GMQL Functional Comparison with BEDTools and BEDOPS

Genomic Computing Group
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

This document presents a functional comparison of GMQL with the NGS tools BEDTools and BEDOPS, which are rather popular among biologists for genomic region manipulations. The comparative example below highlights a series of differences between GMQL and BEDTools and BEDOPS, which are hereby summarized.

- Experimental data selection in BEDTools and BEDOPS must be done manually, as these tools do not handle metadata and thus cannot support any selection over datasets. Conversely, GMQL supports selections using metadata.
- Annotation retrieval must also be done manually in BEDTools and BEDOPS, through the use of Genome Browsers (e.g. UCSC and Ensembl). In GMQL, annotations are selected in the same way as experimental data, by using selections.
- Differently from BEDTools and BEDOPS, GMQL does not require sorted inputs, since it automatically sorts them before processing.
- In order to perform the same operation on different samples, BEDTools and BEDOPS require the use of cycles and control structures, embedded in other languages. Conversely, GMQL allows multiple samples to be grouped in the same dataset and GMQL operations are applied in batch, i.e. to every sample in the dataset.
- BEDTools and BEDOPS do not have an internal standard for data format. Different scripts may lead to different output formats. This implies the need for external languages to manipulate outputs (e.g. AWK or SED), with longer scripts, and requires the knowledge of different scripting languages. Conversely, GMQL has a unique internal data format, for every region operation, and the output is always produced in GTF format.

Next, we present a relevant biological example which is fully developed using the three approaches.

Enhancer-gene distal binding relationship

***Description**:*

In this example we want to find distal regulatory elements (i.e. enhancers) that may interact with a gene of interest (the proximal element), through the CTCF transcription factor (TF). The distance of a distal element from a gene is considered starting from the gene transcription start site (TSS). First, we set the distance TF-gene; then we consider the presence of a putative enhancer region using its typical histone modifications. This example is also included in our submitted paper (Example 2); here, we provide a simplified version applied to single samples of a single cell line in order to be directly performed also in BEDTools and BEDOPS.

***Procedure*:**

Find all enriched regions (peaks) in the CTCF transcription factor (TF) ChIP-seq sample from the K562 human cell line which are the nearest regions farther than 100 kb from a transcription start site (TSS). For the same cell line, find also all peaks for the H3K4me1 histone modifications (HM) which are also the nearest regions farther than 100 kb from a TSS. Then, out of the TF and HM peaks found in the same cell line, return all TF peaks that overlap with both HM peaks and known enhancer (EN) regions.

1. GMQL

```
# TASK  1: DATA COLLECTION
# Select and retrieve ChIP-seq, TSS and enhancer data
# One sample selected for each TF, HM, TSS and EN dataset
TF = SELECT(dataType == 'ChipSeq' AND view == 'Peaks' AND cell == 'K562' AND antibody_target == 'CTCF'
      AND lab == 'Broad') HG19_PEAK;
HM = SELECT(dataType == 'ChipSeq' AND view == 'Peaks' AND cell == 'K562' AND
      antibody_target == 'H3K4me1' AND lab == 'Broad') HG19_PEAK;
TSS = SELECT(ann_type == 'TSS' AND provider == 'UCSC') HG19_ANN;
EN = SELECT(ann_type == 'enhancer' AND provider == 'UCSC') HG19_ANN;


# TASK 2: DATA PREPARATION
# Merge multiple overlapping or adjacent regions in each input file
# Although GMQL manages such regions correctly in each operation, including in the following JOINs, this
# step is required for comparison with BEDTools and BEDOPS since both these tools merge such regions
# internally before performing closestBed or closest-features operations, respectively
TF1 = COVER(1, ANY) TF;
HM1 = COVER(1, ANY) HM;
TSS1 = COVER(1, ANY) TSS;
EN1 = COVER(1, ANY) EN;


# TASK 3: DATA PROCESSING
# Extract only TF and HM peaks farther than 100 kb from a TSS
TF2 = JOIN(first(1) after distance 100000, right_distinct) TSS1 TF1;
HM2 = JOIN(first(1) after distance 100000, right_distinct) TSS1 HM1;

# Extract only farther TF peaks overlapping with farther HM peaks and enhancer regions
HM3 = JOIN(distance < 0, int) EN1 HM2;
TF_res = JOIN(distance < 0, right) HM3 TF2;

# STORING RESULTS
MATERIALIZE TF_res;
```

2. BEDTools

```
# TASK 1: DATA COLLECTION
# SELECT is not available in BEDTools; data selection must be done manually and data must be present locally
# Input data saved locally:
#     CTCF peaks:          wgEncodeBroadHistoneK562CtcfStdPk.broadPeak
#     H3K4me1 peaks:       wgEncodeBroadHistoneK562H3k4me1StdPk.broadPeak
#     TSS:                 TSS_h19.bed
#     Enhancers:           VistaEnhancers_hg19.txt


# TASK 2: DATA PREPARATION
# BEDTools requires sorted input data
sortBed -i wgEncodeBroadHistoneK562CtcfStdPk.broadPeak > CTCF_sorted.bed;
sortBed -i wgEncodeBroadHistoneK562H3k4me1StdPk.broadPeak > H3K4me1_sorted.bed;
sortBed -i TSS_hg19.bed > TSS_sorted.bed;
sortBed -i VistaEnhancers_hg19.bed > VistaEnhancers_sorted.bed;

# Merge multiple overlapping or adjacent regions in each input file
# BEDTools requires this operation to flatten all the overlapping regions
bedtools merge -i CTCF_sorted.bed > CTCF_merged.bed;
bedtools merge -i H3K4me1_sorted.bed > H3K4me1_merged.bed;
bedtools merge -i TSS_sorted.bed > TSS_merged.bed;
bedtools merge -i VistaEnhancers_sorted.bed > VistaEnhancers_merged.bed;


# TASK 3: DATA PROCESSING
# Extend TSS regions
# AWK is needed to set 100 kb distance from each TSS, by extending the coordinates of each original TSS
# If the start position of a TSS is less than or equal to 100000, the start coordinate of its extended region
# is set to 1
awk -v OFS="\t" '{ if ( $2-100000 <= 0 ) {print $1,1,$3+100000,$4,$5,$6} else
    {print $1,$2-100000,$3+100000,$4,$5,$6} }' TSS_merged.bed > TSS_extended.bed;

# Extract only TF and HM peaks farther than 100 kb from a TSS, by searching for the closest CTCF or
# H3K4me1 binding site to each extended TSS region
# BEDTools closestBed command produces a "NA" label if the reference file has no related closest peaks.
# To remove "NA" labels, a supplementary AWK operation is required
closestBed -t all -a TSS_extended.bed -b CTCF_merged.bed |
    awk -v OFS="\t" '{ if ($5 > 0) {print $4,$5,$6} }' > CTCF_closest.bed;
closestBed -t all -a TSS_extended.bed -b H3K4me1_merged.bed |
    awk -v OFS="\t" '{ if ($5 > 0) {print $4,$5,$6} }' > H3K4me1_closest.bed;

# CTCF_closest.bed and H3K4me1_closest.bed files contain duplicate regions, because a peak that is the
# closest to multiple TSSs is reported multiple times
# BEDTools lacks a "distinct" operator; AWK must be used to removed duplicates
awk '!x[$0]++' CTCF_closest.bed > CTCF_closest_distinct.bed;
awk '!x[$0]++' H3K4me1_closest.bed > H3K4me1_closest_distinct.bed;

# Extract only farther TF peaks overlapping with farther HM peaks and enhancer regions
bedtools intersect -a VistaEnhancers_merged.bed -b H3K4me1_closest_distinct.bed > H3K4me1_enhancers.bed;
bedtools intersect -a H3K4me1_enhancers.bed -b CTCF_closest_distinct.bed > CTCF_H3K4me1_enhancers.bed;
```

3. BEDOPS

```
# TASK 1: DATA COLLECTION
# SELECT is not available in BEDOPS; data selection must be done manually and data must be present locally
# Input data saved locally:
#     CTCF peaks:         wgEncodeBroadHistoneK562CtcfStdPk.broadPeak
#     H3K4me1 peaks:      wgEncodeBroadHistoneK562H3k4me1StdPk.broadPeak
#     TSS:                TSS_h19.bed
#     Enhancers:          VistaEnhancers_hg19.txt


# TASK 2: DATA PREPARATION
# BEDOPS requires sorted input data
sort-bed wgEncodeBroadHistoneK562CtcfStdPk.broadPeak > CTCF_sorted.bed;
sort-bed wgEncodeBroadHistoneK562H3k4me1StdPk.broadPeak > H3K4me1_sorted.bed;
sort-bed TSS_hg19.bed > TSS_sorted.bed;
sort-bed VistaEnhancers_hg19.bed > VistaEnhancers_sorted.bed;

# Merge multiple overlapping or adjacent regions in each input file
# BEDOPS requires this operation to flatten all the overlapping regions
bedops --merge CTCF_sorted.bed > CTCF_merged.bed;
bedops --merge H3K4me1_sorted.bed > H3K4me1_merged.bed;
bedops --merge TSS_sorted.bed > TSS_merged.bed;
bedops --merge VistaEnhancers_sorted.bed > VistaEnhancers_merged.bed;


# TASK 3: DATA PROCESSING
# Extend TSS regions
# AWK is needed to set 100 kb distance from each TSS, by extending the coordinates of each original TSS
# If the start position of a TSS is less than or equal to 100000, the start coordinate of its extended region
# is set to 1
awk -v OFS="\t" '{ if ( $2-100000 <= 0 ) {print $1,1,$3+100000,$4,$5,$6} else
    {print $1,$2-100000,$3+100000,$4,$5,$6} }' TSS_merged.bed > TSS_extended.bed;

# Extract only TF and HM peaks farther than 100 kb from a TSS, by searching for the closest CTCF or
# H3K4me1 binding site to each extended TSS region
# BEDOPS closest-features command produces a "NA" label if the reference file has no related closest peaks.
# To remove "NA" labels, a supplementary AWK operation is required
closest-features --closest --no-overlaps --no-ref TSS_extended.bed CTCF_merged.bed |
    awk -v OFS="\t" '{ if ($0 != "NA") {print $0} }' > CTCF_closest.bed;
closest-features --closest --no-overlaps --no-ref TSS_extended.bed H3K4me1_merged.bed |
    awk -v OFS="\t" '{ if ($0 != "NA") {print $0} }' > H3K4me1_closest.bed;

# CTCF_closest.bed and H3K4me1_closest.bed files contain duplicate regions, because a peak that is the
# closest to multiple TSSs is reported multiple times
# BEDOPS lacks a "distinct" operator; AWK must be used to removed duplicates
awk '!x[$0]++' CTCF_closest.bed > CTCF_closest_distinct.bed;
awk '!x[$0]++' H3K4me1_closest.bed > H3K4me1_closest_distinct.bed;

# Extract only farther TF peaks overlapping with farther HM peaks and enhancer regions
bedops --intersect VistaEnhancers_merged.bed H3K4me1_closest_distinct.bed > H3K4me1_enhancers.bed;
bedops --intersect H3K4me1_enhancers.bed CTCF_closest_distinct.bed > CTCF_H3K4me1_enhancers.bed;
```

***Discussion*:**

We progressively compared the data processing steps and results produced by the three considered systems when the three programs above are applied to CTCF and H3K4me1 signal enrichment called regions from NGS human samples aligned to the human genome assembly 19, as provided by ENCODE (https://genome.ucsc.edu/ENCODE/dataMatrix/encodeChipMatrixHuman.html), and on transcription start sites (TSS) and enhancer annotations, as provided by the UCSC database (https://genome.ucsc.edu/cgi-bin/hgTables) from SwitchGear Genomics (http://switchgeargenomics.com/) and Vista Enhancer (http://enhancer.lbl.gov/), respectively. Every input dataset contains one single sample (including 80,538 CTCF, 125,713 HeK4me1, 131,780 SwitchGear TSS and 1,339 Vista enhancer regions, respectively); each sample contains overlapping regions. For each step, we tested the concordance between the datasets produced by GMQL, BEDTools and BEDOPS.

In GMQL, overlapping regions are processed independently, while in BEDTools and BEDOPS they are implicitly considered mainly as a unique, merged region, leading to a potential loss of data. To compare the behavior of the three algorithms avoiding differences caused by different strategies for the management of overlapping regions, in all programs we performed a first step to merge overlapping regions.

Using the merged input, with the first two *JOIN* operations in GMQL, corresponding to the *closestBed* and *closest-features* operations in BEDTools and BEDOPS respectively, we obtained the same results for GMQL and BEDOPS: 40,687 CTCF and 41,007 H3K4me1 distinct peaks. BEDTools returned different results since the *closestBed* function cannot find the closest regions non-overlapping reference regions, but it considers overlapping regions (i.e. features) as the closest (i.e. at distance 0).

Another minor source of differences among the three considered systems is represented by the management of ties, i.e. features that are equidistant from a reference region. When occurring, in GMQL ties are both retained, whereas in BEDTools the *-t* option allows the user to specify which feature has to be retained: *left*, *right*, *all*. Conversely, in BEDOPS, the preference is always assigned only to the left feature, reducing the number of returned regions as compared to the other two systems. Although ties are not present in our comparative dataset, they may represent a potential issue for other comparisons.

When searching for regions at a given distance from (or intersecting with) a reference region, the same region can be correctly returned several times when it satisfies the searching constraints with respect to multiple reference regions. These duplicated regions can cause issues when the obtained results are used as input for a subsequent operation. In the GMQL *JOIN*, the *_distinct* option allows avoiding them; this is not possible in both BEDTools and BEDOPS, requiring the use of external languages (e.g. AWK) to remove them.

The second two (and last) *JOIN* (*intersect*) operations returned different final results in the three systems. BEDTools's differences are due to the aforementioned issue in the *closestBed* function, leading to a final result of 67 CTCF peaks. On the other hand, BEDOPS seems to miss some features while doing the intersection, finally reporting 14 CTCF peaks versus 23 in GMQL. The small number of CTCF peaks is mainly due to the very limited amount of available Vista enhancers.