SEQUENCING METHODS AND SOPS: NEXT-GENERATION SEQUENCING AND BIOINFORMATICS (ILLUMINA AND NANOPORE)

Note - **Language and Databases:** Linux/Shell, NCBI, GISAID, Nextstrain, UCSC Genome Browser, Greengenes, RDP and SILVA.

SARS-CoV-2 Whole Genome Sequencing using 1200bp tiling on Oxford Nanopore MinION

AG Velavan
Institute of Tropical Medicine
Universität Tübingen
Wilhelmstr 27
72074 Tuebingen
Germany

Protocol updates

Version	Update summary
1.0	Complete draft protocol
1.1	Adding instruction for using Flongle Flow Cell
1.2	Adding additional Bioinformatics analysis steps, Adding instruction for using Rapid Barcoding Kit 96 (SQK-RBK110.96)

INDEX

1. Abbreviations

bp	base pairs
°C	degree Celsius
cDNA	complementary deoxyribonucleic acid
dNTP	deoxynucleotide triphosphate
µl	microlitres
µM	micromolar
ml	millilitres
NTC	non-template control
PCR	polymerase chain reaction
PC	positive control
RNA	ribonucleic acid
RT	reverse transcription
SARS-CoV-2	severe acute respiratory syndrome coronavirus 2

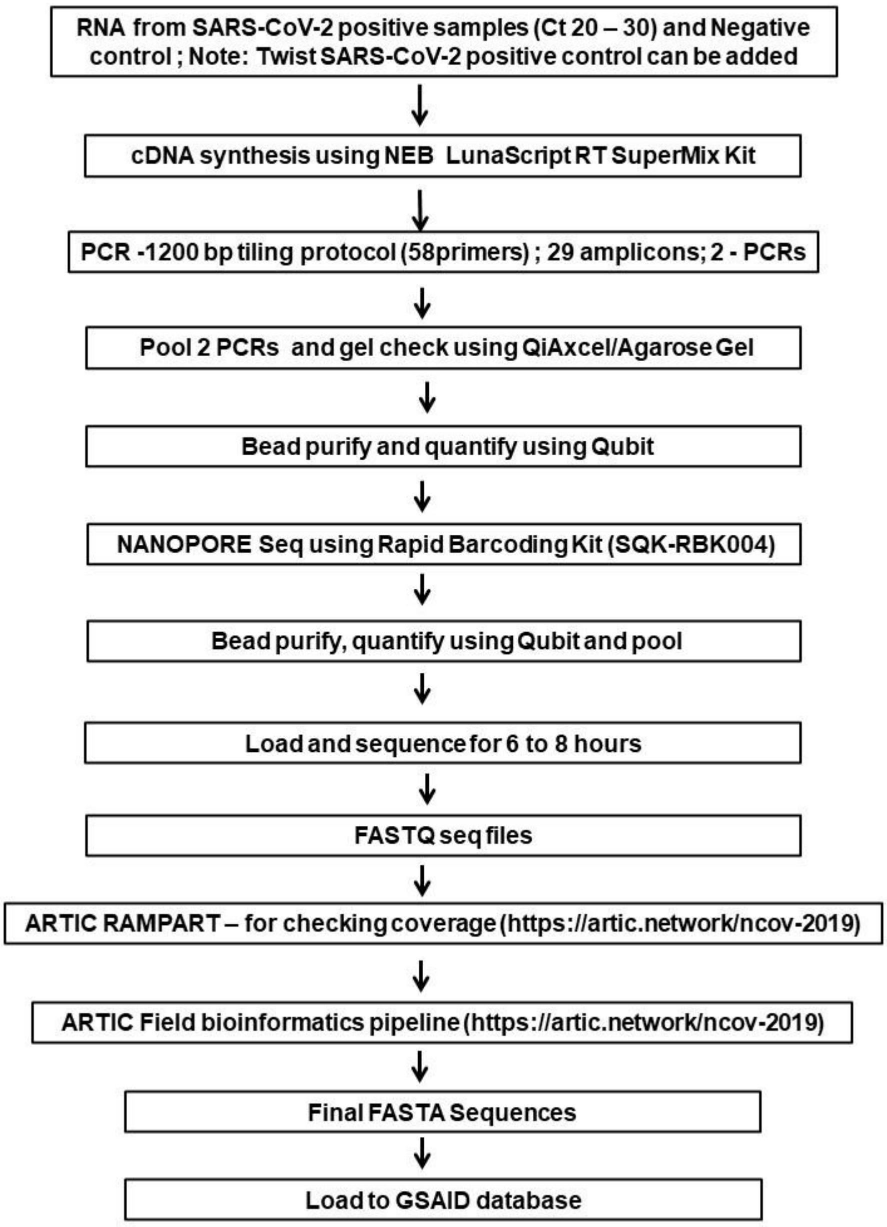
2. Purpose

Whole genome sequencing of SARS-CoV-2 for detecting and characterizing viral pathogens in clinical samples, supporting infection control, viral epidemiology and characterizing evolutionary viral responses to vaccines and treatments.

3. Principal of the assay

Viral RNA is transcribed and amplified by a multiplexed set of primers which generate a pool of 1200bp tiled amplicons. After PCR, samples are barcoded with the Oxford Nanopore Rapid Barcoding Kit. The Rapid Barcoding Kit using a transposase which simultaneously cleaves template molecules and attaches one of 12 unique barcoded tags to the cleaved ends. Barcoded samples are pooled, and Rapid Sequencing Adapters are then added to the tagged ends. Samples are sequenced on MinION device.

Flow diagram SARS-CoV-2 whole genome sequencing using Oxford MinION Nanopore


RNA from SARS-CoV-2 positive samples (Ct 20 – 30) and Negative control ; Note: Twist SARS-CoV-2 positive control can be added
cDNA synthesis using NEB LunaScript RT SuperMix Kit
PCR -1200 bp tiling protocol (58primers) ; 29 amplicons; 2 - PCRs
Pool 2 PCRs and gel check using QiAxcel/Agarose Gel
Bead purify and quantify using Qubit
NANOPORE Seq using Rapid Barcoding Kit (SQK-RBK004)
Bead purify, quantify using Qubit and pool
Load and sequence for 6 to 8 hours
FASTQ seq files
ARTIC RAMPART – for checking coverage (https://artic.network/ncov-2019)
ARTIC Field bioinformatics pipeline (https://artic.network/ncov-2019)
Final FASTA Sequences
Load to GSAID database

4. Material

Isolated SARS-CoV-2 RNA from swab material

5. Pre-analytics

5.1 RNA isolation and quantification

Nucleic acid extraction using the QIAamp Viral RNA Mini Kit (protocols – according to the instruction manual)

Whole genome sequencing is performed on samples which were tested SARS-CoV-2 positive by RT- qPCR. Ct value of viral RNA input should be between 20-30. If Ct is between 12-15, then dilute the sample 100-fold in water, if between 15-18 then dilute 10-fold in water. This will reduce the likelihood of PCR-inhibition. It is good practice to carry a negative control (e.g. water) through the entire process from cDNA preparation to sequencing.

Frozen, archived RNA is thawed in a cooling rack on a PCR workstation.

6. Equipment

- PCR thermo cycler
- Gel electrophoresis system
- Invitrogen™ Qubit™ 4 Fluorometer
- Invitrogen™ DynaMag™-2 Magnet
- MinION Mk1B Oxfore Nanopore
- PC like i7-10700KF / 64GB RAM / 1TB SSD
- Vortexer
- Tabletop centrifuge
- PCR workstation
- Pipettes set (P1000, P100, P10)

7. Reagent and Consumables

- LunaScript® RT SuperMix Kit (NEB, cat# E3010)
- SuperScript IV (Thermo Fisher Scientific, cat# 18090050)
- dNTP mix, 10mM each (Thermo Fisher Scientific, cat# R0192)
- RNase OUT (Thermo Fisher Scientific, cat# 10777019)
- Random Hexamers 50µM (Thermo Fisher Scientific, cat# N8080127)
- Agencourt Ampure XP Beads (Beckman Coulter, cat # A63880/A63881)
- dsDNA BR assay kit (Thermo Fisher Scientific, cat# Q32850)
- Nuclease Free water (Thermo Fisher Scientific, cat# AM9916)
- Q5 Hot Start High Fidelity Polymerase (NEB, cat# M0493S)
- Absoluter Ethanol (AppliChem, cat# A1613,2500PE)
- Rapid Barcoding kit (Oxford Nanopore Technologies, cat# SQK-RBK004)
- Flongle Sequencing Expansion (Oxford Nanopore Technologies, cat# EXP-FSE001)
- Flow Cell Priming Kit (Oxford Nanopore Technologies, cat# EXP-FLP002)
- SpotON Flow Cell R9.4.1 (Oxford Nanopore Technologies, cat# FLO-MIN106D)
- Flongle device - flow cell and adapter (Oxford Nanopore Technologies, cat# ADP-FLG001 and cat# FLO-FLG001

- Qubit assay tubes (Thermo Fisher Scientific, cat# Q32856)
- 1,5ml Eppendorf Safe-Lock (Eppendorf, cat# 0030120086)
- 0.2ml PCR tubes
- Disposable filter tips

8. Primers

PCR- POOL 1

Primer Name	Sequence	Start	Pool	Length	Tm
SARSCoV_1200_1_LEFT	ACCAACCAACTTTCGATCTCTTGT	30	1	24	60.69
SARSCoV_1200_1_RIGHT	GGTTGCATTCATTTGGTGACGC	1205	1	22	61.49
SARSCoV_1200_3_LEFT	GGCTTGAAGAGAAGTTTAAGGAAGGT	2153	1	26	61.19
SARSCoV_1200_3_RIGHT	GATTGTCCTCACTGCCGTCTTG	3257	1	22	61.5
SARSCoV_1200_5_LEFT	ACCTACTAAAAAGGCTGGTGGC	4167	1	22	60.55
SARSCoV_1200_5_RIGHT	AGCATCTTGTAGAGCAGGTGGA	5359	1	22	61.16
SARSCoV_1200_7_LEFT	ACCTGGTGTATACGTTGTCTTTGG	6283	1	24	60.8
SARSCoV_1200_7_RIGHT	GCTGAAATCGGGGCCATTTGTA	7401	1	22	61.53
SARSCoV_1200_9_LEFT	AGAAGTTACTGGCGATAGTTGTAATAACT	8253	1	29	60.59
SARSCoV_1200_9_RIGHT	TGCTGATATGTCCAAAGCACCA	9400	1	22	60.29
SARSCoV_1200_11_LEFT	AGACACCTAAGTATAAGTTTGTTCGCA	10343	1	27	60.74
SARSCoV_1200_11_RIGHT	GCCCACATGGAAATGGCTTGAT	11469	1	22	61.8
SARSCoV_1200_13_LEFT	ACCTCTTACAACAGCAGCCAAAC	12450	1	23	61.55
SARSCoV_1200_13_RIGHT	CGTCCTTTTCTTGGAAGCGACA	13621	1	22	61.38
SARSCoV_1200_15_LEFT	TTTTAAGGAATTACTTGTGTATGCTGCT	14540	1	28	60.06
SARSCoV_1200_15_RIGHT	ACACACAACAGCATCGTCAGAG	15735	1	22	61.12
SARSCoV_1200_17_LEFT	TCAAGCTTTTTGCAGCAGAAACG	16624	1	23	61.28
SARSCoV_1200_17_RIGHT	CCAAGCAGGGTTACGTGTAAGG	17754	1	22	61.19
SARSCoV_1200_19_LEFT	GGCACATGGCTTTGAGTTGACA	18596	1	22	61.91
SARSCoV_1200_19_RIGHT	CCTGTTGTCCATCAAAGTGTCCC	19678	1	23	61.62
SARSCoV_1200_21_LEFT	TCTGTAGTTTCTAAGGTTGTCAAAGTGA	20553	1	28	60.58
SARSCoV_1200_21_RIGHT	GCAGGGGGTAATTGAGTTCTGG	21642	1	22	60.95
SARSCoV_1200_23_LEFT	ACTTTAGAGTCCAACCAACAGAATCT	22511	1	26	60.18
SARSCoV_1200_23_RIGHT	TGACTAGCTACACTACGTGCCC	23631	1	22	61.52
SARSCoV_1200_25_LEFT	TGCTGCTACTAAAATGTCAGAGTGT	24633	1	25	60.51
SARSCoV_1200_25_RIGHT	CATTTCCAGCAAAGCCAAAGCC	25790	1	22	61.45
SARSCoV_1200_27_LEFT	TGGATCACCGGTGGAATTGCTA	26744	1	22	61.75
SARSCoV_1200_27_RIGHT	TGTTCGTTTAGGCGTGACAAGT	27894	1	22	60.74
SARSCoV_1200_29_LEFT	TGAGGGAGCCTTGAATACACCA	28677	1	22	61.1
SARSCoV_1200_29_RIGHT	TAGGCAGCTCTCCCTAGCATTG	29790	1	22	61.61

PCR- POOL 2

Primer Name	Sequence	Start	Pool	Length	Tm
SARSCoV_1200_2_LEFT	CCATAATCAAGACTATTCAACCAAGGGT	1100	2	28	61.27
SARSCoV_1200_2_RIGHT	ACAGGTGACAATTTGTCCACCG	2266	2	22	61.33
SARSCoV_1200_4_LEFT	GGAATTTGGTGCCACTTCTGCT	3144	2	22	61.66

SARSCoV_1200_4_RIGHT	CCTGACCCGGGTAAGTGGTTAT	4262	2	22	61.49
SARSCoV_1200_6_LEFT	ACTTCTATTAAATGGGCAGATAACAACTG	5257	2	29	60.18
SARSCoV_1200_6_RIGHT	GATTATCCATTCCCTGCGCGTC	6380	2	22	61.75
SARSCoV_1200_8_LEFT	CAATCATGCAATTGTTTTTCAGCTATTTTG	7298	2	30	60.39
SARSCoV_1200_8_RIGHT	TGACTTTTTGCTACCTGCGCAT	8385	2	22	61.39
SARSCoV_1200_10_LEFT	TTTACCAGGAGTTTTCTGTGGTGT	9303	2	24	60.32
SARSCoV_1200_10_RIGHT	TGGGCCTCATAGCACATTGGTA	10451	2	22	61.5
SARSCoV_1200_12_LEFT	ATGGTGCTAGGAGAGTGTGGAC	11372	2	22	61.48
SARSCoV_1200_12_RIGHT	GGATTTCCCACAATGCTGATGC	12560	2	22	60.48
SARSCoV_1200_14_LEFT	ACAGGCACTAGTACTGATGTCGT	13509	2	23	61.12
SARSCoV_1200_14_RIGHT	GTGCAGCTACTGAAAAGCACGT	14641	2	22	61.94
SARSCoV_1200_16_LEFT	ACAACACAGACTTTATGAGTGTCTCT	15608	2	26	60.18
SARSCoV_1200_16_RIGHT	CTCTGTCAGACAGCACTTCACG	16720	2	22	61.17
SARSCoV_1200_18_LEFT	GCACATAAAGACAAATCAGCTCAATGC	17622	2	27	62.03
SARSCoV_1200_18_RIGHT	TGTCTGAAGCAGTGGAAAAGCA	18706	2	22	60.68
SARSCoV_1200_20_LEFT	ACAATTTGATACTTATAACCTCTGGAACAC	19574	2	30	60.15
SARSCoV_1200_20_RIGHT	GATTAGGCATAGCAACACCCGG	20698	2	22	61.39
SARSCoV_1200_22_LEFT	GTGATGTTCTTGTTAACAACTAAACGAACA	21532	2	30	61.44
SARSCoV_1200_22_RIGHT	AACAGATGCAAATCTGGTGGCG	22612	2	22	62.03
SARSCoV_1200_24_LEFT	GCTGAACATGTCAACAACTCATATGA	23518	2	26	60.13
SARSCoV_1200_24_RIGHT	ATGAGGTGCTGACTGAGGGAAG	24736	2	22	61.74
SARSCoV_1200_26_LEFT	GCCTTGAAGCCCCTTTTCTCTA	25690	2	22	60.29
SARSCoV_1200_26_RIGHT	AATGACCACATGGAACGCGTAC	26857	2	22	61.5
SARSCoV_1200_28_LEFT	TTTGTGCTTTTTAGCCTTTCTGCT	27784	2	24	60.14
SARSCoV_1200_28_RIGHT	GTTTGGCCTTGTTGTTGTTGGC	29007	2	22	61.82

*Positions relative to MN908947/SARS-CoV-2/Wuhan-Hu-1

- Dilute this primer pool 1:10 in molecular grade water, to generate 10μM primer stocks. It is recommended that multiple aliquots of each primer pool are made to in case of degradation or contamination.

- If you have ordered each primer independently and need to generate primer pool stocks: add 5μL of each primer from Pool 1 to a 1.5mL Eppendorf labeled "Pool 1 (100μM)" and each primer from Pool 2 to a 1.5mL Eppendorf labelled "Pool 2 (100μM)". These are your 100μM stocks of each primer pool.

- Primers should be diluted and pooled in the mastermix cabinet which should be cleaned with decontamination wipes and UV sterilized before and after use.

9. Assay procedure

9.1. cDNA synthesis

cDNA synthesis using NEB LunaScript® RT SuperMix Kit is recommended since its protocol is quick and convenient. For samples with high Ct value (between 28-30), cDNA synthesis using Thermo Fisher kit is recommended for greater efficiency and coverage.

❖ ***cDNA synthesis using NEB LunaScript® RT SuperMix Kit (E3010)***

- Mix components briefly and spin down if necessary.
- Prepare cDNA synthesis reaction as described below:

Component	**1 reaction**
Template RNA	8 µL
LunaScript RT SuperMix (5X)	2 µL
Total	10 µL

- Incubate reactions in a thermocycler with the following steps:

CYCLE STEP	**TEMPERATURE**	**TIME**	**CYCLES**
Primer Annealing	25°C	2 minutes	1
cDNA Synthesis	55°C	20 minutes	
Heat Inactivation	95°C	1 minute	
Hold	4°C		

❖ ***cDNA synthesis using Thermo Fisher***

- Add the following components to a 0.2mL RNase-free tube on ice

Component	**1 reaction**
50µM random hexamers	1 µL
10mM dNTPs mix (10mM each)	1 µL
Template RNA	11 µL
Total	13 µL

- Anneal primers: Gently mix by pipetting and pulse spin down the tube. Incubate at 65°C for 5 minutes.
- Snap cool in a prechilled metal rack or on ice for 2 minutes.
- Prepare the mastermix in the mastermix cabinet as follow

Component	**1 reaction**	
SuperScript IV Buffer	4 µL	
100mM DTT	1 µL	
RNaseOUT RNase Inhibitor	1 µL	
SSIV Reverse Transcriptase	1 µL	
Total	20 µL	

- Add 20µL of the mastermix to the annealed template RNA (13µL) in a sample addition cabinet.
- Gently mix by pipetting and pulse spin the tube.

- Incubate the reaction in a preheated PCR machine:

CYCLE STEP	TEMPERATURE	TIME	CYCLES
Reverse transcribe RNA	42°C	50 minutes	1
Inactivate enzyme	70°C	10 minutes	
Hold	4°C		

Safe stopping point: Next for PCR amplification: Store at –20°C for up to one week, or –70°C for long term storage.

9.2 Multiplex PCR

- Thaw reagents: Thaw, mix, and briefly centrifuge each component before use.
- A PCR master mix for each pool should be made up in the master mix cabinet and aliquoted into 0.2mL PCR strip tubes. Tubes should be wiped down when entering and leaving the master mix cabinet.
- Prepare reaction mix for each sample as follow

Component	Pool 1	Pool 2
5X Q5 Reaction Buffer	5 µL	5 µL
10 mM dNTPs	0.5 µL	0.5 µL
Q5 Hot Start DNA Polymerase	0.25 µL	0.25 µL
Primer pool 1 or pool 2 (10µM)	1.1 µL	1.1 µL
Nuclease-free water	15.9 µL	15.9 µL
Total	22.5 µL	22.5 µL

- Add 2.5 µL cDNA to each tube and mix well by pipetting
- Pulse centrifuge the tubes to collect the contents at the bottom of the tube
- Set up the following program on a thermal cycler (3-step protocol)

Cycle step	Temperature	Time	Cycles
Initial denaturation	98°C	30 seconds	1
Denaturation Annealing and extension	98°C 65°C	15 seconds 5minutes	30
Hold	4°C	Indefinite	1
NOTE: Cycle number should be 25 for Ct 18-21 up to a maximum of 35 cycles for Ct 35. Typically use 30 cycles.			

9.3 Pooling

- At this stage, care should be taken with amplified PCR products. Only open tubes in a designated post-PCR workspace with equipment that is separate from areas where primers and master mixes are handled.
- Label a 1.5mL Eppendorf tube for each sample and combine the two pools the PCR reaction as follows:

Component	Volume
Pool 1 PCR reaction	25 µL
Pool 2 PCR reaction	25 µL
Total	50 µL

9.4 Analyzing on agarose gel electrophoresis

- Using 5µL of pooled PCR to check the appearance of 1200bp bands on 1.1% agarose gel

9.5 Purification by AMPure XP Bead

- Resuspend the AMPure XP beads by vortexing.
- Prepare fresh 80% ethanol in Nuclease-free water.
- Prepare 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl buffer.
- To the entire pooled sample (45µL - 50µL), add 25µL of resuspended AMPure XP beads, and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at RT.
- Briefly spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 200µl of freshly prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.
- Repeat the previous step.
- Spin down and place the tube back on the magnet. Pipette off any residual ethanol and briefly allow to dry.
- Remove the tube from the magnetic rack and resuspend pellet in 22µl of 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl. Incubate for 2 minutes at RT.
- Pellet the beads on a magnet until the eluate is clear and colorless.
- Remove and retain 20µL of eluate into a clean 1.5ml Eppendorf DNA LoBind tube.

9.6 Quantify the pool using Qubit dsDNA Assay

- Set up the required number 0.5mL Qubit assay tubes for standards and samples. Note: the standards require two tubes.
- Label tube lids. Do not label the side of the tube as this could interfere with the sample read.
- Prepare the Qubit working solution by diluting the Qubit dsDNA HS reagent 1:200 in Qubit dsDNA HS buffer. Use a clean plastic tube each time you prepare Qubit working solution. Do not mix the working solution in a glass container.
- Prepare sufficient Qubit working solution to accommodate all standards and samples.
- Add 190uL of Qubit working solution to each of the tubes used for standards.
- Add 10uL of each qubit standard to the appropriate tube, mix by vertexing 2-3 seconds.

- Add 197uL of Qubit working solution to each assay tube.
- Add 3uL each sample to the assay tubes, then mix by vertexing 2-3 seconds. The final volume in each tube should be 200uL.
- Allow all tubes to incubate at room temperature for 2 minutes.
- Sample concentration can now be measured on the Qubit Fluorometer.

9.7 Normalization

- Label a 0.2ml PCR tube for each sample
- Adjust the amount of DNA in the tube to be 100ng total per sample in 7.5μL molecular grade water. For example:

DNA conc. by Qubit (ng/ul)	Volume (ul) to have 100 ng	Molecular grade water (ul)	Total volume (ul)
100	1	6.5	7.5

- Use 1ul of the NTC, even if there is no detectable DNA in the PCR reaction

9.8 Barcoding and Concentrate Library

❖ **Using Rapid Barcoding Kit (SQK-RBK004)**

- Set up on ice the following reaction for each sample

Component	**Volume**
DNA amplicons from step 7 (100ng)	7.5 μL
Fragmentation Mix RB 01-12 (one for each sample)	2.5 μL
Total	10 μL

- Mix gently be flicking the tube, and spin down
- Incubate at 30°C for 1 minute and then at 80°C for 1 minute and briefly cool on ice.
- Pool all barcoded samples (max 12 samples), noting the total volume.
- Resuspend the AMPure XP beads by vortexing.
- Prepare fresh 80% ethanol in Nuclease-free water.
- Prepare 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl buffer.
- To the entire pooled barcoded sample, add an equal volume of resuspended AMPure XP beads, and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at RT.
- Spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 200μl of freshly prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.
- Repeat the previous step.
- Spin down and place the tube back on the magnet. Pipette off any residual 80% ethanol and briefly allow to dry.

- Remove the tube from the magnetic rack and resuspend pellet in 12μl of 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl. Incubate for 2 minutes at RT.
- Pellet the beads on a magnet until the eluate is clear and colourless.
- Remove and retain 10μl of eluate into a clean 1.5ml Eppendorf DNA LoBind tube.

Safe stopping point: Store at –20°C for up to one week if not planning to load on MinION.

- Add 1μl of RAP to 10μl of barcoded DNA (Total 11μL pool).
- Mix gently by flicking the tube, and spin down.
- Incubate the reaction for 5 minutes at RT.
- The prepared library is used for loading to the MinION flow cell. Store the library on ice until ready to load.

❖ **Using Rapid Barcoding Kit 96 (SQK-RBK110.96)**

- The Rapid Barcoding Kit 96 contains sufficient reagents for 12 pooled libraries using 24 barcodes (6 pooled libraries using 48 barcodes and 3 pooled libraries using 96 barcodes).
- Please refer to use all reagents in Rapid Barcoding Kit 96 (SQK-RBK110.96) including Rapid Adapter F (RAP-F), SPRI beads (SPRI), Sequencing Buffer II (SBII), Loading Beads II (LBII), Elution Buffer (EB), Flush Tether (FLT) and Flush Buffer (FB)
- Briefly centrifuge and place the barcodes plate on ice
- In 0.2 ml thin-walled PCR tubes or a PCR plate 96, mix the following. The Rapid Barcodes can be transferred using a multichannel pipette:

Component	**Volume**
DNA amplicons from step 9.7 (100ng)	7.5 μL
Rapid Barcodes (RB01-96, one for each sample)	2.5 μL
Total	10 μL

- Mix well by pipetting. Seal the plate and spin down in a centrifuge.
- Incubate the tubes or plate at 30°C for 1 minutes and then at 80°C for 1 minutes. Briefly put the tubes or plate on ice to cool.
- Pool all barcoded samples in your desired ratio, noting the total volume.
- Resuspend the SPRI beads by vortexing.
- To the entire pooled barcoded sample from Step 7, add an equal volume of resuspended SPRI beads and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at room temperature.
- Prepare at least 3 ml of fresh 80% ethanol in nuclease-free water.
- Spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 1.5 ml of freshly-prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.

- Repeat the previous step.Briefly spin down and place the tube back on the magnet. Pipette off any residual ethanol. Allow to dry for 30 seconds, but do not dry the pellet to the point of cracking.
- Remove the tube from the magnetic rack and resuspend the pellet in 15 µl Elution Buffer (EB). Incubate for 10 minutes at room temperature.
- Pellet the beads on a magnet until the eluate is clear and colourless.
- Remove and retain 15 µl of eluate into a clean 1.5 ml Eppendorf DNA LoBind tube.
- Remove and retain the eluate which contains the DNA library in a clean 1.5 ml Eppendorf DNA LoBind tube
- Dispose of the pelleted beads
- Transfer 11 µl of the sample into a clean 1.5 ml Eppendorf DNA LoBind tube.
- Add 1 µl of Rapid Adapter F (RAP F) to 11 µl of barcoded DNA
- Mix gently by flicking the tube, and spin down.
- Incubate the reaction for 5 minutes at room temperature.

9.9 Load to MinION and sequence

❖ **Priming and loading the SpotON Flow Cell R9.4.1**

- Set up the MinION flow cell and host computer, including MinKNOW software.
- Open the MinKNOW GUI from the desktop icon and establish a local connection.
- Inset a flow cell into MinION.
- Click “Check Flow Cells” at the bottom of the screen then click “Start test.” Check the number of active pores available. When the check is complete, it is reported in the Notification panel. Check to ensure it has enough pores for a good sequencing run (warranty for flow cells: 800 nanopores or above checked within 5 days of receipt).
- Thaw the Sequencing Buffer (SQB), Loading Beads (LB), Flush Tether (FLT) and one tube of Flush Buffer (FB) or Sequencing Buffer II (SBII), Loading Beads II (LBII) or Loading Solution (LS, if using), Flush Tether (FLT) and Flush Buffer (FB) if using Rapid Barcoding Kit 96 (SQK-RBK110.96) at room temperature
- Thoroughly mix the Sequencing Buffer (SQB) and Flush Buffer (FB) tubes by vortexing. Spin down the Flush Tether (FLT) tube, and return to ice.
- To prepare the flow cell priming mix, add 30 µl of thawed and mixed Flush Tether (FLT) to 1.17 ml of thawed and mixed Flush Buffer (FB), and mix by vortexing at room temperature.Open the lid of the nanopore sequencing device and slide the flow cell's priming port cover clockwise 90 degrees.
- Set a P1000 pipette to 200uL, insert the tip into the priming port, turn the wheel until the dial shows 220-230uL, or until you can see a small volume of buffer entering the pipette tip. Do not remove more than this.
- Visually check that there is continuous buffer from the priming port across the sensor array.

- Load 800uL of the priming mix into the flow cell via the priming port, avoiding the introduction of air bubbles. Wait for 5 minutes.
- Thoroughly mix the contents of the Loading Beads (LB) by pipetting.
- Prepare library for loading as follows:
  With library prepared by Rapid Barcoding Kit (cat# SQK-RBK004)

Component	**Volume (uL)**
Sequencing Buffer (SQB)	34
Loading Beads (LB) mixed immediately before use	25.5
Nuclease free water	4.5
DNA Library (11µL pool)	11
Total	75

With library prepared by Rapid Barcoding Kit 96 (SQK-RBK110.96)

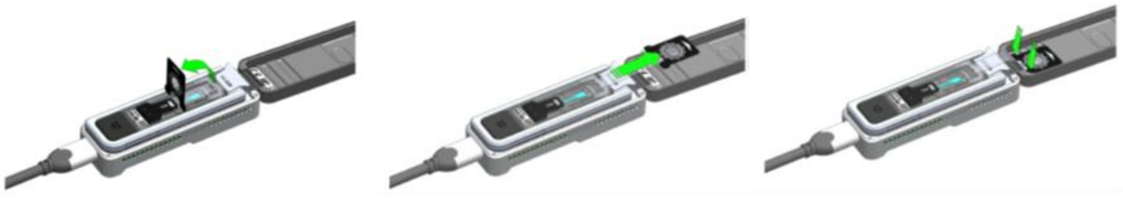
Component	**Volume (uL)**
Sequencing Buffer II (SBII)	37.5
Loading Beads II (LBII) mixed immediately before use	25.5
DNA Library (12 µL pool)	11
Total	75

- Gently lift the SpotON sample port cover to make the SpotON sample port accessible.
- Load 200µL of the priming mix into the flow cell via the priming port (not theSpotON sample port), avoiding the introduction of air bubbles.
- Mix the prepared library gently by pipetting up and down just prior to loading.
- Add 75uL of sample to the flow cell via the SpotON sample port in a dropwise fashion. Ensure each drop flows into the port before adding the next.
- Gently replace the SpotON sample port cover, making sure the bung enters the SpotON port, close the priming port and replace the MinION lid.
- Start the sequencing run using the MinKNOW software.

❖ **Priming and Loading the Flongle flow cell**

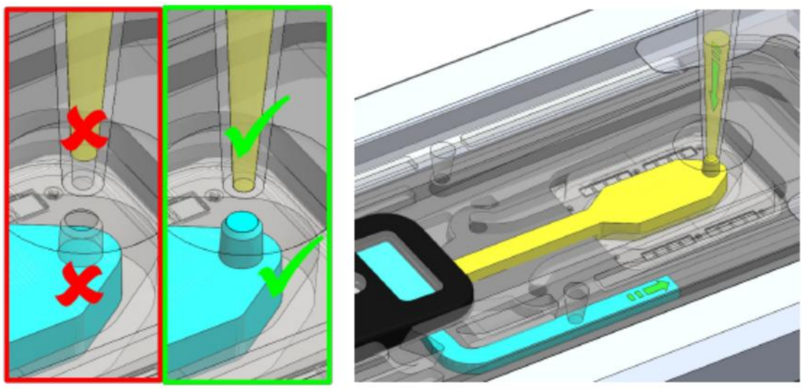
- Set up the MinION flow cell and host computer, including MinKNOW software.
- Open the MinKNOW GUI from the desktop icon and establish a local connection.
- Place the Flongle adapter into the MinION.
- The adapter should sit evenly and flat on the MinION Mk1B or GridION platform. This ensures the flow cell assembly is flat during the next stage.
- Place the flow cell into the Flongle adapter and press the flow cell down until you hear a click. Users should NOT touch the reverse side of the Flongle flow cell array or the contact pads on the Flongle adapter.

- The flow cell should sit evenly and flat inside the adapter, to avoid any bubbles forming inside the fluidic compartments.
- Click “Check Flow Cells” at the bottom of the screen then click “Start test.” Check the number of active pores available. When the check is complete, it is reported in the Notification panel. Check to ensure it has enough pores for a good sequencing run (warranty for flow cells: 50 nanopores or above checked within 5 days of receipt).
- Thaw the Sequencing Buffer II (SQBII), Loading Beads II (LBII), Flush Tether (FLT) and Flush Buffer (FB) at room temperature before placing the tubes on ice.
- Thoroughly mix by vortexing, spin down and return to ice.
- In a fresh 1.5ml Eppendorf DNA LoBind tube, mix 117μl of Flush Buffer (FB) with 3μl of Flush Tether (FLT) and mix by pipetting.
- Peel back the seal tab from the Flongle flow cell, up to a point where the sample port is exposed.



Lift up the seal tab; Pull the seal tab to open access to the sample port; Hold the seal tab open by using adhesive on the tab to stick to the MinION Mk 1B lid

- To prime your flow cell with the mix of Flush Buffer (FB) and Flush Tether (FLT) that was prepared earlier, ensure that there is no air gap in the sample port or the pipette tip. Place the P200 pipette tip inside the sample port and slowly dispense the priming fluid into the Flongle flow cell. To avoid flushing the flow cell too vigorously, load the priming mix by twisting the pipette plunger down.



- Vortex the vial of Loading Beads II (LBII). Note that the beads settle quickly, so immediately prepare the Sequencing Mix in a fresh 1.5ml Eppendorf DNA LoBind tube for loading the Flongle, as follows:

Component	**Volume (uL)**
Sequencing Buffer II (SBII)	15
Loading Beads II (LBII)	10
DNA Library	5
Total	30

- To add the Sequencing Mix to the flow cell, ensure that there is no air gap in the sample port or the pipette tip. Place the P100 tip inside the sample port and slowly dispense the Sequencing Mix into the flow cell by twisting the pipette plunger down.
- Seal the Flongle flow cell using the adhesive on the seal tab.
- Replace the sequencing platform lid.

10. Ending the experiment

When to STOP:

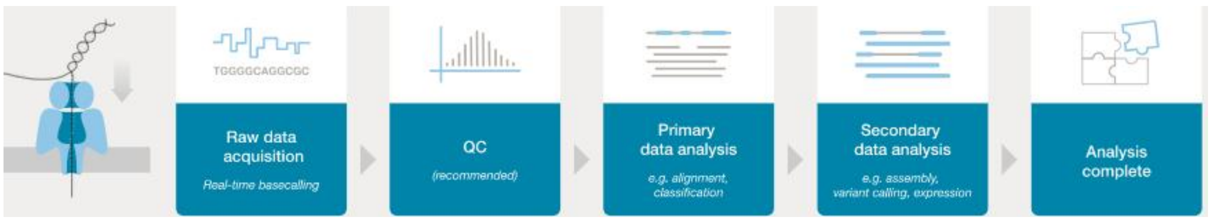
Depending on the variation in coverage of each amplicon, generally, you will need greater than 20,000 reads per sample to confidently assemble and call variants. This can typically be achieved on a MinION flow cell in four to six hours when running 12 samples using Fast Base Calling.

After your sequencing experiment is complete, if you would like to reuse the flow cell, please follow the Wash Kit instructions, and store the washed flow cell at 2-8°C, OR

Follow the returns procedure by washing out the flow cell ready to send back to Oxford Nanopore.

11. Basic bioinformatic for data analysis

MinKNOW, the operating software that drives nanopore sequencing devices, carries out several core tasks, including data acquisition, real-time analysis and feedback, local basecalling, and data streaming. MinKNOW produces FAST5 (HDF5) files and/or FASTQ files, based on preferences. FAST5 files contain raw signal data that can be used for basecalling. In our work, we obtain raw data in FASTQ format that contain nucleotide sequence data and their corresponding quality scores. The entire data analysis can be carried in 5 steps starting from raw data acquisition from MinKNOW.



First step of the analysis is basecalling which is usually completed by MinKNOW software. This step includes barcoding/demultiplexing, adapter trimming and alignment. For the rest of the analysis, you need to use bioinformatics workflows on a Linux/Mac computer. You are advised to watch some bioinformatics lectures on YouTube to get a good insight on these workflows. One such easy to follow lecture series from Dr. Simon Cockell of University of Newcastle, UK can be viewed at

https://www.youtube.com/c/SimonCockell/videos

In addition, keeping in view of your upcoming training you are also suggested to watch one recorded video on Nanopore data analysis. This lecture focusses on how to analyse raw data obtained from MinKNOW software using bioinformatics pipelines.

https://nanoporetech.com/resource-centre/bioinformatics-workflows-sars-cov-2-raw-nanopore-reads-consensus-genomes-using

Besides the above, some basic knowledge in Linux, Miniconda, Docker and GitHub are essential. For editing the code, Visual studio code is also a nice to have software that be easily installed on any OS. Linux and Mac operating systems are ideal for bioinformatics. However, if your PC has Windows 10 as OS, you could still install Linux and share all your data between the two operating systems without having to reboot the PC.

If you are a beginner in bioinformatics without any prior knowledge in Linux or any programming language, you can still do the data analysis, as the bioinformatics workflows for Nanopore are meant to cater the needs of biologists who wants to do sequence analysis by themselves.

11.1 Linux

Linux is an open-source operating system that is very secure, fast and reliable. Linux is most preferred for bioinformatics workflow as it can easily handle large datasets.

Installation of Linux on Windows 10 PC or laptop

On a Windows 10 PC or laptop, Linux can be installed as WSL-2 (Windows-subsystem for Linux, version 2). WSL is freely downloadable from Microsoft store on Windows 10 PC.

IMPORTANT: System should be x64 systems: Windows Version 1903 or higher, with Build 18362 or higher.

The information on Windows build can be found at
Start > Settings > System > About (under windows specifications)

More details on installation of WSL-2 can be read at the following URL

https://docs.microsoft.com/en-us/windows/wsl/install-win10#manual-installation-steps

Users are advised to follow step-by-step instructions under the section "Install WSL & update to WSL-2" at the above website. When asked for a username, you may give a short name or initials, but without any spaces. When you type your password, you may not see cursor moving, but Linux stores your password. You need to confirm the password by typing again. For beginners in Linux, either Ubuntu 18.04 or 20.04 is ideal as there is lot of reading material on internet. WSL does not have a graphical user interface (GUI) and hence some basic commands are essential for easy manoeuvre at the terminal prompt. A very good intro to Linux can be found at

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

There is lot of literature on Ubuntu on internet. For a start, following websites are useful from bioinformatics perspective.

1. http://www.cellbiol.com/bioinformatics_web_development/chapter-2-the-linux-operating-system-setting-up-a-linux-web-server/basic-linux-shell-commands/

2. https://bioinformaticsworkbook.org/Appendix/Unix/UnixCheatSheet.html#gsc.tab=0

11.2 Conda and Miniconda

Conda is an open-source package management and environment management system that runs on any OS. It has been created for Python programs, although it can package and distribute software for any language. Conda as a package manager helps you find and install packages. You will learn more about this package manager as you use it regularly.

Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others. To know more about Conda / miniconda, please refer to the following website.

https://docs.conda.io/projects/conda/en/latest/

Install Miniconda on WSL

At Ubuntu terminal, you need to type the following commands at $ prompt

$ wget https://repo.continuum.io/miniconda/ Miniconda3-py39_4.9.2-Linux-x86_64.sh

Click Enter button. The above command will download the miniconda from internet.

$ bash Miniconda3-py39_4.9.2-Linux-x86_64.sh

The above command will install the miniconda application on your Linux.

If your system is Mac,

$ wget https://repo.continuum.io/miniconda/Miniconda3-py39_4.9.2-MacOSX-x86_64.sh

Followed by

$ bash Miniconda3-py39_4.9.2-MacOSX-x86_64.sh

11.3 Docker

Docker is an open-source software platform to build, deploy or execute applications by using containers. As the name indicates, containerization isolate applications with each container having a separate environment at the same time sharing the underlying OS kernel. Docker provides a good platform to run multiple applications at the same time, each needing its own environment. Containerization is very similar to virtual machines, but unlike VMs, containers are faster and use less system resources. Each Docker container has an image file to create a run-time environment for the supposed application.

Installing Docker

Docker can be easily installed on WSL2 based windows by following instructions as mentioned in the below URL

https://docs.docker.com/docker-for-windows/wsl/

11.4 GitHub

GitHub is an Open-source Community where people around the globe work together on Open-source projects and make contributions. GitHub is also a code hosting platform for collaboration and version control. Importantly, GitHub is a repository for bioinformatics pipelines, and in our work, we use GitHub to download Nanopore workflows.

For more information and basic commands on GitHub, please refer to

https://guides.github.com/introduction/git-handbook/

11.5 Visual Studio Code

This editor can be downloaded freely from **https://code.visualstudio.com** depending on the type of OS. This is very useful software to edit the code, besides providing an excellent platform to integrate linux terminal, Git and docker.

12. Nanopore MinION data analysis

In our program, we use NGS pipelines developed by Artic Network project. These workflows can be modified to suit the analysis of Ebola, SARS-CoV-2, Influenza viruses etc. For further details on ARTIC Network, please refer to

https://artic.network/1-about.html

12.1 Quality control of sequencing data using RAMPART

For running RAMPART, you need to have Docker installed on your PC. For further details on this pipeline, you are advised to read the information at the following website

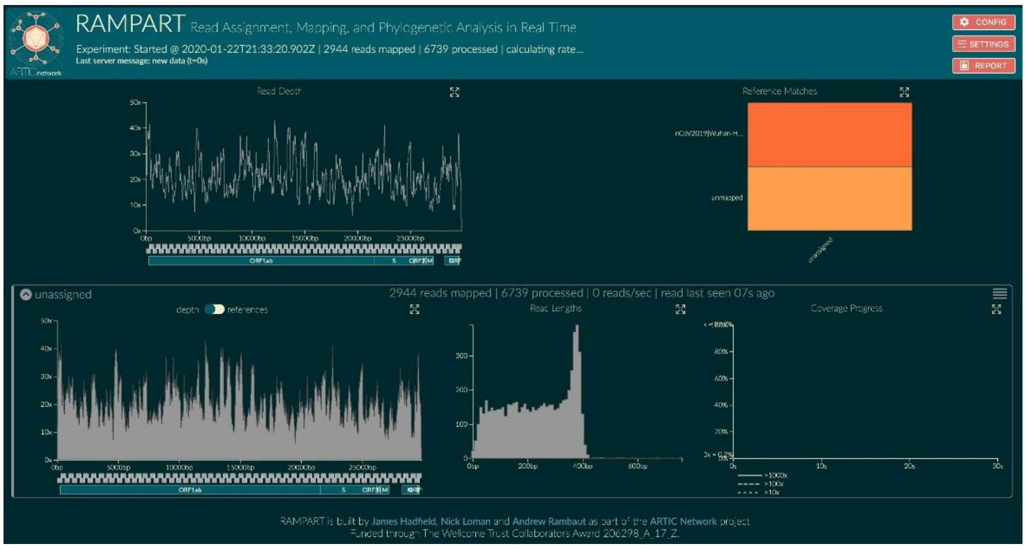
https://hub.docker.com/r/ontresearch/artic_rampart

For using RAMPART, at Ubuntu terminal, you must navigate to the directory that contained the FASTQ files collected from MinKNOW. Then type in the following commands at the $ prompt

```
$ docker pull ontresearch/artic_rampart
```

```
$ docker run -it -e LOCAL_USER_ID=`id -u $USER` --mount
type=bind,source="$(pwd)",target=/data -p 3000:3000 -p 3001:3001
ontresearch/artic_rampart:latest
```

Open a browser and type localhost:3000 to visualize the results. Typically, successful run should return a picture similar to the one below


RAMPART Read Assignment, Mapping, and Phylogenetic Analysis in Real Time

Experiment: Started @ 2020-01-22T21:33:20.902Z | 2944 reads mapped | 6739 processed | calculating rate...
Last server message: new data (t~0s)

2944 reads mapped | 6739 processed | 0 reads/sec | read last seen 07s ago

RAMPART is built by James Hadfield, Nick Loman and Andrew Rambaut as part of the ARTIC Network project
Funded through The Wellcome Trust Collaborators Award 206298_A_17_Z.

This docker image is developed and updated regularly by ONT (Oxford Nanopore Technologies) Research. Care should be taken that the latest image is being pulled from docker hub.

The above pipeline will process sequence reads that were basecalled by MinKNOW. The program will look for. FASTQ files in the specified path, de-multiplex the reads, looking for any barcodes that were used when creating the sequencing library. Subsequently, the reads are mapped to a set of reference genomes. Finally the aligned reads are visualised in the browser using **https://localhost:3000**.

De-multiplexing is carried by porechop. This program is an adapter trimming and demultiplexing package. If the demultiplexing is already done by Guppy, then this step can be skipped. For more details on porechop, the original source code and information can be accessed at https://github.com/rrwick/Porechop

The second step is reference mapping, and this is done by minimap2. This program performs a pairwise alignment for a given FASTA sequence and corresponding reference. Additional information on minimap2 program could be found at https://github.com/lh3/minimap2.

The entire RAMPART could be carried in real-time as and when reads are obtained from MinKNOW program. However, this analysis is quiet demanding for PC and requires high-end configuration with high RAM (>64gb) additional GPU (graphics processing unit)

12.2 Field-Bioinformatics

This pipeline will be carried out on the FASTQ files obtained after demultiplexing. This includes read filtering, primer trimming, amplicon coverage normalization, variant calling and consensus building.

Installing the pipeline
(Adapted from https://github.com/artic-network/fieldbioinformatics)

1. Downloading the source:

```
$ git clone https://github.com/artic-network/fieldbioinformatics
$ cd fieldbioinformatics
```

2. Install dependencies:

The artic pipeline has several software dependencies, and these could be installed by executing the below .yml file.

```
$ conda env create -f environment.yml
$ conda activate artic
```

3. Install the pipeline:

```
$ python setup.py install
```

4. Test the pipeline:

First check the pipeline can be called.

```
$ artic -v
```

5. Finally, execute the pipeline tests:

```
$ ./test-runner.sh medaka
```

The above workflow runs medaka tool that is specific to analyse Nanopore sequencing data. To adapt the workflow to the analysis of SARS-CoV-2 NGS add our primerschemes and the modified script (run1.sh) to the fieldbioinformatics-folder. Change the in- and output definitions in the script for each run.

The workflow principally includes samtools and minimap2.The SAMtools is an acronym for sequence alignment/map and this software parses sequences and returns output in SAM/BAM format. The BAM is binary alignment/map is a compressed version. This package is used in conjunction with BCFtools to generate consensus sequence and variant calling. The consensus sequence is stored in fasta format and variants in a file with .vcf extension.

12.3 High accuracy basecalling with guppy

High accuracy basecalling with guppy needs high computing capacities and should only be run on computers with additional graphic-card (GPU). System requirements can be checked here:

https://community.nanoporetech.com/protocols/Guppy-protocol/v/gpb_2003_v1_revx_14dec2018/guppy-software-overview

You have to have the newest update of your GPU-driver installed, check for updates either at the nvidia-homepage or under “Software&Updates” in your linux-menu. With the command below you can see which driver-version you are using.

```
$ nvidia-smi
```

Make sure to download a GPU-version of the guppy-software or you can go via the Oxford Nanopore's deb repository to load the newest version:

1. Add Oxford Nanopore's deb repository to your system (this is to install Oxford Nanopore Technologies-specific dependency packages):

```
$ sudo apt-get update

$ sudo apt-get install wget lsb-release

$ export PLATFORM=$(lsb_release -cs)

$ wget -O- https://mirror.oxfordnanoportal.com/apt/ont-repo.pub | sudo apt-key add -
$ echo "deb http://mirror.oxfordnanoportal.com/apt ${PLATFORM}-stable non-free" |
$ sudo tee /etc/apt/sources.list.d/nanoporetech.sources.list
```

```
$ sudo apt-get update
```

2. To install the .deb for Guppy, use the following command:

```
$ sudo apt update
```

```
$ sudo apt install ont-guppy
```

Type the command below to see if your installation was successful, if it was, you should see the guppy-manual

```
$ guppy_bascaller
```

To run guppy use the following command, change the input and outputpath according to your data

```
$ guppy_basecaller –input_path ~/data/fast5_pass/barcode01 –save_path ~/data/fast5/barcode01 –flowcell FLO-MIN106 –kit SQK-RBK004 -x "cuda:0"
```

The output will be fastq files, which have to go through the fieldbioinformatics-workfolw to get fasta files.

If you get "[guppy/error] main: GPU mode not available in this build" try to update guppy with the following commands:

```
$ sudo apt install guppy
```

```
$ sudo apt get-update
```

12.4 Information on nomenclatures, variants, mutations, outbreaks about SARS-CoV-2

Nomenclatures

I. GISAID Clades: (e.g. L, V, S, O, G, GH, GR, GV, GRY)

II. NeXTstrain Clades: Major clades: 19A, 19B, and 20A–20I

III. PANGOLIN Lineages: Phylogenetic Assignment of Named Global Outbreak Lineages

Following are important web links that gives information on: Global/Country/State/City to local, Circulating variants, New viral lineages, Variants of interest, under investigation, of concern, Distribution over time and space, No of mutation accumulating and Much more…….

https://cov-lineages.org/resources.html

https://nextstrain.org/ncov/gisaid/global

https://outbreak.info/

To compare between GISAID, Clade, Pangolin and WHO nomenclature check here: https://www.who.int/en/activities/tracking-SARS-CoV-2-variants/

12.4.1 GISAID

https://www.gisaid.org/

GISAID is a platform to share data from all influenza viruses and the coronavirus causing COVID-19. This includes genetic sequence and related clinical and epidemiological data associated with human viruses, and geographical as well as species-specific data associated with avian and other animal viruses. Every time you analyse a new sample you should upload the sequence as quickly as possible to the platform.

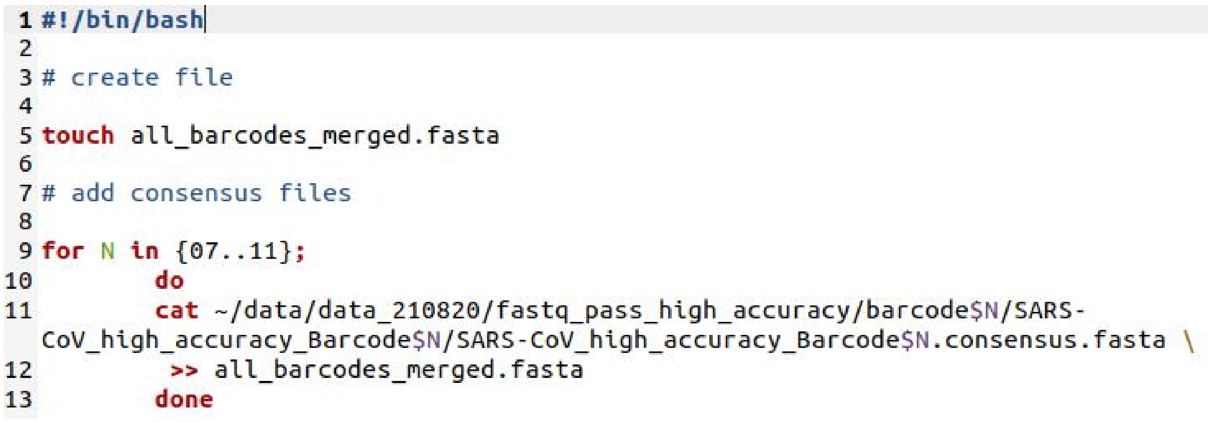
You can single upload one sequence at the time or batch-upload as described below:

https://www.gisaid.org/epiflu-applications/submitting-data-to-epiflutm/

12.4.2 Nextclade and Pangolin to identify the variants and lineages from the sequences

Nextclade and Pangolin are webtools that identify differences between your sequence and a reference sequence and assign the sequences to described variants.

To prepare your data for this you have to merge all fasta files, all barcodes, to one. You can do this using the cat-command multiple times or the bash-script below.

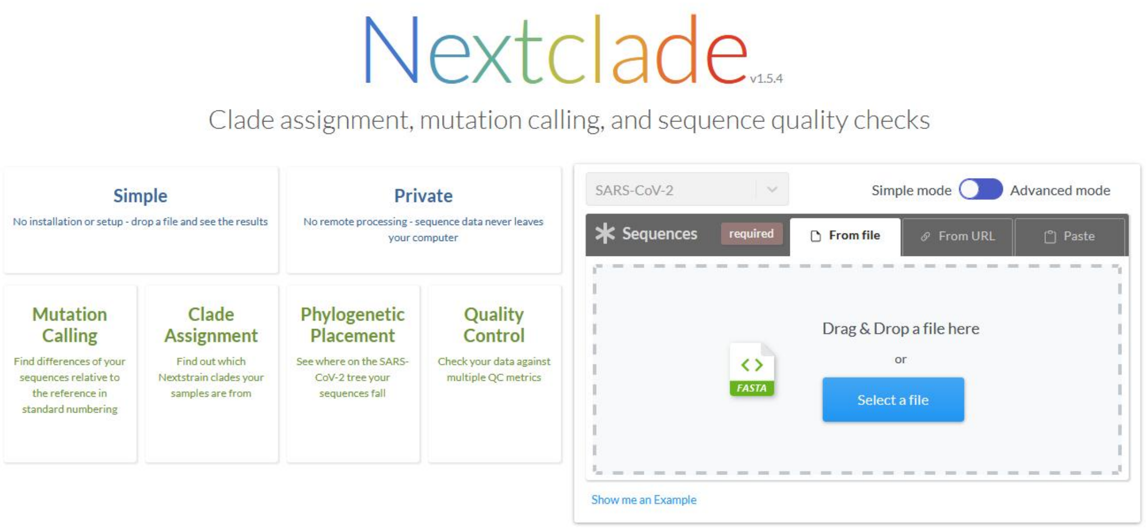


```
#!/bin/bash

# create file

touch all_barcodes_merged.fasta

# add consensus files

for N in {07..11};
        do
        cat ~/data/data_210820/fastq_pass_high_accuracy/barcode$N/SARS-CoV_high_accuracy_Barcode$N/SARS-CoV_high_accuracy_Barcode$N.consensus.fasta \
         >> all_barcodes_merged.fasta
        done
```

You will have to modify the script and put the path to your fasta files.

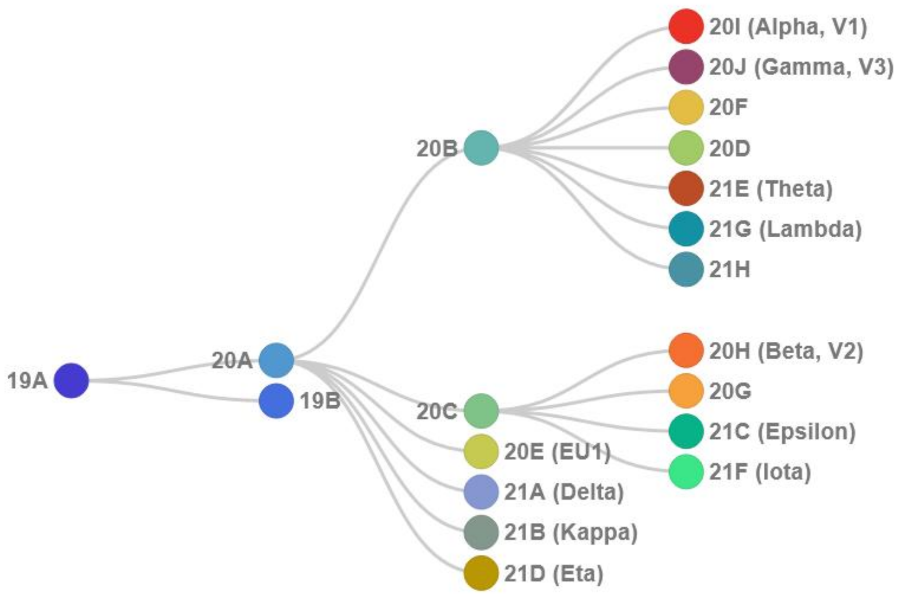
- **Nextclade:**

https://clades.nextstrain.org/results


Nextclade v1.5.4
Clade assignment, mutation calling, and sequence quality checks
Simple
No installation or setup - drop a file and see the results
Private
No remote processing - sequence data never leaves your computer
Mutation Calling
Find differences of your sequences relative to the reference in standard numbering
Clade Assignment
Find out which Nextstrain clades your samples are from
Phylogenetic Placement
See where on the SARS-CoV-2 tree your sequences fall
Quality Control
Check your data against multiple QC metrics
SARS-CoV-2
Simple mode
Advanced mode
Sequences
required
From file
From URL
Paste
Drag & Drop a file here
or
Select a file
FASTA
Show me an Example

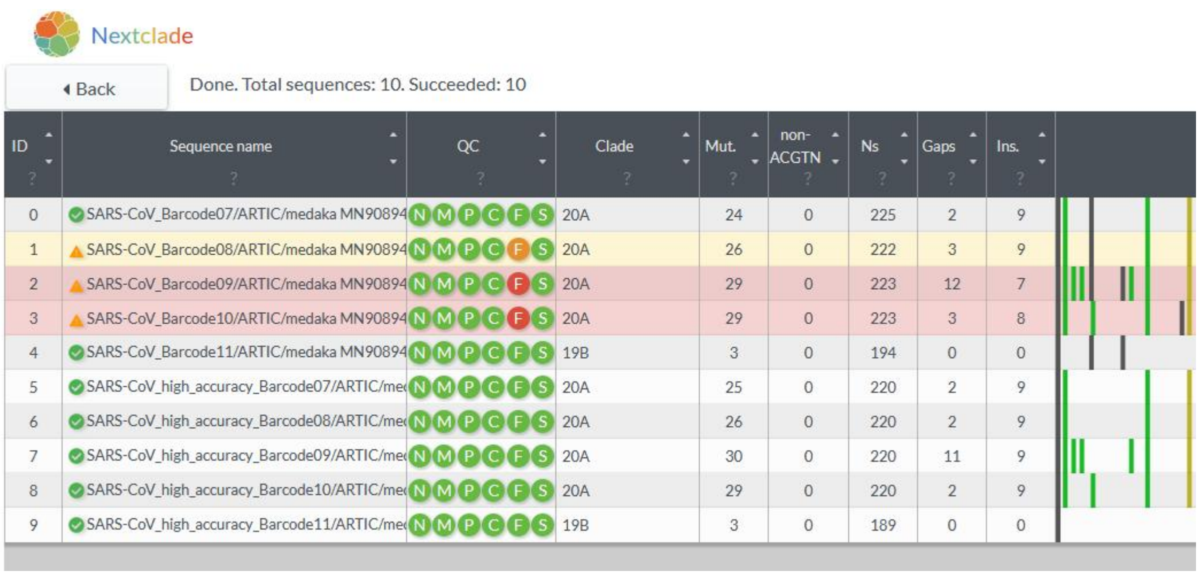
Drag the merged fasta file.

Nextclade groups variants into clades defined by specific signature mutations.



18 major clades were defined on 30AUG2021

After dragging the file, nextclade will automatically start to analyse the data and will show you the results as below.



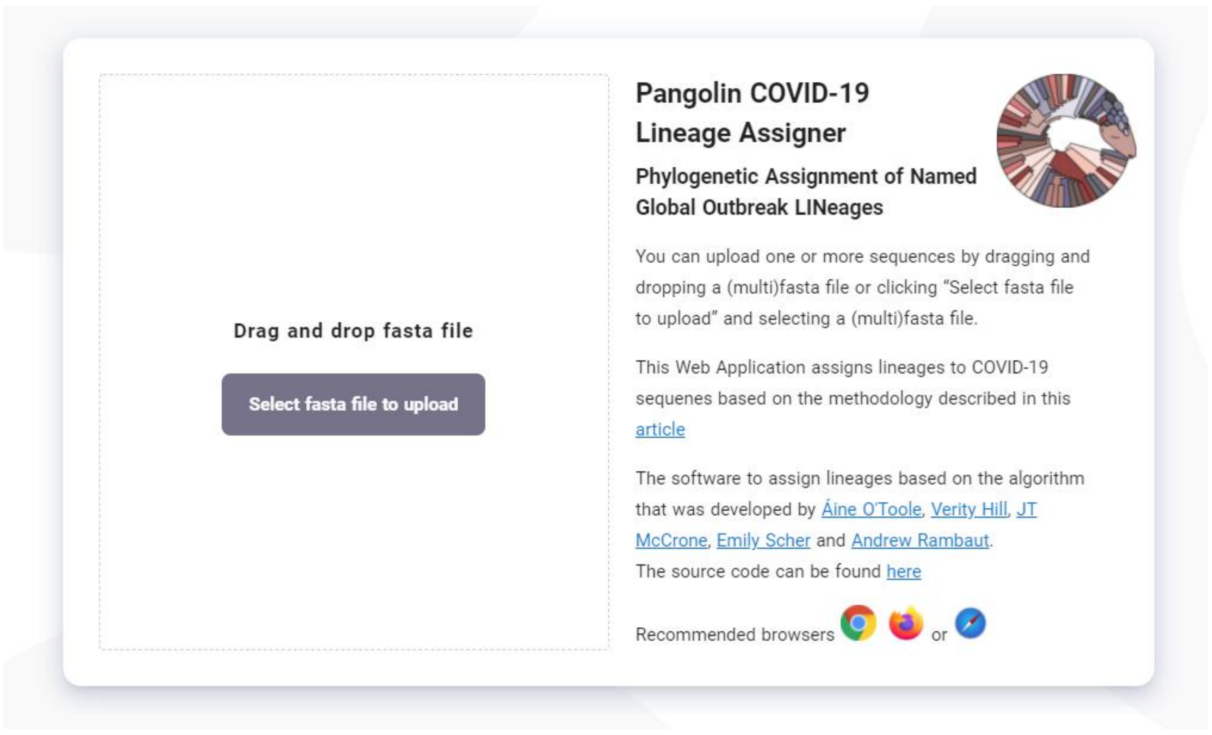
Done. Total sequences: 10. Succeeded: 10

ID	Sequence name	QC	Clade	Mut.	non-ACGTN	Ns	Gaps	Ins.
0	SARS-CoV_Barcode07/ARTIC/medaka MN90894	N M P C F S	20A	24	0	225	2	9
1	SARS-CoV_Barcode08/ARTIC/medaka MN90894	N M P C F S	20A	26	0	222	3	9
2	SARS-CoV_Barcode09/ARTIC/medaka MN90894	N M P C F S	20A	29	0	223	12	7
3	SARS-CoV_Barcode10/ARTIC/medaka MN90894	N M P C F S	20A	29	0	223	3	8
4	SARS-CoV_Barcode11/ARTIC/medaka MN90894	N M P C F S	19B	3	0	194	0	0
5	SARS-CoV_high_accuracy_Barcode07/ARTIC/me	N M P C F S	20A	25	0	220	2	9
6	SARS-CoV_high_accuracy_Barcode08/ARTIC/me	N M P C F S	20A	26	0	220	2	9
7	SARS-CoV_high_accuracy_Barcode09/ARTIC/me	N M P C F S	20A	30	0	220	11	9
8	SARS-CoV_high_accuracy_Barcode10/ARTIC/me	N M P C F S	20A	29	0	220	2	9
9	SARS-CoV_high_accuracy_Barcode11/ARTIC/me	N M P C F S	19B	3	0	189	0	0

❖ **Pangolin:**

Pangolin is similar to Nextclade but uses a different methodology and nomenclature for variant lineages.

How to use Pangolin? See: https://pangolin.docs.cog-uk.io/



Pangolin COVID-19 Lineage Assigner

Phylogenetic Assignment of Named Global Outbreak LINeages

You can upload one or more sequences by dragging and dropping a (multi)fasta file or clicking "Select fasta file to upload" and selecting a (multi)fasta file.

This Web Application assigns lineages to COVID-19 sequenes based on the methodology described in this article

The software to assign lineages based on the algorithm that was developed by Áine O'Toole, Verity Hill, JT McCrone, Emily Scher and Andrew Rambaut.
The source code can be found here

Recommended browsers or

Loading fasta files for analysis: https://pangolin.cog-uk.io/

All pango-lineages: https://cov-lineages.org/lineage_list.html

12.5 Phylogeny

For phylogeny you can use a program called **MegaX** https://megasoftware.net/ which comes with a graphical user interface. There is a free version which allows all important steps, for more features you can buy a full version. But MegaX is rather slow, especially for large numbers of samples.

Alternatively there are some command line tools.

MAFFT https://mafft.cbrc.jp/alignment/software/ is a multiple sequence alignment program. It can be installed via the command line

```
$ sudo apt install mafft
```

To run mafft type:

`$ mafft all_barcodes_merged.fasta > all_barcodes_alinged.fasta` (input > output)

IQTREE http://www.iqtree.org/ is a phylogeny program. It can be installed via the command line.

```
$ sudo apt install iqtree
```

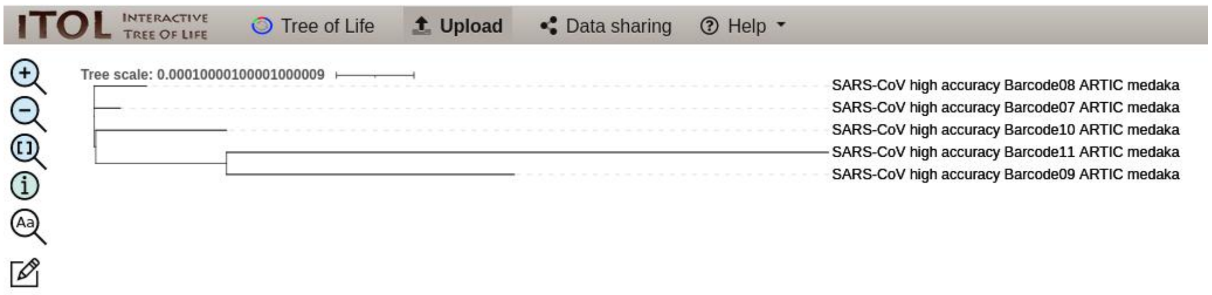
To run type:

```
$ iqtree -s all_barcodes_alinged.fasta -bb 1000
```

"-bb 100" is to set the ultrafast bootstrap (UFBoot) feature to 1000 replicates which will give more unbiased values.

To visualize your phylogenetic analysis you can load the treefile of the iqtree-output to the **ITOL-website**: https://itol.embl.de/upload.cgi

A tree will automatically be generated and can then be modified.


ITOL INTERACTIVE TREE OF LIFE
Tree of Life
Upload
Data sharing
Help
Tree scale: 0.00010000100001000009
SARS-CoV high accuracy Barcode08 ARTIC medaka
SARS-CoV high accuracy Barcode07 ARTIC medaka
SARS-CoV high accuracy Barcode10 ARTIC medaka
SARS-CoV high accuracy Barcode11 ARTIC medaka
SARS-CoV high accuracy Barcode09 ARTIC medaka

13. References

1. Geoghegan JL, Ren X, Storey M, Hadfield J, Jelley L, Jefferies S, Sherwood J, Paine S, Huang S, Douglas J, Mendes FK, Sporle A, Baker MG, Murdoch DR, French N, Simpson CR, Welch D, Drummond AJ, Holmes EC, Duchêne S, de Ligt J. Genomic epidemiology reveals transmission patterns and dynamics of SARS-CoV-2 in Aotearoa New Zealand. Nat Commun. 2020 Dec 11;11(1):6351. doi: 10.1038/s41467-020-20235-8. PMID: 33311501.

2. Nikki Freed, Olin Silander 2020. nCoV-2019 sequencing protocol (RAPID barcoding, 1200bp amplicon). https://www.protocols.io/view/ncov-2019-sequencing-protocol-rapid-barcoding-1200-bgggjttw?version_warning=no&step=1

3. Nikki E Freed, Markéta Vlková, Muhammad B Faisal, Olin K Silander, Rapid and inexpensive whole-genome sequencing of SARS-CoV-2 using 1200 bp tiled amplicons and Oxford Nanopore Rapid Barcoding, Biology Methods and Protocols, Volume 5, Issue 1, 2020, bpaa014, https://doi.org/10.1093/biomethods/bpaa014

4. https://store.nanoporetech.com/eu/rapid-barcoding-kit.html

5. https://help.geneious.com/hc/en-us/articles/360045070991-Assembly-of-SARS-CoV-2-genomes-from-tiled-amplicon-Illumina-sequencing-using-Geneious-Prime
6. Nanopore Protocol, Rapid Barcoding Kit 96 (SQK-RBK110.96) Version: RBK_9126_v110_revD_24Mar2021
7. https://artic.network/ncov-2019/ncov2019-bioinformatics-sop.html

Bacterial Whole Genome Sequencing using long-read/Hybrid Assembly

AG Velavan
Institute of Tropical Medicine
Universität Tübingen
Wilhelmstr 27
72074 Tuebingen
Germany

Protocol updates

Version	**Update summary**
1.0	Complete draft protocol

INDEX:

1 Abbreviations

bp	base pairs
°C	degree Celsius
DNA	deoxyribonucleic acid
dNTP	deoxynucleotide triphosphate
µl	microliters
µM	micromolar
ml	Milliliters
NTC	non-template control
PCR	polymerase chain reaction
PC	positive control
ONT	Oxford Nanopore Technology
RNA	ribonucleic acid
RT	reverse transcription

2 Purpose

Complete bacterial genome is archived for downstream analysis including species identification, phylogeny, resistance and virulence determinants, capsular types, plasmids, insertion sequences, phage regions,… by using bioinformatic tools.

3 Principal of the assay

A same bacterial DNA isolate is used to obtain both short reads (Illumina reads) and long reads (ONT reads). Short reads set is created by Illumina DNA Prep Kit which uses an enzyme to fragment the DNA while Rapid Barcoding kit and MinION is used to make long reads. Hybrid assembly is performed on both long and short reads. To archive a high-quality, complete genome, a deep (>100×) long-read set and a medium (~50×) short-read set is recommended. Sequencing can be multiplexed by using barcode per sample.

4 Material

Isolated genomic bacterial DNA

5 Pre-analytics

DNA isolation and quantification

Nucleic acid extraction using the Monarch® Genomic DNA Purification Kit (protocols – according to the instruction manual)

DNA Purity: This protocol is optimized for DNA with 260/280 absorbance ratio values of 1.8–2.0, which indicates a pure DNA sample.

For Illumina library preparation: adjust the amount of DNA in the tube to be 200ng total per sample in 30 µL molecular grade water. For ONT Library preparation: adjust the amount of DNA in the tube to be 100ng total per sample in 7.5 µL molecular grade water. Using the same DNA for both short-read and long-read sequencing is required to avoid hybrid sequencing mismatch.

6 Equipment

- PCR thermo cycler
- Gel electrophoresis system
- Invitrogen™ Qubit™ 4 Fluorometer
- Invitrogen™ DynaMag™-2 Magnet
- MinION Mk1B Oxford Nanopore
- PC like i7-10700KF / 64GB RAM / 1TB SSD
- Vortex
- Tabletop centrifuge
- PCR workstation
- Pipettes set (P1000, P100, P10)

7 Reagent and Consumables

- Illumina® DNA Prep, (M) Tagmentation (24 Samples or 96 Samples) (Illumina, cat# 20018704 or #20018705)
- IDT® for Illumina® DNA/RNA UD Indexes Set A, Tagmentation (96 Indexes, 96 Samples) (Illumina, cat# 20027213)
- MiSeq Reagent Kit v3 (600-cycle) (Illumina, cat #MS-102-3003)
- Agencourt Ampure XP Beads (Beckman Coulter, cat# A63880/A63881)
- dsDNA BR assay kit (Thermo Fisher Scientific, cat# Q32850)
- Nuclease Free water (Thermo Fisher Scientific, cat# AM9916)
- Absoluter Ethanol (AppliChem, cat# A1613,2500PE)

- Rapid Barcoding kit (Oxford Nanopore Technologies, cat# SQK-RBK004)
- SpotON Flow Cell R9.4.1 (Oxford Nanopore Technologies, cat# FLO-MIN106D)
- Qubit assay tubes (Thermo Fisher Scientific, cat# Q32856)
- 1,5ml Eppendorf Safe-Lock (Eppendorf, cat# 0030120086)
- 0.2 ml PCR tubes
- Disposable filter tips
- 96-well PCR plate

8 Assay procedure

8.1 Library preparation for Illumina

8.1.1 Tagment Genomic DNA

This step uses the Bead-Linked Transposomes (BLT) to tagment DNA, which is a process that fragments and tags the DNA with adapter sequences.

- Prepare the following consumables:

Item	Storage Instructions
BLT	2°C to 8° C Bring to room temperature. Vortex to mix. Do not centrifuge before pipetting.
TB1	-25°C to -15°C Bring to room temperature. Vortex to mix.

- Save the following TAG program on the thermal cycler: 55°C for 15 minutes and Hold at 10°C
- Procedure

- Add 30µl input DNA (200ng) to each well of a 96-well PCR plate
- Vortex BLT (yellow cap) vigorously for 10 seconds to resuspend. Repeat as necessary.
- Combine the following volumes to prepare the tagmentation master mix. Multiply each volume by the number of samples being processed. Reagent overage is included in the volume to ensure accurate pipetting.

BLT	11 µL
TB1	11 µL

- Vortex the tagmentation master mix thoroughly to resuspend.
- Divide the tagmentation master mix volume equally into an 8-tube strip.
- Using a 200 µL multichannel pipette, transfer 20 µL tagmentation master mix to each well of the plate containing a sample. Use fresh tips for each sample column.
- Discard the 8-tube strip after the tagmentation master mix has been dispensed.

- Pipette each sample 10 times to resuspend.

- Seal the plate, place on the preprogrammed thermal cycler, and run the TAG program.

8.1.2 Post Tagmentation Cleanup

This step washes the adapter-tagged DNA on the BLT before PCR amplification.

- Preparation

Item	Storage Instructions
TSB	15°C to 30°C If precipitates are observed, heat at 37°C for 10 minutes, and then vortex until precipitates are dissolved. Use at room temperature
TWB	15°C to 30°C Use at room temperature.

- Save the following PTC program on the thermal cycler: 37°C for 15 minutes and hold at 10°C
- Procedure

- Add 10 µL TSB to the tagmentation reaction.

- Slowly pipette each well 10 times to resuspend the beads.

- Seal the plate, place on the preprogrammed thermal cycler, and run the PTC program.

- Place the plate on the magnetic stand and wait until liquid is clear (~3 minutes).

- Using a multichannel pipette, remove and discard supernatant.

- Wash two times as follows:

  a) Remove the sample plate from the magnetic stand and use a deliberately slow pipetting technique to add 100 µL TWB directly onto the beads. A deliberately slow pipetting technique minimizes the potential of TWB foaming to avoid incorrect volume aspiration and incomplete mixing.
  b) Pipette slowly until beads are fully resuspended.
  c) Place the plate on the magnetic stand and wait until the liquid is clear (~3 minutes).
  d) Using a multichannel pipette, remove and discard supernatant.

- Remove the plate from the magnetic stand and use a deliberately slow pipetting technique to add 100 µL TWB directly onto the beads.

- Pipette each well slowly to resuspend the beads.

- Seal the plate and place on the magnetic stand until the liquid is clear (~3 minutes). Keep on the magnetic stand until step 4 of the Procedure section in Amplify Tagmented DNA.

- The TWB remains in the wells to prevent overdrying of the beads.

8.1.3 Amplify Tagmented DNA

This step amplifies the tagmented DNA using a limited-cycle PCR program. The PCR step adds Index 1 (i7) adapters, Index 2 (i5) adapters, and sequences required for sequencing cluster generation.

- Preparation

Item	Storage Instructions
EPM	-25°C to -15°C Thaw on ice. Invert to mix, then briefly centrifuge.
Index Adapters	-25°C to -15°C Thaw at room temperature. Spin briefly before use.

- Save the following BLT PCR program on a thermal cycler using the appropriate number of PCR cycles, indicated in the table below.

CYCLE STEP	TEMPERATURE	TIME	CYCLES
	68°C	3 minutes	
	98°C	3 minutes	
Amplification	98°C	45 seconds	5
	62°C	30 seconds	
	68°C	2 minutes	
Extension	68°C	1 minutes	
Hold	4°C		

- Procedure

- Combine the following volumes to prepare the PCR master mix. Multiply each volume by the number of samples being processed. Reagent overage is included in the volume to ensure accurate pipetting.

EPM	22 µL
Nuclease-free water	22 µL

- Vortex, and then centrifuge the PCR master mix at 280 × g for 10 seconds.
- With the plate on the magnetic stand, use a 200 µL multichannel pipette to remove and discard supernatant. Foam that remains on the well walls does not adversely affect the library.
- Remove from the magnet.
- Immediately add 40 µL PCR master mix directly onto the beads in each sample well.
- Immediately pipette to mix until the beads are fully resuspended. Alternatively, seal the plate and use a plate shaker at 1600 rpm for 1 minute.
- Seal the sample plate and centrifuge at 280 × g for 3 seconds.

- Add the appropriate index adapters to each sample. 5µl i7 adapter 5µl i5 adapter 96 plex (dual index) 96-well
- Using a pipette set to 40 µL, pipette 10 times to mix. Alternatively, seal the plate and use a plate shaker at 1600 rpm for 1 minute.
- Seal the plate and then centrifuge at 280 × g for 30 seconds.
- Place on the thermal cycler and run the BLT PCR program.

SAFE STOPPING POINT: If you are stopping, store at 2°C to 8°C for up to 3 days.

8.1.4 Clean Up Libraries

This step uses double-sided bead purification procedure to purify the amplified libraries.

Item	Storage Instructions
SPB	2°C to 8°C Let stand at room temperature for 30 minutes. Vortex and invert to mix.
RSB	-25°C to -15°C Thaw and bring to room temperature. Vortex to mix
	Prepare fresh 80% EtOH from absolute ethanol

- Procedure

- Centrifuge at 280 × g for 1 minute to collect contents at the bottom of the well.
- Place the plate on the magnetic stand and wait until the liquid is clear (~5 minutes).
- Transfer 45 µL supernatant from each well of the PCR plate to the corresponding well of a new midi plate.
- Vortex and invert SPB multiple times to resuspend.
- For standard DNA input, perform the following steps.
  a) Add 40 µL nuclease-free water to each well containing supernatant.
  b) Add 45 µL SPB to each well containing supernatant.
  c) Pipette each well 10 times to mix. Alternatively, seal the plate and use a plate shaker at 1600 rpm for 1 minute.
  d) Seal the plate and incubate at room temperature for 5 minutes.
  e) Place on the magnetic stand and wait until the liquid is clear (~5 minutes).
  f) During incubation, thoroughly vortex the SPB (undiluted stock tube), and then add 15µl to each well of a new midi plate.
  g) Transfer 125 µL supernatant from each well of the first plate into the corresponding well of the second plate (containing 15 µL undiluted SPB).
  h) Pipette each well in the second plate 10 times to mix. Alternatively, seal the plate and use a plate shaker at 1600 rpm for 1 minute.
  i) Discard the first plate.

- Incubate the sealed midi plate at room temperature for 5 minutes.
- Place on the magnetic stand and wait until the liquid is clear (~5 minutes).
- Without disturbing the beads, remove and discard supernatant.
- Wash two times as follows.
    a) With the plate on the magnetic stand, add 200 µL fresh 80% EtOH without mixing.
    b) Incubate for 30 seconds.
    c) Without disturbing the beads, remove and discard supernatant.
- Use a 20 µL pipette to remove and discard residual EtOH.
- Air-dry on the magnetic stand for 5 minutes.
- Remove from the magnetic stand.
- Add 32 µL RSB to the beads.
- Pipette to resuspend.
- Incubate at room temperature for 2 minutes.
- Place the plate on the magnetic stand and wait until the liquid is clear (~2 minutes).
- Transfer 30 µL supernatant to a new 96-well PCR plate.

<u>SAFE STOPPING POINT:</u> If you are stopping, seal the plate, and store at -25°C to -15°C for up to 30 days.

8.1.5 Check libraries Quantity and Quality

- Using Qubit dsDNA High Sensitivity assay to measure libraries concentration
- Using QIAxcel to measure Average Lib size

8.1.6 Normalize libraries

2. Use the following formula to convert from ng/µl to nM:

$$\frac{(\text{concentration in ng/µl})}{(660\ \text{g/mol} \times \text{average library size in bp})} \times 10^6 = \text{concentration in nM}$$

- All libraries are diluted to 4mM/µL using 10mM Tris Ph.8
- Pool all the libraries into one tube: 5 µL of each sample -> 4nM pooled library.

8.1.7 Denature and Dilute Library

- Prepare the following consumables
- Denature a 4 nM Library

- Combine 5 µL of 4nM pooled library and 5µl of 0.2 N NaOH in a microcentrifuge tube. Vortex briefly and then centrifuge at 280 × g for 1 minute.

- Incubate at room temperature for 5 minutes.

- Dilute denatured library to 20pM library: Add 990µl prechilled HT1 to 10 µL denatured library. The result is 1 mL of a 20pM library.
- Dilute 20pM library to 10pM library: Combine 300µl of 20pM library and 300µl of Prechilled HT1 in a microcentrifuge tube. Invert to mix and then pulse centrifuge.

MiSeq Reagent Kit v2 Dilute the denatured PhiX control to 12.5pM, which produces an optimal cluster density using v2 reagents. In this protocol, we use 9pM denatured PhiX as control.

- Dilute PhiX to 4 nM: Combine 2 µL of 10 nM PhiX library and 3 µL of 10 mM Tris-Cl, pH 8.5 with 0.1% Tween20 in a microcentrifuge tube.
- Denature PhiX Control

- Combine 5µl of 4nM PhiX library and 5µl of 0.2N NaOH in a microcentrifuge tube.
- Vortex briefly to mix, Centrifuge at 280 × g for 1 minute.
- Incubate at room temperature for 5 minutes.

- Dilute Denatured PhiX to 20 pM: Add 990µl prechilled HT1 to 10 µL the denatured PhiX library. The result is 1 mL of a 20pM PhiX library. Invert to mix.
- Dilute Denatured PhiX to 9pM: Add 300 µL of 20pM PhiX to 300 µL of Prechilled HT1. The result is 600µl of a 9pM PhiX library.
- Combine Library and PhiX Control

- Combine 6 µL of denatured PhiX control and 594µl denatured library.
- Set aside on ice until you are ready to load it onto the reagent cartridge.

8.2 Library preparation for ONT

8.2.1 Using Rapid Barcoding Kit (SQK-RBK004)

- Set up on ice the following reaction for each sample

Component	**Volume**
DNA amplicons from step 7 (100ng)	7.5 µL
Fragmentation Mix RB 01-12 (one for each sample)	2.5 µL
Total	10 µL

- Mix gently be flicking the tube, and spin down
- Incubate at 30°C for 1 minute and then at 80°C for 1 minute and briefly cool on ice.
- Pool all barcoded samples (max 12 samples), noting the total volume.
- Resuspend the AMPure XP beads by vortexing.
- Prepare fresh 80% ethanol in Nuclease-free water.

- Prepare 10mM Tris-HCl pH 7.5-8.0 with 50 mM NaCl buffer.
- To the entire pooled barcoded sample, add an equal volume of resuspended AMPure XP beads, and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at RT.
- Spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 200 µL of freshly prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.
- Repeat the previous step.
- Spin down and place the tube back on the magnet. Pipette off any residual 80% ethanol and briefly allow to dry.
- Remove the tube from the magnetic rack and resuspend pellet in 12 µL of 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl. Incubate for 2 minutes at RT.
- Pellet the beads on a magnet until the eluate is clear and colourless.
- Remove and retain 10µl of eluate into a clean 1.5 mL Eppendorf DNA LoBind tube.

Safe stopping point: Store at –20°C for up to one week if not planning to load on MinION.

- Add 1 µL of RAP to 10 µL of barcoded DNA (Total 11µL pool).
- Mix gently by flicking the tube, and spin down.
- Incubate the reaction for 5 minutes at RT.
- The prepared library is used for loading to the MinION flow cell. Store the library on ice until ready to load.

8.2.2 Using Rapid Barcoding Kit 96 (SQK-RBK110.96)

- The Rapid Barcoding Kit 96 contains sufficient reagents for 12 pooled libraries using 24 barcodes (6 pooled libraries using 48 barcodes and 3 pooled libraries using 96 barcodes).
- Please refer to use all reagents in Rapid Barcoding Kit 96 (SQK-RBK110.96) including Rapid Adapter F (RAP-F), SPRI beads (SPRI), Sequencing Buffer II (SBII), Loading Beads II (LBII), Elution Buffer (EB), Flush Tether (FLT) and Flush Buffer (FB)
- Briefly centrifuge and place the barcodes plate on ice
- In 0.2 ml thin-walled PCR tubes or a PCR plate 96, mix the following. The Rapid Barcodes can be transferred using a multichannel pipette:

Component	**Volume**
DNA amplicons from step 9.7 (100ng)	7.5 µL
Rapid Barcodes (RB01-96, one for each sample)	2.5 µL
Total	9 µL

- Mix well by pipetting. Seal the plate and spin down in a centrifuge.
- Incubate the tubes or plate at 30°C for 1 minutes and then at 80°C for 1 minutes. Briefly put the tubes or plate on ice to cool.
- Pool all barcoded samples in your desired ratio, noting the total volume.
- Resuspend the SPRI beads by vortexing.
- To the entire pooled barcoded sample from Step 7, add an equal volume of resuspended SPRI beads and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at room temperature.
- Prepare at least 3 ml of fresh 80% ethanol in nuclease-free water.
- Spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 1.5 ml of freshly-prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.
- Repeat the previous step. Briefly spin down and place the tube back on the magnet. Pipette off any residual ethanol. Allow to dry for 30 seconds, but do not dry the pellet to the point of cracking.
- Remove the tube from the magnetic rack and resuspend the pellet in 15 µl Elution Buffer (EB). Incubate for 10 minutes at room temperature.
- Pellet the beads on a magnet until the eluate is clear and colourless.
- Remove and retain 15 µL of eluate into a clean 1.5 mL Eppendorf DNA LoBind tube.
- Remove and retain the eluate which contains the DNA library in a clean 1.5 mL Eppendorf DNA LoBind tube
- Dispose of the pelleted beads
- Transfer 11 µL of the sample into a clean 1.5 mL Eppendorf DNA LoBind tube.
- Add 1 µL of Rapid Adapter F (RAP F) to 11 µL of barcoded DNA
- Mix gently by flicking the tube, and spin down.
- Incubate the reaction for 5 minutes at room temperature.

8.2.3 Load to MinION and sequence

- Priming and loading the SpotON Flow Cell R9.4.1

- Set up the MinION flow cell and host computer, including MinKNOW software.
- Open the MinKNOW GUI from the desktop icon and establish a local connection.
- Inset a flow cell into MinION.
- Click "Check Flow Cells" at the bottom of the screen then click "Start test." Check the number of active pores available. When the check is complete, it is reported in the Notification panel. Check to ensure it has enough pores for a good sequencing run (warranty for flow cells: 800 nanopores or above checked within 5 days of receipt).
- Thaw the Sequencing Buffer (SQB), Loading Beads (LB), Flush Tether (FLT) and one tube of Flush Buffer (FB) or Sequencing Buffer II (SBII), Loading Beads II (LBII) or Loading Solution (LS, if using), Flush Tether (FLT) and Flush Buffer (FB) if using Rapid Barcoding Kit 96 (SQK-RBK110.96) at room temperature
- Thoroughly mix the Sequencing Buffer (SQB) and Flush Buffer (FB) tubes by vortexing. Spin down the Flush Tether (FLT) tube, and return to ice.
- To prepare the flow cell priming mix, add 30 μL of thawed and mixed Flush Tether (FLT) to 1.17 mL of thawed and mixed Flush Buffer (FB), and mix by vortexing at room temperature. Open the lid of the nanopore sequencing device and slide the flow cell's priming port cover clockwise 90 degrees.
- Set a P1000 pipette to 200 uL, insert the tip into the priming port, turn the wheel until the dial shows 220-230 uL, or until you can see a small volume of buffer entering the pipette tip. Do not remove more than this.
- Visually check that there is continuous buffer from the priming port across the sensor array.
- Load 800 uL of the priming mix into the flow cell via the priming port, avoiding the introduction of air bubbles. Wait for 5 minutes.
- Thoroughly mix the contents of the Loading Beads (LB) by pipetting.
- Prepare library for loading as follows:

With library prepared by Rapid Barcoding Kit (cat# SQK-RBK004)

Component	**Volume (uL)**
Sequencing Buffer (SQB)	34
Loading Beads (LB) mixed immediately before use	25.5
Nuclease free water	4.5
DNA Library (11μL pool)	11
Total	75

With library prepared by Rapid Barcoding Kit 96 (SQK-RBK110.96)

Component	Volume (uL)
Sequencing Buffer II (SBII)	37.5
Loading Beads II (LBII) mixed immediately before use	25.5
DNA Library (12 µL pool)	11
Total	75

- Gently lift the SpotON sample port cover to make the SpotON sample port accessible.
- Load 200 µL of the priming mix into the flow cell via the priming port (not the SpotON sample port), avoiding the introduction of air bubbles.
- Mix the prepared library gently by pipetting up and down just prior to loading.
- Add 75 uL of sample to the flow cell via the SpotON sample port in a dropwise fashion. Ensure each drop flows into the port before adding the next.
- Gently replace the SpotON sample port cover, making sure the bung enters the SpotON port, close the priming port and replace the MinION lid.
- Start the sequencing run using the MinKNOW software.

When to STOP:

Depending on the variation in coverage of each barcoded samples, generally, a deep (>100×) long-read set and a medium (~50×) short-read set is recommended for an accurate hybrid assembly. If long reads only assembly is performed, more depth is better up to ~200×. Coverage can be rough estimated as follow: C = sequenced bases/ genome size.

After your sequencing experiment is complete, if you would like to reuse the flow cell, please follow the Wash Kit instructions, and store the washed flow cell at 2-8°C

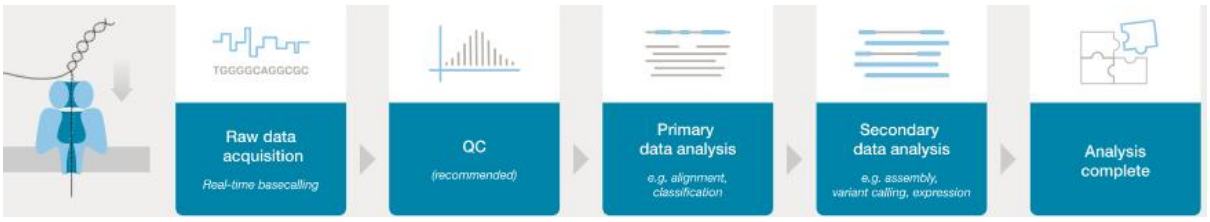
OR

Follow the returns procedure by washing out the flow cell ready to send back to Oxford Nanopore.

9 Data analysis

1. Basic bioinformatic for data analysis

MinKNOW, the operating software that drives nanopore sequencing devices, carries out several core tasks, including data acquisition, real-time analysis and feedback, local basecalling, and data streaming. MinKNOW produces FAST5 (HDF5) files and/or FASTQ files, based on preferences. FAST5 files contain raw signal data that can be used for basecalling. In our work, we obtain raw data in FASTQ format that contain nucleotide sequence data and their corresponding quality scores. The entire data analysis can be carried in 5 steps starting from raw data acquisition from MinKNOW.



First step of the analysis is basecalling which is usually completed by MinKNOW software. This step includes barcoding/demultiplexing, adapter trimming and alignment. For the rest of the analysis, you need to use bioinformatics workflows on a Linux/Mac computer. You are advised to watch some bioinformatics lectures on YouTube to get a good insight on these workflows. One such easy to follow lecture series from Dr. Simon Cockell of University of Newcastle, UK can be viewed at

https://www.youtube.com/c/SimonCockell/videos

In addition, keeping in view of your upcoming training you are also suggested to watch one recorded video on Nanopore data analysis. This lecture focusses on how to analyse raw data obtained from MinKNOW software using bioinformatics pipelines.

https://nanoporetech.com/resource-centre/bioinformatics-workflows-sars-cov-2-raw-nanopore-reads-consensus-genomes-using

Besides the above, some basic knowledge in Linux, Miniconda, Docker and GitHub are essential. For editing the code, Visual studio code is also a nice to have software that be easily installed on any OS. Linux and Mac operating systems are ideal for bioinformatics. However, if your PC has Windows 10 as OS, you could still install Linux and share all your data between the two operating systems without having to reboot the PC.

If you are a beginner in bioinformatics without any prior knowledge in Linux or any programming language, you can still do the data analysis, as the bioinformatics workflows for Nanopore are meant to cater the needs of biologists who wants to do sequence analysis by themselves.

11.1 Linux

Linux is an open-source operating system that is very secure, fast and reliable. Linux is most preferred for bioinformatics workflow as it can easily handle large datasets.

Installation of Linux on Windows 10 PC or laptop

On a Windows 10 PC or laptop, Linux can be installed as WSL-2 (Windows-subsystem for Linux, version 2). WSL is freely downloadable from Microsoft store on Windows 10 PC.

IMPORTANT: System should be x64 systems: Windows Version 1903 or higher, with Build 18362 or higher.

The information on Windows build can be found at
Start > Settings > System > About (under windows specifications)

More details on installation of WSL-2 can be read at the following URL

https://docs.microsoft.com/en-us/windows/wsl/install-win10#manual-installation-steps

Users are advised to follow step-by-step instructions under the section "Install WSL & update to WSL-2" at the above website. When asked for a username, you may give a short name or initials, but without any spaces. When you type your password, you may not see cursor moving, but Linux stores your password. You need to confirm the password by typing again. For beginners in Linux, either Ubuntu 18.04 or 20.04 is ideal as there is lot of reading material on internet. WSL does not have a graphical user interface (GUI) and hence some basic commands are essential for easy manoeuvre at the terminal prompt. A very good intro to Linux can be found at

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

There is lot of literature on Ubuntu on internet. For a start, following websites are useful from bioinformatics perspective.

1. http://www.cellbiol.com/bioinformatics_web_development/chapter-2-the-linux-operating-system-setting-up-a-linux-web-server/basic-linux-shell-commands/

2. https://bioinformaticsworkbook.org/Appendix/Unix/UnixCheatSheet.html#gsc.tab=0

11.2 Conda and Miniconda

Conda is an open-source package management and environment management system that runs on any OS. It has been created for Python programs, although it can package and distribute software for any language. Conda as a package manager helps you find and install packages. You will learn more about this package manager as you use it regularly.

Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others. To know more about Conda / miniconda, please refer to the following website.

https://docs.conda.io/projects/conda/en/latest/

Install Miniconda on WSL

At Ubuntu terminal, you need to type the following commands at $ prompt

```
$ wget https://repo.continuum.io/miniconda/Miniconda3-py39_4.9.2-Linux-x86_64.sh
```

Click Enter button. The above command will download the miniconda from internet.

```
$ bash Miniconda3-py39_4.9.2-Linux-x86_64.sh
```

The above command will install the miniconda application on your Linux.

If your system is Mac,

```
$ wget https://repo.continuum.io/miniconda/Miniconda3-py39_4.9.2-MacOSX-x86_64.sh
```

Followed by

```
$ bash Miniconda3-py39_4.9.2-MacOSX-x86_64.sh
```

11.3 Docker

Docker is an open-source software platform to build, deploy or execute applications by using containers. As the name indicates, containerization isolate applications with each container having a separate environment at the same time sharing the underlying OS kernel. Docker provides a good platform to run multiple applications at the same time, each needing its own environment. Containerization is very similar to virtual machines, but unlike VMs, containers are faster and use less system resources. Each Docker container has an image file to create a run-time environment for the supposed application.

Installing Docker

Docker can be easily installed on WSL2 based windows by following instructions as mentioned in the below URL

https://docs.docker.com/docker-for-windows/wsl/

11.4 GitHub

GitHub is an Open-source Community where people around the globe work together on Open-source projects and make contributions. GitHub is also a code hosting platform for collaboration and version control. Importantly, GitHub is a repository for bioinformatics pipelines, and in our work, we use GitHub to download Nanopore workflows.

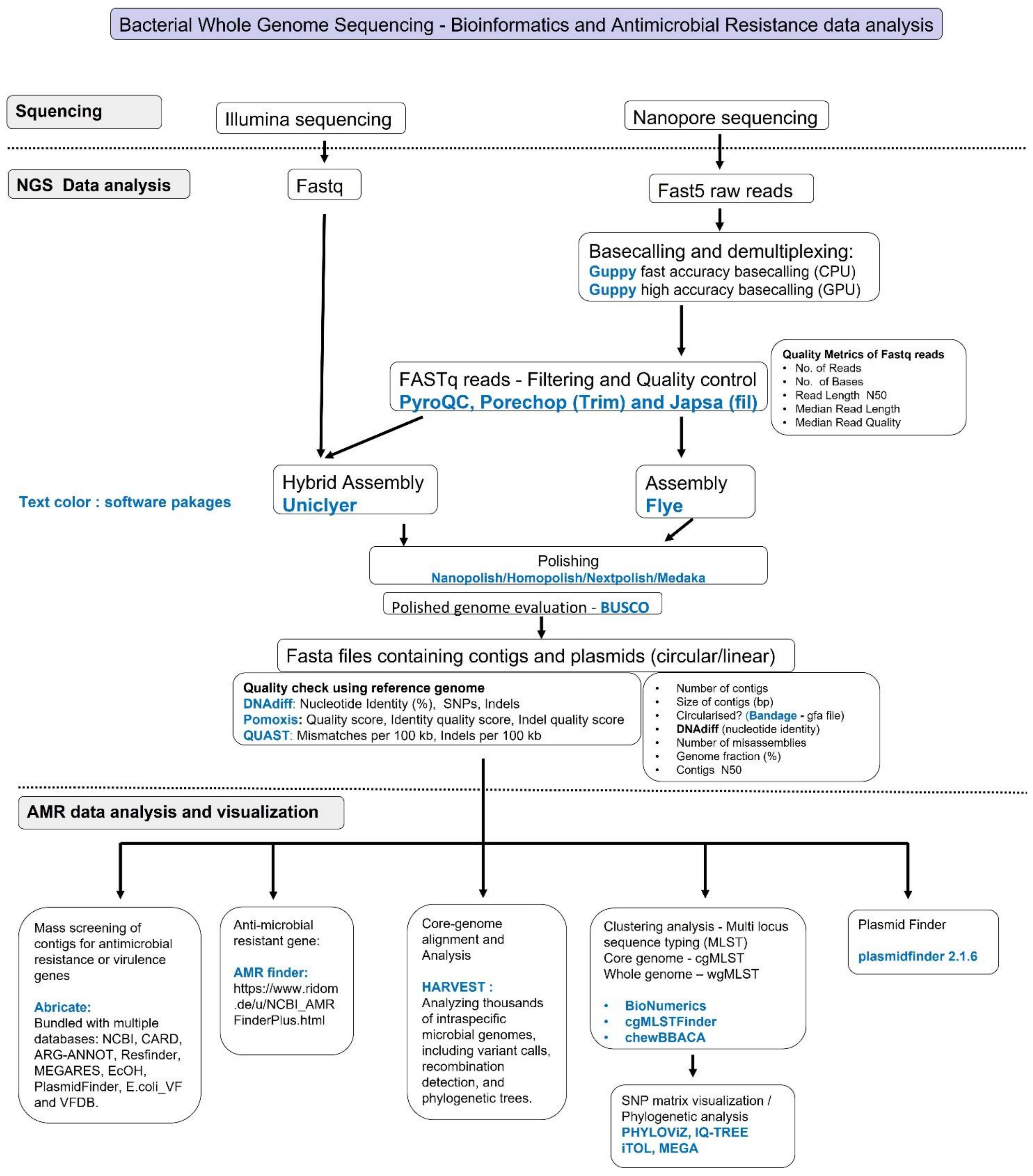
For more information and basic commands on GitHub, please refer to

https://guides.github.com/introduction/git-handbook/

11.5 Visual Studio Code

This editor can be downloaded freely from **https://code.visualstudio.com** depending on the type of OS. This is very useful software to edit the code, besides providing an excellent platform to integrate linux terminal, Git and docker.

***Analysis Plan*:**


Bacterial Whole Genome Sequencing - Bioinformatics and Antimicrobial Resistance data analysis
Squencing
Illumina sequencing
Nanopore sequencing
NGS Data analysis
Fastq
Fast5 raw reads
Basecalling and demultiplexing:
Guppy fast accuracy basecalling (CPU)
Guppy high accuracy basecalling (GPU)
FASTq reads - Filtering and Quality control
PyroQC, Porechop (Trim) and Japsa (fil)
Quality Metrics of Fastq reads
• No. of Reads
• No. of Bases
• Read Length N50
• Median Read Length
• Median Read Quality
Text color : software pakages
Hybrid Assembly
Uniclyer
Assembly
Flye
Polishing
Nanopolish/Homopolish/Nextpolish/Medaka
Polished genome evaluation - BUSCO
Fasta files containing contigs and plasmids (circular/linear)
Quality check using reference genome
DNAdiff: Nucleotide Identity (%), SNPs, Indels
Pomoxis: Quality score, Identity quality score, Indel quality score
QUAST: Mismatches per 100 kb, Indels per 100 kb
• Number of contigs
• Size of contigs (bp)
• Circularised? (Bandage - gfa file)
• DNAdiff (nucleotide identity)
• Number of misassemblies
• Genome fraction (%)
• Contigs N50
AMR data analysis and visualization
Mass screening of contigs for antimicrobial resistance or virulence genes
Abricate:
Bundled with multiple databases: NCBI, CARD, ARG-ANNOT, Resfinder, MEGARES, EcOH, PlasmidFinder, E.coli_VF and VFDB.
Anti-microbial resistant gene:
AMR finder:
https://www.ridom.de/u/NCBI_AMRFinderPlus.html
Core-genome alignment and Analysis
HARVEST :
Analyzing thousands of intraspecific microbial genomes, including variant calls, recombination detection, and phylogenetic trees.
Clustering analysis - Multi locus sequence typing (MLST)
Core genome - cgMLST
Whole genome – wgMLST
• BioNumerics
• cgMLSTFinder
• chewBBACA
SNP matrix visualization / Phylogenetic analysis
PHYLOViZ, IQ-TREE
iTOL, MEGA
Plasmid Finder
plasmidfinder 2.1.6

9.1 Ilumina data

- The raw data come from Ilumina should be compressed in the form of paired end data.

  For example: **R1.fq.gz** and **R2.fq.gz** (for one single sequence)
- Using fastp to preprocess raw data from Illumina (https://github.com/OpenGene/fastp)
  - Install: There are many ways to install, but I suggest installing with conda (it might not be the latest version but still can be used without any problems).

    ```
    $ conda install -c bioconda fastp
    ```
  - Usage:

    ```
    $ fastp -i in.R1.fq.gz -I in.R2.fq.gz -o out.R1.fq.gz -O out.R2.fq.gz
    ```

    (The input and output files can be named as you want)

9.2 Nanopore data

- Using fast5 files (raw data from nanopore) as input for high accuracy basecalling (with flow cell and kit name) by guppy (GPU – version)
  - Usage:

    ```
    $ guppy_basecaller –compress_fastq –i ~/data/fast5_pass/barcode01 –s ~/data/fast5/barcode01 –flowcell FLO-MIN106 –kit SQK-RBK004 -x "cuda:0"
    ```

  -compress_fastq: compress the fastq output (in gz form)

  -i: input path, the directory of the input files

  -s: save path, the directory where the output will be saved

  -flowcell: name of the flow cell

  -kit: name of the kit

  -x: GPU using. It may occur error sometimes. Try "cuda:auto" instead of "cuda:0"

9.3 Hybrid assembly by Unicycler https://github.com/rrwick/Unicycler.git

- Requirement: try to install all the dependency package before install Unicycler
  - Linux or macOS
  - Python 3.4 or later
  - C++ compiler with C++14 support:
    - GCC 4.9.1 or later
    - Clang 3.5 or later
    - ICC also works (though I don't know the minimum required version number)
  - setuptools (only required for installation of Unicycler)

- For short-read or hybrid assembly:
    - SPAdes v3.14.0 or later (spades.py) (can be installed by conda)
- For long-read or hybrid assembly:
    - Racon (racon)
- For rotating circular contigs:
    - BLAST+ (makeblastdb and tblastn)

– Install:

```
$ git clone https://github.com/rrwick/Unicycler.git
$ cd Unicycler
$ python3 setup.py install
```

- If the last command complains about permissions, you may need to run it with "sudo".

– Usage:

- Move to the directory where the inputs are saved. All the input files should be put in the same folder for easier process.

```
$ unicycler -1 short_reads_1.fastq.gz -2 short_reads_2.fastq.gz -l long_reads.fastq.gz -o output_dir
```

-1, -2: Ilumina data input (after qc by fastp)

-l:nanopore data input (after hac by guppy)

-o: directory where the output files will be saved

9.4 Long read only assembly by Flye https://github.com/fenderglass/Flye

– Building Requirements

- Python 2.7 or 3.5+ (with setuptools package installed)
- C++ compiler with C++11 support (GCC 4.8+ / Clang 3.3+ / Apple Clang 5.0+)
- GNU make
- Git
- Core OS development headers (zlib, ...)

– Install: using conda

```
$ conda install flye
```

– Usage: flye (--pacbio-raw | --pacbio-corr | --pacbio-hifi | --nano-raw |

```
        --nano-corr | --nano-hq ) file1 [file_2 ...]
        --out-dir PATH
```

9.5 Quality check of the output from Unicycler by CheckM
(https://github.com/Ecogenomics/CheckM)

- Install: using conda

  $ conda create -n checkm python=3.9 (#create a new environment called "checkm")

  $ conda activate checkm (# activate checkm environment just created)

  $ conda install numpy matplotlib pysam (# install dependency package)

  $ conda install hmmer prodigal pplacer (# install dependency package)

  $ pip3 install checkm-genome

- Reference data:
  - CheckM relies on a number of precalculated data files which can be downloaded from https://data.ace.uq.edu.au/public/CheckM_databases/ and must be decompress into a directory. The path to this data can be set using the CHECKM_DATA_PATH environmental variable, e.g.:

    $ export CHECKM_DATA_PATH=/path/to/my_checkm_data

  - Alternatively, the following command can be run to inform CheckM of where the files have been placed:

    $ checkm data setRoot <checkm_data_dir>

- Usage: we use the "Lineage-specific Workflow"

  $ checkm lineage_wf <input folder> <output folder>

  (All genomes to be analyzed must reside in a single <input> directory)

9.6 Search for anti-microbial resistant gene
9.6.1 By AMR finder (https://github.com/ncbi/amr/wiki)

- Install: using conda
  - Activate an environment you just created (e.g., the checkm environment above or you can create new environment)
  - $ conda install -y -c bioconda -c conda-forge ncbi-amrfinderplus
- Usage:
  - Change directory to the folder contains sequence that need to search (e.g., <input folder> above)
  - $ amrfinder -n <nucleotide_fasta>

9.6.2 By Abricate (https://github.com/tseemann/abricate)

Mass screening of contigs for antimicrobial resistance or virulence genes. It comes bundled with multiple databases: NCBI, CARD, ARG-ANNOT, Resfinder, MEGARES, EcOH, PlasmidFinder, Ecoli_VF and VFDB.

- Install: using conda

```
$ conda install -c conda-forge -c bioconda -c defaults abricate
$ abricate --check
$ abricate –list
```

- Usage:

```
$ abricate assembly.fa
$ abricate assembly.fa.gz
$ abricate assembly.gbk
$ abricate assembly.gbk.bz2
```

The default database is *ncbi*. You can choose a different database using the --db option

9.7 Core-genome alignment and Analysis (https://harvest.readthedocs.io/en/latest/)

Harvest is a suite of core-genome alignment and visualization tools for quickly analyzing thousands of intraspecific microbial genomes, including variant calls, recombination detection, and phylogenetic trees.

https://github.com/katholt/Kleborate/wiki

On going

10 References

1. Illumina DNA Prep Reference Guide Document # 1000000025416 v09 June 2020
2. https://store.nanoporetech.com/eu/rapid-barcoding-kit.html
3. Nanopore Protocol, Rapid Barcoding Kit 96 (SQK-RBK110.96) Version: RBK_9126_v110_revD_24Mar2021
4. Wick RR, Judd LM, Gorrie CL, Holt KE. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. PLoS Comput Biol 2017.

Important: Video links on AMR:

https://en.wikipedia.org/wiki/Carbapenem-resistant_enterobacteriaceae

Gut microbiota profiling with full-length 16S rRNA using MinION Oxford Nanopore Technology (ONT)

and
Metagenomic and Metabolomic analysis

AG Velavan
Institute of Tropical Medicine
Universität Tübingen
Wilhelmstr 27
72074 Tuebingen
Germany

Protocol updates

Version	**Update summary**
1.0	Complete draft protocol

INDEX

1. Abbreviations

bp	base pairs
°C	degree Celsius
DNA	deoxyribonucleic acid
dNTP	deoxynucleotide triphosphate
µl	microlitres
µM	micromolar
ml	millilitres
NTC	non-template control
PCR	polymerase chain reaction
PC	positive control
RNA	ribonucleic acid
RAP	Rapid Sequencing Adapter

2. Purpose

Due to the presence of both highly conserved (adequate for universal primers and phylogenetic signal) and highly variant regions (different across species), the 16S rRNA gene is often used for sequence-based bacterial identification.

3. Principal of the assay

DNA is amplified and barcoded with the Oxford Nanopore 16S Barcoding Kit (SQK-RAB204). Fragment of whole 16S rRNA gene (about 1500bp) is amplified, at the same time, barcode are attached to the end of the sequence. There are 24 unique barcodes, allowing the user to pool up to 24 different samples in one sequencing experiment. Barcoded samples are pooled, and Rapid 1D Sequencing Adapters are then added to the tagged ends. Samples are sequenced on MinION device.

4. Material

High molecular weight genomic DNA

5. Pre-analytics

For this protocol, you will need 10 ng high molecular weight genomic DNA: Transfer 10 ng genomic DNA into a DNA LoBind tube and adjust the volume to 10 µl with nuclease-free water. Mix thoroughly by flicking the tube, to avoid unwanted shearing. Spin down briefly in a microfuge

6. Equipment

- PCR thermo cycler
- Gel electrophoresis system
- Invitrogen™ Qubit™ 4 Fluorometer
- Invitrogen™ DynaMag™-2 Magnet
- MinION Mk1B Oxfore Nanopore
- PC like i7-10700KF / 64GB RAM / 1TB SSD
- Vortexer
- Tabletop centrifuge
- PCR workstation
- Pipettes set (P1000, P100, P10)

7. Reagent and Consumables

- LongAmp Taq 2X Master Mix (NEB, cat# M0287)
- Agencourt Ampure XP Beads (Beckman Coulter, cat# A63880/A63881)
- dsDNA BR assay kit (Thermo Fisher Scientific, cat# Q32850)
- Nuclease Free water (Thermo Fisher Scientific, cat# AM9916)
- Absoluter Ethanol (AppliChem, cat# A1613,2500PE)
- 16S Barcoding Kit (Oxford Nanopore Technologies, cat# SQK-RAB204)
- 10 mM Tris-HCl pH 8.0 with 50 mM NaCl
- Flongle Sequencing Expansion (Oxford Nanopore Technologies, cat# EXP-FSE001)
- Flow Cell Priming Kit (Oxford Nanopore Technologies, cat# EXP-FLP002)
- SpotON Flow Cell R9.4.1 (Oxford Nanopore Technologies, cat# FLO-MIN106D)
- Flongle device - flow cell and adapter (Oxford Nanopore Technologies, cat# ADP-FLG001 and cat# FLO-FLG001
- Qubit assay tubes (Thermo Fisher Scientific, cat# Q32856)
- 1.5 ml Eppendorf DNA LoBind tubes
- 0.2ml PCR tubes
- Disposable filter tips

8. Assay procedure

8.1. Amplification

Thaw the 16S Barcodes at room temperature, mix by pipetting up and down, and spin down briefly. Keep the barcodes on ice until ready to use.

- In a 0.2 ml thin-walled PCR tube, mix the following:

Component	**1 reaction**
Nuclease-free water	14 µL
Input DNA (10 ng)	10 µL
16S Barcode, at 10 µM	1 µl
LongAmp Taq 2X master mix	25 µl
Total	50 µL

- Mix gently by flicking the tube, and spin down.
- Amplify using the following cycling conditions:

CYCLE STEP	**TEMPERATURE**	**TIME**	**CYCLES**
Initial denaturation	95 °C	1 minutes	1
Denaturation	95°C	20 seconds	25
Annealing	55°C	30 seconds	
Extension	65°C	2 minutes	
Final extension	65°C	5 minutes	1
Hold	4 °C	∞	
NOTE: If the amount of input material is altered, the number of PCR cycles may need to be adjusted to produce the same yield. Typically use 25 cycles.			

Safe stopping point: Store at –20°C for up to one week, or –70°C for long term storage.

8.2. Analyzing on agarose gel electrophoresis

- Using 5µL of PCR product to check the appearance of 1500bp bands on 1.1% agarose gel

8.3. Purification by AMPure XP Bead

- Resuspend the AMPure XP beads by vortexing.
- Prepare fresh 80% ethanol in Nuclease-free water.
- Prepare 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl buffer.
- To the entire pooled sample (45µL - 50µL), add 25µL of resuspended AMPure XP beads, and mix by flicking the tube.
- Incubate on a Hula mixer (rotator mixer) for 5 minutes at RT.
- Briefly spin down the sample and pellet on a magnet. Keep the tube on the magnet, and pipette off the supernatant.
- Keep the tube on the magnet and wash the beads with 200µl of freshly prepared 80% ethanol without disturbing the pellet. Remove the ethanol using a pipette and discard.

- Repeat the previous step.
- Spin down and place the tube back on the magnet. Pipette off any residual ethanol and briefly allow to dry.
- Remove the tube from the magnetic rack and resuspend pellet in 12µl of 10mM Tris-HCl pH 7.5-8.0 with 50mM NaCl. Incubate for 2 minutes at RT.
- Pellet the beads on a magnet until the eluate is clear and colorless.
- Remove and retain 10µL of eluate into a clean 1.5ml Eppendorf DNA LoBind tube.

8.4. Quantify the pool using Qubit dsDNA Assay

- Set up the required number 0.5mL Qubit assay tubes for standards and samples. Note: the standards require two tubes.
- Label tube lids. Do not label the side of the tube as this could interfere with the sample read.
- Prepare the Qubit working solution by diluting the Qubit dsDNA HS reagent 1:200 in Qubit dsDNA HS buffer. Use a clean plastic tube each time you prepare Qubit working solution. Do not mix the working solution in a glass container.
- Prepare sufficient Qubit working solution to accommodate all standards and samples.
- Add 190uL of Qubit working solution to each of the tubes used for standards.
- Add 10uL of each qubit standard to the appropriate tube, mix by vertexing 2-3 seconds.
- Add 197uL of Qubit working solution to each assay tube.
- Add 3uL each sample to the assay tubes, then mix by vertexing 2-3 seconds. The final volume in each tube should be 200uL.
- Allow all tubes to incubate at room temperature for 2 minutes.
- Sample concentration can now be measured on the Qubit Fluorometer.

8.5. Pooling and Normalization

- Pool all barcoded libraries in the desired ratios to a total of 50-100 fmoles in 10 µl of 10 mM Tris-HCl pH 8.0 with 50 mM NaCl. For 16S amplicons of ~1500 bp, 50-100 fmoles equates to ~50-100 ng.
- Add 1 µl of RAP to the barcoded DNA.
- Mix gently by flicking the tube, and spin down.
- Incubate the reaction for 5 minutes at RT.
- The prepared library is used for loading into the MinION flow cell. Store the library on ice until ready to load.

8.6. Load to MinION and Sequence

❖ **Priming and loading the SpotON Flow Cell**

- Set up the MinION flow cell and host computer, including MinKNOW software.
- Open the MinKNOW GUI from the desktop icon and establish a local connection.
- Inset a flow cell into MinION.

- Click “Check Flow Cells” at the bottom of the screen then click “Start test.” Check the number of active pores available. When the check is complete, it is reported in the Notification panel. Check to ensure it has enough pores for a good sequencing run (warranty for flow cells: 800 nanopores or above checked within 5 days of receipt).
- Thaw the Sequencing Buffer (SQB), Loading Beads (LB), Flush Tether (FLT) and one tube of Flush Buffer (FB) at room temperature before placing the tubes on ice.
- Thoroughly mix the Sequencing Buffer (SQB) and Flush Buffer (FB) tubes by vortexing. Spin down the Flush Tether (FLT) tube, mix by pipetting, and return to ice.
- Prepare the flow cell priming mix: Add 30uL of thawed and mixed Flush Tether (FLT) directly to the tube of thawed and mixed Flush Buffer (FB), and mix by pipetting up and down.
- Open the lid of the nanopore sequencing device and slide the flow cell's priming port cover clockwise 90 degrees.
- Set a P1000 pipette to 200uL, insert the tip into the priming port, turn the wheel until the dial shows 220-230uL, or until you can see a small volume of buffer entering the pipette tip. Do not remove more than this.
- Visually check that there is continuous buffer from the priming port across the sensor array.
- Load 800uL of the priming mix into the flow cell via the priming port, avoiding the introduction of air bubbles. Wait for 5 minutes.
- Thoroughly mix the contents of the Loading Beads (LB) by pipetting.
- Prepare library for loading my mixing:

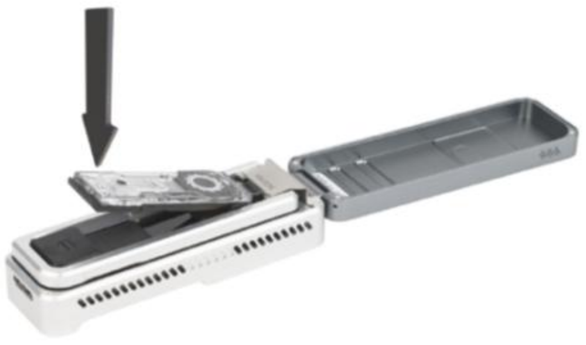
Component	**Volume (uL)**
Sequencing Buffer (SQB)	34
Loading Beads (LB)	25.5
Nuclease free water	4.5
DNA Library (11µL pool)	11
Total	75

- Gently lift the SpotON sample port cover to make the SpotON sample port accessible.
- Load 200µL of the priming mix into the flow cell via the priming port (not theSpotON sample port), avoiding the introduction of air bubbles.
- Mix the prepared library gently by pipetting up and down just prior to loading.
- Add 75uL of sample to the flow cell via the SpotON sample port in a dropwise fashion. Ensure each drop flows into the port before adding the next.
- Gently replace the SpotON sample port cover, making sure the bung enters the SpotON port, close the priming port and replace the MinION lid.

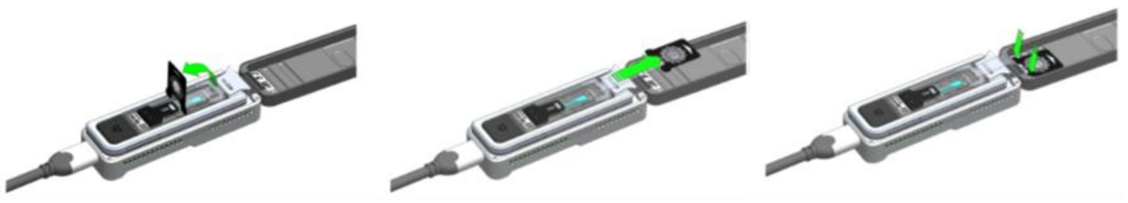
- Start the sequencing run using the MinKNOW software.

❖ Loading the Flongle flow cell

- Set up the MinION flow cell and host computer, including MinKNOW software.
- Open the MinKNOW GUI from the desktop icon and establish a local connection.
- Place the Flongle adapter into the MinION.
- The adapter should sit evenly and flat on the MinION Mk1B or GridION platform. This ensures the flow cell assembly is flat during the next stage.
- Place the flow cell into the Flongle adapter and press the flow cell down until you hear a click. Users should NOT touch the reverse side of the Flongle flow cell array or the contact pads on the Flongle adapter.

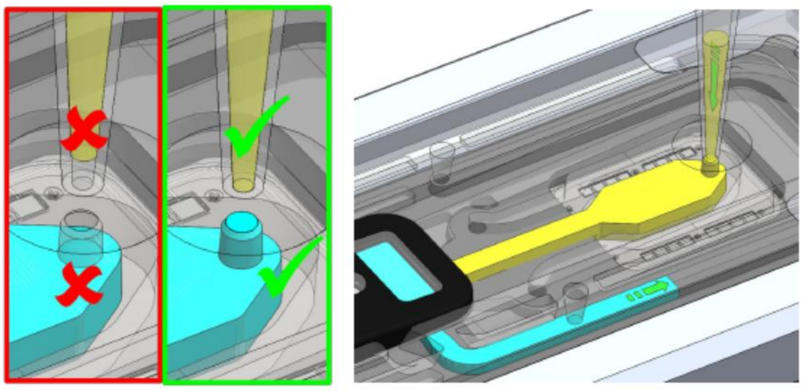


- The flow cell should sit evenly and flat inside the adapter, to avoid any bubbles forming inside the fluidic compartments.
- Click “Check Flow Cells” at the bottom of the screen then click “Start test.” Check the number of active pores available. When the check is complete, it is reported in the Notification panel. Check to ensure it has enough pores for a good sequencing run (warranty for flow cells: 50 nanopores or above checked within 5 days of receipt).
- Thaw the Sequencing Buffer II (SQBII), Loading Beads II (LBII), Flush Tether (FLT) and Flush Buffer (FB) at room temperature before placing the tubes on ice.
- Thoroughly mix by vortexing, spin down and return to ice.
- In a fresh 1.5ml Eppendorf DNA LoBind tube, mix 117µl of Flush Buffer (FB) with 3µl of Flush Tether (FLT) and mix by pipetting.
- Peel back the seal tab from the Flongle flow cell, up to a point where the sample port is exposed.



Lift up the seal tab; Pull the seal tab to open access to the sample port; Hold the seal tab open by using adhesive on the tab to stick to the MinION Mk 1B lid

- To prime your flow cell with the mix of Flush Buffer (FB) and Flush Tether (FLT) that was prepared earlier, ensure that there is no air gap in the sample port or the pipette tip. Place the P200 pipette tip inside the sample port and slowly dispense the priming fluid into the Flongle flow cell. To avoid flushing the flow cell too vigorously, load the priming mix by twisting the pipette plunger down.



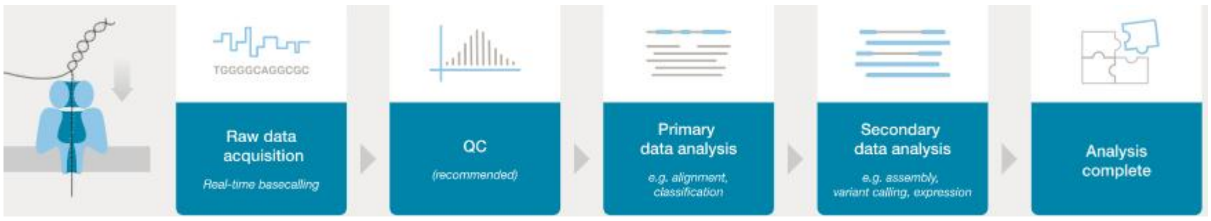
- Vortex the vial of Loading Beads II (LBII). Note that the beads settle quickly, so immediately prepare the Sequencing Mix in a fresh 1.5ml Eppendorf DNA LoBind tube for loading the Flongle, as follows:

Component	**Volume (uL)**
Sequencing Buffer II (SBII)	15
Loading Beads II (LBII)	10
DNA Library	5
Total	30

- To add the Sequencing Mix to the flow cell, ensure that there is no air gap in the sample port or the pipette tip. Place the P100 tip inside the sample port and slowly dispense the Sequencing Mix into the flow cell by twisting the pipette plunger down.
- Seal the Flongle flow cell using the adhesive on the seal tab.
- Replace the sequencing platform lid.

9. Data analysis and Basic Bioinformatic

MinKNOW, the operating software that drives nanopore sequencing devices, carries out several core tasks, including data acquisition, real-time analysis and feedback, local basecalling, and data streaming. MinKNOW produces FAST5 (HDF5) files and/or FASTQ files, based on preferences. FAST5 files contain raw signal data that can be used for basecalling. In our work, we obtain raw data in FASTQ format that contain nucleotide sequence data and their corresponding quality scores. The entire data analysis can be carried in 5 steps starting from raw data acquisition from MinKNOW.



First step of the analysis is basecalling which is usually completed by MinKNOW software. This step includes barcoding/demultiplexing, adapter trimming and alignment. For the rest of the analysis, you need to use bioinformatics workflows on a Linux/Mac computer. You are advised to watch some bioinformatics lectures on YouTube to get a good insight on these workflows. One such easy to follow lecture series from Dr. Simon Cockell of University of Newcastle, UK can be viewed at

https://www.youtube.com/c/SimonCockell/videos

In addition, keeping in view of your upcoming training you are also suggested to watch one recorded video on Nanopore data analysis. This lecture focusses on how to analyse raw data obtained from MinKNOW software using bioinformatics pipelines.

https://nanoporetech.com/resource-centre/bioinformatics-workflows-sars-cov-2-raw-nanopore-reads-consensus-genomes-using

Besides the above, some basic knowledge in Linux, Miniconda, Docker and GitHub are essential. For editing the code, Visual studio code is also a nice to have software that be easily installed on any OS. Linux and Mac operating systems are ideal for bioinformatics. However, if your PC has Windows 10 as OS, you could still install Linux and share all your data between the two operating systems without having to reboot the PC.

If you are a beginner in bioinformatics without any prior knowledge in Linux or any programming language, you can still do the data analysis, as the bioinformatics workflows for Nanopore are meant to cater the needs of biologists who wants to do sequence analysis by themselves.

9.1. Linux

Linux is an open-source operating system that is very secure, fast and reliable. Linux is most preferred for bioinformatics workflow as it can easily handle large datasets.

Installation of Linux on Windows 10 PC or laptop

On a Windows 10 PC or laptop, Linux can be installed as WSL-2 (Windows-subsystem for Linux, version 2). WSL is freely downloadable from Microsoft store on Windows 10 PC.

IMPORTANT: System should be x64 systems: Windows Version 1903 or higher, with Build 18362 or higher.

The information on Windows build can be found at Start > Settings > System > About (under windows specifications)

More details on installation of WSL-2 can be read at the following URL

https://docs.microsoft.com/en-us/windows/wsl/install-win10#manual-installation-steps

Users are advised to follow step-by-step instructions under the section "Install WSL & update to WSL-2" at the above website. When asked for a username, you may give a short name or initials, but without any spaces. When you type your password, you may not see cursor moving, but Linux stores your password. You need to confirm the password by typing again. For beginners in Linux, either Ubuntu 18.04 or 20.04 is ideal as there is lot of reading material on internet. WSL does not have a graphical user interface (GUI) and hence some basic commands are essential for easy manoeuvre at the terminal prompt. A very good intro to Linux can be found at

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

There is lot of literature on Ubuntu on internet. For a start, following websites are useful from bioinformatics perspective.

1. http://www.cellbiol.com/bioinformatics_web_development/chapter-2-the-linux-operating-system-setting-up-a-linux-web-server/basic-linux-shell-commands/

2. https://bioinformaticsworkbook.org/Appendix/Unix/UnixCheatSheet.html#gsc.tab=0

9.2. Conda and Miniconda

Conda is an open-source package management and environment management system that runs on any OS. It has been created for Python programs, although it can package and distribute software for any language. Conda as a package manager helps you find and install packages. You will learn more about this package manager as you use it regularly.

Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others. To know more about Conda / miniconda, please refer to the following website.

https://docs.conda.io/projects/conda/en/latest/

Install Miniconda on WSL

At Ubuntu terminal, you need to type the following commands at $ prompt

```
$ wget https://repo.continuum.io/miniconda/ Miniconda3-py39_4.9.2-Linux-x86_64.sh
```

Click Enter button. The above command will download the miniconda from internet.

```
$ bash Miniconda3-py39_4.9.2-Linux-x86_64.sh
```

The above command will install the miniconda application on your Linux.

If your system is Mac,

```
$ wget https://repo.continuum.io/miniconda/Miniconda3-py39_4.9.2-MacOSX-x86_64.sh
```

Followed by

```
$ bash Miniconda3-py39_4.9.2-MacOSX-x86_64.sh
```

9.3. Docker

Docker is an open-source software platform to build, deploy or execute applications by using containers. As the name indicates, containerization isolate applications with each container having a separate environment at the same time sharing the underlying OS kernel. Docker provides a good platform to run multiple applications at the same time, each needing its own environment. Containerization is very similar to virtual machines, but unlike VMs, containers are faster and use less system resources. Each Docker container has an image file to create a run-time environment for the supposed application.

Installing Docker

Docker can be easily installed on WSL2 based windows by following instructions as mentioned in the below URL

https://docs.docker.com/docker-for-windows/wsl/

9.4. GitHub

GitHub is an Open-source Community where people around the globe work together on Open-source projects and make contributions. GitHub is also a code hosting platform for collaboration and version control. Importantly, GitHub is a repository for bioinformatics pipelines, and in our work, we use GitHub to download Nanopore workflows.

For more information and basic commands on GitHub, please refer to

https://guides.github.com/introduction/git-handbook/

9.5. Visual Studio Code

This editor can be downloaded freely from **https://code.visualstudio.com** depending on the type of OS. This is very useful software to edit the code, besides providing an excellent platform to integrate linux terminal, Git and docker.

10. Full-length 16S rRNA ONT data analysis using Centrifuge

[Centrifuge] is a novel microbial classification engine that enables rapid, accurate and sensitive labeling of reads and quantification of species on desktop computers. The system uses a novel indexing scheme based on the Burrows-Wheeler transform (BWT) and the Ferragina-Manzini (FM) index, optimized specifically for the metagenomic classification problem. Centrifuge requires a relatively small index (4.7 GB for all complete bacterial and viral genomes plus the human genome) and classifies sequences at very high speed, allowing it to process the millions of reads from a typical high-throughput DNA sequencing run within a few minutes. Together these advances enable timely and accurate analysis of large metagenomics data sets on conventional desktop computers

The Centrifuge homepage is http://www.ccb.jhu.edu/software/centrifuge

The Centrifuge paper is available at https://genome.cshlp.org/content/26/12/1721

The Centrifuge poster is available at http://www.ccb.jhu.edu/people/infphilo/data/Centrifuge-poster.pdf

10.1. Quick guide

Installation from source

https://github.com/DaehwanKimLab/centrifuge

```
git clone https://github.com/infphilo/centrifuge
cd centrifuge
make
sudo make install prefix=/usr/local
```

Building indexes

We provide several indexes on the Centrifuge homepage at http://www.ccb.jhu.edu/software/centrifuge. Centrifuge needs sequence and taxonomy files, as well as sequence ID to taxonomy ID mapping. See the MANUAL files for details. We provide a Makefile that simplifies the building of several standard and custom indices

```
cd indices
make p+h+v                  # bacterial, human, and viral genomes [~12G]
make p_compressed           # bacterial genomes compressed at the species level
[~4.2G]
make p_compressed+h+v       # combination of the two above [~8G]
```

It can be used the following commands to run centrifuge and to visualise

centrifuge output (Check where sample?)

to run centrifuge

1. <u>*conda activate centrifuge*</u>

2. centrifuge -x p_compressed+h+v /home/ngs/analysis/*sample.fastq* --report-file report_ *sample*.txt -S results_ *sample*.txt

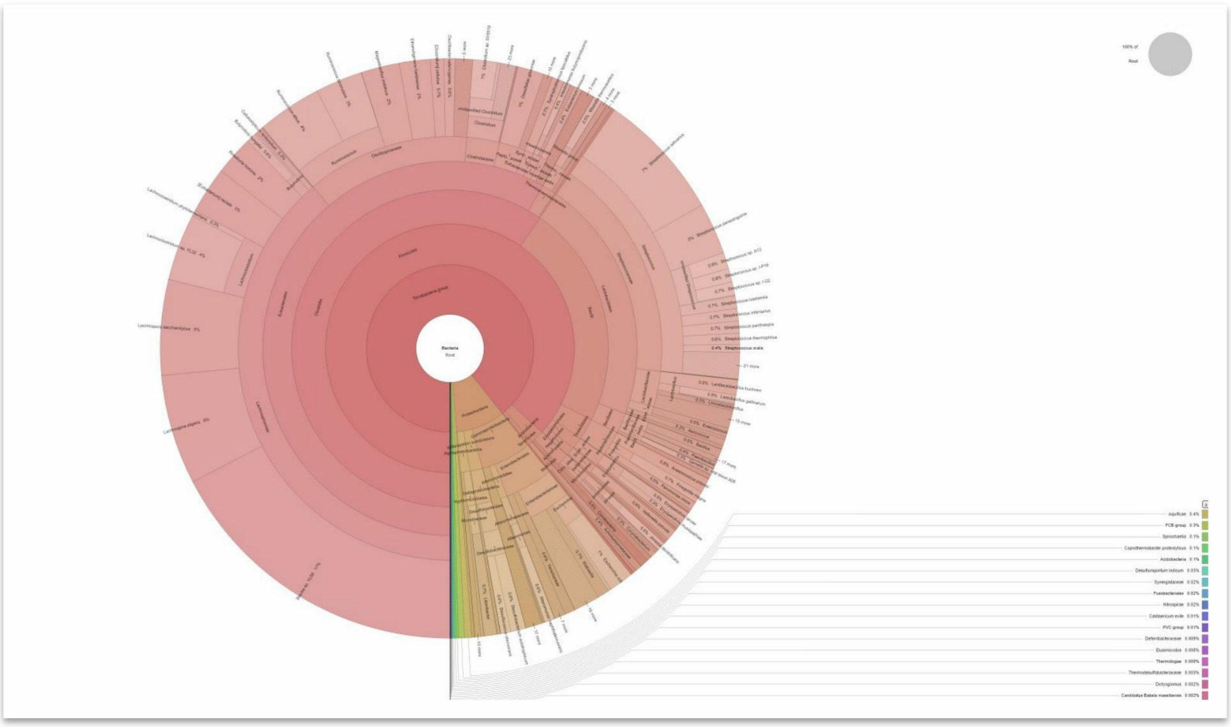
3. centrifuge-kreport -x p_compressed+h+v ./results_ *sample*.txt > evolved-kreport_ *sample*.txt

to visualise centrifuge output data using krona

4. <u>*conda activate krona*</u>

5. cat results_ *sample*.txt | cut -f 1,3 > results_ *sample*.krona

6.ktImportTaxonomy results_ *sample*.krona



10.2. KRONA

Krona allows hierarchical data to be explored with zooming, multi-layered pie charts. Krona charts can be created using an Excel template or KronaTools, which includes support for several bioinformatics tools and raw data formats. The interactive charts are self-contained and can be viewed with any modern web browser.

Installing

https://github.com/marbl/Krona/wiki/Installing

To install KronaTools, unpack the archive, cd to the resulting directory on a command line, and run ./install.pl. If installing to /usr/local (the default), root privileges will be needed (use "sudo ./install.pl" in OS X).
Options for install.pl:

- "--prefix <path>" - scripts will be installed in the bin directory within this path. The default is /usr/local/.
- "--taxonomy <path>" - taxonomy files will be stored in this directory when updateTaxonomy.sh or updateAccessions.sh is run. The default is taxonomy/ within the unpacked KronaTools directory. If the taxonomy database was installed in a previous version of Krona Tools, it can be reused by moving it to the to new KronaTools folder or by pointing to it with this option.

10.3. PAVIAN

evolved-kreport_*sample*.txt file **can be also visualised using Pavian**

Pavian is a interactive browser application for analyzing and visualization metagenomics classification results from classifiers such as Kraken, KrakenUniq, Kraken 2, Centrifuge and MetaPhlAn. Pavian also provides an alignment viewer for validation of matches to a particular genome.

For more information look at the publication at https://doi.org/10.1093/bioinformatics/btz715

Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen identification. FP Breitwieser, SL Salzberg - Bioinformatics, 2020

Thank you for citing the publication if Pavian helps in your research :).

You can try out Pavian at https://fbreitwieser.shinyapps.io/pavian/.

Installation and deployment

Pavian is a R package, and thus requires R to run. Look here for how to install R. On Windows, you probably need to install Rtools. On Ubuntu, install r-base-dev. Once you started R, the following commands will install the package:

```
if (!require(remotes)) { install.packages("remotes") }
remotes::install_github("fbreitwieser/pavian")
```

To run Pavian from R, type

```
pavian::runApp(port=5000)
```

Pavian will then be available at http://127.0.0.1:5000 in the web browser of your choice. Alternatively, you can install and test Pavian with the following command:

```
shiny::runGitHub("fbreitwieser/pavian", subdir = "inst/shinyapp")
```

To try out Pavian, load the example files directly from the interface.

Installing R samtools

The alignment viewer uses Rsamtools. To install this package from Bioconductor, use the following commands

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("Rsamtools")
```

Installing to Shinyapps.io

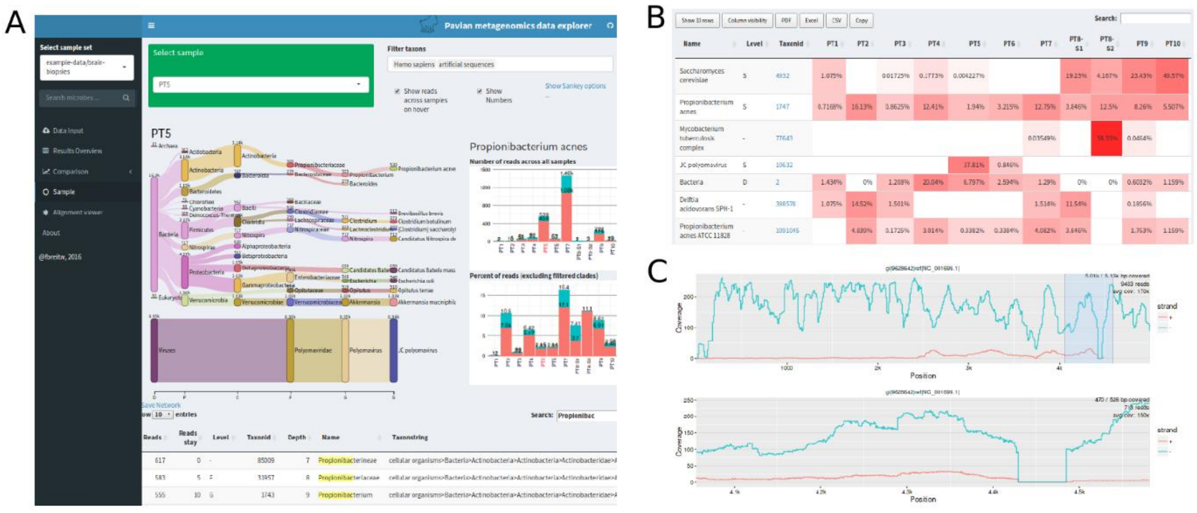
In order to install to Shinyapps.io, because of the Bioconductor repo dependencies, you need to first set the options using setRepositories() in R. At that point a `rsconnect::deployApp('pavian/inst/shinapp/') should work.

Docker image

As an alternative to installing Pavian in R, a Docker image is available at florianbw/pavian. When you run this docker image, Pavian will start automatically on port 80, which you need to make available to the hosting machine. On the shell, you can pull the image and remap the Docker port to port 5000 with the following commands:

```
docker pull 'florianbw/pavian'
docker run --rm -p 5000:80 florianbw/pavian
```

Screenshots


A
B
C
Pavian metagenomics data explorer
Select sample
Filter taxons
PT5
Propionibacterium acnes
Number of reads across all samples
Percent of reads (excluding filtered clades)
Coverage
Position
strand

Name	Level	Taxonid	PT1	PT2	PT3	PT4	PT5	PT6	PT7	PT8-S1	PT8-S2	PT9	PT10
Saccharomyces cerevisiae	S	4932	1.075%		0.01725%	0.1773%	0.004227%			19.23%	4.167%	23.43%	49.57%
Propionibacterium acnes	S	1747	0.7168%	16.13%	0.8625%	12.41%	1.94%	3.215%	12.75%	3.846%	12.5%	8.26%	5.507%
Mycobacterium tuberculosis complex	-	77643							0.03549%		58.33%	0.0464%	
JC polyomavirus	S	10632					37.81%	0.846%					
Bacteria	D	2	1.434%	0%	1.208%	20.04%	6.797%	2.594%	1.29%	0%	0%	0.6032%	1.159%
Delftia acidovorans SPH-1	-	398578	1.075%	14.52%	1.501%				1.514%	11.54%		0.1856%	
Propionibacterium acnes ATCC 11828	-	1091045		4.839%	0.1725%	3.014%	0.3382%	0.3384%	4.082%	3.846%		1.763%	1.159%

11. Full-length 16S rRNA ONT data analysis using Kraken2

Kraken 2 is the newest version of Kraken, a taxonomic classification system using exact k-mer matches to achieve high accuracy and fast classification speeds. This classifier matches each k-mer within a query sequence to the lowest common ancestor (LCA) of all genomes containing the given k-mer. The k-mer assignments inform the classification algorithm. [see: Kraken 1's Webpage for more details].

Kraken 2 provides significant improvements to Kraken 1, with faster database build times, smaller database sizes, and faster classification speeds. These improvements were achieved by the following updates to the Kraken classification program:

1. **Storage of Minimizers:** Instead of storing/querying entire k-mers, Kraken 2 stores minimizers (l-mers) of each k-mer. The length of each l-mer must be ≤ the k-mer length. Each k-mer is treated by Kraken 2 as if its LCA is the same as its minimizer's LCA.
2. **Introduction of Spaced Seeds:** Kraken 2 also uses spaced seeds to store and query minimizers to improve classification accuracy.
3. **Database Structure:** While Kraken 1 saved an indexed and sorted list of k-mer/LCA pairs, Kraken 2 uses a compact hash table. This hash table is a probabilistic data structure that allows for faster queries and lower memory requirements. However, this data structure does have a <1% chance of returning the incorrect LCA or returning an LCA for a non-inserted minimizer. Users can compensate for this possibility by using Kraken's confidence scoring thresholds.
4. **Protein Databases:** Kraken 2 allows for databases built from amino acid sequences. When queried, Kraken 2 performs a six-frame translated search of the query sequences against the database.
5. **16S Databases:** Kraken 2 also provides support for databases not based on NCBI's taxonomy. Currently, these include the 16S databases: Greengenes, SILVA, and RDP.

KRAKEN 2 AND OTHER TOOLS

The following tools are compatible Kraken 2. They are designed to assist users in analyzing and visualizing Kraken results.

Bracken allows users to estimate relative abundances within a specific sample from Kraken 2 classification results. Bracken uses a Bayesian model to estimate abundance at any standard taxonomy level, including species/genus-level abundance.

Pavian has also been developed as a comprehensive visualization program that can compare Kraken 2 classifications across multiple samples.

KrakenTools is a suite of scripts to assist in the analysis of Kraken results. KrakenTools is an ongoing project led by Jennifer Lu.

11.1. System Requirements

- **Disk space**: Construction of a Kraken 2 standard database requires approximately 100 GB of disk space. A test on 01 Jan 2018 of the default installation showed 42 GB of disk space was used to store the genomic library files, 26 GB was used to store the taxonomy information from NCBI, and 29 GB was used to store the Kraken 2 compact hash table.

  Like in Kraken 1, we strongly suggest against using NFS storage to store the Kraken 2 database if at all possible.

- **Memory**: To run efficiently, Kraken 2 requires enough free memory to hold the database (primarily the hash table) in RAM. While this can be accomplished with a ramdisk, Kraken 2 will by default load the database into process-local RAM; the --memory-mapping switch to kraken2 will avoid doing so. The default database size is 29 GB (as of Jan. 2018), and you will need slightly more than that in RAM if you want to build the default database.

- **Dependencies**: Kraken 2 currently makes extensive use of Linux utilities such as sed, find, and wget. Many scripts are written using the Bash shell, and the main scripts are written using Perl. Core programs needed to build the database and run the classifier are written in C++11, and need to be compiled using a somewhat recent version of g++ that will support C++11. Multithreading is handled using OpenMP. Downloads of NCBI data are performed by wget and rsync. Most Linux systems will have all of the above listed programs and development libraries available either by default or via package download.

  Unlike Kraken 1, Kraken 2 does not use an external *k*-mer counter. However, by default, Kraken 2 will attempt to use the dustmasker or segmasker programs provided as part of NCBI's BLAST suite to mask low-complexity regions (see [Masking of Low-complexity Sequences]).

  MacOS NOTE: MacOS and other non-Linux operating systems are *not* explicitly supported by the developers, and MacOS users should refer to the Kraken-users group for support in installing the appropriate utilities to allow for full operation of Kraken 2. We will attempt to use MacOS-compliant code when possible, but development and testing time is at a premium and we cannot guarantee that Kraken 2 will install and work to its full potential on a default installation of MacOS.

  In particular, we note that the default MacOS X installation of GCC does not have support for OpenMP. Without OpenMP, Kraken 2 is limited to single-threaded operation, resulting in slower build and classification runtimes.

- **Network connectivity**: Kraken 2's standard database build and download commands expect unfettered FTP and rsync access to the NCBI FTP server. If you're working behind a proxy, you may need to set certain environment variables (such as ftp_proxy or RSYNC_PROXY) in order to get these commands to work properly.
  Kraken 2's scripts default to using rsync for most downloads; however, you may find that your network situation prevents use of rsync. In such cases, you can try the --use-ftp option to kraken2-build to force the downloads to occur via FTP.
- **MiniKraken**: At present, users with low-memory computing environments can replicate the "MiniKraken" functionality of Kraken 1 in two ways: first, by increasing the value of *k* with respect to *l* (using the --kmer-len and --minimizer-len options to kraken2-build); and secondly, through downsampling of minimizers (from both the database and query sequences) using a hash function. This second option is performed if the --max-db-size option to kraken2-build is used; however, the two options are not mutually exclusive. In a difference from Kraken 1, Kraken 2 does not require building a full database and then shrinking it to obtain a reduced database.

11.2. Installation

To begin using Kraken 2, you will first need to install it, and then either download or create a database.

Kraken 2 consists of two main scripts (kraken2 and kraken2-build), along with several programs and smaller scripts. As part of the installation process, all scripts and programs are installed in the same directory. After installation, you can move the main scripts elsewhere, but moving the other scripts and programs requires editing the scripts and changing the $KRAKEN2_DIR variables in the main scripts.
Once an install directory is selected, you need to run the following command in the directory where you extracted the Kraken 2 source:

```
./install_kraken2.sh $KRAKEN2_DIR
```

(Replace $KRAKEN2_DIR above with the directory where you want to install Kraken 2's programs/scripts.)
The install_kraken2.sh script should compile all of Kraken 2's code and setup your Kraken 2 program directory. Installation is successful if you see the message "Kraken 2 installation complete."
Once installation is complete, you may want to copy the main Kraken 2 scripts into a directory found in your PATH variable (e.g., "$HOME/bin"):

```
cp $KRAKEN2_DIR/kraken2{,-build,-inspect} $HOME/bin
```

After installation, you're ready to either create or download a database.

11.3. Kraken 2 Databases

A Kraken 2 database is a directory containing at least 3 files:

- hash.k2d: Contains the minimizer to taxon mappings
- opts.k2d: Contains information about the options used to build the database
- taxo.k2d: Contains taxonomy information used to build the database

None of these three files are in a human-readable format. Other files may also be present as part of the database build process, and can, if desired, be removed after a successful build of the database.

In interacting with Kraken 2, you should not have to directly reference any of these files, but rather simply provide the name of the directory in which they are stored. Kraken 2 allows both the use of a standard database as well as custom databases; these are described in the sections [Standard Kraken 2 Database] and [Custom Databases] below, respectively.

11.4. Standard Kraken 2 Database

To create the standard Kraken 2 database, you can use the following command:

```
kraken2-build --standard --db $DBNAME
```

(Replace "$DBNAME" above with your preferred database name/location. Please note that the database will use approximately 100 GB of disk space during creation, with the majority of that being reference sequences or taxonomy mapping information that can be removed after the build.)
This will download NCBI taxonomic information, as well as the complete genomes in RefSeq for the bacterial, archaeal, and viral domains, along with the human genome and a collection of known vectors (UniVec_Core). After downloading all this data, the

build process begins; this can be the most time-consuming step. If you have multiple processing cores, you can run this process with multiple threads, e.g.:

```
kraken2-build --standard --threads 24 --db $DBNAME
```

Using 32 threads on an AWS EC2 r4.8xlarge instance with 16 dual-core hyperthreaded 2.30 GHz CPUs and 244 GB of RAM, the build process took approximately 35 minutes in Jan. 2018.

The build process itself has two main steps, each of which requires passing over the contents of the reference library:

1. **Estimation** of the capacity needed in the Kraken 2 compact hash table. This uses a low-memory method to reliably estimate the number of minimizers present in the reference library given the selected parameters *k* and *l*.
2. **Population** of the hash table (and conversion of the taxonomy to an internal format). This step is a second pass over the reference library to find minimizers and then place them in the database.

(There is one other preliminary step where sequence IDs are mapped to taxonomy IDs, but this is usually a rather quick process and is mostly handled during library downloading.)

Unlike Kraken 1's build process, Kraken 2 does not perform checkpointing after the estimation step. This is because the estimation step is dependent on the selected *k* and *l* values, and if the population step fails, it is likely because *k* needs to be increased (reducing the overall memory requirements).

11.5. Classification

To classify a set of sequences, use the kraken2 command:

```
kraken2 --db $DBNAME seqs.fa
```

Output will be sent to standard output by default. The files containing the sequences to be classified should be specified on the command line. Sequences can also be provided through standard input using the special filename /dev/fd/0.

The kraken2 program allows several different options:

- **Multithreading**: Use the --threads NUM switch to use multiple threads.
- **Quick operation**: Rather than searching all *l*-mers in a sequence, stop classification after the first database hit; use --quick to enable this mode.
- **Hit group threshold**: The option --minimum-hit-groups will allow you to require multiple hit groups (a group of overlapping k-mers that share a common minimizer that is found in the hash table) be found before declaring a sequence classified, which can be especially useful with custom databases when testing to see if sequences either do or do not belong to a particular genome.
- **Sequence filtering**: Classified or unclassified sequences can be sent to a file for later processing, using the --classified-out and --unclassified-out switches, respectively.
- **Output redirection**: Output can be directed using standard shell redirection (| or >), or using the --output switch.
- **Compressed input**: Kraken 2 can handle gzip and bzip2 compressed files as input by specifying the proper switch of --gzip-compressed or --bzip2-compressed.

- **Input format auto-detection**: If regular files (i.e., not pipes or device files) are specified on the command line as input, Kraken 2 will attempt to determine the format of your input prior to classification. You can disable this by explicitly specifying --gzip-compressed or --bzip2-compressed as appropriate. Note that use of the character device file /dev/fd/0 to read from standard input (aka stdin) will **not** allow auto-detection.
- **Paired reads**: Kraken 2 provides an enhancement over Kraken 1 in its handling of paired read data. Rather than needing to concatenate the pairs together with an N character between the reads, Kraken 2 is able to process the mates individually while still recognizing the pairing information. Using the --paired option to kraken2 will indicate to kraken2 that the input files provided are paired read data, and data will be read from the pairs of files concurrently.
  Usage of --paired also affects the --classified-out and --unclassified-out options; users should provide a # character in the filenames provided to those options, which will be replaced by kraken2 with "_1" and "_2" with mates spread across the two files appropriately. For example:
  ```
    kraken2 --paired --classified-out cseqs#.fq seqs_1.fq seqs_2.fq
  ```
  will put the first reads from classified pairs in cseqs_1.fq, and the second reads from those pairs in cseqs_2.fq.

To get a full list of options, use kraken2 --help.

11.6. Output Formats

Standard Kraken Output Format

Each sequence (or sequence pair, in the case of paired reads) classified by Kraken 2 results in a single line of output. Kraken 2's output lines contain five tab-delimited fields; from left to right, they are:

1. "C"/"U": a one letter code indicating that the sequence was either classified or unclassified.

2. The sequence ID, obtained from the FASTA/FASTQ header.

3. The taxonomy ID Kraken 2 used to label the sequence; this is 0 if the sequence is unclassified.

4. The length of the sequence in bp. In the case of paired read data, this will be a string containing the lengths of the two sequences in bp, separated by a pipe character, e.g. "98|94".

5. A space-delimited list indicating the LCA mapping of each *k*-mer in the sequence(s). For example, "562:13 561:4 A:31 0:1 562:3" would indicate that:

   - the first 13 *k*-mers mapped to taxonomy ID #562
   - the next 4 *k*-mers mapped to taxonomy ID #561
   - the next 31 *k*-mers contained an ambiguous nucleotide
   - the next *k*-mer was not in the database
   - the last 3 *k*-mers mapped to taxonomy ID #562

Note that paired read data will contain a "|:|" token in this list to indicate the end of one read and the beginning of another.
When Kraken 2 is run against a protein database (see [Translated Search]), the LCA hitlist will contain the results of querying all six frames of each sequence. Reading frame data is separated by a "-:-" token.

Kraken 1 offered a kraken-translate and kraken-report script to change the output into different formats. Through the use of kraken2 --use-names, Kraken 2 will replace the taxonomy ID column with the scientific name and the taxonomy ID in parenthesis (e.g., "Bacteria (taxid 2)" instead of "2"), yielding similar functionality to Kraken 1's kraken-translate script. The sample report functionality now exists as part of the kraken2 script, with the use of the --report option; the sample report formats are described below.

Sample Report Output Format

Like Kraken 1, Kraken 2 offers two formats of sample-wide results. Kraken 2's standard sample report format is tab-delimited with one line per taxon. The fields of the output, from left-to-right, are as follows:

1. Percentage of fragments covered by the clade rooted at this taxon
2. Number of fragments covered by the clade rooted at this taxon
3. Number of fragments assigned directly to this taxon
4. A rank code, indicating (U)nclassified, (R)oot, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies. Taxa that are not at any of these 10 ranks have a rank code that is formed by using the rank code of the closest ancestor rank with a number indicating the distance from that rank. E.g., "G2" is a rank code indicating a taxon is between genus and species and the grandparent taxon is at the genus rank.
5. NCBI taxonomic ID number
6. Indented scientific name

The scientific names are indented using space, according to the tree structure specified by the taxonomy.

By default, taxa with no reads assigned to (or under) them will not have any output produced. However, if you wish to have all taxa displayed, you can use the --report-zero-counts switch to do so. This can be useful if you are looking to do further downstream analysis of the reports, and want to compare samples. Sorting by the taxonomy ID (using sort -k5,5n) can provide a consistent line ordering between reports.
In addition, we also provide the option --use-mpa-style that can be used in conjunction with --report. This option provides output in a format similar to MetaPhlAn's output. The output with this option provides one taxon per line, with a lowercase version of the rank codes in Kraken 2's standard sample report format (except for 'U' and 'R'), two underscores, and the scientific name of the taxon (e.g., "d__Viruses"). The full taxonomy of each taxon (at the eight ranks considered) is given, with each rank's name separated by a pipe character (e.g., "d__Viruses|o_Caudovirales"). Following this version of the taxon's scientific name is a tab and the number of fragments assigned to the clade rooted at that taxon.

11.7. Translated Search

Kraken 2 allows users to perform a six-frame translated search, similar to the well-known BLASTX program. To do this, Kraken 2 uses a reduced 15 amino acid alphabet and stores amino acid minimizers in its database. LCA results from all 6 frames are combined to yield a set of LCA hits, which is then resolved in the same manner as in Kraken's normal operation.

To build a protein database, the --protein option should be given to kraken2-build (either along with --standard, or with all steps if building a custom database).

11.7.1. Custom Databases

We realize the standard database may not suit everyone's needs. Kraken 2 also allows creation of customized databases.

To build a custom database:

1. Install a taxonomy. Usually, you will just use the NCBI taxonomy, which you can easily download using:

2. kraken2-build --download-taxonomy --db $DBNAME
   This will download the accession number to taxon maps, as well as the taxonomic name and tree information from NCBI. These files can be found in $DBNAME/taxonomy/ . If you need to modify the taxonomy, edits can be made to the names.dmp and nodes.dmp files in this directory; you may also need to modify the *.accession2taxid files appropriately.
   Some of the standard sets of genomic libraries have taxonomic information associated with them, and don't need the accession number to taxon maps to build the database successfully. These libraries include all those available through the --download-library option (see next point), except for the plasmid and non-redundant databases. If you are not using custom sequences (see the --add-to-library option) and are not using one of the plasmid or non-redundant database libraries, you may want to skip downloading of the accession number to taxon maps. This can be done by passing --skip-maps to the kraken2-build --download-taxonomy command.
3. Install one or more reference libraries. Several sets of standard genomes/proteins are made easily available through kraken2-build:
   - archaea: RefSeq complete archaeal genomes/proteins
   - bacteria: RefSeq complete bacterial genomes/proteins
   - plasmid: RefSeq plasmid nucleotide/protein sequences
   - viral: RefSeq complete viral genomes/proteins
   - human: GRCh38 human genome/proteins
   - fungi: RefSeq complete fungal genomes/proteins
   - plant: RefSeq complete plant genomes/proteins
   - protozoa: RefSeq complete protozoan genomes/proteins
   - nr: NCBI non-redundant protein database
   - nt: NCBI non-redundant nucleotide database
   - UniVec: NCBI-supplied database of vector, adapter, linker, and primer sequences that may be contaminating sequencing projects and/or assemblies
   - UniVec_Core: A subset of UniVec chosen to minimize false positive hits to the vector database

- env_nr and env_nt are no longer supported by NCBI and therefore are no longer available for download.

To download and install any one of these, use the --download-library switch, e.g.:
kraken2-build --download-library bacteria --db $DBNAME
Multiple libraries can be downloaded into a database prior to building by issuing multiple kraken2-build --download-library commands, e.g.:
kraken2-build --download-library archaea --db $DBNAME
kraken2-build --download-library viral --db $DBNAME
The above commands would prepare a database that would contain archaeal and viral genomes; the --build option (see below) will still need to be used after downloading these libraries to actually build the database, however.
(Note that downloading nr requires use of the --protein option, and that UniVec and UniVec_Core are incompatible with the --protein option.)

Other genomes can also be added, but such genomes must meet certain requirements:

- Sequences must be in a FASTA file (multi-FASTA is allowed)
- Each sequence's ID (the string between the > and the first whitespace character on the header line) must contain either an NCBI accession number to allow Kraken 2 to lookup the correct taxa, or an explicit assignment of the taxonomy ID using kraken:taxid (see below).

Sequences not downloaded from NCBI may need their taxonomy information assigned explicitly. This can be done using the string kraken:taxid|XXX in the sequence ID, with XXX replaced by the desired taxon ID. For example, to put a known adapter sequence in taxon 32630 ("synthetic construct"), you could use the following:
>sequence16|kraken:taxid|32630 Adapter sequence
CAAGCAGAAGACGGCATACGAGATCTTCGAGTGACTGGAGTTCCTTGGC
ACCCGAGAATTCCA
The kraken:taxid string must begin the sequence ID or be immediately preceded by a pipe character (|). Explicit assignment of taxonomy IDs in this manner will override the accession number mapping provided by NCBI.
If your genomes meet the requirements above, then you can add each sequence to your database's genomic library using the --add-to-library switch, e.g.:
kraken2-build --add-to-library chr1.fa --db $DBNAME
kraken2-build --add-to-library chr2.fa --db $DBNAME
Note that if you have a list of files to add, you can do something like this in bash:

```
for file in chr*.fa
do
   kraken2-build --add-to-library $file --db $DBNAME
done
```

Or even add all *.fa files found in the directory genomes:
find genomes/ -name '*.fa' -print0 | xargs -0 -I{} -n1 kraken2-build --add-to-library {} --db $DBNAME

(You may also find the -P option to xargs useful to add many files in parallel if you have multiple processors.)

4. Once your library is finalized, you need to build the database. This can be done with the command:

5. kraken2-build --build --db $DBNAME
   The --threads option is also helpful here to reduce build time.
   By default, the values of *k* and *l* are 35 and 31, respectively (or 15 and 12 for protein databases). These values can be explicitly set with the --kmer-len and minimizer-len options, however. Note that the minimizer length must be no more than 31 for nucleotide databases, and 15 for protein databases. Additionally, the minimizer length *l* must be no more than the *k*-mer length. There is no upper bound on the value of *k*, but sequences less than *k* bp in length cannot be classified.

   Kraken 2 also utilizes a simple spaced seed approach to increase accuracy. A number *s* < *l*/4 can be chosen, and *s* positions in the minimizer will be masked out during all comparisons. Masked positions are chosen to alternate from the second-to-last position in the minimizer; e.g., *s* = 5 and *l* = 31 will result in masking out the 0 positions shown here:

   111 1111 1111 1111 1111 1101 0101 0101
   By default, *s* = 7 for nucleotide databases, and *k* = 0 for protein databases. This can be changed using the --minimizer-spaces option along with the --build task of kraken2-build.

A full list of options for kraken2-build can be obtained using kraken2-build --help.
After building a database, if you want to reduce the disk usage of the database, you can use the --clean option for kraken2-build to remove intermediate files from the database directory.

11.7.2. Masking of Low-complexity Sequences

Low-complexity sequences, e.g. "ACACACACACACACACACACACACAC", are known to occur in many different organisms and are typically less informative for use in alignments; the BLAST programs often mask these sequences by default. Using this masking can help prevent false positives in Kraken 2's results, and so we have added this functionality as a default option to Kraken 2's library download/addition process.

Kraken 2 uses two programs to perform low-complexity sequence masking, both available from NCBI: dustmasker, for nucleotide sequences, and segmasker, for amino acid sequences. These programs are available as part of the NCBI BLAST+ suite. If these programs are not installed on the local system and in the user's PATH when trying to use kraken2-build, the database build will fail. Users who do not wish to install these programs can use the --no-masking option to kraken2-build in conjunction with any of the --download-library, --add-to-library, or --standard options; use of the --no-masking option will skip masking of low-complexity sequences during the build of the Kraken 2 database.

11.7.3. Special Databases

To support some common use cases, we provide the ability to build Kraken 2 databases using data from various external databases. These external databases may not follow the NCBI taxonomy, and so we've provided mechanisms to automatically create a taxonomy that will work with Kraken 2 (although such taxonomies may not be identical to NCBI's).

To build one of these "special" Kraken 2 databases, use the following command:

```
kraken2-build --db $DBNAME --special TYPE
```

where the TYPE string is one of the database names listed below.
At present, the "special" Kraken 2 database support we provide is limited to pre-packaged solutions for some public 16S sequence databases, but this may grow in the future.

11.7.4. 16S Databases

For targeted 16S sequencing projects, a normal Kraken 2 database using whole genome data may use more resources than necessary. A Kraken 2 database created from a well-curated genomic library of just 16S data can provide both a more efficient solution as well as a more accurate set of predictions for such projects. We provide support for building Kraken 2 databases from three publicly available 16S databases:

- Greengenes (Kraken 2 database name: greengenes), using all available 16S data.
- RDP (Kraken 2 database name: rdp), using the bacterial and archaeal 16S data.
- SILVA (Kraken 2 database name: silva), using the Small subunit NR99 sequence set.

Note that these databases may have licensing restrictions regarding their data, and it is your responsibility to ensure you are in compliance with those restrictions; please visit the databases' websites for further details. The kraken2-build script only uses publicly available URLs to download data and then converts that data into a form compatible for use with Kraken 2.
Furthermore, if you use one of these databases in your research, please visit the corresponding database's website to determine the appropriate and up-to-date citation.

11.7.5. Confidence Scoring

At present, we have not yet developed a confidence score with a probabilistic interpretation for Kraken 2. However, we have developed a simple scoring scheme that has yielded good results for us, and we've made that available in Kraken 2 through use of the --confidence option to kraken2. The approach we use allows a user to specify a threshold score in the [0,1] interval; the classifier then will adjust labels up the tree until the label's score (described below) meets or exceeds that threshold. If a label at the root of the taxonomic tree would not have a score exceeding the threshold, the sequence is called unclassified by Kraken 2 when this threshold is applied.
A sequence label's score is a fraction *C*/*Q*, where *C* is the number of *k*-mers mapped to LCA values in the clade rooted at the label, and *Q* is the number of *k*-mers in the sequence that lack an ambiguous nucleotide (i.e., they were queried against the

database). Consider the example of the LCA mappings in Kraken 2's output given earlier:

"562:13 561:4 A:31 0:1 562:3" would indicate that:

- the first 13 *k*-mers mapped to taxonomy ID #562
- the next 4 *k*-mers mapped to taxonomy ID #561
- the next 31 *k*-mers contained an ambiguous nucleotide
- the next *k*-mer was not in the database
- the last 3 *k*-mers mapped to taxonomy ID #562

In this case, ID #561 is the parent node of #562. Here, a label of #562 for this sequence would have a score of *C*/*Q* = (13+3)/(13+4+1+3) = 16/21. A label of #561 would have a score of *C*/*Q* = (13+4+3)/(13+4+1+3) = 20/21. If a user specified a --confidence threshold over 16/21, the classifier would adjust the original label from #562 to #561; if the threshold was greater than 20/21, the sequence would become unclassified.

11.7.6. Inspecting a Kraken 2 Database's Contents

The kraken2-inspect script allows users to gain information about the content of a Kraken 2 database. The output format of kraken2-inspect is identical to the reports generated with the --report option to kraken2. Instead of reporting how many reads in input data classified to a given taxon or clade, as kraken2's --report option would, the kraken2-inspect script will report the number of minimizers in the database that are mapped to the various taxa/clades. For example, the first five lines of kraken2-inspect's output on an example database might look like this:

```
$ kraken2-inspect --db EXAMPLE_DB | head -5
100.00% 1770368409      1581179 R       1       root
 96.50% 1708407622      58003   R1      131567    cellular organisms
 91.28% 1615910070      985309  D       2           Bacteria
 43.89% 777062062       1312736 P       1224          Proteobacteria
 18.62% 329590216       555667  C       1236            Gammaproteobacteria
```

This output indicates that 555667 of the minimizers in the database map directly to the Gammaproteobacteria class (taxid #1236), and 329590216 (18.62%) of the database's minimizers map to a taxon in the clade rooted at Gammaproteobacteria. For more information on kraken2-inspect's options, use its --help option.

11.7.7. Distinct minimizer count information

The KrakenUniq project extended Kraken 1 by, among other things, reporting an estimate of the number of distinct k-mers associated with each taxon in the input sequencing data. This allows users to better determine if Kraken's classifications are due to reads distributed throughout a reference genome, or due to only a small segment of a reference genome (and therefore likely false positive).

Thanks to the generosity of KrakenUniq's developer Florian Breitwieser in allowing parts of the KrakenUniq source code to be licensed under Kraken 2's MIT license, this distinct counting estimation is now available in Kraken 2. Development work by Martin Steinegger and Ben Langmead helped bring this functionality to Kraken 2.

At present, this functionality is an optional *experimental feature* -- meaning that we may later alter it in a way that is not backwards compatible with previous versions of the feature.

To use this functionality, simply run the kraken2 script with the additional --report-minimizer-data flag along with --report, e.g.:

```
kraken2 --db $DBNAME --report k2_report.txt --report-minimizer-data \
    --output k2_output.txt sequence_data.fq
```

This will put the standard Kraken 2 output (formatted as described in [Standard Kraken Output Format]) in k2_output.txt and the report information in k2_report.txt. Within the report file, two additional columns will be present, e.g.:

normal report format:

```
36.40    182    182    S2    211044    Influenza A virus (A/Puerto Rico/8/1934(H1N1))
```

modified report format:

```
36.40    182    182    1688    18    S2    211044    Influenza A virus (A/Puerto Rico/8/1934(H1N1))
```

In this modified report format, the two new columns are the fourth and fifth, respectively representing the number of minimizers found to be associated with a taxon in the read sequences (1688), and the estimate of the number of distinct minimizers associated with a taxon in the read sequence data (18). This would indicate that although 182 reads were classified as belonging to H1N1 influenza, only 18 distinct minimizers led to those 182 classifications.

The format with the --report-minimizer-data flag, then, is similar to that described in [Sample Report Output Format], but slightly different. The fields in this new format, from left-to-right, are:

1. Percentage of fragments covered by the clade rooted at this taxon
2. Number of fragments covered by the clade rooted at this taxon
3. Number of fragments assigned directly to this taxon
4. Number of minimizers in read data associated with this taxon (**new**)
5. An estimate of the number of distinct minimizers in read data associated with this taxon (**new**)
6. A rank code, indicating (U)nclassified, (R)oot, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies. Taxa that are not at any of these 10 ranks have a rank code that is formed by using the rank code of the closest ancestor rank with a number indicating the distance from that rank. E.g., "G2" is a rank code indicating a taxon is between genus and species and the grandparent taxon is at the genus rank.
7. NCBI taxonomic ID number
8. Indented scientific name

We decided to make this an optional feature so as not to break existing software that processes Kraken 2's standard report format. However, this new format can be converted to the standard report format with the command:

```
cut -f1-3,6-8 k2_new_report.txt > k2_std_report.txt
```

As noted above, this is an *experimental feature*. We intend to continue development on this feature, and may change the new format and/or its information if we determine it to be necessary.

For background on the data structures used in this feature and their interaction with Kraken, please read the KrakenUniq paper, and please cite that paper if you use this functionality as part of your work.

11.7.8. Kraken 2 Environment Variables

The kraken2 and kraken2-inpsect scripts supports the use of some environment variables to help in reducing command line lengths:

- **KRAKEN2_NUM_THREADS**: if the --threads option is not supplied to kraken2, then the value of this variable (if it is set) will be used as the number of threads to run kraken2. (This variable does not affect kraken2-inspect.)
- **KRAKEN2_DB_PATH**: much like the PATH variable is used for executables by your shell, KRAKEN2_DB_PATH is a colon-separated list of directories that will be searched for the database you name if the named database does not have a slash (/) character. By default, Kraken 2 assumes the value of this variable is "." (i.e., the current working directory). This variable can be used to create one (or more) central repositories of Kraken databases in a multi-user system. Example usage in bash:
- export KRAKEN2_DB_PATH="/home/user/my_kraken2_dbs:/data/kraken2_dbs:"

  This will cause three directories to be searched, in this order:

  i. /home/user/my_kraken2_dbs
  ii. /data/kraken2_dbs
  iii. the current working directory (caused by the empty string as the third colon-separated field in the KRAKEN2_DB_PATH string)

  The search for a database will stop when a name match is found; if two directories in the KRAKEN2_DB_PATH have databases with the same name, the directory of the two that is searched first will have its database selected.
  If the above variable and value are used, and the databases /data/kraken2_dbs/mainDB and ./mainDB are present, then
  kraken2 --db mainDB sequences.fa
  will classify sequences.fa using /data/kraken_dbs/mainDB; if instead you wanted to use the mainDB present in the current directory, you would need to specify a directory path to that database in order to circumvent searching, e.g.:
  kraken2 --db ./mainDB sequences.fa
  Note that the KRAKEN2_DB_PATH directory list can be skipped by the use of any absolute (beginning with /) or relative pathname (including at least one /) as the database name.
- **KRAKEN2_DEFAULT_DB**: if no database is supplied with the --db option, the database named in this variable will be used instead. Using this variable, you can avoid using --db if you only have a single database that you usually use, e.g. in bash:
- export KRAKEN2_DEFAULT_DB="/home/user/kraken2db"
- kraken2 sequences.fa > kraken2.output

This will classify sequences.fa using the /home/user/kraken2db database. Note that the value of KRAKEN2_DEFAULT_DB will also be interpreted in the context of the value of KRAKEN2_DB_PATH if you don't set KRAKEN2_DEFAULT_DB to an absolute or relative pathname. Given the earlier example in this section, the following:

```
  export KRAKEN2_DEFAULT_DB="mainDB"
  kraken2 sequences.fa
```

will use /data/kraken_dbs/mainDB to classify sequences.fa.

To run kraken2, it can be used the following command

kraken2 --/home/ngs/kraken2/DB --/home/ngs/kraken2/sample.fastq –report /home/ngs/kraken2/results/sample.kreport –output /home/ngs/kraken2/results/sample.kraken

output files:

sample.kreport

sample.kraken

sample.kreport format file can visualise using Pavian.

12. Full-length 16S rRNA ONT data analysis using Bracken

Bracken (**B**ayesian **R**eestimation of **A**bundance with **K**rak**EN**) is a highly accurate statistical method that computes the abundance of species in DNA sequences from a metagenomics sample. Braken uses the taxonomy labels assigned by Kraken, a highly accurate metagenomics classification algorithm, to estimate the number of reads originating from each species present in a sample. Kraken classifies reads to the best matching location in the taxonomic tree, but does not estimate abundances of species. We use the Kraken database itself to derive probabilities that describe how much sequence from each genome is identical to other genomes in the database, and combine this information with the assignments for a particular sample to estimate abundance at the species level, the genus level, or above. Combined with the Kraken classifier, Bracken produces accurate species- and genus-level abundance estimates even when a sample contains two or more near-identical species.

12.1. Installation

Bracken is a companion program to Kraken 1 or Kraken 2 While Kraken classifies reads to multiple levels in the taxonomic tree, Bracken allows estimation of abundance at a single level using those classifications (e.g. Bracken can estimate abundance of species within a sample).

Prior to installing Bracken, please install Kraken: Kraken can be downloaded from here: http://ccb.jhu.edu/software/kraken/

Easy Bracken Installation:

```
bash install_bracken.sh
```

Hard Bracken Installation:

```
cd src/ && make
Add bracken/bracken-build and scripts in src/ to your PATH
```

12.2. RUNNING BRACKEN: EASY VERSION

Steps 0/1 are run once per database. If you would like to generate Bracken files for multiple read lengths, repeat Step 1 specifying the same database but different read lengths. The script will skip any step already complete.

If you run Kraken using one of the pre-built MiniKraken databases, you can find corresponding Bracken files here. Do not run bracken-build with MiniKraken.

Step 0: Build a Kraken 1.0 or Kraken 2.0 database

```
    kraken-build --db ${KRAKEN_DB} --threads ${THREADS}
    kraken2-build --db ${KRAKEN_DB} --threads ${THREADS}
```

- `${KRAKEN_DB}` is the path to a built Kraken database which also must contain:
    - the taxonomy/nodes.dmp file
    - and library sequences `*.fna`, `*.fa`, or `*.fasta` in the `library` directory.

Step 1: Generate the Bracken database file (databaseXmers.kmer_distrib)

It is highly encouraged for users to run the following scripts with 10-20 threads. [if run single-threaded, kraken/kraken2 and kmer2read_distr will take hours-days] Please note that the flags for this script are single lettered

If Kraken 1.0 or Kraken 2.0 is included in your PATH, run the following

```
    bracken-build -d ${KRAKEN_DB} -t ${THREADS} -k ${KMER_LEN} -l ${READ_LEN}
```

Otherwise, direct the program using "-x" to the installation/location of the ./kraken or ./kraken2 scripts

```
    bracken-build -d ${KRAKEN_DB} -t ${THREADS} -k ${KMER_LEN} -l ${READ_LEN} -x
${KRAKEN_INSTALLATION}

        `${KRAKEN_DB}`  = location of the built Kraken 1.0 or Kraken 2.0 database
        `${THREADS}`     = number of threads to use with Kraken and the Bracken
scripts
        `${KMER_LEN}`   = length of kmer used to build the Kraken database
                                Kraken 1.0 default kmer length = 31
                                Kraken 2.0 default kmer length = 35
                                Default set in the script is 35.
        `${READ_LEN}`   = the read length of your data
                                e.g., if you are using 100 bp reads, set it to
`100`.
```

Step 2: Run Kraken 1.0 or Kraken 2.0 AND Generate a report file

Kraken 1.0 requires a 2-step process to generate the report file needed by Bracken

```
    kraken --db ${KRAKEN_DB} --threads ${THREADS} ${SAMPLE}.fq > ${SAMPLE}.kraken
    kraken-report --db ${KRAKEN_DB} ${SAMPLE}.kraken > ${SAMPLE}.kreport
```

Kraken 2.0 requires the addition of the --report flag

```
    kraken2  --db  ${KRAKEN_DB}  --threads  ${THREADS}  --report  ${SAMPLE}.kreport
${SAMPLE}.fq > ${SAMPLE}.kraken
```

Step 3: Run Bracken for Abundance Estimation

```
    bracken   -d   ${KRAKEN_DB}   -i   ${SAMPLE}.kreport   -o   ${SAMPLE}.bracken   -r
${READ_LEN} -l ${LEVEL} -t ${THRESHOLD}
```

12.3. RUNNING BRACKEN: HARD VERSION

Step 0: Build a Kraken 1.0 or Kraken 2.0 database

```
    kraken-build --db ${KRAKEN_DB} --threads ${THREADS}
    kraken2-build --db ${KRAKEN_DB} --threads ${THREADS}
```

- `${KRAKEN_DB}` is the path to a built Kraken database which also must contain:
  - the taxonomy/nodes.dmp file
  - and library sequences `*.fna`, `*.fa`, or `*.fasta` in the `library` directory.

Step 1: Generate the Bracken database file (databaseXmers.kmer_distrib)

- It is highly encouraged for users to run the following scripts with 20 threads.

Step 1a: Search all library input sequences against the database

Run the following scripts WITHIN the Kraken database folder:

```
    kraken --db=${KRAKEN_DB} --threads=10 <( find -L library \(-name "*.fna" -o -
name "*.fa" -o -name "*.fasta" \) -exec cat {} + )  > database.kraken
    kraken2 --db=${KRAKEN_DB} --threads=10 <( find -L library \(-name "*.fna" -o -
name "*.fa" -o -name "*.fasta" \) -exec cat {} + )  > database.kraken
```

Step 1b: Compute classifications for each perfect read from one of the input sequences

```
    /src/kmer2read_distr  --seqid2taxid  ${KRAKEN_DB}/seqid2taxid.map  --taxonomy
${KRAKEN_DB}/taxonomy          --kraken          database.kraken          --output
database${READ_LEN}mers.kraken
        -k ${KMER_LEN} -l ${READ_LEN} -t ${THREADS}

        `${KRAKEN_DB}`  = location of the built Kraken 1.0 or Kraken 2.0 database
        `${THREADS}`    = number of threads to use [recommended: 20]
        `${KMER_LEN}`   = length of kmer used to build the Kraken database
                                Kraken 1.0 default kmer length = 31
                                Kraken 2.0 default kmer length = 35
                                [default: 35]
        `${READ_LEN}`   = the read length of your data
                                e.g., if you are using 100 bp reads, set it to
`100`.
```

Step 1c: Generate the kmer distribution file

The kmer distribution file is generated using the following command line:

```
python generate_kmer_distribution.py -i database${READ_LEN}mers.kraken -o database${READ_LEN}mers.kmer_distrib
```

Step 2: Run Kraken 1.0 or Kraken 2.0 AND Generate a report file

Kraken 1.0 requires a 2-step process to generate the report file needed by Bracken

```
    kraken --db ${KRAKEN_DB} --threads ${THREADS} ${SAMPLE}.fq > ${SAMPLE}.kraken
    kraken-report --db ${KRAKEN_DB} ${SAMPLE}.kraken > ${SAMPLE}.kreport
```

Kraken 2.0 requires the addition of the --report flag

```
    kraken2 --db ${KRAKEN_DB} --threads ${THREADS} --report ${SAMPLE}.kreport ${SAMPLE}.fq > ${SAMPLE}.kraken
```

Step 3: Run Bracken for Abundance Estimation

Given the expected kmer distribution for genomes in a kraken database along with a kraken report file, the number of reads belonging to each species (or genus) is estimated using the estimate_abundance.py file, run with the following command line:

python estimate_abundance.py -i ${SAMPLE}.kreport -k database${READ_LEN}mers.kmer_distrib -l ${CLASSIFICATION_LVL} -t ${THRESHOLD} -o ${BRACKEN_OUTPUT_FILE}.bracken
The following required parameters must be specified:

- ${SAMPLE}.kreport - the kraken report generated for a given dataset
- database${READ_LEN}mers.kmer_distrib - the file generated by generate_kmer_distribution.py
- {BRACKEN_OUTPUT_FILE}.bracken - the desired name of the output file to be generated by the code

The following optional parameters may be specified:

- ${CLASSIFICATION_LVL} - Default = 'S'. This specifies that abundance estimation will calculate estimated reads for each species. Other possible options are K (kingdom level), P (phylum), C (class), O (order), F (family), and G (genus).
- ${THRESHOLD} - Default = 10. For species classification, any species with <= 10 (or otherwise specified) reads will not receive any additional reads from higher taxonomy levels when distributing reads for abundance estimation. If another classification level is specified, thresholding will occur at that level.

Output Kraken-Style Bracken Report

By default, this script will also recreate the report file using the new Bracken numbers.

1. The new report file will be found in the same folder as the original report file, with "bracken" included in the name.
2. Levels below the estimate-level will not be printed.
3. Any levels whose reads were below the threshold will not be included

4. Percentages will be re-calculated for the remaining levels
5. Unclassified reads will not be included in the report.

Example abundance estimation

The following sample input and output files are included in the sample_data/ folder: sample_test.report - Kraken report file generated from the kraken-report command. sample_kmer_distr_75mers.txt - example kmer distribution file generated by generate_kmer_distribution.py sample_output_species_abundance.txt - Bracken species abundance estimation for sample_test.report sample_output_bracken.report - Kraken report style file with all reads redistributed to the species level
Due to size constraints, the following files are not included in the sample_data/ folder: sample_test.kraken - Kraken output file used to generate the Kraken report file database.kraken - Initial Kraken classification of every genome database75mers.kraken_cnts - Counting of kmer abundances
The following commands were used to generate each individual file:

1.
```
kraken --db${KRAKEN_DB} --threads=10 sample.fa > sample_test.kraken
 kraken-report --db=${KRAKEN_DB} sample_test.kraken > sample_test.report
```

2.
```
kraken --db=${KRAKEN_DB} --fasta_input --threads=10 <( find -L library -name "*.fna" -o -name "*.fa" -o -name "*.fasta" -exec cat {} + ) > database.kraken perl count-kmer-abubndances.pl --db=${KRAKEN_DB} --read-length=75 database.kraken > database75mers.kraken_cnts
```

3.
```
python generate_kmer_distribution.py -i database75mers.kraken_cnts -o sample_kmer_distr_75mers.txt
```

4.
```
python estimate_abundance.py -i sample_test.report -k sample_kmer_distr_75mers.txt -l S -t 10 -o sample_output_species_abundance.txt
```

12.4. Kraken Tools

For news and updates, refer to the github page: https://github.com/jenniferlu717/KrakenTools/

KrakenTools is a suite of scripts to be used for post-analysis of Kraken/KrakenUniq/Kraken2/Bracken results. Please cite the relevant paper if using KrakenTools with any of the listed programs.

Links to Kraken github pages

1. Kraken 1
2. Kraken 2
3. KrakenUniq
4. Bracken

For issues with any of the above programs, please open a github issue on their respective github pages. This github repository is dedicated to only the scripts provided here.

Scripts included in KrakenTools

1. extract_kraken_reads.py
2. combine_kreports.py
3. kreport2krona.py
4. kreport2mpa.py
5. combine_mpa.py
6. filter_bracken_out.py
7. fix_unmapped.py
8. make_ktaxonomy.py
9. make_kreport.py

Running Scripts:

No installation required. All scripts are run on the command line as described.

Users can make scripts executable by running

```
chmod +x myscript.py
./myscript.py -h
```

extract_kraken_reads.py

This program extract reads classified at any user-specified taxonomy IDs. User must specify the Kraken output file, the sequence file(s), and at least one taxonomy ID. Additional options are specified below. As of April 19, 2021, this script is compatible with KrakenUniq/Kraken2Uniq reports.

1. extract_kraken_reads.py usage/options

python extract_kraken_reads.py

- -k, --kraken MYFILE.KRAKEN.............Kraken output file
- -s, -s1, -1, -U SEQUENCE.FILE..........FASTA/FASTQ sequence file (may be gzipped)
- -s2, -2 SEQUENCE2.FILE.................FASTA/FASTQ sequence file (for paired reads, may be gzipped)
- -o, --output2 OUTPUT.FASTA.............output FASTA/Q file with extracted seqs
- -t, --taxid TID TID2 etc...............list of taxonomy IDs to extract (separated by spaces)

Optional:

- -o2, --output2 OUTPUT.FASTA.............second output FASTA/Q file with extracted seqs (for paired reads)

- --fastq-output.........................Instead of producing FASTA files, print FASTQ files (requires FASTQ input)
- --exclude..............................Instead of finding reads matching specified taxids, finds reads NOT matching specified taxids.
- -r, --report MYFILE.KREPORT.............Kraken report file (required if specifying --include-children or --include-parents)
- --include-children.....................include reads classified at more specific levels than specified taxonomy ID levels.
- --include-parents......................include reads classified at all taxonomy levels between root and the specified taxonomy ID levels.
- --max #................................maximum number of reads to save.
- --append...............................if output file exists, appends reads
- --noappend.............................[default] rewrites existing output file

2. extract_kraken_reads.py input files

Input sequence files must be either FASTQ or FASTA files. Input files can be gzipped or not. The program will automatically detect whether the file is gzipped and whether it is FASTQ or FASTA formatted based on the first character in the file (">" for FASTA, "@" for FASTQ)

3. extract_kraken_reads.py paired input/output

Users that ran Kraken using paired reads should input both read files into extract_kraken_reads.py as follows:

extract_kraken_reads.py -k myfile.kraken -s1 read1.fq -s2 reads2.fq
Given paired reads, the script requires users to provide two output file names to contain extracted reads:

extract_kraken_reads.py -k myfile.kraken -s1 read1.fq -s2 reads2.fq -o extracted1.fq -o2 extracted2.fq
The delimiter (--delimiter or -d) option has been removed.
`extract_kraken_reads.py -k myfile.kraken ... -o reads_S1.fa -o2 reads_s2.fa

4. extract_kraken_reads.py --exclude flag

By default, reads classified at specified taxonomy IDs will be extracted (and any taxids selected using --include-parents/--include-children. However, specifying --exclude will cause the reads NOT classified at any specified taxonomy IDs.
For example:

1. extract_kraken_reads.py -k myfile.kraken ... --taxid 9606 --exclude ==> extract all reads NOT classified as Human (taxid 9606).
2. extract_kraken_reads.py -k myfile.kraken ... --taxid 2 --exclude --include-children ==> extract all reads NOT classified as Bacteria (taxid 2) or any classification in the Bacteria subtree.
3. extract_kraken_reads.py -k myfile.kraken ... --taxid 9606 --exclude --include-parents ==> extract all reads NOT classified as Human or any classification in the direct ancestry of Human (e.g. will exclude reads classified at the Primate, Chordata, or Eukaryota levels).

5. extract_kraken_reads.py --include-parents/--include-children flags

By default, only reads classified exactly at the specified taxonomy IDs will be extracted. Options --include-children and --include parents can be used to extract reads classified within the same lineage as a specified taxonomy ID. For example, given a Kraken report containing the following:

```
[%]     [reads] [lreads][lvl]   [tid]       [name]
100     1000    0       R       1           root
100     1000    0       R1      131567         cellular organisms
100     1000    50      D       2             Bacteria
0.95    950     0       P       1224             Proteobacteria
0.95    950     0       C       1236               Gammaproteobacteria
0.95    950     0       O       91347                Enterobacterales
0.95    950     0       F       543                  Enterobacteriaceae
0.95    950     0       G       561                    Escherichia
0.95    950     850     S       562                      Escherichia coli
0.05    50      50      S1      498388                      Escherichia coli C
0.05    50      50      S1      316401                      Escherichia coli ETEC
```

1. extract_kraken_reads.py [options] -t 562 ==> 850 reads classified as *E. coli* will be extracted
2. extract_kraken_reads.py [options] -t 562 --include-parents ==> 900 reads classified as *E. coli* or Bacteria will be extracted
3. extract_kraken_reads.py [options] -t 562 --include-children ==> 950 reads classified as *E. coli*, *E. coli C*, or *E. coli ETEC* will be extracted
4. extract_kraken_reads.py [options] -t 498388 ==> 50 reads classified as *E. coli C* will be extracted
5. extract_kraken_reads.py [options] -t 498388 --include-parents ==> 950 reads classified as *E. coli C*, *E. coli*, or Bacteria will be extracted
6. extract_kraken_reads.py [options] -t 1 --include-children ==> All classified reads will be extracted

---

combine_kreports.py

This script combines multiple Kraken reports into a combined report file.

1. combine_kreports.py usage/options

python complete_kreports.py

- -r 1.KREPORT 2.KREPORT........................Kraken-style reports to combine
- -o COMBINED.KREPORT...........................Output file

Optional:

- --display-headers.............................include headers describing the samples and columns [all headers start with #]
- --no-headers..................................do not include headers in output

- --sample-names................................give abbreviated names for each sample [default: S1, S2, ... etc]
- --only-combined...............................output uses exact same columns as a single Kraken-style report file. Only total numbers for read counts and percentages will be used. Reads from individual reports will not be included.

2. combine_kreports.py output

Percentage is only reported for the summed read counts, not for each individual sample.

The output file therefore contains the following tab-delimited columns:

- perc............percentage of total reads rooted at this clade
- tot_all ....total reads rooted at this clade (including reads at more specific clades)
- tot_lvl.........total reads at this clade (not including reads at more specific clades)
- 1_all...........reads from Sample 1 rooted at this clade
- 1_lvl...........reads from Sample 1 at this clade
- 2_all..........""
- 2_lvl...........""
- etc..
- lvl_type........Clade level type (R, D, P, C, O, F, G, S....)
- taxid...........taxonomy ID of this clade
- name............name of this clade

---

kreport2krona.py

This program takes a Kraken report file and prints out a krona-compatible TEXT file

1. kreport2krona.py usage/options

python kreport2krona.py

- -r/--report MYFILE.KREPORT........Kraken report file
- -o/--output MYFILE.KRONA..........Output Krona text file

Optional:

- --no-intermediate-ranks...........[default]only output standard levels [D,P,C,O,F,G,S]
- --intermediate-ranks..............include non-standard levels

2. kreport2krona.py example usage

```
kraken2 --db KRAKEN2DB --threads THREADNUM --report MYSAMPLE.KREPORT \
   --paired SAMPLE_1.FASTA SAMPLE_2.FASTA > MYSAMPLE.KRAKEN2
```

```
python kreport2krona.py -r MYSAMPLE.KREPORT -o MYSAMPLE.krona
ktImportText MYSAMPLE.krona -o MYSAMPLE.krona.html
```
Krona information: see https://github.com/marbl/Krona.

3. kreport2krona.py example output

```
--no-intermediate-ranks
   6298       Unclassified
   8          k__Bacteria
   4          k__Bacteria     p_Proteobacteria
   6          k__Bacteria     p_Proteobacteria    c__Gammaproteobacteria
   ...
--intermediate-ranks
   6298       Unclassified
   79          x__root
   0          x__root     x__cellular_organisms
   8          x__root     x__cellular organisms   k__Bacteria
   4          x__root     x__cellular organisms   k__Bacteria     p__Proteobacteria
   6             x__root      x__cellular organisms    k__Bacteria       p__Proteobacteria
c__Gammaproteobacteria
   ....
```

---

kreport2mpa.py

This program takes a Kraken report file and prints out a mpa (MetaPhlAn) -style TEXT file

1. kreport2mpa.py usage/options

python kreport2mpa.py

- -r/--report MYFILE.KREPORT........Kraken report file
- -o/--output MYFILE.MPA.TXT........Output MPA-STYLE text file

Optional:

- --display-header..................display header line (#Classification, MYFILE.KREPORT) [default: no header]
- --no-intermediate-ranks...........[default] only output standard levels [D,P,C,O,F,G,S]
- --intermediate-ranks..............include non-standard levels
- --read-count......................[default] use read count for output
- --percentages.....................use percentage of total reads for output

2. kreport2mpa.py example usage

```
kraken2 --db KRAKEN2DB --threads THREADNUM --report MYSAMPLE.KREPORT
\
   --paired SAMPLE_1.FASTA SAMPLE_2.FASTA > MYSAMPLE.KRAKEN2
```

```
python kreport2mpa.py -r MYSAMPLE.KREPORT -o MYSAMPLE.MPA.TXT
```
3. kreport2mpa.py example output

The output will contain one tab character inbetween the classification and the read count.

```
--no-intermediate-ranks/--read-count
   #Classification                                  SAMPLE.KREPORT
   k__Bacteria                                      36569
   k__Bacteria|p__Proteobacteria                              21001
   k__Bacteria|p__Proteobacteria|c__Gammaproteobacteria     11648
   ...
--intermediate-ranks/--read-count
   #Classification                                  SAMPLE.KREPORT
   x__cellular_organisms                                 38462
   x__cellular_organisms|k__Bacteria                           36569
   x__cellular_organisms|k__Bacteria|p__Proteobacteria      21001
   ...
```

combine_mpa.py

This program combines multiple outputs from kreport2mpa.py. Files to be combined must have been generated using the same kreport2mpa.py options.

Important:

1. Input files to combine_mpa.py cannot be a mix of intermediate/no intermediate rank outputs.
2. Input files should be generated using the same Kraken database.
3. Input files cannot be a mix of read counts/percentage kreport2mpa.py outputs. **combine_mpa.py** will not test the input files prior to combining.

If no header is in a given sample file, the program will number the files "Sample #1", "Sample #2", etc.

1. combine_mpa.py usage/options

```
python combine_mpa.py
```

- -i/--input MYFILE1.MPA MYFILE2.MPA.......Multiple MPA-STYLE text files (separated by spaces)
- -o/--output MYFILE.COMBINED.MPA..........Output MPA-STYLE text file

2. combine_mpa.py example output

```
   #Classification                                  Sample #1    Sample #2
   k__Bacteria                                      36569        20034
   k__Bacteria|p__Proteobacteria                              21001        18023
   k__Bacteria|p__Proteobacteria|c__Gammaproteobacteria     11648        15000
```

filter_bracken_out.py

This program takes the output file of a Bracken report and filters the desired taxonomy IDs.

1. filter_bracken_out.py usage/options

python filter_bracken_out.py

- -i/--input MYFILE.BRACKEN..........Bracken output file
- -o/--output MYFILE.BRACKEN_NEW.....Bracken-style output file with filtered taxids
- --include TID TID2.................taxonomy IDs to include in output file [space-delimited]
- --exclude TID TID2.................taxonomy IDs to exclude in output file [space-delimited]

User should specify either taxonomy IDs with --include or --exclude. If both are specified, taxonomy IDs should not be in both lists and only taxonomies to include will be evaluated.
When specifying the --include flag, only lines for the included taxonomy IDs will be extracted to the filtered output file. The percentages in the filtered file will be re-calculated so the total percentage in the output file will sum to 100%.

When specifying the --exclude flag alone, all lines in the Bracken file will be preserved EXCEPT for the lines matching taxonomy IDs provided.

2. filter_bracken_out.py example usage

This program can be useful for isolating a subset of species to better understand the distribution of those particular species in the sample.

For example:

- python filter_bracken_out.py [options] --include 1764 1769 1773 1781 39689 will allow users to get the relative percentages of *Mycobacterium avium, marinum, tuberculosis, leprae, and gallinarum* in their samples.

In other cases, users may want to focus on the distribution of all species that are NOT the host species in a given sample. This program can then recalculate percentage distributions for species when excluding reads for the host.

For example, given this output:

```
name                    tax_id     tax_lvl     kraken....  added...   new.... fraction...
Homo sapiens               9606        S          ...        ....       999000  0.999000
Streptococcus pyogenes   1314        S          ...        ....       10      0.000001
Streptococcus agalactiae 1311        S          ...        ....       5       0.000000
Streptococcus pneumoniae 1313        S          ...        ....       3       0.000000
Bordetella pertussis     520         S          ...        ....       20      0.000002
```

```
    ...
```

Users may not be interested in the 999,000 reads that are host DNA, but would rather know the percentage of non-host reads for each of the non-host species. Using python filter_bracken_out.py [options] --exclude 9606 allows better resolution of the non-host species, allowing each of the fraction of reads to be recalculated out of 1,000 instead of 1,000,000 reads in the above example. The output would then be:

```
    name                  tax_id     tax_lvl    kraken....  added...   new.... fraction...
    Streptococcus pyogenes   1314       S          ...        ....       10      0.01000
    Streptococcus agalactiae 1311       S          ...        ....       5       0.05000
    Streptococcus pneumoniae 1313       S          ...        ....       3       0.03000
    Bordetella pertussis     520        S          ...        ....       200     0.20000
    ...
```

fix_unmapped.py

When building a Kraken database, an "unmapped.txt" file may be generated if a taxonomy for a given sequence is not found. This script can search through any accession2taxid files provided and the unmapped.txt file and generate a seqid2taxid.map file to be appended to the one already generated.

1. fix_unmapped.py usage/options

python fix_unmapped.py

- -i/--input unmapped.txt...........Any file containing accession IDs to map
- --accession2taxid REF_FILES.......Any tab-delimited file with 4 columns, accessions = column 1, taxonomy IDs = column 3
- -o/--output OUT_FILE.............Output tab-delimited file with 2 columns: accessions and taxids

Optional:

- -r/--remaining....................file containing any unmapped accession IDs after search [default: still_unmapped.txt]

2. fix_unmapped.py example usage

```
rm *.k2d
mv seqid2taxid.map seqid2taxid_1.map
python fix_unmapped -i unmapped.txt --accession2taxid taxonomy/*accession2taxid
-o seqid2taxid_temp.map
cat seqid2taxid_1.map seqid2taxid_temp.map
kraken2-build --build --db . --threads 4
```

make_ktaxonomy.py

For future KrakenTools scripts, this program generates a single text file that contains all of the taxonomy information required. This program is intended to generate a single text taxonomy file for any Kraken 1, Kraken 2, or KrakenUniq database.

Important: The output of this program does not replace any Kraken database file (do not replace your taxo.k2d or .db files).

1. make_ktaxonomy.py usage/options

python make_ktaxonomy.py

- --nodes taxonomy/nodes.dmp...........nodes.dmp file in Kraken DB taxonomy/ folder
- --names taxonomy/names.dmp...........names.dmp file in Kraken DB taxonomy/ folder
- --seqid2taxid seqid2taxid.map........seqid2taxid.map file generated by kraken-build/kraken2-build/krakenuniq-build when building the database. This is a 2-column tab-delimited file containing sequence IDs and taxonomy IDs.
- -o/--output OUT_FILE.................Output text file. More details below

The program will inform users if a taxonomy ID is listed in the seqid2taxid.map file but not in either the nodes.dmp or the names.dmp files.

2. make_ktaxonomy.py output file format

The output file is similar to the nodes.dmp/names.dmp file format, but not identical. Each of the following columns is separated by a tab-vertical line-tab (e.g. \t|\t).

1. taxonomy ID
2. parent taxonomy ID
3. rank type (R = root, D = domain/superkingdom, P = phylum, etc.)
4. level number (distance from root)
5. name

For ranks outside of the traditional taxonomy ranks (R, D, P, C, O, F, G, S), the rank type will be assigned based on the closest parent, with a number to specify distance from that parent. For example, the strains will be labeled with S1 while ranks inbetween Genus and Species will be labeled with G1, G2, etc.
Currently, names for each node are selected based on the first name listed in the names.dmp file or the name designated as scientific name. scientific names will be preferred over all others.
3. make_ktaxonomy.py required input

1. taxonomy/nodes.dmp
2. taxonomy/names.dmp
3. seqid2taxid.map

4. KrakenTools scripts requiring make_ktaxonomy.py output:

1. make_kreport.py

---

make_kreport.py

This program will generate a kraken-style report file from the kraken output file. Currently, this only generates reports for Kraken 1 or Kraken 2. This program does not currently work for KrakenUniq output files (to be completed in a future project).

This program requires that users first generate the taxonomy file created by make_ktaxonomy.py.

1. make_kreport.py usage/options

python make_kreport.py

- -i/-k/--input KRAKEN_FILE........default Kraken output file (5 tab-delimited columns, taxid in third column)
- -t/--taxonomy TAXONOMY_FILE......output from make_ktaxonomy.py
- -o/--output REPORT_FILE..........output Kraken report file (6 tab-delimited columns)

Optional

- --use-read-len...................make report using summed read lengths instead of read counts

2. make_kreport.py example

Given a Kraken 2 database KRAKENDB/ and sample file EXAMPLE_READS.fq, the following commands can be used to generate a Kraken report file with this script.
python make_ktaxonomy.py --nodes KRAKENDB/taxonomy/nodes.dmp --names KRAKENDB/taxonomy/names.dmp --seqid2taxid KRAKENDB/seqid2taxid.map -o KRAKENDB/mydb_taxonomy.txt
kraken2 --db KRAKENDB --threads 4 EXAMPLE_READS.fq > EXAMPLE.kraken2
python make_kreport.py -i EXAMPLE.kraken2 -t KRAKENDB/mydb_taxonomy.txt -o EXAMPLE.kreport2
3. make_kreport.py --use-read-len option

By default, the output Kraken report will list read counts for each taxonomy ID. However, if all read lengths are not the same, users can add the --use-read-len option, which will result in reporting summed read lengths for each taxon.
4. make_kreport.py output format

The output format for kreport.py is identical to the format generated by kraken-report or the --report switch with kraken2. The output file contains 6 tab-delimited columns as follows:

1. Percentage of total reads
2. Reads classified within sub-tree

3. Reads classified at this specific node (reads cannot be more specifically classified)
4. Level type (R = root, K = kingdom, P = phylum, etc)
5. Taxonomy ID
6. Name (preceeded by spaces to indicate distance from root)

12.5. Diversity Kraken Tools

For news and updates, refer to the github page: https://github.com/jenniferlu717/KrakenTools/

KrakenToolsDiversity is a suite of scripts to be used for post-analysis of Kraken/KrakenUniq/Kraken2/Bracken results. Please cite the relevant paper if using KrakenTools with any of the listed programs.

Scripts included in Diversity KrakenTools

1. alpha_diversity.py
2. beta_diversity.py

Running Scripts:

No installation required. All scripts are run on the command line as described.

Users can make scripts executable by running

```
chmod +x myscript.py
./myscript.py -h
```

---

alpha_diversity.py

This program calculates alpha diversity, from the Bracken abundance estimation file. User must specify Bracken output file, and type of alpha diversity to be calculated. Specific options are specified below.

1. alpha_diversity.py usage/options

```
python alpha_diversity.py
```

- -f, --filename MYFILE.BRACKEN..........Bracken output file
- -a, --alpha TYPE......................Single letter alpha diversity type (Sh, BP, Si, ISi, F)

2. alpha_diversity.py input file

Input Bracken file must be in standard Bracken output file format and must be run specifically for Abundance Estimation. Example format:

```
name              tax_id     tax_lvl    kraken....  added...   new.... fraction...
   Homo sapiens         9606       S          ...        ....      999000  0.999000
   Streptococcus pyogenes   1314        S          ...        ....       10      0.000001
   Streptococcus agalactiae 1311        S          ...        ....       5       0.000000
   Streptococcus pneumoniae 1313        S          ...        ....       3       0.000000
   Bordetella pertussis     520         S          ...        ....       20      0.000002
```

Link to Brack github for reference: Bracken

3. alpha_diversity.py alpha type input

By default, the program will calculate Shannon's alpha:

```
python alpha_diversity.py -f myfile.bracken
```

Users can specify which type of alpha diversity from this set:

- Sh......Shannon's alpha diversity
- BP.....Berger-Parker's alpha
- Si.....Simpson's diversity
- ISi.....Inverse Simpson's diversity
- F.......Fisher's index

To calculate berger-parker's alpha:

```
python alpha_diversity.py -f myfile.bracken -a BP
```

---

beta_diversity.py

This program calculates the beta diversity (Bray-Curtis dissimilarity) from kraken, krona and bracken files.

To calculate the pairwise dissimilarity score, you can call the script with two or more files as follows:

```
python beta_diversity.py -i file1.bracken file2.bracken [...] --type bracken
```

The script supports bracken files, kraken and kracken2 report files and krona files. Additionally, you can pass any tab-separated file and specify the columns with the taxonomical information and read counts (--cols).
To only compute the dissimilarity score on a certain taxonomical level (only supported for kraken and krona files), you can pass --level S (S, G, F or O for species, genus, family and order level).
For more information, please take a look at the help page.

```
python beta_diversity.py --help
```

Braken is two steps from fastq file to estimation abundance txt in our experiment

1.kraken2 --db /home/ngs/kraken2/DB /home/ngs/kraken2/sample.fastq --report /home/ngs/kraken2/results/sample.kreport --threads 24 --output /home/ngs/kraken2/results/sample.kraken output files:

sample.kraken

sample.kreport

2. python est_abundance.py -i /home/ngs/kraken2/results/sample.kreport -k ~/kraken2/Bracken/results/abun/database_kmer_distr_1500mers.txt -l S -t 10 -o /home/ngs/kraken2/results/sample_output_species_abundance.txt

output file:

sample_output_species_abundance.txt

for alpha diversity:

python alpha_diversity.py -f /home/ngs/kraken2/results/sample_output_species_abundance.txt -a **Sh**

for beta diversity:

python beta_diversity.py -i /home/ngs/kraken2/results/**sample1**_output_species_abundance.txt /home/ngs/kraken2/results/**sample2**_output_species_abundance.txt --type bracken --**level S** > /home/ngs/kraken2/results/**beta_diversity.txt**

13. Full-length 16S rRNA ONT data analysis using NanoCLUST

NanoCLUST is de novo clustering and consensus building for ONT 16S sequencing data**.**

The pipeline is built using Nextflow, a workflow tool to run tasks across multiple compute infrastructures in a very portable manner. It comes with docker containers making installation trivial and results highly reproducible.

* **Before starting Please check** "Computing requirements note for NanoCLUST"

to sure whether you have enough RAM to run NanoCLUST

Computing requirements note for NanoCLUST

Clustering step uses up to 32-36GB RAM when working with a real dataset analysis and default parameters (umap_set_size = 100000). Setting umap_set_size to 50000, will diminish memory consumption to 10-13GB RAM. When running the pipeline, kmer_freqs or mostly read_clustering **processes could be terminated with status 137** when not enough RAM.

Nextflow automatically uses all available resources in your machine. More cpu threads enable the pipeline to compute and classify the different clusters at the same time and hence reduces the overall execution time.

Using the -with-trace option, it is possible to get an execution trace file which includes computing times and memory consumption metrics for all pipeline processes

Quick Start

i. Install nextflow
ii. Install docker or conda
iii. Clone the NanoCLUST repository from below Website with below command in Linux and test the pipeline on a minimal dataset with a single command and docker/conda profiles.

https://github.com/genomicsITER/NanoCLUST

gh repo clone genomicsITER/NanoCLUST

*Download a BLAST database in the NanoCLUST dir for cluster sequence classification. For NCBI 16S rRNA database:

```
mkdir db db/taxdb
wget https://ftp.ncbi.nlm.nih.gov/blast/db/16S_ribosomal_RNA.tar.gz && tar -xzvf
16S_ribosomal_RNA.tar.gz -C db
wget https://ftp.ncbi.nlm.nih.gov/blast/db/taxdb.tar.gz && tar -xzvf taxdb.tar.gz
-C db/taxdb


#Using docker profile with container-based dependencies (recommended).
nextflow run main.nf -profile test,docker
```

iv. Start running your own analysis!

Run a single sample analysis inside NanoCLUST dir *using default parameters*:

```
nextflow run main.nf \
             -profile docker \
             --reads 'sample.fastq' \
             --db "db/16S_ribosomal_RNA" \
             --tax "db/taxdb/"
```

Run with the following command for our computer

```
nextflow run main.nf --profile docker --reads '/home/ngs/nanoclust/sample.fastq'
--db "db/16S_ribosomal_RNA" --"db/taxdb/" –max_memory '58.GB'
```

After running, please check the below outputs

hdbscan.output.png

rel_abundance_.....png

If it is necessary, “UMAP Clustering and polishing parameters” in nexflow.config file can be set up for umap_set size, cluster_sel_epsilon, min_cluster_size, polishing_reads.

14. Metabolomics analysis of full-length 16S rRNA ONT data using Paprica

Paprica (PAthway PRediction by phylogenetIC placement)

Paprica conducts metabolic inference on (preferably, but not exclusively, NGS) 16S rRNA gene libraries. Instead of using an OTU based approach however, it uses a phylogenetic placement approach. This provides a more intuitive connection between its “hidden state prediction” and library analysis components, and allows resolution at the strain and species level for some spots on the prokaryotic phylogenetic tree.

Paprica uses pathways shared between the members of all clades on a reference tree to determine what pathways are likely to be associated with a phylogenetically placed read.

Paprica was designed to use a significant amount of resources up front to construct a database and draft metabolic models for all available completed genomes. You can avoid this by using the provided database.

To run paprica you will need:

```
## Install some packages
        sudo apt-get install build-essential
        sudo apt-get install git
        sudo apt-get install zip

## Install python dependencies, including external python tools
        pip3 install numpy
        pip3 install biopython
        pip3 install joblib
        pip3 install pandas
        pip3 install seqmagick
        pip3 install termcolor

## Install RAxML
        #git clone https://github.com/stamatak/standard-RAxML.git
        #cd standard-RAxML
        #sudo make -f Makefile.AVX2.PTHREADS.gcc
        #rm -f *.o

## Install RAxML-ng
        wget           https://github.com/amkozlov/raxml-ng/releases/download/0.9.0/raxml-
        ng_v0.9.0_linux_x86_64.zip
        unzip raxml-ng_v0.9.0_linux_x86_64.zip

## Install infernal
        cd ~
        wget http://eddylab.org/infernal/infernal-1.1.2-linux-intel-gcc.tar.gz
        tar -xzvf infernal-1.1.2-linux-intel-gcc.tar.gz
        mv infernal-1.1.2-linux-intel-gcc infernal

## Install gappa
        git clone --recursive https://github.com/lczech/gappa.git
        cd gappa;make
```

	cd ~
## Install epa-ng	
	## Double check that you have all dependencies as described here: https://github.com/Pbdas/epa-ng#installation.
	## If the compiler yells at you about not having zlib, you will need to have zlib1g-dev installed, not just zlib1g!
	sudo apt-get install autotools-dev libtool flex bison cmake automake autoconf
	git clone https://github.com/Pbdas/epa-ng.git
	cd epa-ng;make
	cd ~
## Modify PATH in .bashrc	
	TEMPNAME=`whoami`
	echo "## added by paprica installer" >> .bashrc
	echo "PATH=/home/${TEMPNAME}/pplacer:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/.local/bin:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/infernal/binaries:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/infernal/easel:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/raxml-ng:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/paprica:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/epa-ng/bin:"'$PATH' >> .bashrc
	echo "PATH=/home/${TEMPNAME}/gappa/bin:"'$PATH' >> .bashrc
	echo "export PATH" >> .bashrc
	source .bashrc
## Download paprica - redundant cause that's probably how you got this script	
	git clone https://github.com/bowmanjeffs/paprica.git
	cd paprica
	chmod a+x *py
	chmod a+x *sh

Paprica can be run on OSX (see tutorial linked above), or preferably, on Linux or Windows using the Windows Subsystem for Linux (see the linux_install.sh script as a guide).

If you use Docker or Singularity you can download a Docker image with docker pull jsbowman/paprica:latest

A pipeline to conduct a metabolic inference from 16S rRNA gene sequence libraries. Check out the Wiki and tutorials (listed above) to get started. Once you've installed the depenencies the following commands will get you started:

```
git clone https://github.com/bowmanjeffs/paprica.git
cd paprica
chmod a+x *py
chmod a+x *sh
./paprica-run.sh test bacteria
```

Alternatively, if you're using the Docker image you can skip installing the dependencies and just do:

```
docker pull jsbowman/paprica:latest
docker run -it jsbowman/paprica
cd /paprica
./paprica-run.sh test bacteria
```

To run paprica the following command can be used in our experiment

- We use fasta file as input for the paprica metabolomic analysis

```
cd /paprica
./paprica-run.sh sample bacteria
```

15. References