+/- Message

```
## Loading required package: knitr
```

+/- R Code

Quantitative Big Imaging

author: Kevin Mader date: 13 March 2014 width: 1440 height: 900 transition: rotate

Advanced Segmentation and Labeling

Course Outline

- 20th February - Introductory Lecture
- 27th February - Filtering and Image Enhancement (A. Kaestner)
- 6th March - Basic Segmentation, Discrete Binary Structures
- 13th March - **Advanced Segmentation**
- 20th March - Analyzing Single Objects
- 27th March - Analyzing Complex Objects
- 3rd April - Spatial Distribution
- 10th April - Statistics and Reproducibility
- 17th April - Dynamic Experiments
- 8th May - Big Data
- 15th May - Guest Lecture - Applications in Material Science
- 22th May - Project Presentations

Literature / Useful References

- Jean Claude, Morphometry with R
  - Online (http://link.springer.com/book/10.1007%2F978-0-387-77789-4) through ETHZ
  - Buy it (http://www.amazon.com/Morphometrics-R-Use-Julien-Claude/dp/038777789X)
- John C. Russ, “The Image Processing Handbook”,(Boca Raton, CRC Press)
  - Available online (http://dx.doi.org/10.1201/9780203881095) within domain ethz.ch (or proxy.ethz.ch / public VPN)
- Markov Random Fields for Image Processing Lecture (https://www.youtube.com/watch?v=vRN_j2j-CC4)
- Markov Random Fields Chapter (http://www.cise.ufl.edu/%7Eanand/pdf/bookchap.pdf)
- Fuzzy set theory

(http://www.academia.edu/4978200/Applications_of_Fuzzy_Set_Theory_and_Fuzzy_Logic_in_Image_Processing)
- Active Contours / Snakes (http://link.springer.com/article/10.1007%2FBF00133570#page-1)

Lesson Outline

- Motivation
  - Many Samples
  - Difficult Samples
- Thresholding
  - Automated Methods
  - Hysteresis Method
  - K-Means Clustering
  - Fuzzy Models
  - Probabalistic Models
- Working with Segmented Images
  - Contouring
  - Component Labeling

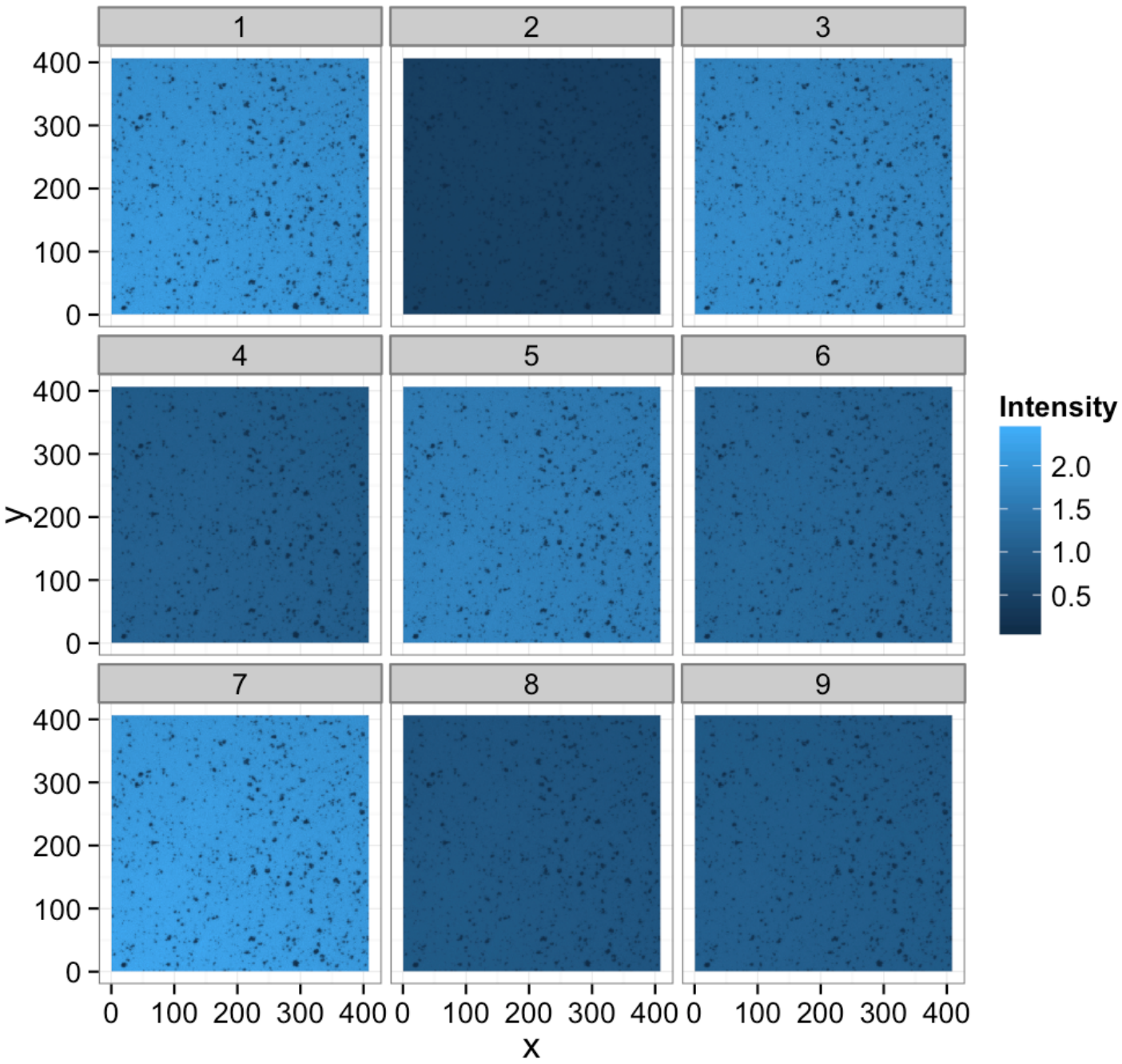
What we covered last time

- Understanding models and interpreting histograms
- Choosing a threshold
  - Examining more complicated, multivariate data sets
- Improving segementation with morphological operations
  - Filling holes
- Partial Volume Effect Caution

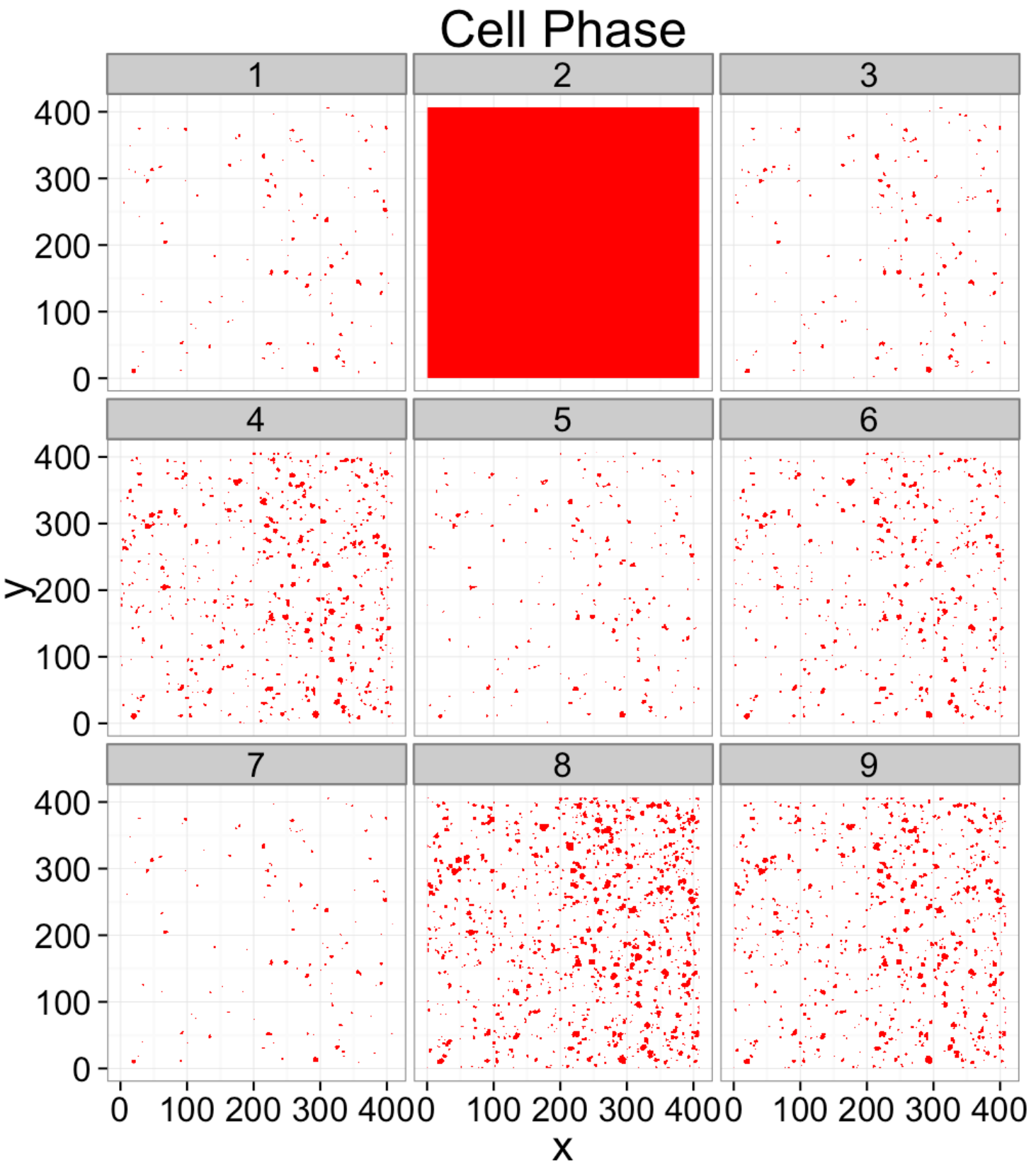
Where segmentation fails: Inconsistent Illumination

With inconsistent or every changing illumination it may not be possible to apply the same threshold to every image.

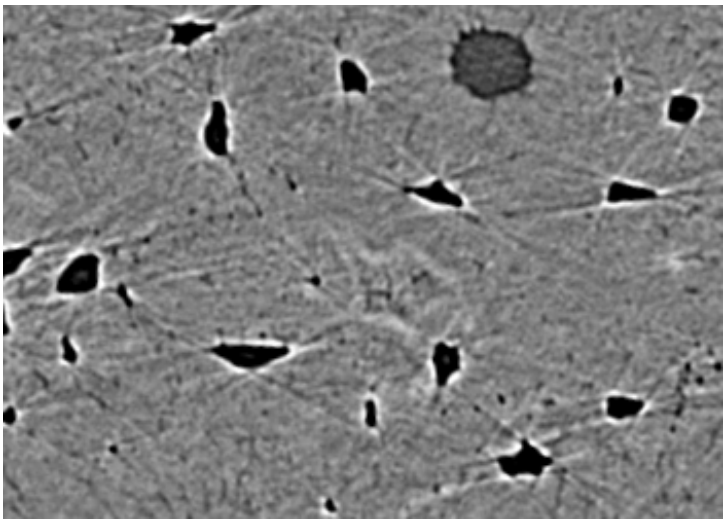
+/- R Code



+/- R Code


Cell Phase
1
2
3
4
5
6
7
8
9
y
x
0
100
200
300
400

Where segmentation fails: Canaliculi



Here is a bone slice

1. Find the larger cellular structures (osteocyte lacunae)
2. Find the small channels which connect them together

The first task

is easy using a threshold and size criteria (we know how big the cells should be)

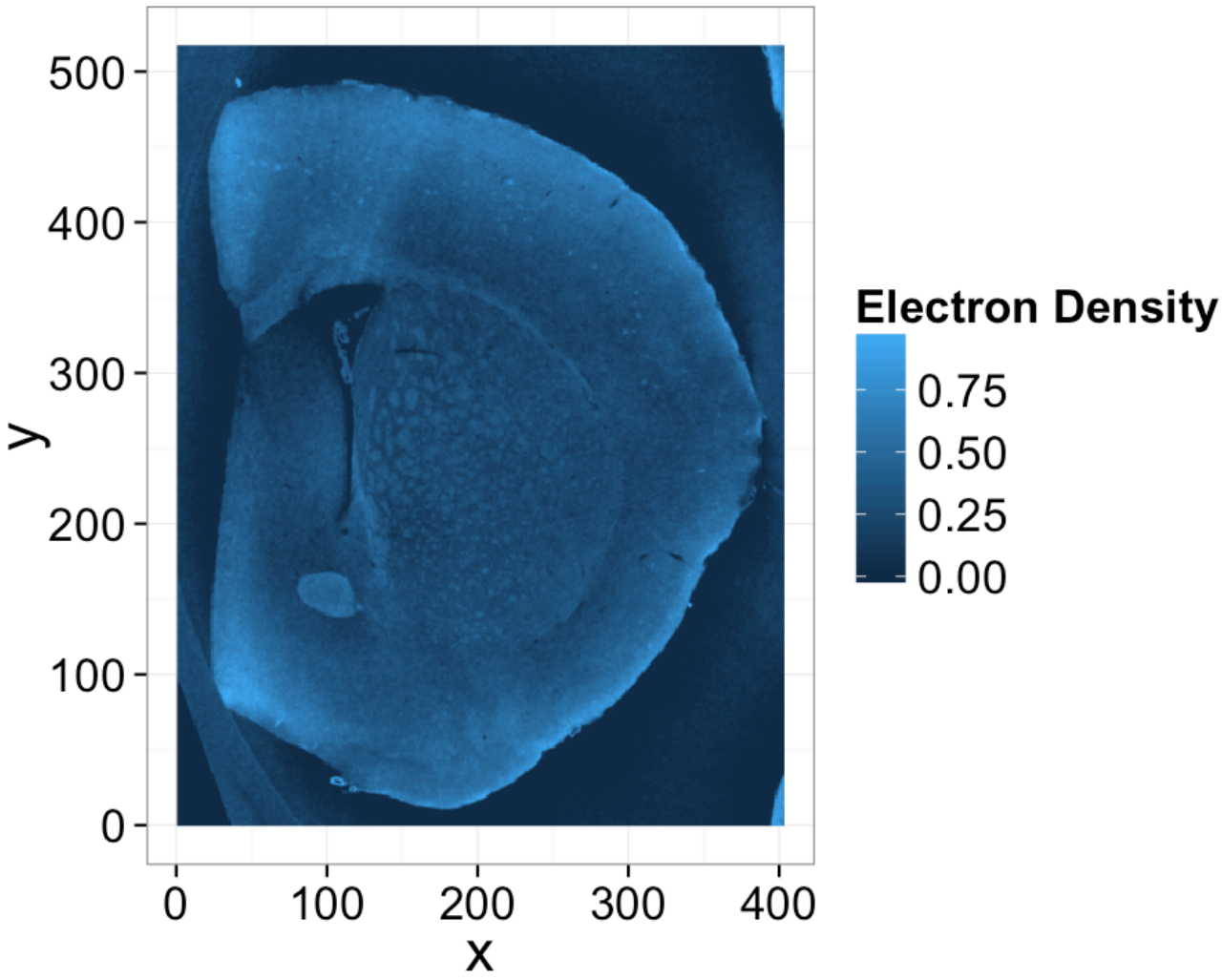
The second

is much more difficult because the small channels having radii on the same order of the pixel size are obscured by partial volume effects and noise.

Where segmentation fails: Brain Cortex

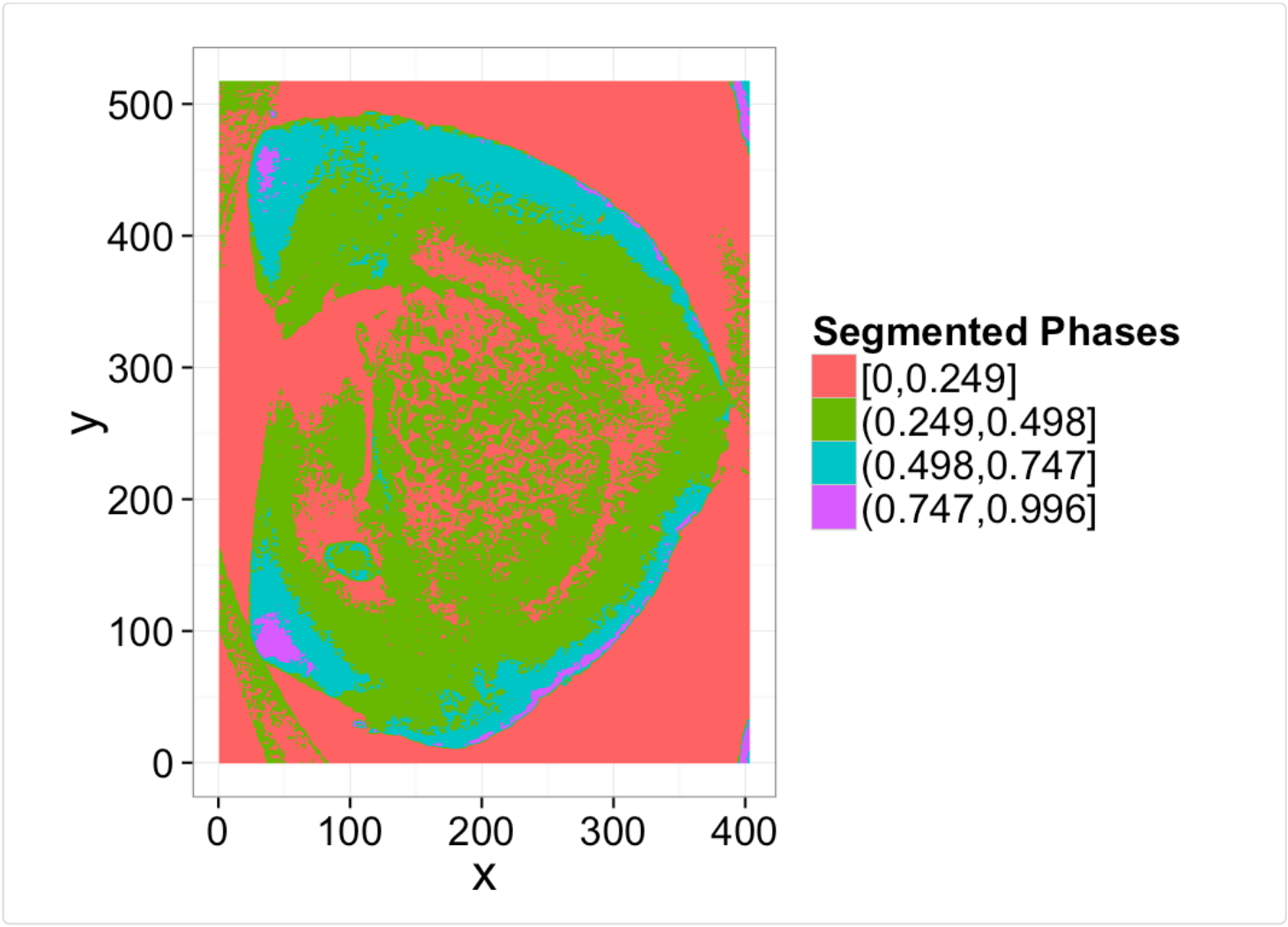
- The cortex is barely visible to the human eye
- Tiny structures hint at where cortex is located

+/- R Code

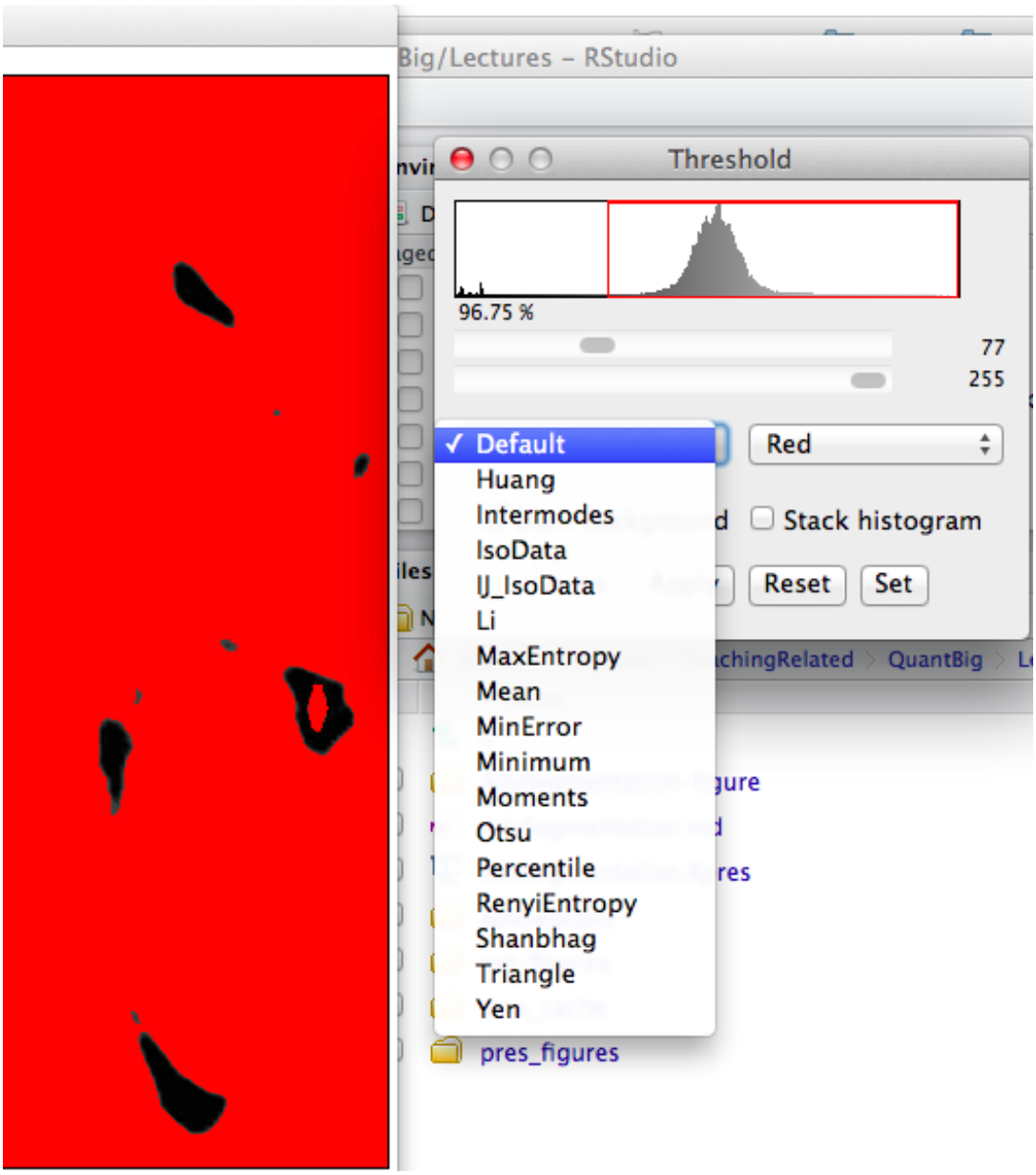


- A simple threshold is insufficient to finding the cortical structures
- Other filtering techniques are unlikely to magicially fix this problem

+/- R Code



Automated Threshold Selection



Given that applying a threshold is such a common and signficant step, there have been many tools developed to automatically (unsupervised) perform it. A particularly important step in setups where images are rarely consistent such as outdoor imaging which has varying lighting (sun, clouds). The methods are based on several basic principles.

Automated Methods

Histogram-based methods

Just like we visually inspect a histogram an algorithm can examine the histogram and find local minimums between two peaks, maximum / minimum entropy and other factors

- Otsu, Isodata, Intermodes, etc

Image based Methods

These look at the statistics of the thresheld image themselves (like entropy) to estimate the threshold

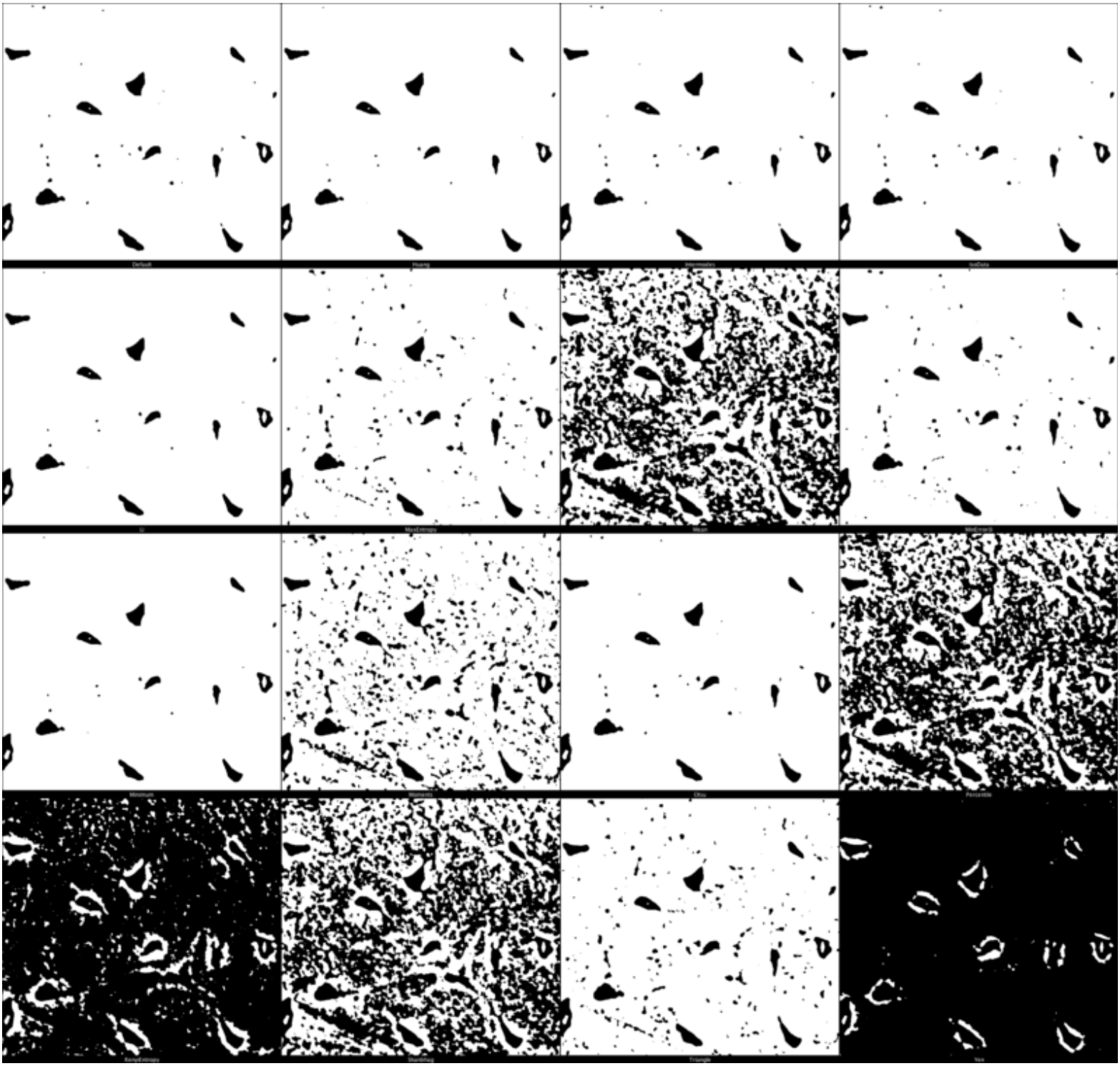
Results-based Methods

These search for a threshold which delivers the desired results in the final objects. For example if you know you have an image of cells and each cell is between 200-10000 pixels the algorithm runs thresholds until the objects are of the desired size

- More specific requirements need to be implemented manually

Fiji -> Adjust -> Auto Threshold

There are many methods and they can be complicated to implement yourself. FIJI offers many of them as built in functions so you can automatically try all of them on your image



Pitfalls

While an incredibly useful tool, there are many potential pitfalls to these automated techniques.

Histogram-based

These methods are very sensitive to the distribution of pixels in your image and may work really well on images with equal amounts of each phase but work horribly on images which have very high amounts of one phase compared to the others

Image-based

These methods are sensitive to noise and a large noise content in the image can change statistics like entropy significantly.

Results-based

These methods are inherently biased by the expectations you have. If you want to find objects between 200 and 1000 pixels you will, they just might not be anything meaningful.

Realistic Approaches for Dealing with these Shortcomings

Imaging science rarely represents the ideal world and will never be 100% perfect. At some point we need to write our master's thesis, defend, or publish a paper. These are approaches for more qualitative assessment we will later cover how to do this a bit more robustly with quantitative approaches

Model-based

One approach is to try and simulate everything (including noise) as well as possible and to apply these techniques to many realizations of the same image and qualitatively keep track of how many of the results accurately identify your phase or not. Hint: >95% seems to convince most biologists

Sample-based

Apply the methods to each sample and keep track of which threshold was used for each one. Go back and apply each threshold to each sample in the image and keep track of how many of them are correct enough to be used for further study.

Worst-case Scenario

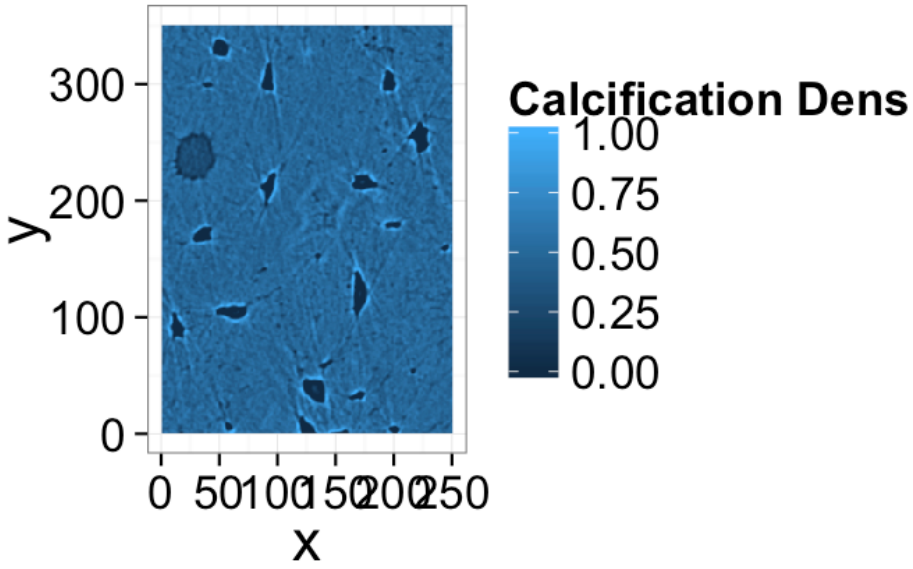
Come up with the worst-case scenario (noise, misalignment, etc) and assess how inacceptable the results are. Then try to estimate the quartiles range (75% - 25% of images).

Hysteresis Thresholding

For some images a single threshold does not work

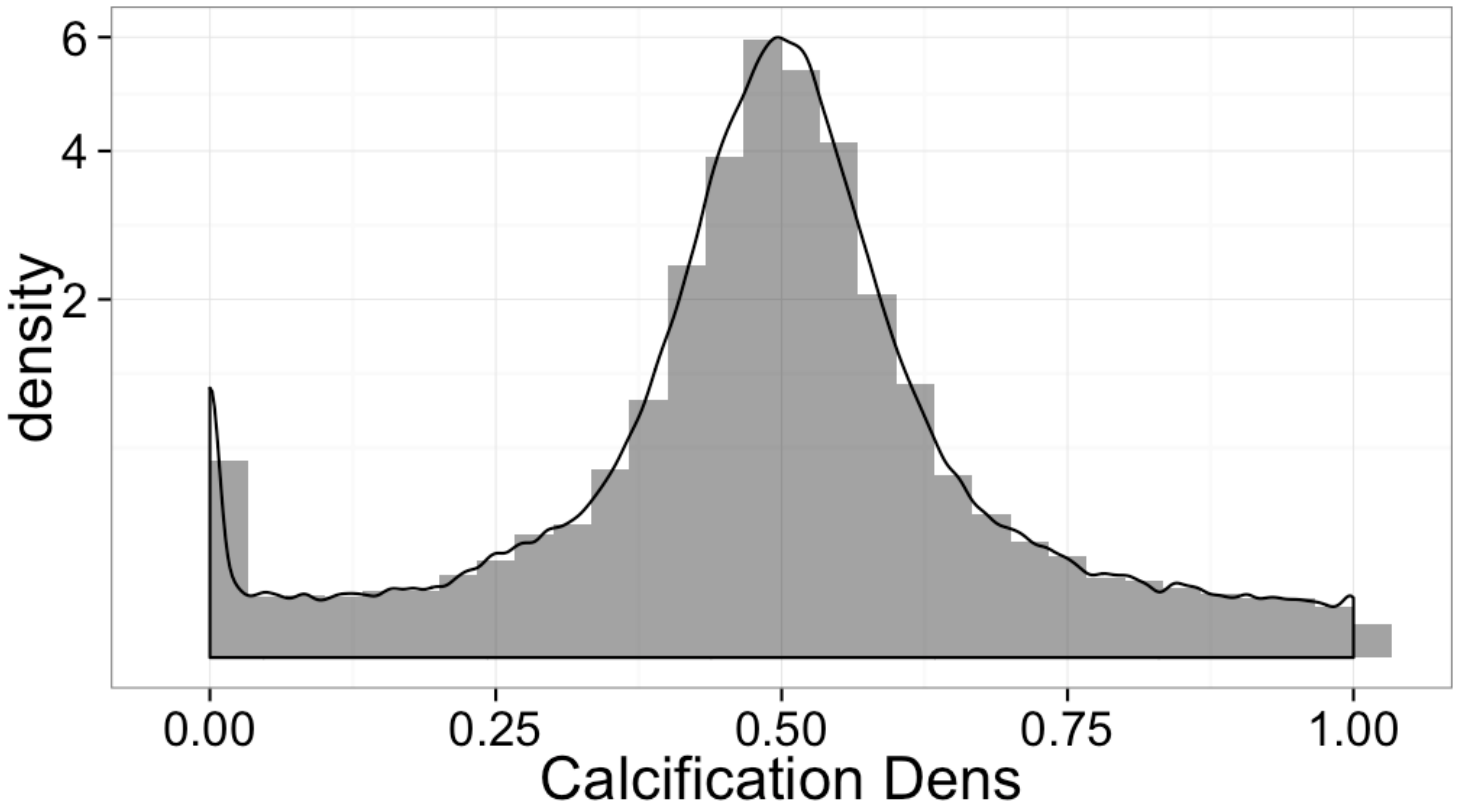
- large structures are very clearly defined
- smaller structures are difficult to differentiated (see partial volume effect (http://bit.ly/1mW7kdP))

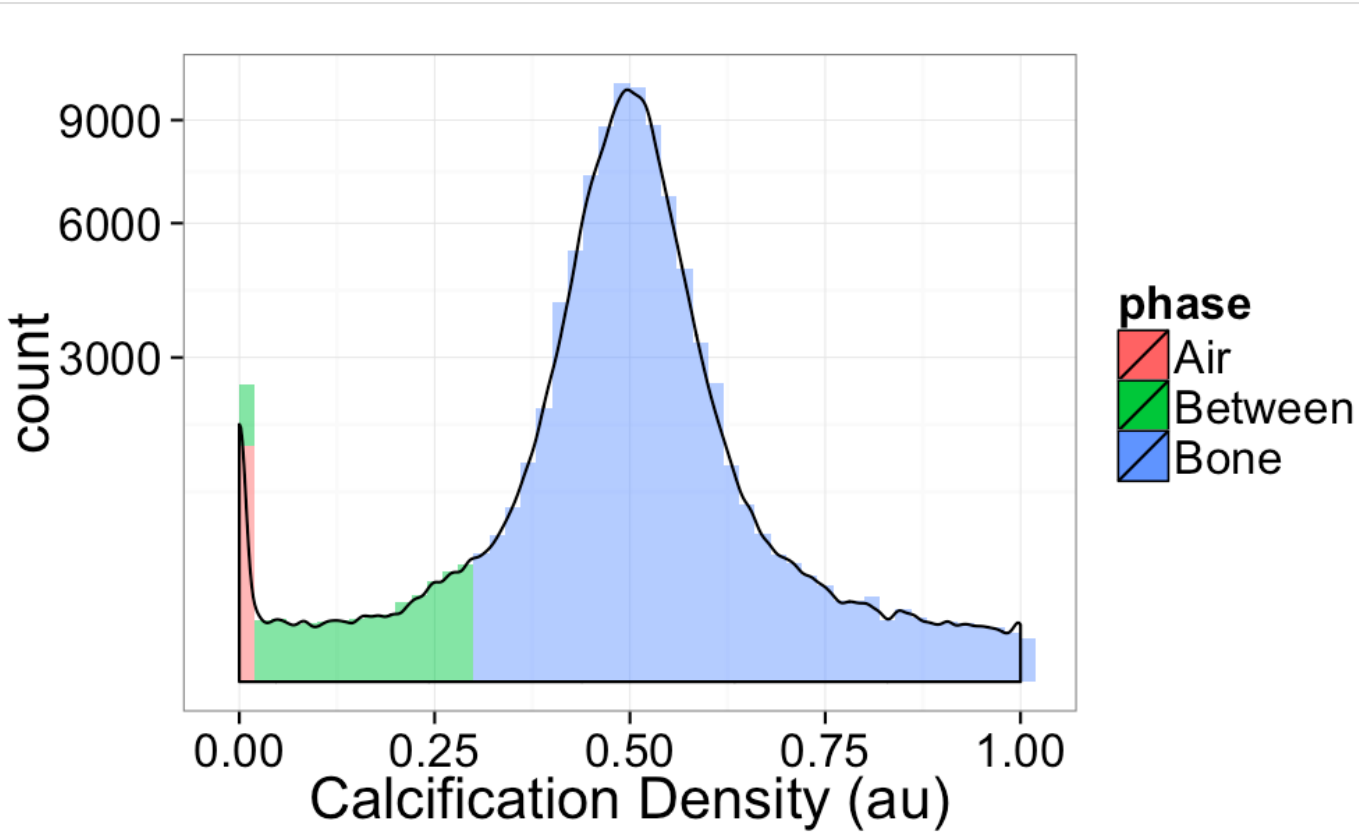
ImageJ Source (http://imagejdocu.tudor.lu/doku.php?id=plugin:segmentation:hysteresis_thresholding:start)



+/- R Code

+/- R Code

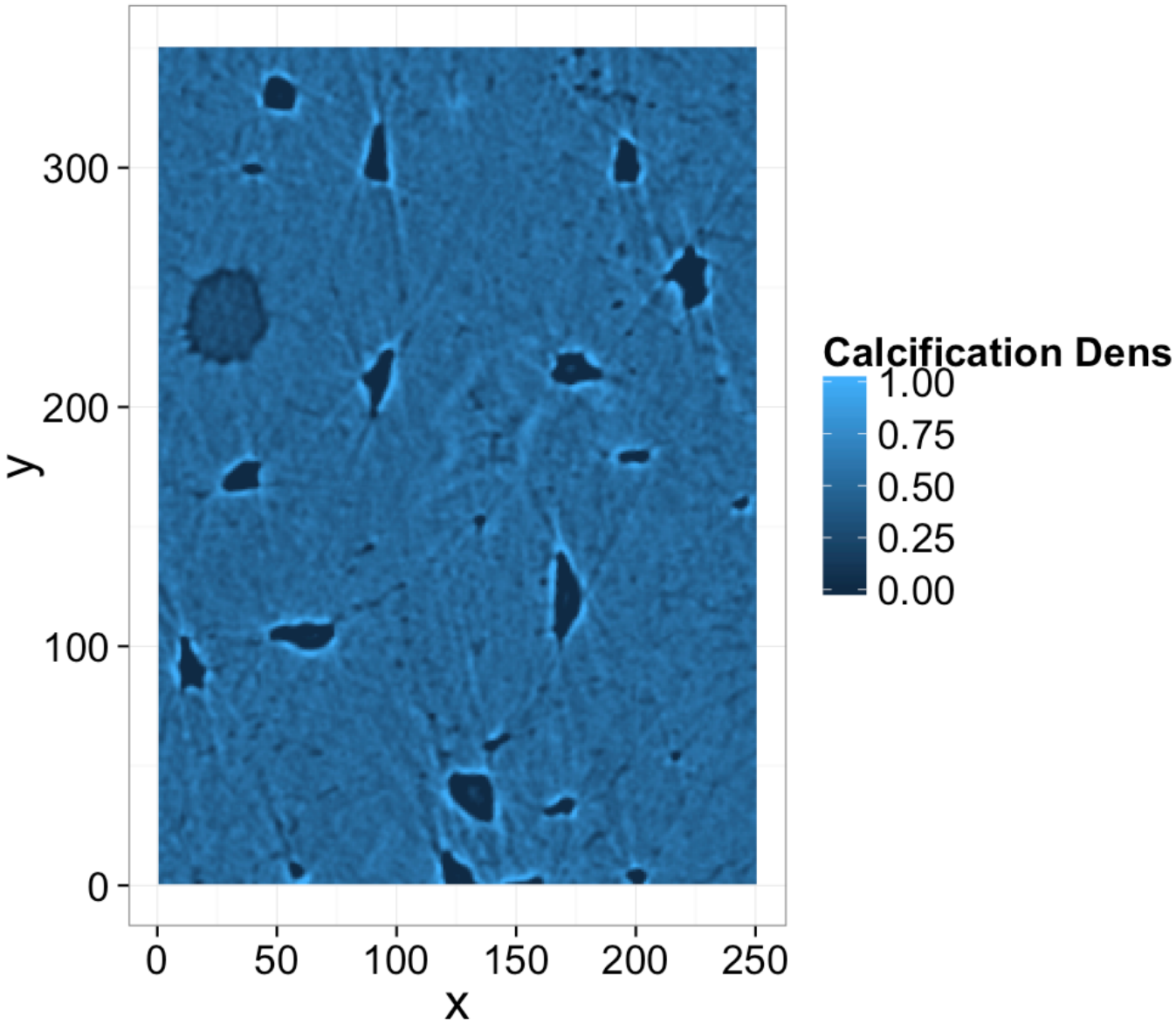

6
4
2
density
0.00
0.25
0.50
0.75
1.00
Calcification Dens


9000
6000
3000
count
0.00
0.25
0.50
0.75
1.00
Calcification Density (au)
phase
Air
Between
Bone

+/- R Code

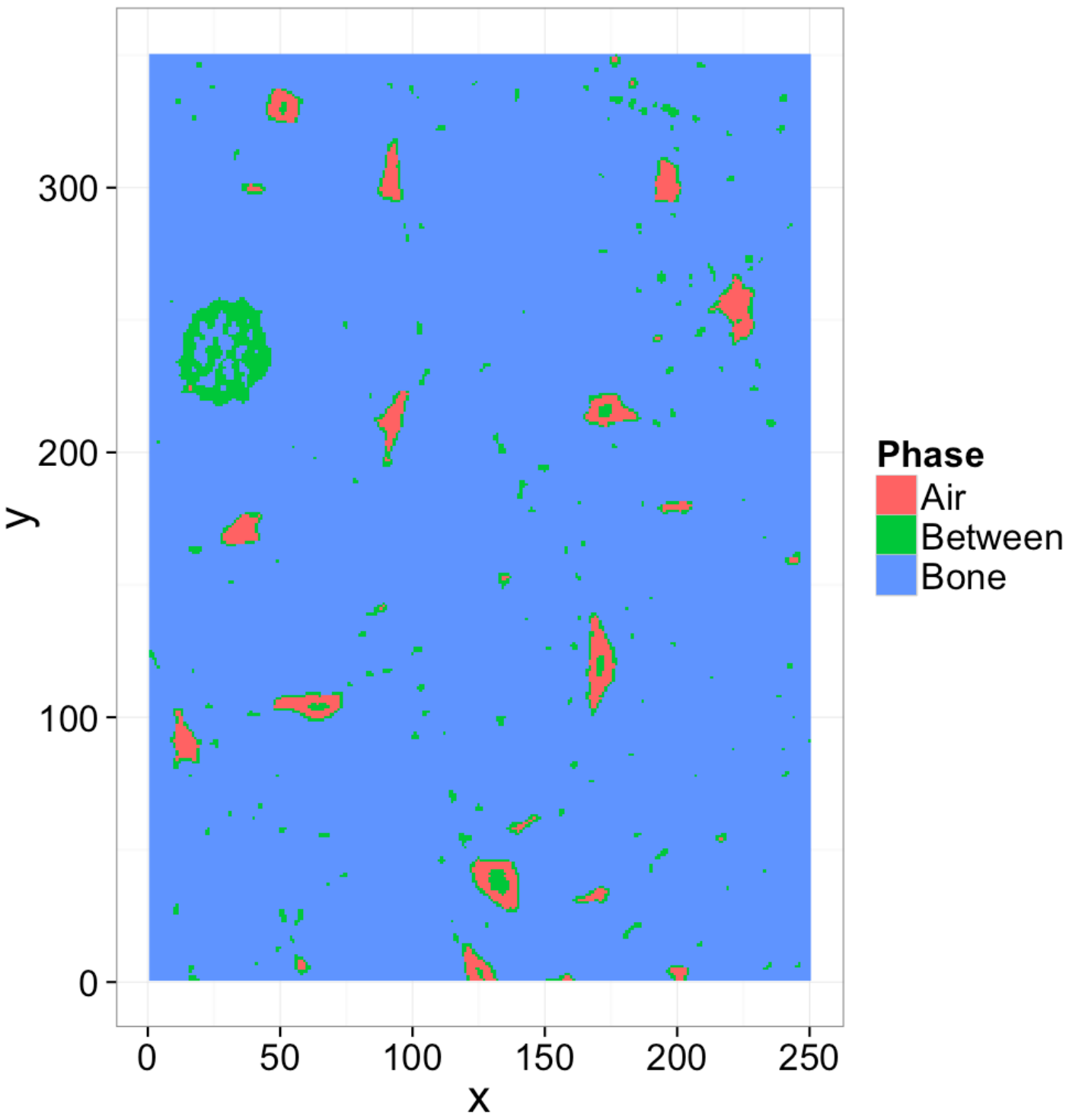
Hysteresis Thresholding

Comparing the original image with the three phases



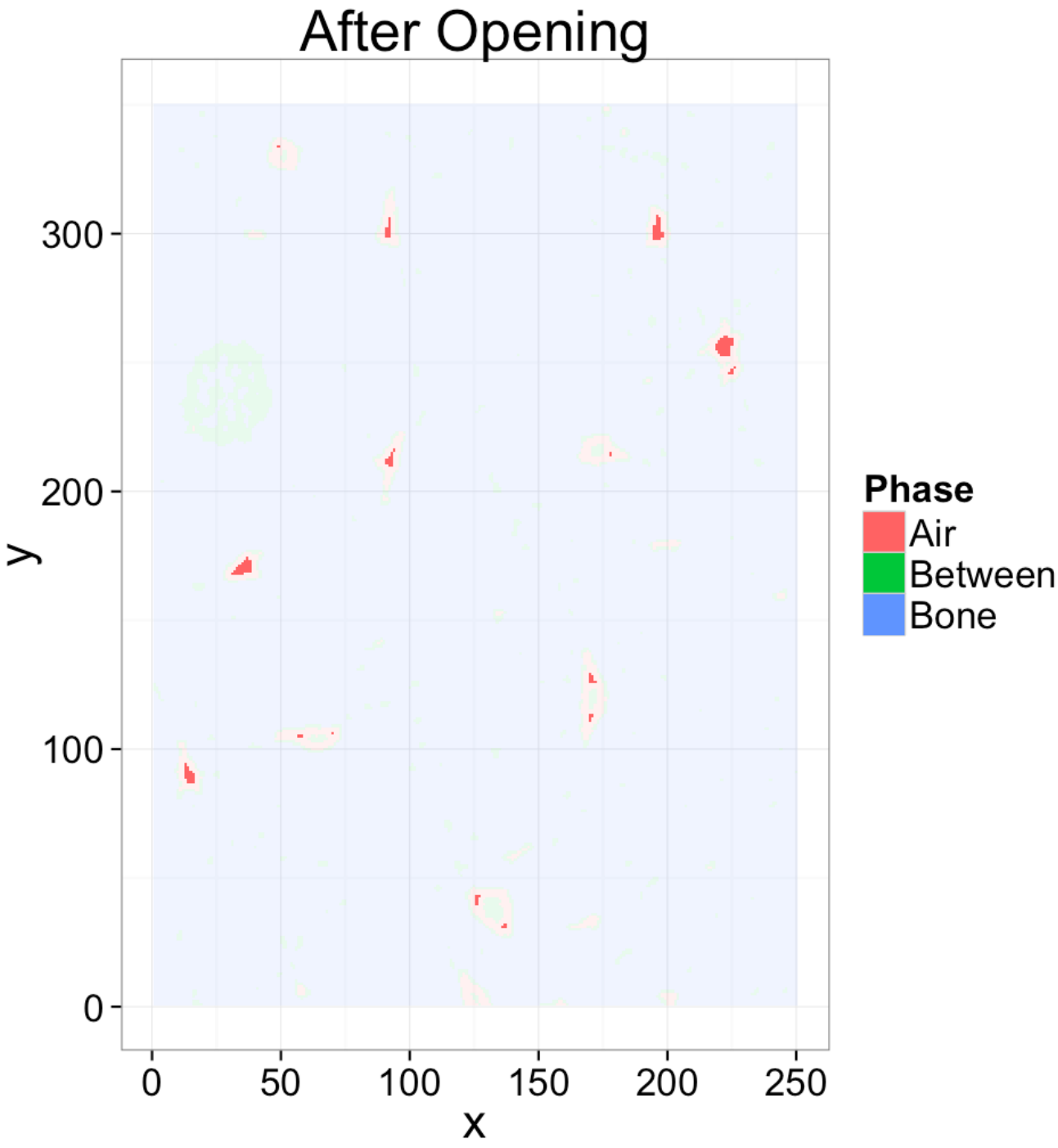
+/- R Code

+/- R Code


300
200
100
0
y
0
50
100
150
200
250
x
Phase
Air
Between
Bone

Hysteresis Thresholding: Reducing Pixels

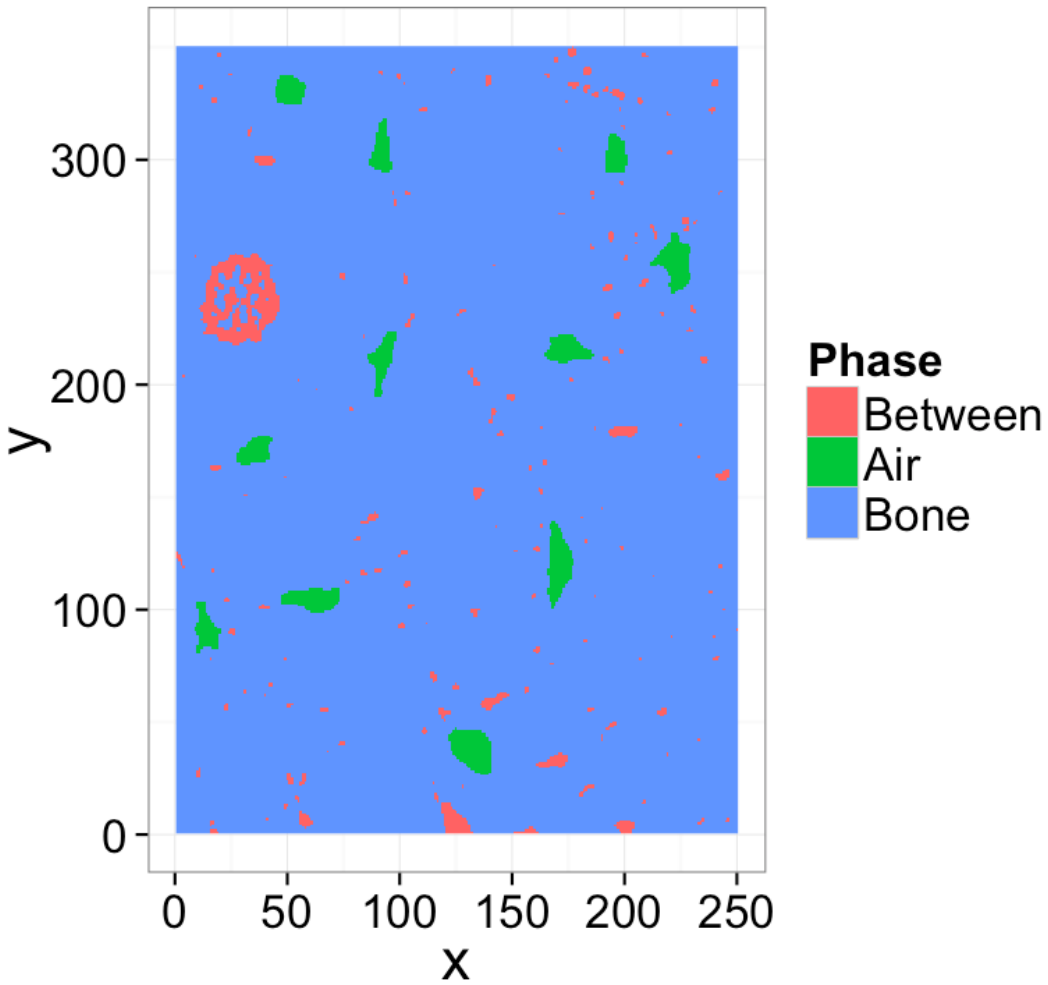
Now we apply two important steps. The first is to remove the objects which are not cells (too small) using an opening operation.



+/- R Code

+/- R Code

The second step to keep the *between* pixels which are connected (by looking again at a neighborhood $\mathcal{N}$) to the *air* voxels and ignore the other ones. This goes back to our original supposition that the smaller structures are connected to the larger structures



+/- R Code

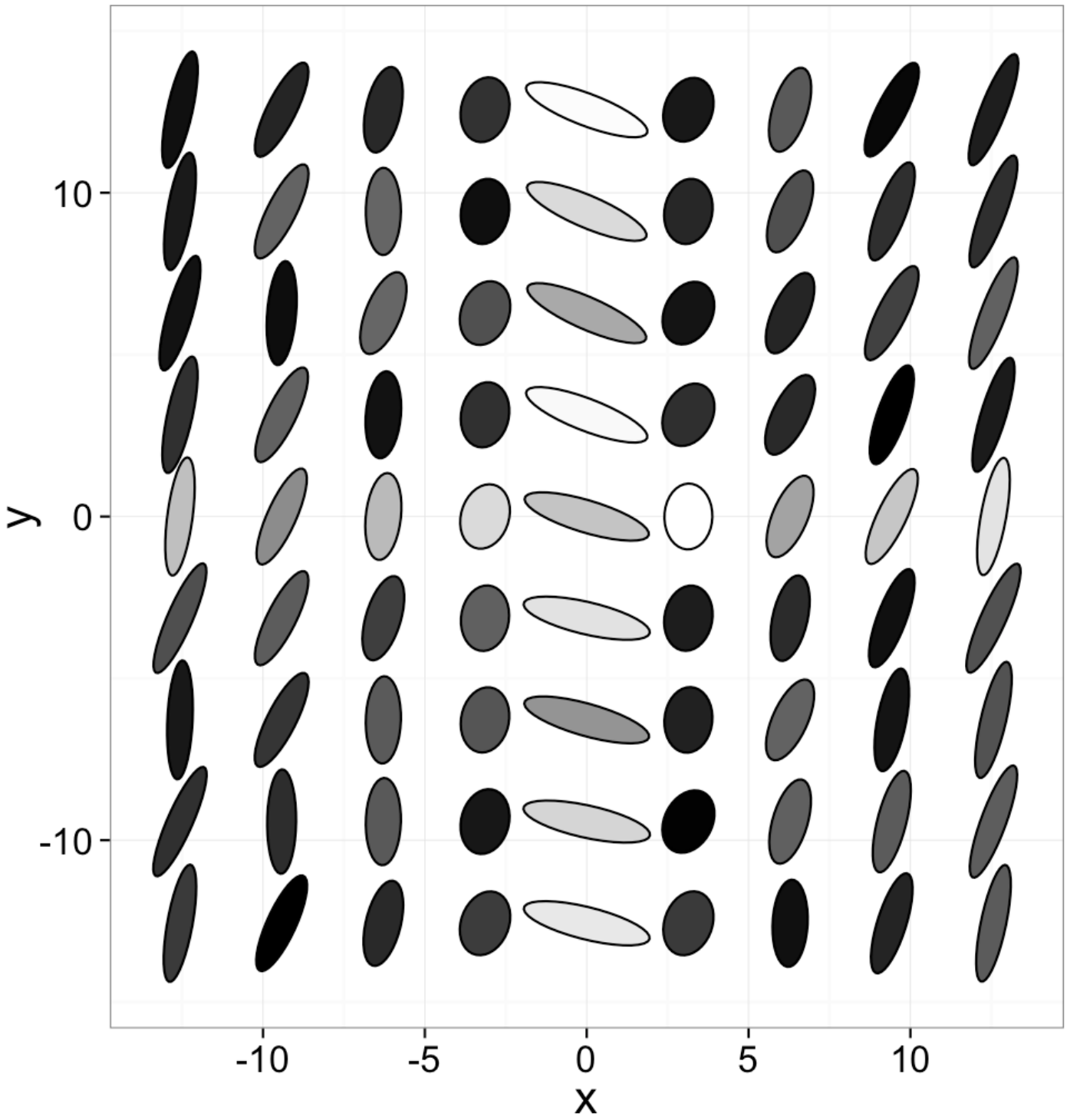
More Complicated Images

As we briefly covered last time, many measurement techniques produce quite rich data.

- Digital cameras produce 3 channels of color for each pixel (rather than just one intensity)
- MRI produces dozens of pieces of information for every voxel which are used when examining different *contrasts* in the system.
- Raman-shift imaging produces an entire spectrum for each pixel
- Coherent diffraction techniques produce 2- (sometimes 3) diffraction patterns for each point.

$$I(x,y) = \hat{f}(x,y)$$

+/- R Code



K-Means Clustering / Classification

- Automatic clustering of multidimensional data into groups based on a distance metric
- Fast and scalable to petabytes of data (Google, Facebook, Twitter, etc. use it regularly to classify customers, advertisements, queries)

- Segementation / Thresholding is really a classification task: given the gray value classify the pixel as either bone or air

K-Means Algorithm

We give as an initial parameter the number of groups we want to find and possible a criteria for removing groups that are too similar

1. Randomly create center points (groups) in vector space
2. Assigns group to data point by the “closest” center
3. Recalculate centers from mean point in each group
4. Go back to step 2 until the groups stop changing

What vector space to we have?

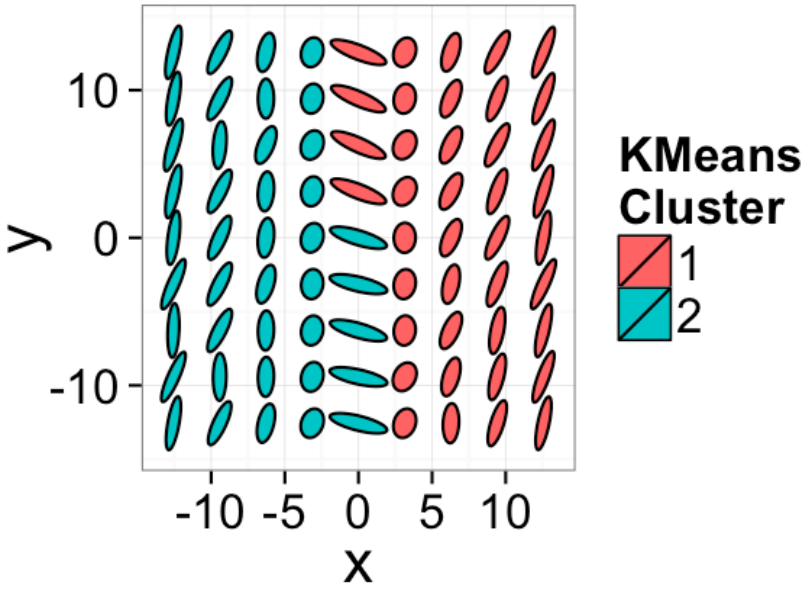
- Sometimes represent physical locations (classify swiss people into cities)
- Can include intensity or color (K-means can be used as a thresholding technique when you give it image intensity as the vector and tell it to find two or more groups)
- Can also include orientation, shape, or in extreme cases full spectra (chemically sensitive imaging)

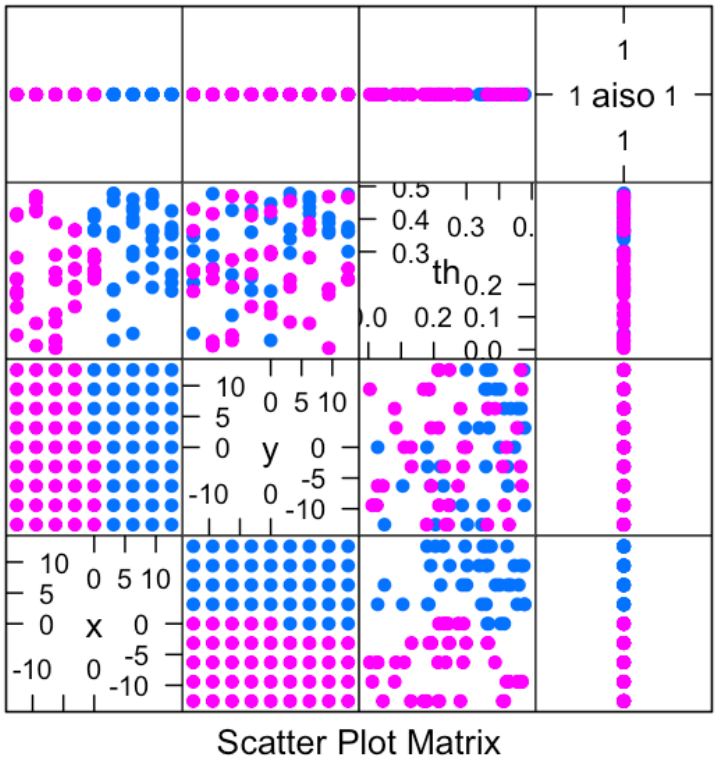
Note: If you look for 2 groups you will almost always find two groups, whether or not they make any sense

K-Means Example

```
objColor=kmeans(indata,2);
```

+/- R Code

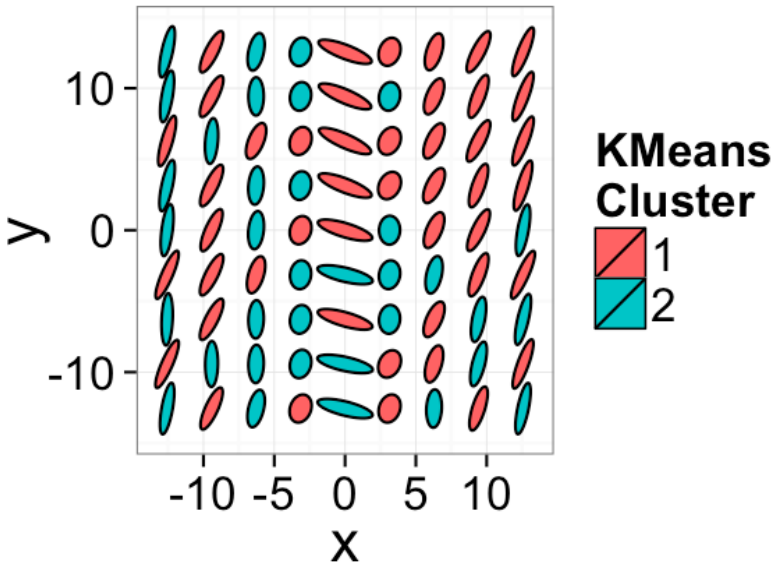




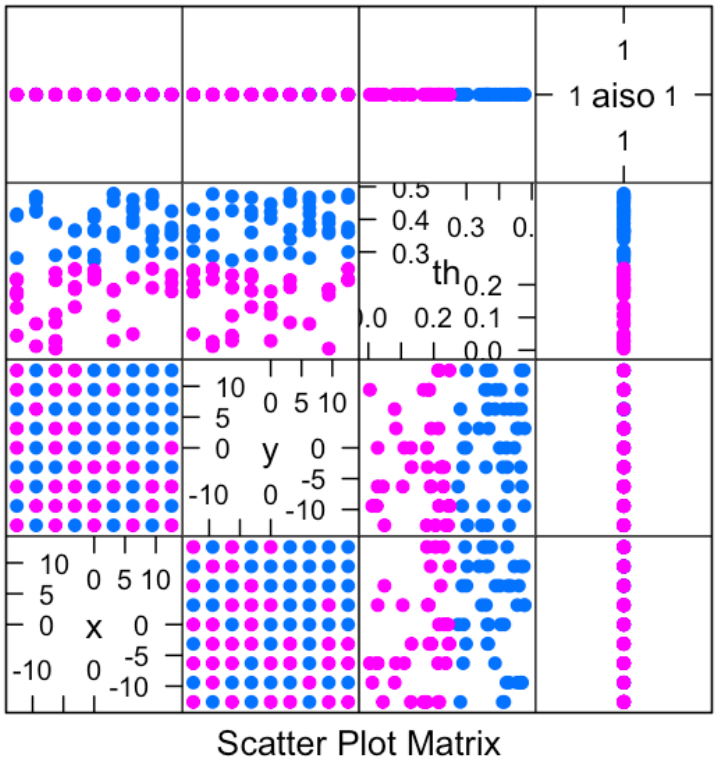
+/- R Code

Or just the orientation

```
objColor=kmeans(indata(:,[aisoCol,thCol]),2);
```



+/- R Code



+/- R Code

Probabilistic Models of Segmentation

A more general approach is to use a probabilistic model to segmentation. We start with our image $I(\vec{x}) \forall \vec{x} \in \mathbb{R}^N$ and we classify it into two phases α and β

$$P(\{\vec{x}, I(\vec{x})\}|\alpha) \propto P(\alpha) + P(I(\vec{x})|\alpha) + P(\sum_{x' \in \mathcal{N}} I(\vec{x'})|\alpha)$$

- $P(\{\vec{x}, f(\vec{x})\}|\alpha)$ the probability a given pixel is in phase α given we know it's position and value (what we are trying to estimate)
- $P(\alpha)$ probability of any pixel in an image being part of the phase (expected volume fraction of that phase)
- $P(I(\vec{x})|\alpha)$ probability adjustment based on knowing the value of I at the given point (standard threshold)
- $P(f(\vec{x'})|\alpha)$ are the collective probability adjustments based on knowing the value of a pixels neighbors (very simple version of Markov Random Field (http://en.wikipedia.org/wiki/Markov_random_field) approaches)

Fuzzy Classification

Fuzzy classification based on Fuzzy logic (http://en.wikipedia.org/wiki/Fuzzy_logic) and Fuzzy set theory (http://www.academia.edu/4978200/Applications_of_Fuzzy_Set_Theory_and_Fuzzy_Logic_in_Image_Processing) and is a general catagory for multi-value logic instead of simply **true** and **false** and can be used to build **IF** and **THEN** statements from our probabilistic models.

Instead of

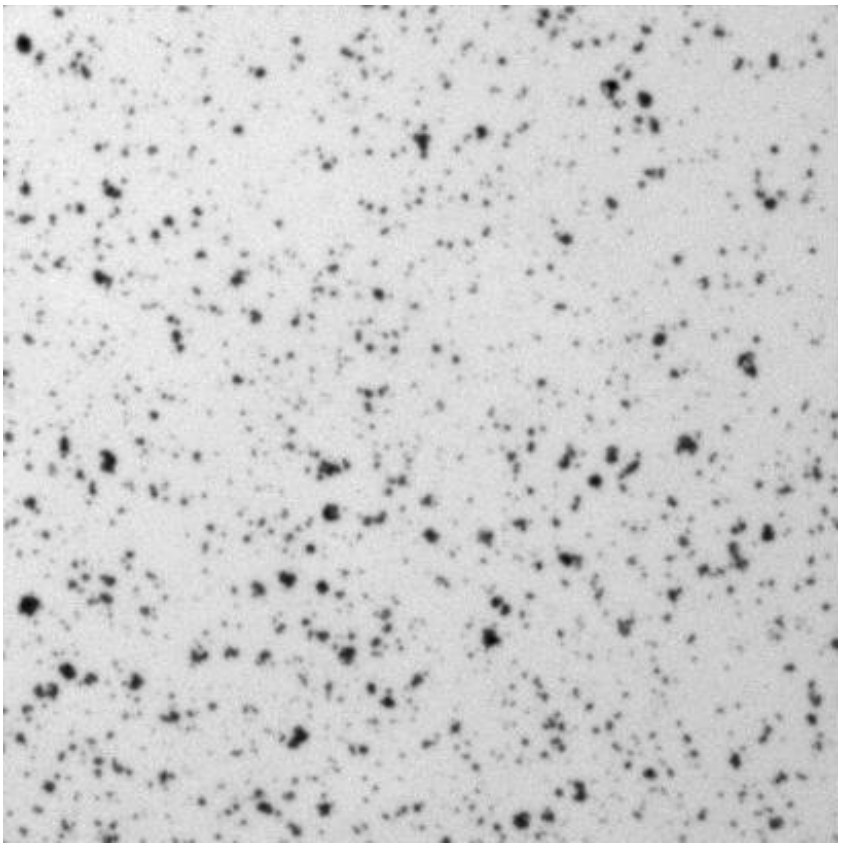
$$P(\{\vec{x}, I(\vec{x})\}|\alpha) \propto P(\alpha) + P(f(\vec{x})|\alpha) +$$

$$P(\sum_{x' \in \mathcal{N}} f(\vec{x'})|\alpha)$$

Clear simple rules

which encompass aspects of filtering, thresholding, and morphological operations

- **IF** the intensity if dark (< 100)
  - **AND** a majority of the neighborhood ($\mathcal{N}$) values are dark (< 100)
- **THEN** it is a cell



Beyond

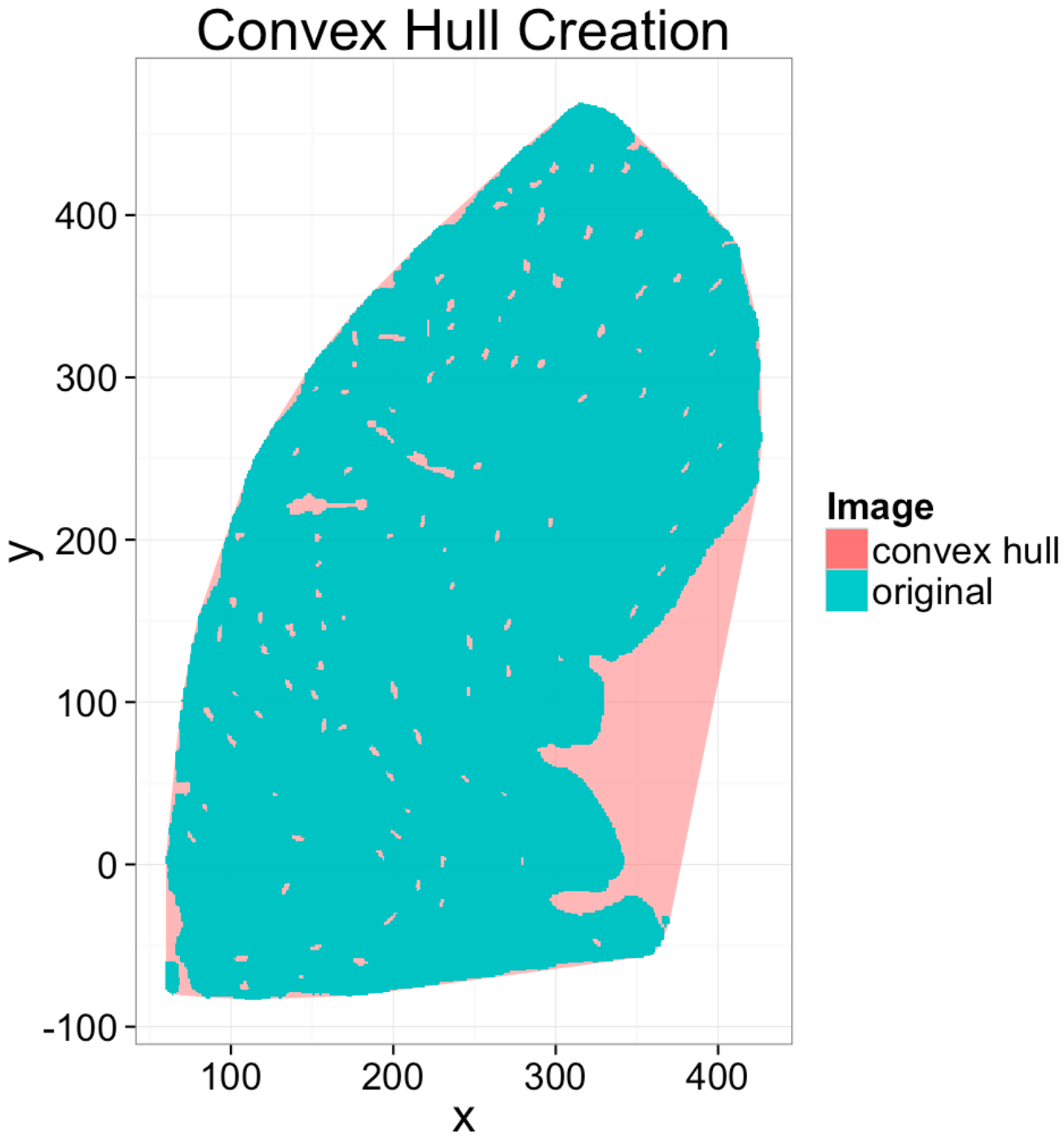
- A multitude of other techniques exist for classifying groups and courses in Data Science and Artificial Intelligence go into much greater details.
- These techniques are generally underused because they are complicated to explain and robustly test and can arouse suspicion from reviewers.
    - Because of their added complexity it is easier to manipulate these methods to get desired results from almost any dataset
    - But if the approach is based on a physical model of the images and the underyling system it is acceptable
- Additionally they usually require some degree of implementation (coding).

Contouring

Expanding on the hole filling issues examined before, a general problem in imaging is identifying regions of interest with in an image. For samples like brains it is done to identify different regions of the brain which are responsible for different functions. In material science it might be done to identify a portion of the sample being heated or understress. There are a number of approaches depending on the clarity of the data and the

Convex Hull Approach

takes all of the points in a given slice or volume and finds the smallest convex 3D volume which encloses all of those points.



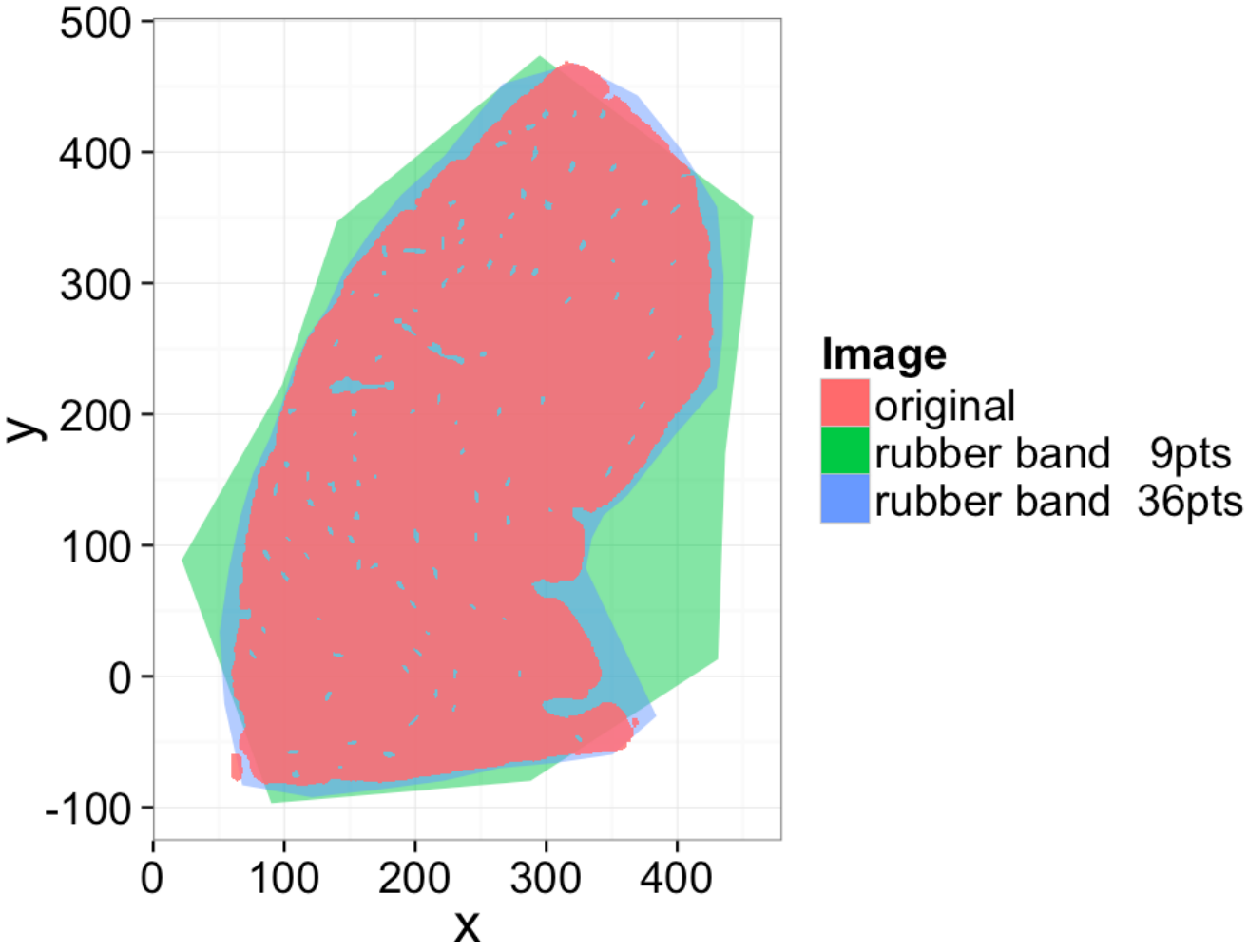
+/- R Code

Rubber Band

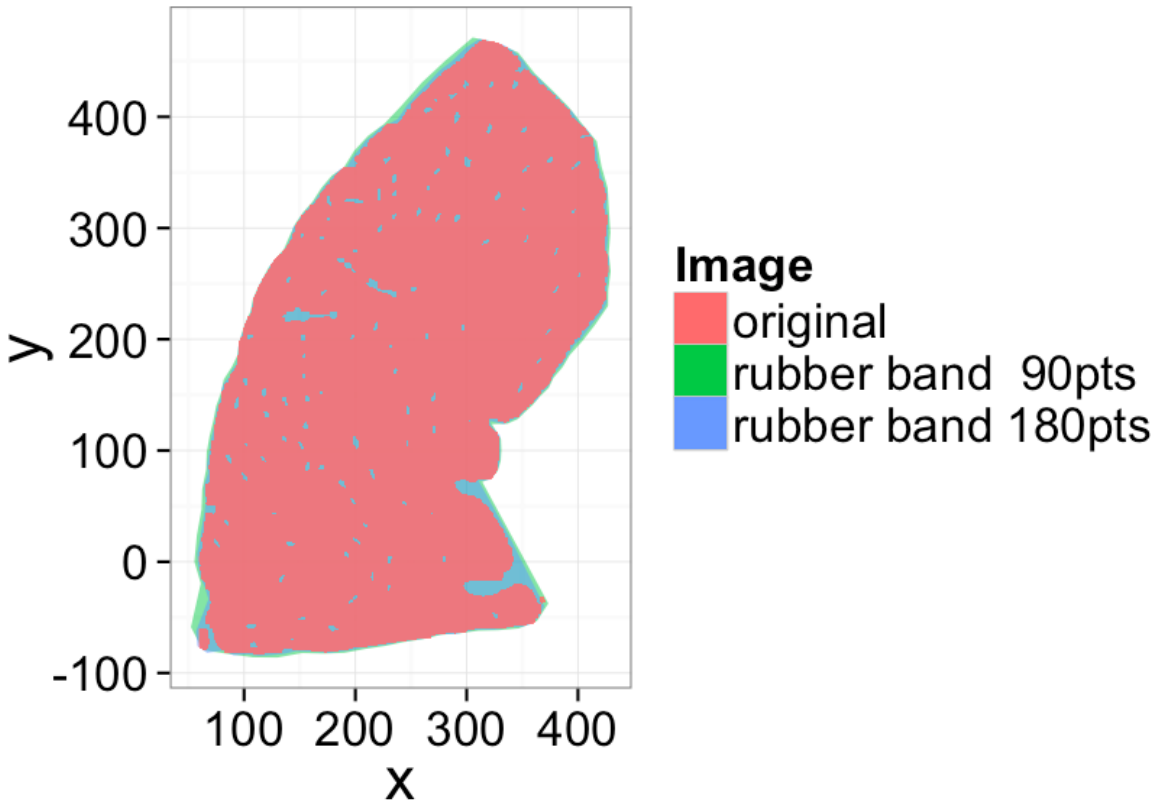
+/- R Code

Useful for a variety of samples (needn't be radially symmetric) and offers more flexibility in step size, smoothing function etc than convex hull.

1. Calculates the center of mass.
2. Transforms sample into Polar Coordinates
3. Calculates a piecewise linear fit $r = f(\theta)$

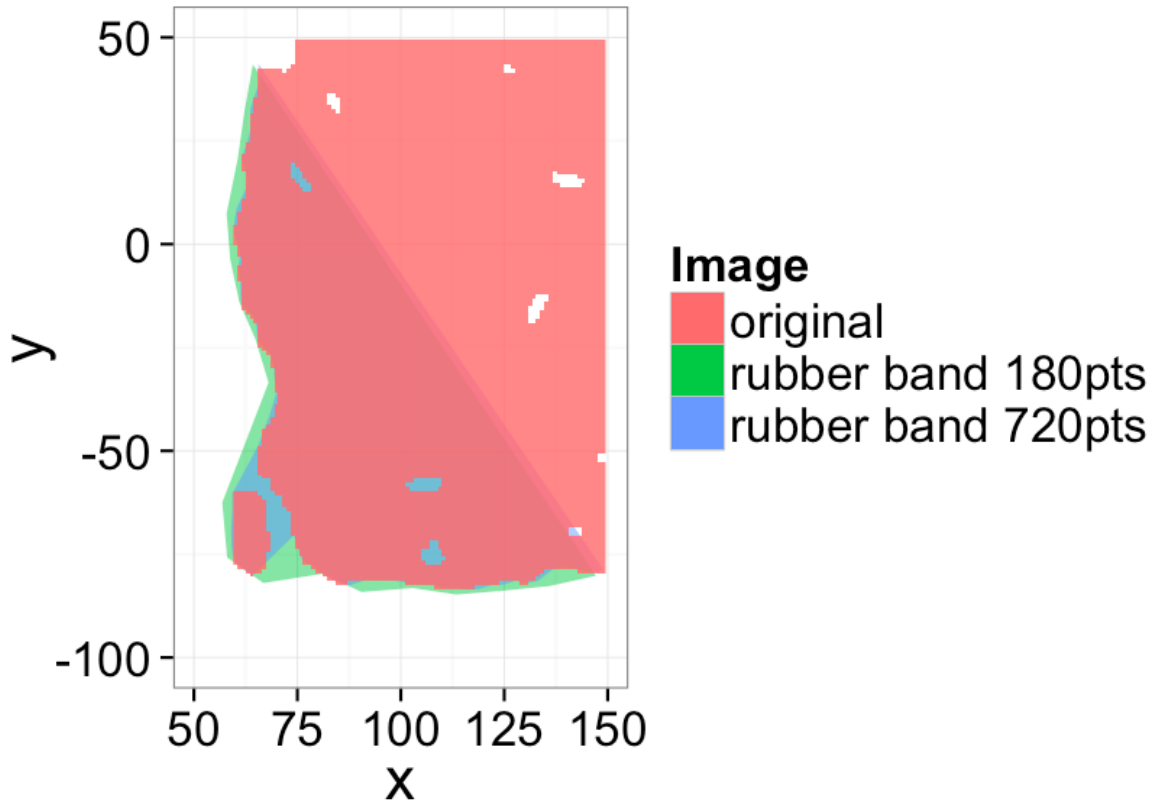


+/- R Code


400
300
200
100
0
-100
y
100
200
300
400
x
Image
original
rubber band 90pts
rubber band 180pts

+/- R Code

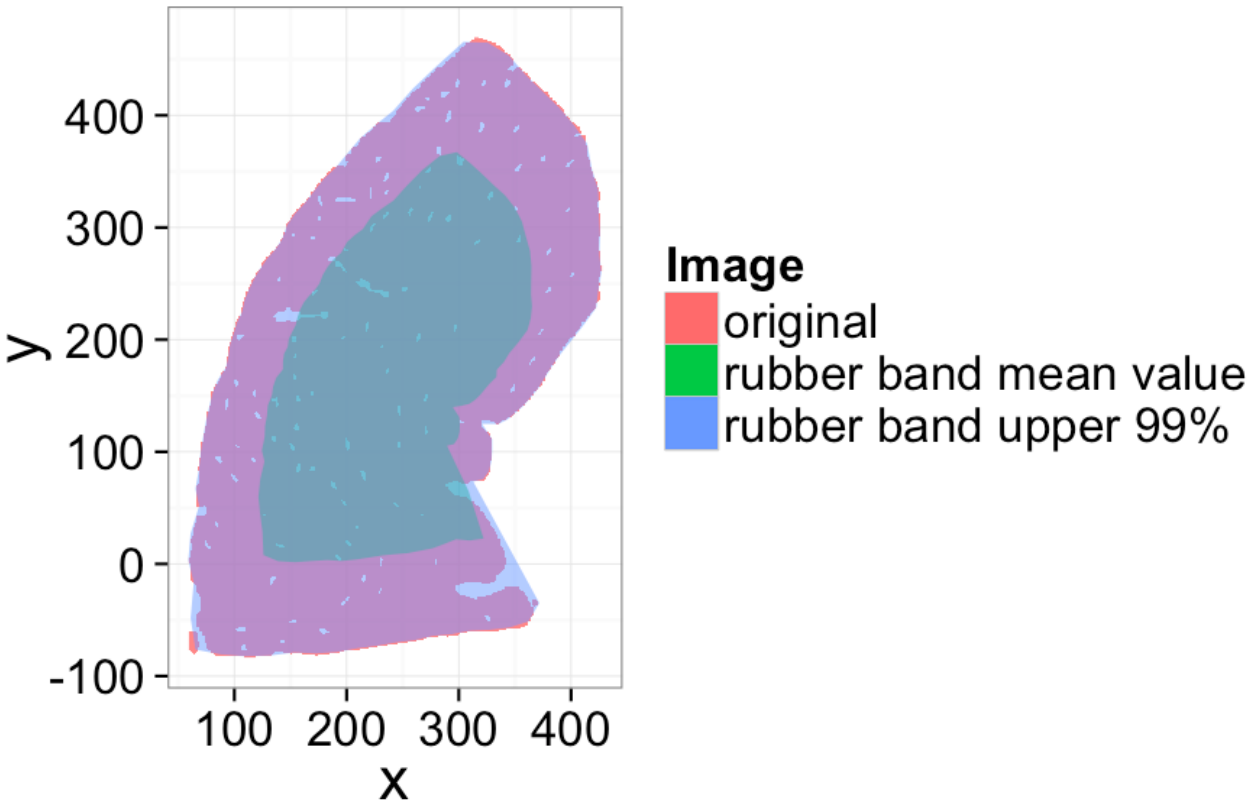
+/- R Code



Rubber Band: More flexible constraints

If we use quartiles or the average instead of the maximum value we can make the method less sensitive to outlier pixels

+/- R Code



Contouring: Guided Methods

type:alert

Many forms of guided methods exist, the most popular is known simply as the *Magnetic Lasso* in Adobe Photoshop (video (http://people.ee.ethz.ch/%7Emaderk/videos/MagneticLasso.swf)).

The basic principal behind many of these methods is to optimize a set of user given points based on local edge-like information in the image. In the brain cortex example, this is the small gradients in the gray values which our eyes naturally seperate out as an edge but which have many gaps and discontinuities.

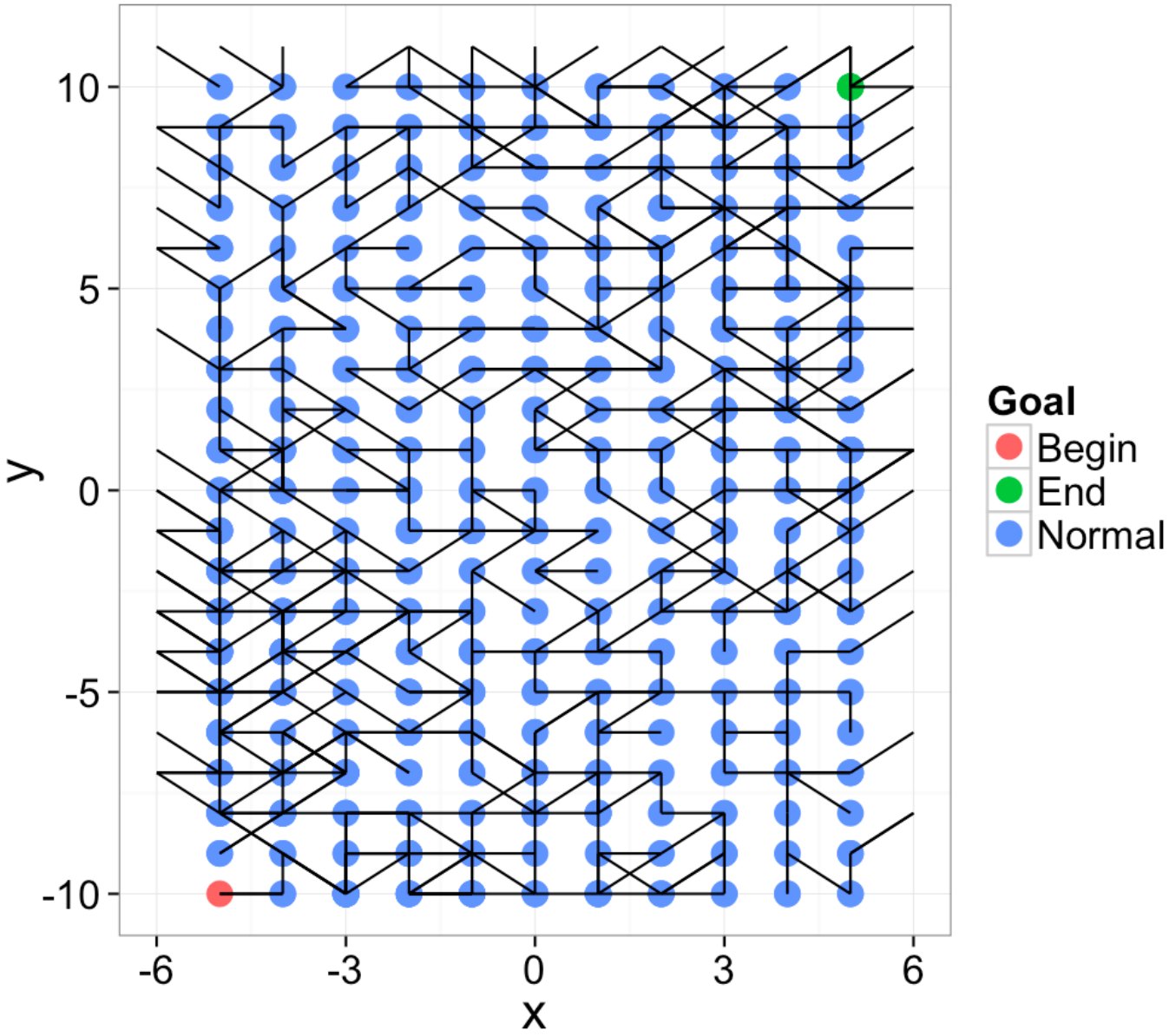
Active Contours / Snakes (http://link.springer.com/article/10.1007%2FBF00133570#page-1)

Component Labeling

Once we have a clearly segmented image, it is often helpful to identify the sub-components of this image. The easist method for identifying these subcomponents is called component labeling which again uses the neighborhood $\mathcal{N}$ as a criterion for connectivity, resulting in pixels which are touching being part of the same object.

In general, the approach works well since usually when different regions are touching, they are related. It runs into issues when you have multiple regions which agglomerate together, for example a continuous pore network (1 object) or a cluster of touching cells

The more general formulation of the problem is for networks (roads, computers, social). Are the points start and finish connected?



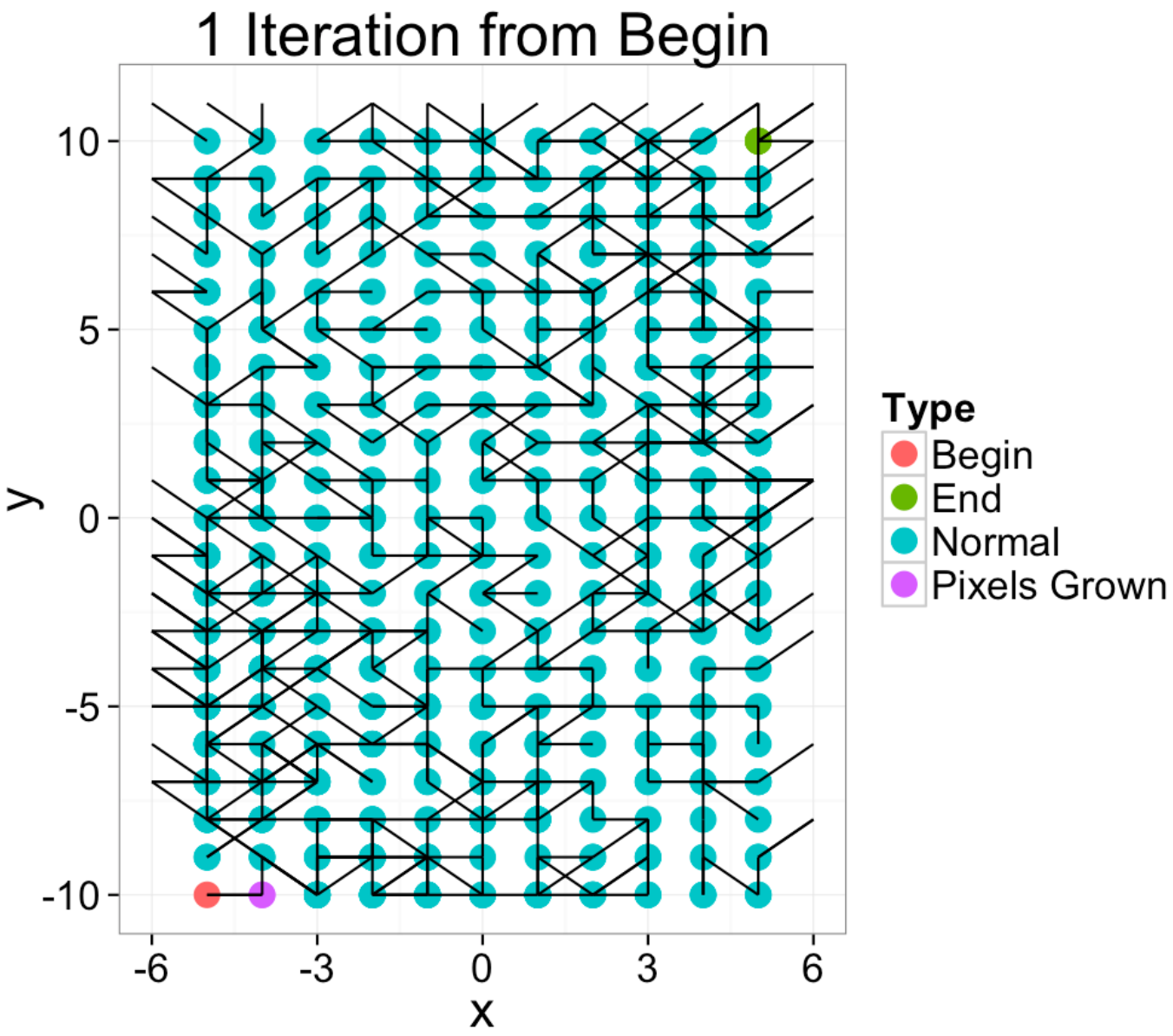
+/- R Code

Component Labeling: Algorithm

We start out with the network and we *grow* **Begin** to its connections. In a brushfire (http://www.sciencedirect.com/science/article/pii/S0921889007000966)-style algorithm

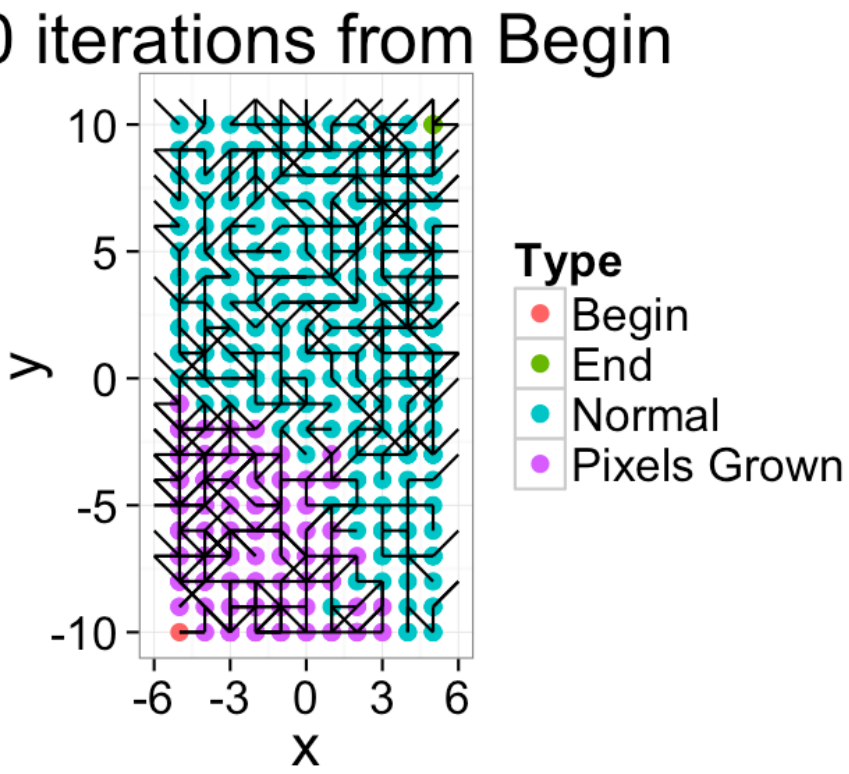
- For each point $(x, y) \in \{\text{Begin}, \text{Pixels Grown}\}$
  - For each point $(x', y') \in \mathcal{N}(x, y)$
  - Set the label to *Pixels Grown*
- Repeat until no more labels have been changed

+/- R Code



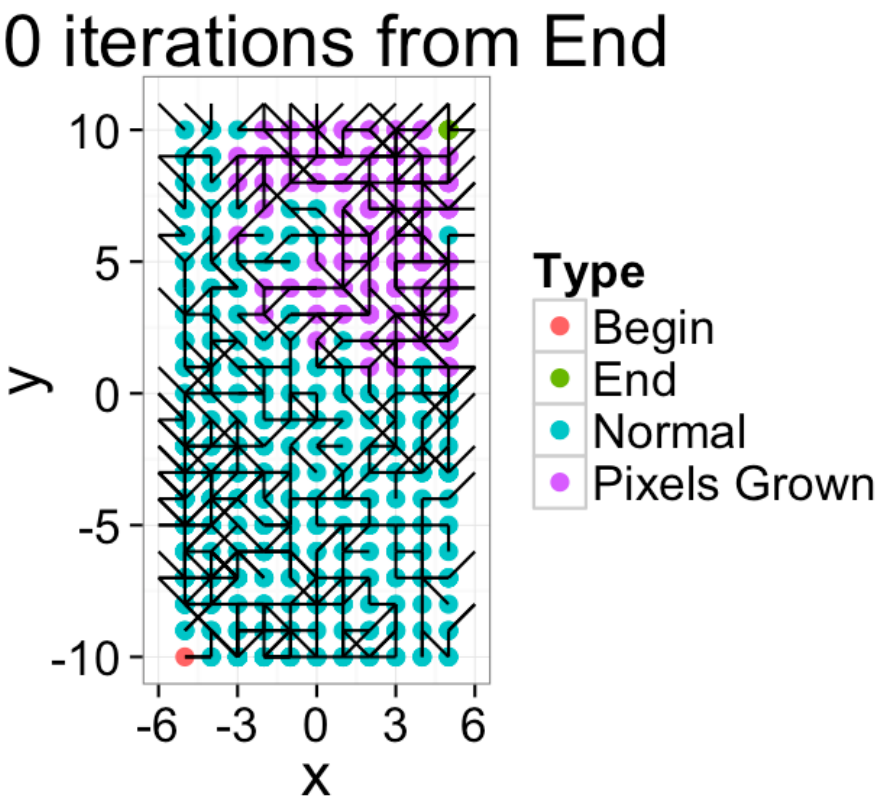
+/- R Code

Component Labeling: Algorithm Steps

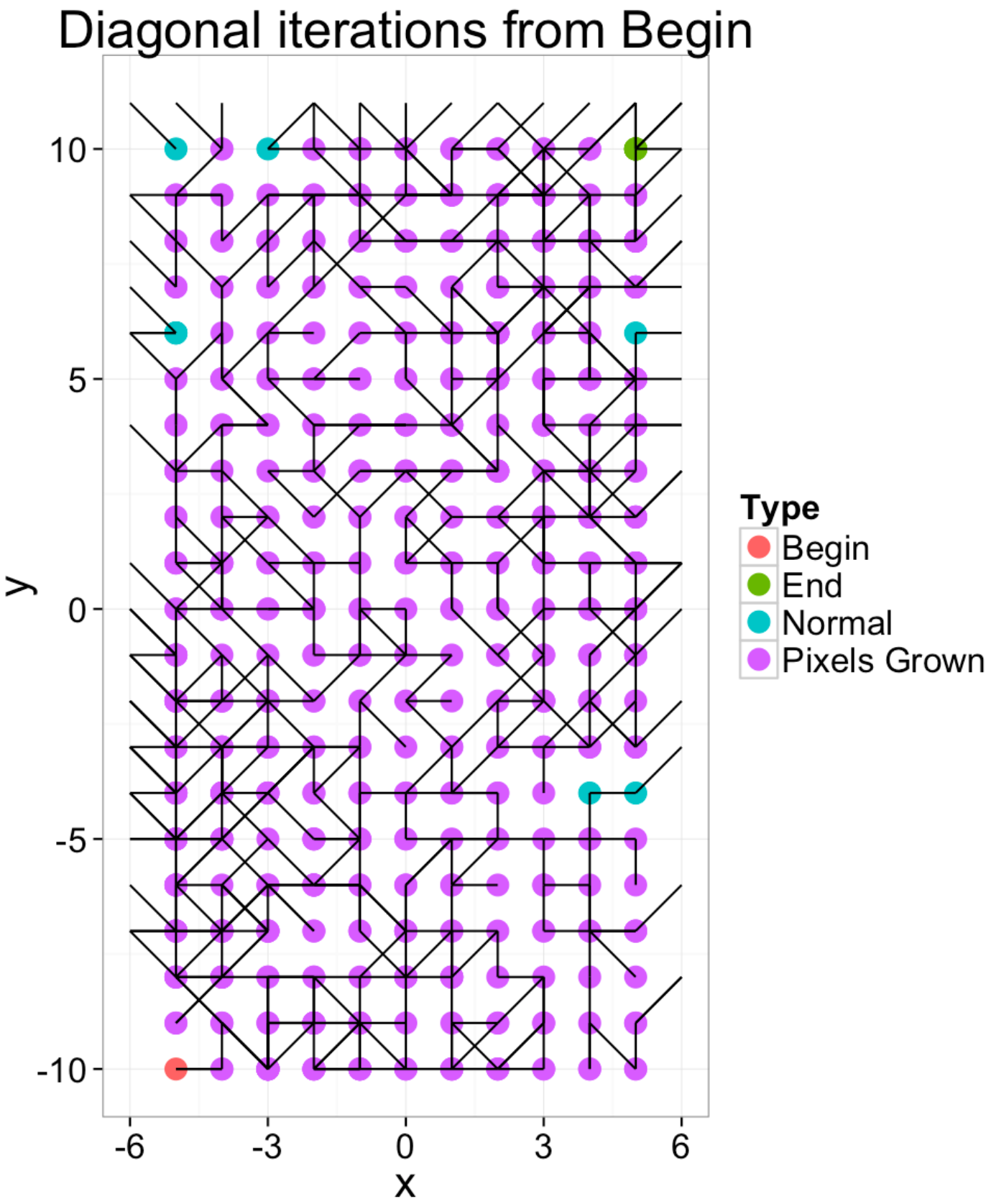


+/- R Code

+/- R Code


10 iterations from End
10
5
0
-5
-10
y
-6
-3
0
3
6
x
Type
Begin
End
Normal
Pixels Grown

+/- R Code


Diagonal iterations from Begin
10
5
0
-5
-10
y
-6
-3
0
3
6
x
Type
Begin
End
Normal
Pixels Grown

Component Labeling: Images Algorithm

Same as for networks but the neighborhood is defined with a kernel (circular, full, line, etc) and labels must be generate for the image

- Assign a unique label to each point in the image

$$L(x,y) = y * \mathrm{Im}_{width} + x$$

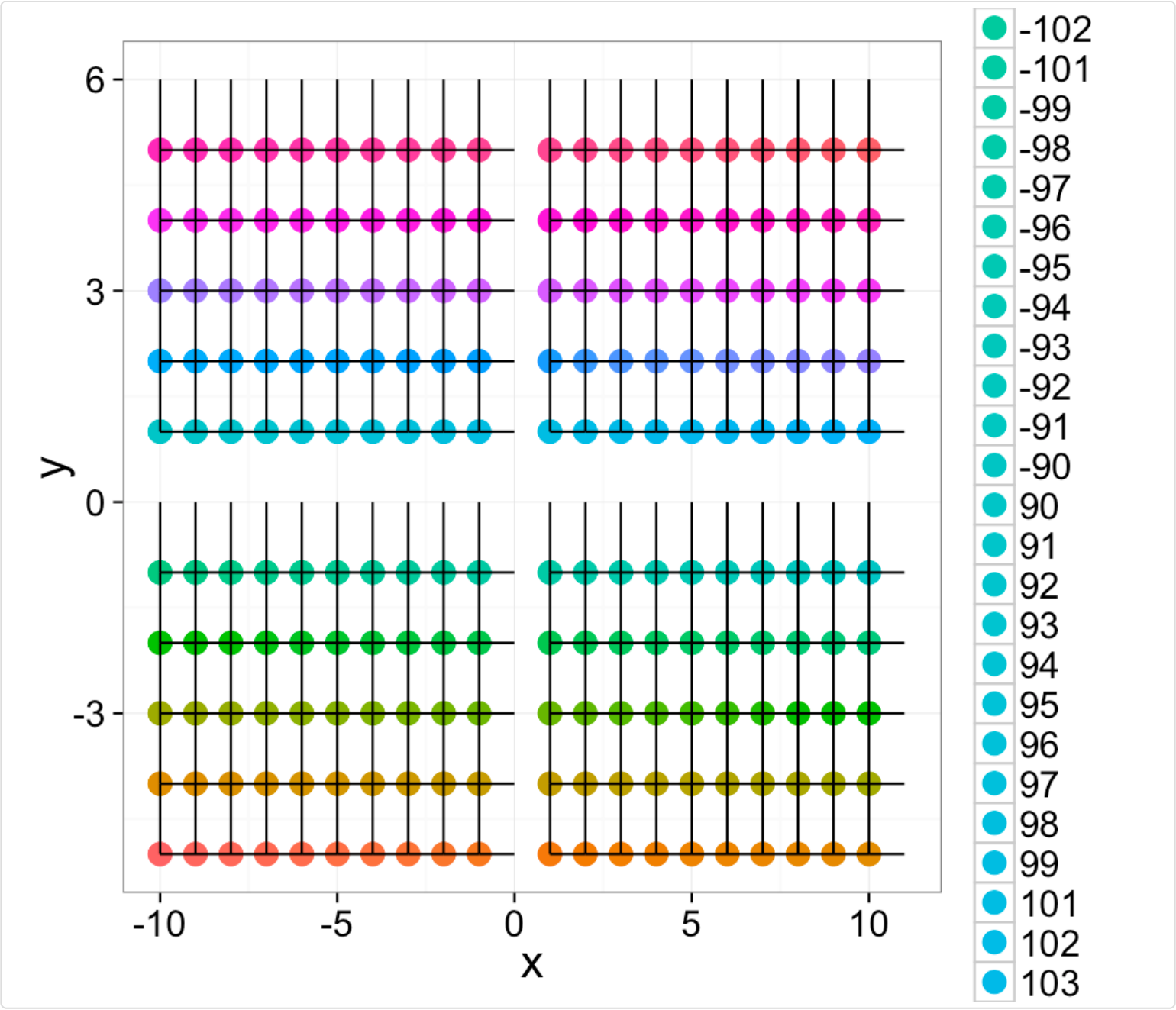
- For each point (x,y)
  - Check each point $(x',y') \in \mathcal{N}(x,y)$
  - Set

$$L(x,y) = \min(L(x,y), L(x',y'))$$

$$L(x',y') = L(x,y)$$

- Repeat until no more L values are changed

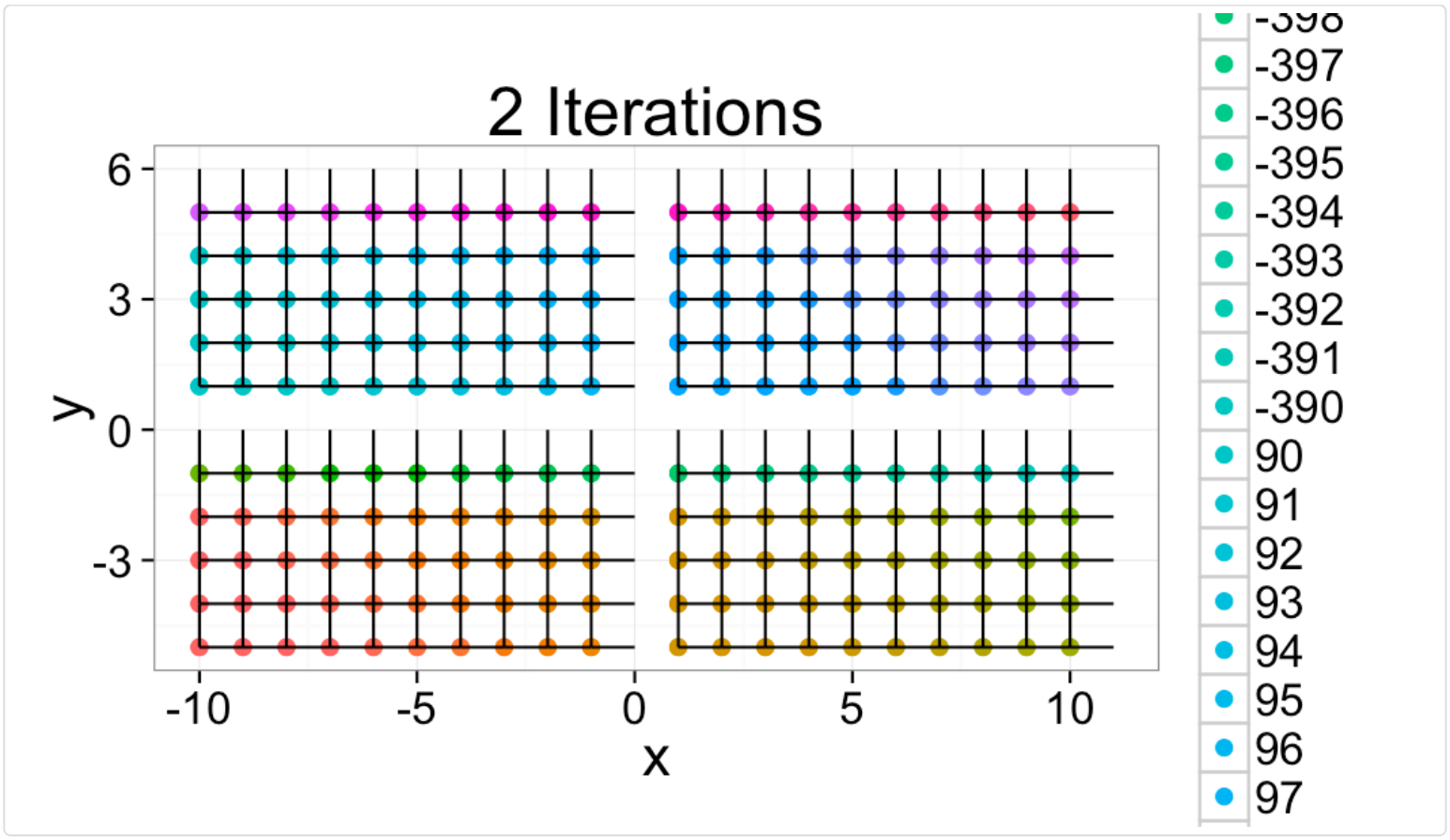
+/- R Code

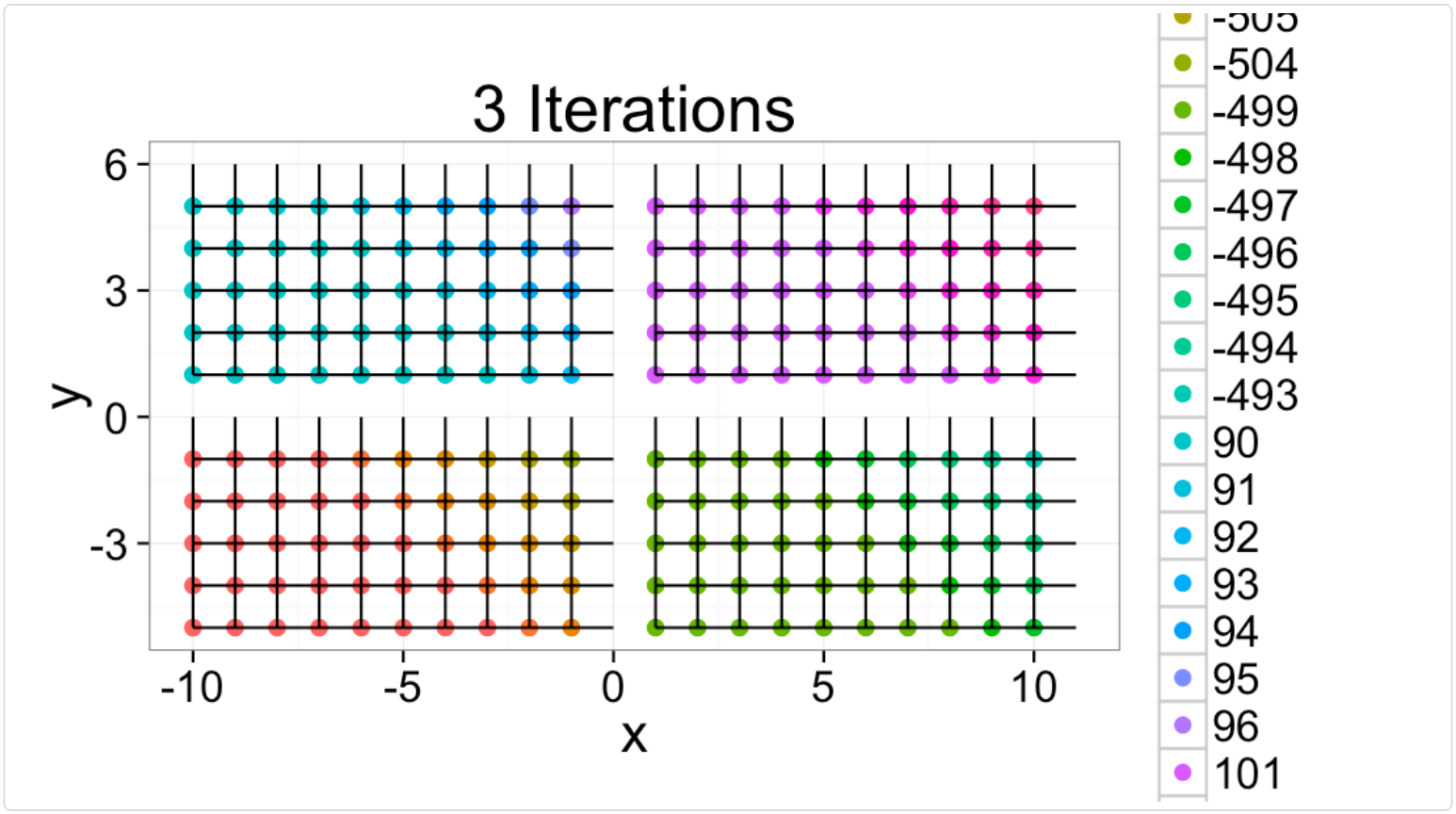


Component Labeling: Image Algorithm

+/- R Code

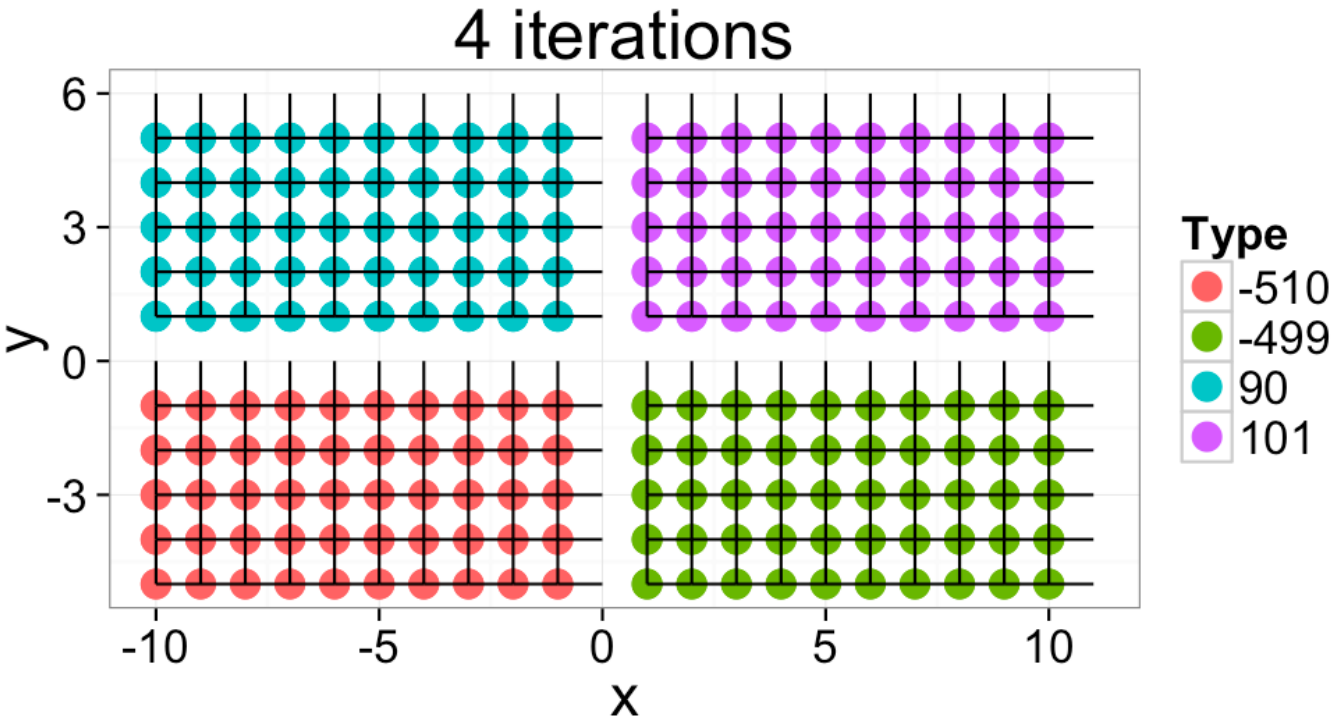
+/- R Code


2 Iterations
6
3
0
-3
y
-10
-5
0
5
10
x
-398
-397
-396
-395
-394
-393
-392
-391
-390
90
91
92
93
94
95
96
97


3 Iterations
6
3
0
-3
y
-10
-5
0
5
10
x
-504
-499
-498
-497
-496
-495
-494
-493
90
91
92
93
94
95
96
101

+/- R Code

The image very quickly converges and after 4 iterations the task is complete. For larger more complicated images with thousands of components this task can take longer, but there exist much more efficient algorithms (https://www.cs.princeton.edu/%7Ers/AlgsDS07/01UnionFind.pdf) for labeling components which alleviate this issue.



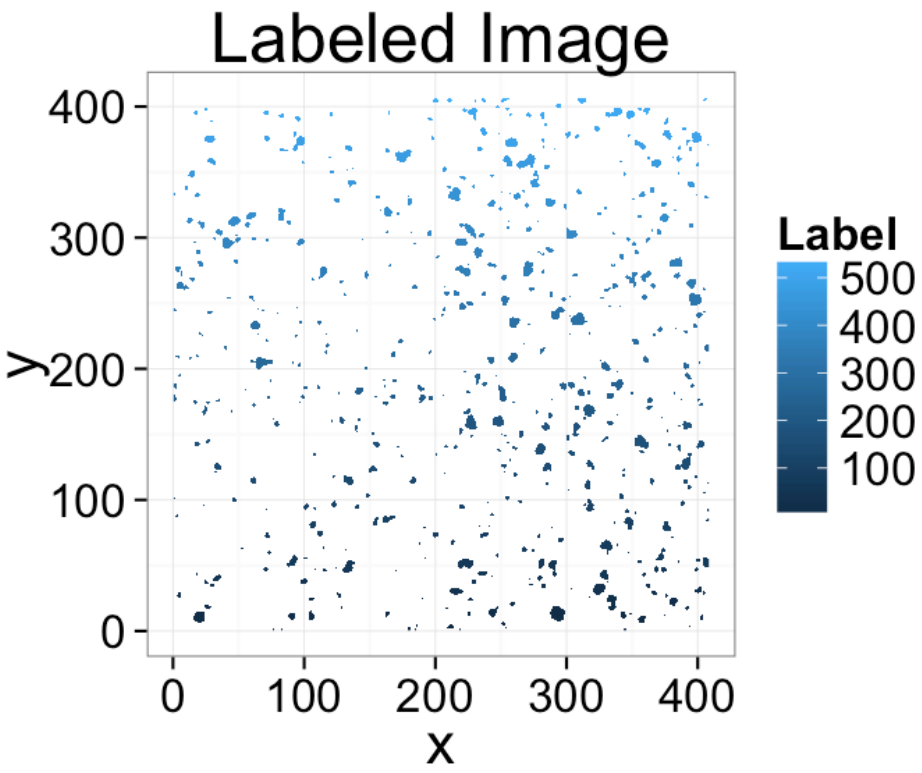
+/- R Code

Component Labeling: Image Algorithm

In reality the component labeling algorithms are usually implemented, but it is important to understand how they work and their relationship with neighborhood in order to interpret the results correctly.

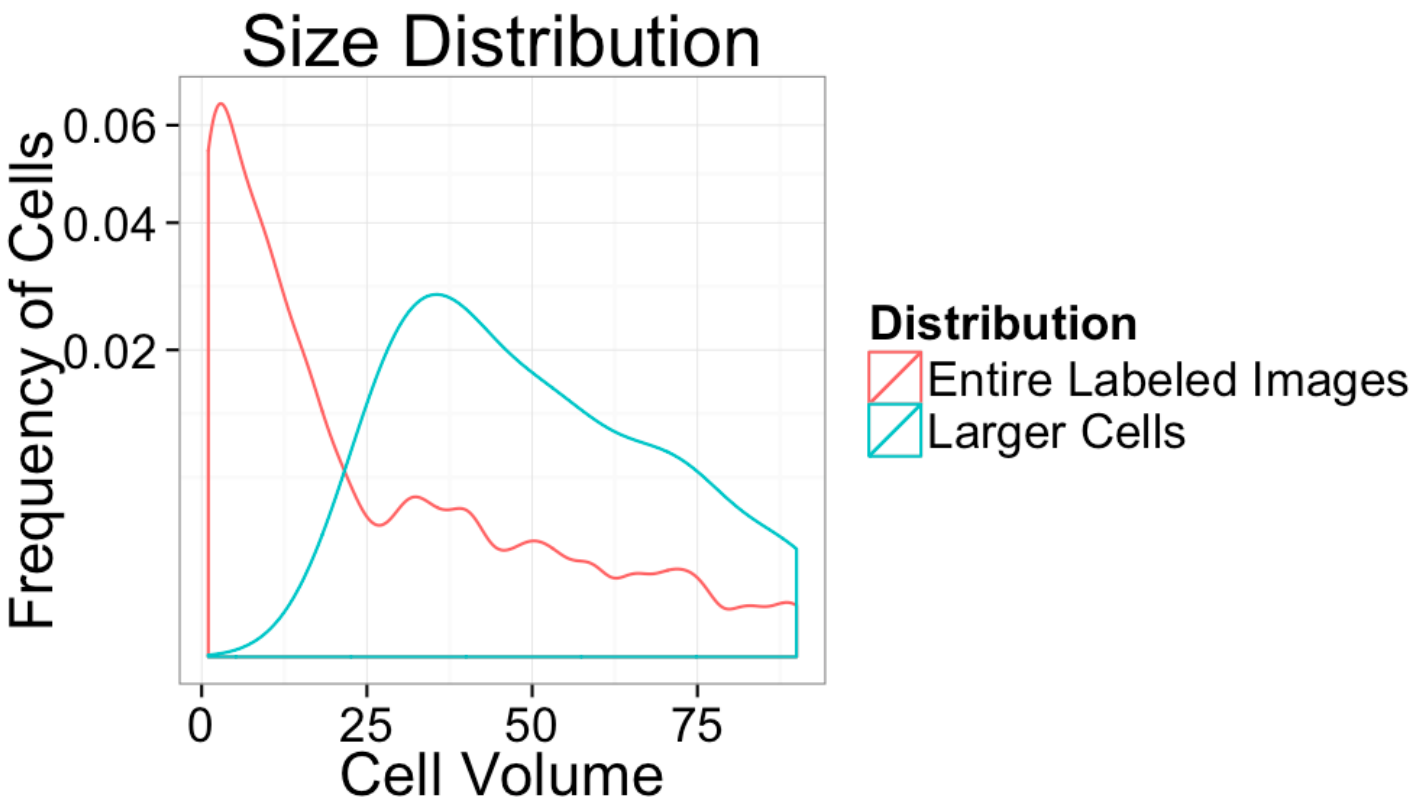
```
labImg=bwlabel(imName)
```

Component Labeling: Image Algorithm

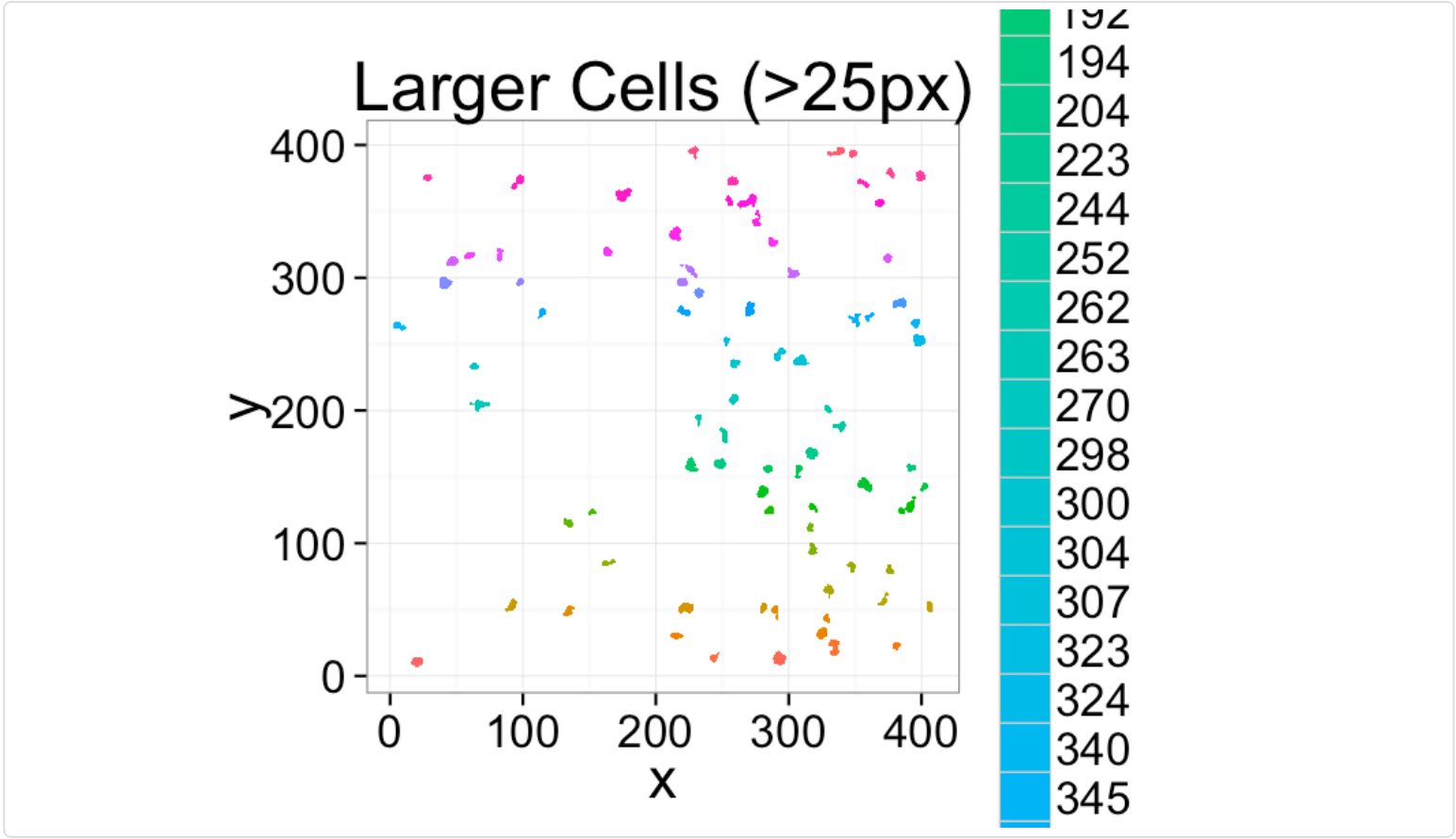


+/- R Code

+/- R Code


Size Distribution
Frequency of Cells
0.06
0.04
0.02
0
25
50
75
Cell Volume
Distribution
Entire Labeled Images
Larger Cells

+/- R Code



Component Labeling: Beyond

Now all the voxels which are connected have the same label. We can then perform simple metrics like

- counting the number of voxels in each label to estimate volume.
- looking at the change in volume during erosion or dilation to estimate surface area

Next week we will cover how more detailed analysis can be performed on these data.