Chapter 5 examples

Brooke Anderson

3/4/2020

```
library(tidyverse)
library(viridis)
```

Computing distances

```
ex <- tibble(mx = c(rep(0, 3), rep(1, 3)),
       my = c(1, 0, 1, 1, 0, 1),
       mz = c(rep(1, 3), 0, 1, 1))
ex
```

```
## # A tibble: 6 x 3
##      mx    my    mz
##   <dbl> <dbl> <dbl>
## 1     0     1     1
## 2     0     0     1
## 3     0     1     1
## 4     1     1     0
## 5     1     0     1
## 6     1     1     1
```

```
ex %>%
  as.matrix() %>%  # Convert the dataframe to a matrix (required input for `dist`) %>%
  t() %>%  # Transpose the matrix
  dist()
```

```
##          mx       my
## my 1.732051
## mz 2.000000 1.732051
```

This has created a triangle-shaped object with the "dist" class. Each entry in the triangle gives the distance between one pair of observations. You can easily change the distance metric that's calculated for these entries with the `method` parameter. The default is Euclidean distance, but you can also put "maximum", "manhattan", "canberra", "binary" or "minkowski".

For example, to incstead calculate the distances using the *binary distance metric*, change the code to:

```
ex %>%
  as.matrix() %>%
  t() %>%
  dist(method = "binary")
```

```
##           mx        my
## my 0.6000000
## mz 0.6666667 0.5000000
```

With another call to `as.matrix`, you can reconvert from a "dist" class object to a matrix:

```
ex %>%
  as.matrix() %>%
  t() %>%
  dist(method = "binary") %>%
  as.matrix()
```

```
##           mx  my        mz
## mx 0.0000000 0.6 0.6666667
## my 0.6000000 0.0 0.5000000
## mz 0.6666667 0.5 0.0000000
```

Notice that the cells of the triangle get repeated twice in this matrix format.

Trying to calculate distance with HIV strains

```
mut <- read_csv("data/HIVmutations.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
mut
```

```
## # A tibble: 5 x 57
##    p10F  p10R  p11I  p20I  p20T  p20V  p23I  p24I  p30N  p32I  p33F  p34Q  p35G
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     0     0     0     0     0     0     0     0     1     0     0
## 2     0     0     0     0     0     0     0     0     0     0     1     0     0
## 3     0     0     0     0     1     0     0     0     0     0     1     0     0
## 4     0     0     0     0     0     0     0     1     0     0     1     1     0
## 5     0     0     0     0     1     0     0     0     0     0     0     0     0
## # ... with 44 more variables: p43T <dbl>, p46I <dbl>, p46L <dbl>, p46V <dbl>,
## #   p47A <dbl>, p47V <dbl>, p48M <dbl>, p50L <dbl>, p50V <dbl>, p53L <dbl>,
## #   p53Y <dbl>, p54A <dbl>, p54L <dbl>, p54M <dbl>, p54S <dbl>, p54T <dbl>,
## #   p54V <dbl>, p55R <dbl>, p58E <dbl>, p66F <dbl>, p67F <dbl>, p71I <dbl>,
## #   p73A <dbl>, p73C <dbl>, p73S <dbl>, p73T <dbl>, p74A <dbl>, p74P <dbl>,
## #   p74S <dbl>, p76V <dbl>, p79A <dbl>, p82A <dbl>, p82F <dbl>, p82S <dbl>,
## #   p82T <dbl>, p84A <dbl>, p84C <dbl>, p84V <dbl>, p85V <dbl>, p88S <dbl>,
## #   p88T <dbl>, p89V <dbl>, p90M <dbl>, p95F <dbl>
```

I think that for this each row is a strain and each column is a gene (?), where a "1" indicates a mutation and a "0" indicates none.

We can use an tile plot to check it out. It'll be easier to do this if you first "pivot" the data to make it longer, like this:

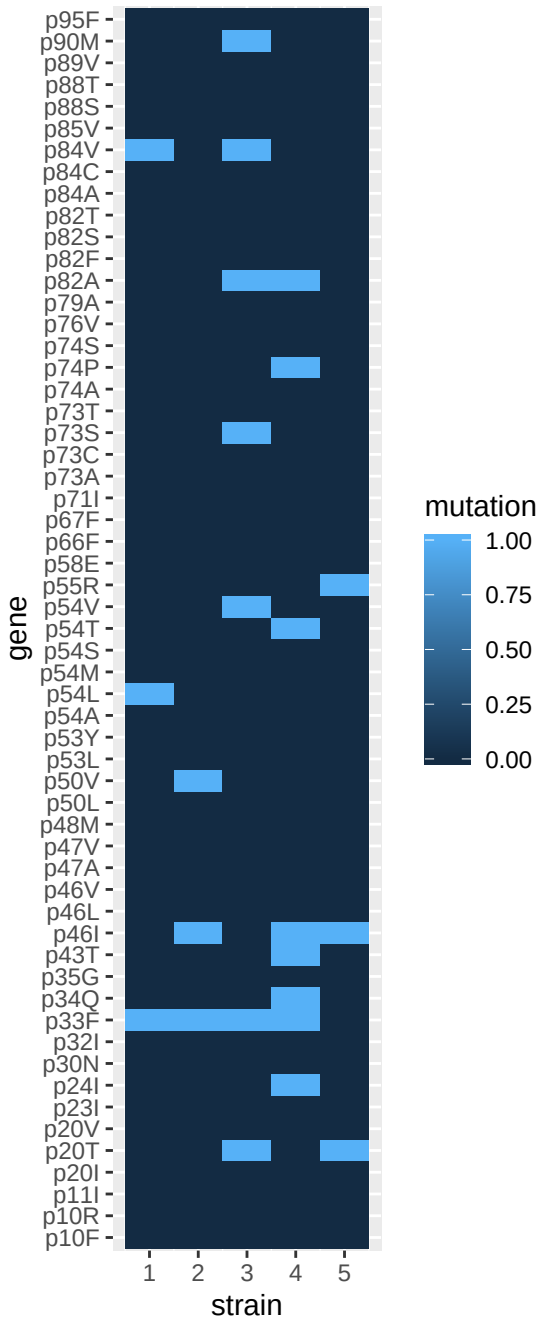
```
mut %>%
  mutate(index = 1:n()) %>%
  pivot_longer(cols = p10F:p95F, names_to = "gene", values_to = "mutation")
```

```
## # A tibble: 285 x 3
##    index gene  mutation
##    <int> <chr>    <dbl>
```

```
##  1     1 p10F         0
##  2     1 p10R         0
##  3     1 p11I         0
##  4     1 p20I         0
##  5     1 p20T         0
##  6     1 p20V         0
##  7     1 p23I         0
##  8     1 p24I         0
##  9     1 p30N         0
## 10     1 p32I         0
## # ... with 275 more rows
```

In this format, it's easy to make a tile plot (it's easier to see if the strains go along the x-axis and the genes along the y-axis):

```
mut %>%
  mutate(strain = 1:n()) %>%
  pivot_longer(cols = p10F:p95F, names_to = "gene", values_to = "mutation") %>%
  ggplot(aes(y = gene, x = strain, fill = mutation)) +
  geom_tile()
```



Since the data for each strain is a string of "0"s and "1"s, the binary distance metric might make sense here.

```
mut %>%
  dist(method = "binary")
```

```
##           1         2         3         4
## 2 0.8000000
## 3 0.7500000 0.8888889
## 4 0.9000000 0.7777778 0.8461538
## 5 1.0000000 0.8000000 0.8888889 0.9000000
```

It looks like strains 1 and 3 are closest. You can also try the Jaccard index:

```
library(vegan) # This package has a function for calculating the Jaccard index
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-6
```

```
mut %>%
  vegdist(method = "jaccard")
```

```
##           1         2         3         4
## 2 0.8000000
## 3 0.7500000 0.8888889
## 4 0.9000000 0.7777778 0.8461538
## 5 1.0000000 0.8000000 0.8888889 0.9000000
```

In this example, the Jaccard dissimilarity values are exactly the same as the binary distances.

You can also try the correlation-based distance:

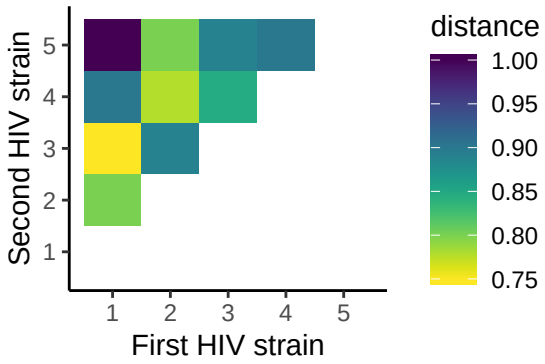
```
corr_mat_mut <- mut %>%
  t() %>%
  cor() # Calculates the correlation matrix among values in a matrix
sqrt(2 * (1 - corr_mat_mut)) %>% # Go from correlation matrix to distance matrix for
  as.dist()                      # correlation-based distance
```

```
##          1        2        3        4
## 2 1.186342
## 3 1.104026 1.302931
## 4 1.318368 1.133893 1.298780
## 5 1.452966 1.186342 1.302931 1.318368
```

Absolute values are different, but again it looks like strains 1 and 3 are closest.

For any of these, you can plot with a tile plot again:

```
mut %>%
  vegdist(method = "jaccard") %>%
  as.matrix() %>% # Can't go straight from a distance class to a tibble, so go through matrix
  as_tibble() %>%
  mutate(strain_1 = 1:n()) %>% # Add a column with strain 1 for each comparison
  pivot_longer(cols = 1:5, names_to = "strain_2", values_to = "distance") %>%
  mutate_at(vars(strain_1:strain_2), as.numeric) %>%
  mutate(distance = ifelse(strain_2 <= strain_1, NA, distance)) %>%
  ggplot(aes(x = strain_1, y = strain_2, fill = distance)) +
  geom_tile() +
  scale_fill_viridis(na.value = "white", direction = -1) +
  theme_classic() +
  labs(x = "First HIV strain",
       y = "Second HIV strain")
```



Clustering

This example used the `clusterExperiment` package. It's got a really nice vignette at https://bioconductor.org/packages/release/bioc/vignettes/clusterExperiment/inst/doc/clusterExperimentTutorial.html.

The example data here is from a single cell RNA sequencing experiment on ...

```
# If you need to install the packages, uncomment the following lines
# BiocManager::install("clusterExperiment")
# BiocManager::install("scRNAseq")

library(clusterExperiment)
library(scRNAseq)
```

The data used is the "fluidigm" dataset. It has a helpful with the citation to the original data (try `?fluidigm` to access that helpfile).

The data's in a "Summarized Experiment" class. You can access the data in that class using the `assay` method and the meta-data with `colData`.

```
data("fluidigm")
class(fluidigm)
```

```
## [1] "SummarizedExperiment"
## attr(,"package")
## [1] "SummarizedExperiment"
```

```
# Check out top left square of data
# I think columns are samples and rows are genes
fluidigm %>%
  assay() %>%
  `[`(1:10, 1:7) # sneaky trick to index out rows 1 to 10 and columns 1 to 7
```

```
##          SRR1275356 SRR1274090 SRR1275251 SRR1275287 SRR1275364 SRR1275269
## A1BG              0          0          0          0          0          0
## A1BG-AS1          0          0          0          0          0          0
## A1CF              0          0          0          0          0          0
## A2M               0          0          0         31          0         46
## A2M-AS1           0          0          0          0          0          0
## A2ML1             0          0          0          0          0          0
## A2MP1             0          0          8          0          0          0
## A3GALT2           0          0          0          0          0          0
## A4GALT            0          0          0          0          0          0
## A4GNT             0          0          0          0          0          0
##          SRR1275263
```

```
## A1BG              0
## A1BG-AS1          0
## A1CF              0
## A2M               0
## A2M-AS1           0
## A2ML1             0
## A2MP1             1
## A3GALT2           0
## A4GALT            0
## A4GNT             0
```

```
# Get the start of the metadata
fluidigm %>%
  colData() %>%
  head()
```

```
## DataFrame with 6 rows and 28 columns
##               NREADS  NALIGNED    RALIGN TOTAL_DUP    PRIMER INSERT_SZ
##             <numeric> <numeric> <numeric> <numeric> <numeric> <numeric>
## SRR1275356  10554900   7555880   71.5862   58.4931 0.0217638       208
## SRR1274090    196162    182494   93.0323   14.5122 0.0366826       247
## SRR1275251   8524470   5858130   68.7213   65.0428 0.0351827       230
## SRR1275287   7229920   5891540   81.4884   49.7609 0.0208685       222
## SRR1275364   5403640   4482910   82.9609   66.5788 0.0298284       228
## SRR1275269  10729700   7806230   72.7536   50.4285 0.0204349       245
##             INSERT_SZ_STD COMPLEXITY     NDUPR PCT_RIBOSOMAL_BASES
##                 <numeric>  <numeric> <numeric>           <numeric>
## SRR1275356            63   0.868928  0.343113              2e-06
## SRR1274090           133   0.997655   0.93573                  0
## SRR1275251            89   0.789252  0.201082                  0
## SRR1275287            78     0.8981  0.538191                  0
## SRR1275364            76   0.890693   0.39166                  0
## SRR1275269            99   0.879414  0.431169                  0
##             PCT_CODING_BASES PCT_UTR_BASES PCT_INTRONIC_BASES
##                    <numeric>     <numeric>          <numeric>
## SRR1275356         0.125806      0.180954           0.613229
## SRR1274090         0.309822      0.412917           0.205185
## SRR1275251         0.398461      0.473884           0.039886
## SRR1275287          0.19642      0.227592           0.498944
## SRR1275364         0.138617      0.210406           0.543941
## SRR1275269         0.333077      0.354635           0.248331
##             PCT_INTERGENIC_BASES PCT_MRNA_BASES MEDIAN_CV_COVERAGE
##                        <numeric>      <numeric>          <numeric>
## SRR1275356             0.080008        0.30676            1.49577
## SRR1274090             0.072076       0.722739            1.00758
## SRR1275251              0.08777       0.872345            1.24299
## SRR1275287             0.077044       0.424013           0.775981
## SRR1275364             0.107035       0.349024            1.44137
## SRR1275269             0.063957       0.687712             0.6171
##             MEDIAN_5PRIME_BIAS MEDIAN_3PRIME_BIAS MEDIAN_5PRIME_TO_3PRIME_BIAS
##                      <numeric>          <numeric>                    <numeric>
## SRR1275356                 0           0.166122                      1.03625
## SRR1274090          0.181742           0.698991                      0.29351
## SRR1275251                 0           0.340046                     0.201518
## SRR1275287          0.010251           0.350915                     0.292838
```

```
## SRR1275364                  0           0.204074                     0.619863
## SRR1275269            0.05796           0.345502                      0.28448
##            sample_id.x           Lane_ID LibraryName avgLength     spots
##            <character>       <character> <character> <integer> <integer>
## SRR1275356   SRX534610 D24VYACXX130502:4      GW16_2       202   9818076
## SRR1274090   SRX534823                 1       NPC_9        60     95454
## SRR1275251   SRX534623 D24VYACXX130502:4      GW16_8       202   7935952
## SRR1275287   SRX534641 D24VYACXX130502:1    GW21+3_2       202   6531944
## SRR1275364   SRX534614 D24VYACXX130502:7     GW16_23       202   4919561
## SRR1275269   SRX534632 D24VYACXX130502:4      GW21_8       202   9969377
##            Biological_Condition Coverage_Type Cluster1 Cluster2
##                     <character>   <character> <factor> <factor>
## SRR1275356                GW16          High     IIIb      III
## SRR1274090                 NPC           Low       1a        I
## SRR1275251                GW16          High       NA      III
## SRR1275287              GW21+3          High       1c        I
## SRR1275364                GW16          High     IIIb      III
## SRR1275269                GW21          High       NA        I
```

This data needs some pre-processing. In the book example, the steps they take are:

1. Limit to only samples where sequencing depth is "high"
2. Limit to only genes with at least 10 reads in at least 10 cells
3. Normalize the data. They use quantile normalization with the `normalizeQuantiles` package from the `limma` package.

Step 1:

```
fluidigm_high <- fluidigm[ , fluidigm$Coverage_Type == "High"]
fluidigm       # This reduces the samples in the data from 130...
```

```
## class: SummarizedExperiment
## dim: 26255 130
## metadata(3): sample_info clusters which_qc
## assays(4): tophat_counts cufflinks_fpkm rsem_counts rsem_tpm
## rownames(26255): A1BG A1BG-AS1 ... ZZEF1 ZZZ3
## rowData names(0):
## colnames(130): SRR1275356 SRR1274090 ... SRR1275366 SRR1275261
## colData names(28): NREADS NALIGNED ... Cluster1 Cluster2
```

```
fluidigm_high  # ... to 65
```

```
## class: SummarizedExperiment
## dim: 26255 65
## metadata(3): sample_info clusters which_qc
## assays(4): tophat_counts cufflinks_fpkm rsem_counts rsem_tpm
## rownames(26255): A1BG A1BG-AS1 ... ZZEF1 ZZZ3
## rowData names(0):
## colnames(65): SRR1275356 SRR1275251 ... SRR1275366 SRR1275261
## colData names(28): NREADS NALIGNED ... Cluster1 Cluster2
```

Step 2:

```
high_read_genes <- fluidigm_high %>%
  assay() %>%
  as_tibble(rownames = NA) %>%
  rownames_to_column(var = "gene") %>%
  mutate(index = 1:n()) %>%
```

```
  pivot_longer(-gene) %>%
  mutate(over_10 = value > 10) %>%
  group_by(gene) %>%
  summarize(n_over_10 = sum(over_10)) %>%
  dplyr::filter(n_over_10 >= 10) %>%
  pull(gene)
head(high_read_genes)
```

```
## [1] "A2M"   "AAAS"  "AACS"  "AADAT" "AAGAB" "AAK1"
```

```
fluidigm_high <- fluidigm_high[rownames(fluidigm_high) %in% high_read_genes, ]

fluidigm_high
```

```
## class: SummarizedExperiment
## dim: 7426 65
## metadata(3): sample_info clusters which_qc
## assays(4): tophat_counts cufflinks_fpkm rsem_counts rsem_tpm
## rownames(7426): A2M AAAS ... ZZEF1 ZZZ3
## rowData names(0):
## colnames(65): SRR1275356 SRR1275251 ... SRR1275366 SRR1275261
## colData names(28): NREADS NALIGNED ... Cluster1 Cluster2
```

```
library(limma)
```

```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```
norm_counts <- fluidigm_high %>%
  assay() %>%
  normalizeQuantiles() %>%
  round()
norm_counts %>%
  `[`(1:4, 1:4)
```

```
##       SRR1275356 SRR1275251 SRR1275287 SRR1275364
## A2M            0          1         40          1
## AAAS           0        472          0          1
## AACS          62        100        275          1
## AADAT          0          1          0          1
```

```
# Replace this back into the SummarizeExperiment class object
assays(fluidigm_high) <- list(normalized_counts = norm_counts)
```

Now you can do the clustering:

```
clus_results <- clusterMany(fluidigm_high, clusterFunction = "pam",
                            ks = c(5, 7, 9), isCount = TRUE,
                            dimReduce = "var",
                            nVarDims = c(60, 100, 150))
clus_results
```

```
## class: ClusterExperiment
## dim: 7426 65
```

```
## reducedDimNames: no reduced dims stored
## filterStats: no valid filtering stats stored
## -----------
## Primary cluster type: clusterMany
## Primary cluster label: k=5
## Table of clusters (of primary clustering):
##  1  2  3  4  5
## 21 19 12  9  4
## Total number of clusterings: 3
## No dendrogram present
## -----------
## Workflow progress:
## clusterMany run? Yes
## makeConsensus run? No
## makeDendrogram run? No
## mergeClusters run? No
```

Flow cytometry example

```
library(flowCore)
library(flowViz)
fcs_b <- read.FCS("data/Bendall_2011.fcs")
slotNames(fcs_b)
```

```
## [1] "exprs"       "parameters"  "description"
```

```
fcs_b %>%
  exprs() %>%
  `[`(1:2, )
```

```
##      Time Cell_length Ir(190.960)-Dual Ir(192.962)-Dual Rh(102.905)-Dual
## [1,]   15    31.74782        1189.3621              784         84.92314
## [2,]   29    32.73994         725.4294              784         12.57166
##      In(114.903)-Dual Cd(109.903)-Dual Cd(110.904)-Dual Cd(111.902)-Dual
## [1,]          5.67213       15.7740660        -1.376150         6.436958
## [2,]         12.75089       -0.1887358        -2.648802        -1.471898
##      Cd(113.903)-Dual La(138.906)-Dual Pr(140.907)-Dual Nd(141.907)-Dual
## [1,]        6.2128391         14.68693        1970.0000         2.209156
## [2,]       -0.7862989         89.21723         558.0174         3.443998
##      Nd(143.910)-Dual Nd(144.912)-Dual Nd(145.913)-Dual Nd(147.916)-Dual
## [1,]         2.364415       -1.2065018       -0.3819572       -0.3784404
## [2,]        22.432457       -0.6339307        1.2526673       -1.7176632
##      Nd(149.920)-Dual Sm(146.914)-Dual Sm(151.919)-Dual Sm(153.922)-Dual
## [1,]       -3.2443109       -1.7556700        -3.299473         1.632192
## [2,]       -0.8809353       -0.2761891        -2.177210        -2.538092
##      Eu(150.919)-Dual Eu(152.921)-Dual Gd(155.922)-Dual Gd(157.924)-Dual
## [1,]       -0.5645261       -0.2469634       -1.8393580         4.089273
## [2,]       -0.9125634        7.3124208       -0.5999154        20.541170
##      Gd(159.927)-Dual Tb(158.925)-Dual Dy(163.929)-Dual Ho(164.930)-Dual
## [1,]       -0.7250714         8.230175        -85.54974        -2.223529
## [2,]       -2.9493327        -6.057666         27.69015        -6.205233
##      Er(165.930)-Dual Er(166.932)-Dual Er(167.932)-Dual Er(169.935)-Dual
## [1,]        1049.7231         21.69583        -4.815055         4.110293
## [2,]         291.7576         29.38453        -2.034343         1.544883
##      Tm(168.934)-Dual Yb(170.936)-Dual Yb(171.936)-Dual Yb(173.938)-Dual
```

```
## [1,]       -0.1112851        -1.005018         84.42017         3.545024
## [2,]       -1.6356324        -1.415160         27.69215        -3.500782
##      Yb(175.942)-Dual Lu(174.940)-Dual Cd(110,111,112,114) absoluteEventNumber
## [1,]      -1.36180770         9.506894           27.267317                   5
## [2,]      -0.06603064         7.971855           -5.128984                   9
```

```
# Read in a small dataframe that matches the original column
# names to the CD marker names
markers_b <- read_csv("data/Bendall_2011_markers.csv")
```

```
## Parsed with column specification:
## cols(
##   isotope = col_character(),
##   marker = col_character()
## )
```

```
markers_b %>%
  head()
```

```
## # A tibble: 6 x 2
##   isotope          marker
##   <chr>            <chr>
## 1 Nd(144.912)-Dual CD4
## 2 Nd(145.913)-Dual CD8
## 3 Sm(146.914)-Dual CD20
## 4 Gd(157.924)-Dual CD33
## 5 Er(169.935)-Dual CD56
## 6 Ir(190.960)-Dual DNA191
```

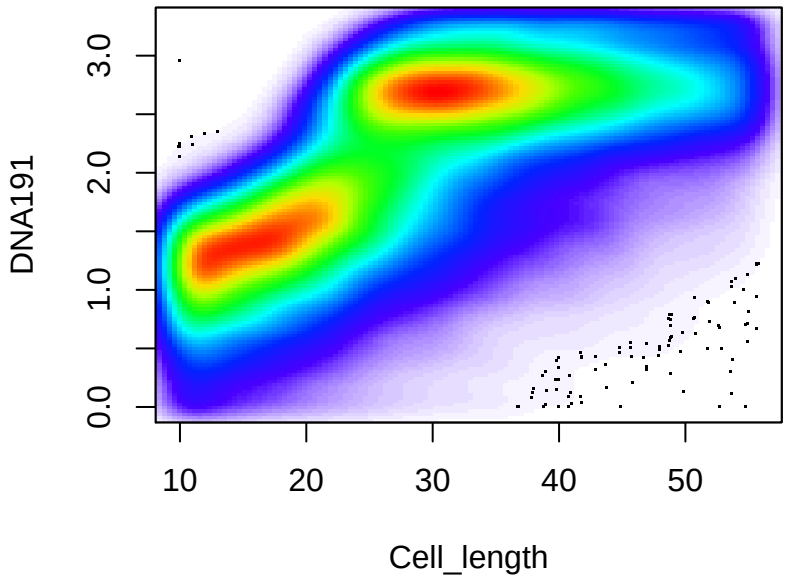
```
# Match these up to the column names and then replace the
# original column names with these
mt <- match(markers_b$isotope, colnames(fcs_b))
colnames(fcs_b)[mt] <- markers_b$marker

# Now you'll see the column names have changed
fcs_b %>%
  exprs() %>%
  `[`(1:2, )
```

```
##      Time Cell_length    DNA191 DNA192 Rh(102.905)-Dual In(114.903)-Dual
## [1,]   15    31.74782 1189.3621    784         84.92314          5.67213
## [2,]   29    32.73994  725.4294    784         12.57166         12.75089
##             CD3 Cd(110.904)-Dual Cd(111.902)-Dual Cd(113.903)-Dual   CD45RA
## [1,] 15.7740660        -1.376150         6.436958        6.2128391 14.68693
## [2,] -0.1887358        -2.648802        -1.471898       -0.7862989 89.21723
##      Pr(140.907)-Dual Nd(141.907)-Dual Nd(143.910)-Dual        CD4        CD8
## [1,]        1970.0000         2.209156         2.364415 -1.2065018 -0.3819572
## [2,]         558.0174         3.443998        22.432457 -0.6339307  1.2526673
##      Nd(147.916)-Dual Nd(149.920)-Dual       CD20 Sm(151.919)-Dual
## [1,]       -0.3784404       -3.2443109 -1.7556700        -3.299473
## [2,]       -1.7176632       -0.8809353 -0.2761891        -2.177210
##      Sm(153.922)-Dual Eu(150.919)-Dual Eu(152.921)-Dual Gd(155.922)-Dual
## [1,]         1.632192       -0.5645261       -0.2469634       -1.8393580
## [2,]        -2.538092       -0.9125634        7.3124208       -0.5999154
##           CD33 Gd(159.927)-Dual Tb(158.925)-Dual Dy(163.929)-Dual
## [1,]  4.089273       -0.7250714         8.230175        -85.54974
```

```
## [2,] 20.541170       -2.9493327        -6.057666         27.69015
##      Ho(164.930)-Dual Er(165.930)-Dual Er(166.932)-Dual Er(167.932)-Dual
## [1,]        -2.223529        1049.7231         21.69583        -4.815055
## [2,]        -6.205233         291.7576         29.38453        -2.034343
##          CD56 Tm(168.934)-Dual Yb(170.936)-Dual Yb(171.936)-Dual
## [1,] 4.110293       -0.1112851        -1.005018         84.42017
## [2,] 1.544883       -1.6356324        -1.415160         27.69215
##      Yb(173.938)-Dual Yb(175.942)-Dual Lu(174.940)-Dual    CD3all
## [1,]         3.545024      -1.36180770         9.506894 27.267317
## [2,]        -3.500782      -0.06603064         7.971855 -5.128984
##      absoluteEventNumber
## [1,]                   5
## [2,]                   9
```

```
flowPlot(fcs_b, plotParameters = c("Cell_length", "DNA191"),
         logy = TRUE)
```



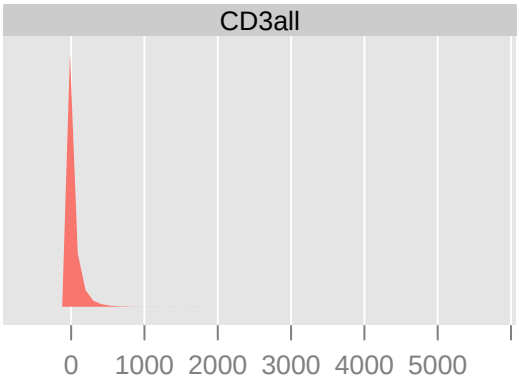

```
asinhtrsf <- arcsinhTransform(a = 0.1, b = 1) # It looks like this function is making a new function!
asinhtrsf %>%
  class()
```

```
## [1] "transform"
## attr(,"package")
## [1] "flowCore"
```

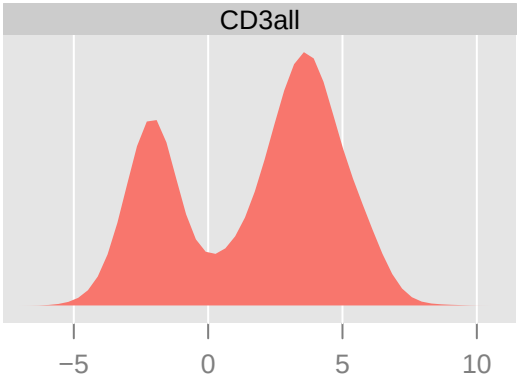
```
is.function(asinhtrsf)
```

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

```
# Transform the data for all the columns *except* 1, 2, and 41
fcs_bt <- transform(fcs_b, transformList(colnames(fcs_b)[-c(1, 2, 41)],
                                         asinhtrsf))
# Plot without the transformation
densityplot(~ `CD3all`, fcs_b)
```



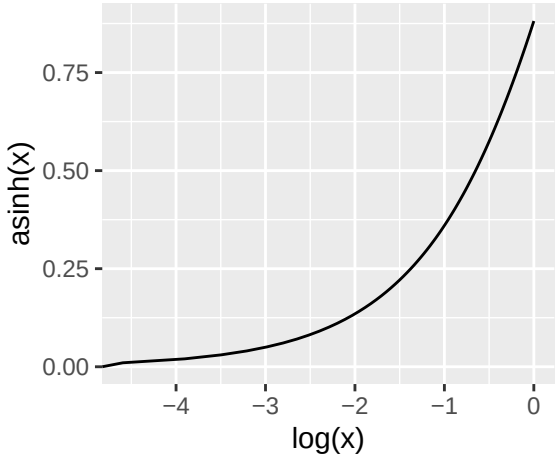

```
# Plot without the transformation
densityplot(~ `CD3all`, fcs_bt)
```



Look a little more at how this transform works:

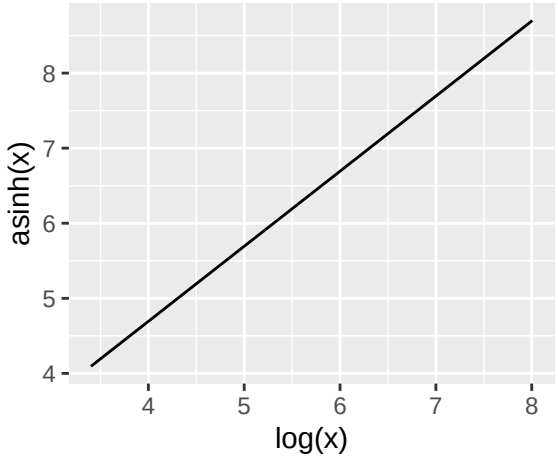
More curved at lower values of x:

```
tibble(x = seq(0, 1, length.out = 100)) %>%
  ggplot(aes(log(x), asinh(x))) +
  geom_line()
```



More like a line at higher values of x:

```
tibble(x = seq(30, 3000, length = 100)) %>%
  ggplot(aes(log(x), asinh(x))) +
  geom_line()
```



Use k-means for clustering:

```
# First make the filter. These are "defined by a single flow parameter"
kf <- kmeansFilter("CD3all" = c("Pop1", "Pop2"),
                   filterId = "myKmFilter")
class(kf)
```

```
## [1] "kmeansFilter"
## attr(,"package")
## [1] "flowCore"
```

```
# Apply the filter to split the data into two populations
fres <- flowCore::filter(fcs_bt, kf) # Need the :: syntax (`filter`)
fres
```

```
## A filterResult produced by the filter named 'myKmFilter'
##  resulting in multiple populations:
##   Pop1
##   Pop2
```

```
class(fres)
```

```
## [1] "multipleFilterResult"
## attr(,"package")
## [1] "flowCore"
```

```
summary(fres)
```

```
## Pop1: 33429 of 91392 events (36.58%)
## Pop2: 57963 of 91392 events (63.42%)
```

```
# create objects with each of these two populations
fcs_bt1 <- flowCore::split(fcs_bt, fres, population = "Pop1")
fcs_bt2 <- flowCore::split(fcs_bt, fres, population = "Pop2")
```

Cluster the cells based on their values for two CD (CD3 and CD56):

```
# Uncomment and run the next line if you need the library
# BiocManager::install("flowPeaks")
library("flowPeaks")

fp <- fcs_bt %>%
  exprs() %>%
  `[`(, c("CD3all", "CD56")) %>% # Pull out two flow parameters
```

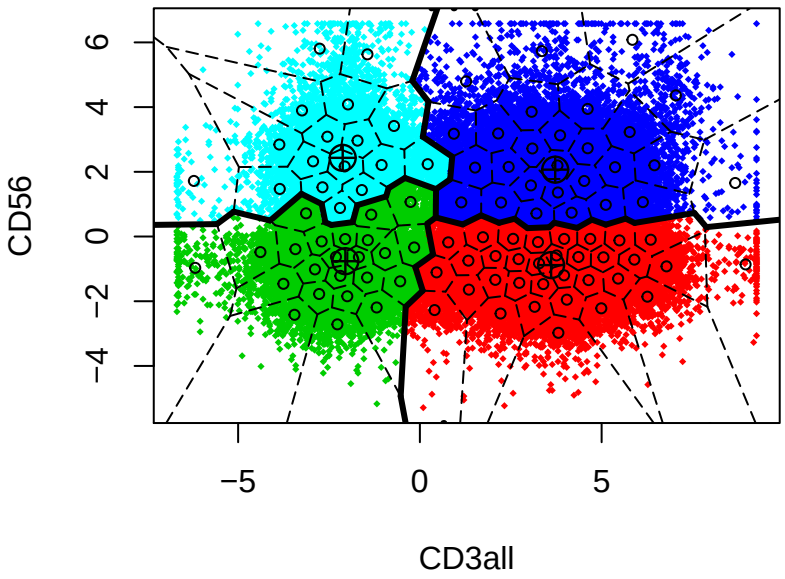
```
  flowPeaks() # Cluser all the cells based on the two parameters?
```

```
##
## Starting the flow Peaks analysis...
##
##     Task A: compute kmeans...
##         step 0, set the intial seeds, tot.wss=17597.8
##         step 1, do the rough EM, tot.wss=11900.7 at 0.217338 sec
##         step 2, do the fine transfer of Hartigan-Wong Algorithm
##                  tot.wss=11846.3 at 0.424264 sec
##         ...finished summarization at 0.428 sec
##
##     Task B: find peaks...
## finished at 0.473 sec
```

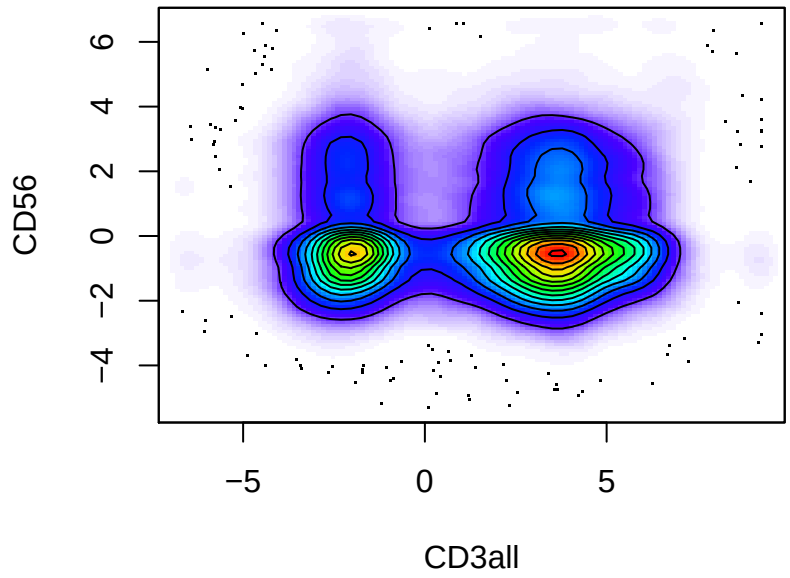
```
summary(fp)
```

```
##      cluster.id     weight CD3all.center CD56.center
## [1,]          1 0.46896884      3.610331  -0.8834935
## [2,]          2 0.25054709     -2.049654  -0.7625281
## [3,]          3 0.19013699      3.723269   2.0670718
## [4,]          4 0.09034708     -2.123329   2.4317792
```

```
# I don't think this plot is giving any of the clustering results...
plot(fp)
```




```
flowPlot(fcs_bt, plotParameters = c("CD3all", "CD56"), logy = FALSE)
fcs_bt %>%
  `[`(, c("CD3all", "CD56")) %>%
  contour(add = TRUE) # Add contour lines to the plot
```



Density-based clustering

```
# Uncomment and run the next line if you need the library
# BiocManager::install("dbscan")
library("dbscan")
mc5 <- fcs_bt %>%
  exprs() %>%
  `[`(,c("CD4", "CD8", "CD20", "CD3all"))
mc5 %>%
  head()
```

```
##             CD4        CD8       CD20    CD3all
## [1,] -0.9547135 -0.2783490 -1.2781241  4.002830
## [2,] -0.5113526  1.1101591 -0.1752901 -2.318107
## [3,] -0.5421367  2.3434486 -0.7316469  3.688635
## [4,]  1.4386085 -1.5734058 -1.4927354  1.440554
## [5,] -1.0765024  3.7359109  5.4406841  3.266283
## [6,]  1.2213971 -0.1953204  1.3793038  3.023840
```

In the `dbscan` call, `eps` is the "size of the epsilon neighborhood":

```
res5 <- dbscan(mc5, eps = 0.65, minPts = 30)
res5
```

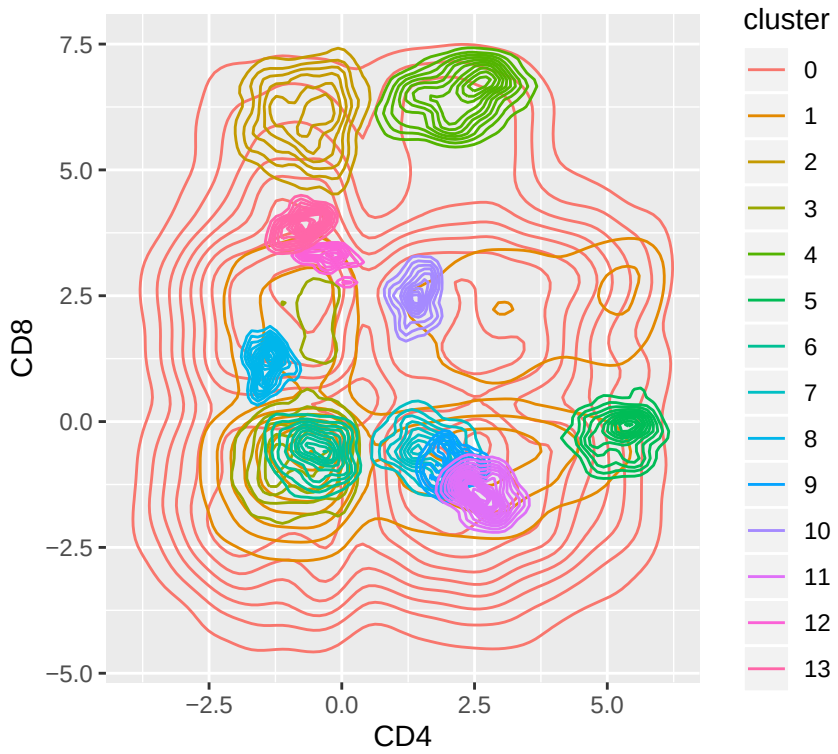
```
## DBSCAN clustering for 91392 objects.
## Parameters: eps = 0.65, minPts = 30
## The clustering contains 13 cluster(s) and 29292 noise points.
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
## 29292 58996  1114  1203   294   108   120   108    24    30    40    33    12
##    13
##    18
##
## Available fields: cluster, eps, minPts
```

```
mc5df <- mc5 %>%
  as_tibble() %>%
```

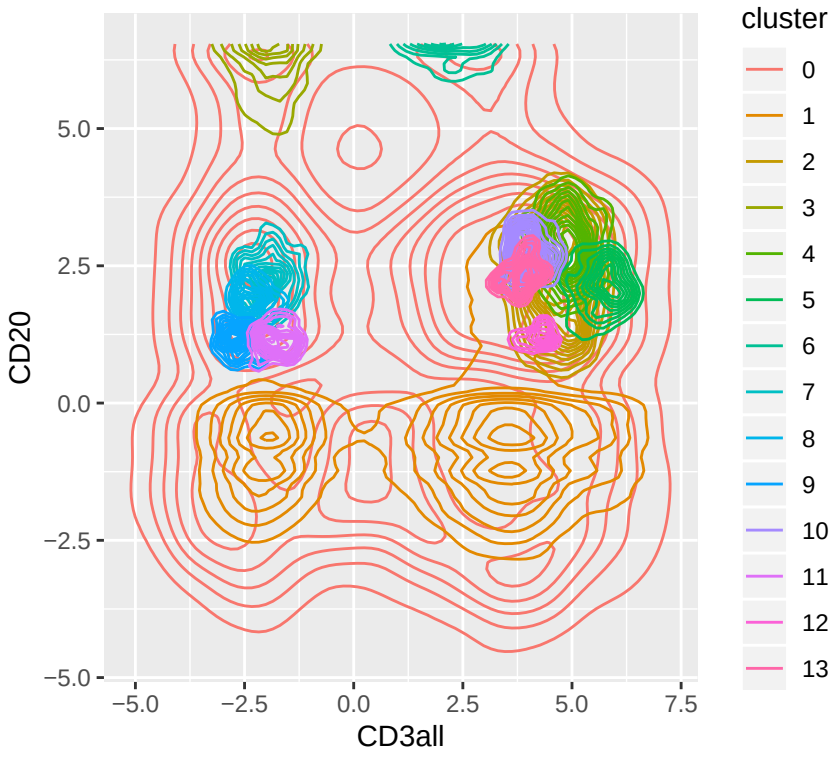
```
  mutate(cluster = res5$cluster) %>%
  mutate(cluster = forcats::as_factor(cluster))
head(mc5df)
```

```
## # A tibble: 6 x 5
##       CD4    CD8   CD20 CD3all cluster
##     <dbl>  <dbl>  <dbl>  <dbl> <fct>
## 1 -0.955 -0.278 -1.28    4.00 1
## 2 -0.511  1.11  -0.175  -2.32 1
## 3 -0.542  2.34  -0.732   3.69 1
## 4  1.44  -1.57  -1.49    1.44 1
## 5 -1.08   3.74   5.44    3.27 0
## 6  1.22  -0.195  1.38    3.02 1
```

```
ggplot(mc5df, aes(x = CD4, y = CD8, color = cluster)) +
  geom_density2d()
```




```
ggplot(mc5df, aes(x = CD3all, y = CD20, color = cluster)) +
  geom_density2d()
```


cluster
0
1
2
3
4
5
6
7
8
9
10
11
12
13
CD20
5.0
2.5
0.0
−2.5
−5.0
−5.0
−2.5
0.0
2.5
5.0
7.5
CD3all