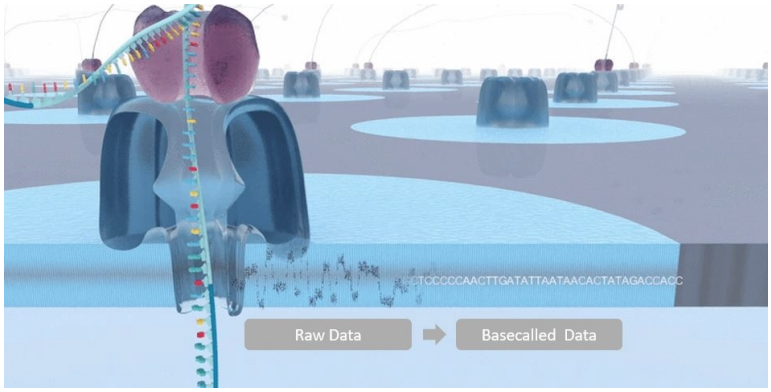
Basecalling overview

Introduction to basecalling

Basecalling is the process of converting the electrical signals generated by a DNA or RNA strand passing through the nanopore into the corresponding base sequence of the strand. The general data flow in a nanopore sequencing experiment is shown below.



Raw data - a direct measurement of the changes in ionic current as a DNA/RNA strand passes through the pore, which are recorded by the MinKNOW software. MinKNOW also processes the signal into "reads", each read corresponding to a single strand of DNA/RNA. These reads are written out as .fast5 files.
These .fast5 files use the HDF5 format to store data (http://www.hdfgroup.org/HDF5/); libraries exist to read and write these in many popular computer languages (e.g. R, Python, Perl, C, C++, Java).

Basecalling - the raw signal is further processed by the basecalling algorithm to generate the base sequence of the read.

Basecalling is made up of a series of steps that are executed one by one. Ionic current measurements from the sequencing device are collected by the MinKNOW software and processed into a read. The reads are transformed into basecalls using mathematical models. The results of these analyses are written into FASTQ files, with a default of 4000 reads per FASTQ file. Alternatively, the user can select a .fast5 file output for additional information about the raw signal (similarly, the default is 4000 reads per .fast5 file).

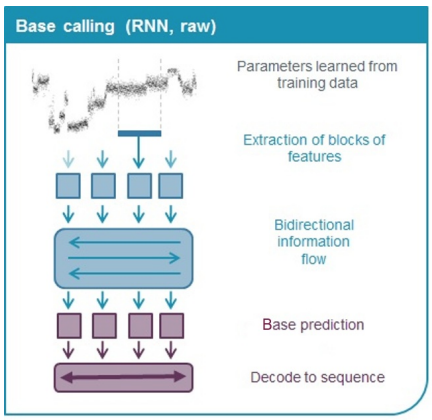
Oxford Nanopore Technologies provides several platforms to allow users to carry out basecalling in real time, as well as executables for users' local infrastructure. Basecalling can be carried out live during the experiment, as post-processing after an experiment has finished, or a combination of these.

Introduction to neural networks

The production version of Oxford Nanopore basecallers convert raw signal data to basecalls using a Recurrent Neural Network (RNN).

A neural network models processes that occur inside the human brain. The network contains nodes arranged in layers, which carry out computations. Neural networks receive and process data, but crucially, they have the ability to 'learn' and improve their predictions for the output over time. Neural networks have been successfully used for diverse applications like pattern recognition (such as handwritten characters, speech recognition), or predicting trends over time.

A recurrent neural network is a class of neural networks in which the output is dependent on past computations. An RNN keeps an internal memory of previously-seen data, so each new computation can use information from several preceding computations.



Some research basecallers at Oxford Nanopore Technologies use different types of neural networks. For example, the Bonito basecaller contains a Convolutional Neural Network (CNN). CNNs are typically used for image and video processing, since the connections between nodes mimic the organisation of the brain's visual cortex.

Analysis used by Oxford Nanopore uses a neural network that has been trained on a range of example DNA sequences (described in more detail in the **Basecaller training** section). Using this training, the network is able to call the sequence of new input data. Note that all of Oxford Nanopore's basecalling is carried out on raw signal data, without converting to an intermediary state.

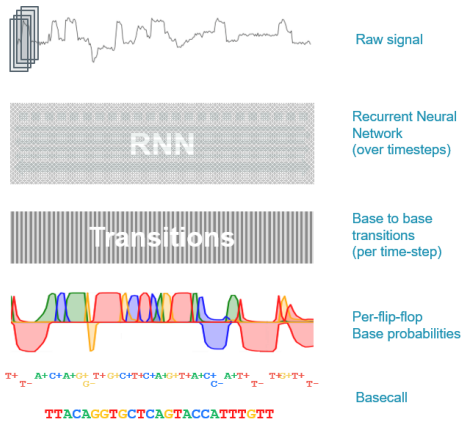
Oxford Nanopore basecallers

Basecaller	Algorithm	Availability
MinKNOW basecaller	Production basecaller on the device software; uses the "Flip-flop" algorithm. This is identical to the algorithm used by stand-alone Guppy, but may be a version behind	Available as a free download (further details in the MinKNOW protocol). The basecalling option is selected when the sequencing experiment is started, and the experiment progress is monitored via the MinKNOW GUI
Guppy	Production basecaller available as an executable; uses the "Flip-flop" algorithm. This can be run on CPUs, but is optimised for real-time basecalling on GPUs. Guppy is the default basecaller on all Oxford Nanopore sequencing devices	Available as a free download. The executable version of the software can be run on the host computer via the command line. GPU compatibility is currently limited to NVIDIA Tesla V100, NVIDIA GTX1080Ti, and NVIDIA Jetson TX2
Research algorithms	Varied	Research algorithms are available through GitHub. The releases are varied, and include features that will be included in future versions of the production basecaller

Basecalling algorithms

The "Flip-flop" basecalling algorithm

The Flip-flop basecalling algorithm uses the raw signal to write out single-base transition probabilities, along with confidence levels for each called base, and the alternative possibilities. Flip-flop basecalling provides higher accuracy basecalls and better homopolymer resolution than previous algorithms from Oxford Nanopore Technologies.



The neural network model in the Flip-flop basecaller performs *label-free basecalling*: instead of labelling raw data with a short sequence of bases like previous "transducer" basecalling algorithms, the model produces likelihoods of transitions between *consecutive* bases. The network considers two states for each base: "flip", which is a transition from one base to the next, and "flop", which is staying on the same base for two time-steps. The model uses Viterbi decoding to assign the likelihood of the "flip" and "flop" state for each base transition. This way, the network can distinguish between long runs of the same base (homopolymers) and multiple “stays” on the same base.

Fast vs High Accuracy models

The Guppy basecaller, which is also integrated in MinKNOW, offers two different Flip-flop models: a High accuracy (HAC) model and a Fast model. The HAC model provides a higher consensus/raw read accuracy than the Fast model. It contains a more computationally-intense Flip-flop architecture that can deliver higher accuracy using the same data produced by nanopore sequencing. It is currently 5-8 times slower than the Fast model.

The Fast Flip-flop model includes a simplified version of the Flip-flop algorithm and delivers the same level of accuracy and basecalling speed to those obtained with the older *transducer* basecalling algorithm. Both models have been trained on the same datasets.

A comparison of the speed and accuracy of the two models is provided in the table and graphs below.
Please note that these numbers represent the theoretical best speeds achievable with the basecallers, and a real biological sample may be basecalled more slowly.

		Data generation in Gbases per hour				
	Output	Flongle	MinION	GridION	P24	P48
Per flow cell	2 Gb Flongle 30 Gb MinION 150 Gb PromethION	0.08	0.42	0.42	2.1	2.1
Per full device		-	-	2.1	50	100

		Basecalling speed in Gbases per hour				
	Raw Read Accuracy	MinION high-spec laptop	MinIT and MinION Mk 1c	GridION Mk 1	P24	P48
R9.4.1 Fast	92.1%	0.13	0.24	40	87	116
R9.4.1 HAC	95%	0.02	0.06	5	18	27
R10.3 Fast	-	-	-	-	-	-
R10.3 HAC	96%	-	0.04	6	11	19

Calling modified bases

Certain base modifications, specifically 6mA dam/5mC dcm and CpG, can be called using the MinKNOW or stand-alone Guppy software. This requires the use of a designated basecalling model which uses the Flip-Flop algorithm and is trained to identify base modifications.

Modified base output consists of two parts:

1. A normal FASTQ record, available either as part of FASTQ files or as FASTQ entries embedded in .fast5 files.
2. A supplementary table provided as part of .fast5 output, which contains estimated probabilities that a particular base in the FASTQ entry is a modified one.

For more information about this model, please refer to the Guppy protocol.

Basecaller training

When developing neural networks, Oxford Nanopore Technologies trains basecall models using data from sequencing experiments. Model development is broken down into two broad categories: training (creating a model) and validation (showing that it works). Currently, two types of models are trained, each requiring their own unique type of training sets: basecall models (which work with basecallers such as Guppy or Bonito) and consensus models (which work with the Medaka sequence polishing tool). Each of these models are validated in different ways and require different types of validation data. The rest of this section will focus on training basecall models.

To train a DNA basecall model, a dataset of reads is selected. Typically, the dataset contains .fast5 files from human, yeast, *E. coli*, *C. elegans*, and ZymoBIOMICS Microbial Community Standard sequencing experiments. The data includes both PCR-amplified reads and native reads that contain base modifications. A portion of the reads is reserved for validating the model, and is not included in the training dataset. The remaining reads are basecalled with the current best basecall model and aligned against the reference genomes. Reads are then filtered by alignment accuracy (minimum 85%) and strand length (minimum 1000 bases); only those reads that pass the minimum criteria are included for model training.

The new basecall model is then trained using the Taiyaki software. To do this, Taiyaki is given the filtered .fast5 dataset and the FASTA file that contains the 'ground truth' reference for each read. It is also supplied with some parameters for how the model should be trained.

Once trained, the quality of the model is validated using the sub-set of reads that were not included in the training dataset. Validation assesses the following parameters:

- Alignment accuracy
- % of strands that align to the reference
- Alignment length
- Strand length
- Identifying strand edges and barcodes
- Homopolymer calling
- Recognising methylation motifs

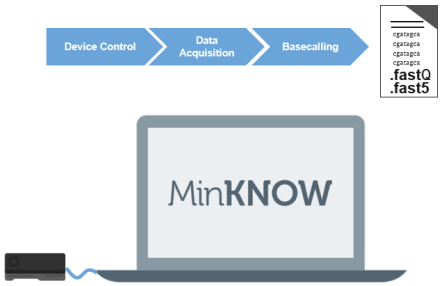
If the validation meets the minimum criteria and the new model is an improvement on the currently-released models, it is then included in one of Oxford Nanopore's basecallers.

Live basecalling

Introduction to basecalling in MinKNOW

For MinION Mk 1B and Flongle, the MinKNOW software presents an option to basecall reads on the local computer. The basecalling is carried out live, as the read files are generated during a sequencing experiment.

Basecalling results are displayed in real-time in the MinKNOW GUI, and data is written out in the FASTQ or .fast5 file format.

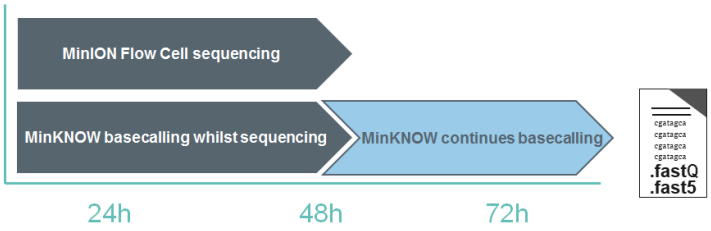


The GridION and PromethION devices have the capacity to perform live basecalling, while keeping up with five flow cells (for GridION) and up to 48 flow cells (for PromethION 48) running simultaneously. Basecalling is carried out directly on the device and uses the MinKNOW software. The basecalled reads are written to the device; users can then export the reads out for further analysis via an Ethernet cable.

MinKNOW basecalling: keep-up vs catch-up

Basecalling with the Fast Flip-flop model can keep up with the speed of data acquisition on all nanopore platforms. With the HAC model, basecalling on a minimum-spec laptop may not keep up with the amount of data generated. Basecalling will therefore continue after the sequencing experiment has run to completion; any reads that have not been basecalled during the experiment will be queued and processed afterwards. This is known as “Catch-up mode”.

The user therefore has two options: either to allow MinKNOW to continue in catch-up mode, or to stop the analysis and basecall the remaining reads at a later time, e.g. using stand-alone Guppy.



Basecalling on the MinIT device

The MinIT is a solution for running a MinION Mk 1B without the requirement for any additional processing power or software. It is pre-configured with the MinKNOW software (containing the Guppy basecaller), and contains GPUs that enable it to keep up with a high-throughput sequencing experiment.

On-demand basecalling using the Guppy software

Guppy basecalling software

Guppy is a data processing toolkit that contains Oxford Nanopore Technologies' basecalling algorithms, and several bioinformatic post-processing features. It is run from the command line in Windows, and on multiple Linux platforms. A selection of configuration files allow basecalling of DNA and RNA libraries, made with Oxford Nanopore Technologies current sequencing kits, in a varied range of flow cells.

The Guppy toolkit contains:

The basecaller

The Guppy basecaller implements a neural network algorithm that allows raw signal data to be interpreted into the five canonical bases of DNA or RNA; namely adenine, guanine, cytosine, thymine and uracil.

Calibration Strand Detection: The basecaller is also capable of detecting calibration strands by aligning calibration sequences. Reads are aligned against a reference using the basecalled data from the DNA CS reagent that is added to the flow cell. Calibration strands serve as a quality control for the pore and

experimental processing. If a read is identified as a calibration strand, no barcoding or alignment steps are performed.

Adapter Strand Trimming: This is the processing and removal of the sequencing adapter (e.g. AMX, BAM, AMII, etc.) sequence provided by the Guppy basecaller.

It will remove the adapter sequences as follows:
- For DNA adapter sequences it will *only* remove the adapter region, up to a characteristic signal in the adapter that is recognised by the basecaller.
- For (m)RNA and, it will attempt to remove all data up to the end of the polyA tail and start of the gene coding region.

Command line, input, and output for the basecaller:
- Command line: `guppy_basecaller`
- Input: .fast5 files containing raw data
- Output: .fast5 files containing basecalled information and/or FASTQ files containing basecalled sequences and a summary file.

The 1D^2 basecaller

The 1D^2 basecaller is only compatible with libraries made using the 1D^2 Sequencing Kit (SQK-LSK309). The basecaller interprets the template and complement strands, one after another. The first strand is sequenced; this is followed by the capture of the double-stranded DNA molecule, which can subsequently be interpreted to produce the reverse complement. This is deconvoluted by the software to produce a higher quality score of the consensus basecalled sequence.

Barcoding/demultiplexing

If a barcoding kit has been used to sequence several different samples on one flow cell, the barcoding option in Guppy can be used to split the reads by barcode. The beginning and the end of each strand are aligned against the barcodes currently provided by Oxford Nanopore Technologies. Demultiplexing occurs directly from the basecalled results.

Alignment

The user can provide a reference file in FASTA or minimap2 index format. If so, the reads are aligned against this reference via the integrated minimap2 aligner using the standard Oxford Nanopore Technologies preset parameters.

Calling modified bases in Guppy

Guppy includes a configuration file that enables the identification of:

- 6mA dam methylation (trained on *E. coli* data)
- 5mC dcm methylation (trained on *E. coli* data)
- 5mC CpG methylation (trained on human data)

This modification detection is done directly from the raw signal of native DNA without the need for any new preparation steps and with almost no additional analysis time.

Modified base output consists of two parts:

1. A normal FASTQ record, the same as from normal basecalling, available either as part of FASTQ files or as FASTQ entries embedded in .fast5 files
2. A supplementary table provided as part of .fast5 output, which contains estimated probabilities that a particular base in the FASTQ entry is a modified one

More information about running the modified base caller and interpreting the results can be found in the Guppy protocol under "Guppy features, settings and analysis".

GPU vs CPU basecalling

Oxford Nanopore Technologies supports GPU basecalling on devices that have built-in compute (MinIT, GridION, PromethION). GPU basecalling is an order of magnitude faster than CPU basecalling, and can keep up with a high-performing sequencing experiment.

The devices above come with MinKNOW pre-installed. Real-time basecalling on the instrument is performed on GPUs, and there is also an option of using GPUs when running standalone Guppy on the instrument.

If sequencing on a MinION Mk 1B and using a laptop/own computer for data analysis, the basecalling will be performed on a CPU and may not keep up with the speed of data acquisition (if using MinKNOW). In this case, basecalling will continue after the sequencing run has completed until all the reads have been processed.

Guppy availability

The Guppy basecalling software is available free of charge to the Nanopore Community. More details on installing and running the software are found in the Guppy protocol.

The Guppy source code is available through the Developer channel to users who have signed the Developer terms and conditions.

Basecall accuracy

Introduction to nanopore sequencing accuracy

Oxford Nanopore's sequencing accuracy is shown as one of several metrics:

- per-base accuracy, denoted by the Phred Q-score (see explanation below)
- single-molecule accuracy, usually denoted by %, e.g. 95% accuracy means that 95 bases out of 100 in a read were called correctly. Note that Oxford Nanopore Technologies represents the average per-read accuracy as the **modal** accuracy from a sequencing run
- consensus accuracy, denoted by the Phred Q-score, e.g. Q40 means that 99.99% of all bases in a consensus sequence were called correctly

Sequencing accuracy is determined by multiple factors such as how a sample is prepared, pore chemistry, and basecalling algorithms. Improvements in these areas over the last several years have led to a steady increase in both single-molecule and consensus accuracy, discussed in more detail below.

Phred quality scores

The Phred quality score defines the quality of each base in the sequence, with values from from 0 to 93. The score is calculated as:

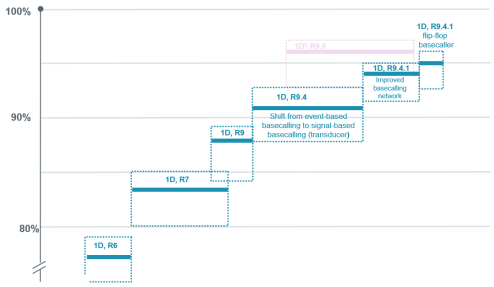
Quality score = -10 x log(Pe)

where Pe is the estimated error probability for each base. For example, an error of 1 in 100 will give a q-score of 20. The q-scores are then encoded in the Sanger format using ASCII, with values of 33 to 126. The quality is then shown as a single character per base.

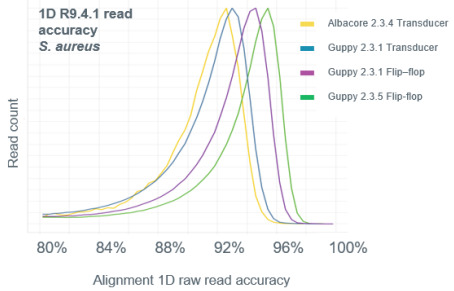
Per-base quality scores are stored together with the base sequence in FASTQ files output by the basecalling algorithms.

Single-molecule accuracy

Single-molecule accuracy has been steadily increasing over the years, following multiple improvements in basecalling algorithms and pore chemistry. Below is a timeline of the increase in modal accuracy (thick blue lines), where the boxes show the accuracy distribution for 90% of reads in a sequencing run. The results are typically generated on a range of samples including *E. coli*, *S. aureus*, and human.



In particular, the switch from a "transducer" basecalling algorithm to the Flip-flop algorithm, which produces a per-base output and can better resolve homopolymeric regions, has improved the modal accuracy from 90% to ~95%.

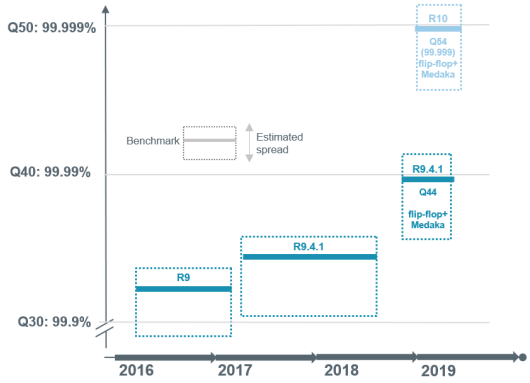


Consensus accuracy

Consensus accuracy represents the accuracy of a consensus sequence made from a pile-up of reads. It is denoted by Q-scores, which have a logarithmic relationship to the basecalling error probability:

Q-score	Number of errors
Q40	1 error in 10,000 bases
Q50	1 error in 100,000 bases
Q60	1 error in 1,000,000 bases

As with single-molecule accuracy, modal consensus accuracy has been increasing over the years with improvements in pore chemistry, basecallers and polishing tools.



The current best consensus accuracy is produced using an Oxford Nanopore Research base-space polishing tool called *medaka*, which is optimised for the latest 'R10' pore chemistry (https://github.com/nanoporetech/medaka).

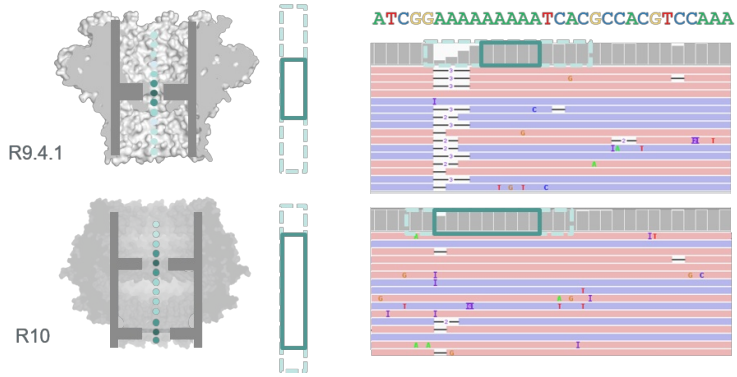
Q44 has been achieved with the R9.4.1 pore, the Flip-flop basecaller and *medaka* polishing, while Q54 can be reached using the R10 pore.

Pore chemistry and its impact on accuracy

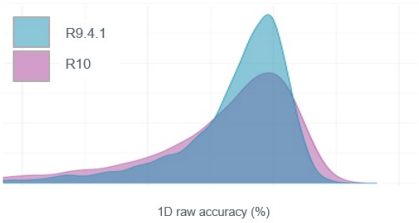
The structure of the nanopore itself determines the raw signal that is produced when the DNA or RNA passes through the pore, and therefore how accurate the basecalling will be.

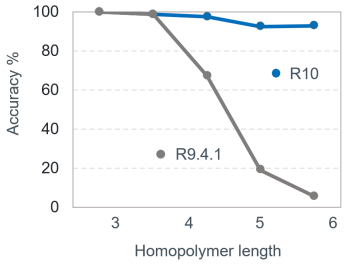
The nanopores used by Oxford Nanopore Technologies have one or more "readers", i.e. constrictions within the pore barrel that interact with the polynucleotides and produce the changes in current levels.

The R9.4.1 version of the pore has a single reader in the middle of the barrel. The new R10 version of the pore has two readers, which means that more bases within the DNA/RNA strand can interact with the pore, and longer homopolymeric regions are better "seen" by the pore (see diagram below).

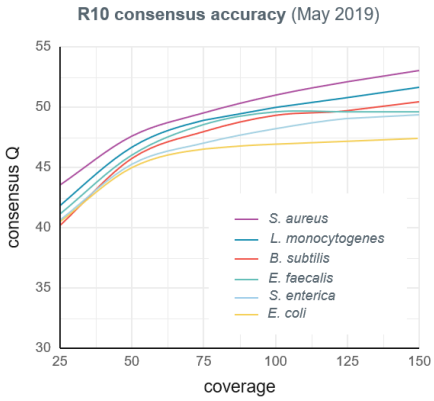


While the R9.4.1 and R10 modal single-molecule accuracies are similar (top graph below), R10 retains higher single-molecule accuracy when reading homopolymers (shown in the bottom graph below), while the accuracy with R9.4.1 drops off.





Consensus accuracy is also higher for R10 than R9.4.1. A recent sequencing analysis of the ZymoBIOMICS Microbial Community Standards with the R10 pore has given a consensus accuracy above Q45 for various bacterial species in the sample at 50X coverage, and Q54 for *S. aureus* at 150X coverage.



Barcoding options

Barcode design

The Oxford Nanopore Technologies barcoding kits can place barcodes at the beginning and end of the strands for multiplexing several different samples in one sequencing experiment. The barcodes will reside in a kit-specific context sequence, and different kits will have different lengths of sequence before and after the barcode. However, the sequences of the barcodes themselves are identical, regardless of kit.

The regions of a barcode

A complete barcode arrangement comprises three sections:

1. The **upstream flanking region**, which comes between the barcode and the sequencing adapter.
2. The **barcode sequence**.
3. The **downstream flanking region**, which comes between the barcode and the sample sequence.

A complete dual-barcode arrangement comprises five sections:

1. The **upstream flanking region**, which comes between the outer barcode and the sequencing adapter.
2. The **outer barcode sequence**.
3. The **mid flanking region**, which comes between the outer barcode and the inner barcode.
4. The **inner barcode sequence**.
5. The **downstream flanking region**, which comes between the inner barcode and the sample sequence.

The barcode sequences remain constant across almost all of Oxford Nanopore Technologies' kits. For example, the flanking regions for barcode 10 in the Rapid Barcoding Kit (SQK-RBK004) are different from the flanking regions for barcode 10 in the native barcoding expansion kit (EXP-NBD114), but the barcode sequence itself is the same.

While native kits use the same barcode sequences as other kits, barcodes 1-12 in the native kit are the reverse complement of the standard barcodes 1-12.

There is one other exception to this: barcode 12a in the Rapid Barcoding Kit SQK-RBK004 has a different barcode sequence to barcode 12 in other kits. For this reason, the oligonucleotide of this sequence is referred to as "barcode 12a".

Barcode and barcode flanking sequences can be found in the Chemistry technical document in the Nanopore Community.

Barcode demultiplexing options

After a barcoded sequencing run has completed, the reads can be split into folders by barcode, using one of Oxford Nanopore's demultiplexing tools:

- The FASTQ Barcoding workflow in EPI2ME
- Barcode demultiplexing in Guppy
- Real-time barcode demultiplexing in MinKNOW

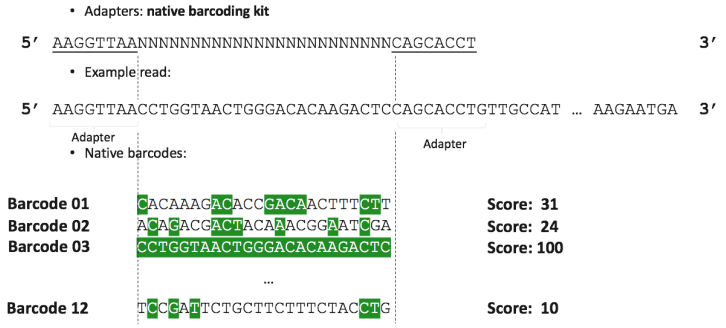
A brief description of each option is provided below.

FASTQ Barcoding workflow in EPI2ME

The EPI2ME platform provides a workflow for analysing sequencing data with barcodes. When this is used, the returned FASTQ files will contain information about the barcodes found. The reads will be returned to a data location entered in the Desktop Agent.

As the basecalled reads are uploaded to the cloud, the first step in the workflow is to check that the read files are of a supported filetype (currently FASTQ), and that the file headers are in the correct format.

The demultiplexing algorithm consists of three main steps:
1. Identify the sequencing kit that was used for a given read
2. Identify the correct barcode
3. Filter the results



1. Identify the sequencing kit that was used for a given read

The algorithm has one ore more 'templates' for each sequencing kit. These templates consist of the fixed regions that flank the barcodes, and a place holder for where the barcode is located. The fixed regions differ between kits and therefore allow to determine the kit that was used.

For example:

Template for the Rapid Barcoding Kit (SQK-RBK004):
GCTTGGGTGTTTAACCNNNNNNNNNNNNNNNNNNNNNNNNNNGTTTTCGCATTTATCGTGAAACGCTTTCGCGTTTTTCGTGCGCCGCTTCA

Template for the Native Barcoding Expansion (EXP-NBD104):
ATTGCTAAGGTTAANNNNNNNNNNNNNNNNNNNNNNNNCAGCACC

The algorithm aligns the templates from all supported kits to a given read using semi-global alignment and computes a length-normalised score. The template with the highest score is most likely to originate from the kit that was used to sequence the given read.

Example:
- The template for SQK-RBK004 has a score of 46
- The template for EXP-NBD104 gets a score of 94

Therefore, it is assumed that the library preparation used EXP-NBD104.
EPI2ME also allows the user specify the kit the was used for sequencing. In this case, step 1 can be skipped.

2. Identify the correct barcode

All the barcodes are aligned separately to the region of the read where the barcode is supposed to be located (the stretch of Ns in the template), to calculate the alignment scores again. Including ~10 bp up- and downstream of the barcode to the alignment has been shown to improve results. The barcode that gives the highest score is mostly likely to be the correct barcode.

For example:

Barcode01: score 31
Barcode02: score 24

Barcode03: score 100
....
Barcode12: score 10

In this case, barcode03 is the correct barcode. The algorithm uses minimum score cutoff of 58: if there are no alignments above this score, the read is reported as unclassified.

3. Filtering the results

It has been shown that reads with different barcodes attached to their 5' and 3' ends result in incorrect classification. To reduce the effect of those reads, the algorithm searches for barcodes independently at the 5' and 3'. If it finds barcodes at both ends that are above the score cutoff, it checks whether the calls are the same. If the two barcodes are not the same, the read is reported as unclassified.

For more information about the barcoding workflow, please see the EPI2ME Technical Document, "Workflows" section.

Barcode demultiplexing in Guppy

The barcoding algorithm in Guppy uses a modified Needleman-Wunsch method. Each barcode is aligned to a section of the basecall, with a score assigned to each base in the sequence depending on whether the base was a match, mis-match or a gap. The combined scores for each barcode alignment are compared, and the barcode with the highest score is chosen as long as the score is above the defined threshold. The barcode sequences can be trimmed from the reads, as a command-line option.

For more details, please see the Guppy protocol, Appendix A.

Barcode demultiplexing in MinKNOW

MinKNOW uses Guppy for both basecalling and barcode demultiplexing, so the barcoding metholodology is identical to stand-alone Guppy. However, MinKNOW performs barcode demultiplexing in real-time, as the sequencing run progresses.

FASTQ files

FASTQ output

FASTQ files can be generated in MinKNOW and Guppy. The default is to write out 4000 reads per FASTQ file, although this number is configurable.

Read .fast5 files from the instrument

Read .fast5 files

A .fast5 file is a type of HDF5 file, which is designed to contain all information needed for analysing nanopore sequencing data and tracking it back to its source. Read .fast5 files contain raw sequencing data for each read.

Default read file location

Windows

C:\data\

Mac OS X

/Library/MinKNOW/

Linux

/var/lib/MinKNOW/

File directory and name

All files output directly from an experiment are located in the same directory. This directory has the structure:

```
{output_dir}/{experiment_id}/{sample_id}/{start_time}_{device_ID}_{flow_cell_id}_{short_protocol_run_id}/
```

output_dir is the configured output directory
experiment_id is the user-entered identifier for a group of runs
sample_id is the user-entered value for a specific sample or run. Multiple **sample_ids** may exist beneath an individual **experiment_id**
start_time is the time at which the protocol started, in YYYYMMDD_HHMM format
device_id is the serial ID of the MinION Mk 1B or device position for GridION/PromethION
flow_cell_id is the flow cell ID (eg: FAH12345), either programmed on the ASIC or entered by the user
short_protocol_run_id is a unique identifier of 7 characters from the protocol ID

Examples of the above naming convention:

MinION Mk 1B

C:\data\MyExperiment\Sample1\20181011_1759_MN12345_FAH12345_0ffe109

GridION

/data/MyExperiment/Sample1/20181011_1759_GA2000_FAH12345_0ffe109

Individual read files are split into .fast5 "pass" and "fail" folders, as well as FASTQ "pass" and "fail" folders within this directory:

{ext}_{status}/{flow cell id}_{run id}_{batch_number}.{ext}

Examples of the above file naming:

fast5_pass/FAK12345_9bf81741599b097c42ba9a6ff6e4d00f02b6534b_0.fast5
fast5_fail/FAK12345_9bf81741599b097c42ba9a6ff6e4d00f02b6534b_0.fast5
fastq_pass/FAK12345_9bf81741599b097c42ba9a6ff6e4d00f02b6534b_0.fastq
fastq_fail/FAK12345_9bf81741599b097c42ba9a6ff6e4d00f02b6534b_0.fastq

File content

This information is provided for users who want to be able to look directly at the read files being produced by MinKNOW, which may be of particular interest to those wishing to carry out custom analysis using their own tools. The following represents the current standard for single-read .fast5 files.

```
/{attributes: file_version}
|-UniqueGlobalKey/
| |-tracking_id/{attributes: asic_id, asic_id_eeprom, asic_temp,
device_id, exp_script_hash, exp_script_name, exp_script_purpose, exp_start_time,
flow_cell_id, heatsink_temp, hostname, protocol_run_id, protocols_version_name,
run_id, version, version_name}
| |-channel_id/{attributes: channel_number, digitisation, offset, range,
sampling_rate}
| |-context_tags/{attributes: set when the experiment is configured}
|-Raw/
| |-Reads/
| |-Read_42/{attributes: start_time, duration, read_number, start_mux,
read_id, median_before}
| |-Signal{samples}
|
```

The root-level **file_version** attribute indicates how the layout of the **UniqueGlobalKey** and **Raw** fields shown above are represented in the file.

The **UniqueGlobalKey** group contains, as attributes, all of the information needed to identify the run and read.

The **Raw** group contains 16-bit DAQ values produced by the instrument. Basecalling algorithms directly operate on this raw signal.

The **Read_#** group contains the read data. The number following the underscore is the read_number. This number is only unique within a specific channel.
The attributes on this group have the following meanings:

- **read_number** - The same as the number after the underscore. This number is unique within the channel, but only really meaningful in respect to the bulk .fast5 file read table
- **read_id** - A unique id for the read. This is a **UUID**, and should be unique for any read from any device
- **start_time** - The start time of the read. This is in samples, so the time has to be divided by the sampling rate to get the start time in seconds
- **duration** - The duration of the read. This is also in samples
- **start_mux** - The MUX setting for the channel when the read began. Due to timing issues this can sometimes reflect what the MUX was just *before* the

read began; this will only matter for reads that start immediately after a MUX change

- **median_before** - The measure of the median current level for the data preceding the read. In most cases this can be used as an estimate of the open pore level

The file structure described here allows the files to be produced incrementally by MinKNOW, which means that analyses can be done whilst the experiment is still in progress.

Intermediate folder

The files in the intermediate folder store unprocessed raw signal data. Once raw signal processing is complete, .fast5 files are generated and stored in the tmp folder, where local basecalling can proceed. The size of unprocessed data is larger than that of the final .fast5/FASTQs, but these files are removed as processing proceeds or at the end of the run.

If the system running the MinION Mk 1B encounters an issue, such as running out of space, the unprocessed data will not be cleared and will remain in the intermediate folder. Due to the real-time streaming nature of the system, this data cannot be processed after the run is stopped.

Basecalled .fast5 files

Single-read .fast5 file structure for local basecalling

```
/{attributes: file_version}
|-UniqueGlobalKey/
| |-tracking_id/{attributes: standard tracking fields}
| |-channel_id/{attributes: channel_number, digitisation, offset, range, sampling_rate}
| |-context_tags/{attributes: set when the experiment is configured}
|-Raw/
| |-Reads/
| |-Read_42/{attributes: start_time, duration, read_number, start_mux, read_id}
| |-Signal{samples}
|-Analyses/
| |-Segmentation_000{attributes: name, version, time stamp}
| | |-Summary/{attributes: return_status}
| | | |-segmentation/{attributes: has_template, has_complement, first_sample_template, duration_template, first_sample_complement, duration_complement}
| |-Basecall_1D_000/{attributes: name, version, time stamp}
| | |-BaseCalled_template/
| | | |-Fastq{string}
| | |-Summary/{attributes: return_status}
| | | |-basecall_1d_template/{attributes: num_events, called_events, mean_qscore, strand_score, sequence_length, stay_prob, step_prob, skip_prob}
|
```

Multi-read .fast5 file structure for local basecalling

```
/{attributes: file_version}
|-read_[read_id]/{top-level group for each read file. Attributes: run_id}
| |-Analyses/
| | |-Segmentation_000{attributes: name, version, time stamp}
| | | |-Summary/{attributes: return_status}
| | | | |-segmentation/{attributes: has_template, has_complement, first_sample_template, duration_template, first_sample_complement, duration_complement}
| | |-Basecall_1D_000/{attributes: name, version, time stamp}
| | | |-BaseCalled_template/
| | | | |-Fastq{string}
| | | |-Summary/{attributes: return_status}
| | | | |-basecall_1d_template/{attributes: num_events, called_events, mean_qscore, strand_score, sequence_length, stay_prob, step_prob, skip_prob}
| |-Raw/{attributes: start_time, duration, read_number, start_mux, read_id}
| | |-Signal{samples}
| |-tracking_id/{attributes: standard tracking fields}
| |-channel_id/{attributes: channel_number, digitisation, offset, range, sampling_rate}
| |-context_tags/{attributes: set when the experiment is configured}
```

Summary group for 1D basecalling

The Summary group contains one sub-group for each main step in the analysis. Within each of these subgroups are key-value pairs stored as attributes that provide useful details about the outcome of that analysis step. These subgroups and attributes are described below.

basecall_1d_template

This section contains details about the 1D basecall of the template section. These results can help with interpretation of the basecalled sequence and can often indicate why there are differences in basecalling accuracy for different reads.

- **num_events** - number of events in the section
- **called_events** - number of events remaining in the section after the basecaller has removed outliers
- **mean_qscore** - Phred score corresponding to the estimated error rate for the entire read
- **strand_score** - scoree indicating how well the data fits the model
- **sequence_length** - length (in bases) of the basecalled sequence. This is not identical to called_events because skips and stays are taken into account when generating the called sequence
- **skip_prob/stay_prob/step_prob** - the relative transition probabilities estimated from the neural network

Data analysis in EPI2ME

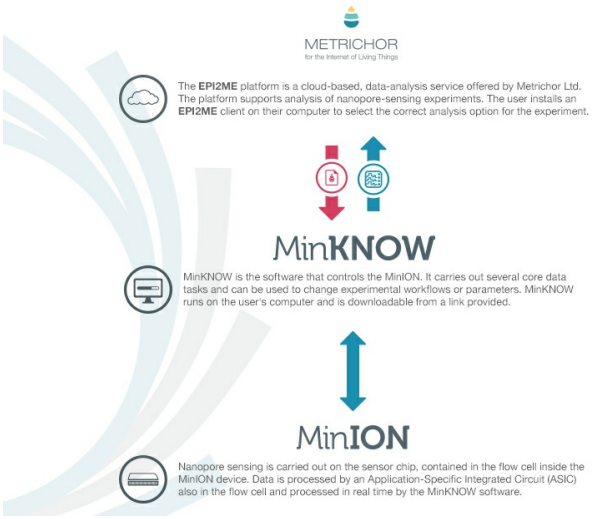
The EPI2ME platform is a cloud-based data analysis service

The EPI2ME platform (created by Metrichor Ltd., a subsidiary of Oxford Nanopore Technologies) offers a number of workflows for end-to-end analysis of nanopore data.

In a typical experiment, the user would prepare a DNA or RNA library for sequencing on a nanopore sequencing device. The sequencing experiment is started from the MinKNOW software, which controls the device and translates the electrical signals from each DNA/RNA read into basecalls. The basecalls for each read are saved in the .fast5 or FASTQ file format.

The basecalled read files can then be uploaded to the EPI2ME platform via a piece of software called the EPI2ME Desktop Agent. The user selects a workflow in the Agent such as 16S alignment or Barcoding, and the Agent transfers the read files one by one into the cloud for real-time analysis.

The analysis results are generated and updated in real-time in the form of a report in the EPI2ME platform.



EPI2ME availability

You can log into the EPI2ME website using the Single Sign-On credentials used for your Nanopore Community account.

Instructions for installing and running the EPI2ME Agent can be found in the EPI2ME protocol.

Oxford Nanopore Technologies tools and pipelines

Oxford Nanopore Technologies tools and pipelines

Oxford Nanopore Technologies' GitHub repository contains a number of data analysis tools created by our R&D division. Most of the tools require some bioinformatics knowledge, and use of the command line. Several pipelines, for example *pinfish* (a tool used for annotating genomes using long-read transcriptomics data), are described in more detail in our Bioinformatics tutorials. These tutorials contain step-by-step instructions for answering a range of biological questions using Oxford Nanopore- or Community-developed analysis tools. Example datasets are provided for practice.

Custom analysis

Third-party tools

The Oxford Nanopore Resource Centre collates all Community-developed data analysis tools (available under the "Tools" tab). Most tools are available on GitHub, and require some knowledge of bioinformatics and use of the command line.