nanoraw Documentation

Release 0.5

Marcus Stoiber

Dec 20, 2018

nanoraw Documentation

nanoraw is a python package (with command line interface) used to analyze nanopore data. This tool primarily allows for the identification of modified DNA bases by comparing native and amplified DNA samples that have been processed by nanopore sequencing. See the *nanoraw* manuscript or more details on the proof of principle for the use of nanoraw to identify modified DNA bases (Pre-print manuscript).

CHAPTER 1

Installation

Official *nanoraw* releases are available through the python package index (nanoraw package site)

Install the latest official release with pip: *pip install nanoraw*

Install the most recent development version with: *pip install git+https://github.com/marcus1487/nanoraw.git*

nanoraw commands are then available on the command line. For all options run *nanoraw -h*. Note that the python package *rpy2* is required for any plotting functions (along the the R package ggplot2).

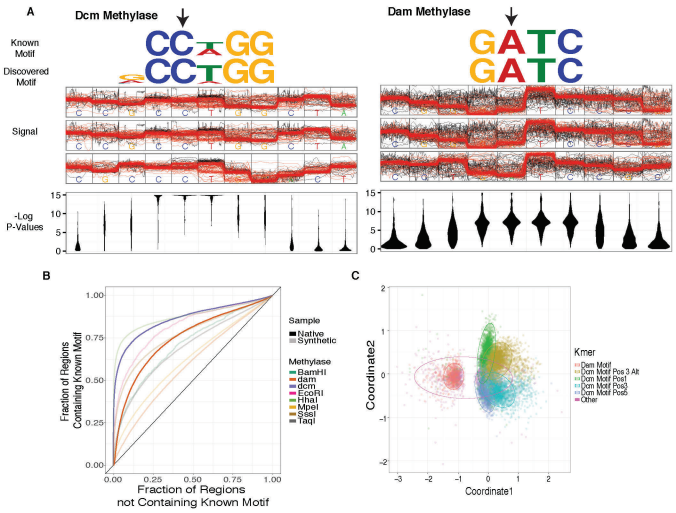

Fig. 1: Detection of in vivo methylated bases, as well as detect capacity and clustering based on raw signal levels.

nanoraw allows for the visualization of raw nanopore signal (below) facilitated by resolving raw signal with a genomic alignment from error-prone basecalls.

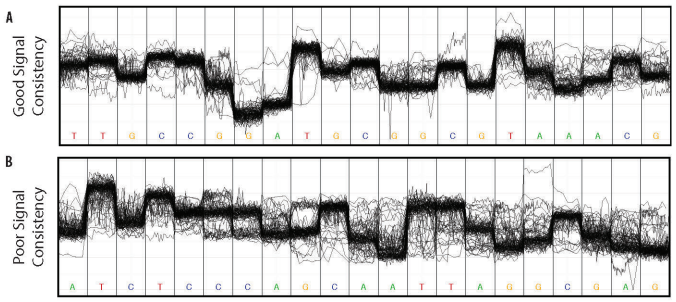

Fig. 2: Example loci with good and poor signal correspondence.

The above plots as well as many others are described in the documentation here.

---

nanoraw produces genome wiggle files for a number of attributes related to nanopore data including significance p/q-values, mean signal levels, and many more.

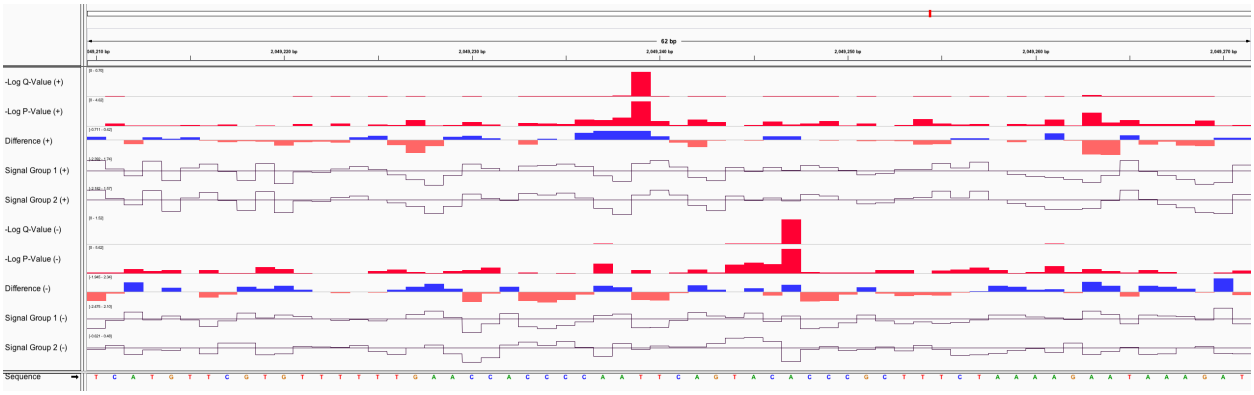

Fig. 3: Example genome browser region with WIG files produced from *nanoraw*

Fig. 4: Example gif scrolling over genomic regions (note that this is not immediately produced by nanoraw, but does not take much more effort).

1.1 Genome Re-squiggle

1.1.1 *genome_resquiggle*

This command re-annotates raw signal to match continguous genomic bases with contiguous sections ("Events") of raw signal. The algorithm begins with a genomic alignment (currently graphmap or bwa-mem) produced by the *genome_resquiggle* command. The full algorithm is described in detail within the manuscript describing nanoraw (Pre-print manuscript).

Required positional arguments for this command are (in order) *fast5_basedir* which specifies a directory containing FAST5 files to process and *genome_fasta* specifying the FASTA file to which mapping should be computed. Additionally, either a bwa or graphmap executable is required (e.g. *–graphmap-executable ./graphmap* or *–bwa-mem-executable ./bwa* assuming the binary is linked to the current directory).

1.1.2 nanoraw FAST5 format

The result of this command is a new group/slot present in each applicable FAST5 file. This slot is specified by the *–corrected-group* option. This slot contains several bits of information (*nanoraw_version* and *basecall_group* specified with the *–basecall-group* and also representing the ONT basecalled slot) along with slots for each 1D genome-resolved read (specified with the *–basecall-subgroups* options). A shortcut for the processing of standard 2D reads is available via the *–2d* option, which sets the *–basecall-subgroups* option appropriately.

Within each basecall subgroup slot the genome-resolved read information is stored. This includes several bits of information related to the normalization used to scale the raw signal: - shift, scale: Applied directly to the raw signal from this read in order to prodcue the normalized signal - lower_lim, upper_lim: Used to apply winsorization to the raw signal - norm_type: The type of normalization applied to the raw signal - outlier_threshold: Specified outlier threshold when this read was corrected

Also stored within each basecall subgroup are datasets containing the alignment details and genome resolved events. - Alignment: This dataset contains the *genome_alignment* and *read_alignment* which define the base-by-base genomic alignement used to start the correction process as well as the original positions of the segments (starting at the beginning of the mapped read) in the *read_segments* slot. - Events: This dataset is similar to the ONT Events dataset providing the following data values for each genomic base: *norm_mean*, *norm_stdev*, *start*, *length*, *base*. The Events slot also contains the *read_start_rel_to_raw* bit of information indicating the 0-based index within the raw signal vector at which the mapped read starts.

The *Alignment* slot also contains several statistics from the alignment including genomic *mapped_start*, *mapped_strand*, and *mapped_chrom*, mapping statistics *num_insertions*, *num_deletions*, *num_matches*, and *num_mismatches* and finally the number of raw observations trimmed from the read by mapping *trimmed_obs_start* and *trimmed_obs_end* (soft clipped bases from mapping).

1.1.3 *genome_resquiggle* Options

Filtering Options

- *–timeout*: Timeout in seconds for the processing of a single read. This can be useful if one would like to skip reads that might take too much time to correct. Default: No timeout.
- *–cpts-limit*: Maximum number of changepoints to attempt to find within a single indel group. (Not setting this option can cause a process to stall and cannot be controlled by the timeout option). This option is also useful to filter and not correct reads that contain very error prone regions. Default: No limit.

Signal Normalization Options

- *–normalization-type*: Type of normalization to apply to raw signal when calculating statistics based on new segmentation. Should be one of {“median”, “ont”, “none”}. “none” will provde the raw integer as the raw signal is stored. “ont” will calculate the pA estimates as in the ONT events mean/sd. “median” will shift by the median of each reads’ raw signal and scale by the MAD. Default: median
- *–outlier-threshold*: Number of scales values (median:MADs; ont:SDs, none:SDs) at which to winsorize the raw signal. This can help avoid strong re-segmentation artifacts from spikes in signal. Set to negative value to disable outlier trimming. Default: 5

Input/Output Options

- *–fast5-pattern*: A pattern to search for a subset of files within fast5-basedir. Note that on the unix command line patterns may be expanded so it is best practice to quote patterns.
- *–overwrite*: Overwrite previous corrected group in FAST5/HDF5 file. WARINGING: Use caution when setting this option that the *–corrected-group* is not set to the group containing the original basecalling information. Note this only effects the group defined by *–corrected-group*.
- *–failed-reads-filename*: Output failed read filenames into a this file with assoicated error for each read. Default: Do not store failed reads. This can be useful in order to track down some failure modes.

Multiprocessing Options

- *–processes*: Specify number of processors to use for processing aligning and resquiggling reads. Note that the number of processors used will be as much as twice as many to process both alignment and resquiggling of reads in separate queues.
- *–align-processes*: Specify number of processors to dedicate to aligning reads.
- *–resquiggle-processes*: Specify number of processors to dedicate to resquiggling reads.

1.1.4 Correction Plotting

In order to interrogate and better understand the genome re-squiggle process two plotting commands are provided (*plot_correction* and *plot_multi_correction*).

plot_correction

This command shows the read correction process for a small region of a single read. In the upper panel are the original ONT (or potentially other) basecalls (along the top of the panel) and the genomic basecalls (along the bottom of the panel) with the raw (normalized) signal shown throughout the middle of the plot. Original basecalls that were inserted are shown in red and similarly genomic bases that were deleted in the original basecalls are shown in red. Mismatched basecalls are also shown in red (at the top of the panel. The lower panel shows the moveing difference used (by default) to compute the new breakpoints when needed near indels.

plot_multi_correction

This plot shows multiple reads in "sequencing time space" (x-axis) anchored at a single position (either chosen randomly or specified). This plot is useful for interogatting the raw sequencing signal at particular regions of interest. This plot can optionally include the original basecalls as well, but the plot can become quite cumbersome with lots of information.

1.1.5 Example commands

Re-squiggle command:

```
nanoraw genome_resquiggle \
      $g1Dir $genomeFn --graphmap-executable ./graphmap \
      --timeout 60 --cpts-limit 100 --normalization-type median \
      --failed-reads-filename testing.signif_group1.failed_read.txt \
      --2d --processes 4 --overwrite
```

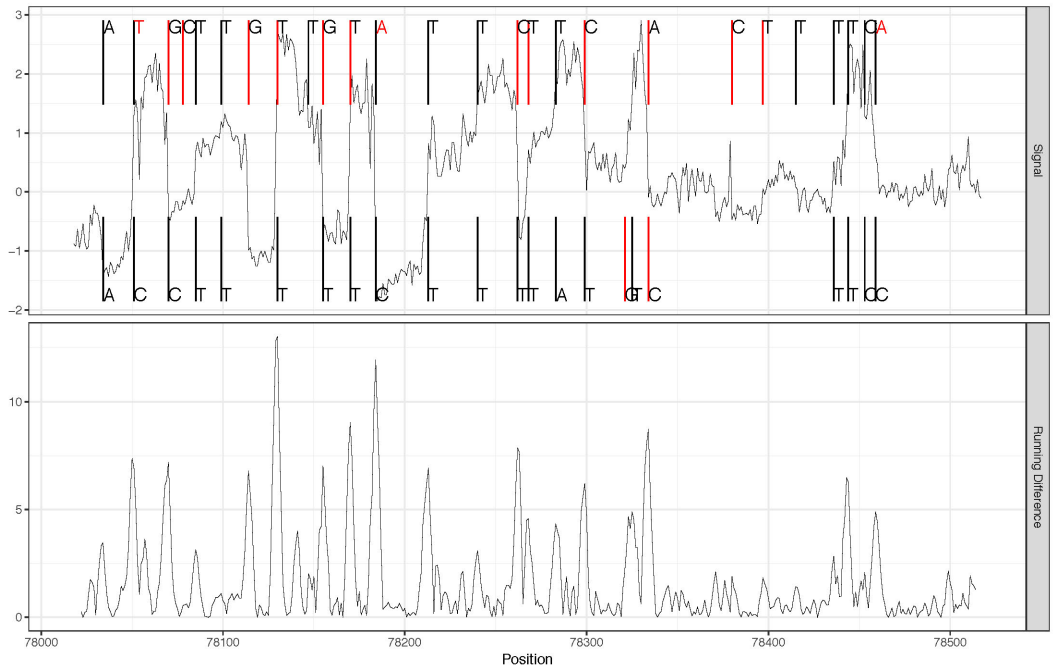

Fig. 5: Read correction example plot.

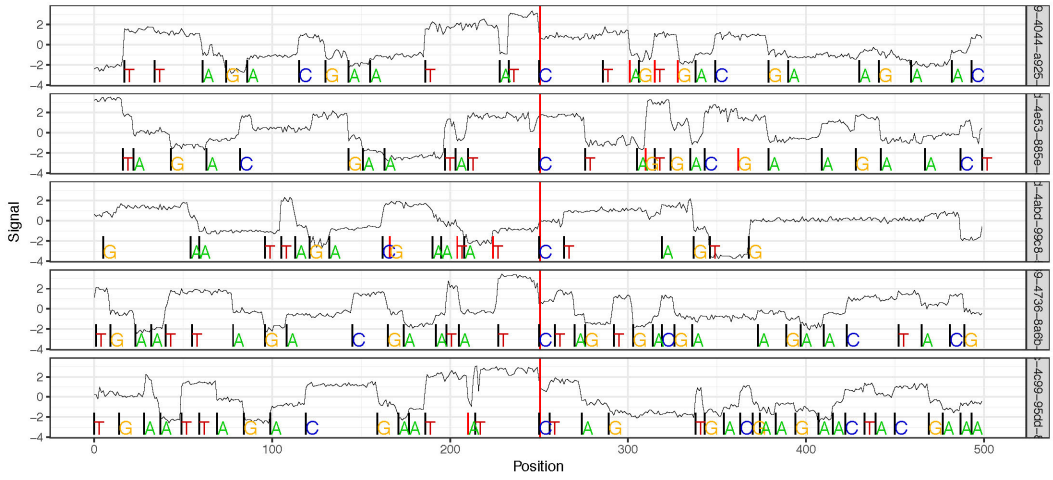

Fig. 6: Mutliple read correction example plot.

Re-squiggle command with bwa:

```
nanoraw genome_resquiggle \
      $g1Dir $genomeFn --bwa-mem-executable ./bwa \
      --timeout 60 --cpts-limit 100 --normalization-type median \
      --corrected-group RawGenomeCorrected_bwamem_000 --overwrite \
      --failed-reads-filename testing.group1.bwamem.failed_read.txt \
      --2d --processes 4
```

Re-squiggle command with pA events:

```
nanoraw genome_resquiggle \
      $g1Dir $genomeFn --graphmap-executable ./graphmap \
      --timeout 60 --cpts-limit 100 --normalization-type ont \
      --corrected-group RawGenomeCorrected_pA_000 --overwrite \
      --failed-reads-filename testing.signif_group1.pA.failed_read.txt \
      --2d --processes 4
```

Correction plotting examples:

```
nanoraw plot_correction --fast5-basedirs $g1Dir --region-type random
nanoraw plot_multi_correction --fast5-basedirs $g1Dir
nanoraw plot_multi_correction --fast5-basedirs $g1Dir \
      --genome-locations "S_aureus:2064835:-" "S_aureus:2064935"
```

1.2 Genome Anchored Plotting Commands

1.2.1 Single Sample Plotting

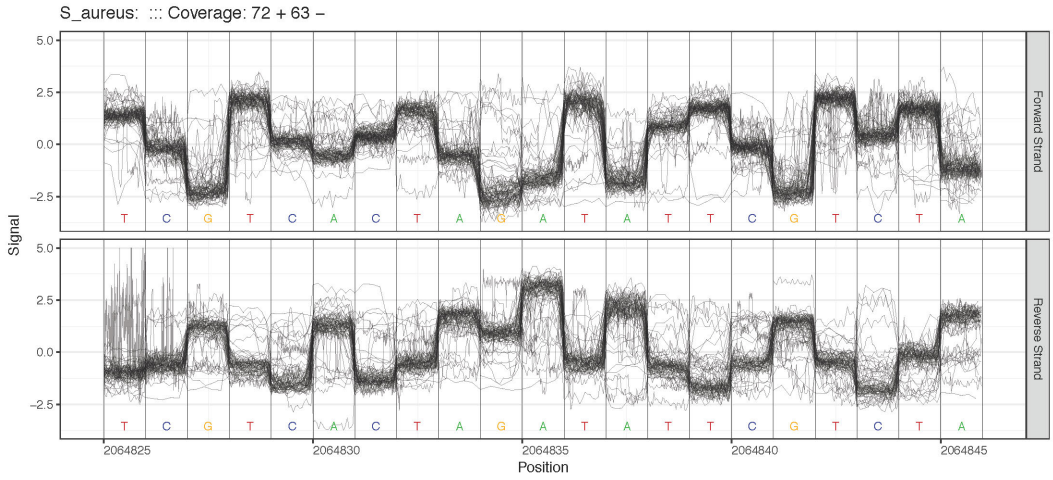

Fig. 7: Single sample plotting example.

Region Plotting Selection Criterion

- Maximum coverage (*plot_max_coverage*)
- Specified Genome Locations (*plot_genome_location*)

- Centered on motif of interest (*plot_motif_centered*)

1.2.2 Two Sample Plotting

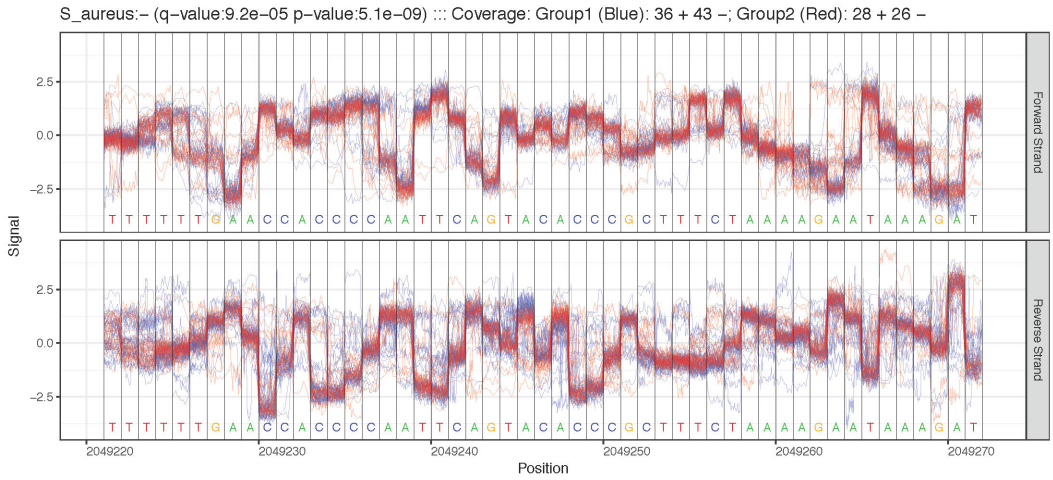

Fig. 8: Two sample plotting example.

Region Plotting Selection Criterion

All those available from single sample plotting plus:

- Maximum difference in mean signal level between two samples (*plot_max_difference*)
- Most significant statistical test between two samples (*plot_most_significant*)

1.2.3 Motif Centered Multi-location with Statistics Plotting

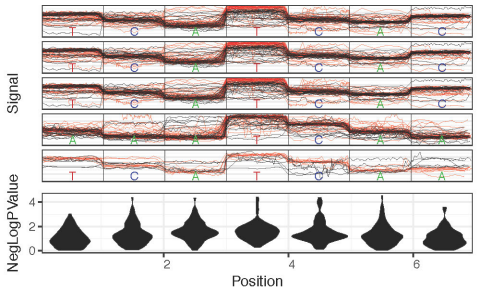

Fig. 9: Motif centered plotting with distribution of statistical test surrounding motif.

Plot statistics distrbution around motif of interest across many genomic locations along with most significant example genomic regions with *plot_motif_sith_stats*.

1.2.4 Common Options

- *–fast5-basedirs*: One or more directories containing FAST5 files that have been "re-squiggled" (*nanoraw genome_resquiggle*). This option is required for all genomic plotting commands.

- *–fast5-basedirs2*: One or more directories containing FAST5 files that have been "re-squiggled" (*nanoraw genome_resquiggle*). These will be group2. This option is required only for *plot_max_difference*, *plot_most_significant* and *plot_motif_with_stats*
- *–pdf-filename*: Filename to store plots from this command. (Default depends on the command)
- *–num-regions*: Number of difference regions to plot. Each region will be on another page of the output PDF and ordered by criterion (if applicable). This option is not valid for *plot_genome_location*.
- *–num-bases*: Number of genomic bases to include in a plot. Selection criterion will apply to the central base of a plotted region.
- *–obs-per-base-filter*: Filter reads for plotting baseed on threshold of percentiles (over all bases in a read) of the number of observations assigned to a base. Format thresholds as "percentile:thresh [pctl2:thresh2 . . .]" E.g. reads with 99th pctl <200 obs and max <5k obs would be "99:200 100:5000". Default is no filter.

1.2.5 Overplotting Options

- *–overplot-threshold*: Number of reads to trigger alternative plot type instead of raw signal due to overplotting. Default depends on command.
- *–overplot-type*: Plot type for regions with higher coverage. Choices: Downsample (default), Boxplot , Quantile, Violin. Examples below.

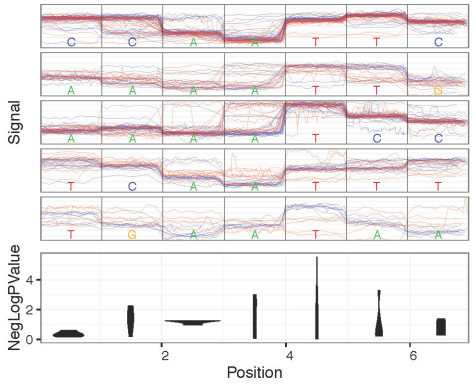

Fig. 10: Downsample Overplotting example

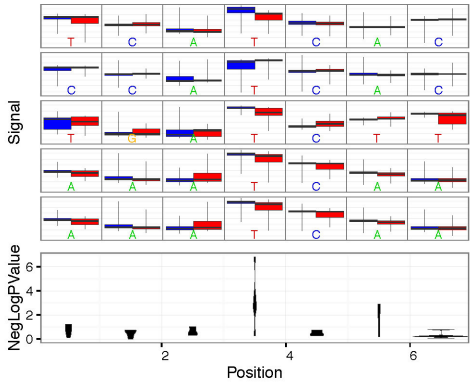

Fig. 11: Boxplot Overplotting example

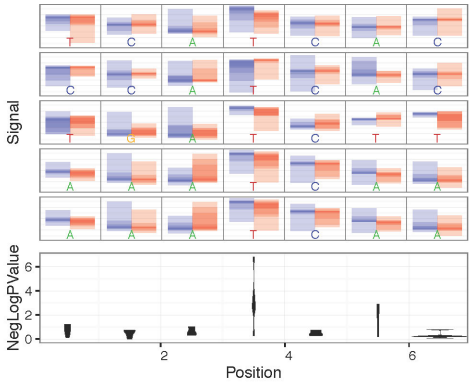

Fig. 12: Quantile Overplotting example

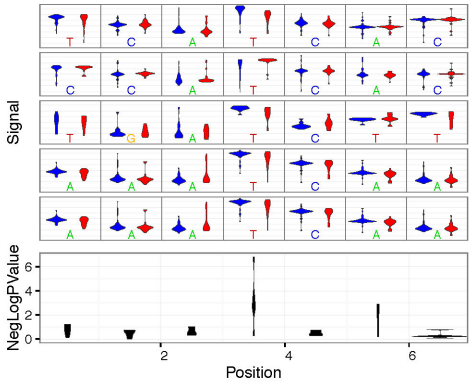

Fig. 13: Violin Overplotting example

1.2.6 Data Slot Options

- *–corrected-group*: FAST5 group to access/plot created by *genome_resquiggle* script. Default: RawGenomeCorrected_000. The default is the default slot used by the *genome_resquiggle* command so this command will not need to be set unless you would like to access an alternatively re-squiggled slot (e.g. including multiple signal normalizations within the same file).
- *–basecall-subgroups*: FAST5 subgroup (under Analyses/[corrected-group]) where individual template and/or complement reads are stored. Default: BaseCalled_template. This is the default supplied by ONT and should work for most cases.
- *–2d*: Input contains 2D reads and both forward and complement should be plotted. Equivalent to *–basecall-subgroups BaseCalled_template BaseCalled_complement*

1.2.7 Command Specific Options

plot_genome_location Option

- *–genome-locations*: Plot signal at specified genomic locations. Regions will be centered on the specified genomic position. Format locations as "chrm:position [chrm2:position2 ...]". E.g. "chr1:1000 chr21:40000 chrY:5000"

plot_motif_centered and *plot_motif_with_stats* Options

- *–motif*: DNA motif of interest. Can be composed of any one letter DNA codes (NEB Single Letter Codes).
- *–genome-fasta*: FASTA file used to map reads with *genome_resquiggle* command. If chromosomes are missing then regions from those chromosomes (or organims if multi-species) will not be considered for plotting.

plot_most_significant and *plot_motif_with_stats* Options

- *–test-type*: Type of significance test to apply. Choices are: mw_utest (default; mann-whitney u-test), ttest.
- *–fishers-method-offset*: Offset up and downstream over which to compute combined p-values using Fisher's method. For example 2 would compute the Fisher's method p-value over a moving window of 5 bases. Default: Do not compute Fihser's method p-values (report raw, base-by-base p-values).
- *–statistics-filename*: Filename to save/load base by base signal difference statistics. If file exists it will be loaded, if it does not exist it will be created to save statistics. Default: Don't save/load. Note that *–test-type* and *–fishers-method-offset* will be ignored if *–statistics-filename* is provided and the file exists.
- *–minimum-test-reads*: Number of reads required from both samples to test for significant difference in signal level. Note that regions with lower coverage levels will not have p-values be computed. Default: 5

plot_most_significant Options

- *–q-value-threshold*: Choose the number of regions to plot by the FDR corrected p-values. Note that *–num-regions* will be ignored if this option is set.
- *–sequences-filename*: Filename to store genomic sequences at selected regions (e.g. for PWM search). Sequences will be stored in FASTA format. Default: None.

plot_motif_with_stats Option

- *–num-context*: Number of bases to plot surrounding motif of interest. Default: 2

1.2.8 Example commands

Single sample genome-anchored plotting functions:

```
nanoraw plot_max_coverage --fast5-basedirs $g1Dir --2d \
  --num-bases 21 --overplot-threshold 1000
nanoraw plot_max_coverage --fast5-basedirs $g1Dir --2d \
  --num-bases 21 --overplot-threshold 1000 \
  --obs-per-base-filter 99:200 100:5000
nanoraw plot_genome_location --fast5-basedirs $g1Dir \
  --genome-locations "S_aureus:2064835" "S_aureus:2064935" \
  --2d --num-bases 21 --overplot-threshold 1000
nanoraw plot_motif_centered --fast5-basedirs $g1Dir --motif AHC \
  --genome-fasta $genomeFn --2d \
  --num-bases 21 --overplot-threshold 1000
nanoraw plot_motif_centered --fast5-basedirs $g1Dir --motif AHC \
  --genome-fasta $genomeFn --2d \
  --num-bases 21 --overplot-threshold 1000 --deepest-coverage
```

Mutliple sample genome-anchored plotting functions:

```
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 1000
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 1000 \
  --obs-per-base-filter 99:200 100:5000
nanoraw plot_genome_location --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir \
  --genome-locations "S_aureus:2064835" "S_aureus:2064935" \
  --2d --num-bases 21 --overplot-threshold 1000
nanoraw plot_motif_centered --fast5-basedirs $g1Dir --motif AHC \
  --genome-fasta $genomeFn \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 1000 --deepest-coverage
```

Mutliple sample statistical testing genome-anchored plotting functions:

```
nanoraw plot_max_difference --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 1000
nanoraw plot_most_significant --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 1000
nanoraw plot_motif_with_stats --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --motif AHC --2d \
  --overplot-threshold 1000 --test-type mw_utest \
  --genome-fasta $genomeFn
```

Overplotting options:

```
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 20 --overplot-type Downsample \
  --pdf-filename Nanopore_read_coverage.max_coverage.Downsample.pdf
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 20 --overplot-type Boxplot \
  --pdf-filename Nanopore_read_coverage.max_coverage.Boxplot.pdf
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 20 --overplot-type Quantile \
  --pdf-filename Nanopore_read_coverage.max_coverage.Quantile.pdf
nanoraw plot_max_coverage --fast5-basedirs $g1Dir \
  --fast5-basedirs2 $g2Dir --2d \
  --num-bases 21 --overplot-threshold 20 --overplot-type Violin \
  --pdf-filename Nanopore_read_coverage.max_coverage.Violin.pdf
```

1.3 Other Plotting

1.3.1 *plot_kmer*

Plot signal distribution acorss kmers. The command allows the selection of the number of bases up and downstream around the central genomic base.

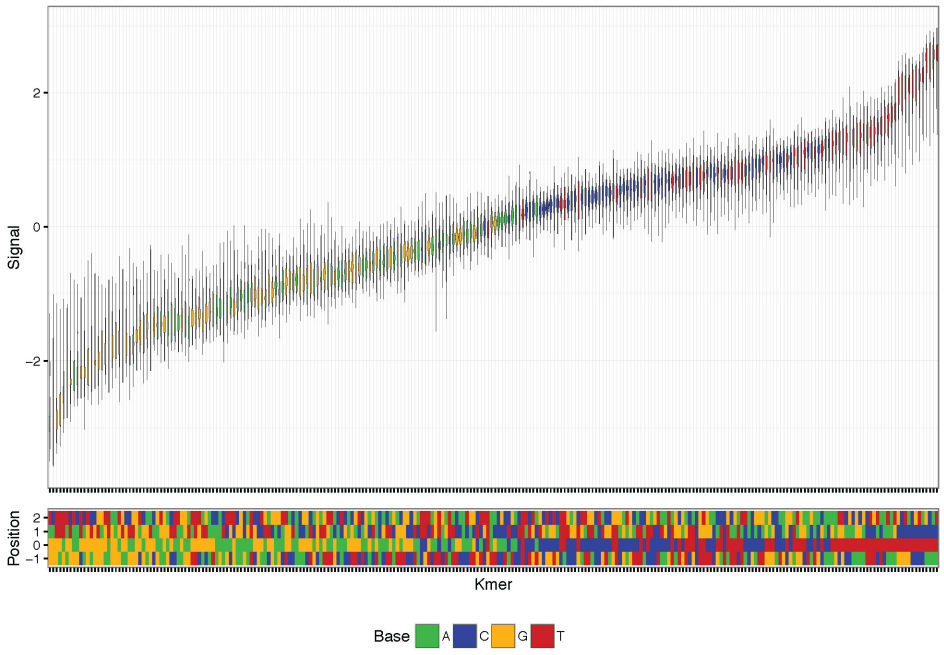

Fig. 14: K-mer signal distribution plotting example.

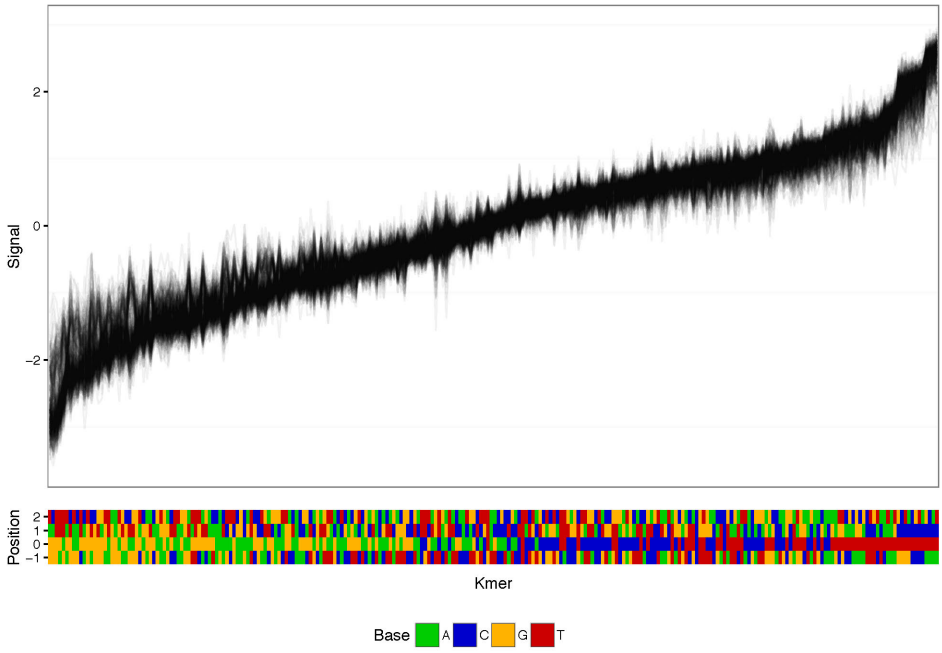

Fig. 15: K-mer signal distribution connecting each read example.

1.3.2 *cluster_most_significant*

Cluster raw signal traces at bases with significant differences. Clustering is done on the difference between two runs (generally a native and amplified sample) centered on the most significantly different bases. Note that this command can be quite slow for many points, so it is advised to only use few points for plotting (<1000).

1.3.3 Example commands

K-mer signal distribution plotting and clustering based on signal:

```
nanoraw plot_kmer --fast5-basedirs $g1Dir
nanoraw plot_kmer --fast5-basedirs $g1Dir --read-mean
nanoraw cluster_most_significant --fast5-basedirs $g1Dir \
      --fast5-basedirs2 $g2Dir --2d \
      --genome-fasta $genomeFn --num-regions 100 \
      --r-data-filename testing.cluster_data.RData \
      --statistics-filename testing.significance_values.txt
```

1.4 Auxiliary Commands

1.4.1 *write_most_significant_fasta*

Write sequences in FASTA format where signal differs the most significantly between two groups. This subcommand is suggested for running downstream motif discovery pipelines. Options are similar or equivalent to significance plotting commands.

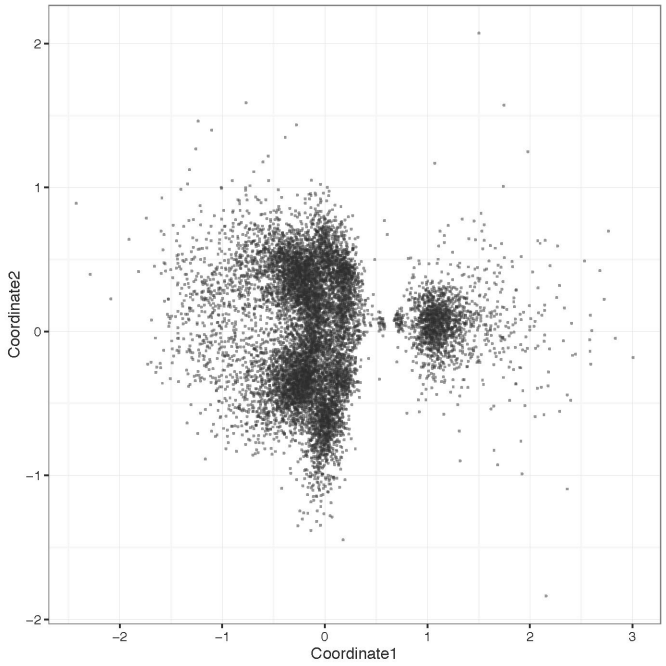

Fig. 16: Signal clustering plotting example.

1.4.2 *write_wiggles*

Write wiggle files for attributes from a group of FAST5 files. The currenly available attributes are *coverage*, *signal*, *signal_sd*, *length* (mean over all reads at each base for last three), and if two FAST5 directories or a statistics file are provided *pvals* (Negative log 10), *qvals* (Negative log 10), and *difference*.

write_wiggles Options

- *–wiggle-types*: Specify which wiggle files should be produced. Options are listed above. (Default: "coverage, pval")
- *–wiggle-basename*: Basename for output files. Two files (plus and minus strand) will be produced for each –wiggle-types supplied (four files will be produced if *–fast5-basedirs2* is supplied for coverage, signal, signal_sd, and length). Filenames will be formatted as "[wiggle-basename].[wiggle-type].[group1/2]?.[plus/minus].wig". Default: Nanopore_data

1.4.3 Example commands

Write FASTA and wiggles examples:

```
nanoraw write_most_significant_fasta --fast5-basedirs $g1Dir \
      --fast5-basedirs2 $g2Dir --2d \
      --genome-fasta $genomeFn
nanoraw write_wiggles --fast5-basedirs $g1Dir \
      --fast5-basedirs2 $g2Dir --2d \
      --wiggle-types coverage signal signal_sd length pvals \
      qvals difference \
      --statistics-filename testing.significance_values.txt
```