

# Cardinal tutorial

Kylie Ariel Bemis

10/10/2019

## Slides and code

The R Markdown for this tutorial is available on Github at:

<https://github.com/kuwisdelu/asilomar2019>

Use R or RStudio to run the document (you may need to update some file paths).

# Installation

First make sure BiocManager is installed.

Then use BiocManager to install Cardinal.

```
if ( !requireNamespace("BiocManager") )  
  install.packages("BiocManager")
```

```
BiocManager::install("Cardinal")
```

For the datasets used in the statistical analysis examples, we'll also install CardinalWorkflows.

```
BiocManager::install("CardinalWorkflows")
```

## Getting started

Load the Cardinal package with `library()`.

```
library(Cardinal)  
  
register(SerialParam())
```

*Note: Cardinal supports parallelization via the `BiocParallel` package. The code above registers a serial (i.e., non-parallel) backend for this session.*

## Cardinal basics

## Example data

To demonstrate reading data and visualization, we'll use example imzML files from [imzml.org](http://imzml.org).

Specifically, we are going to reproduce the first image from the [imzml.org](http://imzml.org) website, which comes from the paper *"Histology by mass spectrometry: label-free tissue characterization obtained from high-accuracy bioanalytical imaging"* by Rompp et al.

The data is available from <https://www.ebi.ac.uk/pride/archive/projects/PXD001283>.

# Importing data

We can use `readMSIData()` to import data from an imzML file.

```
file <- file.path("~/Documents/Datasets/PRIDE/PXD001283",  
                  "HR2MSI mouse urinary bladder S096.imzML")  
  
mouse <- readMSIData(file, mass.range=c(400, 900),  
                     resolution=0.01, units="mz")
```

Providing the mass range and binning resolution will allow faster importing; otherwise, `Cardinal` will try to infer them from reading the data.

mouse

```
## An object of class 'MSProcessedImagingExperiment'
##   <50001 feature, 34840 pixel> imaging dataset
##     imageData(1): intensity
##     featureData(0):
##     pixelData(0):
##     metadata(16): spectrum representation ibd binary type
##       files name
##     run(1): HR2MSI mouse urinary bladder S096
##     raster dimensions: 260 x 134
##     coord(2): x = 1..260, y = 1..134
##     mass range: 400 to 900
##     centroided: FALSE
```



# Accessing pixel metadata

```
pixelData(mouse)
```

```
## PositionDataFrame with 34840 rows and 0 columns
```

```
##                               :run:  coord:x  coord:y
##                               <factor> <integer> <integer>
## 1      HR2MSI mouse urinary bladder S096          1          1
## 2      HR2MSI mouse urinary bladder S096          2          1
## 3      HR2MSI mouse urinary bladder S096          3          1
## 4      HR2MSI mouse urinary bladder S096          4          1
## 5      HR2MSI mouse urinary bladder S096          5          1
## ...
## 34836 HR2MSI mouse urinary bladder S096        256        134
## 34837 HR2MSI mouse urinary bladder S096        257        134
## 34838 HR2MSI mouse urinary bladder S096        258        134
## 34839 HR2MSI mouse urinary bladder S096        259        134
## 34840 HR2MSI mouse urinary bladder S096        260        134
```

# Accessing feature metadata

```
featureData(mouse)
```

```
## MassDataFrame with 50001 rows and 0 columns
##           :mz:
##           <numeric>
##  1           400
##  2          400.01
##  3          400.02
##  4          400.03
##  5          400.04
##  ...          ...
## 49997         899.96
## 49998         899.97
## 49999         899.98
## 50000         899.99
## 50001          900
```

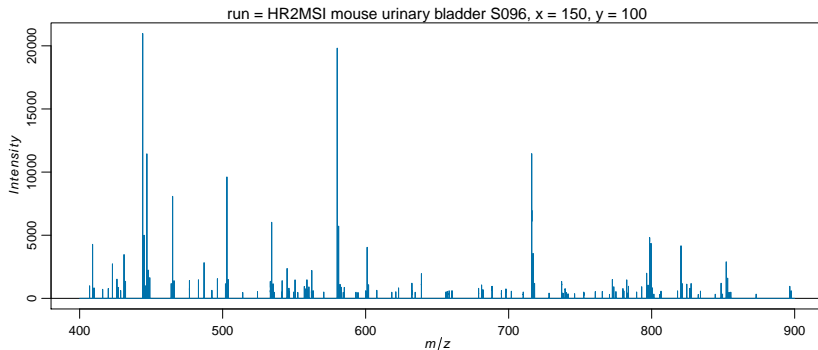
# Accessing the spectra matrix

```
spectra(mouse)
```

```
## <50001 row, 34840 column> sparse_matc :: sparse numeric matrix
##           [,1] [,2] [,3] [,4] [,5] [,6] ...
## [400,]      0    0    0    0    0    0 ...
## [400.01,]    0    0    0    0    0    0 ...
## [400.02,]    0    0    0    0    0    0 ...
## [400.03,]    0    0    0    0    0    0 ...
## [400.04,]    0    0    0    0    0    0 ...
## [400.05,]    0    0    0    0    0    0 ...
## ...      ...    ...    ...    ...    ...    ...
## (67916471/1742034840 non-zero elements: 3.9% density)
## (2.7 MB real | 815 MB virtual)
```

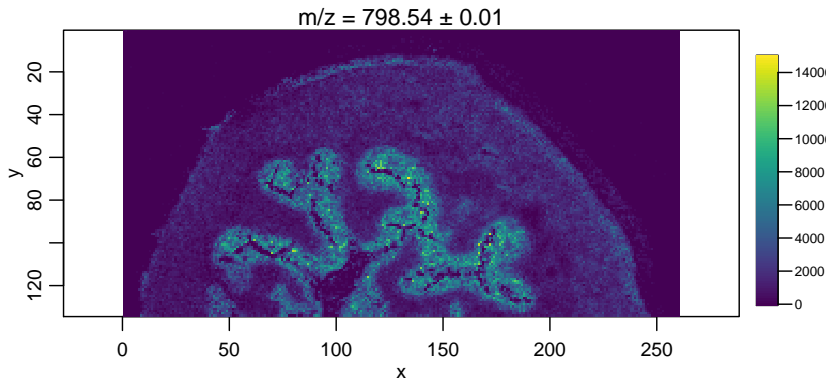
# Visualize mass spectra

```
plot(mouse, coord=list(x=150, y=100))
```



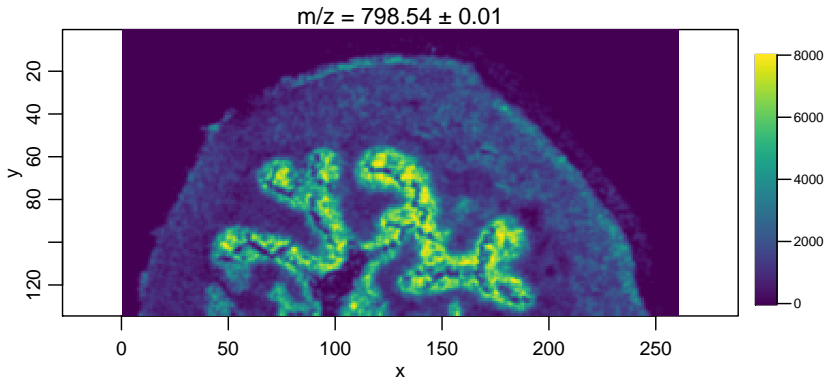
# Visualize ion images

```
image(mouse, mz=798.54, plusminus=0.01)
```



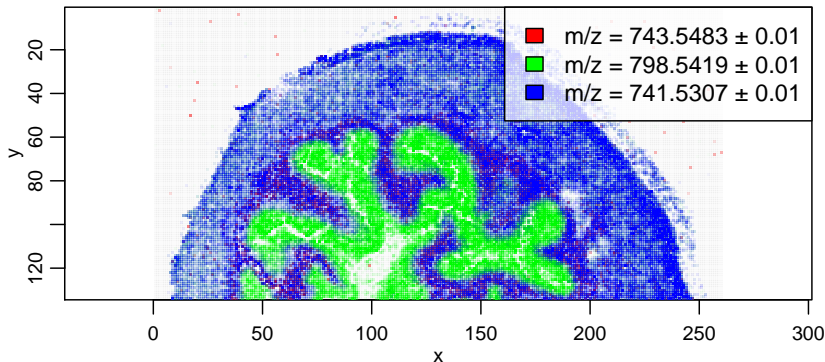
## Visualize ion images (w/ enhancement)

```
image(mouse, mz=798.54, plusminus=0.01,  
      contrast.enhance="suppress",  
      smooth.image="gaussian")
```



## Visualize ion images (superposed)

```
image(mouse, mz=c(743.5483, 798.5419, 741.5307),  
      plusminus=0.01, superpose=TRUE,  
      contrast.enhance="suppress",  
      normalize.image="linear",  
      col=c("red", "green", "blue"))
```



# Segmentation



## Pig fetus section

We will use some pre-processed datasets and pre-calculated analysis results from the `CardinalWorkflows` package.

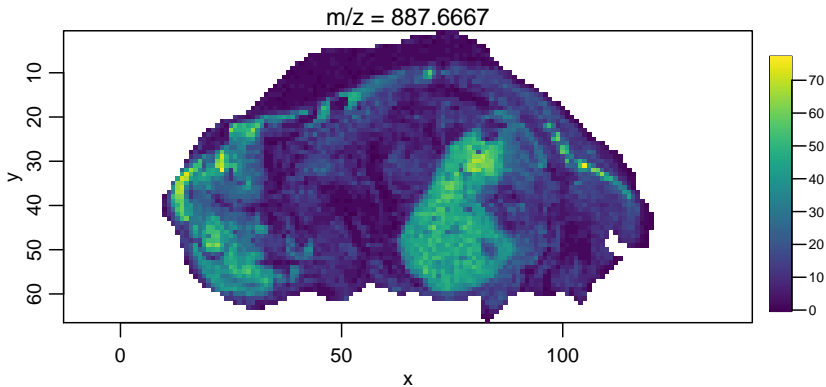
For segmentation, we will use data from a pig fetus cross-section.

```
data(pig206, package="CardinalWorkflows")  
  
pig206_peaks <- as(pig206.peaks, "MSImagingExperiment")
```

\*Note: The datasets provided in `CardinalWorkflows` are stored in an older format, so we need to coerce them to an `MSImagingExperiment` using `as()`.\*

## Pig fetus m/z 888

```
image(pig206_peaks, mz=888)
```





# Loading pre-calculated segmentation results

To save time, we will load some pre-calculated segmentation results.

We'll load results for two types of spatial smoothing  
(method="gaussian" and 'method="adaptive").

```
data(pig206_analyses, package="CardinalWorkflows")
pig206_sscg <- as(pig206.sscg, "SpatialShrunkenCentroids2")
pig206_ssca <- as(pig206.ssca, "SpatialShrunkenCentroids2")
```

\*Note: The datasets provided in CardinalWorkflows are stored in an older format, so we need to coerce them to an MSImagingExperiment using as().\*

```
summary(pig206_sscg)
```

```
## Spatially-aware nearest shrunken centroids:
```

```
##
```

```
## Segmentation / clustering
```

```
## Method = gaussian
```

```
## Distance = chebyshev
```

```
##
```

##	Radius (r)	Init (k)	Sparsity (s)	Classes	Features/Class	BIC
## 1	1	15	0	15	143.00	20509.345
## 2	1	15	3	10	90.80	10191.825
## 3	1	15	6	7	77.29	6856.784
## 4	1	15	9	6	63.33	5465.215
## 5	1	20	0	19	143.00	25700.733
## 6	1	20	3	11	91.73	11194.021
## 7	1	20	6	8	74.38	7877.644
## 8	1	20	9	6	61.67	5505.197
## 9	2	15	0	13	143.00	18046.962
## 10	2	15	3	10	89.70	10204.595
## 11	2	15	6	6	82.17	6306.085
## 12	2	15	9	6	61.83	5178.595
## 13	2	20	0	18	143.00	24352.247
## 14	2	20	3	9	90.00	9300.596
## 15	2	20	6	6	82.33	6318.644
## 16	2	20	9	6	60.33	5039.033

```
summary(pig206_ssca)
```

```
## Spatially-aware nearest shrunken centroids:
```

```
##
```

```
## Segmentation / clustering
```

```
## Method = adaptive
```

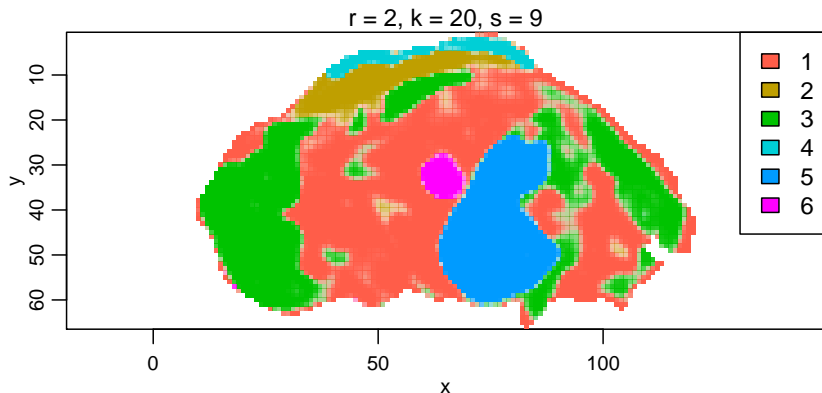
```
## Distance = chebyshev
```

```
##
```

##	Radius (r)	Init (k)	Sparsity (s)	Classes	Features/Class	BIC
## 1	1	15	0	15	143.00	20613.272
## 2	1	15	3	9	100.33	9999.905
## 3	1	15	6	6	85.33	6343.967
## 4	1	15	9	6	62.17	5470.961
## 5	1	20	0	20	143.00	27049.482
## 6	1	20	3	11	83.27	10140.171
## 7	1	20	6	8	68.62	6909.633
## 8	1	20	9	6	61.50	5427.203
## 9	2	15	0	15	143.00	20612.342
## 10	2	15	3	9	99.89	10048.146
## 11	2	15	6	8	69.38	7202.635
## 12	2	15	9	7	55.00	5729.158
## 13	2	20	0	18	143.00	24360.641
## 14	2	20	3	10	90.70	10128.729
## 15	2	20	6	7	78.29	6795.206
## 16	2	20	9	6	65.83	5466.079

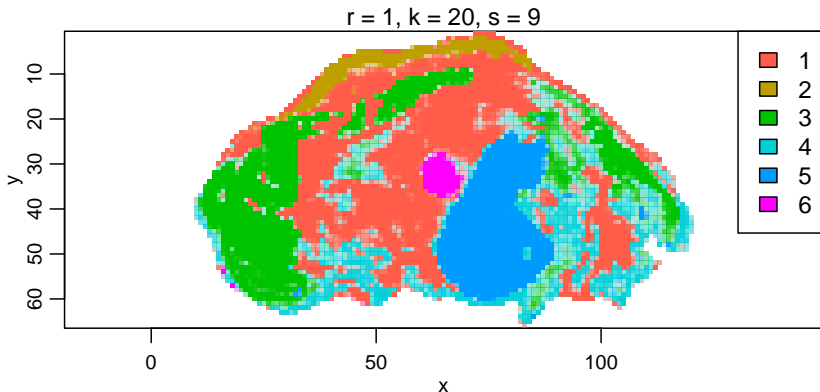
# Probability of segment membership (Gaussian smoothing)

```
image(pig206_sscg, model=list(r=2, k=20, s=9),  
      values="probability")
```



# Probability of segment membership (adaptive smoothing)

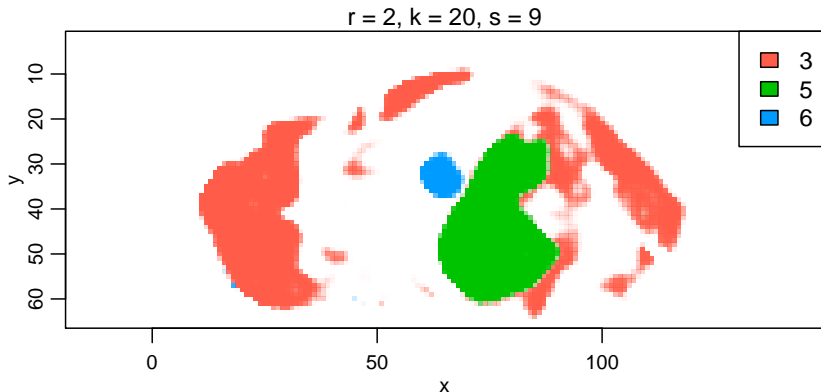
```
image(pig206_ssca, model=list(r=1, k=20, s=9),  
      values="probability")
```





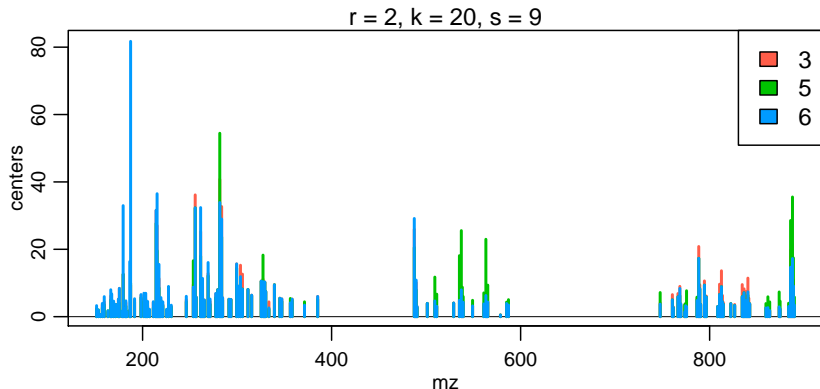
## Brain, liver, and heart segments

```
image(pig206_sscg, model=list(r=2, k=20, s=9),  
      values="probability", column=c(3,5,6))
```



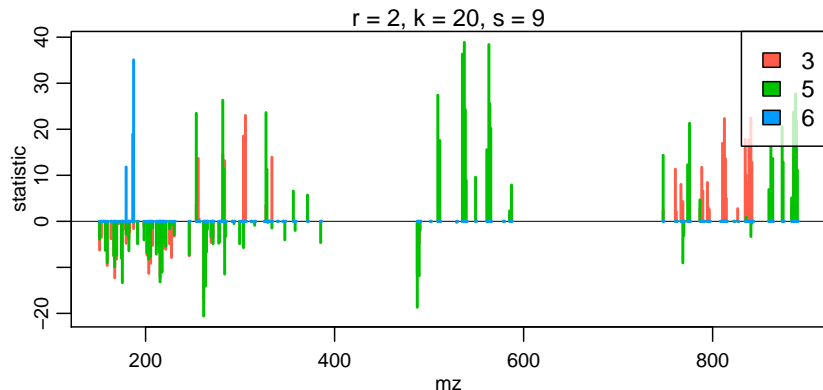
## Mean spectra for brain, liver, and heart

```
plot(pig206_sscg, model=list(r=2, k=20, s=9),  
     values="centers", column=c(3,5,6), lwd=2)
```



## Statistics for brain, liver, and heart

```
plot(pig206_sscg, model=list(r=2, k=20, s=9),  
     values="statistic", column=c(3,5,6), lwd=2)
```



## Most distinctive mass features for heart

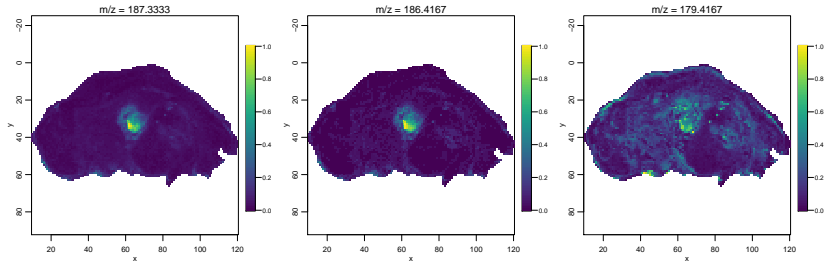
```
topFeatures(pig206_sscg, model=list(r=2, k=20, s=9), class==6)
```

## Top-ranked features:

##	mz	r	k	s	class	centers	statistic
## 1	187.3333	2	20	9	6	81.7135934	35.06540
## 2	186.4167	2	20	9	6	16.3548118	18.92717
## 3	179.4167	2	20	9	6	32.9546437	11.79906
## 4	151.3333	2	20	9	6	3.2799443	0.00000
## 5	153.2500	2	20	9	6	2.0027075	0.00000
## 6	155.2500	2	20	9	6	0.6662069	0.00000
## 7	157.4167	2	20	9	6	3.9341399	0.00000
## 8	159.4167	2	20	9	6	5.9435216	0.00000
## 9	163.3333	2	20	9	6	1.7407288	0.00000
## 10	166.3333	2	20	9	6	7.9829617	0.00000

# Most distinctive mass features for heart

```
image(pig206_peaks, mz=c(187.3, 186.4, 179.4),  
      normalize.image="linear", layout=c(1,3))
```



## Classification

## Renal cell carcinoma (RCC) dataset

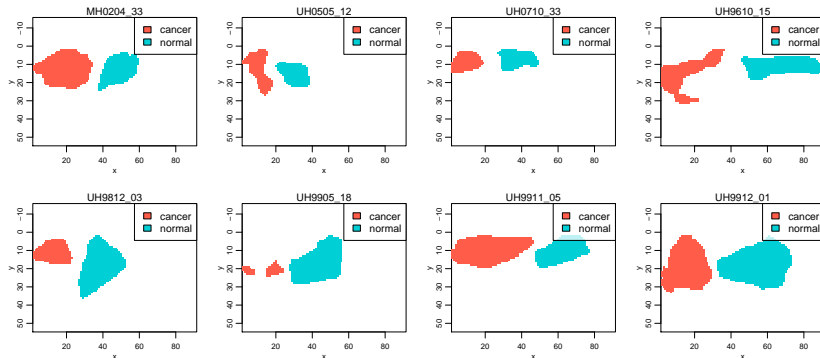
For classification, we will look at a dataset with 8 matched pairs of healthy tissue and tumor.

```
data(rcc, package="CardinalWorkflows")  
  
rcc_binned <- as(rcc.small, "MSImagingExperiment")
```

\*Note: The datasets provided in CardinalWorkflows are stored in an older format, so we need to coerce them to an MSImagingExperiment using as().\*

# Diagnosis

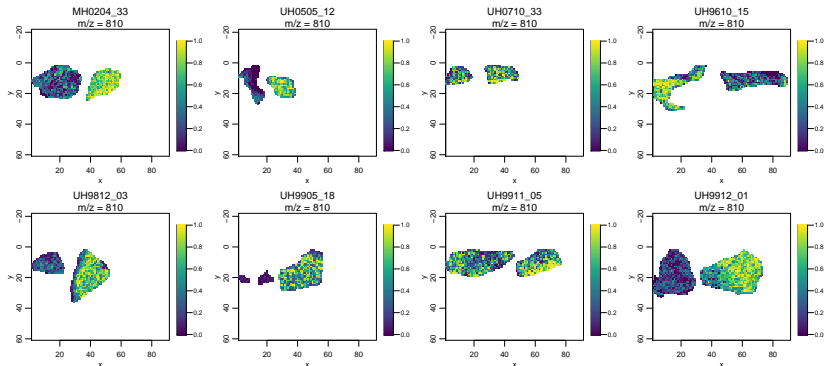
```
image(rcc_binned, diagnosis ~ x * y,  
      layout=c(2,4), key=TRUE)
```





# RCC m/z 810

```
image(rcc_binned, mz=810, layout=c(2,4),  
      contrast.enhance="histogram",  
      normalize.image="linear")
```



# Cross-validation

We use `crossValidate()` to perform cross-validation with spatial shrunk centroids classification.

Each run (with a cancer/normal matched pair) is a separate CV fold.

```
rcc_cv <- crossValidate(rcc_binned,  
                        .y=rcc_binned$diagnosis,  
                        .fun=spatialShrunkCentroids,  
                        r=1, s=0:9,  
                        .fold=run(rcc_binned))
```

## Loading pre-calculated CV results

To save time, we will load some pre-calculated CV results.

```
data(rcc_analyses, package="CardinalWorkflows")  
  
rcc_cv <- as(rcc_cv.sscg, "CrossValidated2")  
rcc_cv$response <- rcc_binned$diagnosis
```

\*Note: The CV results provided in CardinalWorkflows are stored in an older format, so we need to coerce them to the newer version of the class using as().\*

```
summary(rcc_cv)
```

```
## Cross validation:
##
## Classification on 2 classes: cancer normal
## Summarized 8 folds: MH0204_33 UH0505_12 ... UH9911_05 UH9912_01
##
##      r k  s Accuracy Sensitivity Specificity
## 1  1 2  0 0.8429914    0.8476989    0.8159756
## 2  2 2  0 0.8561886    0.8580030    0.8355094
## 3  3 2  0 0.8630894    0.8604488    0.8477564
## 4  1 2  4 0.8490092    0.8483200    0.8264378
## 5  2 2  4 0.8614576    0.8611654    0.8443781
## 6  3 2  4 0.8683942    0.8612681    0.8569161
## 7  1 2  8 0.8547938    0.8488032    0.8424877
## 8  2 2  8 0.8674976    0.8554014    0.8616460
## 9  3 2  8 0.8757184    0.8594661    0.8758228
## 10 1 2 12 0.8594844    0.8458832    0.8524716
## 11 2 2 12 0.8744938    0.8587803    0.8763088
## 12 3 2 12 0.8811212    0.8601993    0.8846880
## 13 1 2 16 0.8657251    0.8499447    0.8669211
## 14 2 2 16 0.8816412    0.8675844    0.8901845
## 15 3 2 16 0.8873204    0.8691277    0.9003744
## 16 1 2 20 0.8687747    0.8452182    0.8850448
## 17 2 2 20 0.8831311    0.8579025    0.9040489
## 18 3 2 20 0.8889099    0.8598918    0.9126307
## 19 1 2 24 0.8642629    0.8229296    0.9028367
## 20 2 2 24 0.8754163    0.8370316    0.9125450
## 21 3 2 24 0.8826314    0.8434021    0.9198243
## 22 1 2 28 0.8127775    0.6927374    0.9346619
## 23 2 2 28 0.8297853    0.7115986    0.9498645
## 24 3 2 28 0.8388546    0.7159251    0.9603586
```

Which is the most accurate model?

```
i <- which.max(summary(rcc_cv)$Accuracy)
summary(rcc_cv)[i,]
```

```
## Cross validation:
```

```
##
```

```
## Classification on 2 classes: cancer normal
```

```
## Summarized 8 folds: MH0204_33 UH0505_12 ... UH9911_05 UH9912_01
```

```
##
```

```
##      r k s Accuracy Sensitivity Specificity
```

```
## 18 3 2 20 0.8889099    0.8598918    0.9126307
```

Fit the model with the best parameters to the full dataset.

```
rcc_sscg <- spatialShrunkenCentroids(rcc_binned,  
                                     y=rcc_binned$diagnosis,  
                                     r=3, s=20)  
summary(rcc_sscg)
```

```
## Spatially-aware nearest shrunken centroids:
```

```
##
```

```
## Classification on 2 classes: cancer normal
```

```
## Method = gaussian
```

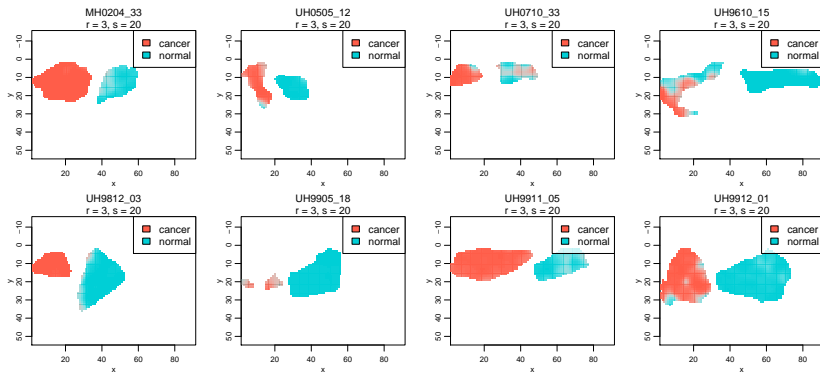
```
## Distance = chebyshev
```

```
##
```

##	Radius (r)	Sparsity (s)	Features/Class	Accuracy	Sensitivity	Specificity
## 1	3	20	43	0.948988	0.92	0.9733495

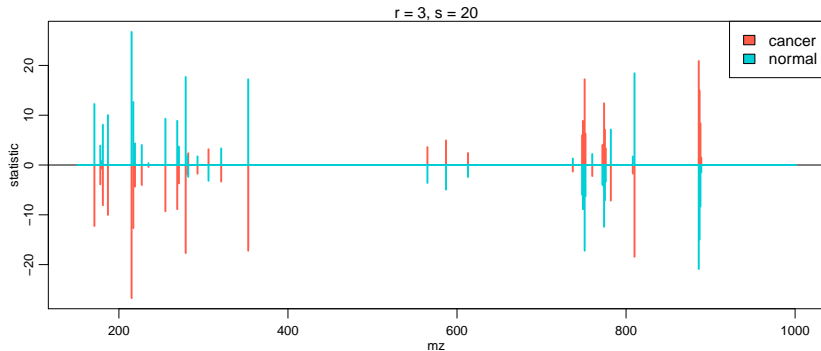
# Predicted probability of cancer/normal

```
image(rcc_sscg, values="probability", layout=c(2,4))
```



# Mass features over/under-expressed in each class

```
plot(rcc_sscg, values="statistic", lwd=2)
```





## Mass features associated with cancer

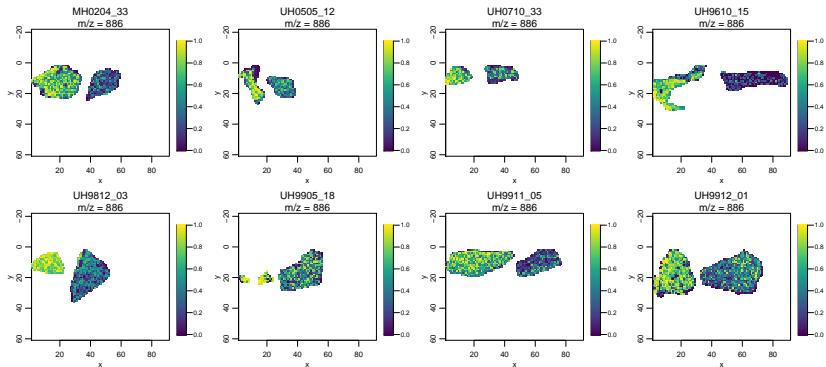
```
topFeatures(rcc_sscg, class=="cancer")
```

```
## Top-ranked features:
```

##	mz	r	k	s	class	centers	statistic
## 1	886	3	2	20	cancer	20.304817	20.877944
## 2	751	3	2	20	cancer	4.202790	17.209928
## 3	887	3	2	20	cancer	10.588845	14.976055
## 4	774	3	2	20	cancer	3.421733	12.399521
## 5	749	3	2	20	cancer	3.866871	8.859075
## 6	750	3	2	20	cancer	2.935949	8.789444
## 7	888	3	2	20	cancer	5.638888	8.341449
## 8	775	3	2	20	cancer	1.937996	7.084207
## 9	752	3	2	20	cancer	1.906379	6.289482
## 10	748	3	2	20	cancer	5.533633	5.977955

# m/z 886 associated with cancer

```
image(rcc_binned, mz=886, layout=c(2,4),  
      contrast.enhance="histogram",  
      normalize.image="linear")
```



## Mass features associated with normal

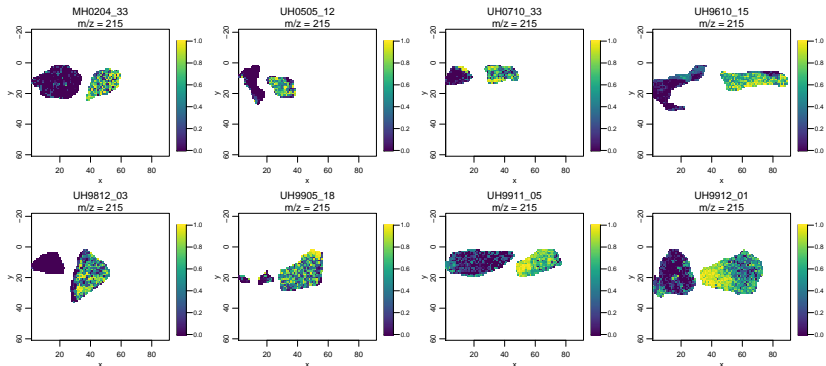
```
topFeatures(rcc_sscg, class=="normal")
```

```
## Top-ranked features:
```

##	mz	r	k	s	class	centers	statistic
## 1	215	3	2	20	normal	6.199255	26.722518
## 2	810	3	2	20	normal	8.845353	18.420318
## 3	279	3	2	20	normal	3.719320	17.676717
## 4	353	3	2	20	normal	2.371743	17.202849
## 5	217	3	2	20	normal	10.376391	12.651647
## 6	171	3	2	20	normal	3.432423	12.246610
## 7	187	3	2	20	normal	2.562495	10.017407
## 8	255	3	2	20	normal	4.666685	9.295445
## 9	269	3	2	20	normal	2.174021	8.869836
## 10	181	3	2	20	normal	3.628515	8.084145

# m/z 215 associated with normal

```
image(rcc_binned, mz=215, layout=c(2,4),  
      contrast.enhance="histogram",  
      normalize.image="linear")
```



## Statistical testing

# Statistical testing

Suppose we want to test which mass features are differentially abundant between cancer and normal tissue?

We have  $N = 8$  subjects per condition.

Two approaches we might take using Cardinal:

- ▶ Average all pixels in each tissue sample and compare means
- ▶ Segment each tissue and compare distinctive segments

Let's explore both.

## Filter mass features by intensity thresholding

```
rcc_filt <- rcc_binned %>%  
  mzFilter(thresh.max=0.25) %>%  
  process()
```

```
rcc_filt
```

```
## An object of class 'MSContinuousImagingExperiment'  
##   <24 feature, 6077 pixel> imaging dataset  
##     imageData(1): intensity  
##     featureData(0):  
##     pixelData(1): diagnosis  
##     processing complete(1): mzFilter  
##     processing pending(0):  
##     run(8): MH0204_33 UH0505_12 ... UH9911_05 UH9912_01  
##     raster dimensions: 90 x 36  
##     coord(2): x = 2..91, y = 2..37  
##     mass range: 179 to 888  
##     centroided: FALSE
```

## Means-based testing

We need to create a variable for each (subject x condition) combination.

Then we can use `meansTest()` to calculate the mean intensities for each group and fit linear models for each mass feature.

```
rcc_filt$sample <- interaction(run(rcc_filt),  
                               rcc_filt$diagnosis)  
  
rcc_mtest <- meansTest(rcc_filt, ~ diagnosis,  
                       groups=rcc_filt$sample)
```



```
summary(rcc_mtest)
```

```
## Means-summarized linear model testing:
##
## Fixed effects: ~diagnosis
##
## Summarized 16 groups: MH0204_33.cancer UH0505_12.cancer ... UH9911_05.normal
## Summarized      UH9912_01.normal
##
## Likelihood ratio test for fixed effects:
##
##      Feature      LR      PValue      FDR
## 1      1  3.0428 8.109691e-02 1.621938e-01
## 2      2 25.0468 5.595576e-07 1.342938e-05
## 3      3  5.3379 2.086668e-02 7.154290e-02
## 4      4  5.7995 1.603029e-02 6.412116e-02
## 5      5  0.1023 7.490393e-01 8.171338e-01
## 6      6  2.4427 1.180745e-01 1.771118e-01
## 7      7  3.6943 5.459962e-02 1.310391e-01
## 8      8  4.8178 2.816663e-02 7.511102e-02
## 9      9  3.2456 7.161508e-02 1.562511e-01
## 10     10 0.0685 7.934699e-01 8.279686e-01
## 11     11 2.1830 1.395439e-01 1.970032e-01
## 12     12 0.0036 9.521536e-01 9.521536e-01
## 13     13 2.6273 1.050369e-01 1.771118e-01
## 14     14 1.3544 2.445037e-01 2.934045e-01
## 15     15 2.4443 1.179504e-01 1.771118e-01
## 16     16 0.6841 4.081620e-01 4.664709e-01
## 17     17 2.0003 1.572729e-01 2.096973e-01
## 18     18 5.0775 2.423862e-02 7.271586e-02
## 19     19 12.5661 3.928116e-04 4.713740e-03
## 20     20 1.7581 1.848666e-01 2.335158e-01
## 21     21 2.6737 1.020161e-01 1.771118e-01
## 22     22 10.1421 1.449245e-03 1.159396e-02
## 23     23 7.4050 6.504441e-03 3.902665e-02
## 24     24 6.0384 1.399762e-02 6.412116e-02
```

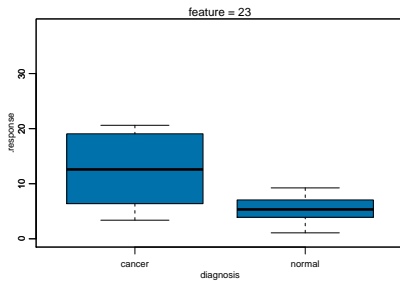
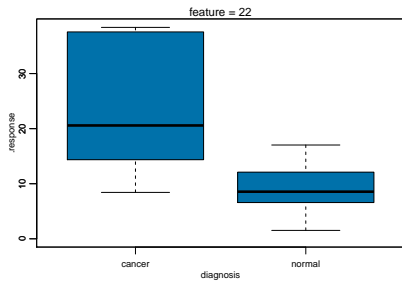
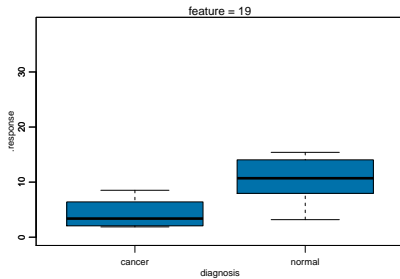
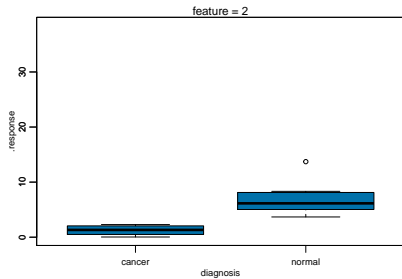
## Significant mass features

```
topFeatures(rcc_mtest, p.adjust="fdr", AdjP < 0.05)
```

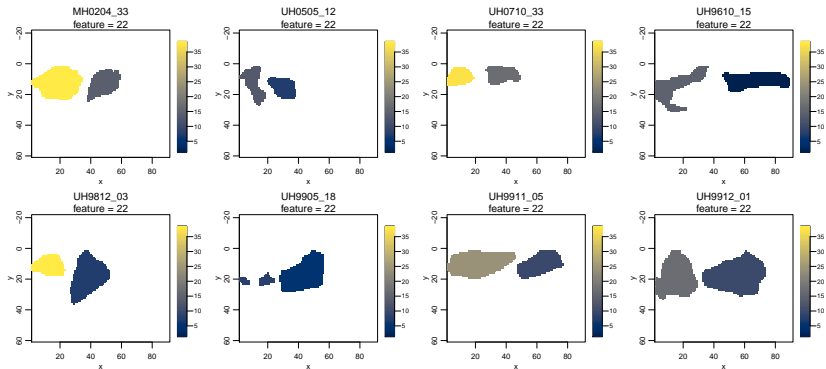
```
## Top-ranked tests: ~diagnosis vs ~1
```

##	mz	feature	LR	PValue	AdjP
## 1	215	2	25.0468	5.595576e-07	1.342938e-05
## 2	810	19	12.5661	3.928116e-04	4.713740e-03
## 3	886	22	10.1421	1.449245e-03	1.159396e-02
## 4	887	23	7.4050	6.504441e-03	3.902665e-02

```
plot(rcc_mtest, model=c(2,19,22,23))
```



```
image(rcc_mtest, model=22, layout=c(2,4))
```



## Segmentation-based testing

To represent potential heterogeneity within the tissue, we can segment each tissue first and then compare the most mean intensities for the segments.

First we use `spatialDGMM()` to fit segmentations for each mass feature, and then `segmentationTest()` fits linear models to the segments.

```
rcc_dgmm <- spatialDGMM(rcc_filt,  
                        r=1, k=2,  
                        groups=rcc_filt$sample)  
  
rcc_stest <- segmentationTest(rcc_dgmm, ~ diagnosis)
```

```
summary(rcc_stest)
```

```
## Segmentation-based linear model testing:
```

```
##
```

```
## Fixed effects: ~diagnosis
```

```
##
```

```
## Summarized 16 groups: MH0204_33.cancer UH0505_12.cancer ... UH9911_05.normal
```

```
## Summarized      UH9912_01.normal
```

```
##
```

```
## Likelihood ratio test for fixed effects:
```

```
##
```

##	Radius (r)	Init (k)	Feature	LR	PValue	FDR
## 1	1	2	1	0.5200	4.708211e-01	0.5649853301
## 2	1	2	2	15.4995	8.252627e-05	0.0009903152
## 3	1	2	3	4.3195	3.767813e-02	0.1507125286
## 4	1	2	4	2.5316	1.115837e-01	0.2434553525
## 5	1	2	5	0.0107	9.175941e-01	0.9574895198
## 6	1	2	6	1.2433	2.648356e-01	0.4254695750
## 7	1	2	7	2.8573	9.096265e-02	0.2334429157
## 8	1	2	8	3.2931	6.957074e-02	0.2087122291
## 9	1	2	9	0.0762	7.825407e-01	0.8536807610
## 10	1	2	10	0.1130	7.367214e-01	0.8419673345
## 11	1	2	11	0.9926	3.191022e-01	0.4254695750
## 12	1	2	12	0.0027	9.582721e-01	0.9582720933
## 13	1	2	13	2.7498	9.726788e-02	0.2334429157
## 14	1	2	14	1.0225	3.119236e-01	0.4254695750
## 15	1	2	15	2.0393	1.532768e-01	0.3065535846
## 16	1	2	16	1.0783	2.990742e-01	0.4254695750
## 17	1	2	17	1.2442	2.646563e-01	0.4254695750
## 18	1	2	18	1.0108	3.147026e-01	0.4254695750
## 19	1	2	19	19.4299	1.043600e-05	0.0002504640
## 20	1	2	20	0.6660	4.144655e-01	0.5235353759
## 21	1	2	21	4.8176	2.817053e-02	0.1352185376
## 22	1	2	22	10.8951	9.641960e-04	0.0077135677
## 23	1	2	23	8.2509	4.073273e-03	0.0244396377
## 24	1	2	24	3.5107	6.097466e-02	0.2087122291

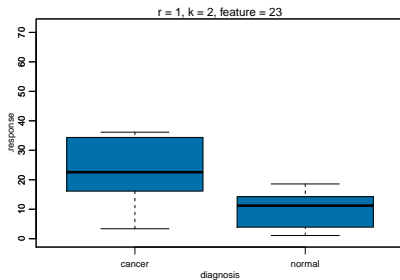
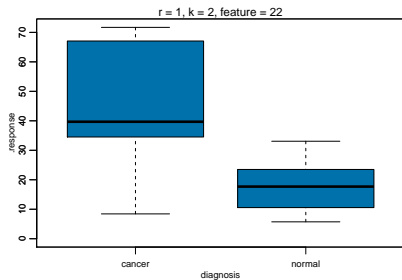
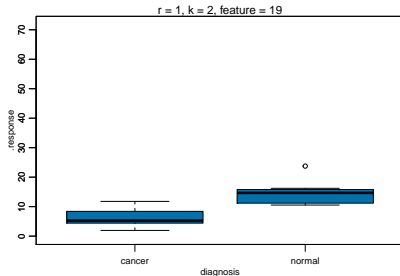
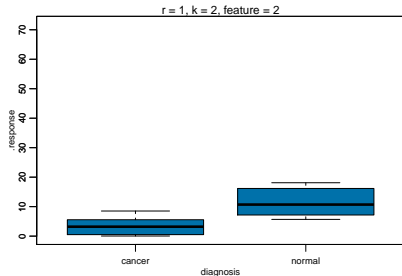
## Significant mass features

```
topFeatures(rcc_stest, p.adjust="fdr", AdjP < 0.05)
```

```
## Top-ranked tests: ~diagnosis vs ~1
```

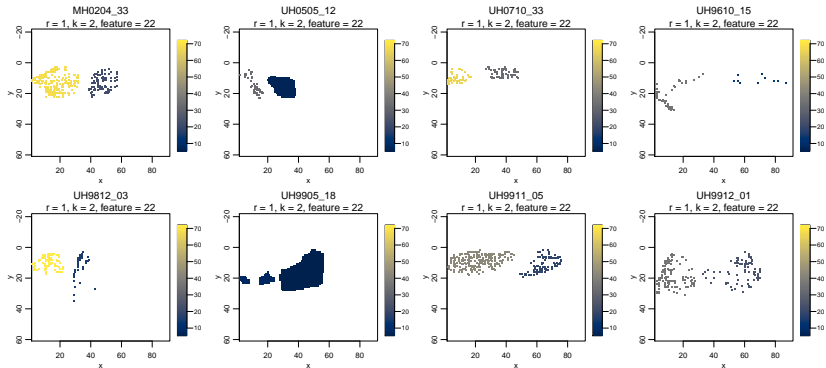
##	mz	r	k	feature	LR	PValue	AdjP
## 1	810	1	2	19	19.4299	1.043600e-05	0.0002504640
## 2	215	1	2	2	15.4995	8.252627e-05	0.0009903152
## 3	886	1	2	22	10.8951	9.641960e-04	0.0077135677
## 4	887	1	2	23	8.2509	4.073273e-03	0.0244396377

```
plot(rcc_stest, model=c(2,19,22,23))
```





```
image(rcc_stest, model=22, layout=c(2,4))
```



## Find mass features colocalized with m/z 886

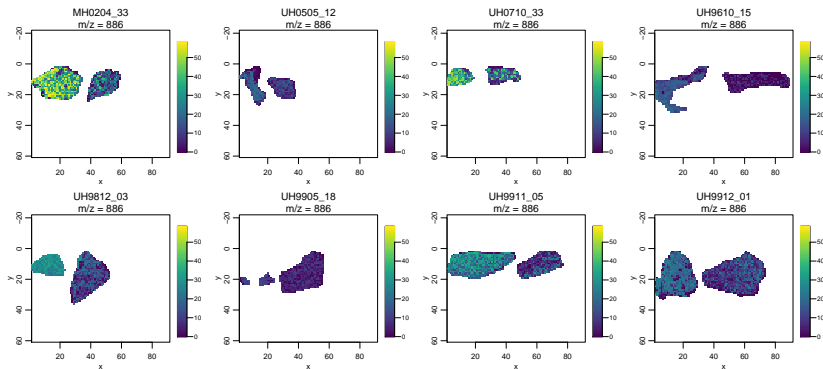
```
colocalized(rcc_binned, mz=886)
```

```
## Colocalized features:
```

##	mz	correlation	M1	M2
## 1	886	1.0000000	1.0000000	1.0000000
## 2	887	0.8842467	0.8390388	0.8390388
## 3	751	0.7898111	0.7169190	0.7176277
## 4	767	0.7423926	0.7238315	0.7238315
## 5	749	0.7419686	0.6846610	0.6851120
## 6	888	0.7355489	0.7718894	0.7718894
## 7	752	0.7204633	0.7090191	0.7092526
## 8	723	0.6955917	0.6770902	0.6773131
## 9	889	0.6708970	0.6583278	0.6587615
## 10	858	0.6443805	0.5687953	0.5691700

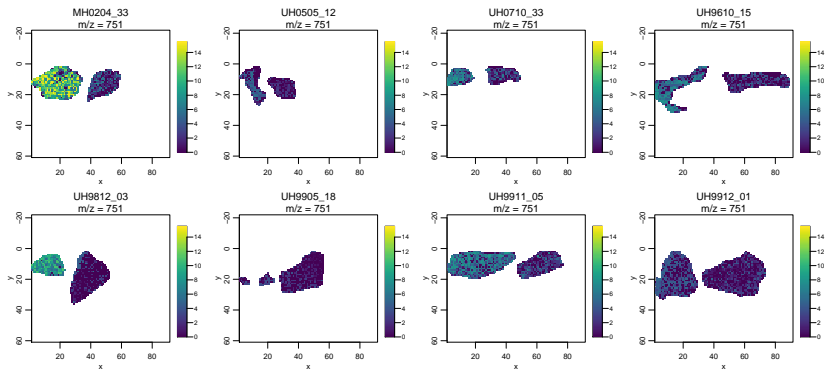
# RCC m/z 886

```
image(rcc_binned, mz=886, layout=c(2,4),  
      contrast.enhance="histogram")
```



# RCC m/z 751

```
image(rcc_binned, mz=751, layout=c(2,4),  
      contrast.enhance="histogram")
```



Interactive visualization

# Interactive visualization

Shiny interface for Cardinal (under development) on Github...

```
if (!requireNamespace("remotes", quietly = TRUE))  
  install.packages("remotes")
```

```
BiocManager::install("kuwisdell/CardinalVis")
```

```
library(CardinalVis)
```

```
pig206 <- as(pig206, "MSImagingExperiment")
```

```
msiVis(pig206)
```

## Questions