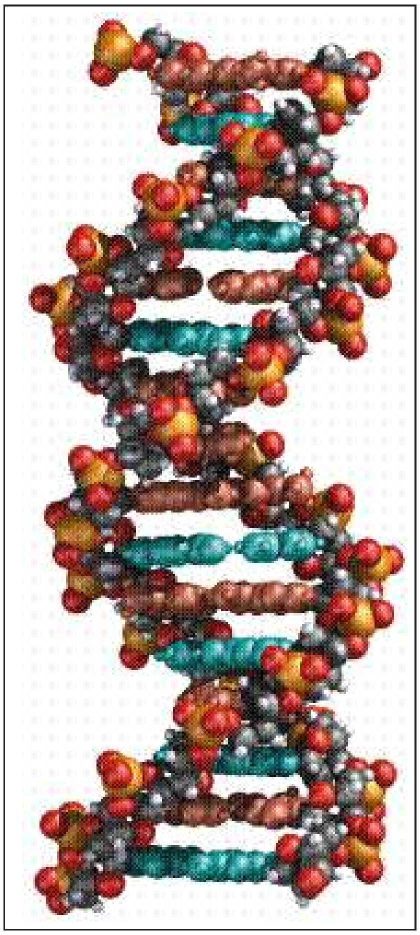
geneid *handbook*



Enrique Blanco García

Roderic Guigó Serra

August 22, 2000

Contents

Chapter 1

What's geneid?

`geneid` is a program to predict genic elements as splice sites, exons or genes, along eukaryotic DNA sequences. `geneid` offers also some rudimentary support to integrate predictions for multiple sources. The program is written in **ANSI C**, and runs on `UNIX` operating systems.

Installation, setup and usage of `geneid` is very easy, and there is a range of options to configure output predictions and program behaviour.

This short handbook explains how to install, and use `geneid`. It is still under construction, and some sections are missing.

`geneid` source code, compiled binaries and documentation are available under the GNU GENERAL PUBLIC LICENSE.

Comments and questions are welcome.

1.1 Where to obtain geneid

`geneid` home page is `http://www1.imim.es/software/geneid/` and `geneid` distribution can be obtained through anonymous ftp to `ftp.imim.es` in the directory `/pub/software/geneid`.

1.2 Help on geneid

If you need help, send a message to `geneid@imim.es`.

Chapter 2

Signals, exons and genes

`geneid` is a program to predict genes in anonymous genomic sequences designed with a hierarchical structure. In the first step, splice sites, start and stop codons are predicted and scored along the sequence using Position Weight Matrices (PWMs). In the second step, exons are built from the sites. Exons are scored as the sum of the scores of the defining sites, plus the the log-likelihood ratio of a Markov Model for coding DNA. From the set of predicted exons, the gene structure is assembled maximizing the sum of the scores of the assembled exons.

2.1 Signals

`geneid` recognizes the following signals using Position Weight Matrices.

Acceptors :
Last intron dinucleotide mandatory ***AG***. Acceptors occure at the boundaries intron/exon – Splice site –

Donors :
First intron dinucleotide mandatory ***GT***. Donor occure at the boundaries exon/intron – Splice site –

Start Codons :
First exon first tri-nucleotide mandatory ***ATG***. Start Codons initiate translation – Translation signal –

Stop Codons :
TGA, TAA, TAG. Stop Codons terminate translation end of the last exon from the genes. – Translation signal –

2.2 Exons

Exons are built from the set of predicted signals.

Frame and Remainder

In `geneid` exons are Open Reading Frames delimited by signals. Exons are assigned a Frame and Remainder, which in `geneid` are understood in the following way:

The *Frame* of an exon is the number of nucleotides (0, 1 or 2) from the first nucleotide in the exon sequence to the first nucleotide in the first complete codon translated from the exon sequence.

The *Remainder* of an exon is the number of nucleotides left (0, 1 or 2) after the last complete codon has been defined from the exon sequence given the frame. The remainder is fully determined by the length and the frame of the exon.

Thus, if E and F are two consecutive exons, they can be asssembled into a gene if (frame(F) + remainder(E)) MOD 3 = 0.

Exon types

`geneid` builds the following exons:

First :
Boundary signals: Start - Donor.
Frame $\in \{0\}$. Remainder $\in \{0,1,2\}$.

Internal :
Boundary signals: Acceptor - Donor.
Frame $\in \{0,1,2\}$. Remainder $\in \{0,1,2\}$.

Terminal :
Boundary signals: Acceptor - Stop.
Frame $\in \{0,1,2\}$. Remainder $\in \{0\}$.

Single Exon Genes :
Boundary signals: Start - Stop.
Frame $\in \{0\}$. Remainder $\in \{0\}$.

2.3 Genes

Gene structures are frame compatible sequences of predicted exons. `geneid` is able to predict one or more genes along a DNA sequence, in both strands simultaneously.

Chapter 3

Installing `geneid`

The `geneid` distribution contains several directories and files. Source code, compiled binaries for a number of architectures and operating systems, and documentation files are included in the distribution.

The distribution is archived and compressed in a single file using the command `tar -zcvf`. The compressed file name is `geneid.tar.gz` (or something similar depending on compiled binaries included). The `geneid` files can be extracted following these instructions:
Under `UNIX`:

- Type `gzip -d geneid.tar.gz`
- Type `tar -xvf geneid.tar`

Under `LINUX`:

- Type `tar -zxvf geneid.tar.gz`

After executing these commands, the directory `geneid` will be created in your working directory.

3.1 File Listing

The `geneid` distribution contains the following files and directories

`bin/`
: compiled binaries

`docs/`
: documentation: this handbook, `geneid` homepage and other documentation.

`include/`
: `geneid.h`. The `geneid` header file.

`objects/`
: object files after compiling the source code.

`param/`
: Parameter files for several organisms. So far, *Homo Sapiens*, and *Drosophila melanogaster*.

`samples/`
: A sequence example and results.

`src/`
: source code of **`geneid`** program.

`GNULicense`
: This software is registered under GNU license.

`Makefile`
: This file is required to build **`geneid`** binary file.

`README`
: This file contains information on how to install and run **`geneid`** history and contact information.

3.2 Compiling `geneid`

After moving into the **`geneid`** directory, you use **`make`** to compile **`geneid`**.
Under `UNIX`, do the following

- `cd geneid`
- `make`

This will generate the **`geneid`** executable file within the **`bin`** subdirectory. Type **`geneid -h`** to test the binary file has been correctly created.

Chapter 4

Running geneid

To run geneid on UNIX systems type:
    geneid [options]... <seqfile>
where <seqfile> is a fasta formated DNA sequence. Predictions are sent to the standard output. In addition, geneid requires a organims specific parameter file (which can be broadly used within the phylogenetic group to which the organism belongs. For instance, the human parameter file can be used as well to predict genes in genomic sequences from vertebrate organisms). The directory param contains parameter files for different organisms.

The parameter file can be specified in three different ways:

1. By default geneid will use a file named param.default which must be in the same directory that the executable geneid program.

2. Through an environmental variable GENEID.

3. Using the command line option -P <filename>

4.1 Command line options

A number of command line options can be used to alter geneid default behaviour.

4.1.1 Prediction options

-P <filename> Parameter filename. Already discussed

-v *Verbose* option.
    Sends to standard error run-time information.

-W Predict only on the forward (+) strand.

-C Predict only on the reverse (–) strand.
    By default, geneid predicts on both strands.

-o Predict exons only. Do not assemble genes.

-O <filename> Do not predict exons.
    Assemble genes from an external set of exons in <filename> (which may be predicted

using a different program, for instance.) Predicted exons must be in `gff` format. The gene model (see section 8.1) specifies the criteria under which the predicted exons (or other genic elements) may be assembled. (see chapter 6).

`-R` <`filename`> Includes external evidence in the predictions.
<`filename`> must be in gff format. This options may be useful when re-annotating a sequence, and the gene structure of some of the sequence regions is already known. The user do not want this structure to be overriden by the new predictions, but may want prediction be produced in other—no so well annotated—regions of the sequence. (see chapter 7).

`-S` <`filename`> Use information from database searches.
Not implemented yet.

4.1.2 Output options

By default, `geneid` output shows only the predicted genes along the query DNA sequence. Through a number of options, however, other predicted genic elements—intermediate, not necessarily in the final prediction— can be included in the output.

`-b` Include predicted Start Codons in the output.

`-d` Include predicted Donor sites in the output.

`-a` Include predicted Acceptor sites in the output.

`-e` Include predicted Stop Codons in the output.

`-f` Include predicted Initial Exons in the output.

`-i` Include predicted Internal Exons in the output.

`-t` Include predicted Terminal Exons in the output.

`-s` Include predicted Single Exon Genes in the output.

`-x` Include all predicted exons sorted in the output.
– This option is equivalent to `-fits`. –

For large sequences these options may generate very large amounts of output.

In addition to the default format, elements in the output can be printed in a number of other different ways.

`-G` `gff` format. (see `http://www.sanger.ac.uk/Software/GFF/GFF_Spec.shtml`)

`-X` eXtended format.
For each exon, `geneid` prints the sequence of the boundary signals and the corresponding scores. (see chapter 5)

Chapter 5

Examples

This section compiles a number of `geneid` runs and the corresponding output. The sequence used through the examples is `Test.fasta` in the `samples` directory of the `geneid` distribution. All these examples have been run using the *Drosophila* parameter file `Dros.param` in the `param` directory.

5.1 Default gene predictions

`geneid` default behaviour is to produce the optimal gene structure predicted along the input sequence. For example, if `geneid` is executed in the following way:

```
geneid -v -P Dros.param samples/Test.fasta > output
```

The output will contain

```
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:17:01 2000
# Optimal Gene Structure. 3 genes. Score = 140.109665
# Gene 1(Forward). 2 exons. 155 aa. Score = 36.870944
   First    6090    6155    1.40 + 0 0   2.15   3.63   12.34   0.00 AA    0: 22 gene_1
Terminal    6881    7276   35.47 + 0 0   3.33   2.30   97.73   0.00 AA   23:155 gene_1

>10|GeneId_predicted_protein_1|156_AA
MVVKAVCVINGDAKGTVFFEQESSGTPVKVSGEVCGLAKGLHGFHVHEFGDNTNGCMSSG
PHFNPYGKEHGAPVDENRHLGDLGNIEATGDCPTKVNITDSKITLFGADSIIGRTVVVHA
DADDLGQGGHELSKSTGNAGARIGCGVIGIAKV*

...
```

5.1.1 Output formats

If `gff` format is selected through the `-G` option, the output is:

```
geneid -vG -P Dros.param samples/Test.fasta > output
```

```
## gff-version 2
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:38:23 2000
# Optimal Gene Structure. 3 genes. Score = 140.109665
# Gene 1(Forward). 2 exons. 155 aa. Score = 36.870944
10 geneid_v1.0    First   6090  6155   1.404 + 0 gene_1 #  AA-0  AA-22
10 geneid_v1.0 Terminal   6881  7276  35.467 + 0 gene_1 # AA-23 AA-155
...
```

If the extended format is selected through the `-X` option:

`geneid -vGX -P Dros.param samples/Test.fasta > output`

the output is

```
## gff-version 2
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:38:15 2000
# Optimal Gene Structure. 3 genes. Score = 140.109665
# Gene 1(Forward). 2 exons. 155 aa. Score = 36.870944
10 geneid_v1.0 Start 6090 6092 2.150 + . 0 # GATTAATTCATTCGAAATGG
10 geneid_v1.0 Donor 6155 6156 3.627 + . 0 # GAGGTGAGAA
10 geneid_v1.0 First 6090 6155 1.404 + 0 gene_1 # AA-0 AA-22
10 geneid_v1.0 Acceptor 6880 6881 3.331 + . 0 # AATCCCTTCATCCCGTCCACAGAG
10 geneid_v1.0 Stop 7274 7276 2.297 + . 0 # AGGTCTAAG
10 geneid_v1.0 Terminal 6881 7276 35.467 + 0 gene_1 # AA-23 AA-155
...
```

5.2 Including signals and exons in the output

5.2.1 Signals

With the `-bdae` options, the user may select the signals to be included in the output

For example, the following command instructs `geneid` to include predicted Start Codons in the output:

`geneid -vob -P Dros.param samples/Test.fasta > output`

```
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:39:18 2000
# Starts(+) predicted in sequence 10: [0,21539]
   Start       22       24 -4.390 + TGCCTGTTGCCAACTGATGT
   Start       67       69 0.194 + TAGAACACTCCTTACAATGC
   Start      109      111 -1.193 + TGTATCAGGATTATCCATGG
   Start      140      142 -2.500 + AGTAGTTCAAATTTGAATGA
   Start      143      145 0.302 + AGTTCAAATTTGAATGATGA
   Start      275      277 -2.182 + TGATTGATTAAGGGGGATGA
   Start      296      298 -1.119 + AAAGGTCTTGTAAAGCATGA
```

Record structure:

1. Signal type
2. Left position
3. Right position
4. Score
5. Strand
6. Region which contains the predicted signal

The output using the `gff` format:

```
geneid -vobG -P Dros.param samples/Test.fasta > output
```

```
## gff-version 2
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:41:24 2000
# Starts(+) predicted in sequence 10: [0,21539]
10 geneid_v1.0 Start 22 24 -4.390 + . # TGCCTGTTGCCAACTGATGT
10 geneid_v1.0 Start 67 69 0.194 + . # TAGAACACTCCTTACAATGC
10 geneid_v1.0 Start 109 111 -1.193 + . # TGTATCAGGATTATCCATGG
10 geneid_v1.0 Start 140 142 -2.500 + . # AGTAGTTCAAATTTGAATGA
10 geneid_v1.0 Start 143 145 0.302 + . # AGTTCAAATTTGAATGATGA
10 geneid_v1.0 Start 275 277 -2.182 + . # TGATTGATTAAGGGGGATGA
10 geneid_v1.0 Start 296 298 -1.119 + . # AAAGGTCTTGTAAAGCATGA
...
```

Record structure:

1. Sequence name

2. Data source - `geneid`
3. Signal type
4. Left position
5. Right position
6. Score
7. Strand
8. Frame(not used)
9. Nucleotide sequence of the signal

5.2.2 Exons

All predicted exons can be included in the output with the following command:

```
geneid -vox -P Dros.param samples/Test.fasta > output
```

```
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Tue Jun 13 10:43:19 2000
# Exons(x) predicted in sequence 10: [0,21539]
   First       20      110  -8.76 - 0 1 -3.67 -0.10  1.24  0.00 31 MDNPDTAPKMCPVGIVRSVLGRLGNFKLYIa
   First       20       83  -8.99 - 0 1 -3.67  1.02 -1.00  0.00 22 MCPVGIVRSVLGRLGNFKLYIa
Internal       20      118  -5.51 - 2 1 -3.67  4.14  3.00  0.00 34 gaFPMDNPDTAPKMCPVGIVRSVLGRLGNFKLYIa
Internal       20       91  -9.84 - 1 2 -3.67 -1.15  0.14  0.00 25 cTKDVPCRHCKECSRTARELQIVHca
Internal       20       91 -10.18 - 2 1 -3.67 -1.15 -0.72  0.00 25 caPKMCPVGIVRSVLGRLGNFKLYIa
Internal       20       83 -10.24 - 0 1 -3.67 -1.07 -1.00  0.00 22 MCPVGIVRSVLGRLGNFKLYIa
Internal       20       83 -10.44 - 2 2 -3.67 -1.07 -1.49  0.00 22 atVPCRHCKECSRTARELQIVHca
...
```

Record structure:

1. Exon type
2. Beginning position
3. End position
4. Score
5. Strand
6. Frame
7. Remainder
8. Acceptor/Start score
9. Donor score
10. Markov Model coding potential score

11. Score derived from similarity to known coding regions (not yet implemented)
12. Number of aminoacids encoded in the exon sequence
13. Amino acid sequence of the translated exon

The output using `-G` (`gff` format):

```
geneid -voxG -P Dros.param samples/Test.fasta > output
```

```
## gff-version 2
## source-version: geneid v 1.0 -- imim.es
## Sequence 10 - Length = 148847 bps
## date Thu Jun 22 09:34:10 2000
# Exons(x) predicted in sequence 10: [0,21539]
10 geneid_v1.0    First 20 110  -8.763 - 0
10 geneid_v1.0    First 20  83  -8.985 - 0
10 geneid_v1.0 Internal 20 118  -5.515 - 2
10 geneid_v1.0 Internal 20  91  -9.836 - 1
10 geneid_v1.0 Internal 20  91 -10.181 - 2
10 geneid_v1.0 Internal 20  83 -10.240 - 0
...
```

Record structure:

1. Sequence name
2. Data source – `geneid`
3. Exon Type
4. Beginning position.
5. End position.
6. Score
7. Strand
8. Frame

Chapter 6

Using `geneid` to assemble predicted exons into gene structures

`geneid` can be used to chain predicted exons, obtained through sources other than `geneid` into coherent gene strucutures. `geneid` will use the score associated to the input predicted exons; then it will predict the gene structure that maximizes the sum of the scores of the assembled exons. It makes sense to use `geneid` in this context, only when the sum of the scores of the assembled exons is a meaningful score.

The set of predicted exons should be given in an additional file in `gff` format; `geneid` is instructed to read this file with the command line option `-O` <`filename`> (see section 4.1.1). This file must be sorted by the starting position of the exons (column 4 in `gff`)

Actually, this file may include predicted genic elements of any type, such as promoters, repeats, etc. and not only exons. A gene model in the parameter file (see chapter 8) lists the rules according with which the chaining of the predicted elements can occur. Elements types are identified through the feature name in the `gff` file. This name must be used in the gene model to refer to the element. The rules specify essentiallly which elements can be chained together and whithin which distances.

The current implementation of this option is not completely satisfactory. The main problem is that `geneid` assumes any predicted genic element to have a frame (see section 2.2). This makes no sense for intergenic elements, such as promoters, CpG Islands, etc. The remainder is automatically computed from the frame and the length of the element. In `geneid`, we assume the frame of the a gene (the frame of the first exon) and the remainder of the last exon from a gene to be zero, and thus we assign frame zero to predicted intergenic elements and force them to be of length multiple of three. This, again, is not satisfactory.

Chapter 7

Introducing external evidence into the predictions

geneid allows to merge the program predictions with external evidence, such as already annotated genes. The external evidence is read from an additional file in gff format. geneid is instructed to read this file with the command line option -R <filename> (see section 4.1.1). This file must be sorted by the starting position of the elements (column 4 in gff).

There is a difference between using the options -O <filename> and -R <filename>. In the first case, only the elements given in <filename> are used to assemble the gene predictions according to the specificied gene model. In the second case, the gene predictions are built from both the elements in <filename>, plus the exons predicted by geneid.

If the elements in <filename> are assigned an score, then they "compete" with the elements (exons) predicted by geneid (if any) to be in the final gene model. For instance,

```
AE002566         exon_traping       Internal    66255    66323    3.27       +         1
```

means that this exon will compete with geneid exons to be in the final prediction.

If no score is given for the element, i.e. the score field in the gff file is a dot ".", then this element is assumed to be mandatory in the final gene prediction (unless a conflicting element with no score is also given in the gff file.) For instance,

```
AE002566         exon_traping       Internal    66255    66323    .       +        1
```

means that this exon must be in the final prediction.

The frame can be specificied in the column 8 of the gff file, or a "." can be used as a wildcard in this position, meaning "any frame". This is not the same that un-enforcing frame consistency. geneid always enforces frame consistency (see section 6).

What geneid currently does when "." is used to specify the frame, is to generate three equivalent elements in the three possible frames, and computing the corresponding remainder in each case. For instance, the record in the input gff file

```
AE002566         exon_traping       Internal    66255    66323    3.27       +         .
```

is internally expanded to

AE002566	exon_traping	Internal	66255	66323	3.27	+	0
AE002566	exon_traping	Internal	66255	66323	3.27	+	1
AE002566	exon_traping	Internal	66255	66323	3.27	+	2

and the optimal one given the rest of the elements - either predicted by `geneid` or given externally - will be included in the final predictions.

`geneid` will use the group field, if given, in the `gff` file. In such a case, `geneid` predictions will not be included within the limits of the group. If no group is given, external evidence and `geneid` predictions may be intermixed in the same group.

Chapter 8

geneid parameter file

`geneid` depends on a parameter file to build the predictions. The parameter file contains mostly the description of the probabilistic model on which the predictions are based. It also contains the so-called gene model, the set of rules to chain gene elements (such exons) into gene predictions (the gene model appears at the bottom of the parameter file). Through the usage of the gene model and the `-O` and `-R` options, `geneid` offers some limited support for the integration of predictions from multiple sources.

8.1 Gene Model

The gene model is the set of rules that specifies the constrains under which predicted gene elements can be chained in the final gene prediction. This constrains refer to the succession of elements in the prediction and to the distances between these elements. The gene model is a set of three column records.

For instance, the rule

`First+:Internal+`	`Internal+:Terminal+`	`40:11000`

indicates that elements (exons) of type `Internal` in the forward strand and of type `Terminal` in the forward strand can be chained inmediatly after elements of type `First` in the forward strand or of type `Internal` in the forward strand. The third column indicates the interval distance at which they can be chained. In this case the predicted elements must be at least 40 bp, and at most 11000 bp apart. This rule specifices the constrains governing intronic conexions. The equivalent rule for intronic conexions in the reverse strand is

`Terminal-:Internal-`	`First-:Internal-`	`40:11000`

The following rule specifices the constrains governing "intergenic" connections:

`Promoter-:First-:Terminal+`	`First+:Promoter+:Terminal-`	`2000:9999999`
`Promoter+`	`First+`	`50:4000`
`First-`	`Promoter-`	`50:4000`

The first rule indicates that elements of type `First` and `Promoter` predicted in the forward strand, and elements of type `Terminal` in the reverse strand can be chained inmediately after elements of type `First` or `Promoter` in the reverse strand, or elements of type `Terminal` in the forward strand. The minimum distance between them in order to be chained is 2000, while there are no restrictions on the maximum distance (9999999 taken as infinity). The second rule specifies that elements of type `First` in the forward strand can also be chained after elements of type `Promoter` predicted in the forward strand. The third rule is the equivalent in the reverse strand.

Only conections explicit in the Gene Model are allowed. Thus, if the user wants elements of type `promoter` to appear necessarily upstream elements of type `First`, the gene model should be instead

`Promoter-:Terminal+`	`Promoter+:Terminal-`	`2000:9999999`
`Promoter+`	`First+`	`50:4000`
`First-`	`Promoter-`	`50:4000`

The current version of `geneid` predicts elements with types `First`, `Internal`, `Terminal`, and `Single` (for single-exon genes). Other elements in the additional files provided with the `-O` or `-R` options are ignored if they do not appear in the gene model.

The gene model does not specify which elements terminate genes (which are assumed to be represented by the group field of the `gff` file). Gene termination is coded within `geneid` `First+` and `Terminal-` start genes, while `Terminal+` and `First-` terminate genes. `Single+` and `Single-` start and terminate genes.

Chapter 9

How to read the `geneid` code

9.1 Code Summary

The structure of the `geneid` code is modular. Subroutines encapsulate different functions. Most of them can be modified independently without changes in the rest of the code. The main program is `geneid.c` routines.
Follows the list of subroutines in `geneid`:

`account.c`

Management of diverse statistics on memory usage and execution time.

`BackupGenes.c`

Save genes between two consecutive sequence fragments.

`BuildInitialExons.c`

Build Initial exons.

`BuildInternalExons.c`

Build Internal exons.

`BuildSingles.c`

Build Single Exons genes.

`BuildSort.c`

Build an exon ordered list.

`BuildTerminalExons.c`

Build Terminal exons.

`CookingGenes.c`

Processing predicted genes for pretty printing.

`CorrectExon.c`

Recompute boundary positions from exons.

`Dictionary.c`

Management of GeneModel hash table.

`DumpHash.c`

Management of the hash table used to share genes between sequence fragments.

`FetchSequence.c`

Process input sequence(uppercase, reverse sequence, ...).

`genamic.c`

Assembling genes using predicted exons.

`geneid.c`

Main program.

`GetSitesWithProfile.c`

Predict signals(Splice Sites and Start Codons).

`GetStopCodons.c`

Search for Stop Codons.

`Output.c`

Management of printing output predictions.

`PrintExons.c`

Print predicted exons—formatted.

`PrintSites.c`

Print predicted signals—formatted.

`readargv.c`

Read options from command line.

`ReadGeneModel.c`

Read GeneModel from parameter file.

`readparam.c`

Read parameter file.

`ReadSequence.c`

Read input sequence from file.

`ReadSR.c`

Read list of similarity regions to known coding sequences (not yet fully implemented)

`RecomputePositions.c`

Recompute reverse signal positions.

`RequestMemory.c`

Request memory for storing the predictions.

`ReverseSequence.c`

Reverse input sequence.

`ScoreExons.c`

Score predicted exons using Markov chains from parameter file.

`SearchEvidence.c`

Search through the list of external evidence.

`SortExons.c`

Sort all exons by the start position.

`SwitchFrames.c`

Exchange frame/remainder of reverse predicted exons.

`SwitchPositions.c`

Exchange left/right signals of reverse exons.

`Translate.c`

Translate predicted exons to protein.

9.2 Main definitions

Definitions and data structures are `geneid.h`, in the `include` directory. These are the main `geneid` parameters:

- `LENGTHSi`.
  Length of fragments from the input sequence.
- `OVERLAP`.
  Length of overlap between two consecutive sequence fragments.
- `RSITES`.
  Ratio of predicted sites per fragment from input sequence. – The maximum number of predicted sites per fragment will be LENGTHSi / RSITES –
- `REXONS`.
  Ratio of predicted exons per fragment from input sequence. – The maximum number of predicted exons per fragment will be LENGTHSi / REXONS –
- `NUMEEVIDENCES`.
  Maximum number of exons/evidences to read from file to assembly genes with `-O|R` options (when using).
- `MAXGENE`.
  Maximum number of predicted genes.

9.3 Running very large sequences

The distributed version of `geneid` works fine on sequences as large as the human Chrosome 22 (34 Mbp). Running `geneid` on larger sequences may produce nonsense predictions. In such a case, you will need to increase the following definition values in the `include/geneid.h` file and recompile `geneid`.

- `RBSITES`.
  Ratio of saved sites per fragment from input sequence. – The maximum number of saved sites two between fragments will be (Length of Sequence) / RBSITES –
- `RBEXONS`.
  Ratio of saved exons per fragment from input sequence. – The maximum number of saved exons between two fragments will be (Length of Sequence) / RBEXONS –

9.4 Speed and Memory needs

`geneid` memory requirements are 50 Mb plus aproximately 1Mb per Mb of input sequence.
In Pentium III running at 550 Mhz, sequences are analyzed at about 2Mb per minute.

Chapter 10

About the autors

Roderic Guigó, Steen Knudsen, Neil Drake and Temple F. Smith contributed to the first version of `geneid`. This version was developed at the Molecular Biology Computer Research Resource (Dana Farber Cancer Institute — Harvard University). It was never distributed, but an e-mail server was latter set up at the Biomolecular Engineering Research Center (Boston University). Kathleen Klose and Steen Knudsen developed the server.

`geneid` has been rewritten from scratch at the Institut Municipal d'Investigació Mèdica in Barcelona. Rewritting of this version started in late 1999. The new code is mostly by Enrique Blanco and Roderic Guigó. Moisès Burset has also made some contributions to the code. Xavier Messeguer from the Universitat Politècnica de Catalunya has contributed to some aspects of the design.

10.1 About the name

During the years `geneid` has been referred to as GeneID, GeneId, geneID, Geneid, etc. We have decided for the simplest, most sensible name: `geneid`

10.2 About the version

Although this is not the first version of `geneid`, it is the first version we are releasing. Therefore, we decided this to be `geneid` version 1.0

10.3 Acknowledgements

This work was partially supported by grant from "Plan Nacional de I+D", BIO98-0443-C02-01, and from the "Ministerio de Educación y Ciencia" (Spain).

geneid related papers

[1] R. Guigó, S. Knudsen, N. Drake, and T. F. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 226:141–157, 1992.

[2] S. Knudsen, R. Guigó, and T. F. Smith. GeneID—a computer server for prediction of genes in DNA sequences. In H.A. Lim, J.W. Fickett, C.R. Cantor, and R.J. Robins, editors, *Proceedings on the Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 545–553. World Scientific, 1993.

[3] R. Guigó. Assembling genes from predicted exons in linear time with dynamic programming. *Journal of Computational Biology*, 5:681–702, 1998.

[4] G. Parra, E. Blanco, and R. Guigó. Geneid in *Drosophila*. *Genome Research*, 10:511–515, 2000.