# Обработка чтений

- FastQC
- Trimmomatic
- Jellyfish

Контроль качества чтений является важным этапом предшествующим сборке генома: перед запуском сборки следует убедиться, что риды имеют адекватное качество, т.е. не содержат большого количества ошибок, и последовательности адаптеров, лигированных в ходе приготовления библиотек, не помешают сборке. Оценку качества ридов мы проведем при помощи FastQC, а удаление адаптеров программой Trimmomatic. Дополнительно, мы посчитаем распределение кратности к-меров в наших данных при помощи программы Jellyfish, чтобы иметь представление о размере и покрытии генома до выполнения сборки.

Для работы с данными создайте папку Project_Intoshia на локальном хранилище /local. В этой папке мы будем работать, и хранить все результаты геномного проекта. Для хранения чтений организуйте подпапку Illumina_reads в /Project_Intoshia

```
cd  /local
mkdir  Project_Intoshia
cd  /local/Project_Intoshia
mkdir  Illumina_reads
```

Скопируйте файлы ридов Intoshia из папки на сервере: /P/NGSPhylo2014/Intoshia_reads в созданную директорию на локальном диске: /local/Project_Intoshia/Illumina_reads

```
cp  /P/NGSPhylo2014/Intoshia_reads/*.gz  /local/Project_Intoshia/Illumina_reads
```

Файлы ридов содержатся в архиве. Для работы с ними их необходимо извлечь из архива

```
cd  /local/Project_Intoshia/Illumina_reads
gunzip *.gz
```

Парно-концевые чтения записаны в формате FASTQ и разделены по файлам в соответствии с направлением чтения вставки: прямые риды (R1) и обратные риды (R2). Также файлы прямых и обратных ридов разбиты на части по 4 миллиона ридов в каждом (001, 002, и т.д.) Для работы имеет смысл свести все части с прямыми чтениями в один файл и также сделать с обратными

```
cat  Intoshia_ATGTCA_L008_R1_001.fastq  Intoshia_ATGTCA_L008_R1_002.fastq \
> Intoshia_ATGTCA_L008_R1.fastq
cat  Intoshia_ATGTCA_L008_R2_001.fastq  Intoshia_ATGTCA_L008_R2_002.fastq \
> Intoshia_ATGTCA_L008_R2.fastq
```

Исходные файлы, т.е. разбитые на части, теперь нам больше не нужны и их можно удалить

```
rm  *001.fastq
rm  *002.fastq
```

Теперь в директории /Illumina_reads есть два файла с ридами: в одном прямые риды, а в другом обратные. Риды записанные в таком формате принимают большинство ассемблеров. Прежде чем

приступить к сборке мы проведем оценку качества чтений и очистим риды от адаптеров.

Для оценки качества воспользуемся программой FastQC. В отличие от большинства используемых нами программ, у FastQC есть собственный графический интерфейс, который можно вызвать командой

```
fastqc
```

В меню программы выберите оба файла с ридами и сохраните результат после завершения анализа. Убедившись, что при приготовлении библиотеки были использованы адаптеры TruSeq, воспользуемся программой Trimmomatic для удаления адаптеров из ридов. Программе необходимо указать файл с адаптерами, который также нужно скопировать в директорию с ридами

```
cp  /P/NGSPhylo2014/Supplementary_files/TruSeq3-PE.ed.fa  \
/local/Project_Intoshia/Illumina_reads
```

Команда для запуска Trimmomatic достаточно громоздкая, поэтому ее имеет смысл записать в виде скрипта. Для этого создайте новый файл trim.sh

```
cat  >  trim.sh
```

Запишите в созданном файле:

```
#!/bin/sh
TrimmomaticPE  -threads 2  -phred33  \
Intoshia_ATGTCA_L008_R1.fastq  Intoshia_ATGTCA_L008_R2.fastq  \
Intoshia_R1.trimmed.fastq  Intoshia_R1.single.fastq  \
Intoshia_R2.trimmed.fastq  Intoshia_R2.single.fastq  \
ILLUMINACLIP:TruSeq3-PE.ed.fa:2:30:10:1:true  MINLEN:55
```

Сохранить и выйти из режима редактирования: Ctrl+D

В скрипте мы вызываем модуль программы Trimmomatic для работы с парно-концевыми чтениями, выделяя программе 2 ядра (-threads 2) и указывая, что качество чтений записано в формате phred33. Далее последовательно записаны названия двух входных файлов с прямыми и обратными чтениями и 4-х выходных файлов: по файлу для парных и одиночных прямых ридов и также для обратных. Последняя строка скрипта содержит название файла с адаптерами и параметры для тримминга ридов: ILLUMINACLIP:<fastaWithAdaptersEtc>:<seed mismatches>:<palindrome clip threshold>:<simple clip threshold>:<minAdapterLength>:<keepBothReads>

Чтобы trim.sh можно было использовать как скрипт необходимо добавить разрешение на исполнение файла. Для добавления разрешений на запуск используйте следующую команду

```
chmod  a+x  trim.sh
```

После запуска созданного скрипта `./trim.sh` в директории должны появиться файлы с обработанными ридами. Нам нужны файлы, которые мы назвали Intoshia_R1.trimmed.fastq и Intoshia_R2.trimmed.fastq – они содержат прямые и обратные риды очищенные от адаптеров.

Дополнительным средством анализа ридов, которое может помочь определить размер генома до

проведения сборки, является распределение кратности к-меров в ридах. Для подсчета кратности к-меров мы воспользуемся программой Jellyfish. Перед запуском программы объединим прямые и обратные риды в один файл и переместим их в новую директорию /Jellyfish под /Project_Intoshia

```
cat Intoshia_R1.trimmed.fastq Intoshia_R2.trimmed.fastq > Intoshia.merged.fastq
cd /local/Project_Intoshia
mkdir Jellyfish
mv /local/Project_Intoshia/Illumina_reads/Intoshia.merged.fastq /local/Project_Intoshia/Jellyfish
cd /local/Project_Intoshia/Jellyfish
```

В директории /local/Project_Intoshia/Jellyfish создадим новый скрипт для запуска программы Jellyfish

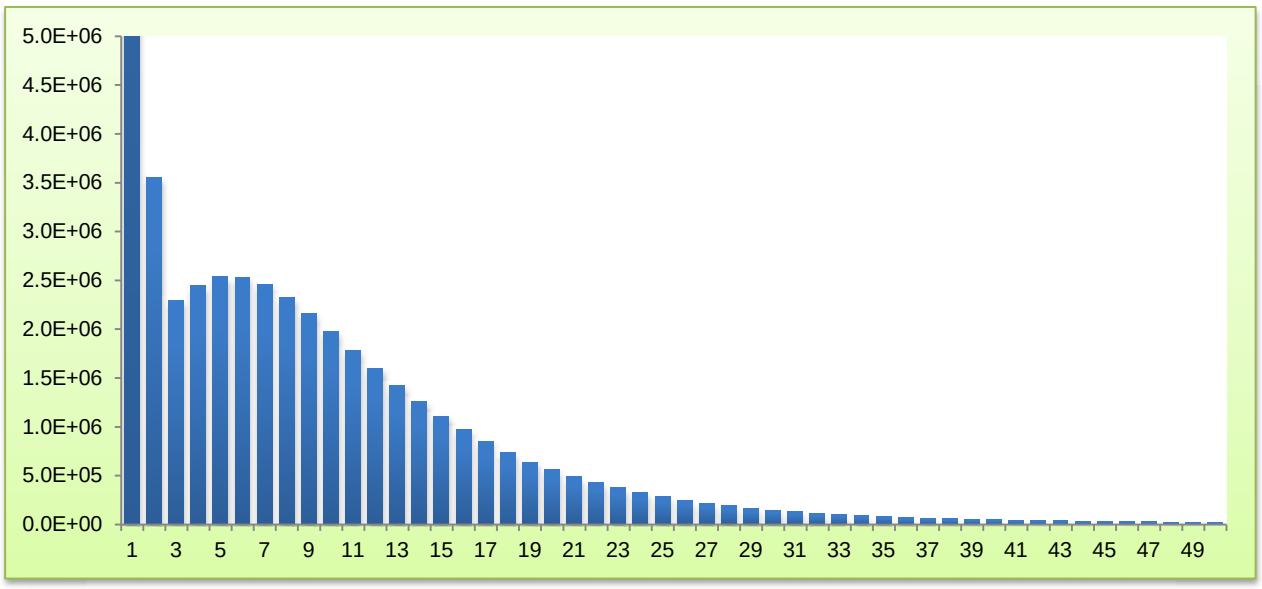
```
cat > jellyfish.sh
```

```
#!/bin/sh
jellyfish count -m 21 -o Intoshia -c 5 -C -s 100M -t 2 Intoshia.merged.fastq
jellyfish stats Intoshia_0 > stats.txt
jellyfish histo Intoshia_0 > histo.txt
```

В скрипте мы выделяем программе 2 ядра (-t 2) и просим посчитать распределение кратности 21-меров (-m 21) используя обе цепи ДНК (-C), с размером хэша в 100 миллионов элементов (-s 100M), выделяя минимум 5 бит на хранение значения кратности каждого к-мера (-c 5). Результат работы будет записан в файлах stats.txt и histo.txt

```
chmod a+x jellyfish.sh
./jellyfish.sh
```

В histo.txt содержится информация о распределении кратности к-меров, которое для этих данных выглядит следующим образом:




Первый пик распределения – к-меры встречающиеся всего один раз. Эти к-меры, вместе со всеми к-мерами слева от первого минимума на гистограмме, преимущественно связаны с ошибками. Вычитая ошибочные к-меры ~31M из числа Distinct к-меров в stats.txt ~65M, получаем приблизительную оценку размера генома - 34 Mb. Второй пик распределения приходится на значение кратности около 5-6, что в

пересчете на нуклеотидное покрытие дает значение около 6-7. Формула для пересчета покрытия:

$$C_n = C_k \cdot \frac{r}{(r-k+1)}$$

$C_n$ – покрытие в чтениях на нуклеотид
$C_k$ – покрытие в чтениях на к-мер
$r$ – длина чтения
$k$ – размер к-мера

# Сборка генома

- Velvet
- Ray
- SPAdes
- QUAST
- CEGMA

Для *de novo* сборки генома *Intoshia* мы опробуем три программы основанные на построении графа де Брёйна: Velvet, Ray и SPAdes. В первом приближении Velvet и Ray не имеют принципиальных отличий помимо реализации алгоритма сборки и разных подходов к поддержке параллелизации. SPAdes отличается от этих программ более принципиально благодаря возможности комбинировать сборки выполненные с разными значениями размера к-мера. Оценку качества сборок генома мы проведем при помощи программы QUAST, и, выбрав наиболее перспективную сборку, выполним оценку полноты сборки генома программой CEGMA.

Работу с ассемблерами мы будем выполнять при помощи СК Ломоносов. Скомпилированные программы для сборки находятся в архиве compiled_programs.tar.gz на сервере компьютерного класса в папке Supplementary_files. Загрузите данный архив в вашу директорию на СК

```
scp  -o identityFile=/path_to_key/key_name  \
/P/NGSPhylo2014/Supplementary_files/compiled_programs.tar.gz \
username@lomonosov.parallel.ru:~/
```

Затем соединитесь с сервером СК в терминале и выполните вход

```
ssh  -o identityFile=/path_to_key/key_name  username@lomonosov.parallel.ru
```

Разархивируйте compiled_programs.tar.gz в корень вашей директории на СК

```
tar  -xvzf  compiled_programs.tar.gz
```

В директории должно появиться 6 новых папок: velvet, ray, spades, raxml, phylobayes, mrbayes. Три папки содержат файлы необходимые для запуска программ сборки, три других содержат исполняемые файлы для программ реконструкции деревьев, которые нам потребуются позже для филогенетического анализа.

Перед запуском программ на СК Ломоносов необходимо подключить библиотеку, при помощи которой производилась их компиляция

```
module add openmpi/1.5.5-gcc
```

## Velvet

Программа Velvet состоит из двух модулей – velveth и velvetg. Первый модуль создает базу данных, записывая информацию о к-мерах в ридах, второй модуль использует эту базу для конструкции графа, выполняя сам процесс сборки.

Для запуска Velvet создайте скрипт velvet.sh в директории /velvet, который последовательно вызовет оба модуля и запишет результат сборки и промежуточные файлы в папку Intoshia-velvet-k33

```
cd  ~/velvet
cat  >  velvet.sh
```

```
#!/bin/sh
export OMP_NUM_THREADS=7
./velveth Intoshia-velvet-k33  33  -fastq  -separate  -shortPaired \
Intoshia_R1.trimmed.fastq  Intoshia_R2.trimmed.fastq
./velvetg Intoshia-velvet-k33  -exp_cov auto  -cov_cutoff auto
```

Параметр OMP_NUM_THREADS устанавливает количество дополнительных потоков для выполнения задачи (на узле 8 ядер). Для velveth, помимо названий файлов с ридами, мы указываем рабочую директорию «Intoshia-velvet-k33», размер к-мера (33), формат входных файлов (-fastq) и тип данных (-shortPaired = парные риды); флаг separate сообщает velveth, что прямые и обратные риды записаны в разных файлах. Модулю velvetg мы передаем рабочую директорию и устанавливаем автоматическое определение среднего покрытия (-exp_cov auto) и порога покрытия для сокращения графа (-cov_cutoff auto).

```
chmod  a+x  velvet.sh
```

Размер к-мера и порог покрытия - два наиболее важных параметра, позволяющие регулировать процесс и результат сборки. Оптимальный размер к-мера часто подбирается эмпирически путем проведения нескольких сборок и сравнения их качества. Порог покрытия служит для отсечения ошибочных к-меров и элементов графа, и его выбор позволяет собирать последовательности выделяющиеся покрытием из остальных данных (например, митохондриальные геномы). При автоматическом выборе порога Velvet выставит его равным половине среднего покрытия, что не всегда совпадает с первым минимумом на гистограмме кратности к-меров.

Запуск программ на СК Ломоносов осуществляется через директиву sbatch из под директории ~/_scratch.

Скопируйте папку velvet в ~/_scratch, также не забыв скопировать в директорию /_scratch/velvet сами файлы ридов. Запуск скрипта выполните следующей командой, запросив один узел (-N1) на разделе test

```
cp  -r  ~/velvet  ~/_scratch
cd  ~/_scratch/velvet
sbatch -p test -N1  ./velvet.sh
```

Проверить состояние задачи можно командой squeue `squeue -u username`

В подпапке Intoshia-velvet-k33 нас интересует конечный результат сборки - последовательности в формате FastA (contigs.fa), также могут оказаться полезными лог файл и stats.txt, содержащий информацию о длине и покрытии каждой последовательности в виде таблицы. Скопируйте только эти три файла в соответствующую папку на основном дисковом разделе и загрузите их с СК на локальный компьютер в новую папку Assemblies

```
cd  ~/velvet
mkdir Intoshia-velvet-k33
cp  ~/_scratch/velvet/Intoshia-velvet-k33/contigs.fa  ~/velvet/Intoshia-velvet-k33/
cp  ~/_scratch/velvet/Intoshia-velvet-k33/Log  ~/velvet/Intoshia-velvet-k33/
cp  ~/_scratch/velvet/Intoshia-velvet-k33/stats.txt  ~/velvet/Intoshia-velvet-k33/
```

```
cd  /local/Project_Intoshia
mkdir  Assemblies
scp  -o identityFile=/path_to_key/key_name -r  \
username@lomonosov.parallel.ru:~/velvet/Intoshia-velvet-k33 \
/local/Project_Intoshia/Assemblies/
```

## Ray

Для пользователей наиболее существенным различием между программами Velvet и Ray является способ параллелизации процесса сборки. В то время как Velvet поддерживает параллелизацию только в пределах одного узла (стандарт OpenMP), Ray поддерживает распределение процессов между узлами (стандарт MPI). Поддержка MPI дает очевидное преимущество Ray на вычислительных кластерах с большим количеством узлов, но малым объемом оперативной памяти на узлах.

Для запуска Ray создадим скрипт ray.sh в директории /ray. В отличие от Velvet, программа Ray не разделена на модули и запускается одной командой

```
cd  ~/ray
cat  >  ray.sh
```

```
#!/bin/sh
./Ray  -k 33  -p Intoshia_R1.trimmed.fastq  Intoshia_R2.trimmed.fastq  -o Intoshia-ray-k33
```

Для Ray мы задаем размер к-мера (33) и указываем файлы с ридами (названия записываются после директивы -p, означающей что риды парные и записаны в отдельных файлах). После директивы -o идет название рабочей директории.

```
chmod  a+x  ray.sh
```

Как и Velvet, Ray позволяет выбирать размер к-мера и настраивать порог покрытия (-use-minimum-seed-coverage). При автоматическом определении порога (по умолчанию) Ray ищет первый минимум на гистограмме кратности к-меров и устанавливает порог равный значению кратности для минимума. Полный список команд для Ray можно найти здесь: http://denovoassembler.sourceforge.net/manual.html

Запустите Ray из ~/_scratch/ray, скопировав папку из основного раздела и переместив в эту директорию риды из папки velvet. В команде sbatch добавьте директиву ompi, необходимую для запуска MPI приложений, и вместо одного узла (-N1) укажите 16 процессоров (-n16)

```
cp  -r  ~/ray  ~/_scratch
mv  ~/_scratch/velvet/*.fastq  ~/_scratch/ray
cd  ~/_scratch/ray
sbatch -p test -n16 ompi ./ray.sh
```

В процессе работы Ray будет выводить подробную статистику по данным в файлы под /Intoshia-ray-k33. Основной результат сборки будет записан в два FastA файла - Contigs.fasta и Scaffolds.fasta; также представляют интерес файл со статистикой по результатам сборки (OutputNumbers.txt), распределение кратности и статистика по к-мерам (CoverageDistribution.txt, CoverageDistributionAnalysis.txt), фактически повторяющие анализ Jellyfish, а также информация о рассчитанной длине вставки (LibraryStatistics.txt).

## SPAdes

Большинство ассемблеров основанных на построении графа де Брёйна, включая Velvet и Ray, рассчитаны на относительно равномерное покрытие генома чтениями. Многочисленные перепады в уровне покрытия приводят к сильной фрагментированности сборок, что становится ощутимым недостатком при работе с некоторыми образцами, например образцами получаемыми с применением полно-геномной амплификации. Один из немногих ассемблеров, которые менее чувствительны к неравномерному покрытию и позволяют достаточно эффективно собирать подобные данные, это SPAdes. Ассемблер SPAdes собирает контиги с разным уровнем покрытия, выполняя последовательные сборки с повышающимся значением размера к-мера: граф из к-меров малого размера позволяет собирать участки с низким покрытием, граф из более длинных к-меров повышает специфичность сборки, разрешая повторяющиеся участки генома.

Для запуска SPAdes создайте следующий скрипт в директории /spades

```
cd  ~/spades
cat  >  spades.sh
```

```
#!/bin/sh
export PATH=$PATH:~/spades/SPAdes-3.0.0-Linux/bin
spades.py -t 8  -m 12 -k 33,55 --only-assembler \
--pe1-1 Intoshia_R1.trimmed.fastq --pe1-2 Intoshia_R2.trimmed.fastq \
-o Intoshia-spades
```

Переменная «путь» указывает на директорию с исполняемыми файлами программы. Для работы SPAdes мы выделяем 8 потоков (-t 8) и 12 гигабайт памяти (-m 12). После директивы -k, через запятую идет перечисление размеров к-мера, которые будут использованы для сборки (33,55); возможность использовать несколько значений для размера к-мера в рамках одной сборки является ключевой чертой данного ассемблера. Флагом --only-assembler мы отключаем процедуру исправления чтений, чтобы ассемблер сразу переходил к этапу сборки. Файлы с парными ридами даются после --pe#-#, где первый # это индекс библиотеки (у нас всего одна), а второй # перечисляет файлы с прямыми и обратными ридами. Промежуточные файлы и результаты будут записаны в /Intoshia-spades

```
chmod  a+x  spades.sh
```

Сохранив скрипт, скопируйте папку /spades из основного раздела в ~/_scratch и переместите в нее файлы с ридами из ~/_scratch/ray. SPAdes можно распареллелить только в пределах одного узла, поэтому в команде sbatch не нужна директива ompi и выделить мы можем только один узел (-N1)

```
cp  -r  ~/spades  ~/_scratch
mv  ~/_scratch/ray/*.fastq  ~/_scratch/spades
cd  ~/_scratch/spades
sbatch -p test -N1  ./spades.sh
```

В папке Intoshia-spades по мере работы ассемблера будет записываться лог и результаты промежуточных сборок; конечный результат будет записан в файлах contigs.fasta и scaffolds.fasta

Для оценки сборок мы воспользуемся программой QUAST. Эта программа рассчитывает и выводит ряд статистических параметров сборки, таких как суммарная длина, число контигов и N50, а также позволяет вывести данные нескольких сборок для удобного сравнения. Сравнивать мы будем заранее подготовленные сборки *Intoshia*, выполненные Velvet, Ray и SPAdes, и записанные в папку Assemblies на сервере.

Скопируйте архив со сборками из /P/NGSPhylo2014/Assemblies/ в локальную директорию /Assemblies и разархивируйте его

```
cp  /P/NGSPhylo2014/Assemblies/assemblies.tar.gz \
/local/Project_Intoshia/Assemblies
cd  /local/Project_Intoshia/Assemblies
tar  -xvzf  assemblies.tar.gz
```

В директорию будут записаны семь новых папок: по три сборки Velvet и Ray с разными значениями размера к-мера (21,33,55) и одна сборка SPAdes.

Для запуска QUAST создайте скрипт quast.sh

```
cat  >  quast.sh
```

```
#!/bin/sh
quast  --threads 2  -t 200  \
./velvet-k21/contigs.fa ./velvet-k33/contigs.fa ./velvet-k55/contigs.fa \
./ray-k21/Scaffolds.fasta ./ray-k33/Scaffolds.fasta ./ray-k55/Scaffolds.fasta \
./spades/scaffolds.fasta \
--labels velvet-k21,velvet-k33,velvet-k55,ray-k21,ray-k33,ray-k55,spades
```

Мы выделяем quast для работы 2 потока (--threads 2) и указываем исключить из сводки статистики последовательности короче 200 нуклеотидов (-t 200); далее мы перечисляем файлы со сборкой, и после команды --labels, через запятую даем название каждой сборке для отображения в результатах (порядок файлов = порядок названий)

```
chmod  a+x  quast.sh
./quast.sh
```

Результат работы QUAST будет записан в директорию /quast_results/results_#, где # соответствует дате и времени запуска программы. Файлы report записаны в нескольких форматах, и содержат сводку статистики для сборок в виде таблицы и графиков. График «cumulative length» отображает суммарную длину последовательностей (ордината), ранжированных по длине от наибольшей до наименьшей (абсцисса).

Дополнительный метод оценки черновых сборок эукариотических геномов предлагает пайплайн CEGMA (Core Eukaryotic Genes Mapping Approach). CEGMA использует набор из ~460 высококонсервативных генов для предсказания их экзон-интронных структур в сборках на основе сходства. Помимо автоматической генерации аннотаций этих генов в исследуемом геноме, которые нам понадобятся для дальнейшей работы, данный анализ сообщает статистику о находках этих генов, что дает некоторую оценку полноты сборки.

Перед запуском программы создайте новую папку cegma под директорией /Assemblies и скопируйте в нее файл с контигами из наиболее целостной сборки (contigs.fasta из /spades)

```
cd  /local/Project_Intoshia/Assemblies
mkdir  cegma
cp  /local/Project_Intoshia/Assemblies/spades/contigs.fasta  \
/local/Project_Intoshia/Assemblies/cegma
```

Для запуска CEGMA дайте следующую команду

```
cd  /local/Project_Intoshia/Assemblies/cegma
cegma  --genome contigs.fasta  -threads 2
```

Мы передаем программе cegma файл с контигами (--genome contigs.fasta) и выделяем для работы 2 потока (-threads 2)

В процессе работы CEGMA будет обращаться к программам BLAST, HMMER, geneid и genewise (эти

программы необходимы для работы пайплайна CEGMA и устанавливаются отдельно), создавая аннотации найденных в contigs.fasta генов из встроенного набора. Результат будет записан в эту же директорию в несколько файлов output.cegma. Для нас наибольший интерес представляют три файла:

output.cegma.fa – аминокислотные последовательности транскриптов предсказанных генов
output.cegma.gff – аннотации в формате GFF
output.completeness_report – сводка статистики по предсказаниям для 248 ультраконсервативных генов

## Предсказание генов

- Augustus
- Scipio

Этот этап геномного проекта мы будем выполнять при помощи Augustus – программы предназначенной для *ab initio* предсказаний генов в эукариотических геномах. Augustus основан на принципе скрытой марковской модели, и требует тренировочный набор аннотированных генов для формирования модели подходящей для данного организма. В качестве тренировочного набора мы возьмем выход программы CEGMA, предсказавшей набор из ~250 генов *Intoshia* основываясь на сходстве с консервативными ортологами из других организмов. Для преобразования генов предсказанных программой CEGMA в тренировочный набор пригодный для подачи Augustus, мы воспользуемся программой Scipio.

В папке Project_Intoshia организуйте новую папку Augustus и в ней подпапку training

```
cd  /local/Project_Intoshia
mkdir  Augustus
cd  /local/Project_Intoshia/Augustus
mkdir  training
```

Для составления тренировочного набора из выхода CEGMA необходимо преобразовать предсказанные аминокислотные последовательности из output.cegma.fa в аннотированные нуклеотидные последовательности генов в формате GenBank. Это можно сделать при помощи программы Scipio и нескольких скриптов поставляемых вместе с Augustus. Для работы со Scipio создайте новую директорию «scipio» под /training и скопируйте в нее output.cegma.fa из /cegma и contigs.fasta из /spades

```
cd  /local/Project_Intoshia/Augustus/training
mkdir  scipio
cp  /local/Project_Intoshia/Assemblies/cegma/output.cegma.fa  \
/local/Project_Intoshia/Augustus/training/scipio
cp  /local/Project_Intoshia/Assemblies/spades/contigs.fasta  \
/local/Project_Intoshia/Augustus/training/scipio
```

В директории /scipio создайте новый скрипт для запуска Scipio

```
cd  /local/Project_Intoshia/Augustus/training/scipio
cat  >  scipio.sh
```

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/scripts
scipio.1.4.1.pl  --min_score 1  --min_identity 100  --min_coverage 95  \
--single_target_hits  --blat_output=prot.vs.genome.psl  contigs.fasta  \
output.cegma.fa  >  scipio.yaml
cat  scipio.yaml  |  yaml2log.1.4.pl >  scipio.log
cat  scipio.yaml  |  yaml2gff.1.4.pl  >  scipio.scipiogff
scipiogff2gff.pl  --in=scipio.scipiogff  --out=scipio.gff
gff2gbSmallDNA.pl  scipio.gff  contigs.fasta  100  training_set.gb
```

Первой командой скрипта мы вызываем программу Scipio и запрашиваем провести картирование аминокислотных последовательностей в output.cegma.fa на контиги в contigs.fasta, требуя полного соответствия (--min_score 1  --min_identity 100) при выравнивании как минимум 95% длины аминокислотной последовательности (--min_coverage 95). Флаг --single_target_hits предотвращает картирование аминокислотной последовательности на более чем один контиг. Следующие команды вызывают скрипты из Augustus, которые преобразуют выход программы Scipio, в конечном итоге записывая результат в training_set.gb. Цифра в последней строке (100) регулирует размер фланкирующих участков гена (в нуклеотидах).

```
chmod  a+x  scipio.sh
./scipio.sh
```

Скрипт сгенерирует файл training_set.gb, в котором должны находиться аннотированные нуклеотидные последовательности генов из CEGMA фланкированные 100 нуклеотидами в формате GenBank.

Перед запуском Augustus скопируйте папку config из /usr/lib/augustus в /local/Project_Intoshia/Augustus. В директории /config/species будет создана новая модель для генома *Intoshia*

```
cp  -R  /usr/lib/augustus/config  /local/Project_Intoshia/Augustus
```

Для создания новой модели «intoshia» напишите следующий скрипт в директории /training

```
cd  /local/Project_Intoshia/Augustus/training
cat  >  new.sh
```

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/scripts
export AUGUSTUS_CONFIG_PATH=/local/Project_Intoshia/Augustus/config
new_species.pl --species=intoshia
```

```
chmod  a+x  new.sh
./new.sh
```

В директории /local/Project_Intoshia/Augustus/config/species должна появиться новая папка intoshia, в

которой будут сохранены параметры модели для предсказания генов. Модифицируйте одну из строк в файле intoshia_parameters.cfg заменив false на true (необходимо для тренировочных наборов созданных по данной схеме):

stopCodonExcludedFromCDS false ⟶ stopCodonExcludedFromCDS true

```
cd /local/Project_Intoshia/Augustus/config/species/intoshia
pico intoshia_parameters.cfg
```

Выход из режима редактирования: Ctrl+X

ИЛИ

```
sed -i 's/stopCodonExcludedFromCDS false/stopCodonExcludedFromCDS true/' \
intoshia_parameters.cfg
```

Теперь все готово для обучения Augustus. Запустите тренировку при помощи следующего скрипта

```
cd /local/Project_Intoshia/Augustus/training
cat > train.sh
```

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/bin
export PATH=$PATH:/usr/lib/augustus/scripts
export AUGUSTUS_CONFIG_PATH=/local/Project_Intoshia/Augustus/config
etraining --species=intoshia --stopCodonExcludedFromCDS=true training_set.gb
augustus --species=intoshia training_set.gb | tee training_test_initial.out
```

В скрипте мы сначала вызываем программу для обучения, передавая ей обучающий набор training_set.gb, а затем проводим процедуру предсказания генов для этого же набора*, записывая результат в training_test_initial.out

```
chmod a+x train.sh
./train.sh
```

В конце файла training_test_initial.out будет записана сводка о чувствительности и специфичности предсказаний для генов из тренировочного набора, которая будет выглядеть примерно таким образом:

| | Sn | Sp |
|---|---|---|
| nucleotide level | 0.869 | 0.982 |
| exon level | 0.504 | 0.63 |
| gene level | 0.128 | 0.128 |

Улучшению этих показателей часто помогает оптимизация метапараметров модели Augustus, которую можно запустить при помощи следующего скрипта

```
cat > optimize.sh
```

* в идеале полагается разделять набор генов на обучающий и проверочный, чтобы не выполнять проверку на том же наборе что был использован для обучения

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/bin
export PATH=$PATH:/usr/lib/augustus/scripts
export AUGUSTUS_CONFIG_PATH=/local/Project_Intoshia/Augustus/config
optimize_augustus.pl  --cpus=2  --species=intoshia  training_set.gb
```

```
chmod  a+x  optimize.sh
./optimize.sh
```

После оптимизации запустите повторную тренировку Augustus

```
cat  >  retrain.sh
```

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/bin
export PATH=$PATH:/usr/lib/augustus/scripts
export AUGUSTUS_CONFIG_PATH=/local/Project_Intoshia/Augustus/config
etraining  --species=intoshia  --stopCodonExcludedFromCDS=true  training_set.gb
augustus  --species=intoshia  training_set.gb  |  tee  training_test.out
```

```
chmod  a+x  retrain.sh
./retrain.sh
```

Сводка о чувствительности и специфичности предсказаний после оптимизации, записанная в конце файла training_test.out, будет выглядеть примерно так:

| | **Sn** | **Sp** |
|---|---|---|
| nucleotide level | 0.974 | 0.975 |
| exon level | 0.834 | 0.817 |
| gene level | 0.488 | 0.476 |

Этих показателей должно быть достаточно для целей филогенетического анализа. Для запуска процедуры предсказания генов для всей сборки скопируйте scaffolds.fasta из /spades в /Augustus и создайте следующий скрипт

```
cp  /local/Project_Intoshia/Assemblies/spades/scaffolds.fasta  \
/local/Project_Intoshia/Augustus
cd  /local/Project_Intoshia/Augustus
cat  >  augustus.sh
```

```
#!/bin/sh
export PATH=$PATH:/usr/lib/augustus/bin
export PATH=$PATH:/usr/lib/augustus/scripts
export AUGUSTUS_CONFIG_PATH=/local/Project_Intoshia/Augustus/config
augustus --species=intoshia scaffolds.fasta > augustus.gtf
getAnnoFasta.pl --seqfile scaffolds.fasta --codingseq=on augustus.gtf
gtf2gff.pl < augustus.gtf --gff3 --out=augustus.gff
```

Первая команда запускает Augustus для scaffolds.fasta, записывая результат в augustus.gtf. Следующие команды вызывают скрипты, преобразующие результаты предсказаний в более универсальный формат (FastA и GFF файлы).

```
chmod a+x augustus.sh
./augustus.sh
```

После завершения задачи, в директории /Augustus должны появиться следующие файлы:

augustus.aa – аминокислотные последовательности транскриптов предсказанных генов
augustus.codingseq – нуклеотидные последовательности транскриптов предсказанных генов
augustus.cdsexons – нуклеотидные последовательности экзонов
augustus.gff – предсказания в формате GFF
augustus.gtf – предсказания в формате GTF

## Кластеризация ортологов

- BLAST
- OrthoMCL
- MCL

Идентификация ортологичных генов в геномах исследуемых организмов необходима как для их функциональной аннотации, так и для филогенетического анализа. Подходы к идентификации ортологов опираются на поиске сходства между последовательностями, в частности на применение и обработку результатов BLAST. Один из подобных подходов реализован программой OrthoMCL. OrthoMCL использует результат all-vs-all BLAST всех генов из набора нескольких геномов, чтобы найти пары потенциальных ортологов и выделить их в группы ортологичных генов при помощи алгоритма кластеризации MCL (Markov Cluster Algorithm).

Начнем с подготовки файлов для all-vs-all BLAST. Организуйте в /Project_Intoshia новую папку

OrthoMCL и подпапку Proteomes

```
cd  /local/Project_Intoshia
mkdir  OrthoMCL
cd  /local/Project_Intoshia/OrthoMCL
mkdir  Proteomes
```

В директорию /OrthoMCL/Proteomes скопируйте архив proteomes.tar.gz из папки Proteomes на сервере и разархивируйте его

```
cp  /P/NGSPhylo2014/Proteomes/proteomes.tar.gz  /local/Project_Intoshia/OrthoMCL/Proteomes
cd  /local/Project_Intoshia/OrthoMCL/Proteomes
tar  -xvzf  proteomes.tar.gz
```

В папке должно появиться 20 .fa файлов – это предсказанные аминокислотные последовательности транскриптов из 18 геномов животных, включая *Intoshia*, и геномов двух ближайших одноклеточных родственников животных, записанные в формате FastA. Файл Intoshia.fa просто переименованный augustus.aa из результата Augustus, а остальные файлы собраны из геномных баз данных, преимущественно из Генбанка NCBI. Эти файлы были взяты из соответствующих баз данных *as is* и не проходили никакой дополнительной обработки. Как результат, названия генов не всегда соответствуют одной схеме. Для стандартизации названий, необходимой для следующих стадий работы, мы воспользуемся программой orthomclAdjustFasta, входящей в инструментарий OrthoMCL.

В папке /Proteomes создайте новый скрипт для запуска orthomclAdjustFasta

```
cd  /local/Project_Intoshia/OrthoMCL/Proteomes
cat  >  adjust.sh
```

```
#!/bin/sh
orthomclAdjustFasta  Adin  Adineta_vaga.fa  1
orthomclAdjustFasta  Amph  Amphimedon_queenslandica.fa  4
orthomclAdjustFasta  Aply  Aplysia_californica.fa  4
orthomclAdjustFasta  Asca  Ascaris_suum.fa  4
orthomclAdjustFasta  Capi  Capitella_teleta.fa  4
orthomclAdjustFasta  Caps  Capsaspora_owczarzaki.fa  1
orthomclAdjustFasta  Dros  Drosophila_melanogaster.fa  4
orthomclAdjustFasta  Echi  Echinococcus_granulosus.fa  4
orthomclAdjustFasta  Homo  Homo_sapiens.fa  4
orthomclAdjustFasta  Hydr  Hydra_vulgaris.fa  4
orthomclAdjustFasta  Hyps  Hypsibius_dujardini.fa  1
orthomclAdjustFasta  Into  Intoshia.fa  1
orthomclAdjustFasta  Ixod  Ixodes_scapularis.fa  4
orthomclAdjustFasta  Mnem  Mnemiopsis_leidyi.fa  1
orthomclAdjustFasta  Mono  Monosiga_brevicollis.fa  1
orthomclAdjustFasta  Nema  Nematostella_vectensis.fa  3
orthomclAdjustFasta  Osca  Oscarella_carmela.fa  1
orthomclAdjustFasta  Schi  Schistosoma_mansoni.fa  4
orthomclAdjustFasta  Stro  Strongylocentrotus_purpuratus.fa  4
orthomclAdjustFasta  Tric  Trichoplax_adhaerens.fa  3
```

В скрипте мы последовательно запускаем orthomclAdjustFasta для каждого .fa файла и цифрой

указываем поле в названии генов, которое будет фигурировать как уникальный идентификатор последовательности в файлах со стандартизированными названиями. Поля в названиях генов считаются после знака > и разделены пробелом или знаком | , пример:

| старый формат | № поля | новый формат |
|---|---|---|
| >CAOG_00001T0 \| CAOG_00001 \| Capsaspora owczarzaki ATCC 30864 hypothetical protein (1182 aa)<br>MPVWFPH.....<br>>CAOG_00002T0 \| CAOG_00002 \| Capsaspora owczarzaki ATCC 30864 hypothetical protein (1268 aa)<br>MDPNEAV..... | 1 | >Caps\|CAOG_00001T0<br>MPVWFPH.....<br>>Caps\|CAOG_00002T0<br>MDPNEAV..... |
| >gi\|304570837\|ref\|XP_002404260.2\| plekhH1, putative [Ixodes scapularis]<br>LPAETSV.....<br>>gi\|251831105\|ref\|XP_002405670.2\| hypothetical protein IscW_ISCW003139 [Ixodes scapularis]<br>MLNCGHN..... | 4 | >Ixod\|XP_002404260.2<br>LPAETSV.....<br>>Ixod\|XP_002405670.2<br>MLNCGHN..... |

```
chmod  a+x  adjust.sh
./adjust.sh
```

В папке /Proteomes должны появиться 20 .fasta файлов со стандартизированными названиями генов. Переместите их в новую подпапку compliantFasta

```
mkdir  compliantFasta
mv  *.fasta  ./compliantFasta
```

Перед запуском BLAST все файлы нужно свести в один. Эта операция выполняется программой orthomclFilterFasta, которая дополнительно проводит фильтрацию последовательностей на предмет длины и количества стоп кодонов. Запустите программу следующей командой

```
orthomclFilterFasta  compliantFasta  20 10
```

orthomclFilterFasta сведет все последовательности из /compliantFasta в единый файл goodProteins.fasta и отбросит белки короче 20 аминокислот и белки содержащие стоп кодоны в более чем 10% своей длины.

Файл goodProteins.fasta можно использовать для all-vs-all BLAST. Для этого создайте папку BLAST под /Project_Intoshia и скопируйте в нее goodProteins.fasta

```
cd  /local/Project_Intoshia
mkdir  BLAST
cp  /local/Project_Intoshia/OrthoMCL/Proteomes/goodProteins.fasta  \
/local/Project_Intoshia/BLAST
```

Для проведения all-vs-all BLAST создайте базу данных из goodProteins.fasta при помощи makeblastdb (из пакета программ standalone BLAST)

```
cd  /local/Project_Intoshia/BLAST
makeblastdb  -in goodProteins.fasta  -dbtype prot  -out goodProteins
```

Запустить BLAST можно следующей командой

```
blastp  -db goodProteins  -query goodProteins.fasta  -out blastresult.tsv  \
-evalue  1e-50  -max_target_seqs  10000  -outfmt 6  -num_threads 2
```

Рекомендуемое значение E-value составляет 1e-05, однако в целях экономии времени на стадии

обработки результатов мы поставили значение 1e-50, что существенно сократит количество детектируемых ортологов в этом анализе. Данная задача может идти от нескольких часов до нескольких дней в зависимости от размера базы данных и степени параллелизации процесса, поэтому мы не будем ждать выполнения этой команды и для следующей стадии работы возьмем заранее посчитанный результат BLAST. Готовый результат можно взять из папки на сервере /P/NGSPhylo2014/BLAST_result

```
cp  /P/NGSPhylo2014/BLAST_result/blastresult.tar.gz   /local/Project_Intoshia/OrthoMCL
```

Разархивируйте blastresult.tar.gz в директорию /OrthoMCL

```
cd  /local/Project_Intoshia/OrthoMCL
tar  -xvzf  blastresult.tar.gz
```

Перед загрузкой результата BLAST в базу OrthoMCL необходимо преобразовать blastresult.tsv при помощи программы orthomclBlastParser, которая переформатирует результат и посчитает процент сходства для каждого хита

```
orthomclBlastParser  blastresult.tsv  ./Proteomes/compliantFasta/  >>  similarSequences.txt
```

Результат будет записан в similarSequences.txt и в таком виде он готов к загрузке в базу OrthoMCL.

Для работы основного модуля OrthoMCL нам потребуется создать и сконфигурировать базу данных MySQL. Для этого вызовите консоль MySQL следующей командой

```
mysql  -u  root  -p
```

пароль:  bad-password

В консоли MySQL создайте новую базу данных «orthomcl»

```
mysql> create database orthomcl;
```

Далее создайте нового пользователя «orthomcl» и дайте ему права на использование базы orthomcl

```
mysql> grant  select,insert,update,delete,create view,create,index,drop  \
on orthomcl.*  to  orthomcl@localhost;
```

Установите пароль для пользователя orthomcl (для удобства установим пароль повторяющий имя пользователя: «orthomcl»)

```
mysql> set password for orthomcl@localhost = password('orthomcl');
```

Выход из консоли MySQL:

```
mysql> exit
```

После создания новой базы данных ее нужно настроить для работы с данными OrthoMCL. Для этого скопируйте шаблон orthomcl.config.template, поставляемый вместе с OrthoMCL, из папки Supplementary_files на сервере в папку OrthoMCL на локальном диске

```
cp  /P/NGSPhylo2014/Supplementary_files/orthomcl.config.template  \
/local/Project_Intoshia/OrthoMCL/orthomcl.config
```

Измените строки dbConnectString, dbLogin и dbPassword в файле orthomcl.config следующим образом

```
dbVendor=mysql
dbConnectString=dbi:mysql:orthomcl:mysql_local_infile=1:localhost
dbLogin=orthomcl
dbPassword=orthomcl
similarSequencesTable=similarSequences
orthologTable=Ortholog
inParalogTable=InParalog
coOrthologTable=CoOrtholog
interTaxonMatchView=InterTaxonMatch
percentMatchCutoff=50
evalueExponentCutoff=-5
oracleIndexTblSpc=NONE
```

Сохраните изменения в orthomcl.config и вызовите программу orthomclInstallSchema для этого файла

```
cd  /local/Project_Intoshia/OrthoMCL
orthomclInstallSchema  orthomcl.config  mysql.log
```

Теперь все должно быть готово для запуска программ основного модуля OrthoMCL. Эти программы организованы по принципу конвейера, т.е. результат работы одной программы сразу передается следующей, поэтому их имеет смысл свести в единый скрипт

```
cat  >  orthopipe.sh
```

```
#!/bin/sh
orthomclLoadBlast  orthomcl.config  similarSequences.txt
orthomclPairs  orthomcl.config  pairs.log  cleanup=yes
orthomclDumpPairsFiles  orthomcl.config
mcl ./mclInput  --abc  -I 2.0  -o groups.txt
orthomclMclToGroups  OG2.0_  10000  <  groups.txt  >  named_groups.txt
```

Программа orthomclLoadBlast загружает преобразованный результат BLAST (similarSequences.txt) в MySQL базу orthomcl (для этого требуется примерно в 5-6 раз больше дискового места, чем занимает сам файл similarSequences.txt). Далее программа orthomclPairs выделяет в загруженных данных пары потенциальных межгеномных ортологов, внутригеномных паралогов и межгеномных ко-ортологов, и рассчитывает вес каждой пары из значений сходства. Эта наиболее емкая по времени стадия пайплайна, которая также требует дополнительного места на диске, так как в процессе работы создаются промежуточные таблицы в базе orthomcl (команда cleanup=yes удаляет промежуточные результаты). После завершения работы orthomclPairs, программа orthomclDumpPairsFiles выводит пары потенциальных ортологов, паралогов и ко-ортологов из базы данных в файлы под директорией /pairs, и создает файл mclInput для подачи алгоритму кластеризации. Кластеризация выполняется отдельной программой MCL, которая не включена в пакет программ OrthoMCL. Для программы кластеризации мы указываем формат входного файла (--abc: 3 поля разделенные табуляцией) и устанавливаем параметр инфляции равный двум (-I 2.0). Параметр инфляции контролирует процесс кластеризации, отражая степень дробления на кластеры. Этот параметр можно варьировать от минимального значения >1; в большинстве работ фигурируют значения в диапазоне 1.2 - 1.7, в частности значение 1.5 было выбрано для базы данных OrthoMCL-DB (http://www.orthomcl.org). Наконец программа orthomclMclToGroups переписывает выход MCL (groups.txt) в (named_groups.txt), присваивая каждой группе номер с

префиксом OG2.0_ начиная счет от 10000

```
chmod  a+x  orthopipe.sh
./orthopipe.sh
```

На этом завершается работа с OrthoMCL. В файле named_groups.txt записан результат работы в виде названия кластера и идентификаторов последовательностей в него входящих. Для более удобного представления результата воспользуйтесь скриптом mcl2table, который располагается в папке на сервере /P/NGSPhylo2014/Supplementary_files

```
cp  /P/NGSPhylo2014/Supplementary_files/mcl2table  \
/local/Project_Intoshia/OrthoMCL/mcl2table
```

Запустите скрипт указав название файла с результатом пайплайна OrthoMCL

```
./mcl2table  -in named_groups.txt
```

Скрипт создаст файл named_groups.txt.table, в котором результат будет представлен в виде таблицы. Содержимое файла можно скопировать в любое средство работы с таблицами (MS Excel, Libre Calc, и пр.) В ячейках таблицы записано количество генов, которое попадает в данный кластер (ряд) в данном геноме (колонка), и дополнительным столбцом идет счет геномов с ненулевым количеством генов в данном кластере.

После завершения работы OrthoMCL, база данных orthomcl автоматически не удаляется, поэтому ее нужно сбросить вручную. Для этого снова вызовите консоль MySQL

```
mysql  -u  root  -p
```

пароль:  bad-password

В консоли дайте команду на сброс базы данных orthomcl

```
mysql> drop database orthomcl;
mysql> exit
```

В следующий раз для запуска OrthoMCL будет достаточно только создать базу данных orthomcl и применить схему orthomcl.config; пользователь orthomcl и права доступа будут сохранены.

# Филогенетический анализ

- MAFFT
- trimAl
- SCaFoS
- Tree-Puzzle
- RAxML
- Phylobayes
- MrBayes

В результате кластеризации ортологичных генов, выполненной при помощи OrthoMCL, у нас появилась возможность выбрать ряд кластеров для филогенетического анализа согласно формальному критерию отбора, минуя трудозатратную стадию ручного поиска и проверки ортологов по выравниваниям. Выбрав кластеры при помощи полученной ранее таблицы, мы воспользуемся программой MAFFT для составления выравниваний ортологов, а затем конкатенируем выравнивания в единую матрицу при помощи программ SCaFoS и Tree-Puzzle. Выравнивания будут предварительно обработаны программой trimAl для удаления сомнительных участков выравнивания. Филогенетический анализ для конкатенированного выравнивания мы проведем на вычислительном кластере, используя три из наиболее популярных программ для реконструкции деревьев: RAxML, MrBayes и Phylobayes.

Полностью автоматизированный подход к поиску ортологов сопряжен с некоторым количеством ошибок, однако мы попытаемся митигировать их вклад в наш анализ применив строгий критерий к отбору кластеров. Для филогенетического анализа мы возьмем только те группы, где 90% геномов имеют всего один ген. Данный критерий легко формализовать в таблице при помощи функции COUNTIF, используя, например, LibreOffice Calc: в качестве диапазона (range) выбираем все геномы в данной строке, а в качестве критерия ставим 1. Функция для первой смысловой строки выглядит так =COUNTIF(B2:U2,1), для второй =COUNTIF(B3:U3,1), и т.д. Скопировав функцию во все строки и ранжировав столбец с функциями от наибольшего значения к наименьшему, в первых ~80 строках таблицы получаем кластеры со значением ≥18, т.е. кластеры соответствующие нашему критерию. Скопируйте названия этих кластеров (в первом столбце) в новый файл selected_OG.txt под директорией /OrthoMCL/selected_OG

```
cd  /local/Project_Intoshia/OrthoMCL
mkdir  selected_OG
cd  /local/Project_Intoshia/OrthoMCL/selected_OG
cat  >  selected_OG.txt
```

В эту же директорию скопируйте файл named_groups.txt из /OrthoMCL и скрипт orthogroups из папки Supplementary_files на сервере

```
cp  /local/Project_Intoshia/OrthoMCL/named_groups.txt  \
/local/Project_Intoshia/OrthoMCL/selected_OG
cp  /P/NGSPhylo2014/Supplementary_files/orthogroups \
/local/Project_Intoshia/OrthoMCL/selected_OG
```

Запустите скрипт orthogroups следующей командой

```
./orthogroups  -in named_groups.txt  -id selected_OG.txt
```

В директории должно появиться ~80 (по числу выбранных кластеров) .gi файлов с номером кластера в

названии и идентификаторами последовательностей из кластера внутри. Каждый из этих файлов будет использован для извлечения самих последовательностей из базы данных BLAST, которую нам нужно создать из файла goodProteins.fasta, сгенерированного в ходе работы с OrthoMCL. Предыдущая база данных, созданная из goodProteins.fasta, не пригодна для извлечения последовательностей из-за несовместимости названий с функцией -parse_seqids программы makeblastdb. Для того чтобы сделать названия совместимыми, сделайте копию goodProteins.fasta в файл goodProteins.parseable.fasta и воспользуйтесь командой sed чтобы заменить '|' на '-' и '>' на '>lcl|' в этом файле

```
cd  /local/Project_Intoshia/BLAST
cp  goodProteins.fasta  goodProteins.parseable.fasta
sed  -i  's/|/-/'  goodProteins.parseable.fasta
sed  -i  's/>/>lcl|/'  goodProteins.parseable.fasta
```

Теперь из goodProteins.parseable.fasta можно создать базу данных с флагом -parse_seqids, чтобы извлекать интересующие нас последовательности по идентификаторам

```
makeblastdb  -in goodProteins.parseable.fasta -dbtype prot \
-out goodProteins.parseable -parse_seqids
```

Последовательности извлекаются из базы данных BLAST программой blastdbcmd (входит в пакет программ standalone BLAST) для каждого кластера (файла с идентификаторами) следующей командой:

blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch OG_num.gi -out OG_num.fasta

где OG_num это название выбранного кластера.

Команда вызывается индивидуально для каждого файла, поэтому все вызовы имеет смысл объединить в скрипт blastdbcmd.sh под /selected_OG, а сам скрипт создать при помощи таблицы

| | | | | |
|---|---|---|---|---|
| blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch | OG_num1 | .gi -out | OG_num1 | .fa |
| blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch | OG_num2 | .gi -out | OG_num2 | .fa |
| blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch | OG_num3 | .gi -out | OG_num3 | .fa |
| blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch | OG_num4 | .gi -out | OG_num4 | .fa |
| … | … | … | … | … |

```
#!/bin/sh
blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch OG_num1.gi -out OG_num1.fa
blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch OG_num2.gi -out OG_num2.fa
blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch OG_num3.gi -out OG_num3.fa
blastdbcmd -db /local/BLAST/goodProteins.parseable -entry_batch OG_num4.gi -out OG_num4.fa
…
```

```
cd  /local/Project_Intoshia/OrthoMCL/selected_OG
chmod  a+x  blastdbcmd.sh
./blastdbcmd.sh
```

Данный скрипт создаст .fa файлы для каждого указанного .gi файла, куда будут записаны последовательности ортологов в FastA формате. Переместите все .fa файлы в новую папку alignments

под /OrthoMCL

```
cd  /local/Project_Intoshia/OrthoMCL
mkdir  alignments
cd  /local/Project_Intoshia/OrthoMCL/selected_OG
mv  *.fa  /local/Project_Intoshia/OrthoMCL/alignments
```

Прежде чем перейти к дальнейшей работе с этими файлами мы изменим формат названий последовательностей. Программы MAFFT и trimAl не чувствительны к формату названий, но для работы SCaFoS все последовательности должны соответствовать формату “taxon-name@sequence-id” и содержать только буквенно-цифровые знаки и знаки ‘.’ и ‘-’ (также допустимо использовать ‘_’ для разделения родовых и видовых названий). Для замены названий мы воспользуемся командой sed и следующим скриптом rename.sh

```
#!/bin/sh
sed -i 's/>lcl|Adin-/>Adineta_vaga@/' *.fa
sed -i 's/>lcl|Amph-/>Amphimedon_queenslandica@/' *.fa
sed -i 's/>lcl|Aply-/>Aplysia_californica@/' *.fa
sed -i 's/>lcl|Asca-/>Ascaris_suum@/' *.fa
sed -i 's/>lcl|Capi-/>Capitella_teleta@/' *.fa
sed -i 's/>lcl|Caps-/>Capsaspora_owczarzaki@/' *.fa
sed -i 's/>lcl|Dros-/>Drosophila_melanogaster@/' *.fa
sed -i 's/>lcl|Echi-/>Echinococcus_granulosus@/' *.fa
sed -i 's/>lcl|Homo-/>Homo_sapiens@/' *.fa
sed -i 's/>lcl|Hydr-/>Hydra_vulgaris@/' *.fa
sed -i 's/>lcl|Hyps-/>Hypsibius_dujardini@/' *.fa
sed -i 's/>lcl|Into-/>Intoshia@/' *.fa
sed -i 's/>lcl|Ixod-/>Ixodes_scapularis@/' *.fa
sed -i 's/>lcl|Mnem-/>Mnemiopsis_leidyi@/' *.fa
sed -i 's/>lcl|Mono-/>Monosiga_brevicollis@/' *.fa
sed -i 's/>lcl|Nema-/>Nematostella_vectensis@/' *.fa
sed -i 's/>lcl|Osca-/>Oscarella_carmela@/' *.fa
sed -i 's/>lcl|Schi-/>Schistosoma_mansoni@/' *.fa
sed -i 's/>lcl|Stro-/>Strongylocentrotus_purpuratus@/' *.fa
sed -i 's/>lcl|Tric-/>Trichoplax_adhaerens@/' *.fa
```

```
cd  /local/Project_Intoshia/OrthoMCL/alignments
chmod  a+x  rename.sh
./rename.sh
```

Все .fa файлы в /alignments теперь должны соответствовать формату SCaFoS.

Выравнивание для каждого .fa файла мы сделаем при помощи программы MAFFT. Команда дается для каждого файла индивидуально, поэтому, как и в случае с blastdbcmd, нам потребуется batch скрипт mafft.sh, который удобнее сделать при помощи таблицы

```
#!/bin/sh
mafft --thread 2 --localpair --maxiterate 1000 --ep 0.123 --legacygappenalty OG_num1.fa > OG_num1.aln.fa
mafft --thread 2 --localpair --maxiterate 1000 --ep 0.123 --legacygappenalty OG_num2.fa > OG_num2.aln.fa
mafft --thread 2 --localpair --maxiterate 1000 --ep 0.123 --legacygappenalty OG_num3.fa > OG_num3.aln.fa
mafft --thread 2 --localpair --maxiterate 1000 --ep 0.123 --legacygappenalty OG_num4.fa > OG_num4.aln.fa
                                                    …
```

Мы выделяем программе mafft два ядра (--thread 2) и запрашиваем выравнивание из расчета на присутствие центрального консервативного участка в белках, который может быть фланкирован вариабельными (--localpair --maxiterate 1000), и указываем использовать старые (не экспериментальные) параметры для расчета длин инделей в выравнивании (--ep 0.123 --legacygappenalty).

```
chmod a+x mafft.sh
./mafft.sh
```

Результаты работы записаны в .aln.fa файлах. Эти файлы содержат выравнивания полных последовательностей генов/белков, и в них, зачастую, помимо адекватно выравненных участков есть участки неподдающиеся выравниванию. Для исключения последних из филогенетического анализа мы воспользуемся программой trimAl. Программа trimAl удаляет невыравненные участки согласно настраиваемым параметрам сходства и допустимой доли пробелов. Для запуска trimAl также потребуется создать batch скрипт trimal.sh

```
#!/bin/sh
trimal -in OG_num1.aln.fa -out OG_num1.fasta -fasta -gt 0.9 -st 0.001 -w 3 -cons 10
trimal -in OG_num2.aln.fa -out OG_num2.fasta -fasta -gt 0.9 -st 0.001 -w 3 -cons 10
trimal -in OG_num3.aln.fa -out OG_num3.fasta -fasta -gt 0.9 -st 0.001 -w 3 -cons 10
trimal -in OG_num4.aln.fa -out OG_num4.fasta -fasta -gt 0.9 -st 0.001 -w 3 -cons 10
                                                    …
```

В команде мы указываем, что хотим удалить из выравнивания позиции с более чем 10% пробелов (-gt 0.9), а также позиции со значением сходства меньше 0.001 (-st 0.001: значение подобрано эмпирически), где сходство будет рассчитываться как среднее с учетом трех фланкирующих позиций (-w 3). Параметр -cons 10 запрещает удалять больше чем 90% позиций из выравнивания.

```
chmod a+x trimal.sh
./trimal.sh
```

Последняя фаза подготовки к филогенетическому анализу это конкатенация обработанных trimAl выравниваний в единую матрицу. Для этого переместите все обработанные файлы (*.fasta) из /alignments в новую папку scafos под /OrthoMCL

```
cd /local/Project_Intoshia/OrthoMCL
mkdir scafos
mv /local/Project_Intoshia/OrthoMCL/alignments/*.fasta /local/Project_Intoshia/OrthoMCL/scafos
```

SCaFoS имеет собственный графический интерфейс, который можно вызвать командой

```
scafos
```

Выберите пункт меню SPECIES PRESENCE и укажите в Input directory путь к папке с выравниваниями

(/local/Project_Intoshia/OrthoMCL/scafos), а в Output directory путь к подпапке step1 (/local/Project_Intoshia/OrthoMCL/scafos/step1). Запустите задачу кнопкой RUN. В процессе работы SCaFoS проверит все файлы в /scafos и запишет лог в директорию /scafos/step1. Если в процессе работы не возникло ошибок, то SCaFoS сразу предложит выполнить конкатенацию, но мы откажемся. Теперь переходите к пункту DATASET ASSEMBLING, и в поле OTUs file введите путь к файлу step1-name.otu: /local/Project_Intoshia/OrthoMCL/scafos/step1/step1-name.otu; а в поле Output directory измените step1 на step2. Отметьте галкой пункт меню Making of Chimera и выберите Minimal evolutionary distance (Tree-Puzzle), также отметив галкой пункт Gamma. Убедитесь что в Output format выбран пункт FASTA и запустите задачу - RUN.

В процессе работы SCaFoS производит конкатенацию данных и обращается к программе Tree-Puzzle для выбора последовательности с наименьшей эволюционной дистанцией, если в индивидуальном выравнивании организм представлен более чем одной последовательностью. Данная процедура отбрасывает все внутригеномные паралоги, которые могли быть объединены в один кластер OrthoMCL как ко-ортологи, и оставляет лишь одну последовательность на геном для каждого кластера, выбирая при этом наименее дивергентный паралог. После завершения работы основной результат (конкатенированное выравнивание) и дополнительные логи будут записаны в /scafos/step2. Нас интересует файл step2.fasta - конкатенированное выравнивание в FastA формате. При конкатенации SCaFoS вводит знаки ‘?’ в позиции выравнивания с отсутствующими последовательностями. Для замены этих знаков на более привычный вариант ‘X’ для аминокислотных и ‘N’ для нуклеотидных последовательностей используйте следующие команды

```
sed -i ‘s/?/X/g’ step2.fasta
```
аминокислоты

```
sed -i ‘s/?/N/g’ step2.fasta
```
нуклеотиды

Для подготовки файла конкатенированных выравниваний к филогенетическому анализу, FastA формат потребуется конвертировать в формат Phylip для RAxML и формат Nexus для байесовских программ. Вот образцы трех форматов:

FastA

```
>Adineta_vaga
VNYLIYRYLSEESHIAHTVPPIQKGLYYTELSLIDAVIVTVLRGHESEVFIDLLASGSGDSTARIWNLEVLHKDVTSLDWNGTLATGSYDGARIWSGLSTLQHKGPIF
>Amphimedon_queenslandica
VNFLVYRYLLEESHVKHTIPPIQKGLQYVELSLIDAVSVKILQGHGSEVFTNTLVTGSGDSTARLWHVSVLHRDVTSVHWSGSVATGSYDGARLWSGISTLRHSGPIF
>Schistosoma_mansoni
VNFLIFRYLQEESQISQSVPPIQKGLQYTELSLIDAVVITVLKGHQSEVFIDMLASGSGDSTARIWNLQVLHKDVTSLDWNGSLATGSYDGARVWNGLTTLQHKGPIF
>Strongylocentrotus_purpuratus
VNFLVYRYLQEESHISQSVPPIQKGLQYVELSLIDAVVATVLRGHESEVFSDLLASGSGDSTARIWNLQVLHKDVTSLDWNGSLATGSYDGARIWSGLSTLQHKGPIF
>Trichoplax_adhaerens
LNYLIYRYLQEESHIHRSVPPVQKGMLYYYISLIDSISSDQRRSSESDIFAELIATGSSDGTAKISQPAVLHYAVTSIVWKGSLLTACADGGRIWQGLATLNHTAPIF
```

Phylip

```
5 127
Adineta_vaga                 VNYLIYRYLSEESHIAHTVPPIQKGLYYTELSLIDAVIVTVLRGHESEVFIDLLASGSGDSTARIWNLEVLHKDVTSL
Amphimedon_queenslandica     VNFLVYRYLLEESHVKHTIPPIQKGLQYVELSLIDAVSVKILQGHGSEVFTNTLVTGSGDSTARLWHVSVLHRDVTSV
Schistosoma_mansoni          VNFLIFRYLQEESQISQSVPPIQKGLQYTELSLIDAVVITVLKGHQSEVFIDMLASGSGDSTARIWNLQVLHKDVTSL
Strongylocentrotus_purpuratus VNFLVYRYLQEESHISQSVPPIQKGLQYVELSLIDAVVATVLRGHESEVFSDLLASGSGDSTARIWNLQVLHKDVTSL
Trichoplax_adhaerens         LNYLIYRYLQEESHIHRSVPPVQKGMLYYYISLIDSISSDQRRSSESDIFAELIATGSSDGTAKISQPAVLHYAVTSI
```

Nexus

```
#NEXUS
begin data;
dimensions ntax=5 nchar=127;
format interleave=no datatype=protein gap=- missing=X;
matrix
Adineta_vaga                  VNYLIYRYLSEESHIAHTVPPIQKGLYYTELSLIDAVIVTVLRGHESEVFIDLLASGSGDSTARIWNLEVLHKDVTSL
Amphimedon_queenslandica      VNFLVYRYLLEESHVKHTIPPIQKGLQYVELSLIDAVSVKILQGHGSEVFTNTLVTGSGDSTARLWHVSVLHRDVTSV
Schistosoma_mansoni           VNFLIFRYLQEESQISQSVPPIQKGLQYTELSLIDAVVITVLKGHQSEVFIDMLASGSGDSTARIWNLQVLHKDVTSL
Strongylocentrotus_purpuratus VNFLVYRYLQEESHISQSVPPIQKGLQYVELSLIDAVVATVLRGHESEVFSDLLASGSGDSTARIWNLQVLHKDVTSL
Trichoplax_adhaerens          LNYLIYRYLQEESHIHRSVPPVQKGMLYYYISLIDSISSDQRRSSESDIFAELIATGSSDGTAKISQPAVLHYAVTSI
;
end;
```

При помощи скрипта fastatophylip (в папке Supplementary_files на сервере) файл в формате FastA можно автоматически конвертировать в формат Phylip

```
cp  /P/NGSPhylo2014/Supplementary_files/fastatophylip \
/local/Project_Intoshia/OrthoMCL/scafos/step2
cd  /local/Project_Intoshia/OrthoMCL/scafos/step2
./fastatophylip -in step2.fasta
```

Данные в формате Phylip будут записаны в step2.phy. Для конвертирования в формат Nexus достаточно изменить файл в формате Phylip, добавив соответствующий заголовок и окончание (см. образец). Модифицируйте step2.phy и сохраните данные в формате Nexus в отдельный файл step2.nex

Скопируйте Phylip и Nexus файлы в новую папку trees под /Project_Intoshia, дав этим файлам более описательное название, например intoshiaOG

```
cd  /local/Project_Intoshia
mkdir trees
cp  /local/Project_Intoshia/OrthoMCL/scafos/step2/step2.phy \
/local/Project_Intoshia/trees/intoshiaOG.phy
cp  /local/Project_Intoshia/OrthoMCL/scafos/step2/step2.nex \
/local/Project_Intoshia/trees/intoshiaOG.nex
```

Филогенетический анализ мы будем выполнять при помощи СК Ломоносов. Скопируйте в папку raxml на сервере СК файл intoshiaOG.phy, а в папки phylobayes и mrbayes - intoshiaOG.nex

```
scp  -o identityFile=/path_to_key/key_name  /local/Project_Intoshia/trees/intoshiaOG.phy \
username@lomonosov.parallel.ru:~/raxml
scp  -o identityFile=/path_to_key/key_name  /local/Project_Intoshia/trees/intoshiaOG.nex \
username@lomonosov.parallel.ru:~/phylobayes
scp  -o identityFile=/path_to_key/key_name  /local/Project_Intoshia/trees/intoshiaOG.nex \
username@lomonosov.parallel.ru:~/mrbayes
```

Соединитесь с сервером СК в терминале и выполните вход

```
ssh  -o identityFile=/path_to_key/key_name  username@lomonosov.parallel.ru
```

На сервере СК подключите библиотеку openmpi/1.5.5-gcc

```
module add openmpi/1.5.5-gcc
```

# RAxML

Для запуска RAxML в директории /raxml создайте следующий скрипт raxml.sh

```
cd ~/raxml
cat  >  raxml.sh
```

```
#!/bin/sh
./raxmlHPC-PTHREADS-SSE3 -T 8 -f a -m PROTGAMMAGTR -N 100 -p 12345 -x 12345 \
 -s intoshiaOG.phy -n intoshiaOG.tre
```

Мы выделяем программе 8 ядер (-T) и просим выполнить поиск ML дерева вместе с быстрым бутстрэпом «rapid bootstrap» (-f a) размером в 100 реплик (-N 100). Расчет правдоподобия будет использовать модель эволюции GTR «General Time Reversible», параметры которой (вероятности замещений) оптимизирует сама программа; также будет учитываться разная скорость замен в сайтах, моделируемая гамма распределением (-m PROTGAMMAGTR). Цифры рядом с параметрами -p и -x особого значения не имеют - они служат для инициализации генератора случайных чисел.

```
chmod  a+x  raxml.sh
```

Запуск программ на СК Ломоносов осуществляется через директиву sbatch из под директории ~/_scratch. Скопируйте папку raxml в ~/_scratch и запустите скрипт следующей командой, запросив один узел (-N1) на разделе test

```
cp  -r  ~/raxml  ~/_scratch
cd  ~/_scratch/raxml
sbatch -p test -N1  ./raxml.sh
```

Проверить состояние задачи можно командой squeue `squeue -u username`

После завершения работы в директории ~/_scratch/raxml должны появиться следующие файлы:

RAxML_info.intoshiaOG.tre – лог выполнения задачи RAxML
RAxML_bestTree.intoshiaOG.tre – дерево с лучшим значением правдоподобия из найденных
RAxML_bootstrap.intoshiaOG.tre – деревья полученные для реплик бутстрэпа
RAxML_bipartitions.intoshiaOG.tre – дерево bestTree со значениями поддержки (формат 1)
RAxML_bipartitionsBranchLabels.intoshiaOG.tre – дерево bestTree со значениями поддержки (формат 2)

Скопируйте результат в директорию ~/raxml из ~/_scratch/raxml и загрузите папку raxml с сервера СК на локальный компьютер

```
scp  -o identityFile=/path_to_key/key_name -r  username@lomonosov.parallel.ru:~/raxml \
/local/Project_Intoshia/trees
```

# Phylobayes

Запуск программы Phylobayes осуществляется аналогично RAxML при помощи следующего скрипта pb1.sh

```
cd  ~/phylobayes
cat  >  pb1.sh
```

```
#!/bin/sh
./pb_mpi -d intoshiaOG.nex -dgam 4 -cat -poisson intoshiaOG_chain1
```

Для байесовского анализа мы запустим параллельно две независимые цепи chain1 и chain2 (создайте такой же скрипт для второй цепи intoshiaOG_chain2 и сохраните в pb2.sh). В качестве модели эволюции выбрана модель CAT-Poisson, которая рассчитывает индивидуальный профиль замещений для каждого сайта (-cat) и комбинирует его с равномерным глобальными значением вероятности замещений (-poisson). Мы также будем учитывать разную скорость замен в сайтах согласно гамма распределению с 4-мя категориями скоростей (-dgam 4).

```
chmod  a+x  pb1.sh pb2.sh
```

Скопируйте папку phylobayes в ~/_scratch и запустите скрипт следующей командой, запросив 8 ядер (-n8) для каждой цепи на разделе test и добавив директиву ompi

```
cp  -r  ~/phylobayes  ~/_scratch
cd  ~/_scratch/phylobayes
sbatch -p test -n8 ompi ./pb1.sh
sbatch -p test -n8 ompi ./pb2.sh
```

В данной конфигурации задача phylobayes не имеет определенной точки завершения: обе цепи будут продолжать аккумулировать результаты случайного розыгрыша, пока задача не будет снята. Если же счет был остановлен, то его можно продолжить для любой цепи с момента остановки

```
pb_mpi intoshiaOG_chain1
```

Критерием для прекращения счета является конвергенция оптимизируемых параметров модели и топологии в пределах одной цепи, а также между независимыми цепями. Изменение таких показателей как значение правдоподобия, длина дерева, коэффициент альфа отражено в файле .trace для каждой цепи, а сами деревья записаны в файле .treelist. Проверить динамику показателей в .trace файлах можно при помощи утилиты Tracer или средствами R, MS Excel, Libre Calc и пр. Для проверки сходимости топологий и параметров между цепями используйте программы bpcomp и tracecomp (прилагаются к phylobayes)

```
bpcomp -x 1000 10 -c 0.05 -o intoshiaOG  intoshiaOG_chain1 intoshiaOG_chain2
```

```
tracecomp -x 1000 -o intoshiaOG  intoshiaOG_chain1 intoshiaOG_chain2
```

В командах следует указать размер burn-in, т.е. количество начальных точек, которые будут исключены из анализа (-x 1000*). Для bpcomp также можно указать частоту отбора деревьев (10: использовать только каждое 10-е дерево) и значение апостериорной вероятности, ниже которого узлы дерева будут коллапсированы (-c 0.05).

bpcomp запишет консенсусное дерево в intoshiaOG.con.tre; средний и максимальный показатель расхождения в бипартициях между цепями будет записан в intoshiaOG.bpdiff

tracecomp запишет показатели расхождения и эффективный объем выборки для параметров в intoshiaOG.contdiff

Разработчики Phylobayes рекомендуют следующее очень общее руководство для оценки сходимости цепей:

**bpcomp** (tree files)

- maxdiff < 0.1: good run.
- maxdiff < 0.3: acceptable: gives a good qualitative picture of the posterior consensus.
- 0.3 < maxdiff < 1: the sample is not yet sufficiently large, and the chains have not converged, but this is on the right track.
- if maxdiff = 1 even after 10000 points, this indicates that at least one of the runs is stuck in a local maximum.

**tracecomp** (parameter files)

- maxdiff < 0.1 and minimum effective size > 100: good run.
- maxdiff < 0.3 and minimum effective size > 50: acceptable run.

Следует помнить, что сходимость параметров и топологий не гарантирована даже при очень длительном прогоне цепей. Если в сходимости цепей долгое время не наблюдается положительной динамики, то счет можно завершить.

## MrBayes

Команда для запуска MrBayes дается несколько иначе, чем для Phylobayes или RAxML: настройки всех параметров запуска MrBayes записываются отдельным блоком в Nexus файле вместе с самими данными. Сам блок команд выглядит следующим образом

```
begin mrbayes;
set autoclose=yes nowarnings=yes quitonerror=yes;
log start filename=intoshiaOG.log;
prset aamodelpr=fixed(gtr);
lset rates=gamma ngammacat=4;
mcmc
ngen=3000000 nruns=2 nchains=4
samplefreq=1000 printfreq=10000
relburnin=yes burninfrac=0.5;
sump;
sumt contype=allcompat conformat=Figtree;
end;
```

* размер burn-in подбирается в соответствие с точкой перехода цепей к «стационарной» стадии оптимизации параметров, т.е. стадии блуждания около найденного оптимального значения без проявления тренда

После директивы запуска MrBayes (begin mrbayes;) мы записали строку для контроля общего поведения программы, подразумевающее неинтерактивное выполнение задачи (set autoclose=yes nowarnings=yes quitonerror=yes;). Следующая строка (log start filename=intoshiaOG.log;) указывает в каком файле вести лог исполнения задачи. Далее идет выбор модели эволюции: как и для RAxML, мы будем использовать модель GTR (prset aamodelpr=fixed(gtr);) и 4 категории для распределения скоростей (lset rates=gamma ngammacat=4;). В отличие от Phylobayes, параметры длины, сэмплинга и burn-in цепей даются MrBayes сразу при запуске (под директивой mcmc). Для данной задачи мы запросили 3 миллиона генераций в 2-х независимых запусках по 4-е цепи в каждом (ngen=3000000 nruns=2 nchains=4); частота сэмплинга: 1 на 1000 генераций, с выводом результатов в лог каждые 10000 генераций (samplefreq=1000 printfreq=10000); первые 50% точек цепи будут исключены из анализа как burn-in (relburnin=yes burninfrac=0.5;). Директивы sump и sumt выводят результаты прогона цепей для значений оптимизируемых параметров и деревьев. Для sump и sumt можно указать размер burn-in отличный от mcmc, но мы будем использовать единое значение. Для вывода консенсусного дерева мы дополнительно указываем тип консенсуса и формат дерева (contype=allcompat conformat=Figtree;): с contype=allcompat sumt не будет коллапсировать узлы с поддержкой <0.5. Блок команд завершается директивой end;

Данный блок можно сразу записать в Nexus файл или поместить в отдельный файл (mb.comm.txt) и добавить к Nexus файлу следующим образом

```
cat  intoshiaOG.nex  mb.comm.txt >> intoshiaOG.mb.nex
```

Создайте скрипт mb.sh для запуска программы

```
cat  >  mb.sh
```

```
#!/bin/sh
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/mrbayes/beagle-lib
./mb  intoshiaOG.mb.nex
```

```
chmod  a+x  mb.sh
```

Скопируйте папку mrbayes в ~/_scratch и запустите скрипт следующей командой, запросив 8 ядер (-n8) на разделе test (количество ядер ≤ суммарное количество цепей)

```
cp  -r  ~/mrbayes  ~/_scratch
cd  ~/_scratch/mrbayes
sbatch -p test -n8  ompi  ./mb.sh
```

Счет будет остановлен после прохождения запрошенного числа генераций, и результаты будут записаны в несколько файлов:

intoshiaOG.mb.nex.pstat – статистика по оптимизируемым параметрам
intoshiaOG.mb.nex.lstat – статистика по значению правдоподобия
intoshiaOG.mb.nex.con.tre – консенсусное дерево (формат Figtree)
intoshiaOG.mb.nex.trprobs – исследованные в процессе MCMC деревья, ранжированные по значению апостериорной вероятности
intoshiaOG.mb.nex.parts – файл с бипартициями

intoshiaOG.mb.nex.tstat
intoshiaOG.mb.nex.vstat } статистика по бипартициям

Также в процессе работы, помимо лога, MrBayes будет вести запись параметров и деревьев:

intoshiaOG.mb.nex.run1.p
intoshiaOG.mb.nex.run2.p
intoshiaOG.mb.nex.run1.t
intoshiaOG.mb.nex.run2.t

Как и для .trace файлов Phylobayes, за ходом цепей можно следить при помощи Tracer или другими средствами.

N.B. В последних версиях MrBayes предусмотрена возможность продолжения счета после плановой или внеплановой остановки. Для этого достаточно перезапустить счет, предварительно добавив append=yes под директивой mcmc в блоке команд Nexus файла.

| Программа | Источник | Назначение |
|---|---|---|
| FastQC | http://www.bioinformatics.babraham.ac.uk/projects/fastqc/ | оценка качества чтений |
| Trimmomatic | http://www.usadellab.org/cms/?page=trimmomatic | удаление последовательностей адаптеров из чтений |
| Jellyfish | http://www.genome.umd.edu/jellyfish.html | вычисление распределения частот к-меров в ридах |
| Velvet | https://www.ebi.ac.uk/~zerbino/velvet/ | ассемблер NGS данных (OpenMP) |
| Ray | http://denovoassembler.sourceforge.net/download.html | ассемблер NGS данных (MPI) |
| SPAdes | http://bioinf.spbau.ru/ru/spades | ассемблер NGS данных с возможностью работать с неравномерным покрытием |
| QUAST | http://bioinf.spbau.ru/ru/quast | оценка качества сборок по статистическим параметрам |
| CEGMA | http://korflab.ucdavis.edu/datasets/cegma/ | оценка полноты сборки эукариотического генома |
| Scipio | http://www.webscipio.org/webscipio/download_scipio | картирование аминокислотной последовательности транскрипта на нуклеотидную последовательность гена |
| Augustus | http://bioinf.uni-greifswald.de/augustus/ | *ab initio* предсказание генов в эукариотическом геноме |
| BLAST | http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download | поиск последовательностей по сходству |
| OrthoMCL | http://www.orthomcl.org/common/downloads/software/ | пайплайн для кластеризации ортологов |
| MCL | http://micans.org/mcl/src/ | реализация алгоритма Markov Cluster |
| MAFFT | http://mafft.cbrc.jp/alignment/software/ | выравнивание последовательностей |
| trimAl | http://trimal.cgenomics.org/downloads | удаление сомнительных участков из выравнивания |
| SCaFoS | http://megasun.bch.umontreal.ca/Software/scafos/scafos_download.html | конкатенация выравниваний |
| TREE-PUZZLE | http://www.tree-puzzle.de/ | реконструкция деревьев методом максимального правдоподобия |
| RAxML | http://sco.h-its.org/exelixis/web/software/raxml/index.html | реконструкция деревьев методом максимального правдоподобия |
| Phylobayes | http://megasun.bch.umontreal.ca/People/lartillot/www/index.htm | реконструкция деревьев методом Байеса |
| MrBayes | http://mrbayes.sourceforge.net/download.php | реконструкция деревьев методом Байеса |