SUPPLEMENTARY MATERIALS

InterARTIC: an interactive web application for whole-genome nanopore sequencing analysis of SARS-CoV-2 and other viruses

James M. Ferguson, Hasindu Gamaarachchi, Thanh Nguyen, Alyne Gollon, Stephanie Tong, Chiara Aquilina-Reid, Rachel Bowen-James & Ira W. Deveson

	Laptop	**GridION**
Model	Dell XPS	GridION Mk1
Processor	Intel i7-8750H	Intel i7-7700
RAM	16 GB	64 GB
Disk	1TB NVMe SSD	4TB NVMe SSD (RAID configuration)
O/S	Ubuntu 16.04.7 LTS on WSL for Windows 10	Ubuntu 16.04.7 LTS
Threads used	4	4

Supplementary Table 1. Specifications of computers used to run example workflows.

					Laptop run-time (min:sec)				GridION run-time (min:sec)			
Set	**Virus**	**Amplicons**	**Samples**	**Pipeline**	**Total**	**Gather**	**Demux**	**Analysis**	**Total**	**Gather**	**Demux**	**Analysis**
1	SARS-CoV-2	Eden V1 2.5kb	10	Nanopolish	32:19	00:07	17:56	14:05	25:09	00:05	13:42	11:33
				Medaka-Longshot	30:24	0:11	17:53	12:09	21:41	00:05	13:43	07:45
3	Ebola	Artic V1 400bp	2	Nanopolish	03:57	00:02	02:23	01:30	02:55	00:01	01:42	01:10
				Medaka-Longshot	04:09	00:03	02:24	01:42	02:54	00:01	01:40	01:12

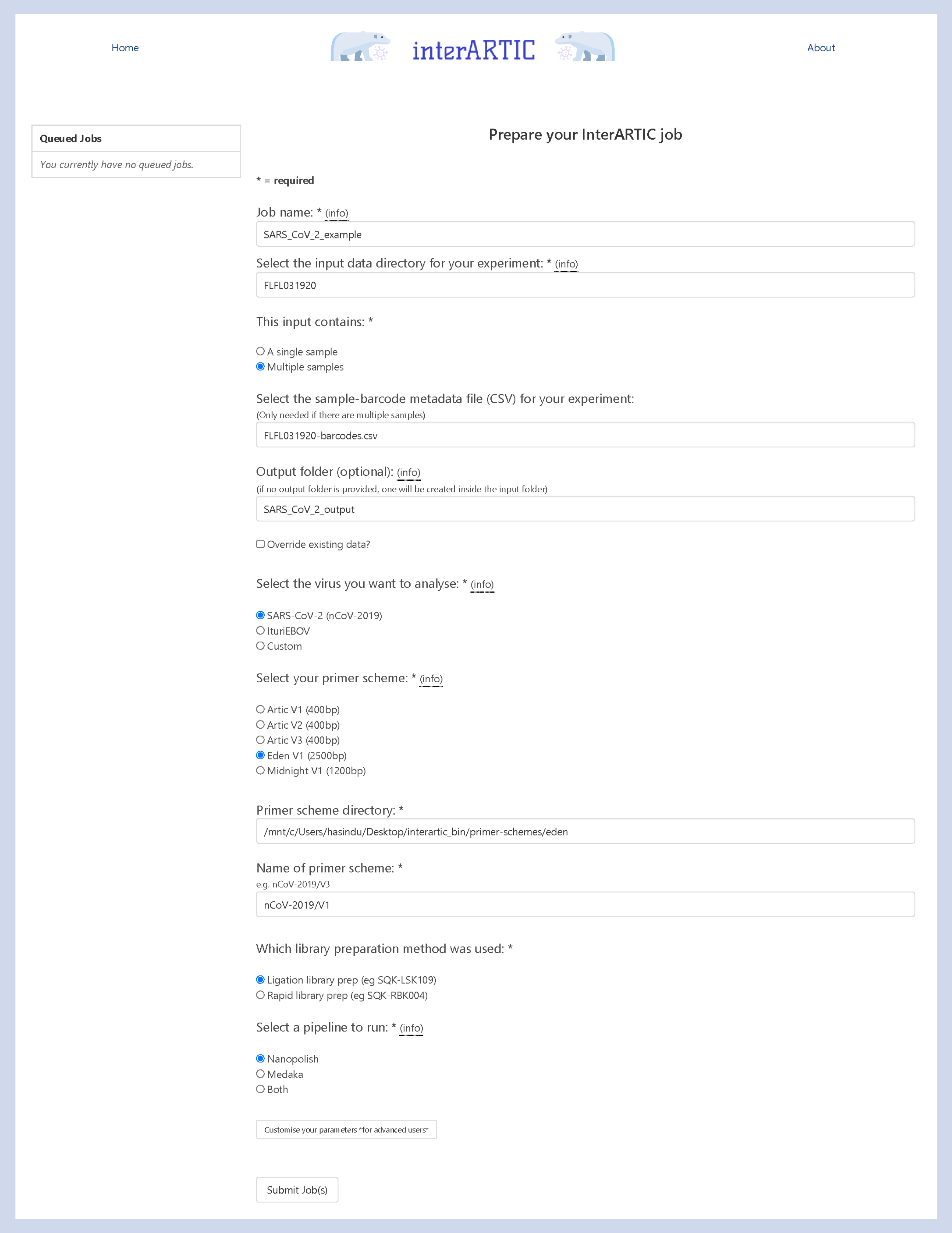
Supplementary Table 2. Indicative run-times for *InterARTIC* workflow on example viral WGS datasets.

Supplementary Note 1. The art of snake charming

InterARTIC development involved the use of the Python programming language and depends on several third-party Python modules and software written predominantly in Python (e.g., *Flask*, *Celery*, *ARTIC* tools, etc). The Python ecosystem (including the language itself, in addition to Python libraries) has limited backward compatibility. As a result, Python software is often compatible only with the exact version of the Python interpreter and library versions it was developed with (sometimes specific even to the minor version level). Python virtual environments and Anaconda are designed to resolve issues related to version compatibility but - at least in our experience - software installation via these methods can be complicated, especially for novice users.

Fortunately, the Python interpreter is predominantly written in C. Generally speaking, both the C language and system libraries have good backward compatibility. For instance, GLIBC is fully backward compatible. Thus, if you compile a C program on an older Linux system (e.g., Ubuntu 14) with an older compiler (e.g., gcc 4.8) and statically link third party libraries with limited backward compatibility, while dynamically linking the basic backward compatible libraries, the compiled binary would be portable on most (if not all) modern Linux systems. Of course, x86_64 binaries will not work on ARM processors, but if you compile for an older x86_64 instruction-set, it will work on all modern x86_64 processors, thanks to the backward compatibility in processor instruction sets. ARM also benefits from a similar level of backward compatibility.

In summary, if the relevant Python interpreter, all the modules, and third party software are compiled and packaged with your Python code, it will be “portable”. We call this process “snake charming", since it prevents Python modules from biting one another. For interested developers, we provide detailed instructions in the InterARTIC GitHub on how snake charming was used in the development of *InterARTIC*, and how to use this technique to improve their own tools.



Queued Jobs

You currently have no queued jobs.

Prepare your InterARTIC job

*** = required**

Job name: * (info)

SARS_CoV_2_example

Select the input data directory for your experiment: * (info)

FLFL031920

This input contains: *

○ A single sample
● Multiple samples

Select the sample-barcode metadata file (CSV) for your experiment:
(Only needed if there are multiple samples)

FLFL031920-barcodes.csv

Output folder (optional): (info)
(if no output folder is provided, one will be created inside the input folder)

SARS_CoV_2_output

□ Override existing data?

Select the virus you want to analyse: * (info)

● SARS-CoV-2 (nCoV-2019)
○ IturiEBOV
○ Custom

Select your primer scheme: * (info)

○ Artic V1 (400bp)
○ Artic V2 (400bp)
○ Artic V3 (400bp)
● Eden V1 (2500bp)
○ Midnight V1 (1200bp)

Primer scheme directory: *

/mnt/c/Users/hasindu/Desktop/interartic_bin/primer-schemes/eden

Name of primer scheme: *
e.g. nCoV-2019/V3

nCoV-2019/V1

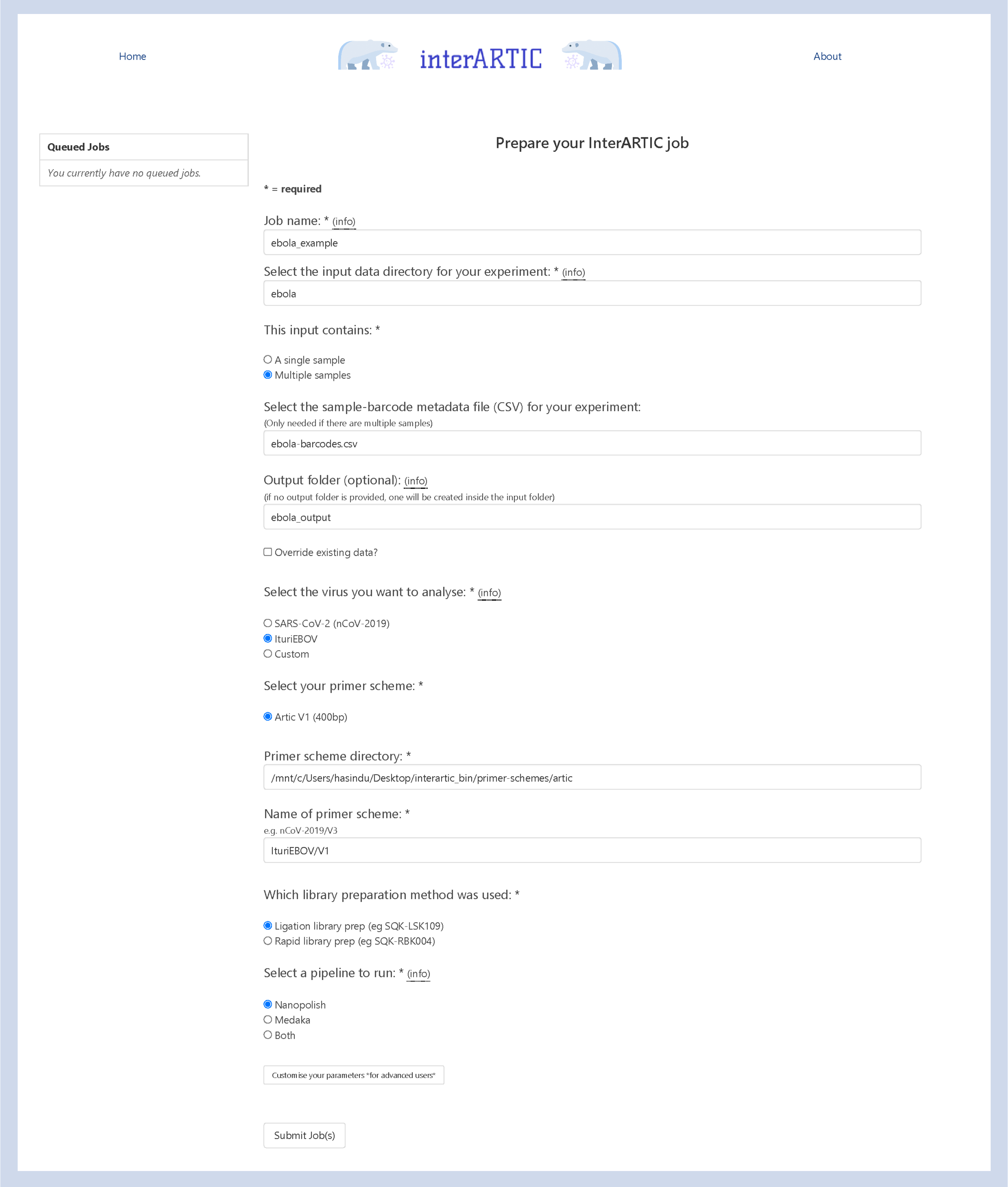
Which library preparation method was used: *

● Ligation library prep (eg SQK-LSK109)
○ Rapid library prep (eg SQK-RBK004)

Select a pipeline to run: * (info)

● Nanopolish
○ Medaka
○ Both

Supplementary Figure 1. Example job configuration for *InterARTIC* analysis of SARS-CoV-2 whole-genome sequencing.



Queued Jobs

You currently have no queued jobs.

Prepare your InterARTIC job

*** = required**

Job name: * (info)

ebola_example

Select the input data directory for your experiment: * (info)

ebola

This input contains: *

○ A single sample
● Multiple samples

Select the sample-barcode metadata file (CSV) for your experiment:
(Only needed if there are multiple samples)

ebola-barcodes.csv

Output folder (optional): (info)
(if no output folder is provided, one will be created inside the input folder)

ebola_output

□ Override existing data?

Select the virus you want to analyse: * (info)

○ SARS-CoV-2 (nCoV-2019)
● IturiEBOV
○ Custom

Select your primer scheme: *

● Artic V1 (400bp)

Primer scheme directory: *

/mnt/c/Users/hasindu/Desktop/interartic_bin/primer-schemes/artic

Name of primer scheme: *
e.g. nCoV-2019/V3

IturiEBOV/V1

Which library preparation method was used: *

● Ligation library prep (eg SQK-LSK109)
○ Rapid library prep (eg SQK-RBK004)

Select a pipeline to run: * (info)

● Nanopolish
○ Medaka
○ Both

Supplementary Figure 2. Example job configuration for *InterARTIC* analysis of Ebola whole-genome sequencing.