SPATIAL TRANSCRIPTOMICS DATA ANALYSIS: THEORY AND PRACTICE

PRACTICAL SESSION 3

DR SIMON J COCKELL

ELEFTHERIOS (LEFTERIS) ZORMPAS

BIOSCIENCES INSTITUTE,

FACULTY OF MEDICAL SCIENCES,

NEWCASTLE UNIVERSITY

23/07/2023

Practical session 3

In this practical session, we will demonstrate the application of the most commonly used spatial analysis tools to STx data, and how we work with coordinate data alongside expression data.

Practical session 3

- spdep: a collection of functions to create spatial weights matrix objects and perform spatial data analysis like regional aggregation and tests for spatial *'autocorrelation'*.
- sf: *Simple Features for R* → a standardized way to encode spatial vector data.
- GWmodel: is a suite of models where there are spatial regions where a suitably localised calibration provides a better description.

3.1 Background

3.1.1 Main geocomputational data structures

a) Spatial geometries can be points, lines, polygons and pixels.

b) Neighbour lists are special types of lists that contain information about the neighbours of each polygon.
   - The neighbours can be defined either by adjacency or by distance.

c) Distance matrices contain the distances between different points and can be either weighted or unweighted.
   - The weighted distances are usually objective to each point and its neighbours.

3.1 Background

3.1.2 The *sf* objects

The following seven simple feature types are the most common:

type	description
`POINT`	zero-dimensional geometry containing a single point
`LINESTRING`	sequence of points connected by straight, non-self intersecting line pieces; one-dimensional geometry
`POLYGON`	geometry with a positive area (two-dimensional); sequence of points form a closed, non-self intersecting ring; the first ring denotes the exterior ring, zero or more subsequent rings denote holes in this exterior ring
`MULTIPOINT`	set of points; a MULTIPOINT is simple if no two Points in the MULTIPOINT are equal
`MULTILINESTRING`	set of linestrings
`MULTIPOLYGON`	set of polygons
`GEOMETRYCOLLECTION`	set of geometries of any type except GEOMETRYCOLLECTION

3.1 Background

3.1.2 The *sf* objects

```
class(nc)
```

```
## [1] "sf"         "data.frame"
```

```
print(nc[9:15], n = 3)
```

```
## Simple feature collection with 100 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## precision:      double (default; no precision model)
## First 3 features:
##   BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                           geom
## 1  1091     1      10  1364     0      19 MULTIPOLYGON(((-81.47275543...
## 2   487     0      10   542     3      12 MULTIPOLYGON(((-81.23989105...
## 3  3188     5     208  3616     6     260 MULTIPOLYGON(((-80.45634460...
```

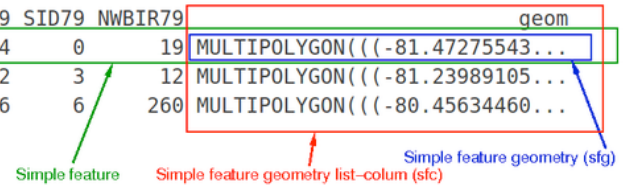


Figure 3.1: Overview of the `sf` object.

You can transform them into data frames only

```
nc.no_sf <- as.data.frame(nc)
class(nc.no_sf)
```

```
## [1] "data.frame"
```

However, such objects:

1. no longer register which column is the geometry list-column
2. no longer have a plot method, and
3. lack all of the other dedicated methods for class sf

3.1 Background

3.1.2 The *sf* objects: simple feature geometry list-column

```
(nc_geom <- st_geometry(nc))
```

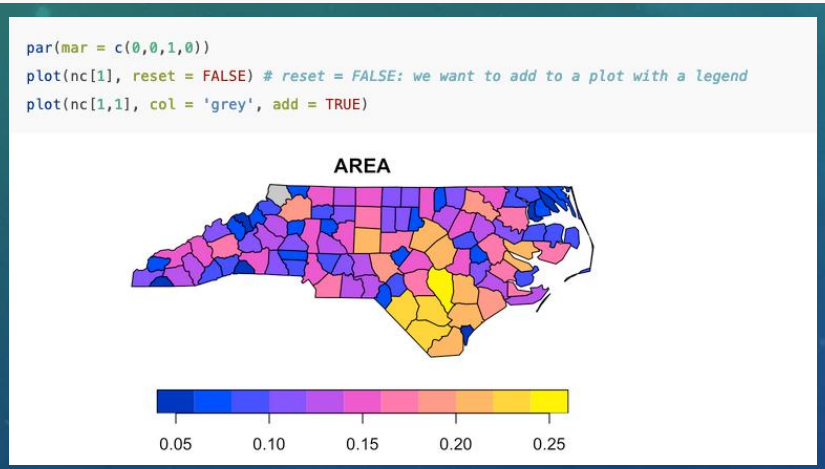
```
## Geometry set for 100 features
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
## First 5 geometries:
```

```
## MULTIPOLYGON (((-81.47276 36.23436, -81.54084 3...
```

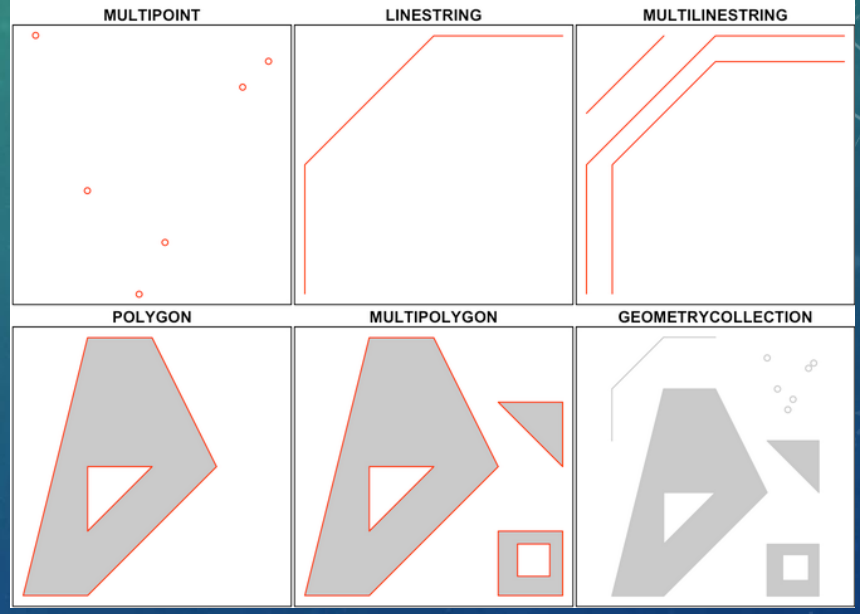
```
## MULTIPOLYGON (((-81.23989 36.36536, -81.24069 3...
```

```
## MULTIPOLYGON (((-80.45634 36.24256, -80.47639 3...
```

```
par(mar = c(0,0,1,0))
plot(nc[1], reset = FALSE) # reset = FALSE: we want to add to a plot with a legend
plot(nc[1,1], col = 'grey', add = TRUE)
```



The below figure illustrates the different types of geometries:



3.2 Data structures preparation

In this tutorial session, we will be using a human steatotic kidney dataset from the Liver Atlas (Guilliams et al. 2022). Specifically, we will use the *JBO019* sample.

We will also use a modified S4 object: SpatialFeaturesExperiment object which is an extension of the SpatialExperiment (SPE) object that we used in the 2nd practical session.

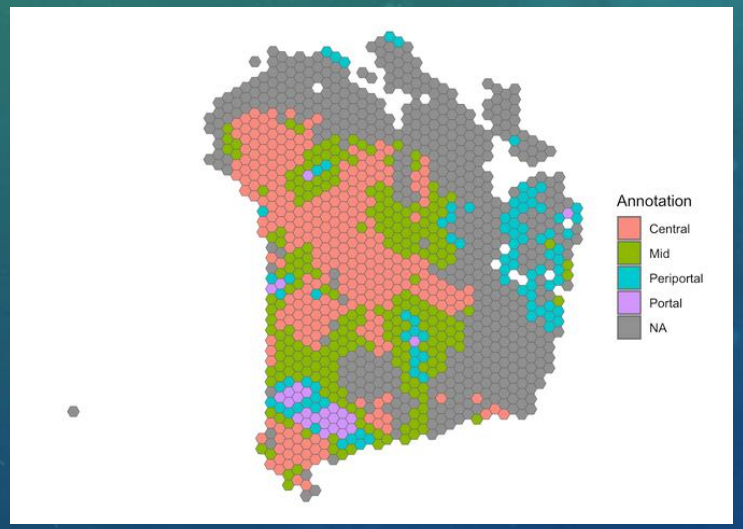
➢To Do: Load the data and work on the practical's code

3.3 Spot-level Quality Control

Calculating QC metrics

```
is_mito <- grepl("(^MT-)|(^mt-)", rowData(sfe)$symbol)
sfe <- addPerLocQC(sfe, gTruth = ground_truth, assay = "counts", 2, subsets = list(mito = is_mito))
sfe <- addGeometries(sfe, samples = sampleDir, sample_id = sampleNames, res = "fullres")
sfe <- addPerGeneQC(sfe, assay = "counts", version = NULL, mirror = NULL)
```

Plotting annotation



colData

DataFrame with 1185 rows and 15 columns

	in_tissue	array_row	array_col	sample_id	Barcode	Capt_area	annotation	index	sparsity
	<logical>	<integer>	<integer>	<character>	<character>	<character>	<character>	<character>	<numeric>
AAACAAGTATCTCCCA-1	TRUE	50	102	JBO019	AAACAAGTATCTCCCA-1	1	NA	spot_1	0.910410
AAACATTTCCCGGATT-1	TRUE	61	97	JBO019	AAACATTTCCCGGATT-1	1	NA	spot_2	0.967805
AAACCCGAACGAAATC-1	TRUE	45	115	JBO019	AAACCCGAACGAAATC-1	1	Mid	spot_3	0.864958
AAACGAGACGGTTGAT-1	TRUE	35	79	JBO019	AAACGAGACGGTTGAT-1	1	Central	spot_4	0.835818
AAACTAACGTGGCGAC-1	TRUE	8	110	JBO019	AAACTAACGTGGCGAC-1	1	NA	spot_5	0.995418
...	...	...	...	...	...	...	...	...	...
TTGTAATCCGTACTCG-1	TRUE	35	55	JBO019	TTGTAATCCGTACTCG-1	1	NA	spot_1181	0.933716
TTGTGAACCTAATCCG-1	TRUE	56	90	JBO019	TTGTGAACCTAATCCG-1	1	NA	spot_1182	0.955831
TTGTGCAGCCACGTCA-1	TRUE	60	74	JBO019	TTGTGCAGCCACGTCA-1	1	NA	spot_1183	0.978252
TTGTGTTTCCCGAAAG-1	TRUE	51	59	JBO019	TTGTGTTTCCCGAAAG-1	1	NA	spot_1184	0.956778
TTGTTGTGTGTCAAGA-1	TRUE	31	77	JBO019	TTGTTGTGTGTCAAGA-1	1	Mid	spot_1185	0.852160

	sum	detected	subsets_mito_sum	subsets_mito_detected	subsets_mito_percent	total
	<numeric>	<integer>	<numeric>	<integer>	<numeric>	<numeric>
AAACAAGTATCTCCCA-1	13443	2933	1021	12	7.59503	13443
AAACATTTCCCGGATT-1	2648	1054	285	12	10.76284	2648
AAACCCGAACGAAATC-1	27733	4421	2087	12	7.52533	27733
AAACGAGACGGTTGAT-1	32973	5375	821	12	2.48992	32973
AAACTAACGTGGCGAC-1	400	150	182	11	45.50000	400
...	...	...	...	...	...	...
TTGTAATCCGTACTCG-1	7612	2170	733	11	9.62953	7612
TTGTGAACCTAATCCG-1	4299	1446	515	12	11.97953	4299
TTGTGCAGCCACGTCA-1	1452	712	54	10	3.71901	1452
TTGTGTTTCCCGAAAG-1	3831	1415	422	11	11.01540	3831
TTGTTGTGTGTCAAGA-1	27755	4840	906	12	3.26428	27755

rowData

DataFrame with 32738 rows and 17 columns

	gene_name	mean	detected	total	JBO019.sparsity	JBO019.total	JBO019.nLocations	JBO019.s_min	JBO019.max
	<character>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<integer>	<numeric>	<numeric>
ENSG00000243485	MIR1302-10	0.00000000	0.000000	0	1.000000	0	0	Inf	0
ENSG00000237613	FAM138A	0.00000000	0.000000	0	1.000000	0	0	Inf	0
ENSG00000186092	OR4F5	0.00000000	0.000000	0	1.000000	0	0	Inf	0
ENSG00000238009	RP11-34P13.7	0.00590717	0.590717	7	0.994093	7	7	1	1
ENSG00000239945	RP11-34P13.8	0.00000000	0.000000	0	1.000000	0	0	Inf	0
...	...	...	...	...	...	...	...	...	...
ENSG00000215635	AC145205.1	0	0	0	1	0	0	Inf	0
ENSG00000268590	BAGE5	0	0	0	1	0	0	Inf	0
ENSG00000251180	CU459201.1	0	0	0	1	0	0	Inf	0
ENSG00000215616	AC002321.2	0	0	0	1	0	0	Inf	0
ENSG00000215611	AC002321.1	0	0	0	1	0	0	Inf	0

	JBO019.s_mean	JBO019.s_median	JBO019.s_SD	JBO019.p_mean	JBO019.p_median	JBO019.p_SD	JBO019.s_CV	JBO019.p_CV
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG00000243485	NaN	NA	NA	0.00000000	0	0.0000000	NA	NaN
ENSG00000237613	NaN	NA	NA	0.00000000	0	0.0000000	NA	NaN
ENSG00000186092	NaN	NA	NA	0.00000000	0	0.0000000	NA	NaN
ENSG00000238009	1	1	0	0.00590717	0	0.0766631	0	1297.8
ENSG00000239945	NaN	NA	NA	0.00000000	0	0.0000000	NA	NaN
...	...	...	...	...	...	...	...	...
ENSG00000215635	NaN	NA	NA	0	0	0	NA	NaN
ENSG00000268590	NaN	NA	NA	0	0	0	NA	NaN
ENSG00000251180	NaN	NA	NA	0	0	0	NA	NaN
ENSG00000215616	NaN	NA	NA	0	0	0	NA	NaN
ENSG00000215611	NaN	NA	NA	0	0	0	NA	NaN

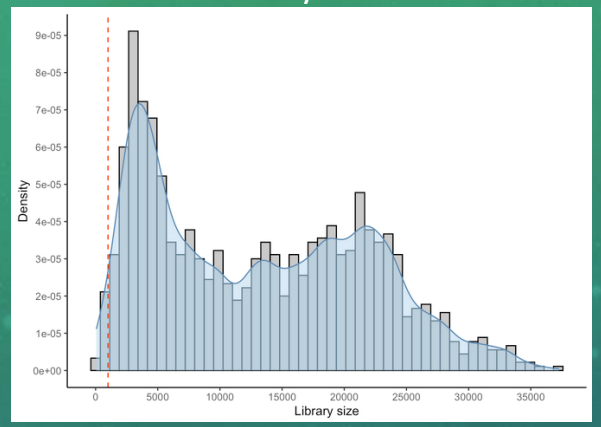
colGeometries

```
List of length 3
names(3): spotPoly spotCntd spotHex
```

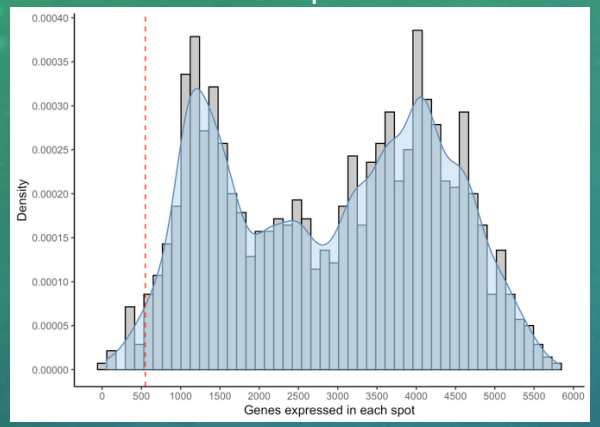
3.3 Spot-level Quality Control

Selecting thresholds

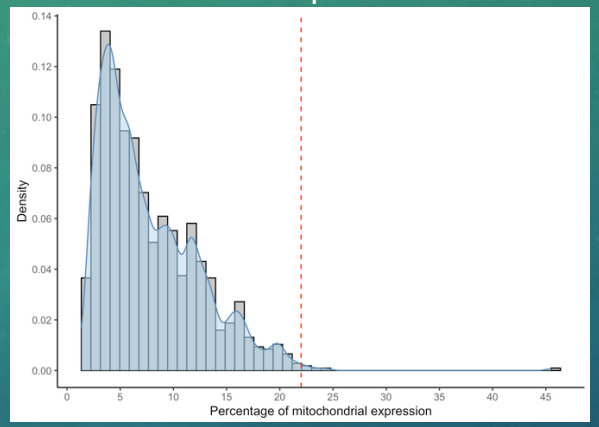
Library size



Genes expressed

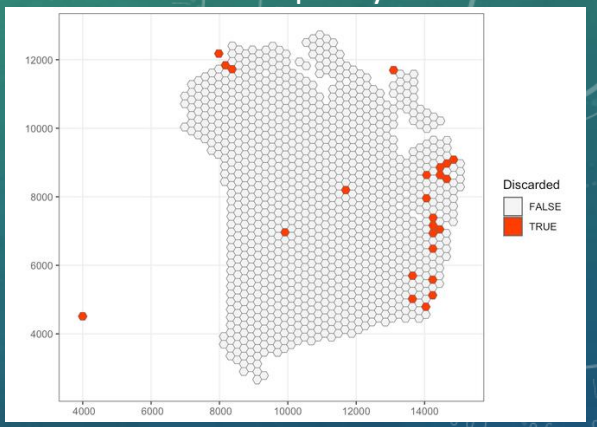


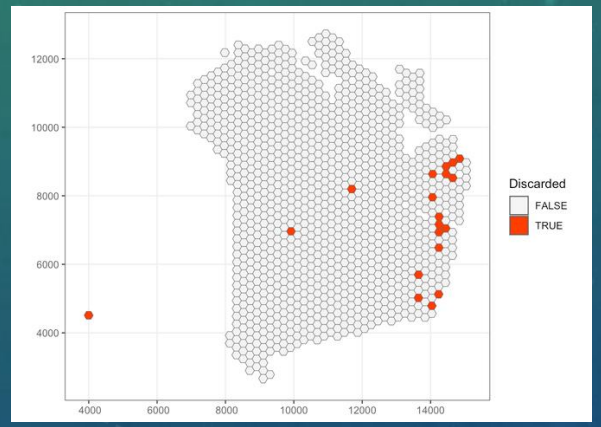
% mito expression

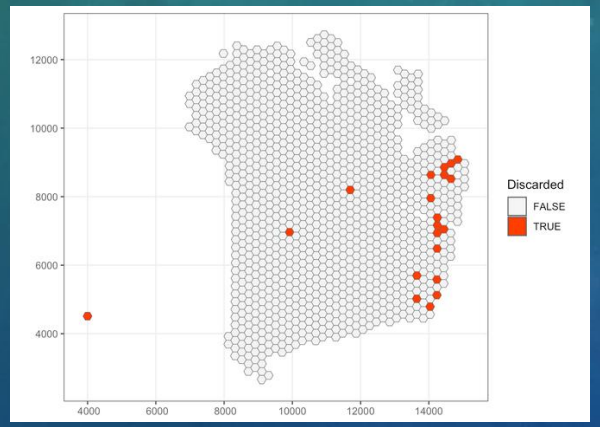


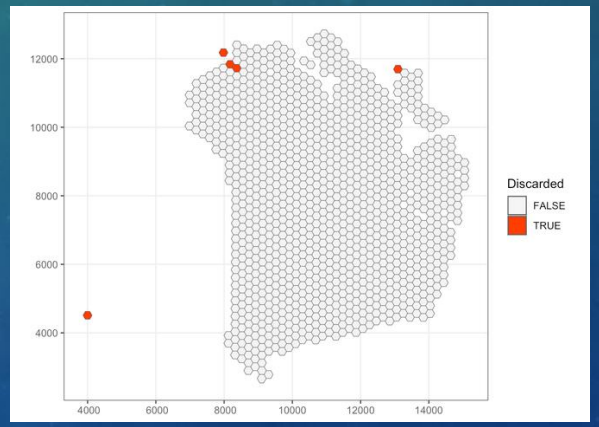
Applying thresholds

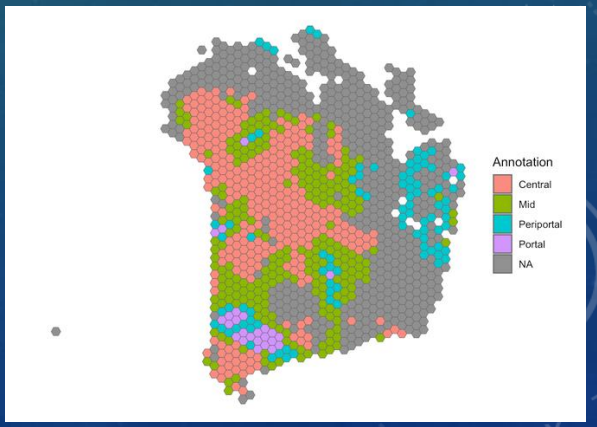
Discarded low-quality locations









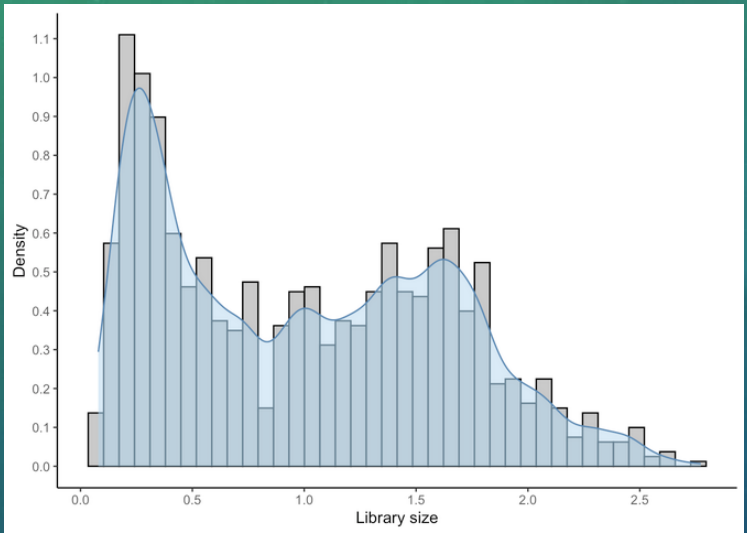


Our QC does not remove any biology – so we can assume it is correct and move on.

3.4 Normalisation of counts

```
## Calculate library size factors
sfe <- computeLibraryFactors(sfe)
## Have a look at the size factors
summary(sizeFactors(sfe))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07961 0.36902 0.95469 1.00000 1.54936 2.77256
```



Library size factors are used for the normalisation

```
# calculate logcounts using library size factors
sfe <- logNormCounts(sfe)
```

3.5 Gene-level Quality Control

Calculating extra QC metrics

```
rowData(sfe)[["JBO019.s_logMean"]] <- rowSums(assay(sfe, "logcounts")) / rowData(sfe)[["JBO019.nLocations"]]
```

Set and apply filters

```
is_zero <- rowData(sfe)$total == 0
is_logLow <- rowData(sfe)[["JBO019.s_logMean"]] <= 1
discard_gs <- is_zero | is_mito | is_logLow
table(discard_gs)
```

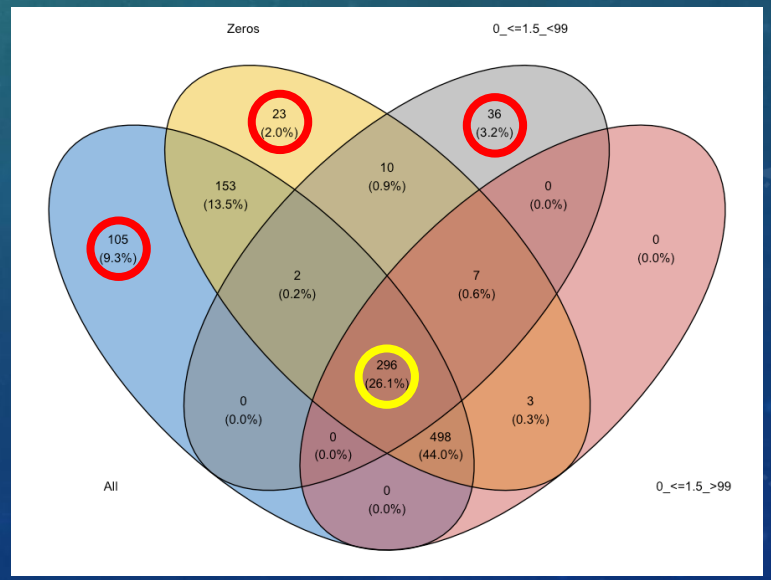
```
## discard_gs
## FALSE  TRUE
##  8535 24203
```

```
rowData(sfe)$discard <- discard_gs

## FEATURE SELECTION
## remove mitochondrial and other genes
sfe <- sfe[!rowData(sfe)$discard, ]
```

- ✓ Same dataset
- ✓ Same spot QC
- ✓ Same log-normalisation
- ✓ Same `modelGeneVar( )` parameters
- ✓ Same `getTopHVGs( )` parameters
  - ➢ *var.field = bio*
  - ➢ *prop = 0.1*
  - ➢ *var.threshold = 0*
  - ➢ *fdr.threshold = NULL*

- ❖ Different filtering after log-normalisation/before variance modelling:
  - All: no filtering at all
    (32738 genes input)
  - Zeros: filtered non-expressed genes
    (19157 genes input)
  - 0_<=1.5_<99: filtered zeros + sample mean < 1.5 + sparsity < 99%
    (7709 genes input)
  - 0_<=1.5_>99: filtered zeros + sample mean < 1.5 + sparsity > 99%
    (13788 genes input)

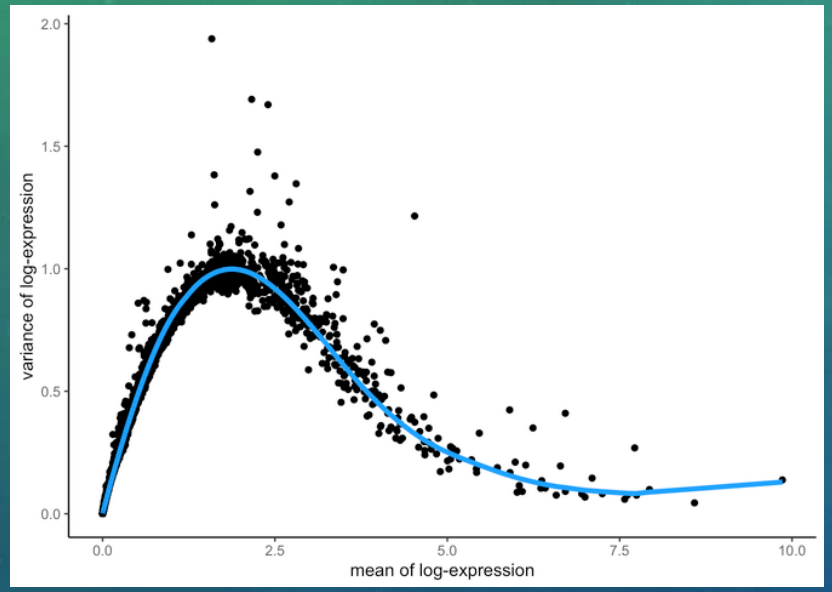


Unique

Common between all

2.3 Selecting genes

Highly Variable Genes (HVGs)




```r
## Select top HVGs
top_hvgs <- getTopHVGs(dec,
                       var.field = "bio",
                       prop = 0.5,
                       var.threshold = 0,
                       fdr.threshold =  0.05)
```

We select the top 50% of genes based on their variability with an FDR < 0.05.

3.7 Neighbour graph and distance matrix

Adding spatial weights and neighbours

Neighbour relationships coding styles

1. **B**: is the basic binary coding (1 for neighbour, 0 for no neighbour).
2. **W**: is row standardised (sums over all links to n).
3. **C**: is globally standardised (sums over all links to n).
4. **U**: is equal to C divided by the number of neighbours (sums over all links to unity).
5. **S**: is the variance-stabilizing coding scheme (sums over all links to n).
6. **minmax**: divides the weights by the minimum of the maximum row sums and maximum column sums of the input weights; It is similar to the C and U styles.

Binary W matrix

	A	B	C	D	E	F	G	H	I	Row Sum
A	0	1	0	1	1	0	0	0	0	3
B	1	0	1	1	1	1	0	0	0	5
C	0	1	0	0	1	1	0	0	0	3
D	1	1	0	0	1	0	1	1	0	5
E	1	1	1	1	0	1	1	1	1	8
F	0	1	1	0	1	0	0	1	1	5
G	0	0	0	1	1	0	0	1	0	3
H	0	0	0	1	1	1	1	0	1	5
I	0	0	0	0	1	1	0	1	0	3

Row standardised W matrix

	A	B	C	D	E	F	G	H	I	Row Sum
A	0	0.3	0	0.3	0.3	0	0	0	0	1
B	0.2	0	0.2	0.2	0.2	0.2	0	0	0	1
C	0	0.3	0	0	0.3	0.3	0	0	0	1
D	0.2	0.2	0	0	0.2	0	0.2	0.2	0	1
E	0.12	0.12	0.12	0.12	0	0.12	0.12	0.12	0.12	1
F	0	0.2	0.2	0	0.2	0	0	0.2	0.2	1
G	0	0	0	0.3	0.3	0	0	0.3	0	1
H	0	0	0	0.2	0.2	0.2	0.2	0	0.2	1
I	0	0	0	0	0.3	0.3	0	0.3	0	1

3.7 Neighbour graph and distance matrix

Adding spatial weights and neighbours

Neighbour graph types

(a) Contiguity-based - "poly2nb"

(b) Graph-based - "tri2nb", "soi.graph", "gabrielneigh", "relativeneigh"

(c) Distance-based - "knearneigh", "dnearneigh"

Distance-based types of weights

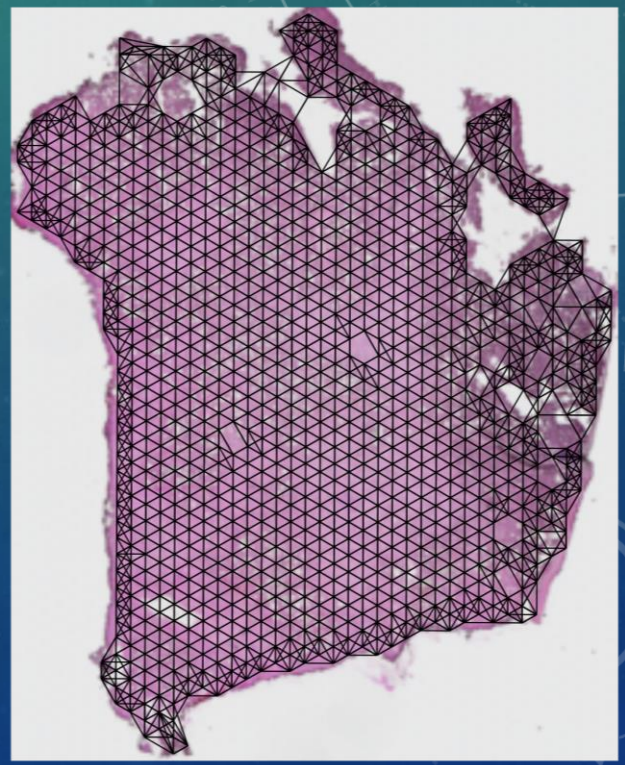
- IDW - inverse distance weighting
- DPD - double-power distance weights
- EXP - exponential distance decay

1. **idw**: $wij=d-\alpha ij$,
2. **exp**: $wij=\exp(-\alpha \cdot dij)$,
3. **dpd**: $wij=[1-(dij/d\max)\alpha]\alpha$,

```
## Add neighbour graph and distance matrix
sfe <- addSpatialNeighGraphs(sfe, sampleName, type = "knearneigh", style = "W", distMod = "raw", k = 6)
```

colGraphs

```
## $col
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 1161
## Number of nonzero links: 6966
## Percentage nonzero weights: 0.5167959
## Average number of links: 6
## Non-symmetric neighbours list
##
## Weights style: W
## Weights constants summary:
##      n       nn   S0       S1       S2
## W 1161 1347921 1161 376.8333 4674.667
```



3.7 Neighbour graph and distance matrix

Generate distance matrices

```
## Calculate a simple distance matrix
sfe <- addDistMat(sfe, p = 2)
```

Global Model	$w_{ij} = 1$
Gaussian	$w_{ij} = \exp\left(-\frac{1}{2}\left(\frac{d_{ij}}{b}\right)^2\right)$
Exponential	$w_{ij} = \exp\left(-\frac{\lvert d_{ij}\rvert}{b}\right)$
Box-car	$w_{ij} = \begin{cases} 1 & \text{if } \lvert d_{ij}\rvert < b, \\ 0 & \text{otherwise} \end{cases}$
Bi-square	$w_{ij} = \begin{cases} (1-(d_{ij}/b)^2)^2 & \text{if } \lvert d_{ij}\rvert < b, \\ 0 & \text{otherwise} \end{cases}$
Tri-cube	$w_{ij} = \begin{cases} (1-(\lvert d_{ij}\rvert/b)^3)^3 & \text{if } \lvert d_{ij}\rvert < b, \\ 0 & \text{otherwise} \end{cases}$

Table 1: Six kernel functions; w_{ij} is the j-th element of the diagonal of the matrix of geographical weights $W(u_i, v_i)$, and d_{ij} is the distance between observations i and j, and b is the bandwidth.

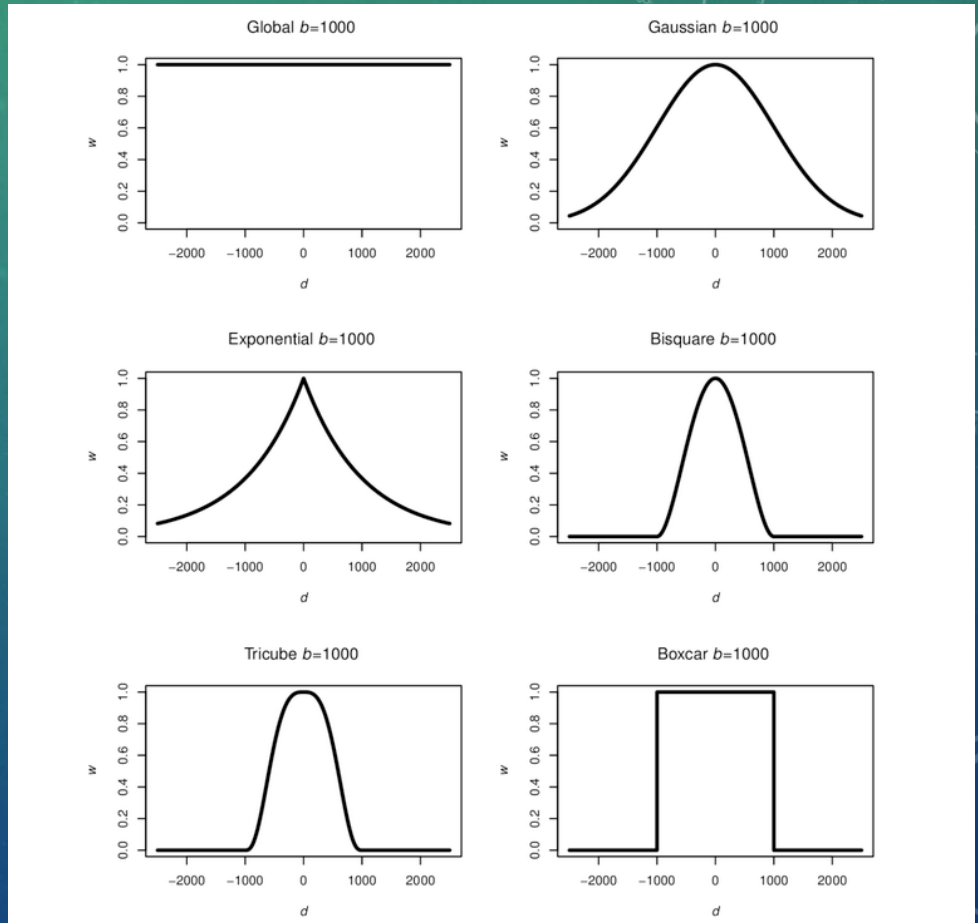


Figure 1: Plot of the six kernel functions, with the bandwidth $b = 1000$, and where w is the weight, and d is the distance between two observations.

AKNOWLEDGEMENTS

Eleftherios Zormpas

Newcastle University

Dr Simon J Cockell

Newcastle University

Dr Rachel Queen

Newcastle University

Prof. Alex Comber

UNIVERSITY OF LEEDS

iSMB feedback form:

ISMB ECCB 2023 LYON FRANCE ISCB

Newcastle University

UKRI Medical Research Council

MRC DiMeN Doctoral Training Partnership