XYalign Documentation

Release 1.1.5

Tim Webster, Tanya Phung, Madeline Couse, Bruno Grande, Eric I

Oct 27, 2018

Contents:

Authors Timothy Webster, Madeline Couse, Bruno Grande, Eric Karlins, Tanya Phung, Phillip Richmond, Whitney Whitford, Melissa Wilson Sayres

Date Oct 27, 2018

Release 1.1.5

Download Github Repository

Forum/Mailing List Google Group

Issues Please submit any bugs, problems, or feature requests as issues via Github

The high degree of similarity between gametologous sequences on the sex chromosomes can lead to the misalignment of sequencing reads and substantially affect variant calling. Here we present XYalign, a new tool that (1) aids in the inference of sex chromosome content using NGS data, (2) remaps reads based on the inferred sex chromosome complement of the individual, and (3) outputs quality, depth, and allele-balance metrics across chromosomes.

CHAPTER 1

Installation

1.1 Operating System

XYalign has been tested on Linux and Mac operating systems, but has not been tested on Windows. This isn't to say it won't work, however we are unprepared to offer any Windows support at this time.

1.2 Requirements

XYalign has a number of required Python packages and external programs:

```
Python: 2.7

Python packages:
        matplotlib
        numpy
        pandas
        pybedtools
        pysam
        scipy

External Programs:
        bbmap (XYalign uses repair.sh and shuffle.sh from this suite of tools)
        bedtools
        bwa
        platypus
        sambamba
        samtools
```

Note: Bedtools is required for pybedtools and must be added to one's *PATH*. XYalign will check that it is available by calling *bedtools*. Other external programs do not, however, need to be on one's *PATH* and can be provided to XYalign using the appropriate flag(s):

```
--repairsh_path
--shufflesh_path
--bwa_path
--platypus_path
--samtools_path
--sambamba_path
```

1.3 Obtaining XYalign

We strongly recommend users install and manage all packages and programs using Anaconda. To do so:

1. First download and install either Anaconda or Miniconda (both work well, Miniconda is a lightweight version of Anaconda).

- Be sure to allow Anaconda to append to your PATH (it will ask for permission to do so during installation)
  - You can check this after installation with the command (from the command line):

    ```
    which python
    ```

    which should point you to the python installed in your Anaconda or Miniconda directory.

2. Linux and Mac users can finish installation with the following commands (note that \ indicates a continuation of the command on the next line):

```
conda config --add channels r

conda config --add channels defaults

conda config --add channels conda-forge

conda config --add channels bioconda

conda create -n xyalign_env xyalign
```

This assumes you're installing into a new environment called "xyalign_env".

Note: You *need* to add channels in this order. Doing so will ensure priority of channels will go in the order bioconda > conda-forge > defaults > r. This is important because the source of bzip2 (required for many programs) needs to be conda-forge (the version in defaults will cause many programs to miss a required library).

You can then load the new environment (containing all required programs and packages) with:

```
source activate xyalign_env
```

To use Bioconda to simply install XYalign into your current environment, load the channels in using the commands listed above and then type:

```
conda install xyalign
```

In all cases, this will install XYalign, its dependancies, and all external programs that it calls.

1.4 Pip

XYalign can also be installed using `pip`, a tool used for installing Python packages, with the command:

```
pip install xyalign
```

However, note that this will not install any external programs that XYalign calls on for its various functions.

CHAPTER 2

Usage Overview

2.1 The Basics

2.1.1 Requirements

The different modules of XYalign have slightly different requirements, but in general you'll need: a **bam file** and the **reference fasta file** used to generate it (it's critically important, as using a different fasta will cause errors). XYalign also requires a *list of chromosomes* to analyze, the *name of the X chromosome*, and the *name of the Y chromosome* (if in the assembly). The chromosome names must *exactly* match those in the bam header and reference fasta - 'chr19' is not equivalent to '19', for example.

You also need a variety of python packages and external programs installed. See *Installation* for more information.

2.1.2 The Pipeline

Xyalign is composed of the following modules that can be thought of as steps in the pipeline (with the exception of CHROM_STATS):

```
PREPARE_REFERENCE
ANALYZE_BAM
CHARACTERIZE_SEX_CHROMS
STRIP_READS
REMAPPING
CHROM_STATS
```

Each of these modules can be invoked as a command line flag with no arguments (e.g., `--PREPARE_REFERENCE`), and XYalign will execute *only that module*. If no flags are provided, XYalign will run the full pipeline in the following order: PREPARE_REFERENCE -> ANALYZE_BAM -> CHARACTERIZE_SEX_CHROMS -> STRIP_READS -> REMAPPING -> ANALYZE_BAM. This will:

1. Prepare two reference genomes - one with the Y chromosome masked, the other with both X and Y unmasked. In both cases, XYalign will optionally mask other regions of the genome provided in an input bed file (using the flag `--reference_mask <file1.bed> <file2.bed> ...`).

2. Analyze the bam file to calculate metrics related to read balance, read depth, and mapping quality. Read depth and mapping quality are calculated in windows, and either `--window_size <integer window size>` or `--target_bed <path to target bed file>` must be provided. `--window_size` is the fixed size of windows to use in a nonoverlapping sliding window analysis in bases (e.g., `10000` for 10 kb windows). `--target_bed` is a bed file of targets to use as windows, e.g. exome capture targets.

3. Plot read balance, depth, and mapq for each chromosome, and output bed files of high and low quality regions, based on either default or user-defined thresholds.

4. Run a series of tests comparing ANALYZE_BAM metrics for each chromosome. If the flag `--CHARACTERIZE_SEX_CHROMS` is invoked, XYalign will carry out the bam analysis steps above and then proceed to these tests.

5. Strip and sort reads mapping to the sex chromosomes, map to the reference with the appropriate masking (step 1) based on the results of step 4, and replace the sex chromosome alignments in the original bam file with these new ones.

6. Analyze the new bam file as in steps 4 and 5.

`CHROM_STATS` provides quicker, coarser statistics and is designed for cases in which a reference genome is well-understood and when multiple samples are available.

2.1.3 Suggested Command Lines

Below we highlight example command lines, as well as useful optional flags for each module (PREPARE_REFERENCE, ANALYZE_BAM, CHARACTERIZE_SEX_CHROMS, STRIP_READS, REMAPPING, CHROM_STATS) as well as the full pipeline. You can find a complete list of command line flags, their descriptions, and their defaults from the command line:

```
xyalign -h
```

In all examples, `reference.fasta` is our input reference in fasta format, `input.bam` is our input bam file (created using `reference.fasta`), `sample1` is the ID of our sample, and `sample1_output` is the name of our desired output directory. We'll analyze chromosomes named 'chr19', 'chrX', and 'chrY', with chrX representing the X chromosome and chrY representing the Y chromosome. We'll assume that all programs are in our `PATH` and can be invoked by typing the program name from the command line without any associated path (e.g., `samtools`). We'll also assume that we're working on a cluster with 4 cores available to XYalign.

1. PREPARE_REFERENCE

```
xyalign --PREPARE_REFERENCE --ref reference.fasta \
--output_dir sample1_output --sample_id sample1 --cpus 4 --reference_mask mask.bed \
--x_chromosome chrX --y_chromosome chrY
```

Here, `mask.bed` is a bed file containing regions to mask in *both* output reference genomes (e.g., coordinates for the pseudoautosomal regions on the Y chromosome). More than one can be included as well (e.g., `--reference_mask mask.bed mask2.bed`).

This will output two reference genomes, one with the Y chromosome completely masked (defaults to `sample1_output/reference/xyalign_noY.masked.fa`) and one with an unmasked Y (defaults to `sample1_output/reference/xyalign_withY.masked.fa`). These default *names* can be changed with the `--xx_ref_out_name` and `--xy_ref_out_name` flags. With these flags, files will still be deposited in `sample1_output/reference`. To deposit these files in a specific location, use `--xx_ref_out` and `--xy_ref_out` with the full path to and name of desired output files. You can optionally use BWA to index the output fasta files as well by using the "–bwa_index" flag.

2. ANALYZE_BAM

```
xyalign --ANALYZE_BAM --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 --window_size 10000 \
--chromosomes chr19 chrX chrY --x_chromosome chrX --y_chromosome chrY
```

Here, 10000 is the fixed window size to use in (nonoverlapping) sliding window analyses of the bam file. If you're working with targeted sequencing data (e.g. exome), you can provide a list of regions to use instead of windows. For example, if your regions are in `targets.bed` you would add the flag: `--targed_bed targets.bed`.

This command line will default to a minimum quality of 30 (SNP), genotype quality of 30 (SNP), variant depth of 4 (SNP), and mapping quality of 20 (bam window). These can be set with the flags `--variant_site_quality`, `--variant_genotype_quality`, `--variant_depth`, and `--mapq_cutoff`, respectively. One can also apply depth filters to bam windows with `--min_depth_filter` and `--max_depth_filter`.

This will output a series of plots in `sample1_output/plots`, bed files containing high and low quality windows in `sample1_output/bed`, and the entire dataframe with values for each measure in each window in `sample1_output/bed`.

3. CHARACTERIZE_SEX_CHROMS

```
xyalign --CHARACTERIZE_SEX_CHROMS --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 --window_size 10000 \
--chromosomes chr19 chrX chrY --x_chromosome chrX --y_chromosome chrY
```

Settings here are identical to 3 because the first step of CHARACTERIZE_SEX_CHROMS involves running ANALYZE_BAM.

In addition to everything in ANALYZE_BAM, CHARACTERIZE_SEX_CHROMS will output the results of a series of statistical tests in `sample1_output/results`.

4. STRIP_READS

```
xyalign --STRIP_READS --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 \
--chromosomes chr1 chr2 chr3 chr4 chr5 --xmx 2g \
--fastq_compression 5
```

This will strip the reads, by read group, from chromosomes 1-5 and output a pair of fastqs per read group, as well as the read groups themselves, and a text file connecting fastqs with their respective read groups in the directory '' sample1_output/fastq''. If we were working with single-end reads, we would have had to include the flag '' –single_end . Here, the reference file isn't used at all (it's a general requirement of XYalign), so a dummy file can be used in its place. To strip reads from the entire genome (including unmapped), use `` --chromosomes ALL. `--xmx` tells the Java programs that XYalign is calling how much memory to use (e.g., `--xmx 2g` provides 2 GB ram). `--fastq_compression` determines the compression level of output fastqs (between 0 and 9, with 0 leaving files uncompressed).

5. REMAPPING

```
xyalign --REMAPPING --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 \
--chromosomes chr19 chrX chrY --x_chromosome chrX --y_chromosome chrY \
--xx_ref_in sample1_output/reference/xyalign_noY.masked.fa \
--xy_ref_in sample1_output/reference/xyalign_withY.masked.fa \
--y_absent
```

Here, we've input our reference genomes generated in step 1 (if we don't, XYalign will repeat that step). We've also used the flag `--y_absent` to indicate that there is no Y chromosome in our sample (perhaps as the result of step 3, or outside knowledge). If a Y is present, we would have used `--y_present` instead. REMAPPING requires

one of those two flags, as it does not involve any steps to estimate sex chromosome content (those are carried out in CHARACTERIZE_SEX_CHROMS). Note that REMAPPING will run STRIP_READS first.

5. Full pipeline

And if we want to run the full XYalign pipeline on a sample, we'd use a command line along the lines of:

```
xyalign --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 --reference_mask mask.bed \
--window_size 10000 --chromosomes chr19 chrX chrY \
--x_chromosome chrX --y_chromosome chrY
```

We could have optionally provided preprocessed reference genomes with `--xx_ref_in` and `--xx_ref_in`, as in 4. We could have also used `--y_absent` or `--y_present` to force mapping to a certain reference. Because we didn't include either of these two flags, XYalign will use `--sex_chrom_calling_threshold` to determine the sex chromosome complement (default is 2.0).

6. CHROM_STATS

```
xyalign --CHROM_STATS --use_counts --bam input1.bam input2.bam input3.bam --ref null \
--output_dir directory_name --sample_id analysis_name --chromosomes chr19 chrX chrY
```

Here, `--use_counts` simply grabs the number of reads mapped to each chromosome from the bam index. It's by far the fastest, yet coarsest option. Running without this flag will calculated depth and mapq along each chromosome for more detail, but this will take longer.

2.2 Recommendations for Incorporating XYalign into Pipelines

While the full XYalign pipeline will be useful in certain situations, we feel that the following pipeline is better suited to most users' needs and will save time and space.

1. Use XYalign PREPARE_REFERENCE to prepare Y present and Y absent genomes.

2. Preliminarily map reads to the standard reference (or Y present) and sort the bam file using any mapper and sorting algorithm. We have found that one can usually use smaller dataset for this step (e.g., a whole exome sequencing run or one lane of a whole genome sequencing run).

3. Run CHARACTERIZE_SEX_CHROMS, to analyze the bam file, output plots, and estimate ploidy. If a number of samples are available and sex chromosomes are well-differentiated (as in humans), consider using CHROM_STATS with plot_count_stats.

4. Remap reads to the fasta produced in 1 corresponding to the sex chromosome complement characterized in 3. E.g., if Y is not detected, map to Y absent. This time run full pipeline of mapping, sorting, removing duplicates, etc., using users' preferred tools/pipeline.

5. Optionally run ANALYZE_BAM on bam file produced in 4.

6. Call variants using user-preferred caller.

7. Analyze variants taking into account ploidy estimated in 3, and consider masking low quality regions using bed files output in 5.

2.3 XYalign - Speed and Memory

The minimum memory requirements for XYalign are determined by external programs, rather than any internal code. Right now, the major limiting step is bwa indexing of reference genomes which requires 5-6 GB of memory to index

a human-sized genome. In addition, in certain situations (e.g., removing all reads from deep coverage genome data with a single - or no - read group) the STRIP_READS module will require a great deal of memory to sort and match paired reads (the memory requirement is that of the external program repair.sh).

The slowest parts of the pipeline also all involve steps relying on external programs, such as genome preparation, variant calling, read mapping, swapping sex chromosome alignments, etc. In almost all cases, you'll see substantial increases in the speed of the pipeline by increasing the number of threads/cores. You must provide information about the number of cores available to XYalign with the `--cpus` flag (XYalign will assume only a single thread is available unless this flag is set).

2.4 Exome data

XYalign handles exome data, with a few minor considerations. In particular, either setting `--window_size` to a smaller value, perhaps 5000 or less, or inputting targets instead of a window size (`--target_bed targets.bed`) will be critical for getting more accurate window measures. In addition, users should manually check the results of CHARACTERIZE_SEX_CHROMS for a number of samples to get a feel for expected values on the sex chromosomes, as these values are likely to vary among experimental design (especially among different capture kits).

2.5 Nonhuman genomes

XYalign will theoretically work with any genome, and on any combination of chromosomes or scaffolds (see more on the latter below). Simply provide the names of the chromosomes/scaffolds to analyze and the names of the sex chromosomes (e.g., `--chromosomes chr1a chr1b chr2 lga lgb --x_chromosome lga --y_chromosome lgb` if our x_linked scaffold was lga and y_linked scaffold was lgb, and we wanted to compare these scaffolds to chromosomes: chr1a chr1b and chr2). However, please note that, as of right now, XYalign does not support multiple X or Y chromosomes/scaffolds (we are planning on supporting this soon though).

Keep in mind, however, that read balance, mapq, and depth ratios might differ among organisms, so default XYalign settings will likely not be appropriate in most cases. Instead, if multiple samples are available, we recommend running XYalign's CHARACTERIZE_SEX_CHROMS on each sample (steps 2-3 in "Recommendations for Incorporating XYalign into pipelines" above) using the same output directory for all samples. One can then quickly concatenate results (we recommend starting with bootstrap results) and plot them to look for clustering of samples (see the XYalign publication for examples of this).

2.6 Analyzing arbitrary chromosomes

Currently, XYalign requires a minimum of two chromosomes (an "autosome and an "x chromosome") for analyses in ANALYZE_BAM and CHARACTERIZE_SEX_CHROMS (and therefore, the whole pipeline) These chromosomes, however, can be arbitrary. Below, we highlight two example cases: looking for evidence of Trisomy 21 in human samples, and running the full XYalign pipeline on a ZW sample (perhaps a bird, squamate reptile, or moth).

If one wanted to look for evidence of Trisomy 21 in human data mapped to hg19 (which uses "chr" in chromosome names), s/he could use a command along the lines of:

```
xyalign --CHARACTERIZE_SEX_CHROMS --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 --window_size 10000 \
--chromosomes chr1 chr10 chr19 chr21 --x_chromosome chr21
```

This would run the CHARACTERIZE_SEX_CHROMS module, systematically comparing `chr21` to `chr1`, `chr10`, and `chr19`.

To run the full pipeline on a ZW sample (in ZZ/ZW systems, males are ZZ and females are ZW), one could simply run a command like (assuming the Z scaffold was named "scaffoldz" and the W scaffold was named "scaffoldw"):

```
xyalign --ref reference.fasta --bam input.bam \
--output_dir sample1_output --sample_id sample1 --cpus 4 --reference_mask mask.bed \
--window_size 10000 --chromosomes scaffold1 scaffoldz scaffoldw --x_chromosome␣
↪scaffoldz \
--y_chromosome scaffoldw
```

In this example, it's important that the the "X" and "Y" chromosomes are assigned in this way because PREPARE_REFERENCE (the first step in the full pipeline) will create two reference genomes: one with the "Y" completely masked, and one with both "X" and "Y" unmasked. This command will therefore create the appropriate references (a ZW and a Z only). Other organisms or uses might not require this consideration.

2.7 Using XYalign as a Python library

All modules in the XYalign/xyalign directory are designed to support the command line program XYalign. However, some classes and functions might be of use in other circumstances. If you've installed XYalign as described in *Installation*, then you should be able to import XYalign libraries just like you would for any other Python package. E.g.:

```
from xyalign import bam
```

Or:

```
import xyalign.bam
```

2.8 Full List of Command-Line Flags

This list can also be produced with the command:: xyalign -h

Flags:

```
-h, --help            show this help message and exit
--bam [BAM [BAM ...]]
                                        Full path to input bam files. If more than␣
↪one
                                        provided, only the first will be used for␣
↪modules
                                        other than --CHROM_STATS
--cram [CRAM [CRAM ...]]
                                        Full path to input cram files. If more than␣
↪one
                                        provided, only the first will be used for␣
↪modules
                                        other than --CHROM_STATS. Not currently␣
↪supported.
--sam [SAM [SAM ...]]
                                        Full path to input sam files. If more than␣
↪one
                                        provided, only the first will be used for␣
↪modules
                                        other than --CHROM_STATS. Not currently␣
↪supported.
```

```
--ref REF             REQUIRED. Path to reference sequence (including file
                      name).
--output_dir OUTPUT_DIR, -o OUTPUT_DIR
                      REQUIRED. Output directory. XYalign will create a
                      directory structure within this directory
--chromosomes [CHROMOSOMES [CHROMOSOMES ...]], -c [CHROMOSOMES [CHROMOSOMES ...]]
                      Chromosomes to analyze (names must match reference
                      exactly). For humans, we recommend at least chr19,
                      chrX, chrY. Generally, we suggest including the sex
                      chromosomes and at least one autosome. To analyze all
                      chromosomes use '--chromosomes ALL' or '--chromosomes
                      all'.
--x_chromosome [X_CHROMOSOME [X_CHROMOSOME ...]], -x [X_CHROMOSOME [X_CHROMOSOME ...]]
                      Names of x-linked scaffolds in reference fasta (must
                      match reference exactly).
--y_chromosome [Y_CHROMOSOME [Y_CHROMOSOME ...]], -y [Y_CHROMOSOME [Y_CHROMOSOME ...]]
                      Names of y-linked scaffolds in reference fasta (must
                      match reference exactly). Defaults to chrY. Give None
                      if using an assembly without a Y chromosome
--sample_id SAMPLE_ID, -id SAMPLE_ID
                      Name/ID of sample - for use in plot titles and file
                      naming. Default is sample
--cpus CPUS           Number of cores/threads to use. Default is 1
--xmx XMX             Memory to be provided to java programs via -Xmx. E.g.,
                      use the flag '--xmx 4g' to pass '-Xmx4g' as a flag
                      when running java programs (currently just repair.sh).
                      Default is 'None' (i.e., nothing provided on the
                      command line), which will allow repair.sh to
                      automatically allocate memory. Note that if you're
                      using --STRIP_READS on deep coverage whole genome
                      data, you might need quite a bit of memory, e.g. '--
                      xmx 16g', '--xmx 32g', or more depending on how many
                      reads are present per read group.
--fastq_compression {0,1,2,3,4,5,6,7,8,9}
                      Compression level for fastqs output from repair.sh.
                      Between (inclusive) 0 and 9. Default is 3. 1 through 9
                      indicate compression levels. If 0, fastqs will be
                      uncompressed.
```

```
--single_end          Include flag if reads are single-end and NOT paired-
                                    end.
--version, -V         Print version and exit.
--no_cleanup          Include flag to preserve temporary files.
--PREPARE_REFERENCE   This flag will limit XYalign to only preparing
                                    reference fastas for individuals with and␣
↪without Y
                                    chromosomes. These fastas can then be␣
↪passed with each
                                    sample to save subsequent processing time.
--CHROM_STATS         This flag will limit XYalign to only analyzing
                                    provided bam files for depth and mapq␣
↪across entire
                                    chromosomes.
--ANALYZE_BAM         This flag will limit XYalign to only analyzing the bam
                                    file for depth, mapq, and (optionally) read␣
↪balance
                                    and outputting plots.
--CHARACTERIZE_SEX_CHROMS
                                    This flag will limit XYalign to the steps␣
↪required to
                                    characterize sex chromosome content (i.e.,␣
↪analyzing
                                    the bam for depth, mapq, and read balance␣
↪and running
                                    statistical tests to help infer ploidy)
--REMAPPING           This flag will limit XYalign to only the steps
                                    required to strip reads and remap to masked
                                    references. If masked references are not␣
↪provided,
                                    they will be created.
--STRIP_READS         This flag will limit XYalign to only the steps
                                    required to strip reads from a provided bam␣
↪file.
--logfile LOGFILE     Name of logfile. Will overwrite if exists. Default is
                                    sample_xyalign.log
--reporting_level {DEBUG,INFO,ERROR,CRITICAL}
                                    Set level of messages printed to console.␣
↪Default is
                                    'INFO'. Choose from (in decreasing amount of
                                    reporting) DEBUG, INFO, ERROR or CRITICAL
--platypus_path PLATYPUS_PATH
                                    Path to platypus. Default is 'platypus'. If␣
↪platypus
                                    is not directly callable (e.g., '/path/to/
↪platypus' or
                                    '/path/to/Playpus.py'), then provide path␣
↪to python as
                                    well (e.g., '/path/to/python /path/to/
↪platypus'). In
                                    addition, be sure provided python is␣
↪version 2. See
                                    the documentation for more information␣
↪about setting
                                    up an anaconda environment.
--bwa_path BWA_PATH   Path to bwa. Default is 'bwa'
--samtools_path SAMTOOLS_PATH
                                    Path to samtools. Default is 'samtools'
```

```
--repairsh_path REPAIRSH_PATH
                                        Path to bbmap's repair.sh script. Default is
                                        'repair.sh'
--shufflesh_path SHUFFLESH_PATH
                                        Path to bbmap's shuffle.sh script. Default␣
↪is
                                        'shuffle.sh'
--sambamba_path SAMBAMBA_PATH
                                        Path to sambamba. Default is 'sambamba'
--bedtools_path BEDTOOLS_PATH
                                        Path to bedtools. Default is 'bedtools'
--platypus_calling {both,none,before,after}
                                        Platypus calling withing the pipeline␣
↪(before
                                        processing, after processing, both, or␣
↪neither).
                                        Options: both, none, before, after.
--no_variant_plots    Include flag to prevent plotting read balance from VCF
                                        files.
--no_bam_analysis     Include flag to prevent depth/mapq analysis of bam
                                        file. Used to isolate platypus_calling.
--skip_compatibility_check
                                        Include flag to prevent check of␣
↪compatibility between
                                        input bam and reference fasta
--no_perm_test        Include flag to turn off permutation tests.
--no_ks_test          Include flag to turn off KS Two Sample tests.
--no_bootstrap        Include flag to turn off bootstrap analyses. Requires
                                        either --y_present, --y_absent, or
                                        --sex_chrom_calling_threshold if running␣
↪full
                                        pipeline.
--variant_site_quality VARIANT_SITE_QUALITY, -vsq VARIANT_SITE_QUALITY
                                        Consider all SNPs with a site quality␣
↪(QUAL) greater
                                        than or equal to this value. Default is 30.
--variant_genotype_quality VARIANT_GENOTYPE_QUALITY, -vgq VARIANT_GENOTYPE_QUALITY
                                        Consider all SNPs with a sample genotype␣
↪quality
                                        greater than or equal to this value.␣
↪Default is 30.
--variant_depth VARIANT_DEPTH, -vd VARIANT_DEPTH
                                        Consider all SNPs with a sample depth␣
↪greater than or
                                        equal to this value. Default is 4.
--platypus_logfile PLATYPUS_LOGFILE
                                        Prefix to use for Platypus log files. Will␣
↪default to
                                        the sample_id argument provided
--homogenize_read_balance HOMOGENIZE_READ_BALANCE
                                        If True, read balance values will be␣
↪transformed by
                                        subtracting each value from 1. For example,␣
↪0.25 and
                                        0.75 would be treated equivalently. Default␣
↪is False.
--min_variant_count MIN_VARIANT_COUNT
                                        Minimum number of variants in a window for␣
↪the read
```

```
                                      balance of that window to be plotted. Note␣
↪that this
                                      does not affect plotting of variant counts.␣
↪Default is
                                      1, though we note that many window averages␣
↪will be
                                      meaningless at this setting.
--reference_mask [REFERENCE_MASK [REFERENCE_MASK ...]]
                                      Bed file containing regions to replace with␣
↪Ns in the
                                      sex chromosome reference. Examples might␣
↪include the
                                      pseudoautosomal regions on the Y to force␣
↪all
                                      mapping/calling on those regions of the X␣
↪chromosome.
                                      Default is None.
--xx_ref_out_name XX_REF_OUT_NAME
                                      Desired name for masked output fasta for␣
↪samples
                                      WITHOUT a Y chromosome (e.g., XX, XXX, XO,␣
↪etc.).
                                      Defaults to 'xyalign_noY.masked.fa'. Will␣
↪be output in
                                      the XYalign reference directory.
--xy_ref_out_name XY_REF_OUT_NAME
                                      Desired name for masked output fasta for␣
↪samples WITH
                                      a Y chromosome (e.g., XY, XXY, etc.).␣
↪Defaults to
                                      'xyalign_withY.masked.fa'. Will be output␣
↪in the
                                      XYalign reference directory.
--xx_ref_out XX_REF_OUT
                                      Desired path to and name of masked output␣
↪fasta for
                                      samples WITHOUT a Y chromosome (e.g., XX,␣
↪XXX, XO,
                                      etc.). Overwrites if exists. Use if you␣
↪would like
                                      output somewhere other than XYalign␣
↪reference
                                      directory. Otherwise, use --xx_ref_name.
--xy_ref_out XY_REF_OUT
                                      Desired path to and name of masked output␣
↪fasta for
                                      samples WITH a Y chromosome (e.g., XY, XXY,␣
↪etc.).
                                      Overwrites if exists. Use if you would like␣
↪output
                                      somewhere other than XYalign reference␣
↪directory.
                                      Otherwise, use --xy_ref_name.
--xx_ref_in XX_REF_IN
                                      Path to preprocessed reference fasta to be␣
↪used for
                                      remapping in X0 or XX samples. Default is␣
↪None. If
```

```
                                        none, will produce a sample-specific␣
↪reference for
                                        remapping.
--xy_ref_in XY_REF_IN
                                        Path to preprocessed reference fasta to be␣
↪used for
                                        remapping in samples containing Y␣
↪chromosome. Default
                                        is None. If none, will produce a sample-
↪specific
                                        reference for remapping.
--bwa_index BWA_INDEX
                                        If True, index with BWA during PREPARE_
↪REFERENCE. Only
                                        relevantwhen running the PREPARE_REFERENCE␣
↪module by
                                        itself. Default is False.
--read_group_id READ_GROUP_ID
                                        If read groups are present in a bam file,␣
↪they are
                                        used by default in remapping steps. However,
↪ if read
                                        groups are not present in a file, there are␣
↪two
                                        options for proceeding. If '--read_group_id␣
↪None' is
                                        provided (case sensitive), then no read␣
↪groups will be
                                        used in subsequent mapping steps. Otherwise,
↪ any other
                                        string provided to this flag will be used␣
↪as a read
                                        group ID. Default is '--read_group_id␣
↪xyalign'
--bwa_flags BWA_FLAGS
                                        Provide a string (in quotes, with spaces␣
↪between
                                        arguments) for additional flags desired for␣
↪BWA
                                        mapping (other than -R and -t). Example: '-
↪M -T 20 -v
                                        4'. Note that those are spaces between␣
↪arguments.
--sex_chrom_bam_only  This flag skips merging the new sex chromosome bam
                                        file back into the original bam file (i.e.,␣
↪sex chrom
                                        swapping). This will output a bam file␣
↪containing only
                                        the newly remapped sex chromosomes.
--sex_chrom_calling_threshold SEX_CHROM_CALLING_THRESHOLD
                                        This is the *maximum* filtered X/Y depth␣
↪ratio for an
                                        individual to be considered as having␣
↪heterogametic
                                        sex chromsomes (e.g., XY) for the REMAPPING␣
↪module of
                                        XYalign. Note here that X and Y chromosomes␣
↪are simply
```

```
                                        the chromosomes that have been designated␣
↪as X and Y
                                        via --x_chromosome and --y_chromosome. Keep␣
↪in mind
                                        that the ideal threshold will vary␣
↪according to sex
                                        determination mechanism, sequence homology␣
↪between the
                                        sex chromosomes, reference genome,␣
↪sequencing methods,
                                        etc. See documentation for more detail.␣
↪Default is
                                        2.0, which we found to be reasonable for␣
↪exome, low-
                                        coverage whole-genome, and high-coverage␣
↪whole-genome
                                        human data.
--y_present           Overrides sex chr estimation by XYalign and remaps
                                        with Y present.
--y_absent            Overrides sex chr estimation by XY align and remaps
                                        with Y absent.
--window_size WINDOW_SIZE, -w WINDOW_SIZE
                                        Window size (integer) for sliding window␣
↪calculations.
                                        Default is 50000. Default is None. If set␣
↪to None,
                                        will use targets provided using --target_
↪bed.
--target_bed TARGET_BED
                                        Bed file containing targets to use in␣
↪sliding window
                                        analyses instead of a fixed window width.␣
↪Either this
                                        or --window_size needs to be set. Default␣
↪is None,
                                        which will use window size provided with
                                        --window_size. If not None, and --window_
↪size is None,
                                        analyses will use targets in provided file.␣
↪Must be
                                        typical bed format, 0-based indexing, with␣
↪the first
                                        three columns containing the chromosome␣
↪name, start
                                        coordinate, stop coordinate.
--exact_depth         Calculate exact depth within windows, else use much
                                        faster approximation. *Currently exact is␣
↪not
                                        implemented*. Default is False.
--whole_genome_threshold
                                        This flag will calculate the depth filter␣
↪threshold
                                        based on all values from across the genome.␣
↪By
                                        default, thresholds are calculated per␣
↪chromosome.
--mapq_cutoff MAPQ_CUTOFF, -mq MAPQ_CUTOFF
                                        Minimum mean mapq threshold for a window to␣
↪be
```

```
                                        considered high quality. Default is 20.
--min_depth_filter MIN_DEPTH_FILTER
                                        Minimum depth threshold for a window to be␣
↪considered
                                        high quality. Calculated as mean depth *
                                        min_depth_filter. So, a min_depth_filter of␣
↪0.2 would
                                        require at least a minimum depth of 2 if␣
↪the mean
                                        depth was 10. Default is 0.0 to consider␣
↪all windows.
--max_depth_filter MAX_DEPTH_FILTER
                                        Maximum depth threshold for a window to be␣
↪considered
                                        high quality. Calculated as mean depth *
                                        max_depth_filter. So, a max_depth_filter of␣
↪4 would
                                        require depths to be less than or equal to␣
↪40 if the
                                        mean depth was 10. Default is 10000.0 to␣
↪consider all
                                        windows.
--num_permutations NUM_PERMUTATIONS
                                        Number of permutations to use for␣
↪permutation
                                        analyses. Default is 10000
--num_bootstraps NUM_BOOTSTRAPS
                                        Number of bootstrap replicates to use when
                                        bootstrapping mean depth ratios among␣
↪chromosomes.
                                        Default is 10000
--ignore_duplicates   Ignore duplicate reads in bam analyses. Default is to
                                        include duplicates.
--marker_size MARKER_SIZE
                                        Marker size for genome-wide plots in␣
↪matplotlib.
                                        Default is 10.
--marker_transparency MARKER_TRANSPARENCY, -mt MARKER_TRANSPARENCY
                                        Transparency of markers in genome-wide␣
↪plots. Alpha in
                                        matplotlib. Default is 0.5
--coordinate_scale COORDINATE_SCALE
                                        For genome-wide scatter plots, divide all␣
↪coordinates
                                        by this value.Default is 1000000, which␣
↪will plot
                                        everything in megabases.
--include_fixed INCLUDE_FIXED
                                        Default is False, which removes read␣
↪balances less
                                        than 0.05 and greater than 0.95 for␣
↪histogram
                                        plotting. True will include all values.␣
↪Extreme values
                                        removed by default because they often swamp␣
↪out the
                                        signal of the rest of the distribution.
--use_counts          If True, get counts of reads per chromosome for
```

```
                                        CHROM_STATS, rather than calculating mean␣
↪depth and
                                        mapq. Much faster, but provides less␣
↪information.
                                        Default is False
```

CHAPTER 3

Frequently Asked Questions

3.1 Does XYalign require X and Y chromosomes?

In principle, no, it doesn't. The focus on X and Y chromosomes stems from our initial interest in characterizing technical biases and aneuploidies affecting variant calling on the sex chromosomes in large human genomic datasets. Hence, the terminology we use throughout. You can provide the name of any chromosome or scaffold to `--x_chromosome` and `--y_chromosome`, and an arbitrary number of chromosome/scaffold names to `--chromosomes`. See *Usage Overview* for an example of how this might work. We plan to generalize XYalign in the future to make this easier.

3.2 Will XYalign work with genomes from other organisms?

Yes, but with some caveats. As discussed above, you can provide any chromosome names to `--x_chromosome` and `--y_chromosome`. So, if your organism has Z and W chromosomes, this might look like `--x_chromosome chrZ` and `--y_chromosome chrW`. However, we advise users to interpret results cautiously, as XYalign's default settings for human X and Y chromosomes are likely inappropriate for many other organisms. This is especially the case for ZW systems, or reference genomes without sequences for the Y (or equivalent) chromosome. In addition, XYalign does not currently accept multiple X or Y scaffolds. We plan to address these phenomena in future releases.

CHAPTER 4

API

4.1 xyalign

4.1.1 xyalign package

Subpackages

Submodules

xyalign.assemble module

xyalign.assemble.**bwa_mem_mapping_sambamba**(*bwa_path*, *samtools_path*, *sambamba_path*, *reference*, *output_prefix*, *fastqs*, *threads*, *read_group_line*, *bwa_params*, *cram=False*)

Maps reads to a reference genome using bwa mem. If output is in bam format, will sort using sambamba, else will sort with samtools

Parameters **bwa_path** : str

The path to bwa

samtools_path : str

The path to samtools

sambamba_path : str

The path to sambamba

reference : reftools.RefFasta() object

reftools.RefFasta() object of reference genome (in fasta format)

output_prefix : str

The prefix (including path) to the desired output files

fastqs : list

Fastqs, e.g. [‘sample_1.fastq’, ‘sample_2.fastq’]

threads : int

The number of threads/cpus to use

read_group_line : str

Read group info for bwa to add. If ‘None’, will not add read group.

bwa_params : list

Bwa parameters to be joined into a string and inserted into the command

cram : bool

If True, will output a sorted cram, else a sorted bam. Default is False.

Returns str

Path to output bam file (indexed)

Raises **RuntimeError**

If fastq or reference files cannot be found

xyalign.bam module

class xyalign.bam.**BamFile**(*filepath*, *samtools='samtools'*, *no_initial_index=False*)

A class for working with external bam files

Attributes

filepath	(str) Full path to external bam file.
samtools	(str) Full path to samtools. Default = ‘samtools’

is_indexed()

Checks that bam index exists, is not empty, and is newer than bam.

Returns bool

True if bam index exists and is newer than bam, False otherwise.

index_bam()

Indexes a bam using samtools (‘samtools index file.bam’).

Returns bool

True if successful.

Raises **RuntimeError**

If return code from external call is not 0.

get_chrom_length(*chrom*)

Extract chromosome length from BAM header.

Parameters **chrom** : str

The name of the chromosome or scaffold.

Returns **length** : int

The length (integer) of the chromsome/scaffold

Raises **RuntimeError**

If chromosome name not present in bam header

chromosome_lengths()

Returns tuple

chromosome lengths ordered by sequence order in bam header

chromosome_names()

Returns tuple

chromosome names ordered by sequence order in bam header

chromosome_bed(*output_file*, *chromosome_list*)

Takes list of chromosomes and outputs a bed file with the length of each chromosome on each line (e.g., chr1 0 247249719).

Parameters **output_file** : str

Name of (including full path to) desired output file

chromosome_list : list

Chromosome/scaffolds to include

Returns str

output_file

Raises **RuntimeError**

If chromosome name is not in bam header.

check_chrom_in_bam(*chromosome_list*)

Checks to see if all chromosomes in chromosome_list are in bam file

Parameters **chromosome_list** : list

Chromosomes/scaffolds to check

Returns list

List of chromosomes not in bam file

sort_bam(*sorted_bam*, *query_name=False*)

Sorts bam file by coordinate (query_name=False) or query name (query_name=True)

Parameters **sorted_bam** : str

Full path to (including desired name of) output bam file

query_name : bool

If True, sort by query name (read ID), else sort by coordinate

Returns BamFile() object

BamFile() object of new, sorted bam file

extract_regions(*regions*, *new_bam*, *rg_id=None*)

Extracts regions from a bam file into new bam file.

Parameters **regions** : list

regions from which reads will be stripped

new_bam : str

Full path to and desired name of output bam file

rg_id : str or None

Path to text file containing read group ids to use when isolating regions. If None, all reads from regions will be extracted.

Returns BamFile() object

BamFile() object of new bam file (containing extracted regions)

extract_read_group(*new_bam*, *rg_id*)

Extracts all reads belonging to a given RG ID from a bam file into new bam file.

Parameters **new_bam** : str

Full path to and desired name of output bam file

rg_id : str

Path to text file containing read group ids to use when isolating regions.

Returns BamFile() object

BamFile() object of new bam file (containing extracted regions)

strip_reads(*repairsh*, *shufflesh*, *single*, *output_directory*, *output_prefix*, *regions*, *repair_xmx*, *compression*, *cleanup=True*, *default_rg='None'*)

Strips reads from a bam or cram file in provided regions and outputs sorted fastqs containing reads, one set of fastq files per read group.

Parameters **repairsh** : str

Path to repair.sh (from BBmap)

shufflesh : str

Path to shuffle.sh (from BBmap)

single : bool

If true output single-end fastq, otherwise output paired-end fastqs

output_directory : str

The directory for ALL output (including temporary files)

output_prefix : str

The name (without path) to use for prefix to output fastqs

regions : list

regions from which reads will be stripped

repair_xmx : str

If "None", repair.sh will allocate its own memory. Otherwise value will be provided in the form of -Xmx4g, where 4g is the value provided as repair_xmx

compression : int

Desired compression level (0-9) for output fastqs. If 0, fastqs will be uncompressed.

cleanup : bool

If true, will clean up temporary files.

default_rg : str

If "None", no default read group will be created. Otherwise, default read group will be string provided. This read group will consist exclusively of an ID.

Returns list

A two-item list containing the path to a text file pairing read group names with associated output fastqs, and a text file containing a list of @RG lines associated with each read group

analyze_bam(*chrom*, *duplicates*, *exact*, *window_size*, *target_file=None*)

Analyze BAM (or CRAM) file for depth and mapping quality across genomic windows.

Parameters **chrom** : str

The name of the chromosome to analyze

duplicates : bool

If True, duplicates included in analyses.

exact : bool

If True, mean depth is calculated exactly within each window. If False, an accurate (and much faster) approximation is used

window_size

If int, the window size to use for sliding window analyses, if None intervals from target_file

target_file : str

Path to bed_file containing regions to analyze instead of windows of a fixed size. Will only be engaged if window_size is None

Returns pandas dataframe

pandas data frame with the columns: "chrom", "start", "stop", "depth", "mapq"

chrom_stats(*chrom*, *duplicates*)

Analyze BAM (or CRAM) file for depth and mapping quality across a single chromosome.

Parameters **chrom** : str

The name of the chromosome to analyze

duplicates : bool

If True, duplicates included in analyses.

Returns tuple

(mean_depth, mean_mapq)

chrom_counts()

Get read counts per chrom from a bamfile

platypus_caller(*platypus_path*, *log_path*, *ref*, *chroms*, *cpus*, *output_file*, *regions_file=None*)

Uses platypus to make variant calls on provided bam file

Parameters **platypus_path** : str

Path to platypus

log_path : str

Path to and name of desired log file for platypus

ref : str

Path to reference sequence

chroms : list

Chromosomes to call variants on, e.g., ["chrX", "chrY", "chr19"]

cpus : int

Number of threads/cores to use

output_file : path

Path to and name of the output vcf

regions_file : {str, None}

If not None, must be path to bed file containing regions to call variants in. If None, calls in call regions of provided chromosomes. Default = None.

Returns int

Exit code of the platypus call

xyalign.bam.**switch_sex_chromosomes_sambamba**(*samtools_path*, *sambamba_path*, *bam_orig*, *bam_new*, *sex_chroms*, *output_directory*, *output_prefix*, *threads*, *pg_header_dict*, *cram=False*)

Removes sex chromosomes from original bam and merges in remmapped sex chromosomes, while retaining the original bam header (and adding new @PG line)

Parameters **samtools_path** : str

The path to samtools

sambamba_path :

The path to sambamba

bam_orig : str

The path to the original full bam file

bam_new : str

The path to the bam file containing the remapped sex chromosomes

sex_chroms : list

Sex chromosomes (to be removed from bam_orig)

output_directory : str

The path to directory where all files (inc. temp) will be output

output_prefix : str

The name (without path) to use as prefix for all files

threads : int

The number of threads/cpus to use

pg_header_dict : dict

dictionary with information to be included in the new @PG line

- **must contain:** Key = 'CL', value = list of command line values Key = 'ID', value = string of program ID
- **optional:** Key = 'VN', value = string of program version

cram : bool

If True, will treat input as cram files and output cram files. Otherwise, will treat input as bam. Defaule is False. True is currently unsupported.

Returns str

Bam or cram file path with new sex chromosomes, but all others intact.

Raises **RuntimeError**

If cram is not False.

xyalign.bam.**samtools_merge**(*samtools_path*, *bam_list*, *output_prefix*, *threads*)

Merges bam files using samtools.

Parameters **samtools_path** : str

The path to samtools

bam_list : list

Bam files to be merged. Merging order will match order of this list.

output_prefix : str

Returns str

path to merged bam

xyalign.ploidy module

xyalign.ploidy.**permutation_test_chromosomes**(*data_frame*, *first_chrom*, *second_chrom*, *chrom_column*, *value_column*, *num_perms*, *output_file=None*)

Runs a permutation test comparing mean values of two chromosomes.

Parameters **data_frame** : pandas dataframe

first_chrom : str

The name of the first chromosome in comparison

second_chrom : str

The name of the second chromosome in comparison

chrom_column : str

The name of the column containing chromosome names

value_column : str

The name of the column containing the value of interest

num_perms : int

The number of permutations to use

output_file : {str, None}

If not None, will print results to this file

Returns tuple

(mean of first chrom, mean of second chrom, p-value)

xyalign.ploidy.**ks_two_sample**(*data_frame*, *first_chrom*, *second_chrom*, *chrom_column*, *value_column*, *output_file=None*)

Runs a Two-sample Kolmogorov-Smirnov test

Parameters **data_frame** : pandas dataframe

first_chrom : str

The name of the first chromosome in comparison

second_chrom : str

The name of the second chromosome in comparison

chrom_column : str

The name of the column containing chromosome names

value_column : str

The name of the column containing the value of interest

output_file : {str, None}

If not None, will print results to this file.

Returns tuple

(ks_statistic, ks_pvalue)

xyalign.ploidy.**bootstrap**(*data_frame*, *first_chrom*, *second_chrom*, *chrom_column*, *value_column*, *num_reps*, *output_file=None*)

Bootstraps the 95 percent confidence interval of the mean ratio of measure for two chromosomes (chrom1 / chrom2).

Parameters **data_frame** : pandas dataframe

first_chrom : str

The name of the first chromosome in comparison

second_chrom : str

The name of the second chromosome in comparison

chrom_column : str

The name of the column containing chromosome names

value_column : str

The name of the column containing the value of interest

num_reps : int

The number of bootstrap replicates to use

output_file : {str, None}

If not None, will print results to this file.

Returns tuple

(mean ratio, 0.025 percentile, 0.975 percentile)

xyalign.reftools module

class xyalign.reftools.**RefFasta**(*filepath*, *samtools='samtools'*, *bwa='bwa'*, *no_initial_index=False*)

A class for working with external reference fasta files

Attributes

filepath	(str) Full path to external bam file.
samtools	(str) Full path to samtools. Default = 'samtools'
bwa	(str) Full path to bwa. Default = 'bwa'

is_faidxed()

Checks that fai index exists, is not empty and is newer than reference.

Returns bool

True if fai index exists and is newer than fasta, False otherwise.

index_fai()

Create fai index for reference using samtools ('samtools faidx ref.fa')

Returns bool

True if successful

Raises **RuntimeError**

If return code from external call is not 0

index_bwa()

Index reference using bwa

Returns bool

True if successful

Raises **RuntimeError**

If return code from external call is not 0

check_bwa_index()

Checks to see if bwa indices are newer than fasta.

Returns bool

True if indices exist and are newer than fasta. False otherwise.

conditional_index_bwa(*bwa='bwa'*)

Indexes if indices are the same age or older than the fasta. Use RefFasta.index_bwa() to force indexing.

Parameters **bwa** : str

Path to bwa program (default is 'bwa')

check_seq_dict()

Checks that sequence dictionary exists, is not empty and is newer than reference.

Returns bool

True if seq dict exists and is newer than fasta, False otherwise.

seq_dict(*out_dict=None*)
Create sequence dictionary .dict file using samtools

Parameters **out_dict** : str

The desired file name for the sequence dictionary - defaults to adding '.dict' to the fasta name

Returns bool

True if exit code of external call is 0.

Raises **RuntimeError**

If external call exit code is not 0.

conditional_seq_dict()
Creates sequence dictionary if .dict the same age or older than the fasta, or doesn't exist.

Use RefFasta.seq_dict() to force creation.

mask_reference(*bed_mask*, *output_fasta*)
Creates a new masked references by hardmasking regions included in the bed_mask

Parameters **bed_mask** : str

Bed file of regions to mask (as N) in the new reference

output_fasta : str

The full path to and filename of the output fasta

Returns str

Path to new (indexed and masked) fasta

isolate_chroms(*new_ref_prefix*, *chroms*, *bed_mask=None*)
Takes a reference fasta file and a list of chromosomes to include and outputs a new, indexed (and optionally masked) reference fasta.

Parameters **new_ref_prefix** : str

The desired path to and prefix of the output files

chroms : list

Chromosomes to include in the output fasta

bed_mask : str

Bed file of regions to mask (as N) in the new reference

Returns str

Path to new, indexed (optionally masked) fasta

get_chrom_length(*chrom*)
Extract chromosome length from fasta.

Parameters **chrom** : str

The name of the chromosome or scaffold.

Returns **length** : int

The length (integer) of the chromsome/scaffold

Raises **RuntimeError**

If chromosome name not present in bam header

chromosome_bed(*output_file*, *chromosome_list*)
Takes list of chromosomes and outputs a bed file with the length of each chromosome on each line (e.g., chr1 0 247249719).

Parameters **output_file** : str

Name of (including full path to) desired output file

chromosome_list : list

Chromosome/scaffolds to include

Returns str

output_file

Raises **RuntimeError**

If chromosome name is not in fasta.

chromosome_lengths()

Returns tuple

Chromosome lengths ordered by sequence order in fasta

chromosome_names()

Returns tuple

Chromosome names ordered by sequence order in fasta

xyalign.utils module

xyalign.utils.**validate_external_prog**(*prog_path*, *prog_name*)
Checks to see if external program can be called using provided path

Parameters **prog_path: path to call program**

prog_name: name of program

Returns int

0

xyalign.utils.**validate_dir**(*parent_dir*, *dir_name*)
Checks if directory exists and if not, creates it.

Parameters **parent_dir** : Parent directory name

dir_name : Name of the new directory

Returns bool

whether the directory existed

xyalign.utils.**check_bam_fasta_compatibility**(*bam_object*, *fasta_object*)
Checks to see if bam and fasta sequence names and lengths are equivalent (i.e., if it is likely that the bam file was generated using the fasta in question).

Parameters **bam_object** : BamFile() object

fasta_object: RefFasta() object

Returns bool

True if sequence names and lengths match. False otherwise.

xyalign.utils.**check_compatibility_bam_list**(*bam_obj_list*)

Checks to see if bam sequence names and lengths are equivalent (i.e., if it is likely that the bam files were generated using the same reference genome).

Parameters **bam_obj_list** : list

List of bam.BamFile() objects

Returns bool

True if sequence names and lengths match. False otherwise.

xyalign.utils.**merge_bed_files**(*output_file*, **bed_files*)

This function simply takes an output_file (full path to desired output file) and an arbitrary number of external bed files (including full path), and merges the bed files into the output_file

Parameters **output_file** : str

Full path to and name of desired output bed file

***bed_files**

Variable length argument list of external bed files (include full path)

Returns str

path to output_file

xyalign.utils.**make_region_lists_genome_filters**(*depthAndMapqDf*, *mapqCutoff*, *min_depth*, *max_depth*)

Filters a pandas dataframe for mapq and depth based on using all values from across the entire genome

Parameters **depthAndMapqDf** : pandas dataframe

Must have 'depth' and 'mapq' columns

mapqCutoff : int

The minimum mapq for a window to be considered high quality

min_depth : float

Fraction of mean to set as minimum depth

max_depth : float

Multiple of mean to set as maximum depth

Returns tuple

(passing dataframe, failing dataframe)

xyalign.utils.**make_region_lists_chromosome_filters**(*depthAndMapqDf*, *mapqCutoff*, *min_depth*, *max_depth*)

Filters a pandas dataframe for mapq and depth based on thresholds calculated per chromosome

Parameters **depthAndMapqDf** : pandas dataframe

Must have 'depth' and 'mapq' columns

mapqCutoff : int

The minimum mapq for a window to be considered high quality

min_depth : float

Fraction of mean to set as minimum depth

max_depth : float

Multiple of mean to set as maximum depth

Returns tuple

(passing dataframe, failing dataframe)

xyalign.utils.**output_bed**(*outBed*, **regionDfs*)

Concatenate and merges dataframes into an output bed file

Parameters **outBed** : str

The full path to and name of the output bed file

***regionDfs**

Variable length list of dataframes to be included

Returns int

0

xyalign.utils.**output_bed_no_merge**(*outBed*, **regionDfs*)

Concatenate dataframes into an output bed file. This will preserve all columns after the first three as well.

Parameters **outBed** : str

The full path to and name of the output bed file

***regionDfs**

Variable length list of dataframes to be included

Returns int

0

xyalign.utils.**chromosome_wide_plot**(*chrom*, *positions*, *y_value*, *measure_name*, *sampleID*, *output_prefix*, *MarkerSize*, *MarkerAlpha*, *Xlim*, *Ylim*, *x_scale=1000000*)

Plots values across a chromosome, where the x axis is the position along the chromosome and the Y axis is the value of the measure of interest.

Parameters **chrom** : str

Name of the chromosome

positions : numpy array

Genomic coordinates

y_value : numpy array

The values of the measure of interest

measure_name : str

The name of the measure of interest (y axis title)

sampleID : str

The name of the sample

output_prefix : str

Full path to and prefix of desired output plot

MarkerSize : float

Size in points^2

MarkerAlpha : float

Transparency (0 to 1)

Xlim : float

Maximum X value

Ylim : float

Maximum Y value

x_scale : int

Divide all x values (including Xlim) by this value. Default is 1000000 (1MB)

Returns int

0

xyalign.utils.**hist_array**(*chrom*, *value_array*, *measure_name*, *sampleID*, *output_prefix*)

Plots a histogram of an array of values of interest. Intended for mapq and depth, but generalizeable. Separate function from variants.hist_read_balance because that function eliminates fixed variants, while this function will plot all values.

Parameters **chrom** : str

Name of the chromosome

value_array : numpy array

Read balance values

measure_name : str

The name of the measure of interest (y axis title)

sampleID : str

Sample name or id to include in the plot title

output_prefix : str

Desired prefix (including full path) of the output files

Returns int

0 if plotting successful, 1 otherwise.

xyalign.utils.**plot_depth_mapq**(*window_df*, *output_prefix*, *sampleID*, *chrom_length*, *MarkerSize*, *MarkerAlpha*, *x_scale=1000000*)

Creates histograms and genome-wide plots of various metrics.

Parameters **window_df** : pandas dataframe

Columns must include chrom, start, depth, and mapq (at least)

output_prefix : str

Path and prefix of output files to create

sampleID : str

Sample ID

chrom_length: int

Length of chromosome

x_scale : int

Divide all x values (including Xlim) by this value for chromosome_wide_plot. Default is 1000000 (1MB)

Returns int

0

xyalign.utils.**before_after_plot**(*chrom*, *positions*, *values_before*, *values_after*, *measure_name*, *sampleID*, *output_prefix*, *MarkerSize*, *MarkerAlpha*, *Xlim*, *YMin='auto'*, *YMax='auto'*, *x_scale=1000000*, *Color='black'*)

Plots difference between before/after values (after minus before) across a chromosome.

Parameters **chrom** : str

Name of the chromosome

positions : numpy array

Genomic coordinates

values_before : numpy array

The values of the measure of interest in the "before" condidtion

values_after : numpy array

The values of the measure of interest in the "after" condidtion

measure_name : str

The name of the measure of interest (for y-axis title)

sampleID : str

The name of the sample

output_prefix : str

Full path to and prefix of desired output plot

MarkerSize : float

Size in points^2

MarkerAlpha : float

Transparency (0 to 1)

Xlim : float

Maximum X value

YMin : str, int, or float

If "auto", will allow matplotlib to automatically determine limit. Otherwise, will set the y axis minimum to the value provided (int or float)

YMax : str, int, or float

If "auto", will allow matplotlib to automatically determine limit. Otherwise, will set the y axis maximum to the value provided (int or float)

x_scale : int

Divide all x values (including Xlim) by this value. Default is 1000000 (1MB)

Color : str

Color to use for points. See matplotlib documentation for acceptable options

Returns int

0 if plotting successful, 1 otherwise

xyalign.variants module

class xyalign.variants.**VCFFile**(*filepath*, *bgzip='bgzip'*, *tabix='tabix'*, *no_initial_compress=False*)

A class for working with external vcf files.

Attributes

filepath	(str) Full path to external vcf file
bgzip	(str) Full path to bgzip. Default = 'bgzip'
tabix	(str) Full path to tabix. Default = "tabix"

is_bgzipped()

Checks to see if vcf file is gzipped, simply by looking for a .gz or .bgz ending. If .gz or .bgz ending exists, assumes file is compressed using bgzip.

Returns bool

True if ends in .gz, False otherwise

compress_vcf()

Compresses vcf file using bgzip.

Returns bool

True if successful

Raises **RuntimeError**

If return code from external call is not 0

index_vcf()

Indexes vcf file using tabix. If file does not end in .gz, will compress with bgzip (by calling self.compress_vcf).

Note: Files MUST be compressed using bgzip.

Returns bool

True if successful.

Raises **RuntimeError**

If return code from external call is not 0.

parse_platypus_VCF(*site_qual*, *genotype_qual*, *depth*, *chrom*)

Parse vcf generated by Platypus to grab read balance. Note that this is hard-coded to Platypus (version 0.8.1) and will not generalize to vcfs generated with other programs (and, potentially, other versions of Platypus)

Parameters **site_qual** : int

Minimum (PHRED) site quality at which sites should be included

genotype_qual : int

Minimum (PHRED) genotype quality at which sites should be included

depth : int

Minimum depth at which sites should be included

chrom : str

Name of the chromosome to include

Returns tuple

five corresponding arrays of the same length: (position across the chromosome, site quality, read balance, genotype quality, and depth)

plot_variants_per_chrom(*chrom_list*, *sampleID*, *output_prefix*, *site_qual*, *genotype_qual*, *depth*, *MarkerSize*, *MarkerAlpha*, *bamfile_obj*, *variant_caller*, *homogenize*, *dataframe_out*, *min_count*, *window_size*, *x_scale=1000000*, *target_file=None*, *include_fixed=False*)

Parses a vcf file and plots read balance in separate plots for each chromosome in the input list

Parameters **chrom_list** : list

Chromosomes to include

sampleID : str

Sample ID (for plot titles)

output_prefix : str

Full path to and prefix of desired output plots

site_qual : int

Minimum (PHRED) site quality at which sites should be included

genotype_qual : int

Minimum (PHRED) genotype quality at which sites should be included

depth : int

Minimum depth at which sites should be included

MarkerSize : float

Size of markers (matplotlib sizes) to use in the figure

MarkerAlpha : float

Transparency (matplotlib values, 0 to 1) of markers

bamfile_obj : BamFile() object

Used to get chromosome lengths only

variant_caller : str

Variant caller used to generate vcf - currently only "platypus" supported

homogenize: bool

If True, all read balance values less than 0.5 will be transformed by subtracting the value from 1. For example, the values 0.25 and 0.75 would be treated as equivalent.

dataframe_out : str

Full path of file to write pandas dataframe to. Will overwire if exists

min_count : int

Minimum number of variants to include a window for plotting.

window_size

If int, the window size to use for sliding window analyses, if None intervals from target_file

x_scale : int

Divide all x values (including Xlim) by this value. Default is 1000000 (1MB)

target_file : str

Path to bed_file containing regions to analyze instead of windows of a fixed size. Will only be engaged if window_size is None

include_fixed : bool

If False, only plots histogram for values between 0.05 and 1.0. If True, plots histogram of all variants.

Returns int

0 if variants to analyze; 1 if no variants to analyze on any chromosome

xyalign.variants.**read_balance_per_window**(*chrom*, *positions*, *readBalance*, *sampleID*, *homogenize*, *chr_len*, *window_size*, *target_file=None*)

Calculates mean read balance per genomic window (defined by size or an external target bed file) for a given chromosome. Takes as input an array of positions and an array of read balances - the order of which must correspond exactly. In addition, the positions are expected to ALL BE ON THE SAME CHROMOSOME and be in numerically sorted order (i.e., the output of parse_platypus_VCF())

Parameters **chrom** : str

Name of the chromosome

positions : numpy array

Positions along the chromosome (same length as readBalance)

readBalance : numpy array

Read balance corresponding with the positions in the positions array

sampleID : str

Sample name or id to include in the plot title

homogenize: bool

If True, all read balance values less than 0.5 will be transformed by subtracting the value from 1. For example, the values 0.25 and 0.75 would be treated as equivalent.

chr_len : int

Length of chromosome. Ignored if target_file is provided.

window_size

If int, the window size to use for sliding window analyses, if None intervals from target_file

target_file : str

Path to bed file containing regions to analyze instead of windows of a fixed size. Will only be engaged if window_size is None

Returns pandas dataframe

With columns: "chrom", "start", "stop", "balance", and "count"

xyalign.variants.**plot_read_balance**(*chrom*, *positions*, *readBalance*, *sampleID*, *output_prefix*, *MarkerSize*, *MarkerAlpha*, *homogenize*, *chrom_len*, *x_scale=1000000*)

Plots read balance at each SNP along a chromosome

Parameters **chrom** : str

Name of the chromosome

positions : numpy array

Positions along the chromosome (same length as readBalance)

readBalance : numpy array

Read balance corresponding with the positions in the positions array

sampleID : str

Sample name or id to include in the plot title

output_prefix : str

Desired prefix (including full path) of the output files

MarkerSize : float

Size of markers (matplotlib sizes) to use in the figure

MarkerAlpha : float

Transparency (matplotlib values) of markers for the figure

homogenize: bool

If True, all read balance values less than 0.5 will be transformed by subtracting the value from 1. For example, the values 0.25 and 0.75 would be treated as equivalent.

chrom_len : int

Length of chromosome

x_scale : int

Divide all x values (including Xlim) by this value. Default is 1000000 (1MB)

Returns int

0

xyalign.variants.**hist_read_balance**(*chrom*, *readBalance*, *sampleID*, *homogenize*, *output_prefix*, *include_fixed=False*)

Plots a histogram of read balance values between 0.05 and 1.0 (non-incusive)

Parameters **chrom** : str

Name of the chromosome

readBalance : list or numpy array

Read balance values

sampleID : str

Sample name or id to include in the plot title

homogenize: bool

If True, all read balance values less than 0.5 will be transformed by subtracting the value from 1. For example, the values 0.25 and 0.75 would be treated as equivalent.

output_prefix : str

Desired prefix (including full path) of the output files

include_fixed : bool

If False, only plots histogram for values between 0.05 and 1.0. If True, plots histogram of all variants.

Returns int

0 if plotting successful, 1 otherwise.

xyalign.xyalign module

xyalign.xyalign.**parse_args**()
Parse command-line arguments

Returns Parser argument namespace

xyalign.xyalign.**ref_prep**(*ref_obj*, *ref_mask*, *ref_dir*, *xx*, *xy*, *y_chromosome*, *samtools_path*, *bwa_path*, *bwa_index*)
Reference prep part of XYalign pipeline.

- Creates two reference fasta files. Both will include masks provied with

ref_mask. One will additionally have the entire Y chromosome hard masked.

- Indexes (.fai, .dict, and optionally bwa indices) both new references

Parameters **ref_obj** : RefFasta() object

A reftools.RefFasta() object of a fasta reference file to be processed

ref_mask : list or None

List of files to use to *hard-mask* references. None will ignore masking.

ref_dir : str

Path to output directory

xx : str

Path to XX output reference

xy : str

Path to XY output reference

y_chromosome : str

Name of Y chromosome in fasta

samtools_path : str

The path to samtools (i.e, "samtools" if in path)

bwa_path : str

The path to bwa (i.e, "bwa" if in path)

bwa_index : bool

If True, create bwa indices. Don't if False.

Returns tuple

Paths to two masked references (y_masked, y_unmasked)

xyalign.xyalign.**chrom_stats**(*bam_obj_list*, *chrom_list*, *use_counts*)

Runs chrom stats module.

Calculates mean depth and mapq across entire scaffolds for a list of bam files

Returns tuple

Tuple containing two dictionaries with results for depth and mapq, respectively. Or, if use_counts is True, returns a tuple containing the count dictionary and None.

xyalign.xyalign.**bam_analysis**(*input_bam_obj*, *platypus_calling*, *platypus_path*, *vcf_log*, *ref_obj*, *input_chroms*, *cpus*, *out_vcf*, *no_variant_plots*, *window_size*, *target_bed*, *sample_id*, *readbalance_prefix*, *variant_site_quality*, *variant_genotype_quality*, *variant_depth*, *marker_size*, *marker_transparency*, *homogenize_read_balance*, *data_frame_readbalance*, *min_variant_count*, *no_bam_analysis*, *ignore_duplicates*, *exact_depth*, *whole_genome_threshold*, *mapq_cutoff*, *min_depth_filter*, *max_depth_filter*, *depth_mapq_prefix*, *bam_data_frame*, *output_bed_high*, *output_bed_low*, *use_bed_for_platypus*, *coordinate_scale*, *fixed*)

Runs bam analyis part of XYalign pipeline on bam file.

- (Optionally) calls variants using Platypus
- (Optionally) parses and filters Platypus vcf, and plots read balance
- **(Optionally) Calculates window based metrics from the bam file:** depth and mapq
- (optionally) Plots window-based metrics
- Outputs two bed files: high quality windows, and low quality windows.

Parameters **input_bam_obj** : bam.BamFile() object

platypus_calling : bool

If True, will call and analyze variants

platypus_path : str

Command to call platypus (e.g, "platypus")

vcf_log : str

Path to file for platypus log

ref_obj : reftools.RefFasta() object

input_chroms : list

Chromosomes to analyze

cpus : int

Number of threads/cpus

out_vcf : str

Output vcf path/name

no_variant_plots : bool

If True, will not plot read balance

window_size : int or None

Window size for sliding window analyses (both bam and vcf). If None, will use regions in target_bed

target_bed : str or None

Path to bed file containing targets to use in sliding window analyses

sample_id : str

readbalance_prefix : str

Prefix, including full path, to use for output files for readbalance analyses

variant_site_quality : int

Minimum site quality (PHRED) for a site to be included in readbalance analyses

variant_genotype_quality : int

Minimum genotype quality for a site to be included in read balance analyses

variant_depth : int

Minimum depth for a site to be included in read balance analyses

marker_size : float

Marker size for plotting genome scatter plots

marker_transparency: float

Value to use for marker transparency in genome scatter plots

homogenize_read_balance : bool

If true, will subtract values less than 0.5 from 1. I.e., 0.25 and 0.75 would be treated equivalently

data_frame_readbalance: str

Path of output file for full read balance dataframe

min_variant_count : int

Minimum number of variants in a given window for the window to be plotted in window-based read balance analyses

no_bam_analysis : bool

If True, no bam analyses will take place

ignore_duplicates : bool

If True, duplicates excluded from bam analyses

exact_depth : bool

If True, exact depth calculated in each window. Else, a much faster approximation will be used

whole_genome_threshold : bool

If True, values for depth filters will be calculated using mean from across all chromosomes included in analyses. Else, mean will be taken per chromosome

min_depth_filter : float

Minimum depth threshold for a window to be considered high. Calculated as mean depth * min_depth_filter.

max_depth_filter : float

Maximum depth threshold for a window to be considered high. Calculated as mean depth * min_depth_filter.

depth_mapq_prefix : str

Prefix, including full path, to be used for files output from depth and mapq analyses

bam_data_frame : str

Full path to output file for dataframe containing all data from bam analyses

output_bed_high : str

Full path to output bed containing high quality (i.e., passing filters) windows

output_bed_low : str

Full path to output bed containing low quality (i.e., failing filters) windows

use_bed_for_platypus : bool

If True, use output_bed_high as regions for Platypus calling

coordinate_scale : int

Divide all coordinates by this value for plotting. In most cases, 1000000 will be ideal for eukaryotic genomes.

fixed : bool

If False, only plots histogram for values between 0.05 and 1.0 (non-inclusive). If True, plots histogram of all variants.

Returns tuple

(list of pandas dataframes with passing windows, list of pandas dataframes with failing windows)

xyalign.xyalign.**ploidy_analysis**(*passing_df*, *failing_df*, *no_perm_test*, *no_ks_test*, *no_bootstrap*, *input_chroms*, *x_chromosome*, *y_chromosome*, *results_dir*, *num_permutations*, *num_bootstraps*, *sample_id*)

Runs the ploidy analysis part of XYalign.

- Runs permutation test to systematically compare means between

every possible pair of chromosomes

- Runs K-S two sample test to systematically compare distributions between

every possible pair of chromosomes

- Bootstraps the mean depth ratio for every possible pair of chromosomes

Parameters **passing_df** : list

Passing pandas dataframes, one per chromosome

failing_df : list

Failing pandas dataframes, one per chromosome

no_perm_test : bool

If False, permutation test will be run

no_ks_test : bool

If False, KS test will be run

no_bootstrap : bool

If False, bootstrap analysis will be run

input_chroms : list

Chromosomes/scaffolds to analyze

x_chromosome : list

X-linked scaffolds

y_chromosome : list

Y-likned scaffolds

results_dir : str

Full path to directory to output results

num_permutations : int

Number of permutations

num_bootstraps : int

Number of bootstrap replicates

sample_id : str

Returns dictionary

Results for each test. Keys: perm, ks, boot.

xyalign.xyalign.**remapping**(*input_bam_obj*, *y_pres*, *masked_references*, *samtools_path*, *sambamba_path*, *repairsh_path*, *shufflesh_path*, *bwa_path*, *bwa_flags*, *single_end*, *bam_dir*, *fastq_dir*, *sample_id*, *x_chromosome*, *y_chromosome*, *cpus*, *xmx*, *fastq_compression*, *cleanup*, *read_group_id*)

Runs remapping steps of XYalign.

- Strips, sorts, and re-pair reads from the sex chromosomes (collecting read group information)
- Maps (with sorting) reads (with read group information) to appropriate reference based on presence (or not) of Y chromosome
- Merge bam files (if more than one read group)

Parameters **input_bam_obj** : bam.BamFile() object

y_pres : bool

True if Y chromosome present in individual

masked_references : tuple

Masked reference objects (xx, xy)

samtools_path : str

Path/command to call samtools

sambamba_path : str

Path/command to call sambamba

repairsh_path : str

Path/command to call repair.sh

shufflesh_path : str

Path/command to call shuffle.sh

bwa_path : str

Path/command to call bwa

bwa_flags : str

Flags to use for bwa mapping

single_end : bool

If True, reads treated as single end

bam_dir : str

Path to output directory for bam files

fastq_dir : str

Path to output directory for fastq files

sample_id : str

x_chromosome : list

X-linked scaffolds

y_chromosome : list

Y-linked scaffolds

cpus : int

Number of threads/cpus

xmx : str

Value to be combined with -Xmx for java programs (i.e., 4g would result in -Xmx4g)

fastq_compression : int

Compression level for fastq files. 0 leaves fastq files uncompressed. Otherwise values should be between 1 and 9 (inclusive), with larger values indicating more compression

cleanup : bool

If True, will delete temporary files

read_group_id : str

ID to use to add read group information

Returns str

Path to bam containing remapped sex chromsomes

xyalign.xyalign.**swap_sex_chroms**(*input_bam_obj*, *new_bam_obj*, *samtools_path*, *sambamba_path*, *x_chromosome*, *y_chromosome*, *bam_dir*, *sample_id*, *cpus*, *xyalign_params*)

Switches sex chromosmes from new_bam_file with those in original bam file

Parameters **input_bam_obj** : bam.BamFile() object

Original input bam file object

new_bam_obj : bam.BamFile() object

Bam file object containing newly mapped sex chromosomes (to insert)

samtools_path : str

Path/command to call samtools

sambamba_path : str

Path/command to call sambamba

x_chromosome : list

X-linked scaffolds

y_chromosome : str

Y-linked scaffolds

bam_dir : str

Path to bam output directory

sample_id : str

cpus : int

Number of threads/cpus

xyalign_params : dict

Dictionary of xyalign_params to add to bam header

Returns str

Path to new bam file containing original autosomes and new sex chromosomes

xyalign.xyalign.**main**()

Module contents

CHAPTER 5

Release History

5.1 1.1.5

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v1.1.5
- Fix bug in variants.read_balance_per_window in calculating final window length

5.2 1.1.3

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v1.1.2
- More work making utility scripts available in pip and bioconda

5.3 1.1.2

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v1.1.2
- Fixed import errors for the utility scripts (plot_cout_stats, plot_window_differences, and explore_thresholds)

5.4 1.1.1

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v1.1.1
- Couple of minor documentation and testing updates

5.5 1.1.0

- **Updates across all of XYalign, but most substantial were:**

  - The addition of plot_cout_stats, plot_window_differences, and explore_thresholds
  - XYalign now outputs Adobe Illustrator compatible pdfs for figures
  - Fixing some bugs in VCF parsing
  - Allowing plotting of fixed variants in read balance figures, if desired
- See detailed list of changes here: https://github.com/WilsonSayresLab/XYalign/blob/master/xyalign/CHANGELOG.txt

5.6 1.0.0

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v1.0.0
- Major updates across all of XYalign
- See detailed list of changes here: https://github.com/WilsonSayresLab/XYalign/blob/master/xyalign/CHANGELOG.txt

5.7 0.1.1

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v0.1.1
- Minor documentation updates

5.8 0.1.0

- Available: https://github.com/WilsonSayresLab/XYalign/releases/tag/v0.1
- Initial release
- Released April 5, 2017
- Full support of human-style reference genomes with X and Y chromosomes.
- No support for reference genomes without Y chromosome

5.9 0.0.1 Prerelease

- Development version until April 4, 2017

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

x

Index

P

R

S

V

X