9. Supplementary data

S.1-Supplementary codes:

Supplementary code 1. **Bash script for assemblies**. Each assembler command line is inserted into a time command for measuring the total assembly time. $filepath is the variable including the absolute path of the input reads. $resultdir is the variable including the absolute path of the results directory (folder) and $size is the variable including the estimate genome size of the community.

```
#!/bin/bash

#Ask for fastq file path

echo Please enter absolute path of file:

read filepath

echo Now please enter absolute path of results directory:

read resultdir

cd $resultdir

#All results are in $resultdir

#_____________________________________________________________________________

echo ~~~ Calling assemblers ~~~

#Create txt for time recording

touch assembly_tpo.txt

#NECAT_____________________________________________________________________________

echo ~~~ Starting NECAT ~~~

#NECAT in time file

echo Time NECAT: >>assembly_tpo.txt

cd NECAT/

#CORRECTION + ASSEMBLY + BRIDGING

{ time Necat.pl bridge config.txt 2>> ../assembly.stderr ; } 2>> ../assembly_tpo.txt

#Separation in time file

echo    >> ../assembly_tpo.txt

echo ~~~ NECAT finished ~~~
```

```
cd $resultdir

#RedBean (wdtgb2)_____________________________________________________________

echo ~~~ Running RedBean ~~~

#Redbean in time txt

echo Time Redbean: >>assembly_tpo.txt

echo    >>assembly_tpo.txt

#Folder for RedBean results

mkdir Redbean

cd Redbean/

echo ~~~ Step1. wdtgb2 starting ~~~

echo Step1: >>../assembly_tpo.txt

#Assembler

{ time /home/darwin/Descargas/Programas/Redbean/wtdbg2/wtdbg2 -x ont -t 16 -g 47m -i
$filepath -fo step1 2>>../assembly.stderr ; } 2>> ../assembly_tpo.txt

#Space

echo   >>../assembly_tpo.txt

echo ~~~ Step2. wtpoa-cns starting ~~~

echo Step 2: >>../assembly_tpo.txt

#Consenser

{ time /home/darwin/Descargas/Programas/Redbean/wtdbg2/wtpoa-cns -t 16 -i
step1.ctg.lay.gz -fo assembly.ctg.fa 2>>../assembly.stderr ; } 2>>
../assembly_tpo.txt

#Space

echo   >>../assembly_tpo.txt

echo ~~~ RedBean finished ~~~

#Return to directory of results

cd $resultdir

#Raven_________________________________________________________________________

echo ~~~ Running Raven ~~~

echo Raven: >>assembly_tpo.txt

mkdir Raven
```

```
cd Raven/

#Running Raven and storing the time

{ time raven –threads 16 $filepath > assembly.fa 2>>../assembly.stderr ; }
2>>../assembly_tpo.txt

echo   >>../assembly_tpo.txt

echo ~~~ Raven finished ~~~

cd $resultdir

#Flye_____________________________________________________________________

echo ~~~ Running metaFlye ~~~

#Flye in txt

echo Time Flye: >>assembly_tpo.txt

#Running Flye and measuring the time

{ time flye –nano-raw $filepath –out-dir $resultdir/Flye –genome-size 47m –threads 16
–meta –plasmids 2>>assembly.stderr ; } 2>>assembly_tpo.txt

#space on file

echo    >>assembly_tpo.txt

echo ~~~ metaFlye finished ~~~

#Pomoxis__________________________________________________________________

#The environment for Pomoxis must be previously initialized, the command can be found
in activate.txt, among Pomoxis program files

echo ~~~ Running Pomoxis ~~~

echo Time Pomoxis: >>assembly_tpo.txt

#Running Pomoxis and measuring time

{ time /home/darwin/Descargas/Programas/pomoxis/scripts/mini_assemble -i $filepath -o
$resultdir/Pomoxis -p assembly -l 47mb -t 16 2>>assembly.stderr ; }
2>>assembly_tpo.txt

echo    >>assembly_tpo.txt

echo ~~~ Pomoxis finished ~~~

#Canu_____________________________________________________________________

echo ~~~ Running Canu 2.0 ~~~

echo Canu: >>assembly_tpo.txt

#Running Canu and measuring time
```

```
{ time /home/darwin/Descargas/Programas/canu-2.0/Linux-amd64/bin/canu -p assembly -d
$resultdir/Canu genomeSize=47m corOutCoverage=10000 corMhapSensitivity=high
corMinCoverage=0 redMemory=32 oeaMemory=32 batThreads=16 batMemory=60 -nanopore
$filepath 2>>assembly.stderr ; } 2>>assembly_tpo.txt

#batThreads was set to 16 and the recommended batMemory=200 was changed to 60 due to
canu failure and warnings of CPU resources when 1st running it

echo ~~~ Canu finished ~~~

echo ~~~ All assemblies finished. You can now check your results ~~~
```

Supplementary code 2. **Bash script for Racon polishing**. $reads is the variable including the path to the fastq file containing the sequenced reads. The file containing each assembly draft is assembly.fasta, and assembly_racon.fasta is the file containing the new polished draft.

```
#!/bin/bash

#This script is for running minimap2 + racon. Run the script on the target folder
Correction

#Raven and Pomoxis already have run racon as part of their pipeline (2 and 4 times
respectively)

# First, we create the alignment using minimap2, then we use that alignment for
running Racon once, then we erase sam files

echo Please enter fastq reads file absolute path

read reads

echo Starting correction with Racon

echo Redbean

mkdir Redbean

cd Redbean

echo Indexing draft assembly

minimap2 -x map-ont -d indexed_draft.mmi assembly.fasta

echo Aligning

minimap2 -ax map-ont assembly.fasta $reads > aln.sam

echo Polishing with racon

racon -t16 $reads aln.sam assembly.fasta > assembly_racon.fasta

rm aln.sam

cd ../

echo NECAT

mkdir NECAT

cd NECAT

echo Indexing

minimap2 -x map-ont -d indexed_draft.mmi assembly.fasta

echo Aligning

minimap2 -ax map-ont assembly.fasta $reads > aln.sam
```

```
echo Polishing

racon -t16 $reads aln.sam assembly.fasta > assembly_racon.fasta

rm aln.sam

cd ../

echo Flye

mkdir Flye

cd Flye

echo Indexing

minimap2 -x map-ont -d indexed_draft.mmi assembly.fasta

echo Aligning

minimap2 -ax map-ont assembly.fasta $reads > aln.sam

echo Polishing

racon -t16 $reads aln.sam assembly.fasta > assembly_racon.fasta

rm aln.sam

cd ../

echo Canu

mkdir Canu

cd Canu

echo Indexing

minimap2 -x map-ont -d indexed_draft.mmi assembly.fasta

echo Aligning

minimap2 -ax map-ont assembly.fasta $reads > aln.sam

echo Polishing

racon

rm aln.sam

cd ../

echo Racon polishing finished
```

Supplementary code 3. **Bash script for running medaka after Racon polishing.** Valid for running Medaka without Racon if the input files are substituted by the draft assemblies obtained after first assembly.

```
#!/bin/bash

#Run from Toshiba/Morgane_TFG cd Benchmarking_SRA/Assembly_even

#Bench data: for Albacore v2.0.2, recommended model is r941_trans, but it is not
supported in new versions of medaka

#The data will be run in Mekaka v 0.11.5 with its default model.

echo Starting correction with Medaka

echo Bench data polishing

echo Starting with BenchEv dataset

echo Redbean

medaka_consensus -i equimolar_all.fastq -d ./Correction/Redbean/assembly_racon.fasta
-t 16 -o ./Correction/Redbean

echo Raven

medaka_consensus -i equimolar_all.fastq -d ./Assembly/Raven/assembly.fa -t 16 -o
./Correction/Raven

echo Pomoxis

medaka_consensus -i equimolar_all.fastq -d ./Assembly/Pomox_2/assembly_final.fa -t 16
-o ./Correction/Pomoxis

echo NECAT

medaka_consensus -i equimolar_all.fastq -d ./Correction/NECAT/assembly_racon.fasta -t
16 -o ./Correction/NECAT

echo Flye

medaka_consensus -i equimolar_all.fastq -d ./Correction/Flye/assembly_racon.fasta -t
16 -o ./Correction/Flye

echo Canu

medaka_consensus -i equimolar_all.fastq -d ./Correction/Canu/assembly_racon.fasta -t
16 -o ./Correction/Canu

cd ../Assembly_uneven

echo Starting with BenchHE dataset

echo Redbean

medaka_consensus -i heterogeneous_all.fastq -d
./Correction/Redbean/assembly_racon.fasta -t 16 -o ./Correction/Redbean

echo Raven

medaka_consensus -i heterogeneous_all.fastq -d ./Assembly/Raven/assembly.fa -t 16 -o
./Correction/Raven
```

```
echo Pomoxis

medaka_consensus -i heterogeneous_all.fastq -d ./Assembly/Pomox_2/assembly_final.fa -
t 16 -o ./Correction/Pomoxis

echo NECAT

medaka_consensus -i heterogeneous_all.fastq -d
./Correction/NECAT/assembly_racon.fasta -t 16 -o ./Correction/NECAT

echo Flye

medaka_consensus -i heterogeneous_all.fastq -d ./Correction/Flye/assembly_racon.fasta
-t 16 -o ./Correction/Flye

echo Canu

medaka_consensus -i heterogeneous_all.fastq -d ./Correction/Canu/assembly_racon.fasta
-t 16 -o ./Correction/Canu

cd ../../BMock12_SRA

#BMock12 dataset, obtained with Albacore v2.3.1, recommended model r941_trans

echo Starting with BMock12 dataset

echo Redbean

medaka_consensus -i BMock_por.fastq -d ./Correction/Redbean/assembly_racon.fasta -t
16 -o ./Correction/Redbean

echo Raven

medaka_consensus -i BMock_por.fastq -d ./Assembly/Raven/assembly.fa -t 16 -o
./Correction/Raven

echo Pomoxis

medaka_consensus -i BMock_por.fastq -d ./Assembly/Pomoxis/assembly_final.fa -t 16 -o
./Correction/Pomoxis

echo NECAT

medaka_consensus -i BMock_por.fastq -d ./Correction/NECAT/assembly_racon.fasta -t 16
-o ./Correction/NECAT

echo Flye

medaka_consensus -i BMock_por.fastq -d ./Correction/Flye/assembly_racon.fasta -t 16 -
o ./Correction/Flye

cd ../atcc_subsample

#MSA2006 dataset, obtained with Guppy v2.3.5

echo Starting with MSA2006 dataset

echo Redbean
```

```
medaka_consensus -i first_half_pore.fastq -d
./Correction/Redbean/assembly_racon.fasta -t 16 -o ./Correction/Redbean

echo Raven

medaka_consensus -i first_half_pore.fastq -d ./Assembly/Raven/assembly.fa -t 16 -o
./Correction/Raven

echo NECAT

medaka_consensus -i first_half_pore.fastq -d ./Correction/NECAT/assembly_racon.fasta
-t 16 -o ./Correction/NECAT

echo Flye

medaka_consensus -i first_half_pore.fastq -d ./Correction/Flye/assembly_racon.fasta -
t 16 -o ./Correction/Flye

echo Polishing with Medaka finished.
```

S.2-Assembly complications and solutions

These events helped tunning assembly parameters and input data.

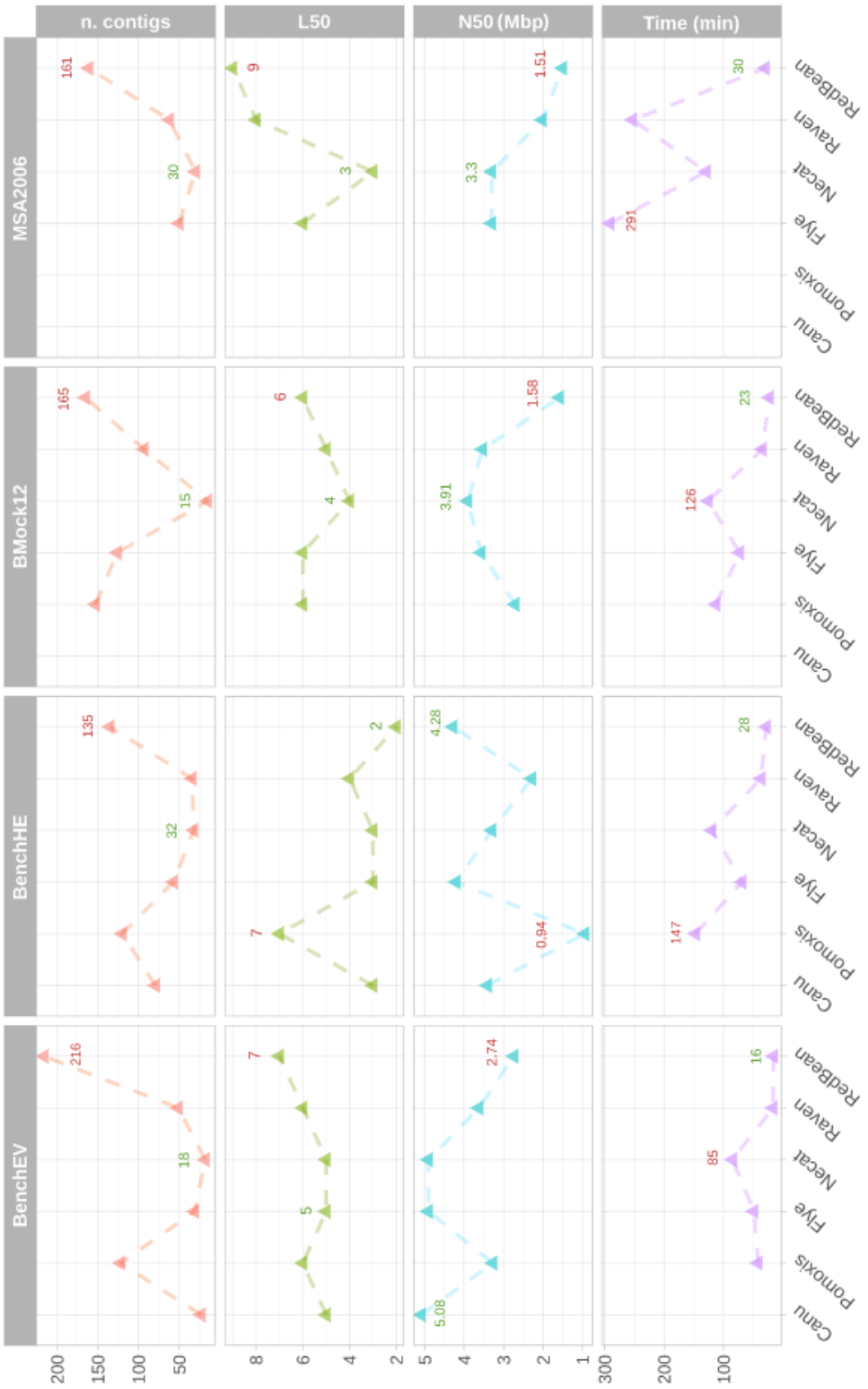
When trying to run Canu for the first sample, an error message was displayed with respect to the machine configuration memory. It advised to change batMemory and batThreads parameters: “16 CPUs and 63 GB detected, cannot run task, change batMemory and/or batThreads. The values were therefore changed to 60 and 16, respectively.

Furthermore, the MSA2006 dataset had to be subsampled because it was too large (60.2Gb) to be porechoped or assembled in the Darwin computer. Flye and Pomoxis and Porechop failed, while Necat gave a very low assembled fraction. For this reason, the datased was split in two and the assembly was run again, using only the first half of the reads.

S.3-Supplementary figures

Title
Benchmarking the MinION : Evaluating long reads for microbial profiling
Shotgun metagenome data of a defined mock community using Oxford Nanopore, PacBio and Illumina technologies
Complete, closed bacterial genomes from microbiomes using nanopore sequencing
Long-read based de novo assembly of low-complexity metagenome samples results in finished genomes and reveals insights into strain
Synthetic microbe communities provide internal reference standards for metagenome sequencing and analysis
De novo Nanopore read quality improvement using deep learning
Implications of error-prone long-read whole-genome shotgun sequencing on characterizing reference microbiomes
Discovering and exploiting multiple types of DNA methylation from individual bacteria and microbiome using nanopore sequencing
Preprint: Analysis procedures for assessing recovery of high quality, complete, closed genomes from Nanopore long read metagenome
Pathogen Detection and Microbiome Analysis of Infected Wheat Using a Portable DNA Sequencer
Metagenomic Profiling of Microbial Pathogens in the Little Bighorn River, Montana
Near-complete Lokiarchaeota genomes from complex environmental samples using long and short read metagenomic analyses
Novel prosthecate bacteria from the candidate phylum Acetothermia
Generating closed bacterial genomes from long-read nanopore sequencing of microbiomes
New tools for diet analysis: nanopore sequencing of metagenomic DNA from rat stomach contents to quantify diet
Deciphering taxonomic and functional diversity of fungi as potential bioindicators within confluence stretch of Ganges and Yamuna Rivers,
Improving recovery of member genomes from enrichment reactor microbial communities using MinION–based long read metagenomics
Rapid MinION profiling of preterm microbiota and antimicrobial-resistant pathogens
Genetic repertoires of anaerobic microbiomes driving generation of biogas
Long-read based de novo assembly of low-complexity metagenome samples results in finished genomes and reveals insights into strain
Stationary and portable sequencing-based approaches for tracing wastewater contamination in urban stormwater systems
Ultra-deep, long-read nanopore sequencing of mock microbial community standards

Supplementary figure 1. **List of studies containing ONT metagenomic data considered for this project.** Studies were retrieved during the first two weeks of April 2020. Green = selected studies. Red = discarded after first filtration. Yellow = discarded after second filtration. Grey = Not evaluable (data employed by Latorre-Pérez *et al.* (2019)).



Supplementary figure 2. **General performance of assemblers (higher size).**

Microorganisms

	BenchEV						BenchHE					
	Canu	Flye	Necat	Pomoxis	Raven	Redbean	Canu	Flye	Necat	Pomoxis	Raven	Redbean
C. glutamicum	202.08	188.29	193.50	192.38	203.14	201.90	179.41	148.69	155.77	233.13	158.24	501.80
B. licheniformis	357.43	341.88	344.34	351.86	346.47	346.69	263.63	249.63	254.50	431.01	254.66	252.36
X. campestris	84.14	71.74	72.57	81.65	72.62	78.96	1,070.57	1,555.09	-	1,263.61	-	1,195.75
E. hormaechei	262.75	250.53	248.83	365.97	328.64	303.40	339.82	319.80	365.48	421.13	368.12	368.02
S. fonticola	158.70	155.45	156.92	210.50	160.90	172.84	157.55	154.01	156.89	177.28	166.42	171.53
A. xylosoxidans	106.29	91.24	92.30	98.42	96.69	99.71	1,343.65	-	-	-	-	1,353.09
M. luteus	205.70	186.38	198.40	214.10	220.09	203.73	1,396.93	1,793.14	-	1,272.67	843.27	1,437.65
Cr. sakazakii	297.34	291.56	286.36	461.85	339.79	337.85	236.78	205.64	204.57	449.77	230.17	260.19
S. saprophyticus	249.59	239.61	240.84	245.81	243.21	242.04	-	-	-	-	-	-
Ch. violaceum	97.84	91.54	92.18	104.75	97.19	96.29	0.00	-	-	-	-	-
P. odorifer	316.46	313.22	314.64	323.56	318.55	316.12	-	-	-	-	-	-
D. solani	147.00	146.85	145.70	185.74	155.43	163.31	-	-	-	-	-	-

	BMock12				
	Flye	Necat	Pomoxis	Raven	Redbean
Cohaes. ES047	182.89	183.65	189.98	188.99	186.36
Halomonas HL4	205.26	197.20	399.01	228.29	243.59
Halomonas HL93	248.50	228.04	383.95	245.84	241.91
Marin. LV10M	203.71	206.36	223.19	216.64	215.84
Marin. LV10R	162.99	163.59	173.84	173.20	171.63
Micr. DSM43904	1,543.59	-	1,151.22	1,289.75	962.53
Micr. DSM43913	1,546.92	-	1,162.35	1,280.16	1,182.66
Micr. DSM45161	-	-	-	-	-
Muric. ES050	310.71	311.04	333.82	314.38	319.37
Propion. ES041	860.16	712.89	924.70	909.67	935.06
Psychr. LV10R	222.83	222.81	279.50	226.65	228.92
Thioclava ES032	185.64	190.57	199.33	195.31	196.22

	MSA2006			
	Flye	Necat	Raven	Redbean
Enterobacter	86.69	335.09	112.29	107.61
Bacter. 9343	30.21	33.49	29.86	35.83
Bifidobacterium	128.41	-	145.81	149.55
Clostridioides	94.85	94.56	79.37	97.83
E. coli K12	110.06	92.59	164.03	172.38
Bacter. 8482	35.69	-	32.90	42.17
Salmonella	111.44	222.83	166.34	157.75
Fusobacterium	105.45	-	98.25	121.83
Helicobacter	364.30	386.17	360.65	378.02
Lactobacillus	20.46	21.07	20.86	22.11
Enterococcus	50.91	52.92	50.61	51.98
Yersinia	60.37	170.26	67.01	70.14

Plasmids

	BenchEV						BenchHE					
	Canu	Flye	Necat	Pomoxis	Raven	Redbean	Canu	Flye	Necat	Pomoxis	Raven	Redbean
Cr. sakaz. CSK1	607.00	285.41	275.81	304.57	285.68	289.05	298.17	175.72	176.77	185.82	190.70	196.70
Cr. sakaz. CSK2	1,964.36	587.64	-	429.80	-	-	567.03	202.55	-	-	-	-
Cr. sakaz. CSK3	853.02	359.17	370.40	360.34	366.80	381.62	851.15	267.51	252.33	249.52	242.39	248.80
S. sapr. pSSP1	629.32	213.24	221.04	232.23	260.72	332.82	-	-	-	-	-	-
S. sapr. pSSP2	2,789.68	315.02	385.07	1,351.11	381.51	486.77	-	-	-	-	-	-

	MSA2006			
	Flye	Necat	Raven	Redbean
Enter. pECLA	76.58	101.82	87.70	130.24
Enter. pECLB	60.25	73.89	-	92.84
Enteroc. pTEF1	70.87	63.33	241.90	501.20
Enteroc. pTEF2	57.24	60.70	292.92	622.85
Enteroc. pTEF3	183.71	27.84	180.03	-
Bacter. pBF9343	60.18	51.97	46.60	44.58

Supplementary figure 3. **Indels per assembler and microorganism, after Polishing**. Squares are filled proportionally to the number of total Indels.