Introduction to *proBAMr*

Xiaojing Wang

October 29, 2024

Contents

1 Why proBAM file **1**

2 What is proBAM file **2**

3 How to build proBAM file **3**
3.1 Preparing annotation files 3
3.2 Preparing PSMs table 4
3.3 Generate SAM file using `PSMtab2SAM` 5
3.4 Convert SAM file to BAM and index 6
3.5 Visulize proteomics data in IGV 7

4 Session Information **7**

1 Why proBAM file

Recent advances of sequencing technologies have reformed our conception of genomic data analysis, storage and interpretation, instigating more research interest in exploring human proteome at a parallel scale. Shotgun proteomics holds this promise by surveying proteome both qualitatively and quantitatively. Over the last years large amount of proteomics data has been accumulated, an emerging demand is to combine these efforts to catalogue the wide dynamic range of protein expression and complexity of alternative isoforms. However, this task is daunting due to the fact that different studies use varying databases, search engines and assembly tools. Such a challenge calls for an efficient approach of integrating data from different proteomics studies and even with genomic data.

Here we provide an R package, *proBAMr*, that maps identified PSMs to the genome in BAM format, a binary format for efficient data storage and fast access in genomic research field. This method differs from other approaches because of its ability of building connections between peptide and genomic location and simultaneously maintaining spectra count information. PSMs are aligned under the same coordination framework regardless of the annotation systems (e.g. RefSeq, ENSEMBL) of the input proteomics data, which enables flexible protein assembly switch between different annotation or at different level (gene or protein). When genomic/transcriptomic information of the same individual is available, this approach allows the co-analysis with -omics data together.

2 What is proBAM file

Table 1: Mandatory field definition of proBAM file and compare to original BAM format for genomic studies.

Mandatory fields in BAM and proBAM				
No.	NAME	TYPE	BAM Description	proBAM Description
1	QNAME	String	Query template NAME	Spectrum name
2	FLAG	Int	Bitwise FLAG	Bitwise FLAG
3	RNAME	String	Reference sequence NAME	Reference sequence NAME
4	POS	Int	1-based leftmost mapping POSition	1-based leftmost mapping POSition
5	MAPQ	Int	MAPping Quality (Phred-scaled)	-
6	CIGAR	String	Extended CIGAR string (operations: MIDNSHP)	CIGAR string
7	RNEXT	String	Mate Reference NAME ('=' if same as RNAME)	-
8	PNEXT	Int	1-Based leftmost Mate POSition	-
9	TLEN	Int	observed Template LENgth	-
10	SEQ	String	segment SEQuence	Coding sequence
11	QUAL	String	Query QUALity (ASCII-33=Phred base quality)	-

To take full advantage of tools developed for processing BAM files in genomics studies, we designed proBAM by incorporating features from the BAM file format and other features specifically for proteomics. Like BAM, it contains a header section and an alignment section. A full description of the BAM format is available at `http://samtools.github.io/hts-specs/SAMv1.pdf`. A PSGM (peptide-spectrum-genomic location-match) is the basic unit in proBAM and is similar to a read in NGS data. In Table 1, we compared the BAM and proBAM description of each mandatory column in the alignment section.

Table 2: FLAG description in proBAM file.

FLAG description in proBAM**		
Bit	Description	FLAG
0x00	Peptide map to forward strand	0
0x10	Peptide map to reverse strand	16
0x100	Peptide is NOT the rank=1 peptide for the spectrum	256
0x400	Decoy peptide	1024
0x4	Unmapped peptide	4

proBAM allows for 5 FLAG values due to the less complicated requirements by shotgun proteomics

data (Table 2). For the same reason, the CIGAR tag in proBAM file only supports 'M' for match/mismatch and 'N' for skipped bases on the reference.

The optional filed keep extra information from proteomics experiment platform or search engines. The definition and value format of each optional column is descripted in Table 3. It is important to note that this table is extendable depending on continuous development and input from the community.

Table 3: Optional field definition of proBAM file.

Mandatory fields in proBAM*		
TAG	TYPE	Description
NH	i	Number of genomic locations the peptide mapping to
XL	i	Number of peptides the spectrum mapping to
XP	Z	Peptide sequence
XR	Z	Reference peptide sequence
XS	f	PSM score
XQ	f	PSM Q-value
XC	i	Peptide Charge
XA	Z	Whether the peptide is well annotated (0: yes ; 1: partially unknown; 2: totally unknown)
XM	Z	Modification
XN	i	Number of mis-cleavage
XT	i	0: non-tryptic; 1: semi-tryptic; 2: tryptic
XG	Z	Peptide type. N: normal peptide; V: variant peptide; J: novel junction peptides; D: decoy; U: unmapped

The optional field follow the rule TAG:TYPE:VALUE defined by BAM file. There are three type of VALUE format: i, Singed 32-bit integer; Z, Printable string; f, Single-precision floating number.

3 How to build proBAM file

3.1 Preparing annotation files

To map proteomics data to the genome, numerous pieces of genome annotation information are needed, such as genome elements region boundary, protein coding sequence and protein sequence et al. It is possible to manually download these data from different public resources (e.g. NCBI, UCSC and ENSEMBL) and then parse them to an appropriate format. To make this process more efficient and autonomous, we provide functions to prepare the gene/transcript annotation files from UCSC, ENSEMBL and GENCODE. The `PrepareAnnotationRefseq` and `PrepareAnnotationEnsembl` were included in another R package *customProDB* `http://bioconductor.org/packages/3.0/bioc/html/customProDB.html`. Here, we provide the function `PrepareAnnotationGENCODE` to prepare the annotation from GENCODE. This function requires users to download GTF file, coding sequence and protein sequence FASTA files from GENCODE ftp `ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/`. Users should use the same version of annotations through the same project annalysis. All the annotations are saved to a specified directory for latter use.

```
> library(proBAMr)
```

```
> gtfFile <- system.file("extdata", "test.gtf", package="proBAMr")
> CDSfasta <- system.file("extdata", "coding_seq.fasta", package="proBAMr")
> pepfasta <- system.file("extdata", "pro_seq.fasta", package="proBAMr")
> annotation_path <- tempdir()
> PrepareAnnotationGENCODE(gtfFile, CDSfasta, pepfasta,
+                 annotation_path, dbsnp=NULL,
+                 splice_matrix=FALSE, COSMIC=FALSE)
```

3.2 Preparing PSMs table

After preparing all the annotation files, the R package *pepXMLTab* is used to extract confident PSMs and related information from pepXML files. Other tools are also applicable at this step, as long as it generates similar tabular files, as shown below.

```
> passedPSM <- read.table(system.file("extdata", "passedPSM.tab",
+     package="proBAMr"), sep='\t', header=TRUE)
> passedPSM[1:3, ]
```

```
                                  spectrum
1 00463_H12_P003361_B00L_A00_R1.9484.9484.2
2 00463_H12_P003361_B00L_A00_R1.9501.9501.2
3 00463_H12_P003361_B00L_A00_R1.9526.9526.2
                               spectrumNativeID start_scan end_scan
1 controllerType=0 controllerNumber=1 scan=9484       9484     9484
2 controllerType=0 controllerNumber=1 scan=9501       9501     9501
3 controllerType=0 controllerNumber=1 scan=9526       9526     9526
  precursor_neutral_mass assumed_charge index retention_time_sec
1               1945.011              2  1604           5941.112
2               1945.019              2  1614           5951.951
3               1945.016              2  1631           5963.760
  hit_rank            peptide peptide_prev_aa peptide_next_aa
1        1 VNPTVFFDIAVDGEPLGR               M               V
2        1 VNPTVFFDIAVDGEPLGR               M               V
3        1 VNPTVFFDIAVDGEPLGR               M               V

1 ENST00000415933.1|ENSG00000196262.9|OTTHUMG00000023687.5|OTTHUMT00000341788.1|PPIA
2 ENST00000415933.1|ENSG00000196262.9|OTTHUMG00000023687.5|OTTHUMT00000341788.1|PPIA
3 ENST00000415933.1|ENSG00000196262.9|OTTHUMG00000023687.5|OTTHUMT00000341788.1|PPIA
  num_tot_proteins calc_neutral_pep_mass    massdiff num_tol_term
1                4              1944.995 -0.01667663           2
2                4              1944.995 -0.02461120           2
3                4              1944.995 -0.02192565           2
  num_missed_cleavages num_matched_ions tot_num_ions      mvh
1                    0               21           31 46.20442
2                    0               26           31 60.50605
3                    0               22           31 52.51659
```

```
  mzFidelity    xcorr modification NTT
1   81.97849 4.276119         <NA>   1
2  104.25397 5.334539         <NA>   1
3   86.18693 5.057394         <NA>   1
```

3.3 Generate SAM file using `PSMtab2SAM`

The function `PSMtab2SAM` first finds the peptide location in protein sequences, then maps the coding sequence of the peptide back to the genome according to the annotation.

```
> load(system.file("extdata/GENCODE", "exon_anno.RData", package="proBAMr"))
> load(system.file("extdata/GENCODE", "proseq.RData", package="proBAMr"))
> load(system.file("extdata/GENCODE", "procodingseq.RData", package="proBAMr"))
> options(stringsAsFactors=FALSE)
> passedPSM <- read.table(system.file("extdata", "passedPSM.tab",
+     package="proBAMr"), sep='\t', header=TRUE)
> SAM <- PSMtab2SAM(passedPSM, XScolumn='mvh', exon, proteinseq,
+     procodingseq)
> write.table(SAM, file=paste(tempdir(), '/test.sam', sep=''),
+         sep='\t', quote=FALSE, row.names=FALSE, col.names=FALSE)
> dim(SAM)

[1] 40 21

> SAM[20:27, ]

                                       QNAME X1    X2        X3  X4
110 00463_H12_P003361_B00L_A00_R1.0.1.7307 16 chr11  65622810 255
113 00463_H12_P003361_B00L_A00_R1.0.1.7350  0  chr7  44839340 255
114 00463_H12_P003361_B00L_A00_R1.0.1.7441  0  chr7  44836381 255
115 00463_H12_P003361_B00L_A00_R1.0.1.7457  0  chr7  44836381 255
116 00463_H12_P003361_B00L_A00_R1.0.1.7898 16  chr5 133509648 255
117 00463_H12_P003361_B00L_A00_R1.0.1.7915 16  chr5 133509648 255
118 00463_H12_P003361_B00L_A00_R1.0.1.7933 16  chr5 133509648 255
119 00463_H12_P003361_B00L_A00_R1.0.1.7952 16  chr1  26230237 255
             X5 X6 X7 X8
110         42M  *  0  0
113         45M  *  0  0
114 12M2453N24M  *  0  0
115 12M2453N24M  *  0  0
116         51M  *  0  0
117         51M  *  0  0
118         51M  *  0  0
119         39M  *  0  0
                                                        X9 X10    X11
110          CAAAGGCTTGCCCTCCAGGGAGATGACGGCACTGCCCCCCAG   * XA:Z:0
```

```
113       TCCATCTATGGGGAGAAATTTGAAGATGAGAACTTCATCCTAAAG   * XA:Z:0
114                GTCTCCTTTGAGCTGTTTGCAGACAAGGTCCCAAAG   * XA:Z:0
115                GTCTCCTTTGAGCTGTTTGCAGACAAGGTCCCAAAG   * XA:Z:0
116 TTTGGCAATTTCCACATCAACTTCAAATATCTCTCCATCAGAACTCTGCAA   * XA:Z:0
117 TTTGGCAATTTCCACATCAACTTCAAATATCTCTCCATCAGAACTCTGCAA   * XA:Z:0
118 TTTGGCAATTTCCACATCAACTTCAAATATCTCTCCATCAGAACTCTGCAA   * XA:Z:0
119             CCGAGGGCTGAGAATCAGCTCAAAAGCCTGGCCTGAGGC   * XA:Z:0
        NH     XL                      XP     XC           XS     XM
110 NH:i:1 XL:i:1    XP:Z:LGGSAVISLEGKPL XC:i:2 XS:f:30.6722 XM:Z:-
113 NH:i:1 XL:i:1   XP:Z:SIYGEKFEDENFILK XC:i:2 XS:f:22.4909 XM:Z:-
114 NH:i:1 XL:i:1      XP:Z:VSFELFADKVPK XC:i:2  XS:f:21.321 XM:Z:-
115 NH:i:1 XL:i:1      XP:Z:VSFELFADKVPK XC:i:2 XS:f:25.2581 XM:Z:-
116 NH:i:1 XL:i:1 XP:Z:LQSSDGEIFEVDVEIAK XC:i:2 XS:f:30.9829 XM:Z:-
117 NH:i:1 XL:i:1 XP:Z:LQSSDGEIFEVDVEIAK XC:i:2 XS:f:42.8235 XM:Z:-
118 NH:i:1 XL:i:1 XP:Z:LQSSDGEIFEVDVEIAK XC:i:2 XS:f:33.9951 XM:Z:-
119 NH:i:1 XL:i:1     XP:Z:ASGQAFELILSPR XC:i:2 XS:f:23.4655 XM:Z:-
        XN     XT     XG
110 XN:i:0 XT:i:2 XG:Z:N
113 XN:i:1 XT:i:2 XG:Z:N
114 XN:i:1 XT:i:2 XG:Z:N
115 XN:i:1 XT:i:2 XG:Z:N
116 XN:i:0 XT:i:2 XG:Z:N
117 XN:i:0 XT:i:2 XG:Z:N
118 XN:i:0 XT:i:2 XG:Z:N
119 XN:i:0 XT:i:2 XG:Z:N
```

3.4 Convert SAM file to BAM and index

Add the header to the SAM file. Converted them to the binary BAM files using samtools `http://samtools.sourceforge.net/`. Sort and index them for fast access.

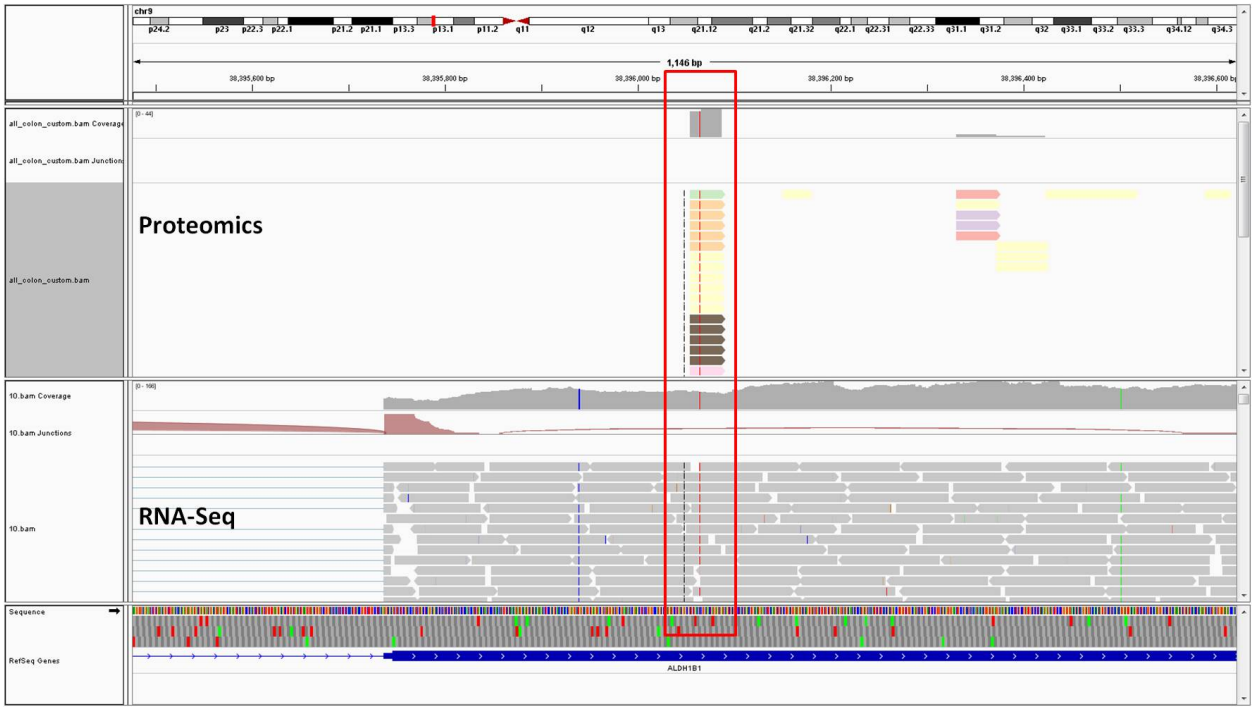
The bullet list below summarizes the steps after the SAM file been generated.

```
> paste('cat header test.sam > test_header.sam')

[1] "cat header test.sam > test_header.sam"

> paste('samtools view -S -b test_header.sam > test_header.bam')

[1] "samtools view -S -b test_header.sam > test_header.bam"

> paste('samtools sort test_header.bam > test_header_sort')

[1] "samtools sort test_header.bam > test_header_sort"

> paste('samtools index test_header_sort')

[1] "samtools index test_header_sort"
```

3.5 Visulize proteomics data in IGV

The proBAM files can be visulized in IGV directly. Furthermore, users can co-visulize their proteomics data with the paired genomics/transcriptomics data, as shown in Fig 1.

Figure 1: IGV snapshot of a homozygous mutation in gene ALDH1B1 in both proteomics and RNA-Seq data (inside read box)



4 Session Information

```
R version 4.4.1 (2024-06-14)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.1 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.20-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
```

```
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets
[7] methods   base

other attached packages:
[1] proBAMr_1.40.0       AnnotationDbi_1.68.0 Biobase_2.66.0
[4] IRanges_2.40.0       S4Vectors_0.44.0     BiocGenerics_0.52.0

loaded via a namespace (and not attached):
 [1] KEGGREST_1.46.0             SummarizedExperiment_1.36.0
 [3] rjson_0.2.23                httr2_1.0.5
 [5] lattice_0.22-6              generics_0.1.3
 [7] vctrs_0.6.5                 tools_4.4.1
 [9] bitops_1.0-9                curl_5.2.3
[11] parallel_4.4.1              tibble_3.2.1
[13] fansi_1.0.6                 RSQLite_2.3.7
[15] blob_1.2.4                  pkgconfig_2.0.3
[17] Matrix_1.7-1                dbplyr_2.5.0
[19] lifecycle_1.0.4             GenomeInfoDbData_1.2.13
[21] compiler_4.4.1              stringr_1.5.1
[23] Rsamtools_2.22.0            Biostrings_2.74.0
[25] progress_1.2.3              codetools_0.2-20
[27] GenomeInfoDb_1.42.0         RCurl_1.98-1.16
[29] yaml_2.3.10                 pillar_1.9.0
[31] crayon_1.5.3                BiocParallel_1.40.0
[33] DelayedArray_0.32.0         cachem_1.1.0
[35] abind_1.4-8                 tidyselect_1.2.1
[37] digest_0.6.37               stringi_1.8.4
[39] dplyr_1.1.4                 restfulr_0.0.15
[41] biomaRt_2.62.0              fastmap_1.2.0
[43] grid_4.4.1                  cli_3.6.3
[45] SparseArray_1.6.0           magrittr_2.0.3
[47] S4Arrays_1.6.0              GenomicFeatures_1.58.0
[49] utf8_1.2.4                  XML_3.99-0.17
[51] filelock_1.0.3              prettyunits_1.2.0
[53] UCSC.utils_1.2.0            rappdirs_0.3.3
[55] bit64_4.5.2                 XVector_0.46.0
[57] httr_1.4.7                  matrixStats_1.4.1
[59] bit_4.5.0                   png_0.1-8
```

```
[61] hms_1.1.3                   memoise_2.0.1
[63] GenomicRanges_1.58.0        BiocIO_1.16.0
[65] BiocFileCache_2.14.0        rtracklayer_1.66.0
[67] txdbmaker_1.2.0             rlang_1.1.4
[69] glue_1.8.0                  DBI_1.2.3
[71] xml2_1.3.6                  jsonlite_1.8.9
[73] R6_2.5.1                    MatrixGenerics_1.18.0
[75] GenomicAlignments_1.42.0    zlibbioc_1.52.0
```