gpGrouper Documentation and Methodology

Alexander Saltzman

July 8, 2018

Contents

1 Overview of the Program

gpGrouper is a program for grouping peptides from bottom-up proteomics data. It implements logic for handling peptides shared across multiple proteins and genes. It is designed to map peptides into gene products instead of proteins to cut down on the number of shared peptides. This is not a hard requirement, but grouping into proteins instead of gene products significantly reduces the number of data points with unique peptides, making quantification much more difficult.

The program distributes the precursor areas for shared peptides across multiple gene products. It distributes the area for a given shared peptide to a gene product based on the ratio of that gene product's unique peptide area over the unique peptide areas for all gene products. In the case of a shared peptide that has no gene products with unique peptide area, the shared peptide is simply divided evenly among all gene products it belongs to. A gene product with peptides that are form subset of another gene product that also has unique peptides is not allocated any peptide area and gets an value of zero.

Additionally, peptides that are shared between species are distributed across them based on the estimated ratio of gene products within each species. This ratio is calculated via the sum of all peptides unique to each species. This is useful for PDX samples which have additional peptide overlap between species.

All of these shared peptide distributions are estimations that do not necessarily reflect reality. However, we do believe that this area distribution is more accurate than alternative methods such as the principle of parsimony - in which the protein (or gene product) with the most unique peptides is given all of the shared peptide area. Additionally, gpGrouper also reports the unique to gene product areas if that method of quantification is desired in downstream analysis.

gpGrouper is implemented in Python and Pandas and is available for download on github.

2 Parameters

2.1 Definitions

Below are a list of values calculated for each gene product their definition.

IDSet Designation of peptide overlap with other GeneID.

- 1 Has at least 1 peptide unique for that GeneID.
- 2 Shares all peptides with other GeneIDs, is **not** a subset of any other GeneID.
- 3 Shares all peptides with other GeneIDs, is a subset of at least one other GeneID.

IDGroup / IDGroup u2g Quality bins for each GeneID based on the best peptide based on all and unique to gene peptides (see Table 1).

GPGroup A unique identifier for a set of peptides, similar to a protein group in other software. IDSet 1 GeneIDs have unique GPGroups and overlapping IDSet2 GeneIDs share GPGroups.

GPGroup All A list of all GPGroups that collectively represent the GPGroups each to which a peptide belongs.

S (designation) A filter by strict PSMs/Peptides, which are PSMs with an IDGroup ≤ 3.

u2g (designation) A filter by unique to gene PSMs/peptides.

Coverage Total peptide coverage (ratio) for each gene product.

Coverage_u2g Total peptide coverage using unique-to-gene (ratio) for each gene product.

ProteinGI_GIDGroups Pipe delimited (|) sets of distinguishable ProteinGIs

AreaSum max The maximum possible GeneID area, no area redistribution

AreaSum gpcAdj The GeneID area as calculated after the division of each PSM's area by the total number of GeneIDs that map to each PSM.

AreaSum u2g all Same as **AreaSum gpcAdj** but after filtering by unique peptides.

AreaSum u2g 0 Same as **AreaSum gpcAdj** but after filtering by unique peptides with zero miscuts.

AreaSum dstrAdj The GeneID area after adjustment of each shared PSM's area among each mapped GeneID based on the GeneID area as calculated by unique PSMs. IDSet 2 GeneIDs that have no unique PSMs are adjusted by dividing by the total number of IDSet 2 GeneIDs. IDSet 3 GeneIDs recieve a **AreaSum dstrAdj** of 0.

SRA Strict/Relaxed/All quality bins. S = IDSet 1 & IDGroup_u2g 1-3 or IDSet 2 & IDGroup 1-3. R = (IDSet1 & IDGroup 1-5 & IDGroup_u2g 4-5) or (IDSet 2 & IDGroup4-5). A = (IDSet 1 & IDGroup 1-9 & IDGroup_u2g 6-9) or (IDSet 2 & IDGroup 6-9) or IDSet 3

2.2 IDGroup Bins

Default IDGroup bins, designed around Mascot IonScores. This is customizable at runtime.

Table 1: IDGroup Bins

IonScore (i)	q-value (q)	IDGroup
i $\geq$ 30	q $\leq$ 0.01	1
30 i $\geq$ 20	q $\leq$ 0.05	2
30 i $\geq$ 20	q $\leq$ 0.01	3
20 i $\geq$ 20	q $\leq$ 0.05	4
20 i $\geq$ 10	q $\leq$ 0.01	5
10 i $\geq$ 10	q $\leq$ 0.05	6
10 i $\geq$ 10	q $\leq$ 0.01	7
$0 \leq i$	q $\leq$ 0.05	8
$0 \leq i$	q $\leq$ 1.00	9

These IonScore cutoff bins (10, 20, 30) have been empirically translated to SpectrumMill and MaxQuant PSM Scores:

SpectrmMill 7, 10, 13

MaxQuant (Andromeda) 66, 91, 114

3 Initial Processing

Explanation of the internal workings of the grouping process.

3.1 Set Up

Each experiment is held in a UserData container class created at startup. First, in the `set_up` function, the input PSMs file is loaded and some initial set up operations are performed. Columns are renamed if appropriate (see below). If a `q-value` column is not found, it is assigned by dividing the `Posterior Error Probability` column by 10 for a rough approximation. If a `MissedCleavages` column is not found, the number of missed cleavages for each PSM is calculated. For Thermo Proteome Discoverer files, the `SequenceModi` annotation column is assigned which annotates the amino acid sequence position ally with modifications. The number of modifications is also recorded; for TMT experiments the TMT modifications are not counted toward the modification count.

MaxQuant derived PSM files already have this annotated and are not calculated.

3.2 Column Renaming

If the column aliases dictionary is provided, the header columns in the input PSMs file are renamed appropriately. An input dictionary containing the proper name mapping to the potential incoming names is used to look for matches:

```
IonScore -> Ionscore, Ions Score, ionscore
Sequence -> Sequence, Annotated Sequence
```

the `column_identifier` function filters the one to many mapping to a 1:1 mapping with the standardized name with the incoming name found in each PSMs file as appropriate. This filtered mapping is then used to change the original column names.

3.3 Matching to Database

The input databases are used to match with the input PSM files. Each sequence is digested *in silico* into peptides and matched to the input PSMs files. In the current implementation, a relation is first made between each peptide in the database to the indices in the database in which the peptide is present. During this process the peptide capacity for each entry is also calculated[1]. These indices are stored in the PSMs data for extraction of the relevant metadata later. This, as well as chunked enzymatic digestion of the peptidome, is done to reduce the memory footprint. Next, the genes, proteins, homologenes, taxa, as well as the counts for each of these is accumulated for each PSM based on the indices.

4 Grouping

4.1 Filtering IDs for Multiple Taxa

gpGrouper has an option to specify a list of IDs to ignore when estimating the ratio of each taxon when appropriate. This file structure is just a simple list with each ID on a separate line. Typically it is worth filtering out keratins, which are a common human contaminant that will over-estimate the amount of human.

4.2 Assigning IDG

Each PSM is binned into one of 8 possible `IDG` (quality) bins (1 being the best). First, each PSM is assigned 1, 3, 5, or 7 based on their IonScore. The defaults assign 1 to IonScore >= 30, 3 to IonScore between 20 and 30, 5 with IonScore between 10 and 20, and 7 with IonScore less than 10. Then, each PSM with a q-value greater than 0.01 has their IDG bin increased by 1 to yield the 8 possible IDG values.

4.3 Redundant Peak Removal

Often, the same peptide is identified multiple times as it is eluting off of the column. We filter these redundant PSMs by dropping these duplicates and keeping the one with the highest IonScore. This best PSM gets a `Peak_UseFLAG` of 1, while the duplicates get a value of 0. Duplicates are PSMs with an identical `SpectrumFile` (mass spec fraction), `SequenceModi`, `Charge`, and `PrecursorArea`.

4.4 Summing Areas of Similar PSMs

The same PSM may be observed multiple times, for example across mass spec fractions. For PSMs with `Peak_UseFLAG = 1`, the areas for PSMs with the same Modified Sequence and Charge are summed to yield the `SequenceArea`.

4.5 AUC Re-Flagging

Similar to the considerations for removing redundant peaks, multiple PSMs may have the same `SequenceArea`. Here, duplicates are PSMs with the same `SequenceArea`, `Charge`, and `SequenceModi`. The best (based on `IDG`) PSM is given a `AUC_reflagger` value of 1 and any others are given a value of 0.

[1] The number of single miscut peptides that result from the protein sequence at or above a minimum specified length (set here to 7).

4.6 Splitting PSMs on GeneID

Each PSM record is duplicated for each identifier it maps to. So this:

```
PSM  GeneList  IDG  SequenceArea  ...
0    1,2,3     30       100
```

becomes this:

```
PSM  GeneList  GeneID  IDG  SequenceArea  oriFLAG  ...
0    1,2,3     1       1        100           1
0    1,2,3     2       1        100           0
0    1,2,3     3       1        100           0
```

All information is duplicated, with the original record marked as such with the created `oriFLAG` column. This is used later when different Identifiers are assigned different distributed area. Filtering by `oriFLAG = 1` is useful when performing analytics on the original PSMs data.

4.7 AUC and PSM Flags

`AUC_UseFLAG` and `UseFLAG` columns are designations for the use of each PSM for area and count calculations. Both are zeroed out if any value falls outside the preset filter value range. These filters include the minimum and maximum charge, the minimum ion score, the minimum q-value, the minimum PEP value, and the maximum `IDG` value. There is some redundancy here, i.e. if something is below the minimum ion score it will be above the maximum `IDG` value.

Thermo's Proteome Discoverer tags each PSM with a `PSMAmbiguity` value. This can take a value of `Ambiguous` or `Unambiguous` depending on the ambiguity of the PSM. For example, a PSM may be equally matched to have a leucine or isoleucine in a certain position due to the equal masses of the amino acids, or the position of a modification may be ambiguous due to the lack of a b or y ion. If a given PSM has a `PeakUseFLAG` of 0 yet also has a `PSMAmbiguity` of `Unambiguous`, the AUC and PSM flags are set to 1. Else if the PSM is `Ambiguous` the flags are set to 0.

Finally, if the `AUC_reflagger` flag is 0 (see above), the `AUC_UseFLAG` is set to 0.

4.8 PSM Area Redistribution Based on TaxonID

If there is more than one taxon present in the data (based on an input database with multiple taxa) the areas of PSMs shared across taxa are divided up appropriately. A flag is available to turn this off if desired.

First, an estimated ratio of each taxon is calculated by dividing the sum of all PSMs unique to taxon by the sum of all PSMs unique to any one taxon. Then each PSM that is shared between taxa is multiplied by that ratio. As an example, for a PSM shared between human and mouse, the record associated with the human gene identifier is multiplied by the human taxon ratio and the record associated with the mouse gene identifier is multiplied by the mouse taxon ratio.

Currently this is only set up to support two taxa. More than two taxa will not cause an error in the program, but the logic is not set up correctly. Specifically, logic needs to implemented to deal with a PSM that is shared between $n < N$ total taxa; currently it will simply be multiplied by the ratio of one taxon divided by the sum of all.

4.9 Isobaric Labels

gpGrouper supports isobaric labeled experiments. The program looks for the reporter ion values if specified. Like other columns, aliases can be set ahead of time in a gpGrouper config file. The PSMs are further split based on the different isobaric labels. A `PrecursorArea_split` is calculated for each PSM for each label, which is the original `PrecursorArea` multiplied by the ratio of that particular label over the sum of all of the labels.

4.10 SILAC

4.11 Peptide and PSM Counts

The total, strict, unique to gene, and strict unique to gene PSM and peptide counts are calculated for each gene identifier. A strict peptide is defined as having a `IDG` $<=3$.

5 Area Calculations

PSMs with `AUC_UseFLAG` of 1 are used for area calculations. Area calculations are performed in steps. First, the non-distributed areas are calculated for each GeneID. Then the distributed area for each shared PSM is able to be calculated. Finally, the distributed area for each GeneID is calculated.

5.1 Non-Distributed Areas

Non-distributed area is calculated through the aggregation of PSMs in a variety of ways. Maximum area without any filtering or manipulation (`nGPArea_Sum_max`), gene count normalized area (`nGPArea_Sum_cgpAdj`), unique to gene area (`nGPArea_Sum_cgpAdj`), and unique to gene area after filtering for no miscuts (`nGPArea_Sum_cgpAdj`) are calculated.

5.2 Distribution of PSM Areas

Here the assignment of `PrecursorArea_dstrAdj` occurs. For PSMs with only a single gene identifier they simply take on a copy of their `SequenceArea` value. PSMs with a `AUC_UseFLAG` of 0 receive a distributed area of 0. PSMs that map to multiple gene identifiers are redistributed. This occurs separately for each gene identifier for a given PSM.

5.2.1 PSMs with mappings that have nonzero unique to gene area

For a given PSM mapping to GeneID n with a total `SequenceArea` s:

$$distArea_n = s_n \times \frac{u2g_n}{\sum_i^N u2g_i} \tag{1}$$

with N the total set of GeneIDs that the PSM maps and $u2g_i$ the unique to gene area for each of the N GeneIDs. Note : see below for a special case of this scenario when the unique to gene area is zero yet there are unique to gene peptides.

5.2.2 PSMs with mappings that do not have unique to gene area

For a given PSM with multiple GeneID mappings, none of which have any unique to gene areas:

$$distArea_n = s_n \times \frac{1}{M} \times \frac{u2taxon_n}{\sum_i^N u2taxon_i} \tag{2}$$

with M the total **count** of GeneIDs that the PSM maps, and $u2taxon_i$ the unique to gene area of each of the N taxons.

For experiments with only one taxon this last term is 1.

1. Special Case : Shared Peptides with Unique to Gene Peptide Area of 0. It is possible for a PSM to exist without a quantified precursor area. On rare occasions, a situation can emerge in which all peptides that are unique to a specific gene have no quantified precursor area (the only observed case thus far has been for genes with a single unique peptide). In these cases, the distributed area for these shared peptides will be zero for that gene. In the simplest case with a peptide shared among

two genes, one of them having a unique area of 0, all of the peptide are will be redistributed to the other gene. However, this behavior is more in line with how peptide area is distributed for set 3 genes - no unique peptides and is a subset of another gene.

5.3 Gene Level Distributed Area

For gene level distributed area, `nGPArea_Sum_dstrAdj`, the `PrecursorArea_dstrAdj` calculated previously is summed on a per gene (and per label, as appropriate) level. Any IDSet 3 GeneID is assigned an area of 0.

5.4 iBAQ Calculation

iBAQ (intensity based absolute quantification) is calculated by dividing the AUC by the peptide capacity - the number of peptides that result from a given protein. Here, the peptide capacity this is calculated on the gene level, consistent with this gene-centric approach to proteomics. For the cases in which there are multiple isoforms per gene, the peptide capacity is the average number of singly miscut peptides across all of the isoforms.