

Руководство пользователя программы lossgainRSL v6.31

Содержание

1	Общие сведения.....	2
2	Установка и проверка	3
2.1	Версия для Windows	3
2.2	Версия для linux	3
3	Синтаксис командной строки	4
4	Описание входных данных.....	7
4.1	Таблица генов.....	8
4.2	Таблица ортологов.....	9
4.3	Таблица паралогов.....	10
4.4	Матрица сходства белков.....	11
4.5	Таблица ортологичных групп.....	12
4.6	Таблица кластеров белков.....	13
5	Файл конфигурации	13
5.1	Секция [mode] – режим работы программы.....	14
5.2	Секция [data] – входные данные.....	17
5.3	Секция [fields] – имена используемых полей.....	18
5.4	Секция [species] – имена видов и групп	20
5.5	Секция [predicate] – предикат для отбора генов	20
5.5.1	Оператор SET и его разновидности.....	21
5.5.2	Операторы IN и INBK	24
5.5.3	Метки и оператор GOTO	27
5.5.4	Оператор ADD	27
5.5.5	Проверка двухвидовых условий. Операторы BOR/EOR/EOS, BAND/EAND.....	28
5.5.6	Операторы OVER/IN2/NEXT и проверка трёхвидовых условий	32
5.5.7	Отладка предиката.....	35
6	Описание выходной информации.....	36
6.1	Протокол работы lossgainRSL	36
6.2	Выходной файл	42
7	Использование других источников входных данных.....	42
7.1	Добавление геномов, отсутствующих в Ensembl	42
7.1.1	Составление новой таблицы генов вручную	43
7.1.2	Импорт таблицы генов из RefSeq	44
7.1.3	Дополнение таблиц ортологов и паралогов с помощью утилиты addspecies.....	45
7.2	Использование альтернативной информации об ортологии генов.....	47
7.3	Использование входных данных исключительно из баз NCBI.....	48

1 Общие сведения

Программа lossgainRSL предназначена для предсказания потерь и приобретений генов между несколькими множествами видов. Она позволяет для фиксированного *опорного* вида (генома) находить его гены, присутствующие и/или отсутствующие в нескольких множествах видов в соответствии с заданной логической функцией (предикатом). Элементарное условие всегда имеет форму «Ген X присутствует в множестве видов S », а предикат представляет собой булеву функцию, образованную из произвольного числа таких условий с помощью логических связок И/ИЛИ/НЕ и скобок. Множества видов не обязательно должны быть таксономическими группами, а могут состояться по произвольным признакам; в частности, множество S может состоять из одного вида и один вид может входить в несколько множеств. Присутствие гена в множестве видов S требует, чтобы он был не менее чем у заданного числа p видов из S (p – параметр условия). 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 гомологичные генам, расположенным в окрестности X' . Число множеств 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/>), RefSeq (<https://www.ncbi.nlm.nih.gov/refseq/>) и 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 MPI, 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).

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 папку. Предлагаются три варианта модуля:

- `lossgainrsl.exe` – вариант для 32-битных версий Windows;
- `lossgainrsl64nompi.exe` – однопроцессорный вариант для Windows 64 бит;
- `lossgainrsl64.exe` – параллельный вариант для Windows 64 бит с MPI.

Для использования параллельного варианта программы в системе Windows необходимо установить распространяемый пакет Microsoft MPI v10.0, который доступен по адресу: <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 распространяемые пакеты Microsoft Visual C++ для Visual Studio 2013 той же, что и программа, разрядности, которые доступны по адресам: http://download.microsoft.com/download/f/1/7/f17b06dd-cc21-40ed-b62a-b390516e90db/vcredist_x86.exe и http://download.microsoft.com/download/f/1/7/f17b06dd-cc21-40ed-b62a-b390516e90db/vcredist_x64.exe.

Проверка работоспособности установленной программы для Windows осуществляется с помощью контрольного примера, доступного для загрузки на странице программы по адресу <http://lab6.iitp.ru/ru/lossgainrsl/example.zip>:

1. Загрузить архив `example.zip` с контрольным примером и распаковать его в рабочий каталог, 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 MPI,
`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.`
Затем следует сравнить содержимое полученного файла `result.txt` с имеющимся в папке примера образцом `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 Open MPI (<https://www.open-mpi.org/>). Используемые на некоторых высокопроизводительных кластерах библиотеки MVARICH и др. реализации также применимы, но могут потребовать небольших изменений в Makefile. 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 `make` без параметров.
5. В иных случаях следует предварительно внести изменения в `Makefile` (рекомендуется сохранить исходный файл, а работать с его копией). Нужно указать команду запуска компилятора C++ для построения программ с MPI (`CXX=`) 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 MPI 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 [options] [%name=value...] [config] [outfile]
```

Здесь указано имя программы в linux; в среде Windows указывается `lossgainrsl` (для 32-битной версии), `lossgainrsl64nomp` (64-битная без поддержки MPI) или `lossgainrsl64` (64-битная с поддержкой Microsoft 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 (в алфавитном порядке):

- b Если указать эту опцию, то все имеющиеся в предикате операторы xIN , включая и любые разновидности, выполняются как операторы $xINBK$, т.е. с обратной проверкой генов-кандидатов (подробнее см. [5.5.2](#)). И наоборот, если данная опция указана в форме $--b$, любые операторы $xINBK$ в составе предиката заменяются на xIN , т.е. без выполнения обратной проверки.

- cN Обычно, когда эта опция не указана или указано $N=0$, программа для текущего гена опорного вида ищет в каждом проверяемом виде только *первый* гомолог этого гена, и при его наличии переходит к проверке следующего гена опорного вида. Если указано $-c1$, то после нахождения первого гомолога программа будет искать в том же виде следующий гомолог того же гена и т.д. вплоть до нахождения *всех* гомологов в каждом виде. При этом работа программы сильно замедляется, а объём выдаваемых результатов увеличивается, так что данную опцию не следует применять в случае сложного предиката с большим числом видов.

- gN Эта опция позволяет управлять периодичностью вывода информации в процессе поиска генов опорного вида, удовлетворяющих заданному предикату. Значение N указывает шаг, с которым на консоль выдаются порядковые номера проверяемых генов в каждой геномной последовательности (хромосоме, скэфолде, контиге и т.п.). Нулевое значение N вообще отменяет эту выдачу, что позволяет немного ускорить счёт. Если опция не указана, по умолчанию используется шаг 10.

- iN Эта опция учитывается только при использовании во входных данных матриц сходства белков; она информирует программу о наличии в этих матрицах необязательных полей (подробнее см. [4.4](#)). Если указано $-i1$, программа считает, что записи каждой матрицы содержат поля 5 и 6 с целочисленными значениями прямого и обратного ранга (см. [Табл. 2](#)). Если указано $-i2$, то предполагается, что эти же поля содержат плавающие значения в интервале $[0, 1]$. Если опция отсутствует или указано $N=0$, то матрицы обрабатываются обычным образом, как содержащие только обязательные поля. Данная опция позволяет ускорить работу программы при использовании в предикате ненулевых значений параметра ORTH/VBH ([5.5.1](#)).

- l Если в командной строке указана эта опция, то все операторы IN (или $INBK$) в предикате интерпретируются как LIN ($LINBK$), т.е. в не-опорном виде свидетели ищутся только в левой полуокрестности гена-кандидата в гомологии, *включающей* сам этот ген. Подробнее см. [5.5.2](#).

- ll То же самое, но все операторы $IN/INBK$ интерпретируются как $LLIN/LLINBK$, т.е. в не-опорном виде свидетели ищутся только в левой полуокрестности гена-кандидата в гомологии, *не включающей* сам этот ген. Подробнее см. [5.5.2](#).

- n Если указать эту опцию, программа `lossgainRSL` будет принудительно исполняться в однопроцессорном режиме, даже если используется версия с MPI. Может использоваться при отсутствии или несовместимой версии библиотек MPI, установленных в операционной системе. Если данная опция не помогает, имеется версия программы без MPI (доступна в разделе файлов для загрузки).

- qN Опция позволяет сократить объем протокола, выдаваемого программой на консоль. По умолчанию $N=0$ и выдается максимально подробный протокол. Если указано -q1, подавляется выдача сообщений об ошибках в таблицах генов (например, о повторении одного и того же белка). Если указано -q2, программа не воспроизводит заданные в исходной конфигурации список видов и предикат, а также не выдает сообщения об ошибках в таблицах гомологов (в т.ч. ортологов, паралогов). Наконец, если указано -q3, программа не выдает также подробности чтения исходных файлов, а только ход поиска генов и конечный результат, так что объем консольной выдачи наименьший.

- r То же самое, но все операторы IN[BK] интерпретируются как RIN[BK], т.е. в неопорном виде свидетели ищутся только в правой полуокрестности гена-кандидата в гомологи, *включающей* сам этот ген. Подробнее см. [5.5.2](#).

- rr То же самое, но все операторы IN[BK] интерпретируются как RRIN[BK], т.е. в неопорном виде свидетели ищутся только в правой полуокрестности гена-кандидата в гомологи, *не включающей* сам этот ген. Подробнее см. [5.5.2](#).

- x Выдавать только найденные гены опорного вида и их гомологи у других видов, даже если в файле конфигурации указан режим вывода синтетических 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 --x.

- %name=value Программа позволяет использовать в файле конфигурации символические обозначения числовых параметров в форме %name, где name – произвольный идентификатор. 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, 5.5e-6. Значение в форме строки символов не должны содержать пробелов. Если команда используется в составе командного сценария (скрипта) Windows, знак процента перед именем параметра необходимо дублировать.

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

- config* Имя файла конфигурации, 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, предполагается 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 `>logfile`, где `logfile` – 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>debugfile`, где `debugfile` – желаемое имя файла, которое может включать путь.

Программа `lossgainRSL` 64-битной версии обеспечивает параллельное выполнение в среде MPI. Для запуска в мультипроцессорном режиме командная строка снабжается префиксом следующего вида:

```
mpirun -np P lossgainRSL [...]
```

где `P` – число используемых логических процессоров. Здесь указаны имена программ в Linux; в среде Windows следует указывать `mpiexec` и `lossgainrsl64`. Прочие аргументы [...] командной строки описаны выше.

4 Описание входных данных

Предпочтительный режим работы с программой `lossgainRSL` предполагает использование входных данных из базы данных Ensembl (www.ensembl.org). Естественно, при этом множество рассматриваемых видов ограничивается представленными в этой базе данных (по состоянию на апрель 2022 г. в версию 106 базы входят 308 геномов позвоночных). 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 MySQL-серверов 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 должен сам сформировать нужные файлы вручную или с применением собственных программ или скриптов. Еще три вида альтернативных данных, которые поддерживает текущая версия программы, описаны в разделах [4.4–4.6](#).

4.1 Таблица генов

Эта таблица обязательна 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.). Порядок строк произвольный. 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.

Имя в Ensembl v106	Описание
Protein stable ID	Идентификатор белка в БД Ensembl
Gene stable ID	Идентификатор гена в БД Ensembl
Chromosome/scaffold name	Имя последовательности верхнего уровня (хромосомы, скэфолда, контига...)
Gene start (bp)	Позиция начала гена, если он на прямой цепи (иначе – конца)
Gene end (bp)	Позиция конца гена, если он на прямой цепи (иначе – начала)
Strand	Указатель цепи (1 означает прямую цепь, –1 обратную)
Gene name*	Символическое имя гена
Gene description*	Аннотация гена

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 106 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 совпадают с именами на входе. Все соответствия имён полей устанавливаются в файле конфигурации и при их изменении не требуется изменять программу.

Примеры таблиц генов содержатся в каталоге genes в составе контрольного примера к программе (<http://lab6.iitp.ru/ru/lossgainsrsl/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 106 (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](#). Поля появятся в записи файла в том порядке, в котором они были отмечены.
8. Нажать слева 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 загружаться файл с именем mart_export.txt. Переименовать его в Human.tsv и направить в каталог, где будут находиться все таблицы генов.

4.2 Таблица ортологов

Эта таблица требуется для опорного вида, а также для каждого вида-заменителя, участвующего в трёхвидовой условии (подробнее см. [5.5.6](#)). Представляет собой файл в TSV-формате, 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. опорного или заменителя), второй – из какого-то другого вида, участвующего в задаче.

Подчеркнём, что таблица ортологов для опорного 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/lossgainsrsl/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 106 (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, после чего свернуть пункт REGION.
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. (Следует развернуть тот из нескольких пунктов, в котором представлены виды с именем, начинающимся на нужную букву, в данном случае [F-J]). 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 начнет загружаться 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 средствами ОС. 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 в файле конфигурации задано, что при выборе ортологических генов должны учитываться не только ортологи, но и их паралоги, либо требуется выводить число паралогов выбранных генов (поле с именем # в [5.3](#)). Таблица паралогов требуется для каждого вида, в котором будут братья 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/lossgains/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 106 (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 загружаться файл с именем mart_export.txt. Переименовать его в Mouse.tsv и направить 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 всех видов, участвующих в задаче, хотя в этом случае в них могут использоваться 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 обязательные поля 1–4, перечисленные в [Табл. 2](#), и может содержать необязательные поля 5–6 из той же таблицы. 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 пару с наибольшим значением raw score и наименьшим значением E-value.

Табл. 2. Список полей записи матрицы сходства белков.

№	Описание
1	Код белка из вида, совпадающего с именем матрицы. Если все таблицы генов соответствуют описанию в разделе 4.1 , кодом служит идентификатор белка в Ensembl. Если таблицы генов имеют совместимый формат, но используют другие идентификаторы белков, то должен быть указан один из таких идентификаторов, иначе строка игнорируется (о чём выдаётся предупреждение). Программа далее агрегирует эти данные путём формирования матрицы для генов (выбирается пара белков с наибольшим сходством), поэтому в данном поле может прямо указываться код гена.
2	Код белка или гена из того же или другого вида, участвующего в данной задаче. Справедливы те же замечания, что и для первого поля.
3	Величина Raw score (или вес, вычисленный на её основе) для оптимального локального выравнивания двух аминокислотных последовательностей, соответствующих указанным белкам.

- 4 Величина E-value для этого оптимального выравнивания.
 - 5 Необязательное поле, содержащее *ранг* данной записи, т.е. номер хита этого белка «первого» вида во «втором» виде, в порядке убывания качества хита. Иначе говоря, при фиксированных белке первого вида и втором виде это поле содержит 1 для лучшего хита, 2 для следующего по качеству выравнивания, и т.д.
 - 6 Необязательное поле, содержащее *обратный ранг* данной записи, т.е. номер хита этого белка «второго» вида в «первом» виде, в порядке убывания качества хита. Иначе говоря, при фиксированных белке второго вида и первом виде это поле содержит 1 для лучшего хита, 2 для следующего по качеству выравнивания, и т.д. В частности, для пары белков, являющихся взаимно лучшими хитами (ВВН), оба поля 5 и 6 содержат 1. Это поле может присутствовать только при наличии поля 5. При отсутствии хита в обратную сторону, поле может быть опущено или содержать специальное значение обратного ранга 999.
- Примечание.** Указанное содержимое полей 5 и 6 предполагает запуск программы с опцией `-i1`. Если используется режим `-i2`, то оба эти поля должны содержать плавающее число в интервале от 0 (наилучший хит) до 1 (отсутствующий хит).

Пример фрагмента матрицы сходства белков, импортированного в Excel, представлен на листе 'Xenopus_scores' файла <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. Другой пример фрагмента матрицы представлен на листе 'Zebrafish_weights' того же файла. Здесь поля 1 и 2 содержат идентификаторы генов в NCBI, а третье поле – вес данного хита, обычно между 0 и 1, и также присутствует поле 5 со значением ранга.

4.5 Таблица ортологичных групп

Данные этого типа содержатся целиком в одной таблице и заменяют собой всю информацию разделов [4.1–4.3](#), 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/>). 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 группы ортологов). Благодаря этому, порядок и набор полей в таблице могут быть достаточно свободными, но каждая строка должна включать по меньшей мере поля, перечисленные в [Табл. 1](#), с одним исключением: вместо поля Protein ID с идентификатором белка обязательно поле Species с идентификатором одного из видов, указанных в файле конфигурации. Имена и порядок полей в строках таблицы может не совпадать с указанными в [Табл. 1](#), правильное соответствие задается в файле 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>.

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, 4.3). Однако таблицы генов (4.1) требуются для каждого вида, участвующего в задаче. Таблица кластеров белков представляет собой текстовый файл в формате TSV, который формируется с помощью ранее разработанной программы кластеризации белков (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](#).

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>.

5 Файл конфигурации

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, однако 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 появления 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 одной информационной строки, где в единственном поле указаны в произвольном порядке 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.

Символ	Описание
	<i>Вид входных данных</i> (необходимо указать один из вариантов)
H	Основной режим работы программы, в котором используются входные данные из БД Ensembl: таблицы генов (4.1), ортологов (4.2) и, если требуется, паралогов (4.3).
A, W	Входными данными служат таблицы генов (4.1) и матрицы сходства белков (4.4). В режиме A матрица принудительно делается симметричной (для обоих симметрично расположенных элементов берется лучший из двух хитов), а в режиме W матрица используется как есть, даже если она асимметрична. Следует иметь в виду, что режим A заметно медленнее и вместо того, чтобы постоянно его использовать, стоит поискать возможность построения симметричных матриц при их первоначальном формировании.
Q	Единственными входными данными служит только таблица ортогрупп (4.5).
C	В качестве входных данных используются таблицы генов (4.1) и единая таблица кластеров белков (4.6).
	<i>Вариант расстояния между генами внутри окрестности</i> (см. Рис. 1)
E	С радиусом окрестности сравнивается расстояние между наиболее удалёнными друг от друга концами генов. Этот режим действует по умолчанию.
B	С радиусом окрестности сравнивается расстояние между серединами генов.
S	С радиусом окрестности сравнивается межгенное расстояние, т.е. расстояние между ближайшими концами генов (отрицательное, если гены перекрываются).
	<i>Добавочные условия синтении*</i> (Рис. 2)
O	Взаимно ортологичные гены синтеничных блоков должны располагаться в одинаковом порядке, а их направление безразлично.

Символ	Описание
D	Взаимно ортологичные гены синтеничных блоков должны иметь одинаковое направление, а их порядок не имеет значения.
M	Условия синтении включают и зеркальный вариант расположения, т.е. когда порядок и направление всех генов синтеничного блока инвертируются.
<p data-bbox="370 373 852 415"><i>Представление выходных данных</i></p> <p data-bbox="272 426 1485 709">X Если указан этот режим, то из каждого найденного синтеничного блока генов в выходном файле появится только искомый ген опорного вида и его гомологи в других видах (в одной строке), а информация о свидетелях не выдается. В противном случае в выходной файл будет выдан целиком синтеничный блок (как группа последовательных строк), причём первым всегда выдаётся искомый ген опорного вида и его гомологи у других видов, а в последующих строках – свидетели. Эту настройку можно впоследствии изменить опциями командной строки (глава 3).</p> <p data-bbox="272 720 1510 793">I В режиме X этот параметр игнорируется. Если не задан режим X, в выходной файл между синтеничными блоками будет вставляться пустая строка.</p> <p data-bbox="272 804 1507 1129">J В режиме X этот параметр игнорируется. Если не задан режим X, в каждую строку выходного файла, включая пустую вследствие параметра I, добавляются в конец два поля: первое содержит идентификатор искомого гена в опорном виде, для которого найден данный синтеничный блок, второе – номер строки в этом блоке. Например, если синтеничный блок в опорном виде состоит из гена <i>Id</i> и двух генов-свидетелей, в выходном файле появятся 4 строки (для самого <i>Id</i>, 1-го свидетеля, 2-го свидетеля и пустая), у которых в добавленных полях стоят пары (<i>Id</i>, 0), (<i>Id</i>, 1), (<i>Id</i>, 2) и (<i>Id</i>, 3) соответственно. Это позволяет в дальнейшем легко упорядочивать блоки по генам опорного вида.</p>	
<p data-bbox="370 1150 1166 1192"><i>Отладочные режимы</i> (игнорируются при запуске с MPI)</p> <p data-bbox="272 1203 1485 1339">G Указывает, что данная секция файла конфигурации содержит еще одну или более строк с идентификаторами генов опорного вида, обработка которых должна быть особо отражена в протоколе. Такой протокол отладки выдается в стандартный поток <i>stderr</i> (см. главу 3).</p> <p data-bbox="272 1350 1442 1465">P То же, но дополнительные строки содержат идентификаторы белков опорного вида. В отладочном протоколе будет отражена обработка генов, кодирующих указанные белки.</p> <p data-bbox="272 1476 1507 1591">T То же, но дополнительные строки содержат идентификаторы последовательностей в геноме опорного вида (имена хромосом или скэффолдов). В протоколе будет отражена обработка всех генов указанных последовательностей.</p> <p data-bbox="272 1602 1490 1814">N То же, но дополнительные строки содержат порядковые номера геномных последовательностей и (опционально) генов в них. Нумерация во всех случаях начинается с 0. Номер гена указывается после символа подчеркивания, например, 493_5 означает шестой по порядку ген в 494-й по порядку последовательности в геноме опорного вида. В отладочном протоколе будет отражена обработка указанных (или всех) генов перечисленных последовательностей.</p>	

Символ	Описание
V	Данный параметр может использоваться вместе с любым из вышеуказанных в данной группе; он включает максимально подробную отладочную выдачу. Следует иметь в виду, что при этом могут формироваться протоколы гигантских размеров, так что рекомендуется перенаправлять выдачу в файл и указывать как можно меньше генов.

Примечание:

- * Режимы этой группы не альтернативны, а могут указываться 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.

Сразу после строки, задающей режимы работы программы, могут идти одна или более строк с указанием объектов, для которых требуется выдать отладочную информацию; в роли таких объектов могут выступать идентификаторы или порядковые номера генов, белков или геномных последовательностей. Каждая строка содержит один или несколько объектов; в качестве разделителя используется символ 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 имеющиеся строки с информацией для отладки игнорируются. Дальнейшие рекомендации по отладке приведены в разделе [5.5.7](#).

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

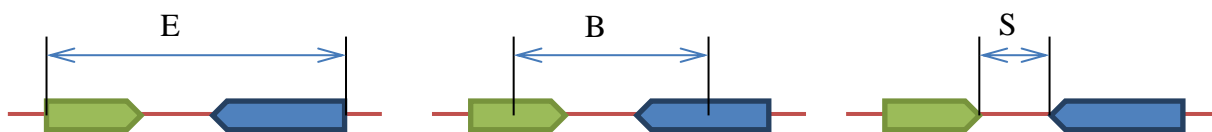


Рис. 1. Варианты учёта попадания генов в окрестность.

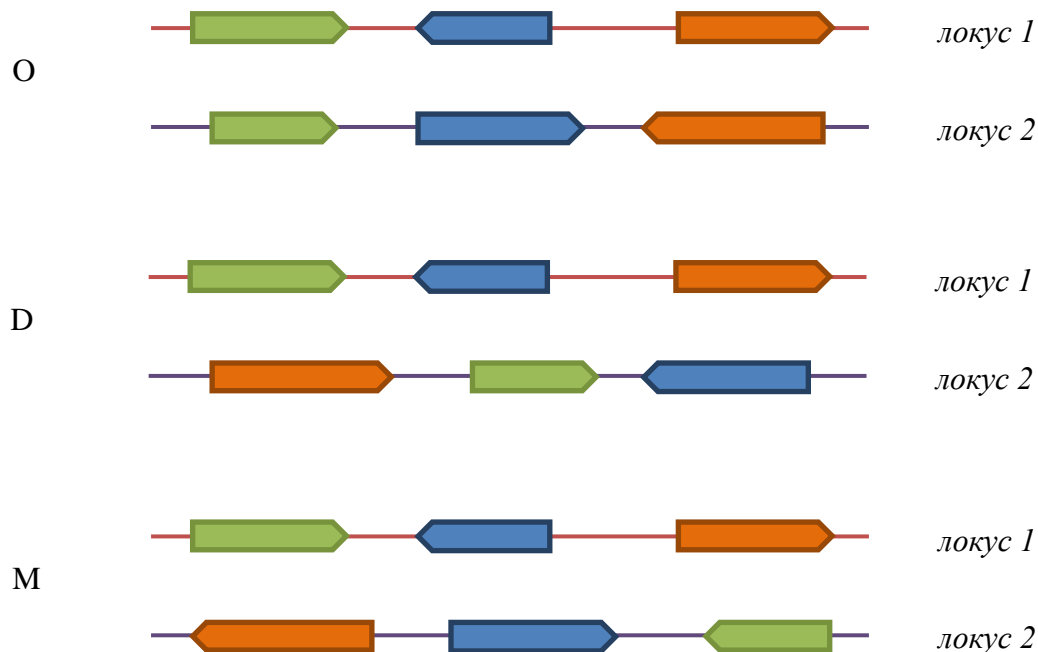


Рис. 2. Варианты учёта синтении двух локусов. О – одинаковый порядок генов, D – одинаковые направления генов, М – допускается зеркальный вариант расположения.

5.2 Секция [data] – входные данные

В этой секции файла конфигурации указываются папки со входными данными каждого типа и расширения соответствующих файлов. В качестве имени файла (или его переменной части) 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]
I genes\*.tsv
O orthologs\*.tsv
P paralogs\*.tsv
H homologs\*.tsv
A scores\*.tsv
Q orthogroups\odb10.tsv
C protein-clusters.tsv
```

Табл. 4. Типы входных данных.

Код	Содержание второго поля для этого типа входных данных
I	Относительный или абсолютный путь к каталогу, где находятся таблицы генов (см. раздел 4.1). Вместо имени файла указывается звёздочка, а после неё расширение имени файла, если есть. Таким образом, все таблицы генов должны находиться в указанном каталоге и иметь одинаковое расширение (или расширения).
O	Относительный или абсолютный путь к каталогу, где находятся таблицы ортологов (см. раздел 4.2). Вместо имени файла указывается звёздочка, а после неё расширение имени файла, если есть. Таким образом, все таблицы ортологов должны находиться в одном каталоге и иметь одинаковое расширение (или расширения).
P	Относительный или абсолютный путь к каталогу, где находятся таблицы паралогов (см. раздел 4.3). Вместо имени файла указывается звёздочка, а после неё расширение имени файла, если есть. Таким образом, все таблицы паралогов должны находиться в указанном каталоге и иметь одинаковое расширение (или расширения).
H	Относительный или абсолютный путь к каталогу, где находятся единые таблицы ортологов и паралогов для всех или некоторых видов. Фактически каждая такая таблица эквивалентна объединению двух таблиц, ортологов и паралогов, описанных в разделах 4.2 , 4.3 . Вместо имени файла указывается звёздочка, а после неё расширение имени файла, если есть. Таким образом, все таблицы гомологов должны находиться в указанном каталоге и иметь одинаковое расширение (или расширения).
A	Относительный или абсолютный путь к каталогу, где находятся матрицы сходства белков (см. раздел 4.4). Вместо имени файла указывается звёздочка, а после неё расширение имени файла, если есть. Таким образом, все матрицы сходства белков должны находиться в указанном каталоге и иметь одинаковое расширение (или расширения).
Q	Относительный или абсолютный путь и полное имя файла с таблицей ортогрупп (раздел 4.5). Если таблица разбита на несколько частей, то указывается необходимое число таких строк. Части таблицы будут объединяться в том порядке, как они указаны в файле конфигурации. Строка заголовка с именами полей требуется только в первой части таблицы (но может присутствовать и в последующих).
C	Относительный или абсолютный путь и полное имя файла с таблицей кластеров белков (раздел 4.6). Если таблица разбита на несколько частей, то указывается необходимое число таких строк. Части таблицы будут объединяться в том порядке, как они указаны в файле конфигурации. Строка заголовка с именами полей требуется только в первой части таблицы (но может присутствовать и в последующих).

5.3 Секция [fields] – имена используемых полей

Таблицы генов ([4.1](#)), ортогрупп ([4.5](#)) и белковых кластеров ([4.6](#)) 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 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](#) (в том числе, совместно с кластерами белков [4.6](#)), то поле с указанным здесь именем содержит идентификатор белка в БД Ensembl или ином источнике. Если же используется таблица ортогрупп [4.5](#), здесь указывается имя поля, содержащего идентификатор *вида* (см. [5.4](#)).
- 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, скэффолда, 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 v106, эта секция может выглядеть так:

```
[fields]
Protein stable ID      Protein
Gene stable ID        Gene
Chromosome/scaffold name  Contig
Gene start (bp)       Start
Gene end (bp)         End
Strand                +|-
Gene name              Name
Gene description       Description
```

Кроме того, имеется особое поле с выделенным именем #, которое также можно запросить в числе дополнительных полей. Это поле в явном виде не присутствует во входных файлах, но при необходимости может быть сформировано программой и напечатано в выходном файле. Оно содержит число паралогов у того гена, для которого выдается. Это поле всегда нужно указывать *последним*, и в списке полей здесь, и при выдаче (в секции [predicate]), см. [5.5.4](#).

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 опорного 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 состояться на основании любых желаемых признаков. Используемый синтаксис не позволяет одному виду входить сразу в несколько групп; это ограничение обходится путём повторения имени вида в другой группе.

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 опорного 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 аварийное завершение 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 для этого предикат. Так, 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, обрабатывая по очереди все гены опорного вида. Для каждого очередного гена 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 опорному 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 опорного вида *R* и другого вида *A* (*двухвидовое* условие) или опорного вида *R*, т.н. вида-заменителя *S* и другого вида *A* (*трёхвидовое* условие). Условия проверяются в той последовательности, как они записаны, однако очерёдность 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 опорного 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 (1) достигнут его конец (за которым неявно следует метка YES, поэтому текущий ген опорного вида будет отобран и появится в выходном файле), либо (2) происходит переход на метку NO (тогда содержимое выходного файла не меняется). В обоих случаях программа переходит к проверке предиката для следующего гена опорного вида *R*. Работа lossgainRSL заканчивается после проверки всех генов опорного вида.

В последующих параграфах даётся систематическое описание всех возможностей языка описания предикатов. Примеры законченных предикатов можно найти в конфигурационных файлах в составе контрольного примера (<http://lab6.iitp.ru/ru/lossgainrsl/example.zip>).

5.5.1 Оператор SET и его разновидности

Оператор служит для установки или изменения значений параметров отбора искомых генов. Как и прочие операторы, он должен целиком записываться в одной строке. Содержит от 1 до 11 полей, разделённых знаком табуляции. Как минимум, должно присутствовать первое поле. Синтаксис оператора:

```
SET [ , w, r, b, k, s, e, l, h1, h2 ] [WIT[NESS]=w] [RANGE=r] [ {ORTH|BBH}=b ] [RANK=k]
[SCORE=s] [EVAL=e] [LAMBDA=l] [HOLD1=h1] [HOLD2=h2] [HEAD[ING]=CS-list]
```

Всего имеется 9 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 пропускать, сохраняя запятые, если дальше ещё есть значения), либо в последующих полях в произвольном порядке и количестве (каждый параметр опознаётся 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 HEAD или HEADING. Он определяет вид строки заголовка в выходном файле.

Значения числовых параметров можно указывать либо числом, либо символической ссылкой в форме *%name*, где *name* – 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. Тем же способом можно определять параметры в виде строк символов для переменных полей заголовка в операторе HEAD, однако такие символьные строки не должны содержать пробелов.

Табл. 5. Параметры программы, устанавливаемые оператором SET.

Имя	Ключевое слово	Значения	По умолчанию	Описание
<i>w</i>	WIT WITNESS	0, 1, 2	2	Число генов-свидетелей в составе синтетического блока генов, в дополнение к основному гену <i>X</i> .
<i>r</i>	RANGE	> 0	2000000	Радиус окрестности, т.е. расстояние в парах оснований, в пределах которого ищутся свидетели для основного гена (см. также Рис. 1). Может указываться суффикс <i>K k</i> или <i>M m</i> для значений в <i>Kbr</i> или <i>Mbr</i> , соответственно. Например, радиус окрестности 200 килобаз можно задать значениями 200000, 200K или 0.2M.
<i>b</i>	ORTH BVH	0...511	0	<p>1) Для исходных данных из Ensembl допускаются значения 0...7 (три бита), где каждый бит указывает условие выбора гомолога (1 – только ортолог, 0 – ортолог или его паралог). Младший бит относится к проверяемому гену <i>X</i>, следующий – к 1-му свидетелю <i>Y</i>, старший – ко 2-му свидетелю <i>Z</i>. Например, <i>b=1</i> означает, что для <i>X</i> берутся только ортологи, а для <i>Y</i> и <i>Z</i> – ортологи и их паралоги; а <i>b=7</i> означает, что гомологами для <i>X</i>, <i>Y</i> и <i>Z</i> считаются только ортологи.</p> <p>2) В случае матриц сходства белков (см. 4.4) этот параметр ограничивает выбор гомологов только лучшими хитами. Смысл битов: 1 – для гена <i>X</i> берётся BVH (взаимно лучший хит), 2 – для гена <i>Y</i> берётся BVH, 4 – для гена <i>Z</i> берётся BVH, 8 – для гена <i>X</i> берётся VH (лучший хит вперед), 16 – для гена <i>Y</i> берётся VH, 32 – для гена <i>Z</i> берётся VH, 64 – для гена <i>X</i> берётся лучший хит назад, 128 – для гена <i>Y</i> берётся лучший хит назад, 256 – для гена <i>Z</i> берётся лучший хит назад. Отметим, что если для гена установлены оба бита VH (вперед и назад), то автоматически будет установлен и бит BVH.</p>

Имя	Ключевое слово	Значения	По умолчанию	Описание
<i>k</i>	RANK	> 0	10	<p>Параметр учитывается, только если в качестве входных данных используются матрицы сходства белков (4.4). Учитываются только такие хиты, ранг которых не превышает указанного порога.</p> <p>Например, если указано значение 3 и программа запущена с опцией <i>-i1</i>, для каждого белка/гена программа будет учитывать в каждом виде <i>A</i> не более трёх лучших хитов, даже если в матрице имеется больше. В режиме <i>-i2</i> значение <i>k</i> – действительное число и будут учитываться все хиты с меньшим значением ранга в интервале [0, 1]; их число заранее не известно (см. раздел 3).</p>
<i>s</i>	SCORE	≥ 0	0.001	<p>Параметр учитывается, только если в качестве входных данных используются матрицы сходства белков (4.4). Гомологичными считаются только белки/гены, при выравнивании которых величина <i>raw score</i> или вес не меньше указанного порога.</p>
<i>e</i>	EVAL	≥ 0	1e-5	<p>Параметр учитывается, только если в качестве входных данных используются матрицы сходства белков (4.4). Гомологичными считаются только белки/гены, при выравнивании которых величина <i>E-value</i> не больше указанного порога.</p>
<i>l</i>	LAMBDA LAMB	0–1	0	<p>Параметр учитывается, только если в качестве входных данных используются матрицы сходства белков (4.4); он задает величину λ в условии обратной проверки (см. 5.5.2).</p>
<i>h1</i>	HOLD1	0...7	1	<p>Смысл трёх битов значения параметра – указать, какие гены не меняются при проверке предиката для текущего гена опорного вида, а какие могут свободно варьироваться:</p> <ul style="list-style-type: none"> 1 – проверка выполняется с одинаковым геном <i>X</i>, 2 – проверка выполняется с одинаковым геном-свидетелем <i>Y</i> (иначе условие не выполнено), 4 – проверка выполняется с одинаковым <i>Z</i>. <p>Например, если указано значение 7 и предикат содержит два проверяемых условия, то в обоих должна участвовать одна и та же тройка генов.</p>
<i>h2</i>	HOLD2	0...7	1	<p>Параметр имеет значение только для трёхвидового условия (5.5.6); он аналогичен <i>h1</i>, но регулирует связь между двумя элементарными условиями в составе трёхвидового. В основном режиме, когда данные берутся из Ensembl, следует указывать 1.</p>

Имя	Ключевое слово	Значения	По умолчанию	Описание
–	HEAD HEADING	Значением является текстовая строка, которая задаёт заголовки полей в выходном файле слева направо, разделяя их запятой (поэтому запятая не должна появляться в заголовке поля). Если параметр опущен, заголовки полей в выходном файле совпадают с именами соответствующих полей входного файла или, если указаны, их алиасами в секции [fields] файла конфигурации (см. 5.3). При формировании текстовой строки могут использоваться символьные параметры в форме %name, значения которых задаются в командной строке.		

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

Если требуется (пере)установить не сразу все, а только 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:

```
SET RANGE=1M WITNESS=1
```

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

```
RANGE      500K
WITNESS    1
```

Помимо оператора SET и его разновидностей, параметры можно для краткости устанавливать прямо в операторах, осуществляющих проверку, таких как IN, OVER, IN2, через запятую вслед за кодом оператора – 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 Операторы IN и INBK

Этот оператор проверяет наличие текущего гена X опорного вида R в указанном другом виде A или группе видов. Такая проверка признается успешной, если одновременно выполнены два условия: (1) в A 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 отличных друг от друга генов-свидетелей, например, Y и Z , у которых есть несовпадающие гомологи Y' и Z' , соответственно, в окрестности гена X' радиуса r' в A (Рис. 3). Радиус интерпретируется с учётом выбранного режима определения окрестности (см. Табл. 3 в разделе 5.1), на Рис. 3 изображён вариант 'E'. В этом условии участвует два вида, поэтому оно называется *двухвидовым*. Как показано стрелками на рисунке, в нём выполняется проверка гомологичности в направлении от опорного вида R к другому виду A ; такую проверку назовём *прямой*. Требования гомологичности для проверяемого гена и для свидетелей могут быть разными, например, для свидетеля может выполняться более строгая проверка, чем для гена X , что изображается соответственно сплошной и штриховой стрелкой.

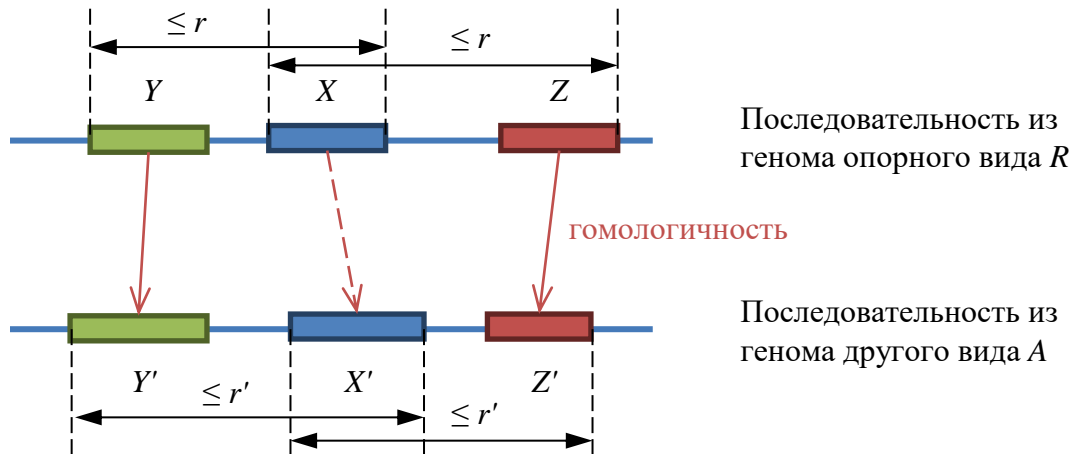


Рис. 3. Прямая проверка двухвидового условия при одном из вариантов синтении.

Если используются матрицы сходства белков (4.4), двухвидовое условие может включать *обратную проверку*, для чего используется форма INBK оператора IN (Рис. 4). Такая проверка выполняется для найденного в результате прямой проверки гомолога X' и призвана подтвердить, что в виде R отсутствует альтернативный ген U , гомологом которого в A также является X' , причём U существенно более гомологичен X' , чем X гомологичен X' . Для этого проверяется условие: (3) в R нет такого гена U , для которого X' является гомологом в A с выполнением тех же условий в отношении свидетелей, что и для X , и такого, что справедливо неравенство:

$$\begin{cases} W(X', U) / W(X, X') > \lambda & \text{если } W(\bullet) - \text{raw score} \\ W(X', U) - W(X, X') > \lambda & \text{если } W(\bullet) - \text{вес,} \end{cases}$$

где $W(\bullet)$ – значение поля 3 матрицы сходства белков (Табл. 2. Список полей записи матрицы сходства белков. Табл. 2), а λ – параметр l в операторе SET (5.5.1).

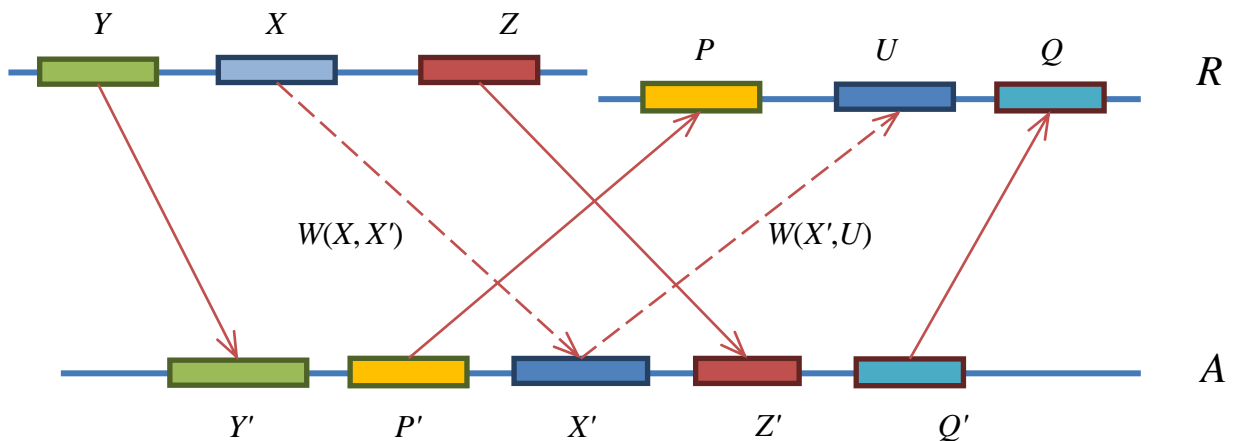


Рис. 4. Двухвидовое условие с прямой и обратной проверкой.

Синтаксис оператора IN/INBK предусматривает два или три поля:

IN [BK] [, w , r , b , k , s , e , l , h1 , h2] *species* [*yes-label* | *-no-label*]

Здесь в первом поле оператора 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 виде *A*, т.е. используется в качестве *r'* на [Рис. 3](#), а в качестве радиуса *r* окрестности в *R* 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 вида *A* или группы видов – одно из имён, перечисленных в секции [species] файла конфигурации ([5.4](#)). 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/INBK 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 *многократно вложенный цикл* проверки заданного условия: вначале по всем видам из группы, затем для каждого вида – по последовательностям в составе его генома, для каждой – по её генам, для каждого гена – по гомологичным генам в другом виде, по потенциальным свидетелям и их гомологам и т.д. В варианте INBK глубина вложения циклов еще больше за счет проверки потенциальных генов *U* и свидетелей для них. 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 *yes-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/INBK. Любые дальнейшие действия выполняются только после окончания или прерывания цикла.

Имеются также разновидности оператора IN, позволяющие проверять в не-опорном виде (или видах) *A* наличие свидетелей для гомолога не в целой его окрестности, а отдельно в каждой полуокрестности, включая или исключая область самого гомолога. Синтаксис этих операторов такой же, как у оператора IN выше, но код оператора дополняется одним из четырех префиксов, в результате чего образуется новое ключевое слово:

LIN – условие (2) выполнено для *левой* (меньшие значения позиции) полуокрестности гена *X'* в виде *A*, *включая* область самого гена *X'*;

LLIN – условие (2) выполнено для *левой* (меньшие значения позиции) полуокрестности гена *X'* в виде *A*, *не включая* область самого гена *X'*;

RIN – условие (2) выполнено для *правой* (бóльшие значения позиции) полуокрестности гена X' в виде A , *включая* область самого гена X' ;

RRIN – условие (2) выполнено для *правой* (бóльшие значения позиции) полуокрестности гена X' в виде A , *не включая* область самого гена X' .

За исключением области в виде A , которая проверяется при поиске нужного числа свидетелей, действие этих разновидностей оператора IN ничем не отличается от основного варианта с симметричной окрестностью. Все они также допускают обратную проверку, т.е. независимо от указанного префикса код оператора может дополняться суффиксом ВК, например, LLINBK. Однако ограничение поиска свидетелей лишь одной полуокрестностью относится только к **прямой** проверке, а в ходе обратной проверки анализируются полные окрестности генов X' и U , потому что главная цель обратной проверки – уменьшить ошибки при выборе гомолога X' текущего гена X .

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 добавить в текущую строку выходного файла необходимые поля из определённой строки таблицы генов для указанного в операторе вида. Синтаксис оператора ADD:

```
ADD      species      [ [-]field1 [ ,field2 . . . ] ]
```

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. Имена полей могут указываться с использованием параметров командной строки в форме *%name* (глава [3](#)).

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

- Если в операторе ADD указан опорный вид *R*, то в выходной файл переносятся данные текущего (проверяемого) гена *X* из *R*.
- 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 опорного вида со свидетелями в этом другом виде. Если оператор IN для данного вида не выполнялся, возникает ошибка времени выполнения, что говорит о необходимости исправления предиката.
- Если для вида *A* условие *не было выполнено* (нет гомолога или свидетелей), выводятся поля с пустым содержимым.
- 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 во входных файлах), 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 опорного вида выдаётся только одна строка выходного файла, содержащая данные для гена *X* опорного 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* опорного вида и, если есть, 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 синтетическом блоке фиксированный: *X, Y, Z* (или *X, Y* или *X*), т.е. искомый ген опорного вида всегда стоит в первой строке блока.

Окончательный перенос подготовленных строк 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. Операторы BOR/EOR/EOS, BAND/EAND

Семантику проверки двухвидовых условий в различных вариантах удобно описывать на примере. Пусть опорный вид – лягушка и кроме того рассматриваются две группы видов, рыбы и млекопитающие, в каждой из которых два представителя. Предполагаются данные из Ensembl, поэтому обратная проверка не используется. Пусть 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
Fugu
*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 для текущего гена X лягушки, есть ли он 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 X 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 X лягушки, за исключением идентификатора белка. Ещё один оператор ADD добавляет в текущую строку выходного файла два поля с идентификатором и именем найденного гена X' . (Предполагаем, что программа работает в режиме X из Табл. 3 – выдача только искомым 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 неясно, 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 исходных 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.

(B) Для исправления указанного недостатка можно составить более сложный предикат с разветвлением:

```

[predicate]
SET, 2, 2000K, 7
HEADING Xenopus, Contig, Start, End, Name, Description, Danio, Name, Fugu, Name
IN Danio -NotDanio
ADD Xenopus Gene, Contig, Start, End, Name, Description
ADD Danio Gene, Name
IN Fugu
ADD Fugu Gene, Name
GOTO Final
:NotDanio
IN Fugu -NO
ADD Xenopus Gene, Contig, Start, End, Name, Description
ADD Danio Gene, Name
ADD Fugu Gene, Name
:Final
IN, 1, , 2 *Mammal NO

```

В этом случае каждое поле выходного файла содержит (или не содержит) данные только для 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 иметь значительную длину.

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 перейдет к метке NotDanio. Иначе выполняется следующий за IN оператор ADD, в котором для генов опорного 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.

Начиная с метки Final, две ветви снова соединяются и делается та же заключительная проверка, что и в предикате (A). В зависимости от её результатов, сформированная строка либо записывается в выходной файл, либо теряется, если ген *X* 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 (или BOR/EOS), которые играют роль скобок, охватывающих несколько независимых проверок, для успеха которых устанавливается общий количественный порог.

(С) Эквивалентный варианту (В) предикат с использованием этих операторов имеет вид:

```
[predicate]
SET, 2, 2000K, 7
HEADING Xenopus, Contig, Start, End, Name, Description, Danio, Name, Fugu, Name
BOR
IN Danio
ADD Xenopus Gene, Contig, Start, End, Name, Description
ADD Danio Gene, Name
IN Fugu
ADD Fugu Gene, Name
EOR, 1 -NO
IN, 1, , 2 *Mammal NO
```

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, но если проверка была неуспешной, содержимое добавляемых полей будет пустым. Собственно проверка комплексного условия выполняется оператором EOR, который ставится после последней проверяемой альтернативы и имеет следующий синтаксис:

```
EOR [ , n] [yes-label | -no-label]
```

Здесь n – минимально необходимое число выполненных альтернативных проверок, при котором всё условие считается выполненным. Если n 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 опорного вида. Предусмотрена также разновидность этого оператора с аналогичным синтаксисом, но немного отличающейся семантикой:

```
EOS [ , n] [yes-label | -no-label]
```

Отличие состоит в том, что здесь условие считается выполненным, если выполнены *более* n альтернативных проверок (в отличие от EOR, где используется нестрогое неравенство). Пара операторов BOR/EOR (или BOR/EOS) 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:

(D) Эквивалентный варианту (C) предикат с использованием операторов BAND/EAND:

```
[predicate]
SET, 2, 2000K, 7
HEADING      Xenopus, Contig, Start, End, Name, Description, Danio, Name, Fugu, Name
BAND
IN            Danio
ADD          Xenopus      Gene, Contig, Start, End, Name, Description
ADD          Danio        Gene, Name
IN            Fugu
ADD          Fugu         Gene, Name
EAND, 2      NO
IN, 1, , 2   *Mammal     NO
```

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 – опорный R и еще два, из которых в одном должен быть гомолог текущего гена (такой гомолог будем называть *заменителем* текущего гена, а сам вид – *видом-заменителем* S), а в другом виде A – есть или нет гомолога для гена-заменителя ([Рис. 5](#)). 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 транзитивно замкнуто, нужда в трёхвидовом условии возникает только в том случае, когда в окрестности гена X 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, что ген X имеет в другом виде гомолог X'' с сохранением синтении.

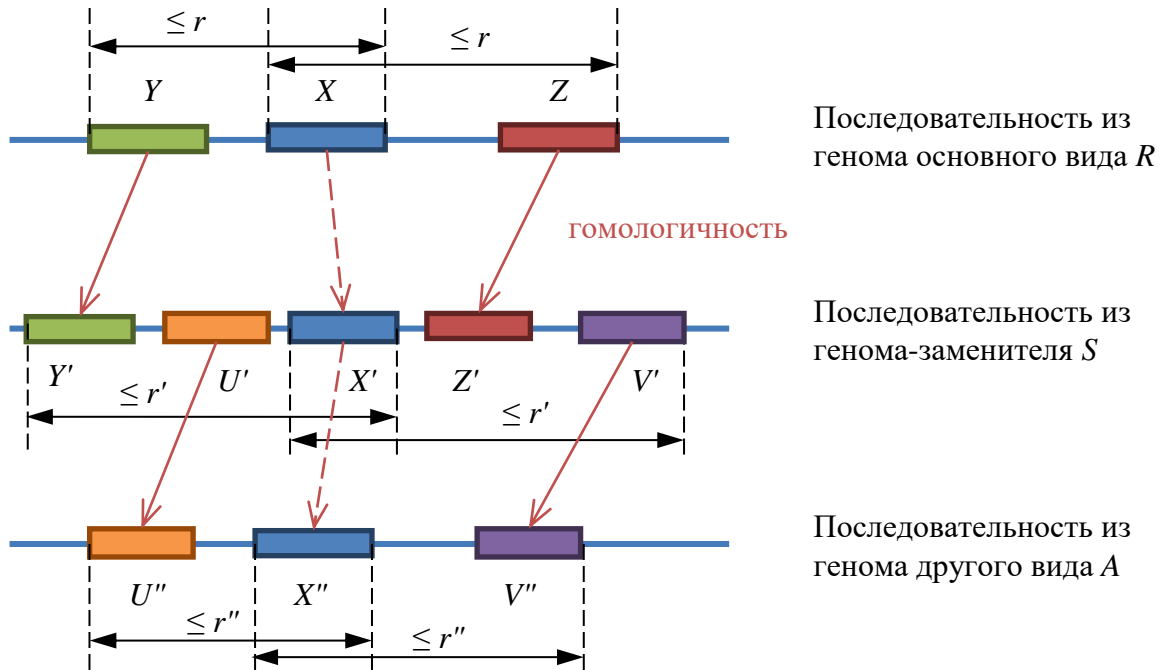


Рис. 5. Выполнение трёхвидового условия при одном из вариантов синтении.

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. 5). 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 и матриц сходства белков), то следует указывать значение 1.

Напомним, что двухвидовое условие определяется единственным оператором IN, который задаёт вложенный цикл перебора генов указанного в нём не-опорного вида A или группы

видов (5.5.2). Чтобы определить трёхвидовое условие, используются три оператора – OVER, IN2 и NEXT. Операторы OVER и IN2 задают два вложенных друг в друга цикла аналогичного перебора генов в двух не-опорных видах, а оператор NEXT указывает конец тела внешнего цикла. Очередность и синтаксис указанных операторов следующие:

```
OVER [ , w , r , b , k , s , e , l , h1 , h2 ]   species1   [ yes-label | -no-label ]
IN2 [ , w , r , b , k , s , e , l , h1 , h2 ]   species2   [ yes-label | -no-label ]
...
NEXT
```

Внешний цикл определяется оператором OVER, в котором вместо *species1* указывается имя вида-заменителя *S* или группы видов, в которых ищется ген-заменитель. Ген-заменитель *X'* должен быть гомологом текущего гена *X* опорного вида *R* и, кроме того, в окрестности *X'* должны присутствовать гены-свидетели *Y'* и *Z'*, которые являются гомологами генов *Y* и *Z* в окрестности гена *X* опорного 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 произойдёт, когда найдется первый подходящий ген-заменитель (без учета условий внутреннего цикла), в противном случае программа переходит к следующему оператору. Такой вариант условия, по-видимому, не имеет большого 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 *X*, для которого проверяется наличие или отсутствие в виде *A* (или группе видов) – аргумент *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 *X'*.

Следует проявлять осторожность при включении в тело цикла операторов с побочным эффектом, таких как ADD, поскольку это может привести к заикливанию 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.

Начиная с версии 6.31, программа `lossgainRSL` поддерживает обратную проверку в трёхвидовых условиях, которая (по желанию) выполняется при поиске гомологов независимо в геномах S и A , для чего используются разновидности операторов, `OVERBK` и `IN2BK`, соответственно. Сама обратная проверка осуществляется аналогично оператору `INBK` (5.5.2) и точно также дополнительно управляется опциями `-b` и `--b` в командной строке запуска программы (см. 3).

Замечание. В текущей версии программы `lossgainRSL` не реализована поддержка префиксов в операторах `OVER` и `IN2`, в отличие от оператора `IN` (5.5.2), поэтому в трёхвидовом условии невозможен поиск свидетелей только в полуокрестности потенциального гомолога. Такая возможность может быть реализована в будущих версиях программы.

5.5.7 Отладка предиката

По существу, записанный в конфигурационном файле предикат тоже является *программой*, которая для каждого очередного гена X исполняется в режиме интерпретации программой `lossgainRSL`. Как любая программа, предикат может содержать ошибки, приводящие к получению ошибочных результатов или отказам при работе `lossgainRSL`, таким как аварийное завершение или заикливание. 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;
- 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 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 заикливается, по протоколу можно установить, на каком контиге и гене опорного вида это произошло (последние выданные на консоль значения в строке `сККК xLLL`). Поскольку по умолчанию номера генов для скорости выдаются с шагом 10, имеет смысл повторить запуск с опцией `-g1` (см. главу 3).

Найденные значения KKK 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 KKK_LLL. Подробнее см. [Табл. 3](#) в разделе [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 and homology analysis utility (v6.31)
```

```
arguments: for_manual.ini result.txt
```

```
Taxa: 6 Species: 41 Mode: H E X
```

В первой строке указан номер версии программы, в следующей – указанные при её запуске аргументы. После этого показано число групп видов, которые определены в файле конфигурации (6) и общее число видов (41), а также активные режимы работы программы ([5.1](#)). При запуске программы в среде MPI указывается ещё число параллельно работающих ветвей.

```
*[1]: mus_musculus
*Human[1]: homo_sapiens
*Apes[5]: gorilla_gorilla nomascus_leucogenys pan_paniscus pan_troglodytes pongo_abelii
*Cercopithecidae[12]: cercocebus_atys chlorocebus_sabaeus colobus_angolensis palliatus
macaca_fascicularis macaca_mulatta macaca_nemestrina mandrillus_leucophaeus papio_anubis
piliocolobus_tephrosceles rhinopithecus_bieti rhinopithecus_roxellana
theropithecus_gelada
*Other_mammals[13]: bos_taurus canis_lupus_familiaris delphinapterus_leucas
equus_caballus heterocephalus_glaber_female loxodonta_africana lynx_canadensis
monodelphis_domestica myotis_lucifugus panthera_leo physeter_catodon
rhinolophus_ferrumequinum ursus_maritimus
*Other_primates[9]: aotus_nancymae callithrix_jacchus carlito_syrichtha cebus_capucinus
microcebus_murinus otolemur_garnettii prolemur_simus propithecus_coquereli
saimiri_boliviensis boliviensis
```

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 *Human, состоящая из одного генома человека; группа *Apes из пяти геномов человекообразных обезьян, перечисленных после двоеточия (горилла, гиббон, бонобо, шимпанзе, орангутан); группа *Cercopithecidae (мартышковые) из указанных 12 геномов; группа *Other_mammals, состоящая из 13 геномов, и группа *Other_primates из 9 геномов. Этот фрагмент не выдаётся, если была указана опция -q2 или -q3.

Далее приводятся результаты синтаксического разбора предиката (также не выдаются, если в командной строке была указана опция -q2 или -q3).

```
The predicate (scenario) consists of 87 terms:
```

0	SET		1	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1	
1	SET		2	2	5000000	7	10.000	0.001	1.0e-005	0.400	1	1	
2	IN	*Human	NO	3	1	5000000	7	10.000	0.001	1.0e-005	0.400	1	1
3	SET		4	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1	

Здесь в первой строке указано общее число операторов, распознанных в описании предиката (87), и последующие 87 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 WITNESS (2), радиус окрестности RANGE (2 Mbp), ORTH (7), RANK (10), SCORE (0.001), EVAL (10^{-5}), LAMBDA (0.4), HOLD1 (1) и HOLD2 (1). Эти значения параметров будут использоваться по умолчанию; они могут временно заменяться с помощью других операторов SET, примером чего служит оператор с номером 1. Как видно из протокола, радиус окрестности временно изменён на 5 мегабаз). Затем оператор IN проверяет, есть ли текущий ген мыши у человека, причем только для этого оператора используется один свидетель. Если да, то выполняется переход к метке NO (т.е. ген пропускается), иначе программа переходит к следующему оператору (3). Еще один оператор SET в последней строке фрагмента восстанавливает прежнее значение RANGE.

4	ADD	mus_musculus	5	Fields: Gene ID,Region,Start,End,Strand,Label,Description								
5	BOR		6									
6	IN	gorilla_gorilla	7	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1
7	ADD	gorilla_gorilla	8	Fields: Gene ID								

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 для выдачи могли запрашиваться альтернативные имена (алиасы). Следующий оператор BOR указывает, что с оператора 6 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 гориллы. Независимо от результата проверки, программа переходит к оператору 7, 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 установленными. Оператор 7 ADD добавляет в строку выходного файла еще одно поле с идентификатором первого найденного оператором IN подходящего гена гориллы. Если проверка была неуспешной и подходящего гена не найдено, то добавляется пустое поле. Разумеется, выбор нужных полей для копирования в выходной файл остаётся за пользователем.

8	IN	nomascus_leucogenys	9	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1
9	ADD	nomascus_leucogenys	10	Fields: Gene ID								
10	IN	pan_paniscus	11	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1
11	ADD	pan_paniscus	12	Fields: Gene ID								
12	IN	pan_troglodytes	13	2	2000000	7	10.000	0.001	1.0e-005	0.400	1	1

```

13 ADD pan_troglodytes 14 Fields: Gene ID
14 IN pongo_abelii 15 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
15 ADD pongo_abelii 16 Fields: Gene ID
16 EOR 17 NO 4

```

Вышеприведённый фрагмент содержит ещё четыре аналогичных проверки с другими геномами. После каждой проверки в выходную строку добавляется найденный ген из проверяемого вида или пустое поле, если такого гена не нашлось. Последняя строка во фрагменте содержит оператор 16 EOR (5.5.5), завершающий эту серию проверок. В нём указаны две метки 17 и NO (вторая виртуальная, она не транслируется в номер реального оператора), а также значение параметра $n=4$, которое означает, что если 4 и более (т.е. все, за исключением, быть может, одной) проверок были успешными, то *этот терм* предиката считается выполненным и программа переходит к проверке следующего термина. В противном случае *весь* предикат не выполнен и программа переходит к проверке очередного гена мыши.

```

17 BOR 18
18 IN cercocebus_atys 19 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
19 ADD cercocebus_atys 20 Fields: Gene ID
20 IN chlorocebus_sabaeus 21 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
21 ADD chlorocebus_sabaeus 22 Fields: Gene ID
22 IN colobus_angolensis_palliatus 23 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
23 ADD colobus_angolensis_palliatus 24 Fields: Gene ID
24 IN macaca_fascicularis 25 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
25 ADD macaca_fascicularis 26 Fields: Gene ID
26 IN macaca_mulatta 27 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
27 ADD macaca_mulatta 28 Fields: Gene ID
28 IN macaca_nemestrina 29 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
29 ADD macaca_nemestrina 30 Fields: Gene ID
30 IN mandrillus_leucophaeus 31 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
31 ADD mandrillus_leucophaeus 32 Fields: Gene ID
32 IN papio_anubis 33 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
33 ADD papio_anubis 34 Fields: Gene ID
34 IN ptilocolobus_tephrosceles 35 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
35 ADD ptilocolobus_tephrosceles 36 Fields: Gene ID
36 IN rhinopithecus_bieti 37 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
37 ADD rhinopithecus_bieti 38 Fields: Gene ID
38 IN rhinopithecus_roxellana 39 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
39 ADD rhinopithecus_roxellana 40 Fields: Gene ID
40 IN theropithecus_gelada 41 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
41 ADD theropithecus_gelada 42 Fields: Gene ID
42 EOR 43 NO 2

```

Приведённый выше фрагмент содержит терм предиката, аналогичный предыдущему, но в нём проверка идёт против другого множества, состоящего из 12 видов мартышковых, и используется другое значение параметра $n=2$.

```

43 IN bos_taurus 44 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
44 ADD bos_taurus 45 Fields: Gene ID
45 IN canis_lupus_familiaris 46 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
46 ADD canis_lupus_familiaris 47 Fields: Gene ID
47 IN delphinapterus_leucas 48 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
48 ADD delphinapterus_leucas 49 Fields: Gene ID
49 IN equus_caballus 50 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
50 ADD equus_caballus 51 Fields: Gene ID
51 IN heterocephalus_glaber_female 52 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
52 ADD heterocephalus_glaber_female 53 Fields: Gene ID
53 IN loxodonta_africana 54 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
54 ADD loxodonta_africana 55 Fields: Gene ID
55 IN lynx_canadensis 56 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
56 ADD lynx_canadensis 57 Fields: Gene ID
57 IN monodelphis_domestica 58 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
58 ADD monodelphis_domestica 59 Fields: Gene ID
59 IN myotis_lucifugus 60 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
60 ADD myotis_lucifugus 61 Fields: Gene ID
61 IN panthera_leo 62 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1

```



```

62 ADD panthera_leo 63 Fields: Gene ID
63 IN physeter_catodon 64 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
64 ADD physeter_catodon 65 Fields: Gene ID
65 IN rhinolophus_ferrumequinum 66 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
66 ADD rhinolophus_ferrumequinum 67 Fields: Gene ID
67 IN ursus_maritimus 68 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
68 ADD ursus_maritimus 69 Fields: Gene ID
69 IN aotus_nancymaae 70 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
70 ADD aotus_nancymaae 71 Fields: Gene ID
71 IN callithrix_jacchus 72 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
72 ADD callithrix_jacchus 73 Fields: Gene ID
73 IN carlito_syrichta 74 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
74 ADD carlito_syrichta 75 Fields: Gene ID
75 IN cebus_capucinus 76 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
76 ADD cebus_capucinus 77 Fields: Gene ID
77 IN microcebus_murinus 78 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
78 ADD microcebus_murinus 79 Fields: Gene ID
79 IN otolemur_garnettii 80 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
80 ADD otolemur_garnettii 81 Fields: Gene ID
81 IN prolemur_simus 82 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
82 ADD prolemur_simus 83 Fields: Gene ID
83 IN propithecus_coquereli 84 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
84 ADD propithecus_coquereli 85 Fields: Gene ID
85 IN saimiri_boliviensis_boliviensis 86 2 2000000 7 10.000 0.001 1.0e-005 0.400 1 1
86 ADD saimiri_boliviensis_boliviensis YES Fields: Gene ID

```

Приведённая выше заключительная группа операторов предиката не содержит никаких условных переходов и, следовательно, не влияет на отбор генов мыши. Она служит для выяснения того, присутствует или отсутствует текущий ген мыши у других рассматриваемых видов. В частности, это позволяет, не проводя новый счёт, на базе полученных данных изучить другие варианты условия отбора генов мыши, с участием других видов. Если вычисление предиката дошло до этого фрагмента, значит предикат выполнен. Его последний оператор 86 ADD завершается переходом на метку YES и сформированная ранее строка будет записана в выходной файл. Иначе, если где-то ранее произошёл переход на метку NO, предикат не выполняется и строка не будет записана. В обоих случаях программа переходит к проверке следующего гена X опорного вида, начиная снова с оператора 0.

Следующий фрагмент консольного протокола перечисляет виды, гены которых будут представлены в строке выходного файла (слева направо). Если предикат формирует строки с переменным форматом, показан вариант с наибольшим числом генов. Данный фрагмент не выдаётся, если в командной строке была указана опция -q2 или -q3.

```

Maximum 40 gene(s) per line: mus_musculus gorilla_gorilla nomascus_leucogenys
pan_paniscus pan_troglodytes pongo_abelii cercocebus_atys chlorocebus_sabaeus
colobus_angolensis_palliatus macaca_fascicularis macaca_mulatta macaca_nemestrina
mandrillus_leucophaeus papio_anubis piliocolobus_tephrosceles rhinopithecus_bieti
rhinopithecus_roxellana theropithecus_gelada bos_taurus canis_lupus_familiaris
delphinapterus_leucas equus_caballus heterocephalus_glaber_female loxodonta_africana
lynx_canadensis monodelphis_domestica myotis_lucifugus panthera_leo physeter_catodon
rhinolophus_ferrumequinum ursus_maritimus aotus_nancymaae callithrix_jacchus
carlito_syrichta cebus_capucinus microcebus_murinus otolemur_garnettii prolemur_simus
propithecus_coquereli saimiri_boliviensis_boliviensis

```

В следующих двух строках приводятся сведения о прочитанной таблице генов опорного вида (не выдаются, если указана опция -q3). 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 (35), которые здесь

для единообразия именуется контигами. Далее, c1 – число контигов, содержащих только один ген (3), c2 – число контигов, содержащих ровно два гена (2).

```
Reading gene info file genes\mus_musculus.tsv ...
Lines read: 112608 genes: 22468 proteins: 67158 contigs: 35 c1: 3 c2: 2
```

Далее по мере чтения таблиц генов выводится аналогичная информация об остальных видах (если не указана опция -q3):

```
Reading gene info file genes\homo_sapiens.tsv ...
Lines read: 176163 genes: 24337 proteins: 117680 contigs: 1220 c1: 936 c2: 69
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\nomascus_leucogenys.tsv ...
Lines read: 47561 genes: 20794 proteins: 40527 contigs: 336 c1: 225 c2: 49
Reading gene info file genes\pan_paniscus.tsv ...
Lines read: 52282 genes: 21210 proteins: 43232 contigs: 243 c1: 167 c2: 22
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\cercocebus_atys.tsv ...
Lines read: 53602 genes: 20926 proteins: 46067 contigs: 549 c1: 129 c2: 26
Reading gene info file genes\chlorocebus_sabaeus.tsv ...
Lines read: 28077 genes: 19165 proteins: 19255 contigs: 119 c1: 65 c2: 9
Reading gene info file genes\colobus_angolensis_palliatu...
Lines read: 47036 genes: 20617 proteins: 40399 contigs: 730 c1: 113 c2: 41
Reading gene info file genes\macaca_fascicularis.tsv ...
Lines read: 65967 genes: 22504 proteins: 49919 contigs: 147 c1: 58 c2: 27
Reading gene info file genes\macaca_mulatta.tsv ...
Lines read: 62443 genes: 21761 proteins: 48770 contigs: 175 c1: 108 c2: 27
Reading gene info file genes\macaca_nemestrina.tsv ...
Lines read: 54045 genes: 21060 proteins: 46238 contigs: 449 c1: 105 c2: 19
Reading gene info file genes\mandrillus_leucophaeus.tsv ...
Lines read: 47767 genes: 20841 proteins: 40903 contigs: 1538 c1: 313 c2: 153
Reading gene info file genes\papio_anubis.tsv ...
Lines read: 53012 genes: 21647 proteins: 45181 contigs: 418 c1: 275 c2: 70
Reading gene info file genes\piliocolobus_tephrosceles.tsv ...
Lines read: 54302 genes: 24588 proteins: 41245 contigs: 3537 c1: 2848 c2: 190
Reading gene info file genes\rhinopithecus_bieti.tsv ...
Lines read: 52671 genes: 20966 proteins: 43730 contigs: 2510 c1: 1007 c2: 248
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\theropithecus_gelada.tsv ...
Lines read: 43834 genes: 22515 proteins: 36976 contigs: 414 c1: 337 c2: 39
Reading gene info file genes\bos_taurus.tsv ...
Lines read: 43267 genes: 21880 proteins: 37538 contigs: 128 c1: 66 c2: 13
Reading gene info file genes\canis_lupus_familiaris.tsv ...
Lines read: 55790 genes: 20257 proteins: 45094 contigs: 221 c1: 134 c2: 33
Reading gene info file genes\delphinapterus_leucas.tsv ...
Lines read: 33898 genes: 19091 proteins: 30992 contigs: 262 c1: 124 c2: 11
Reading gene info file genes\equus_caballus.tsv ...
Lines read: 54352 genes: 20955 proteins: 44934 contigs: 299 c1: 178 c2: 48
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\loxodonta_africana.tsv ...
Lines read: 28849 genes: 20033 proteins: 25635 contigs: 583 c1: 228 c2: 81
Reading gene info file genes\lynx_canadensis.tsv ...
Lines read: 38883 genes: 19131 proteins: 35180 contigs: 36 c1: 5 c2: 5
Reading gene info file genes\monodelphis_domestica.tsv ...
Lines read: 50160 genes: 21384 proteins: 36557 contigs: 425 c1: 258 c2: 59
Reading gene info file genes\myotis_lucifugus.tsv ...
Lines read: 26842 genes: 19728 proteins: 20719 contigs: 2418 c1: 1169 c2: 305
```

```

Reading gene info file genes\panthera_leo.tsv ...
Lines read: 34373 genes: 19550 proteins: 31178 contigs: 459 c1: 262 c2: 58
Reading gene info file genes\physeter_catodon.tsv ...
Lines read: 35311 genes: 19717 proteins: 32206 contigs: 1279 c1: 676 c2: 243
Reading gene info file genes\rhinolophus_ferrumequinum.tsv ...
Lines read: 36358 genes: 19533 proteins: 33717 contigs: 60 c1: 13 c2: 6
Reading gene info file genes\ursus_maritimus.tsv ...
Lines read: 39675 genes: 18724 proteins: 34052 contigs: 463 c1: 124 c2: 24
Reading gene info file genes\aotus_nancymaae.tsv ...
Lines read: 51062 genes: 20412 proteins: 42510 contigs: 898 c1: 268 c2: 66
Reading gene info file genes\callithrix_jacchus.tsv ...
Lines read: 61545 genes: 22615 proteins: 43000 contigs: 545 c1: 483 c2: 13
Reading gene info file genes\carlito_syrichta.tsv ...
Lines read: 38316 genes: 18398 proteins: 31791 contigs: 5933 c1: 2658 c2: 1086
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\microcebus_murinus.tsv ...
Lines read: 45983 genes: 18895 proteins: 38078 contigs: 76 c1: 32 c2: 2
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\prolemur_simus.tsv ...
Lines read: 43803 genes: 20354 proteins: 38056 contigs: 2231 c1: 888 c2: 205
Reading gene info file genes\propithecus_coquereli.tsv ...
Lines read: 37914 genes: 17925 proteins: 32202 contigs: 810 c1: 185 c2: 52
Reading gene info file genes\saimiri_boliviensis_boliviensis.tsv ...
Lines read: 48819 genes: 19380 proteins: 40695 contigs: 320 c1: 45 c2: 16

```

Эти сведения не выдаются, если в командной строке была указана опция `-q3` (но выдаются с опциями `-q0...-q2`). То же относится и к следующим строкам, сообщающим о прочитанных таблицах ортологов и, если требуется, паралогов для опорного и других видов согласно конкретному предикату. В данном случае – только одна строка о чтении таблицы ортологов для мыши. В конце каждой строки этой части протокола приводится число упорядоченных пар ортологичных (или паралогичных) генов в указанной таблице. Описанная выдача характерна для выбранного режима работы H (данные из Ensembl). В других режимах работы программы выдаётся информация о других прочитанных входных файлах.

```
Reading orthologs from orthologs\mus_musculus.tsv ... 747042
```

Затем выдаются две строки, показанные ниже. В первой упоминается имя опорного вида, число последовательностей верхнего уровня (для краткости «контигов»), из которых состоит его геном (35), и общее число генов (22468). 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: 35 contigs, 22468 genes.
With several genes: 32 contigs, 22465 genes.
```

После этого начинается собственно поиск генов, во время которого программа `lossgainRSL` выдаёт на консоль сообщения вида

```
c5 x130
```

где первое число указывает порядковый номер проверяемого контига опорного 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: 230
```

```
54s: 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.

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.

Если для программы был задан режим `X` (5.1), то для каждого найденного гена опорного 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 `J` 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/lossgainrsl/example.zip>). Выходной файл допускает удобный импорт в электронную таблицу Excel, что упрощает последующую обработку и анализ полученных результатов. Пример импорта выходного файла представлен на листе 'Result' файла <http://lab6.iitp.ru/ru/lossgainrsl/dataformat.xlsx>.

7 Использование других источников входных данных

7.1 Добавление геномов, отсутствующих в Ensembl

База данных Ensembl уже в нынешней редакции содержит полные геномы более 300 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 добавить новый геном опорного вида, который должен браться из Ensembl, по крайней мере, в нынешней реализации. Если опорный 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 неструктурированном 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 код этой геномной последовательности верхнего уровня по сводному каталогу (accession), причём без номера версии. Его можно безопасно использовать в качестве 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).

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 универсальный алгоритм для автоматического формирования данного поля, но при ручном заполнении таблицы генов трудностей обычно 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 иначе: использовать готовый FASTA-файл всех белковых последовательностей нового вида, заменяя в нём заголовки на присвоенные идентификаторы белков или, наоборот, брать для таблицы генов в качестве идентификатора белка содержимое первого поля заголовка в существующем файле.

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 имеет место для многих видов из GenBank. В составе эталонной сборки обычно есть файл feature_table, из которого несложно получить нужную таблицу генов. Например, для тихоокеанского моржа *Odobenus rosmarus divergens* этот файл расположен по адресу: ftp://ftp.ncbi.nlm.nih.gov/genomes/all/annotation_releases/9708/101/GCF_000321225.1_Oros_1.0/GCF_000321225.1_Oros_1.0_feature_table.txt.gz, и для получения таблицы генов можно применить скрипт MakeGeneTable.py (автор – О.А. Зверков) из состава пакета gbinput, который можно загрузить со страницы программы, <http://lab6.iitp.ru/ru/lossgainrsl/>. Скрипт необходимо запускать из той папки, куда загружен вышеуказанный файл; там же будет сформирована и таблица генов. Для других мест расположения файлов или для других видов соответствующие изменения необходимо внести в строки 4 и 5 скрипта.

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 опорного 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 опорного 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 в составе пакета `gbinput` 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 опорного и нового видов, а также всех пар белков опорного 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 `per` для каждого вида, например, ftp://ftp.ensembl.org/pub/current_fasta/mus_musculus/pep/ для мыши. Для геномов из RefSeq и GenBank аналогичные файлы имеются в каталоге соответствующей сборки, например, `GCF_000321225.1_Oros_1.0_protein.faa.gz` для моржа (полный адрес каталога см. 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 опорного и нового видов, указанные в секции `[species]` файла конфигурации (0). Подчеркнём, что идентификаторы белков обоих видов в этих матрицах сходства белков должны *в точности* совпадать с указанными в таблицах генов опорного и нового видов. Это существенно, потому что последовательности белков в формате FASTA часто имеют идентификаторы с номером версии, в отличие от используемых программой таблиц 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 `addspecies64`. Помимо опций, обязательные аргументы имеют следующий смысл: `source` – имя добавляемого нового вида, `target` – имя существующего вида (опорного 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):

```
[add]
EVALUE      1e-10
ORTHOLOG    0.35
PARALOG     0.35
```

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 выравнивания каждого белка с самим собой. Если вычисленный вес больше или равен указанному порогу, 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 *n* 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 M (в дополнение к перечисленным в Табл. 4), например, в форме строки
M matrix*-*.tsv
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.
- U То же, что и -M, 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 паралога 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 O 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), 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), описанные выше. Эти матрицы должны находиться в каталоге, указанном для типа данных A в разделе [data] файла конфигурации (5.2), 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) или таблица кластеров (4.5), а таблицы генов (4.1) берутся по-прежнему из Ensembl.

Наконец, БД Ensembl может не использоваться вообще, а в качестве входных данных будет браться таблица ортологичных групп (4.5), составленная на основе данных OrthoDB. Также можно составить таблицы генов (4.1) и матрицы сходства белков (4.4) целиком на материале NCBI, как описано в .

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.

7.3 Использование входных данных исключительно из баз NCBI

Представленная в NCBI информация об ортологии генов не претендует на полноту и носит фрагментарный характер, а во многих случаях перенаправляет запрос на OrthoDB. Поэтому самым подходящим вариантом входных данных для программы lossgainRSL являются таблицы генов (4.1) и матрицы сходства белков (4.4). Для более точного определения истинных гомологов следует сохранить в матрице для каждого гена X в виде R возможно большее число его лучших гомологов X' в *каждом* виде A , например, до 10-15, если все они достаточно достоверны, скажем, с отсечкой по порогу E-value < 0.001, что позволит в дальнейшем выбрать подходящее значение параметра k – наибольшего ранга хита (Табл. 5). А также использовать при поиске гомологов обратную проверку (5.5.2). Ниже описывается типичная последовательность действий для подготовки входных данных.

Подготовка структуры директорий

Создать рабочую папку (например, с именем work) и скопировать туда нужную версию программы lossgainRSL и других упоминаемых ниже утилит и скриптов, собранных в единый пакет gbinput, который можно загрузить с главной страницы <http://lab6.iitp.ru/ru/lossgainrsl/>. Создать в этой папке будущий файл конфигурации с именем config.ini (см. главу 5) и еще один текстовый файл species.txt, первая строка которого содержит заголовки полей “filename”, “protein” и “feature”, разделённые символом табуляции, а содержание последующих строк описывается ниже. Затем создать в рабочей папке ряд подпапок с помощью следующих командами:

```
mkdir blastdb
mkdir dup
mkdir feature
mkdir fasta
mkdir gene
mkdir hits
mkdir homolog
mkdir protein
```

Выбор видов и получение исходных файлов NCBI

Выбор зависит от проводимого исследования, но следует предпочесть хорошо собранные геномы, содержащиеся в RefSeq. Для каждого генома необходимо выбрать уникальный идентификатор вида, который не должен содержать пробелов. Обычно за основу берется латинское двучленное название вида, например, pan_troglodytes. Необходимо также записать каталожный код (accession) выбранного генома, в данном случае GCF_002880755.1, с помощью которого можно выйти в соответствующий раздел FTP-сайта NCBI:

<ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/002/880/755/>

Этот каталог обычно содержит только символическую ссылку на папку с данными генома, из которой необходимо загрузить два упакованных файла, содержащие аминокислотные последовательности всех известных белков (протеом) и таблицу характеристик генома (feature_table), в данном случае с именами:

```
GCF_002880755.1_Clint_PTRv2_protein.faa.gz
GCF_002880755.1_Clint_PTRv2_feature_table.txt.gz
```

Для каждого вида первый файл распаковать в папку `protein`, второй в папку `feature`. В очередную строку файла `species.txt` внести три поля, разделяя их символом табуляции: первое поле содержит идентификатор вида, второе – имя распакованного файла протеома, третье – имя распакованного файла характеристик. Пример готового файла `species.txt` после его импорта в Excel содержится на листе ‘Species_list’ файла <http://lab6.iitp.ru/ru/lossgainrs1/dataformat.xlsx>. Одновременно в раздел `[species]` файла `config.ini` (5.4) необходимо заносить присвоенные идентификаторы видов.

Построение таблиц генов

Запустить из рабочей папки `work` скрипт `gb2en.py` (автор – О.А. Зверков), который является усовершенствованием скрипта из раздела 7.1.2 для одновременной обработки таблиц характеристик всех геномов, перечисленных в файле `species.txt`, а не одного указанного. Для этого на компьютере должен быть установлен интерпретатор python версии 3.10 или выше. В результате в папке `gene` будут получены файлы в формате TSV с именами `species.tsv`, где `species` – идентификатор вида; каждый из этих файлов соответствует спецификации раздела 4.1.

Предварительная обработка протеомов

Цель предобработки – устранение дубликатов белков и их индексирование, что достигается командой:

```
fastaprep -r 0 -u 0 -w 1 species.txt species.txt
```

В результате в папке `fasta` будут получены файлы в формате FASTA с именами `species.fas`, а в папке `dup` – рабочие таблицы с именами `species.tsv`, где `species` – идентификатор вида. Размер каждого файла FASTA обычно меньше, чем соответствующего протеома в папке `protein`, поскольку отсутствует часть белков и неиспользуемая информация в заголовках.

Нахождение белковых хитов

Возможны два варианта действий:

Вариант 1

С помощью программы ‘makeblastdb’ из пакета BLAST v2.12.0+ или более поздней версии построить отдельную базу данных для каждого вида. Например, для указанного выше вида использовать команду:

```
makeblastdb -in fasta/pan_troglodytes.fas -dbtype prot -out blastdb/pan_troglodytes
```

Для других видов в команде необходимо соответственно изменить идентификатор вида в аргументах `-in` и `-out`.

Далее для каждой пары видов с идентификаторами `species1` и `species2`, не учитывая порядок и в том числе с одинаковыми идентификаторами, выполнить команду:

```
blastp -outfmt "6 qseqid sseqid score evalue" -evalue 0.1 -max_target_seqs 1000  
-max_hsps 1 -use_sw_tback -query fasta/species1.fas -db blastdb/species2 -out  
hits/species1-species2.tsv
```

Этот вариант реалистичен, если число рассматриваемых видов не превышает 25-30, т.е. в общей сложности надо составить, запланировать и выполнить до 1000 команд.

Примечания:

1. Счёт по вышеприведённой команде может продолжаться десятки часов, поэтому рекомендуется использовать суперкомпьютер для параллельного исполнения сразу многих таких команд для разных пар видов. При использовании многоядерных CPU также можно ускорить счет добавлением в команду параметра `-num_threads n`, где n целое число, большее 1.
2. Если число видов превышает 10-12, рекомендуется заранее создать в папке `hits` подпапки, поименованные идентификаторами всех видов, и последний аргумент командной строки выше указывать в форме `hits/species1/species1-species2.tsv`, чтобы работа файловой системы не замедлялась из-за большого числа файлов в папке `hits`.
3. В качестве более быстрой альтернативы пакету BLAST+ можно использовать пакет DIAMOND. Его запуск необходимо проводить с параметрами, обеспечивающими получение в папке `hits` файлов того же формата и содержания, что выдаёт `blastp`.

Вариант 2

Собрать все файлы FASTA в один, используя в Windows команду

```
copy /a fasta/*.faa /b fasta/common.fasta
```

или в linux команду

```
cat fasta/*.faa >fasta/common.fasta
```

Построить единую базу для всех рассматриваемых видов командой:

```
makeblastdb -in fasta/common.fasta -dbtype prot -out blastdb/common
```

Затем для каждого вида с идентификатором `species` выполнить команду:

```
blastp -outfmt "6 qseqid sseqid score evalue" -evalue 0.1 -max_target_seqs 10000  
-max_hsp 1 -use_sw_tback -query fasta/species.fas -db blastdb/common -out  
hits/species.tsv
```

Этот вариант удобнее применять при большом числе рассматриваемых видов, когда число их пар становится неприемлемо большим. Здесь число запускаемых команд равно числу видов, но каждая из них будет выполняться существенно дольше, чем в первом варианте (время увеличится кратно числу видов). Также, если число видов превышает 30-50, рекомендуется увеличить значение параметра `-max_target_seqs`, например, до 30000-50000.

Независимо от выбранного варианта, для нахождения белковых хитов с помощью `blastp` рекомендуется использовать суперкомпьютер. При использовании программы `diamond` это можно сделать и на достаточно мощном ПК или сервере.

Построение матриц сходства генов

Эти матрицы, являющиеся частным случаем матриц сходства белков (4.4) и способствующие ускорению работы, строятся с помощью программы `posthit`. Это написанная на C++ утилита командной строки Windows или linux (только версии 64 бит) с поддержкой MPI. Запуск в однопроцессорном режиме производится командой:

```
posthit -b 1 -c 0 -i hits/*.tsv -n 10 -r 2 -w 0 list.txt
```

В результате в папке `homolog` будут построены отдельные матрицы для каждого вида, в каждой записи которых содержатся все шесть полей из Табл. 2, где в поле 3 стоят значения `raw score`. Для получения матриц весов в команде выше необходимо указать опцию `-w 1`.