EuGène: an open gene finder for eukaryotes

Philipe Bardou Marie-Josée Cros Sylvain Foissac Jérome Gouzy
Annick Moisan Céline Noirot
Thomas Schiex
Applied Mathematics and Computer Science Dept.
INRA Toulouse, France

December 22, 2006

Contents

Overview

EUGÈNE is a sophisticated open gene finder for eukaryotic organisms. It has been developped thanks to funding by INRA (permanent scientists and engineers), Génoplante and the french ministry of research (with one PhD student). It generates text, HTML and graphical outputs.

EUGÈNE uses a graph based model to predict genes that covers both HMM based or more complex Bayesian net based or Conditional Markov Field probabilistic predictions. The model is fixed but complex (with 43 different states) and covers Exon, Intron, UTR, UTR introns... each with a possible explicit distribution on length. The prediction itself relies on an optimal linear time and space algorithm for prediction.

Even if the gene model is fixed, the sources of information taken into account be EUGÈNE for prediction are extremely varied and can be easily extended by creating so-called plugins. Currently, EUGÈNE can use around 30 different plugins integrating statistical information (Markov models at DNA or amino acid level, WAM, Support Vector machine based signal prediction...), similarity information (Est, cDNA, proteins) and homology (exon conservation). It can also integrate predictions from other gene predictors if needed.

In order to integrate all this information, EUGÈNE does not use maximum likelihood estimation for all parameters but parameters optimized by maximum of prediction quality on expertized data sets (minimizing empirical risk on a given dataset).

The software called `eugene` is written in C++ and is distributed under the artistic license.

Chapter 1

Quick Start

1.1 Annoting a sequence

Here is a small example based on the `SYNO_ARATH.fasta` sequence. For reference information on the software, see chapter 2.

In order to first collect information on the sequence (splice sites, translation start predictions…) we will have to use the `getsites4eugene.pl` script. This script directly queries the Netgene2, SPlicePredictor and NetStart web servers. Alternatively, if you have installed these programs locally, you can use the `lgetsites4eugene.pl` script (you must modify it and indicate the paths to the executables).

```
> ./getsites4eugene.pl Sequences/SYNO_ARATH.fasta
started on sam dec 7 13:44:35 CET 2002

processing Sequences/SYNO_ARATH.fasta

NetStart [2*1 request(s)]: F1..R1..done
NetGene2 [1   request(s)]: 1..FR..done
SplicePredictor: done
finished on sam dec 7 13:47:14 CET 2002
```

The script creates the files that contains information about the sequence in the same directory as the fasta file itself. The extensions used are `.splices` for NetGene2, `.spliceP` for SplicePredictor, `.starts` for NetStart (in each case, a `R` is added for the reverse strand).

We are now ready to use EuGène on this sequence. Because the sequence lacks context around the CDS of the gene, we inform EuGène that the prediction should start and end in intergenic mode using the `-s` flag. This behavior can also be controlled by all the `Prior` parameters in the program parameter file (see section 2.1).

EuGène produce two kind of output: textual and graphical. To manage this outputs several options could be use. Two of them (more details see chapter 2):

- `-p a|d|g|h|l|s|o`: if we want, we may ask for multiple textual output. For example an HTML output and an GFF output using the `-phg` flag. Two files will be created '`SYNO_ARATH.html`' and '`SYNO_ARATH.gff`'. `-po` allows to print prediction on stdout.
- `-g`: activates the graphical output (with `-ph` flag `-g` is on).

```
>../src/eugene -s -po Sequences/SYNO_ARATH.fasta
----------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
```

```
----------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM......done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2............done
Reading splice site file (NetGene2)...........forward, reverse done
Loading Sensor.SPred..........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Optimal path length = 4869.7709

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

SYNO_.1.0        Utr5    +         5      28       24     NA      NA      NA      NA     0.0
SYNO_.1.1        Init    +        29     421      393     +1      +2      28     422     0.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1     513     583     0.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3     698     810     0.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2     913    1019     0.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2    1270    1409     0.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1    1521    1603     0.0
SYNO_.1.7        Intr    +      1694    1801      108     +1      +2    1693    1802     0.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2    1852    1922     0.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1    2013    2089     0.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3    2180    2265     0.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2    2359    2447     0.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3    2711    2883     0.0
SYNO_.1.13        Intr    +      2966    3092      127     +1      +2    2965    3093     0.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2    3188    3305     0.0
SYNO_.1.0        Utr3    +      3305    3362       58     NA      NA      NA      NA     0.0

----------------------------------------------------------------------
```

1.2 Using transcribed sequences

If you want to exploit similarities with cDNA/EST sequences, you have to inform EUGÈNE of existing similarities. These similarities should be available in a file with the `.est` extension. The format of this file is described in the `Est` plugin section 2.3.3.2. It can easily be created from an existing FASTA databank of EST and cDNA using a patched version of `sim4`. The patch is provided with EUGÈNE.

```
> sim4 Sequences/SYNO_ARATH.fasta cDNA A=6 > seqs/SYNO_ARATH.fasta.est
```

With an old dbEST databank completed with the cDNA databank PlantGene, we get the following file:

```
> cat Sequences/SYNO_ARATH.fasta.est
    32     421 1844 0 0 ATAJ644    1  390
   514     582 1844 0 0 ATAJ644  391  459
   699     809 1844 0 0 ATAJ644  460  570
   914    1018 1844 0 0 ATAJ644  571  675
  1271    1408 1844 0 0 ATAJ644  676  813
  1522    1602 1844 0 0 ATAJ644  814  894
  1694    1771 1844 0 0 ATAJ644  895  972
  1853    1921 1844 0 0 ATAJ644  973 1041
  2014    2088 1844 0 0 ATAJ644 1042 1116
  2181    2264 1844 0 0 ATAJ644 1117 1200
  2360    2446 1844 0 0 ATAJ644 1201 1287
  2712    2882 1844 0 0 ATAJ644 1288 1458
  2966    3092 1844 0 0 ATAJ644 1459 1585
  3189    3447 1844 0 0 ATAJ644 1586 1844
    32     375 347 0 0 N97006    1  347
  3099    3379 297 0 0 AV525988   51  347
  3071    3092 256 0 1 AI994358    1   22
  3189    3421 256 0 1 AI994358   23  256
   658     672 61 0 1 AV521563    1   14
   765     813 61 0 1 AV521563   15   61
```

We can now ask EuGène for a new prediction, including this new evidence using the `-d` flag (equivalently, the `Est` plugin can be activated by modifying EuGène parameter file). When evidence from transcribed sequences is available, EuGène will automatically report in the last column of its output the percentage of bases of the element (exon, UTR...) which is consistent with the available evidence. Here, the gene is almost completely covered by the available transcribed sequences. The `Est` plugin also mentions if transcribed sequences are rejected and why. The information from two transcribed sequences is rejected. The first one because no splice site has been found near one of the intron border detected by the EST, another one because it was inconsistent with a sequence considered as more reliable.

```
>../src/eugene -s -po -d Sequences/SYNO_ARATH.fasta
---------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
---------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM.....done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2...........done
Reading splice site file (NetGene2)..........forward, reverse done
Loading Sensor.SPred.........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart)................forward, reverse done
Loading Sensor.Transcript.....done
Loading Sensor.Est...........done
Reading cDNA hits...........3 sequences read
Optimal path length = 4885.5924

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

SYNO_.1.0        Utr5    +         5      28       24     NA      NA      NA      NA     0.0
SYNO_.1.1        Init    +        29     421      393     +1      +2      28     422    99.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1     513     583   100.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3     698     810   100.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2     913    1019   100.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2    1270    1409   100.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1    1521    1603   100.0
SYNO_.1.7        Intr    +      1694    1771       78     +1      +2    1693    1772   100.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2    1852    1922   100.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1    2013    2089   100.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3    2180    2265   100.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2    2359    2447   100.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3    2711    2883   100.0
SYNO_.1.13        Intr    +      2966    3092      127     +1      +2    2965    3093   100.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2    3188    3305   100.0
SYNO_.1.0        Utr3    +      3305    3441      137     NA      NA      NA      NA   100.0

---------------------------------------------------------------------
```

An additional postprocessing can be requested to the plugin using the `-E` flag. For each gene predicted, the plugin will analyze each transcribed sequence matching the gene and report its consistency with the prediction in '`SYNO_ARATH.misc_info`' file.

```
>../src/eugene -s -po -d -E Sequences/SYNO_ARATH.fasta
---------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
---------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM.....done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2...........done
Reading splice site file (NetGene2)..........forward, reverse done
Loading Sensor.SPred.........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
```

```
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Loading Sensor.Est............done
Reading cDNA hits............3 sequences read
Optimal path length = 4885.5924

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

SYNO_.1.0        Utr5    +         5      28       24     NA      NA      NA      NA     0.0
SYNO_.1.1        Init    +        29     421      393     +1      +2      28     422    99.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1     513     583   100.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3     698     810   100.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2     913    1019   100.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2    1270    1409   100.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1    1521    1603   100.0
SYNO_.1.7        Intr    +      1694    1771       78     +1      +2    1693    1772   100.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2    1852    1922   100.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1    2013    2089   100.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3    2180    2265   100.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2    2359    2447   100.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3    2711    2883   100.0
SYNO_.1.13        Intr    +      2966    3092      127     +1      +2    2965    3093   100.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2    3188    3305   100.0
SYNO_.1.0        Utr3    +      3305    3441      137     NA      NA      NA      NA   100.0

---------------------------------------------------------------------
> cat SYNO_ARATH.misc_info
#=========================================#
#=           Gene informations           =#
#=========================================#
SYNO_.1          EuGene_misc       CDS        29        3304        1704        +        .        Full        29..421,51
SYNO_.1          EuGene_misc       Gene        5        3441        3437        +        .        Full        5..28:29..
#=========================================#
#=             Est evidences             =#
#=========================================#
SYNO_.1.1        EuGene_cDNA       Exon        29        421        391        +        .        99       ATAJ644(99,1)
SYNO_.1.2        EuGene_cDNA       Exon        514        582        69        +        .        100       ATAJ644(100,
SYNO_.1.3        EuGene_cDNA       Exon        699        809        111        +        .        100       ATAJ644(100
SYNO_.1.4        EuGene_cDNA       Exon        914        1018        105        +        .        100       ATAJ644(10
SYNO_.1.5        EuGene_cDNA       Exon        1271        1408        138        +        .        100       ATAJ644(1
SYNO_.1.6        EuGene_cDNA       Exon        1522        1602        81        +        .        100       ATAJ644(10
SYNO_.1.7        EuGene_cDNA       Exon        1694        1771        78        +        .        100       ATAJ644(10
SYNO_.1.8        EuGene_cDNA       Exon        1853        1921        69        +        .        100       ATAJ644(10
SYNO_.1.9        EuGene_cDNA       Exon        2014        2088        75        +        .        100       ATAJ644(10
SYNO_.1.10        EuGene_cDNA       Exon        2181        2264        84        +        .        100       ATAJ644(1
SYNO_.1.11        EuGene_cDNA       Exon        2360        2446        87        +        .        100       ATAJ644(1
SYNO_.1.12        EuGene_cDNA       Exon        2712        2882        171        +        .        100       ATAJ644(
SYNO_.1.13        EuGene_cDNA       Exon        2966        3092        127        +        .        100       ATAJ644(
SYNO_.1.14        EuGene_cDNA       Exon        3189        3304        116        +        .        100       ATAJ644(
SYNO_.1.0        EuGene_cDNA       UTR3        3305        3441        137        +        .        100       ATAJ644(1
SYNO_.1          EuGene_cDNA       CDS        29        3304        3273        +        .        99       ATAJ644(99,1
SYNO_.1          EuGene_cDNA       Gene        5        3441        3410        +        .        99       ATAJ644(99,1
```

1.3 Using protein similarities

If one wants also to exploit similarities with homologous proteins, a similar file format can be used (see the corresponding plugin). The plugin can analyze similarities from several databases, each being associated with a specific "level". For each level, a confidence is defined in EUGÈNE's parameter file. Usually, 3 databases are used: SwissProt, PIR and TrEMBL (from the highest confidence to the lowest). Each collection of similarity is stored in a file with an extension `.blast` followed by the level of the database (`.blast0`, `.blast1`, ..., `.blast9`). The script used create these files from the output of NCBI-BLASTX is copyrighted and is therefore not distributed with EUGÈNE. It is not difficult to design another one. Here is an extract from `SYNO_ARATH.fasta.blast0`:

```
2820 2861 36 3e-08 +3  sp_O07683_SYD_HALSA; 335 348
2972 3088 41 3e-08 +2  sp_O07683_SYD_HALSA; 359 397
3185 3298 113 3e-08 +2  sp_O07683_SYD_HALSA; 398 435
353 418 45 2e-13 +2  sp_O24822_SYD_HALVO; 13 34
1850 1915 67 2e-13 +2  sp_O24822_SYD_HALVO; 202 223
2775 2858 72 2e-13 +3  sp_O24822_SYD_HALVO; 318 345
3191 3280 104 2e-13 +2  sp_O24822_SYD_HALVO; 397 426
```

```
353 418 51 7e-12 +2  sp_O26328_SYD_METTH; 21 42
1271 1414 70 7e-12 +2  sp_O26328_SYD_METTH; 141 188
1850 1954 62 7e-12 +2  sp_O26328_SYD_METTH; 210 244
3191 3280 93 7e-12 +2  sp_O26328_SYD_METTH; 401 430
```

To exploit this information, the `-b` flag must be used, optionally followed by the set of levels to be exploited ("`012`" means level 0, 1 and 2). We start EuGène and ask for both EST and proteic similarities analysis. We again enforce the use of an intergenic mode on the beginning and end of the sequence. And similar to the EST, an additional postprocessing can be requested to the plugin using the `-B` flag.

```
>../src/eugene -s -po -d -b012 -B Sequences/SYNO_ARATH.fasta
---------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
---------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM......done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.BlastX.........done
Reading BlastX data, level...0 1 2 done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2............done
Reading splice site file (NetGene2)...........forward, reverse done
Loading Sensor.SPred..........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Loading Sensor.Est............done
Reading cDNA hits............3 sequences read
Optimal path length = 4907.0231

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

SYNO_.1.0        Utr5    +         5      28       24     NA      NA      NA      NA     0.0
SYNO_.1.1        Init    +        29     421      393     +1      +2      28     422    99.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1     513     583   100.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3     698     810   100.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2     913    1019   100.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2    1270    1409   100.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1    1521    1603   100.0
SYNO_.1.7        Intr    +      1694    1771       78     +1      +2    1693    1772   100.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2    1852    1922   100.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1    2013    2089   100.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3    2180    2265   100.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2    2359    2447   100.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3    2711    2883   100.0
SYNO_.1.13        Intr    +      2966    3092      127     +1      +2    2965    3093   100.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2    3188    3305   100.0
SYNO_.1.0        Utr3    +      3305    3441      137     NA      NA      NA      NA   100.0

---------------------------------------------------------------------
> cat SYNO_ARATH.misc_info
#========================================#
#=          Gene informations           =#
#========================================#
SYNO_.1         EuGene_misc       CDS        29        3304        1704        +        .        Full        29..421,51
SYNO_.1         EuGene_misc       Gene        5        3441        3437        +        .        Full        5..28:29..
#========================================#
#=          Protein evidences           =#
#========================================#
SYNO_.1.1       EuGene_prot       Exon       29        421        393        +        .        100        sp_O23573_O2
SYNO_.1.2       EuGene_prot       Exon       514        582        69        +        .        100        sp_O23573_O2
SYNO_.1.3       EuGene_prot       Exon       699        809        72        +        .        64        sp_O23573_O23
SYNO_.1.4       EuGene_prot       Exon       914        1018        102        +        .        97        sp_O23573_O
SYNO_.1.5       EuGene_prot       Exon       1271        1408        138        +        .        100        sp_O23573
SYNO_.1.6       EuGene_prot       Exon       1522        1602        81        +        .        100        sp_O23573_
SYNO_.1.7       EuGene_prot       Exon       1694        1771        78        +        .        100        sp_P43829_
SYNO_.1.8       EuGene_prot       Exon       1853        1921        69        +        .        100        sp_P54263_
SYNO_.1.9       EuGene_prot       Exon       2014        2088        75        +        .        100        sp_P17242_
SYNO_.1.10       EuGene_prot       Exon       2181        2264        81        +        .        96        sp_O96198_
SYNO_.1.11       EuGene_prot       Exon       2360        2446        87        +        .        100        sp_P43829
SYNO_.1.12       EuGene_prot       Exon       2712        2882        171        +        .        100        sp_O2357
SYNO_.1.13       EuGene_prot       Exon       2966        3092        127        +        .        100        sp_O2357
SYNO_.1.14       EuGene_prot       Exon       3189        3304        110        +        .        94        sp_P5227
```

```
SYNO_.1          EuGene_prot        CDS        29        3304        1653        +        .        97        sp_O23573_O2
```

1.4 Injecting further information

Additional user information on the intronic, exonic, intergenic aspect of the sequence or on specific signals can be injected using a specific simple langage (see the the `User` plugin, section 2.3.4.4). This plugin is activated using the `-U` flag. The plugin will read the file with extension `.user`. For example, imagine I know that the 5'UTR of the gene extends from position 2 to 28. I can inject this information in EUGÈNE as follows (the `10.0` gives a strong influence to this information, see the plugin description):

```
utr5 f [2..28] 10.0
```

If I use EUGÈNE with the `-U` flag (and even without `-s`), we get the following prediction:

```
>../src/eugene -po -U Sequences/SYNO_ARATH.fasta
---------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
---------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM......done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2............done
Reading splice site file (NetGene2)...........forward, reverse done
Loading Sensor.SPred..........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Loading Sensor.User...........done
Reading user data............done
Optimal path length = 4597.4208

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

SYNO_.1.0        Utr5    +         1      28       28     NA      NA      NA      NA     0.0
SYNO_.1.1        Init    +        29     421      393     +1      +2      28     422     0.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1     513     583     0.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3     698     810     0.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2     913    1019     0.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2    1270    1409     0.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1    1521    1603     0.0
SYNO_.1.7        Intr    +      1694    1801      108     +1      +2    1693    1802     0.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2    1852    1922     0.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1    2013    2089     0.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3    2180    2265     0.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2    2359    2447     0.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3    2711    2883     0.0
SYNO_.1.13        Intr    +      2966    3092      127     +1      +2    2965    3093     0.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2    3188    3305     0.0
SYNO_.1.0        Utr3    +      3305    3362       58     NA      NA      NA      NA     0.0

---------------------------------------------------------------------
```

This type of plugin allows to explore alternative predictions. Eg. one may wonder what is the optimal prediction if the splice site at position `2966` is not used. It suffices to forbid this splice site using the following user file:

```
acceptor f 2966 0.0
```

and to ask for a new prediction.

```
>../src/eugene -po -U Sequences/SYNO_ARATH.fasta
---------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
---------------------------------------------------------------------
Loading sequence...SYNO_ARATH, 3699 bases read, GC Proportion = 38.8%
Loading Sensor.MarkovIMM......done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2............done
Reading splice site file (NetGene2)...........forward, reverse done
Loading Sensor.SPred..........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Loading Sensor.User...........done
Reading user data............done
Optimal path length = 4873.3548

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac       Do       Pr

SYNO_.1.1        Intr    +         1     421      421   Unk.      +2        0     422     0.0
SYNO_.1.2        Intr    +       514     582       69     +1      +1      513     583     0.0
SYNO_.1.3        Intr    +       699     809      111     +1      +3      698     810     0.0
SYNO_.1.4        Intr    +       914    1018      105     +1      +2      913    1019     0.0
SYNO_.1.5        Intr    +      1271    1408      138     +1      +2     1270    1409     0.0
SYNO_.1.6        Intr    +      1522    1602       81     +1      +1     1521    1603     0.0
SYNO_.1.7        Intr    +      1694    1801      108     +1      +2     1693    1802     0.0
SYNO_.1.8        Intr    +      1853    1921       69     +1      +2     1852    1922     0.0
SYNO_.1.9        Intr    +      2014    2088       75     +1      +1     2013    2089     0.0
SYNO_.1.10        Intr    +      2181    2264       84     +1      +3     2180    2265     0.0
SYNO_.1.11        Intr    +      2360    2446       87     +1      +2     2359    2447     0.0
SYNO_.1.12        Intr    +      2712    2882      171     +1      +3     2711    2883     0.0
SYNO_.1.13        Intr    +      2924    3092      169     +1      +2     2923    3093     0.0
SYNO_.1.14        Term    +      3189    3304      116     +2      +2     3188    3305     0.0
SYNO_.1.0        Utr3    +      3305    3362       58     NA      NA       NA      NA     0.0

---------------------------------------------------------------------
```

Other plugins are described in the reference section of this document.

Chapter 2

Reference documentation

To be executed, EuGène needs at least one file: this is the so-called parameter file. EuGène behavior is entirely controlled by a set of parameters whose default values are available in this file. These default values can be altered by editing this file or for some values through flags in the command line (such as `-d`, see the quick start chapter). Command line flags override any value in the parameter file. The name of the parameter file that EuGène seeks is obtained by adding the suffix "`.par`" to the name of the EuGène command itself. As it is distributed, EuGène command's name is `eugene` and accordingly the parameter file is `eugene.par`. If at some point you want to use several different parameter files, you can simply use symbolic links to the `eugene` binary executable. Using a symbolic link, with a specific name to call EuGène will enable you to load a different parameter file whose name is derived from the symbolic link name by ading '`'.par`" The parameter file is first sought in the local directory. If this fails, the value of the environment variable `EUGENEDIR` is used as a second possible path.

EuGène gathers all informations on the FASTA sequences through so-called "plugins" also called "sensors". A plugin is a small software component that can be dynamically loaded and that can inform EuGène about likely exonic, intronic, utr, intergenic regions and about signals in the sequence (either splice sites, translation starts and stops, transcription starts and stops and possible frameshifts). Plugins can typically embody Markov models (that characterize exonic, intronic... regions) or splice site detectors or others. Available sensors are stored in the `PLUGINS` directory and are dynamically loaded by EuGène according to the parameters.

The typical call to EuGène is:

```
eugene <fasta files>
```

where each FASTA file contains one single DNA sequence. In this case, the first action of EuGène is to seek and load the parameter file. All the parameters in this file are either used by EuGène or by the plugins. Each plugin may have its own parameters. The following section describes all the parameters used by EuGène. Information about the parameters used by plugins is provided in each plugin section (see section 2.3).

2.1 EuGène's general parameters

Here is a list of all the parameters not related to a plugin which control EuGène's behavior. When a command line flag exists that can modify the corresponding parameter, it is indicated. All the parameters that control EuGènés behavior are available in the parameter file. This file has a relatively strict formatting. Each line can either be a comment line (the first character in the line must be a `#`) or a parameter definition. Empty lines are not allowed. A parameter definition is composed of two strings of character. The first one is the name of the parameter, the second is its value. Everything is case sensitive. The definition order is not important.

- EuGene.version: specifies the EUGÈNE version. After having load the parameter file, EUGÈNE checks that the parameter file version is consistent with the executable version.
- EuGene.organism: name of the considered organism.
- EuGene.PluginsDir: specify the location of the directory where plugins could be found (".so" files, sensor specific models files); that is the path to the PLUGINS directory.
- EuGene.sloppy: in the default (non-sloppy) mode, EUGÈNE will stop and abort if some needed parameters in the parameter file is missing. If the parameter is set to 1 then a simple warning is emitted. Not advised unless you know what you do.
- EuGene.ExonPrior, EuGene.IntronPrior, EuGene.InterPrior, EuGene.FivePrimePrior, EuGene.ThreePrimePrior: prior on the initial/final state of prediction. The "-s" command line flag can override these priors by setting all the non intergenic priors to 0.0. This forces EUGÈNE to start and and its prediction in intergenic mode.
- EuGene.InitExDist, EuGene.IntrExDist, EuGene.TermExDist, EuGene.SnglExDist, EuGene.IntronDist, EuGene.InterDist, EuGene.5PrimeDist, EuGene.3PrimeDist: EUGÈNE can use explicit penalty distributions on the length of the elements predicted. This can be an initial exon, and intermediary exon, a terminal exon, a single exon gene, an intron an intergenic region, a 5' UTR region or a 3' UTR region. Each parameter specifies the filename of an explicit penalty distribution file (see the following section).
- EuGene.SplicedStopPen: indicates the penalty for predicting genes containing in-frame spliced STOPs. This is basically set to an infinite value in order to avoid prediction containing spliced STOPs but setting this to 0.0 can be useful for pseudo-gene prediction...
- Output.truncate: in the text output, each gene element predicted if prefixed by the FASTA sequence id (or the filename if no FASTA id is available). This is truncated to the number of caracters indicated. If set to 0 (or FALSE), the full id is used.
- Output.MinCDSLen: any predicted gene whose CDS length in number of nucleotides is lower than this is filtered out from the output.
- Output.UTRtrim: EuGene is natively capable of predicting UTR. If desired however, the UTR prediction of EuGene can be trimmed to be exactly consistent with the transcript evidence available as provided by the Est plugin. If no EST evidence is available, this means that all UTR predictions will be removed from the output.
- Output.stepid: in the text output, step for numbering genes.
- Output.graph: if set, requests graphical PNG output. This can also be set using the -g command line flag. The PNG filename is composed by the seq name (w/o the .fasta suffix) completed by the number of the figure + .png extension (possibly, start/end positions will be inserted too if -u/-v is used).
- Output.resx, Output.resy: controls the horizontal and vertical resolution of the PNG images generated by EUGÈNE.
- Output.gfrom, Output.gto: respectively controls which part of the sequence is to be plotted (eg. for zooming). The default value for both is -1 which corresponds to the whole sequence. These parameters can also be set using the -u and -v. flags
- Output.glen: controls the number of nucleotides that will appear on a single image. The value -1 corresponds to a default adaptative mechanism which plots min (6000,length to visualize). The "length to visualize" is computed from the value given to Output.gfrom and Output.gto.

- `Output.golap`: controls how successives PNG images overlap. It must be set to the number of overlapping nucleotides between 2 successives PNG images. Default is -1 which heuristically determines this based on resolution and number of nuc. per image. This parameter can also be set using the `-c` command line flag.
- `Output.normopt`: indicates the way the score are normalized accross the possibles states (phase 1, 2, 3, -1, -2, -3, introns and intergenic states).
  - 0: no normalization
  - 1: normalize accross all states
  - 2: normalize each coding phase w.r.t. to the non coding score only.

  Default is 1. Does not affect prediction, only graphical output.
- `Output.window`: sets the half-size of the smoothing window used to plot the scores. Default is 48. This does not affect prediction, only graphical output. It can be set using the `-w` command line flag.
- `Output.intron`: allows to print introns in the textual output. Default is 0 (no introns).
- `Output.format`: controls the format of the textual outpout. May be `o` (stdout), `d` (detailed), `l` (long), `s` (short), `h` (html), `g` (gff) or `a` (araset format). Default is `l`. This can be overrided using the `-p` command line flag. `o`: print the prediction on stdout using the same format than `l`. All the others print the prediction in files which name are composed by the name of the sequence file (w/o the extension .fasta, .tfa, .fsa or .txt) completed by `.egn.debug (d)`, `.egn (l)`, `.egn.short (s)`, `.html (h)`, `.gff (g)`, `.gff3 (g)` or `.egn.ara (a)`. Multiple format can be selected (`ohg` for example). When GFF is requested, both GFF1 and GFF3 are produced.
- `Output.offset`: allows to offset the nucleotide position of the prediction. That is, the prediction for nucleotide at position i of the given sequence is printed as nucleotide $i+$ the offset. Useful to perform prediction on an extracted sequence without loosing the original position. Can also be set using the `-o` command line flag.
- `Output.Prefix`: indicates the directory where all non stderr/stdout output (eg. PNG images, HTML and GFF files...) should go. Default is the current directory.

Specification of explicit penalty distributions on length

As in semi-Markov models, EUGÈNE uses explicit distribution of penalties on the length of all predicted elements. The dynamic programming inside EUGÈNE garantees that EUGÈNE will run in linear time and space in the length of the sequence in all cases.

The distributions handled by EUGÈNE are made of 3 components. First, there is a region of forbidden length (minimum length), then a region with an arbitrary penalty distribution, then a region with a linear variation of the penalty. From a probabilitic point of view, this means an exponential tail.

Although EUGÈNE is linear in time in the sequence length, it is also typically linear in time in the sum of the size of the two first regions. For the moment, all existing EUGÈNE instances use explicit distributions with an empty arbitrary region (the distribution is just a minimum length followed by an exponential tail).

Explicit distributions must be specified in distribution files. Each line in a distribution file contains a length and a penalty. The first length used specifies the minimum allowed length. Then each line specifies a point of the explicit distribution. Linear interpolation is used between points. Then the last length used specifies the start of the linear tail. The last slope used becomes the slop of the linear tail.

A typical distribution file is given below:

```
3 0.0
4 2.0
6 4.0
```

It specifies a minimum length of 3. We then have an explicit distribution region with penalty 0.0 for 3, 2.0 for 4, 3.0 for 5 (linear interpolation), then 4.0 at 6. As this is the last point and the slope is 1, the rest of the distribution will be linear with slope 1.

2.2 Splice variant prediction

Since version 3.4, EUGÈNE allows to predict splice variants based purely on experimental data (alternative transcripts observed through EST). The feature is still experimental and is activated using the `-a` flag or equivalently by setting the parameter `AltEst.use` to 1 or TRUE.

In this case, EUGÈNE will look for a file with the same name as the sequence file and with a suffix '`.alt.est`'. This file has the same format as the '`.est`' used by the Est pugin (see later) and contains information about genomic region with high quality similarity with EST. The spliced alignment algorithm used ti create this file should be of high quality, with clear exon-intron frontiers associated with splice sites (use for example GeneSeqer or faster GenomeThreader).

EUGÈNE will analyze these EST and look for pairs of EST which are inconsistent one with the other (there is one nucleotide mapped to an exon by one which is mapped to an intron/gap by the other). Each element of such a pair will be used to try to produce a prediction that follows the EST structure. If the prediction is different from the optimal prediction, the genen variant structure will be also output.

This feature is controlled by a number of parameters with the '`AltEst`' prefix in the parameter file. The only parameters that you could change are the parametrs regarding length thresholds, used for filtering (`AltEst.maxEstLength`, `AltEst.minEstLength`, `AltEst.maxIn`, `AltEst.minIn`, `AltEst.maxEx` and `AltEst.minEx` which speak for themselves.

Every alignement is also "trimmed" by an amount of `AltEst.exonucleasicLength` on the first and last hit to account for possible spurious short matches. If these hits are shorter than this amount, they are removed from the available data.

Using the sequence `At5g18830.fasta.genomicAJ011613.fasta` and the associated information found in the `doc/Sequences/` directory, we can test this as follows:

```
>../src/eugene -s -a -po Sequences/At5g18830.fasta.genomicAJ011613.fasta
-------------------------------------------------------------------
EuGene rel. 3.4 - Arabidopsis -
Parameters file /home/tschiex/Desktop/eugene/cfg/eugene.par loaded.
-------------------------------------------------------------------
Loading sequence...gi|5931636|emb|AJ011613.1|ATH011613, 4920 bases read, GC Proportion = 40.0%
Loading Sensor.MarkovIMM......done
Reading IMM... 1 2 3 4 5 6 7 ...done
Loading Sensor.EuStop.........done
Loading Sensor.IfElse.........done
Loading Sensor.NG2............done
Reading splice site file (NetGene2)...........forward, reverse done
Loading Sensor.SPred..........done
Reading splice site file (Splice Predictor)...forward, reverse done
Loading Sensor.MarkovConst....done
Loading Sensor.NStart.........done
Reading start file (NetStart).................forward, reverse done
Loading Sensor.Transcript.....done
Optimal path length = 6501.0711

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

Reading alt. spl. evidence... 11 read, 9 removed (0 incl., 9 unsp., 0 no alt.spl., 0 len.), 1 inc. pairs, 2 kept ... done
Optimal path length = 6505.9537

    Seq         Type    S       Lend    Rend   Length  Phase   Frame      Ac      Do      Pr

gi|59.1.0        Utr5    +         83     118       36     NA      NA      NA      NA     0.0
gi|59.1.1        Init    +        119     669      551     +1      +2     118     670     0.0
gi|59.1.2        Intr    +       1066    1238      173     +3      +2    1065    1239     0.0
gi|59.1.3        Intr    +       1390    1616      227     +2      +3    1389    1617     0.0
gi|59.1.4        Intr    +       1690    1764       75     +1      +1    1689    1765     0.0
gi|59.1.5        Intr    +       1867    1974      108     +1      +1    1866    1975     0.0
```

```
gi|59.1.6         Intr    +       2291    2408      118      +1       +2     2290     2409      0.0
gi|59.1.7         Intr    +       2525    2660      136      +2       +1     2524     2661      0.0
gi|59.1.8         Intr    +       2886    3039      154      +3       +1     2885     3040      0.0
gi|59.1.9         Intr    +       3114    3674      561      +1       +3     3113     3675      0.0
gi|59.1.10         Term    +       3928    4230      303      +1       +1     3927     4231      0.0
gi|59.1.0         Utr3    +       4231    4268      38       NA       NA      NA       NA      0.0

gi|59.1a.0         Utr5    +         83     118       36      NA       NA       NA       NA      0.0
gi|59.1a.1         Init    +        119     669      551      +1       +2      118      670      0.0
gi|59.1a.2         Intr    +       1066    1238      173      +3       +2     1065     1239      0.0
gi|59.1a.3         Intr    +       1390    1616      227      +2       +3     1389     1617      0.0
gi|59.1a.4         Intr    +       1690    1764       75      +1       +1     1689     1765      0.0
gi|59.1a.5         Intr    +       1867    1974      108      +1       +1     1866     1975      0.0
gi|59.1a.6         Intr    +       2291    2408      118      +1       +2     2290     2409      0.0
gi|59.1a.7         Intr    +       2525    2660      136      +2       +1     2524     2661      0.0
gi|59.1a.8         Intr    +       2886    3039      154      +3       +1     2885     3040      0.0
gi|59.1a.9         Intr    +       3114    3596      483      +1       +3     3113     3597      0.0
gi|59.1a.10         Term    +       3928    4230      303      +1       +1     3927     4231      0.0
gi|59.1a.0         Utr3    +       4231    4353      123      NA       NA       NA       NA      0.0

----------------------------------------------------------------------
```

We can see that two predictions are produced for the same region. In this case, it is just one alternative splice site that has been used for the exon number 8 in the gene.

2.3 Plugins

Plugins are small software components that can be dynamically loaded by EUGÈNE. Although it is completely transparent to the end-user, every plugin loaded by EUGÈNE must be written in C++ and be a subclass of the Sensor class. This class provides essentially four methods:

- constructor: when instanciated, a plugin receives an instance number (specified in the parameter file) and a DNA sequence (instance of the `DNASeq` class). The instance number allows to load several identical plugins using different parameters. A plugin with a parameter `X` and instance number `n` will fetch parameter `X[n]` in the parameter file. On instanciation, the plugin should load all data needed to handle the sequence. If the plugin depends on optimizable parameters (parameters whose name is followed by a `*`), then the final configuration that may depend on these parameters must be postponed in the `Init` method.
- `Init`: receives as argument the sequence to process (an instance of the `DNASeq` class) and performs the extra initializations that depends on optimizable parameters values (parameters whose name is followed by a `*`).
- `GiveInfo`: receives as argument the sequence to process (an instance of the `DNASeq` class), a position on the sequence and a `Data` instance. The `Data` data-structure can receive predictions on all signals and contents scores known to EUGÈNE.
- `Plot`: receives as argument the sequence to process (an instance of the `DNASeq` class) and plots all the predictions made by the sensor.
- `PostAnalyse`: receives as argument the prediction of EUGÈNE and may check it against its own prediction and report support or inconsistencies.

The `Plot` and `PostAnalyse` methods are often empty. The `Init` is usually limited to the reloading of optimizable parameters (see the source of the `Est` or `BlastX` plugins for exceptions).

2.3.1 Loading plugins

When EUGÈNE starts, plugins are loaded and instanciated following parameters in the parameter file. The `Sensor.*.use` may activate or desactivate the corresponding sensor (which must be available in the PLUGINS directory). If the parameter value is set to `0` or `FALSE`, the plugin is not used. If the parameter value is set to `1`, then a single instance of the plugin is loaded. If the parameter is set to an integer value, then this number of instances of the plugin are created.

Below is the list of minimum plugins which are activated by default by the *Arabidopsis thaliana* version of EUGÈNE.

```
Sensor.Transcript.use   1
Sensor.EuStop.use       1
Sensor.NStart.use       1
Sensor.IfElse.use       1 (with 2 splice site prediction plugins)
Sensor.MarkovIMM.use    1
Sensor.MarkovConst.use  1
```

Sensors are loaded and instanciated following an increasing order of priorities. The priority of a given type of plugin is defined by the value of the corresponding `Sensor.*` parameter. Here is an example of actual priorities:

```
Sensor.Transcript       1
Sensor.FrameShift       1
Sensor.IfElse           1
Sensor.EuStop           1
Sensor.NStart           1
Sensor.MarkovIMM        1
Sensor.User             20
Sensor.Est              30
```

The `Sensor.Est` is loaded last because it has the highest priority. This is important since the sensor actually uses the information provided by other sensors (splice site prediction sensors) that then have to be loaded before.

Several instances of the same sensor can be loaded. Eg., if you are dealing with an organism that has a large GC% range, one may use several `Sensor.MarkovIMM`. Imagine you want to use one model for sequences who have a GC% below 50 and another for higher GC%. This can be achieved by instaciating 2 such sensors.

```
Sensor.MarkovIMM        2
```

When these sensors will be instanciated, they will look for specific parameters. The first instance will use the usual parameters or parameters followed by `[0]` for this plugin class, the second instance will use parameters followed by `[1]`.

```
MarkovIMM.matname[0]    lowGC.mat
MarkovIMM.minGC[0]      0
MarkovIMM.maxGC[0]      50
MarkovIMM.matname[1]    highGC.mat
MarkovIMM.minGC[1]      50
MarkovIMM.maxGC[1]      100
```

As the example show, it is equivalent to define the parameter `MarkovIMM.matname[0]` (or any parameter followed by `[0]` and the parameter `MarkovIMM.matname`.

We now descrive each plugin, its behavior and parameters.

2.3.2 `Signal plugins`

2.3.2.1 `Sensor.ATGpr`

Description This plugin injects possible translation starts as predicted by the ATGpr program. The plugin reads the prediction of the programs from two files whose names are derived from the sequence name by adding the `.atgpr` and `.atgprR` suffix (respectively prediction for the forward and reverse strand).

The sensor is activated by setting the value 1 for the parameter `Sensor.ATGpr.use` in the parameter file. The score for acceptor and donor prediction is rescaled by the parameters `ATGpr.startP*` and `ATGpr.startB*` (see below).

Here is an example of ATGpr parameters definition.

```
ATGpr.startP*       0.002
ATGpr.startB*       0.001
Sensor.ATGpr.use    1        # Use ATGpr sensor
Sensor.ATGpr        1        # Sensor priority
```

Input files format The files with `.atgpr` and `.atgprR` suffix are obtained by running ATGpr which can be obtained at `http://www.hri.co.jp/atgpr/` and using the detailed output of the software.

Here is an extract from `SYNO_ARATH.fasta.atgpr`:

```
1        0.26    2       AXXATGG 29       460      144      Yes
2        0.12    3       tXXATGc 168      302      45       Yes
3        0.04    1       GXXATGc 196      234      13       Yes
4        0.04    3       GXXATGG 237      302      22       Yes
5        0.04    2       AXXATGG 371      460      30       Yes
6        0.04    2       tXXATGa 518      550      11       Yes
7        0.04    2       AXXATGG 521      550      10       Yes
8        0.04    3       tXXATGc 528      575      16       Yes
9        0.04    3       GXXATGt 546      575      10       Yes
10       0.04    1       GXXATGa 553      597      15       Yes
11       0.04    2       cXXATGc 563      583      7        Yes
12       0.04    2       GXXATGa 575      583      3        Yes
13       0.04    3       cXXATGa 645      692      16       Yes
14       0.04    1       GXXATGt 661      699      13       Yes
15       0.04    2       GXXATGt 812      910      33       Yes
[...]
```

Filtering input information No filtering.

Integration of information The integrated score for start prediction is read (column 2). The score read s is rescaled using the `ATGpr.startP*` (*P*) and `ATGpr.startB*` (*B*) as follows:

$$s' = e^{-P} * s^B$$

All predictions that use a predicted start receive a $\log(s')$ penalty while those that go through a predicted start while they could have used it receive a $\log(1 - s')$ penalty.

Post analyse No post analyse.

Graph Predicted starts are visible on exonix tracks as blue vertical lines whose length indicates the site score.

2.3.2.2 Sensor.EuStop

Description This simple plugin predicts translation stops. It is able to deal with noisy sequences and will eg. predict a possible stop on `TGN`.

The sensor is activated by setting the value 1 for the parameter `Sensor.EuStop.use` in the parameter file.

The penalty payed for using a Stop is defined by the `EuStop.stopP*` parameter.

Here is an example of EuStop parameters definition.

```
EuStop.stopP*           4.155   # Stop penalty
Sensor.EuStop.use       1    # Use EuStop sensor
Sensor.EuStop           1       # Sensor priority
```

Input files format No input files needed.

Integration of information In the case of non degenerated sequences, all predictions using a Stop to end a terminal exon are given an extra `EuStop.stopP*` penalty. All predictions going through an in-phase Stop in an exonic state receive an infinite penalty.

Post analyse No post analyse.

Graph Every predicted Stop is ploted as a small vertical red bar in the corresponding phase on the exonics tracks.

2.3.2.3 Sensor.FrameShift

Description This plugin predicts possible frameshifts (either insertions or deletions) at each position of the sequence with a uniform cost. The parameters `FrameShift.Ins*` `FrameShift.Del*` give the corresponding penalties.

The sensor is activated by either:

- the `-f` argument .
- the value 1 for the parameter `Sensor.FrameShift.use` in the parameter file.

Here is an example of FrameShift parameters definition.

```
FrameShift.Ins*         1e999.0
FrameShift.Del*         1e999.0
Sensor.FrameShift.use   1     # Use FrameShift sensor
Sensor.FrameShift       1        # Sensor priority
```

Input files format No input files needed.

Integration of information All predictions that use a frameshift (going from one coding phase to another coding phase) are given an extra `FrameShift.Ins*` or `FrameShift.Del*` penalty according to the phase change.

Post analyse No post analyse.

Graph Every predicted frameshift is plotted as a vertical red line that connect the exonic prediction blocks in the 2 corresponding phase.

2.3.2.4 `Sensor.GSplicer`

Description The GSplicer sensor injects possible splice sites as predicted by the GeneSplicer program. The plugin reads the prediction of the program from one file whose name is derived from the sequence name by adding the `.Gsplicer` suffix. This file describes the predicted splice sites for the forward and reverse strand. The sensor is activated by the value 1 for the parameter `Sensor.GSplicer.use` in the parameter file. Here is an example of GSplicer parameters definition :

```
GSplicer.coefAcc*    0.8300      #
GSplicer.penAcc*     7.7000      # GSplicer parameters (rescaling)
GSplicer.coefDon*    1.3153      # See the Integration of
GSplicer.penDon*     10.1600     # information section.
Sensor.GSplicer.use  1        # Use GSplicer sensor
Sensor.GSplicer      1           # Sensor priority
```

Input files format The files `.Gsplicer` describe the predicted splice sites sorted by position on the sequence (as given by GeneSplicer). The format of a line is : <End5> <End3> <Score> <confidence> <splice_site_type>

Here is an extract from `SYNO_ARATH.fasta.Gsplicer` :

```
422 423 17.275659 High donor
512 513 15.534963 High acceptor
583 584 9.516534 Medium donor
697 698 6.432014 Medium acceptor
745 746 2.028095 Medium donor
810 811 9.255683 Medium donor
896 897 9.772425 Medium acceptor
1019 1020 10.580889 Medium donor
1105 1106 5.318046 Medium acceptor
1177 1178 7.345249 Medium acceptor
1203 1204 3.989773 Medium donor
1269 1270 15.249301 High acceptor
[...]
```

These files can be obtained by launching the GeneSplicer software available at:

- web site : `http://www.tigr.org/tdb/GeneSplicer/index.shtml`.
- ftp serveur : `ftp://ftp.tigr.org/pub/software/GeneSplicer`.

GeneSplicer is launched with the command:

```
genesplicer SEQ_FASTA GENOME_TRAINING_DIRECTORY [options] > SEQ_FASTA.Gsplicer
```

where options are:

- $-a\ t$: Choose t as a threshold for the acceptor sites
- $-d\ t$: Choose t as a threshold for the donor sites
- $-e\ n$: The maximum acceptor score within n bp is chosen
- $-i\ n$: The maximum donor score within n bp is chosen

Filtering input information No filter.

Integration of information The procedure consists in weigthing the graph used by EuGène. For each predicted site the edge corresponding to the good transition is weighted by: $(s_i * coef) - pen$. Where s_i is the score given by GeneSplicer, $coef$ and pen are given in the parameter file.

A set of 8 vectors is used. The vectors are:

- $vPosAccF$ the forward acceptor predicted positions
- $vValAccF$ the forward acceptor score at position $vPosAccF$
- $vPosAccR$ the reverse acceptor predicted positions
- $vValAccR$ the reverse acceptor score at position $vPosAccR$
- $vPosDonF$ the forward donor predicted positions
- $vValDonF$ the forward donor score at position $vPosDonF$
- $vPosDonR$ the reverse donor predicted positions
- $vValDonR$ the reverse donor score at position $vPosDonR$

For each position if one of the 4 "position vectors" contains the query position:

- for forward and reverse acceptor sites : $(vValAcc*coefAcc)-penAcc$ is added to the corresponding transition (intron track to exon track according to the phase)
- for forward and reverse donor sites : $(vValDon*coefDon)-penDon$ is added to the corresponding transition (exon track to intron track according to the phase)

Post analyse No post analyse.

Graph Predicted splice sites are visible on the intronic tracks as green (donor) and magenta (acceptor) vertical lines whose length indicates the site score.

2.3.2.5 `Sensor.NG2`

Description This plugin injects possible splice sites as predicted by the NetGene2 program. The plugin reads the prediction of the program from two files whose names are derived from the sequence name by adding the `.splices` and `.splicesR` suffixes (respectively prediction for the forward and reverse strand).

The sensor is activated by setting the value 1 for the parameter `Sensor.NG2.use` in the parameter file. The score for acceptor and donor prediction is rescaled by the parameters `NG2.accP*` and `NG2.accB*` for acceptors and `NG2.donP*` and `NG2.donB*` for donors (see below).

Here is an example of NG2 parameters definition.

```
NG2.accP*        0.903
NG2.accB*        5.585
NG2.donP*        0.980
NG2.donB*        27.670
Sensor.NG2.use   1
Sensor.NG2       1      # Sensor priority
```

Input files format The files with `.splices` and `.splicesR` suffixes are obtained by running NetGene2 which can be obtained at `http://www.cbs.dtu.dk/services/NetGene2/` and using the detailed output of the software.

Here is an extract from `SYNO_ARATH.fasta.splices`:

```
    396 C  0      0      0.903 2  0.835  0.862  0      0       -     -
    397 T  0      0      0.858 3  0.828  0.869  0      0       -     -
    398 C  0      0      0.873 1  0.822  0.876  0      0       -     -
    399 A  0      0      0.826 2  0.816  0.882  0      0       -     -
    400 G  0      0      0.869 3  0.809  0.889  0      0      -2.337 359
    401 A  0      0      0.862 1  0.803  0.896  0      0       -     -
    402 G  0      0      0.794 2  0.798  0.901  0      0      -2.337 359
    403 C  0      0      0.809 3  0.792  0.907  0      0       -     -
    404 A  0      0      0.833 1  0.786  0.914  0      0       -     -
    405 G  0      0      0.823 2  0.780  0.920  0      0      -2.337 359
    406 T  0      0      0.845 3  0.774  0.926  0      0       -     -
    407 G  0      0      0.792 1  0.769  0.932  0      0       -     -
    408 T  0      0      0.745 2  0.764  0.936  0      0       -     -
    409 C  0      0      0.792 3  0.759  0.942  0      0       -     -
    410 A  0      0      0.802 1  0.753  0.948  0      0       -     -
    411 C  0      0      0.764 2  0.749  0.953  0      0       -     -
[...]
```

To run Netgene2, the following parameters are used (for *Arabidopsis thaliana*): `netgene2 -a -e -p -r -s at <Fasta sequence>`.

Filtering input information No filtering.

Integration of information The integrated score for donor/acceptor prediction is read (columns 9 and 10). If it is not available (extremities of the sequence) then the non integrated score is used (columns 3 and 4).

The score read s is rescaled using the `NG2.accP*` (*P*) and `NG2.accB*` (*B*) parameters for acceptors and `NG2.donP*` (*P*) and `NG2.donB*` (*B*) parameters for donors as follows:

$$s' = B * s^P$$

All predictions that use a predicted splice site receive a $\log(s')$ penalty while those that go through a predicted splice site while they could have used it receive a $\log(1 - s')$ penalty.

Post analyse No post analyse.

Graph Predicted splice sites are visible on the intronic tracks as green (donor) and magenta (acceptor) vertical lines whose length indicates the site score.

2.3.2.6 `Sensor.NStart`

Description This plugin injects possible translation starts as predicted by the NetStart program. The plugin reads the prediction of the program from two files whose names are derived from the sequence name by adding the `.starts` and `.startsR` suffixes (respectively prediction for the forward and reverse strand).

The sensor is activated by setting the value `1` for the parameter `Sensor.NStart.use` in the parameter file. The score for acceptor and donor prediction is rescaled by the parameters `NStart.startP*` and `NStart.startB*` (see below).

Here is an example of NetStart parameters definition.

```
NStart.startP*      0.052
NStart.startB*      0.308
Sensor.NStart.use   1       # Use NStart sensor
Sensor.NStart       1       # Sensor priority
```

Input files format The files with `.starts` and `.startsR` suffix are obtained by running NetStart which can be obtained at `http://www.cbs.dtu.dk/services/NetStart/` and using the detailed output of the software.

Here is an extract from `SYNO_ARATH.fasta.starts`:

```
  1089    0.256     -
  1146    0.214     -
  1251    0.618     Yes
  1299    0.197     -
  1474    0.526     Yes
  1535    0.112     -
  1559    0.490     -
  1638    0.401     -
  1674    0.569     Yes
  1678    0.147     -
  1740    0.299     -
  1752    0.187     -
[...]
```

To run NetStart, the following parameters are used (for *Arabidopsis thaliana*): `netstart -at <Fasta sequence>`.

Filtering input information No filtering.

Integration of information The integrated score for start prediction is read (column 2). The score read s is rescaled using the `NStart.startP*` (*P*) and `NStart.startB*` (*B*) as follows:

$$s' = e^{-P} * s^B$$

All predictions that use a predicted start receive a $\log(s')$ penalty while those that go through a predicted start while they could have used it receive a $\log(1 - s')$ penalty.

Post analyse No post analyse.

Graph Predicted starts are visible on exonix tracks as blue vertical lines whose length indicates the site score.

2.3.2.7 **Sensor.PatConst**

Description This plugin predicts signals at each occurence of a pattern on the sequence. The corresponding uniform costs for using or rejecting a signal can be set using the `PatConst.patP*[i]` / and `PatConst.patPNo*[i]` parameters.

The sensor is activated by setting the value 1 (one instance of the plugin) or an integer (i instance) for the parameter `Sensor.PatConst.use` in the parameter file.

Here is an example of PatConst parameters definition (2 instances) :

```
PatConst.type[0]        donor    # Possible types : start insertion deletion
PatConst.pat[0]         GC       #  transstart transstop stop acceptor donor
PatConst.newStatePos[0] 1        # Position of the new state in the pattern
PatConst.patP*[0]       -25
PatConst.patPNo*[0]     0
#
PatConst.type[1]        acceptor
PatConst.pat[1]         AG
PatConst.newStatePos[1] 3
PatConst.patP*[1]       -40
PatConst.patPNo*[1]     0
#
Sensor.PatConst.use     2
Sensor.PatConst         1        # Sensor priority
```

Input files format No file input.

Integration of information All predictions that use a predicted signal receive the corresponding `PatConst-.patP*[i]` penalty while those that go through a predicted splice site while they could have used it receive a `PatConst.patPNo*[i]` penalty.

Post analyse No post analyse.

Graph No plot.

2.3.2.8 **Sensor.PepSignal**

Description This plugin injects possible translation starts as predicted by the Predotar program (http://genoplante-info.infobiogen.fr/predotar/) that looks for peptide adressing sequences after every occurrence of an ATG. The plugin reads the predictions (both in forward and reverse strand) of the program from a file whose name is derived from the sequence name by adding the `.psignal` suffix.

The sensor is activated by setting the value `1` for the parameter `Sensor.PepSignal.use` in the parameter file. The score for start prediction is rescaled by the parameters `PepSignal.startP*` and `PepSignal.startB*` (see below).

Here is an example of PepSignal parameters definition.

```
PepSignal.startP*       0.9
PepSignal.startB*       0.1
Sensor.PepSignal.use    1       # Use PepSignal sensor
Sensor.PepSignal        10      # Sensor priority
```

Input files format The file `.psignal` describes the predicted start sites sorted by position on the sequence (as given by Predotar). The format of a line is : `<position> <strand> <score> <comment>`

Here is an example used for test:

```
175 start_rev 0.024083 test
188 start 0.000151 test
195 start_rev 0.010081 test
261 start 0.001628 test
270 start 0.000026 test
[...]
```

This file can be obtained by launching the Predotar software available at
`http://genoplante-info.infobiogen.fr/predotar/predotar.html`

Filtering input information No filtering.

Integration of information The score (column 3) read s is rescaled using the `PepSignal.startP*` (*P*) and `PepSignal.startB*` (*B*) as follows:

$$s' = e^{-P}.s^{B}$$

All predictions that use a predicted start receive a $\log(s')$ penalty while those that go through a predicted start while they could have used it receive a $\log(1 - s')$ penalty.

Post analyse No post analyse.

Graph Predicted starts are visible on exonix tracks as blue vertical lines whose length indicates the site score.

2.3.2.9 `Sensor.SMachine`

Description This plugin injects possible start and splice sites as predicted by the SpliceMachine program. For more detail, see the publication Degroeve, S., Saeys, Y., De Baets, B., Rouzé, P., Van de Peer, Y. (2004) Predicting splice sites from high-dimensional local context representations Bioinformatics.

There are two possible outputs for this program, the direct raw output directly outputs SVM values (positive or negative real numbers with arbitrary magnitude) or a rescaled output (fitting to a sigmoid) with a positive ouput between 0 and 1 (with a probabilistic interpretation). The two different outputs may lead to different prediction performances in EUGÈNE.

The plugin reads the predictions of the program from two files whose names are derived from the sequence name by adding the `.spliceMSt` and `.spliceMAD` suffixes (respectively prediction for the starts and splices sites). The sensor is activated by setting the value 1 for the parameter `Sensor.SMachine.use` in the parameter file. The score for start, acceptor and donor prediction is rescaled by the parameters `SMachine.startP*` and `SMachine.startB*` for starts, `SMachine.accP*` and `SMachine.accB*` for acceptors and `SMachine.donP*` and `SMachine.donB*` for donors (see below).

The parameter `SMachine.isScaled` indicates how the scores of SpliceMachine are integrated in EUGÈNE (the details of the scaling used in each case is given below. Note that this is the second rescaling if the sigmoid fitting has been used in SpliceMachine). The parameter `SMachine.cmd` contains the command which is launch if the predictions files do not exist.

Here is an example of SMachine parameters definition.

```
SMachine.cmd           "splicemachine.pl "
SMachine.isScaled      1
SMachine.accP*         0.102032725565
SMachine.accB*         5.585
SMachine.donP*         0.020202707318
SMachine.donB*         27.670
SMachine.startP*       0.052
SMachine.startB*       0.308
Sensor.SMachine.use    1                # Use SMachine sensor
Sensor.SMachine        10               # Sensor priority
```

Input files format The files with `.spliceMSt` and `.spliceMAD` suffixes are obtained by running SpliceMachine which can be obtained at
`http://bioinformatics.psb.ugent.be/webtools/splicemachine/`

Here is an extract from a `.spliceMSt` file:

```
175 start_rev 0.024083
188 start 0.000151
195 start_rev 0.010081
261 start 0.001628
270 start 0.000026
[...]
```

Here is an extract from a `.spliceMAD` file:

```
210 acceptor_rev 0.066414
245 donor_rev 0.001345
628 acceptor 0.066414
1309 donor 0.000039
[...]
```

Filtering input information No filtering.

Integration of information The integrated score for start and donor/acceptor prediction is read (columns 3). The score read s is rescaled using the `SMachine.startP*` (*P*) and `SMachine.startB*` (*B*) parameters for starts, `SMachine.accP*` (*P*) and `SMachine.accB*` (*B*) parameters for acceptors and `SMachine.donP*` (*P*) and `SMachine.donB*` (*B*) parameters for donors.

If (`SMachine.isScaled` is set to `0`) then the rescaled score s' is:

$$s' = B * s - P$$

used when the signal is used. If the signal is not used, no penalty occurs.

If (`SMachine.isScaled` is set to `1`) then the rescaled score s' is:

$$s' = B \log(s) - P$$

when the signal is used, and $\log(1.0 - s^B * e^{-P})$ otherwise.

If (`SMachine.isScaled` is set to `2`) then the rescaled score s' is

$$s' = B \log(s) - P$$

when the signal is used, nothing otherwise.

Post analyse No post analyse.

Graph Predicted starts are visible on exonix tracks as blue vertical lines whose length indicates the site score. Predicted splice sites are visible on the intronic tracks as green (donor) and magenta (acceptor) vertical lines whose length indicates the site score.

2.3.2.10 `Sensor.SpliceWAM`

Description The goal of the SpliceWAM sensor is to detect splice sites and to give them a score reflecting the context accordance with given models. A score is attributed at each potential acceptor and donor AG/GT site according to Weight Array Method (see Zhang and Marr, *Comput Appl Biosci.* 1993 Oct;9(5):499-509), or Weighted Array Matrix models (Salzberg, *Comput Appl Biosci* 1997 Aug;13(4):365-76). A WAM describes a consensus motif of a functional signal, and is composed by one markovian model per each position of the motif. Here the motifs are defined by the AG/GT (assumed to be present in all splice sites) plus two flanking contexts (used by the WAM). Globally, the score of a motif is a function of the emission probabilities of this motif given a true positive model and a false positive model.

The sensor is activated by setting the parameter `Sensor.SpliceWAM.use` to 1. The user have to specify in the parameter file the base name (prefix) of the model files (`SpliceWAM.donmodelfilename` and `SpliceWAM.accmodelfilename`), the size of the context (`SpliceWAM.NbNtBeforeGT`, `SpliceWAM.NbNtAfterGT` and `SpliceWAM.NbNtBeforeAG`, `SpliceWAM.NbNtAfterAG`), the order of the markovian models `SpliceWAM.MarkovianOrder` (the same for each position of the motifs), and the scaling parameters `SpliceWAM.DonScaleCoef*`, `SpliceWAM.DonScalePenalty*`, `SpliceWAM.AccScaleCoef*` and `SpliceWAM.ScalePenalty*`.

Here is an example of SpliceWAM parameters definition.

```
SpliceWAM.MarkovianOrder        1
SpliceWAM.donmodelfilename      WAM/WAM.ARA.DON.L9
SpliceWAM.NbNtBeforeGT          3
SpliceWAM.NbNtAfterGT           4
SpliceWAM.DonScaleCoef*         2.9004
SpliceWAM.DonScalePenalty*      -7.5877
SpliceWAM.accmodelfilename      WAM/WAM.ARA.ACC.L7
SpliceWAM.NbNtBeforeAG          2
SpliceWAM.NbNtAfterAG           3
SpliceWAM.AccScaleCoef*         2.9004
SpliceWAM.AccScalePenalty*      -7.5877
Sensor.SpliceWAM.use            1               # Use SpliceWAM sensor
Sensor.SpliceWAM                1                  # Sensor priority
```

Input files format This SpliceWAM Sensor requires a true positive and a false positive model file per motif position and for each type so sites. These files have to be present in the path given by `SpliceWAM.donmodelfilename` and `SpliceWAM.donmodelfilename` from the plugins directory (see `EuGene.PluginsDir` parameter). These models can be generated using `WAMbuilder.cc` (see `eugene/src/SensorPlugins/0_SensorTk/GetData/README`). The file name of a model is a concatenation of the base name (prefix) specified in the parameter file, an extension (suffix) specified in the `WAM.h` file (`.TP.` for true positive and `.FP.` for false positive), and a number between 00 and 99 indexing the position in the motif (restricting thus the motif length to a maximum of 100 nt).

As an example, with the base name `WAM.ARA.DON.L9` (refering to *A.thaliana* models of 9nt-length donor motif), one can found these files:

```
WAM.ARA.DON.L9.FP.00
WAM.ARA.DON.L9.FP.01
```

```
...
WAM.ARA.DON.L9.FP.07
WAM.ARA.DON.L9.FP.08
WAM.ARA.DON.L9.TP.00
WAM.ARA.DON.L9.TP.01
...
WAM.ARA.DON.L9.TP.08
```

These files are in binary form, each containing the properties of a markovian model (see documentations of `WAMbuilder.cc` and `markov.cc`).

Filtering input information Each binary model file is verified when loaded, checking if 3 expected properties of its markovian model are verified: the order, the alphabet size, and the total number of possible words (these 3 values are automatically included during the models generation by `WAMbuilder.cc`). This test is done in the loading file method "`chargefichier`" in markov.cc.

Integration of information At each AG/GT occuurence in the genomic sequence, a score is assigned depending on the AG/GT flanking context. If there isn't enough context, e.g. in the sequence extremities, nothing is done. This score is provided by a scaled sum of likelihood ratio, computed as following.

Let be P_i^t the emission probability of the nucleotid at position i in the motif according to the True Positive model, and P_i^f the emission probability of the nucleotid given by the False Positive model. The score given by the WAM for the entire motif M of length L is:

$$S_M = \sum_{i=0}^{L} log\left(\frac{P_i^t}{P_i^f}\right)$$

This score is then scaled with the `SpliceWAM.DonScaleCoef*`/`SpliceWAM.AccScaleCoef*` and the `SpliceWAM.DonScalePenalty*`/`SpliceWAM.AccScalePenalty*` parameters, following this formula :

$$S_M.\texttt{SpliceWAMScaleCoef}\star + \texttt{SpliceWAMScalePenalty}\star$$

This rescaled score is finally integrated into the EuGène graph on the intron/exon transition edges at the corresponding positions. The score applies only to the edge corresponding to the situation where the signal is used. The edge corresponding to the situation where the signal is not used is unchanged.

Post analyse No Post-Analyse.

Graph Vertical green/magenta lines (whose length is function of the score) are plotted on the intron track on the corresponding strand for each splice site occurrence whose score is higher than a defined threshold. This threshold is defined in the `SpliceWAM.cc` file as -`plotscoreincrease`.

2.3.2.11 `Sensor.SPred`

Description This plugin injects possible splice sites as predicted by the SplicePredictor program. The plugin reads the prediction of the program from two files whose names are derived from the sequence name by adding the `.spliceP` and `.splicePR` suffixes (respectively prediction for the forward and reverse strand).

The sensor is activated by setting the value 1 for the parameter `Sensor.SPred.use` in the parameter file. The score for acceptor and donor prediction is rescaled by the parameters `SPred.accP*` and `SPred.accB*` for acceptors and `SPred.donP*` and `SPred.donB*` for donors (see below).

Here is an example of SPred parameters definition.

```
SPred.accP*        0.987
SPred.accB*        3.850
SPred.donP*        0.929
SPred.donB*        10.800
Sensor.SPred.use   1
Sensor.SPred       1      # Sensor priority
```

Input files format The files with `.spliceP` and `.splicePR` suffix are obtained by running SplicePredictor which can be obtained at `http://bioinformatics.iastate.edu/cgi-bin/sp.cgi`.

Here is an extract from `SYNO_ARATH.fasta.spliceP`:

```
[...]
A     <- 568 tgacttcggatgcAGaa   0.006 0.000 0.000  3 (1 1 1) IAEEEDA-E-EEEDIAD
A     <- 571 cttcggatgcagaAGgg   0.001 0.000 0.000  3 (1 1 1) AEEEDAE-E-EEDIADI
D -->    573           aggGTatga 0.176 0.009 0.000  6 (2 3 1) EEEDAEE-E-EDIADII
A     <- 582 gaagggtatgatcAGgt   0.003 0.000 0.000  3 (1 1 1) EEDAEEE-E-DIADIII
D ----->  583          cagGTaatt 0.964 0.256 0.788 15 (5 5 5) EDAEEEE-D-IAEEEED
A     <- 658 tatgataatccttAGac   0.001 0.000 0.000  3 (1 1 1) DAEEEED-I-AEEEEDI
A   <--- 698 ttgggtggataatAGgt   0.273 0.049 0.266 10 (3 3 4) AEEEEDI-A-EEEEDII
D ->     699           tagGTagaa 0.006 0.000 0.000  3 (1 1 1) AEEEDIA-E-EEEDIII
A     <- 702 gtggataataggtAGaa   0.001 0.000 0.000  3 (1 1 1) IIADIAD-I-IIIIIAD
D ->     709           ctgGTtcga 0.005 0.000 0.000  3 (1 1 1) IADIADI-I-IIIIAED
A     <- 744 cagtatctgtacaAGgt   0.007 0.000 0.000  3 (1 1 1) ADIADII-I-IIIAEDI
D ->     745           aagGTacta 0.042 0.000 0.000  3 (1 1 1) DIADIII-I-IIAEDIA
A     <- 756 aaggtactattgtAGct   0.007 0.000 0.000  3 (1 1 1) IADIIII-I-IAEDIIA
A     <- 760 tactattgtagctAGcc   0.002 0.000 0.000  3 (1 1 1) ADIIIII-I-AEDIIAE
A     <- 764 attgtagctagccAGgg   0.025 0.000 0.023  4 (1 1 2) DIIIIII-A-EDIIAEE
D ->     792           aagGTggag 0.057 0.001 0.000  3 (1 1 1) IIIIIIA-E-DIIAEEE
[...]
```

Filtering input information No filtering.

Integration of information One of the SplicePredictor scores s for a given position is rescaled using the $\log(\alpha.s^{\beta})$ function. The four parameters `SPred.accP*`, `SPred.accB*`, `SPred.donP*`, `SPred.donB*` indicates the values of these α and β parameters for acceptor sites and donor sites respectively. These parameters have been estimated on existing data.

Post analyse No post analyse.

Graph Predicted splice sites are visible on the intronic tracks as green (donor) and magenta (acceptor) vertical lines whose length indicates the site score.

2.3.2.12 `Sensor.StartWAM`

Description The goal of the StartWAM sensor is to detect the translation start codons and to give them a score reflecting the context accordance with given models. A score is attributed at each potential start codons (ATG), according to Weight Array Method (see Zhang and Marr, *Comput Appl Biosci.* 1993 Oct;9(5):499-509), or Weighted Array Matrix models (Salzberg, *Comput Appl Biosci* 1997 Aug;13(4):365-76). A WAM describes a consensus motif of a functional signal, and is composed by one markovian model per each position of the motif. Here the motif is defined by the ATG (present in all start codons) plus the two flanking contexts (used by the WAM). Globally, the score of a motif is a function of the emission probabilities of this motif given a true positive model and a false positive model.

The sensor is activated by setting the parameter `Sensor.StartWAM.use` to 1. The user have to specify in the parameter file the base name (prefix) of the model files (`StartWAM.modelfilename`), the size of the context (`StartWAM.NbNtBeforeATG, StartWAM.NbNtAfterATG`), the order of the markovian models `StartWAM.MarkovianOrder` (the same for each position of the motif), and the scaling parameters `StartWAM.ScaleCoef*, StartWAM.ScalePenalty*`.

Here is an example of StartWAM parameters definition.

```
StartWAM.modelfilename      WAM/WAM.ARA.START  # base name of the model files
StartWAM.NbNtBeforeATG      3                  # amount context
StartWAM.NbNtAfterATG       3                  # aval context
StartWAM.MarkovianOrder     1                  # order of the markovian models
StartWAM.ScaleCoef*         0.1594             # scaling parameter
StartWAM.ScalePenalty*      -3.1439            # scaling parameter
Sensor.StartWAM.use         1               # Use StartWAM sensor
Sensor.StartWAM             1                  # Sensor priority
```

Input files format This StartWAM sensor requires a true positive and a false positive model file per motif position. These files have to be present in the path given by `StartWAM.modelfilename` from the plugins directory (see `EuGene.PluginsDir` parameter). These models can be generated using `WAMbuilder.cc` (see `eugene/src/SensorPlugins/0_SensorTk/GetData/README`). The file name of a model is a concatenation of the base name (prefix) specified in the parameter file, an extension (suffix) specified in the `WAM.h` file (`.TP.` for true positive and `.FP.` for false positive), and a number between 00 and 99 indexing the position in the motif (restricting thus the motif length to a maximum of 100 nt).

As an example, with the base name `WAM.ARA.START9` (refering to *A.thaliana* models of 9nt-length start motif), one can found these files:

```
WAM.ARA.START9.FP.00
WAM.ARA.START9.FP.01
...
WAM.ARA.START9.FP.07
WAM.ARA.START9.FP.08
WAM.ARA.START9.TP.00
WAM.ARA.START9.TP.01
...
WAM.ARA.START9.TP.08
```

These files are in binary form, each containing the properties of a markovian model (see documentations of `WAMbuilder.cc` and `markov.cc`).

Filtering input information Each binary model file is verified when loaded, checking if 3 expected properties of its markovian model are verified: the order, the alphabet size, and the total number of possible words (these 3 values are automatically included during the models generation by `WAMbuilder.cc`). This test is done in the loading file method "`chargefichier`" in markov.cc.

Integration of information At each ATG of the genomic sequence a score is assigned depending on the ATG flanking context. If there isn't enough context, e.g. in the sequence extremities, nothing is done. This score is provided by a scaled sum of likelihood ratio, computed as following.

Let be P_i^t the emission probability of the nucleotid at position i in the motif according to the True Positive model, and P_i^f the emission probability of the nucleotid given by the False Positive model. The score given by the WAM for the entire motif M of length L is:

$$S_M = \sum_{i=0}^{L} log\left(\frac{P_i^t}{P_i^f}\right)$$

This score is then scaled with the `StartWAM.ScaleCoef*` and the `StartWAM.ScalePenalty*` parameters, following this formula :

$$S_M.\texttt{StartWAMScaleCoef}\star + \texttt{StartWAMScalePenalty}\star$$

This rescaled score is finally integrated into the EUGÈNE graph on the UTR5 → EXON transition edge just before the considered ATG. The score applies only to the edge corresponding to the situation where the signal is used. The edge corresponding to the situation where the signal is not used is unchanged.

Post analyse No Post-Analyse.

Graph Vertical blue lines (whose length is function of the score) are plotted on the corresponding frame for each start codon which score is higher than a defined treshold. This threshold is defined in the `StartWAM.cc` file as -`PlotScoreIncrease`.

2.3.2.13 `Sensor.Transcript`

Description This simple plugin predicts a possible transcription start and stop at every position, all with the same uniform cost. These costs are respectively set by the `Transcript.Start*` and `Transcript.Stop*` parameters.

The sensor is activated by setting the value 1 for the parameter `Sensor.Transcript.use` in the parameter file.

Here is an example of Transcript parameters definition.

```
Transcript.Start*       4.155
Transcript.Stop*        4.155
Sensor.Transcript.use   1   # Use Transcript sensor
Sensor.Transcript       1      # Sensor priority
```

Input files format No input file needed.

Integration of information All predictions that go through a transcription start (resp. stop) are penalized by the corresponding `Transcript.Start*` (resp. `Transcript.Stop*`) parameter value.

Post analyse No post analyse.

Graph No plotting.

2.3.3 `Content plugins`

2.3.3.1 `Sensor.BlastX`

Description The BlastX sensor allows to exploit similarities with homologous proteins. The similarities influence exon and intron detection. Similarities from several databases can be exploited. Usually 3 databases are used: SwissProt, PIR and TrEMBL.

A label i (that could vary from 0 to 9) is assigned at each considered database. Files describing a collection of similarities with a sequence have an extension .blast<i> (`.blast0, ..., .blast9`).

The user has to specify the list of labels to consider, the confidence accorded to each, the minimum length of an intron and a number of amino acids involved in intron incitation. The sensor is activated by either:

- the `-b` argument that allows to specify the labels to consider, for example `-b092` to use the levels 0, 9, 2 (files `.blast0, .blast9, .blast2`),
- the value 1 for the parameter `Sensor.BlastX.use` in the parameter file and the labels to consider in the `.BlastX.levels` parameter.

The confidence in analyzes have to be specified in the parameter file giving values to the parameters `BlastX.level<i>*`. The minimum length of an intron is defined in the `BlastX.minIn` parameter. A number of amino acids defined in the `BlastX.blastxM*` parameter that allows to define if 2 similarities are near (see the paragraph Integration of information). Finally the `BlastX.postProcess` parameter (when set to 1) allows to request to analyse how BlastX information are integrated in the final prediction.

Here is an example of BlastX parameters definition.

```
BlastX.postProcess 1     # analyse prediction accorded to BlastX information
BlastX.levels      012      # use levels 0, 1, and 2
BlastX.levels      0        # make gap active on level 0
BlastX.level0*     0.2      # confidence in the level 0
BlastX.level1*     0.0      # confidence in the level 1
BlastX.level2*     0.0      # confidence in the level 2
BlastX.blastxM*    10       # nb of amino acids implicated in intron incitation
BlastX.minIn       50       # minimum length of intron
Sensor.BlastX.use  1     # Use BlastX sensor
Sensor.BlastX      1        # Sensor priority
```

Input files format The files `.blast<i>` describe a collection of similarities sorted by protein and by position on the sequence. One similarity S is described per line.

The format of a line is:

<b^S> <e^S> <s^S> <v^S> <p^S> `<protein name>` <bp^S> <ep^S>

where:

- b^S and e^S are the begin and the end of the similarity S on the sequence,
- s^S is the score of the similarity S,
- v^S is the e-value given by BlastX and ignored by EUGÈNE,
- p^S is the phase: +1, +2, +3, -1, -2, -3,
- bp^S and ep^S are the begin and the end of the similarity S on the protein.

Here is an extract from `SYNO_ARATH.fasta.blast0`:

```
2820 2861 36 3e-08 +3  sp_O07683_SYD_HALSA; 335 348
2972 3088 41 3e-08 +2  sp_O07683_SYD_HALSA; 359 397
3185 3298 113 3e-08 +2  sp_O07683_SYD_HALSA; 398 435
353 418 45 2e-13 +2  sp_O24822_SYD_HALVO; 13 34
1850 1915 67 2e-13 +2  sp_O24822_SYD_HALVO; 202 223
2775 2858 72 2e-13 +3  sp_O24822_SYD_HALVO; 318 345
3191 3280 104 2e-13 +2  sp_O24822_SYD_HALVO; 397 426
353 418 51 7e-12 +2  sp_O26328_SYD_METTH; 21 42
1271 1414 70 7e-12 +2  sp_O26328_SYD_METTH; 141 188
1850 1954 62 7e-12 +2  sp_O26328_SYD_METTH; 210 244
3191 3280 93 7e-12 +2  sp_O26328_SYD_METTH; 401 430
```

These files can be obtained directly from the output BlastX files by parsing them with the `blast_parser.pl` script. The BlastX is launched with the command:

```
blastall -p blastx -d DATABASE_MULTIFASTA_PROTEIC_FILE -g F -F T -b
500000 -v 500000 -e 1e-6 -i QUERY_GENOMIC_SEQUENCE_FASTA >
TEMPORY_BLAST_RESULT_FILE
```

and the final `.blast`$< i >$ files are obtained with:

```
blast_parser.pl TEMPORY_BLAST_RESULT_FILE | sort -n -k 1,1 | sort -s
-k 6,6 > QUERY_GENOMIC_SEQUENCE_FASTA.blast0
```

For more explanation, see the README file in the directory `eugene/src/SensorPlugins/BlastX/GetData`.

Filtering input information Similarities with a length higher than 15,000 nucleodites are rejected. A message “Similarity of extreme length rejected” is printed to alert the user.

Integration of information The procedure consists first, in computing information at the nucleotide level and second, in weigthing the graph used by EUGÈNE.

A/ Computing information at the nucleotide level

A-1/ Extracting information

Each similarity S is considered, one after the other. A set of 3 variables is computed for nucleotide in position i. The variables are:

- s_i the score of the nucleotide at position i
- c_i the confidence in s_i,
- p_i the phase of s_i : +1, +2, +3, -1, -2, -3 for exon and 0 for intron,

Let l^S be the length of the similarity in nucleotide.

$$l^S = (ep^S - bp^S - 1) * 3$$

<u>Valuation for exon position</u>

- from $i = b^S$ to $i = e^S$
  - $s_i = s^S/l^S$
  - $c_i = c^S$
  - $p_i = p^S$

<u>Valuation for intron position</u>

Intron is only possible if:

- the similarities before or after in the same file are on the same protein, strand and near. That is have the same protein name, the same sense and have a maximum distance or overlap of `BlastX.blastxM*` amino acids.
- the distance in nucleotide on the sequence is upper than `BlastX.minIn`.

Considering a similarity S with 2 similarities before and after in accordance with these conditions, an intron is incitated on a small length of `BlastX.minIn`/2 on both sides of S.

- from $i = b^S -$ BlastX.minIn$/2$ to $i = b^S$
- from $i = e^S$ to $i = e^S +$ BlastX.minIn$/2$
- the following values are given at each position:
  - $s_i = s^S/l^S$
  - $c_i = c^S$
  - $p_i = 0$

If the `activegaps` parameter is active for a level, then all the bases in the gap and not mentionned above are also penalized but only for intergenic, UTR and UTR introns states. The penality used is the minimum of the weights used for the left and right HSP.

A-2/ Combining extracted information

When all the similarities have been handled, if a position has several set of variables, the set with the highest confidence is kept. In case of egal confidence, the set with the higher score is kept.

B/ Weighting the graph

For each i with a set of variables:

- if p_i codes for exon then $s_i.c_i$ is added to the content score of nucleotide i in the corresponding exon phase (a track between 0 and 5),
- if p_i codes for intron then $s_i.c_i$ is added to the intron score of nucleotide i (tracks 6 and 7),

Note: in fact, instead of rewarding the correct track (like described here), all the tracks except the according one(s) are penalized, with a penalty equal to $-|s_i.c_i|$.

Post analyse The correspondance between BlastX information and prediction is analyzed if the `-B` flasg is provided or if the `BlastX.PostProcess` parameter is set to `1`.

For each predicted CDS, from the start codon to the stop, the percentage of nucleotides supported by a proteic similarity is displayed.

Graph Grey horizontal lines are ploted on the exon tracks for only the 3 first levels to consider (dark grey for the first, grey for the second, and light grey for the third).

2.3.3.2 `Sensor.Est`

Description This sensor is intended to take into account information from aligned transcribed sequences, both complete cDNA and EST. The existence of a hit (resp. gap) in the spliced alignment will influence intergenic, exonic and intronic state costs by penalizing states that are incompatible with the alignment. The spliced alginments must be performed beforehand using a spliced aligner such as `sim4` or `spidey`. The output of these aligners must be converted in the adequate format (see below).

The sensor is activated by either:

- the `-d` argument .
- the value 1 for the parameter `Sensor.Est.use` in the parameter file.

The behavior of the plugin is controlled by 4 parameters:

- `Est.est*P` indicates the penalty for violating a transcribed evidence
- `Est.estM` gives the amount of "fuzzyness" allowed in interpreting a hit/gap border. The nucleotides which are less than `Est.estM` nucleotides away from this border are considered as neither in a hit or a gap.
- `Est.utrP*` is a penalty introduced to try to limit the extension of UTR beyond the frontier of transcribed evidence when there is some. For a defined length, the adequate UTR states that precede or follow an uninterrupted stretch of transcribed evidence will be penalized by the logarithm of this parameter.
- `Est.utrM` gives the number of UTR nucleotides that will be penalized using the previous penalty on the border of transcribed evidence.
- `Est.StrongDonor*` gives a threshold T on donor strength inside intronless EST. If a given Donor with scores $a|b$ such that $\log(\frac{a}{b}) > T$ then the intronless EST is rejected because it goes through a very strong donor site.
- `Est.MinDangling` gives the minimum length of the first and last match region in a genomic spliced alignment. If the length is below this, then it is assumed to be a false match and it is ignored.
- `Est.MaxIntron` gives the maximum length for the first and last gap (representing introns) in a spliced alignment. If the length exceeds this maximum, then the corresponding regions are ignored in the alignment (the long gap and the dangling hit).
- `Est.MaxInternalIntron` gives the maximum length for any gap (representing an intron) in a spliced alignment. If the length exceeds this maximum, then the corresponding gap is ignored as a probable bad spliced alignment or chimeric data..

The sensor is also capable of a postprocessing analyse described below and activated either by the `-E` argument or by setting the value 1 for the parameter `Sensor.Est.PostProcess` in the parameter file.

The `Sensor.Est` is loaded last because it has the highest priority. This is important since the sensor actually uses the information provided by other sensors (splice site prediction sensors) that then have to be loaded before.

Here is an example of Est parameters definition.

```
Est.PostProcess  FALSE    # Don't postprocess
Est.estP*        -0.4
Est.estM         6
Est.utrP*        0.35
Est.utrM         5
Est.StrongDonor* 0.95
Est.MinDangling  18
Est.MaxIntron    15000
Sensor.Est.use   1     # Use EST sensor
Sensor.Est       20       # Sensor priority: the highest one
```

Input files format The input is obtained in a `.est` file with 8 fields of information per ligne:

<b^S> <e^S> <s^S> <x> <b^S> `<est name>` <bq^S> <eq^S>

where:

- b^S and e^S are the begin and the end of the similarity S on the genomic sequence,
- s^S is the score of the similarity S (number of identical bases)

- x is unused for now
- b^S is the strand where the similarity occurs (forward = 0, reverse = 1). This information is not used anymore by the plugin which decides the strand of similarity by itself if there is anough information.
- `<est name>` is the name of the EST/cDNA sequence. Each sequence MUST have a unique name.
- <bq^S> <eq^S> are the begin and the end of the similarity S on the query (EST/cDNA) sequence.

The lines in the file must be ordered by sequence name first (all the hits of a given EST are put together) and by increasing <bq^S> <eq^S>.

Here is an example for the `SYNO_ARATH.tfa.est` file:

```
  32     421 1844  0  0 ATAJ644    1   390
 514     582 1844  0  0 ATAJ644  391   459
 699     809 1844  0  0 ATAJ644  460   570
 914    1018 1844  0  0 ATAJ644  571   675
1271    1408 1844  0  0 ATAJ644  676   813
1522    1602 1844  0  0 ATAJ644  814   894
1694    1771 1844  0  0 ATAJ644  895   972
1853    1921 1844  0  0 ATAJ644  973  1041
2014    2088 1844  0  0 ATAJ644 1042  1116
2181    2264 1844  0  0 ATAJ644 1117  1200
2360    2446 1844  0  0 ATAJ644 1201  1287
2712    2882 1844  0  0 ATAJ644 1288  1458
2966    3092 1844  0  0 ATAJ644 1459  1585
3189    3447 1844  0  0 ATAJ644 1586  1844
  32     375 347  0  0 AT00622    1   347
3071    3092 256  0  1 AI994358    1    22
3189    3421 256  0  1 AI994358   23   256
```

All the hits of the ATAJ644 are clustered together and sorted with increasing <bq^S> <eq^S> . In practice, this file can be directly constructed from an EST/cDNA bank and the sequence using a modified version of `sim4`. This version outputs splices alignements in the correct format (using the flag `A=6`) and only outputs hits with a coverage of more than 80% and with a similarity either than 90%.

A patch file `sim4.patch` is available in the plugin source directory as well as an awk script that put a FASTA sequence bank in a pure sim4 format (no upper case, no degenerated code). This seems useless on recent sim4 versions.

Filtering input information The EST information goes through a complex filtering process. First all hits are loaded. Successive hits of a same sequence are considered as a single alignment. For every spliced alignement, the plugin checks if a splice site of the correct type as been predicted near the border of each gap (less than `Est.estM` bases aways of the border). This is checked on each strand. If a strand does not contain the necessary splice sites, then it is considered as impossible. If neither strand contains adequate splice sites, the sequence is discarded (filtered).

All remaining alignments are sortered by 1) decreasing number of detected gaps then by 2) length (this tends to put cDNA or spliced EST alignments first) and 3) by the alphabetical order of the sequence names (to avoid sorting ambiguities). Any sequence that is inconsistent with previous sequence in this order (in the sense that they indicate that a given nucleotide is part of an intron, resp. exon, while a previous sequence indicates that the nucleotide is part of an exon (resp. intron) is discarded (filtered).

Any unspliced sequence that crosses a donor site that is predicted with a sufficiently strong confidence (See `Est.StrongDonor*`) is filtered out.

Integration of information Using filtered EST/cDNA sequence that are all consistent, every nucleotide can either be located:

Hit inside a matching segment that can occur on the forward strand on the reverse strand or both (as identified during filtering).

Gap or inside a gap segment that can occur on the forward strand on the reverse strand or both (as identified during filtering).

None or otherwise outside of any existing hit or gap (or less than `Est.estM` bases aways of the border of such a segment)

For a Hit: all intronic and intergenic tracks as well as UTR and exonic tracks on a strand incompatible with the hit are penalized with `Est.estP*`.

For a Gap: all exonic and intergenic tracks as well as UTR and intronic tracks on a strand incompatible with the hit are penalized with `Est.estP*`.

When no information exists (None), the UTR tracks are penalized by `Est.utrP*` if and only if there is a Hit evidence at less than `Est.utrM` bases away.

Post analyse The correspondance between transcribed sequence information and prediction is analyzed if the `-E` argument is activated or if the `Est.PostProcess` parameter is set to `1`.

For each coding exon predicted, the percentage of nucleotides supported by a transcribed sequence is displayed. This count includes all EST/cDNA, including those filtered.

For each predicted transcript (from 5'UTR to 3'UTR), each available EST/cDNA sequence that overlap the transcript is compared to the prediction. The transcribed sequence is then classified as:

- Filtered (if it was filtered in the initial filtering process)
- Inconsistent (if it is incompatible with the prediction)
- Full transcript Support (if it is completely consistent with the predicted transcript on all the predicted transcript length)
- Full Coding Support (if it is completely consistent with the predicted CDS on all the predicted CDS length, from start to stop codon)
- Support (if it is otherwise consistent with the prediction)

Finally, for each predicted transcript (from 5'UTR to 3'UTR) and predicted CDS (from start to stop), the number of predicted nucleotides that are supported by existing transcripts (filtered or not) is reported.

Graph Non filtered transcribed evidence are plotted as blue horizontal blocks (hits) separated by thin blue lines (gap) on the intronic tracks. If the strand of the transcription has been identified during filtering, the blocks and lines occurr only on the corresponding IR or IF track.

Filtered sequences are plotted as gray blocks and lines just above (below) the forward (reverse) intronic tracks.

2.3.3.3 **Sensor.Homology**

Description This sensor is intended to take into account information from one or more homologous DNA sequences, usually genomic sequences from other species (inter-genomic homology), other sequences from the same genome (intra-genomic homology), or transcript sequences from normalized cDNA sets. The underlying general idea is that during evolution, functional genomic regions (*e.g.* exons) tend to be more conserved than non-functional ones (*e.g.* introns). The sensor increases the coding score of a genomic position that is included in a conserved region.

Homology detection has to be performed beforehand by the `TBlastX` software. Resulting alignment files require a specific format (see below).

The sensor is activated by either :

- the `-t` argument
- the value 1 for the parameter `Sensor.Homology.use` in the parameter file.

Here is an example of Homology parameters definition :

```
Homology.TblastxP*       0
Homology.TblastxB*       0.0595
Homology.protmatname     BLOSUM80
Sensor.Homology.use      1          # Use Homology sensor
Sensor.Homology          1          # Sensor priority
```

`Homology.protmatname` is the name of the file containing the amino acid substitution score matrix used to measure the base homology score at the proteic level, before scaling (standard text format). `Homology.TblastxP` and `Homology.TblastxB` parameters are used to scale the information given by homology regions. For more details, please refer to the publication "EUGÈNE'HOM: a generic similarity-based gene finder using multiple homologous sequences" (Foissac *et. al*, *Nucleic Acids Res.*, 2003, 31(13):3742-5).

Input files format The files `.tblastx` describe a collection of similarities sorted by subject sequence and by position on the query sequence.

These files are obtained by blasting with `TBlastx` the genomic sequence (the query) against a set of other DNA sequences, and by parsing the results with the `ParseBlastXML.pl` script.

The TBlastX is launched with the command:

```
blastall -p tblastx -i QUERY_GENOMIC_SEQUENCE_FASTA -d
DATABASE_MULTIFASTA_FILE -F T -M SUBSTITUTION_MATRIX_FILE -e 1e-6 -b
50000 -m 7 > TEMPORY_BLAST_RESULT_FILE
```

Note: in order to reduce the number of "phantom frame" hits, the amino acid substitution matrix (e.g. BLOSUM62) should be modified by setting every score involving a STOP codon (lines and column noted with a star *) to a huge penalty (e.g. -500).

and the final `.tblastx`$< i >$ files are obtained with:

```
ParseBlastXML.pl TEMPORY_BLAST_RESULT_FILE > QUERY_GENOMIC_SEQUENCE_FASTA.tblastx
```

For more explanation, see the README file in the directory `eugene/src/SensorPlugins/Homology/GetData`.

One similarity S is described per line.

The format of a line is similar to the ".blast" file format (BlastX sensor) with an additional column displaying the translated sequence (amino acid alphabet) of the subject matching region:

<b^S> <e^S> <s^S> <v^S> <p^S> `<subject seq. name>` <bp^S> <ep^S> `<AA_SEQ>`

where:

- b^S and e^S are the begin and the end of the similarity S on the query sequence,
- s^S is the score of the similarity S,
- v^S is the e-value given by TBlastX and ignored by EUGÈNE,
- p^S is the phase: +1, +2, +3, -1, -2, -3,
- bp^S and $ep^S$are the begin and the end of the similarity S on the subject sequence,
- AA_SEQ is the amino acid sequence translated from the subject nucleic region.

Here is an example of the format:

```
831 878 51 7e-26 -3 ATHA10A_809_856 809 856 TLQLHGRRYVETTVFV
828 878 48 1e-20 +3 ATHA10A_806_856 806 856 RDKHRCFHVSSAMKLEG
1572 1652 109 7e-114 -3 ATHA10A_1349_1429 1349 1429 IPWSNLLELKSTPMILEAPAILAPSAA
1738 1821 108 1e-153 +1 ATHA10A_1493_1576 1493 1576 FDMLLAAKEFGVTECVNPKDHDKPIQQV
830 877 86 1e-153 +2 ATHA10A_808_855 808 855 GQTPLFPRIFGHEAGG
590 625 44 1e-20 +2 ATHA10A_562_597 562 597 LQLLWHGKPESH
```

Filtering input information Similarities with a length higher than 15,000 nucleodites are rejected. A message "Similarity of extreme length rejected. Check tblastx file <NAME>" is printed to alert the user.

Integration of information The TBlastX search returns a set of High Scoring Pairs (HSP), each in a given frame. All pairs of HSP which overlap and are in the same frame are clustered together in so-called "HSP contigs". To associate an homology score to a given nucleotide n_i in the context of a coding region, we consider (if it exists), the single HSP contig HC that overlaps the nucleotide in the sequence and which is in the same frame. Let n be the maximum number of HSPs in the cluster that overlap a single position. Let $c(n_i)$ be the codon that contains n_i in the sequence, for each HSP h in the cluster that overlaps the codon $c(n_i)$, we define $S(c(n_i), h)$ to be the matrix substitution score for the amino acid coded by $c(n_i)$ in the HSP alignment. This score is considered as equal to zero for non overlapping HSP. The homology score for the nucleotide in the context of the coding region considered is defined as:

$$HS(n_i) = \frac{1}{n} \sum_{h \in HC} S(c(n_i), h)$$

The resulting score provided by the sensor after scaling is equal to

$$\texttt{Homology.TblastxB}\,.HS(n_i) + \texttt{Homology.TblastxP}$$

Post analyse None.

Graph HSP clusters are represented as grey blocks whose thickness is proportional to the number of hits at a given position and whose darkness is proportional to the homology score at this position.

2.3.3.4 `Sensor.MarkovConst`

Description A simulated content sensor that gives constant probabilities (as indicated in the `MarkovConst-.Coding*`, `MarkovConst.Intron*`, `MarkovConst.UTR5*`, `MarkovConst.UTR3*` and `Markov-Const.Inter*` parameters). to all positions for each region type. As the MarkovIMM sensor, the plugin is controlled by two further parameters: `MarkovConst.minGC` and `MarkovConst.maxGC` which indicate the GC scope of the contents sensor. If the GC% of the sequence is out of the scope, the plugin will give an equal null loglikelihood to all types of regions.

Used for testing purposes and for simulating the exponential length distributions of HMM.

Here is an example of MarkovConst parameters definition.

```
MarkovConst.Coding*      1.0
MarkovConst.Intron*      1.0
MarkovConst.UTR5*        0.999
MarkovConst.UTR3*        0.999
MarkovConst.Inter*       1.0
MarkovConst.minGC[0]     0
MarkovConst.maxGC[0]     100
Sensor.MarkovConst.use  1   # Use MarkovConst sensor
Sensor.MarkovConst       1      # Sensor priority
```

Input files format No input files needed.

Integration of information If the GC% of the sequence handled is between `MarkovConst.minGC` and `MarkovConst.maxGC` then in every position, in all possible states, the prediction is penalized by the logarithm of the corresponding parameter: `MarkovConst.Coding*`, `MarkovConst.Intron*`, `MarkovConst.UTR5*`, `MarkovConst.UTR3*` and `MarkovConst.Inter*`.

Post analyse No post analyse.

Graph No plotting.

2.3.3.5 `Sensor.MarkovIMM`

Description This plugin injects coding/intronic/utr/intergenic likelihood as modeled by interpolated Markov models (introduced in Glimmer, see S. Salzberg, A. Delcher, S. Kasif, and O. White.*Microbial gene identification using interpolated Markov models* Nucleic Acids Research 26:2 (1998), 544-548). These models are defined in a so-called matrices file (located in the directory specified by the `EuGene.PluginsDir` parameter) whose name is indicated by the `MarkovIMM.matname` parameter. Depending on the matrices file, this may contain IMM for exons, introns and intergenic data and also optionnally 5' and 3' UTR regions. If these 2 last IMMs are absent from the matrices file, intronic models are used for UTR.

The plugin is controlled by two further parameters: `MarkovIMM.minGC` and `MarkovIMM.maxGC` which indicate the GC scope of the matrices. If the GC% of the sequence is out of the scope, the plugin will give an equal null loglikelihood to all types of regions.

By instanciating multiple `MarkovIMM` plugins (see section 2.3), this enables the use of several IMM according to the GC% of the input sequence.

Instead of using a specific intergenic IMM, it is possible to simply use the sequence GC% as the basis for a O^{th} order Markov model by setting `MarkovIMM.UseM0asIG` to `1`.

The sensor is activated by either:

- the `-m` argument followed by the filename of the set of Markov models.
- the value 1 for the parameter `[Sensor.MarkovIMM.use` in the parameter file.

Here is an example of MarkovIMM parameters definition.

```
MarkovIMM.matname    Ara2UTR.mat
MarkovIMM.minGC      0
MarkovIMM.maxGC      100
MarkovIMM.UseM0asIG  FALSE
Sensor.MarkovIMM.use 1
Sensor.MarkovIMM     1       # Sensor priority
```

Input files format No input files needed beyond the IMM matrix file.

Integration of information On the forward intronic, exonic and UTR tracks, the probability that the nucleotide at position i appears given the nucleotides that follow him is fetched in the IMM matrix file. The logarithm of this probability is used as a penalty on the corresponding track.

For reverse tracks, the same process is used but the probability used is the probability that the nucleotide at position i appears given the nucleotides that precede him.

For the intergenic track, the probability used is the mean of the probability that the nucleotide at position i appears given the nucleotides that precede him and the probability that the nucleotide at position i appears given the nucleotides that follow him. This guarantees that a sequence and its reverse complement will receive the same weights exactly.

Post analyse No post analyse.

Graph The likelihood of a subsequence of width `Output.window` is computed for each IMM model and normalized over all these. The corresponding normalized likelihood is plotted as a thin black line on each track of the graphical output.

2.3.3.6 `Sensor.MarkovProt`

Description This plugin injects coding/non coding likelihood as modeled by proteic Markov models. These models are defined in a matrices file (located in the directory specified by the `EuGene.PluginsDir` parameter) whose name is indicated by the `MarkovProt.matname` parameter. The order of the Markov model must be given in `MarkovProt.maxorder` while the actual order to use is set by `MarkovProt.order`.

The plugin is controlled by two further parameters: `MarkovProt.minGC` and `MarkovProt.maxGC` which indicate the GC scope of the matrices. If the GC% of the sequence is out of the scope, the plugin will give an equal null loglikelihood to all types of regions.

The sensor is activated by either:

- the `-M` argument followed by the filename of the set of models.
- the value 1 for the parameter `[Sensor.MarkovProt.use` in the parameter file.

Here is an example of MarkovProt parameters definition.

```
MarkovProt.matname      SwP41.noFragm.mininfo1.order2.bin
MarkovProt.minGC        0
MarkovProt.maxGC        100
MarkovProt.maxorder     2
MarkovProt.order        2
Sensor.MarkovProt.use   1    # Use MarkovProt sensor
Sensor.MarkovProt       1       # Sensor priority
```

Input files format No input files needed beyond the markov matrix files.

Integration of information For coding tracks, assuming a uniform codon usage, the probability of the coding tracks is decomposed as the product of choosing a codon and then emitting the corresponding amino acid in the corresponding phase. The logarithm of the probability is used for weighting.

For other tracks, a simple GC% model is used to compute a background probability. The logarithm of the probability is used for weighting.

Post analyse No post analyse.

Graph Same as in the MarkovIMM plugin.

2.3.3.7 `Sensor.Repeat`

Description The plugin allows to exploit the output of repeated sequences detector such as RepeatMasker by penalizing exonic, inronic or UTR states when repeats are detected.

The sensor is activated by either :

- the `-r` argument
- the value 1 for the parameter `Sensor.Repeat.use` in the parameter file.

The penalties used when a repeat exists are `Repeat.IntronPenalty*`, `Repeat.ExonPenalty*` and `Repeat.UTRPenalty*` respectively.

Here is an example of Repeat parameters definition.

```
Repeat.UTRPenalty*      0.0
Repeat.IntronPenalty*   0.1
Repeat.ExonPenalty*     1.0
Sensor.Repeat.use       1    # Use Repeat sensor
Sensor.Repeat           1       # Sensor priority
```

Input files format The file with a `.ig` suffix is needed. Each line of the file contains the beginning and the end of a region detected as a repeat. The positions must be sorted in increasing positions. Such a file can be obtained by eg. reformatting RepeatMasker output.

Here is an extract from a typical `.ig` file:

```
4800    5006
22494   22758
22703   22772
22841   23017
22929   23017
29433   29703
[...]
```

Filtering input information No filtering.

Integration of information For exonic, intronic and UTR tracks, all positions that occur in a repeat interval as reported in the `.ig` file are penalized using the corresponding `Repeat.IntronPenalty*`, `Repeat.ExonPenalty*` and `Repeat.UTRPenalty*` penalties.

Post analyse No post analyse.

Graph Repeat intervals are visualized as grey blocks in the intergenic track.

2.3.4 Mixed signal/content plugins

2.3.4.1 Sensor.AnnotaStruct

Description The sensor allows to seamlessly modify EUGÈNE underlying graph weighting using a small langage that can directly modify the weights of signals and contents edges in the graph. The plugin offers both high-level entries and low-level entries in GFF format.

The high-level entries allow to take into account information on:

- **transcribed sequences** (involving exons, introns, UTR, transcription start and transcription stop and splice sites) that may come from alignment of transcribed sequences (using spliced alignment algorithms such as sim4 or PASA).
- **CDS** (involving exons, introns, translation start, translation stop and splice sites) that may come from other gene predictors that may predict CDS (either *ab initio* or homology based predictors).

The high-level entries are actually automatically expanded in elementary (low-level) information as the plugin reads the data. The way the expansion takes place is user-controllable through parameters.

Compared to the `Est` plugin, there is no data filtering performed here which means that the plugin should rather be used on consistent and fairly reliable data (eg. on existing gene predictions, cDNA or EST cluster alignements rather than simple EST alignments that would be better handled using the `Est` plugin).

The low-level entries allow to directly modify every edge of the underlying prediction graph of EUGÈNE as the (now obsolete) `User` plugin allowed. The weights of all signals edges (transcription start and stop, translation start and stop, splice sites, insertions and deletions) and contents edges (exons, introns, UTR, UTR introns and intergenic regions) can be directly modified using this plugin.

The sensor is activated by using the value 1 for the parameter `Sensor.AnnotaStruct.use` in the parameter file.

Here is an example of AnnotaStruct parameters definition :

```
AnnotaStruct.FileExtension    gff
AnnotaStruct.Exon*            10
AnnotaStruct.Intron*          10
AnnotaStruct.CDS*             10
AnnotaStruct.StartType        p
AnnotaStruct.Start*           1
AnnotaStruct.StopType         p
AnnotaStruct.Stop*            1
AnnotaStruct.AccType          p
AnnotaStruct.Acc*             1
AnnotaStruct.DonType          p
```

```
AnnotaStruct.Don*              1
AnnotaStruct.TrStartType       p
AnnotaStruct.TrStart*          1
AnnotaStruct.TrStopType        p
AnnotaStruct.TrStop*           1
Sensor.AnnotaStruct.use        1         # Use AnnotaStruct sensor
Sensor.AnnotaStruct            1         # Sensor priority
```

Input files format The plugin reads a GFF format file. Each line in this file forms an elementary information which is directly interpreted by the plugin independently of other lines. A GFF line is formed by a sequence of separated fields: sequence name, source, feature, start, end, score, strand and frame. The sequence name and source fields are ignored by the plugin and can be set to user informative values.

Each line may either represent a high-level or a low-level information. Low-level informations use specific features for specifying which signals and contents edges should be modified. For signals, the following features are recognized:

- `trStart`: for transcription starts.
- `trStop`: for transcription stops.
- `start`: for translation starts (ATG).
- `stop`: for translation stops.
- `acc`: for acceptor splice sites.
- `don`: for donor splice sites.
- `ins`: for insertion (frameshift).
- `del`: for deletion (frameshift).

In this case, the start and the strand field are used to indicate the signal position. The score field is used to indicate the weight that will be used to modify the existing weight. It is either a floating point value between `-1e999` and `1e999` (that match $-\infty$ and ∞ respectively in the format used) or a floating point between 0.0 and 1.0 preceded by the letter p (like probability).

1. In the first case, the score indicated is directly added to the weight of the signal edge (that corresponds to the fact that the signal is used). The other signal edge is unmodified.
2. In the second case, the score s that appears after the p is treated as a (conditional) probability. The edge that corresponds to the fact that the signal is used receive a weight $\log(s)$ and the other edge $\log(1-p)$.

For contents edges, the following features are recognized:

- `exon`: for coding exons.
- `intron`: for introns separating coding exons.
- `utr5`: for 5' UTR (untranslated terminal regions).
- `utr3`: for 3' UTR.
- `utr`: for both 5' or 3' UTR.
- `intronutr`: for UTR introns.

The start and end fields together with the strand field delimit the region considered. All corresponding contents edges will be modified by the weight indicated in the score field.

High-level information may either be used to express knowledge about potential *transcribed sequences* or potential *coding sequences*. For information about CDS regions, the following features may be used:

- `E.Init`: for an initial exon, this will automatically expand in the weight modification of a translation start and a donor site at the corresponding extremities on the indicated strand (using parameters `AnnotaStruct.Start*` and `AnnotaStruct.Don*` respectively as weights) and the contents modification for the exon in the corresponding frame and strand (using the score indicated in the `AnnotaStruct.CDS*` parameter).
- `E.Intr`: for an intermediary exon, this will automatically expand in the weight modification of a donor and an acceptor site at the corresponding extremities on the indicated strand (using parameters `AnnotaStruct.Don*` and `AnnotaStruct.Acc*` respectively as weights) and the contents modification for the exon in the corresponding frame and strand (using the score indicated in the `AnnotaStruct.CDS*` parameter).
- `E.Term`: for a terminal exon, this will automatically expand in the weight modification of an acceptor and a stop signal at the corresponding extremities on the indicated strand (using parameters `AnnotaStruct.Acc*` and `AnnotaStruct.Stop*` respectively as weights) and the contents modification for the exon in the corresponding frame and strand (using the score indicated in the `AnnotaStruct.CDS*` parameter).
- `E.Sngl`: for a single exon gene, this will automatically expand in the weight modification of a translation start and stop signal at the corresponding extremities on the indicated strand (using parameters `AnnotaStruct.Start*` and `AnnotaStruct.Stop*` respectively as weights) and the contents modification for the exon in the corresponding frame and strand (using the score indicated in the `AnnotaStruct.CDS*` parameter).
- `UTR5`, `UTR3`, `UTR`: although not part of the CDS, some gene predictors may predict UTR (non coding part of exons). These 3 features allow to inject this information by respectively reweighting a transcription start, stop or both using the corresponding `AnnotaStruct.TrStart*`, `AnnotaStruct.TrStop*` parameters and then by reweighting the UTR5, UTR3 or both contents edges (using the `AnnotaStruct.CDS*` parameter).
- `Intron`: equivalent to `intron` except that the weight used comes from the `AnnotaStruct.CDS*` parameter).

For information about transcribed sequences, the following features are recognized:

- `E.Any`: any exon in the biological sense *i.e.* either an exon or a UTR in the EUGÈNE sense. Frame is typically unknown (in this case, all coding frame in the indicated strand are considered). The corresponding contents region are modified accordingly to the `AnnotaStruct.Exon*` parameter.
- `E.First`: the first biological exon (containing UTR and possibly part of CDS too). A transcription start signal is weighted according to the `AnnotaStruct.TrStart*` parameter value. The UTR5 and coding exon contents edge are reweighted according to the `AnnotaStruct.Exon*` parameter value.
- `E.Last`: the last biological exon. A transcription stop signal is weighted according to the `AnnotaStruct.TrStop*` parameters value. The UTR3 and coding exon contents edge are reweighted according to the `AnnotaStruct.Exon*` parameter value.
- `E.Extreme`: used for a biological exon on the extremity (either first or last). A transcription start and stop are generated at each respective extremities according to the `AnnotaStruct.TrStart*` and `AnnotaStruct.TrStop*` parameter values. The UTR5, UTR3 and coding exon contents edges are reweighted according to the `AnnotaStruct.Exon*` parameter value.

Here is a high-level CDS based example:

```
ATSYNO FGENESH E.Init  3 33 0 + 3
ATSYNO FGENESH E.Term 45 75 0 + 3
```

Filtering input information No filtering beside syntax checking.

Integration of information The underlying graph edges are directly modified as indicated.

Post analyse No post analyse.

Graph No plotting.

2.3.4.2 `Sensor.IfElse`

Description This plugin is used to combine the predictions of two existing plugins. It listens to a first plugin. For each possible predictable item, if this plugin predicts something then this prediction is used. If the plugin does not predict anything, then the output of the second plugin is used.

The plugin needs only two parameters to be informed: `IfElse.SensorIf` and `IfElse.SensorElse` which indicate the names of the two slave plugins. The two slave plugins will be loaded with an instance number equal to one plus the instance number of the IfElse sensor itself (allowing for nested IfElse).

Here is an example of IfElse parameters definition which uses the NG2 Sensor if it predicts something or else the SPred sensor.

```
IfElse.SensorIf         NG2
IfElse.SensorElse       SPred
Sensor.IfElse.use       1    # Use IfElse sensor
Sensor.IfElse           1       # Sensor priority
```

In this case, since the IfElse is loaded as a first plugin (instance 0), the two slave plugins will be instanciated as instance number one. The parameters for the 2 plugins must therefore be suffixed by `[1]`.

Input files format No input files needed beyond those used by the slave sensors.

Filtering input information No filtering.

Integration of information The “If” plugin is called. For each of the possible information type (signal and contents), if nothing is predicted by it, the prediction of the second plugin is used instead.

Post analyse No post analyse beyond the post analyze in the slave plugins.

Graph Nothing beyond the plotting in the slave plugins.

2.3.4.3 `Sensor.Riken`

Description The plugin allows to exploit 5'/3' EST extracted from the extremities of full-length cDNA. This type of data was produced by the Riken institute for *Arabidopsis thaliana*. By mapping such EST to the genomic sequence, it is possible to know the positions where a gene (transcript) must start (5' side) and stop (3' side). The plugin assumes that this mapping has been done and that the coordinates of the extremities of the 5' and 3' EST of full-length clones have been determined before hand.

The sensor is activated by either :

- the `-R` argument
- the value 1 for the parameter `Sensor.Riken.use` in the parameter file.

The plugin is controled by several parameters, most of which control sanity checks (see below). The `Riken.RAFLPenalty*` parameter controls the amount of penalty used to force EUGÈNE to predict a gene on a region defined by a valid EST pair.

Here is an extract of Riken parameters definition :

```
Riken.Min_est_diff              100
Riken.Max_overlap               60
Riken.Max_riken_length          60000
Riken.Max_riken_est_length      3000
Riken.Min_riken_length          120
Riken.Min_riken_est_length      10
Riken.StrandRespect             0
Riken.RAFLPenalty*              -120
Sensor.Riken.use                1     # Use Riken sensor
Sensor.Riken                    7        # Sensor priority
```

Input files format A file with extension `.riken` is read. Each line must contain the positions of the extremities of the match of the 5' EST then the name of the 5' EST, the same thing for the 3'EST and finally the name of the clone.

Here is an exert of a typical `.riken` file:

```
417757  418379  AV826766        418902  419330  AV796216        0907A18
341382  342036  AU235278        340748  341549  AU225941        1201K23
40318   40969   AV821185        38800   39323   AV781490        0208M10
309757  310341  AV830906        308043  308392  AV813791        0980B11
387624  388227  AU236666        387383  387834  AU227623        1514C21
148345  148909  AV822910        147090  147960  AV783778        0513A17
```

Filtering input information The plugin uses several parameters that control sanity checks on the input data.

- the `Riken.Max_riken_length` parameter controls the maximum length for a transcript. If an EST pair defines a transcript with a length of more than this number of base pairs, then it is ignored. A typical value is 60kb (for *Arabidopsis thaliana*).
- the `Riken.Min_riken_length` parameter controls the minimum length for a transcript. If an EST pair defines a transcript with a length lower than this number of base pairs, then it is ignored. A typical value is 120b (for *Arabidopsis thaliana*).
- the `Riken.Max_riken_est_length` parameter controls the maximum length of the genomic sequence matching one EST. If either the 5' or the 3' EST exceed this length, then the EST pair is rejected. A typical value is 3kb (for *Arabidopsis thaliana*).

- the `Riken.Min_riken_est_length` parameter controls the minimum length of the genomic sequence matching one EST. If either the 5' or the 3' EST are below this length, then the EST pair is rejected. A typical value is 10 bp (for *Arabidopsis thaliana*).
- the `Riken.StrandRespect` parameter controls whether the 5'/3' information available for the EST is taken into account or ignored. If this parameter is set to `0`, then the information is ignored and the prediction of a gene is "forced" in the region but with no constraint on the strand. Otherwise, and if the 5'/3' EST pair is separated enough to decide the strand, then the prediction of a gene is forced on the strand detected.
- the `Riken.Min_est_diff` parameter controls the minimum distance of separation between the 5' and 3' EST (computed as the sum of the distances of the left and right extremities of the two genomic sequences mapping the 2 EST) that is sufficient to deduce the strand of the gene. A typical value is 100 bp (for *Arabidopsis thaliana*).
- the `Riken.Max_overlap` parameter controls how information on "overlapping" regions is handled. If two EST pairs define transcribed regions with a large overlap (larger than the parameter value), then it is likely that they refer to the same gene (as far as they are detected as being on the same strand). In this case, the two EST pairs are taken as one (merged by taking the leftmost extremity as the new left extremity and the rightmost as the right extremity). If the two overlapping regions are not on the same strand, then they are considered as inconsistent and the orientation is forgotten.

  If the two regions have a small overlap (lower than the parameter value), then it is likely that there are 2 different genes with overlapping UTR. Because EUGÈNE cannot predict overlapping UTR, then the extremities are modified so that they do not overlap anymore.

Integration of information Basically, when a genomic region is validated, the plugin forces EUGÈNE to predict one single gene in the region. This is done by penalizing all tracks but the intergenic track just before and after the gene extremities and by penalizing the intergenic track on the genomic region itself. If the strand is also considered as detected, then all tracks on the other strand are also penalized. Although an infinite (eg. `-1e999` in the current double format) penalty would seem more appropriate, we advocate for a strong finite penalty to avoid stupid uselss predictions in case of data inconsistency.

Post analyse No post analyze.

Graph The Riken information is plotted on the output graph as two small corners delimiting the region on the intergenic track. The corners are colored differently according to the strand detected for the transcribed region.

2.3.4.4 `Sensor.User`

The user plugin is superseded by the AnnotaStruct[1] plugin. It is conserved for compatibility reasons.

Description The sensor allows to seamlessly modify EUGÈNE underlying graph weighting using a small langage that can modify the weights of signals and contents edges in the graph. The plugin priority is significant here because it may directly set signal weights to a specific value (to eg. remove a signal prediction). Contents edges are just penalized (a weight is added to the existing weight).

The sensor is activated by either :

- the `-U` argument

[1] Also know as the BarbaTruc plugin.

- the value 1 for the parameter `Sensor.User.use` in the parameter file.

Here is an example of User parameters definition :

```
Sensor.User.use  1       # Use User sensor
Sensor.User      10         # Sensor priority
```

Input files format A small language is used to indicate which edges should have their weight modified. Signal weight modification is indicated by the keywords `start`, `stop`, `acceptor` and `donor`. Each line should indicate the type of signal edge which should be modified, the strand (either `f` or `r`), the position and the weight p to use for the signal. p can take any value between 0.0 and 1.0. The two signal edges (corresponding respectivelyto the fact the the signal is used or not) are respectively set to $\log(p)$ and $\log(1-p)$.

For contents edges, the keywords `exon`, `intron`, `utr3`, `utr5` and `intergenic` are used. Then the strand is indicated (either `f` or `r`), then the region where contents edge weights should be modified (as an interval `[begin..end]`). Finally, the weight modification p is given as a floating point value. Any floating point value can be used including two special values denoted `infinity` and `-infinity` (with an obvious meaning).

Information on signals (stop/start/splice sites) use the following syntax:

```
<type> <strand> <position> <value>
```

where:

- `<type>` can be start, stop, acceptor, donor
- `<strand>` can be f for forward and r for reverse
- `<position>` is a position on the sequence
- `<value>` is the cost associated with the signal. It can range from 0.0 to 1.0. A value of 0.0 means that no site of this type exists at this position. The cost will override any existing cost.

Informations on regions have the following syntax:

```
<type> <strand>{<phase>} [<start>..<end>] <value>
```

where

- `<type>` can be exon, intron, utr3, utr5, intergenic
- `<strand>` if either f for forward or r for reverse
- `<phase>` is only used for exons and introns. The phase of an exon indicates the position of the codons wrt. to the start/end of the sequence (modulo 3, same as reported by BlastX). The phase of an intron specifies how the codon is split by the splice site: between 2 codons, after 1 nucleotode or after 2. Basically, if you don't care of the intron "phase" just put the same information for all 3 phases.
- `<start>` and `<end>` are positions on the sequence that indicate the scope of the information.
- `<value>` is a score that can take any floating point value plus the extra value "infinity" that represent "minus infinity". This cost will be added to the existing cost. A positive cost helps predicting something consistent with the information, a negative one prevents from predicting something consistent with the information. Thus a cost "infinity" effectively forbids the prediction.

Here is an example:

```
utr5 f [2..28] 10.0
acceptor f 2966 0.0
```

which delete any existing acceptor prediction at position 2966 on the forward strand and favors UTR5 prediction on the 2-28 region.

Filtering input information No filtering.

Integration of information The underlying graph edges are directly modified as indicated. Pay attention that signal edge weights are *set* to the value indicated while contents edge weights are just incremented by the value indicated. The signal behavior is useful to be able to delete an existing signal.

Post analyse No post analyse.

Graph No plotting.

2.3.5 `Others plugins`

2.3.5.1 `Sensor.GCPlot`

Description The GCPlot sensor allows to add to the graphical representation a plot of basic composition statistics on the sequence. The sensor is activated by setting the parameter `Sensor.GCPlot.use` to `1` in the parameter file. The composition statistics represented can be arbitrarily chosen. For example, the GC%=$\frac{G+C}{A+T+G+C}$ is selected by setting `GCPlot.Up` to `GC` and `GCPlot.Over` to `ATGC`. Statistics on the 3rd base of each codon are automatically computed and plotted.

The color (integer between 0 and 8), the smoothing window width and specific zooming factors can be given. The zooming factor for the 3rd base in each codon is zoomed using specific zooming factor `GCPlot.Zoom3`

Here is an example of a GCPlot parameter definition :

```
GCPlot.Up          GC
GCPlot.Over        ATGC
GCPlot.Smooth      98
GCPlot.Color       5     # light green
GCPlot.Zoom        2.0
GCPlot.Zoom3       1.0
Sensor.GCPlot.use  1  # use GCPlot sensor
Sensor.GCPlot      1     # sensor priority
```

Input files format No input file.

Integration of information This sensor does not influence prediction.

Post analyse No post analyse.

Graph The composition statistics is plotted on the intergenic (IG) track. The same statistics computed on the 3rd position of each codon is plotted on the 6 exonic tracks.

2.3.5.2 **`Sensor.GFF`**

Description The GFF sensor allows to add to the graphical representation an annotation provided in a GFF format. Note that the provided GFF annotation could be an EUGÈNEprediction given in GFF format (obtained using the $-pg$ argument). This could allow to visualise two predictions on the same graph.

For a sequence, the plugin reads the annotation from one file whose name is derived from the sequence name by adding the `.gff` suffix. The sensor is activated by either :

- the `-G` argument
- the value 1 for the parameter `Sensor.GFF.use` in the parameter file.

Here is an example of GFF parameters definition :

```
Sensor.GFF.use  1       # Use GFF sensor
Sensor.GFF      1          # Sensor priority
```

Input files format The file `.gff` describes an annotation for a sequence. The format of a line is : `<seqname> <source> <feature> <start> <end> <score> <strand> <frame>`. Seqname, source and score fields are ignored.

Example:

```
seqName  EuGene  Utr5    1       199     0       -       .
seqName  EuGene  Utr5    340     359     0       +       .
seqName  EuGene  Init    360     393     0       +       2
seqName  EuGene  Intr    596     732     0       +       0
seqName  EuGene  Intr    830     876     0       +       1
seqName  EuGene  Intr    961     1286    0       +       1
seqName  EuGene  Intr    1396    1478    0       +       2
seqName  EuGene  Intr    1573    1648    0       +       0
seqName  EuGene  Intr    1757    1818    0       +       0
seqName  EuGene  Intr    1962    2057    0       +       2
seqName  EuGene  Intr    2145    2306    0       +       2
seqName  EuGene  Term    2491    2607    0       +       0
seqName  EuGene  Utr3    2608    2626    0       +       .
```

Note: only exons are plotted, this file is parsing by the frame field (no '.' in the frame field).

Filtering input information No filter.

Integration of information This sensor does not affect prediction.

Post analyse No post analyse.

Graph Orange horizontal lines are plotted on the exon tracks.

Documentation of the Plotter sensor

2.3.5.3 **Sensor.Plotter**

Description The Plotter sensor allows to add to the graphical representation the GC%, the GC3% and the two quotients A/T+A and T/T+A.

The sensor is activated by the value 1 for the parameter `Sensor.Plotter.use` in the parameter file.

Here is an example of Plotter parameters definition :

```
Plotter.GC          1         #
Plotter.GC3         1         # 0 -> no plot  -  1 -> plot
Plotter.A|T/A+T     1         #
Sensor.Plotter.use  1      # Use GFF sensor
Sensor.Plotter      1         # Sensor priority
```

Input files format No input files needed.

Integration of information This sensor does not affect prediction.

Post analyse No post analyse.

Graph The GC% is plotted as a thin turquoise line on the intergenic track. The GC3% is plotted as a thin turquoise line on each exon tracks. The T/T+A quotient is plotted as a thin orange line on the forward intron track. The A/T+A quotient is plotted as a thin orange line on the reverse intron track.

2.3.5.4 **Sensor.Tester**

Description The Tester sensor allows to evaluate signal sensors. For a sequence, the plugin reads the truth gene coordinates (note only one complete gene) in GFF format from one file whose name is derived from the sequence name by adding the `.gff` suffix.

Depending of the value of the parameter `Tester.Make`, two independant tests could be done. If `Tester.Make` is set to TEST, the positive positions (where the sensor detects a signal) are analysed (compared to the thruth). The results are written in a file `test.<sensorName.gff>`.

If the parameter `Tester.Make` is set to SPSN, the positions of the canonical coding of the considered signal (ATG for Start; TAG, TAA, TGA for Stop; AG for acceptor; GT, GC for donors) are analysed. The considering signal is defined following the value given at the `Tester.SPSN.Eval` parameter. Four variables are computed:

- TP, number of True Positive
- FN, number of False Negative
- FP, number of False Positive
- TN, number of True Negative

With these variables, two others are evaluated:

- Sn = TP/(TP+FN), sensitivity
- Sp = TP/(TP+FP), specificity

All these variables are computed for all score value given by the sensor. Each value is in turn considered as a threshold (if the score is higher than the threshold the information is considered as positive).
The values of the variables are put on stdout. Here is an example.

```
Thres.          Nb      TP      FP      TN      FN      Sens.           Spec.
-3.7297         1       144     16776   230     0       0.851064        100
-3.68888        1       144     16775   231     0       0.851114        100
-3.61192        1       144     16774   232     0       0.851164        100
-3.57555        1       144     16773   233     0       0.851215        100
[...]
```

Where Thres. is the threshold value, Nb is the number of observation of the threshold as a score.

To be plotted, specificity and sensibility are also written in the file Sensor.<sensorName>.SpSn and if the Tester.SPSN.Eval parameter is set to SPLICE, in the files Sensor.<sensorName>.Acc (for acceptor only), Sensor.<sensorName>.Don (for donor only).
Note that specificity and sensitivity are written in the .SpSn, .Acc, .Don files, only if TP+FP and TP+FN are higher than Tester.SPSN.MinNumbers. This to avoid Sp and Sn based on small effective.

The sensor is activated by the value 1 for the parameter Sensor.Tester.use. A parameter Tester.Sensor indicates which sensor to test. An other parameter Tester.Sensor.Instance defines wich instance (see the 2.2.1 Loading plugins section) of sensor to consider.

Here is an example of Tester parameters definition:

```
Tester.Make             SPSN      # SPSN, TEST
Tester.Sensor           EuStop
Tester.Sensor.Instance  0
Tester.SPSN.MinNumbers  100       # greater than 0
Tester.SPSN.Eval        STOP      # START, STOP, SPLICE
Sensor.Tester.use       1         # use Tester sensor
Sensor.Tester           1         # sensor priority
```

Input files format The file .gff describes the truth coordinates of only one complete gene in GFF format. The format of a line is :<seqname> <source> <feature> <start> <end> <score> <strand> <frame>. Seqname, source, score and frame fields are ignored.

Example:

```
seqName EuGene  UTR5    866     885     0       +       .
seqName EuGene  E.Init  886     931     0       +       0
seqName EuGene  E.Intr  1014    2366    0       +       1
seqName EuGene  E.Term  2444    2481    0       +       1
seqName EuGene  UTR3    2482    2632    0       +       .
```

Note : Feature field must be UTR5, UTR3, E.Init, E.Intr, E.Term or E.Sngl (UTR states are optional).

Output files format For the parameter Tester.Make set to TEST, a file (test.<sensorName.gff>) is created if it does not exist. For each predicted signals the Tester sensor write one line in the output file. The format of this line is : <seqname> <source> <feature> <score> <start> <end> <strand> <frame> <T/F> <state>.

Where:

- <seqname> is the 7 first characters of the sequence file name.

- <source> is the name of the tested sensor.
- <feature> is the feature type name (can be ‘Start’, ‘Acc’, ‘Don’ and ‘Stop’).
- <start> is the predicted signal position.
- <end> is always ‘.’.
- <score> is the score given by the tested sensor.
- <strand> is ‘+’ for forward and ‘-’ for reverse.
- <frame> is always ‘.’.
- <T/F> is ‘True’ for real site and ‘False’ for the others.
- <state> is the real state according to the predicted signal position (can be ‘IG’, ‘UTR’, ‘ExonF’, ‘ExonR’, ‘IntronF’ or ‘IntronR’).

Here is an extract of `test.NG2.gff` :

```
[...]
seqName     NG2     Acc      835       .   -2.26       -       .   False       IG
seqName     NG2     Acc      869       .  -15.03       +       .   False      UTR
seqName     NG2     Acc      918       .  -10.96       -       .   False    ExonF
seqName     NG2     Don      931       .   -0.02       +       .    True    ExonF
seqName     NG2     Don      962       .  -33.06       -       .   False  IntronF
seqName     NG2     Don      973       .  -27.76       +       .   False  IntronF
seqName     NG2     Don     1011       .   -4.10       -       .   False  IntronF
seqName     NG2     Acc     1013       .   -0.10       +       .    True  IntronF
seqName     NG2     Acc     1050       .   -7.52       +       .   False    ExonF
seqName     NG2     Don     1050       .  -27.76       +       .   False    ExonF
[...]
```

For the parameter `Tester.Make` set to SPSN, the file `Sensor.<sensorName>.SpSn` contains on each line a value of specificity and sensitivity for a threshold taken from the lowest to the highest.
For splice detectors sensors, two other files are also written `Sensor.<sensorName>.Acc` (acceptor only), `Sensor.<sensorName>.Don` (donor only) with the same format.

Filtering input information No filter.

Integration of information This sensor does not affect prediction.

Post analyse No post analyse.

Graph No plot.

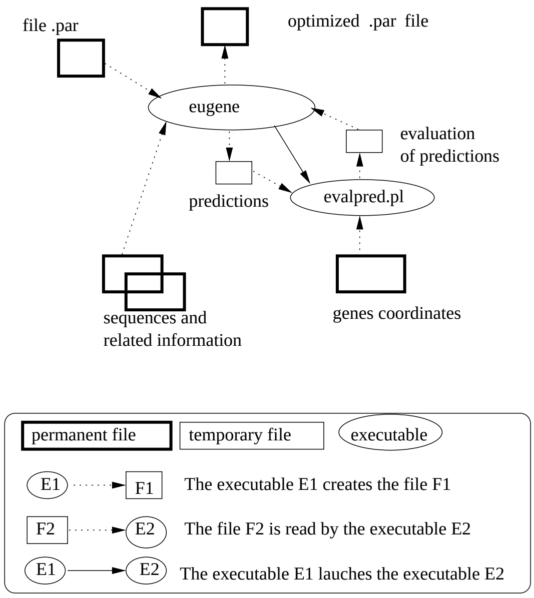

Figure 2.1: Input and output files for parameters optimization

2.4 Optimization of Plugins parameters

The value of some numerical plugins parameters (specified in the parameter file with a name finishing with an '*') can be optimized on a reference set of sequences (with their related information) for which genes positions are known. The idea is to adapt the values of parameters to increase as much as possible the quality of prediction of genes and exons. The figure 2.1 details the general function of the software with input and ouput files.

The optimization can be lauched with the `-Z` argument on the command line or with the `ParaOptimization.Use` parameter set to `1`.

After updating the parameter file `eugene.par` (which sensors to use,...), the software is lauched with the usual command line specifying as argument the reference sequences to consider. At the end, the software creates a new parameter file called `eugene.<date>.OPTI.par` (for example, `eugene.30Sep-2003.OPTI.par`) with the new value for the optimized parameters.

For parameters optimization, the inputs to be specified in the parameter file are:

- the parameters to optimize with their value domain,
- the optimization algorithm to use: genetic algorithm, Line Search, genetic algorithm and Line Search,
- the parameters of the optimization algorithm, and for the Line Search algorithm complementary information on the parameters to optimize (initial value, step of discretization, ...),
- a file with the coordinates of genes for the sequences set.

Here is a simplified example of optimization parameters definition in the parameter file.

```
#################################################################
################## PARAMETERS OPTIMIZATION ####################
#################################################################
ParaOptimization.Use           1
ParaOptimization.TrueCoordFile Araset.coord
```

```
ParaOptimization.Algorithm     GENETIC+LINESEARCH
ParaOptimization.Test          FALSE
ParaOptimization.Trace         1
#
ParaOptimization.NbParameter   3
#
ParaOptimization.Para.Name[0]   NStart.startP*
ParaOptimization.Para.Min[0]    0.001
ParaOptimization.Para.Max[0]    15
#
ParaOptimization.Para.Name[1]   NStart.startB*
ParaOptimization.Para.Min[1]    0.001
ParaOptimization.Para.Max[1]    15
#
ParaOptimization.Para.Name[2]   EuStop.stopP*
ParaOptimization.Para.Min[2]    0
ParaOptimization.Para.Max[2]    6
#
################## Genetic ####################################
Genetic.NbRun 2
Genetic.NbGeneration 20
Genetic.NbElement 50
Genetic.Seed 4
Genetic.CrossOverProbability 0.6
Genetic.MutationProbability 0.2
Genetic.SelectionType 1         # 0: roulette wheel
#                                 1:stochastic remainder without replacement
Genetic.ScalingType 1           # 0: no scaling
#                                 1: Sigma Truncation scaling
#                                 2: Power Law scaling
Genetic.Sharing 0.9             # 0: no sharing
#                                 1: sharing, looking for clusters which best
#                                    elt fitness is at least n% of the overall
#                                    best element of the population
Genetic.Clustering 1
Genetic.Elitism 0.9             # 0: none
#                                 n: elitism; keeps the best elt if no sharing,
#                                    and keeps the best elt of each cluster
#                                    which best_elt fitness is at
#                                    least n% of the overallbest elt if sharing
Genetic.SA.Mutation FALSE          # Simulated Annealing mutation
Genetic.SA.CrossOver FALSE         # Simulated Annealing crossover
#
#
######### LINESEARCH ##########################################
LineSearch.NbMaxCycle 1
LineSearch.NbMinCycle 1
LineSearch.NbMaxStab 2
LineSearch.DivInter 10
LineSearch.Alpha 0.6
LineSearch.EvolutionMini 0.001
LineSearch.Seed ALEA
#
LineSearch.NbCluster 2
LineSearch.Cluster[0] LINKED
LineSearch.Cluster[1] IDENTICAL
#
LineSearch.Para.Step[0]         0.001
LineSearch.Para.Init[0]         7.5
LineSearch.Para.MinInit[0]      0.001
LineSearch.Para.MaxInit[0]      15
LineSearch.Para.Cluster[0]      0
#
LineSearch.Para.Step[1]         0.001
LineSearch.Para.Init[1]         7.5
LineSearch.Para.MinInit[1]      0.001
LineSearch.Para.MaxInit[1]      15
LineSearch.Para.Cluster[1]      0
#
LineSearch.Para.Step[2]         0.001
LineSearch.Para.Init[2]         3
LineSearch.Para.MinInit[2]      0
LineSearch.Para.MaxInit[2]      6
LineSearch.Para.Cluster[2]      1
#
```

2.5 Command line flags

- `b`: activates the plugin `Sensor.BlastX`.
- `B`: postprocessing activation of the plugin `Sensor.BlastX`.
- `c`: controls how successives PNG images overlap (parameter `Output.golap`). It must be followed by the number of overlapping nucleotides between 2 successives PNG images. Default is heuristically determined based on resolution and number of nuc. per image.
- `d`: activates the plugin `Sensor.Est`.
- `D`: allows to specify a value to a parameter (syntax: -D<para>=<value>).
- `E`: enables EST and cDNA post-predition analysis (parameter `Est.PostProcess`) of the Est sensor: after each transcript prediction, all matching EST are analyzed and the consistency of the EST with the prediction is analyzed. At the end, the number of bases of the exon/intron structure predicted which are consistent with at least one EST/cDNA are reported.
- `f`: the frameshift penalty. A large value prevents EuGène from predicting frameshifts (the default).
- `g`: graph required.
- `G`: activates the plugin `Sensor.GFF`.
- `h`: help
- `l`: controls the number of nucleotides that will appear on a single image (parameter `Output.glen`). Default is min (6,000 length to visualize). The length to visualize is computed from the value given to -u and -v (default is all sequence)
- `m`: activates the plugin `Sensor.MarkovIMM` and specifies the filename of the set of Markov models that will be used by the MarkovIMM sensor (parameter `MarkovIMM.matname`).
- `M`: activates the plugin `Sensor.MarkovProt` and specifies the filename of the set of Markov models that will be used by the MarkovProt sensor (parameter `MarkovProt.matname`).
- `n`: followed by 0 1 or 2. Indicates the way the score are normalized accross the possibles states (phase 1,2,3,-1,-2,-3, introns and intergenic states).
  - `0`: no normalization
  - `1`: normalize accross all states
  - `2`: normalize each coding phase w.r.t. to the non coding score only.

  Default is 1 (parameter `Output.normopt`). Does not affect prediction, only text/graphical output.

- `o`: allows to offset the nucleotide position of the prediction (parameter `Output.offset`). That is, the prediction for nucleotide at position i of the given sequence is printed as nucleotide $i+$ the offset. Useful to perform prediction on an extracted sequence without loosing the original position.
- `O`: allows to specify an output directory (the textttOutput.Prefix parameter value).
- `p`: controls the format of the textual outpout (parameter `Output.format`). May be d (detailed), l (long), s (short), h (html), g (gff) or a (araset format). Default is l.
- `r`: activates the plugin `Sensor.Repeat`.
- `R`: activates the plugin `Sensor.Riken`.

- `s`: forces non partial gene mode prediction. This forbids predictions that start and end in intergenic mode and therefore prevents the occurrence of partial gene structures on the border of the sequence. Useful if EUGÈNE lacks context around the gene and you know a single (or only complete) gene appears on the sequence. In practice this simply sets the parameters `EuGene.ExonPrior`, `EuGene.IntronPrior`, `EuGene.FivePrimePrior` and `EuGene.ThreePrimePrior` to 0.0.
- `t`: activates the plugin `Sensor.Homology`
- `u`: controls the part of the sequence whose prediction will be displayed in the graphical output (parameter `Output.gfrom`). It must be followed by the position of the 1st nuc. which will be plotted on graphical output (allows for zoom'in). Default is 1.
- `U`: activates the User information sensor (parameter `Sensor.User.use`). This sensor reads user informations stored in .user file. These informations use a small language. The language can contain two types of statements. Statements on signals (translation start, splice sites) and on the sequence itself (coding, non coding...).
- `v`: controls the part of the sequence whose prediction will be displayed in the graphical output (parameter `Output.gfrom`). It must be followed by the position of the last nuc. which will be plotted on graphical output (allows for zoom'in). Default is the sequence length.
- `w`: followed by half the size of the smoothing window for the scores (parameter `Output.window`). Default is 48. Does not affect prediction, only graphical output.
- `x`: controls the horizontal resolution of the PNG images generated by EuGene (parameter `Output.resx`). Default is 900.
- `y`: controls vertical resolution of the PNG images generated by EuGene (parameter `Output.resy`). Default is 400.
- `Z`: allows to ask for a parameters optimization (equivallent to set the `ParaOptimization.Use` parameter to 1).

Index of command line flags for sensors

For more information, have a look to www-bia.inra.fr/T/EuGene. This gives a rough idea of EUGÈNE reliability and the meaning of the graphical output (PDF file, poster on EUGÈNE).