How-to-RollUp

Thursday, 20 April 2017 13:49

Understanding the inner workings of Rollup functions -

Function : Rrollup

Input :

1. Data which has samples in columns and peptides/proteins in rows,
2. Accessions for the entity on which you want to collapse the data i.e protein accession, peptide sequence
3. Various numerical parameters

Calls : protein.rollup1, protein.rollup2, remove.outliers, plotCurrProt.RefRup, rm.outlier.1, outlier.1, remove.duplicates and CountRollup

Start : RRollup

1. Filter the data only to keep those rows that contain data in > "minPres" columns. Default is 50% .
    a. In our data, we only have 2 columns "Heavy" and "Light"
    b. So 50% of two columns = 1
    c. We keep all rows where there is a value in at least one of the columns
    d. So we can get value in Heavy or Light or both columns

2. The filter above applies to both the data and the corresponding accession numbers/sequences.

3. Normally, for a given protein you have multiple peptide hits. However, due to the nature of the experiment, you could have exactly one peptide for a given protein. These are referred to as "One-Hit-Wonders". For a given accession, you can decide whether or not you want to keep it if it is one-hit wonder.
    a. Weirdly enough, for "oneHitWonders=TRUE", the cut-off is 2. Huh ?

4. Calculate table of Accessions be it peptide sequence or protein accession
    a. We have 5929 unique peptide sequences where there is an abundance value for "Heavy" or "Light" or "both.

5. We decide what the cut-off for keeping peptide sequences is using "minPep".
   a. We have chosen 3.
   b. So any accessions with fewer than 3 peptides will be discarded
   c. There are 2355 peptide sequences represented by >= 3 rows/sets of values (sigIPI) = Nprots
   d. There are 3574 peptide sequences represented by <3 rows/sets of values

6. We calculate a value "threshold" which is equal to the minPres (50%) * number of columns (samples) rounded to 2 decimal place
   a. Ours is 50*2 = 100 1

7. Here, I use the first peptide as an example to run through the Rollup part of the code contains data
   a. prot = 1

      ```
      sigIPI[1] = "AAAAAAAAAAAAAAAGAGAGAK"
      ```

   b. Grab all the rows in the sigIPI matrix which are equal to the protein of interest and put it in 'pidx'

      ```
      [1]  2364  3566  5711 11341 13751 15103
      ```

   c. Extract data for the protein from the 'Data' object

      ```
      > currProtData
            light.log heavy.log
      10540  24.96499  21.83846
      17781  23.73169  20.59878
      28243  23.12108  20.33429
      55703  21.76830  16.84107
      67588  25.60964  21.48689
      75222  22.40458  19.00525
      ```

   d. Check that the number of samples that have values exceeds threshold of 1. This is true as we have already filtered for this.
   e. Call another rollup function called "protein.rollup1" which takes "currProtData" as input.
   f. Check how many samples each peptide is detected in

      ```
      > pepCounts
      10540 17781 28243 55703 67588 75222
          2     2     2     2     2     2
      ```

   g. If more than one peptide in the group is detected in the same number of (maximum) samples, then
      i. add abundances for each peptide across all samples
      ii. Find peptide with maximum abundance across all samples

iii. Make the peptide with the maximum abundance the reference

```
> totalAbundances
   10540    17781    28243    55703    67588
75222
46.80345 44.33047 43.45537 38.60937 47.09653
41.40983
```

iv. Reference is '67588' with abundance of 47.09

v. Subtract the value of the reference from each of the other peptides. Subtract sample1 reference from sample 1 and sample2 reference from sample 2.

```
> currSelAdj
       light.log  heavy.log
10540 -0.6446489  0.3515652
17781 -1.8779486 -0.8881144
28243 -2.4885636 -1.1526032
55703 -3.8413398 -4.6458210
67588  0.0000000  0.0000000
75222 -3.2050643 -2.4816422
```

vi. Don't understand the point of "overlapMedians" section, lines 224-227

vii. Similarly, we calculate currentSelAdj but never go onto use it as we replace it with currentSel which is just a scaling function.

viii. Only applies if there are fewer samples with values than in "minOverlap". MinOverlap is 3 and we only have 2 samples so this doesn't make a difference.

h. Perform a Grubbs test on the data to remove any outlier peptides and only retain the non-outliers. In our case, we retain all peptides

i. Scale all values to be centred around mean if the "center" option is TRUE which it is in our case. Else keep the values as is.

j. If the mode is "median", calculate column-wise median (one per sample) of the scaled, outlier free data. Else calculate the "mean" for the same data.

k. Calculate the standard deviation for the same data.

l. Across all peptides, calculate the median or mean of scaled values for each sample

m. Return the scaled values for all samples, scaled value for non-outlier samples, meadian/mean value for each sample and the standard deviation for non-outlier samples.

n. Add peptide name to the scaled values

```
> rownames(protData) <- proteinNames
> protData
```

```
                         light.log heavy.log
AAAAAAAAAAAAAAAGAGAGAK  1.633059 -1.633059
```

o. Follow simple steps to include oneHitWonders in a similar way. For our peptide, we have 6 matches so not a one-hit wonder.
p. Assign this value to "outProtData"
q. If requested with "reportCount = T", then return the count of how many peptide matches there were (here = 6)

8. Run step 7 for the next peptide
9. With each step, the results are amalgamated.
10. Return the list containing scaledData, rolledUpData, count of matches used in Rollup, standard deviation in data across all peptides