Streamlining Next-Gen Data Quality Control with Fastp in Miniconda Environment

Let's get started with *what Fastp is?*

Welcome to the world of data quality control in bioinformatics! ***Fastp*** is a robust and user-friendly tool designed to perform fast all-in-one preprocessing for Next-Generation Sequencing (NGS) data. It's your go-to solution for ensuring the quality and reliability of your sequencing data, a crucial step before diving into in-depth analyses.

Before installation, create a new environment in Miniconda for ***Fastp using:***

Command: ***conda create -n fastp***

This command creates a virtual environment named ***"fastp"*** *using Conda, a package manager. It allows isolated Python environments for managing dependencies and projects.*

And then activate the environment by typing: ***conda activate fastp***

This command activates a specific conda environment named "fastp". When you activate an environment, you're essentially switching to that environment and any subsequent installations or commands will be applied within it.

Once the environment (fastp) is activated let's understand installation and usage:

Installation in Miniconda Environment:

To get started, let's make sure you have ***Fastp*** installed in your Miniconda environment. Here's how you can do it:

Command: conda install -c bioconda fastp

This command installs ***Fastp*** and its dependencies, ensuring a smooth experience in your Miniconda environment.

Let's break down the command:

• `conda install`: This is the command to install packages using Conda.

• `-c bioconda`: This flag specifies the Conda channel from which to install the package, in this case, the "bioconda" channel.

• `fastp`: This is the name of the package you want to install.

```
(base) praneet@pop-os:~$ conda activate fastp
(fastp) praneet@pop-os:~$ conda install -c bioconda fastp
Collecting package metadata (current_repodata.json): done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: unsuccessful attempt using repodata from current_repodata.json, retrying with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/praneet/miniconda3/envs/fastp

  added / updated specs:
    - fastp


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    libdeflate-1.17            |       h5eee18b_1          64 KB
    ------------------------------------------------------------
                                          Total:          64 KB
```

```
The following NEW packages will be INSTALLED:

  _libgcc_mutex      pkgs/main/linux-64::_libgcc_mutex-0.1-main
  _openmp_mutex      pkgs/main/linux-64::_openmp_mutex-5.1-1_gnu
  fastp              bioconda/linux-64::fastp-0.23.2-hb7a2d85_2
  isa-l              pkgs/main/linux-64::isa-l-2.30.0-h7f8727e_0
  libdeflate         pkgs/main/linux-64::libdeflate-1.17-h5eee18b_1
  libgcc-ng          pkgs/main/linux-64::libgcc-ng-11.2.0-h1234567_1
  libgomp            pkgs/main/linux-64::libgomp-11.2.0-h1234567_1
  libstdcxx-ng       pkgs/main/linux-64::libstdcxx-ng-11.2.0-h1234567_1


Proceed ([y]/n)? y


Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(fastp) praneet@pop-os:~$
```

After packages are installed and downloaded, a text pops up “Proceed ([y]/[n])? Type y and enter” and the installation is done.

As you have successfully installed ***Fastp*** in your local system, now let’s take some simple raw sequence files and use Fastp to perform quality checks.

How to do that?

I got you covered.

Here we are working with a dataset with accession ID “SRR12486989, " a paired-end dataset. If you're not sure how to fetch datasets check out my previous blog about ***Prefetch and SRA-toolkit.*** Once your datasets are fetched perform ***Fastp*** here I have fetched a dataset SRR12486989 from SRA using prefetch commands.

Command: fastp -i SRR12486989_1.fastq -I SRR12486989_2.fastq -o out.R1.fq.gz -o out.R2.fq.gz

Command breakdown:

- -i SRR12486989_1.fastq: Specifies the input file containing forward reads.
- -I SRR12486989_2.fastq: Specifies the input file containing reverse reads.
- -o out.R1.fq.gz: Specifies the output file for processed forward reads in compressed FASTQ format.
- -o out.R2.fq.gz: Specifies the output file for processed reverse reads in compressed FASTQ format.

This command processes the input files, conducts quality checks, and saves the cleaned reads in the specified output files.

```
(base) praneet@pop-os:~$ fastp -i SRR12486989_1.fastq -I SRR12486989_2.fastq  -o out.R1.fq.gz -O out.R2.fq.gz
Read1 before filtering:
total reads: 5440369
total bases: 408449568
Q20 bases: 408449568(100%)
Q30 bases: 408449568(100%)

Read2 before filtering:
total reads: 5440369
total bases: 408438400
Q20 bases: 408438400(100%)
Q30 bases: 408438400(100%)

Read1 after filtering:
total reads: 589893
total bases: 43570928
Q20 bases: 43570928(100%)
Q30 bases: 43570928(100%)

Read2 after filtering:
total reads: 589893
total bases: 43564207
Q20 bases: 43564207(100%)
Q30 bases: 43564207(100%)
```

Working of Fastp:

Here's a detailed breakdown of what happens before filtering and after filtering using Fastp, and the key quality metrics it provides:

Before Filtering: Raw Data

When you first obtain raw sequencing data, it usually contains various issues such as adapter contamination, low-quality bases, and sequencing artifacts. Fastp steps in to address these problems.

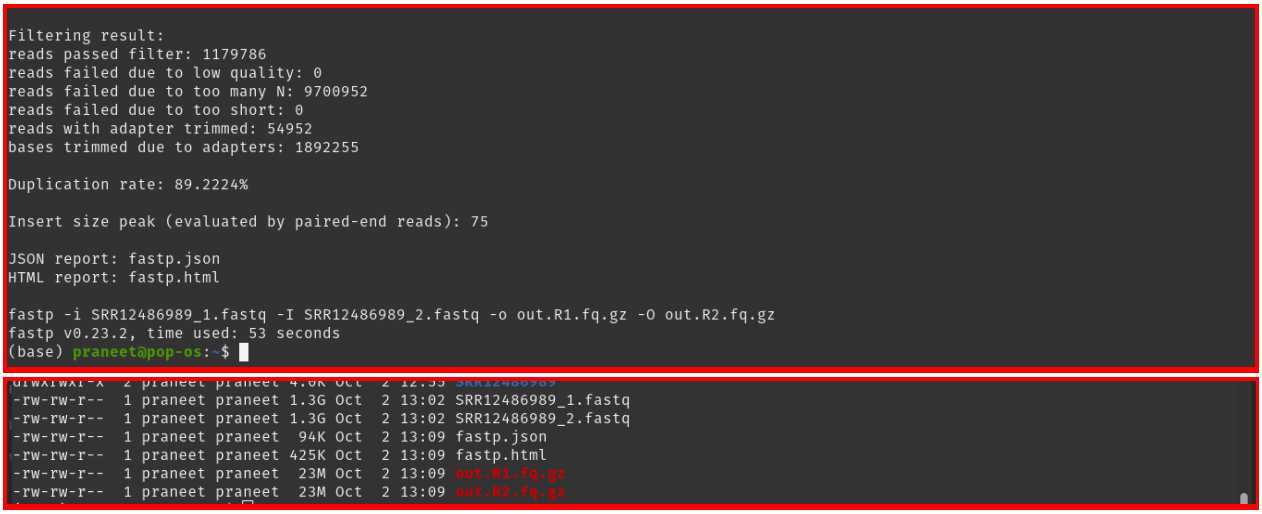
- **Total Reads:** The total number of reads in the input data file.

- **Total Bases:** The total number of bases in all the reads combined.
- **Q20 Bases:** The number of bases with a quality score of 20 or higher. A quality score of 20 corresponds to a 1% chance of an error.
- **Q30 Bases:** The number of bases with a quality score of 30 or higher. A quality score of 30 corresponds to a 0.1% chance of an error.

After Filtering: Processed Data

After Fastp filtering, the data is cleaned and optimized for downstream analysis. Here's what happens to the data:

- **Filtered Reads:** The number of reads that pass Fastp's quality control filters. Reads failing these filters (due to low quality, adapter contamination, etc.) are discarded.
- **Filtered Bases:** The total number of bases in the filtered reads.
- **Q20 Bases (After Filtering)**: The number of bases with a quality score of 20 or higher in the filtered reads.
- **Q30 Bases (After Filtering)**: The number of bases with a quality score of 30 or higher in the filtered reads.



Understanding the Filtering Process:

Adapter Removal: Fastp trims adapter sequences from the ends of the reads. Adapters are short DNA sequences used during sequencing that need to be removed for accurate analysis.

Quality Filtering: Reads with low-quality bases are removed. Fastp uses quality scores to identify unreliable bases. Bases below a certain threshold (like Q20 or Q30) might indicate sequencing errors and are filtered out.

Length Filtering: Extremely short reads, often resulting from sequencing artifacts, are removed. Fastp allows you to set a minimum length threshold for the reads.

PolyX Trimming: Sequences with poly-X (homopolymer) tails can be problematic. Fastp can trim these tails to improve the data quality.

Interpreting the Metrics:

- **Total Reads and Total Bases:** These metrics represent the volume of data you started with. Higher values are generally better but don't guarantee quality.
- **Q20 and Q30 Bases:** The higher the percentage of Q20 and Q30 bases, the more accurate your data. Higher Q20 and Q30 percentages indicate high sequencing quality and reliability.
- **Filtered Reads and Filtered Bases:** After filtering, you're left with high-quality, reliable reads. These are the reads you should use for downstream analyses like alignment, variant calling, or assembly.

Fastp generates detailed reports in JSON and HTML formats, providing graphical representations of these metrics. These reports help you visualize the quality of your data and make informed decisions about its usability in your research.

Recognizing Output Files: HTML and JSON

There are two kinds of output files that you'll find after executing ***Fastp***:

JSON Files: Comprehensive quality control metrics are produced in JSON files using Fastp. The number of reads read length distribution and per-base quality scores are just a few of the comprehensive details these files offer regarding the caliber of your data.

HTML Files: In an aesthetically pleasing format, Fastp creates HTML reports that summarize the quality control metrics. These reports may be opened in any web browser and are quite helpful for quickly visualizing the quality of the data.

Why It's Important to Check for Quality:

In NGS data processing, quality control is essential for several reasons:

Data Reliability: Guarantees the accuracy and dependability of the data you're evaluating, avoiding misunderstandings brought on by subpar reads.

Downstream analysis: Accurate downstream analyses, like variant calling, assembly, and alignment, produce more dependable outcomes when they utilize high-quality data.

Resource Efficiency: Removing low-quality reads speeds up processing for later analysis and maximizes computational resources.

Conclusion:

Empowering Your NGS Journey,

In conclusion, Fastp in your Miniconda environment empowers your NGS data analysis journey. By ensuring the quality of your data, you pave the way for accurate, reliable, and insightful biological discoveries. Remember, understanding your data from the very beginning is the key to unlocking the secrets hidden within the sequences.

Happy sequencing! 🧬🔍✨

https://academic.oup.com/bioinformatics/article/34/17/i884/5093234
https://github.com/OpenGene/fastp
https://anaconda.org/bioconda/fastp

** Streamlining NGS Data QC with Fastp in a Miniconda Environment - My Medium Series Continues! **

**Building on the buzz from my previous article about SRA-TOOLS, Prefetch, and Fasterq-dump, I'm excited to unveil Part 2, delving into the world of Fastp! **

In this new piece, we'll explore:

- ⚡ Fastp's superpowers: Adapter trimming, quality filtering, base correction, and more!
- Setting up Fastp within a Miniconda environment for seamless integration.
- Practical workflows and best practices to optimize Fastp for your specific needs.

I will be happy to know about your issues Troubleshooting tips and tricks to conquer common challenges with confidence.

**Whether you're an NGS expert or just starting, this article is packed with valuable insights and practical takeaways to help you streamline your data quality control processes. **

Ready to dive in? Check out the full article here: [Link to your Medium article]

#NGS #dataprocessing #bioinformatics #Fastp #Miniconda #qualitycontrol #datascience #research #openscience

**I'd love to hear your experiences with Fastp! Share your thoughts and feedback in the comments below.