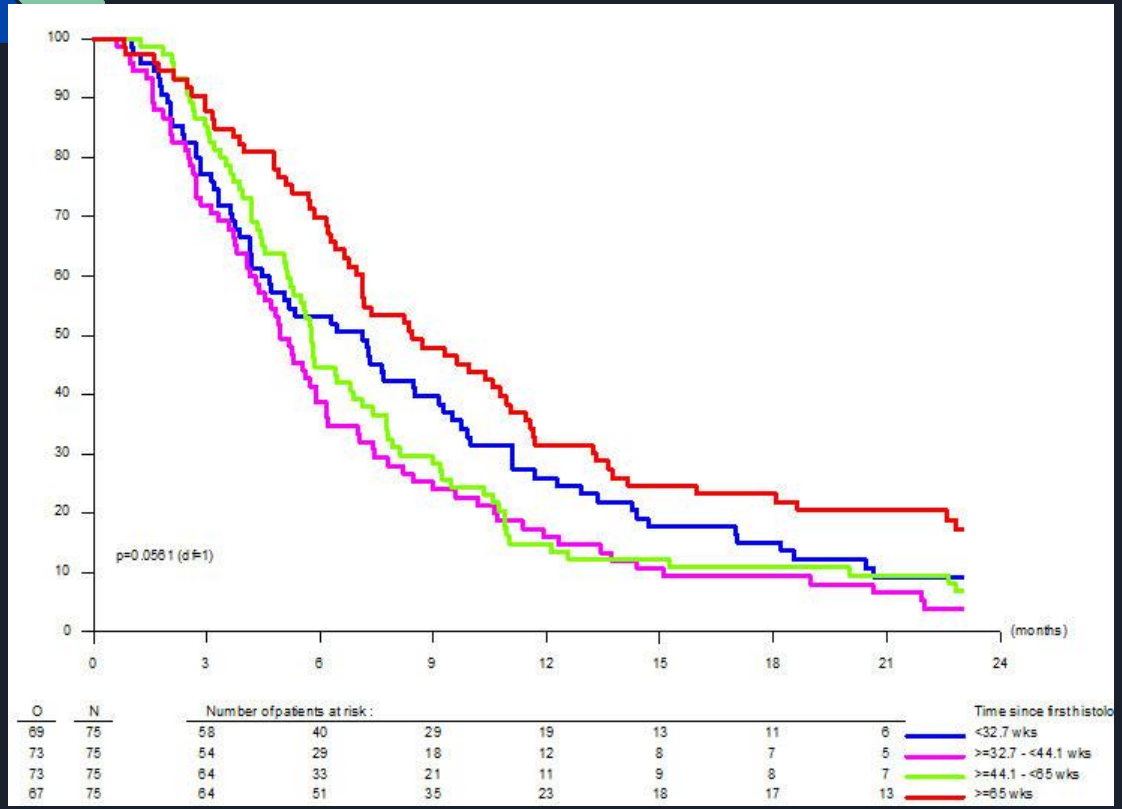
AstraZeneca AI Challenge

Team Russia
E Santhosh Kumar (SHA02921) (CS16B107)

Shaastra 2021

General Properties of Kaplan Meier Charts

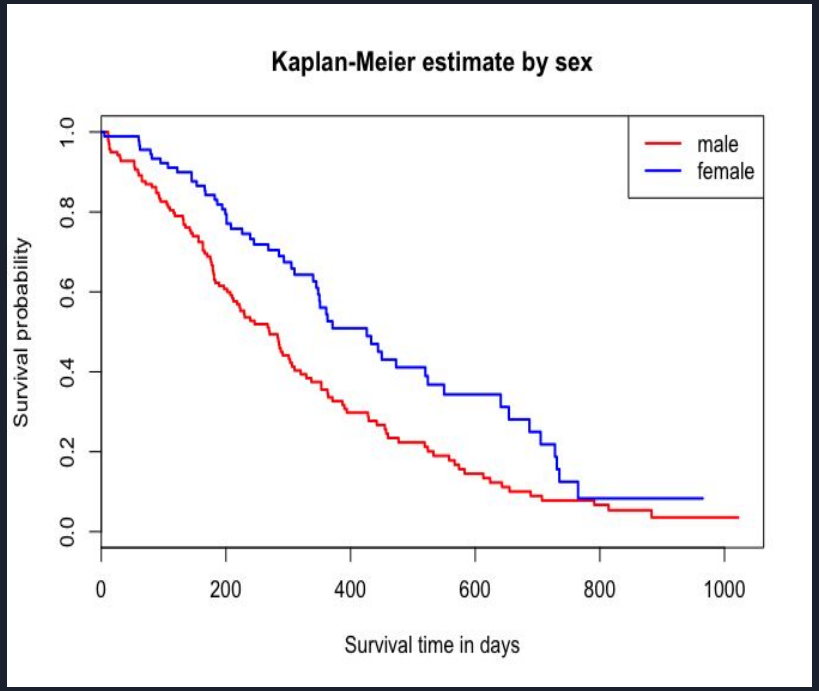


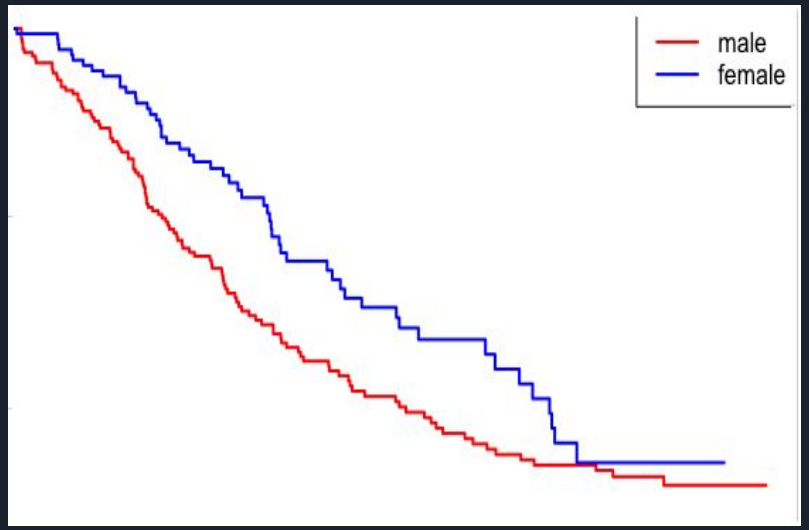
- Y axis is probability -> range [0, 1]
- Monotonically increasing or decreasing curves
- All curves in a chart begin at the same point

Key Processing Phases

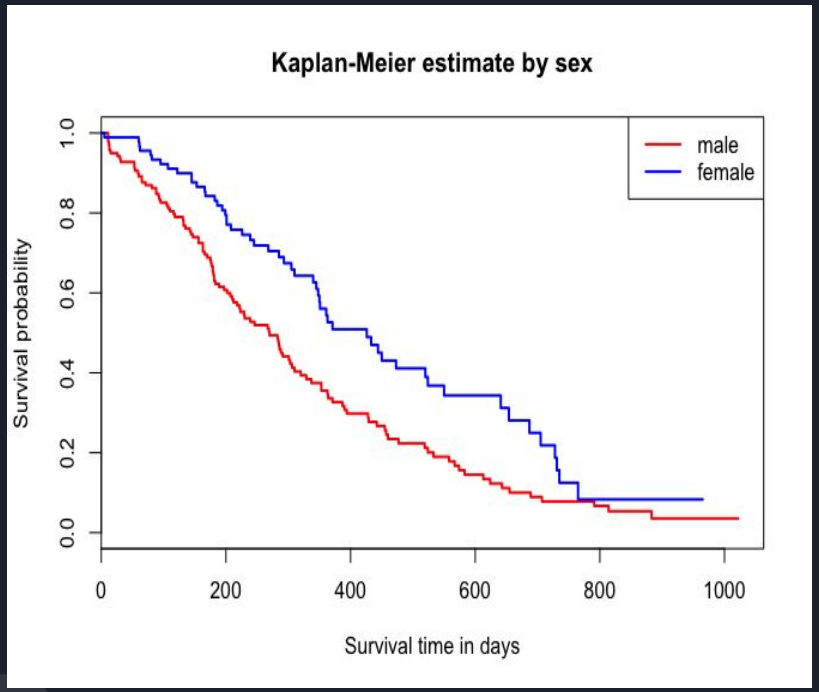
1. Identifying axes / ROI
2. Segmenting Plots
3. Interpolating Coordinates
4. Identifying range of X axis

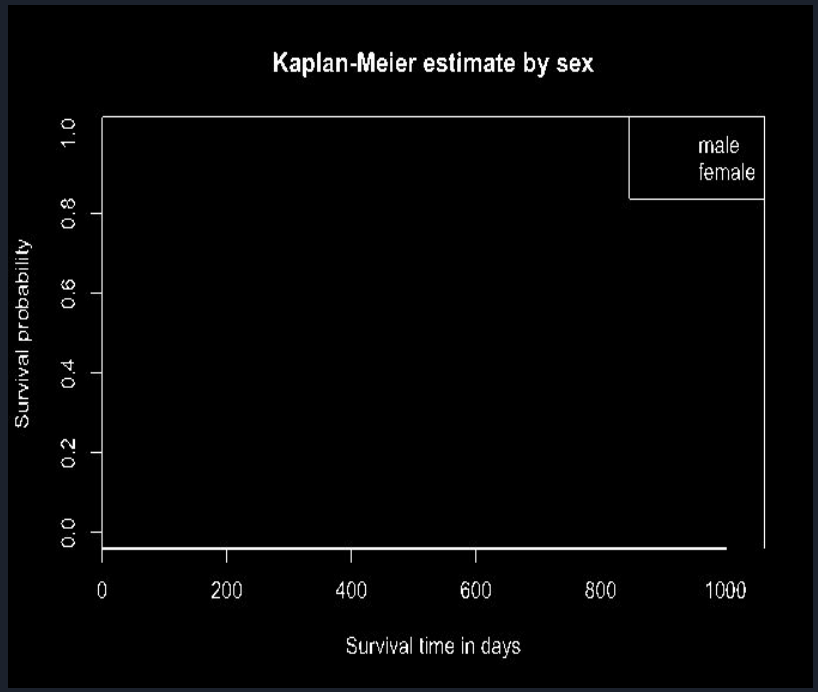
1. Identifying Axes / ROI





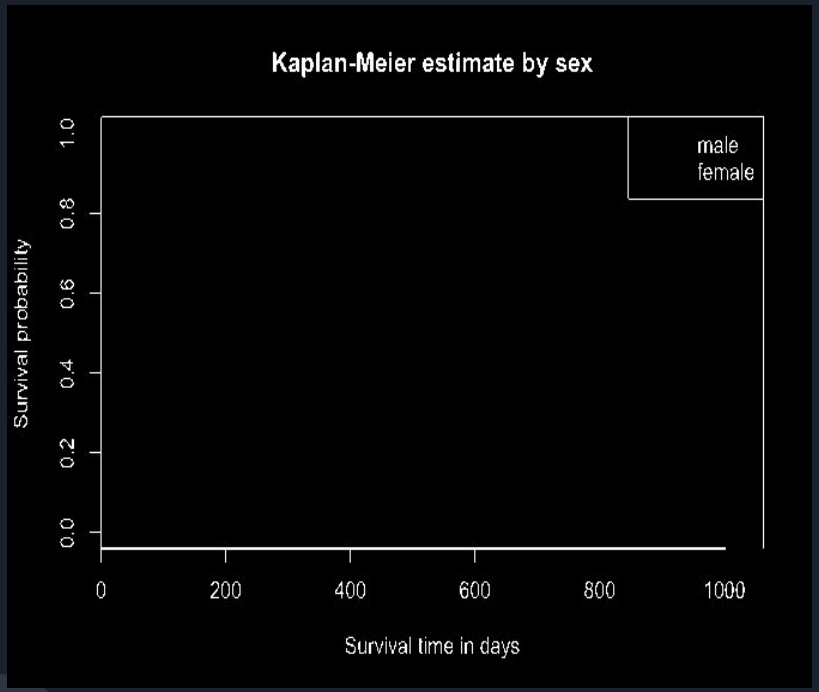
Step 1:
Mask only black areas by thresholding with HSV value

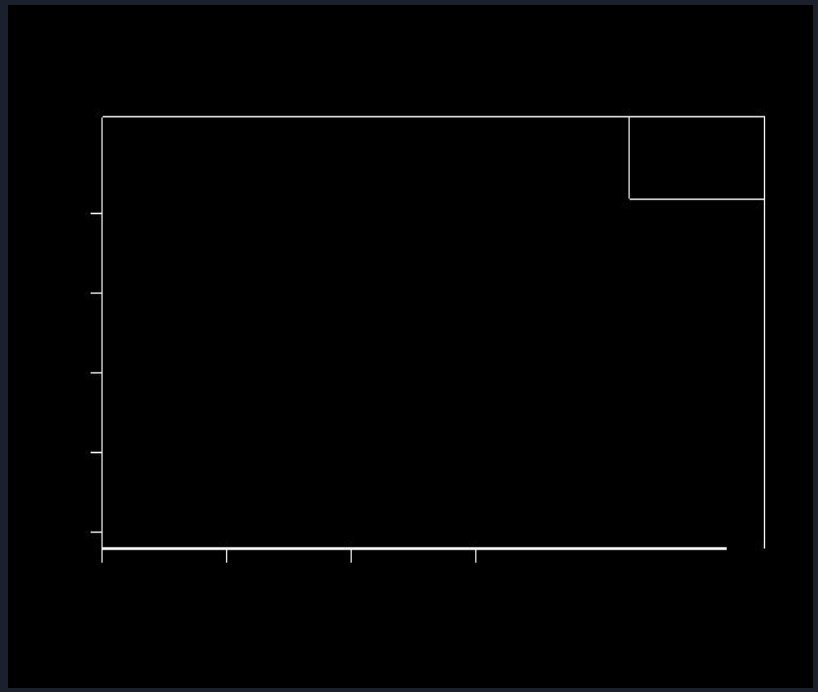




Assumption: axes are always black/dark

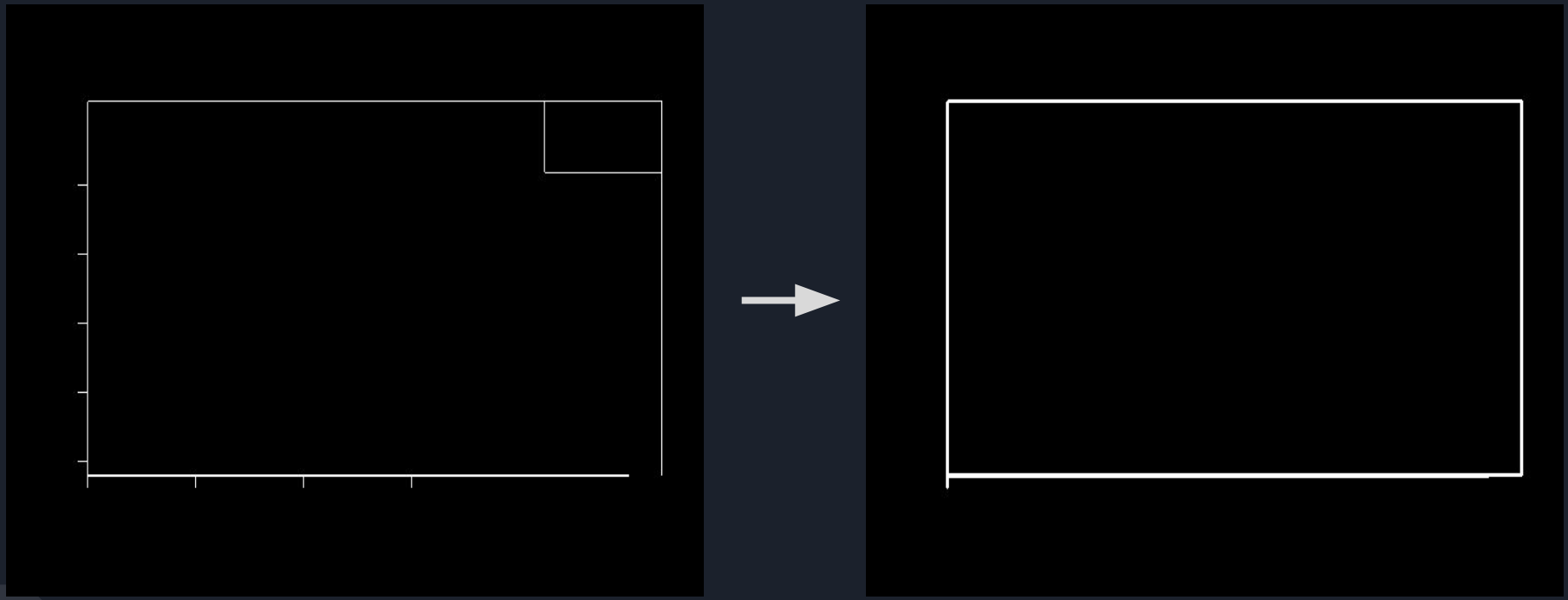
Step 2:
Remove connected components that are neither tall nor wide





Assumption: Y and X axes are among the tallest and widest components of the image respectively

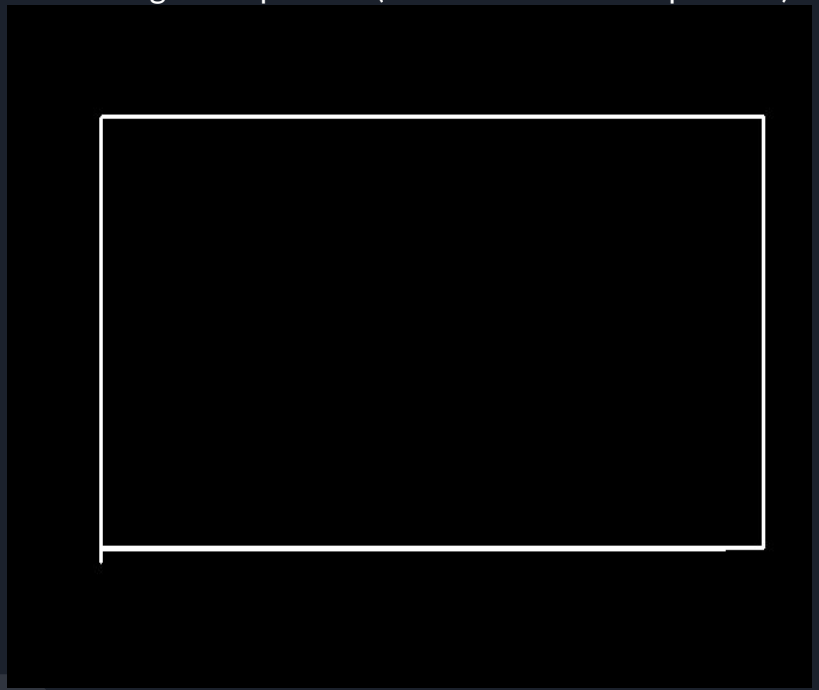
Step 3:
Identify line segments in mask using HoughLinesP



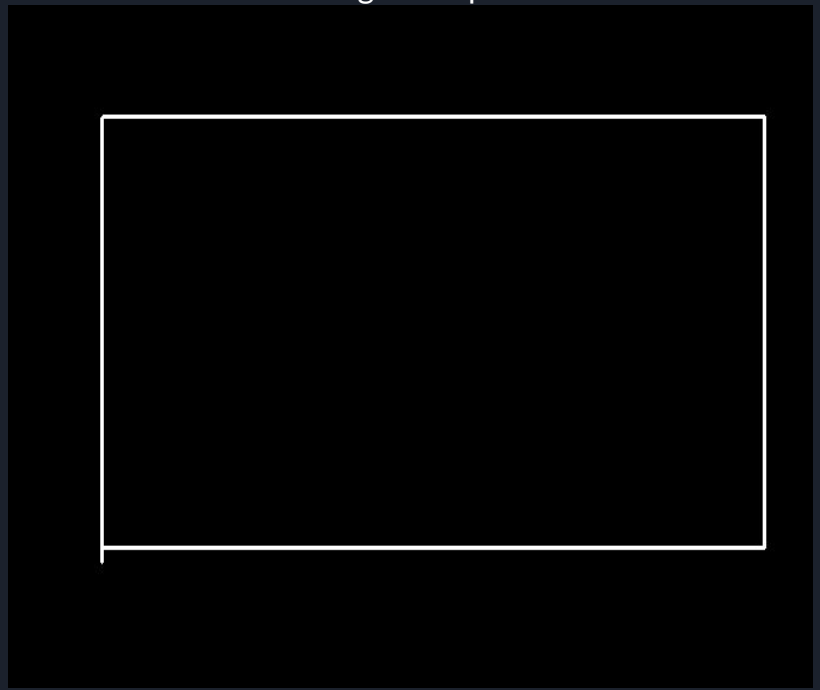
Assumption: Axes are ideally single straight lines. In practice, they are at least piece-wise straight

Step 4:
Merge line segments that are end-to-end or close-by duplicates

5 line segments present (x axis broken and duplicated)

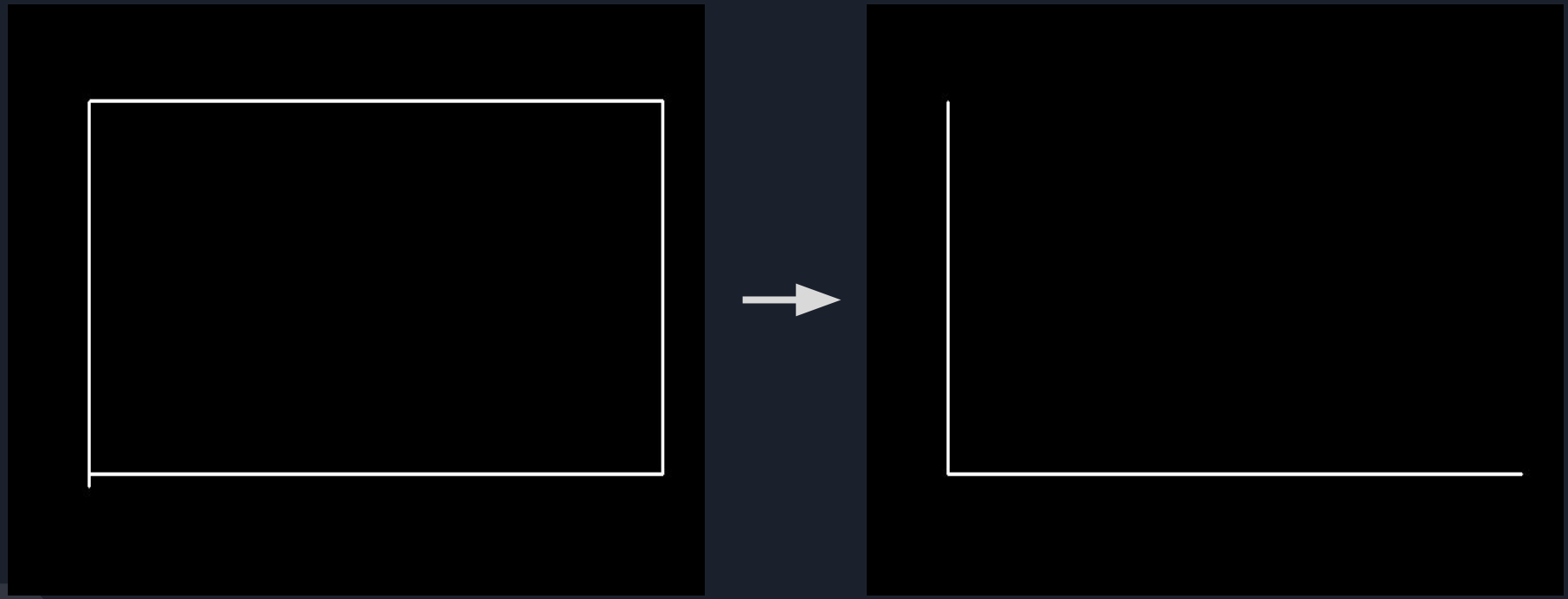


4 line segments present



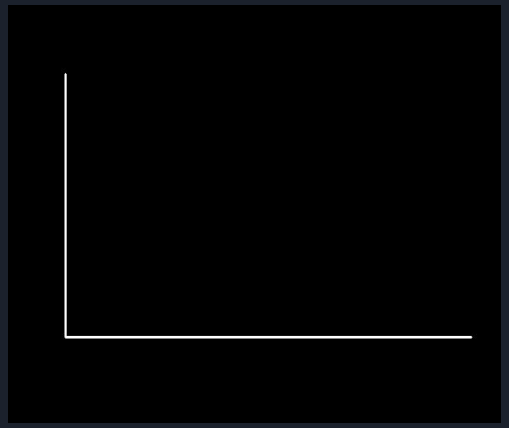
Reason: HoughLinesP may detect duplicate (because of thickness) or broken lines
Source: https://stackoverflow.com/questions/45531074/how-to-merge-lines-after-houghlinesp

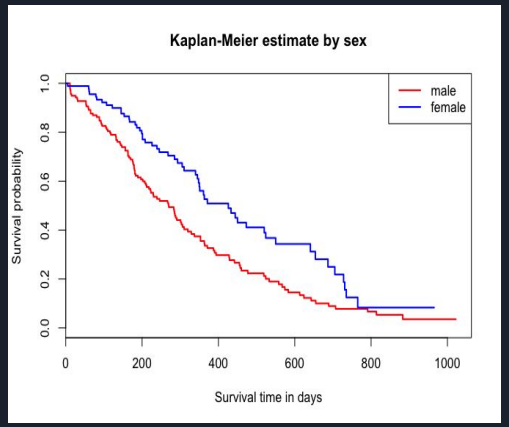
Step 5:
Identify X and Y axes from identified long lines

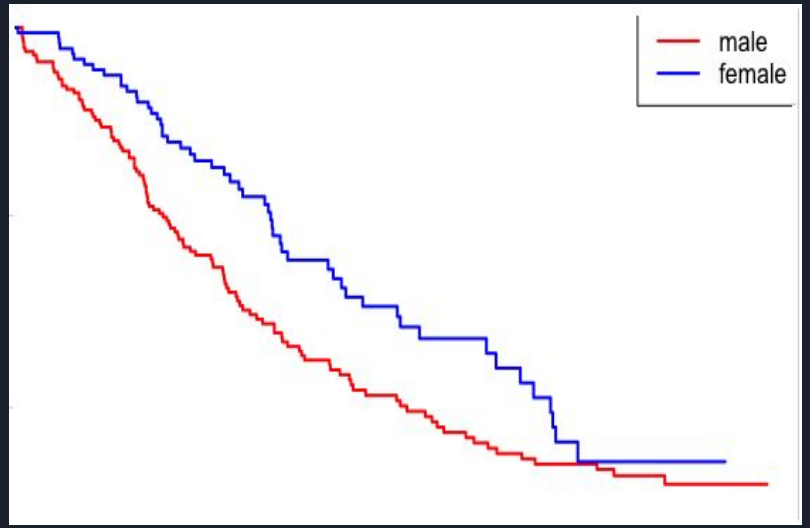


Assumptions: 1) axes can only be horizontal or vertical
2) Plots always have axes at least in the left and bottom
3) The actual chart has to be in the central area of the image

Step 6:
Crop image. Mask out black axes from the foreground of ROI



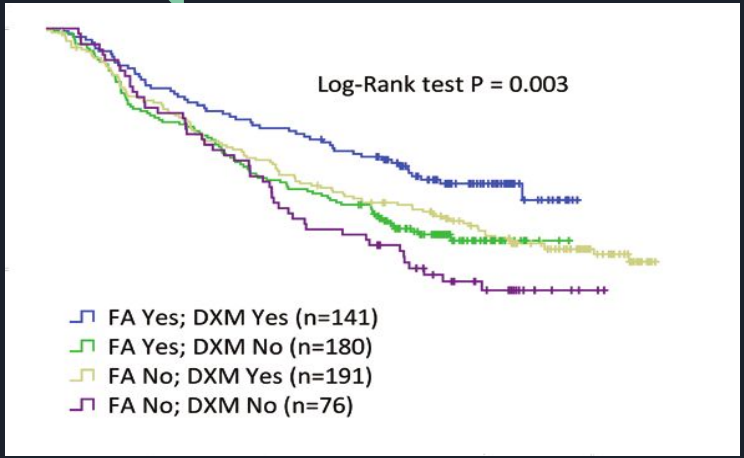




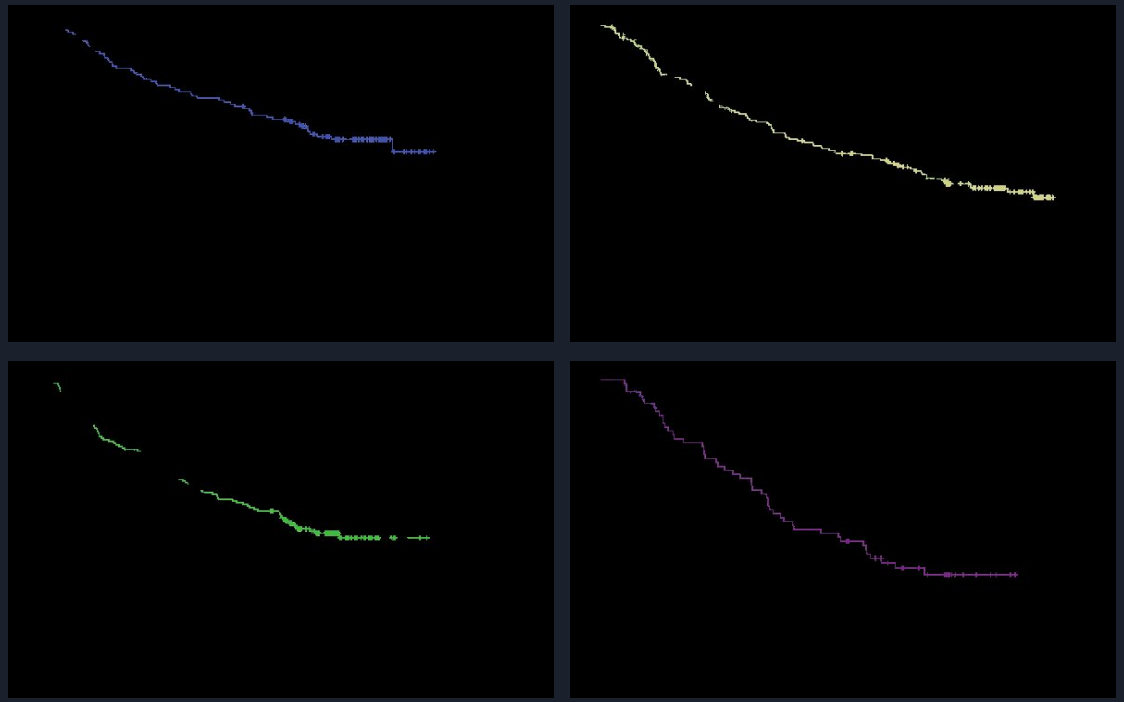
(Notice that the black axes have been thresholded out)

Reason: The axes themselves should not be misinterpreted as plots in the next steps
Assumption: Axes are dark (low value in HSV)

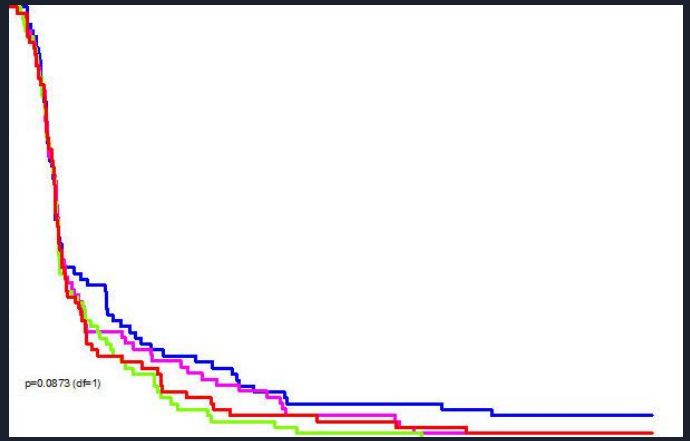
2. Segmenting Plots

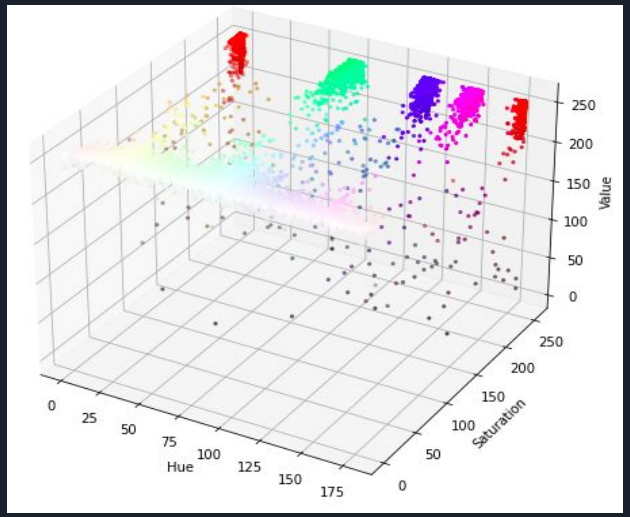


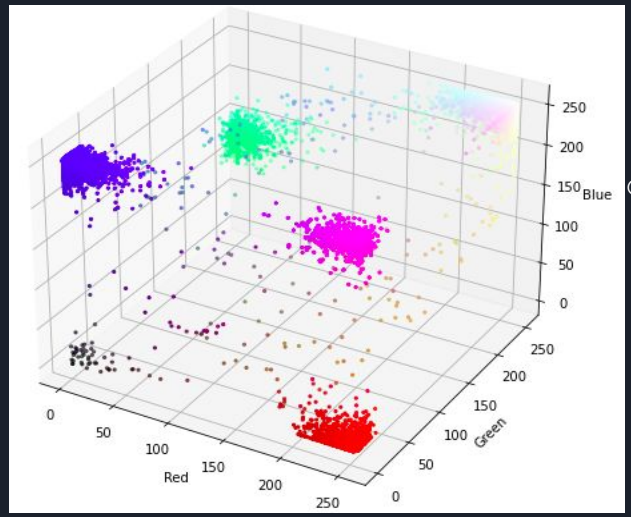
(Cropped ROI)



(Boolean masks for each label)



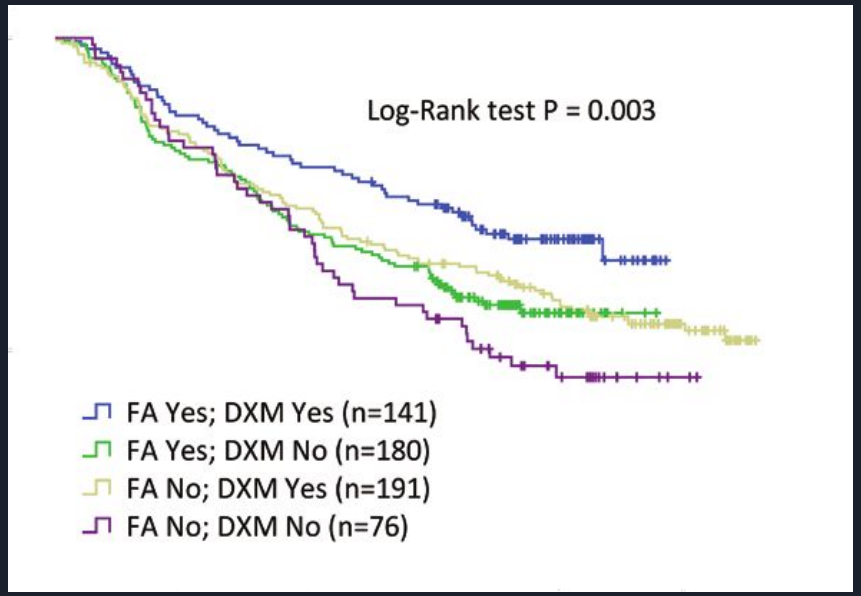


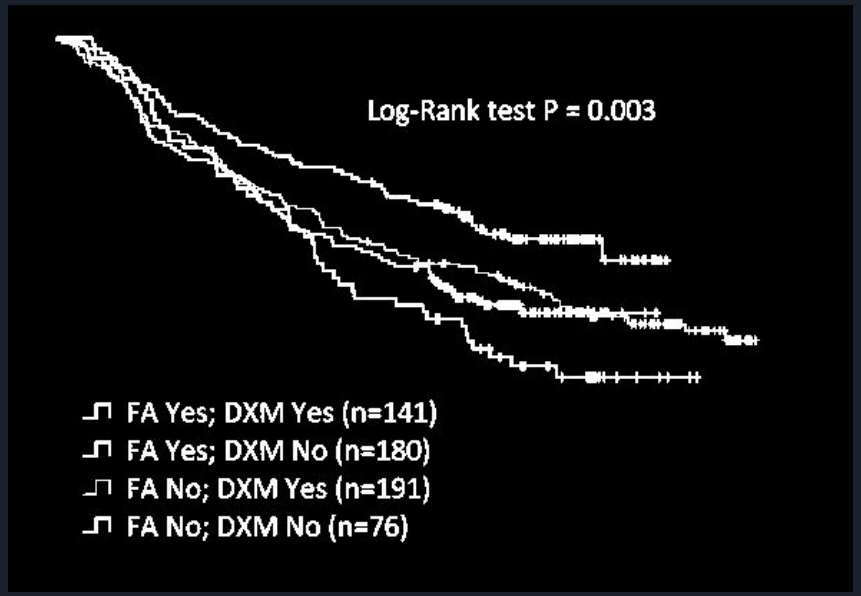


- Expectation:
  - Image contains few (k) unique pixel values apart from black and white, each corresponding to a label
- Reality:
  - Each label is associated with a cluster of noisy pixel values
  - Specks of colours of one label are present in unexpected locations like
    - legend of chart
    - other labels' plots
    - black areas like text, axes
  - Other unexpected colours occurring with non-negligible frequency

Step 1: Preprocessing for Clustering

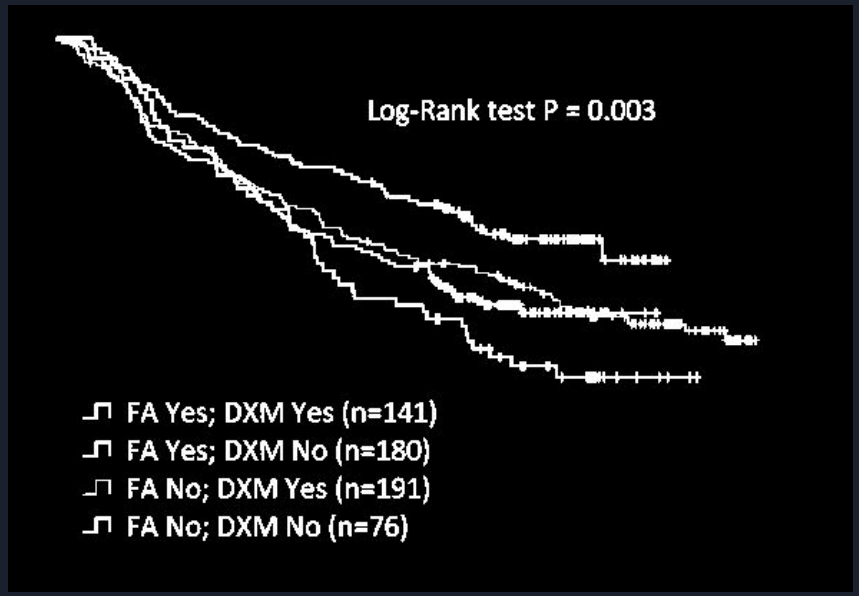
- Denoise using an edge-preserving bilateral filter
- Mask out white background by thresholding with HSV value

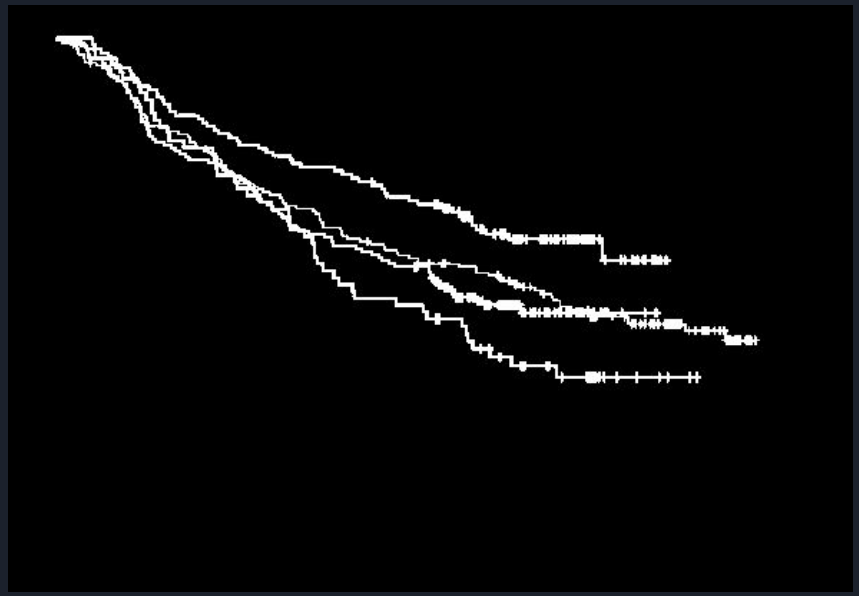




Assumption: Background is the one and only white area

Step 2: Preprocessing for Clustering
Retain only the connected component with largest area in foreground

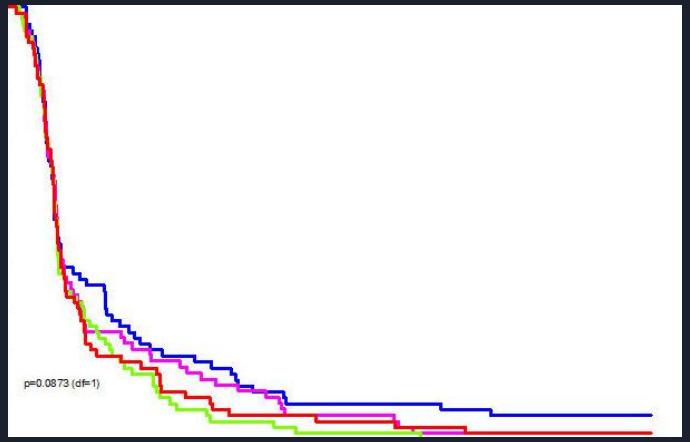


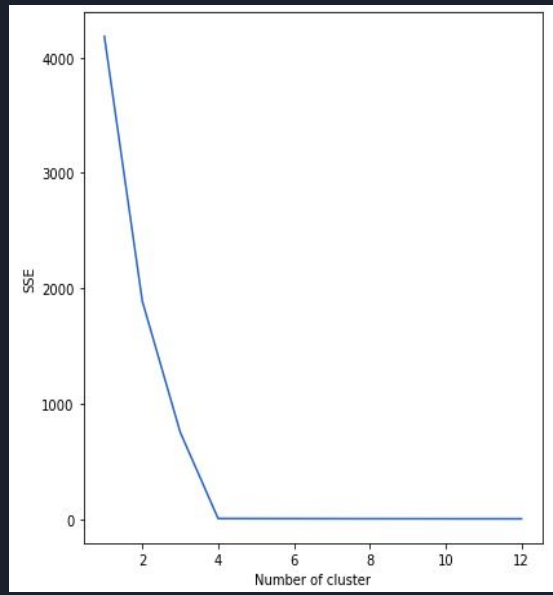


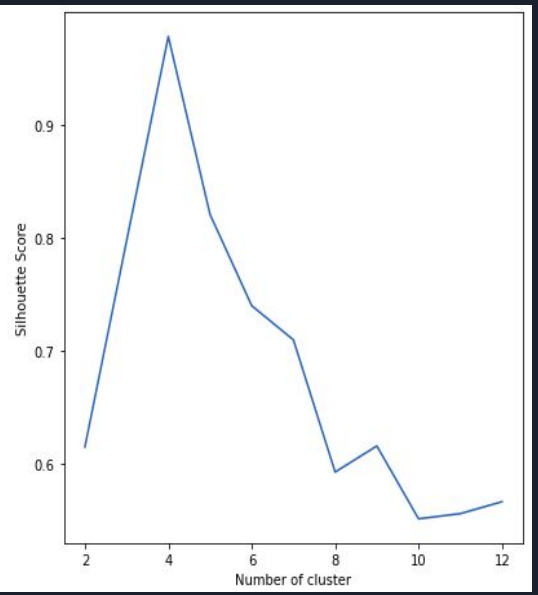
Reason: Expected to take away the legend, other text, random noise
Assumptions: 1) All curves in a chart begin at the same point
2) the union of all plots is the largest connected component in the non-white foreground

Step 3: K - means clustering on RGB Pixel values

- Perform clustering using a range of k values
- Choose the k with max Silhouette Coefficient of random sample





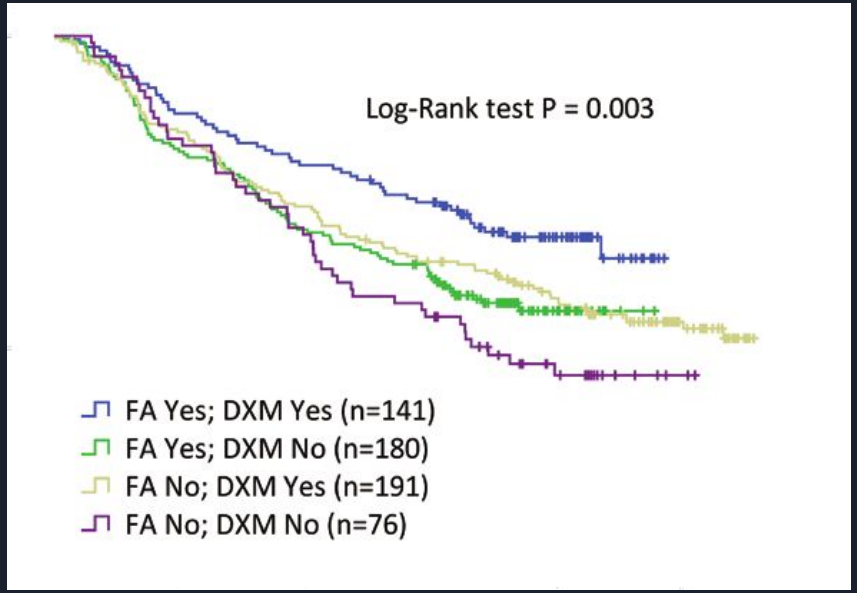


- We don't cluster in HSV space as visually close colours (red) can be farther apart in this space
- Silhouette Coefficient is only computed with a random sample as it is an expensive $O(n^2)$ algo
  - Note: We may still end up with extra clusters that need pruning

Step 4: Filter Clustering Predictions

Image may contain extra pixel clusters (apart from the ones corresponding to valid plots) because of reasons like

- plot lines having shadows / border effects
- Pixel value approximations while rendering the chart as image. E.g,
  - Varying shades of the plot's colour in the edges between plot and white background
  - Completely random colours in edges between plots of different colours



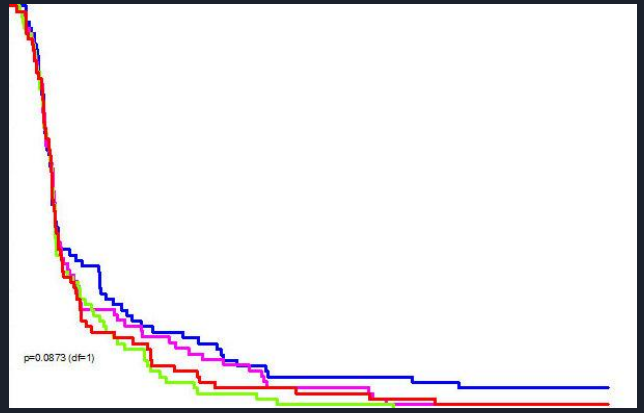
Steps to eliminate these extra clusters

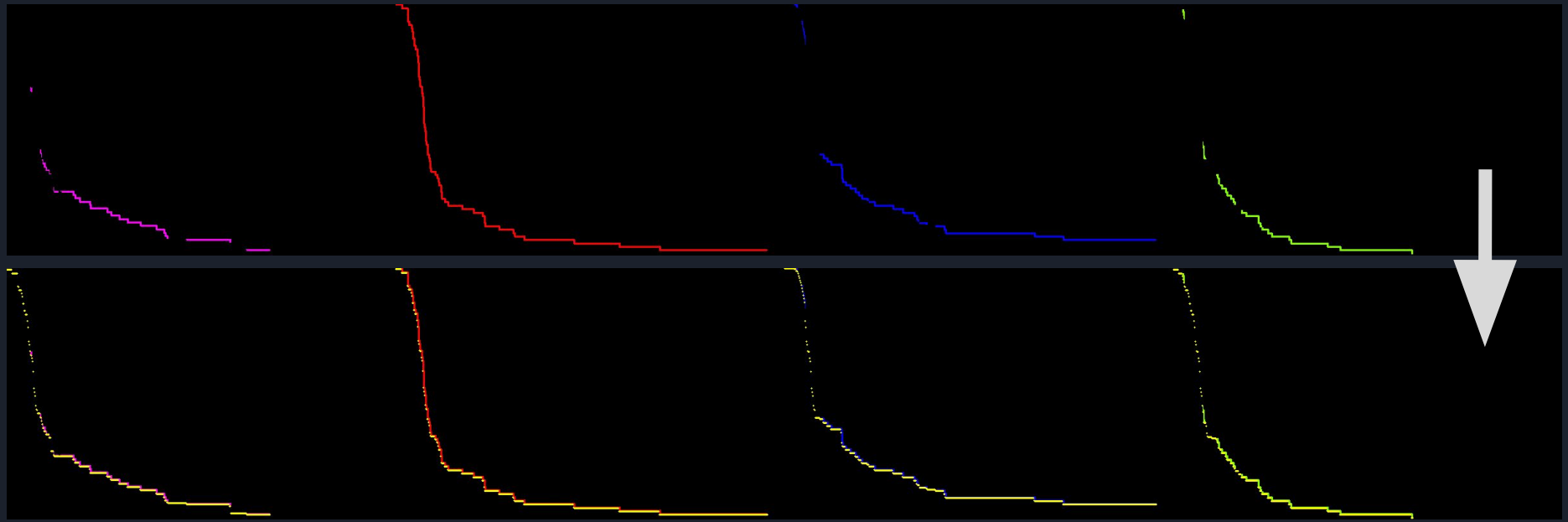
- In the mask of each cluster, remove components of size smaller than a minimum threshold (removes noisy blobs)
- Eliminate clusters with No. of pixels less than a minimum threshold. Threshold is a fraction of size of the largest cluster.

(Initial Clusters)

(Filtered Clusters)

3. Interpolating Coordinates

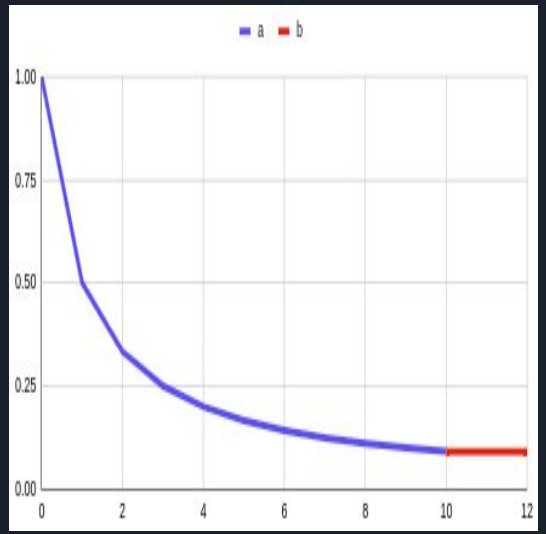




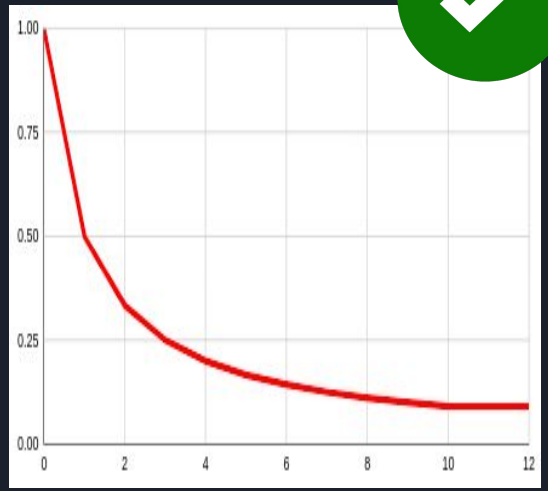
Why is simple linear interpolation in incomplete areas incorrect?

Assumptions:

- The various plots in the chart are superimposed (overlapped) in a fixed (unknown) order
- All plots begin from the same point



Correct interpolation of red plot based on overlapping blue plot



Incorrect linear interpolation of red plott

Algorithm

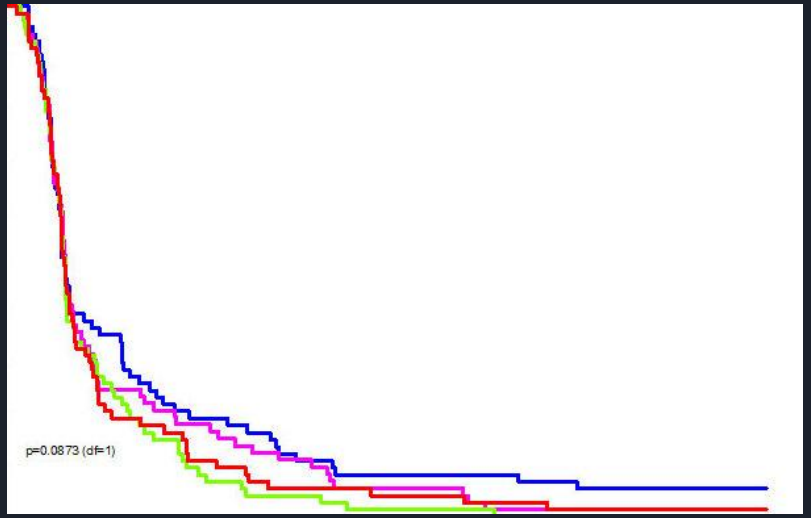
- Logic: Keep interpolating plots from top-most colour to bottom-most colour (in order of overlapping)
- Steps:
  - Identify top-most colour
    - It is the colour visible at the starting point (over all other colours)
    - Its plot is a single connected component
  - Initialize *current_overlap_mask* as the mask of top colour
  - While there are colours yet to be added,
    - Iterate over all remaining colours
    - The next colour to be added is the one who's incomplete sections can be connected through monotonically decreasing paths in the *current_overlap_mask*. (we use a path finding algorithm here)
    - Once the next colour is chosen, add its mask to *current_overlap_mask*

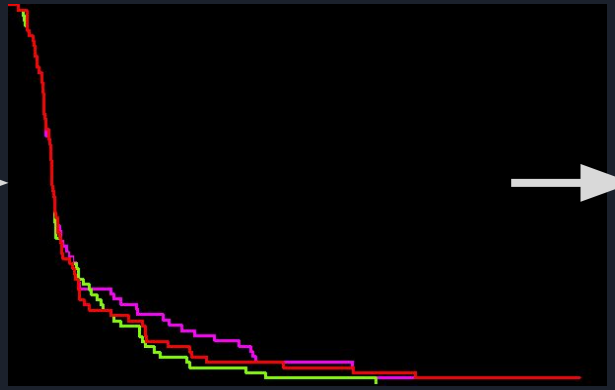
Note: 1) This algorithm is for monotonically decreasing plots. For monotonically increasing plots, we flip the input and pass to the algorithm.

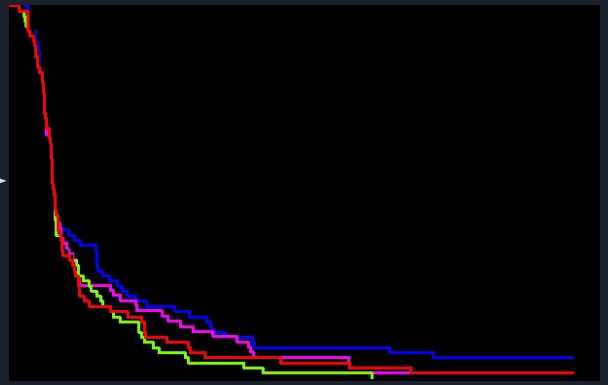
2) In practice, when none of the remaining colours are perfectly completed by the *current_overlap_mask*, we proceed with the colour with least incomplete manhattan distance

Algorithm - Example run

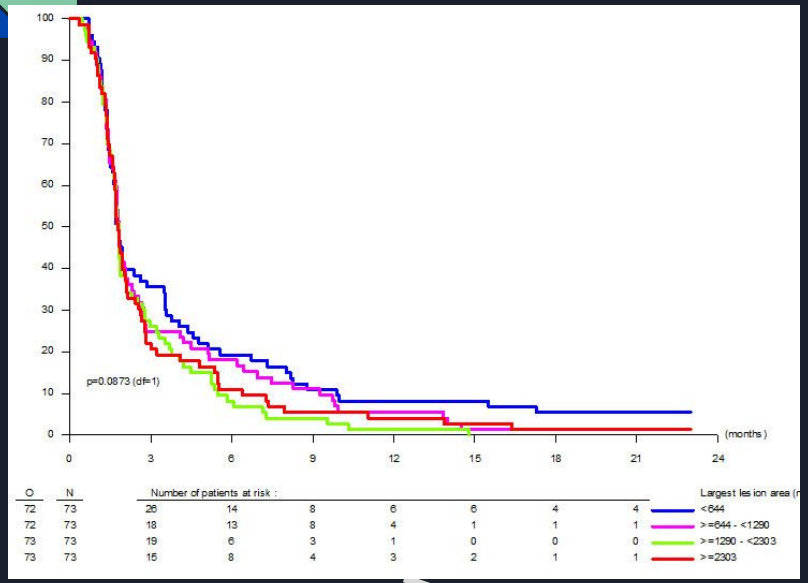
Note that the order of colour addition matches the overlapping order in the input image

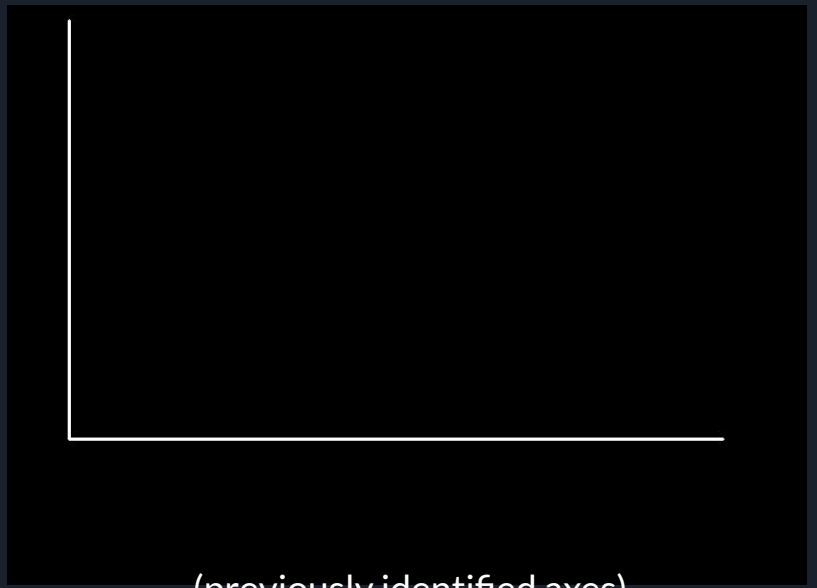






4. Identifying range of X axis



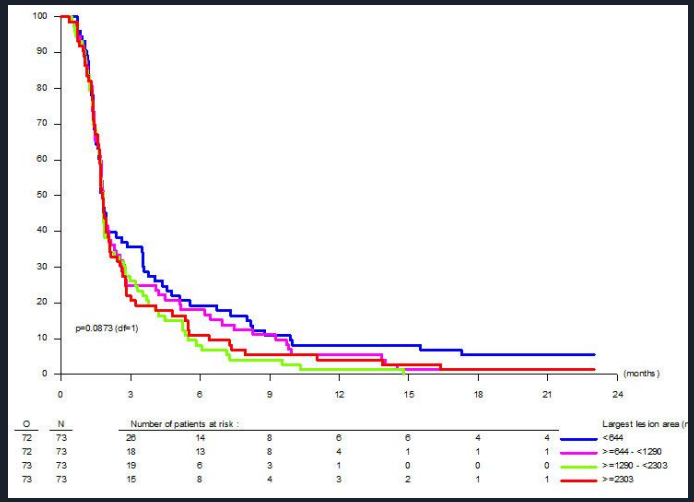


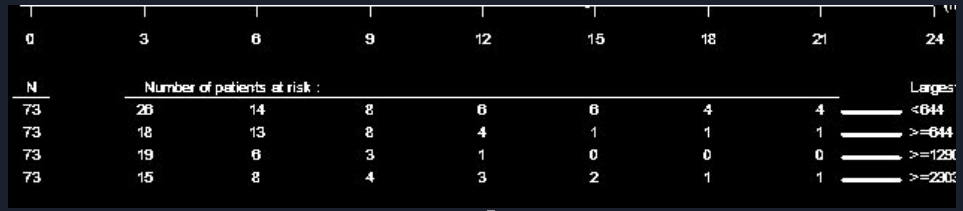
(previously identified axes)

range of x axis = 24

Step 1: Isolating ROI of xlabels

- Crop and threshold region of image below x axis
- Remove components wider than a threshold length (eliminates the x-axis along with x-ticks)
- Isolate the few consecutive rows on top that have white pixels. These are the rows with x-labels





Assumptions: x axis and labels are all dark

Step 2: using Optical Character Recognition (OCR) to get xrange

- Use Tesseract OCR Engine (v4.1.1) to extract text from the crop
  - Used in single line mode. Recognize only digits, floating point (.) and minus (-)
- Split text into numbers. Calculate the most common difference of consecutive numbers.
- If this difference occurs more than a relative threshold frequency (high confidence), multiply it with the number of numbers recognized to get xrange.

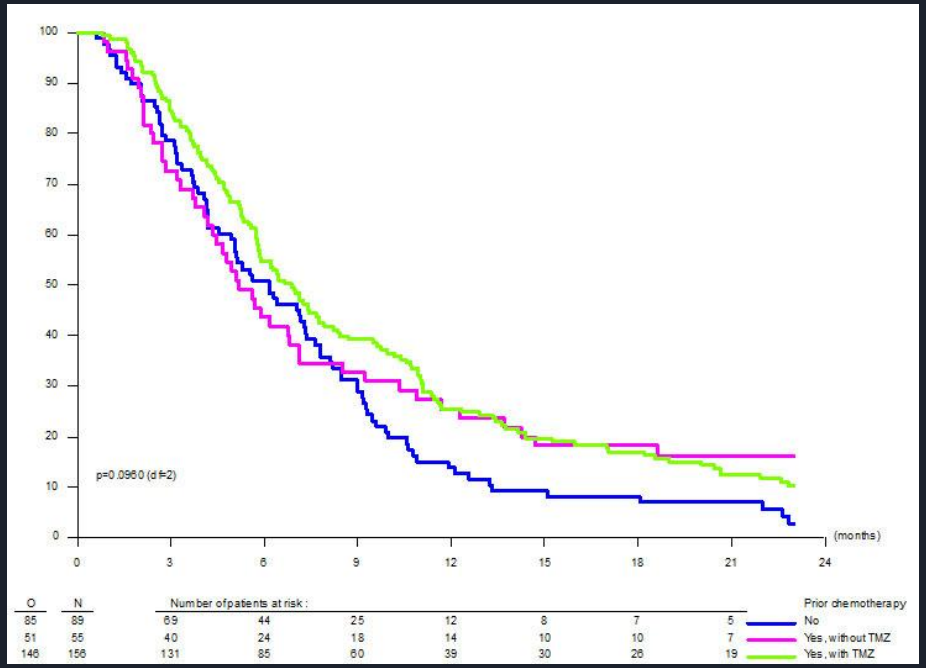
text = “0 3 8 9 12 15 18 21 24”

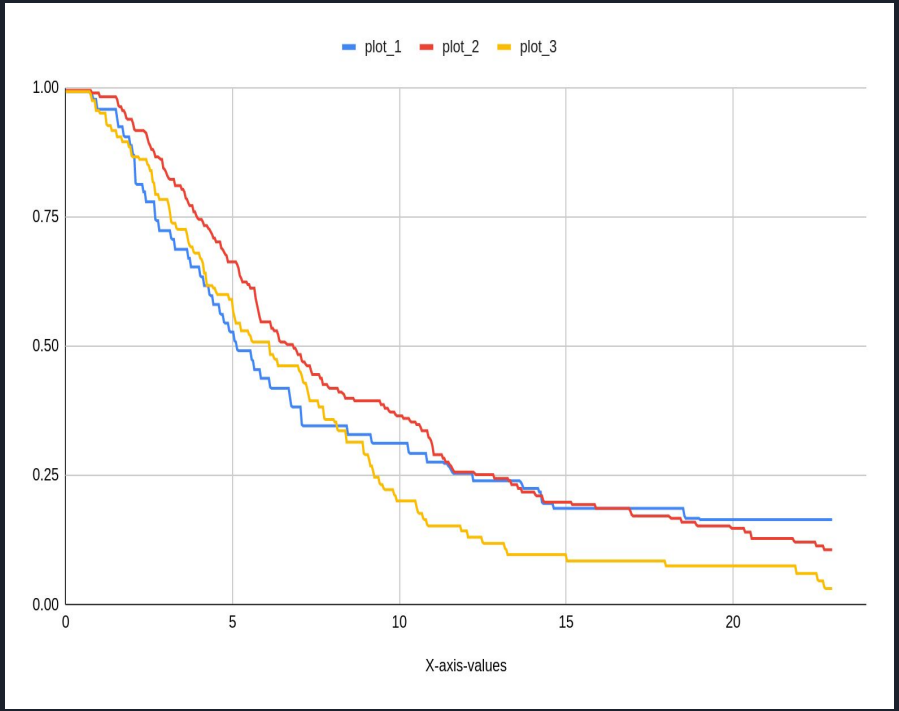
xrange = 24

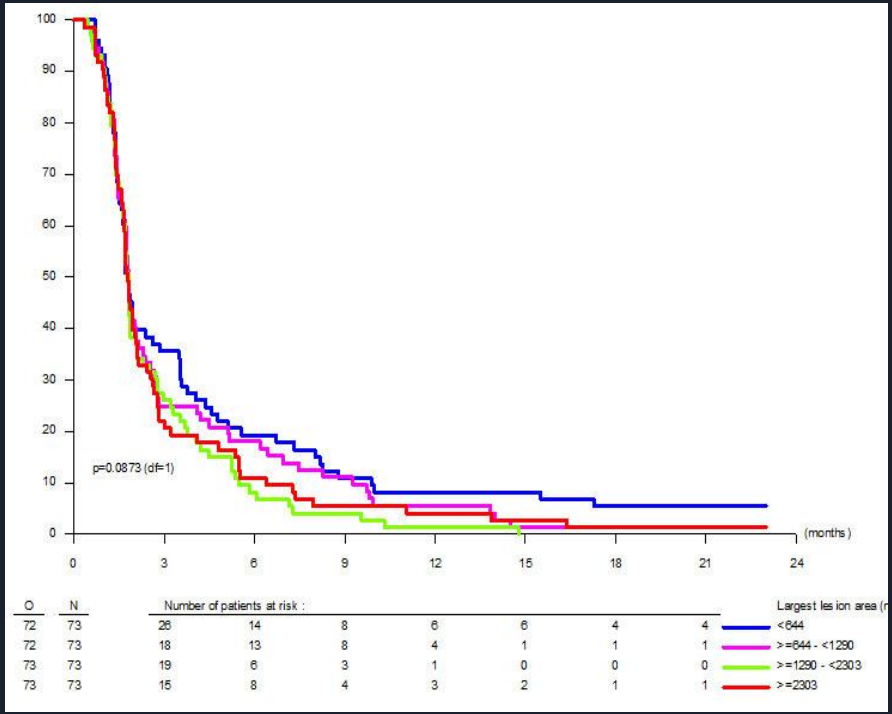
Note: If the above pipeline fails at any point, we use the default normalized xrange of 1.0
Assumptions: 1) The first xlabel is 0
2) xlabels follow an arithmetic progression

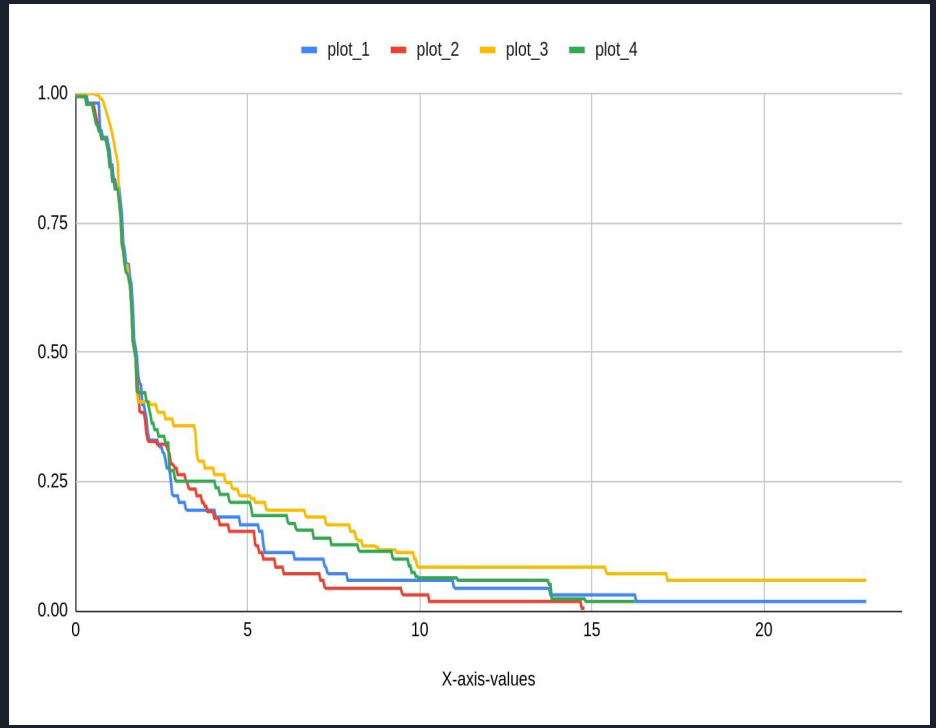
Results

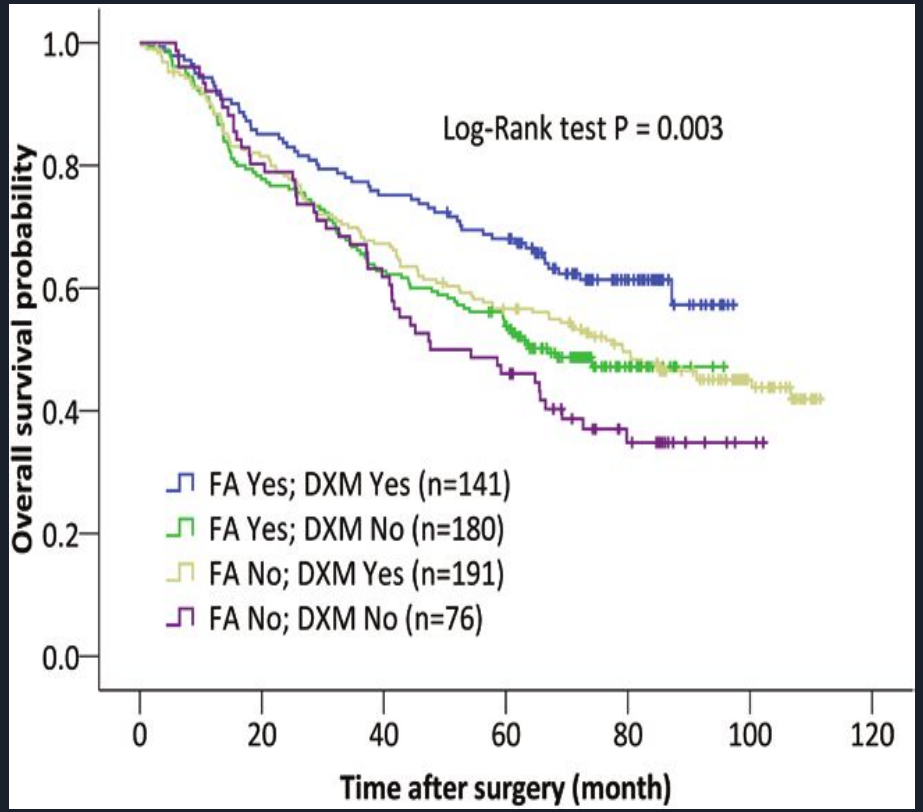
Comparing sample input charts with charts generated on Google Sheets using generated data

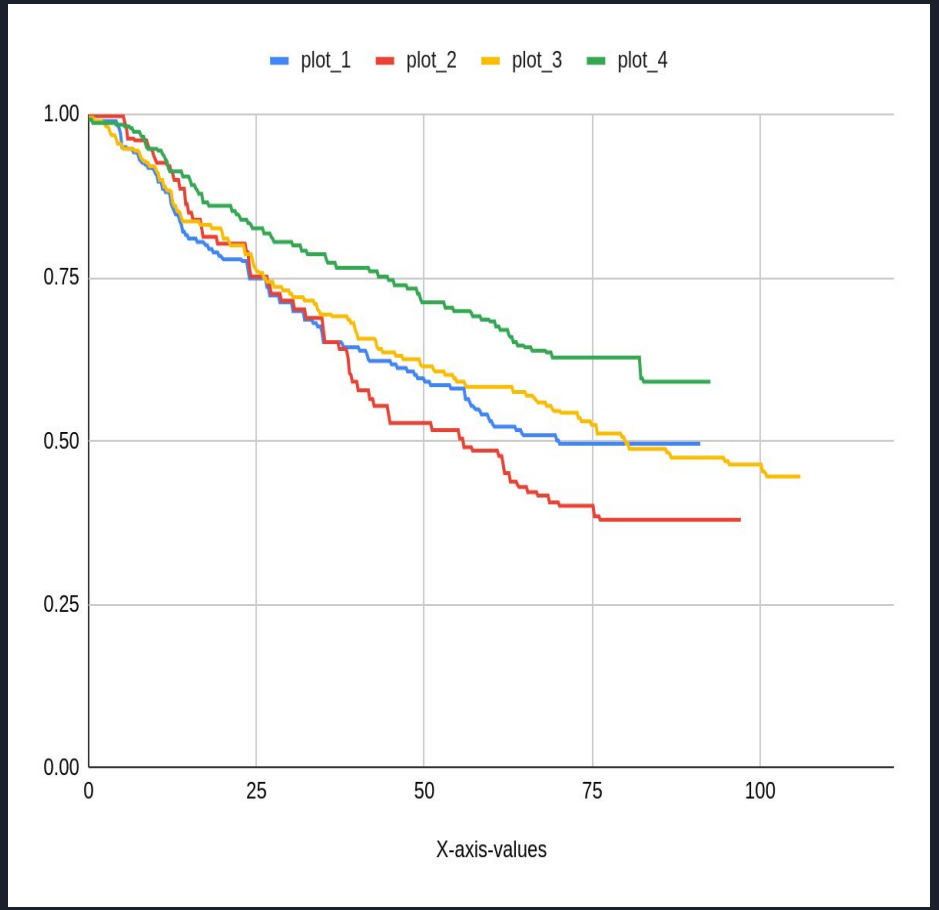

100
90
80
70
60
50
40
30
20
10
0
p=0.0960 (df=2)
0
3
6
9
12
15
18
21
24
(months)
O
N
Number of patients at risk :
Prior chemotherapy
85 89 69 44 25 12 8 7 5 No
51 55 40 24 18 14 10 10 7 Yes, without TMZ
146 156 131 85 60 39 30 26 19 Yes, with TMZ

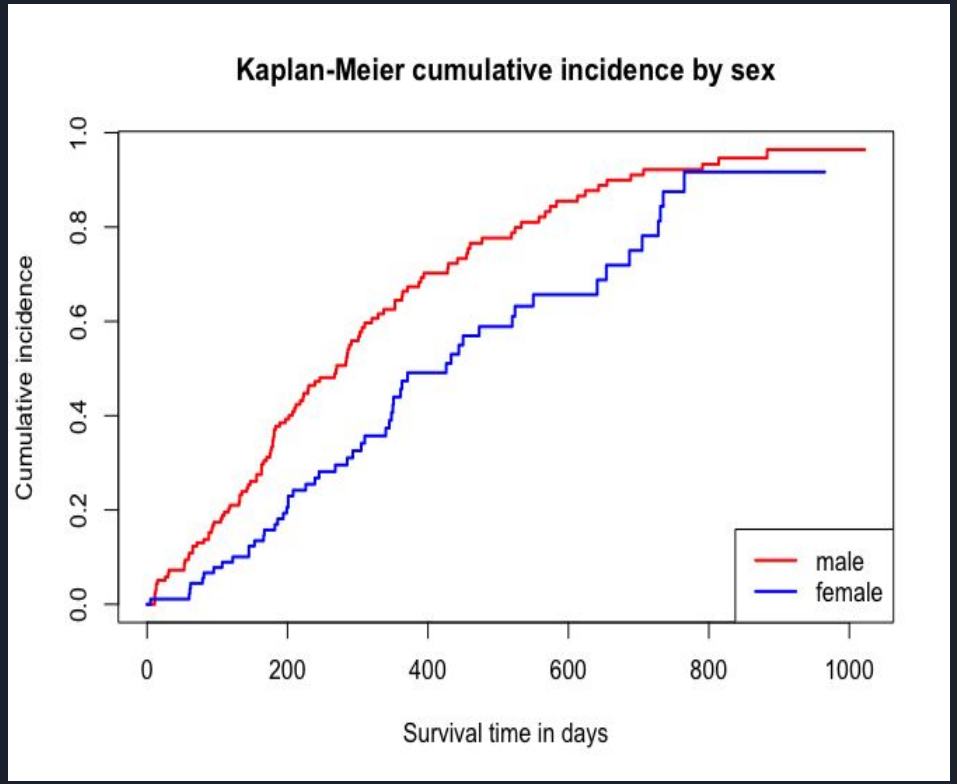

plot_1
plot_2
plot_3
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
X-axis-values

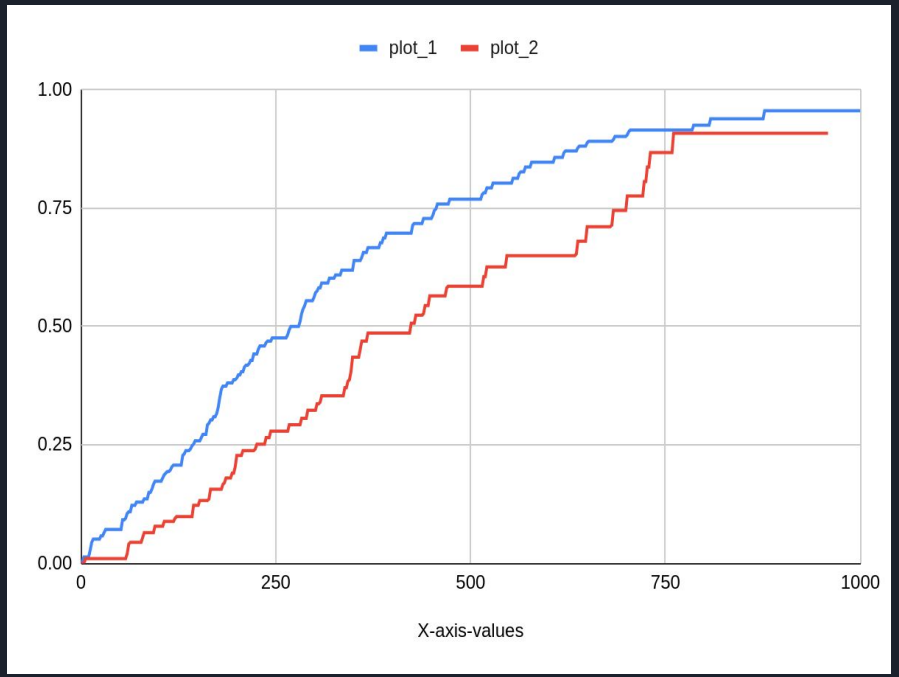

100
90
80
70
60
50
40
30
20
10
0
p=0.0873 (df=1)
0
3
6
9
12
15
18
21
24
(months)
O
N
Number of patients at risk :
72 73 26 14 8 6 6 4 4 <644
72 73 18 13 8 4 1 1 1 >=644 - <1290
73 73 19 6 3 1 0 0 0 >=1290 - <2303
73 73 15 8 4 3 2 1 1 >=2303
Largest les ion area (r

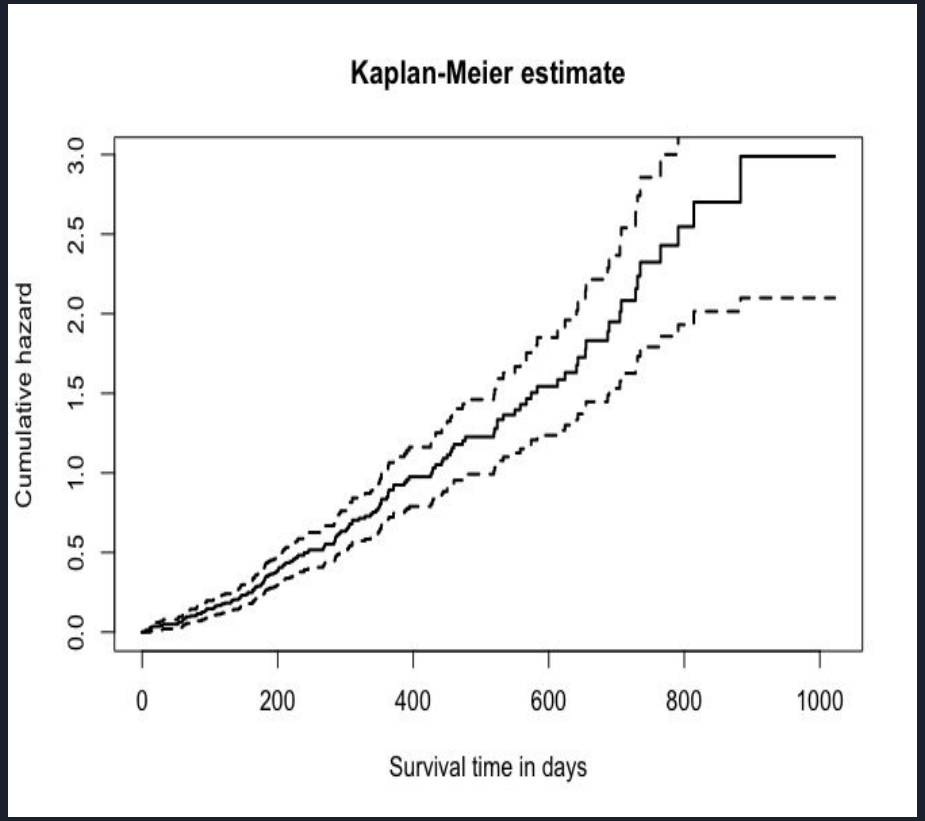

plot_1
plot_2
plot_3
plot_4
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
X-axis-values

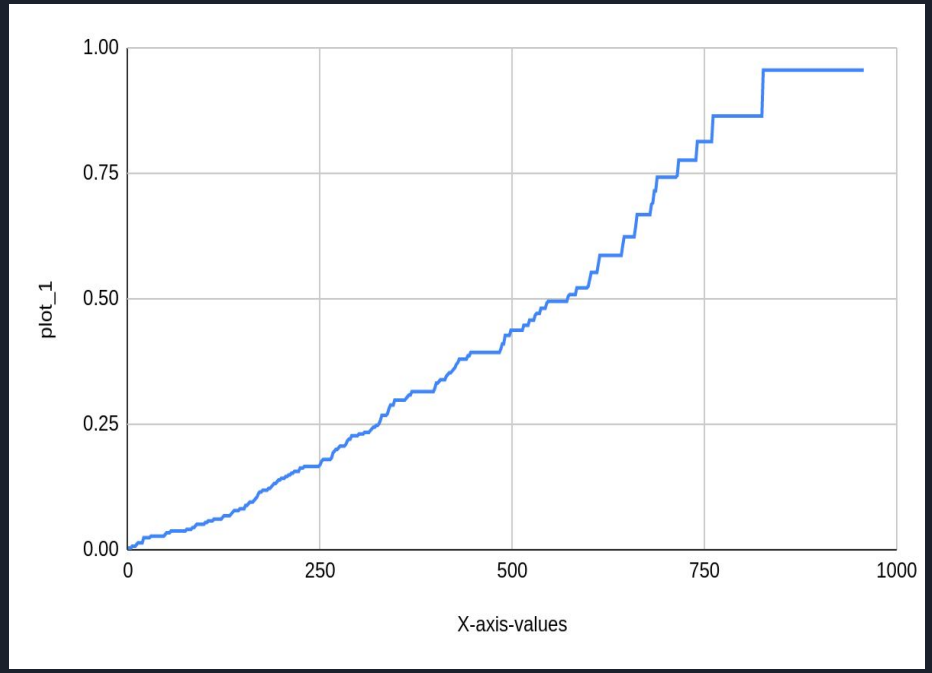

Log-Rank test P = 0.003
Overall survival probability
1.0
0.8
0.6
0.4
0.2
0.0
FA Yes; DXM Yes (n=141)
FA Yes; DXM No (n=180)
FA No; DXM Yes (n=191)
FA No; DXM No (n=76)
0
20
40
60
80
100
120
Time after surgery (month)


plot_1
plot_2
plot_3
plot_4
1.00
0.75
0.50
0.25
0.00
0
25
50
75
100
X-axis-values


Kaplan-Meier cumulative incidence by sex
Cumulative incidence
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Survival time in days
male
female


plot_1
plot_2
1.00
0.75
0.50
0.25
0.00
0
250
500
750
1000
X-axis-values


Kaplan-Meier estimate
Cumulative hazard
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
200
400
600
800
1000
Survival time in days


plot_1
1.00
0.75
0.50
0.25
0.00
0
250
500
750
1000
X-axis-values

Next Steps

- Improving x axis scale identification
  - Current implementation assumes that xlabels begin at value 0. Further, it uses tesseract's basic single line OCR mode. It fails in some cases when xlabels are highly disfigured.
  - Can try to use other OCR modules.
  - Can try to isolate individual xlabels and perform OCR on each separately.

- Handling dashed lines in graph
  - The current implementation ignores dashed line plots because some examples use dashed lines to show error bounds (and not the actual plot itself).
  - The current code relies on plots being the large connected components for the following
    - Filtering out areas like the legend where smaller sections of the same colours may be found
    - Identifying the starting point of all plots
  - If dashed lines are to be supported, these specific functionalities alone need to be reworked. The coordinate interpolation logic works for dashed lines too.

THANK YOU