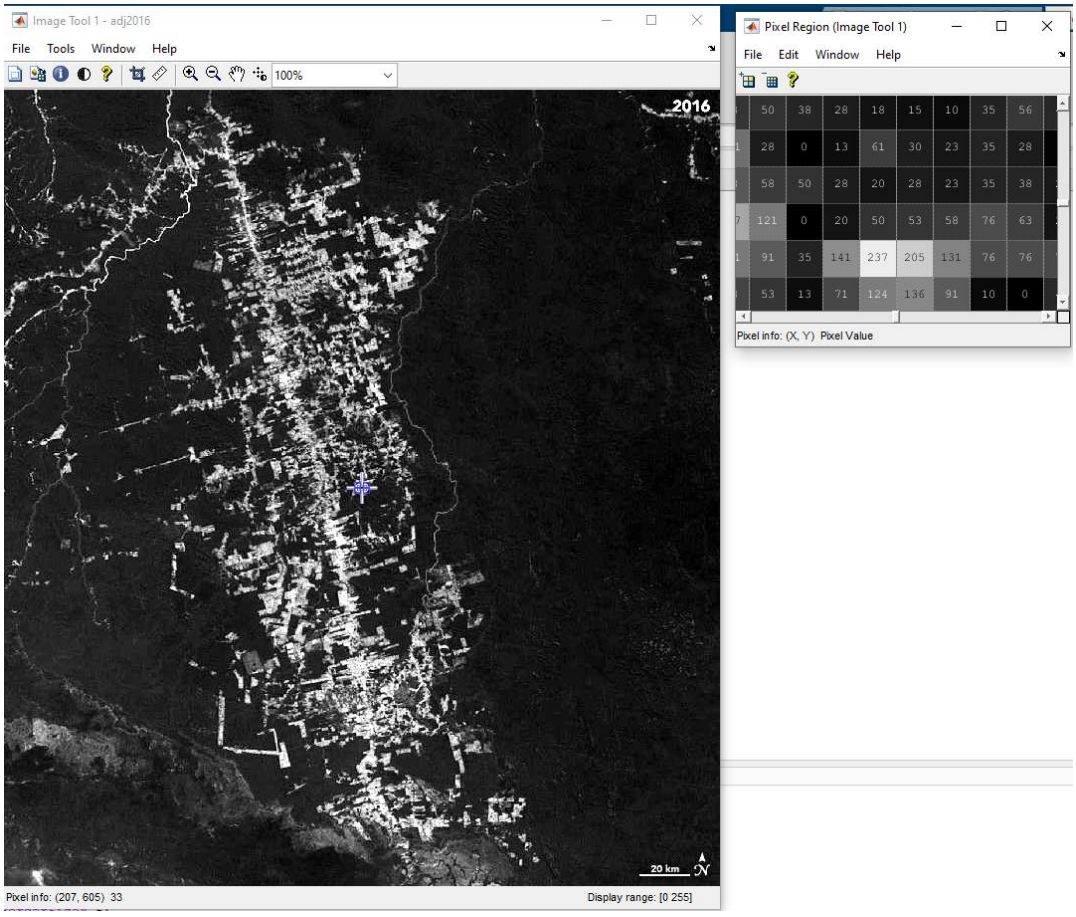
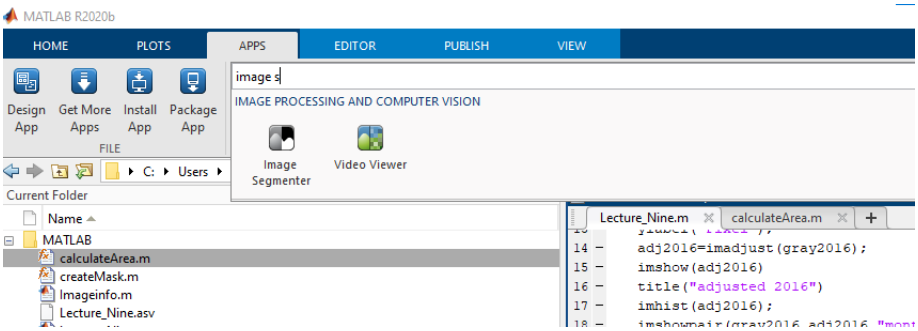
Lecture 11: Image Processing in MATLAB

1. Using `imtool` function we can check which pixels belong to the deforested areas.

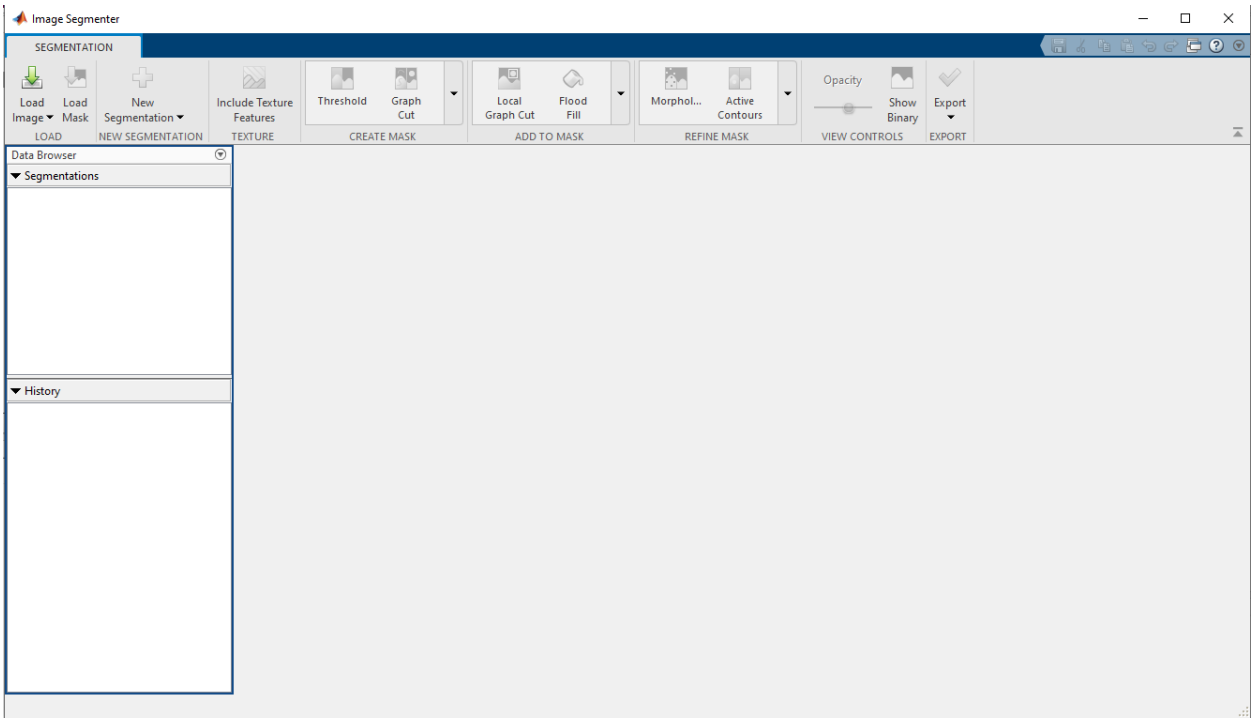
```
imtool(adj2016)
```



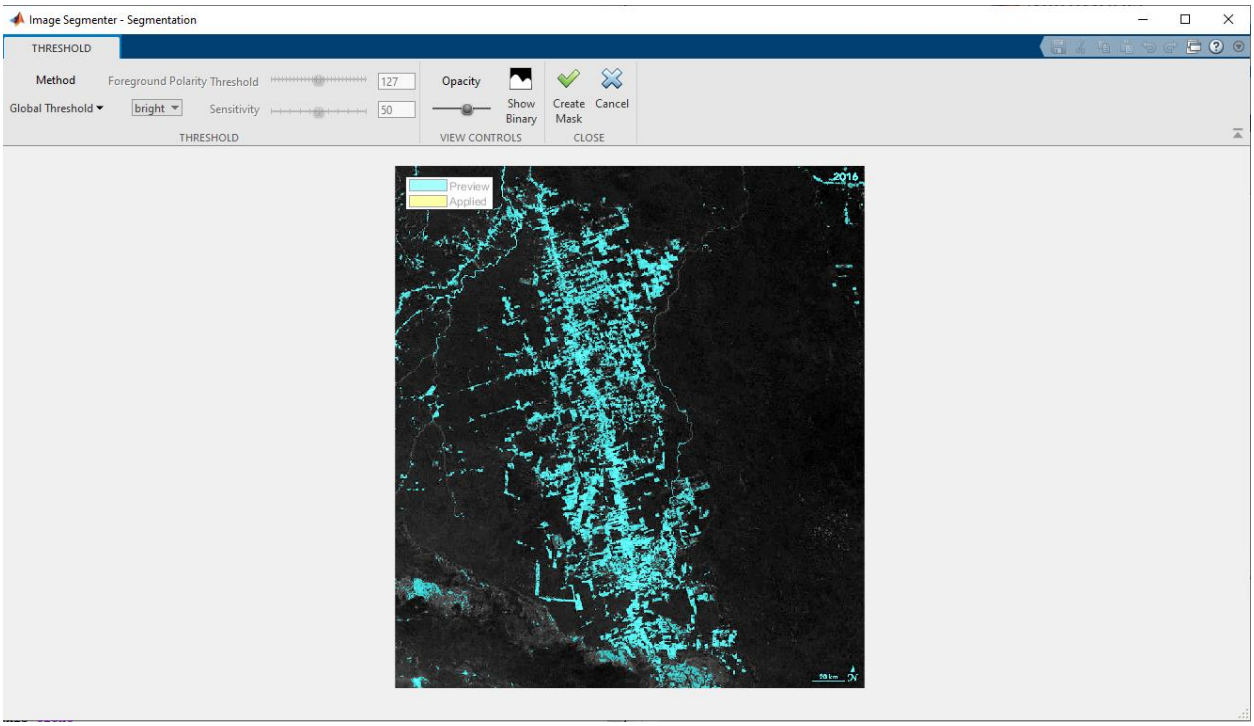
2. One way is to threshold the intensity values. To find the right one, you can use the Image Segmenter app. You can find it in the Apps tab and the Image Processing and Computer Vision section.



3. We'll start by loading our image.
   We'll use the adjusted grayscale image from 2016.
   We already adjusted the contrast, so we don't have to do it again.



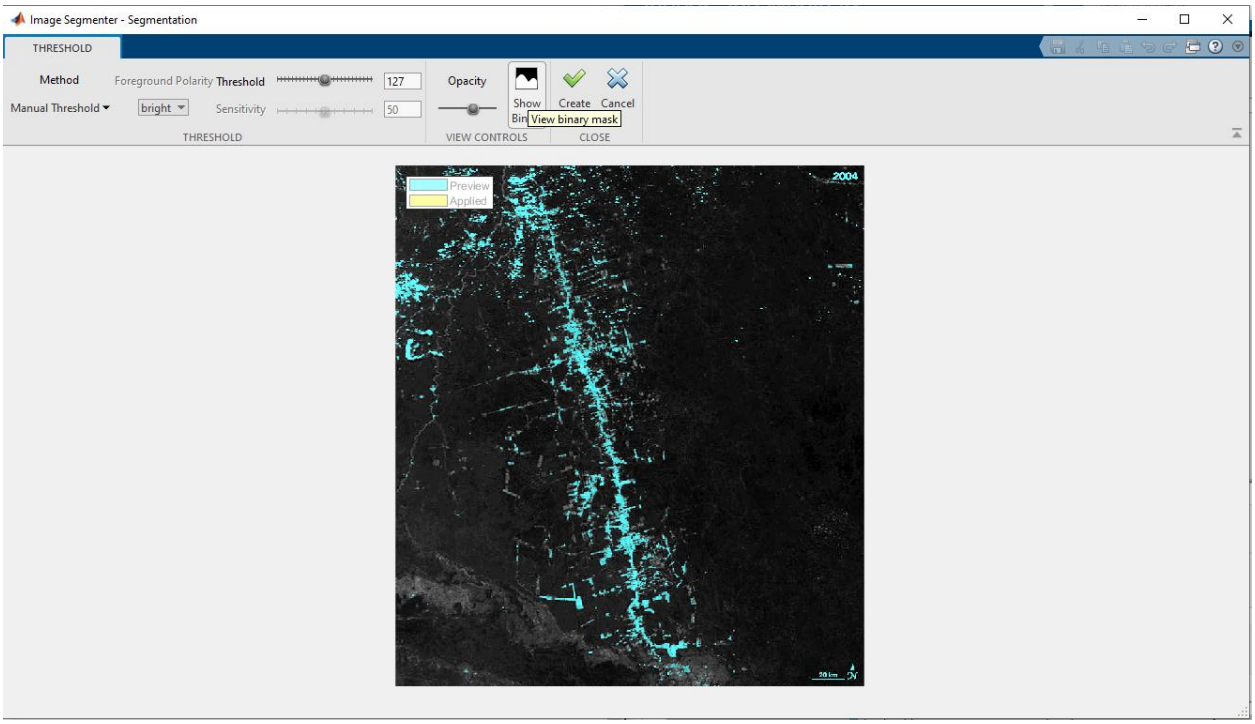
4. Let's start with a Threshold tool. The blue areas indicate what will be segmented based on your current threshold settings.



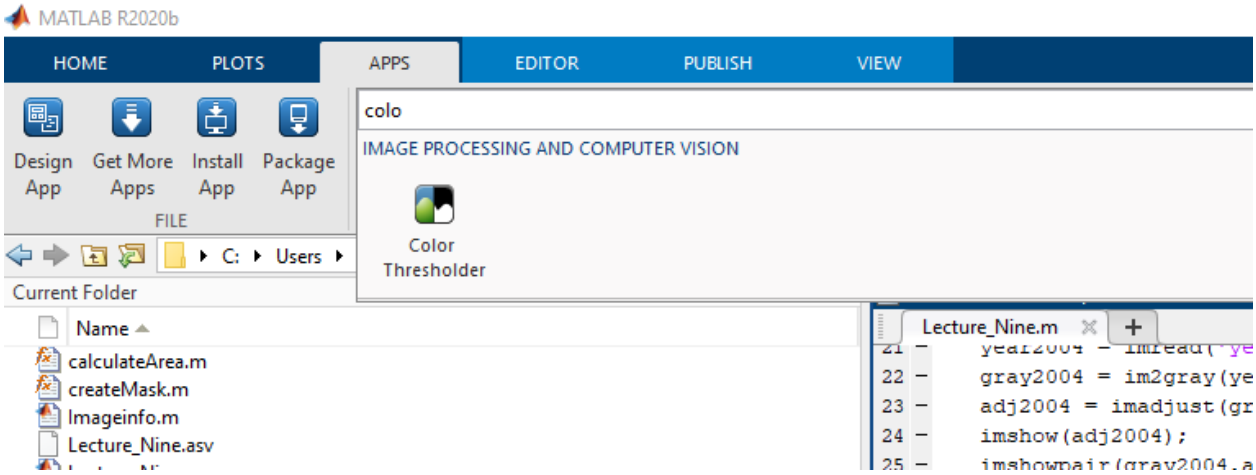
5. The default tries to automatically pick an optimal threshold. But you can also manually choose a threshold with manual option. A low threshold isolates the deforested areas well, but also falsely segments some areas that aren't deforested. A high threshold removes those false segmentations, but then you lose some areas that are deforested. It's all about balance. In this case, a threshold around 100 does a good job.
6. To view the result, Click Show Binary. It looks pretty good. But we can see a few small problems. For example, river that shouldn't be segmented.
7. Morphology, allow us to modify the shape of the segmented areas. Choose the type of operation. Since we want to remove the river, let's try the Open Mask operation, which removes areas smaller than a given shape.
8. For the shape, choose one that resembles the areas you want to keep. In our case, the areas are square. Next, set a size for the shape. If it's too small, then you might not remove the river, but if it's too big, you'll remove deforested areas. In this case, a length of 3 removes the river and leaves the rest mostly intact.
9. OK, we've segmented one image. So let's try another one. How about 2004?

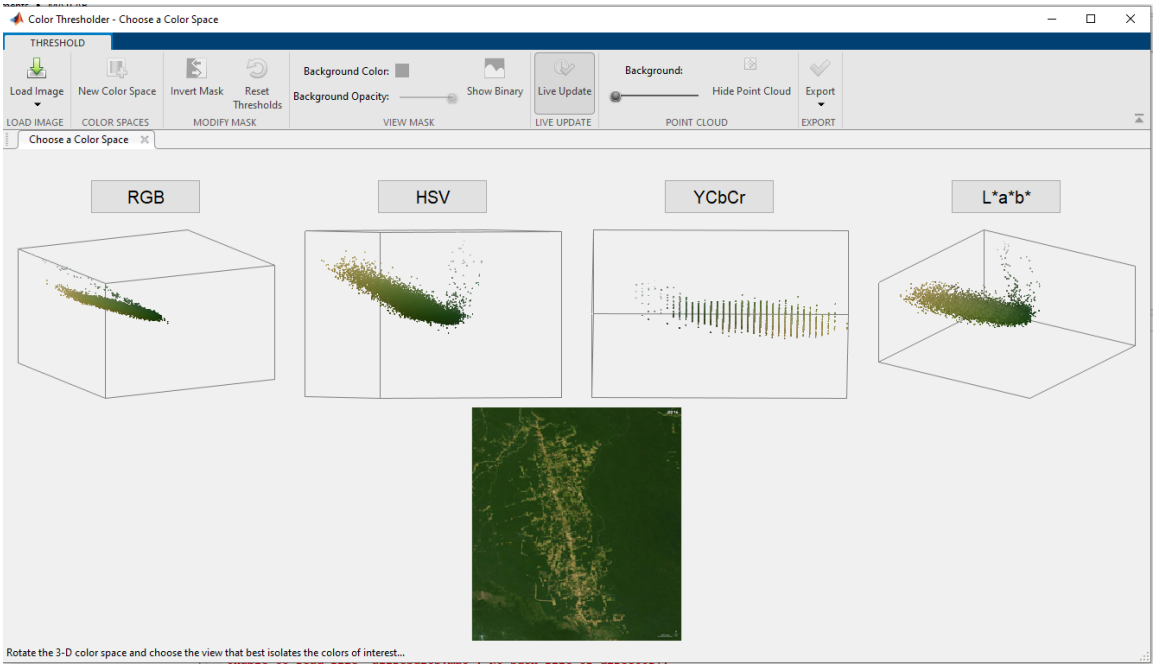
```
year2004 = imread("year2004.jpg");
gray2004 = im2gray(year2004);
adj2004 = imadjust(gray2004);
```

10. Then back in the app, let's import the adjusted image. It looks like the same threshold also segments most of the deforested areas. However, this image contains some clouds from the atmosphere that block our view.

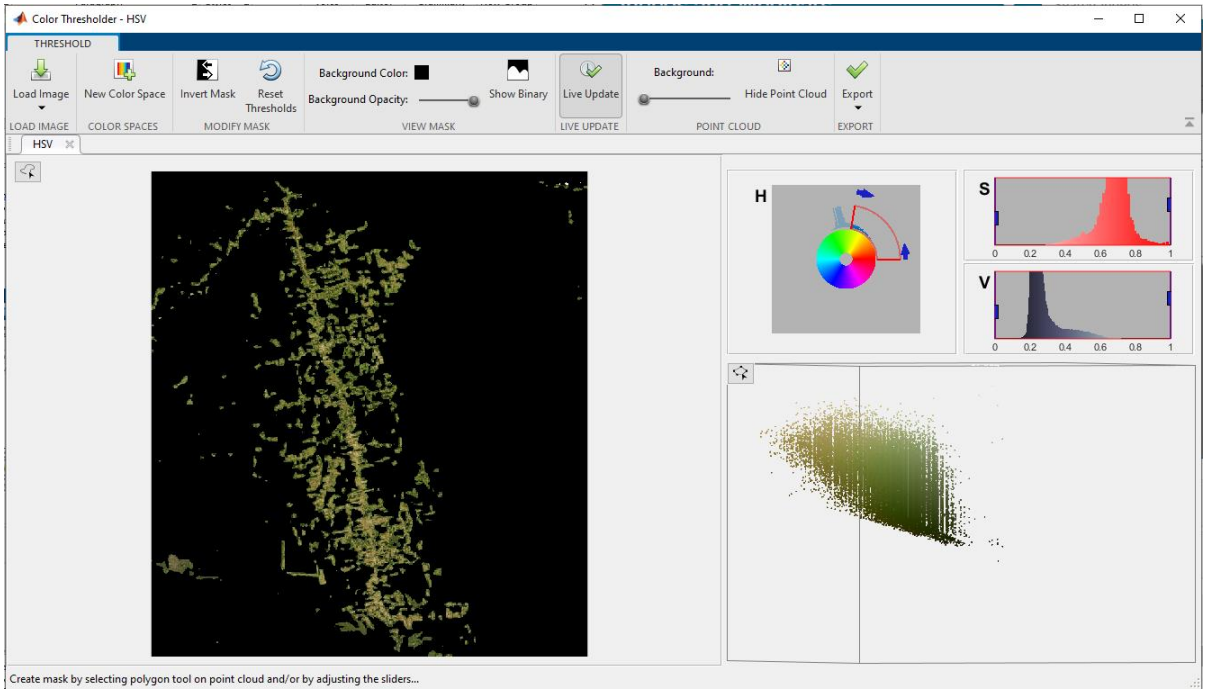


11. So it looks like we're not going to be able to use intensity to segment this image. How about color? The deforested areas have a distinct brown color that's different from the white and dark green colors in the rest of the image. We can segment based on color using the Color Thresholder app. Let's use the image from 2016, because it has more of that brown color that we're looking for.





12. They visualize the colors in a 3D space. You'll want to pick the one that isolates the brown color the most. These three are all equally good. So let's just pick one and start. How about HSV?
13. Hue determines the color. Saturation sets the shade. And value describes the brightness. Let's start with hue.



14. You can export the image itself, but since we're going to do this for multiple images, let's generate a function instead. We'll save it as createMask in our current folder.
15. let's try it out on the image from 2004. Last time, the clouds got in our way. This time, it looks like our function was able to remove the clouds while preserving the deforested areas.
16. Let's try it out on the image from 2004. Last time, the clouds got in our way. This time, it looks like our function was able to remove the clouds while preserving the deforested areas.

```
bw2004 = createMask(year2004);
imshowpair(year2004,bw2004,"montage")
title("2004 with color thresholding")
```

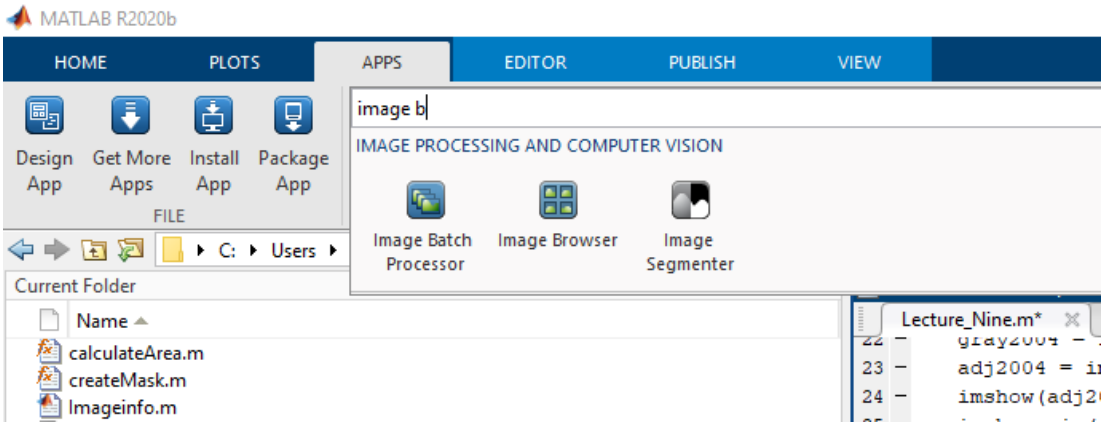
17. Calculate the area of the deforested regions. The regionprops function calculates a lot of useful information about the segmented regions in an image. We're only interested in the area of each region. So let's use the Area option. Then, to get the total area, we'll add them all up.

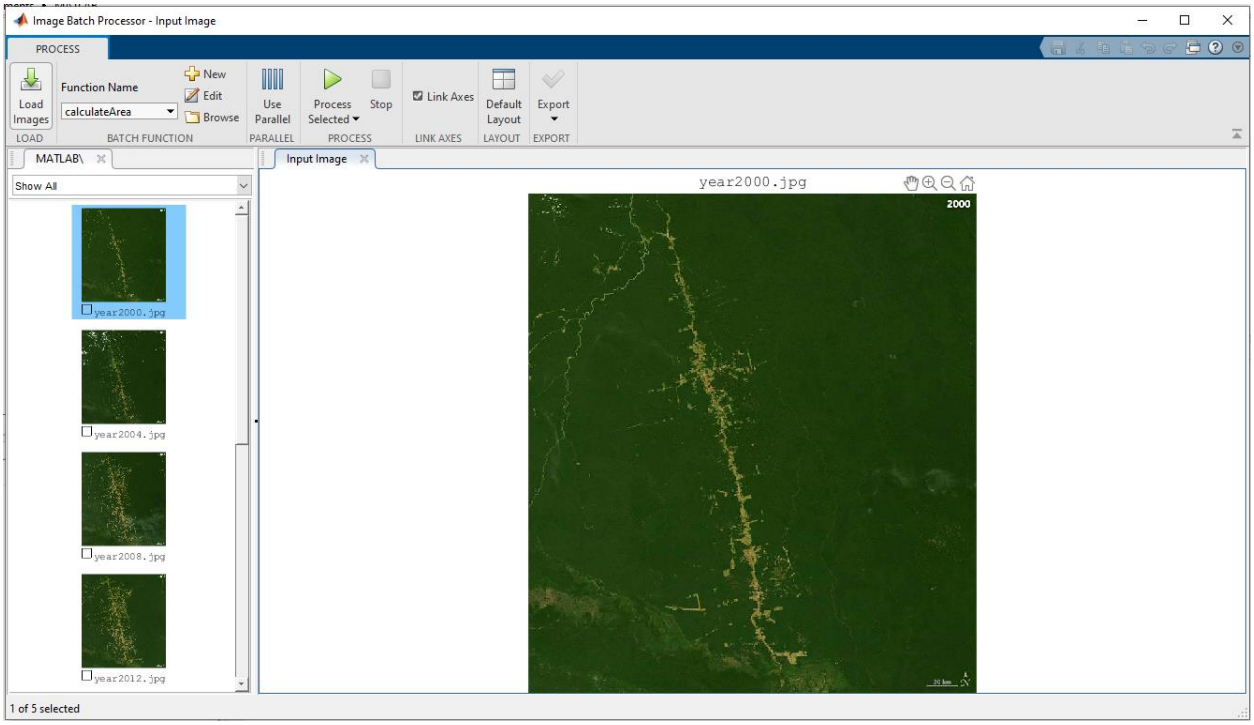
```
props = regionprops("table",bw2004,"all")
props = regionprops("table",bw2004,"Area")
areaPixels = sum(props.Area)
```

18. Now, this area is in pixels. So we need to convert it to square kilometers
19. To measure the scale, we can use the Image Viewer app. All right, let's Zoom in. The app has a ruler tool that we can use to measure the length of this line. It looks like 20 kilometers is about 51 pixels.

```
px2km = (51/20)^2
areaKmSq = round(areaPixels*px2km)
```

20. Now, we'll need to do this for all the images, which we can do using the Image Batch Processor app.



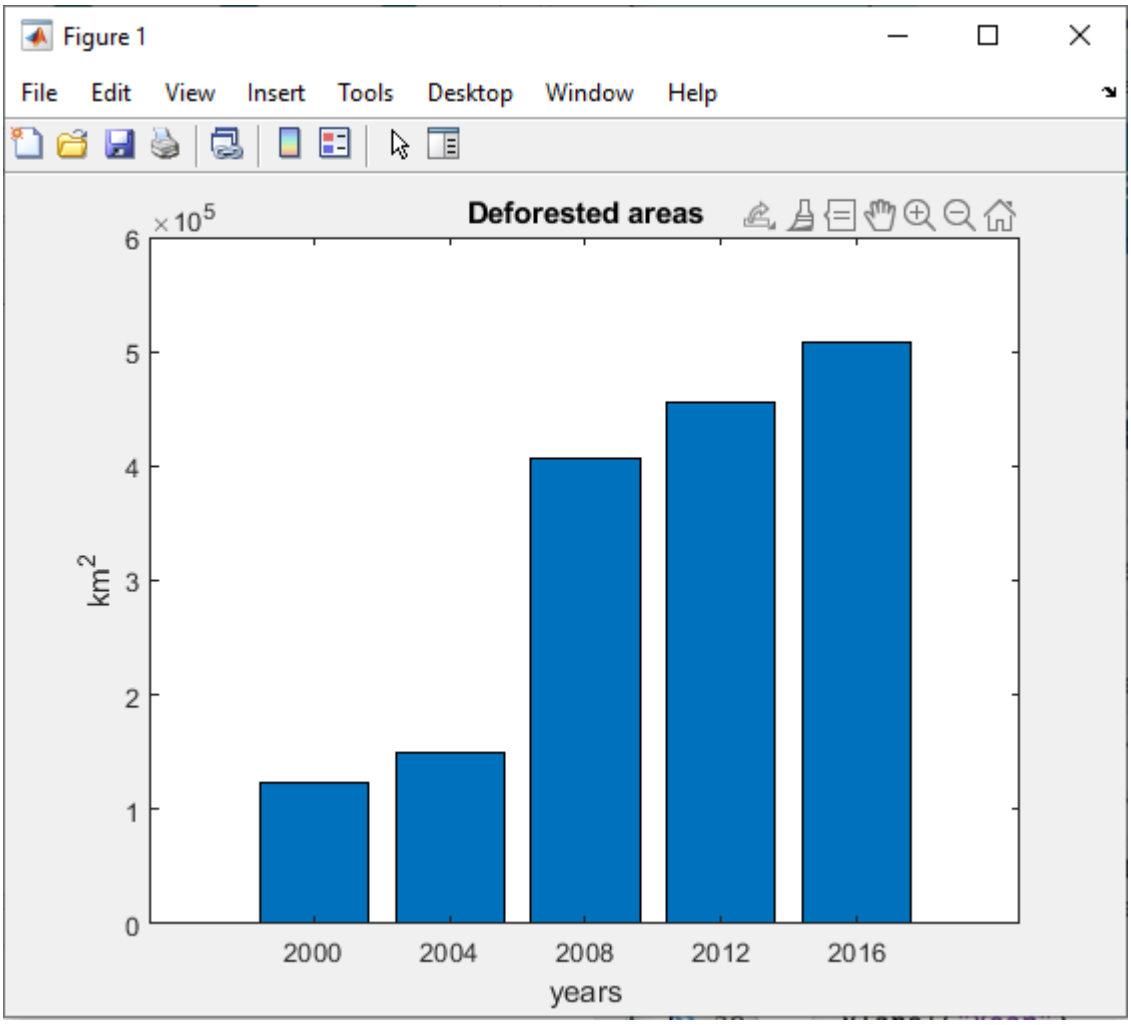


21. To calculate and segment the area of each image a program is written and save as shown below:

```
function results = calculateArea(I)
% Segment
BW = createMask(I);
% Area in pixels
props = regionprops("table",BW,"Area");
areaPixels = sum(props.Area);
% Area in km^2
px2km = (51/20)^2;
areaKmSq = round(areaPixels*px2km);
% Format into structure
results.BW = BW;
results.areaKmSq = areaKmSq;
```

22. Let's export to the workspace so we can analyze the results further. We'll include everything in a table named all results.
23. Create a bar chart of the areas so we can see the trend. There was a considerable jump between 2004 and 2008. After that, there's a significant reduction in the deforestation rate, showing the impact of conservation efforts.

```
years = 2000:4:2016;
areas = allresults.areaKmSq;
bar(years,areas)
title("Deforested areas in km^{2}")
xlabel("Year")
ylabel("km^{2}")
```



24. To see all the segmented images, we can use the functions bellow:

```
segmentedImages = allresults.BW;
montage(segmentedImages,...
    "BackgroundColor","White",...
    "BorderSize",5)
title("All segmented images")
```

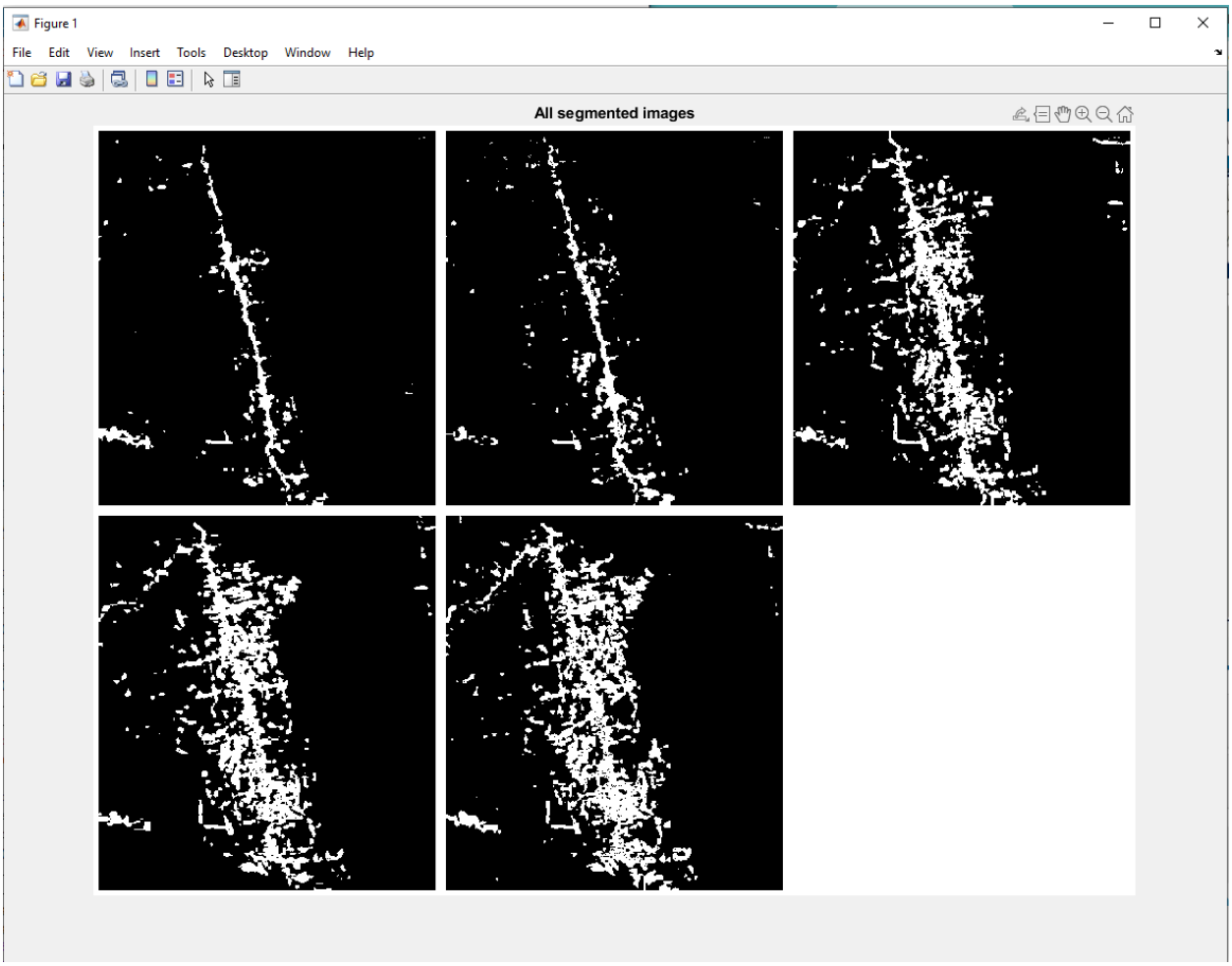


Figure 1
File Edit View Insert Tools Desktop Window Help
All segmented images