Testing STR calls against the GIABTR benchmark

Andrea Finocchio

Files

File	Description
VCF	A *.repeats.vcf file from DRAGEN
ADOTTO_VCF	Truthset of variants (`HG002_GRCh38_TandemRepeats_v1.0.vcf.gz`)
ADOTTO_REGIONS	TR regions defined in GIABTR (`HG002_GRCh38_TandemRepeats_v1.0.bed.gz`)
CATALOG_REGIONS	Regions in the caller's catalog in BED format
GENOME_FASTA	Genome in FASTA format

ADOTTO_VCF is the VCF truthset (with SNPs) and ADOTTO_REGIONS are the TR regions in BED format. The ADOTTO_REGIONS are divided in two tiers, 1 and 2, Tier1 being the high confidence TR. I subset both ADOTTO_VCF and ADOTTO_REGIONS to the Tier1 regions, which are the "high confidence TRs" but this is not required.

Introduction

The calls need to be converted from VCF 4.4 format to SV format for Truvari to use. I wrote a script called `Truvarizer.py` that takes a VCF 4.4 (**where alleles have been split**) and outputs a minimal VCF in SV format compatible with Truvari. Since we don't make sequence calls for STRs, the script just inserts as many copies of the motif as needed in the REF and ALT fields and sets the correct SVLEN and SVTYPE.

The REF field sequence should not be that important because it is filled again with the reference sequence using `bcftools`.

Steps

The steps below can be chained, but it's useful to show the VCF records changes at each step. The starting VCF will have an entry in EH format:

```
chr1    1585949 .   A   <STR17>,<STR18> .   PASS
 ↪  END=1585976;REF=9;RL=27;RU=AAT;VARID=chr1_1585949_1585976;REPID=chr1_1585949_1585976
 ↪  GT:SO:REPCN:REPCI:ADSP:ADFL:ADIR:LC
 ↪  1/2:SPANNING/SPANNING:17/18:17-17/18-18:11/6:16/16:0/0:39.689562
```

The same record in VCF 4.4 format:

```
chr1    1585949 chr1_1585949_1585976    A   <CNV:TR>,<CNV:TR>   .   PASS
 ↪  SVTYPE=CNV;EVENTTYPE=STR;SVLEN=27,27;END=1585976;REFRUC=9.00;RUS=AAT;CN=1.89,2.00;
 ↪  RUC=17.00,18.00;RUCCHANGE=8.000000,9.000000;CNVTRLEN=24.000000,27.000000
 ↪  GT:CN   1/2:3.888889
```

The transformation script `EH VCF4.4` is named EHVCFConverter.py.

The VCF4.4 formatted calls are then transformed in SV calls that `truvari` can work with in the following steps.

```
bcftools filter -i 'FILTER="PASS"' VCF \ # <1>
  | bcftools norm -N -d any \ # <2>
  | bcftools norm -N -m-any -o 1_split_variants.vcf                    ③
```

① Keep only `PASS` variants and filter out REF calls
② Remove duplicates before splitting the alleles
③ Split alleles

```
chr1    1585949 chr1_1585949_1585976    A   <CNV:TR>    .   PASS
 ↪  SVTYPE=CNV;EVENTTYPE=STR;SVLEN=27;END=1585976;REFRUC=9;RUS=AAT;CN=1.89;
 ↪  RUC=17;RUCCHANGE=8;CNVTRLEN=24  GT:CN   1/0:3.88889
chr1    1585949 chr1_1585949_1585976    A   <CNV:TR>    .   PASS
 ↪  SVTYPE=CNV;SVLEN=27;END=1585976;REFRUC=9;RUS=AAT;CN=2;
 ↪  RUC=18;RUCCHANGE=9;CNVTRLEN=27   GT:CN   0/1:3.88889
```

> The `-N` option is key otherwise `bcftools` would try to normalise variants with unpredictable (and often wrong) results.

After the alleles have been split they can be transformed in SV format (see `Truvarizer.py`) by naively filling the REF and ALT fields with a made-sequence of repeats in base of the predicted haplotype length. After a few refinement steps that can be used as input for `truvari bench`.

```
python3 ../tools/EHconverters/Truvarizer.py -i 1_split_variants.vcf -o
 ↪  2_fill_ref_alt.vcf                                                    ①
bcftools sort 2_fill_ref_alt.vcf -o 3_sorted.vcf                          ②
bgzip -c 3_sorted.vcf > 3_sorted.vcf.gz
tabix -p vcf 3_sorted.vcf.gz
```

① Transform from VCF 4.4 (VNTR) format to SV format with artificial REF/ALT fields
② Sort

```
chr1    1585949 chr1_1585949_1585976    AAATAATAATAATAATAATAATAATAAT
 ↪  AAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAAT    .   PASS
 ↪  SVTYPE=INS;SVLEN=24;RUS=AAT GT:CN   1/0:3.88889
chr1    1585949 chr1_1585949_1585976    AAATAATAATAATAATAATAATAATAAT
 ↪  AAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAAT .   PASS
 ↪  SVTYPE=INS;SVLEN=27;RUS=AAT GT:CN   0/1:3.88889
```

```
bcftools +fill-from-fasta 3_sorted.vcf.gz -- -c REF -f $GENOME_FASTA >
 ↪  4_fix_ref.vcf                                                        ①
bgzip -c 4_fix_ref.vcf > 4_fix_ref.vcf.gz
tabix -p vcf 4_fix_ref.vcf.gz
```

① Fill the REF sequence (which is now just the repeated motif) with the reference sequence

```
chr1    1585949 chr1_1585949_1585976    AAATAATAATAATAATAATAATAATAAT
 ↪  AAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAAT    .   PASS
 ↪  SVTYPE=INS;SVLEN=24;RUS=AAT GT:CN   1/0:3.88889
chr1    1585949 chr1_1585949_1585976    AAATAATAATAATAATAATAATAATAAT
 ↪  AAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAATAAT .   PASS
 ↪  SVTYPE=INS;SVLEN=27;RUS=AAT GT:CN   0/1:3.88889
```

In this case the REF sequence is identical but it can be different in some cases. The REF sequece has to be identical to the reference sequence passed to `truvari refine`, otherwise it will error out. The ALT sequence is unchanged by this, it just allows the haplotype extraction with `bcftools` during `refine` to work without errors.

```
mkdir truvari_bench
VCF=4_fix_ref.vcf.gz
truvari bench -b $ADOTTO_VCF \
  -c $VCF \
  --includebed $ADOTTO_REGIONS \
  --sizemin 5 \ # <1>
  --sizefilt 5 \ # <1>
  --pick ac \ # <2>
  -o truvari_bench/
```

① the minimum size of a base (truth) call to be considered.
② the minimum size of a comp (input) call to be considered (0 too to disable).

```
truvari refine \
  -t 48 --use-original-vcfs \
  --reference $GENOME_FASTA \
  --align mafft \
  --regions $CATALOG_REGIONS \ # <1>
  truvari_bench/
```

① optional, only to be used if the benchmark has to be restricted to the catalog regions

The results after the refine step are in the `region_summary.json`. They are aggregated per region since the phab MSA workflow recalls variants which makes the variant-level numbers difficult to interpret. The variant level numbers are still present in `variant_summary.json` but those would be "new" variant that are called after the MSA alignemnt of haplotypes and reference during the `refine` step.

Output: `region_summary.json`


```
{
    "TP": 12384,
    "TN": 31755,
    "FP": 295,
    "FN": 729,
    "base P": 13166,
    "base N": 31930,
    "comp P": 12876,
    "comp N": 32220,
```

```
    "PPV": 0.961789375582479,                                    ①
    "TPR": 0.9406045875740544,                                   ②
    "TNR": 0.994519260883182,                                    ③
    "NPV": 0.9855679702048417,
    "ACC": 0.9787786056412986,
    "BA": 0.9675619242286182,
    "F1": 0.9510790261884648,
    "UND": 5
}
```


① Positive Predictive Value (a.k.a. precision) = TP / comp P
② True Positive Rate (a.k.a. recall) = TP / base P
③ True Negative Rate (a.k.a. specificity) = TN / base N