Построение пайплайна получения генетических вариантов. На примере фреймворка Toil

Автор

Мазяр Алексей Николаевич, группа 21213

Ссылка на официальный сайт о Toil

https://toil.readthedocs.io/en/latest/index.html#

Использованный результат секвенирования

https://www.ncbi.nlm.nih.gov/sra/SRX8340760[accn]

Описание файлов

bash_pipeline.sh

Пайплайн, реализованный на Bash. Принимает два аргумента: путь к файлу с референсным геномом, путь к файлу с результатом секвенирования.

quality_pipeline.py

Пайплайн, реализованный на Python с использование фреймворка Toil. Принимает два аргумента: путь к файлу с референсным геномом, путь к файлу с результатом секвенирования.

launch_toil_pipeline.sh

Скрипт, запускающий quality_pipeline.py. Он нужен для настройки логирования toil, так как изначально этот фреймворк всё пишет в stderr. Принимает два аргумента: путь к файлу с референсным геномом, путь к файлу с результатом секвенирования.

quality_check.py

Скрипт на Python для парсинга результата работы samtools flagstat для оценки качества отображения.

toil_hello_world.py

“Hello world” на Python с использование фреймворка Toil.

SRR11788637_on_hg38.txt

Результат работы samtools flagstat на загруженных ридах из SRR11788637 и референсном геноме из файла hg38.fa.gz.

log.txt

Логи Toil’a.

progress.log.txt

Логи использованных в пайплайне программ.

Инструкция по установке

1. Установить Python версии 3.8 или выше.
2. pip install toil

Официальная инструкция: https://toil.readthedocs.io/en/latest/gettingStarted/install.html

Инструкция по запуску пайплайна

1. Установить пайплайн согласно инструкции выше
2. Скачать желаемые файл с референсным геномом (.fa) и файлы с результатом секвенирования (.fq/.fastq/.fastq.gz)
3. Запустить launch_toil_pipeline.sh, передав ему .fa и .fq файлы в качестве аргументов

Визуализация

Фреймворк Toil не даёт возможности визуализировать структуру пайплайна, поэтому я изобразил её вручную с использованием сервиса https://excalidraw.com/. Результат - файл schema.svg.
В отличии от абстрактного алгоритма здесь реализовано несколько вспомогательных функций:

- mk_outdir - создаёт или очищает директорию out/, в которой будут записаны все результаты работы пайплайна
- clean_fastqc - удаляет .zip файл, полученный в результате работы fastqc
- echo_success - выводит на терминал сообщение об удачном завершении работы пайплайна

На схеме видно, что от корневого узла mk_outdir отходят две основные ветки: fastqc и minimap2. Так как обе операции не зависят от результатов работы друг друга, их можно выполнять “параллельно” (в Python они всё равно выполнятся последовательно). Более

того, echo_success “зависит” от результатов работы обоих веток, что позволяет определить удачное завершение работы пайплайна как удачное завершение обоих веток.

Про логи

В Toil логи это и есть вывод в stderr, который можно дополнять собственными сообщениями, используя job.log(“...”). В реализованном пайплайне такие сообщения выводятся в начале выполнения каждого шага.
Таким образом, выведенные результаты работы и лог-файл объединены.