Руководство пользователя программы lossgainRSL v.6.20

Содержание

Общие сведения $\mathbf 1$

Программа lossgainRSL предназначена для предсказания потерь и приобретений генов между несколькими наборами видов. Она позволяет для фиксированного базового вида находить его гены, присутствующие и/или отсутствующие у нескольких наборов видов в соответствии с заданной логической функцией (предикатом). Элементарное условие всегда имеет форму «Ген X присутствует в наборе видов S », а предикат представляет собой булеву функцию, составленную из произвольного числа таких условий с помощью связок И, ИЛИ, НЕ. Наборы видов не обязательно являются таксономическими группами, а могут составляться по произвольным признакам; в частности, набор S может состоять из одного вида и любой вид может входить в несколько наборов. Присутствие гена в наборе видов S требует, чтобы он был не менее чем у заданного числа р видов из $S(p - nq)$ араметр набора S). Наличие гена X у какого-то вида понимается не только в смысле наличия в геноме этого вида гена X' – ортолога или близкого гомолога X, а с учётом синтении: в окрестности гена X должны присутствовать другие гены («свидетели»), соответственно ортологичные генам в окрестности X'. Число наборов видов и их состав, проверяемый предикат, число свидетелей и другие требования синтении, размеры окрестностей и степень близости гомологов являются параметрами программы, что позволяет без переделки применять её для широкого круга исследований.

Текущая версия программы lossgainRSL ориентирована прежде всего на виды и геномные данные, содержащиеся в базе данных Ensembl (http://www.ensembl.org/), однако в данном руководстве описаны и другие возможности, в частности, касающиеся использования данных GenBank (https://www.ncbi.nlm.nih.gov/genbank/) u OrthoDB (https://www.orthodb.org/). Программа написана на C++ и представляет собой утилиту командной строки Windows или Linux. Поддерживаются 32- и 64-битные версии ОС; учитывая требования к оперативной памяти, для решения задач с большим числом видов рекомендуется использовать 64-битную версию. Программа допускает параллельную мультипроцессорную обработку в среде МРІ, для чего в операционной системе должна быть установлена её реализация, соответствующая стандарту MPI v2.1 или выше.

Программа lossgainRSL разработана в Лаборатории математических методов и моделей в биоинформатике Института проблем передачи информации им. А.А. Харкевича Российской академии наук (http://lab6.iitp.ru), авторы – Рубанов Л.И., Селиверстов А.В., Любецкий В.А. Текущая версия исполняемых модулей lossgainRSL для Windows и исходного кода для Linux (по лицензии GNU GPL) доступны на странице http://lab6.iitp.ru/ru/lossgainrsl/ вместе с контрольным примером и настоящим руководством.

Минимальные требования программы: 32- или 64-битная версия ОС Windows или Linux (рекомендуется 64 bit), одно процессорное ядро с тактовой частотой 1 ГГц (рекомендуется 3-4 ГГц), объем оперативной памяти 2 ГБ (для большого числа видов рекомендуется 8 ГБ).

 $\overline{2}$ Установка и проверка

2.1 Версия для Windows

В среде Microsoft Windows установка состоит в загрузке со страницы программы архива с нужным вариантом исполняемого модуля и последующей распаковке содержимого в рабочую директорию. Имеется три варианта модуля:

- lossgainrsl.exe вариант для 32-битных версий Windows;
- lossgainrsl64nompi.exe однозадачный вариант для 64-битных версий Windows;
- lossgainrsl64.exe параллельный вариант для 64-битных версий Windows с MPI.

Для использования параллельного варианта программы в системе Windows необходимо установить свободно распространяемый продукт Microsoft MPI v10.0, которую можно загрузить с адреса: [https://www.microsoft.com/en-us/download/details.aspx?id=57467.](https://www.microsoft.com/en-us/download/details.aspx?id=57467) С другими версиями MPI работа исполняемого модуля не гарантируется.

Помимо этого, для работы исполняемых модулей программы в среде Windows может потребоваться установить пакет свободно распространяемых библиотек Microsoft Visual С++ 2008 SP1 соответствующей разрядности, которые можно загрузить с адресов: <https://www.microsoft.com/ru-ru/download/details.aspx?id=2092> (x64) <https://www.microsoft.com/ru-ru/download/details.aspx?id=5582> (x86).

Проверка работоспособности установленной программы осуществляется с помощью контрольного примера, доступного для загрузки на странице программы:

- 1. Загрузить архив example.zip с контрольным примером и распаковать его в рабочую директорию, где будет создана папка example.
- 2. Скопировать в эту папку выбранный вариант (или варианты) исполняемого модуля (см. выше).
- 3. Запустить командный процессор и перейти в эту папку.
- 4. В зависимости от проверяемого варианта программы, ввести одну из команд:
	- run 32-битная версия,
	- $run64$ 64-битная версия без МРІ,

run64mpi 64-битная версия с MPI (проверка на 2 процессорах).

5. В двух первых случаях через небольшое время будет выведено сообщение: Test completed OK.

Это свидетельствует о работоспособности проверяемого варианта программы.

6. При проверке версии с MPI будет выдано сообщение: Xs: Completed OK.

Следует сравнить содержимое полученного файла result.txt c имеющимся в папке примера образцом result4.txt (или result4.ref). Это удобно делать путём импорта в Excel. Полученный файл должен содержать такие же четырёхстрочные группы, но, возможно, в другом порядке. Различия в последнем столбце следует игнорировать.

2.2 Версия для Linux

Установка осуществляется путём компиляции исходного кода, доступного для загрузки на странице программы. Для использования параллельной версии программы в системе Linux должна быть предварительно установлена какая-либо реализация библиотек MPI v2.1 или выше; рекомендуется использовать текущую версию openMPI ($\frac{https://www.open-mpi.org/}{https://www.open-mpi.org/})$. Установка и проверка программы включает следующие шаги:

- 1. Загрузить в рабочий каталог дистрибутив программы, который обычно имеет имя lossgainrsl-*x.yy*.tar.gz, где *x.yy* – номер текущей версии.
- 2. Распаковать дистрибутив командой tar –xf *имя_файла*
- 3. Перейти в созданный каталог командой cd lossgainrsl-*x.yy*
- 4. В простейшем случае, когда компилятор C++ имеет в системе имя $g++$ и параллельная обработка не предполагается, достаточно ввести команду таке без параметров.
- 5. В иных случаях следует предварительно внести изменения в макебне (рекомендуется сохранить исходный файл, а работать с его копией). Нужно указать команду запуска компилятора C++ для построения программ с MPI ($cx =$) и используемые параметры компилятора (CXXFLAGS=). Несколько типичных вариантов в закомментированном виде имеются в начале существующего файла.
- 6. После внесения изменений в Makefile ввести команду make или make имя файла, если результат изменений сохранён в новом файле с указанным именем.
- 7. После успешной компиляции в каталоге . /bin должен быть создан исполняемый файл lossgainRSL. Если выданы сообщения об ошибках или предупреждения, их необходимо проанализировать: возможно, причина в некорректном содержании Makefile. Перед тем, как снова запускать make, следует выдать команду make clean для очистки следов предыдущего запуска.
- 8. Если программа успешно построена, выдать команду make test.
- 9. Через короткое время на консоль будет выдано сообщение Example completed OK, что свидетельствует о работоспособности программы.
- 10. Для проверки программы в режиме МРІ на двух процессорах выдать команду make mpitest. В этом случае оценка результатов проводится так же, как описано в предыдущем разделе, п. 6.

3 Синтаксис командной строки

В однопроцессорном режиме программа lossgainRSL запускается с помощью командной строки следующего вида:

lossgainRSL $[-x \mid -x]$ $[-q \mid -qq]$ $[-qM]$ $[-n]$ $\{ \text{\%name} = \text{value...} \}$ $[config]$ $[\text{outfile}]$

Здесь указано имя программы в Linux; в среде Windows указывается lossgainrsl (32битная версия), lossgainrs164nompi (64-битная без поддержки MPI) или lossgainrs164 (64-битная с поддержкой MS-MPI). За исключением имён файлов в Linux, регистр символов в командной строке безразличен. Все аргументы командной строки необязательные. Для получения краткой справки по формату командной строки программу следует запустить с любым из аргументов ? -? - h. Опции программы указываются одной буквой, которой предшествует символ - или /, а для параметров также и %. В текущей версии предусмотрены следующие опции:

- Выдавать только найденные гены базового вида и их ортологи у других видов, $-x$ даже если в файле конфигурации указан режим вывода синтеничных блоков генов, т.е. вместе с генами-свидетелями. И наоборот, чтобы выдавать гены вместе со свидетелями, даже если в конфигурационном файле выбран режим вывода только генов, данную опцию следует указать в форме -- х.
- Опция позволяет сократить объем протокола, выдаваемого программой на $pq-|p$ консоль. По умолчанию выдаётся максимально подробный протокол. Если указано - а, программа не выдаёт заданные в исходной конфигурации список видов и предикат. Если указано - ад, программа также не выдаёт подробности

чтения исходных файлов и поиска генов, а только конечный результат, так что объем выдачи минимальный.

- $-\Box M$ Эта опция позволяет управлять периодичностью вывода информации в процессе поиска генов базового вида, удовлетворяющих заданному предикату. Значение М указывает шаг, с которым на консоль выдаются порядковые номера проверяемых генов в каждой геномной последовательности (хромосоме, контиге и т.п.). Нулевое значение т вообще отменяет эту выдачу, что позволяет немного ускорить счёт. Если опция не указана, по умолчанию используется шаг 10.
- Если указать эту опцию, программа lossgainRSL будет принудительно $-n$ исполняться в однопроцессорном режиме, даже если используется версия с МРІ. Может использоваться при отсутствии или несовместимой версии библиотек МРІ, установленных в операционной системе. Если данная опция не помогает, имеется версия программы без MPI (доступна в разделе файлов для загрузки).
- $\frac{1}{2}$ тате=value Программа позволяет использовать в файле конфигурации символические обозначения числовых параметров в форме **name*, где *name* - произвольное имя. Значения всех параметров должны быть заданы в командной строке с помощью соответствующего количества таких аргументов, в произвольном порядке. Числовые значения справа от знака равенства могут быть целыми или дробными, в том числе с указанием десятичного порядка, например, 5.5е-6. Если команда используется в составе сценария (скрипта) Windows, знак процента перед именем параметра необходимо продублировать.

Опции могут появляться в произвольном порядке в любом месте командной строки. Первый аргумент, который не является опцией или параметром программы, рассматривается как имя файла конфигурации (config), второй – как имя выходного файла (outfile).

- Имя файла конфигурации, которое может включать путь к каталогу. Если в среде config Windows имя содержит пробелы или другие особо обрабатываемые символы, аргумент необходимо заключить в двойные кавычки. Если имя не указано, берётся файл config.ini в текущем каталоге. Формат и содержимое конфигурационного файла описаны в разделе 5 данного руководства.
- outfile Имя выходного файла, которое может включать путь к каталогу. Если в среде Windows имя содержит пробелы или другие особо обрабатываемые символы, аргумент необходимо заключить в двойные кавычки. Если имя не указано, создаётся файл result.txt в текущем каталоге. Описание выходного файла приведено в разделе 6.2 данного руководства.

Протокол работы lossgainRSL (6.1) выводится в стандартный поток *stdout* (по умолчанию – на консоль); его можно перенаправить в файл добавлением в конце командной строки аргумента >logfile1, где logfile1 - желаемое имя файла, которое может включать путь.

В программе предусмотрен отладочный режим, управление которым осуществляется через файл конфигурации. Протокол отладки выводится в стандартный поток *stderr* (по умолчанию также на консоль, что может быть неудобным). Чтобы перенаправить протокол отладки в

файл, в конец командной строки добавляется аргумент 2 >logfile2, где logfile2 – желаемое имя файла, которое может включать путь.

Программа lossgainRSL 64-битной версии обеспечивает параллельное выполнение в среде МРІ. Для запуска в мультипроцессорном режиме используется командная строка следующего вила:

mpirun -np P lossgainRSL $[-x]-x]$ $[-q]-qq]$ $[-qM]$ $[\n\%name = value...]$ $[jobfile]$ $[outfile]$

где P – число используемых логических процессоров. Здесь указаны имена программ в Linux; в среде Windows указываются mpiexec и lossgainrs164. Остальные аргументы командной строки описаны выше.

 \blacktriangleleft Описание входных данных

Предпочтительный режим работы с программой lossgainRSL предполагает использование входных данных из базы данных Ensembl (www.ensembl.org). Естественно, при этом набор рассматриваемых видов ограничивается представленными в этой базе данных (по состоянию на март 2019 г. в версию 95 базы позвоночных входят 162 вида). Возможности работы с видами, отсутствующими в Ensembl, обсуждаются в главе 7.

При использовании данных из Ensembl программе требуются, в зависимости от заданной конфигурации, два или три типа входных файлов, которые описываются в последующих разделах 4.1-4.3. Эти файлы данных можно получить двумя способами; выбор подходящего остаётся за пользователем программы:

- 1) Пользователь составляет нужный запрос на языке SQL и с помощью подходящей программы-клиента подает его для обработки на один из общедоступных MySQLсерверов Ensembl (дальнейшую информацию см. на странице http://www.ensembl.org/info/data/mysql.html). Этот способ подробно не описывается, т.к. рассчитан на квалифицированных пользователей, знающих язык SQL и изучивших схемы баз данных Ensembl. Кроме того, сложный SQL-запрос может обрабатываться слишком долго, так что публичный сервер будет его принудительно прерывать. В таких случаях пользователю предлагается загрузить и установить у себя зеркало всей базы данных, что предусмотрено её авторами, но сопряжено с трудностями.
- 2) Пользователь обращается к веб-интерфейсу инструмента для извлечения данных BioMart на сайте Ensembl (http://www.ensembl.org/biomart/martview/) и формирует нужные данные в интерактивном режиме. Это не требует специальных знаний, но несколько более трудоёмко. В некоторых случаях для слияния файла из частей могут дополнительно потребоваться внешние средства (команды ОС или подходящий текстовый редактор). Далее в разделах 4.1-4.3 излагается именно такой способ получения входных данных каждого типа.

Как описано в главе 7, программа lossgainRSL позволяет вместо информации из БД Ensembl использовать другие источники, в частности, GenBank и OrthoDB, но в этом случае пользователь сам должен сформировать нужные файлы вручную или с применением собственных программ/скриптов. Еще три вида альтернативных данных, которые поддерживает текущая версия программы, описаны в разделах 4.4-4.6.

4.1 Таблица генов

Эта таблица требуется для каждого вида, участвующего в задаче; она содержит идентификаторы всех белков и генов в геноме данного вида, координаты генов, их имена и аннотации. Таблица представляет собой текстовый файл в формате TSV (tab-separated values - поля, разделённые символом табуляции) и содержит в первой строке имена полей. Каждая последующая строка описывает один белок (для белок-кодирующих генов) или РНК (транспортную, рибосомную и т.д.). Упорядоченность строк произвольная. Имя таблицы генов должно совпадать с именем вида в конфигурационном файле, все таблицы генов должны находиться в одном каталоге и иметь одинаковое расширение имени.

Набор и порядок полей в разных таблицах генов может быть различным, но каждая таблица обязана, по меньшей мере, содержать поля, перечисленные в Табл. 1.

Табл. 1. Список полей таблицы генов.

Примечание: Поля, отмеченные *, строго говоря, не обязательны для работы lossgainRSL, но рекомендуются к включению в состав таблицы генов.

Указанные в таблице имена полей соответствуют версии 95 БД Ensembl; программа не предполагает, что эти имена неизменны, однако имя одного и того же поля должно быть одинаковым во всех таблицах генов. Точно так же, имена полей аналогичного содержания в выходном файле не обязательно будут совпадать с именами на входе. Все соответствия имён полей устанавливаются в файле конфигурации и при их изменении не потребуется изменять программу.

Примеры таблиц генов содержатся в каталоге genes в составе контрольного примера к программе (http://lab6.iitp.ru/ru/lossgainrsl/example.zip). Укажем последовательность действий для получения, например, файла Human.tsv:

- 1. Вызвать интерфейс BioMart (http://www.ensembl.org/biomart/martview/)
- 2. В выпадающем списке CHOOSE DATABASE выбрать Ensemble Genes 95 (или последующий выпуск).
- 3. В выпадающем списке CHOOSE DATASET выбрать Human genes (...).
- 4. Щелкнуть слева по ссылке Filters и развернуть справа пункт REGION.
- 5. Отметить опцию Chromosome/scaffold и выбрать в списке справа хромосому Y.
- 6. Щелкнуть слева по ссылке Attributes и отметить справа опцию Features, а затем развернуть ниже пункт GENE.
- 7. Убрать все имеющиеся пометки полей и затем последовательно выбрать все поля, перечисленные в [Табл. 1.](#page-6-1) Поля появятся в записи файла в том порядке, в котором они были отмечены.
- 8. Нажать вверху на кнопку Results. Справа указать Export all results to File, TSV и пометить Unique results only, затем нажать кнопку Go.
- 9. В папку пользователя для загрузки файлов из Интернет будет загружен файл с именем mart_export.txt. Переименовать его в Human.tsv и затем перенести в каталог, где будут находиться все таблицы генов.

4.2 Таблица ортологов

Эта таблица требуется для базового вида, а также для каждого вида-заменителя, участвующего в трёхвидовом условии (подробнее см. [5.5.6\)](#page-27-0). Представляет собой файл в TSVформате, но строка заголовка не требуется (а если есть – игнорируется, в том числе внутри файла). Каждая строка таблицы содержит два поля, в которых указаны идентификаторы двух ортологичных генов в Ensembl; первый из того вида, для которого строится таблица (т.е. базового или заменителя), второй – из какого-то другого вида, участвующего в задаче.

Подчеркнём, что таблица ортологов для базового вида или вида-заменителя должна содержать ортологи *каждого* гена этого вида *во всех* остальных видах, поэтому она может достигать большого объема. Упорядоченность строк произвольная. Имя таблицы ортологов должно совпадать с именем вида в конфигурационном файле; все такие таблицы должны находиться в одном каталоге и иметь одинаковое расширение имени.

Пример таблицы ортологов содержится в папке orthologs в составе контрольного примера к программе [\(http://lab6.iitp.ru/ru/lossgainrsl/example.zip\)](http://lab6.iitp.ru/ru/lossgainrsl/example.zip). Укажем последовательность действий для получения этого файла, Mouse.tsv:

- 1. Вызвать интерфейс BioMart [\(http://www.ensembl.org/biomart/martview/\)](http://www.ensembl.org/biomart/martview/)
- 2. В выпадающем списке CHOOSE DATABASE выбрать Ensemble Genes 95 (или позднее).
- 3. В выпадающем списке CHOOSE DATASET выбрать Mouse genes (...).
- 4. Щелкнуть слева по ссылке Filters и развернуть справа пункт REGION.
- 5. Отметить опцию Chromosome/scaffold и выбрать в списке справа хромосому Y.
- 6. Щелкнуть слева по ссылке Attributes и отметить справа опцию Homologues, а затем развернуть ниже пункт GENE.
- 7. Оставить отмеченным только поле Gene stable ID и свернуть пункт GENE.
- 8. Развернуть ниже пункт ORTOLOGUES и найти в нём второй вид, в котором берутся ортологи, например, Human. Отметить поле Human gene stable ID.
- 9. Щелкнуть слева по ссылке Filters, свернуть справа пункт REGION и развернуть MULTI SPECIES COMPARISONS. Отметить опции Homologue filters и Only, затем выбрать в выпадающем списке Orthologous Human Genes.
- 10. Нажать вверху на кнопку Results. Справа указать Export all results to File, TSV и пометить Unique results only, затем нажать кнопку Go.
- 11. В папку для загрузки файлов из Интернет будет загружен файл с именем mart_export.txt. Переименовать его в Mouse_Human.txt.
- 12. Щелкнуть слева по ссылке Attributes, убрать прежнюю отметку поля и проделать п.п. 8-12 со следующим видом в качестве второго, и т.д., пока не будут перебраны все вторые виды для данного первого.
- 13. Объединить все полученные выше файлы в произвольном порядке. Например, в командной строке Windows для этого используется команда: copy /a Mouse *.txt /b Mouse.tsv
- 14. Перенести полученный файл Mouse.tsv в папку, где будут находиться таблицы ортологов.

4.3 Таблица паралогов

Эта таблица нужна, если при выборе ортологичных генов должны учитываться не только ортологи, но и их паралоги (такой режим может быть задан в файле конфигурации). Таблица паралогов требуется для каждого вида, в котором могут браться паралоги, и представляет собой файл в TSV-формате; строка заголовка не требуется, а если она есть - игнорируется. Каждая строка таблицы содержит два поля, в которых указаны идентификаторы в Ensembl двух паралогичных генов одного и того же вида. Файл должен содержать все упорядоченные (т.е. включая перестановку) пары паралогичных генов данного вида, но очередность их безразлична. Имя таблицы паралогов должно совпадать с именем вида в конфигурационном файле, все такие таблицы должны находиться в одном каталоге и иметь одинаковое расширение имени.

Пример таблицы паралогов содержится в папке paralogs в составе контрольного примера к программе (http://lab6.iitp.ru/ru/lossgainrsl/example.zip). Укажем последовательность действий для получения этого файла, Mouse.tsv:

- 1. Вызвать интерфейс BioMart (http://www.ensembl.org/biomart/martview/)
- 2. В выпадающем списке CHOOSE DATABASE выбрать Ensemble Genes 95 (или позднее).
- 3. В выпадающем списке CHOOSE DATASET выбрать Mouse genes (...).
- 4. Щелкнуть слева по ссылке Filters и развернуть справа пункт REGION.
- 5. Отметить опцию Chromosome/scaffold и выбрать в списке справа хромосому Y.
- 6. Свернуть справа пункт REGION и развернуть пункт MULTI SPECIES COMPARISONS. Отметить опции Homologue filters и Only, затем выбрать в выпадающем списке Paralogous Mouse Genes.
- 7. Щелкнуть слева по ссылке Attributes и отметить справа опцию Homologues, а затем развернуть ниже пункт GENE.
- 8. Оставить отмеченным только поле Gene stable ID и свернуть пункт GENE.
- 9. Убедиться, что слева в пункте Attributes не осталось никаких других полей, кроме Gene stable ID, в противном случае снять соответствующие пометки справа в пунктах GENE или ORTHOLOGUES. Развернуть справа пункт PARALOGUES и отметить в нём поле Mouse paralogue gene stable ID.
- 10. Нажать вверху на кнопку Results. Справа указать Export all results to File, TSV и пометить Unique results only, затем нажать кнопку Go.
- 11. В папку для загрузки файлов из Интернет будет загружен файл с именем mart export.txt. Переименовать его в Mouse.tsv и перенести в папку, где будут находиться все таблицы паралогов.

4.4 Матрица сходства белков

Этот вид входных данных является альтернативой для таблиц ортологов и паралогов Ensembl (разделы 4.2–4.3). Заметим, что таблицы генов (4.1) тем не менее нужны для каждого вида,

участвующего в задаче, хотя в общем случае в них могут встречаться идентификаторы белков и генов не из БД Ensembl.

Требуется отдельный текстовый файл в формате TSV для каждого вида из задачи, причём строка заголовка не нужна, а если присутствует – игнорируется. Каждая информационная строка матрицы должна содержать четыре поля, перечисленные в [Табл. 2.](#page-9-1) Имя матрицы сходства белков должно совпадать с именем вида в конфигурационном файле; все матрицы должны находиться в одном каталоге и иметь одинаковое расширение имени.

Порядок записей в матрице сходства белков произвольный, но значения третьего и четвертого полей должны быть согласованными в пределах одной матрицы и для разных матриц. Например, если матрицы строятся с помощью программы BLASTP, следует всюду использовать одинаковые параметры, включая таблицу замен аминокислот (скажем, BLOSUM62). Уникальности записей не требуется, т.е. одна и та же неупорядоченная пара кодов белков может встречаться неоднократно; в таких случаях программа lossgainRSL использует наибольшее значение Raw score и наименьшее значение E-value.

Пример фрагмента матрицы сходства белков, импортированного в Excel, представлен на листе Xenopus_scores файла [http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx.](http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx) Такую матрицу можно сформировать с помощью собственного скрипта, вызывающего программу BLASTP, при условии, что предварительно из Ensembl или иного используемого источника получены аминокислотные последовательности для каждого кода белка из всех видов данной задачи.

4.5 Таблица ортологичных групп

Данные этого типа содержатся целиком в одной таблице и заменяют собой всю информацию разделов [4.1](#page-6-0)[–4.3,](#page-8-0) нормально получаемую из БД Ensembl. Таблица представляет собой один текстовый файл в формате TSV, который можно сформировать с помощью собственной программы, в частности, на основании данных БД OrthoDB [\(https://www.orthodb.org/\)](https://www.orthodb.org/). Физически таблица может быть разбита на несколько файлов, перечисленных в файле конфигурации; они будут логически объединены программой.

Первая строка таблицы должна содержать заголовки полей в последующих строках (за исключением пустых строк, разделяющих кластеры). Благодаря этому, порядок и набор полей в таблице могут быть достаточно свободными, но каждая строка должна включать по меньшей мере поля, перечисленные в $Ta6n$. 1, с одним исключением: вместо поля Protein stable ID с идентификатором белка требуется поле Species с идентификатором вида, который указан в файле конфигурации. Имена и порядок полей в строках таблицы не обязательно совпадают с указанными в [Табл. 1,](#page-6-1) правильное соответствие устанавливается с помощью файла конфигурации. Идентификаторы генов и последовательностей берутся из того источника, который использован для построения таблицы.

Подчеркнем, что здесь *одна* таблица содержит сведения о генах *всех* видов, участвующих в задаче. Важную роль играет упорядоченность строк в таблице. Строки, каждая из которых описывает один ген в каком-то виде, сгруппированы в виде *ортогрупп* из попарно ортологичных генов (относящихся к другим или тому же виду), а одна ортогруппа от другой отделяется пустой строкой. Порядок строк внутри ортогруппы не играет роли. Гены, не имеющие ортологов, исключены из таблицы.

Пример фрагмента таблицы ортогрупп, импортированного в Excel, представлен на листе Orthogroups файла [http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx.](http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx)

4.6 Таблица кластеров белков

Данные этого типа содержатся целиком в одной таблице и заменяют собой информацию об ортологах и паралогах, нормально получаемую из БД Ensembl (разделы [4.2,](#page-7-0) [4.3\)](#page-8-0). Однако таблицы генов [\(4.1\)](#page-6-0) требуются для каждого вида, участвующего в задаче. Таблица кластеров белков представляет собой один текстовый файл в формате TSV, который формируется с применением ранее разработанной программы кластеризации белков [\(http://lab6.iitp.ru/ru/pr_protclust/\)](http://lab6.iitp.ru/ru/pr_protclust/). Физически таблица может быть разбита на несколько файлов, перечисленных в файле конфигурации; они будут логически объединены программой.

Первая строка таблицы должна содержать заголовки полей в последующих строках (за исключением пустых строк, разделяющих кластеры). Порядок и набор полей в таблице могут быть достаточно свободными, но каждая строка должна содержать поле с идентификатором белка или гена, из числа указанных в [Табл. 1.](#page-6-1)

Подчеркнем, что здесь *одна* таблица содержит сведения о белках или генах для *всех* видов, участвующих в задаче. Важную роль играет упорядоченность строк в таблице. Строки, каждая из которых описывает один белок/ген в каком-то виде, сгруппированы в виде *кластеров* из попарно ортологичных генов/белков (в том числе из того же вида), а один кластер от другого отделяется пустой строкой. Порядок строк внутри кластера не играет роли. Ортологичность определяется по тому варианту сплайсинга, который даёт белки с наибольшим сходством, поэтому каждый ген встречается в таблице не более одного раза. Кластеры, состоящие из единственного гена (синглетоны), исключены из таблицы.

Пример фрагмента таблицы кластеров белков, импортированного в Excel, представлен на листе Gene_clusters файла [http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx.](http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx)

Файл конфигурации $\overline{\mathbf{5}}$

Файл конфигурации служит для управления работой lossgainRSL и описания решаемой задачи. Содержимое файла критически важно: как правило, ошибки при работе программы связаны с неправильным содержимым конфигурационного файла. Наличие файла конфигурации обязательно для запуска программы; его имя должно быть указано в командной строке либо совпадать с принятым по умолчанию (см. главу 3).

Файл создаётся в обычном текстовом формате; переносы строк запрещены. Если в строке конфигурационного файла содержится несколько полей, то разделителем полей служит один или более символов табуляции. Точка, запятая, пробел могут свободно использоваться внутри полей и не являются разделителем полей. Файл конфигурации может содержать комментарии, которые игнорируются программой. А именно, игнорируются строки, начинающиеся символами "; (точка с запятой) или "//" (двойная наклонная черта). Кроме того, игнорируется часть любой информационной строки, начинающаяся с символов "//" (двойная наклонная черта).

Файл конфигурации для программы lossgainRSL должен содержать пять секций, которые описываются в нижеследующих подразделах в порядке их расположения в файле. Другие секции конфигурационного файла могут присутствовать, но игнорируются программой. Каждая секция начинается со строки заголовка, которым служит имя секции в квадратных скобках. Содержимое секции занимает последующие строки вплоть до заголовка следующей секции или конца файла. Несколько вариантов конфигурационного файла имеются в составе контрольного примера (http://lab6.iitp.ru/ru/lossgainrsl/example.zip).

5.1 Секция [mode] – режим работы программы

Данная секция в большинстве случаев состоит из единственной информационной строки, в единственном поле которой в произвольном порядке указываются символы из перечисленных в Табл. 3, обязательно с соблюдением регистра. Смысл каждого символа объясняется в таблице. Не все символы могут сочетаться друг с другом; каждая группа взаимно альтернативных символов содержится между двумя горизонтальными линиями. Символы, отсутствующие в таблице, в том числе пробел, запятая, знак табуляции игнорируются и могут использоваться в качестве необязательных разделителей.

Табл. 3. Символы управления режимами lossgainRSL.

Примечание:

Режимы этой группы не альтернативны, а могут появляться в любом сочетании или вообще отсутствовать. Последнее означает, что единственным условием синтении является наличие генов-свидетелей в окрестности проверяемого гена с заданным радиусом, независимо от их порядка и ориентации.

Вслед за строкой, задающей режимы работы программы, могут идти одна или более строк с указанием объектов, для которых требуется подробная отладочная информация; в роли таких объектов могут выступать идентификаторы или порядковые номера генов, белков или геномных последовательностей. Каждая строка содержит один или несколько объектов, а в качестве разделителя используется знак табуляции. Эти данные интерпретируются в соответствии с запрошенным отладочным режимом из Табл. 3; ошибочные идентификаторы или номера игнорируются. Если не был запрошен ни один из отладочных режимов, все присутствующие строки с информацией для отладки игнорируются. Дополнительные рекомендации по отладке приведены в разделе 5.5.7.

Следующие рисунки иллюстрируют предусмотренные в программе варианты учёта радиуса окрестности и проверки синтении.

Рис. 1. Варианты учёта радиуса окрестности.

Рис. 2. Варианты добавочного учёта синтении. О – одинаковый порядок генов, D одинаковые направления генов, М - допускается зеркальный вариант расположения.

5.2 Секция [data] - входные данные

В этой секции файла конфигурации указываются каталоги, где хранятся входные данные каждого типа, и расширения соответствующих файлов. В качестве имени файла в большинстве случаев должно использоваться имя, присвоенное виду (имена видов содержатся в секции [species] файла конфигурации, см. 5.4). Поэтому здесь вместо имени файла чаще всего указывается символ *.

Секция состоит из необходимого числа однотипных строк, очередность которых не имеет значения. Каждая строка описывает один вид данных и состоит из двух полей, разделённых знаком табуляции. Первое поле содержит односимвольный код типа данных; соответствующее содержание второго поля представлено в Табл. 4. Программа использует только те виды данных, которые соответствуют выбранному режиму работы, а прочие игнорируются. Поэтому данная секция конфигурационного файла может иметь, например, в ОС Windows следующий «универсальный» вид:

```
[data]
\mathbbm{I}genes\*.tsv
\circorthologs\*.tsv
\mathsf{P}paralogs\*.tsv
Hhomologs\*.tsv
\mathbb Ascores\*.tsv
       orthogroups\odb10.tsv
Q
\overline{C}protein-clusters.tsv
```


Табл. 4. Типы входных данных.

5.3 Секция [fields] – имена используемых полей

Таблицы генов [\(4.1\)](#page-6-0), ортогрупп [\(4.5\)](#page-9-0) и белковых кластеров [\(4.6\)](#page-10-0) могут содержать большое число полей в каждой записи и имеют достаточно свободный формат: не лимитирован ни набор полей, ни их очерёдность в записи, что упрощает подготовку исходных данных. Однако среди них есть ряд полей, необходимых для работы алгоритма или нужных для перенесения в выходные данные. Поэтому в обязательном порядке требуется, чтобы эти файлы содержали строку заголовка с именами *всех* имеющихся в записи полей, а в этой

секции файла конфигурации необходимо указать, поле с каким именем содержит данные нужного программе типа. При этом одновременно предоставляется возможность указать для используемых полей более удобные альтернативные имена (алиасы).

Данная секция содержит по меньшей мере шесть строк, в каждой из них одно или два поля, разделённых одним или более знаком табуляции. Первое поле должно содержать точное имя интересующего поля данных в заголовке входного файла, второе (необязательное) – назначенный ему алиас. Эти шесть строк должны идти строго в следующем порядке:

- 1) *Код белка/вида*. Если используются таблицы генов [4.1](#page-6-0) (в том числе, совместно с кластерами белков [4.6\)](#page-10-0), то поле с указанным здесь именем содержит идентификатор белка в БД Ensembl или ином источнике. Если же используется таблица ортогрупп [4.5,](#page-9-0) здесь указывается имя поля, содержащего идентификатор вида (см. [5.4\)](#page-16-0).
- 2) *Код гена*. Поле с указанным именем содержит идентификатор гена в БД Ensembl или ином источнике.
- 3) *Код последовательности*. Поле с указанным именем содержит идентификатор геномной последовательности верхнего уровня – хромосомы, скэфолда, контига и т.п. – в БД Ensembl или ином источнике.
- 4) *Позиция начала гена*. Поле с указанным именем содержит начальную позицию гена в соответствующей последовательности, т.е. номер первого нуклеотида. (Начальная позиция всегда меньше конечной, независимо от того, на какой цепи расположен ген.)
- 5) *Позиция конца гена*. Поле с указанным именем содержит конечную позицию гена в соответствующей последовательности, т.е. номер последнего нуклеотида. (Конечная позиция всегда больше начальной, независимо от того, на какой цепи расположен ген.)
- 6) *Указатель цепи*. Содержимое поля с этим именем указывает цепь ДНК, на которой находится ген: 1 для положительной (прямой) цепи, –1 для отрицательной (обратной), либо в символьной форме, знаком «плюс» или «минус».

Перечисленные шесть полей необходимы для работы программы lossgainRSL и должны присутствовать обязательно. Далее могут идти строки, описывающие другие желаемые поля, например, содержащие *имя гена*, его *аннотацию* и т.д. Чтобы поле данных могло появиться в выходном файле, оно должно быть описано в данной секции. Например, если таблицы генов берутся из Ensembl v95, эта секция может выглядеть так:

5.4 Секция [species] – имена видов и групп

Назначение этой секции конфигурационного файла – сообщить программе lossgainRSL имена всех видов, участвующих в задаче, и, если нужно, определить группы, объединяющие ту или иную часть этих видов. В роли имени вида может выступать любой уникальный идентификатор, присвоенный пользователем, но поскольку этот идентификатор используется в качестве имени файла, он должен удовлетворять требованиям операционной системы.

Рекомендуется ограничиваться алфавитно-цифровыми символами и использовать вместо пробелов символ подчёркивания. Использование букв верхнего и нижнего регистров должно быть единообразным внутри файла конфигурации, даже если файловая система не различает регистр.

По соглашению, первым в этой секции указывается имя базового вида, очередность прочих видов не играет роли. Имя каждого вида занимает отдельную строку.

Иногда условия отбора искомых генов можно сформулировать более компактно, используя понятие группы видов, к которым это условие применяется однотипно. Группа не обязательно должна быть таксономической группой видов, а может составляться на основании любых желаемых признаков. Используемый синтаксис не позволяет одному виду входить сразу в несколько групп; это ограничение обходится путём повторения имени вида.

Имена групп отличаются от имён видов тем, что всегда начинаются с символа звёздочки (*). Для определения группы её имя указывается в отдельной строке среди имён видов. В эту группу входят все виды, начиная с указанного в следующей строке и вплоть до имени другой группы или конца данной секции. Примеры содержимого этой секции имеются в файлах конфигурации в составе контрольного примера (http://lab6.iitp.ru/ru/lossgainrsl/example.zip).

5.5 Секция [predicate] – предикат для отбора генов

Эта секция файла конфигурации является самой сложной и в то же время наиболее ответственной, поскольку именно она определяет, какие гены базового вида будут отобраны и какая информация поступит в выходной файл. В ней используется специальный процедурный язык, отчасти напоминающий известные языки программирования, но с рядом особенностей и ограниченным набором функций. Следует иметь в виду, что программа lossgainRSL осуществляет лишь самый поверхностный контроль правильности предиката, а основная ответственность лежит на пользователе. Как показывает опыт, в подавляющем большинстве случаев аварии или неправильная работа программы происходят именно из-за ошибок в предикате.

Язык описания предиката позволяет определять достаточно сложные условия отбора искомых генов и свободно экспериментировать с этими условиями в разных сочетаниях, без необходимости изменений в программе. Он, однако, не претендует на применимость в любых ситуациях, и не позволяет формулировать сколь угодно сложные условия. Язык лишь предлагает набор типовых элементов проверки и способов их соединения для построения более сложной процедуры обработки. Пользователь должен принять решение о пригодности имеющихся средств для решения конкретной задачи и самостоятельно составить нужный предикат. Например, иногда решение можно получить в результате нескольких запусков программы с разными предикатами и последующего построения искомого множества генов путём объединения или пересечения множеств, полученных в разных запусках.

Программа lossgainRSL использует предикат по следующей схеме. Для каждого очередного гена базового вида предикат интерпретируется с начала, без учёта истории обработки других генов. Первый его оператор, SET, устанавливает значения параметров, которые в данном проходе будут использоваться применительно к базовому виду и, если не указано иное, прочим видам; для последних параметры можно позже изменять новым оператором SET (или его разновидностью), но такие изменения действуют только до нового изменения или до окончания обработки данного гена.

После начального SET в предикате следует набор проверок заданных пользователем условий, каждое из которых касается генов базового вида и другого вида (двухвидовое условие) или базового вида, вида-заменителя и другого вида (трёхвидовое условие). Условия проверяются в той последовательности, как они записаны, однако очерёдностью и логикой обработки можно управлять в зависимости от результата текущей проверки (условный переход) или с помощью оператора безусловного перехода, для чего используются метки. Предусмотрены две неявно определённые метки YES и NO, переход на которые означает, что текущий ген базового вида удовлетворяет предикату (отбирается) или не удовлетворяет (пропускается), соответственно. С помощью оператора ADD для найденных генов в том или ином виде указываются поля данных гена, содержимое которых надлежит перенести в выходной файл.

Нормальной является ситуация, когда в результате последовательной интерпретации предиката либо достигнут его конец (за которым неявно следует метка YES, поэтому текущий ген базового вида будет отобран и появится в выходном файле), либо происходит переход на метку NO (тогда содержимое выходного файла не меняется). В обоих случаях программа переходит к проверке предиката для следующего гена базового вида. Работа lossgainRSL заканчивается после проверки всех генов базового вида.

В последующих параграфах даётся систематическое описание всех возможностей языка описания предикатов. Примеры законченных предикатов можно найти в конфигурационных файлах в составе контрольного примера $(\frac{http://lab6.i}tp.ru/ru/loss \frac{gain}{s/ex}$ ample.zip).

5.5.1 Оператор SET и его разновидности

Оператор служит для установки или изменения значений параметров отбора искомых генов. Он должен целиком записываться в одной строке. Содержит от 1 до 8 полей, разделённых знаком табуляции. Как минимум, должно присутствовать первое поле. Синтаксис оператора:

```
SET[, w, r, b, h1, h2, e] [WIT[NESS]=w] [RANGE=r] [{ORTH|BBH}=b] [HOLD1=h1]]
[HOLD2=h2] [EVAL=e] [HEAD[ING] = commas separated-text]
```
Всего имеется 6 числовых параметров, их значения можно указывать либо в первом поле оператора SET (через запятую сразу вслед за кодом оператора в фиксированном порядке, хотя не изменяемые значения можно не указывать, сохраняя запятые, если дальше ещё есть значения), либо в последующих полях в произвольном порядке и количестве (каждый параметр опознается по своему ключевому слову). Смысл параметров, допустимые значения и значение по умолчанию, а также ключевое слово, используемое для установки значения каждого параметра, представлены в Табл. 5. В эту таблицу включён также необязательный текстовый параметр программы, который нельзя указать в первом поле вместе с кодом оператора, а только в отдельном поле (или операторе) с ключевым словом НЕАD или HEADING.

Значения числовых параметров можно указывать либо числом, либо символической ссылкой в форме *%пате*, где *пате* – произвольный идентификатор, не совпадающий с другими служебными словами. В последнем случае значение параметра задаётся непосредственно в командной строке запуска программы, как описано в главе 3.

Табл. 5. Параметры программы, устанавливаемые оператором SET.

Оператор SET должен быть первым оператором в предикате, и может также появляться далее, чтобы изменить значение тех или иных параметров. При этом: а) новые значения параметров действуют до следующего изменения или до конца предиката; б) для базового вида всегда используется радиус окрестности из начального оператора SET (или принятый по умолчанию); его нельзя изменить динамически, а новое значение г относится только к другим видам.

Если требуется (пере)установить не все сразу, а один-два параметра lossgainRSL, удобнее перечислять нужные значения не вслед за кодом оператора, а в последующих полях, используя соответствующее ключевое слово из Табл. 5, например: RANGE=1M WITNESS=1 SET

Также допускаются разновидности оператора SET, в которых ключевое слово является кодом оператора, а значение указывается в следующем поле, например:

```
RANGE
               500K
WITNESS
               \mathbf{1}
```
Помимо оператора SET и его разновидностей, параметры можно для краткости устанавливать прямо в операторах, осуществляющих проверку, например, IN, OVER, IN2, точно так же, как это делается в первом поле оператора SET. Такие изменения носят локальный характер и действуют только для того оператора, в котором они сделаны.

5.5.2 Оператор IN

Этот оператор проверяет наличие текущего гена X базового вида в указанном другом виде или группе видов. Такая проверка признается успешной, если одновременно выполнены два условия: (1) в другом виде есть ген X', являющийся гомологом X в том смысле, как это задано действующими значениями параметров, например, ортолог X; и (2) в окрестности гена X

радиуса *r* имеется заданное число несовпадающих генов-свидетелей, например, *Y* и *Z*, у которых есть гомологи *Y'* и *Z'*, соответственно, в окрестности гена *X'* радиуса *r'* в другом виде [\(Рис. 3\)](#page-21-0). Радиус интерпретируется с учётом выбранного режима определения окрестности (см. [Табл. 3](#page-11-2) в разделе [5.1\)](#page-11-1), для [Рис. 3](#page-21-0) это вариант 'E'. В этом условии участвует два вида, поэтому оно называется *двухвидовым*.

Синтаксис оператора IN предусматривает два или три поля:

Здесь в первом поле оператора IN можно изменять текущие значения параметров точно так же, как в первом поле оператора SET, чтобы использовать новые значения при этой проверке (но не далее!). Особенностью является значение *r*: если оно указано, то задаёт радиус окрестности только в другом виде, т.е. используется в качестве *r'* на [Рис. 3,](#page-21-0) а в качестве радиуса *r* окрестности в базовом виде всегда используется то значение, которое было задано оператором SET в начале предиката или принято по умолчанию.

Во втором поле оператора в качестве *species* должно быть указано имя другого вида или группы видов – одно из имён, перечисленных в секции [species] файла конфигурации [\(5.4\)](#page-16-0). Напомним, что имя группы начинается со звёздочки. Если указана группа, то поочерёдно проверяются все входящие в неё виды, и условие считается выполненным, если оно выполнено хотя бы для одного вида. Как только найден такой вид и ген, оставшиеся виды данной группы не проверяются. В противном случае условие считается невыполненным.

Необязательное третье поле оператора IN содержит имя метки, куда осуществляется переход при выполнении или невыполнении условия. Если указана метка без знака минус перед ней (*yes-label*), то в случае выполнении условия делается переход по метке, иначе к следующему оператору. Если перед меткой стоит знак минус (-*no-label*), то в случае выполнения условия программа переходит к следующему оператору, а в противном случае – переходит по метке. Если метка опущена, в обоих случаях выполняется переход к следующему оператору. Дальнейшие сведения о метках приведены в следующем разделе.

Фактически оператор IN определяет вложенный цикл проверки заданного условия по всем видам из группы, по гомологичным генам в другом виде, по потенциальным свидетелям и их гомологам и т.д. Эта проверка продолжается до тех пор, пока не найдётся гомолог и свидетели, для которых условие выполняется, либо не будут проверены все варианты. В первом случае проверка успешная и выполняется переход по метке *ves-label* (если она есть), во втором – неуспешная и выполняется переход по метке *no-label* (если есть). Заметим, что цикл не содержит никаких внутренних операторов, а состоит из единственного оператора IN. Любые дальнейшие действия выполняются только после окончания или прерывания цикла.

5.5.3 Метки и оператор GOTO

Метки внутри предиката используются для изменения естественной последовательности исполнения операторов. В качестве метки может использоваться любой уникальный идентификатор (буквенно-цифровая последовательность, которая начинается с буквы и не совпадает с другой меткой, именем вида, поля таблицы или кодом оператора), причём символы верхнего и нижнего регистров различаются. Две метки, YES и NO, неявно определены во всех случаях (см. выше в начале раздела 5.5). Любая явно определяемая метка указывается в отдельной строке предиката вслед за символом двоеточия, например: :RareCase

Переход по метке RareCase означает, что следующим будет выполняться оператор в строке, идущей вслед за меткой.

Типичным примером использования меток является условный оператор IN (5.5.2), в котором обычно фигурирует метка. Ниже описываются и другие виды условных операторов, в которых также могут использоваться метки.

Для описания предикатов со сложной разветвлённой структурой предусмотрен также оператор безусловного перехода GOTO. Он кодируется в отдельной строке вида: **GOTO** label где во втором поле указывается имя одной из существующих меток, например: **GOTO** RareCase **GOTO YES**

5.5.4 Оператор ADD

Оператор ADD служит для формирования выходного файла желаемого вида. Он позволяет скопировать в текущую строку выходного файла необходимые поля из определённой строки таблицы генов для указанного в операторе вида. Синтаксис оператора ADD:

```
[[-] field] [, field2...]ADD
             species
```
Во втором поле должно стоять имя вида (или группы) из числа указанных в секции [species] файла конфигурации (5.4). В необязательном третьем поле перечисляется через запятую список полей из числа указанных в секции [fields] файла конфигурации (5.3); эти поля таблицы генов указанного вида будут скопированы в выходной файл именно в таком порядке. Если это поле опущено, в выходной файл копируются все имеющиеся поля в том порядке, как они записаны в таблице генов. Если перед списком полей стоит знак минус, в выходной файл копируются все поля, за исключением перечисленных.

Уточним, для каких вида и гена берутся данные из таблицы генов:

- \equiv . Если в операторе ADD указан базовый вид, то в выходной файл переносятся данные текущего (проверяемого) гена Х базового вида.
- Если указан другой вид, для которого ранее в предикате исполнялся оператор IN, то в выходной файл переносятся данные того гена, для которого условие было выполнено, т.е. гомолога X' проверяемого гена базового вида со свидетелями в этом другом виде.
- Если для другого вида условие не было выполнено (нет гомолога или свидетелей), поля формируются с пустым содержимым; то же самое происходит, если в данном проходе по предикату оператор IN для указанного вида не исполнялся.
- Если в качестве species было указано имя группы видов (начинающееся со звёздочки), то переносятся данные из того вида, для которого условие выполнилось первым (что зависит от порядка записей в таблицах ортологов и паралогов), а если условие не выполнилось ни для одного вида из группы, в выходной файл вставляются пустые поля.

Вышеизложенное относится к режиму работы lossgainRSL, в котором для отобранных генов базового вида выдаётся только одна строка выходного файла, содержащая данные для гена X базового вида и, возможно, его гомологов X' в других видах (Рис. 3). Альтернативный режим обеспечивает выдачу еще нескольких строк по числу свидетелей, т.е. для генов Y , Z базового вида и, возможно, их гомологов в других видах. Если в секции [mode] указан такой режим работы программы (5.1), то оператор ADD управляет выдачей данных сразу во все эти строки однотипным образом. Порядок строк в синтеничном блоке фиксированный: *X, Y, Z* (или *X, Y*), т.е. искомый ген базового вида всегда стоит в первой строке блока.

Окончательный перенос подготовленных данных для проверяемого гена в выходной файл происходит только в том случае, если для этого гена выполнен весь предикат, т.е. при проверке достигнут конец предиката или метка YES. В противном случае результаты всех операторов ADD для текущего гена теряются и в выходной файл не заносятся никакие данные.

5.5.5 Проверка двухвидовых условий и операторы BOR/EOR, BAND/EAND

Семантику проверки двухвидовых условий в различных вариантах удобно описывать на примере. Пусть базовый вид – лягушка и кроме того рассматриваются две группы видов, рыбы и млекопитающие, в каждой из которых два представителя. Предположим, что радиус окрестности для всех видов - 2 мегабазы и требуется найти гены лягушки, которые имеют ортолога с двумя свидетелями-ортологами у рыб, но отсутствуют у млекопитающих даже как паралоги и хотя бы с одним ортологичным свидетелем. Секция [species] файла конфигурации (5.4) может в этом примере иметь такой вид:

[species] Xenopus $*$ Fish Danio Fuau *Mammal Mouse Human

(A) Простейший вариант предиката может быть следующим: [predicate] SET, 2, 2000K, 7 IN *Fish $-NO$ ADD Xenopus -Protein ADD *Fish Gene, Name IN, 1 , 2 *Mammal NO

Здесь первый оператор SET устанавливает стандартные значения параметров проверки: два свидетеля, окрестность с радиусом 2 мегабазы, в качестве гомологов рассматриваются ортологи. После этого оператор IN проверяет, имеется ли X у какой-либо рыбы (точнее: есть ли у какой-то рыбы его ортолог X' с двумя свидетелями-ортологами Y', Z' в окрестности заданного размера). Если такого X'нет ни у одной рыбы, выполняется переход на метку NO, ген Х лягушки отбрасывается, в выходной файл ничего не записывается, а программа переходит к проверке следующего X. В противном случае следующий за ним оператор ADD копирует в выходной файл все поля таблицы генов для проверяемого гена Х лягушки, за исключением идентификатора белка. Ещё один оператор ADD добавляет в текущую строку выходного файла два поля с идентификатором и именем найденного гена Х'. (Предполагаем, что программа работает в режиме Х из Табл. 3 – выдача только искомых генов, а имена полей совпадают с перечисленными в разделе 5.3). Затем второй оператор IN проверяет, имеется ли X у кого-то из млекопитающих, причем стандартные значения параметров модифицируются: требуется только один свидетель и в качестве гена X' у млекопитающих проверяются не только ортологи X, но и их паралоги (при наличии). Если у кого-то из млекопитающих ген найден, то выполняется переход на метку NO (ген X отбрасывается, в выходной файл ничего не записывается). В противном случае достигнут конец предиката, т.е. он выполнен и происходит запись подготовленной строки в выходной файл. В этой строке будут все подготовленные данные для искомого гена лягушки и его ортолога у какой-то рыбы.

Недостаток описанного предиката в том, что из выходного файла остаётся неясным, у каких ещё рыб есть отобранные гены лягушки. В каждой строке появляется только тот гомолог, который был найден первым у какой-то рыбы, а о других рыбах сведений нет. Причем очередность проверки видов не фиксированная, а зависит от упорядоченности записей в таблицах ортологов и паралогов. Этот дефект особенно мешает, когда рассматриваемые группы состоят не из двух видов, как в данном примере, а из значительно большего числа.

 (D) $\prod_{\mathcal{B}}$ $\prod_{\mathcal{B}}$

:Final IN, 1 , 2 *Mammal **NO**

В этом случае каждое поле выходного файла содержит (или не содержит) данные только одного вида, так что соответствующим столбцам можно присвоить индивидуальные заголовки оператором HEADING (разновидность оператора SET). Напоминаем, что каждый оператор должен быть полностью записан в одной строке; хотя именно этот оператор может достигать значительной длины.

Здесь с помощью первого оператора IN выясняется, есть ли проверяемый ген X у рыбы Danio. Дальнейшая проверка разветвляется: в случае отсутствия гена у Danio программа перейдёт к метке NotDanio. Иначе выполняется следующий за IN оператор ADD, в котором для генов базового вида перечисляются нужные поля в том порядке, как указывают заголовки; эти поля копируются в формируемую строку для гена X. Еще один оператор ADD добавляет в строку выходного файла два поля, относящиеся к ортологичному гену X' у Danio. После этого второй оператор IN проверяет, есть ли ортологичный ген у рыбы фугу. Если ген есть, к строке добавляется еще два поля, к нему относящиеся; в противном случае оператор ADD добавит для фугу два поля пустого содержания. Далее с помощью безусловного перехода проверка продолжится с метки Final.

В другой ветви, начиная с метки NotDanio, оператор IN проверяет наличие гена у рыбы Fugu; при отсутствии такого выполняется переход на метку NO и в выходной файл не записывается ничего. Если ортолог у фугу найден, первый оператор ADD копирует в выходную строку нужные поля текущего гена лягушки. Второй оператор ADD добавляет в эту строку два пустых поля, поскольку ранее условие оператора IN для Danio не было выполнено. Последний оператор ADD добавляет два поля с данными найденного ортолога у фугу. Таким образом, и в этом случае формируется выходная строка с задуманной структурой полей.

Hачиная с метки Final, две ветви снова соединяются и делается та же заключительная проверка, что и в предикате (А). В зависимости от её результатов, сформированная строка либо записывается в выходной файл, либо теряется, если ген Х отвергнут.

Надо признать, что такой способ не особенно удобен и приводит к излишнему усложнению предиката, особенно когда группа включает не два вида, а больше. И к тому же, трудно управлять тем, у скольких видов из группы ген должен присутствовать. Поэтому в языке описания предиката предусмотрена пара операторов BOR/EOR, которые играют роль скобок, охватывающих несколько независимых проверок, для успеха которых устанавливается общий количественный порог.

```
(C) Эквивалентный варианту (B) предикат с использованием этих операторов имеет вид:
[predicate]
SET, 2, 2000K, 7
HEADING
            Xenopus, Contig, Start, End, Name, Description, Danio, Name, Fugu, Name
BOR
TN
            Danio
ADD
                         Gene, Contig, Start, End, Name, Description
            Xenopus
ADD
            Danio
                         Gene, Name
            Fugu
IN
```
ADD Fuqu Gene, Name $-NO$ $EOR, 1$ IN, $1, 2$ *Mammal $N()$

Оператор BOR не имеет аргументов и ставится в строке, предшествующей первой из проверяемых альтернатив. Сами проверки фактически выполняются безусловно: поскольку в операторах IN не указаны метки, то независимо от результата проверки после них выполняется следующий оператор ADD, однако если проверка была неуспешной, то содержимое добавляемых полей будет пустым. Собственно проверка комплексного условия выполняется оператором EOR, который ставится после последней проверяемой альтернативы и имеет следующий синтаксис:

$[ves-label] - no-label]$ EOR \lceil , $n \rceil$

Здесь n – минимально необходимое число выполненных альтернативных проверок, при котором всё условие считается выполненным. Если *п* не указано, оно считается равным нулю, т.е. условие выполнено всегда; то же и в случае, когда опущена метка. В остальном метка интерпретируется как в операторе IN (5.5.2). Возможность оперативно менять единственное значение n, в том числе с помощью символического параметра прямо в командной строке, значительно повышает удобство экспериментов с различными условиями отбора генов базового вида. Пара операторов BOR/EOR может неоднократно встречаться внутри предиката, однако вложенные пары запрещены.

Другая пара операторных скобок, BAND/EAND, является в чём-то двойственной. В отличие от оператора EOR, который подсчитывает число выполненных альтернативных условий, сравнивая его с порогом, оператор EAND подсчитывает число невыполненных альтернатив:

EAND $[m]$ $[yes-label]$ -no-label]

где *m* – минимально необходимое число невыполненных альтернативных проверок, при котором всё условие считается выполненным. В остальном эта пара операторов аналогична описанной выше, хотя следует иметь в виду важное отличие: если проверка не выполнена, в выходную строку заносятся пустые поля. Поэтому для операторов BOR/EOR в предельном случае, когда выполняются все альтернативные условия, в выходной строке будут заполнены все поля, тогда как для пары BAND/EAND в предельном случае, когда все альтернативы не выполняются, выходная строка будет пустой. Тем не менее, эта пара операторов может оказаться более удобной для задания некоторых предикатов. В рассматриваемом примере это не очевидно; тем не менее, приведем без комментариев запись того же самого предиката с их помощью:

5.5.6 Операторы OVER/IN2/NEXT и проверка трёхвидовых условий

Помимо описанного выше двухвидового условия, в программе lossgainRSL предусмотрена возможность проверки более сложного условия, в котором участвуют сразу три вида базовый и еще два, из которых в одном должен быть гомолог текущего гена (такой гомолог будем называть заменителем текущего гена), а в другом виде - есть или нет гомолога для гена-заменителя (Рис. 4). Такое условие назовём трёхвидовым; оно не выполнено, когда либо вообще нет заменителей, либо ни для какого заменителя нет гомолога в другом виде.

Если используемое отношение гомологичности образует транзитивное замыкание, нужда в трёхвидовом условии может возникнуть только в том случае, когда в окрестности гена Х отсутствуют гомологи генов U'' и V", а в окрестности гена X" нет гомологов генов Y и Z, иначе выполнялось бы обычное двухвидовое условие. Наличие вида-заменителя, у которого в окрестности гена X'есть обе пары гомологов, может рассматриваться как свидетельство того, что ген X имеет в другом виде гомолог X" с сохранением синтении.

Как и в двухвидовом условии, здесь гомологичность либо означает ортологичность, либо учитывает также паралоги, либо опирается на сходство белков и др.; ищется заданное число свидетелей и применяется свой радиус окрестности в каждой последовательности. Этими настройками управляют параметры, задаваемые предшествующими операторами SET (5.5.1) или непосредственно в условных операторах OVER и IN2, которые описаны ниже.

При проверке условия значение параметра HOLD1 используется обычным образом, оно управляет постоянством генов *X*, *Y*, *Z* в течение проверки *всего* предиката для текущего *X*; обычно задаётся значение 1, которое означает, что во всех проверках внутри предиката используется один и тот же ген *X*, а свидетели *Y* и *Z* могут быть свои в каждом элементарном условии.

Значение параметра HOLD2 позволяет указать, какие гены остаются неизменными при переходе от верхнего условия к нижнему (по [Рис. 4\)](#page-27-1). Значение указывается в виде суммы трёх значений битов, имеющих следующий смысл:

1 – должен использоваться один и тот же гомолог гена *X* (обозначенный на рисунке *X'*),

2 – должен использоваться один и тот же гомолог первого свидетеля, т.е. *U' = Y'*, 4 – должен использоваться один и тот же гомолог второго свидетеля, т.е. *V' = Z'*. Как объяснялось выше, если гомологичное отношение транзитивно (что справедливо для

ортогрупп OrthoDB и кластеров белков, но иногда нарушается для ортологов в БД Ensembl), то следует указывать значение 1.

Напомним, что двухвидовое условие определяется единственным оператором IN, который задаёт вложенный цикл перебора генов указанного в нём не-базового вида или группы видов [\(5.5.2\)](#page-20-0). Чтобы определить трёхвидовое условие, используются три оператора – OVER, IN2 и NEXT. Операторы OVER и IN2 задают два вложенных друг в друга цикла перебора генов в двух не-базовых видах, а оператор NEXT указывает конец тела внешнего цикла. Очерёдность и синтаксис указанных операторов следующие:

Внешний цикл определяется оператором OVER, в котором вместо *species1* указывается имя вида или группы видов, в которых ищется ген-заменитель. Ген-заменитель *X'* должен быть гомологом текущего гена *X* базового вида и, кроме того, в окрестности *X'* должны присутствовать гены-свидетели *Y'* и *Z'*, которые являются гомологами генов *Y* и *Z* в окрестности гена *X* базового вида. Радиус окрестности *r'* в геноме вида-заменителя, число свидетелей и используемый вариант гомологии, если они отличны от ранее установленных для всего предиката, можно установить параметрами предшествующего оператора SET или указать непосредственно после кода в первом поле оператора OVER.

Выполнение оператора зависит от указанной в нём метки. Если указана метка без знака минус, переход на неё осуществляется, когда найден первый подходящий ген-заменитель (без учета условий внутреннего цикла), в противном случае программа переходит к следующему оператору. Такой вариант условия, по-видимому, не имеет особого смысла и описан здесь для полноты. Более типична ситуация, когда метка в операторе OVER указана со знаком минус перед ней. Тогда переход по метке осуществляется по окончании перебора всех генов указанного вида или группы видов, если ни для одного из них не выполнены условия обоих циклов или не произошёл переход из тела цикла. Если метка опущена, это интерпретируется как если бы стояла метка -*no-label* , указывающая на следующий оператор после NEXT.

Внутренний цикл определяется оператором IN2, который аналогичен оператору IN (5.5.2) с той единственной разницей, что в роли гена Х, для которого проверяется наличие или отсутствие в другом виде или группе видов species2, выступает очередной ген X', найденный во внешнем цикле. Значения параметров и метки в этом операторе обрабатываются точно так же, как в операторе IN. Например, если указана метка *yes-label* (без знака минус перед ней), то при обнаружении в геноме *species2* гомолога гена X' вместе с необходимыми свидетелями выполняется переход по этой метке, которая может быть как внутри тела цикла, так и вне, скажем, NO. В противном случае, если такого гомолога, X", нет в указанном геноме (или целой группе геномов, если *species2* - имя группы), выполняется следующий оператор тела внешнего цикла, скрытый за многоточием выше. В частности, это может быть другой оператор IN2, который осуществляет другую проверку текущего X', переход по метке и т.д.

Признаком конца тела внешнего цикла является оператор NEXT, не имеющий параметров, который служит программе сигналом возвратиться к оператору OVER и опробовать следующий ген-заменитель Х'.

Следует проявлять осторожность при включении в тело цикла операторов с побочным эффектом, таких как ADD, поскольку это может привести к зацикливанию программы с появлением в выходном файле строк неконтролируемой длины или, по меньшей мере, непостоянного формата. Рекомендуется соблюдать простое правило: после записи данных в выходной файл необходимо прервать выполнение внешнего цикла путём условного или безусловного перехода из него на некоторую внешнюю метку.

5.5.7 Отладка предиката

По существу, записанный в конфигурационном файле предикат есть программа, которую lossgainRSL для каждого очередного гена X исполняет в режиме интерпретации. Как любая программа, предикат может содержать ошибки, приводящие к получению ошибочных результатов или отказам при работе lossgainRSL, таким как аварийное завершение или зацикливание. Некоторые ошибки (неправильный синтаксис оператора, ошибки в ключевых словах, именах полей и видов, неопределённые метки и т.п.) обнаруживаются в процессе разбора файла конфигурации, но иногда они проявляются только в процессе счёта или при анализе полученных результатов. Не пытаясь предложить универсальные методические рекомендации по отладке, ограничимся перечислением средств, предусмотренных в программе lossgainRSL для этой цели, и некоторых типовых приёмов.

В процессе работы программы на консоль выдаётся протокол, который можно перенаправить в файл стандартными средствами операционной системы. Описание протокола приведено в разделе 6.1. В случае любых ошибок, в этом протоколе прежде всего следует проверить:

- $-$ командную строку запуска lossgainRSL (использованные опции и имена файлов);
- номер версии программы; \equiv
- режим работы программы (5.1);
- перечень видов и групп видов, а также список видов в каждой группе;
- результаты разбора предиката. Необходимо убедиться в правильности переходов в $\frac{1}{2}$ результате замены меток номерами операторов, а также что указан правильный геном и значения параметров в каждом операторе, включая установленные по умолчанию;
- сведения о прочитанных файлах исходных данных и их количественные характеристики.

Если программа завершается аварийно или зацикливается, по протоколу можно установить, на каком контиге и гене базового вида это произошло (последние выданные на консоль **xLLL). Поскольку по умолчанию номера генов выдаются с шагом** значения в строке сККК 10, имеет смысл повторить запуск с опцией $-G1$ (см. главу 3).

Найденные значения ККК и LLL можно использовать для получения более полной отладочной выдачи, для чего в секции [mode] файла конфигурации надо дополнительно указать режим отладки программы N в первой строке, а также добавить в эту секцию ещё строку вида ККК LLL. Подробнее см. Табл. 3 в 5.1. Протокол отладки выдаётся в стандартный поток stderr, который удобно перенаправить в файл.

6 Описание выходной информации

6.1 Протокол работы lossgainRSL

Протокол программы по большей части понятен сам по себе; ниже приводится пример реального протокола работы с объяснениями некоторых данных и обозначений. Для удобства описания протокол разрезан на несколько идущих друг за другом частей, выделенных голубым фоном.

```
Synteny analysis utility (version 6.20)
Taxa: 4 Species: 20 Mode: H E I J
```
В первой строке указан номер версии программы. После этого показано число групп видов, которые определены в файле конфигурации (4) и общее число видов (20), а также активные режимы работы программы (5.1). При запуске программы в среде МРІ указывается ещё число параллельно работающих ветвей.

```
*[1]: mus musculus
*Long-lived[3]: heterocephalus_glaber_female homo_sapiens cebus_capucinus
*Putative long-lived[6]: nannospalax galili fukomys damarensis nomascus leucogenys pan troglodytes
pongo abelii gorilla gorilla
*Short-lived[10]: peromyscus_maniculatus_bairdii cavia_porcellus ictidomys_tridecemlineatus
oryctolagus cuniculus mesocricetus auratus rattus norvegicus microcebus murinus
rhinopithecus roxellana otolemur garnettii rhinopithecus bieti
```
Затем перечисляются группы видов вместе с входящими в них видами. Каждая группа начинается с новой строки. В данном случае имеются: безымянная группа *, в которую входит только геном мыши; группа *Long-lived из трёх геномов, имена которых перечислены после двоеточия (самка голого землекопа, человек, белоплечий капуцин); группа *Putative_long-lived из 6 геномов (израильский слепыш, дамарский пескорой, гиббон, шимпанзе, орангутан, горилла); и группа *Short-lived из 10 геномов (олений хомячок, морская свинка, 13-полосовый суслик, кролик, золотистый хомячок, серая крыса, мышиный лемур, рокселланов ринопитек, галаго Гарнетта, чёрный ринопитек). Этот фрагмент не выдаётся, если в командной строке была указана опция - а или - аа.

Далее приводятся результаты синтаксического разбора предиката (также не выдаются, если в командной строке была указана опция - а или - ад).

Здесь в первой строке указано общее число операторов, распознанных в описании предиката (59), и поэтому следующие 59 строк протокола начинаются с номера оператора. Операторы нумеруются с нуля, и самым первым должен быть оператор SET (5.5.1). Для каждого оператора в протокол выдаётся ряд полей, хотя в зависимости от оператора некоторые поля могут быть пустыми или иметь особенности. Применительно к оператору SET во второй строке этого фрагмента, указаны: номер оператора (0), код оператора, номер следующего выполняемого оператора (1), и текущие значения параметров - число генов-свидетелей (2), радиус окрестности (5 Mbp), ORTH (7), HOLD1 (1), HOLD2 (1) и E-value (10⁻⁵). В последней строке оператор IN проверяет, есть ли текущий ген мыши у какого-либо долгоживущего вида (группа *Long-lived). Если да, то выполняется переход к метке NO (т.е. ген пропускается), иначе программа переходит к следующему оператору (2). Дальше в строке перечисляются используемые конкретно при этой проверке значения параметров, которые совпадают с первоначально заданными с одним исключением: значение ОКТН=6 означает, что в долгоживущих видах проверяются не только ортологи текущего гена мыши, но и их паралоги (подробнее см. 5.5.1).

Оператор BOR в начале этого фрагмента указывает, что с оператора 2 начинается серия проверок, из которых не менее определённого числа должны дать положительный результат. Первая из таких проверок вместе с сопутствующими действиями содержится в данном фрагменте. Оператор IN проверяет, есть ли текущий ген мыши вместе с двумя свидетелями в геноме оленьего хомячка. Независимо от результата проверки, программа переходит к оператору 3, потому что в данном случае в операторе IN не была указана метка. Если бы она присутствовала, здесь стояло бы два номера операторов, из которых один всегда следующий оператор, а другой – какой-то еще. По соглашению, первый номер используется в случае успеха проверки, второй – при неуспехе. Если в операторе нет разветвления, как в данном случае, то указывается только один номер следующего оператора. В конце строки оператора IN перечислены текущие значения параметров, которые здесь совпадают с первоначально заданными.

Первый оператор ADD в этом фрагменте копирует в формируемую строку выходного файла перечисленные поля таблицы генов для текущего проверяемого гена мыши. Указываются исходные имена полей из раздела [fields] файла конфигурации (5.3), хотя в выходном файле могут фигурировать альтернативные имена (алиасы). Второй оператор ADD добавляет в строку выходного файла еще одно поле с идентификатором первого найденного оператором IN подходящего гена оленьего хомячка. Если проверка была неуспешной и подходящего гена не нашлось, то содержимое поля пустое. Разумеется, выбор нужных полей для копирования в выходной файл принадлежит пользователю.

В этом фрагменте содержатся ещё пять аналогичных проверок с другими геномами. Поясним, зачем после каждой проверки в выходную строку добавляется не только найденный ген из проверяемого вида, но сначала еще и ген мыши. Действительно, если бы в этом предикате выдавались только искомые гены мыши, то было бы излишним повторять каждый раз данные одного и того же текущего гена *X*. Однако из самого первого фрагмента протокола видно, что для программы запрошен режим выдачи целого синтеничного блока генов, состоящего из *X* и расположенных вблизи генов-свидетелей *Y*, *Z* (то же и для ортологов в других видах). В общем случае для разных видов могут использоваться разные свидетели *Y* и *Z*, поэтому здесь для каждого проверяемого вида снова выдаются гены мыши. Вся выдача для каждого гена *X* будет состоять из *четырёх* строк выходного файла: для гена *X* и его ортологов у других видов, для гена *Y* в разных вариантах и его ортологов у других видов, то же для гена *Z*, плюс пустая строка-разделитель.

Последняя строка во фрагменте выше содержит оператор EOR [\(5.5.5\)](#page-23-0), завершающий эту серию проверок. В нём указаны две метки 22 и NO (вторая виртуальная и не транслируется в номер реального оператора), а также значение параметра *n*=3, которое означает, что если 3 и более (т.е. не менее половины) проверок были успешными, то этот терм предиката считается выполненным и программа переходит к проверке следующего терма. В противном случае предикат не выполнен и программа переходит к проверке очередного гена.

Этот терм предиката аналогичен предыдущему, но в нём проверка идёт против другого множества, состоящего из четырёх видов, и используется другое значение параметра *n*=2. В этом и состоит причина, почему эти две серии проверок не были соединены в одну: фактически, проверяемое условие может быть словесно сформулировано как «ген мыши присутствует не менее чем у половины короткоживущих грызунов и не менее чем у половины короткоживущих приматов».

Дальнейшая проверка предиката проводится с изменённым значением параметра ORTH:

Сделано такое же изменение, как и в операторе 1 выше, но теперь оно будет действовать для всех последующих операторов, вплоть до нового изменения или конца предиката.

Эта группа операторов проверяет, что текущий ген мыши отсутствует хотя бы у одного из предположительно долгоживущих грызунов. Здесь удобнее оказалось использовать операторные скобки BAND/EAND, хотя желаемого результата можно было достичь и так, как делалось в предыдущих термах, изменив содержание заключительного условия.

Наконец, заключительный терм предиката аналогичен предыдущему, но относится к четырём предположительно долгоживущим приматам и использует порог 2, что соответствует половине этого множества.

Последний оператор EAND проверяет условие: если из четырёх проведённых проверок две и более не были успешными, то весь предикат выполняется (переход на метку YES) и ранее сформированные строки будут записаны в выходной файл. Иначе, если неуспешных проверок было меньше двух, предикат не выполняется и строки не записываются. В обоих случаях программа переходит к проверке следующего гена *X* базового вида, начиная с оператора 0.

Maximum 32 gene(s) per line: mus musculus peromyscus maniculatus bairdii mus musculus cavia porcellus mus musculus ictidomys tridecemlineatus mus musculus oryctolagus_cuniculus mus_musculus mesocricetus_auratus mus_musculus rattus_norvegicus mus musculus microcebus murinus mus musculus otolemur qarnettii mus musculus rhinopithecus roxellana mus musculus rhinopithecus bieti mus musculus nannospalax galili mus musculus fukomys damarensis mus musculus nomascus leucogenys mus musculus pan⁻troglodytes mus^{musculus} pongo^{_abelii} mus musculus gorilla gorilla

В этом фрагменте консольного протокола перечислены слева направо виды, гены которых будут представлены в строке выходного файла. Если предикат формирует строки с переменным форматом, показан вариант с наибольшим числом генов. Данный фрагмент не выдаётся, если в командной строке была указана опция -q или -qq.

Reading gene info file genes\mus musculus.tsv ... Lines read: 111762 genes: 22950 proteins: 66760 contigs: 104 c1: 10 c2: 12

Здесь приведены сведения о прочитанной таблице генов базового вида. В первой строке указано полное имя файла и путь к нему, во второй – численные характеристики: общее число строк в файле, число разных генов в этих строках, число белков (напомним, что lossgainRSL учитывает только белок-кодирующие гены) и общее число последовательностей верхнего уровня (104), которые здесь для единообразия именуются контигами. Далее, $c1$ число контигов, содержащих только один ген (10) , с2 - число контигов, содержащих ровно два гена (12).

Далее по мере чтения таблиц генов выводится аналогичная информация об остальных видах:

Reading gene info file genes\heterocephalus glaber female.tsv ... Lines read: 40018 genes: 20774 proteins: 28984 contigs: 366 c1: 92 c2: 17 Reading gene info file genes\homo_sapiens.tsv ... Lines read: 166360 genes: 24154 proteins: 110059 contigs: 1138 c1: 871 c2: 63 Reading gene info file genes\cebus capucinus.tsv ... Lines read: 48350 genes: 20317 proteins: 40677 contigs: 1037 c1: 199 c2: 86 Reading gene info file genes\nannospalax galili.tsv ... Lines read: 32863 genes: 18647 proteins: 26872 contigs: 1433 c1: 354 c2: 135 Reading gene info file genes\fukomys damarensis.tsv ... Lines read: 39247 genes: 17730 proteins: 23413 contigs: 930 c1: 202 c2: 79 Reading gene info file genes\nomascus leucogenys.tsv ... Lines read: 47561 genes: 20794 proteins: 40527 contigs: 336 c1: 225 c2: 49 Reading gene info file genes\pan_troglodytes.tsv ... Lines read: 60146 genes: 23534 proteins: 49949 contigs: 625 c1: 364 c2: 97 Reading gene info file genes\pongo abelii.tsv ... Lines read: 29435 genes: 20424 proteins: 21414 contigs: 53 c1: 0 c2: 2 Reading gene info file genes\gorilla gorilla.tsv ... Lines read: 53486 genes: 21794 proteins: 45194 contigs: 327 c1: 271 c2: 25 Reading gene info file genes\peromyscus maniculatus bairdii.tsv ... Lines read: 33295 genes: 19854 proteins: 28866 contigs: 1424 c1: 399 c2: 137 Reading gene info file genes\cavia_porcellus.tsv ... Lines read: 34344 genes: 18095 proteins: 25582 contigs: 319 c1: 61 c2: 20 Reading gene info file genes\ictidomys tridecemlineatus.tsv ... Lines read: 32687 genes: 18474 proteins: 25958 contigs: 698 c1: 212 c2: 41 Reading gene info file genes\oryctolagus cuniculus.tsv ... Lines read: 24966 genes: 19293 proteins: 20588 contigs: 1053 c1: 472 c2: 174 Reading gene info file genes\mesocricetus auratus.tsv ... Lines read: 29938 genes: 18257 proteins: 25910 contigs: 777 c1: 465 c2: 32 Reading gene info file genes\rattus norvegicus.tsv ... Lines read: 40234 genes: 22250 proteins: 29107 contigs: 133 c1: 78 c2: 14 Reading gene info file genes\microcebus murinus.tsv ... Lines read: 45983 genes: 18895 proteins: 38078 contigs: 76 c1: 32 c2: 2 Reading gene info file genes\rhinopithecus roxellana.tsv ... Lines read: 53493 genes: 21289 proteins: 45897 contigs: 2777 c1: 1001 c2: 343 Reading gene info file genes\otolemur garnettii.tsv ... Lines read: 28567 genes: 19506 proteins: 19986 contigs: 525 c1: 151 c2: 38 Reading gene info file genes\rhinopithecus_bieti.tsv ... Lines read: 52671 genes: 20966 proteins: 43730 contigs: 2510 c1: 1007 c2: 248

Эти сведения не выдаются, если в командной строке была указана опция -qq (но сохраняются с опцией -q). То же относится и к следующим строкам, сообщающим о прочитанных таблицах ортологов для базового вида и паралогов для тех видов, где требуется. Здесь таблицы паралогов нужны для всех долгоживущих видов, потому что для них задано значение параметра ORTH=6 (т.е. гомологи проверяемого гена *X* необходимо искать с учётом паралогов). В конце каждой строки приводится число упорядоченных пар ортологичных (или паралогичных) генов:

```
Reading orthologs from orthologs\mus_musculus.tsv ... 424717
Reading paralogs from paralogs\heterocephalus glaber female.tsv ... 203921
Reading paralogs from paralogs\homo_sapiens.tsv ... 146080
Reading paralogs from paralogs\cebus capucinus.tsv ... 111465
Reading paralogs from paralogs\nannospalax_galili.tsv ... 100929
Reading paralogs from paralogs\fukomys_damarensis.tsv ... 187810
Reading paralogs from paralogs\nomascus_leucogenys.tsv ... 115430
Reading paralogs from paralogs\pan_troglodytes.tsv ... 159621
Reading paralogs from paralogs\pongo_abelii.tsv ... 116494
Reading paralogs from paralogs\gorilla_gorilla.tsv ... 130833
```
Описанная выдача характерна для выбранного режима работы H (данные из Ensembl). Если же входными данными является таблица ортогрупп (4.5) , то вместо двух последних фрагментов выдаётся один следующего содержания:

```
Reading orthogroups from ODB\odb10.txt ...
Orthogroups processed: 24616 Orthologous pairs written: 5742639
Contigs, genes and pairs collected per species:
c: 142/170 g: 21089/21117 p:638301/639471 mus_musculus
c: 275/383 g: 18948/19056 p: 32169/32528 heterocephalus_glaber
c: 197/287 g: 19721/19811 p: 28692/28789 homo_sapiens
c: 823/1049 g: 19367/19593 p: 26885/27346 cebus_capucinus_imitator
c:1182/1836 g: 19238/19892 p: 37390/39369 nannospalax_galili
c: 738/1044 g: 18618/18924 p: 33179/34119 fukomys_damarensis
c: 116/404 g: 19217/19505 p: 26696/27013 nomascus_leucogenys
c: 99/266 g: 20327/20494 p: 28824/29162 pan troglodytes
c: 77/188 g: 19844/19955 p: 29288/29453 pongo_abelii
c: 48/451 g: 19975/20378 p: 28746/29124 gorilla_gorilla_gorilla
c:1121/1819 g: 19995/20693 p: 46419/51173 peromyscus_maniculatus_bairdii
c: 271/357 g: 19107/19193 p: 37927/38694 cavia_porcellus
c: 507/788 g: 19130/19411 p: 34133/35454 ictidomys_tridecemlineatus
c: 554/1017 g: 18663/19126 p: 35464/37858 oryctolagus_cuniculus
c: 357/1292 g: 18560/19495 p: 34776/36294 mesocricetus_auratus
c: 50/137 g: 20980/21067 p: 47959/48258 rattus norvegicus
c: 48/108 g: 19411/19471 p: 29493/29614 microcebus_murinus
c:1715/3015 g: 19117/20417 p: 26630/28537 rhinopithecus_roxellana
c: 351/509 g: 18937/19095 p: 28662/29037 otolemur garnettii
c:1486/2889 g: 19023/20426 p: 25952/27649 rhinopithecus_bieti
```
Здесь в первой строке указано полное имя и путь к файлу ортогрупп (здесь единственному), а во второй – его численные характеристики: число обнаруженных ортогрупп и общее число ортологичных пар генов (неупорядоченных). Ниже для каждого вида, упомянутого в секции [species] конфигурационного файла [\(5.4\)](#page-16-0), приведены характеристики генома – число контигов (c:) и генов (g:), а также число ортологичных пар с участием данного вида (p:). Каждая из характеристик представлена в форме дроби, где в знаменателе указано общее число (например, контигов), а в числителе – сколько из них используется (содержит минимально необходимое число генов). Разумеется, данные из разных источников не одинаковы.

Затем выдаются две строки, показанные ниже. В первой напоминается имя базового вида, число последовательностей верхнего уровня (для краткости «контигов»), из которых состоит его геном (104), и общее число генов (22950). Во второй строке приводятся актуальные данные – сколько контигов и генов будет перебираться в данной задаче (поскольку требуется два свидетеля, пропускаются контиги, содержащие только один или два гена).

mus musculus in total: 104 contigs, 22950 genes. With several genes: 94 contigs, 22940 genes.

После этого начинается собственно поиск генов, во время которого программа lossgainRSL вылаёт на консоль сообшения вила

x130 $c5$

где первое число указывает порядковый номер проверяемого контига базового вида, а второе - порядковый номер гена в этом контиге. Порядок определяется таблицей генов, нумерация в обоих случаях идёт с нуля, причём номера генов отображаются с шагом, заданным опцией -G в командной строке. Эта информация может помочь при отладке (5.5.7).

При успешном завершении работы выдаётся число найденных генов и время счёта:

```
mus musculus candidate genes selected: 57
```
7m: Completed OK.

6.2 Выходной файл

В результате работы lossgainRSL формируется выходной файл, имя и путь к которому были указаны в командной строке (глава 3) или приняты по умолчанию - result.txt в текущей директории. Если файл с нужным именем уже существует, он будет перезаписан заново.

Файл создаётся в текстовом формате TSV (поля данных, разделённые символом табуляции). Первая строка содержит имена каждого поля. Предполагается, что число и тип полей одинаковы во всех строках выходного файла, хотя пользователь может составить предикат, который приведёт к получению строк разного формата. Приоритет отдаётся именам, которые были явно указаны в предикате с помощью ключевого слова HEAD или HEADING (5.5.1); в этом случае пользователь сам отвечает за правильность заголовков и их соответствие полям в информационных строках. Если заголовки не заданы в предикате, то используются альтернативные имена полей (алиасы) из секции [fields] (5.3) файла конфигурации. Если алиасы отсутствуют, используются исходные имена полей, которые имелись в заголовке таблицы генов. Выходной файл допускает удобный импорт в электронную таблицу Excel, что упрощает последующую обработку и анализ полученных результатов.

Если для программы был задан режим X (5.1), то для каждого найденного гена базового вида выдаётся одна строка, набор полей в которой определяется последовательностью операторов ADD, выполненных в процессе проверки предиката для этого гена. Если режим X не задан, сверх того выдаётся по одной строке для каждого гена-свидетеля и его ортологов в другом виде (если число свидетелей по ходу проверки предиката меняется, берётся максимум) плюс одна пустая строка-разделитель после каждой такой группы генов (если в числе режимов программы указано I). В этом случае, чтобы упростить упорядочение групп, состоящих из нескольких строк, в конце строк можно с помощью режима Ј автоматически добавлять два дополнительных поля (подробнее см. 5.1).

Кроме того, если программа lossgainRSL запущена в режиме MPI, в конце каждой строки выходного файла автоматически добавляется поле rank, в котором указан номер параллельной ветви программы, в которой найден данный ген (для целей отладки).

Несколько примеров выходных файлов имеются в составе контрольного примера $(http://lab6.iitp.ru/ru/lossgainsl/example,zip).$

7 Использование других источников входных данных

7.1 Добавление геномов, отсутствующих в Ensembl

База данных Ensembl уже в нынешней редакции содержит полные геномы свыше 150 видов и регулярно пополняется. Тем не менее, возникают задачи, где в число видов требуется включить несколько (подразумевается один-два) видов, которые Ensembl пока не охватывает. В принципе, это возможно, но связано с достаточно трудоёмкой деятельностью по ручному составлению новых и дополнению имеющихся таблиц, либо с разработкой и отладкой соответствующих программ и скриптов для указанной цели.

Ниже описывается в общем содержание работы по добавлению одного нового генома к уже имеющимся входным данным (см. главу 4). Если надо добавить несколько новых геномов, сказанное ниже надо проделать по очереди для каждого из них. Подчеркнём, что так нельзя <u>добавить новый геном базового вида, который должен браться из Ensembl</u>, по крайней мере, в нынешней реализации. Если базовый вид не входит в Ensembl, то *все* данные об ортологии генов необходимо брать из других источников (7.2).

По-видимому, наиболее важным источником новых геномов является GenBank. В принципе, вся необходимая информация там содержится, но часто в разрозненном и недостаточно формализованном виде, что усложняет процедуру составления таблиц 4.1–4.3. Для ручной работы удобнее всего использовать полный геном в формате GBFF; описание ниже относится именно к такому варианту. Если же пользователь lossgainRSL разрабатывает для этой цели собственную программу (или скрипт), он сам должен выбрать подходящий источник и формат входных файлов.

7.1.1 Составление новой таблицы генов вручную

Прежде всего необходимо составить таблицу генов для нового вида. Она должна иметь в точности такой формат, как описано в 4.1, включая заголовки полей. Необходимо, чтобы все таблицы генов, участвующие в задаче, имели согласованные имена полей. Составление таблицы вручную удобно выполнять в Excel, взяв в качестве шаблона одну из имеющихся таблиц генов. Укажем возможные источники данных для полей, перечисленных в Табл. 1.

Chromosome/scaffold name

В строке LOCUS, с которой начинается каждая секция файла в формате GBFF, есть номер по каталогу данной последовательности (accession), причём без номера версии. Его можно безопасно использовать для содержимого данного поля, хотя для большей информативности результатов рекомендуется изучить следующую строку (DEFINITION). Например, если в ней указано Drosophila melanogaster chromosome 2R, то более уместным содержимым поля следует признать 2R. Имя, присвоенное данной последовательности, должно в дальнейшем использоваться в строках для всех белков/генов, аннотации которых найдены в этой секции файла GBFF.

Далее анализу подлежат строки файла от заголовка FEATURES до заголовка ORIGIN.

Gene stable ID, Gene name

Заносить в таблицу имеет смысл только белок-кодирующие гены, для которых в файле имеются предложения gene, mRNA, CDS. В каждом из этих предложений идентификатор белка указан тегом /gene; его лучше использовать в качестве имени гена, а уникальный идентификатор назначать некоторым формальным способом по аналогии с тем, как это сделано в Ensembl.

При внесении гена в таблицу следует учитывать его размер. Если ген состоит из нескольких экзонов, расположенных с большими промежутками, его может быть целесообразно разбить на группы, каждая из которых будет вноситься как отдельный «ген», что следует отразить и в имени (например, добавляя к постоянному имени гена номер части). Ключевым вопросом для такого разбиения является возможность получения аминокислотной последовательности отдельно для каждой части, т.е. края экзонов должны совпадать с границей кодона.

Если для гена приведено несколько транслированных последовательностей в отдельных предложениях CDS, соответствующих разным вариантам сплайсинга, то для него в таблицу следует внести несколько строк с одинаковым идентификатором гена, но разными идентификаторами белков.

Gene start (bp), Gene end (bp), Strand

Координаты начала и конца гена (или его части) в нуклеотидной последовательности имеются в предложениях gene, mRNA, CDS. Если перед координатами стоит ключевое слово complement, то в поле Strand записывается -1 , иначе 1.

Gene description

Для этого поля целесообразно использовать содержимое полей с тегами /note и /product в предложениях gene, mRNA, CDS. Трудно предложить алгоритм для автоматического формирования данного поля, но при ручном заполнении таблицы генов трудностей не возникает.

Protein stable ID

Идентификатор белка следует назначать каждой аминокислотной последовательности с тегом /translation в предложении CDS, для чего следует выработать некоторую формальную процедуру. В результате, как отмечено выше, в таблице генов для нового вида может появиться несколько строк для одного и того же гена, но с разными идентификаторами белков, что отражает возможность альтернативного сплайсинга.

Важно одновременно с наполнением таблицы генов составлять для нового вида файл белков, удобнее всего – в формате FASTA, где именем является назначенный идентификатор белка, а аминокислотная последовательность берётся из упомянутого выше поля с тегом /translation. Можно и наоборот, использовать готовый FASTA-файл всех белковых последовательностей нового вида, заменяя в нём заголовки на присвоенные идентификаторы белков.

Вышеизложенные рекомендации следует считать предварительными, их предстоит уточнить по мере накопления опыта добавления новых видов.

7.1.2 Импорт таблицы генов из RefSeq

Описанная в этом разделе процедура применима в тех случаях, когда собранный полный геном интересующего вида существует в форме эталонной последовательности (RefSeq) NCBI (ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/), что справедливо для многих видов из GenBank. В составе эталонной сборки обычно есть файл feature_table, из которого несложно получить нужную таблицу генов. Например, для кашалота этот файл расположен по адресу ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/002/837/175/GCF_002837175.1_ASM283717v1/GCF_ 002837175.1_ASM283717v1_feature_table.txt.gz, и для получения таблицы генов можно применить следующий скрипт python (автор – О.А. Зверков):

#!python3

```
import csv
import gzip
infile name = 'GCF 002837175.1 ASM283717v1 feature table.txt.gz'
outfile name = 'physeter catodon.tsv'
ens_fields = ('Protein ID', 'Transcript ID', 'Gene ID', 'Region Type',
                 'Region', 'Start', 'End', 'Strand', 'Label', 'Description')
def main():
    with gzip.open(infile_name, 'rt', newline='') as infile, \
          open(outfile name, 'w', newline='') as outfile:
         assert infile.read(2) == '# '
         reader = \text{csv.DictReader}(\text{infile, delimiter='}\t)writer = \text{csv.DictWriter}(\text{outfile, delimiter='}\t),\text{ fieldnames=ens fields})outfile.write('\t'.join(ens fields) + '\n')
         for feature in reader:
              if feature['feature'] == 'CDS':
                  writer.writerow(gbk2ens(feature))
def gbk2ens (feature) :
    region = (feature['chromosome'] if feature['seq type'] == 'chromosome'
                else feature ['qenomic accession'])
    return {
         'Protein ID': feature['product accession'].split('.')[0],
         'Transcript ID': feature['related_accession'].split('.')[0],
         "Gene ID": <br>
'GID' + feature["GeneID"],<br>
'Region Type': feature['seq_type'],
        Region 1ype: Leature Seq_cype 1,<br>
'Region': region,<br>
'Start': feature ['start'],<br>
'End': feature ['end'],<br>
'Strand': {'+': '1', '-': '-1'} [feature ['strand']],<br>
'Label': feature ['symbol'],
         'Description': feature['name'],
    \mathcal{F}if name == ' main ':
    main()
```
7.1.3 Дополнение таблиц ортологов и паралогов с помощью утилиты addspecies

Если в предикате (5.5) не используются трёхвидовые условия, таблица ортологов (4.2) нужна только для базового вида (иначе - ещё и для всех потенциальных видов-заменителей). При добавлении вида, отсутствующего в Ensembl, в эту таблицу необходимо добавить записи для ортологичных пар генов базового и нового видов. Кроме того, если в предикате участвуют паралоги, может потребоваться сформировать таблицу паралогов (4.3) для нового вида. Для этого разработана вспомогательная программа addspecies, которая доступна для загрузки с главной страницы lossgainRSL (http://lab6.iitp.ru/ru/lossgainrsl/).

Для использования программы необходимо предварительно с помощью BLASTP вычислить оценки сходства (scores) для всех пар белков базового и нового видов, а также всех пар белков базового и нового вида по отдельности. Подразумеваются упорядоченные пары, т.е. в обоих направлениях, поскольку в общем случае оценка зависит от того, какой из двух видов

используется в качестве области поиска. Для этого, в свою очередь, требуется получить аминокислотные последовательности для каждого белка обоих видов. Для геномов из Ensembl файлы белков в формате FASTA доступны на ftp-сервере в каталоге рер для каждого вида, например, ftp://ftp.ensembl.org/pub/release-95/fasta/mus musculus/рер/ для мыши. Для геномов из RefSeq и GenBank аналогичные файлы имеются в каталоге соответствующей сборки, например, GCF 002837175.1 ASM283717v1 protein.faa.gz (полный адрес каталога приведен в 7.1.2). Подробности запуска BLASTP не описываются; в результате должны быть получены четыре матрицы сходства белков в формате согласно разделу 4.4, которым удобно присвоить имена spec1-spec2.*, spec2-spec1.*, spec1-spec1.* и spec2-spec2.* где spec1,2 имена базового и нового видов, указанные в секции [species] файла конфигурации (5.4). Подчеркнём, что идентификаторы белков обоих видов в этих матрицах сходства белков должны в точности совпадать с указанными в таблицах генов базового и нового видов. Это существенно, потому что последовательности белков в формате FASTA часто имеют идентификаторы с номером версии, чего обычно нет в таблице генов.

Программа addspecies написана на C++ и представляет собой утилиту командной строки Windows или Linux (32/64 бит). Синтаксис командной строки:

addspecies [options] source target [config]

В среде ОС Windows 64 бит указывается имя программы addspecies 64. Помимо опций, обязательные аргументы имеют следующий смысл: source - имя добавляемого нового вида, target – имя существующего вида (базового или потенциального заменителя). Имена видов берутся из указанных в секции [species] файла конфигурации, в качестве которого можно использовать конфигурационный файл программы lossgainRSL (см. главу 5) с добавлением: перед или после любой существующей секции файла добавляется новая секция [add] следующего содержания (указаны примеры значений):

Имя файла конфигурации (и, если требуется, путь к нему) указывается аргументом *config* в командной строке; по умолчанию предполагается файл config.ini в текущем каталоге. Заметим, что программа lossgainRSL игнорирует секцию [add], а программа addspecies игнорирует ненужные ей секции из числа описанных в главе 5, так что на практике обе программы могут пользоваться одним и тем же конфигурационным файлом.

Параметры в секции [add] имеют следующий смысл:

EVALUE - максимально допустимая величина E-value для выравнивания двух белков из геномов видов *source и target*; пары белков с большим значением E-value пропускаются. ORTHOLOG - минимально допустимый вес сходства белков видов source и target, который вычисляется по формуле $w = 2S_{st} / (S_{ss} + S_{tt})$, где S_{st} – величина raw score для выравнивания двух белков из видов *source* и *target*, а S_{ss} , S_{tt} – та же величина для выравнивании каждого белка с самим собой. Если вычисленный вес больше или равен указанного порога, соответствующие гены считаются ортологичными.

PARALOG - минимально допустимый вес сходства белков нового вида, чтобы считать их паралогами. Вес вычисляется по той же формуле, но для двух белков из одного вида.

Программа addspecies предлагает следующие опции (регистр символов не играет роли):

- $-En$ Устанавливает значение *п* в качестве порога по E-value. Если указана эта опция, значение EVALUE в секции [add] конфигурационного файла игнорируется. По умолчанию используется значение 1e-10 (т.е. 10^{-10}).
- $-Gn$ Шаг, с которым отображаются на экране номера проверяемых генов в каждом контиге, по умолчанию 10. Значение 0 отменяет показ номеров генов.
- $-M$ Если указана эта опция, программа сохраняет на диске четыре матрицы значений E-value и raw score между генами видов source и target. За величины для генов принимаются минимальное (E-value) и максимальное (raw score) значения по всем парам белков, которые кодируются данными генами. Матрицы упорядочиваются по возрастанию идентификатора первого гена, затем по убыванию raw score. Для записи матриц должен быть выделен каталог, указанный в секции [data] файла конфигурации (5.2) с помощью специального кода типа данных М (в дополнение к перечисленным в Табл. 4), например, в форме строки $matrix \$ ^*.tsv $\mathbb M$ В этом случае файлам матриц будут присвоены имена вида species1-species2.tsv, где в роли каждого из species1,2 выступают source и target во всех сочетаниях.
- $-IJ$ То же, что и -М, но без сортировки матриц.
- $-Qn$ Опция предназначена для прекращения работы программы на одном из промежуточных этапов. Если указано значение 0, программа addspecies записывает (если это задано) только матрицы сходства генов и останавливается. Если указано значение 1, сверх того формируется таблица ортологов для вида target. Если указано значение 2 или более, сверх того формируется таблица паралогов для вида source; то же самое выполняется по умолчанию.
- $-On$ Устанавливает новое значение порога для веса ортологов. Если указана эта опция, значение ORTHOLOG в секции [add] конфигурационного файла игнорируется. По умолчанию используется значение 0.5.
- $-*Pn*$ Устанавливает новое значение порога для веса паралогов. Если указана эта опция, значение PARALOG в секции [add] конфигурационного файла игнорируется. По умолчанию используется значение 0.5.
- $-Wfile$ Если указана эта опция, программа записывает найденные ортологичные пары генов из видов *source* и *target* в отдельный файл с указанным именем. По умолчанию ортологичные пары генов добавляются в конец таблицы ортологов вида target. Для записи в обоих случаях используется каталог, указанный для типа данных О в разделе [data] файла конфигурации (5.2).

Что касается таблицы паралогов у нового вида, она записывается в каталог, указанный для типа данных P в разделе [data] файла конфигурации [\(5.2\)](#page-14-0), имя присваивается автоматически, как у других таблиц паралогов.

Чтобы воспользоваться программой addspecies, следует вначале внести необходимые изменения в файл конфигурации (в частности, указать имя нового вида) и затем построить с помощью BLASTP четыре матрицы сходства белков [\(4.4\)](#page-8-1), описанные выше. Эти матрицы должны находиться в каталоге, указанном для типа данных A в разделе [data] файла конфигурации [\(5.2\)](#page-14-0), и иметь имена соответствующего формата. Оптимальные значения порогов для E-value и весов находятся эмпирически. Реализованный в программе алгоритм носит экспериментальный характер; он предварительный и подлежит уточнению.

7.2 Использование альтернативной информации об ортологии генов

Оптимальный алгоритм нахождения ортологов неизвестен; приведённая в Ensembl информация об ортологах и паралогах тоже вызывает нарекания в некоторых случаях. Программа lossgainRSL позволяет использовать и другие источники информации об ортологах, но они должны всегда заменяться *целиком*, т.е. для всех видов и генов из решаемой задачи. Например, вместо таблиц ортологов и паралогов по Ensembl во всех случаях могут применяться матрицы сходства белков [\(4.4\)](#page-8-1) или таблица кластеров [\(4.5\)](#page-9-0), а таблицы генов [\(4.1\)](#page-6-0) берутся по-прежнему из Ensembl.

Наконец, БД Ensembl может не использоваться вообще, а в качестве входных данных будет браться таблица ортологичных групп [\(4.5\)](#page-9-0), составленная на основе данных OrthoDB (подробно не описывается).

При запуске программы с разными источниками информации об ортологах будут, вообще говоря, отбираться различные списки искомых генов, что открывает новые возможности. Укажем две из них. Если стоит задача уменьшить перепредсказание, т.е. получить более короткие списки самых надёжных генов с интересующим свойством, например, для проведения мокрых экспериментов, то этого можно достичь, находя пересечение списков, полученных с разными источниками информации об ортологах. И наоборот, снизить недопредсказание можно путём объединения нескольких независимо полученных списков генов.

Помимо таких жёстких теоретико-множественных операций, как пересечение и объединение списков, в случае трёх и более источников можно применить более мягкие процедуры, например, основанные на голосовании, когда решение о каждом гене окончательного списка принимается простым или квалифицированным большинством. На этом пути предстоят дальнейшие исследования, чтобы максимально интегрировать существующие и будущие представления о гомологии генов, особенно у далеко отстоящих видов.