857 %      sort_cur_y = sort(table_1.Y(combined_and_lone_fibers(i)));
858 %      angle = []; % clears the array during each loop
859 %      slope = []; % array of slopes
860 %      for j = 1:length(cur_x)-1
861 %          % Consider using the polyfit
862 %          numerator = (cur_y(j+1) - cur_y(j));
863 %          denominator = (cur_x(j+1) - cur_x(j));
864 %          % Calculates the fiber angle for each successive point in the fiber
865 %          % angle(j) = atan(numerator/denominator)*180/pi;
866 %          % Calculates the fiber angle for each successive point in the fiber
867 %          angle_calc = atan(numerator/denominator)*180/pi;
868 %          % if isnan(angle_calc)
869 %          % j
870 %          % fprintf('isnan\n');
871 %          % continue % bypass the angle that doesn't
872 %          % elseif (numerator == 0 && denominator == 0)
873 %          if (numerator == 0 && denominator == 0)
874 %              continue % bypass the angle that doesn't exist
875 %          elseif (denominator == 0)
876 %              %angle(j) = 90; % perpendicular line segments
877 %              angle = [angle;90];
878 %              continue
879 %          else
880 %              %slope(j) = numerator/denominator;
881 %              slope = [slope;numerator/denominator];
882 %              if slope(end) < 0 % slope(j) < 0
883 %                  % slopes are negative so add 180 degrees
884 %                  %angle(j) = angle(j) + 180;
885 %                  angle = [angle;angle_calc + 180];
886 %              end
887 %          end
888 %      end
889
890 % Calculate slope & y-intercept from linear fit
891 [F] = Least_Squares(Filt_Fibers_XY);
892 %Slope
893 Filt_Fibers(i).slope = F(2);
894 % inverse tangent of the slope
895 Filt_Fibers(i).Angle = -atan(F(2))*180/pi;
896 % Clear the dataset from the array for the next iteration
897 Filt_Fibers_XY = [];
898
899 % % average the slope for each individual contour ID
900 % Filt_Fibers(i).slope = mean(slope);
901 % %ILM_angle - ...
902 % Filt_Fibers(i).Angle = angle;

```



```

903         % Mean angle of each countour ID
904         Filt_Fibers(i).mean_Angle = mean(angle);
905     end
906
907     for i = 1:length(combined_and_lone_fibers)
908         % Puts each mean angle into an array
909         filt_ang(i) = Filt_Fibers(i).Angle;
910         % Calculates mean fiber length
911         filt_len(i) = Filt_Fibers(i).Length;
912         % Average width of the fiber and puts it into an array
913         filt_wid(i) = mean(Filt_Fibers(i).Width);
914         % Calculates the average fiber area (length*width of pixels)
915         filt_area(i) = mean(Filt_Fibers(i).Area);
916         % Number of points in each contour ID# and puts it into an array
917         filt_num(i) = length(Filt_Fibers(i).Width);
918         % Average slope of each contour ID#
919         filt_slo(i) = Filt_Fibers(i).slope;
920     end
921
922     for i = 1:length(short_segments)
923         cur_x = table_1.X(find(table_1.ContourID == ...
924             short_segments(i)));
925         cur_y = table_1.Y(find(table_1.ContourID == ...
926             short_segments(i)));
927         % cur_xRes = cur_x*x_scale + shift_x;
928         % cur_yRes = cur_y*y_scale + shift_y;
929         No_Filt_Fibers_XY = [cur_x, cur_y];
930         % No_Filt_Fibers(i).XYRes = [cur_xRes, cur_yRes];
931         No_Filt_Fibers(i).Length = unique(table_1.Length(table_1.ContourID == ...
932             short_segments(i)));
933         No_Filt_Fibers(i).Width = table_1.LineWidth(table_1.ContourID == ...
934             short_segments(i));
935         No_Filt_Fibers(i).ID = short_segments(i);
936
937         % Calculate slope & y-intercept from linear fit
938         [F] = Least_Squares(No_Filt_Fibers_XY);
939         %Slope
940         No_Filt_Fibers(i).slope = F(2);
941         % inverse tangent of the slope
942         No_Filt_Fibers(i).Angle = atan(F(2))*180/pi;
943         % Clear the dataset from the array for the next iteration
944         No_Filt_Fibers_XY = [];
945
946         % % average the slope for each individual contour ID
947         % No_Filt_Fibers(i).slope = mean(slope);
948         % %ILM_angle - ...
949         % No_Filt_Fibers(i).Angle = angle;
950         % % Mean angle of each countour ID
951         % % No_Filt_Fibers(i).mean_Angle = mean(angle);
952     end
953
954     for i = 1:length(short_segments)
955         % Puts each mean angle into an array
956         No_filt_ang(i) = No_Filt_Fibers(i).Angle;
957         % Puts each fiber length into an array
958         No_filt_len(i) = No_Filt_Fibers(i).Length;
959         % Average width of the fiber and puts it into an array
960         No_filt_wid(i) = mean(No_Filt_Fibers(i).Width);

```

```

961         % Number of points in each contour ID# and puts it into an array
962         No_filt_num(i) = length(No_Filt_Fibers(i).Width);
963         % Average slope of each contour ID#
964         No_filt_slo(i) = No_Filt_Fibers(i).slope;
965     end
966
967     % %%
968     % % Plot individual fibers on a single sheet
969     % %
970     % % Do not run this on a real image
971     % %
972     % C = hsv(length(segments)); % Color array for the fibers
973     % for i = 1:length(segments)
974     %     figure
975     %     imshow(img);
976     %     hold on
977     %     plot(Filt_Fibers(i).XYRes(:, 1), Filt_Fibers(i).XYRes(:, 2), '.',
↪ 'Color', C(i, :));
978     % end
979     % title('\bf Filterd image', 'fontsize', 18);
980     % %%
981     % [~, index] = sortrows([Filt_Fibers.Length].');
982     % Filt_Fibers = Filt_Fibers(index);
983     % clear index; % Sort the Filt_Fibers by Length
984     %
985     %
986     % % Plot individual fibers on the same sheet just pausing for half a second
987     % C = hsv(length(segments)); % Color array for the fibers
988     % figure
989     % imshow(img);
990     % hold on
991     % for i = 1:length(segments)
992     %     waitbar(i/length(segments));
993     %     plot(Filt_Fibers(i).XYRes(:, 1), Filt_Fibers(i).XYRes(:, 2), '.',
↪ 'Color', C(i, :));
994     %     % pause(0.01)
995     % end
996     % title('\bf Filterd image', 'fontsize', 18);
997     %
998     % %%
999     % [~, index] = sortrows([No_Filt_Fibers.Length].');
1000     % No_Filt_Fibers = No_Filt_Fibers(index);
1001     % clear index; % Sort the Filt_Fibers by Length
1002     %
1003     %
1004     % % Plot individual fibers on the same sheet just pausing for half a second
1005     % C = hsv(length(short_segments)); % Color array for the fibers
1006     % figure
1007     % imshow(img);
1008     % hold on
1009     % for i = 1:length(short_segments)
1010     %     waitbar(i/length(short_segments));
1011     %     plot(No_Filt_Fibers(i).XYRes(:, 1), No_Filt_Fibers(i).XYRes(:, 2), '.',
↪ 'Color', C(i, :));
1012     %     % pause(0.01)
1013     % end
1014     % title('\bf Non-Filterd image', 'fontsize', 18);
1015

```

```

1016
1017 % The combined_and_lone_fibers list needs to be sorted by fiber length
1018 % before calculating attributes such as slope, and angle
1019 for i = 1:length(combined_and_lone_fibers)
1020     % unique length of the connected fibers *1000 for nanometers
1021     len = unique(table_1.Length(table_1.ContourID ==
    ↪ combined_and_lone_fibers(i)));
1022     % converted average angle from y-axis to the x-axis -pi/2
1023     angle = (mean(table_1.AngleOfNormal(table_1.ContourID == ...
1024         combined_and_lone_fibers(i)))-pi)*180/pi;
1025     % angle from calculating the inverse tangent of the slope
1026     calc_ang = firt_ang(i);
1027     %difference in angle
1028     difference = angle - calc_ang;
1029     % density of collagen fibers / ilm length
1030     density(i) = firt_area(i)/ILM_length;
1031     fprintf(['Fiber # %d -- length = %.4f nanometers, -- ' ...
1032         'avg. angle RD = %.2f degrees, -- angle Calc = ' ...
1033         '%.2f degrees, -- angle diff %.2f\n'], ...
1034         combined_and_lone_fibers(i), len, angle, calc_ang, ...
1035         difference);
1036 end
1037 % density of collagen fibers / ilm length
1038 fprintf('Collagen fiber density = %f microns\n', sum(density));
1039
1040 % Plots the histogram of the calculated angles
1041 % figure
1042 % hist(firt_ang);
1043 % title('\bf Calculation of fiber angles');
1044 % fprintf(['Collagen fiber angle is %f \n ' ...
1045     % '(relative to the x-axis)\n'], mean(firt_ang));
1046
1047 % Plots the angle vs. fiber segment length
1048 %figure
1049 %plot(firt_ang, firt_len, '.');
1050 %set(gca, 'XDir', 'reverse');
1051 %xlabel('\bf Fiber Angle');
1052 %ylabel('\bf Fiber Length');
1053 %title('\bf Fiber Angle vs. Length');
1054
1055 % Plots the angle vs. fiber segment length on a polar grid
1056 %figure
1057 % plot(ang, len, '.');
1058 % pax = gca; % 2018a
1059 % pax.ThetaAxisUnits = 'radians'; % 2018a
1060 %polarplot(firt_ang*pi/180, firt_len, '.')
1061 % xlabel('\bf Fiber Angle'); % 2018a
1062 % ylabel('\bf Fiber Length'); % 2018a
1063 % axis([min(ang), max(ang), min(len), max(len)]);
1064 %title('\bf Fiber Angle vs. Length');
1065
1066 % Plot each unique fiber with a different color
1067 % figure
1068 % imshow(img);
1069 % hold on
1070 % C = hsv(length(unique(combined_and_lone_fibers)));
1071 % for i = 1:length(combined_and_lone_fibers)

```

```

1072     %      plot(All_Fibers(i).XYRes(:, 1), All_Fibers(i).XYRes(:, 2), '.', 'color',
↪ C(i, :), 'linewidth', 2);
1073     %      hold on;
1074     % end
1075     % axis image;
1076     % title('\bf Unique ContourID fiber identification');
1077
1078     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1079     % Plot the histogram of the image
1080     % figure
1081     % if synthetic == true
1082     %     img2 = rgb2gray(img); % Converts the RGB image to grayscale
1083     %     % [counts, grayLevels] = imhist(img, 256);
1084     %     imhist(img2); % Looks at the histogram of pixel intensities
1085     % else
1086     %     % [counts, grayLevels] = imhist(img, 256);
1087     %     imhist(img); % Looks at the histogram of pixel intensities
1088     % end
1089     % title('\bf Histogram of TEM image pixel intensities');
1090
1091     % Plot the contour map for the image overlayed with the detected fibers
1092     % h = figure;
1093     % image(img)
1094     % hold on
1095     % contourf(img, 10)
1096     % axis image
1097     % colormap gray
1098     % for i = 1:length(combined_and_lone_fibers)
1099     %     % figure;
1100     %     % imshow(img);
1101     %     hold on
1102     %     x1 = table_1.X(table_1.ContourID == combined_and_lone_fibers(i));
1103     %     y1 = table_1.Y(table_1.ContourID == combined_and_lone_fibers(i));
1104     %     plot(x1*x_scale + shift_x, y1*y_scale + shift_y, '.', 'markersize', 5,
↪ 'color', C(i, :)); % Plot dots instead of connected lines
1105     % %     txt = {'\leftarrow #', num2str(combined_and_lone_fibers(i))}; % i %
↪ Plot the ID # i
1106     % %     text(mean(x1*x_scale) + shift_x, mean(y1*y_scale) + shift_y,
↪ strcat(txt{1}, txt{2}));
1107     % %     title('\bf True Fibers');
1108     % end
1109     %title('\bf True Fibers overlayed on a contour filled plot!');
1110     %saveas(h, strcat(file_name_root, file_name_extension, '_contour.tif')); %
↪ Saves the figure as a Tif
1111
1112     fprintf(fileID, 'Total unique fibers = %d fibers\n', ...
1113         length(combined_and_lone_fibers));
1114     fprintf(fileID, ...
1115         'Width of the rectangle ILM measurement = %d microns\n', ...
1116         ILM_length);
1117     fprintf(fileID, ...
1118         'ILM angle is %f degrees \n (relative to the x-axis)\n', ...
1119         ILM_angle);
1120     fprintf(fileID, ...
1121         'Average ILM thickness is %f nanometers \n', ...
1122         ILM_thickness);
1123     fprintf(fileID, 'Collagen fiber count density = %f \n', ...
1124         length(combined_and_lone_fibers)/ILM_length);

```

```

1125 fprintf(fileID, ...
1126     ['Abs Mean Collagen fiber angle is %f \n ' ...
1127     '(relative to the x-axis)\n'], ...
1128     nanmean(abs(filt_ang)));
1129 fprintf(fileID, ...
1130     ['Abs Median Collagen fiber angle is %f \n ' ...
1131     '(relative to the x-axis)\n'], ...
1132     nanmedian(abs(filt_ang)));
1133 fprintf(fileID, ...
1134     ['Abs Mean Collagen fiber angle is %f \n ' ...
1135     '(relative to the ILM)\n'], ...
1136     nanmean(abs(filt_ang-ILM_angle)));
1137 fprintf(fileID, ...
1138     ['Abs Median Collagen fiber angle is %f \n ' ...
1139     '(relative to the ILM)\n'], ...
1140     nanmedian(abs(filt_ang-ILM_angle)));
1141
1142 %fprintf(fileID, 'ILM slope = %f\n', ILM_slope);
1143 %fprintf(fileID, 'ILM length = %f microns\n', ILM_length);
1144
1145 %fprintf(fileID, 'Mimimum fiber length is %f microns\n', fiber_min_length);
1146
1147 %fprintf(fileID, 'Filtered out %d fiber segments\n', filtered_fibers);
1148 %fprintf(fileID, 'Remaining eligible fibers = %d fibers\n',
↪ length(segments));
1149
1150 % for i = 1:length(combined_and_lone_fibers)
1151 %     fprintf(fileID, 'Fiber # %d -- length = %.4f nanometers, -- avg. angle
↪ RD = %.2f degrees, -- angle Calc = %.2f degrees, -- angle diff %.2f\n',
↪ combined_and_lone_fibers(i), len, angle, calc_ang, difference);
1152 % end
1153 %fprintf(fileID, 'Collagen fiber density = %f microns\n', sum(density)); %
↪ density of collagen fibers / ilm length
1154
1155 fprintf(fileID, 'Average collagen fiber length = %f microns\n', ...
1156     mean(filt_len));
1157 fclose(fileID); % close the txt file for the output information
1158
1159 % Saves the new table with Original Fibril & New Fibril data
1160 writetable(table_1, strcat(file_name_root, file_name_extension, ...
1161     '_Original_and_New_FibrilData', '.csv'))
1162
1163
1164 case 'No'
1165     %Calculate only ILM thickness if no collagen
1166     figure
1167     imshow(img);
1168     % Indicate the five points on the ILM used for thickness measurements
1169     for i = 1:5
1170         f = msgbox(['Select the first two points that define ' ...
1171             'the ILM thickness'], 'ILM');
1172         %     pause(1);
1173         [ILM_thick(i).x, ILM_thick(i).y] = ginput(2);
1174         hold on
1175         plot(ILM_thick(i).x, ILM_thick(i).y, 'g-o', 'linewidth', 1);
1176         % Pythagorean theorem
1177         ILM_thick(i).measurement = sqrt((ILM_thick(i).x(1) - ILM_thick(i).x(2))^2
↪ + ...

```

```

1178         (ILM_thick(i).y(1) - ILM_thick(i).y(2))^2);
1179         delete(f); % Delete the message box
1180     end
1181     for i = 1:5
1182         ILM_measurement(i) = ILM_thick(i).measurement;
1183     end
1184     L{4} = 'ILM thickness measurements';
1185     axis image;
1186     ILM_thickness = mean(ILM_measurement)/x_scale*1000;
1187     fprintf('Average ILM thickness is %f nanometers \n', ILM_thickness);
1188 end

```

1.5 Human Data Analysis

</> **Script 4:** *Python script analyzes human data, performs statistics, and creates figures.* </>

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Nov 23 21:48:15 2020
4
5  @author: Kiffer Creveling
6  """
7
8  import pandas as pd
9  import os
10 import numpy as np
11 import seaborn as sns
12 from statannot import add_stat_annotation
13 import matplotlib.pyplot as plt
14 from matplotlib.patches import PathPatch
15 plt.rcParams['figure.figsize'] = [16, 10]
16 from scipy import stats
17 import pdb
18
19 # In[Functions]
20
21 # fcn for plotting
22 def yfit(x):
23     return slope*x + intercept
24
25 # In[Read values from Database]
26 """ Read from the database """
27
28 df = pd.read_csv('JMP_Data.csv') # Data from JMP
29 df = pd.read_excel('Human Data Paper 2 TEM only (Updated Jul 10 2020).xlsx',
30                   engine='openpyxl')
31 df = pd.read_excel('Human Data Paper 2 TEM only (Updated April 17 2021).xlsx',
32                   engine='openpyxl')
33
34 """ Simplification of code """
35 SF = 'StatisticsFigures' # Figure directory
36 TMD = 'TEM Mean Density'
37 TMA = 'TEM Mean Angle'
38 TAA = 'TEM Angle ABS'

```

```

39 ILM = 'ILM Thickness (nm)'
40 FL = 'Fiber Length (um)'
41 MPF = 'Maximum peel force (mN)'
42 mpf_mN = 'Max peel force (mN)'
43 R = 'Region'
44 Eq = 'Equator'
45 Po = 'Posterior'
46 AG = 'AgeGroup'
47 A60 = 'Age60'
48 Aleq60 = r'Age  $\leq$  60'
49 Ag60 = 'Age  $>$  60'
50 A = 'Age'
51 MN = 'Max [N]'
52 MmN = 'Max [mN]'
53 SSN = 'SS [N]'
54 SSmN = 'SS [mN]'
55
56 # Plot attributes (labels, etc)
57 A_yrs = 'Age (yr.)'
58 A_G = 'Age Group (yr.)'
59 DensityUnit = (r'Collagen Fibril Density
    ↳  $\left(\frac{\text{# of fibrils}}{\text{ILM length (nm)}}\right)$ ')
60 FibrilLengthUnit = r'Collagen Fibril length ( $\mu\text{m}$ )'
61 OrientationUnit = r'Collagen Fibril Angle Relative to the ILM ( $^\circ$ )'
62
63 # convert from N to mN
64 df[mpf_mN] = df[MN]*1000
65 df[SSmN] = df[SSN]*1000
66
67 # Exclude the cells that have duplicates or have been excluded due to
68 # video analysis
69 df = df[df['Excluded'] != 'yes']
70
71 # In[Create AgeGroup bins]
72 bins = [30, 40, 50, 60, 70, 80, 90]
73 labels = ['30-39', '40-49', '50-59', '60-69', '70-79', '80-89']
74 # Create binned AgeGroups
75
76 df[AG] = pd.cut(df[A], bins, labels=labels, right=False)
77
78 bins = [0, 60, 90]
79 labels = [Aleq60, Ag60]
80 # Create binned AgeGroups
81 df[A60] = pd.cut(df[A], bins, labels=labels, right=True)
82
83 # In[Pivot Table]
84 # Simplify pivot table output
85
86 pvtOut = {'count', np.median, np.mean, np.std} # pivot table outputs
87
88 # In[Plots]
89
90 standardError = 68 # Used for confidence intervals
91
92 sns.set_theme(context='paper', style='darkgrid', palette="Paired",
93               font_scale=2)
94 custom_style = {'axes.facecolor': 'white',
95                 'axes.edgecolor': 'black',

```

```

96         'axes.grid': False,
97         'axes.axisbelow': True,
98         'axes.labelcolor': 'black',
99         'figure.facecolor': 'white',
100        'grid.color': '.8',
101        'grid.linestyle': '-',
102        'text.color': 'black',
103        'xtick.color': 'black',
104        'ytick.color': 'black',
105        'xtick.direction': 'out',
106        'ytick.direction': 'out',
107        'lines.solid_capstyle': 'round',
108        'patch.edgecolor': 'w',
109        'patch.force_edgecolor': True,
110        'image.cmap': 'rocket',
111        'font.family': ['sans-serif'],
112        'font.sans-serif': ['Arial', 'DejaVu Sans', 'Liberation Sans',
113                           'Bitstream Vera Sans', 'sans-serif'],
114        'xtick.bottom': True,
115        'xtick.top': False,
116        'ytick.left': True,
117        'ytick.right': False,
118        'axes.spines.left': True,
119        'axes.spines.bottom': True,
120        'axes.spines.right': False,
121        'axes.spines.top': False}
122 # White background with ticks and black border lines, Turns grid off
123 ax = sns.set_style(rc=custom_style)
124
125 def boxPlotBlackBorder(ax):
126     # iterate over boxes in the plot to make each line black
127     for i,box in enumerate(ax.artists):
128         box.set_edgecolor('black')
129         # box.set_facecolor('white')
130
131     # iterate over whiskers and median lines
132     for j in range(6*i, 6*(i+1)):
133         ax.lines[j].set_color('black')
134
135 def smartPlot(data=None, x=None, y=None, hue=None, hue_order=None,
136              addBoxPair=None, ci=None, errcolor=None, capsize=None,
137              plot=None, test=None, sigLoc=None, text_format=None,
138              line_offset=None, line_offset_to_box=None, line_height=None,
139              fontsize=None, legLoc=None, verbose=None, xlabel=None,
140              ylabel=None, legendTitle=None, figName=None, folderName=None,
141              dataPoints=None):
142
143     # barplot
144     f, ax = plt.subplots()
145
146     if plot == 'barplot':
147         ax = sns.barplot(data=data, x=x, y=y, hue=hue, hue_order=hue_order,
148                          ci=ci, errcolor=errcolor, capsize=capsize)
149
150     elif plot == 'boxplot':
151         ax = sns.boxplot(data=data, x=x, y=y, hue=hue, hue_order=hue_order)
152
153     # Statistical test for differences

```



```

154 x_grps = list(data[x].unique()) # List of groups
155 if hue != None:
156     # Create combinations to compare
157     box_pairs_1 = [(x_grps_i, hue_order[0]),
158                   (x_grps_i, hue_order[1]))
159                 for x_grps_i in x_grps]
160     box_pairs = box_pairs_1
161
162     if addBoxPair != None:
163         # Additional box pairs
164         box_pairs = box_pairs_1 + addBoxPair
165
166 elif hue_order != None:
167     box_pairs = [(hue_order[0], hue_order[1])]
168
169 #Stats results and significant differences (SR)
170 SR = add_stat_annotation(ax, plot=plot, data=data, x=x, y=y, hue=hue,
171                          hue_order=hue_order, box_pairs=box_pairs,
172                          test=test, loc=sigLoc, text_format=text_format,
173                          verbose=verbose, comparisons_correction=None,
174                          line_offset=line_offset,
175                          line_offset_to_box=line_offset_to_box,
176                          line_height= line_height,
177                          fontsize=fontsize) # 'bonferroni'
178
179 if plot == 'boxplot':
180     boxPlotBlackBorder(ax) # Make borders black
181
182
183 if dataPoints == True:
184     # Add data points to the box plot
185     sns.stripplot(data=data, x=x, y=y, hue=hue, hue_order=hue_order,
186                  color='.5', size=5, linewidth=1, dodge=True)
187
188     # gather plot attributes for legends
189     handles, labels = ax.get_legend_handles_labels()
190
191     if hue != None:
192         l = plt.legend(handles[0:2], labels[0:2], title=legendTitle)
193
194 else:
195     if hue != None:
196         ax.legend(loc=legLoc).set_title(legendTitle)
197
198 ax.set_xlabel(xlabel)
199 ax.set_ylabel(ylabel)
200 ax = sns.despine() # takes the lines off on the right and top of the graph
201
202 if folderName != None:
203     # If a new folder name is given, put the files there
204
205     # New file path
206     NP = os.path.join(SF, folderName)
207
208     # Create folder if it doesn't exist
209     os.makedirs(NP, exist_ok=True)
210
211 else:

```

```

212     # Put the file in the same folder
213     NP = SF
214
215     f.savefig(os.path.join(NP, '{}.pdf'.format(figName)),
216               bbox_inches='tight')
217     plt.close()
218
219     # Special spacing
220
221     def adjust_box_widths(g, fac):
222         """
223         Adjust the widths of a seaborn-generated boxplot.
224         """
225
226         # iterating through Axes instances
227         for ax in g.axes:
228
229             # iterating through axes artists:
230             for c in ax.get_children():
231
232                 # searching for PathPatches
233                 if isinstance(c, PathPatch):
234                     # getting current width of box:
235                     p = c.get_path()
236                     verts = p.vertices
237                     verts_sub = verts[:-1]
238                     xmin = np.min(verts_sub[:, 0])
239                     xmax = np.max(verts_sub[:, 0])
240                     xmid = 0.5*(xmin + xmax)
241                     xhalf = 0.5*(xmax - xmin)
242
243                     # setting new width of box
244                     xmin_new = xmid - fac*xhalf
245                     xmax_new = xmid + fac*xhalf
246                     verts_sub[verts_sub[:, 0] == xmin, 0] = xmin_new
247                     verts_sub[verts_sub[:, 0] == xmax, 0] = xmax_new
248
249                     # setting new width of median line
250                     for l in ax.lines:
251                         if np.all(l.get_xdata() == [xmin, xmax]):
252                             l.set_xdata([xmin_new, xmax_new])
253
254     # In[TEM mean density by age +/- 60 and region]
255
256     """ TEM mean density by age +/- 60 and region """
257
258     pivotTEM_MeanDensityAgeGroup60 = pd.pivot_table(df, values=TMD,
259                                                       index=[A60, R],
260                                                       aggfunc=pvtOut)
261
262     print('pivotTEM_MeanDensityAgeGroup60')
263     print(pivotTEM_MeanDensityAgeGroup60)
264     # Add the index groups and convert NaN's to "-"'s
265     print(pivotTEM_MeanDensityAgeGroup60.to_latex(index=True, na_rep='-',
266                                                    escape=False,
267                                                    float_format="{:0.3f}".format))
268
269     Folder = 'Density_Age60Region'

```

```

270
271 # Barplot
272 smartPlot(data=df, x=A60, y=TMD, hue=R, hue_order=[Eq, Po], ci='sd',
273           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
274           sigLoc='outside', text_format='star', line_offset=0.0,
275           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
276           legLoc='best', verbose=2,
277           xlabel=A_G, ylabel=DensityUnit, legendTitle=R,
278           figName='BarPlot', folderName=Folder)
279
280 # Boxplot
281 smartPlot(data=df, x=A60, y=TMD, hue=R, hue_order=[Eq, Po], plot='boxplot',
282           test='t-test_ind', text_format='star', sigLoc='outside',
283           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
284           fontsize='small', legLoc='best', verbose=2,
285           xlabel=A_G, ylabel=DensityUnit,
286           legendTitle=R, figName='BoxPlot', folderName=Folder)
287
288 # Boxplot with data
289 smartPlot(data=df, x=A60, y=TMD, hue=R, hue_order=[Eq, Po], plot='boxplot',
290           test='t-test_ind', sigLoc='outside', text_format='star',
291           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
292           fontsize='small', legLoc='best', verbose=2,
293           xlabel=A_G, ylabel=DensityUnit,
294           legendTitle=R, figName='BoxPlotWithData', folderName=Folder,
295           dataPoints=True)
296
297
298 # In[TEM mean density grouped by region]
299
300 """ TEM mean density """
301
302 pivotTEM_MeanDensityRegion = pd.pivot_table(df, values=TMD, index=[R],
303                                              aggfunc=pvtOut)
304
305 print('pivotTEM_MeanDensityRegion')
306 print(pivotTEM_MeanDensityRegion)
307 # Add the index groups and convert NaN's to '-'s
308 print(pivotTEM_MeanDensityRegion.to_latex(index=True, na_rep='-',
309                                           escape=False,
310                                           float_format="{:0.3f}".format))
311
312 Folder = 'Density_Region'
313
314 # Barplot
315 smartPlot(data=df, x=R, y=TMD, hue=None, hue_order=[Eq, Po], ci='sd',
316           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
317           sigLoc='outside', text_format='star', line_offset=0.0,
318           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
319           legLoc='best', verbose=2,
320           xlabel=R, ylabel=DensityUnit, legendTitle=R,
321           figName='BarPlot', folderName=Folder)
322
323 # Boxplot
324 smartPlot(data=df, x=R, y=TMD, hue=None, hue_order=[Eq, Po], plot='boxplot',
325           test='t-test_ind', sigLoc='outside', text_format='star',
326           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
327           fontsize='small', legLoc='best', verbose=2,

```

```

328         xlabel=R, ylabel=DensityUnit,
329         legendTitle=R, figName='BoxPlot', folderName=Folder)
330
331     # Boxplot with data
332     smartPlot(data=df, x=R, y=TMD, hue=None, hue_order=[Eq, Po], plot='boxplot',
333             test='t-test_ind', sigLoc='outside', text_format='star',
334             line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
335             fontsize='small', legLoc='best', verbose=2,
336             xlabel=R, ylabel=DensityUnit,
337             legendTitle=R, figName='BoxPlotWithData', folderName=Folder,
338             dataPoints=True)
339
340     # matched_pairs student's t-test
341     dfTMD = df[df[TMD].notna()]
342
343     dfMP = dfTMD[dfTMD.duplicated(['MatchingID'], keep=False)]
344     f, p = stats.ttest_rel(dfMP[TMD][dfMP[R] == Eq],
345                           dfMP[TMD][dfMP[R] == Po])
346
347     print(f, p, "Matched Pairs Student's t-test")
348
349     f, p = stats.ttest_ind(dfTMD[TMD][dfTMD[R] == Eq],
350                           dfTMD[TMD][dfTMD[R] == Po])
351
352     print(f, p, "Student's t-test")
353
354     # In[TEM mean density grouped by age group decade and region]
355
356     pivotTEM_MeanDensity = pd.pivot_table(df, values=TMD, index=[R, AG],
357                                           aggfunc=pvtOut)
358
359     print('pivotTEM_MeanDensity')
360     print(pivotTEM_MeanDensity)
361     # Add the index groups and convert NaN's to "-"'s
362     print(pivotTEM_MeanDensity.to_latex(index=True, na_rep='-', escape=False,
363                                         float_format="{:0.3f}".format))
364
365     Folder = 'Density_AgeDecadeRegion'
366
367     # Barplot
368     smartPlot(data=df, x=AG, y=TMD, hue=R, hue_order=[Eq, Po], ci='sd',
369             errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
370             sigLoc='outside', text_format='star', line_offset=0.0,
371             line_offset_to_box=0.0, line_height=0.015, fontsize='small',
372             legLoc='best', verbose=2,
373             xlabel=A_G, ylabel=DensityUnit, legendTitle=R,
374             figName='BarPlot', folderName=Folder)
375
376     # Boxplot
377     smartPlot(data=df, x=AG, y=TMD, hue=R, hue_order=[Eq, Po], plot='boxplot',
378             test='t-test_ind', sigLoc='outside', text_format='star',
379             line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
380             fontsize='small', legLoc='best', verbose=2,
381             xlabel=A_G, ylabel=DensityUnit,
382             legendTitle=R, figName='BoxPlot', folderName=Folder)
383
384     # Boxplot with data
385     smartPlot(data=df, x=AG, y=TMD, hue=R, hue_order=[Eq, Po], plot='boxplot',

```

```

386         test='t-test_ind', sigLoc='outside', text_format='star',
387         line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
388         fontsize='small', legLoc='best', verbose=2,
389         xlabel=A_G, ylabel=DensityUnit,
390         legendTitle=R,
391         figName='BoxPlotWithData', folderName=Folder,
392         dataPoints=True)
393
394 # In[ILM thickness vs region age +/- 60]
395
396 """ TEM ILM thickness vs region age +/- 60 """
397
398 pivotTEM_ILM_ThicknessAge60 = pd.pivot_table(df, values=ILM, index=[A60, R],
399                                               aggfunc=pvtOut)
400
401 print('pivotTEM_ILM_ThicknessAge60')
402 print(pivotTEM_ILM_ThicknessAge60)
403 # Add the index groups and convert NaN's to "-"'s
404 print(pivotTEM_ILM_ThicknessAge60.to_latex(index=True, na_rep='-',
405                                           escape=False,
406                                           float_format="{:0.3f}".format))
407
408 Folder = 'ILM_Age60Region'
409
410 # Barplot
411 smartPlot(data=df, x=A60, y=ILM, hue=R, hue_order=[Eq, Po], ci='sd',
412           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
413           sigLoc='outside', text_format='star', line_offset=0.0,
414           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
415           legLoc='best', verbose=2,
416           xlabel=A_G, ylabel=ILM, legendTitle=R,
417           figName='BarPlot', folderName=Folder)
418
419 # Boxplot
420 smartPlot(data=df, x=A60, y=ILM, hue=R, hue_order=[Eq, Po], plot='boxplot',
421           test='t-test_ind', sigLoc='outside', text_format='star',
422           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
423           fontsize='small', legLoc='best', verbose=2,
424           xlabel=A_G, ylabel=ILM,
425           legendTitle=R, figName='BoxPlot', folderName=Folder)
426
427 # Boxplot with data
428 smartPlot(data=df, x=A60, y=ILM, hue=R, hue_order=[Eq, Po], plot='boxplot',
429           test='t-test_ind', sigLoc='outside', text_format='star',
430           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
431           fontsize='small', legLoc='best', verbose=2,
432           xlabel=A_G, ylabel=ILM,
433           legendTitle=R, figName='BoxPlotWithData', folderName=Folder,
434           dataPoints=True)
435
436
437 # In[ILM thickness vs region age group]
438
439 """ ILM thickness vs region and age group """
440
441 pivotTEM_ILM_Thickness = pd.pivot_table(df, values=ILM, index=[AG, R],
442                                         aggfunc=pvtOut)
443

```

```

444 print('pivotTEM_ILM_Thickness')
445 print(pivotTEM_ILM_Thickness)
446 # Add the index groups and convert NaN's to "-"'s
447 print(pivotTEM_ILM_Thickness.to_latex(index=True, na_rep='-',
448                                     escape=False,
449                                     float_format="{:0.3f}".format))
450
451 Folder = 'ILM_Region'
452
453 # Barplot
454 smartPlot(data=df, x=AG, y=ILM, hue=R, hue_order=[Eq, Po], ci=68,
455           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
456           sigLoc='outside', text_format='star', line_offset=0.0,
457           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
458           legLoc='best', verbose=2,
459           xlabel=A_G, ylabel=ILM, legendTitle=R,
460           figName='BarPlot', folderName=Folder)
461
462 # Boxplot
463 smartPlot(data=df, x=AG, y=ILM, hue=R, hue_order=[Eq, Po], plot='boxplot',
464           test='t-test_ind', sigLoc='outside', text_format='star',
465           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
466           fontsize='small', legLoc='best', verbose=2,
467           xlabel=A_G, ylabel=ILM,
468           legendTitle=R, figName='BoxPlot', folderName=Folder)
469
470 # Boxplot with data
471 smartPlot(data=df, x=AG, y=ILM, hue=R, hue_order=[Eq, Po], plot='boxplot',
472           test='t-test_ind', sigLoc='outside', text_format='star',
473           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
474           fontsize='small', legLoc='best', verbose=2,
475           xlabel=A_G, ylabel=ILM,
476           legendTitle=R,
477           figName='BoxPlotWithData', folderName=Folder,
478           dataPoints=True)
479
480
481 # In[ILM fiber length vs region age group decade]
482
483 """ TEM ILM fiber length """
484
485 pivotTEM_FiberLength = pd.pivot_table(df, values=FL, index=[AG, R],
486                                       aggfunc=pvtOut)
487
488 print('pivotTEM_FiberLength')
489 print(pivotTEM_FiberLength)
490 # Add the index groups and convert NaN's to "-"'s
491 print(pivotTEM_FiberLength.to_latex(index=True, na_rep='-',
492                                     escape=False,
493                                     float_format="{:0.3f}".format))
494
495 Folder = 'FibrilLength_AgeDecadeRegion'
496
497 # Barplot
498 smartPlot(data=df, x=AG, y=FL, hue=R, hue_order=[Eq, Po], ci=68,
499           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
500           sigLoc='outside', text_format='star', line_offset=0.0,
501           line_offset_to_box=0.0, line_height=0.015, fontsize='small',

```

```

502         legLoc='best', verbose=2,
503         xlabel=A_G, ylabel=FibrilLengthUnit, legendTitle=R,
504         figName='BarPlot', folderName=Folder)
505
506     # Boxplot
507     smartPlot(data=df, x=AG, y=FL, hue=R, hue_order=[Eq, Po], plot='boxplot',
508             test='t-test_ind', sigLoc='outside', text_format='star',
509             line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
510             fontsize='small', legLoc='best', verbose=2,
511             xlabel=A_G, ylabel=FibrilLengthUnit,
512             legendTitle=R, figName='BoxPlot', folderName=Folder)
513
514     # Boxplot with data
515     smartPlot(data=df, x=AG, y=FL, hue=R, hue_order=[Eq, Po], plot='boxplot',
516             test='t-test_ind', sigLoc='outside', text_format='star',
517             line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
518             fontsize='small', legLoc='best', verbose=2,
519             xlabel=A_G, ylabel=FibrilLengthUnit,
520             legendTitle=R,
521             figName='BoxPlotWithData', folderName=Folder,
522             dataPoints=True)
523
524
525     # In[ILM fiber length vs region age group +/- 60]
526
527     """ TEM ILM fiber length """
528
529     pivotTEM_FiberLengthAge60 = pd.pivot_table(df, values=FL, index=[A60, R],
530             aggfunc=pvtOut)
531
532     print('pivotTEM_FiberLengthAge60')
533     print(pivotTEM_FiberLengthAge60)
534     # Add the index groups and convert NaN's to '-'
535     print(pivotTEM_FiberLengthAge60.to_latex(index=True, na_rep='-',
536             escape=False,
537             float_format="{:0.3f}".format))
538
539     Folder = 'FibrilLength_Age60Region'
540
541     # Barplot
542     smartPlot(data=df, x=A60, y=FL, hue=R, hue_order=[Eq, Po], ci=68,
543             errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
544             sigLoc='outside', text_format='star', line_offset=0.0,
545             line_offset_to_box=0.0, line_height=0.015, fontsize='small',
546             legLoc='best', verbose=2,
547             xlabel=A_G, ylabel=FibrilLengthUnit, legendTitle=R,
548             figName='BarPlot', folderName=Folder)
549
550     # Boxplot
551     smartPlot(data=df, x=A60, y=FL, hue=R, hue_order=[Eq, Po], plot='boxplot',
552             test='t-test_ind', sigLoc='outside', text_format='star',
553             line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
554             fontsize='small', legLoc='best', verbose=2,
555             xlabel=A_G, ylabel=FibrilLengthUnit,
556             legendTitle=R, figName='BoxPlot', folderName=Folder)
557
558     # Boxplot with data
559     smartPlot(data=df, x=A60, y=FL, hue=R, hue_order=[Eq, Po], plot='boxplot',

```

```

560         test='t-test_ind', sigLoc='outside', text_format='star',
561         line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
562         fontsize='small', legLoc='best', verbose=2,
563         xlabel=A_G, ylabel=FibrilLengthUnit,
564         legendTitle=R,
565         figName='BoxPlotWithData', folderName=Folder,
566         dataPoints=True)
567
568 # In[TEM Absolute Angle by age +/- 60 and region]
569
570 """ TEM Absolute Angle """
571
572 pivotTEM_MeanAngleABSAgeGroup60 = pd.pivot_table(df, values=TAA,
573                                                    index=[A60, R],
574                                                    aggfunc=pvtOut)
575
576 print('pivotTEM_MeanAngleABSAgeGroup60')
577 print(pivotTEM_MeanAngleABSAgeGroup60)
578 # Add the index groups and convert NaN's to "-"'s
579 print(pivotTEM_MeanAngleABSAgeGroup60.to_latex(index=True, na_rep='-',
580                                                  escape=False,
581                                                  float_format="{:0.3f}".format))
582
583 Folder = 'ABSAngle_Age60Region'
584
585 # Barplot
586 smartPlot(data=df, x=A60, y=TAA, hue=R, hue_order=[Eq, Po], ci=68,
587           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
588           sigLoc='outside', text_format='star', line_offset=0.0,
589           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
590           legLoc='best', verbose=2,
591           xlabel=A_G, ylabel=OrientationUnit, legendTitle=R,
592           figName='BarPlot', folderName=Folder)
593
594 # Boxplot
595 smartPlot(data=df, x=A60, y=TAA, hue=R, hue_order=[Eq, Po], plot='boxplot',
596           test='t-test_ind', sigLoc='outside', text_format='star',
597           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
598           fontsize='small', legLoc='best', verbose=2,
599           xlabel=A_G, ylabel=OrientationUnit,
600           legendTitle=R, figName='BoxPlot', folderName=Folder)
601
602 # Boxplot with data
603 smartPlot(data=df, x=A60, y=TAA, hue=R, hue_order=[Eq, Po], plot='boxplot',
604           test='t-test_ind', sigLoc='outside', text_format='star',
605           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
606           fontsize='small', legLoc='best', verbose=2,
607           xlabel=A_G, ylabel=OrientationUnit,
608           legendTitle=R,
609           figName='BoxPlotWithData', folderName=Folder,
610           dataPoints=True)
611
612
613 # In[TEM angle]
614
615 pivotTEM_MeanAngle = pd.pivot_table(df, values=TMA, index=[R, AG],
616                                     aggfunc=pvtOut)
617

```



```

618 print('pivotTEM_MeanAngle')
619 print(pivotTEM_MeanAngle)
620 # Add the index groups and convert NaN's to '-'
621 print(pivotTEM_MeanAngle.to_latex(index=True, na_rep='-',
622                                     escape=False,
623                                     float_format="{:0.3f}".format))
624
625 OrientationUnitNoAbs = r'ILM angle  $(^{\circ})$ '
626 Folder = 'Angle_AgeRegion'
627
628 # Barplot
629 smartPlot(data=df, x=AG, y=TMA, hue=R, hue_order=[Eq, Po], ci=68,
630           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
631           sigLoc='outside', text_format='star', line_offset=0.0,
632           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
633           legLoc='best', verbose=2,
634           xlabel=A_G, ylabel=OrientationUnitNoAbs, legendTitle=R,
635           figName='BarPlot', folderName=Folder)
636
637 # Boxplot
638 smartPlot(data=df, x=AG, y=TMA, hue=R, hue_order=[Eq, Po], plot='boxplot',
639           test='t-test_ind', sigLoc='outside', text_format='star',
640           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
641           fontsize='small', legLoc='best', verbose=2,
642           xlabel=A_G, ylabel=OrientationUnitNoAbs,
643           legendTitle=R, figName='BoxPlot', folderName=Folder)
644
645 # Boxplot with data
646 smartPlot(data=df, x=AG, y=TMA, hue=R, hue_order=[Eq, Po], plot='boxplot',
647           test='t-test_ind', sigLoc='outside', text_format='star',
648           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
649           fontsize='small', legLoc='best', verbose=2,
650           xlabel=A_G, ylabel=OrientationUnitNoAbs,
651           legendTitle=R,
652           figName='BoxPlotWithData', folderName=Folder,
653           dataPoints=True)
654
655 # In[TEM ABS angle by age decade group and region]
656
657 pivotTEM_MeanAngleABS = pd.pivot_table(df, values=TAA, index=[R, AG],
658                                         aggfunc=pvtOut)
659 print('pivotTEM_MeanAngleABS')
660 print(pivotTEM_MeanAngleABS)
661 # Add the index groups and convert NaN's to '-'
662 print(pivotTEM_MeanAngleABS.to_latex(index=True, na_rep='-',
663                                       escape=False,
664                                       float_format="{:0.3f}".format))
665
666 Folder = 'ABSAngle_AgeDecadeRegion'
667
668 # Barplot
669 smartPlot(data=df, x=AG, y=TAA, hue=R, hue_order=[Eq, Po], ci=68,
670           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
671           sigLoc='outside', text_format='star', line_offset=0.0,
672           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
673           legLoc='best', verbose=2,
674           xlabel=A_G, ylabel=OrientationUnit, legendTitle=R,
675           figName='BarPlot', folderName=Folder)

```

```

676
677 # Boxplot
678 smartPlot(data=df, x=AG, y=TAA, hue=R, hue_order=[Eq, Po], plot='boxplot',
679           test='t-test_ind', sigLoc='outside', text_format='star',
680           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
681           fontsize='small', legLoc='best', verbose=2,
682           xlabel=A_G, ylabel=OrientationUnit,
683           legendTitle=R, figName='BoxPlot', folderName=Folder)
684
685 # Boxplot with data
686 smartPlot(data=df, x=AG, y=TAA, hue=R, hue_order=[Eq, Po], plot='boxplot',
687           test='t-test_ind', sigLoc='outside', text_format='star',
688           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
689           fontsize='small', legLoc='best', verbose=2,
690           xlabel=A_G, ylabel=OrientationUnit,
691           legendTitle=R,
692           figName='BoxPlotWithData', folderName=Folder,
693           dataPoints=True)
694
695 # In[TEM absolute angle by region]
696
697 pivotTEM_MeanAngleABSRegion = pd.pivot_table(df, values=TAA, index=[R],
698                                               aggfunc=pvtOut)
699
700 print('pivotTEM_MeanAngleABSRegion')
701 print(pivotTEM_MeanAngleABSRegion)
702 # Add the index groups and convert NaN's to "-"'s
703 print(pivotTEM_MeanAngleABSRegion.to_latex(index=True, na_rep='-',
704                                             escape=False,
705                                             float_format="{:0.3f}".format))
706
707 Folder = 'ABSAngle_Region'
708
709 # Barplot
710 smartPlot(data=df, x=R, y=TAA, hue=None, hue_order=[Eq, Po], ci=68,
711           errcolor='black', capsize=.2, plot='barplot', test='t-test_ind',
712           sigLoc='outside', text_format='star', line_offset=0.0,
713           line_offset_to_box=0.0, line_height=0.015, fontsize='small',
714           legLoc='best', verbose=2,
715           xlabel=R, ylabel=OrientationUnit, legendTitle=R,
716           figName='BarPlot', folderName=Folder)
717
718 # Boxplot
719 smartPlot(data=df, x=R, y=TAA, hue=None, hue_order=[Eq, Po], plot='boxplot',
720           test='t-test_ind', sigLoc='outside', text_format='star',
721           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
722           fontsize='small', legLoc='best', verbose=2,
723           xlabel=R, ylabel=OrientationUnit,
724           legendTitle=R, figName='BoxPlot', folderName=Folder)
725
726 # Boxplot with data
727 smartPlot(data=df, x=R, y=TAA, hue=None, hue_order=[Eq, Po], plot='boxplot',
728           test='t-test_ind', sigLoc='outside', text_format='star',
729           line_offset=0.0, line_offset_to_box=0.0, line_height=0.015,
730           fontsize='small', legLoc='best', verbose=2,
731           xlabel=R, ylabel=OrientationUnit,
732           legendTitle=R,
733           figName='BoxPlotWithData', folderName=Folder,

```

```

734         dataPoints=True)
735
736 # In[ILM thickness vs age regression]
737
738 # Linear regression
739 f, ax = plt.subplots()
740 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
741                             "axes.labelsize":12})
742 # dict(Equator="r", Posterior="b") , 'color':'black', 'color':'blue'
743 ax = sns.lmplot(x=A, y=ILM, hue=R, markers=["o", "x"], data=df,
744                 legend_out=False, fit_reg=True, height=5, aspect=1.6,
745                 palette="Set1", truncate=False, ci=95, line_kws={'lw':0})
746 ax.set(ylabel=ILM, xlabel=A_yrs)
747
748 # Remove all NaN's from the data for regressions
749
750 # remove nans from ILM thickness
751 df_no_Nan = df.dropna(subset=[ILM])
752
753 # linear regressions for fitting
754 x = df_no_Nan[A][df_no_Nan[R] == Eq]
755 y = df_no_Nan[ILM][df_no_Nan[R] == Eq]
756
757 x_plot = np.linspace(min(x), max(x), 100)
758
759 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
760 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
761 plt.text(80, yfit(80) + 20, r'$r={:.4f}$'.format(r_value1), color='r',
762          horizontalalignment='left', fontsize=8, weight='semibold') # r value
763
764 # linear regressions for fitting
765 x = df_no_Nan[A][df_no_Nan[R] == Po]
766 y = df_no_Nan[ILM][df_no_Nan[R] == Po]
767
768 x_plot = np.linspace(min(x), max(x), 100)
769 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
770 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
771 plt.text(75, yfit(75) + 20, r'$r={:.4f}$'.format(r_value2), color='b',
772          horizontalalignment='left', fontsize=8, weight='semibold') # r value
773
774 # Axis limits
775 ax.set(ylim=(0, None))
776 ax.set(xlim=(None, None))
777
778 # New path
779 NP = os.path.join(SF, 'ILM_vs_Age')
780
781 # Create folder if it doesn't exist
782 os.makedirs(NP, exist_ok=True)
783
784 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
785 plt.close()
786
787 # In[Max peel force vs ILM thickness]
788
789 # Linear regression
790 f, ax = plt.subplots()
791 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,

```

```

792         "axes.labelsize":12})
793 ax = sns.lmplot(x=ILM, y=mpf_mN, hue=R, markers=["o", "x"], data=df,
794               legend_out=False, fit_reg=True, height=5, aspect=1.6,
795               palette="Set1", truncate=True, ci=95, line_kws={'lw':0})
796 ax.set(xlabel=ILM, ylabel=MPF)
797
798 # Remove all NaN's from the data for regressions
799 # remove nans from ILM thickness & Max
800 df_no_Nan = df.dropna(subset=[ILM, mpf_mN])
801
802 # linear regressions for fitting
803 x = df_no_Nan[ILM][df_no_Nan[R] == Eq]
804 # Convert to N
805 y = df_no_Nan[mpf_mN][df_no_Nan[R] == Eq]
806
807 x_plot = np.linspace(min(x), max(x), 100)
808
809 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
810 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
811 plt.text(500, yfit(500) + 4, r'$r={:.4f}$'.format(r_value1), color='r',
812         horizontalalignment='left', fontsize=8, weight='semibold') # r value
813
814 # linear regressions for fitting
815 x = df_no_Nan[ILM][df_no_Nan[R] == Po]
816 y = df_no_Nan[mpf_mN][df_no_Nan[R] == Po]
817
818 x_plot = np.linspace(min(x), max(x), 100)
819 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
820 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
821 plt.text(1500, yfit(1500) + 1, r'$r={:.4f}$'.format(r_value2), color='b',
822         horizontalalignment='left', fontsize=8, weight='semibold') # r value
823
824 # Axis limits
825 ax.set(ylim=(0, 18))
826 ax.set(xlim=(0, max(x)*1.1))
827
828 # New path
829 NP = os.path.join(SF, 'ILM_vs_MaxPeel')
830
831 # Create folder if it doesn't exist
832 os.makedirs(NP, exist_ok=True)
833
834 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
835 plt.close()
836
837 # In[Max peel force vs ILM thickness by age group]
838
839 # Linear regression
840 f, ax = plt.subplots()
841 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
842                           "axes.labelsize":12})
843 ax = sns.lmplot(x=ILM, y=mpf_mN, hue=A60, markers=["o", "x"], data=df,
844               legend_out=False, fit_reg=True, height=5, aspect=1.6,
845               palette="Set1", truncate=True, ci=95, line_kws={'lw':0})
846 ax.set(xlabel=ILM, ylabel=MPF)
847
848 # Remove all NaN's from the data for regressions
849 # remove nans from ILM thickness & Max

```

```

850 df_no_Nan = df.dropna(subset=[ILM, mpf_mN])
851
852 # linear regressions for fitting
853 x = df_no_Nan[ILM][df_no_Nan[A60] == Aleq60]
854 y = df_no_Nan[mpf_mN][df_no_Nan[A60] == Aleq60] # HmN
855
856 x_plot = np.linspace(min(x), max(x), 100)
857
858 # linear regression
859 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
860
861 # Linear regression line
862 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1)
863 plt.text(1250, yfit(1250) + 0.75, r'$r={:.4f}$'.format(r_value1), color='r',
864         horizontalalignment='left', fontsize=8, weight='semibold') # r value
865
866 # linear regressions for fitting
867 x = df_no_Nan[ILM][df_no_Nan[A60] == Ag60]
868 y = df_no_Nan[mpf_mN][df_no_Nan[A60] == Ag60] # HmN
869
870 x_plot = np.linspace(min(x), max(x), 100)
871 # linear regression
872 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
873
874 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1) # linear regression
875 plt.text(1000, yfit(1000) + 1, r'$r={:.4f}$'.format(r_value2), color='b',
876         horizontalalignment='left', fontsize=8, weight='semibold') # r value
877
878 # Legend
879 plt.legend(loc='best').set_title(A_G) # legend
880
881 # axis limits
882 ax.set(ylim=(0, 18))
883 ax.set(xlim=(0, 2200))
884
885 # New path
886 NP = os.path.join(SF, 'ILM_vs_MaxPeel_Age60')
887
888 # Create folder if it doesn't exist
889 os.makedirs(NP, exist_ok=True)
890
891 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
892 plt.close()
893
894
895 # In[Max peel force vs ILM thickness in the Equator]
896
897 # Linear regression
898 f, ax = plt.subplots()
899 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
900                             "axes.labelsize":12})
901 ax = sns.lmplot(x=ILM, y=mpf_mN, hue=A60, markers=["o", "x"],
902                data=df[df[R] == Eq], legend_out=False, fit_reg=True, height=5,
903                aspect=1.6, palette="Set1", truncate=False, ci=95,
904                line_kws={'lw':0})
905 ax.set(xlabel=ILM, ylabel=MPF)
906
907 # Remove all NaN's from the data for regressions

```

```

908 # remove nans from ILM thickness & Max
909 df_no_Nan = df.dropna(subset=[ILM, mpf_mN])
910
911 # linear regressions for fitting
912 x = df_no_Nan[ILM][(df_no_Nan[A60] == Aleq60) & (df[R] == Eq)]
913 y = df_no_Nan[mpf_mN][(df_no_Nan[A60] == Aleq60) & (df[R] == Eq)] # MmN
914
915 x_plot = np.linspace(min(x), max(x), 100)
916
917 # linear regression
918 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
919
920 # Linear regression line
921 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1)
922 plt.text(500, yfit(500) + 1, r'$r={:.4f}$'.format(r_value1), color='r',
923         horizontalalignment='left', fontsize=8, weight='semibold') # r value
924
925 # linear regressions for fitting
926 x = df_no_Nan[ILM][(df_no_Nan[A60] == Ag60) & (df[R] == Eq)]
927 y = df_no_Nan[mpf_mN][(df_no_Nan[A60] == Ag60) & (df[R] == Eq)] # MmN
928
929 x_plot = np.linspace(min(x), max(x), 100)
930 # linear regression
931 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
932
933 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1) # linear regression
934 plt.text(500, yfit(500) - 1, r'$r={:.4f}$'.format(r_value2), color='b',
935         horizontalalignment='left', fontsize=8, weight='semibold') # r value
936
937 # Legend
938 plt.legend(loc='best').set_title("Equator Age group (yr.)") # legend
939
940 # axis limits
941 ax.set(ylim=(0, 20))
942 # ax.set(xlim=(0, None))
943
944 # New path
945 NP = os.path.join(SF, 'ILM_vs_MaxPeel_Age60_Equator')
946
947 # Create folder if it doesn't exist
948 os.makedirs(NP, exist_ok=True)
949
950 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
951 plt.close()
952
953 # In[Steady state peel force vs ILM density]
954
955 # Linear regression
956 f, ax = plt.subplots()
957 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
958                             "axes.labelsize":12})
959 ax = sns.lmplot(x=TMD, y=SSmN, hue=R, markers=["o", "x"], data=df,
960               legend_out=False, fit_reg=True, height=5, aspect=1.6,
961               palette="Set1", truncate=True, ci=95, line_kws={'lw':0})
962 ax.set(xlabel=DensityUnit, ylabel='Steady state peel force (mN)')
963
964 # Remove all NaN's from the data for regressions
965 # remove nans from ILM thickness & Max

```

```

966 df_no_Nan = df.dropna(subset=[TMD, SSmN])
967 # figure out why zero's aren't being eliminated
968
969 # linear regressions for fitting
970 x = df_no_Nan[TMD][df_no_Nan[R] == Eq]
971 y = df_no_Nan[SSmN][df_no_Nan[R] == Eq]
972
973 x_plot = np.linspace(min(x), max(x), 100)
974
975 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
976 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
977 plt.text(85, yfit(85) + 0.2, r'$r={:.4f}$'.format(r_value1), color='r',
978         horizontalalignment='left', fontsize=8, weight='semibold') # r value
979
980 print('Values for correlation between Steady-state and Equator\n',
981       'P={:.4f}'.format(p_value), 'r={:.4f}'.format(r_value1))
982
983 # linear regressions for fitting
984 x = df_no_Nan[TMD][df_no_Nan[R] == Po]
985 y = df_no_Nan[SSmN][df_no_Nan[R] == Po]
986
987 x_plot = np.linspace(min(x), max(x), 100)
988 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
989 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
990 plt.text(70, yfit(70) + 0.3, r'$r={:.4f}$'.format(r_value2), color='b',
991         horizontalalignment='left', fontsize=8, weight='semibold') # r value
992
993 print('Values for correlation between Steady-state and Posterior\n',
994       'P={:.4f}'.format(p_value), 'r={:.4f}'.format(r_value2))
995
996 # axis limits
997 ax.set(ylim=(0, None))
998 ax.set(xlim=(0, max(df_no_Nan[TMD])*1.05))
999
1000 # New path
1001 NP = os.path.join(SF, 'ILM_vs_SteadyStatePeel_Region')
1002
1003 # Create folder if it doesn't exist
1004 os.makedirs(NP, exist_ok=True)
1005
1006 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
1007 plt.close()
1008
1009 # In[Maximum peel force vs ILM density]
1010
1011 # Linear regression
1012 f, ax = plt.subplots()
1013 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
1014                             "axes.labelsize":12})
1015 ax = sns.lmplot(x=TMD, y=mpf_mN, hue=R, markers=["o", "x"], data=df,
1016               legend_out=False, fit_reg=True, height=5, aspect=1.6,
1017               palette="Set1", truncate=False, ci=95, line_kws={'lw':0})
1018 ax.set(xlabel=DensityUnit, ylabel='Maximum peel force (mN)')
1019
1020 # Remove all NaN's from the data for regressions
1021 # remove nans from ILM thickness & Max
1022 df_no_Nan = df.dropna(subset=[TMD, mpf_mN])
1023 # figure out why zero's aren't being eliminated

```

```

1024
1025 # linear regressions for fitting
1026 x = df_no_Nan[TMD][df_no_Nan[R] == Eq]
1027 y = df_no_Nan[mpf_mN][df_no_Nan[R] == Eq]
1028
1029 x_plot = np.linspace(min(x), max(x), 100)
1030
1031 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
1032 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
1033 plt.text(85, yfit(85) + 0.1, r'$r={:.4f}$'.format(r_value1), color='r',
1034         horizontalalignment='left', fontsize=8, weight='semibold') # r value
1035
1036 # linear regressions for fitting
1037 x = df_no_Nan[TMD][df_no_Nan[R] == Po]
1038 y = df_no_Nan[mpf_mN][df_no_Nan[R] == Po]
1039
1040 x_plot = np.linspace(min(x), max(x), 100)
1041 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
1042 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
1043 plt.text(70, yfit(70) + 0.1, r'$r={:.4f}$'.format(r_value2), color='b',
1044         horizontalalignment='left', fontsize=8, weight='semibold') # r value
1045
1046 # axis limits
1047 ax.set(ylim=(0, None))
1048 # ax.set(xlim=(0, None))
1049
1050 # New path
1051 NP = os.path.join(SF, 'ILM_vs_MaxPeel_Region')
1052
1053 # Create folder if it doesn't exist
1054 os.makedirs(NP, exist_ok=True)
1055
1056 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
1057 plt.close()
1058
1059 # In[Collagen fibril density vs age correlation (regression)]
1060
1061 # Linear regression
1062 f, ax = plt.subplots()
1063 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
1064                             "axes.labelsize":12})
1065 # dict(Equator="r", Posterior="b") , 'color':'black', 'color':'blue'
1066 ax = sns.lmplot(x=A, y=TMD, hue=R, markers=["o", "x"], data=df,
1067               legend_out=False, fit_reg=True, height=5, aspect=1.6,
1068               palette="Set1", truncate=False, ci=95, line_kws={'lw':0})
1069 ax.set(ylabel=DensityUnit, xlabel=A_yrs)
1070
1071 # Remove all NaN's from the data for regressions
1072
1073 # remove nans from ILM thickness
1074 df_no_Nan = df.dropna(subset=[TMD])
1075
1076 # linear regressions for fitting
1077 x = df_no_Nan[A][df_no_Nan[R] == Eq]
1078 y = df_no_Nan[TMD][df_no_Nan[R] == Eq]
1079
1080 x_plot = np.linspace(min(x), max(x), 100)
1081

```



```

1082 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
1083 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
1084 plt.text(80, yfit(80) + 5, r'$r={:.4f}$'.format(r_value1), color='r',
1085          horizontalalignment='left', fontsize=8, weight='semibold') # r value
1086
1087 # linear regressions for fitting
1088 x = df_no_Nan[A][df_no_Nan[R] == Po]
1089 y = df_no_Nan[TMD][df_no_Nan[R] == Po]
1090
1091 x_plot = np.linspace(min(x), max(x), 100)
1092 slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
1093 plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
1094 plt.text(75, yfit(75) + 5, r'$r={:.4f}$'.format(r_value2), color='b',
1095          horizontalalignment='left', fontsize=8, weight='semibold') # r value
1096
1097 # Axis limits
1098 ax.set(ylim=(0, None))
1099 ax.set(xlim=(None, None))
1100
1101 # New path
1102 NP = os.path.join(SF, 'Density_vs_Age')
1103
1104 # Create folder if it doesn't exist
1105 os.makedirs(NP, exist_ok=True)
1106
1107 ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
1108 plt.close()
1109
1110
1111 # In[Collagen fibril Orientation vs age correlation (regression)]
1112
1113 # Linear regression
1114 f, ax = plt.subplots()
1115 sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
1116                             "axes.labelsize":12})
1117 # dict(Equator="r", Posterior="b") , 'color':'black', 'color':'blue'
1118 ax = sns.lmplot(x=A, y=TAA, hue=R, markers=["o", "x"], data=df,
1119                legend_out=False, fit_reg=True, height=5, aspect=1.6,
1120                palette="Set1", truncate=False, ci=95, line_kws={'lw':0})
1121 ax.set(ylabel=OrientationUnit, xlabel=A_yrs)
1122
1123 # Remove all NaN's from the data for regressions
1124
1125 # remove nans from ILM thickness
1126 df_no_Nan = df.dropna(subset=[TAA])
1127
1128 # linear regressions for fitting
1129 x = df_no_Nan[A][df_no_Nan[R] == Eq]
1130 y = df_no_Nan[TAA][df_no_Nan[R] == Eq]
1131
1132 x_plot = np.linspace(min(x), max(x), 100)
1133
1134 slope, intercept, r_value1, p_value, std_err = stats.linregress(x, y)
1135 plt.plot(x_plot, yfit(x_plot), '-', color='r', linewidth=1, label='line')
1136 plt.text(80, yfit(80) + 2, r'$r={:.4f}$'.format(r_value1), color='r',
1137          horizontalalignment='left', fontsize=8, weight='semibold') # r value
1138
1139 print('Collagen fibril Equator orientation\n',

```

```

1140         'P={:.4f}'.format(p_value), 'r={:.4f}'.format(r_value1))
1141
1142     # linear regressions for fitting
1143     x = df_no_Nan[A][df_no_Nan[R] == Po]
1144     y = df_no_Nan[TAA][df_no_Nan[R] == Po]
1145
1146     x_plot = np.linspace(min(x), max(x), 100)
1147     slope, intercept, r_value2, p_value, std_err = stats.linregress(x, y)
1148     plt.plot(x_plot, yfit(x_plot), '-', color='b', linewidth=1, label='line')
1149     plt.text(75, yfit(75) + 2, r'$r={:.4f}$'.format(r_value2), color='b',
1150             horizontalalignment='left', fontsize=8, weight='semibold') # r value
1151
1152     print('Collagen fibril Posterior orientation\n',
1153           'P={:.4f}'.format(p_value), 'r={:.4f}'.format(r_value2))
1154
1155     # Axis limits
1156     ax.set(ylim=(0, None))
1157     ax.set(xlim=(None, None))
1158
1159     # New path
1160     NP = os.path.join(SF, 'Angle_vs_Age')
1161
1162     # Create folder if it doesn't exist
1163     os.makedirs(NP, exist_ok=True)
1164
1165     ax.savefig(os.path.join(NP, 'Regression.pdf'), bbox_inches='tight')
1166     plt.close()
1167
1168     # In[Collagen fibril orientation distributions]
1169
1170     # remove nans from ILM thickness
1171     df_no_Nan = df.dropna(subset=[TAA])
1172
1173     # Normal distribution plots
1174     f, ax = plt.subplots(figsize=(9.6, 6))
1175     sns.set_context("paper", rc={"font.size":12, "axes.titlesize":8,
1176                                "axes.labelsize":12})
1177
1178     ax = sns.kdeplot(data=df_no_Nan, x=TAA, hue=R, hue_order=[Eq, Po], fill=True,
1179                     legend=False, palette='Paired', multiple='layer',
1180                     cut=0, bw_adjust=0.7, alpha=0.3)
1181
1182     ax.set(xlabel=OrientationUnit, ylabel='Kernel Density Estimation')
1183
1184     # Legend
1185     plt.legend(labels=[Eq, Po], loc='best').set_title(R)
1186
1187     # Axis limits
1188     # ax.set(ylim=(0, None))
1189     # ax.set(xlim=(0, 90))
1190
1191     # New path
1192     NP = os.path.join(SF, 'Angle')
1193
1194     # Create folder if it doesn't exist
1195     os.makedirs(NP, exist_ok=True)
1196
1197     plt.savefig(os.path.join(NP, 'Distribution.pdf'), bbox_inches='tight')

```