Image Data

Susan Holmes

Lecture 11 of Stats 366 / Bios 221

Stanford University, July 2019

What You Will Learn in Today's Lecture

- **Read**, **write**, **transform** images in `R`
- Apply **filters** and **morphological operations**
- Cell **segmentation**
- **Extract features** from segmented cells
- Analyze **spatial** patterns using spatial point processes
- Align images using **registration**
- We will look at many **case studies** from the real world
- What is different about **Spatial Data Analysis**?

The EBImage Package

Images Are Just Data

A useful toolkit for handling images in R is the Bioconductor package **EBImage** (Pau et al. 2010)

```
library("EBImage")
```

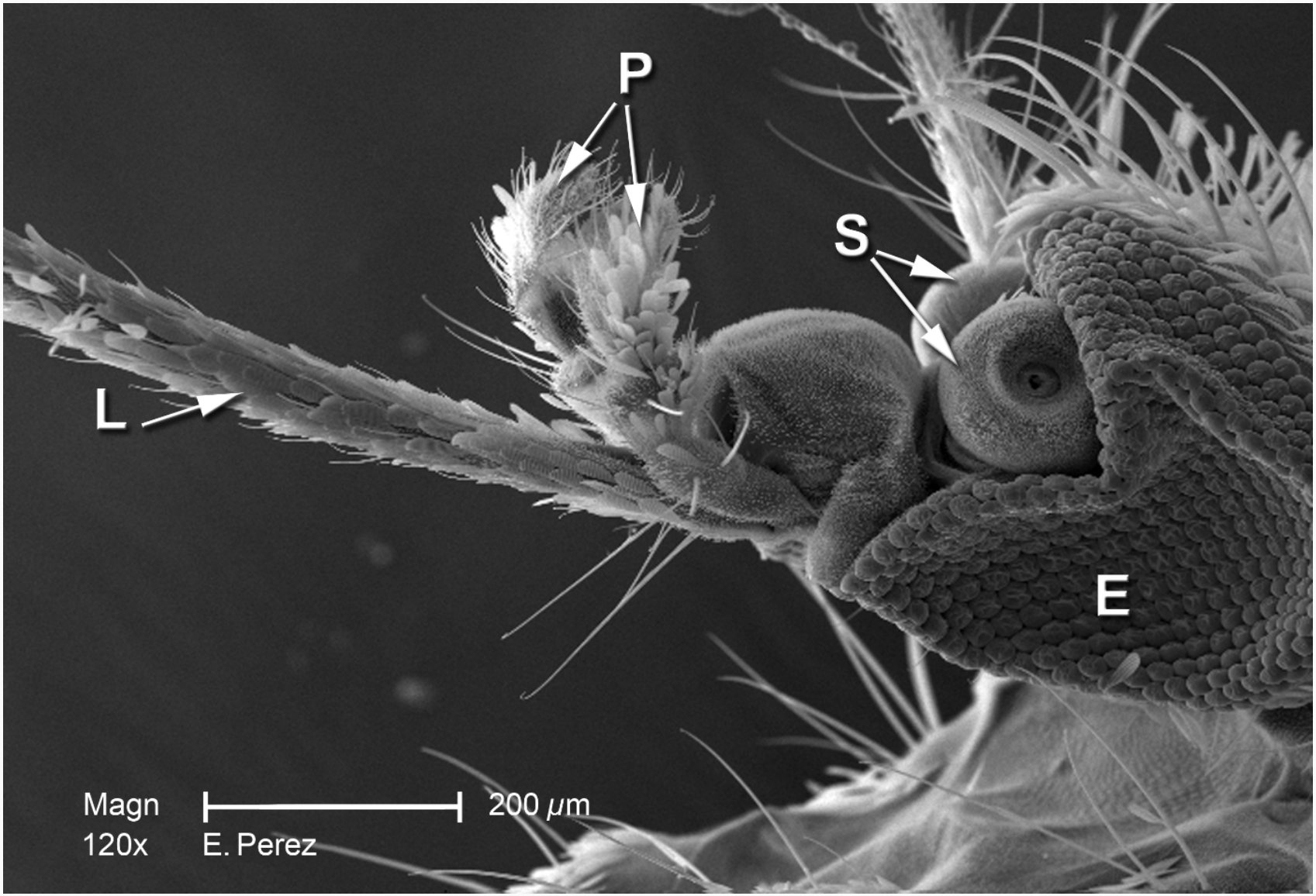
Read Image

We start out by reading in a simple picture to demonstrate the basic functions

```
imagefileloc="http://bios221.stanford.edu/data/mosquito.png
mosq = readImage(imagefileloc)
```

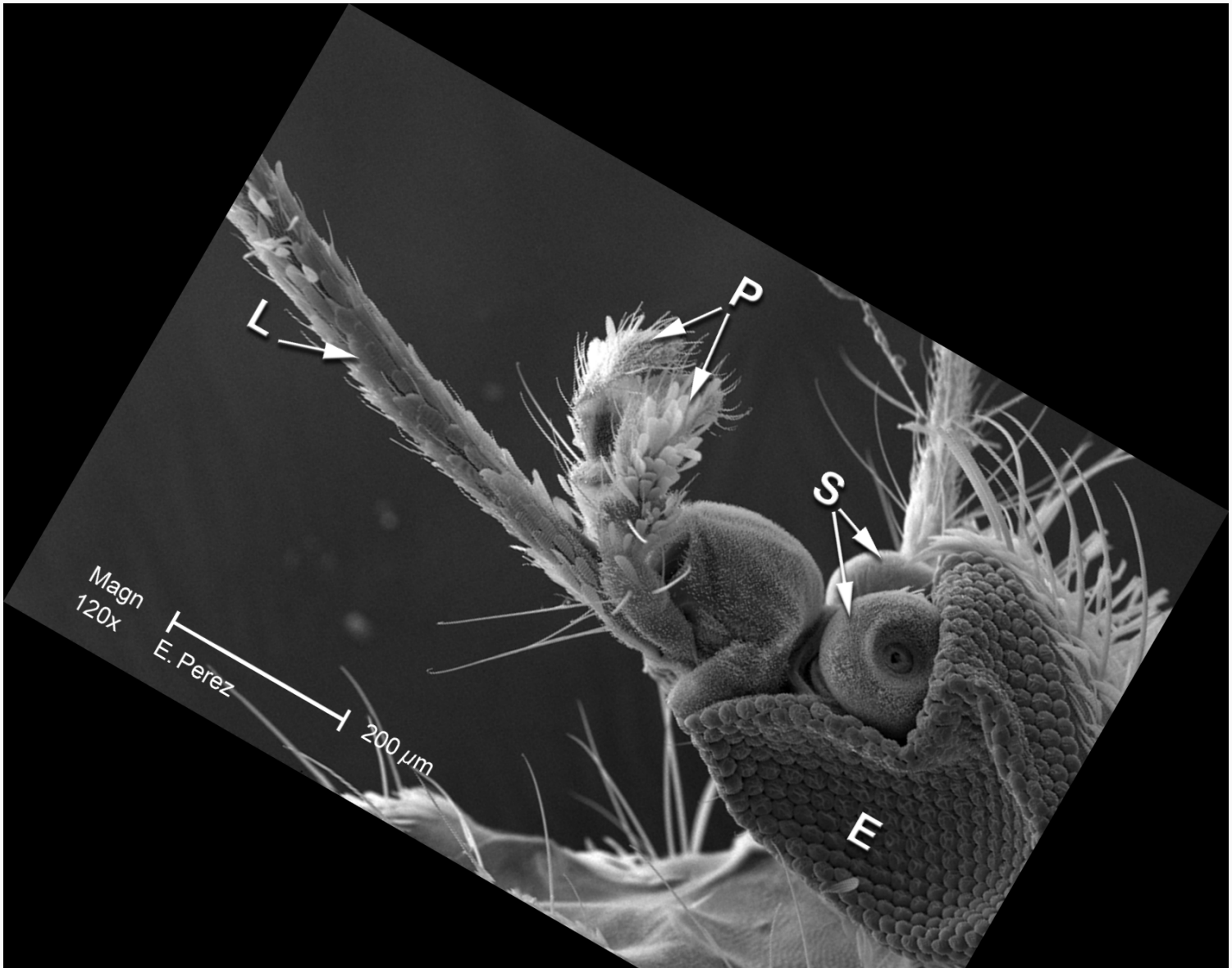
Display Image

```
display(mosq)
```



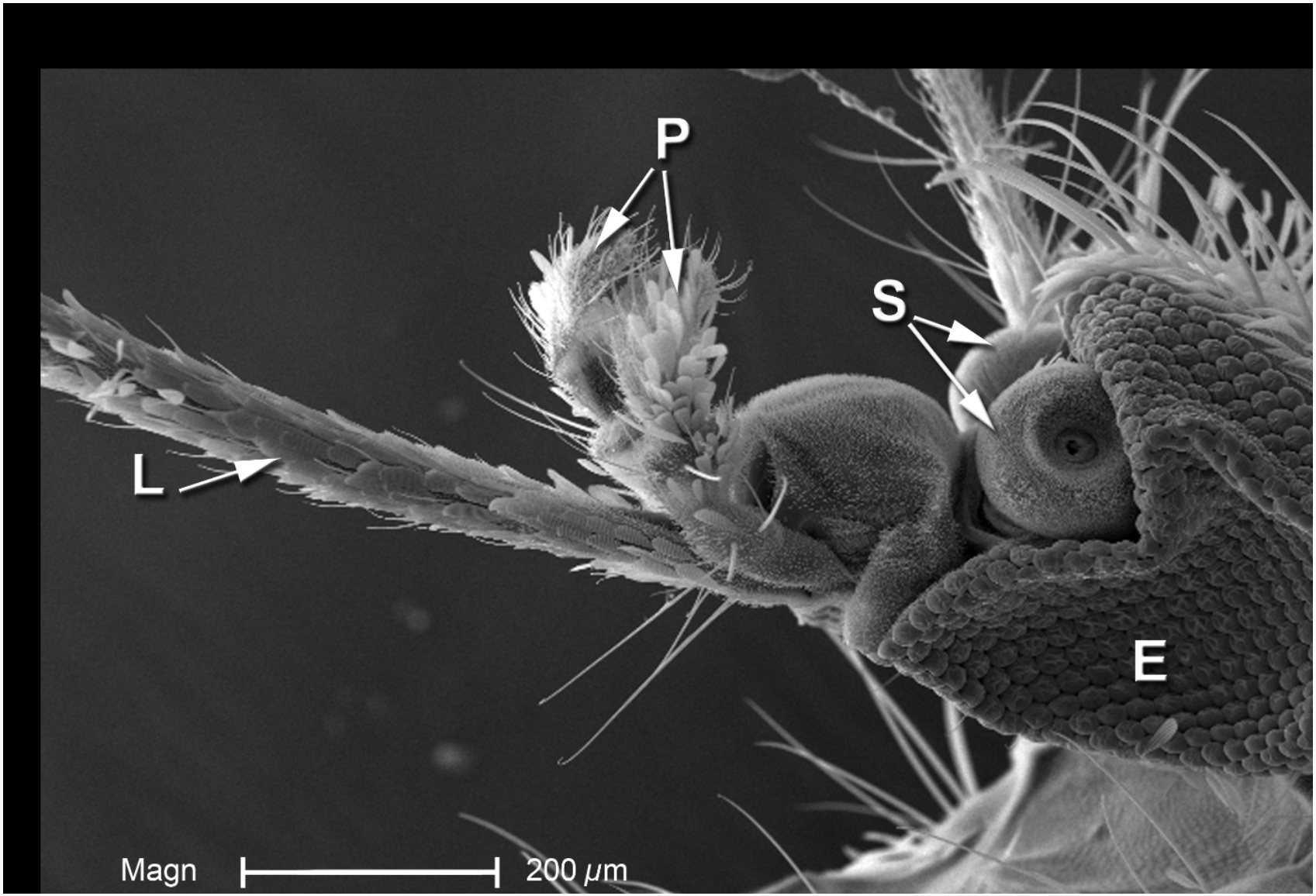
Tranform Image

```
EBImage::rotate(mosq, angle = 30) %>% display()
```



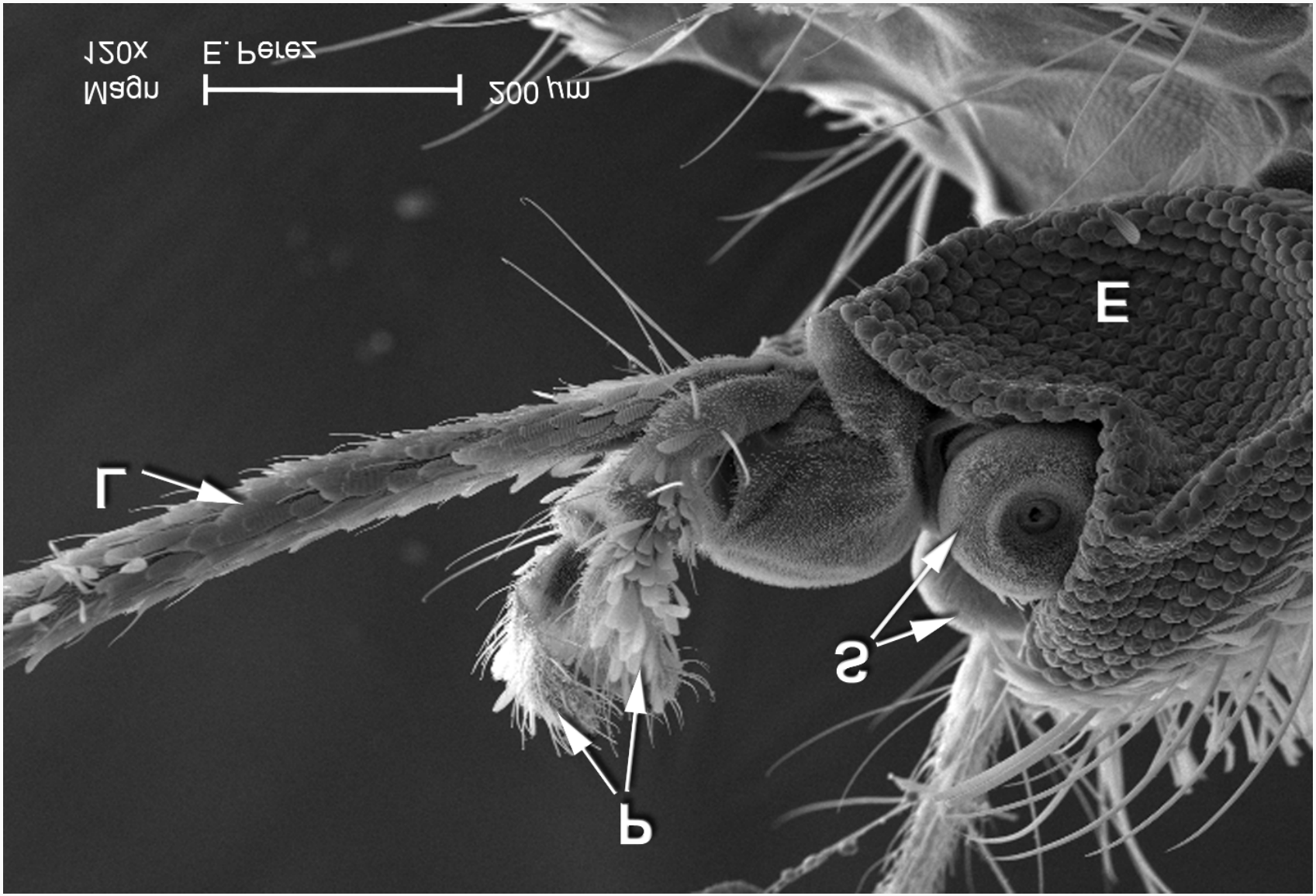
Tranform Image

```
translate(mosq, v = c(40, 70)) %>% display()
```



Tranform Image

```
flip(mosq) %>% display()
```



How Are Images Stored in R?

```
class(mosq)
```

```
## [1] "Image"
## attr(,"package")
## [1] "EBImage"
```

```
dim(mosq)
```

```
## [1] 1400  952
```

```
str(mosq)
```

```
## Formal class 'Image' [package "EBImage"] with 2 slots
##   ..@ .Data    : num [1:1400, 1:952] 0.196 0.196 0.196 0.196 0.196 .
##   ..@ colormode: int 0
```

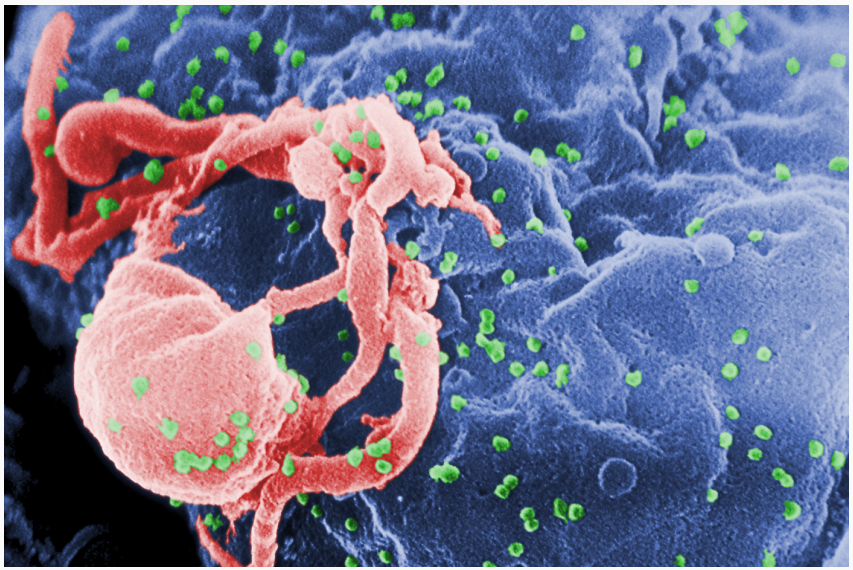
Print Image Object

```
print(mosq)
```

```
## Image
##   colorMode    : Grayscale
##   storage.mode : double
##   dim          : 1400 952
##   frames.total : 1
##   frames.render: 1
##
## imageData(object)[1:5,1:6]
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1960784 0.1960784 0.1960784 0.1960784 0.1960784 0.1960784
## [2,] 0.1960784 0.1960784 0.1960784 0.1960784 0.1960784 0.1960784
## [3,] 0.1960784 0.1960784 0.2000000 0.2039216 0.2000000 0.1960784
## [4,] 0.1960784 0.1960784 0.2039216 0.2078431 0.2000000 0.1960784
## [5,] 0.1960784 0.2000000 0.2117647 0.2156863 0.2000000 0.1921569
```

Load and Display Color Image

```
imagefileloc="http://bios221.stanford.edu/data/hiv.png"
hivc = readImage(imagefileloc)
display(hivc, method = "raster")
```



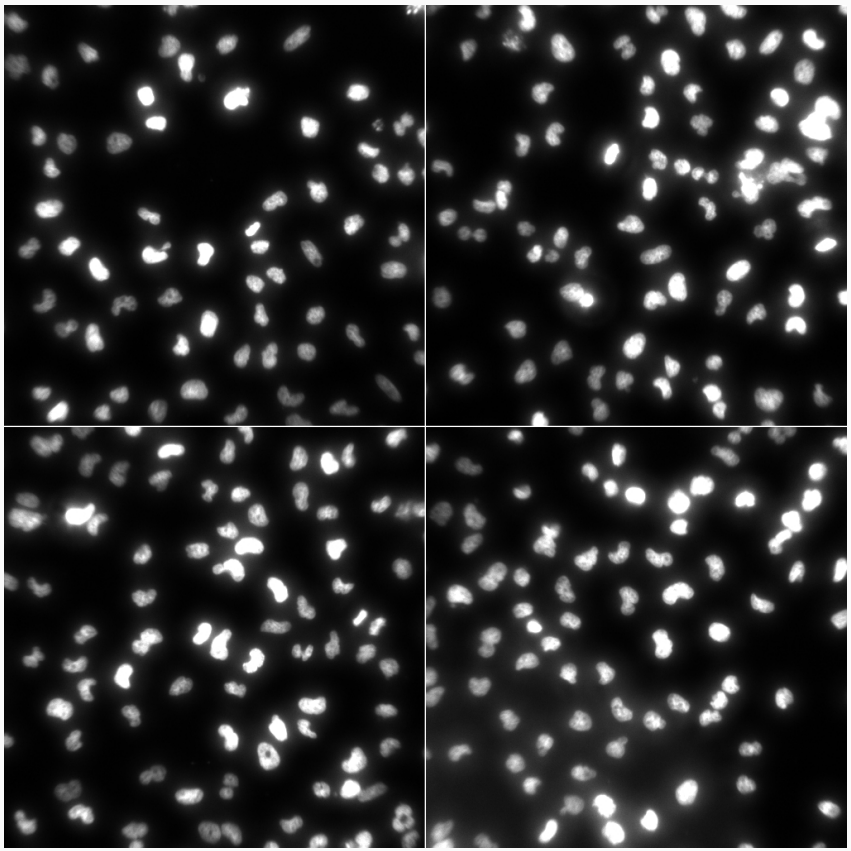
Print Image Object

```
print(hivc)
```

```
## Image
##   colorMode    : Color
##   storage.mode : double
##   dim          : 1400 930 3
##   frames.total : 3
##   frames.render: 1
##
## imageData(object)[1:5,1:6,1]
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    0    0    0    0    0    0
## [3,]    0    0    0    0    0    0
## [4,]    0    0    0    0    0    0
## [5,]    0    0    0    0    0    0
```

Load and Display Multple Images

```
imagefile = system.file("images", "nuclei.tif",
                        package = "EBImage")
nuc = readImage(imagefile)
display(nuc, method = "raster", all = TRUE)
```



Print Image Object

```
print(nuc)
```

```
## Image
##   colorMode    : Grayscale
##   storage.mode : double
##   dim          : 510 510 4
##   frames.total : 4
##   frames.render: 4
##
## imageData(object)[1:5,1:6,1]
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,
## [1,] 0.06274510 0.07450980 0.07058824 0.08235294 0.10588235 0.098039
## [2,] 0.06274510 0.05882353 0.07843137 0.09019608 0.09019608 0.105882
## [3,] 0.06666667 0.06666667 0.08235294 0.07843137 0.09411765 0.094117
## [4,] 0.06666667 0.06666667 0.07058824 0.08627451 0.08627451 0.098039
## [5,] 0.05882353 0.06666667 0.07058824 0.08235294 0.09411765 0.105882
```

Object Oriented Programming in R

- Two most common systems for object oriented programming in R:
  - **S3**: Heavily used in packages on **CRAN**
  - **S4**: Heavily used in **Bioconductor** (in BioC 3.3: 3158 classes and 22511 methods defined in 609 out of 1211 packages)
- As a user it can be useful to recognize S3/S4 objects, explore, manipulate and use the help system
- Years of experience in Bioconductor have shown that S4 classes allow to construct rich and complicated data representations
- Let's compare S3 and S4 classes

S3 Classes

```
df = data.frame(y = rnorm(10), x = rnorm(10))
lm_object = lm(y ~ x, data = df)
class(lm_object)
```

```
## [1] "lm"
```

```
lm_object
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)            x
##      0.4844      -0.1081
```

S3 Classes

```
is.list(lm_object)
```

```
## [1] TRUE
```

```
attributes(lm_object)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"
```

S3 Classes

- In standard R, an S3 object is a list with a class attribute

```
xx = list(a = letters[1:3], b = rnorm(3))
xx
```

```
## $a
## [1] "a" "b" "c"
##
## $b
## [1] -0.3617687  0.8276866  0.8259594
```

S3 Classes

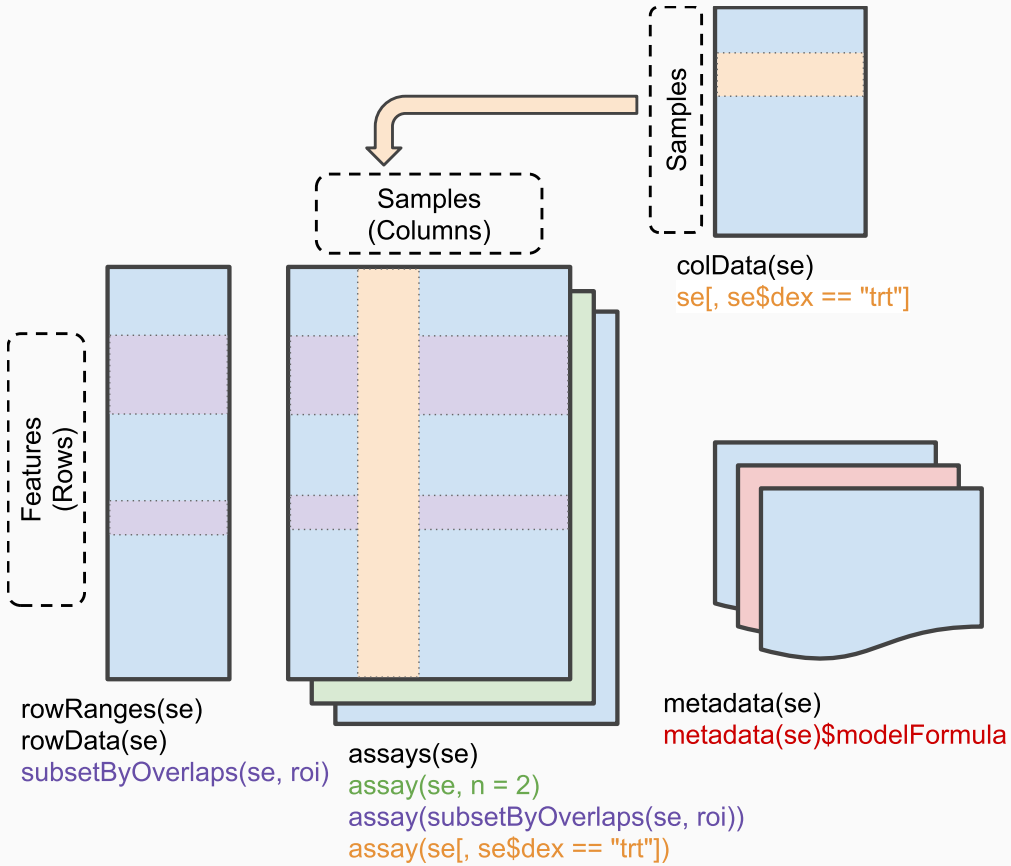
- The problem is that we can assign any class to any list

```
xx = list(a = letters[1:3], b = rnorm(3))
class(xx) = "lm"
xx
```

```
##
## Call:
## NULL
##
## No coefficients
```

S4 Classes

- S4 classes have a formal definition and formal validity checking
- To the end user, this guarantees validity of the object
- Example: SummarizedExperiment will check dimensions of features table, samples table, and assay matrices



S3 vs. S4

- S3 implements a style of OO programming called **generic-function** OO
- This is different from most programming languages, like Java, C++, and C#, which implement **message-passing** OO
- Message-passing:
  - Messages (methods) are sent to objects and the object determines which function to call
  - Object has a special appearance in the method call, usually appearing **before** the name of the method/message: e.g., canvas.drawRect("blue")
- Generic-function:
  - Computations are still carried out via methods
  - Special type of function called a generic function decides which method to call, e.g., drawRect(canvas, "blue")
  - Very casual system, no formal definition of classes

S3 vs. S4

- S4 works similarly to S3, but is more formal
- From a user perspective, there is **one major difference** to S3
  1. Fields of an S4 object are not attributes or named elements, but instead are called slots and are accessed with the special `@` or `slot()` operator (instead of `$` and `[[`)

Image Processing Tools:

Compression, Linear Filters, Morphological Operations, Voronoi Tessellation, Feature Extraction

Compression

- Images contain many **low-complexity** regions: spatial repeats / patterns, values not using the full theoretical dynamical range
- Lossless: Original data can be perfectly reconstructed from the compressed data, e.g., gzip, bzip2; with images: PNG, TIFF (optional)
- TIFF: Very flexible format that be lossless as well as lossy, also allows lots of metadata
- For scientific images: **use lossless** formats

Compression

```
object.size(hivc) %>%
  format(units = "Mb")
```

```
## [1] "29.8 Mb"
```

```
(object.size(hivc) / prod(dim(hivc))) %>%
  format %>%
  paste("per pixel")
```

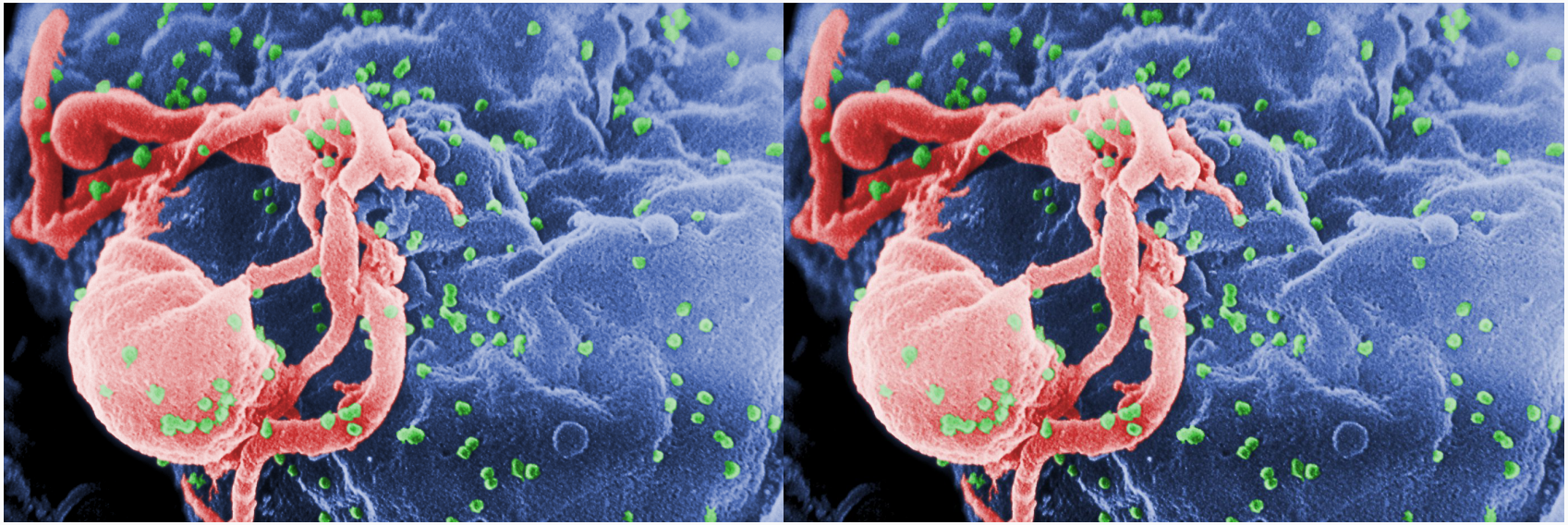
```
## [1] "8 bytes per pixel"
```

```
writeImage(hivc, "hivc.jpeg", quality = 85)
(file.info("hivc.jpeg")$size / 2^20) %>%
  round(digits = 2) %>%
  paste("Mb")
```

```
## [1] "0.28 Mb"
```

Compression

Left: Original image. Right: Compressed image.



Linear Filters

- In image processing, a **kernel**, convolution matrix, or mask is a small matrix
- It is used for blurring, sharpening, embossing, edge detection, and more
- This is accomplished by doing a **convolution** between a kernel and an **image**

Linear Filter

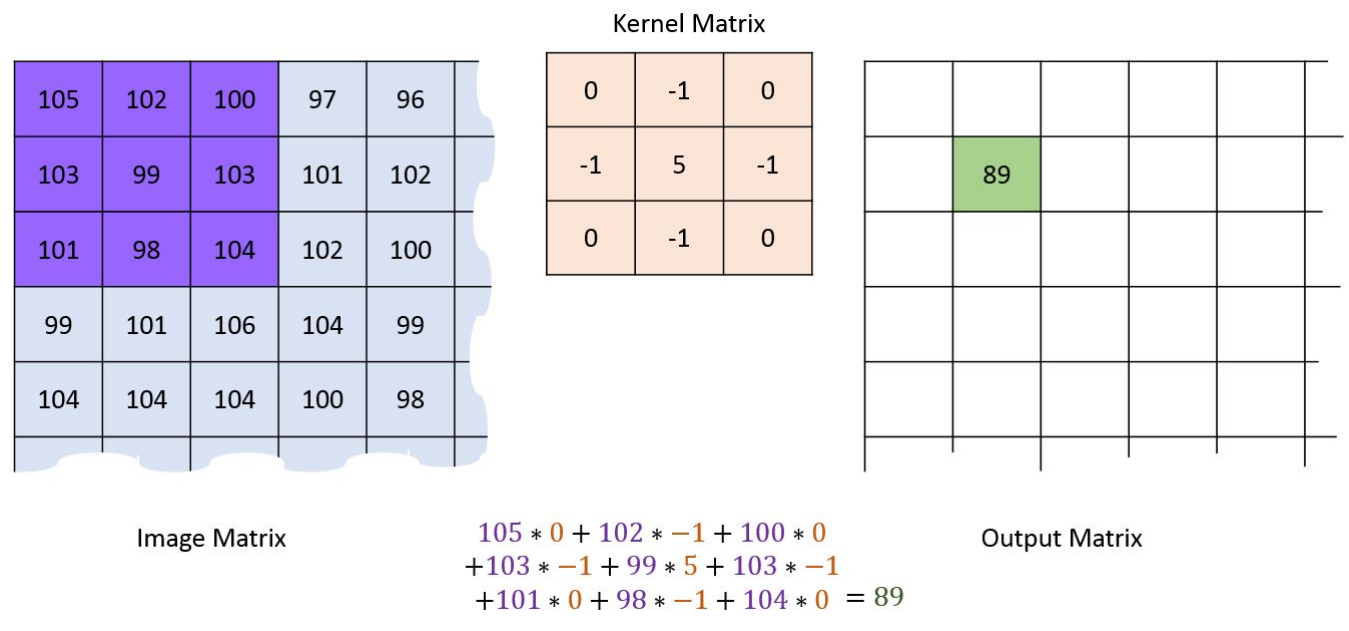


Image source: machinelearninguru.com

Linear Filter

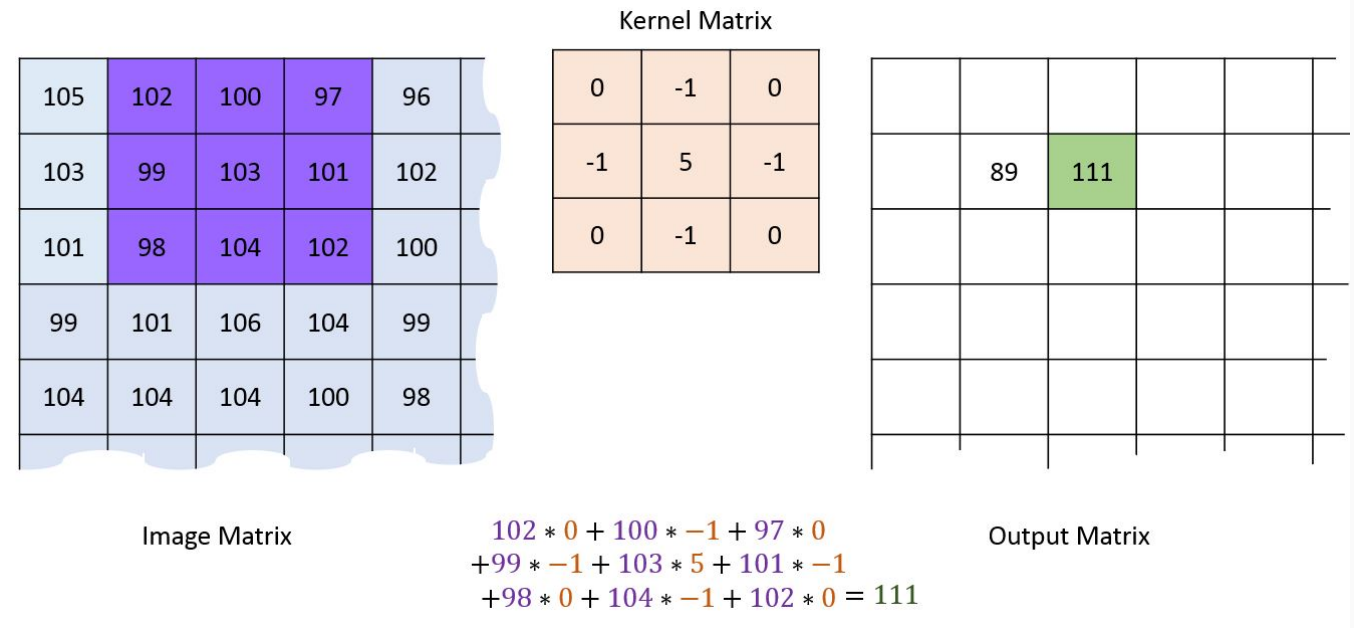


Image source: machinelearninguru.com

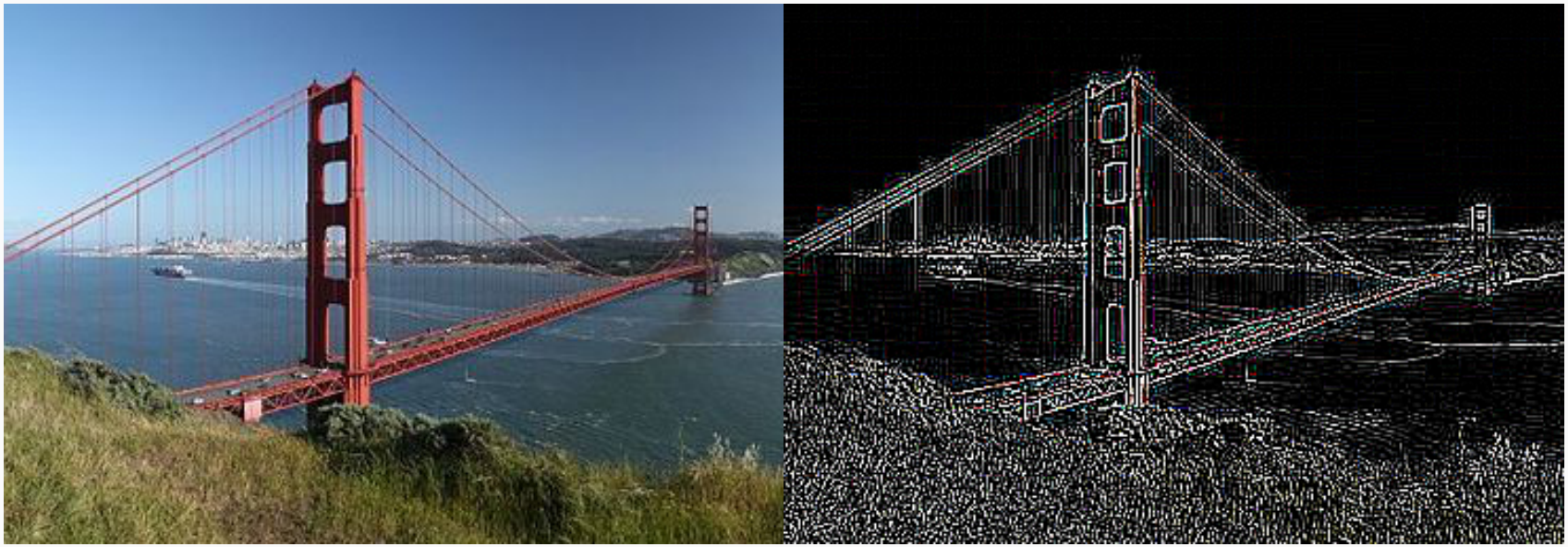
Linear Filter

```
kernel = matrix(-1, nrow = 3, ncol = 3)
kernel[2,2] = 8
kernel
```

```
##      [,1] [,2] [,3]
## [1,]   -1   -1   -1
## [2,]   -1    8   -1
## [3,]   -1   -1   -1
```

Linear Filter: Edge Detection

Left: Original image. Right: Filtered image.



Linear Filter

```
kernel = makeBrush(size = 3, shape = "gaussian",
                   sigma = 1)
kernel
```

```
##             [,1]      [,2]       [,3]
## [1,] 0.07511361 0.1238414 0.07511361
## [2,] 0.12384140 0.2041800 0.12384140
## [3,] 0.07511361 0.1238414 0.07511361
```

Linear Filter: Blurring

Left: Original image. Right: Blurred image.

Morphological Operations

- **erode**: For every foreground pixel, put the mask around it, and if any pixel under the mask is from the background, then set all these pixels to background.

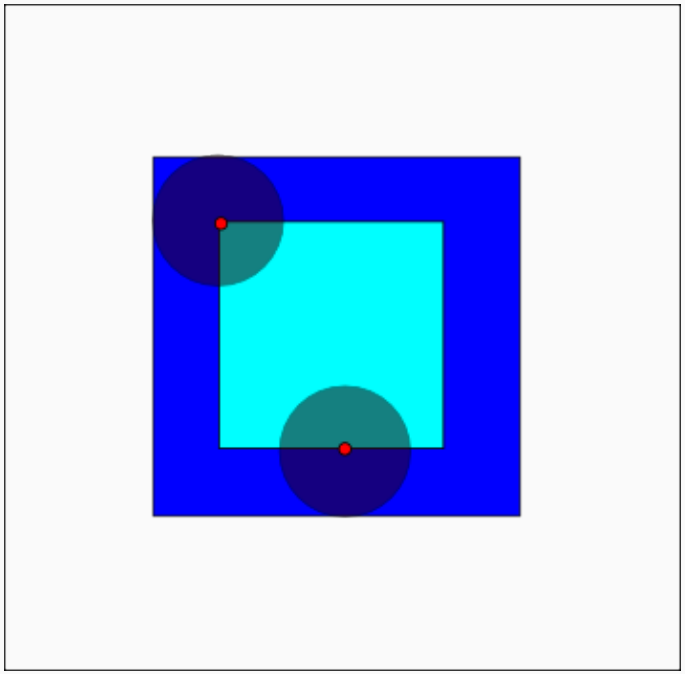


Image source: wiki

Morphological Operations

- **dilate**: For every background pixel, put the mask around it, and if any pixel under the mask is from the foreground, then set all these pixels to foreground.

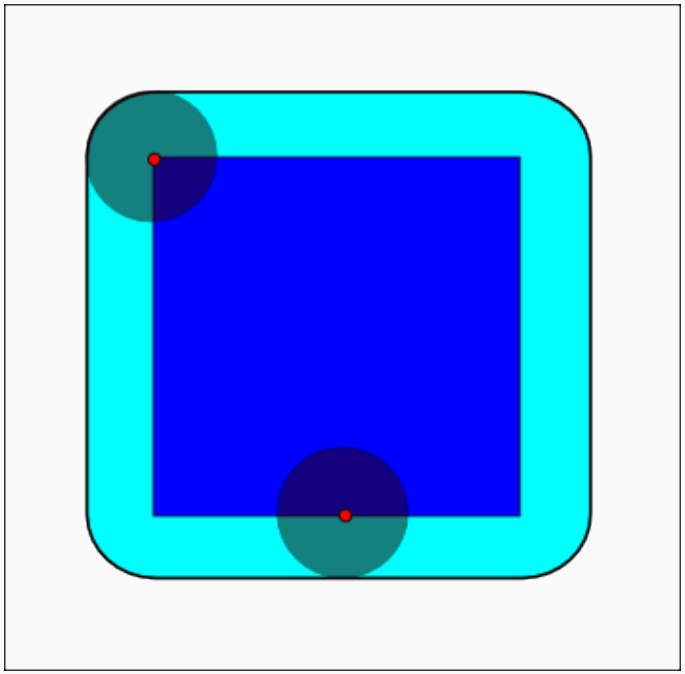


Image source: wiki

Morphological Operations

- **erode + dilate = opening**: Eliminates noise outside objects

Image source: slideshare.net

Morphological Operations

- **dilate + erode = closing**: Eliminates noise within objects

Image source: slideshare.net

Voronoi Tessellation

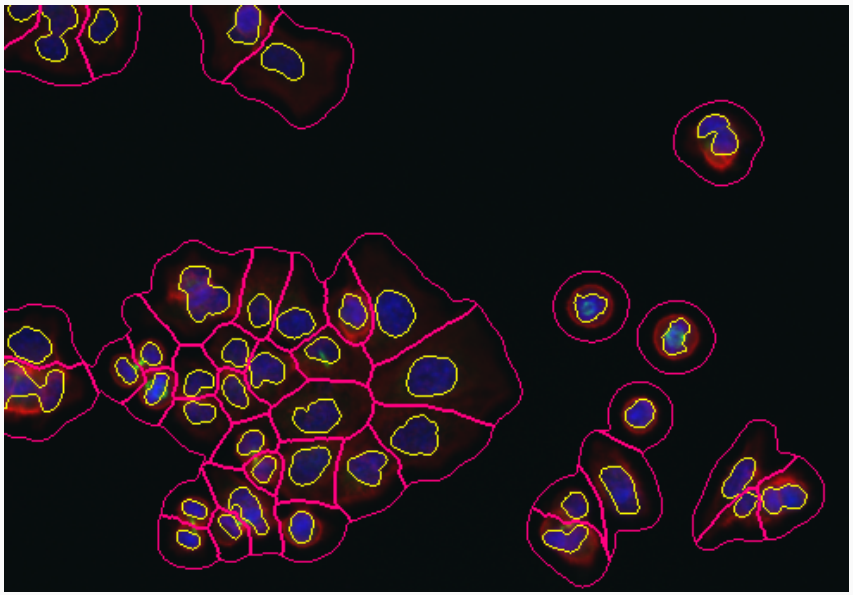
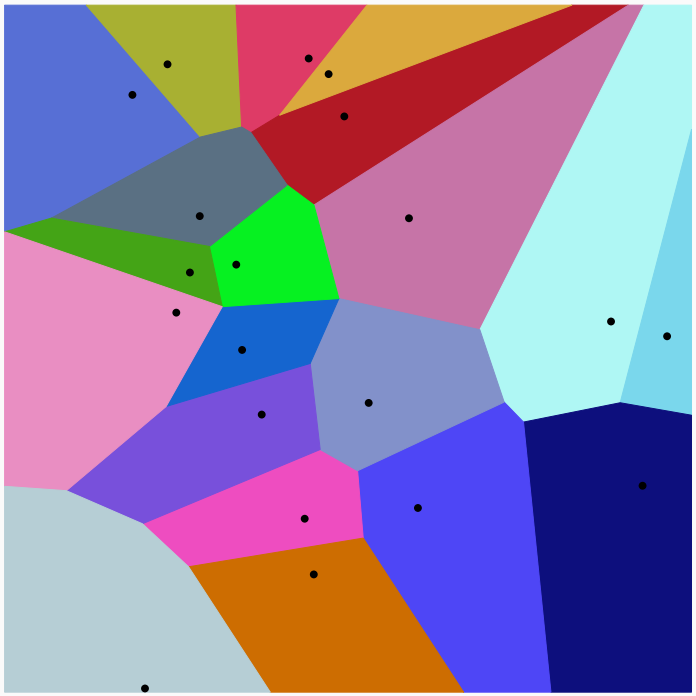


Image source: Figure 11.19 in the book

Voronoi Tessellation: Distances

Animated gif: wiki

Euclidean Distance



Manhattan Distance

Image source: wiki

Voronoi Tessellation: Distances

- Euclidean distance between (x, y) and $(x + dx, y + dy)$:

$$ds^2 = dx^2 + dy^2$$

- Extend it by including image gradient g:

$$ds^2 = \frac{2}{\lambda + 1}[\lambda(dx^2 + dy^2) + (dxg_x)^2 + (dyg_y)^2]$$

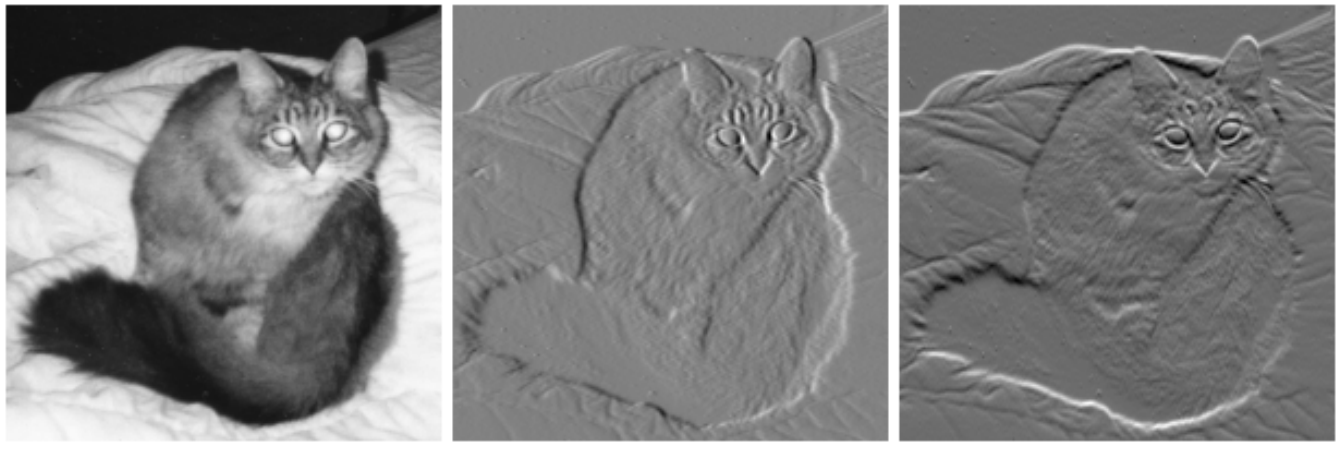


Image source: wiki

Voronoi Tessellation: Distances

$$ds^2 = \frac{2}{\lambda + 1}[\lambda(dx^2 + dy^2) + (dxg_x)^2 + (dyg_y)^2]$$

- λ controls the weight of the Euclidean distance term:
  - For $\lambda \to \infty$, ds tends to the Euclidean distance
  - For $\lambda \to 0$, ds tends to the intensity gradient of the image
  - For $\lambda = 1$, Euclidean distance and image gradient contribute equally

Feature Extraction

- Extract features by processing cell images using using
  - linear filter
  - morphological operators, and
  - Voronoi tessellation
  - Many more...

Feature Extraction

- We can now use multivariate analysis methods:
  - **Clustering**: Detecting cell subpopulations
  - **Classification**: Classifying cells into pre-defined cell types or phenotypes
  - **Differential analysis**: Seeing whether the absolute or relative frequencies of the subpopulations or cell types differ between images that correspond to different biological conditions

Case Study:
Stem Cell Differentiation

Goal

- Methods to characterize mesenchymal stem cells (MSCs) are limited to
  - CD marker expression
  - Plastic adherence
  - Ability to differentiate into adipogenic, osteogenic and chondrogenic precursors
- Stem cells undergoing differentiation should differ in many aspects, such as
  - Morphology
  - Behaviour
- Goal of this study was to predict cell fate of MSCs from morphological features
- Using Time-lapse microscopy: https://www.youtube.com/watch?v=HM17p1vDF9A

Data

- We monitored cells using a time-lapse microscope
- Take an image of each well every 2 hours
- Start 20 min after seeding cells
- Run for 6 days with break after 3 days for media refreshment
- Covered the exponential growing phase until confluence
- The data is stored in tiff images

Cell Segmentation

(1) original image

(2) edge detection

(3) image dilation

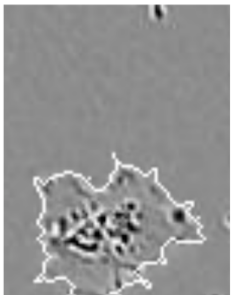
(4) removal of object close to image border

(5) image erosion

(6) removal of small objects

(7) filling of holes inside the cell

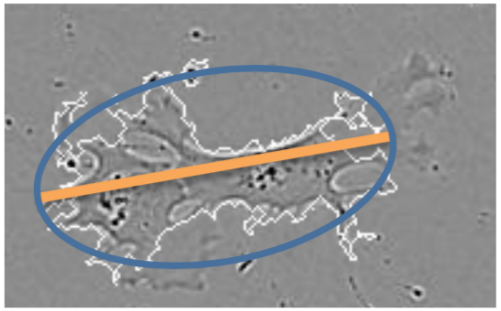
(8) final results: overlay on the original



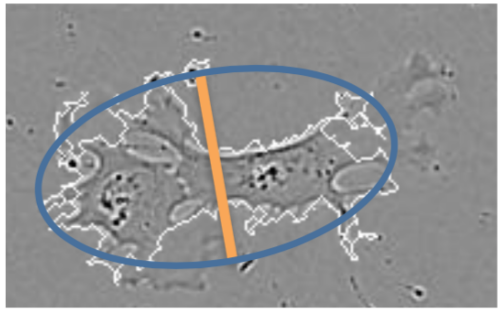
Cell Features

area

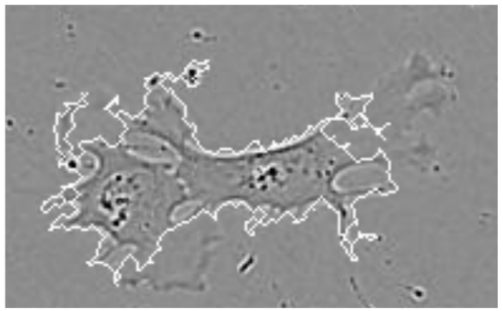
major axis length



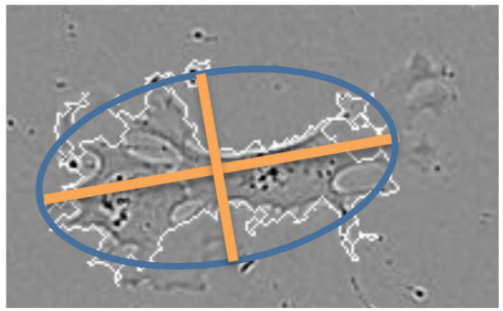
minor axis length



perimeter

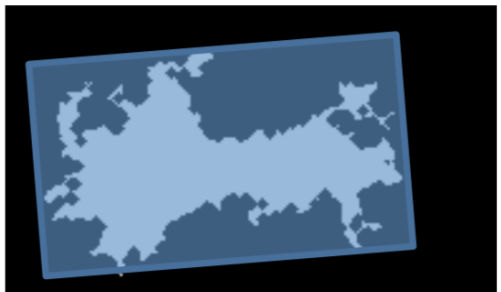


eccentricity

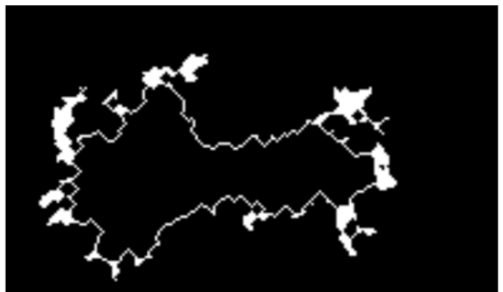


extent

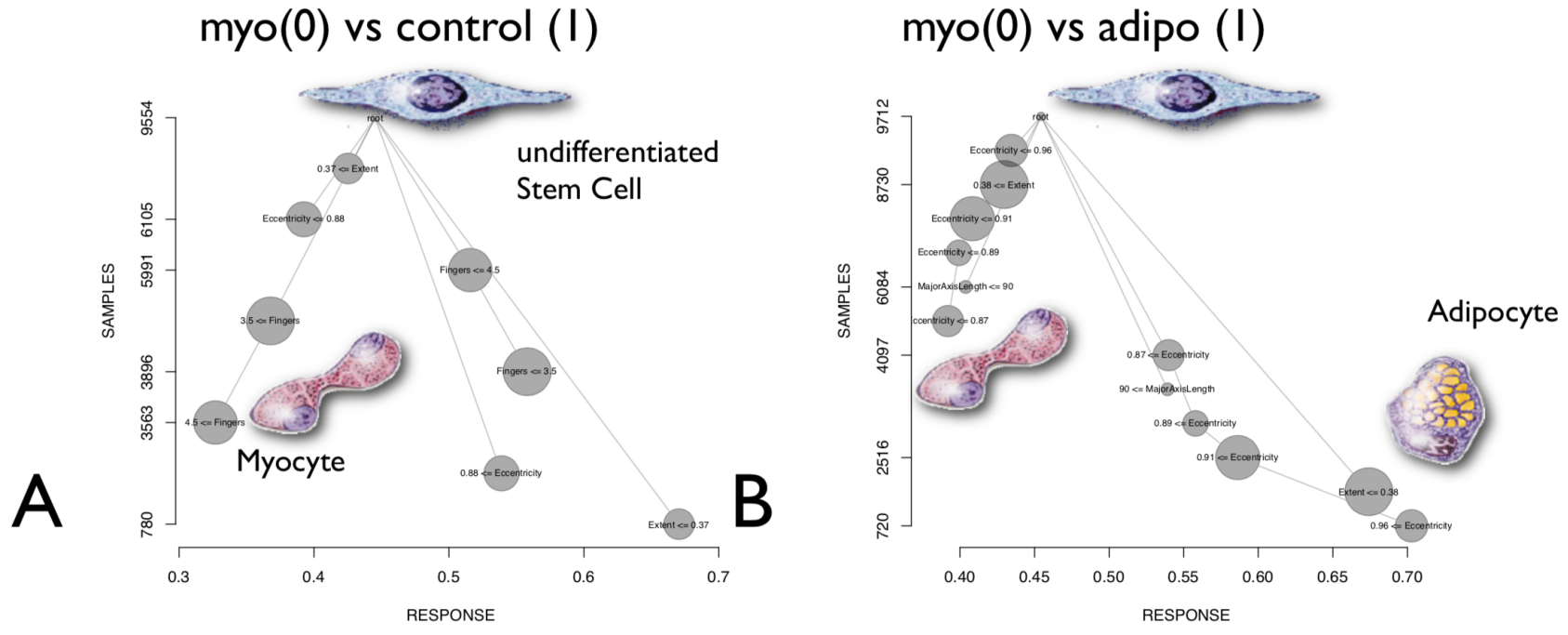
/



finger (filopodia)



Cell Classifier



Results and Conclusions

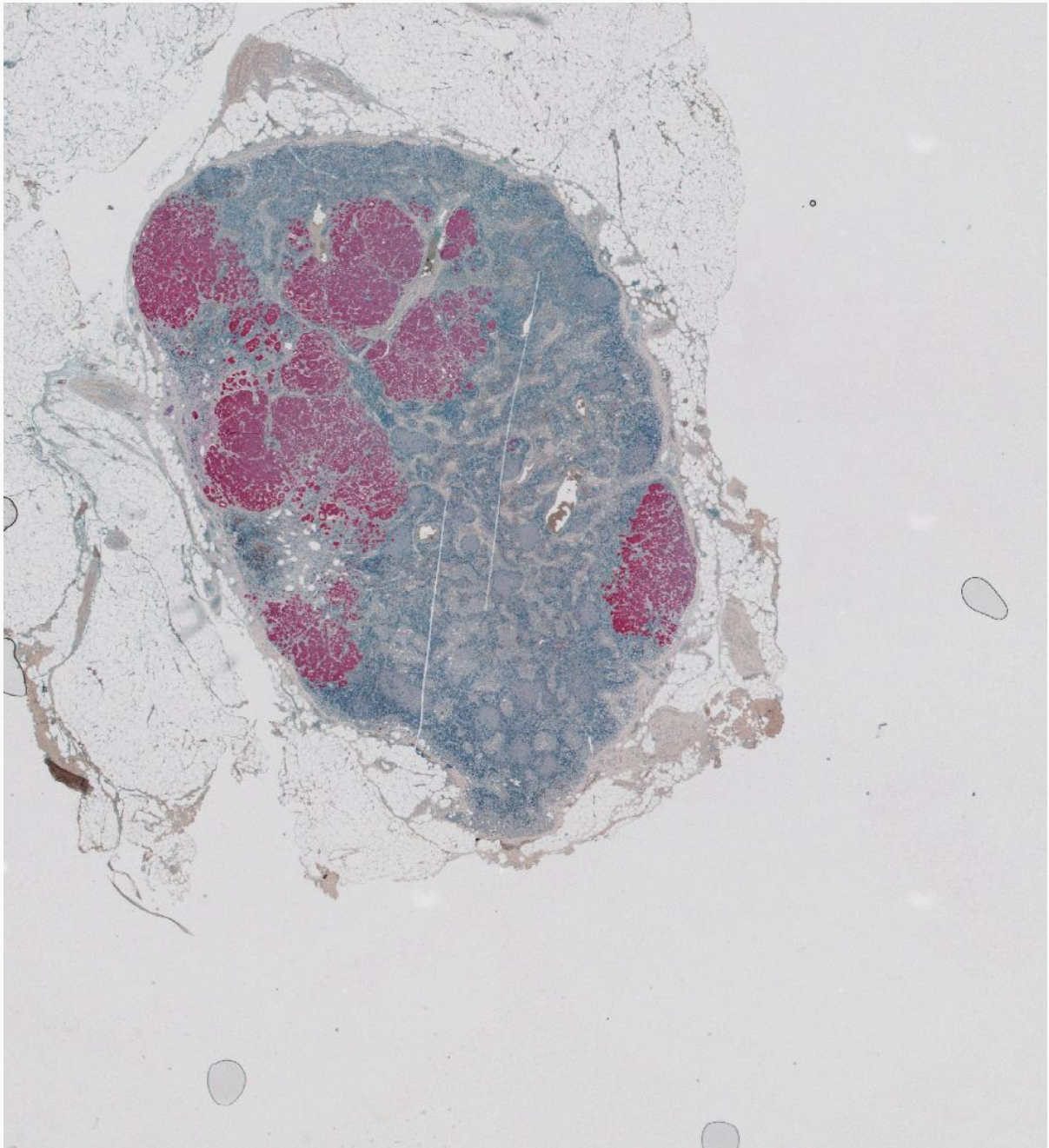
- Our results suggest:
  - Non-invasive automated time-lapse microscopy could potentially be used to predict the hMSCs fate on morphology earlier time-points
  - Might be useful in for clinical application
- More challenges:
  - Cell density
  - Proliferation and possible unknown donor-specific factors
- For more see Seiler et al. (2014)

Spatial Statistics

Motivation

- Now: Will make use of spatial position of cells in our analyses
- Basic question: Are neighboring points clustering? (i.e., do they attract or repel each other?)
  - crime patterns within a city
  - disease patterns within a country
  - soil measurements in a region

Motivation



Interaction Between Immune and Cancer Cells

- Data from (Setiadi 2010)
- Segment cells using the image processing tools we learned about earlier
- We end up with coordinates and the type of all the cells in the image
- We call this type of data a **marked point process**

Interaction Between Immune and Cancer Cells

- Load data table:

```
brcalymphnode
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.boxx spatstat
```

```
## # A tibble: 209,462 x 3
##        x     y class
##    <dbl> <dbl> <fct>
##  1  6355 10382 T_cells
##  2  6356 10850 T_cells
##  3  6357 11070 T_cells
##  4  6357 11082 T_cells
##  5  6358 10600 T_cells
##  6  6361 10301 T_cells
##  7  6369 10309 T_cells
##  8  6374 10395 T_cells
##  9  6377 10448 T_cells
```

Interaction Between Immune and Cancer Cells

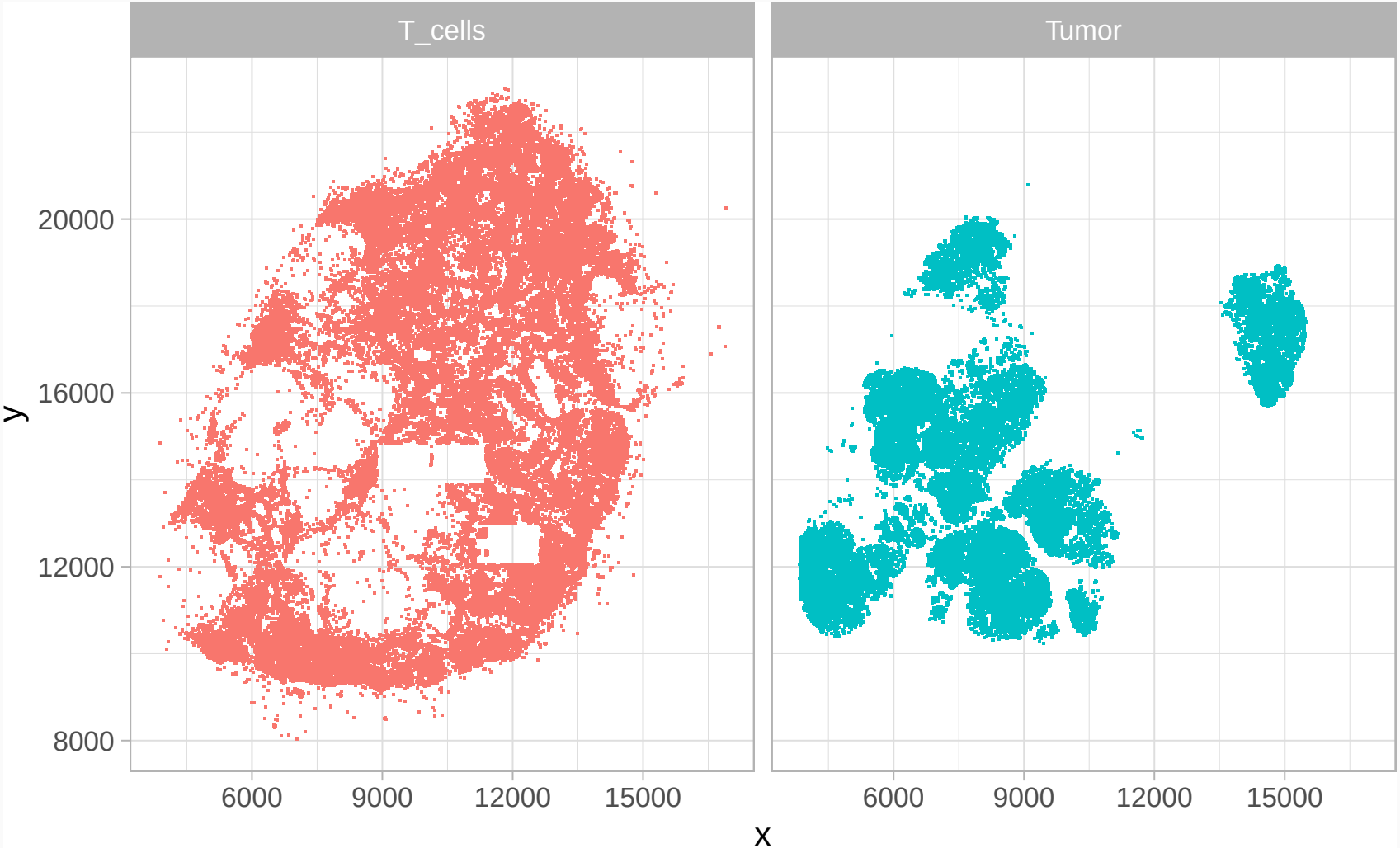
- Count cell types:

```
table(brcalymphnode$class)
```

```
##
##          DCs other_cells     T_cells       Tumor
##          878       77081      103681       27822
```

Interaction Between Immune and Cancer Cells

- Plot the x and y positions of the cells:



Interaction Between Immune and Cancer Cells

- Create spatial point process object **ppp** from package **spatstat**:

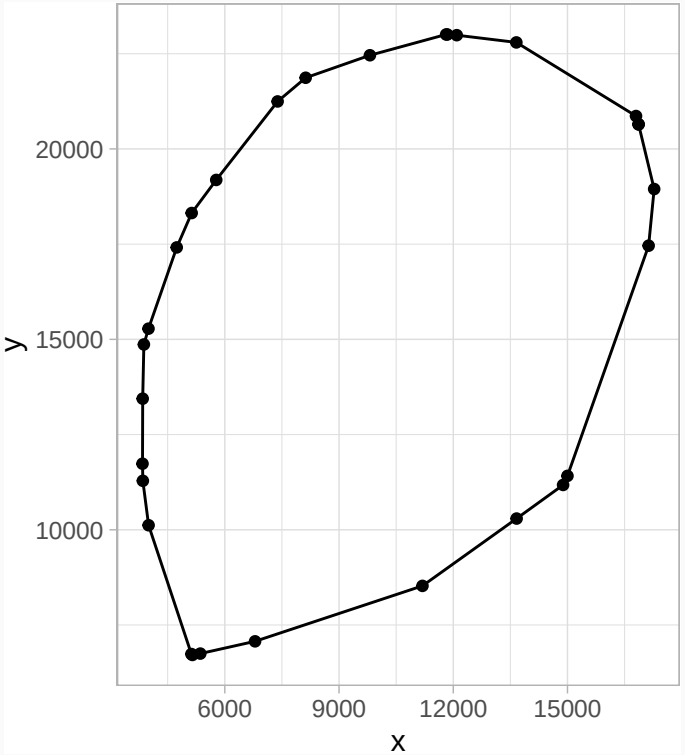
```
ln = with(brcalymphnode,
  spatstat::ppp(x = x, y = y,
                marks = class,
                xrange = range(x), yrange = range(y)))
ln
```

```
## Marked planar point pattern: 209462 points
## Multitype, with levels = DCs, other_cells, T_cells, Tumor
## window: rectangle = [3839, 17276] x [6713, 23006] units
```

Interaction Between Immune and Cancer Cells

- Compute convex hull using package **geometry**:

```
coords = cbind(ln$x, ln$y)
chull = geometry::convhulln( coords )
```



Interaction Between Immune and Cancer Cells

- Call ppp again, this time with the polygon:

```
ln = with(brcalymphnode,
   spatstat::ppp(x = x, y = y, marks = class, poly = coords[ pidx, ],
       check = FALSE))
ln
```

```
## Marked planar point pattern: 209462 points
## Multitype, with levels = DCs, other_cells, T_cells, Tumor
## window: polygonal boundary
## enclosing rectangle: [3839, 17276] x [6713, 23006] units
```

Poisson Process

- Key question: What we expect by chance?
- Strategy: Define a probabilistic model as null model
- The simplest spatial process is the **Poisson process**
- We will use it as a null model against which to compare our data

Poisson Process

- Examples: Rain drops falling on the floor are modeled by a Poisson process
- The number of drops falling on a particular spot depends on
  - rate λ
  - size of the spot
  - but not on what happens at other spots

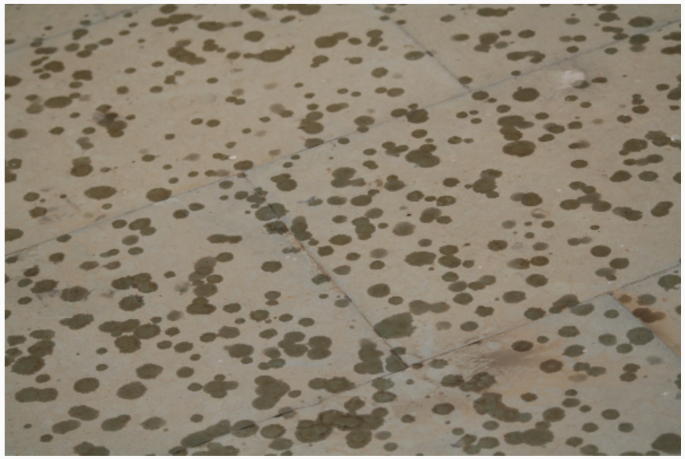


Image source: Figure 11.25 in the book

Poisson Process

- Count the number of points lying in a circle of area a around a point p
- Denote this quantity by $N(p, a)$
- Then look at the mean and covariance of $N(p, a)$
- The mean is
$$\lambda(p) = \lim_{a \to 0} \frac{\mathrm{E}[N(p, a)]}{a}$$
- Stationary process: $\lambda(p) = \text{constant}$

First Order Effects: The Intensity

- Check visually if $\lambda(p)$ is constant by estimating local rate at point p:

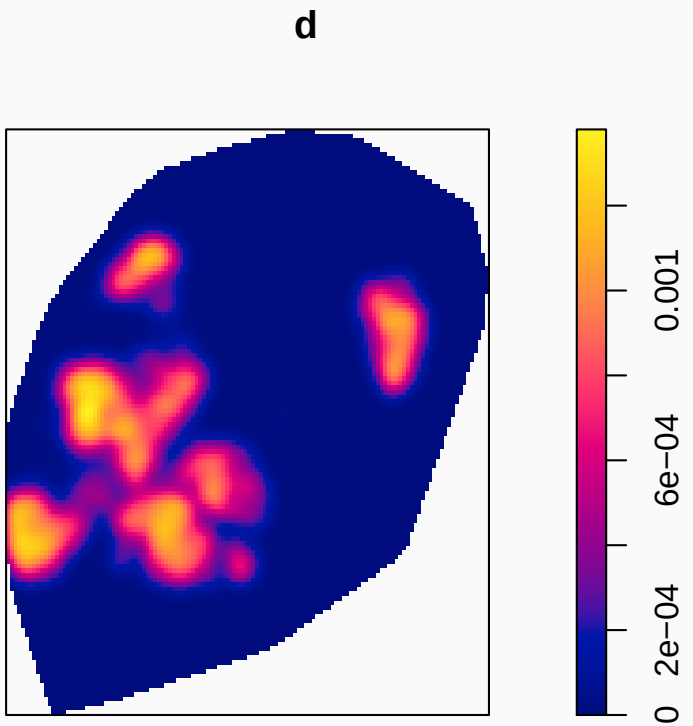
$$\widehat{\lambda}(p) = \sum_i e(p_i) K(p - p_i)$$

  - kernel function K (decays as a function of distance to point p)
  - edge correction factor $e(p_i)$

First Order Effects: The Intensity

- This is implemented in spatstat package:

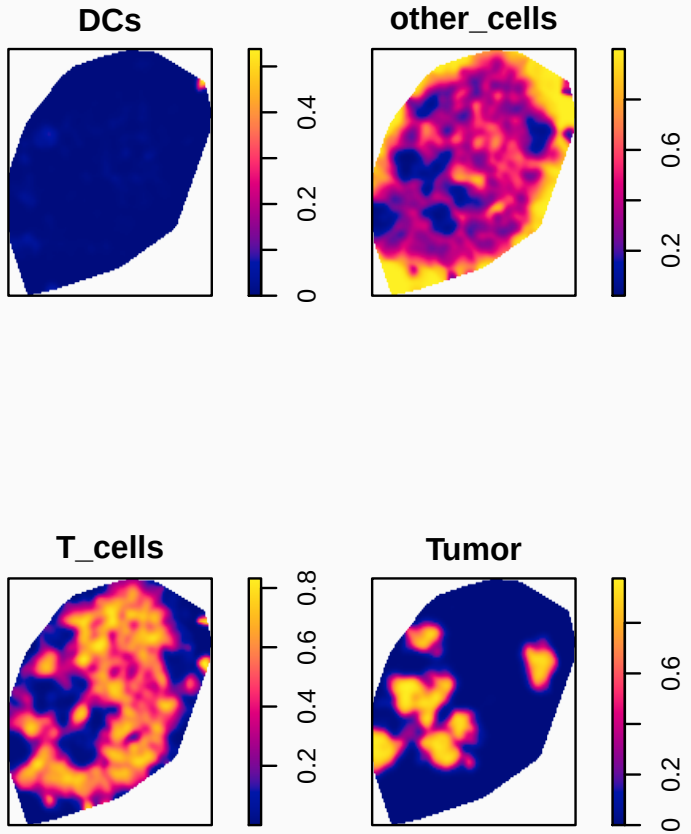
```
d = density(subset(ln, marks == "Tumor"), edge=TRUE,
            diggle=TRUE, sigma=250)
plot(d)
```



First Order Effects: The Intensity

- Probability that a cell is of cell of certain type at p_i **given** any cell present at p_i

relrisk(ln, sigma = 250)

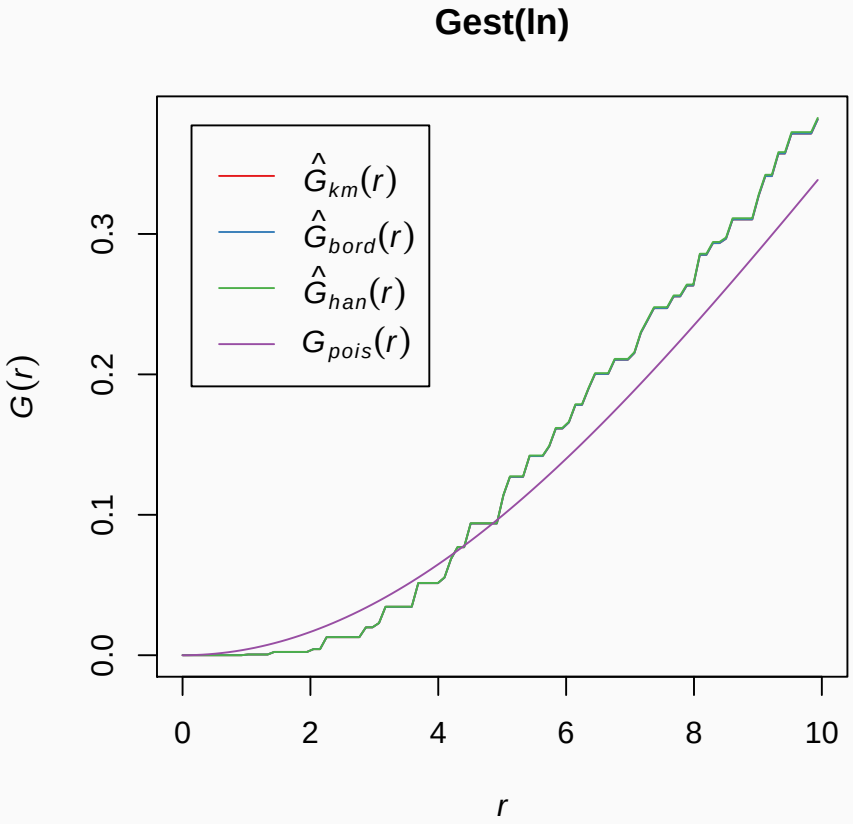


Second Order Effects: Spatial Dependence

- If we pick a point at random in our spatial process, what is the distance R to its nearest neighbor?
- For a homogeneous Poisson process, the cumulative distribution function of this distance is

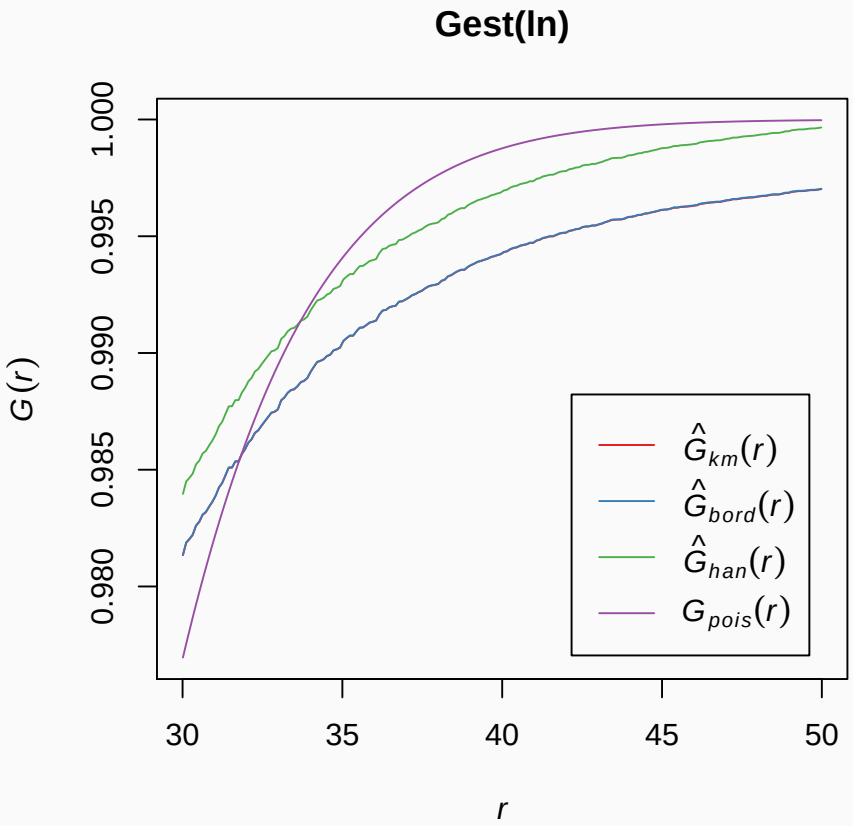
$$G(r) = P(R \leq r) = 1 - e^{-\lambda \pi r^2}$$

Second Order Effects: Spatial Dependence



- Cell to cell distances that are shorter are less likely than for the null model
  - There are essentially no distances below 2
  - Cells have finite size and cannot overlap the same space

Second Order Effects: Spatial Dependence



- There seems to be a trend to avoid very large distances
  - Perhaps indicative of a tendency of the cells to cluster?

Second Order Effects: Ripley's K Function

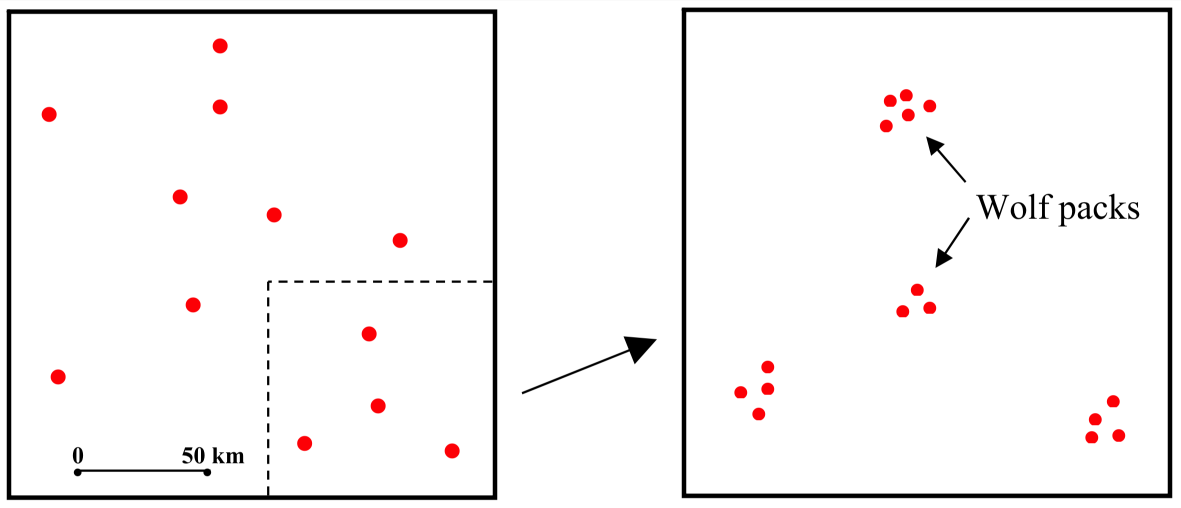


Image source: UPenn ESE 502 Lecture Slides

- This pattern of wolf locations is
  - very clustered at small scales
  - very dispersed at large scales
- Useing nearest-neighbor techniques would only detect small-scale clustering

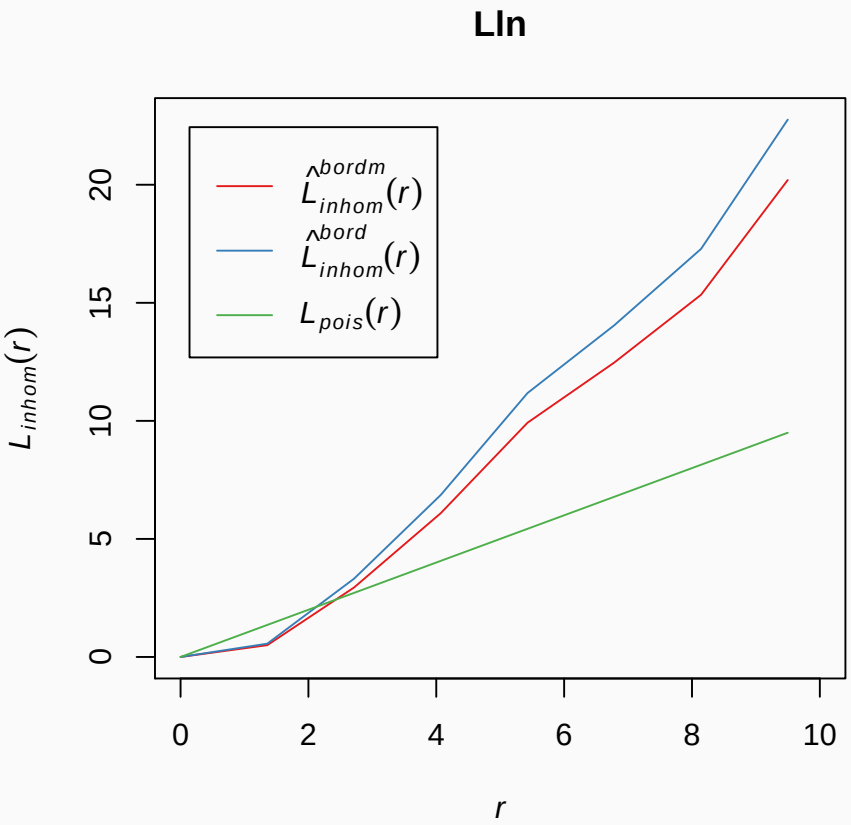
Second Order Effects: Ripley's K Function

- What is the average number of points found near a randomly picked point?
- The K-function (d_{ij} is distance between point i and j):

$$\widehat{K}(r) = \widehat{\lambda}^{-1} \left(\frac{1}{n} \sum_i \sum_{j \neq i} I(d_{ij} < r) \right)$$

- Under the homogeneous Poisson process: $\widehat{K}(r) \approx \pi r^2$ (area of the circle)
- L-function: $\widehat{L}(r) = \sqrt{\frac{\widehat{K}(r)}{\pi}}$
- So, $L(r) = r$ under complete spatial randomness

Second Order Effects: Ripley's K Function



- This data seems to be more clustered compared to a complete spatial random pattern

Medical Image Analysis

Medical Images

- Medical images are **three**-dimensional and EBImage is great for **two**-dimensional images
- A useful toolkit for handling images in R is the package **SimpleITK** (Lowekamp et al. 2013, @yaniv2018simpleitk)
- Also it provides some more advanced feature such as **image registration**

Image Registration

- Image registration is the task of finding a spatial transform mapping one image into another

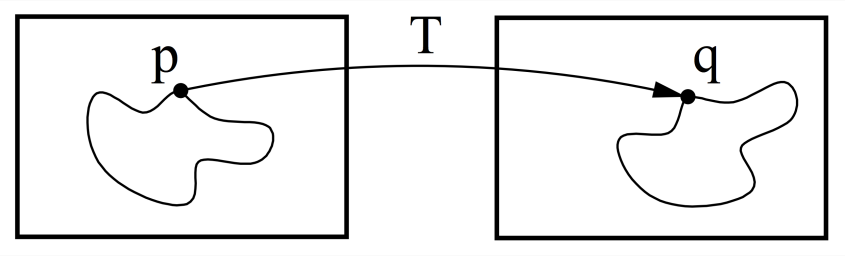


Image source: The ITK Software Guide

Image Registration

- Left: Fixed image. Right: Moving image.

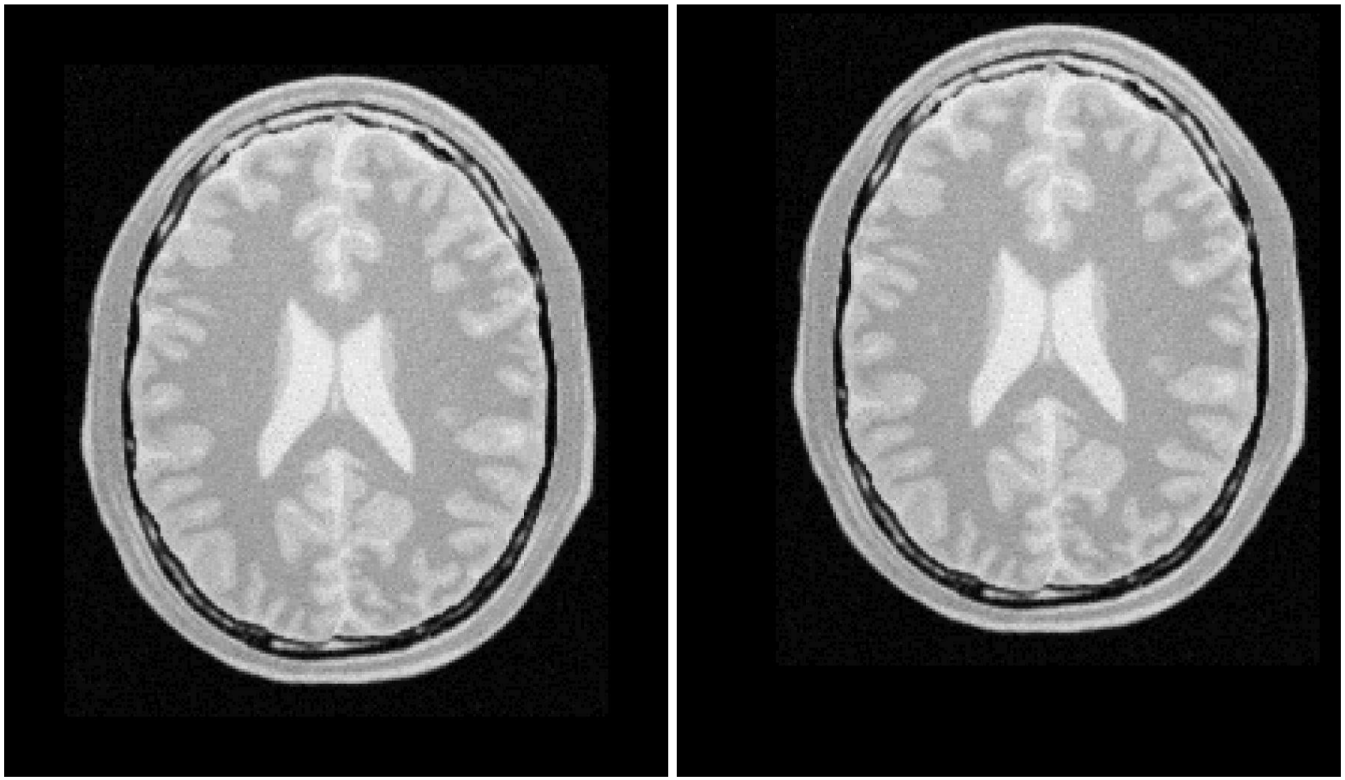


Image source: The ITK Software Guide

Image Registration

- Mapped moving image and its difference with the fixed image before and after registration

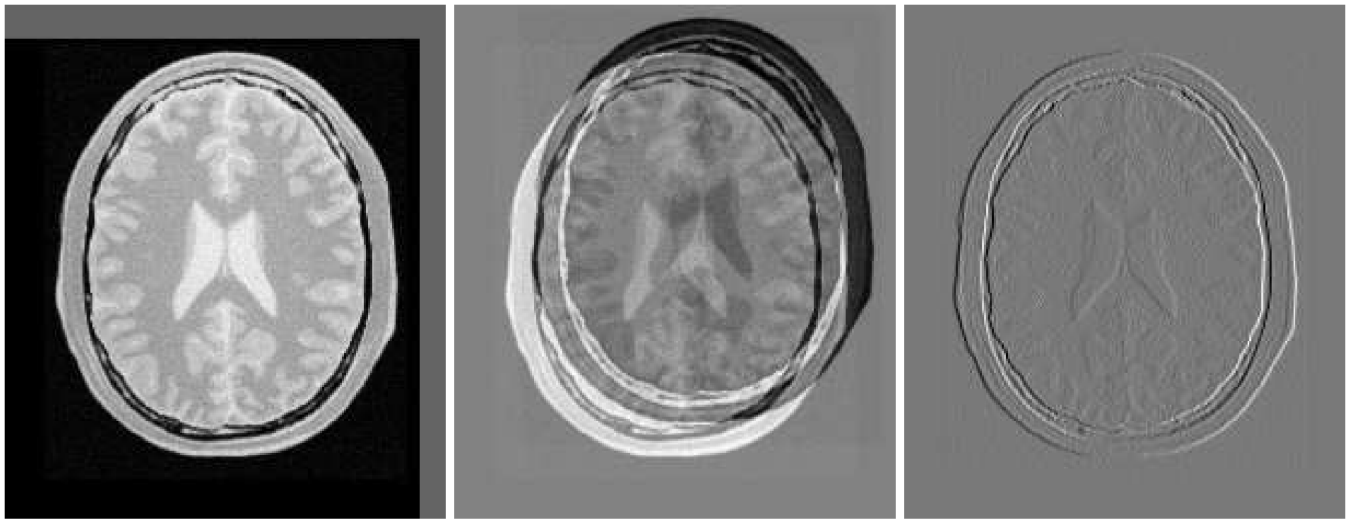


Image source: The ITK Software Guide

The SimpleITK Package

Installation requires cmake and git in the path:

```
devtools::install_github("SimpleITK/SimpleITKRInstaller")
```

For more: SimpleITK-Notebooks

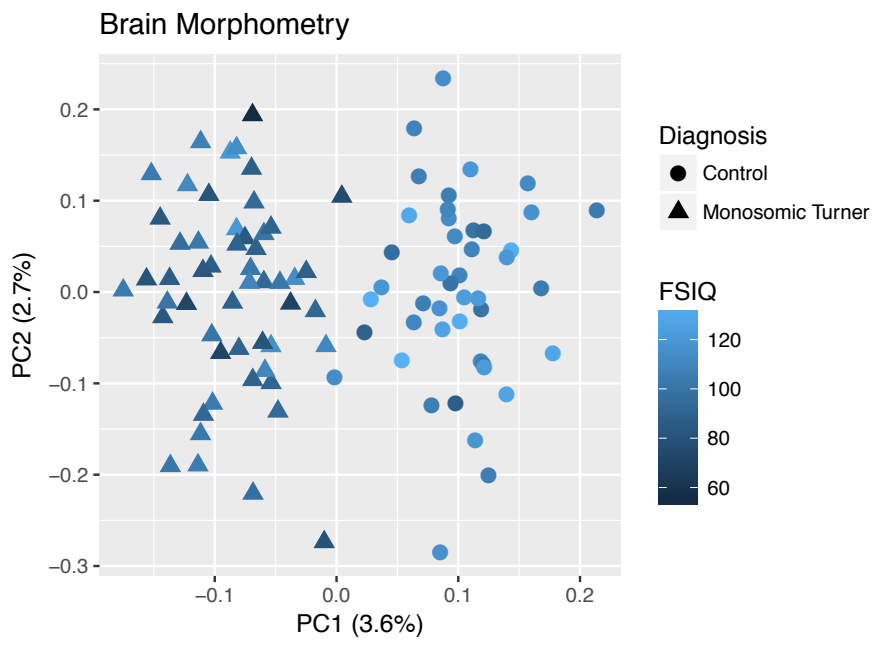
Case Study:
Brain Morphometry

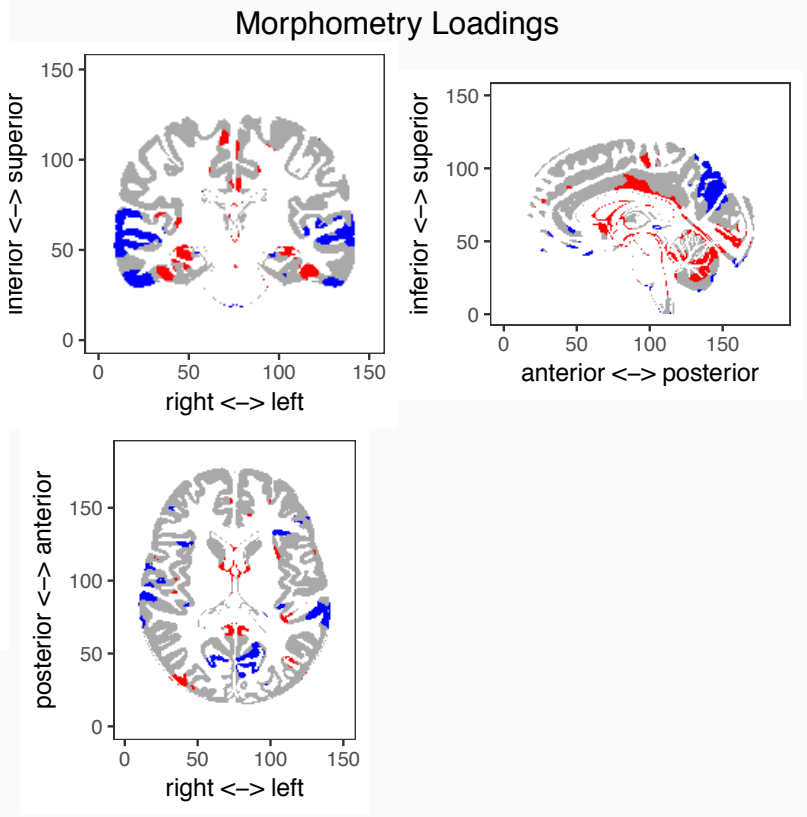
Goal and Methods

- Compare brain morphology between girls with Turner syndrome and age-matched control group
- Use registration to map brains to an average brain
- Compare deformations between groups
- For more see Seiler et al. (2018)

PCA Results

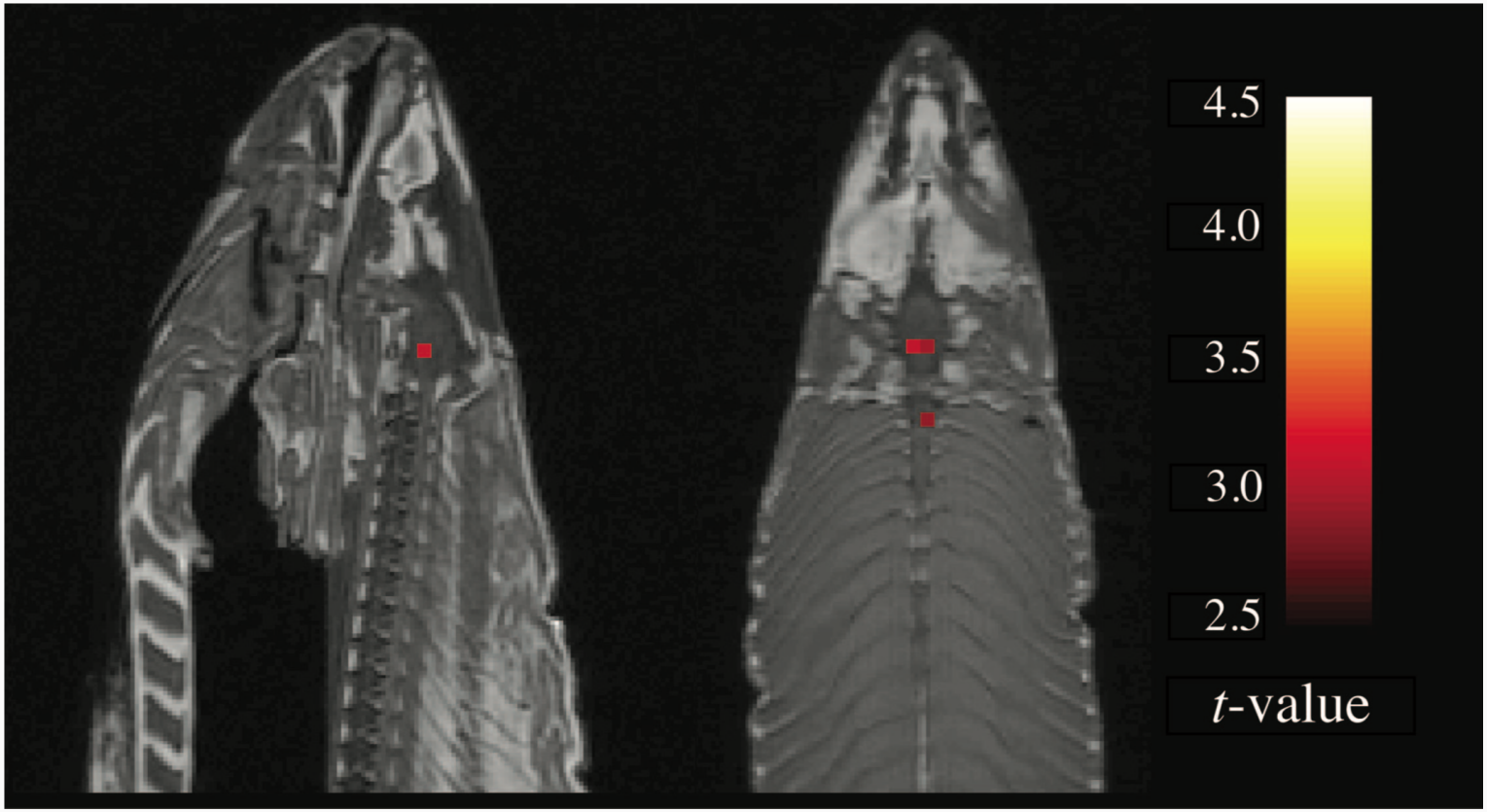
- Left: Participants projected onto PC1 and PC2
- Right: 10% largest positive and 10% negative PC1 loadings
- Blue brain regions are larger in the TS group and smaller in the healthy control group





Case Study:
Brain Activity in Dead Salmon

Results



- Sagittal/axial slices of significant brain voxels in the task > rest contrast

Statistical Issues

- Two clusters passed the uncorrected p-value threshold of 0.001:
  1. One cluster in the medial brain cavity
  2. Another cluster in the upper spinal column
- At that time (around 2010), between 25-40% of studies on published fMRI did NOT adjust p-values for multiple comparisons
- Bennet et al. (2010) published in Journal of Serendipitous and Unexpected Results

Summary

Take Home

- Images are 2 or 3-dimensional **arrays**
- Workflow:
  1. **Extract quantitative features** from images
  2. Statistical summarization, hypothesis testing, ANOVA, clustering and classification
- Common tasks:
  1. **Segmentation**: compute quantitative features not on whole image, but for individual objects
  2. **Spatial statistics**: positions of the objects and how positions relate to each other
  3. **Registration**: In medical image analysis, we usually find transformations between images using and compare the transformations instead of the images

References i

Bennet, C, A Baird, M Miller, and G Wolford. 2010. "Neural Correlates of Interspecies Perspective Taking in the Post-Mortem Atlantic Salmon: An Argument for Proper Multiple Comparisons Correction." *Journal of Serendipitous and Unexpected Results* 1 (1): 1–5.

Lowekamp, Bradley Christopher, David T Chen, Luis Ibáñez, and Daniel Blezek. 2013. "The Design of SimpleITK." *Frontiers in Neuroinformatics* 7. Frontiers: 45.

References ii

Pau, Grégoire, Florian Fuchs, Oleg Sklyar, Michael Boutros, and Wolfgang Huber. 2010. "EBImage – An R Package for Image Processing with Applications to Cellular Phenotypes." *Bioinformatics* 26 (7). Oxford University Press: 979–81.

Seiler, Christof, Amiq Gazdhar, Mauricio Reyes, Lorin M Benneker, Thomas Geiser, Klaus A Siebenrock, and Benjamin Gantenbein-Ritter. 2014. "Time-Lapse Microscopy and Classification of 2D Human Mesenchymal Stem Cells Based on Cell Shape Picks up Myogenic from Osteogenic and Adipogenic Differentiation." *Journal of Tissue Engineering and Regenerative Medicine* 8 (9). Wiley Online Library: 737–46.

Seiler, Christof, Tamar Green, David Hong, Lindsay Chromik, Lynne Huffman, Susan Holmes, and Allan L Reiss. 2018. "Multi-Table Differential Correlation Analysis of Neuroanatomical and Cognitive Interactions in Turner Syndrome." *Neuroinformatics* 16 (1). Springer: 81–93.

Setiadi, Nelson C. AND Kohrt, A. Francesca AND Ray. 2010. "Quantitative, Architectural Analysis of Immune Cell Subsets in Tumor-Draining Lymph Nodes from Breast Cancer Patients and Healthy Lymph Nodes." *PLOS ONE* 5 (8). Public Library of Science: 1–8. https://doi.org/10.1371/journal.pone.0012420.

References iv

Yaniv, Ziv, Bradley C Lowekamp, Hans J Johnson, and Richard Beare. 2018. "SimpleITK Image-Analysis Notebooks: A Collaborative Environment for Education and Reproducible Research." *Journal of Digital Imaging* 31 (3). Springer: 290–303.