TNBC_spe

2022-11-09

Libaries

```
library(readr)
library(dplyr)
library(tidyverse)
library(SpatialExperiment)
```

Reading the CSV output

```
TNBC <- readr::read_csv("/Users/henzhwang/Desktop/TNBC_training/MIBI-TNBC_scdata_counts_mm_matlab_revis
```

```
## Rows: 179194 Columns: 64
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): SITE_02, RECURRENCE_LABEL, mm
## dbl (61): sample_id, patient_id, AGE_AT_DX, STAGE, LATERAL, GRADE, Survival_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Diagnostic of the TNBC dataset

```
# dimension of the TNBC dataset
dim(TNBC)
```

```
## [1] 179194     64
```

```
# number of sample_id and patient_id in the dataset
num_sample_id <- length(unique(TNBC$sample_id))
num_patient_id <- length(unique(TNBC$patient_id))
all.equal(num_sample_id, num_patient_id)
```

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

```
print(paste("There are", num_sample_id, "sample_id and patient_id in the datset."))
```

```
## [1] "There are 39 sample_id and patient_id in the datset."
```

```r
# number of cluster_id in the dataset
print(paste("There are", length(unique(TNBC$cluster_id)), "cluster_id in the datset."))
```

```
## [1] "There are 113 cluster_id in the datset."
```

```r
list(sort(unique(TNBC$cluster_id)))
```

```
## [[1]]
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 113 114 128 129 143 157 158
## [109] 159 172 175 187 211
```

```r
# check to see how many unique cell_type and mm in the datset
print(paste("There are", length(unique(TNBC$cell_type)),
            "cell_type in the dataset,", "\n",
            "they are", list(unique(TNBC$cell_type))))
```

```
## [1] "There are 16 cell_type in the dataset, \n they are c(5, 4, 2, 3, 10, 11, 13, 14, 7, 15, 9, 8, 6
```

```r
print(paste("There are", length(unique(TNBC$mm)),
            "cell_type in the dataset,", "\n",
            "they are", list(unique(TNBC$mm))))
```

```
## [1] "There are 16 cell_type in the dataset, \n they are c(\"B\", \"CD3 T\", \"CD4 T\", \"CD8 T\", \"
```

```r
# check to see if patient_id is unique to sample_id
TNBC_grouped_sampleId <- TNBC %>% group_by(sample_id)

TNBC_grouped_sampleId %>%
  summarise(n = n(),
            num_unique_patient = length(unique(patient_id)), patient_id = unique(patient_id),
            num_unique_image = length(unique(ImageNb)), ImageNb = unique(ImageNb)) #Image 22 and 38 are
```

```
## # A tibble: 39 x 6
##    sample_id     n num_unique_patient patient_id num_unique_image ImageNb
##        <dbl> <int>              <int>      <dbl>            <int>   <dbl>
##  1         1  5199                  1      30824                1       1
##  2         2  3033                  1      30805                1       2
##  3         3  5671                  1      30812                1       3
##  4         4  5381                  1      30838                1       4
##  5         5  4252                  1      30865                1       5
##  6         6  4894                  1      30847                1       6
##  7         7  3308                  1      30854                1       7
##  8         8  3786                  1      30846                1       8
##  9         9  5105                  1      30783                1       9
## 10        10  4066                  1      30781                1      10
## # ... with 29 more rows
```

```
print(paste("Each patient_id is unqiue to one sample_id.")) # patien_id and sample_id are redundant
```

```
## [1] "Each patient_id is unqiue to one sample_id."
```

```
# check to see if cell_type and mm are unique to each other
TNBC %>%
  group_by(mm) %>%
  summarise(n = n(),
            num_unique_cellType = length(unique(cell_type)), cell_type = unique(cell_type))
```

```
## # A tibble: 16 x 4
##    mm               n num_unique_cellType cell_type
##    <chr>        <int>               <int>     <dbl>
##  1 B            17084                   1         5
##  2 CD3 T         1135                   1         4
##  3 CD4 T         9918                   1         2
##  4 CD8 T        13376                   1         3
##  5 DC            2381                   1        10
##  6 DC/Mono       1280                   1        11
##  7 Endothelial    279                   1        13
##  8 Epithelial   31871                   1        14
##  9 Mac           5552                   1         7
## 10 Mesenchymal  27698                   1        15
## 11 Mono/Neu       835                   1         9
## 12 Neu           1365                   1         8
## 13 NK             285                   1         6
## 14 Other        56603                   1        16
## 15 Other immune  8669                   1        12
## 16 T reg          863                   1         1
```

```
print(paste("Each cell_type is unqiue to one mm type."))
```

```
## [1] "Each cell_type is unqiue to one mm type."
```

```
# Check number of clusters_id from FlowSOM for each sample
TNBC_grouped_sampleId %>%
  summarise(n = n(),
            num_unique_cluster = length(unique(cluster_id)))
```

```
## # A tibble: 39 x 3
##    sample_id     n num_unique_cluster
##        <dbl> <int>              <int>
##  1         1  5199                 83
##  2         2  3033                 77
##  3         3  5671                111
##  4         4  5381                109
##  5         5  4252                107
##  6         6  4894                101
##  7         7  3308                 86
##  8         8  3786                 87
##  9         9  5105                109
## 10        10  4066                106
## # ... with 29 more rows
```

```
# check to see how many cell_type in each cluster_id
TNBC %>%
  group_by(cluster_id) %>%
  summarise(n = n(),
            num_unique_cellType = length(unique(mm)), # mm and cell_type are the same
            cell_type_1 = as.character(as.list(unique(mm))[1]),
            cell_type_2 = as.character(as.list(unique(mm))[2]))
```

```
## # A tibble: 113 x 5
##    cluster_id     n num_unique_cellType cell_type_1 cell_type_2
##         <dbl> <int>               <int> <chr>       <chr>
##  1          1  1034                   2 B           Epithelial
##  2          2  1450                   2 B           Epithelial
##  3          3  1991                   2 B           Other
##  4          4  1403                   2 B           Other
##  5          5  2334                   2 DC/Mono     Other
##  6          6  3148                   2 B           Other
##  7          7  3057                   2 CD4 T       Other
##  8          8   906                   2 CD3 T       Other
##  9          9  1973                   2 CD4 T       Other
## 10         10   783                   2 CD4 T       Mesenchymal
## # ... with 103 more rows
```

Construct assay for TNBC spatial experiment object

Test case with sample_id 1 and sample_id 2

The goal is create a count matrix with row names are `sample_id` and column names are `cluster_id` where the input values are the count of cells for cluster_id in one sample.

First we want to take sample 1 and try to contruct the count matrix.

```
# filter out sample 1 from the TNBC dataset
TNBC_sample1 <- TNBC %>% filter(sample_id == 1)
head(TNBC_sample1)
```

```
## # A tibble: 6 x 64
##   sample_id patien~1 AGE_A~2 STAGE SITE_02 LATERAL GRADE RECUR~3 Survi~4 clust~5
##       <dbl>    <dbl>   <dbl> <dbl> <chr>     <dbl> <dbl> <chr>     <dbl>   <dbl>
## 1         1    30824      77    33 C504          2     1 POSITI~    2612      34
## 2         1    30824      77    33 C504          2     1 POSITI~    2612      11
## 3         1    30824      77    33 C504          2     1 POSITI~    2612      31
## 4         1    30824      77    33 C504          2     1 POSITI~    2612      33
## 5         1    30824      77    33 C504          2     1 POSITI~    2612      31
## 6         1    30824      77    33 C504          2     1 POSITI~    2612      31
## # ... with 54 more variables: mm <chr>, cell_type <dbl>, ImageNb <dbl>,
## #   cellLabelInImage <dbl>, cellSize <dbl>, cellRadius <dbl>, centroidX <dbl>,
## #   centroidY <dbl>, majoraxis <dbl>, eccentricity <dbl>, Au <dbl>,
## #   Background <dbl>, betaCatenin <dbl>, Ca <dbl>, CD11b <dbl>, CD11c <dbl>,
## #   CD138 <dbl>, CD16 <dbl>, CD20 <dbl>, CD209 <dbl>, CD3 <dbl>, CD31 <dbl>,
## #   CD4 <dbl>, CD45 <dbl>, CD45RO <dbl>, CD56 <dbl>, CD63 <dbl>, CD68 <dbl>,
## #   CD8 <dbl>, dsDNA <dbl>, EGFR <dbl>, Fe <dbl>, FoxP3 <dbl>, ...
```

```
# # select the required columns
# TNBC_sample1 <- TNBC_sample1 %>%
#   select(c(sample_id, cluster_id, mm, cellLabelInImage, cellSize, cellRadius, centroidX, centroidY))
# head(TNBC_sample1)

# construct count matrix
count_sample1 <- TNBC_sample1 %>%
  group_by(cluster_id) %>%
  summarise(sample_1 = n()) %>%
  #t() %>%
  as.data.frame()
```

We then want to take sample 2 and construct the count matrix

```
# filter out sample 1 from the TNBC dataset
TNBC_sample2 <- TNBC %>% filter(sample_id == 2)
head(TNBC_sample2)
```

```
## # A tibble: 6 x 64
##   sample_id patien~1 AGE_A~2 STAGE SITE_02 LATERAL GRADE RECUR~3 Survi~4 clust~5
##       <dbl>    <dbl>   <dbl> <dbl> <chr>     <dbl> <dbl> <chr>     <dbl>   <dbl>
## 1         2    30805      67    32 C509          2     3 NEGATI~     745      42
## 2         2    30805      67    32 C509          2     3 NEGATI~     745      48
## 3         2    30805      67    32 C509          2     3 NEGATI~     745      20
## 4         2    30805      67    32 C509          2     3 NEGATI~     745      20
## 5         2    30805      67    32 C509          2     3 NEGATI~     745      20
## 6         2    30805      67    32 C509          2     3 NEGATI~     745      48
## # ... with 54 more variables: mm <chr>, cell_type <dbl>, ImageNb <dbl>,
## #   cellLabelInImage <dbl>, cellSize <dbl>, cellRadius <dbl>, centroidX <dbl>,
## #   centroidY <dbl>, majoraxis <dbl>, eccentricity <dbl>, Au <dbl>,
## #   Background <dbl>, betaCatenin <dbl>, Ca <dbl>, CD11b <dbl>, CD11c <dbl>,
## #   CD138 <dbl>, CD16 <dbl>, CD20 <dbl>, CD209 <dbl>, CD3 <dbl>, CD31 <dbl>,
## #   CD4 <dbl>, CD45 <dbl>, CD45RO <dbl>, CD56 <dbl>, CD63 <dbl>, CD68 <dbl>,
## #   CD8 <dbl>, dsDNA <dbl>, EGFR <dbl>, Fe <dbl>, FoxP3 <dbl>, ...
```

```
# construct count matrix
count_sample2 <- TNBC_sample2 %>%
  group_by(cluster_id) %>%
  summarise(sample_2 = n()) %>%
  as.data.frame()
```

We then want to join the two dataframe together,

```
# fully joining the count matrix of sample_1 and sample_2
## let cluster_id be the rownames and take transpose(cluster_id as column name)
example <- full_join(x = count_sample1, y = count_sample2, by = "cluster_id") %>%
  arrange(cluster_id) %>%
  tibble::column_to_rownames(var = "cluster_id") %>%
  t() %>% as.data.frame()

# setting NA value in the dataframe to 0
example[is.na(example)] <- 0
example
```

```
##          2  3 5 6  7  8  9 10 11 12 13 16 17 18 19 20  21 22 23 24 25 26 27 28
## sample_1 1  5 1 0  1 27 19 17 26 32  3  1  6  0 22 29 244 56 19  8  3 15  3  6
## sample_2 0 28 0 6 17  0 11  3  0  0  0  4  3 13 25 77   0  0  0  0  2 11  0  1
##          29 30  31 32 33 34 35 36 37 38 39 40 42 44 45 46 48 49 50 51 52 53 54
## sample_1  8 13 266 11 16 38 18 10 12  0 36 19 19  9 69  9  3  1  1  1  1  0 62
## sample_2 11 66   0  0  0  0  4  0 17  1 16 74  2  0 12  0  6 28 69  0  2  2  1
##          55 56 57 58 59 60 61 63 64  65  66 67 68 69 70 71 73  75 76 77 78 79
## sample_1 70 30 58  3  0  8  2  2  5   0   0 27 11  1  0  1  2   1  4 53 13  0
## sample_2 32 51 21  5  4 27  0  3  2 145 452 13  6  5 13  0  5 476  6 10 22 37
##          80 81 82  83 85  86 87 88 89 90 91 92  93  94  95 96  97 98 99 100 101
## sample_1  2  0  0 377  0 123 28  2  2  3  0  0   0   0   0 14 204 96 25   1   0
## sample_2  2  1  1   6 14  13 40 18  6  5  5  2 185 110 349 43   5 23 10   2 119
##          113 114 128 129 143 157 158 172 175 187 211
## sample_1 122  51 363 173 455 820 257 135   8 420  61
## sample_2   6   2  25  92  79   6   0   0   0   1  16
```

Application to create count matrix

Now we want to apply the above test case of sample_1 and sample_2 to the whole TNBC dataset. We will write a function `create_countMat()` which takes the dataset as an argument and return a count matrix of all sample_id for the input dataset.

The function performs its functionality by the following steps:

1. The function takes in the dataset and creates two lists consists all unique sample_id and cluster_id in the dataset and names as `sampleID_list` and `clusterId_list` correspondingly
2. It uses for loop of the length of the `sampleId_list`. For each `sample_id`, it groups the cell by `cluster_id` and counts the number of them
3. Next is to fully join all samples count matrix `count.matrix` into one data frame by `cluster_id` and sort them by `cluster_id` in ascending order
4. The function then compares if `cluster_id` in `count.matrix` has all the same elements of cluster number as in 'clusterId_list.
5. If step 4 is true, then `cluster_id` will become the row names of `count.matrix`. By taking transpose, the `count.matrix` will have `sample_id` as row names and `cluster_id` as column names. `NA` value will be replaced by 0. Else, the function stops and raises an error
6. The function returns the count matrix `count.matrix`

```r
# write a for loop to create the count matrix of cluster_id for each samples for all sample_id
create_countMat <- function(dataset) {

  # lists of unique sample_id and cluster_id
  sampleId_list <- sort(unique(dataset$sample_id))
  #sampleId_list <- 1:2
  clusterId_list <- sort(unique(dataset$cluster_id))

  # for loop to create count matrix for each sample_id
  for (k in sampleId_list) {
    sampleId <- paste0("sample_", k)

    sample_df <- dataset %>%
      filter(sample_id == k) %>%
      group_by(cluster_id) %>%
      summarise(!!(sampleId) := n())
```

```
      #summarise(!!as.character(k) := n())

    if (k == 1) {
      count.matrix <- sample_df
      # !! indicates evaluate; := indicates assign
      #count.matrix <- sample_df %>% dplyr::rename(!!paste0("sample_", k) := counts)
    } else {
      count.matrix <- dplyr::full_join(x = count.matrix,
                                       y = sample_df,
                                       by = "cluster_id") %>%
        arrange(cluster_id)
    }

  }
  #print(clusterId_list)
  #print(count.matrix$cluster_id)

  #print(isTRUE(all.equal(count.matrix$cluster_id, clusterId_list)))

  if (isTRUE(all.equal(count.matrix$cluster_id, clusterId_list))) {
    count.matrix <- count.matrix %>%
      tibble::column_to_rownames(var = "cluster_id") %>%
      t() %>% as.data.frame()

    # setting NA value in the count matrix to 0
    count.matrix[is.na(count.matrix)] <- 0
  } else {
    stop("Error in counting cluster_id")
  }

  return(count.matrix)
}
```

Test Case: We subset the `TNBC` dataset with `sample_id` less than or equal to 2 (where is the same dataset used to create the `example`), create its count matrix and compare with `example` data frame.

```
TNBC_12 <- TNBC %>% filter(sample_id <= 2)
test <- create_countMat(TNBC_12)

head(test)
```

```
##          2  3 5 6  7  8  9 10 11 12 13 16 17 18 19 20  21 22 23 24 25 26 27 28
## sample_1 1  5 1 0  1 27 19 17 26 32  3  1  6  0 22 29 244 56 19  8  3 15  3  6
## sample_2 0 28 0 6 17  0 11  3  0  0  0  4  3 13 25 77   0  0  0  0  2 11  0  1
##          29 30  31 32 33 34 35 36 37 38 39 40 42 44 45 46 48 49 50 51 52 53 54
## sample_1  8 13 266 11 16 38 18 10 12  0 36 19 19  9 69  9  3  1  1  1  1  0 62
## sample_2 11 66   0  0  0  0  4  0 17  1 16 74  2  0 12  0  6 28 69  0  2  2  1
##          55 56 57 58 59 60 61 63 64  65  66 67 68 69 70 71 73  75 76 77 78 79
## sample_1 70 30 58  3  0  8  2  2  5   0   0 27 11  1  0  1  2   1  4 53 13  0
## sample_2 32 51 21  5  4 27  0  3  2 145 452 13  6  5 13  0  5 476  6 10 22 37
##          80 81 82  83 85  86 87 88 89 90 91 92  93  94  95 96  97 98 99 100 101
## sample_1  2  0  0 377  0 123 28  2  2  3  0  0   0   0   0 14 204 96 25   1   0
## sample_2  2  1  1   6 14  13 40 18  6  5  5  2 185 110 349 43   5 23 10   2 119
```

```
##          113 114 128 129 143 157 158 172 175 187 211
## sample_1 122  51 363 173 455 820 257 135   8 420  61
## sample_2   6   2  25  92  79   6   0   0   0   1  16
```

```
all.equal(test, example)
```

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

As results shown, `test` count matrix is the same count matrix as `example` count matrix.

Now, we want to apply `create_countMat()` to the `TNBC` dataset.

```
test_all <- create_countMat(TNBC)
dim(test_all)
```

```
## [1]  39 113
```

```
sum(test_all)
```

```
## [1] 179194
```

```
all.equal(length(rownames(test_all)), length(unique(TNBC$sample_id)))
```

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

```
all.equal(as.numeric(gsub("sample_", "", rownames(test_all))), sort(unique(TNBC$sample_id)))
```

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

```
head(test_all)
```

```
##          1  2   3   4   5   6   7  8  9  10  11 12 13 14  15  16 17 18  19  20
## sample_1  0  1   5   0   1   0   1 27 19  17  26 32  3  0   0   1  6  0  22  29
## sample_2  0  0  28   0   0   6  17  0 11   3   0  0  0  0   0   4  3 13  25  77
## sample_3 40 16  48  19  26  28  39 25 10   3 127 19 85 63  22  36 20 26  56  36
## sample_4  1  3  34 110 292 279 134  4 44  37  25 21 21 25 159 116 30 23 154  86
## sample_5  3  2 274  10  35  71  30 12 14 119  10 48 22 88  45  51  1  8  44  51
## sample_6  0  1   3   2   4  19  46  0 16  23   0 21  5  7  42 127 67 19  93 128
##          21  22 23 24 25  26  27 28 29 30  31 32  33 34 35  36 37 38  39 40
## sample_1 244  56 19  8  3  15   3  6  8 13 266 11  16 38 18  10 12  0  36 19
## sample_2   0   0  0  0  2  11   0  1 11 66   0  0   0  0  4   0 17  1  16 74
## sample_3 217 130 86 47 41  12  18 22 53 38  47 78 159 35 50  21 10  7  64 40
## sample_4  52   8 18 11 62  28 232 55 87 86   5 65  13 36 16 107 36 53 121 59
## sample_5   8  16 29 34  0   0   0 19 21 34   2  3  13  6  1   0  5 11  54 28
## sample_6   4   2  4 17 72 152 594 12 10 80   2 14   2 12 29 313 34  3  21 35
##          41 42 43 44 45 46 47 48 49 50 51 52 53  54 55 56  57 58 59 60 61 62
## sample_1   0 19  0  9 69  9  0  3  1  1  1  1  0  62 70 30  58  3  0  8  2  0
## sample_2   0  2  0  0 12  0  0  6 28 69  0  2  2   1 32 51  21  5  4 27  0  0
## sample_3 149 92 70 30 49 27  2 37 79 17 54 11 64  24 44 34  42 39 41 16  3  3
## sample_4   4 82  0 34 39 96 57 85 48 28 16 10  1  24 67 94 128 20 29 39 48 13
```

```
## sample_5   8  1 19  0  0  1  2 28 16 15  1 13 36 294 39 20  37 26 16 65 10 18
## sample_6   1 10  6 41  8 48 21 17  2 19  0  9  0   1 19 75  55  7  0 50 12  0
##          63 64  65  66 67  68 69 70 71  72 73  74  75  76  77  78  79 80 81 82
## sample_1  2  5   0   0 27  11  1  0  1   0  2   0   1   4  53  13   0  2  0  0
## sample_2  3  2 145 452 13   6  5 13  0   0  5   0 476   6  10  22  37  2  1  1
## sample_3  7 47  43   1 54 176 52  4  1   1  9  27  96 112 130  85 156 42  0 15
## sample_4 31 31  79  18 70  52 52  4  7  14 14  72 143   8  56  98  14 12 19  8
## sample_5 46 19  53  40 34  72 95 57  2   2 12 245 114  19  79 173  38 23 13 14
## sample_6  0  5  18  62 37  11  7  8 14 115  3   1  38  12  82  62   7  2 58  2
##           83 84  85  86  87  88  89 90 91  92  93  94  95  96  97 98  99 100
## sample_1 377  0   0 123  28   2   2  3  0   0   0   0   0  14 204 96  25   1
## sample_2   6  0  14  13  40  18   6  5  5   2 185 110 349  43   5 23  10   2
## sample_3  26 45 120  65 168  36 140 38  0   1   3  44  14 195 250 84  97 111
## sample_4  25 80   7  14  24  33  32 11 15 164   3  25   6   3  32 32  91  11
## sample_5  15 87  26  36 131 126 113  6  1  13   7  58  10  50  86 98  40  10
## sample_6  19 26  17  40  97  27   6  3 85 329   4  22  19  18  49 62 142   0
##          101 113 114 128 129 143 157 158 159 172 175 187 211
## sample_1   0 122  51 363 173 455 820 257   0 135   8 420  61
## sample_2 119   6   2  25  92  79   6   0   0   0   0   1  16
## sample_3   2  22  31  18  10  28  56  59  14  19  18  43  10
## sample_4  11  16 125  25  60  16   1   0   2   0   4   0   1
## sample_5  23  15  25  29  10  38  10  13  57   4  29   7  42
## sample_6 129  22 230  82 153 123   2   0  16   0 133   0  59
```

By the above results, we have create a `test_all` count matrix with dimension of 39 rows(samples) and 113 columns(cluster_id) with 179194 cells.