QPalma Documentation

Fabio De Bona & Gunnar Rätsch

November 2010

1 Overview

QPALMA [1] is a component of *PALMapper* [2], an alignment tool targeted to accurately and efficiently align both unspliced and spliced reads produced by Next Generation sequencing platforms such as *Illumina Genome Analyzer* or *454*. *PALMapper* is designed to deal with the relatively short length and the possible low quality of reads by combining both the training algorithm and the alignment component of *QPALMA* with the efficient read mapping algorithm of *GenomeMapper* [3].
QPALMA relies on a machine learning strategy similar to Support Vector Machine (SVM) and provides a scoring function to optimally combine several pieces of information, in particular, the (a) alignment information, (b) computational splice-site predictions, (c) read quality values and (d) (optionally) intron lengths. This *QPALMA* scoring model is then used by *PALMapper* to guide a semi-global alignment algorithm that allows for long gaps that correspond to introns. For further details on *QPALMA* itself consult the paper [1]. For details about the learning method see [4].
This documentation describes how to install the training component of *QPALMA* and how to train it based on (a) the reference genome, (b) a set of RNA-seq reads, and (c) a small set of annotated (spliced) transcripts.
The official *QPALMA* project email address is:

qpalma@tuebingen.mpg.de

Alternatively, you may contact Gunnar Rätsch (Gunnar.Raetsch@tuebingen.mpg.de).

2 Installation

2.1 Dependencies

QPALMA is designed to run on *Linux*/UNIX or *Mac OS X* platforms and can be downloaded at this address:
`http://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/software`.
This package is distributed under the GNU Public License (GPL). Read the license before installation. The programs are distributed as C++ and Python source for *QPALMA*. The memory requirement of *QPALMA* very much depends on how many examples are used for training. In most cases 4GB of RAM are sufficient.
The software package has the following dependencies on other packages:

- SWIG, the simple wrapper interface generator (`http://www.swig.org`)
- Python $\geq$ 2.5 (`http://www.python.org`)
- C compiler, for instance the GNU C Compiler gcc (`http://www.gnu.org`). For other compilers, the compiler flags may need to be adapted in `Makefile` files.

- numpy, a python package for numeric computations (`http://numpy.scipy.org`)
- *PALMapper*, the short read aligner used here for preliminary alignments to estimate an error model (see below). This software can be downloaded at this address: `http://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/software/`.
- one of the following optimization toolkits:
  CVXOPT (`http://abel.ee.ucla.edu/cvxopt`, free), or
  MOSEK (`http://www.mosek.com`, commercial, free academic licenses available).
  These both optimizers provide the same performance. However, MOSEK is easier to install and run faster than CVXOPT but CVXOPT is entirely free.
- Standard programs, such as `wget`, `make`, etc.

2.2 Step by step installation guide

1. Go to `http://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/software/` and download the package `qpalma/qpalma-0.9.3.tar.gz` to the home directory.
2. Extract the tar-gzipped file in your home directory as follows:

   ```
   user:~$ cd
   user:~$ tar zxvf qpalma-0.9.3.tar.gz
   ```

   This command decompresses and unpacks the different files contained in the archive to a directory named `qpalma-0.9.3`.
3. From home directory, run the `setup_qpalma.sh` script by typing:

   ```
   user:~$ cd qpalma-0.9.3
   user:~/qpalma-0.9.3$ ./setup_qpalma.sh
   ```

4. Follow the interactive instructions given. First, set *QPALMA* base directory by copying the suggested path.
5. Type `1` or `2` to select the optimizer to use. According to the selected optimizer:
   a. For MOSEK, select between 2 and 4 for downloading complementary binaries according to the architecture of the system or type 1 to use existing installation.
   In short, to know which Linux architecture to choose, type `uname -m` in an other shell window. If the result of the command is `x86_64`, it is a Linux 64-bit system, otherwise it is a 32-bit (`i386`) system. During the setup process, the optimization package MOSEK with a trial license will be downloaded and installed in the directory `./modules/mosek`. The downloaded trial license is restricted to 300 *variables, which is not sufficient for most cases. The user can obtain a trial license without this restriction (but time limited) from*
   `https://www.mosek.com/cgi-bin/trial.py`. *The obtained license file has to be copied to `./modules/mosek/5/licenses/mosek.lic` to get it to work.*
   b. for CVXOPT, press *Enter* to confirm the current installation `bin/` directory.
   The user needs to install CVXOPT manually in the corresponding installation `bin/` directory. It can be downloaded from
   `http://abel.ee.ucla.edu/cvxopt/download/index.html`.

6. The setup script builds the necessary C++ extension modules and the Python code, and creates a configuration file `./bin/qpalma_config.sh`.

3 Training QPALMA

This section describes how to train the *QPALMA* component of *PALMapper* based on the reference genome, a set of RNA-seq reads and a small set of annotated (spliced) transcripts. During this process an error model is estimated from a preliminary alignment of the reads to the reference genome and a set of artificial reads is created from the annotated transcripts and the error model. Both steps are necessary to generate a set of realistic artificial reads with known alignment in order to train *QPALMA*. The training and later the alignment benefit from computational splice sites predictions for the reference genome. The result of the training step is a *QPALMA* parameter file adapted to the specifics of the reads of an RNA-seq experiment, which is required for the alignment protocol.
The *QPALMA* and *PALMapper* releases also contain pre-trained *QPALMA* parameter files, which can be used alternatively. It should be noted, however, that the *QPALMA* parameters optimally trade-off the amount errors in the reads and the information from splice site predictions for one combination of RNA-seq experiment, splice site predictions and genome. Using sharing parameter sets among experiments or genome can lead to a suboptimal alignment performance with *PALMapper*.

3.1 Input files

Training *QPALMA* needs the following files:

- Genome sequence in FASTA format.

- Read data to align in Sanger FASTQ format (see section 5.1 for Sanger FASTQ format).

- A partial genome annotation in GFF3 format. The genome annotation in GFF3 format does not need to be perfect nor complete. However, ideally, it contains a few hundred spliced transcripts or partial transcripts, which are representative of the rest of the genome. The quality of annotation near intron boundaries is most influencing the subsequent steps: a bad annotation can lead to a suboptimal *QPALMA* model and hence affect the quality of the final alignments.

- Donor and acceptor splice site predictions in binary signal prediction format (BSPF) and their associated files (see section 5.2 for BSPF format). There are two ways for obtaining these files: splice site predictions can be computed for a given genome via an appropriate tool (see mGene [5, 6] or ASP [7] for example) or the user can directly download precomputed splice site predictions for a growing list of organisms at `http://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/predictions/splice`. In the last case, the splice site predictions have to be used together with the corresponding version of genome sequence. Alternatively, splice site predictions can be predicted using the Galaxy system (`http://galaxy.tuebingen.mpg.de/`) and then downloaded for local use. The user may also disable the use of splice site predictions to train *QPALMA* by using `-no-ss-pred` parameter in the way explained in step 2 below.

3.2 Training QPALMA on the command-line

1. Open a shell window and go to *QPALMA* directory:

   ```
   user:~$ cd qpalma-0.9.3
   ```

2. Use the command below to train *QPALMA* within the working directory:

   ```
   user:~/qpalma-0.9.3$ ./tools/train_qpalma_artificial_reads.sh \
   <genome_file> <read_file> <anno_file> <gff3_source> <acc_pred_file> \
   <don_pred_file>
   ```

   where:

   - `<genome_file>` is the path to genome sequence file.
   - `<read_file>` is the path to genome read data file.
   - `<anno_file>` is the path to genome annotation file.
   - `<gff3_source>` is the second column of the annotation file corresponding to transcripts.
   - `<acc_pred_file>`, `<don_pred_file>` are the paths to acceptor and donor splice site prediction files. If splice site predictions are not available or if the user wishes to train *QPALMA* without splice site predictions, the files `<acc_pred_file>` and `<don_pred_file>` have to be replaced by the parameter `-no-ss-pred`.

   The result will be a *QPALMA* parameter file that can be used for alignments with *PALMapper*. The run-time is about two hours on a standard desktop computer (depending on a few properties of the transcript annotations).

   The above program calls several scripts that experienced users may want run manually:

   - Computation of a random subset of reads from the original data:

     ```
     user:~/qpalma-0.9.3$ python ./tools/sequences/subsample_fastq.py \
     <read_file> <no_reads> <sample_file>
     ```

     Since the read data file `<read_file>` is quite huge, it is sufficient to take a subset of `<no_reads>` reads as sample source. The output file `<sample_file>` is in Sanger FASTQ format. Since the errors are not uniformly distributed in most read files returned by sequencers, it is not sufficient to simply use the first part of the file.

   - Estimation of an error model:

     ```
     user:~/qpalma-0.9.3$ ./tools/errormodel/create_error_model.sh \
     <sample_file> <genome_file> <no_reads> <pickle_file> \
     <quality_sample>
     ```

     This tool creates an error model from a given read file `<sample_file>`, a number of reads `<no_reads>` and a reference genome `<genome_file>`. The user should use a read file that has either a sufficient size (> 100000 reads) or is a representative subset of a bigger read file that can be created with the sub-sampling script. The user can precise the same number of reads as before to take the whole sub-sample into account. The output files are the error model `<pickle_file>` in PICKLE format and a `<quality_sample>` in plain TXT format.

- *(Optional)* Analysis of the error model:

  ```
  user:~/qpalma-0.9.3$ python ./tools/errormodel/evaluateErrormodel.py \
  -e <pickle_file> -q <quality_sample> -H <plot_file>
  ```

  Optionally, it can be interesting to visualize different properties of the error model of the data for evaluation and analysis of the alignment. From an error model `<pickle_file>` and an associated `<quality_sample>`, an evaluation plot `<plot_file>` in PNG format is created.

- Creation of artificial reads:

  ```
  user:~/qpalma-0.9.3$ ./tools/gff/create_art_reads.sh \
  <out_dir> <pickle_file> <genome_file> <anno_file> <gff3_source> \
  <read_length> <no_transcripts> <no_reads> [<paired>]
  ```

  Based on error model, genome sequence and annotated transcripts in the GFF3 file, the tool will create `<no_reads>` artificial reads of length `<read_length>` that are sampled from `<no_transcripts>` different transcripts. The resulting reads in Sanger-FASTQ format, their optimal alignments in SAM format and the *QPALMA* training set encoded in BED format (described in section 5.3) are written out in `<out_dir>`.

- *QPALMA* training: the training is done by calling the program:

  ```
  user:~/qpalma-0.9.3/bin$ qpalma_train <config_file> <qpalma_train.bed>\
  <working_dir>
  ```

  In addition to the training file encoded in BED format (obtained in the previous step) and the output directory, it requires a configuration file with all necessary information. This configuration file is described in section 5.4.

  For simplicity, one may use a script to automatically generate the configuration file from environment variables that can be set as follows:

  ```
  export QPALMA_max_intron_length=50000         # maximal intron length
  export QPALMA_half_window_size=50000      # window size (=intron len)
  export QPALMA_enable_intron_length=False      # no intron len scoring
  export QPALMA_prb_offset=33                   # quality value offset
  export QPALMA_max_qual=40                     # maximal quality value
  export QPALMA_iterations=50                   # iteration limit
  export QPALMA_C=10                         # regularization parameter
  export QPALMA_num_supp_points=10           # number of pieces in PLIFs
  export QPALMA_genome_dir=<genome_file>        # genome FASTA file
  export QPALMA_acceptor_scores=<acc pred file> # acceptor splice pred.
  export QPALMA_donor_scores=<don pred file>    # donor splice pred.
  export QPALMA_drop_unspliced=True         # no unspliced reads
  ```

  To train *QPALMA* without splice site predictions, the user has to replace the two lines concerning splice site prediction files by the following one:

  ```
  export QPALMA_enable_splice_scores=False    # no splice site pred.
  ```

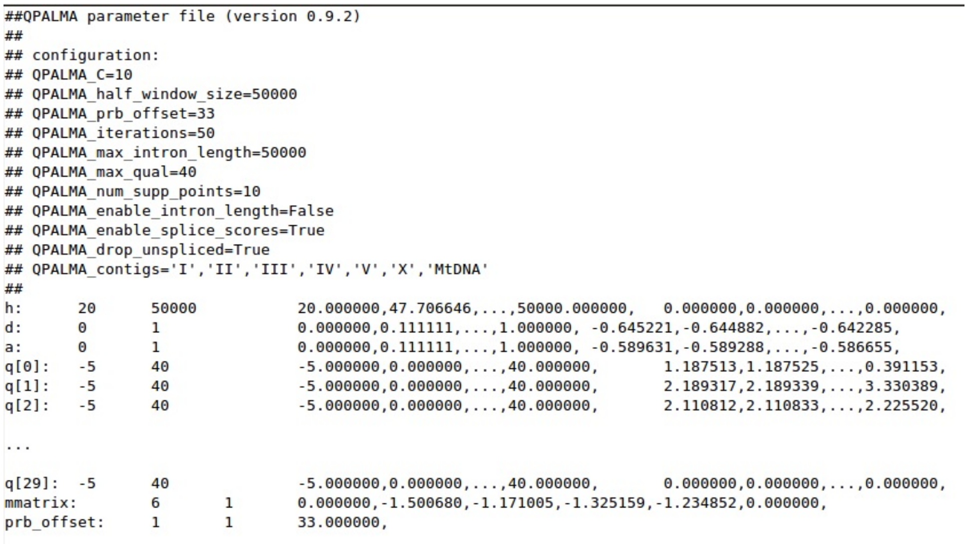
  The script that automatically generates the configuration file and launches the training can be started with:

  ```
  user:~/qpalma-0.9.3$ ./galaxy/qpalma_train.sh <qpalma_train.bed> \
  <working_dir> <parameters.qpalma>
  ```

where `<qpalma_train.bed>` is the *QPALMA* training set based on artificial reads, `<working_dir>` is a directory to store intermediate results, and `<parameters.qpalma>` is the resulting *QPALMA* parameter file. Depending on the parameter settings, training may take between a few minutes to a few hours.

For a deeper comprehension of these scripts, their settings and defaults parameter values, we advise the user to edit manually and to read carefully the file `./tools/train_qpalma_artificial_reads.sh`.

3. A *QPALMA* parameter file called parameters.qpalma has been generated in the working directory. It describes the piece-wise linear functions (PLiFs) for scoring intron lengths, splice sites and possible edit operations. See Figure 1 for more information.



```
##QPALMA parameter file (version 0.9.2)
##
## configuration:
## QPALMA_C=10
## QPALMA_half_window_size=50000
## QPALMA_prb_offset=33
## QPALMA_iterations=50
## QPALMA_max_intron_length=50000
## QPALMA_max_qual=40
## QPALMA_num_supp_points=10
## QPALMA_enable_intron_length=False
## QPALMA_enable_splice_scores=True
## QPALMA_drop_unspliced=True
## QPALMA_contigs='I','II','III','IV','V','X','MtDNA'
##
h:      20      50000           20.000000,47.706646,...,50000.000000,   0.000000,0.000000,...,0.000000,
d:      0       1               0.000000,0.111111,...,1.000000, -0.645221,-0.644882,...,-0.642285,
a:      0       1               0.000000,0.111111,...,1.000000, -0.589631,-0.589288,...,-0.586655,
q[0]:   -5      40              -5.000000,0.000000,...,40.000000,       1.187513,1.187525,...,0.391153,
q[1]:   -5      40              -5.000000,0.000000,...,40.000000,       2.189317,2.189339,...,3.330389,
q[2]:   -5      40              -5.000000,0.000000,...,40.000000,       2.110812,2.110833,...,2.225520,

...

q[29]:  -5      40              -5.000000,0.000000,...,40.000000,       0.000000,0.000000,...,0.000000,
mmatrix:        6       1       0.000000,-1.500680,-1.171005,-1.325159,-1.234852,0.000000,
prb_offset:     1       1       33.000000,
```

Figure 1: Example of a *QPALMA* parameter file generated during the training phase of *QPALMA*. The first lines starting with ## characters describe the parameters used for generating this file. The following lines represent the piece-wise linear functions for intron length (h), donor splice sites (d), acceptor splice sites (a) and edit operations for match/mismatch/gap on read (q). Each piece-wise linear function is defined by the range of possible x-values (length for h, splice site predictions for d and a, quality for q) which is followed by the values for all support points (first, x-values separated by commas then, after a space, y-values separated by commas too). `mmatrix` defines fixed scores for a gap on DNA according to possible aligned base in read. Finally, `prb_offset` is the quality offset for determining the quality value from the ASCII quality character.

3.3 Training QPALMA on an example

QPALMA package includes examples to easily try the training procedure. The small example takes few minutes to complete, the larger example about two hours.

1. Go to examples directory:

```
user:~/qpalma-0.9.3$ cd examples
```

2. Choose the tiny example by typing:

```
user:~/qpalma-0.9.3/examples$ cd tiny_test
```

3. Execute the script `run_qpalma_training.sh` by running the following command:

```
user:~/qpalma-0.9.3/examples/tiny_test$ ./run_qpalma_training.sh
```

   It calls the script `train_qpalma_artificial_reads.sh` detailed above on the data stored in data directory. If everything ran well, the corresponding *QPALMA* parameter file parameters.qpalma is generated in the working directory.

4 Training QPALMA using Galaxy tools

QPALMA can also be trained via a Web service (`http://galaxy.fml.mpg.de/`), which is a customized version of the Galaxy frameworks [8, 9, 10]. There are no difference between these two ways of training *QPALMA* except that using galaxy saves the user from installing the software. Moreover, it is preferable to train *QPALMA* via galaxy if the rest of the study is carried out with galaxy as well. For further details about how to train *QPALMA* using Galaxy tools, please read *PALMapper* tutorial [2].

5 File Formats / Specifications

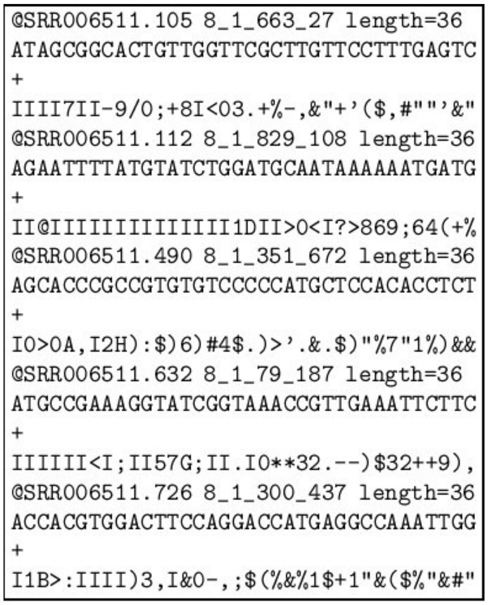
This section introduces all formats and conventions to be used when training *QPALMA*.

5.1 The read file

The read file is encoded in Sanger FASTQ format. A FASTQ file normally uses four lines per sequence. The first line begins with a '@' character and is followed by the read identifier and an optional description. The second line is the nucleotide sequence of the read. The third line begins with a '+' character and is optionally followed by the same identifier (and any description) again than in line 1. The last line encodes the quality values for the sequence in line 2, and must contain the same number of symbols as letters in the sequence. Sanger format encodes a quality score from 0 to 93 using ASCII 33 to 126. For a more detailed description of the FASTQ format consult: `http://en.wikipedia.org/wiki/FASTQ_format`. Figure 2 is an example of a Sanger FASTQ file.

5.2 Splice Scores

As mentioned before, the splice site scores can be generated using an appropriate tool such that mGene [5, 6] or ASP [7]. If you would like to use your own splice site predictions you can create files according to the Binary Signal Prediction Format (BSPF) described below:
For each canonical acceptor (*AG*) and donor site (*GT*/*GC*) *QPALMA* expects a score. The data is organized in files for each signal (acc/don) for each strand (+/-). The information on positions of canonical splice sites and the corresponding scores lies in separate files. Every chromosome or contig leads then to 8 files (acc/don, +/- and pos/score). The position and score files are raw binary files containing the ordered positions and the scores. The positions are stored as unsigned values



```
@SRR006511.105 8_1_663_27 length=36
ATAGCGGCACTGTTGGTTCGCTTGTTCCTTTGAGTC
+
IIII7II-9/0;+8I<03.+%-,&"+'($,#""'&"
@SRR006511.112 8_1_829_108 length=36
AGAATTTTATGTATCTGGATGCAATAAAAAATGATG
+
II@IIIIIIIIIIIII1DII>0<I?>869;64(+%
@SRR006511.490 8_1_351_672 length=36
AGCACCCGCCGTGTGTCCCCCATGCTCCACACCTCT
+
I0>0A,I2H):$)6)#4$.)>'.&.$)"%7"1%)&&
@SRR006511.632 8_1_79_187 length=36
ATGCCGAAAGGTATCGGTAAACCGTTGAAATTCTTC
+
IIIIII<I;II57G;II.I0**32.--)$32++9),
@SRR006511.726 8_1_300_437 length=36
ACCACGTGGACTTCCAGGACCATGAGGCCAAATTGG
+
I1B>:IIII)3,I&0-,;$(%&%1$+1"&($%"&#"
```

Figure 2: Example of a Sanger FASTQ file for input read data.

and the scores as floats. Note that you have to be careful when working in an inhomogeneous cluster environment (endianness, data type size). The positions are 1-based and the assignment of positions and their scores is as follows: The acceptor score positions are the positions right after the *AG* and the donor score positions are the positions right on the *G* of the *GT* or *GC*. For example:

...	3	4	5	6	7	8	9	10	11	...
...	w	g	t	x	y	z	a	g	v	...
...		0.2							0.3	...

We supply a script `asciispf_to_spf.py` (`tools/splicesites/` directory) for conversion of ASCII to binary files. You can use this script as a template to make your own scoring information files.

5.3 Training file format and internal representation

The read input files for *QPALMA* contain the read sequences with their quality as well as some information from the first mapping (see "creation of artificial reads" item in section 3.2). We use the BED format (cf. `http://genome.ucsc.edu/FAQ/FAQformat#format1`) as format for the input file needed for *QPALMA* training. The bed format can be described as follows: each line corresponds to one read and consists of 12 tab-separated entries, namely:

1. **chrom** - The name of the chromosome (e.g. chr3)
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.

4. **name** - Defines the name of the BED line.
5. **score** - A score between 0 and 1000 (not used in *QPALMA*)
6. **strand** - Defines the strand - either '+' or '-'.
7. **thickStart** - (not used in *QPALMA*)
8. **thickEnd** - (not used in *QPALMA*)
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0) (not used in *QPALMA*)
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to blockCount.
12. **blockStarts** - A comma-separated list of block starts. All of the blockStart positions should be calculated relative to chromStart. The number of items in this list should correspond to blockCount.

Strand specific direction means that *QPALMA* assumes that the reads are already in their true orientation and the qualities as well.
Alignment information means that an alignment of a read to a genomic sequence. A mismatch is encoded as $[AG]$ if A is on the sequence and G on the read side. A gap on the sequence (on read side, respectively) is denoted by $[-X]$ ($[X-]$, respectively) with $X \in A, C, G, T, N$.

5.4 The configuration file

In order to run *QPALMA* successfully you will need to create a configuration file, which includes all settings *QPALMA* needs to perform an analysis such as paths to files where the raw data exists as well as sequencing platform used, the number of cluster nodes to employ etc. Its values are in the form:

```
key = value
```

The use of the character # at the beginning of a line introduces a comment.
All valid parameters for *QPALMA* are summarized below. You can also look at the `galaxy/galaxy.conf` file which is a template file used during the execution of the script `qpalma_train.sh` (see *QPALMA* training item in section 3.2) to automatically generate the configuration file from environment variables.
The configuration parameters do not have to be in a particular order. In order to simplify explanations, we have grouped them according to their particular semantics.

5.4.1 General settings for QPALMA

- **perform_checks** - Enables some checks and debugging output.
- **platform** - IGA or 454 for Illumina Genome Analyzer or Roche's 454, respectively.

5.4.2 Data accession

The second group includes parameters needed for accessing the sequence data and the splice site predictions:

- **genome_dir** - The location of the genomic sequence files.
- **acceptor_scores_loc** - The location of the acceptor scores files. It must not appear if the user wishes to train *QPALMA* without splice site predictions or if splice site predictions are not available.
- **donor_scores_loc** - The location of the donor scores files. It must not appear if the user wishes to train *QPALMA* without splice site predictions or if splice site predictions are not available.
- **genome_file_fmt** - A format string describing how valid file names for the DNA sequences in flat file format (letters without white spaces) are generated. See description below.
- **splice_score_file_fmt** - A format string describing how valid file names for the splice site scores are generated.
- **allowed_fragments** - A list of the form "[1,2,4]" describing the valid file name numbers. See description below.
- **half_window_size** - Given an alignment seed position for a given read we cut out the area [seed_pos-half_window_size,seed_pos+half_window_size]. It should be equal to the parameter **max_intron_len** (see below).
- **output_format** - The output format can be blat, shore or mGene.
- **prb_offset** - This value will be substracted from the ASCII quality code (for example an 'h' would correspond to quality 40) to obtain the quality value (choose 33 for Sanger FASTQ files).

Format Strings The genomic sequence and the splice site scores file are accessed using the format strings given above. This works as follows: Suppose we have two chromosomes 1 and 4 we want to align to. The sequences are in flat files:

```
/home/user/genomic_sequence/chromosome_1.flat
/home/user/genomic_sequence/chromosome_4.flat
```

Then we set `genome_dir` to `/home/user/genomic_sequence`, `genome_file_fmt` to `chromosome_%d.flat` and `allowed_fragments` to [1,4].

5.4.3 Additional Configuration Parameters

Parameters needed for training are:

- **C** - this is a parameter trading off between loss term and regularization (similar to C in SVM training; set to 1, if unsure).
- **enable_quality_scores** - You can enable or disable the use of quality scores by setting this parameter to True or False. (This version has only been tested for True.)
- **enable_splice_scores** - You can enable or disable the use of splice site scores by setting this parameter to True or False. If this parameter is set to False, the parameters **donor_scores_loc** and **acceptor_scores_loc** should not appear.

- **enable_intron_length** - You can enable or disable the use of intron length scoring by setting this parameter to True or False.
- **drop_unspliced** - You can enable or disable the use of unspliced reads by setting this parameter to False or True.
- **max_unspliced_reads** - Maximal number of unspliced reads (if **drop_unspliced** is set to False).
- **optimizer** - Either one of CVXOPT or MOSEK.
- **numConstraintsPerRound** - How many constraints per optimizer call should be generated (if unsure set to 50).

5.4.4 PLiFs parameters

For training and prediction purposes you can set the number of support points for the piecewise linear functions of the scoring matrix. This effectively changes the total number of parameters of the model. That means that you can only predict with a parameter vector which was trained with the same settings. (We refer to piecewise linear functions as *plifs*.)

- **numAccSuppPoints** - The number of support points for the plif scoring the acceptor scores.
- **numDonSuppPoints** - The number of support points for the plif scoring the acceptor scores.
- **numLengthSuppPoints** - The number of support points for the plif scoring the acceptor scores.
- **min_intron_len** and **max_intron_len** - Set this to the range of minimum and maximum intron lengths.
- **min_qual** and **max_qual** - Set this to the range of possible quality values. (Illumina GA usually [-5,40]).
- **min_svm_score** and **max_svm_score** - As we work with normalized (in the range [0.0,1.0]) splice site scores this is usually 0.0 resp. 1.0.

5.4.5 “Low-level” parameters

The following last group of parameters includes “low-level” parameters. Just keep the default values of these parameters. Possible extensions of *QPALMA* may use them differently.

- **remove_duplicate_scores**
- **print_matrix**
- **matchmatrixCols** and **matchmatrixRows**
- **numQualPlifs**
- **numQualSuppPoints**
- **totalQualSuppPoints**
- **iter_steps**

6 Internet Resources

`http://www.fml.mpg.de/raetsch/suppl/qpalma`
QPALMA project web-page.
`http://www.fml.mpg.de/raetsch/suppl/palmapper`
PALMapper project web-page.
`http://www.fml.mpg.de/raetsch/suppl/mgene`
mGene project web-page.
`http://www.fml.mpg.de/raetsch/suppl/splice`
ASP project web-page.
`http://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/software/`
http server for downloading QPALMA.
`http://galaxy.fml.mpg.de/`
Galaxy server.

References

[1] F. De Bona, S. Ossowski, K. Schneeberger, and G. Rätsch. Optimal Spliced Alignment of Short Sequence Reads. *Bioinformatics*, 24(16):i174-i180, 2008.

[2] G. Jean, A. Kahles, V.T. Sreedharan, F. De Bona, and G. Rätsch. RNA-Seq Read Alignments with PALMapper. *Curr. Protoc. Bioinformatics*, 32:11.6.1-11.6.38, 2010.

[3] K. Schneeberger, J. Hagmann, S. Ossowski, N. Warthmann, S. Gesing, O. Kohlbacher, and D. Weigel. Simultaneous alignment of short reads against multiple genomes. *Genome Biology*, 10(9):R98, 2009.

[4] I. Tsochantaridis, T. Hofmann, T. Joachims and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. *Proceedings of the 16th International Conference on Machine Learning*, 2004.

[5] G. Schweikert, G. Zeller, A. Zien, J. Behr, C.-S. Ong, P. Philips, A. Bohlen, S. Sonnenburg, and G. Rätsch. mGene: A Novel Discriminative Gene Finding System. *Genome Research*, 19:2133-2143, 2009.

[6] G. Schweikert, J. Behr, A. Zien, G. Zeller, S. Sonnenburg, and G. Rätsch. mGene.web: a web service for accurate computational gene finding. *Nucleic Acids Research*, 37(Suppl. 2):W312W316, 2009.

[7] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S7, 2007.

[8] D. Blankenberg, J. Taylor, I. Schenck, J. He, Y. Zhang, M. Ghent, N. Veeraraghavan, I. Albert, W. Miller, K. Makova, R. Hardison, and A. Nekrutenko. A framework for collaborative analysis of ENCODE data: making largescale analyses biologist-friendly. *Genome Research*, 17(6):960964, 2007.

[9] J. Taylor, I. Schenck, D. Blankenberg, and A. Nekrutenko. Using galaxy to perform large-scale interactive data analyses. *Curr. Protoc. Bioinformatics*, 19:10.5.1-10.5.25, 2007.

[10] D. Blankenberg, G. Von Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, J. Taylor. Galaxy: A WebBased Genome Analysis Tool for Experimentalists. *Curr. Protoc. Molecular Biology*, 89:19.10.1-19.10.21, 2010.