+/- Message

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

+/- R Code

+/- R Code

+/- R Code

Quantitative Big Imaging

author: Kevin Mader date: 26 March 2015 width: 1440 height: 900 css: ../common/template.css transition: rotate

ETHZ: 227-0966-00L

Analysis of Single Objects

Course Outline

+/- R Code

- 19th February - Introduction and Workflows
- 26th February - Image Enhancement (A. Kaestner)
- 5th March - Basic Segmentation, Discrete Binary Structures
- 12th March - Advanced Segmentation
- 19th March - Applying Graphical Models and Machine Learning (A. Lucchi)
- 26th March - Analyzing Single Objects
- 2nd April - Analyzing Complex Objects
- 16th April - Spatial Distribution
- 23rd April - Statistics and Reproducibility
- 30th April - Dynamic Experiments (K. Mader and A. Patera)
- 7th May - Scaling Up / Big Data

- 21th May - Guest Lecture, Applications in Material Science
- 28th 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)
- Principal Component Analysis
    - Venables, W. N. and B. D. Ripley (2002). Modern Applied Statistics with S, Springer-Verlag
- Shape Tensors
    - http://www.cs.utah.edu/~gk/papers/vissym04/ (http://www.cs.utah.edu/%7Egk/papers/vissym04/)
    - Doube, M.,et al. (2010). BoneJ: Free and extensible bone image analysis in ImageJ. Bone, 47, 1076–9. doi:10.1016/j.bone.2010.08.023
    - Mader, K. , et al. (2013). A quantitative framework for the 3D characterization of the osteocyte lacunar system. Bone, 57(1), 142–154. doi:10.1016/j.bone.2013.06.026

Previously on QBI ...

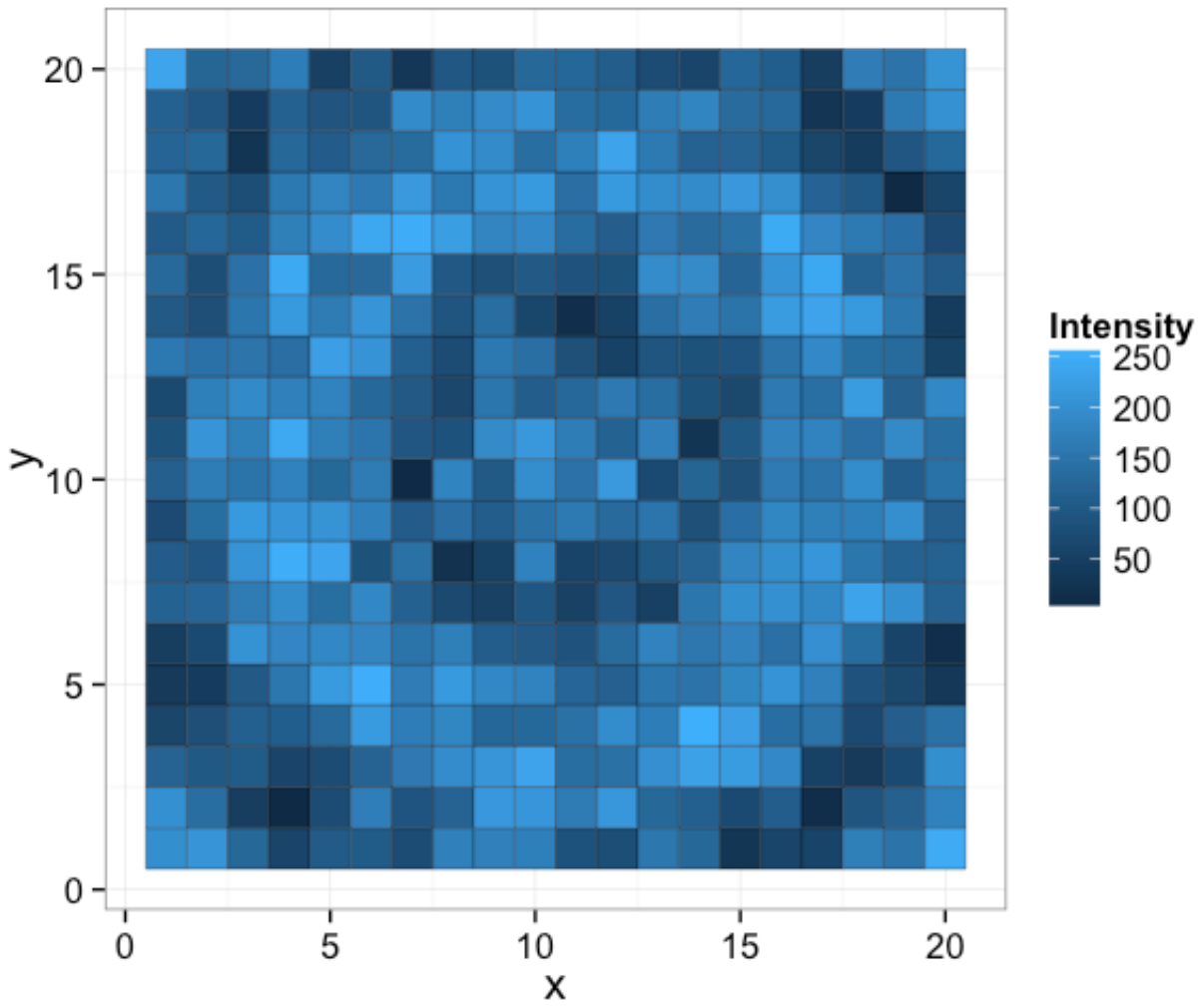
- Image Enhancment
    - Highlighting the contrast of interest in images
    - Minimizing Noise
- Segementation
    - Understanding value histograms
    - Dealing with multi-valued data
- Automatic Methods
    - Hysteresis Method, K-Means Analysis
- Regions of Interest
    - Contouring
- Machine Learning

A Machine Learning Approach to Image Processing

Segmentation and all the steps leading up to it are really a specialized type of learning problem.

+/- R Code

Returning to the ring image we had before, we want to identify the ring from the image



What does identify mean?

- Classify the pixels in the ring as *Foreground*
- Classify the pixels outside of the ring as *Background*

How do we quantify this?

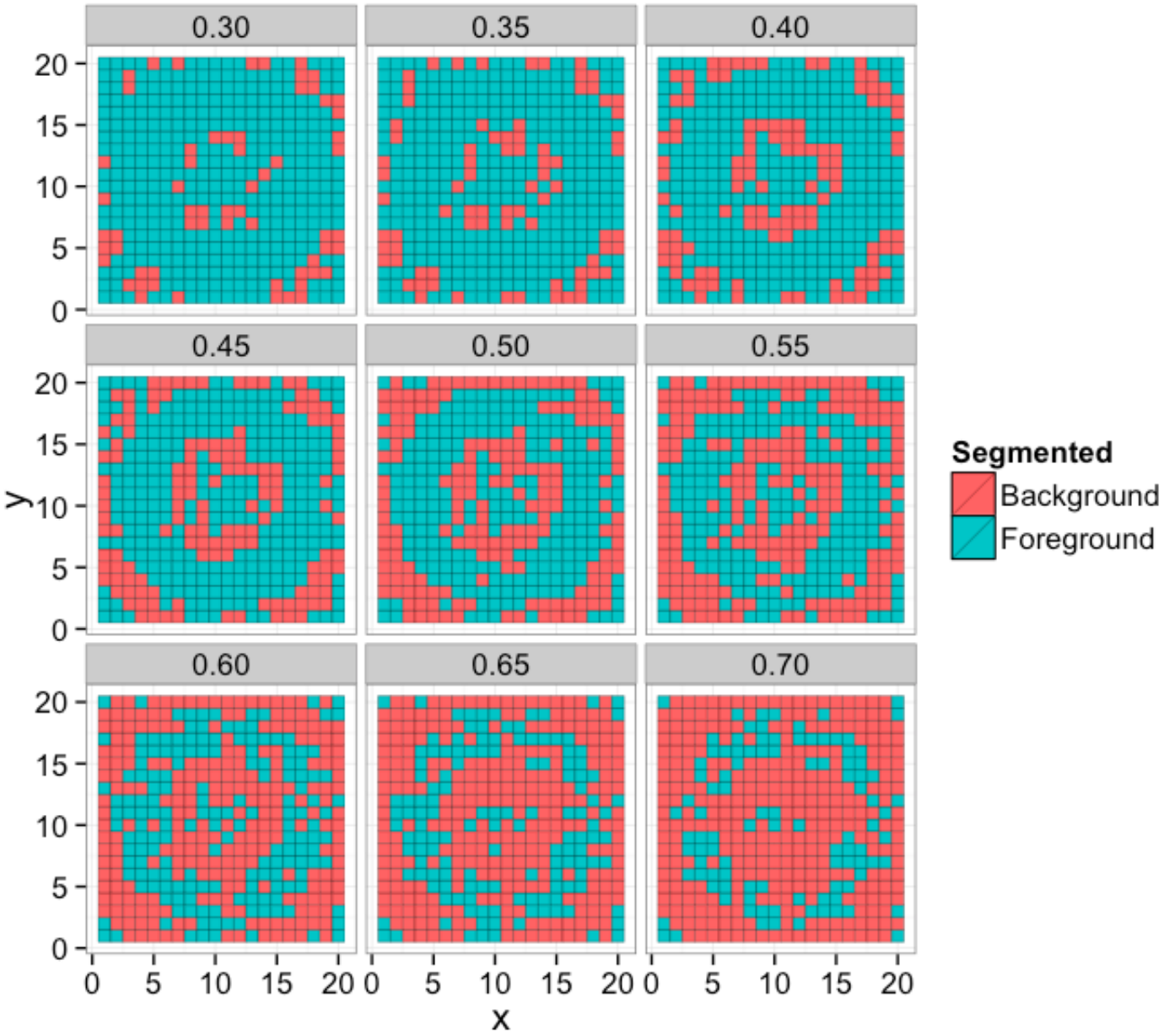
- **True Positive** values in the ring that are classified as *Foreground*
- **True Negative** values outside the ring that are classified as *Background*
- **False Positive** values outside the ring that are classified as *Foreground*
- **False Negative** values in the ring that are classified as *Background*

Ring Threshold Example

Try a number of different threshold values on the image and compare them to the original classification

+/- R Code

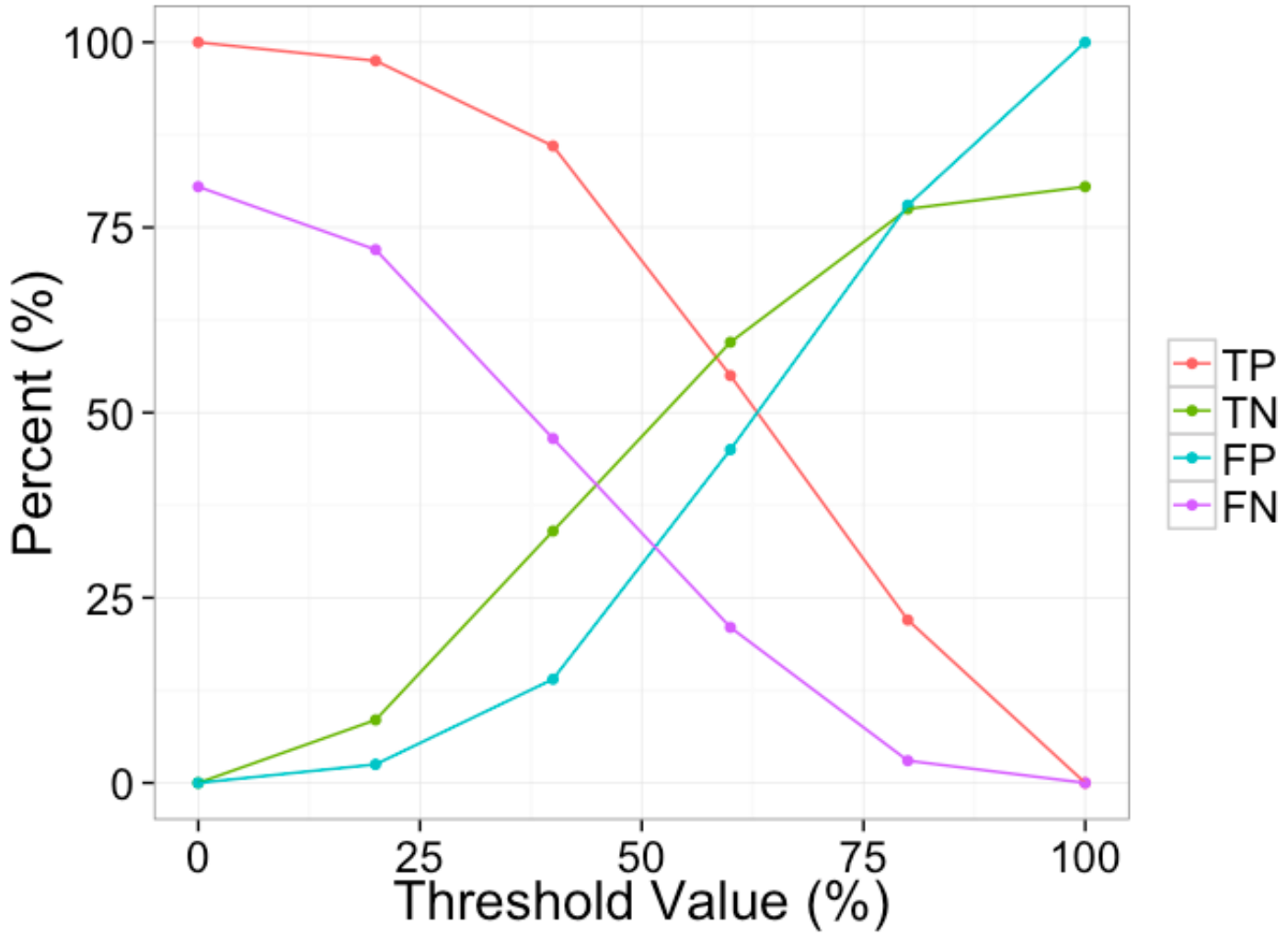
+/- R Code



+/- R Code

Thresh	TP	TN	FP	FN

0.0	200	0	0	161
0.2	195	17	5	144
0.4	172	68	28	93
0.6	110	119	90	42
0.8	44	155	156	6
1.0	0	161	200	0



Apply Precision and Recall

- **Recall** (sensitivity)= $TP/(TP+FN)$
- **Precision** = $TP/(TP+FP)$

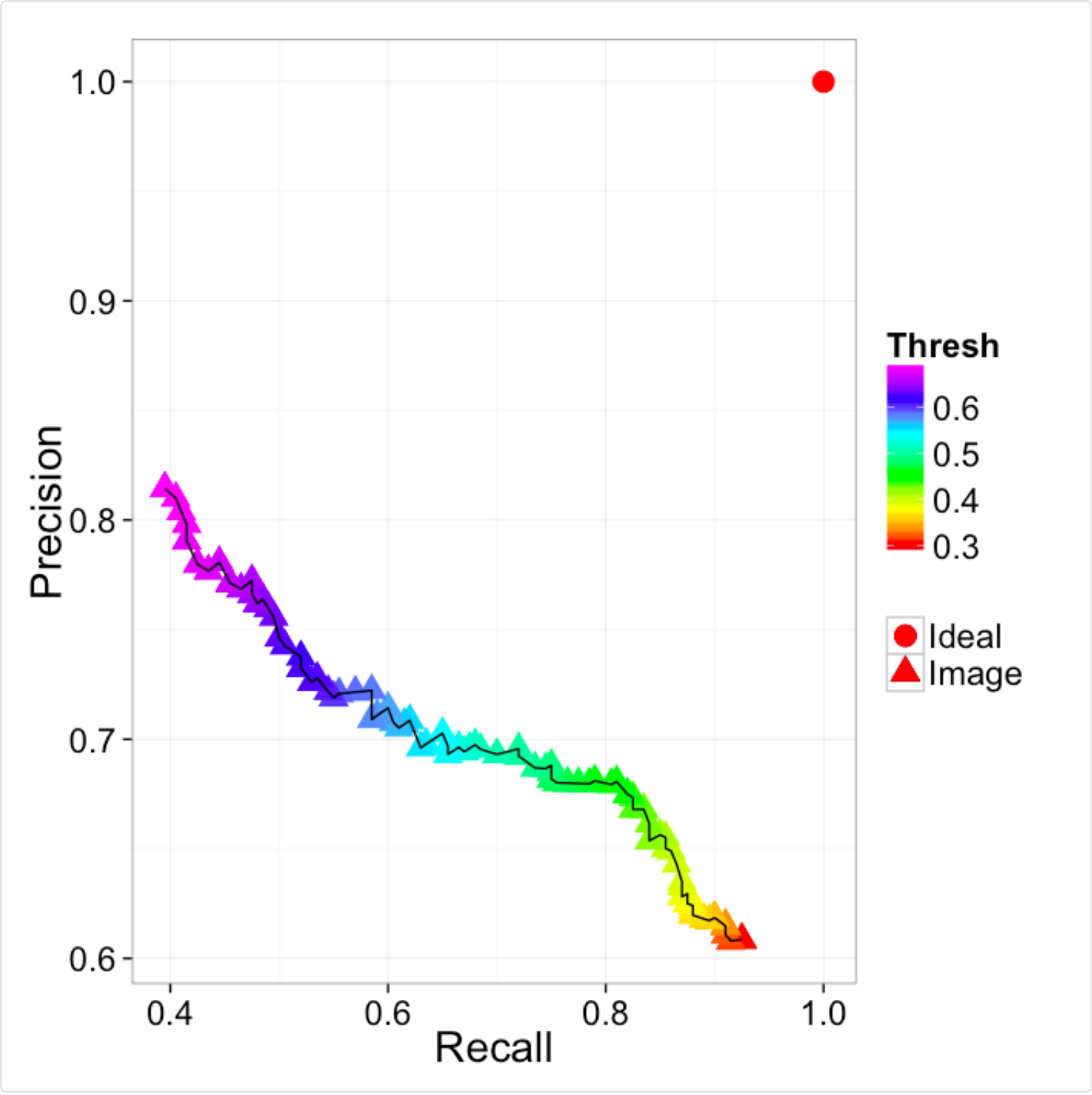
+/- R Code

Thresh	TP	TN	FP	FN	Precision	Recall
0.30	185	42	15	119	0.6085526	0.925
0.38	175	58	25	103	0.6294964	0.875
0.46	157	87	43	74	0.6796537	0.785
0.54	130	104	70	57	0.6951872	0.650
0.62	102	125	98	36	0.7391304	0.510
0.70	79	143	121	18	0.8144330	0.395

ROC Curve

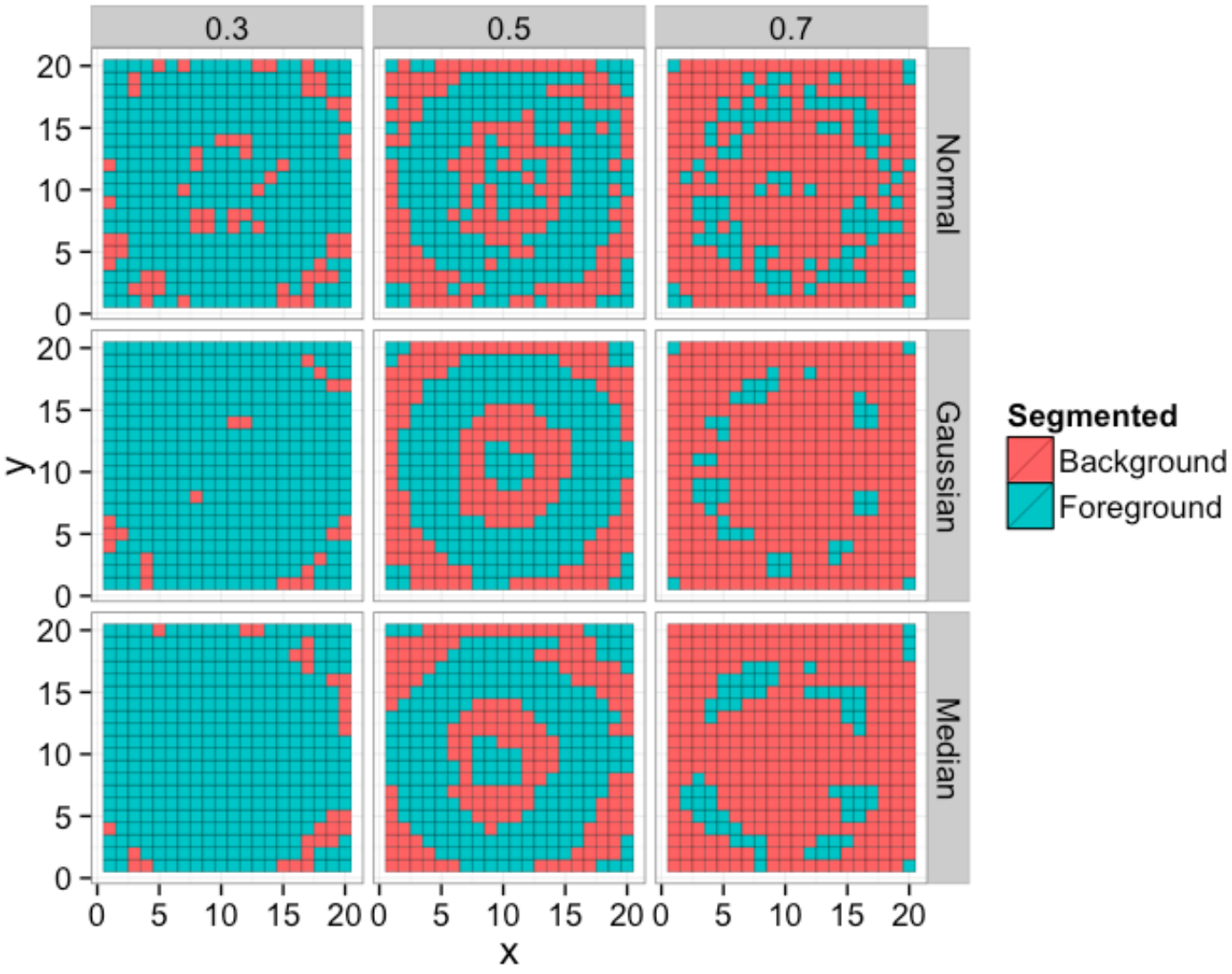
Reciever Operating Characteristic (first developed for WW2 soldiers detecting objects in battlefields using radar). The ideal is the top-right (identify everything and miss nothing)

+/- R Code



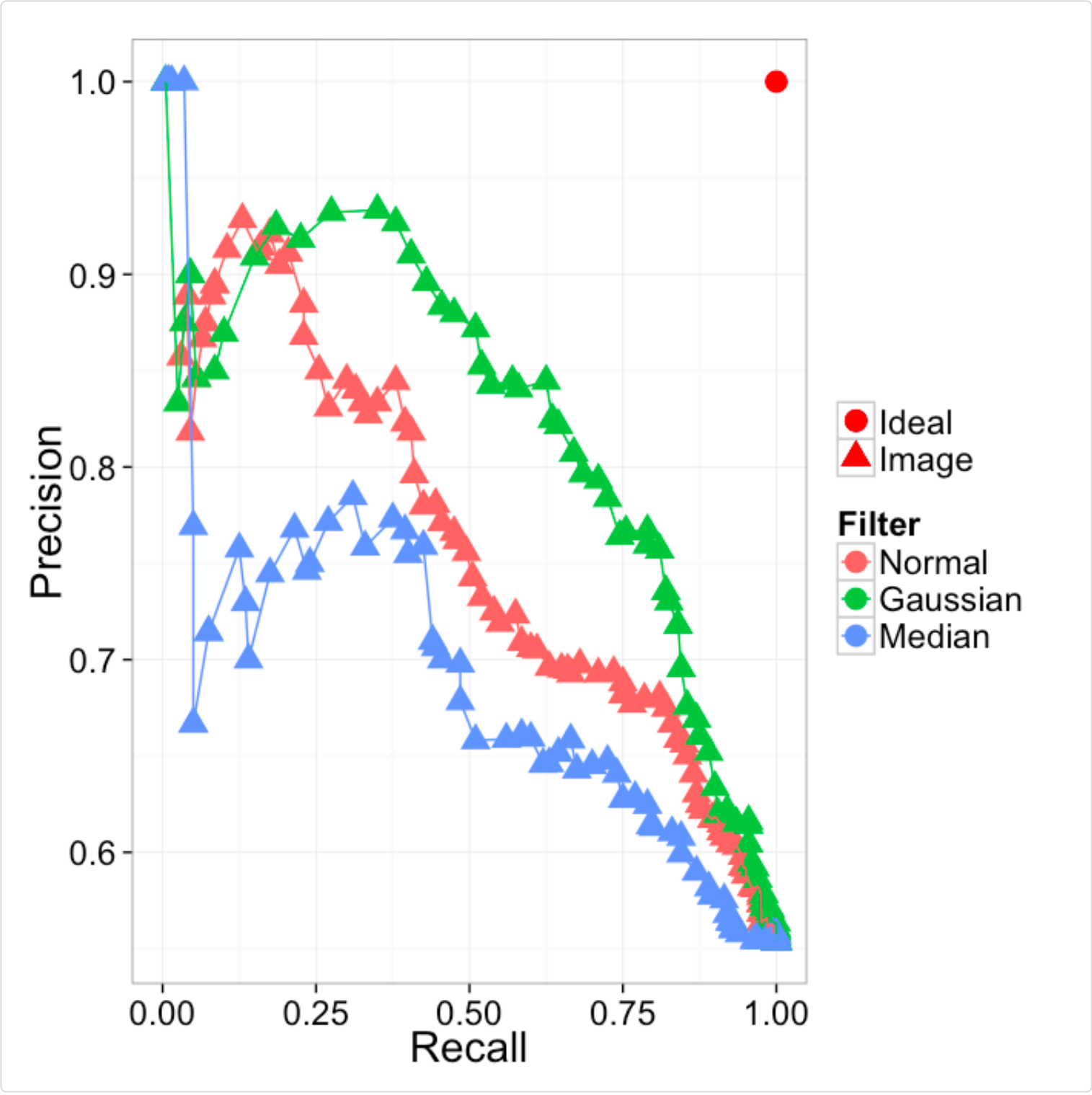
Comparing Different Filters

We can then use this ROC curve to compare different filters (or even entire workflows), if the area is higher the approach is better.


0.3
0.5
0.7
Normal
Gaussian
Median
y
x
0
5
10
15
20
Segmented
Background
Foreground

Different approaches can be compared by area under the curve

+/- R Code



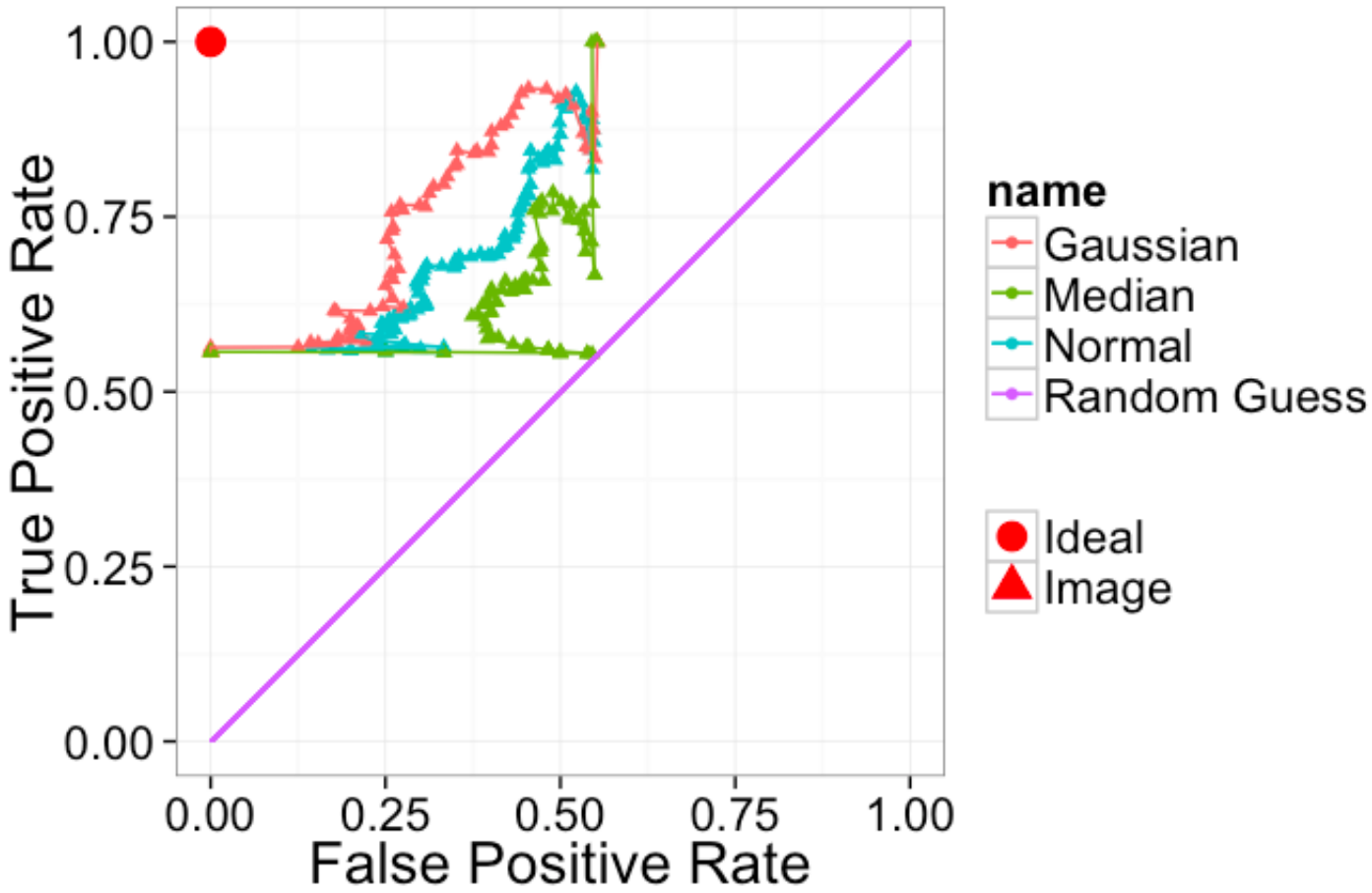
True Positive Rate and False Positive Rate

Another way of showing the ROC curve (more common for machine learning rather than medical diagnosis) is using the True positive rate and False positive rate

- **True Positive Rate** (recall)= $TP/(TP+FN)$

- **False Positive Rate** = $FP/(FP + TN)$

These show very similar information with the major difference being the goal is to be in the upper left-hand corner. Additionally random guesses can be shown as the slope 1 line. Therefore for a system to be useful it must lie above the random line.



Learning Objectives

Motivation (Why and How?)

- How do we quantify where and how big our objects are?
- How can we say something about the shape?
- How can we compare objects of different sizes?
- How can we compare two images on the basis of the shape as calculated from the images?
- How can we put objects into an finite element simulation? or make pretty renderings?

Outline

- Motivation (Why and How?)
- Object Characterization
- Volume
- Center and Extents
- Anisotropy

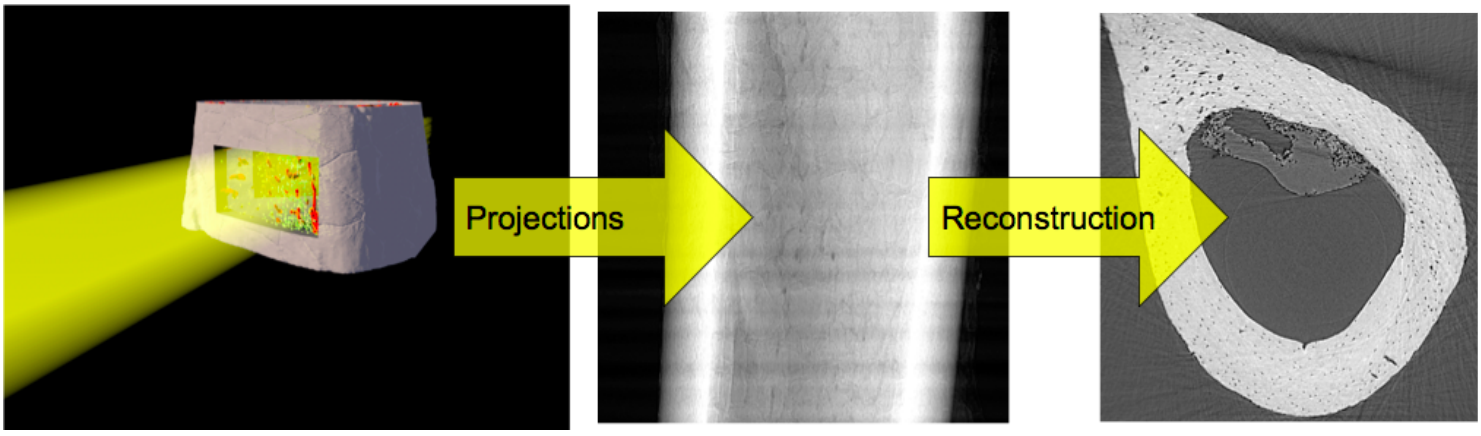
- Shape Tensor
- Principal Component Analysis
- Ellipsoid Representation
- Scale-free metrics
- Anisotropy, Oblateness
- Meshing
    - Marching Cubes
    - Isosurfaces
- Surface Area

Motivation

We have dramatically simplified our data, but there is still too much.

- We perform an experiment bone to see how big the cells are inside the tissue

↓



2560 x 2560 x 2160 x 32 bit

56GB / sample

- Filtering and Enhancement!

↓

- 56GB of less noisy data

- **Segmentation**

↓

2560 x 2560 x 2160 x 1 bit

(1.75GB / sample)

- Still an aweful lot of data

What did we want in the first place

Single number:

- volume fraction,
- cell count,
- average cell stretch,
- cell volume variability

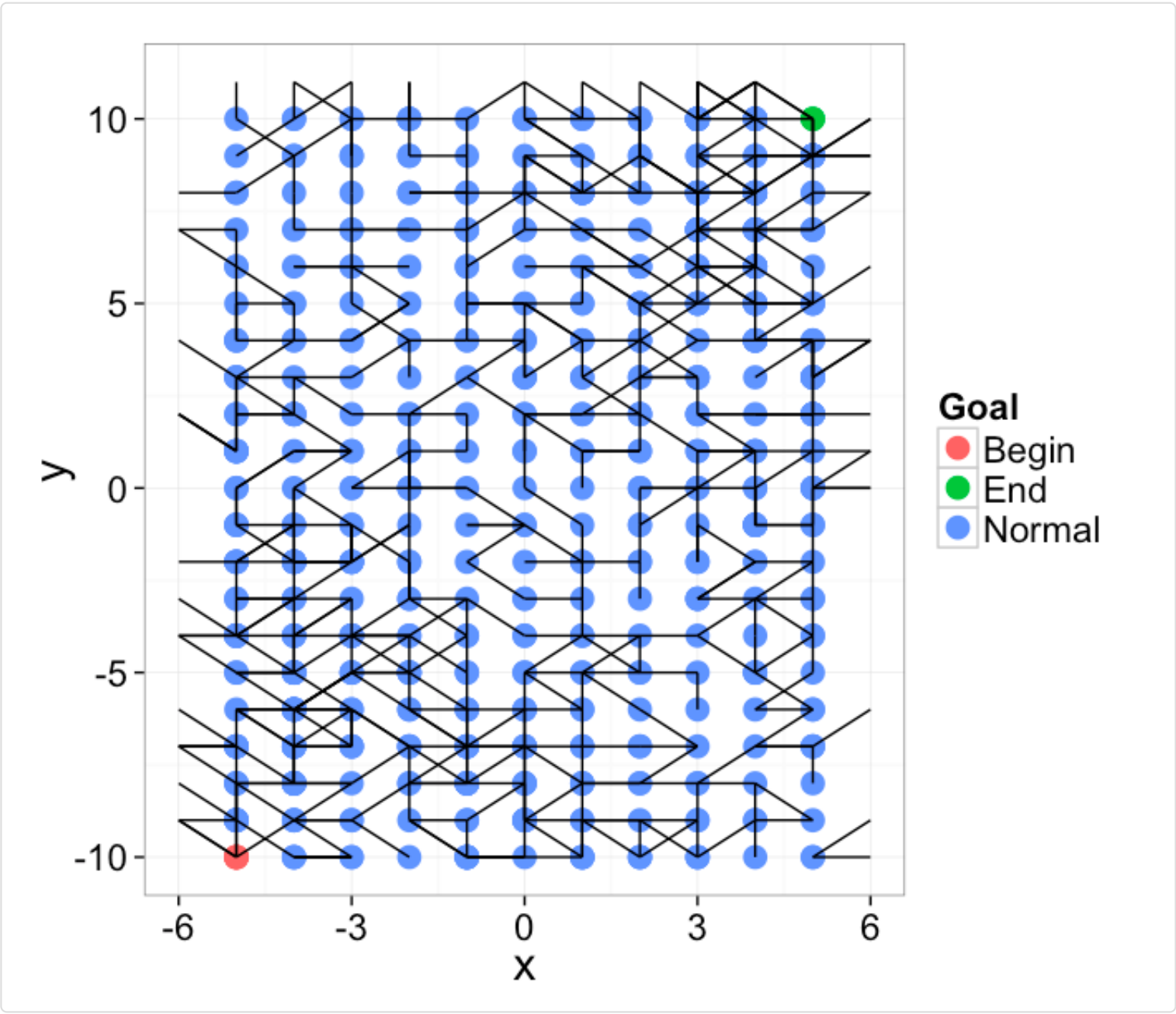
Component Labeling

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?



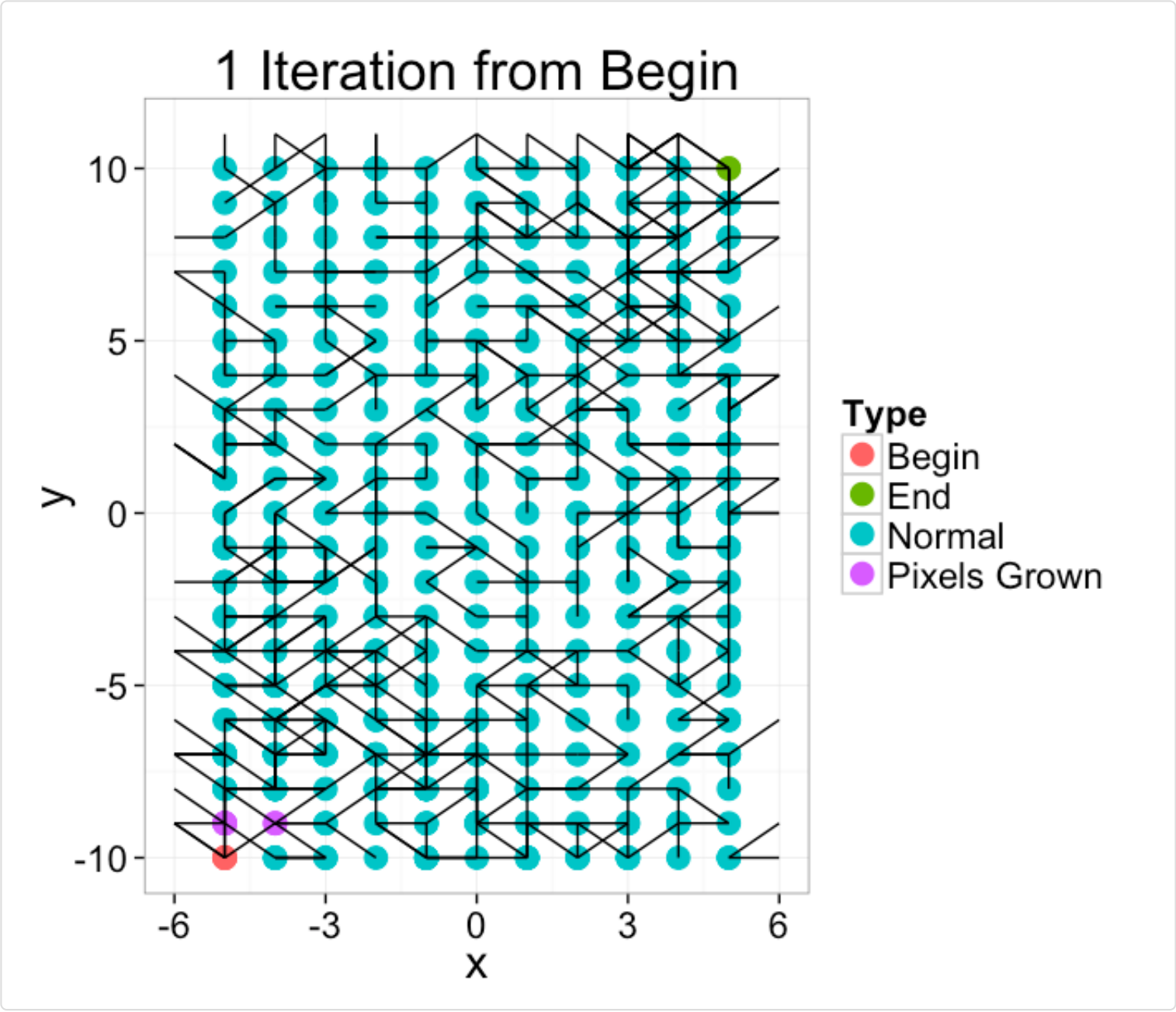
Component Labeling: Algorithm

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

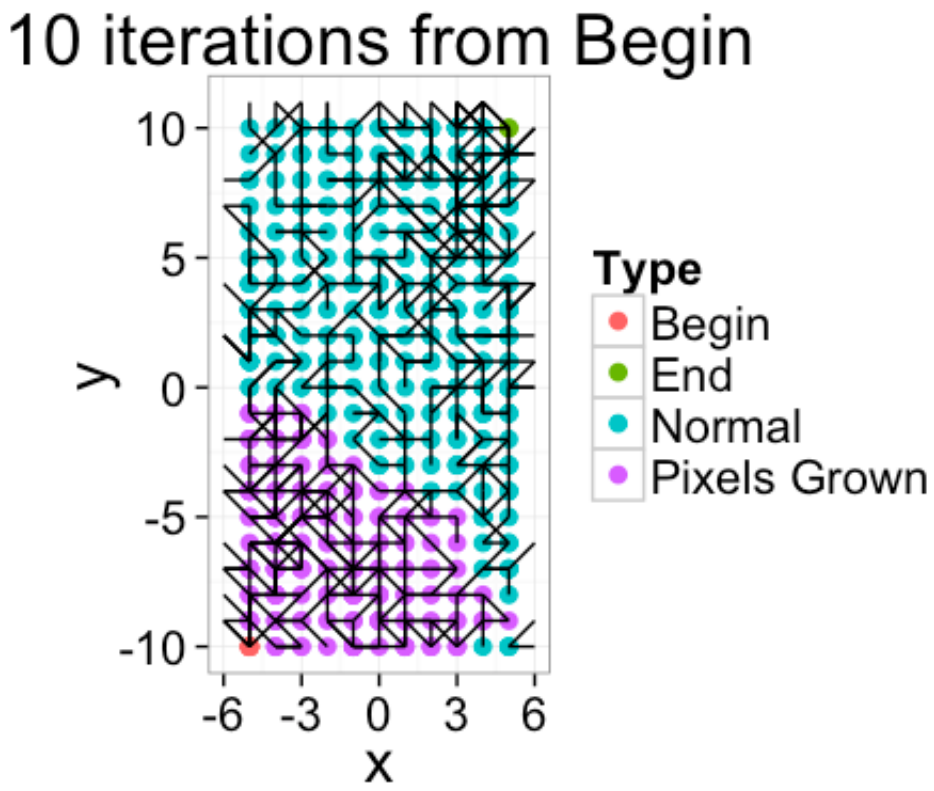
+/- R Code

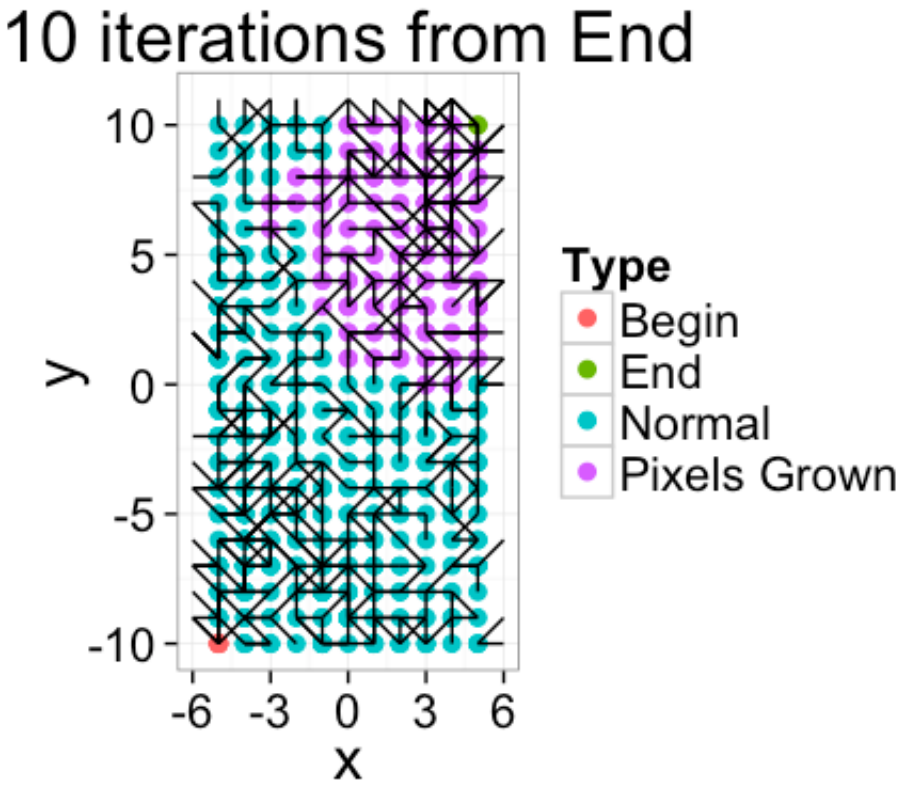
+/- R Code

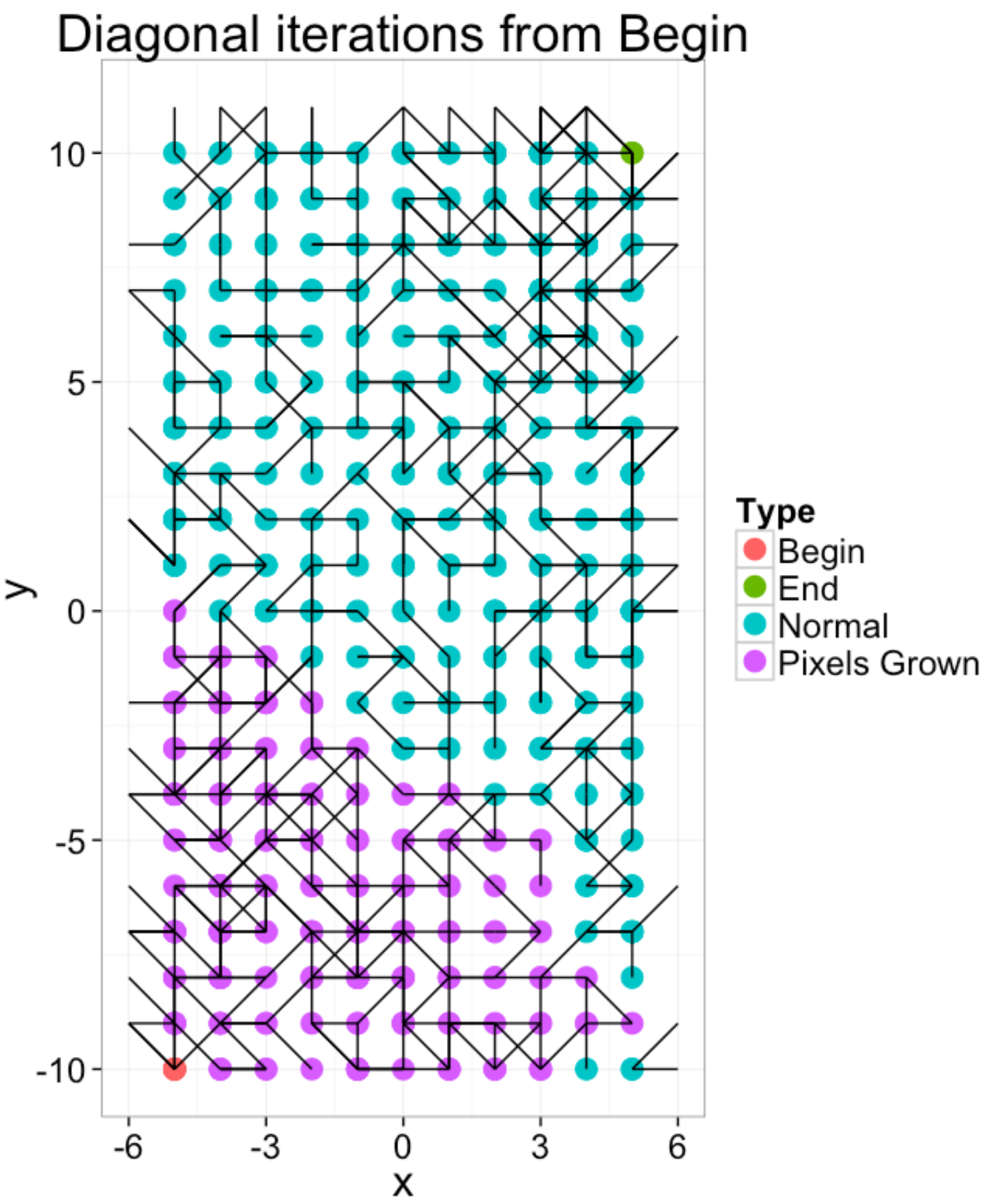


Component Labeling: Algorithm Steps

+/- R Code


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


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


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

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

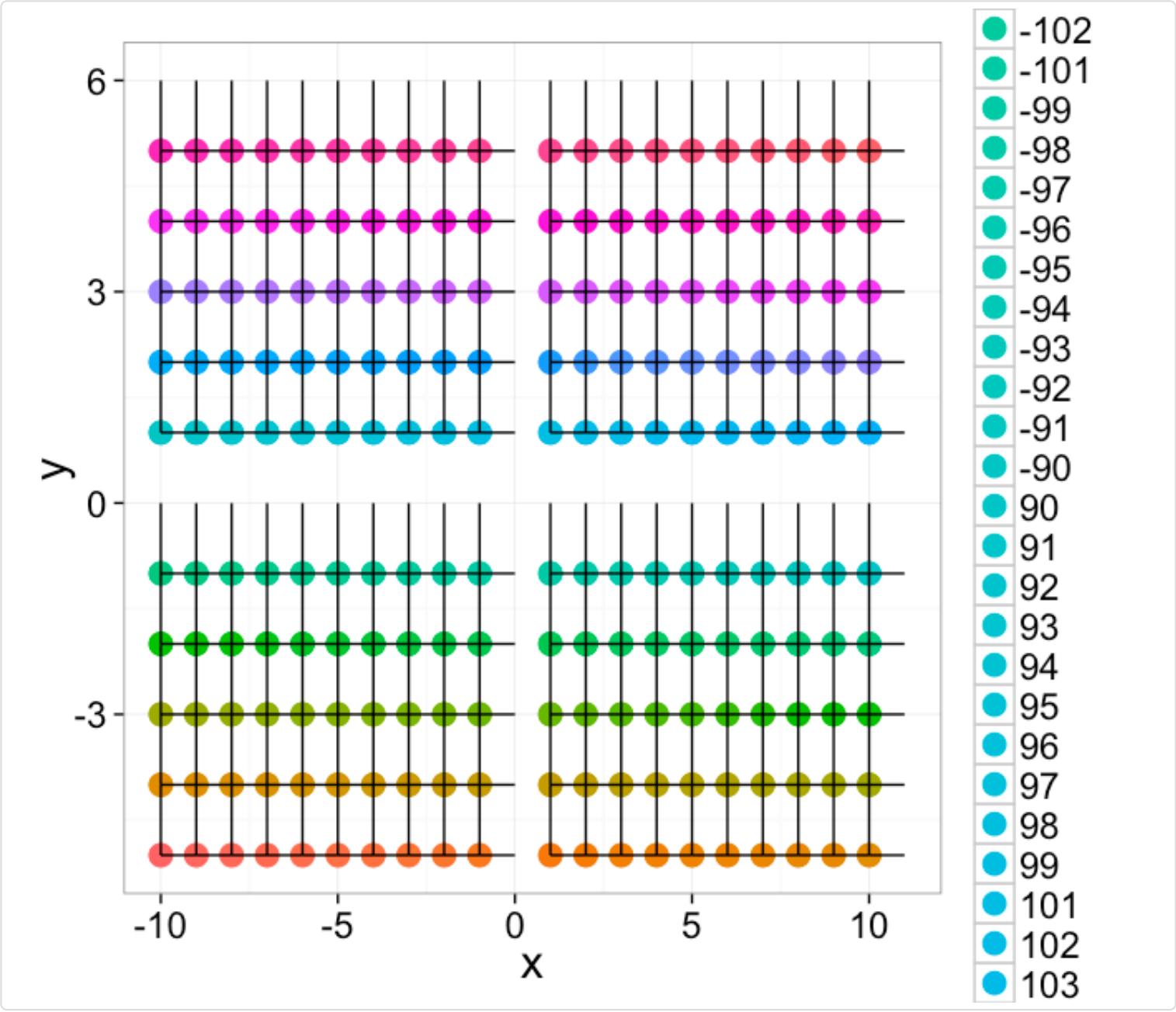
- For each point (x,y)
  - Check each point $(x',y') \in \mathcal{N}(x,y)$
  - Set

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

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

- Repeat until no more L values are changed

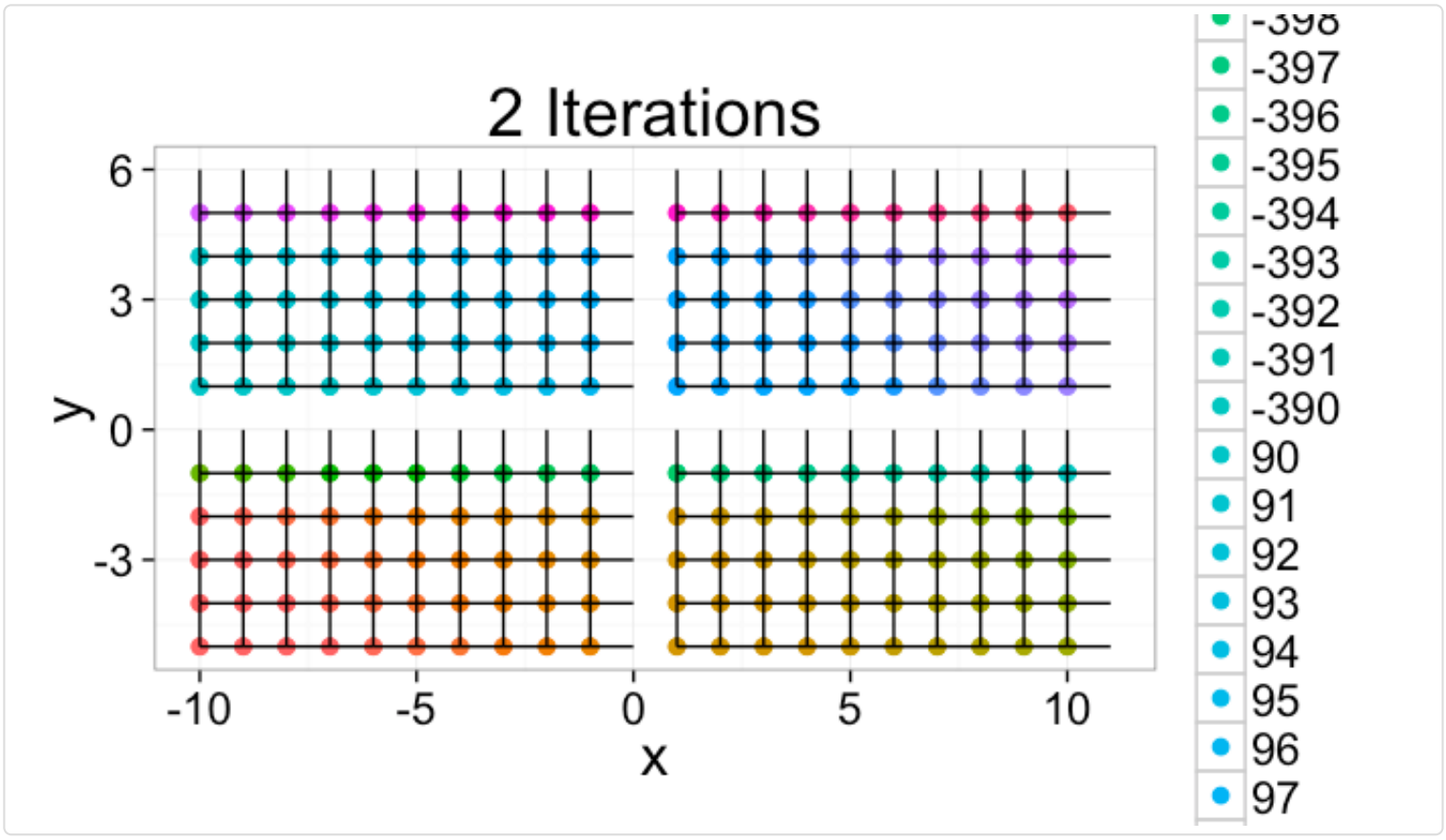
+/- R Code

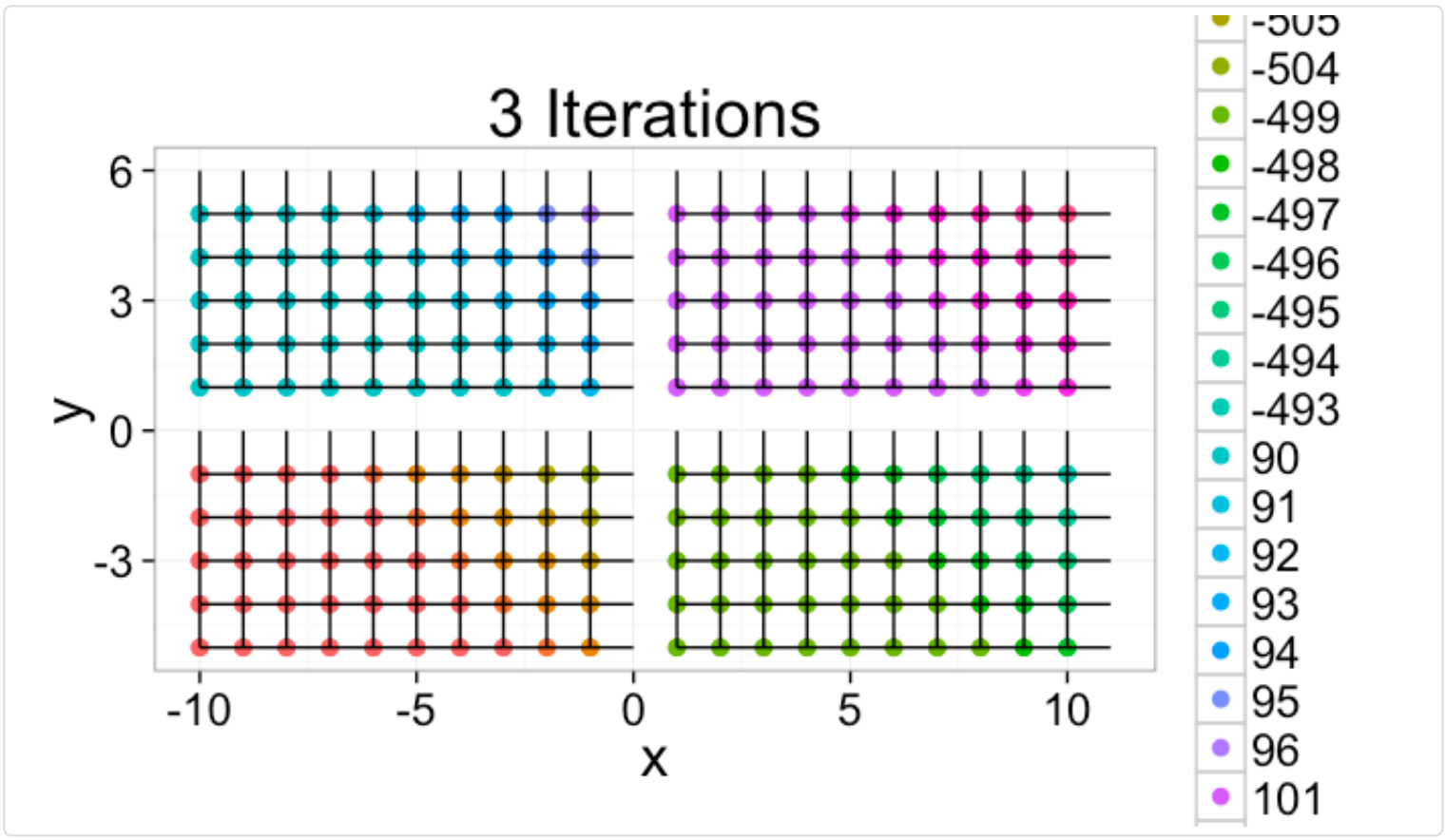


Component Labeling: Image Algorithm

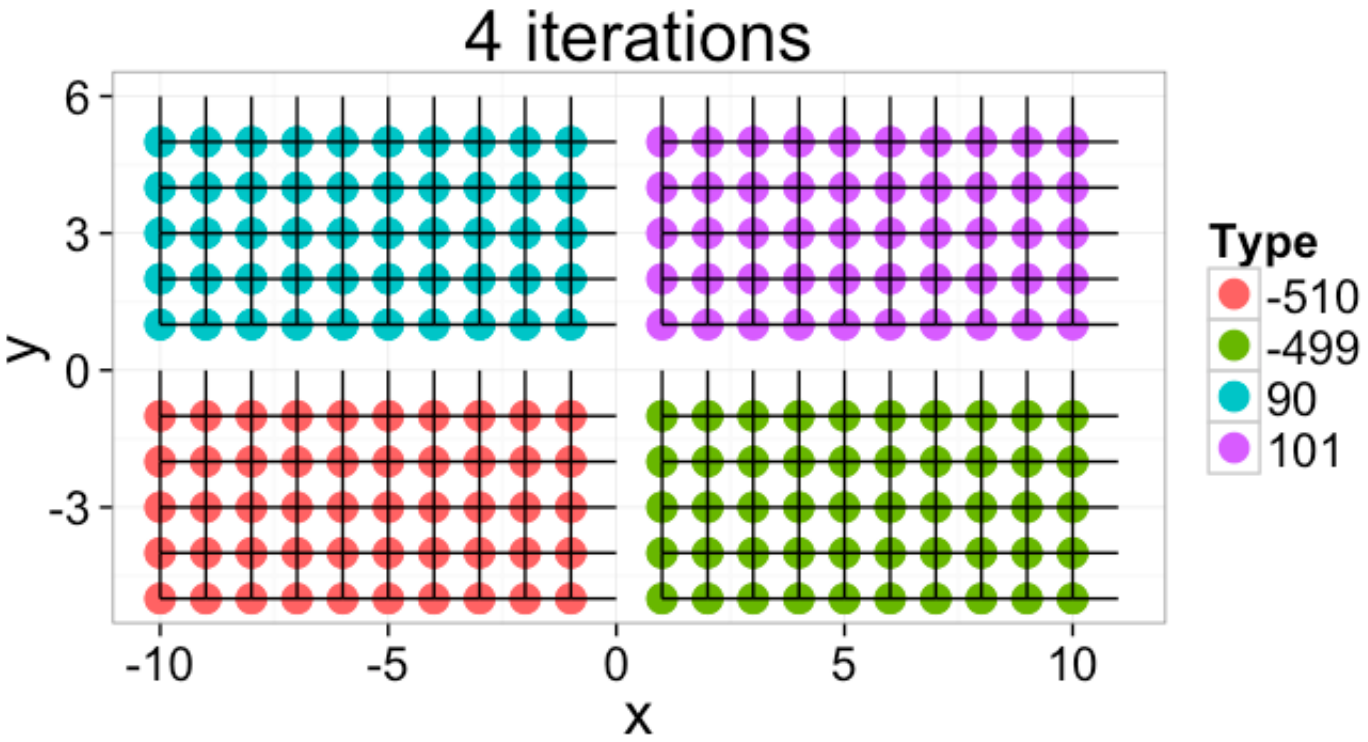
+/- R Code

+/- 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

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.

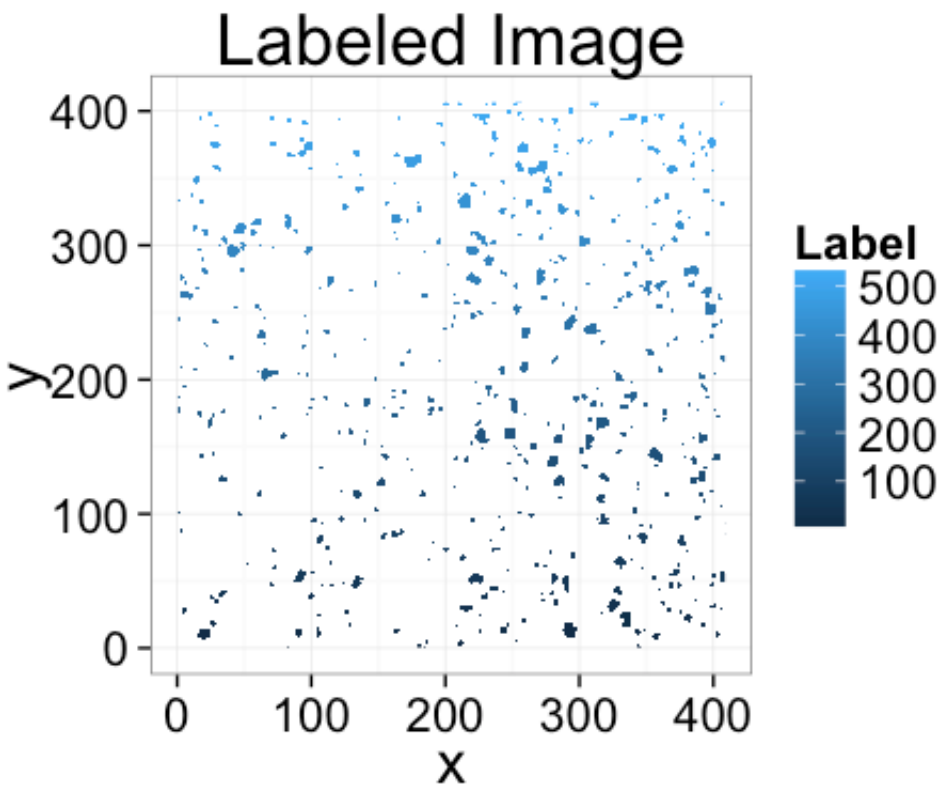


Component Labeling: Image Algorithm

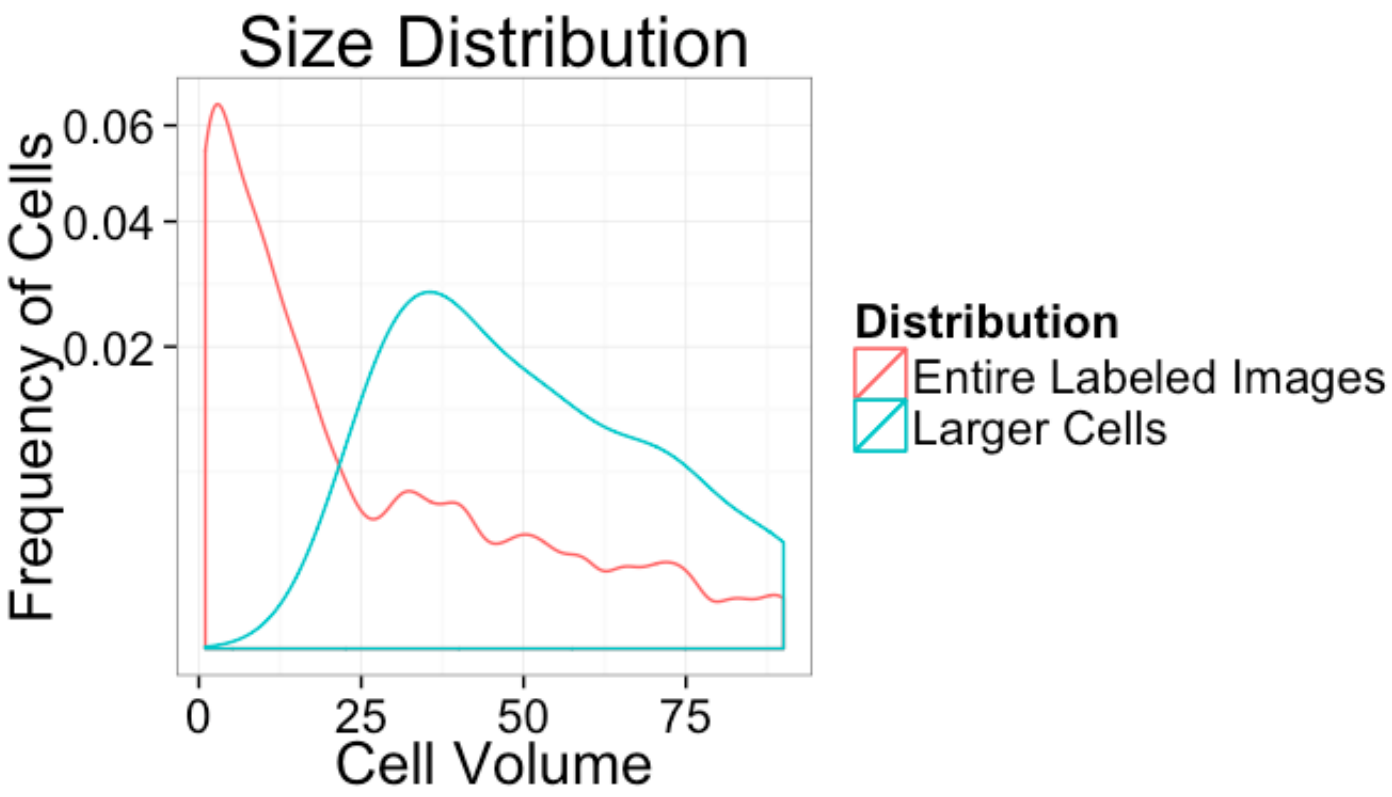
In reality the component labeling algorithms are usually implemented, but it is important to understand how they work and their relationship with neighborhood in order to interpret the results correctly.

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

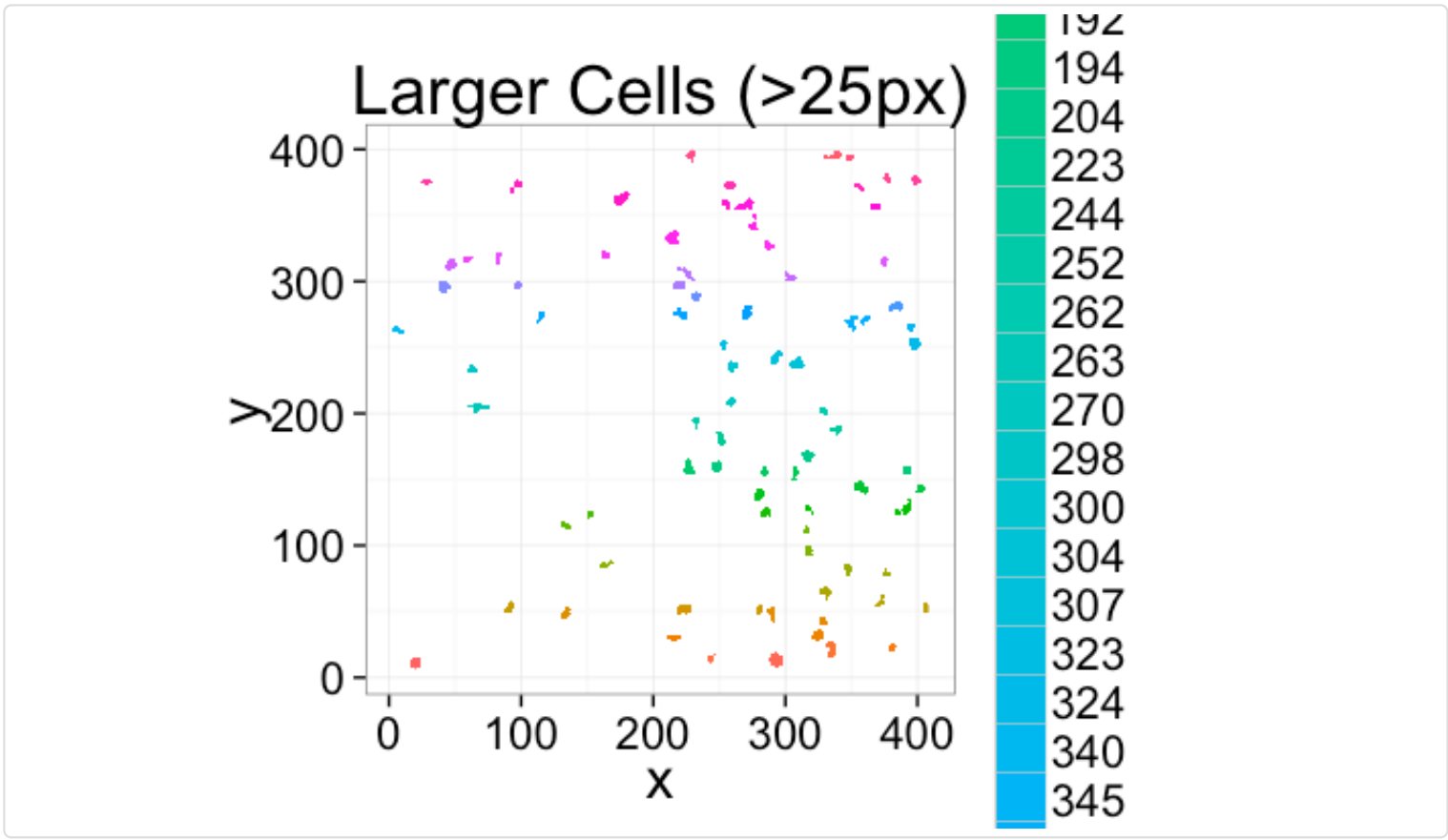
Component Labeling: Image Algorithm



+/- R Code


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

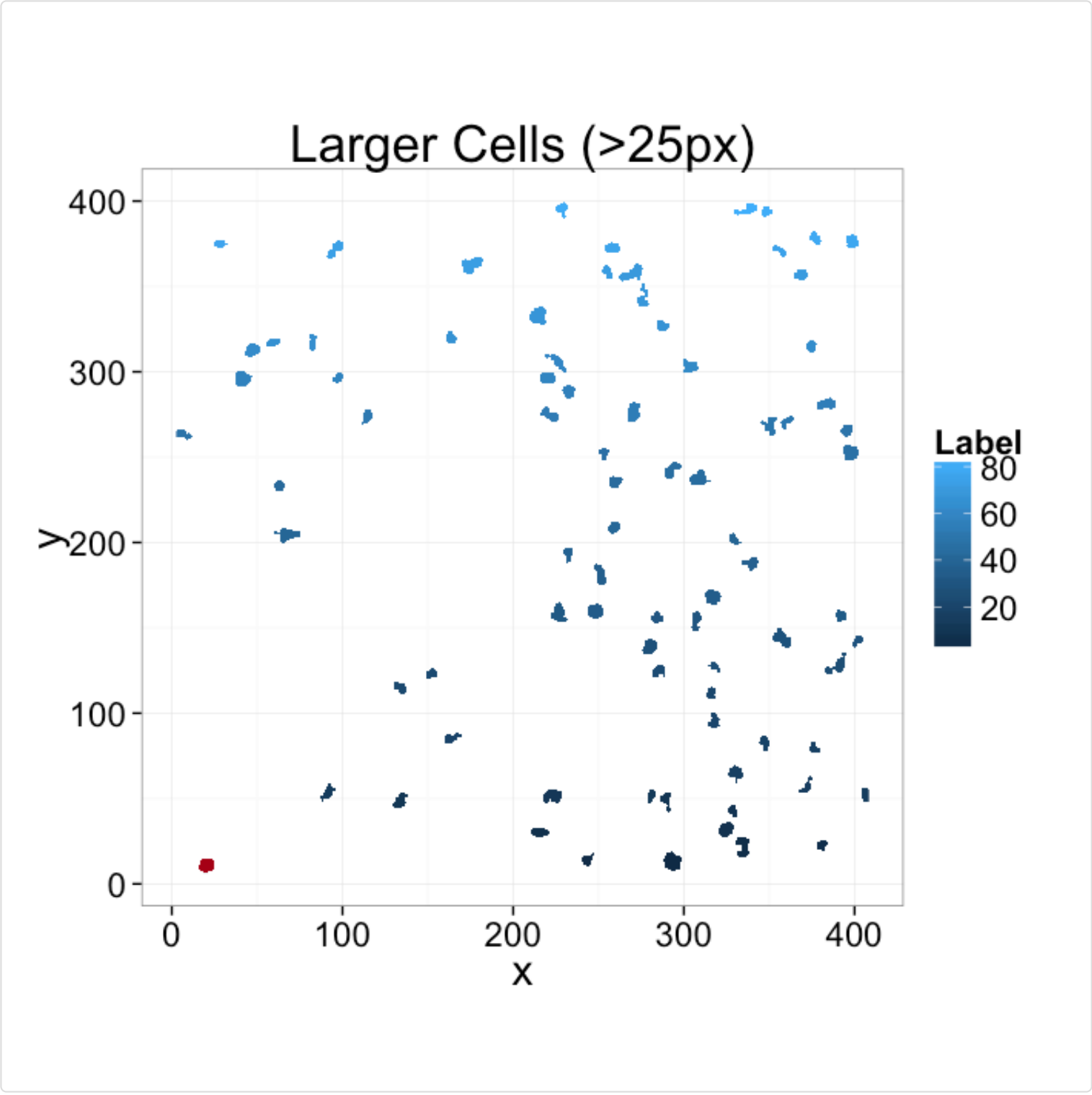
Now all the voxels which are connected have the same label. We can then perform simple metrics like

- 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.

From the labels

+/- R Code



What we would like to to do

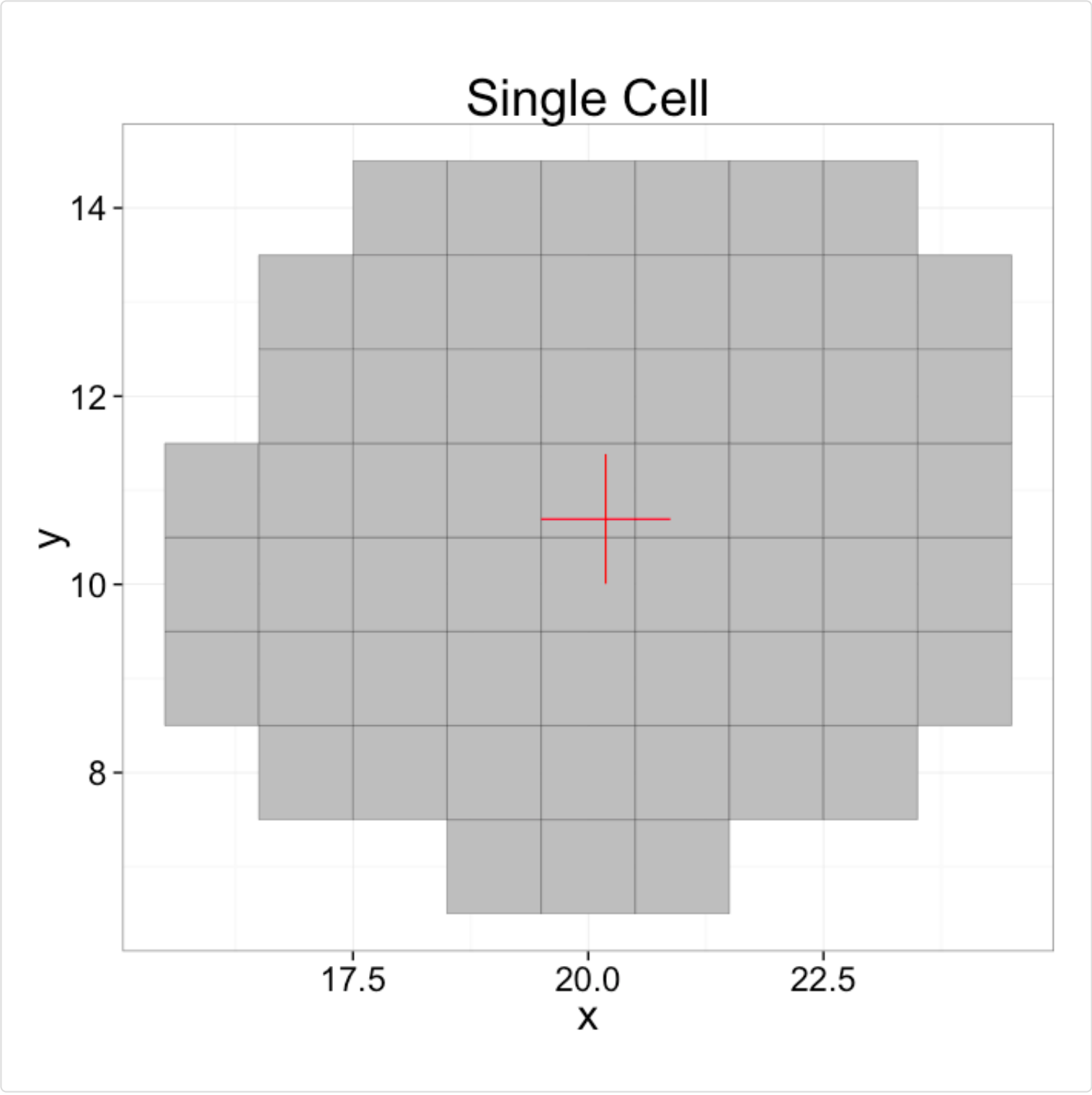
- Count the cells
- Say something about the cells
- Compare the cells in this image to another image

- But where do we start?

COV: With a single object

$$I_{id}(x,y) = \begin{cases} 1, & L(x,y) = id \\ 0, & \text{otherwise} \end{cases}$$

+/- R Code



Define a center

$$\bar{x} = \frac{1}{N} \sum_{\vec{v} \in I_{id}} \vec{v} \cdot \vec{i}$$

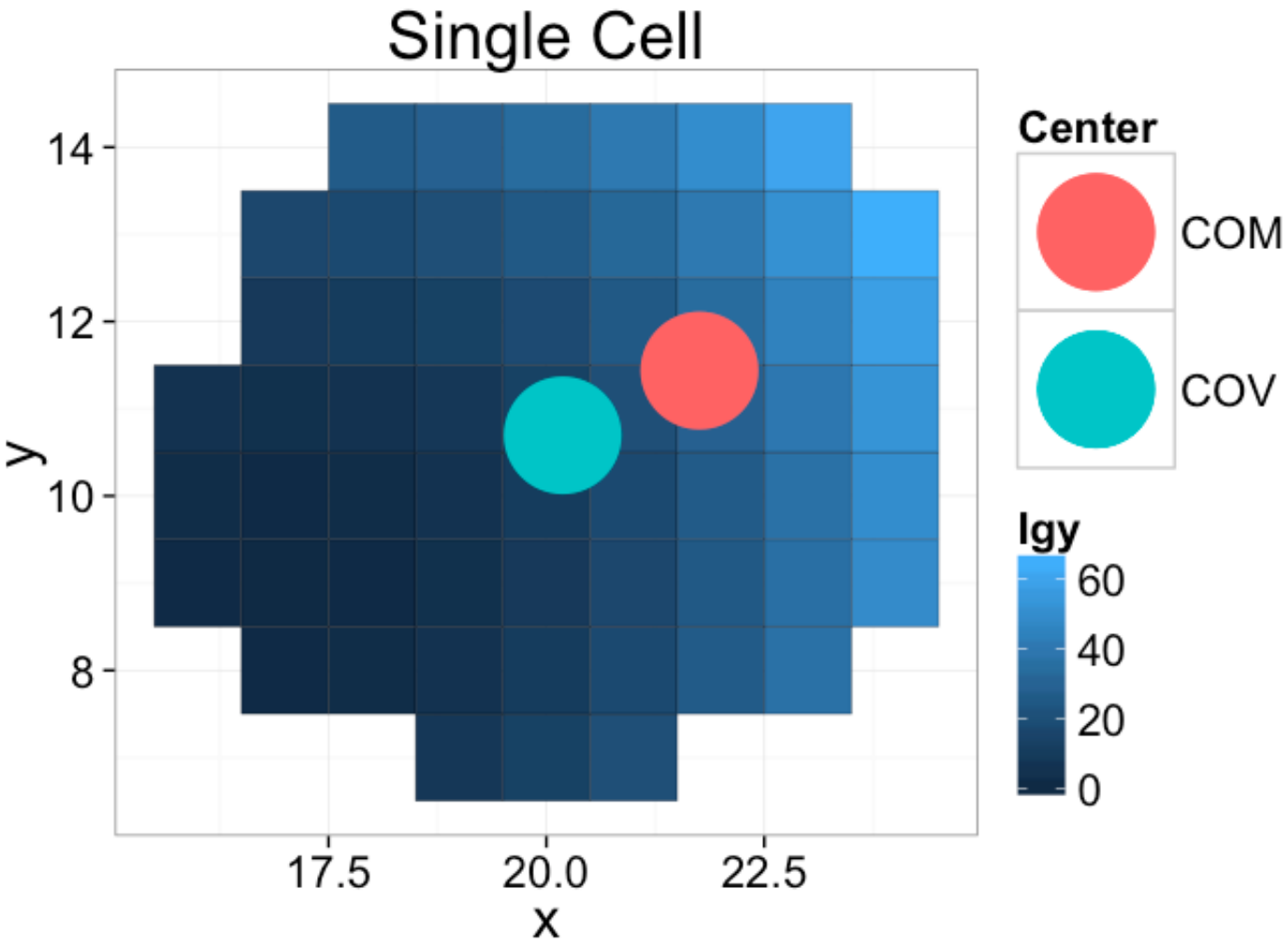
$$\bar{y} = \frac{1}{N} \sum_{\vec{v} \in I_{id}} \vec{v} \cdot \vec{j}$$

$$\bar{z} = \frac{1}{N} \sum_{\vec{v} \in I_{id}} \vec{v} \cdot \vec{k}$$

COM: With a single object

If the gray values are kept (or other meaningful ones are used), this can be seen as a weighted center of volume or center of mass (using I_{gy} to distinguish it from the labels)

+/- R Code



Define a center

$$\Sigma I_{gy} = \frac{1}{N} \sum_{\vec{v} \in I_{id}} I_{gy}(\vec{v})$$

$$\bar{x} = \frac{1}{\Sigma I_{gy}} \sum_{\vec{v} \in I_{id}} (\vec{v} \cdot \vec{i}) I_{gy}(\vec{v})$$

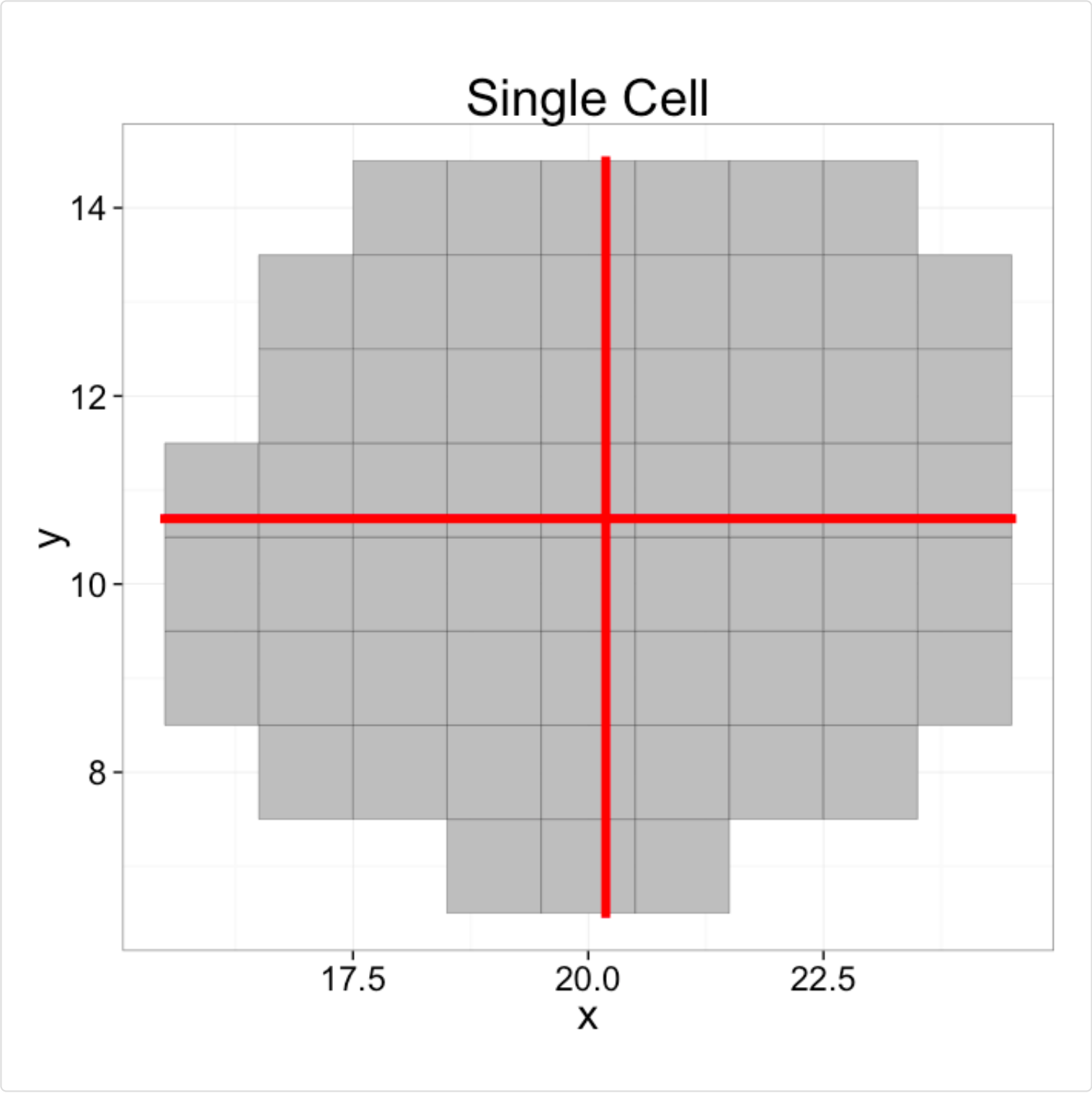
$$\bar{y} = \frac{1}{\Sigma I_{gy}} \sum_{\vec{v} \in I_{id}} (\vec{v} \cdot \vec{j}) I_{gy}(\vec{v})$$

$$\bar{z} = \frac{1}{\Sigma I_{gy}} \sum_{\vec{v} \in I_{id}} (\vec{v} \cdot \vec{k}) I_{gy}(\vec{v})$$

Extents: With a single object

Exents or caliper lenghts are the size of the object in a given direction. Since the coordinates of our image our x and y the extents are calculated in these directions

+/- R Code



Define extents as the minimum and maximum values along the projection of the shape in each direction

$$\text{Ext}_x = \left\{ \forall \vec{v} \in I_{id} : max(\vec{v} \cdot \vec{i}) - min(\vec{v} \cdot \vec{i}) \right\}$$

$$\text{Ext}_y = \left\{ \forall \vec{v} \in I_{id} : max(\vec{v} \cdot \vec{j}) - min(\vec{v} \cdot \vec{j}) \right\}$$

$$\text{Ext}_z = \left\{ \forall \rightarrow \in I_{id} : max(\vec{v} \cdot \vec{k}) - min(\vec{v} \cdot \vec{k}) \right\}$$

- Lots of information about each object now
- But, I don't think a biologist has ever asked "How long is a cell in the x direction? how about y?"

Anisotropy: What is it?

By definition (New Oxford American): ~~varying in magnitude according to the direction of measurement.~~

- It allows us to define metrics in respect to one another and thereby characterize shape.
- Is it tall and skinny, short and fat, or perfectly round

Due to its very vague definition, it can be mathematically characterized in many different very much unequal ways (in all cases 0 represents a sphere)

$$Aiso1 = \frac{\text{Longest Side}}{\text{Shortest Side}} - 1$$

$$Aiso2 = \frac{\text{Longest Side} - \text{Shortest Side}}{\text{Longest Side}}$$

$$Aiso3 = \frac{\text{Longest Side}}{\text{Average Side Length}} - 1$$

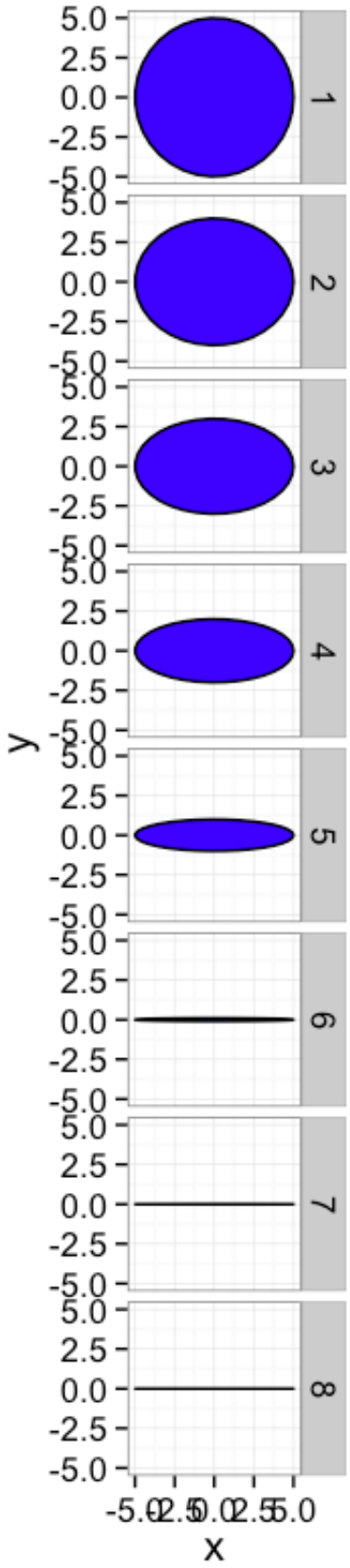
$$Aiso4 = \frac{\text{Longest Side} - \text{Shortest Side}}{\text{Average Side Length}}$$

$$\cdots \rightarrow \text{ ad nauseum}$$

Anisotropy: Which definition makes sense?

Let's take some sample objects

+/- R Code

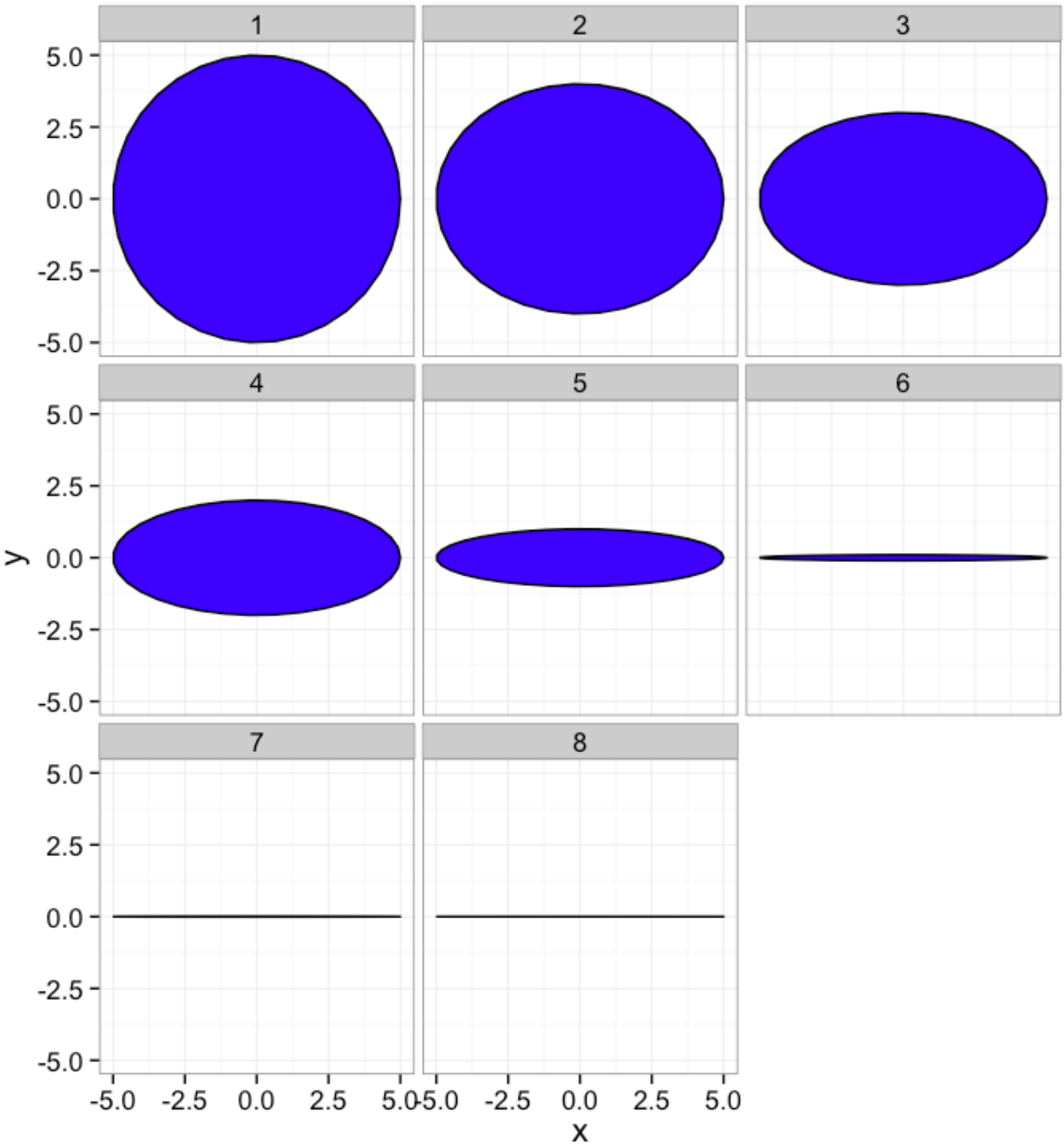

y
x
1
2
3
4
5
6
7
8
5.0
2.5
0.0
-2.5
-5.0

+/- R Code

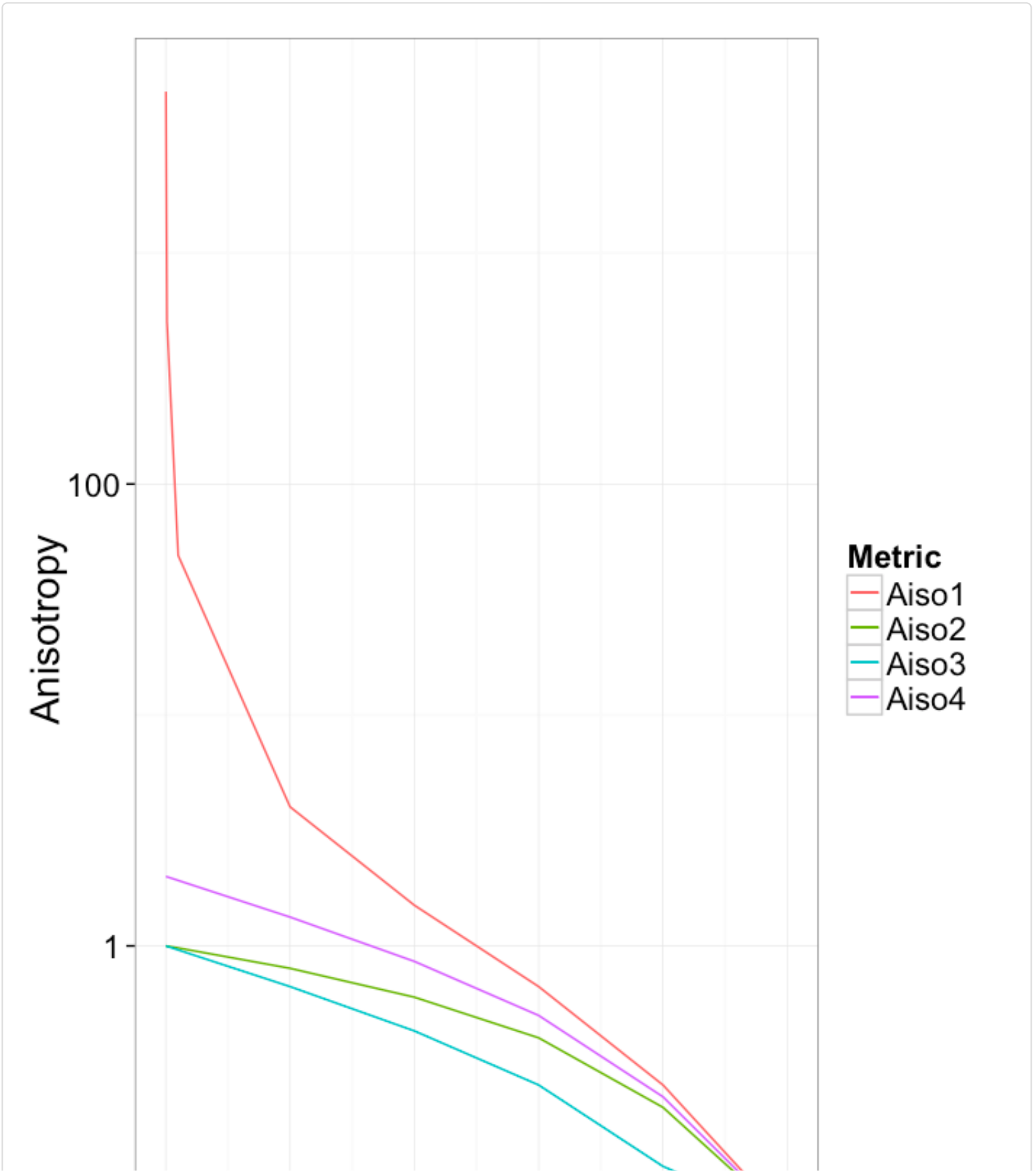
Y Extent	Aiso1	Aiso2	Aiso3	Aiso4
0.00	4999.00	1.00	1.00	2.00
0.01	499.00	1.00	1.00	1.99
0.10	49.00	0.98	0.96	1.92
1.00	4.00	0.80	0.67	1.33
2.00	1.50	0.60	0.43	0.86
3.00	0.67	0.40	0.25	0.50
4.00	0.25	0.20	0.11	0.22
5.00	0.00	0.00	0.00	0.00

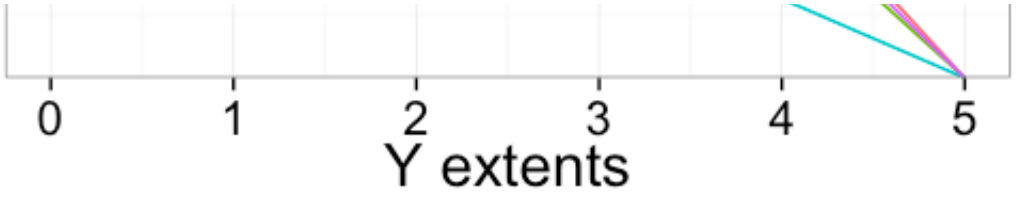
Anisotropy: Which definition makes sense?

+/- R Code


1
2
3
4
5
6
7
8
5.0
2.5
0.0
-2.5
-5.0
y
-5.0
-2.5
0.0
2.5
5.0
x

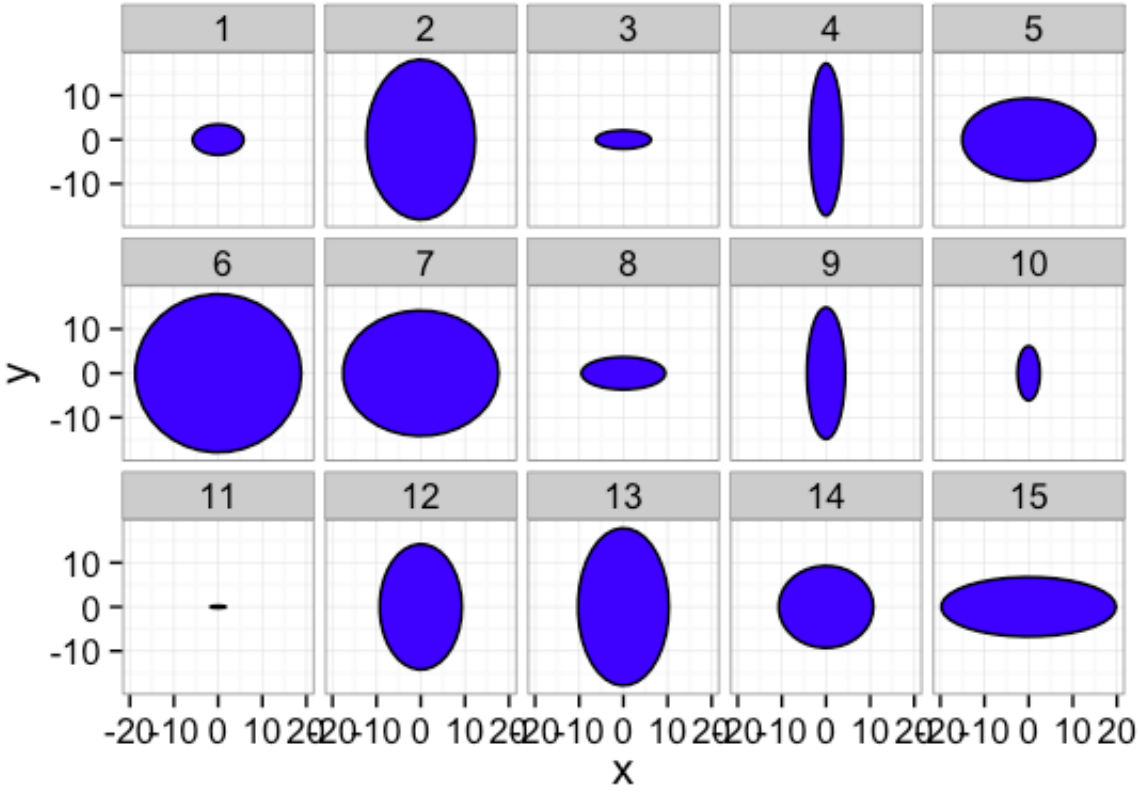
Anisotropy vs x extents



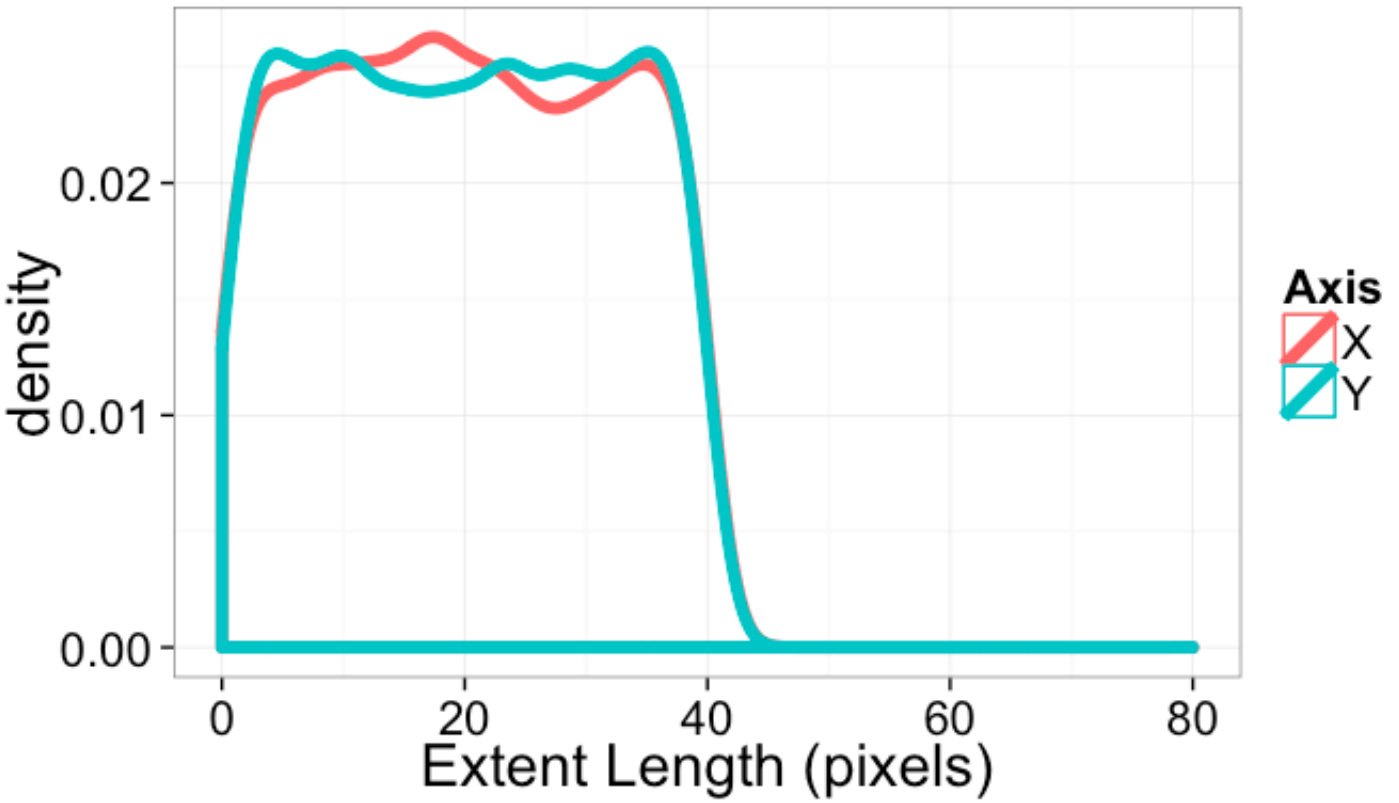


Distributions: Randomly Sized Objects

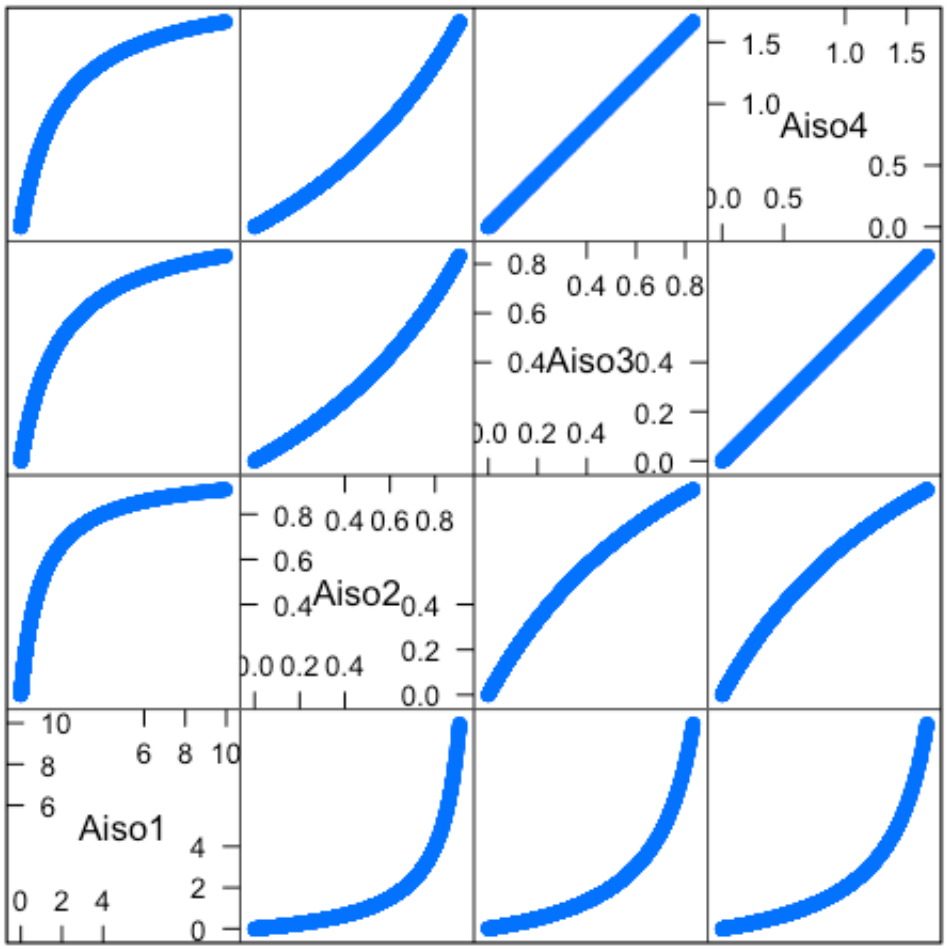
Objects with uniformally distributed, independent x and y extents



+/- R Code

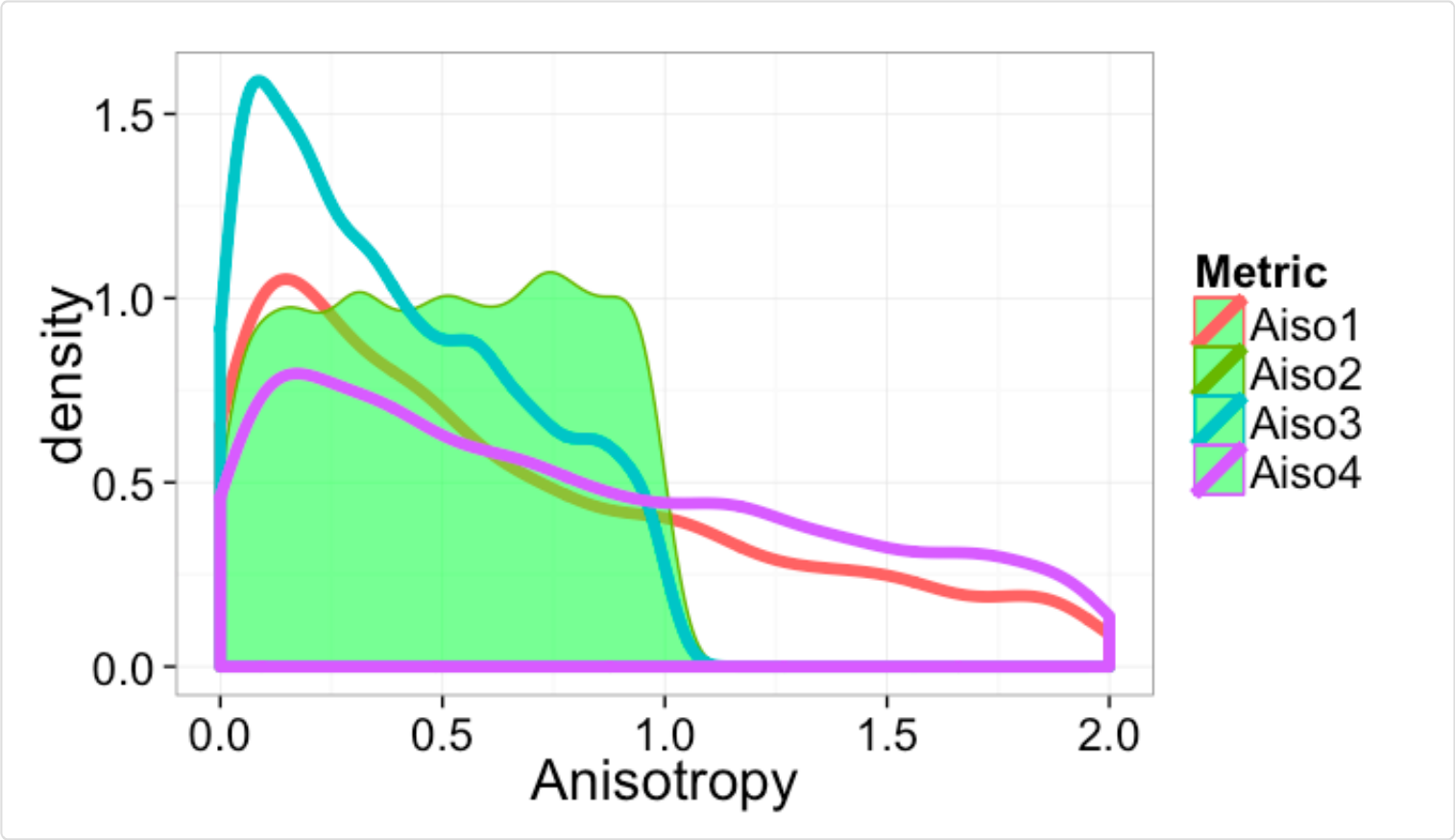


+/- R Code

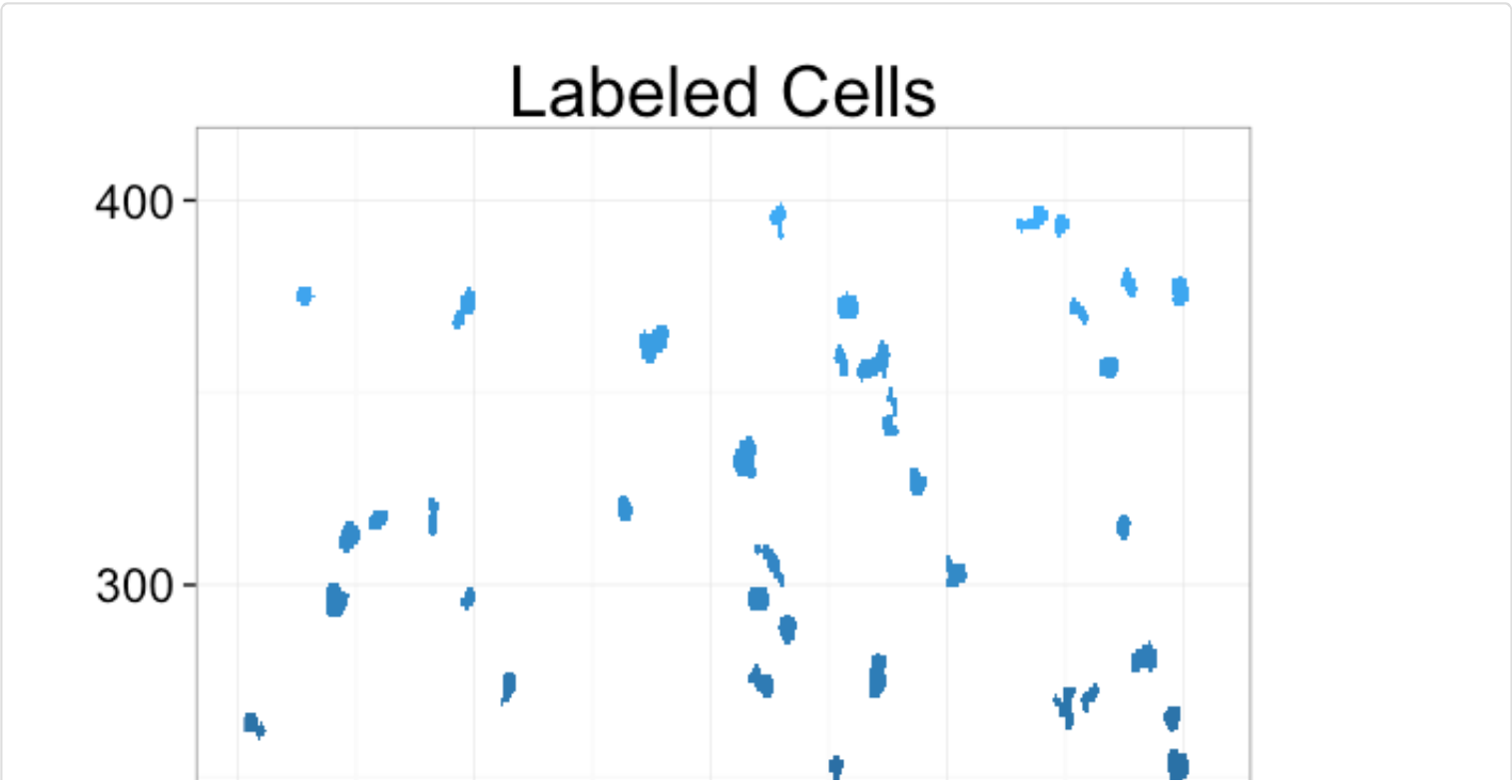


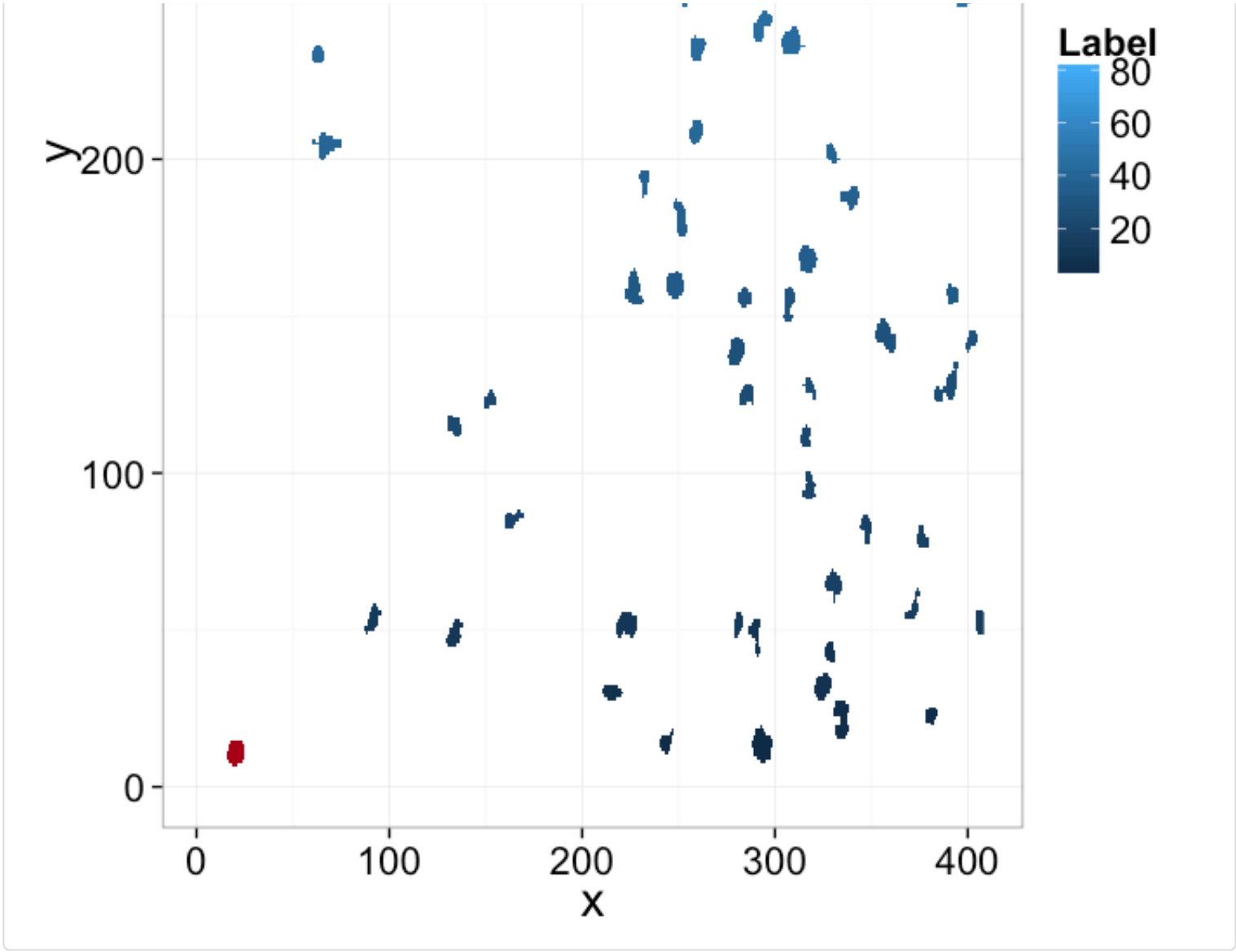
Scatter Plot Matrix

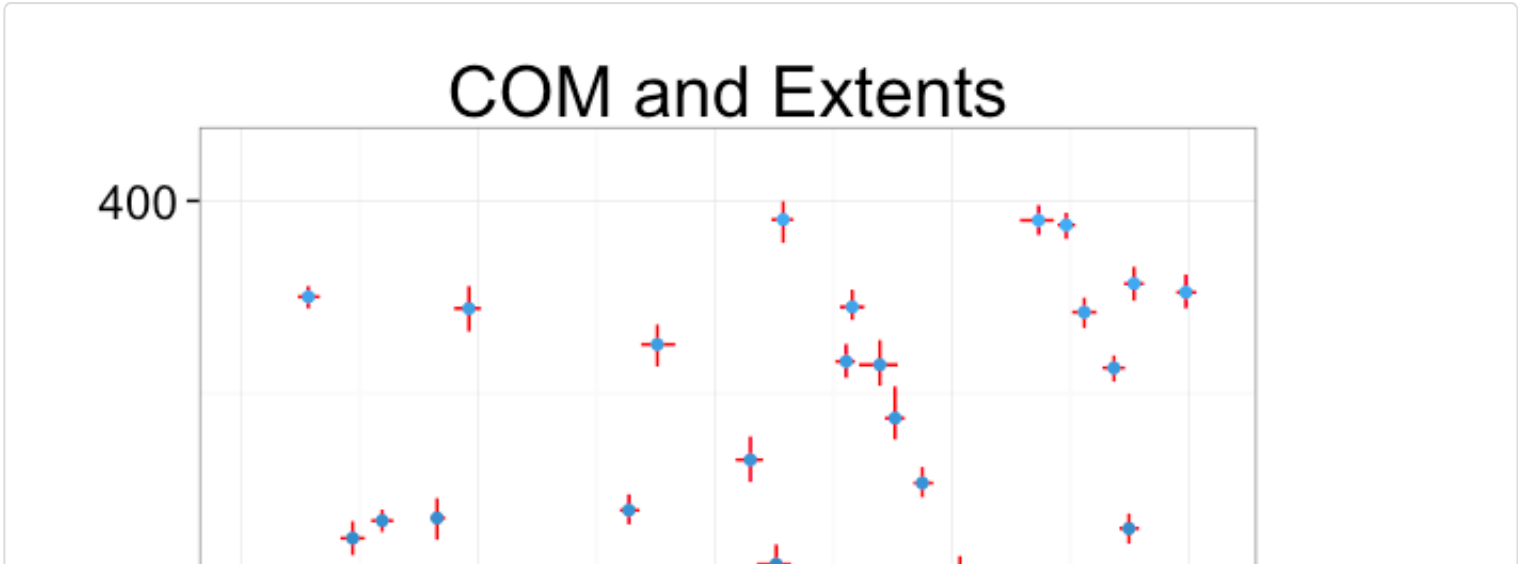
+/- R Code

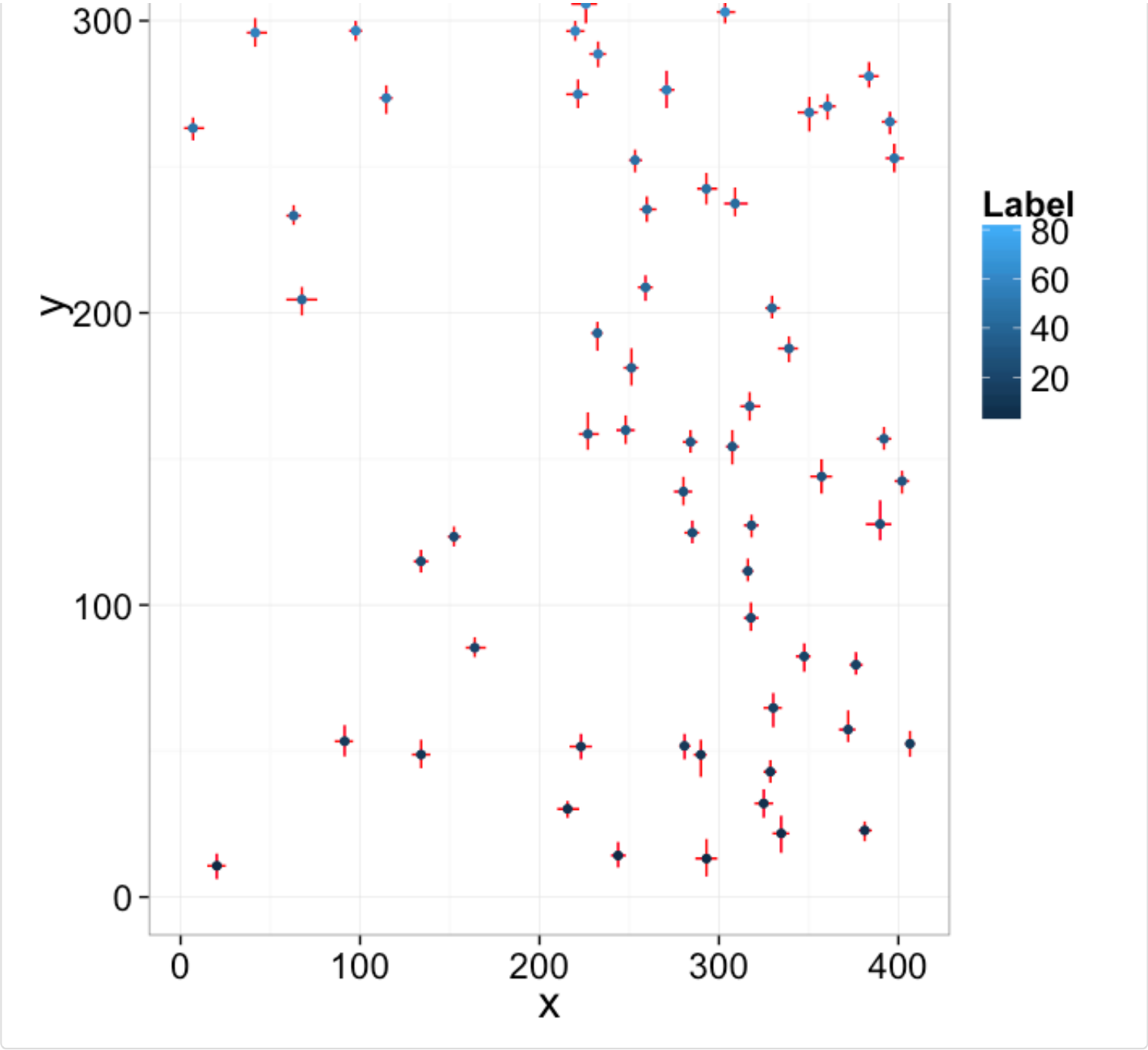


Extents: With all objects



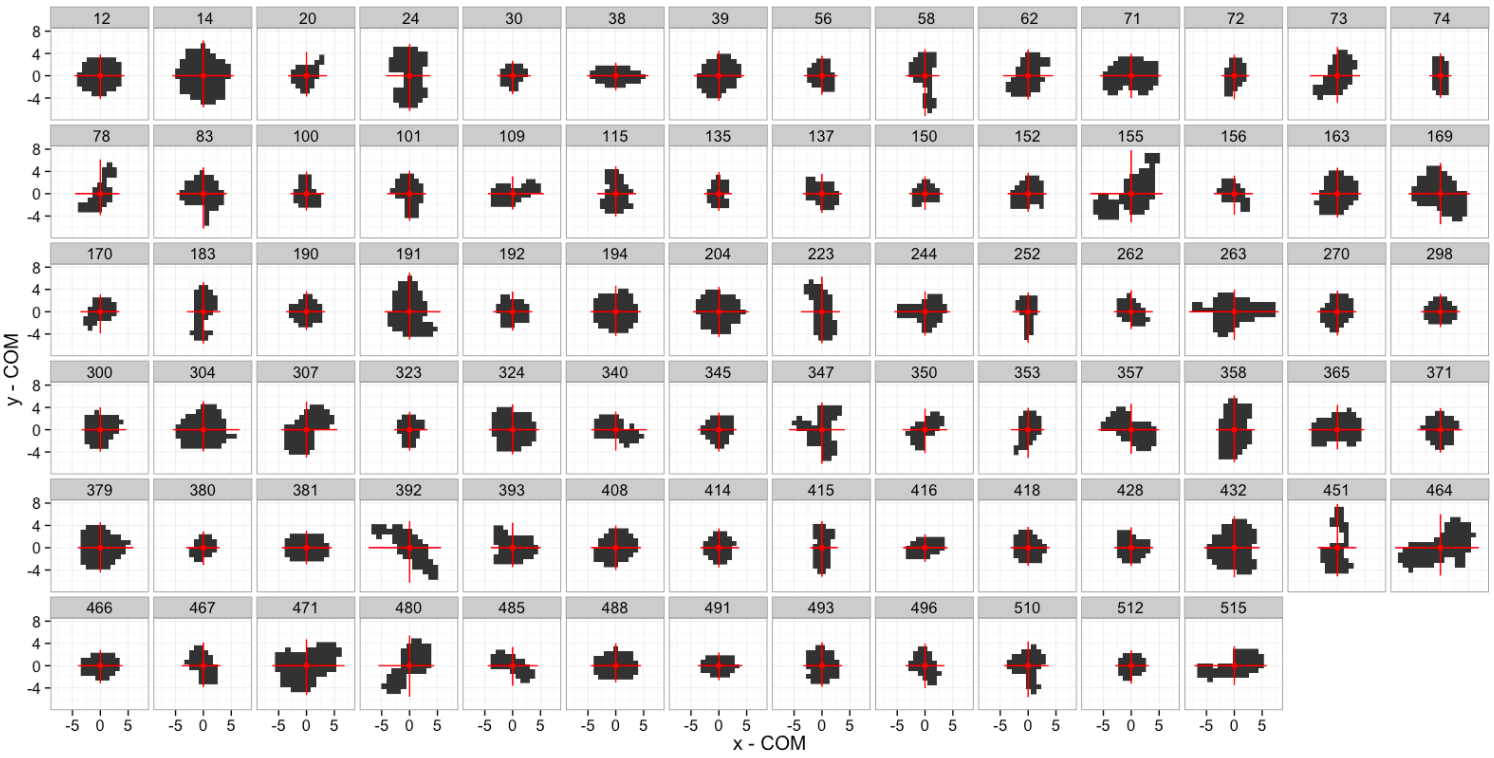

Label
80
60
40
20
y
200
100
0
0
100
200
300
400
x


COM and Extents
400



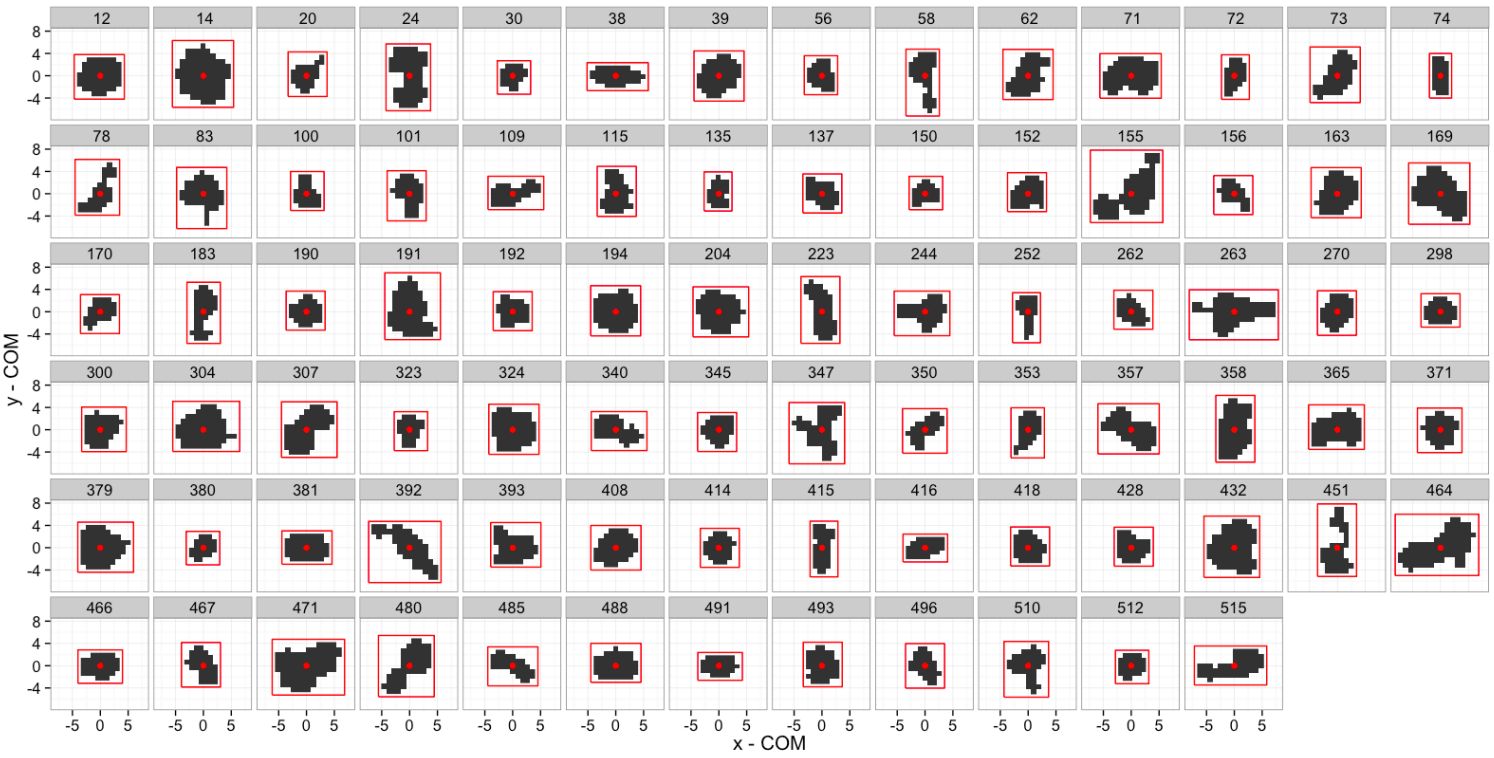
COV and Extents: All Cells

+/- R Code



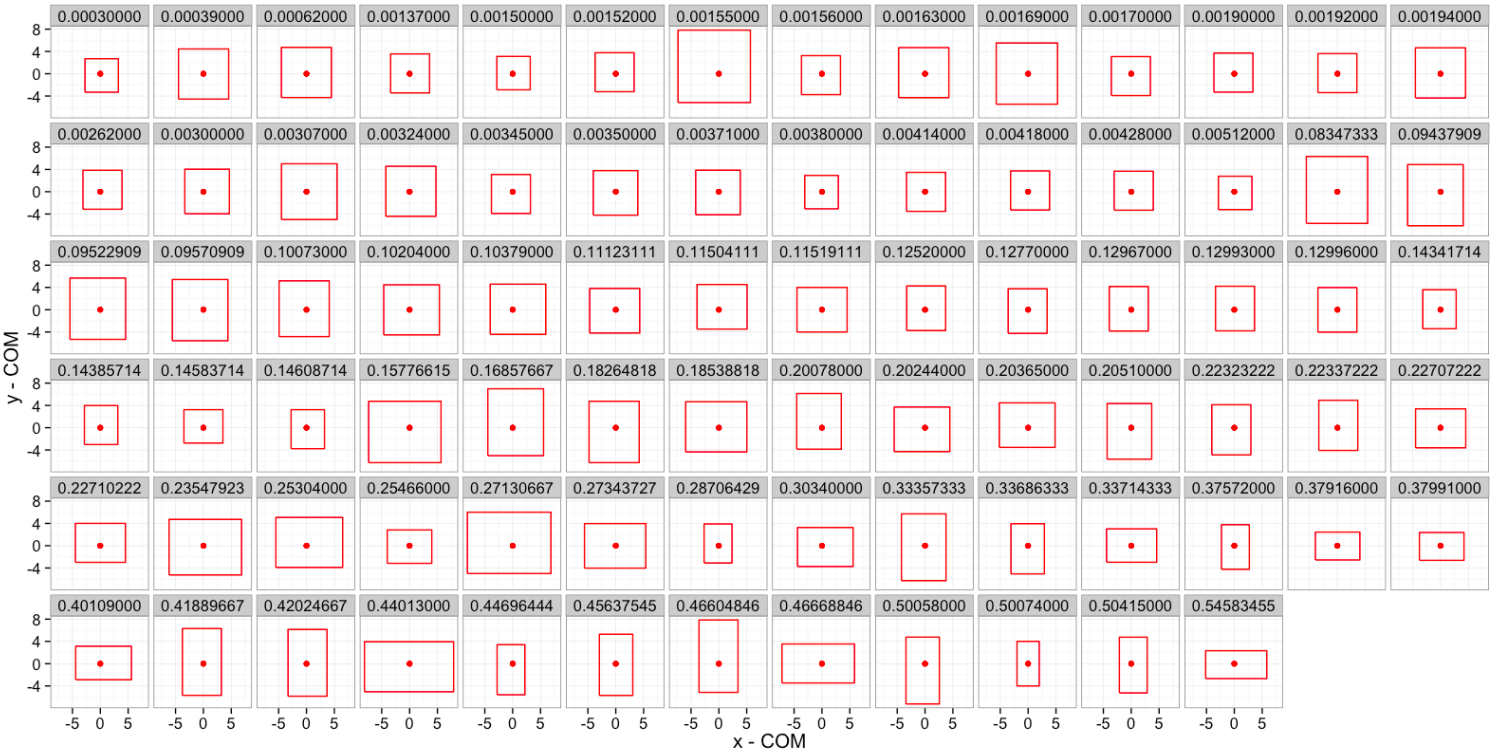
Bounding Box: All Cells

+/- R Code



Sorted by Anisotropy

+/- R Code



Bounding Box is a Poor Approximation

While easy to calculate, the bounding box / extents approach is a very rough approximation for most of the objects in our image. In particular objects which are not parallel to the XY-axes are misrepresented.

Possible Solutions

- Orient the boxes so there is the least amount of empty space or it best fits the data
- Assume the object is an ellipse and do some sort of curve fitting to find the object
- Don't worry about height and width and focus instead on more general characteristics like curvature, thickness (next lecture)

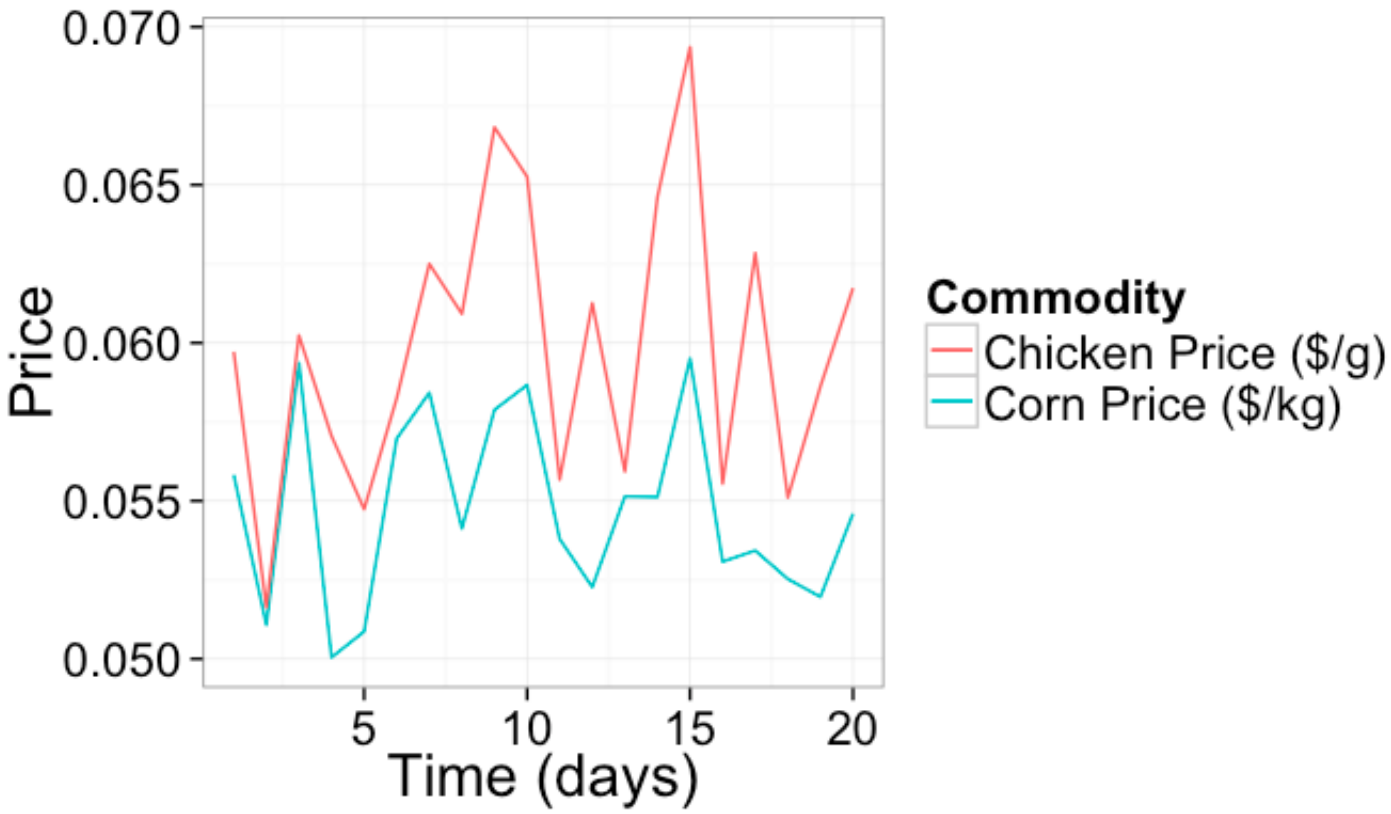
Useful Statistical Tools: Principal Component Analysis

While many of the topics covered in Linear Algebra and Statistics courses might not seem very applicable to real problems at first glance, at least a few of them come in handy for dealing distributions of pixels ~~(they will only be briefly covered, for more detailed review look at some of the suggested material)~~

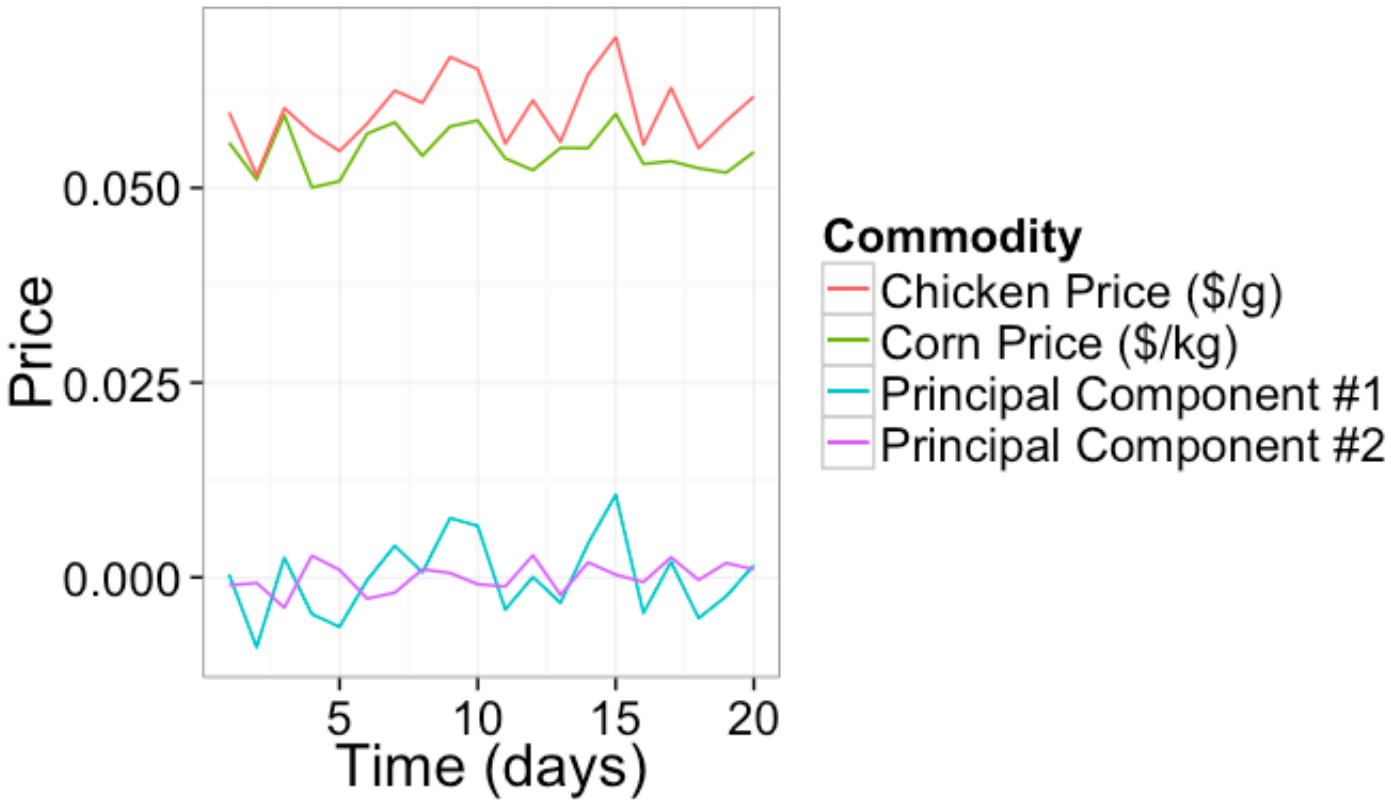
Principal Component Analysis

Similar to K-Means insofar as we start with a series of points in a vector space and want to condense the information. With PCA instead of searching for distinct groups, we try to find a linear combination of components which best explain the variance in the system.

As an example we will use a very simple example of corn and chicken prices vs time



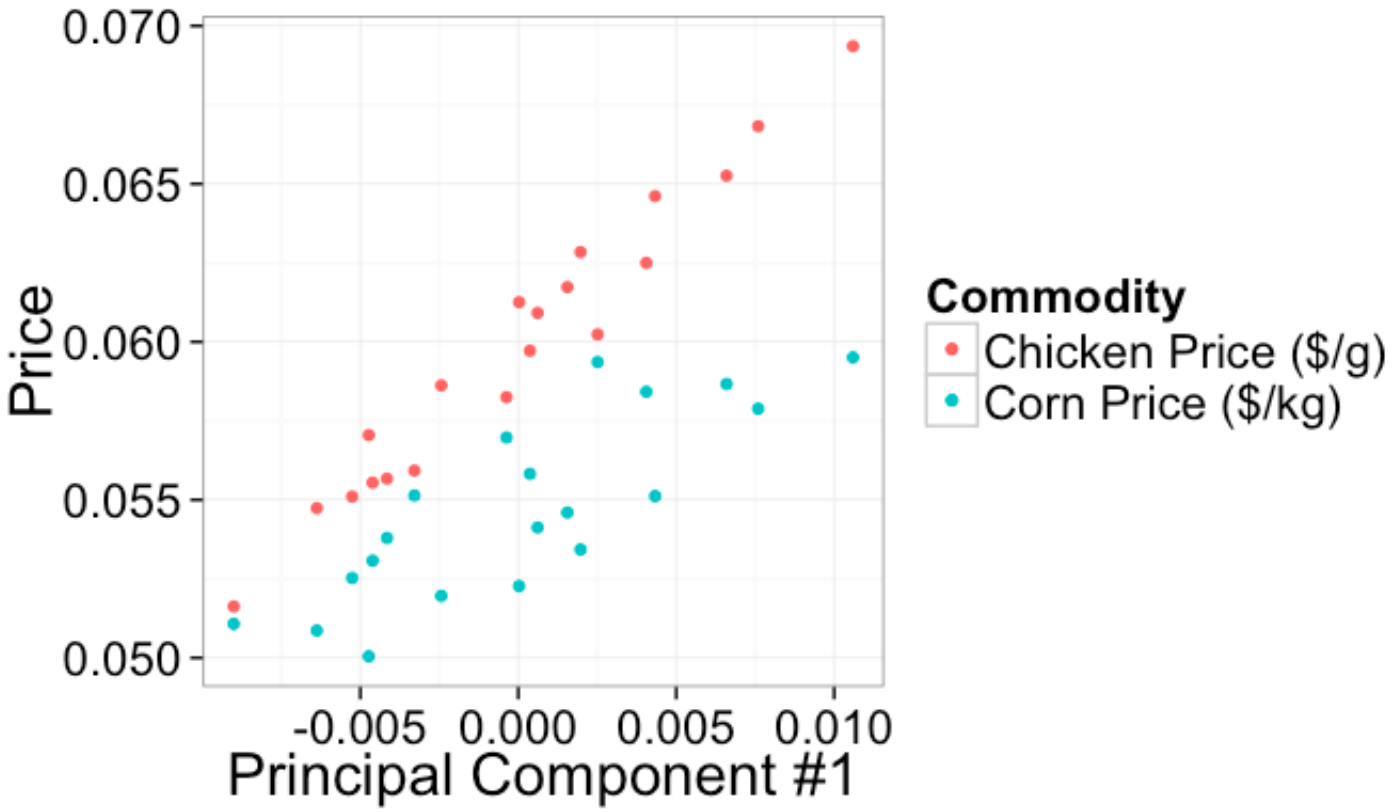
+/- R Code



Useful Statistical Tools: Principal Component Analysis

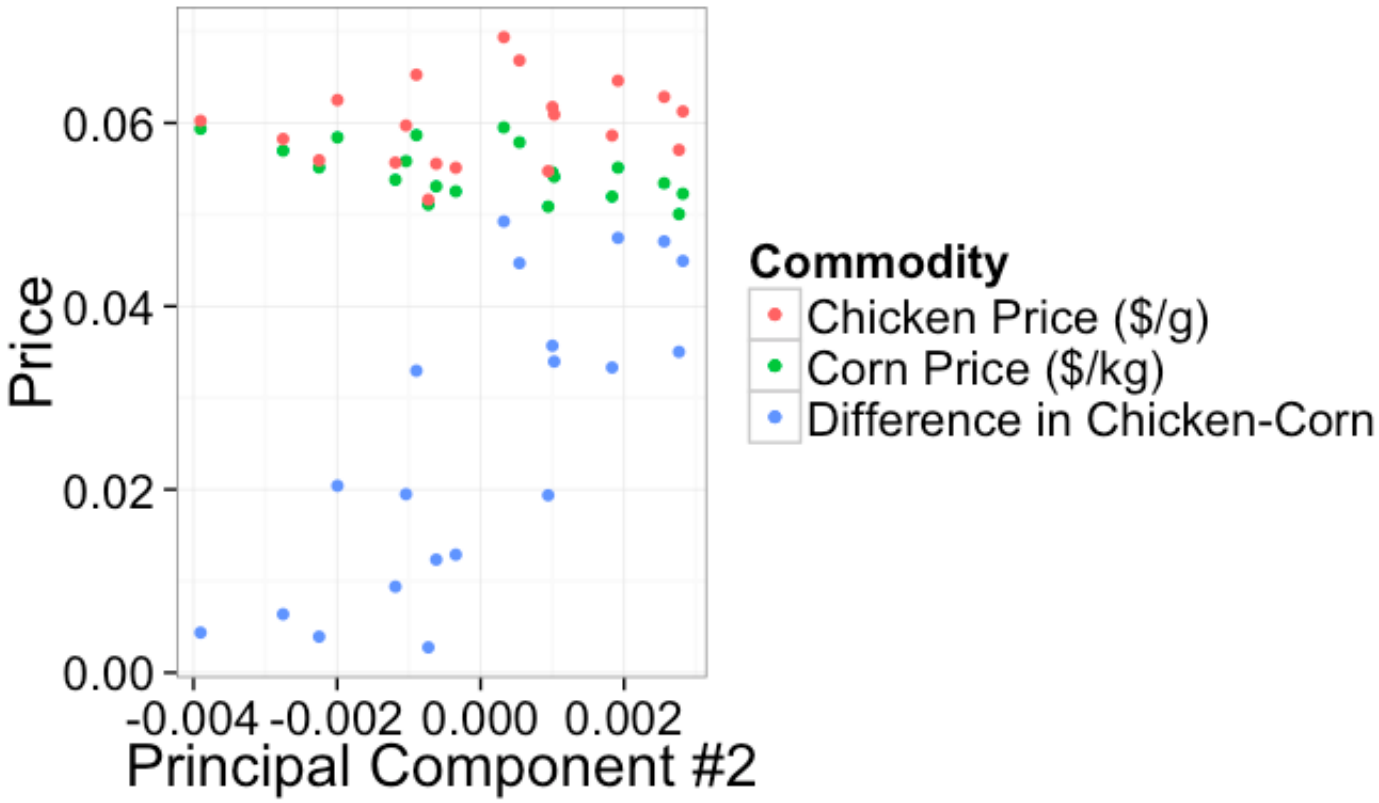
The first principal component condenses the correlated information in both the chicken and corn prices (perhaps the underlying cost of fuel) since it explains the most variance in the final table of corn and chicken prices.

+/- R Code



The second principal component is then related to the unique information seperating chicken from corn prices but neither indices directly themselves (maybe the cost of antibiotics)

+/- R Code



Applied PCA: Shape Tensor

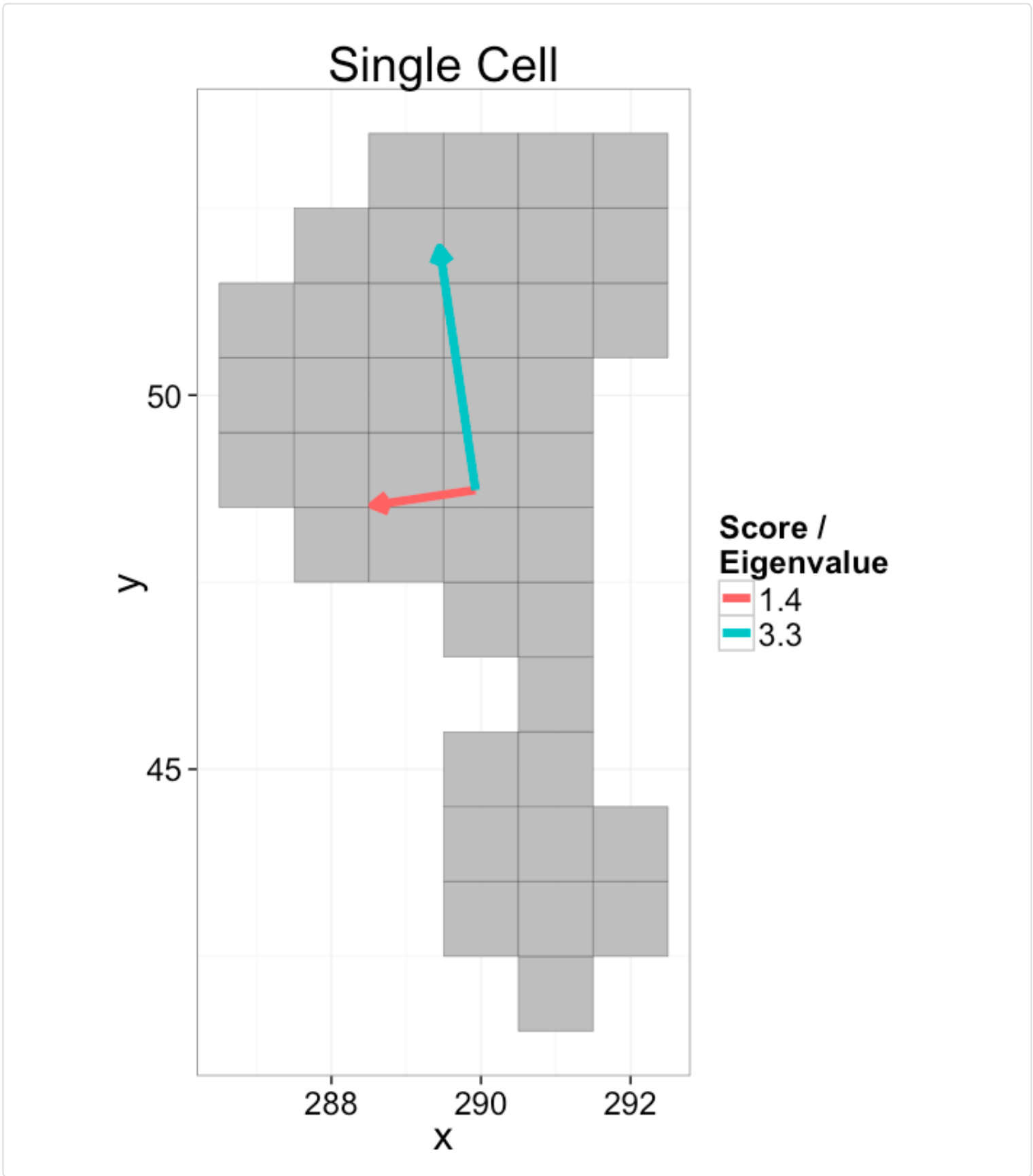
How do these statistical analyses help us?

Going back to a single cell, we have the a distribution of x and y values.

- are not however completely independent
- greatest variance does not lie in either x nor y itself.

A principal component analysis of the voxel positions, will calculate two new principal components (the components themselves are the relationships between the input variables and the scores are the final values.)

+/- R Code


Single Cell
50
45
y
288
290
292
x
Score /
Eigenvalue
1.4
3.3

How did we calculate that?

We start off by calculating the covariance matrix from the list of x, y, and z points that make up our object of interest.

$$COV(I_{id}) = \frac{1}{N} \sum_{\forall \vec{v} \in I_{id}} \begin{bmatrix} \vec{v}_x \vec{v}_x & \vec{v}_x \vec{v}_y & \vec{v}_x \vec{v}_z \\ \vec{v}_y \vec{v}_x & \vec{v}_y \vec{v}_y & \vec{v}_y \vec{v}_z \\ \vec{v}_z \vec{v}_x & \vec{v}_z \vec{v}_y & \vec{v}_z \vec{v}_z \end{bmatrix}$$

We then take the eigentransform of this array to obtain the eigenvectors (principal components, $\vec{\Lambda}_{1...3}$) and eigenvalues (scores, $\lambda_{1...3}$)

$$COV(I_{id}) \longrightarrow \underbrace{\begin{bmatrix} \vec{\Lambda}_{1x} & \vec{\Lambda}_{1y} & \vec{\Lambda}_{1z} \\ \vec{\Lambda}_{2x} & \vec{\Lambda}_{2y} & \vec{\Lambda}_{2z} \\ \vec{\Lambda}_{3x} & \vec{\Lambda}_{3y} & \vec{\Lambda}_{3z} \end{bmatrix}}_{\text{Eigenvectors}} * \underbrace{\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}}_{\text{Eigenvalues}} * \underbrace{\begin{bmatrix} \vec{\Lambda}_{1x} & \vec{\Lambda}_{1y} & \vec{\Lambda}_{1z} \\ \vec{\Lambda}_{2x} & \vec{\Lambda}_{2y} & \vec{\Lambda}_{2z} \\ \vec{\Lambda}_{3x} & \vec{\Lambda}_{3y} & \vec{\Lambda}_{3z} \end{bmatrix}^T}_{\text{Eigenvectors}}$$

The principal components tell us about the orientation of the object and the scores tell us about the corresponding magnitude (or length) in that direction.

Principal Component Analysis: Take home message

- We calculate the statistical distribution individually for x, y, and z and the 'correlations' between them.
- From these values we can estimate the orientation in the direction of largest variance
- We can also estimate magnitude
- These functions are implemented as `princomp` or `pca` in various languages and scale well to very large datasets.

Principal Component Analysis: Elliptical Model

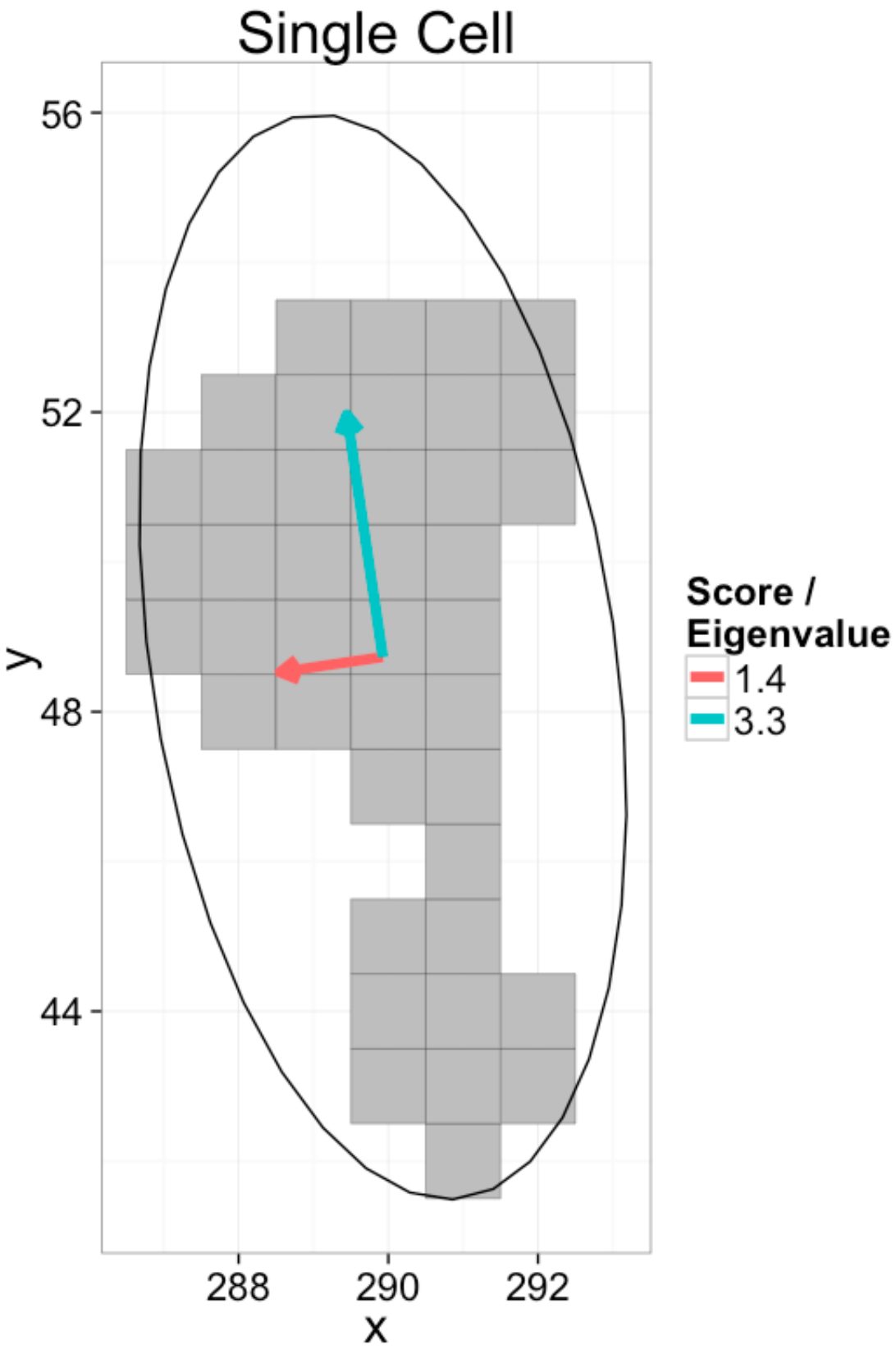
While the eigenvalues and eigenvectors are in their own right useful

- Not obvious how to visually represent these tensor objects
- Ellipsoidal (Ellipse in 2D) representation alleviates this issue

Ellipsoidal Representation

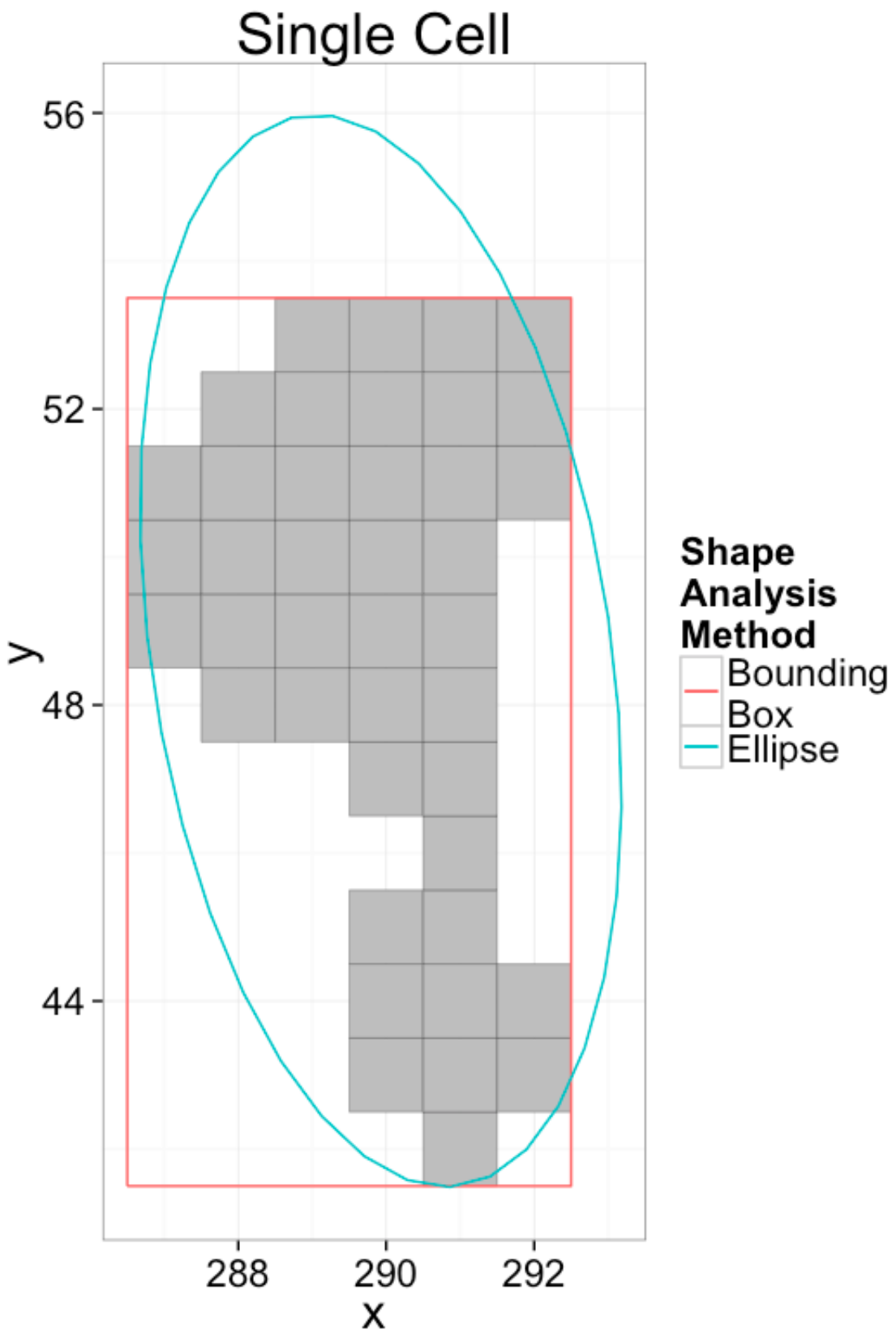
1. Center of Volume is calculated normally
2. Eigenvectors represent the unit vectors for the semiaxes of the ellipsoid
3. $\sqrt{\text{Eigenvalues}}$ is proportional to the length of the semiaxis ($l = \sqrt{5\lambda_i}$), derivation similar to moment of inertia tensor for ellipsoids.

+/- R Code


Single Cell
56
52
48
44
y
288
290
292
x
Score /
Eigenvalue
1.4
3.3

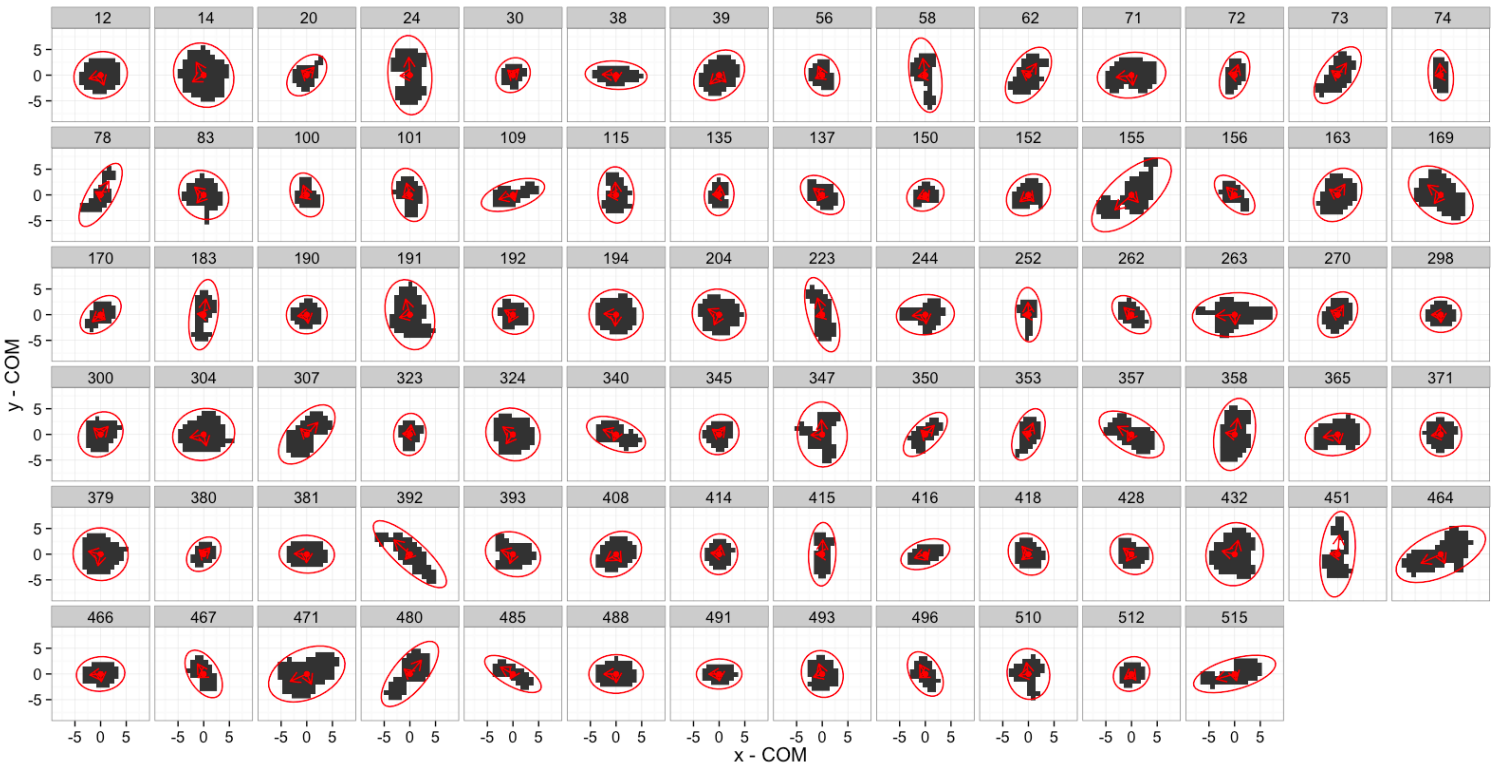
Elliptical Model vs Bounding Box

+/- R Code


Single Cell
56
52
48
44
y
288
290
292
x
Shape Analysis Method
Bounding Box
Ellipse

- The bounding box matches boundaries better
- The ellipse captures the dimensions and shape better
- Extents can be calculated from either method
- Anisotropy can also be calculated from either method

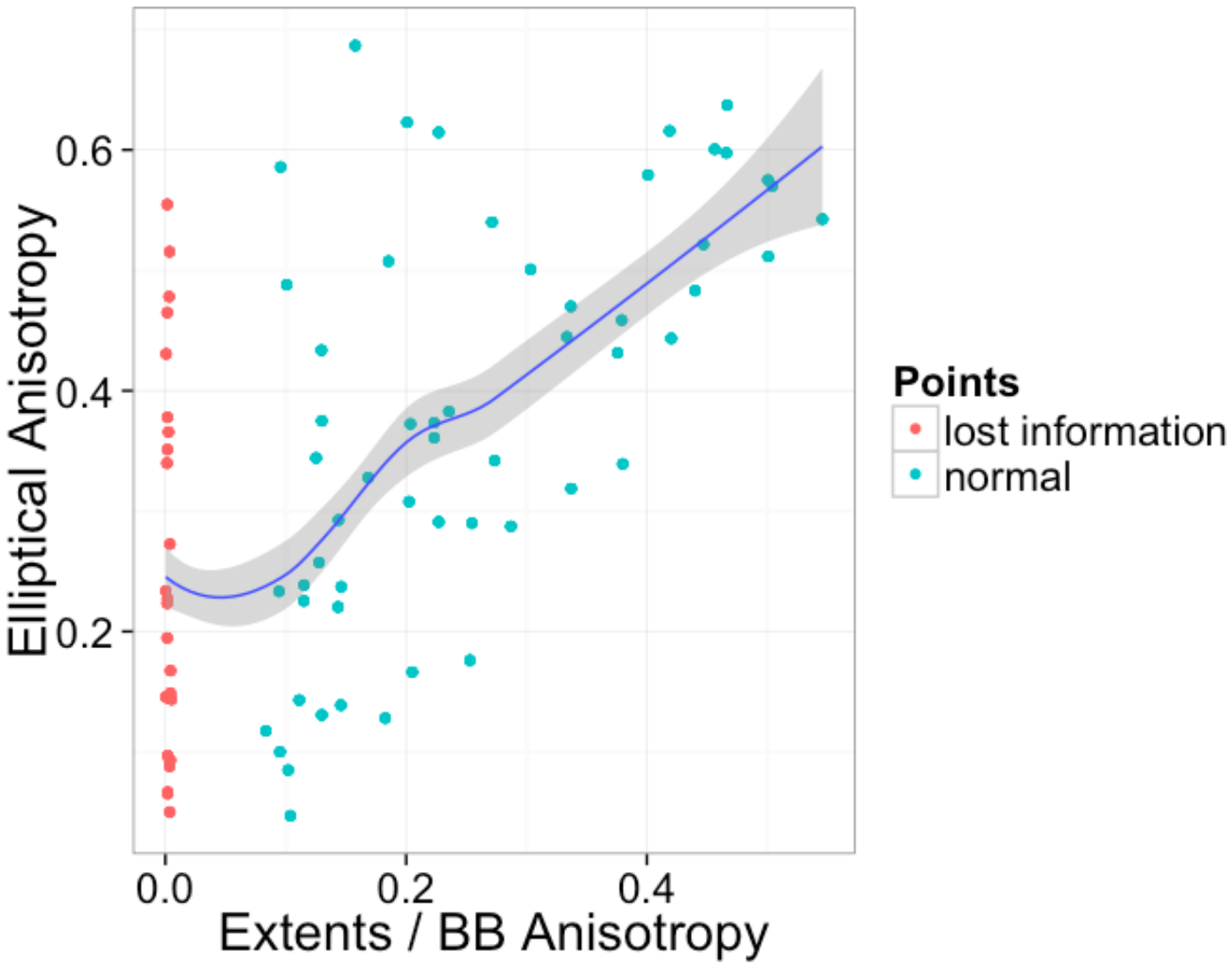
Elliptical Model for All Samples



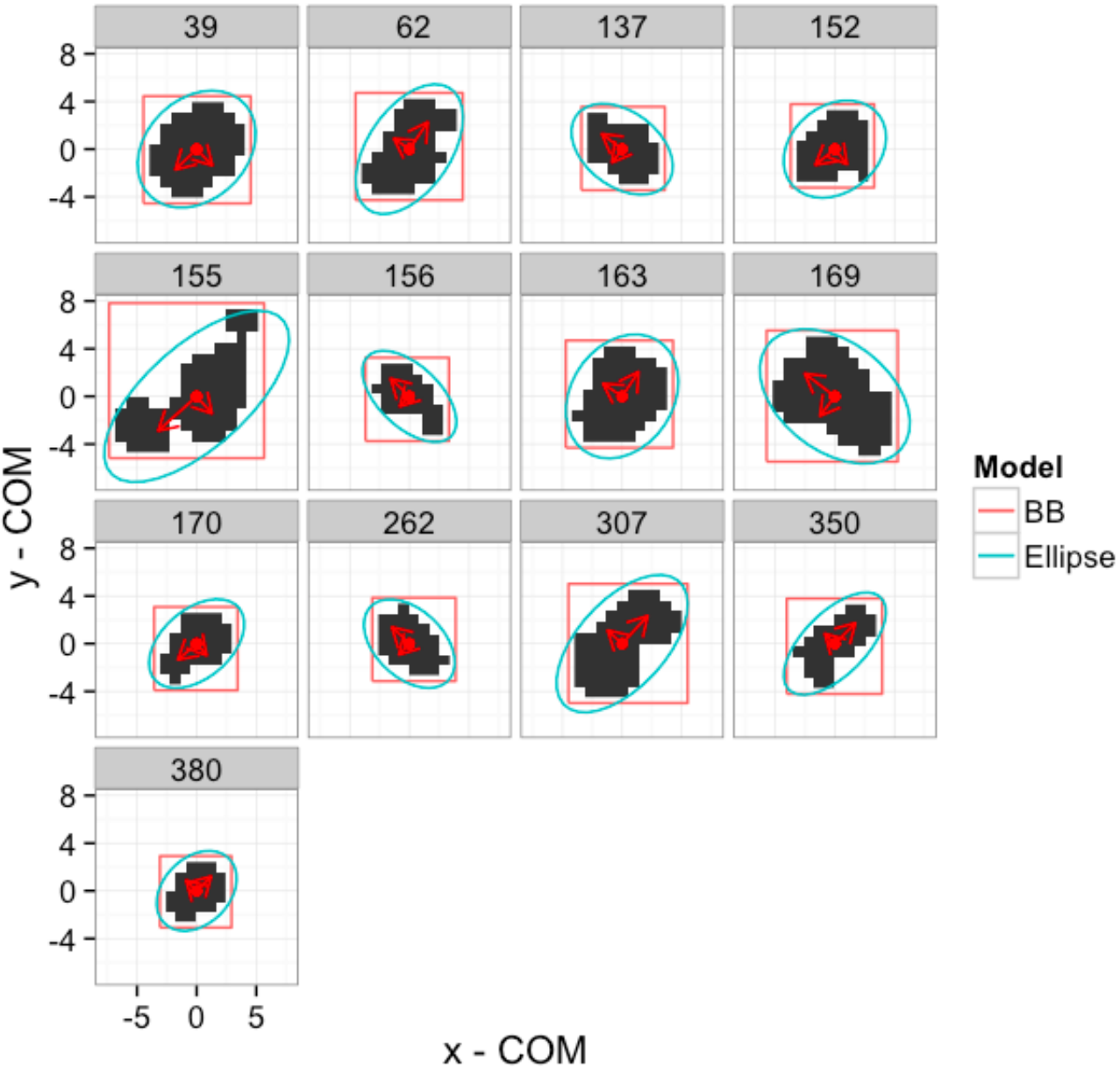
+/- R Code

Comparison between Anisotropy for both models

+/- R Code


Elliptical Anisotropy
0.6
0.4
0.2
0.0
0.2
0.4
Extents / BB Anisotropy
Points
lost information
normal

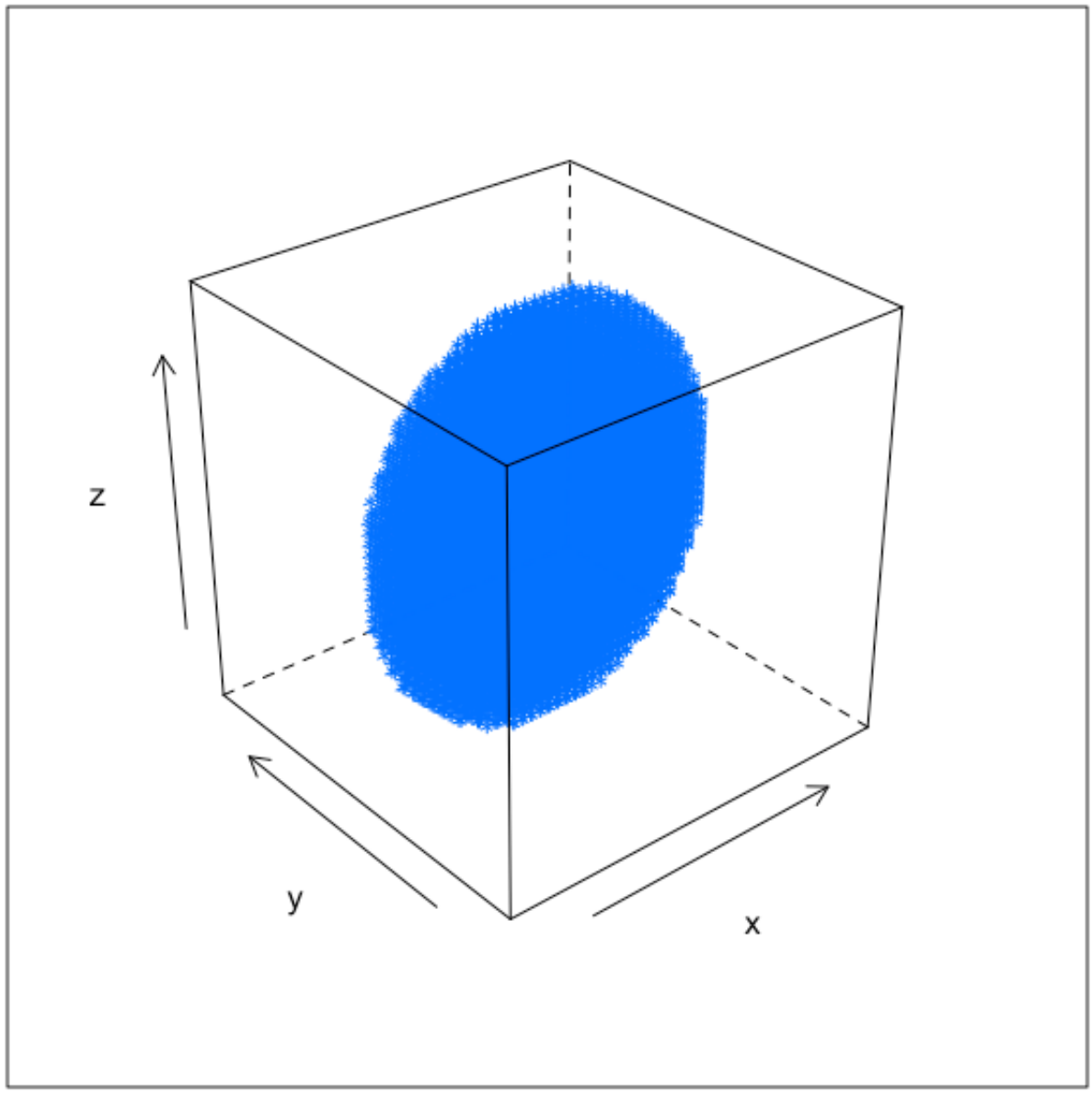
We see that there seems to be a general, albeit weak, correlation between the two measures. The most concerning portion is however the left side where the extents or bounding box method reports 0 anisotropy and the elliptical method reports substancial amounts of it.



3D Shape Analysis

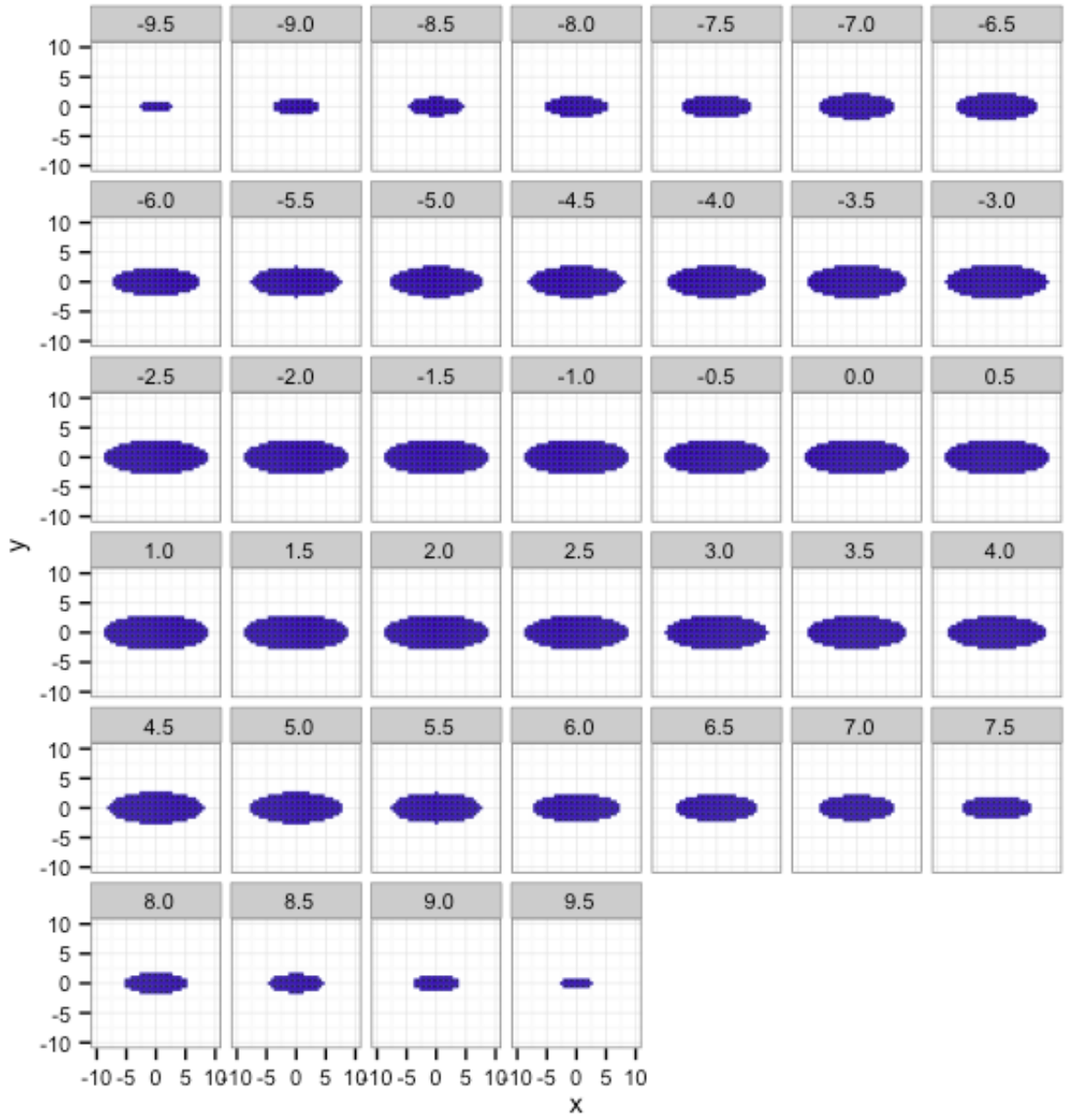
The models we have done are all applicable to both 2D and 3D images. The primary difference is when looking at 3D images there is an extra dimension to consider.

+/- R Code



It can also be shown as a series of 2D slices.

+/- R Code

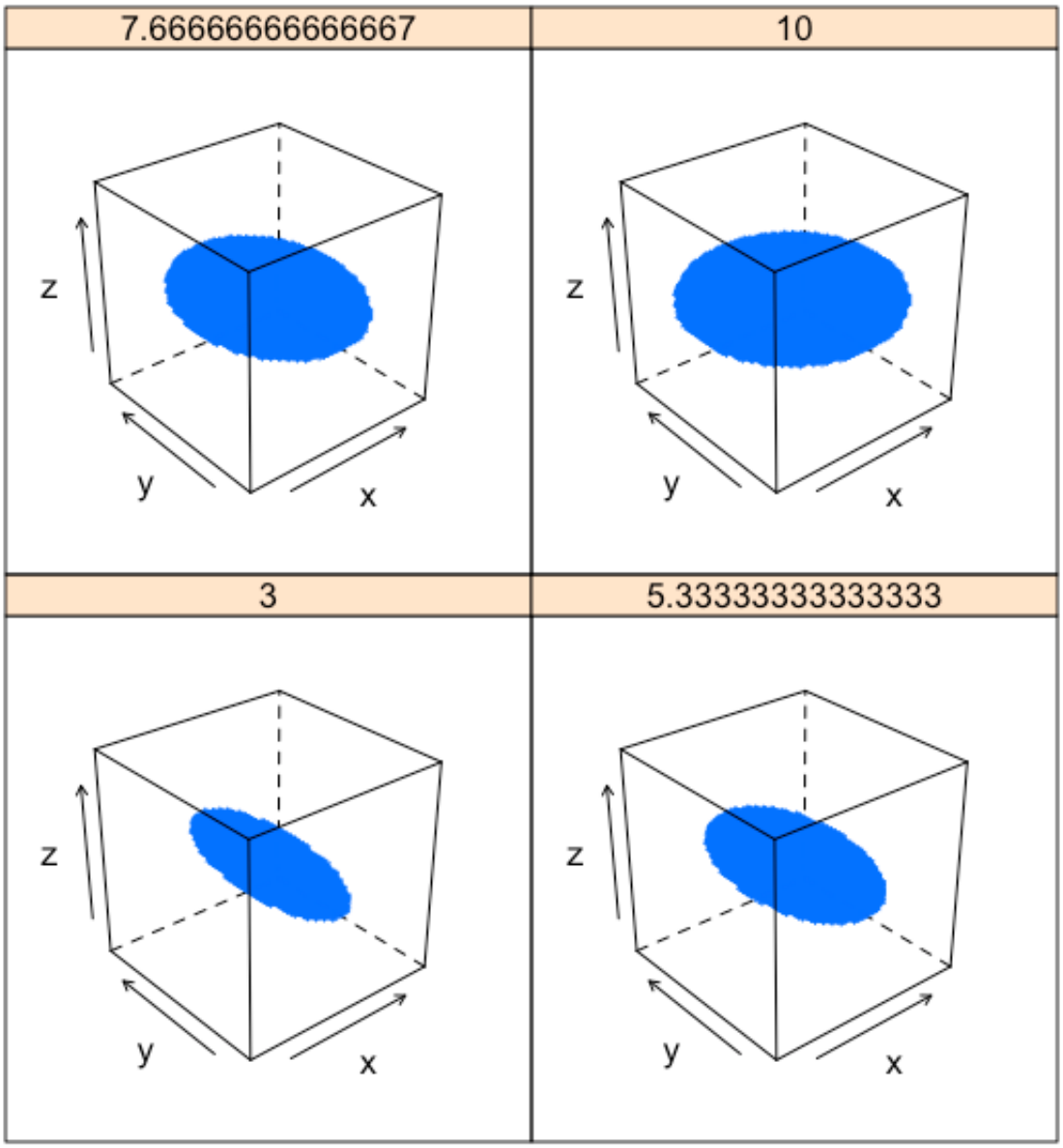


Anisotropy applies to these samples as well but it only gives us information about the shortest and the longest dimensions. Is that enough?

Pancakeness

Each of these images has the exact same anisotropy because the shortest semiaxis length remains

+/- R Code



If we take a slice through the image we can image the final shape looking like a very small thin rod in the case where the second dimension is equal to 3 (short in two directions and long in the other) and a pankcake (short in one direction and long in the other two).

+/- R Code

Oblateness

We can thus introduce a new metric for assessing the second-degree anisotropy in the object and this we shall somewhat more formally call **Oblateness**.

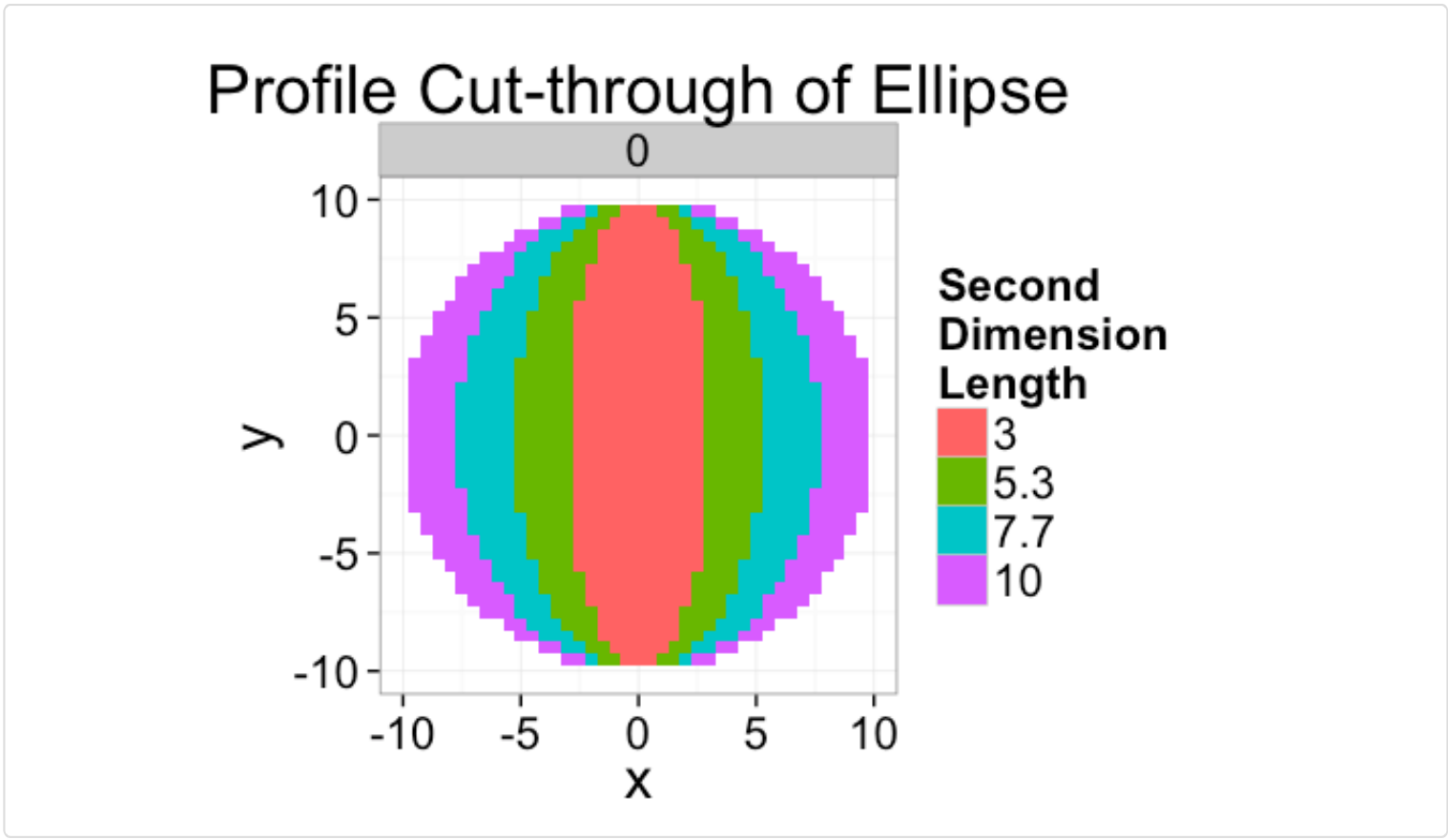
$$\mathrm{Ob} = 2\frac{\lambda_2 - \lambda_1}{\lambda_3 - \lambda_1} - 1$$

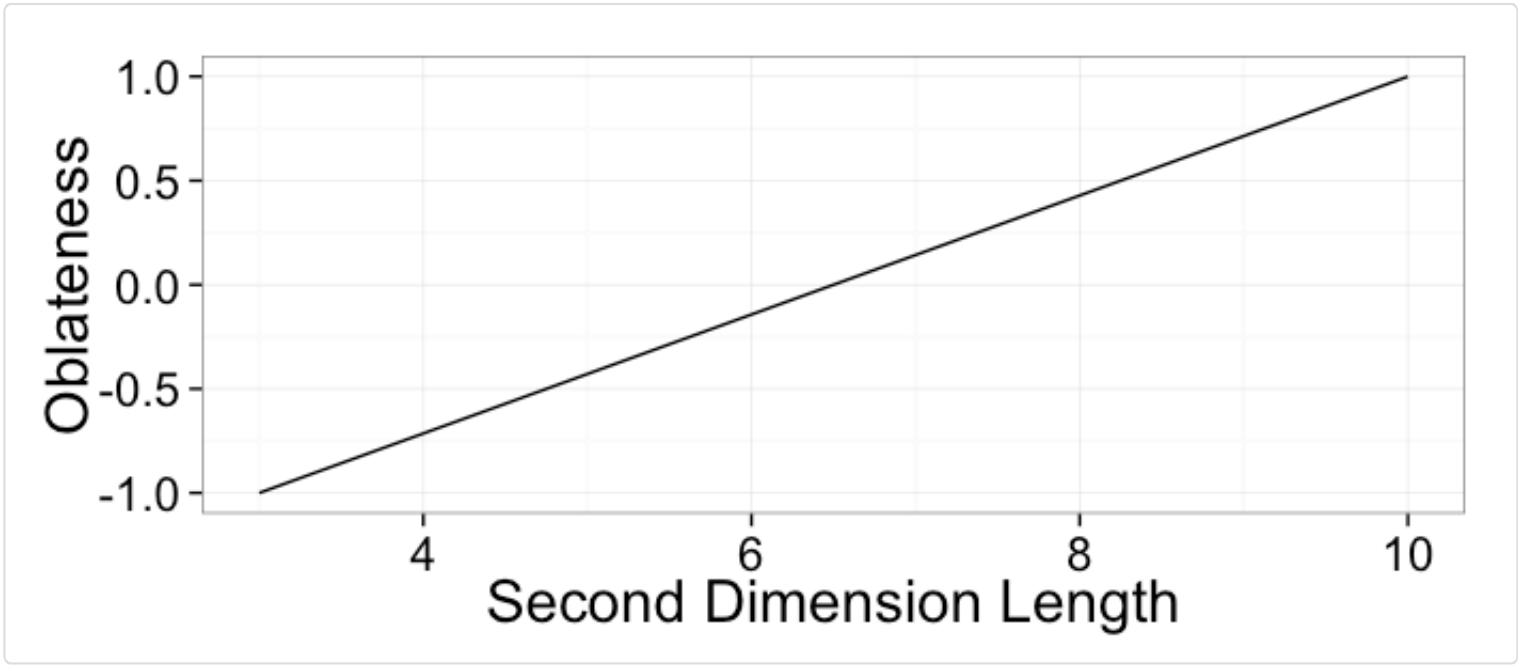
The value like in anisotropy is bound between -1 and 1.

- -1 indicates the middle semiaxis is the same length of the shortest semiaxis $\lambda_2 = \lambda_1$ and thus the structure is

rod-like → **prolate**

- +1 indicates the middle axis is the same length as the longest semiaxis $\lambda_2 = \lambda_3$ and thus the structure is pancake-like → **oblate**





Interfaces / Surfaces

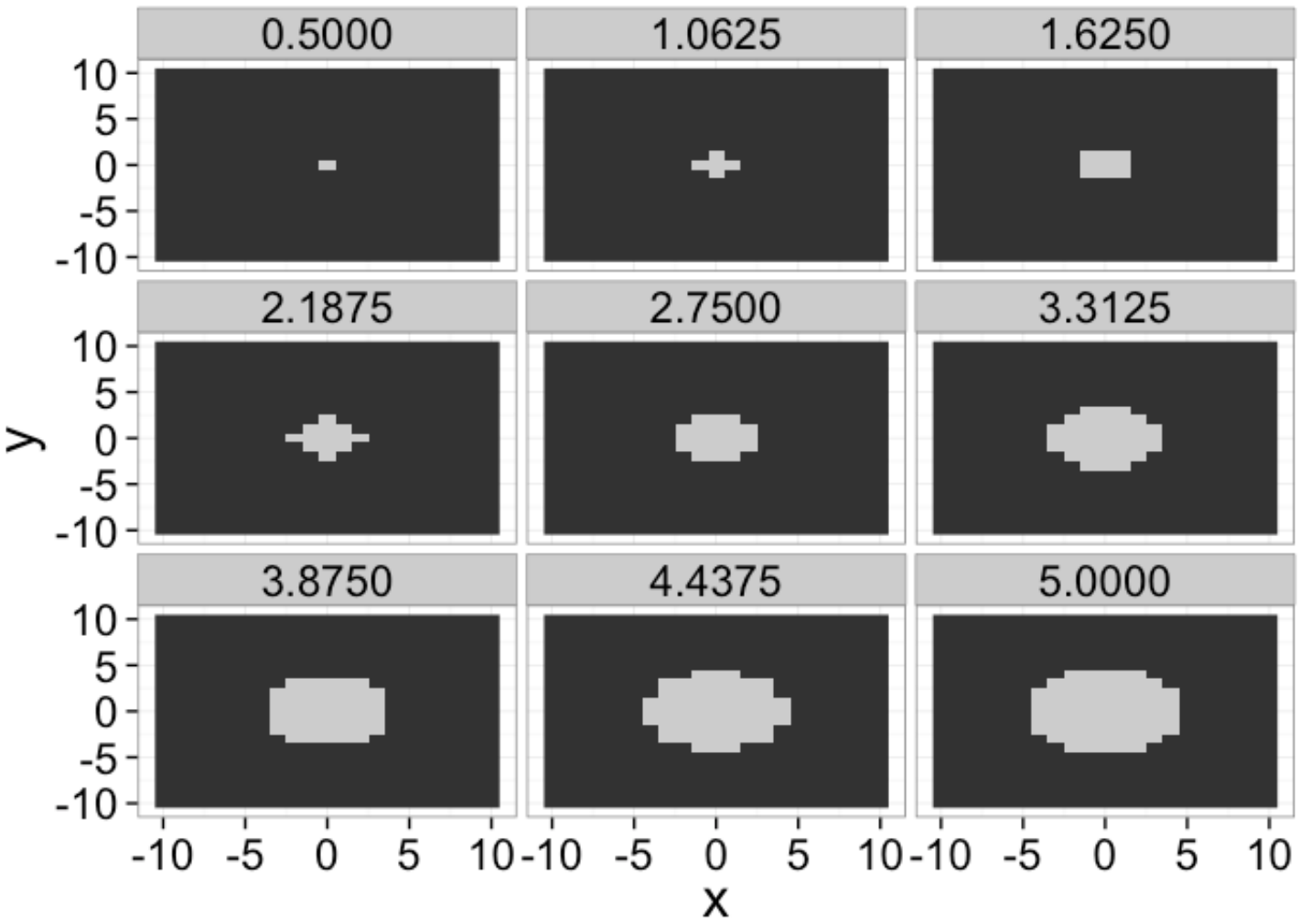
Many physical and chemical processes occur at surfaces and interfaces and consequently these structures are important in material science and biology. For this lecture surface and interface will be used interchangebly and refers to the boundary between two different materials (calcified bone and soft tissue, magma and water, liquid and gas) Through segmentation we have identified the unique phases in the sample under investigation.

- Segmentation identifying volumes (3D) or areas (2D)
- Interfaces are one dimension lower corresponding to surface area (3D) or perimeter (2D)
- Interfaces are important for
    - connectivity of cell networks, particularly neurons
    - material science processes like coarsening or rheological behavior
    - chemical processes (surface-bound diffusion, catalyst action)

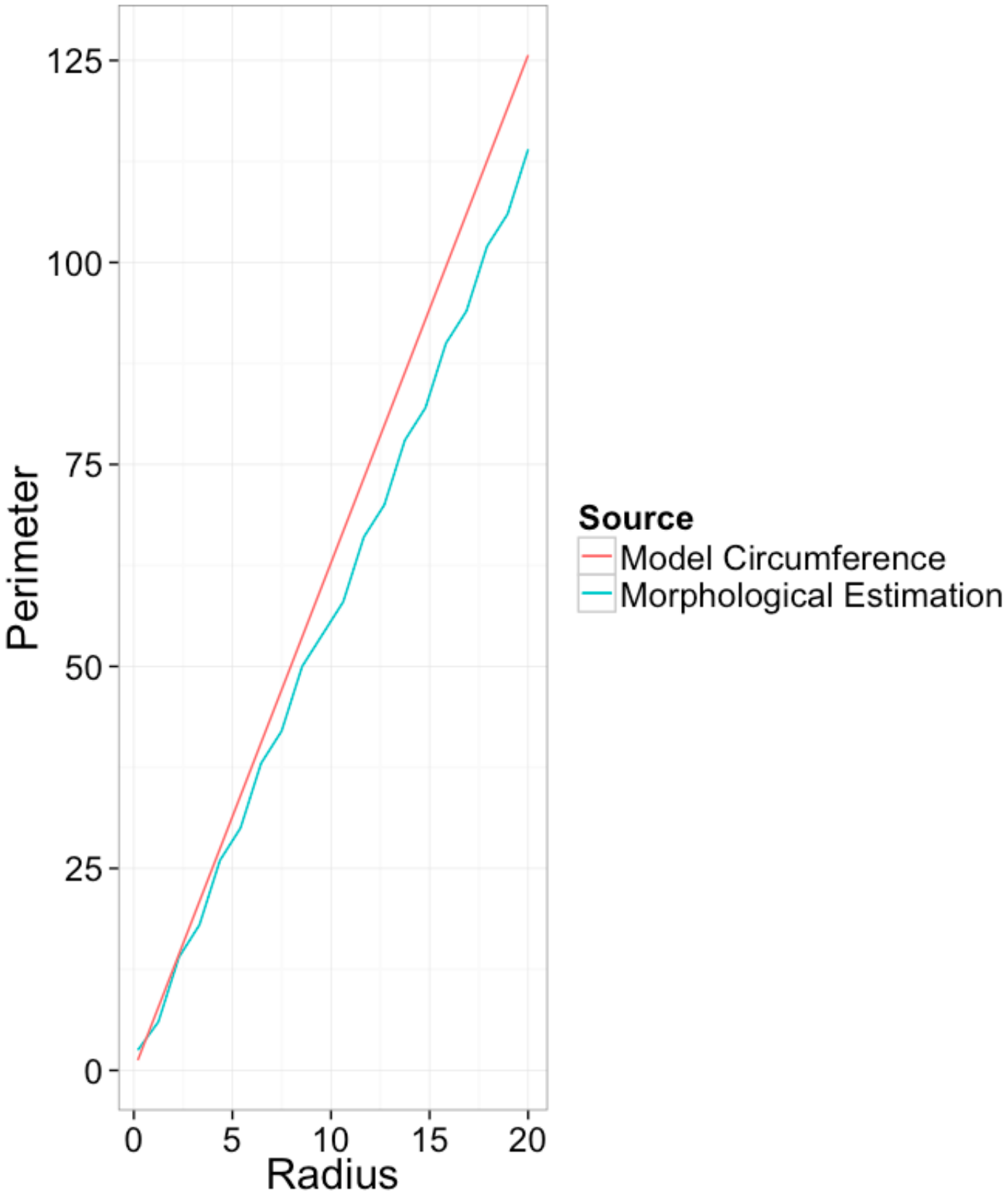
Surface Area / Perimeter

We see that the dilation and erosion affects are strongly related to the surface area of an object: the more surface area the larger the affect of a single dilation or erosion step.

+/- R Code


0.5000
1.0625
1.6250
2.1875
2.7500
3.3125
3.8750
4.4375
5.0000
y
x
10
5
0
-5
-10

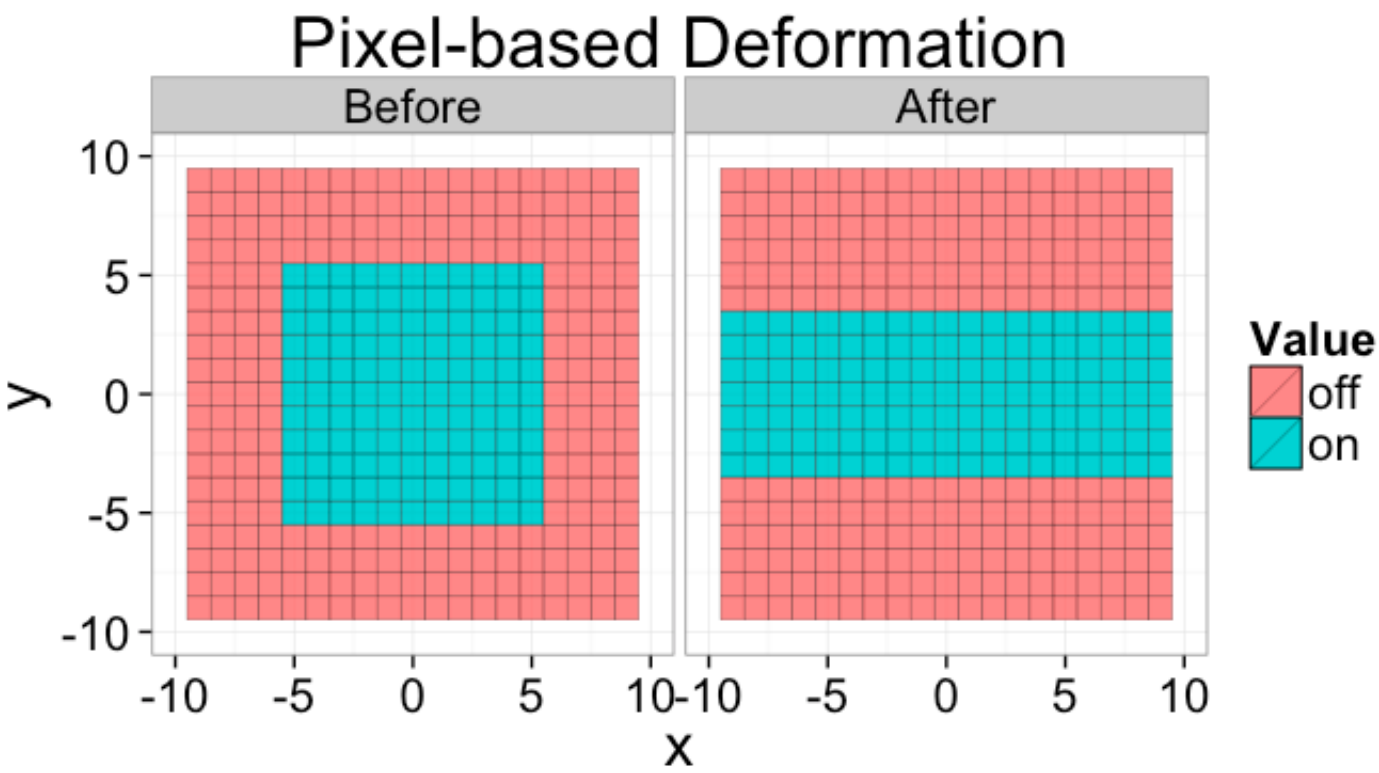
+/- R Code


125
100
75
50
25
0
Perimeter
0
5
10
15
20
Radius
Source
Model Circumference
Morphological Estimation

Meshing

Constructing a mesh for an image provides very different information than the image data itself. Most crucially this comes when looking at physical processes like deformation.

While the images are helpful for visualizing we rarely have models for quantifying how difficult it is to turn a pixel **off**

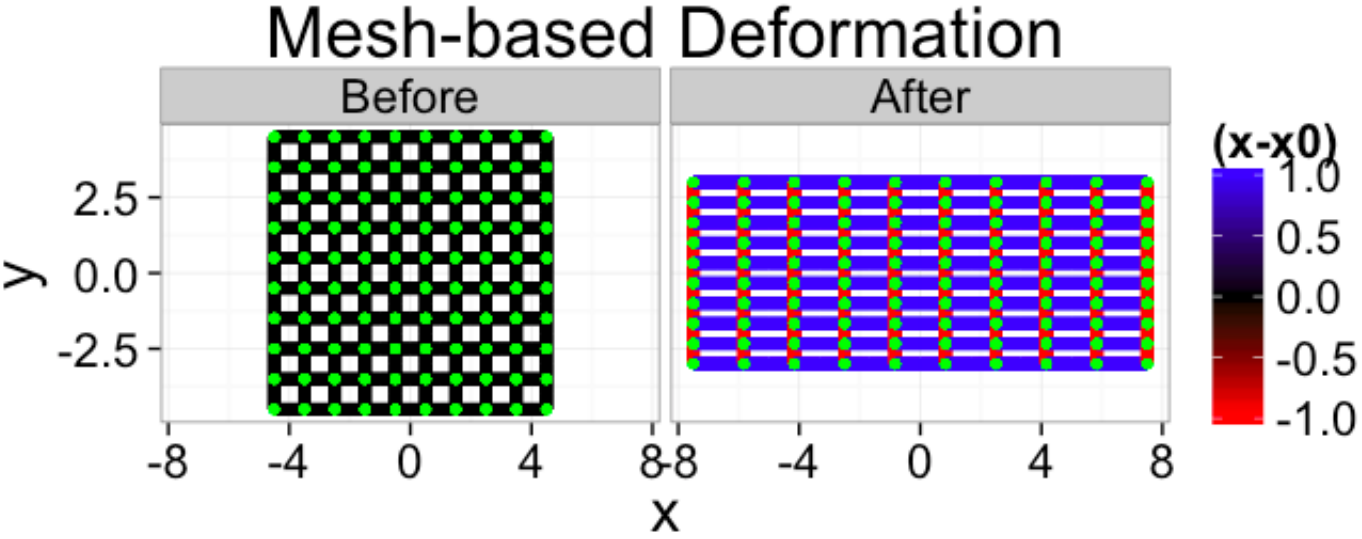


If the image is turned into a mesh we now have a list of vertices and edges. For these vertices and edges we can define forces. For example when looking at stress-strain relationships in mechanics using Hooke's Model

$$\vec{F} = k(\vec{x}_0 - \vec{x})$$

the force needed to stretch one of these edges is proportional to how far it is stretched.

+/- R Code



Meshing

Since we uses voxels to image and identify the volume we can use the voxels themselves as an approimation for the surface of the structure.

- Each 'exposed' face of a voxel belongs to the surface

From this we can create a mesh by

- adding each exposed voxel face to a list of surface squares.
- adding connectivity information for the different squares (shared edges and vertices)

A wide variety of methods of which we will only graze the surface (http://en.wikipedia.org/wiki/Image-based_meshing (http://en.wikipedia.org/wiki/Image-based_meshing))

Marching Cubes

Why

Voxels are very poor approximations for the surface and are very rough (they are either normal to the x, y, or z axis and nothing between). Because of their inherently orthogonal surface normals, any analysis which utilizes the surface normal to calculate another value (growth, curvature, etc) is going to be very inaccurate at best and very wrong at worst.

How (http://en.wikipedia.org/wiki/Marching_cubes)

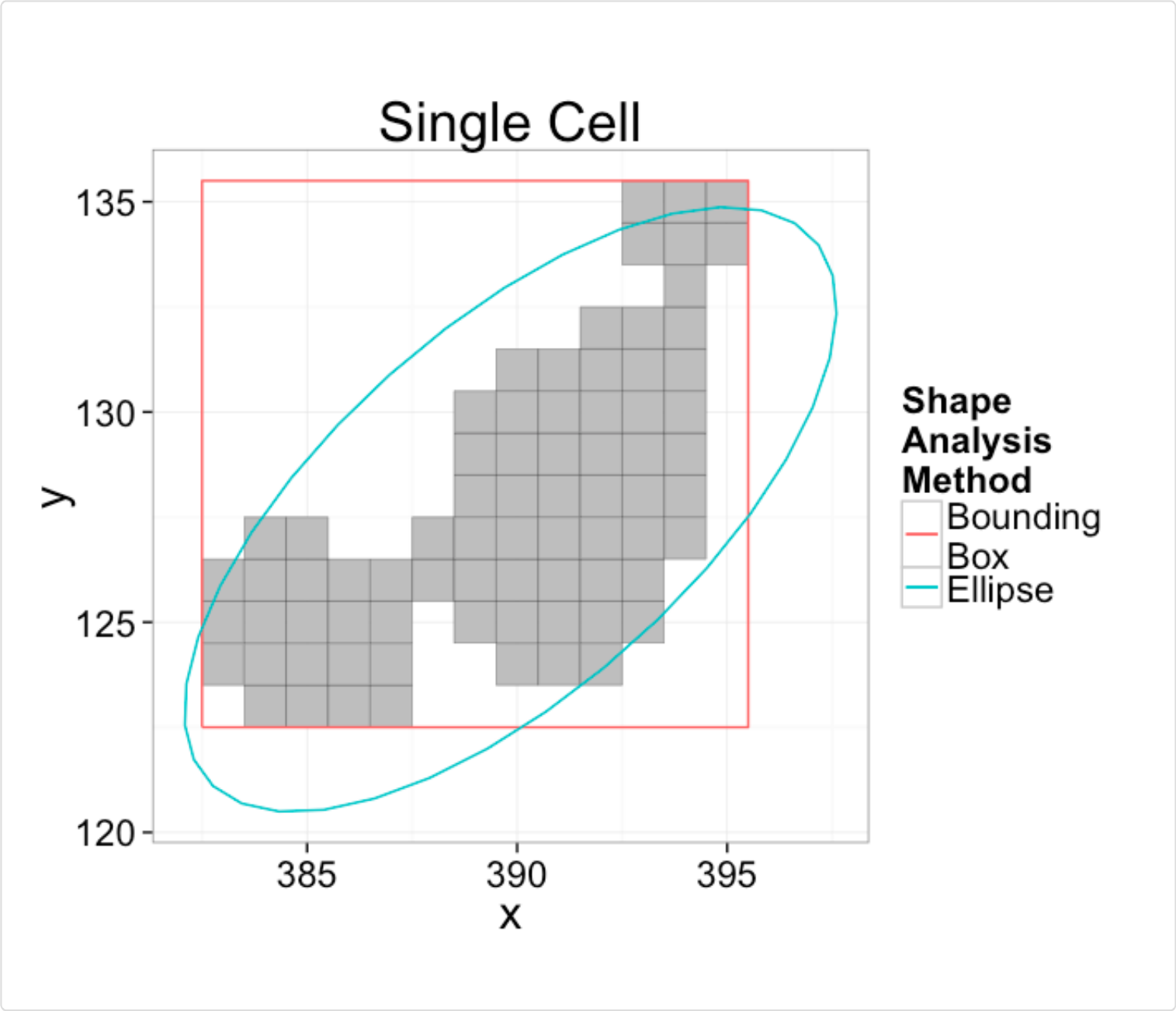
The image is processed one voxel at a time and the neighborhood (not quite the same is the morphological definition) is checked at every voxel. From this configuration of values, faces are added to the mesh to incorporate the most simple surface which would explain the values.

Marching tetrahedra (http://en.wikipedia.org/wiki/Marching_tetrahedra) is for some applications a better suited approach

Next Time on QBI

So while bounding box and ellipse-based models are useful for many object and cells, they do a very poor job with the sample below.

+/- R Code



Why

- We assume an entity consists of connected pixels (wrong)
- We assume the objects are well modeled by an ellipse (also wrong)

What to do?

- Is it 3 connected objects which should all be analzed seperately?
- If we could **divide it**, we could then analyze each spart as an ellipse
- Is it one network of objects and we want to know about the constrictions?

- Is it a cell or organelle with docking sites for cell?
- Neither extents nor anisotropy are very meaningful, we need a **more specific metric** which can characterize