Just Another Metabarcoding Pipeline (JAMP)

Thursday 27 February 2018

Vasco Elbrecht[1]

[1] Centre for Biodiversity Genomics, University of Guelph, Guelph, Ontario, Canada

Training dataset

- DNA from 34 marine specimens equimolar pooled

Leray & Knowlton 2017, PeerJ

- One mock sample sequenced with 7 PCR replicates on 2 MiSeq runs
- Test reproducebility and tagging bias

O'Donnell et al. 2016 Plos ONE

- We will only use MiSeq run one here!

PeerJ

Random sampling causes the low reproducibility of rare eukaryotic OTUs in Illumina COI metabarcoding

Matthieu Leray[1,2] and Nancy Knowlton[1]

[1] National Museum of Natural History, Smithsonian Institution, Washington, D.C., USA
[2] Smithsonian Tropical Research Institute, Smithsonian Institution, Panama City, Balboa, Ancon, Republic of Panama

ABSTRACT

DNA metabarcoding, the PCR-based profiling of natural communities, is becoming the method of choice for biodiversity monitoring because it circumvents some of the limitations inherent to traditional ecological surveys. However, potential sources of bias that can affect the reproducibility of this method remain to be quantified. The interpretation of differences in patterns of sequence abundance and the ecological relevance of rare sequences remain particularly uncertain. Here we used one artificial mock community to explore the significance of abundance patterns and disentangle the effects of two potential biases on data reproducibility: indexed PCR primers and random sampling during Illumina MiSeq sequencing. We amplified a short fragment of the mitochondrial Cytochrome c Oxidase Subunit I (COI) for a single mock sample containing equimolar amounts of total genomic DNA from 34 marine invertebrates belonging to six phyla. We used seven indexed broad-range primers and sequenced the resulting library on two consecutive Illumina MiSeq runs. The total number of Operational Taxonomic Units (OTUs) was ~4 times higher than expected based on the composition of the mock sample. Moreover, the total number of reads for the 34 components of the mock sample differed by up to three orders of magnitude. However, 79 out of 86 of the unexpected OTUs were represented by <10 sequences that did not appear consistently across replicates. Our data suggest that random sampling of rare OTUs (e.g., small associated fauna such as parasites) accounted for most of variation in OTU presence–absence, whereas biases associated with indexed PCRs accounted for a larger amount of variation in relative abundance patterns. These results suggest that random sampling during sequencing leads to the low reproducibility of rare OTUs. We suggest that the strategy for handling rare OTUs should depend on the objectives of the study. Systematic removal of rare OTUs may avoid inflating diversity based on common β descriptors but will exclude positive records of taxa that are functionally important. Our results further reinforce the need for technical replicates (parallel PCR and sequencing from the same sample) in metabarcoding experimental designs. Data reproducibility should be determined empirically as it will depend upon the sequencing depth, the type of sample, the sequence analysis pipeline, and the number of replicates. Moreover, estimating relative biomasses or abundances based on read counts remains elusive at the OTU level.

Submitted 21 October 2016
Accepted 20 January 2017
Published 22 March 2017

Corresponding author
Matthieu Leray,
leray.upmc@gmail.com

Academic editor
Tomas Hrbek

Additional Information and Declarations can be found on page 21

DOI 10.7717/peerj.3006

Subjects Ecology, Molecular Biology
Keywords Indexed PCR primers, Multiplexing, Reproducibility

Leray & Knowlton 2017, PeerJ

Hypothesis

- What do we expect?
- Ideally: one table with 7 identical samples and 34 OTUs **(No biases)**

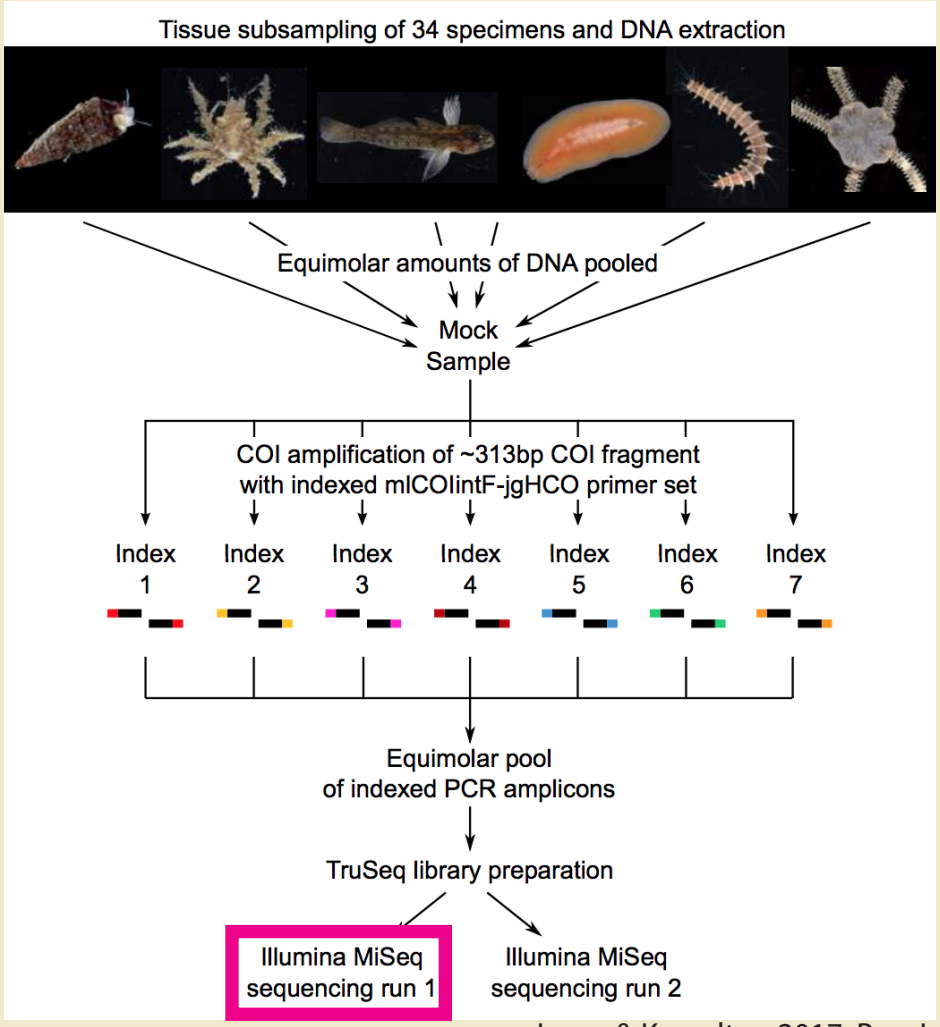
	A	B	C	D	E	F	G	H	I	J
1	sort	ID	L1	L2	L3	L4	L5	L6	L7	sequ
2	1	OTU_1	1000	1000	1000	1000	1000	1000	1000	TCTGGCCGGTAA
3	2	OTU_2	1000	1000	1000	1000	1000	1000	1000	CCTTGCAAGCAA
4	3	OTU_3	1000	1000	1000	1000	1000	1000	1000	ATTAGCAGCTGG
5	4	OTU_4	1000	1000	1000	1000	1000	1000	1000	ACTATCAGGGCC
6	5	OTU_5	1000	1000	1000	1000	1000	1000	1000	TCTTTCTAGTAG
7	6	OTU_6	1000	1000	1000	1000	1000	1000	1000	ATTAGCTGCTGC
8	7	OTU_7	1000	1000	1000	1000	1000	1000	1000	ACTTTCAGCAGG
9	8	OTU_8	1000	1000	1000	1000	1000	1000	1000	TCTCTCAAGTAA

- Not going to happen, let's figure out why in this course!

Detailed laboratory workflow

- Labortatory setup does inform & can complicate your bioinformatic analysis!
- Understanding of both needed!

- Y - forked adpters, randomly on forward and reverse direction = 50:50 reads = diversity
- See: https://youtu.be/sRATXu2Ff08?t=3m6s



Leray & Knowlton 2017, PeerJ

Bioinformatic workflow (JAMP)

1) Download sequence files, `md5`, FastQC quality control
2) Library demultiplexing [`Demultiplexing_shifted()`]
3) Merge paired end reads [`U_merge_PE()`]
4) Trim sequencing primers, for forward & reverse separately [`Cutadapt()`]
5) Reverse complement reverse sequences [`U_revcomp()`]
6) Merge Forward & R.C. reads [custom R script]
7) Filter by length (min max +/- 10bp) [`Minmax()`]
8) Expected error quality filtering [`U_max_ee(max_ee=1)`]
9) Optional: Subset each sample to 60.000 reads [`U_subset()`]
10) Cluster OTUs [`U_cluster_otus()`]
11) Assign taxonomy from BOLD [`Bold_web_hack()`]

1) MiSeq Run 1 (Paired end: R1 + R2)

- Download and extract files if you have not already

```
# Run 1, Read 1:
https://dx.doi.org/10.6084/m9.figshare.4039821.v1 (173.88 MB)
# Run 1, Read 2:
https://dx.doi.org/10.6084/m9.figshare.4039860.v1 (210.25 MB)
```

- Verify that file is downloaded correctly (Mac terminal `md5`, linux `md5sum`)

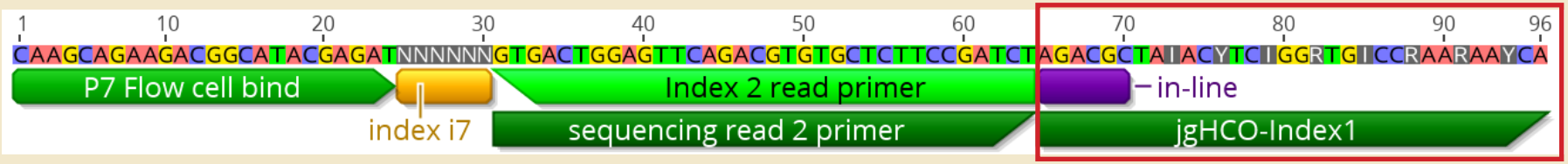
```
# Compare md5 with the ones on figshare

16_S10_L001_R1_001_run1.fastq.gz = a7de39afa0f5ea9a03ed9c3374c80321
16_S10_L001_R2_001_run1.fastq.gz = 1d87ba337ca2399cdc8d10a8ad10fba1
```

- Take a look at both files!
  1) What is simillar?
  2) How could we count the number of sequences?

Tip: View a few lines with `head`

1) Fastq file format

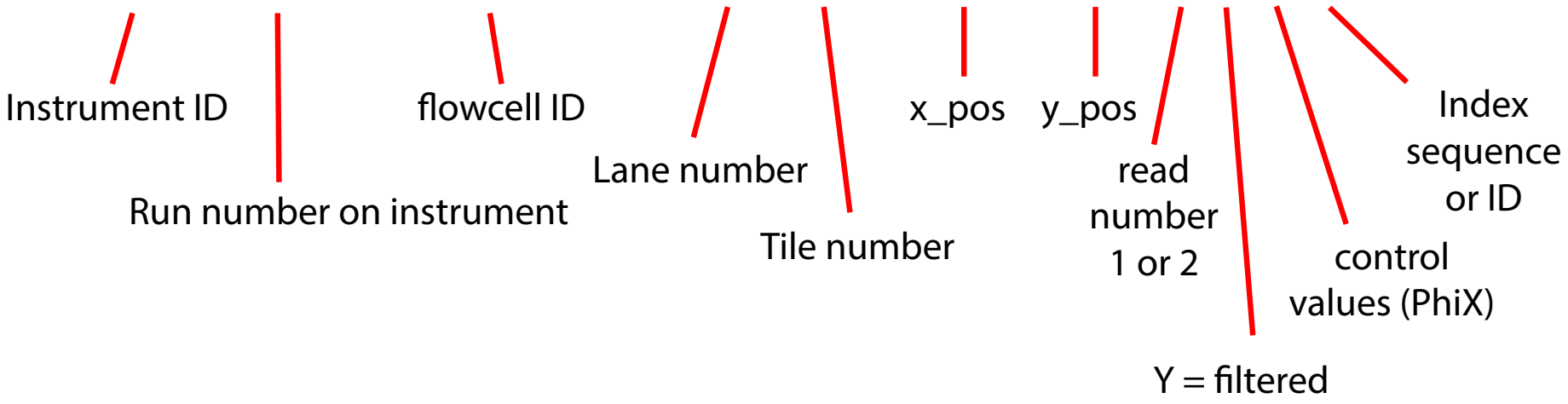
- 4 lines instead of 2 (fasta)
- Includes quality scores (ascii cooded)




```
@M03292:11:000000000-AEF91:1:2101:17896:13658 2:N:0:10
AGACGCTAGACCTCGGGATGGCCGAAGAATCAAAATAGGTGTTGGTATAAAATAGGGTCTCCTCCTC
CTGTTGGGTCAAAGAAAGTAGTGTTTATATTACGGTCTGTTAGTAACATTGTGATTGCTCCGGCAAG
GACTGGTAGGGAGAGTAGTAGTAGAATTGTTGTTAGGAGGATGGATCAAACAAAGAGGGGTGTACGG
TCCATAGTCATCCCAGGAGCGCGCATCTTAATAATTGTGGTGATAAAGTTGATGGAGGCCCTAATTG
AGGAGGCGCCTGCTAGGTTGAGGGAGATGATGG
+
@CCCCGGGGGGGGGGGGGEFGGGGGEEGGGGGGGGGGGEFGFGGGFGGGGGGGGGGGGGGGGGGGGG
GGGFGGGGGGGGGGGGGFGEGFFGGGGGGGGGGGGGGFGGGFGGGGGGGGGFGAFFGFGGGGGGGGGEGF
GGGFGGGGFGDFECCBEGGDFBFGGGCFGGGGGGCFGFGCEGGGG8DFFCFGFGFFGFFGDE?A7CD
FE*@FFEEBFFCAAFA>DDFFFFFB=(*=37?)6)).9@?<;=C3:C=A/).9F3()6?((.(/)-6
4?37:019041((,)3@2).5(43-(,(56<>3
```

1) Illumina file structure

```
16_S10_L001_R1_001_run1.fastq
@M03292:11:000000000-AEF91:1:1101:20000:1095 1:N:0:10

16_S10_L001_R2_001_run1.fastq
@M03292:11:000000000-AEF91:1:1101:20000:1095 2:N:0:10
```



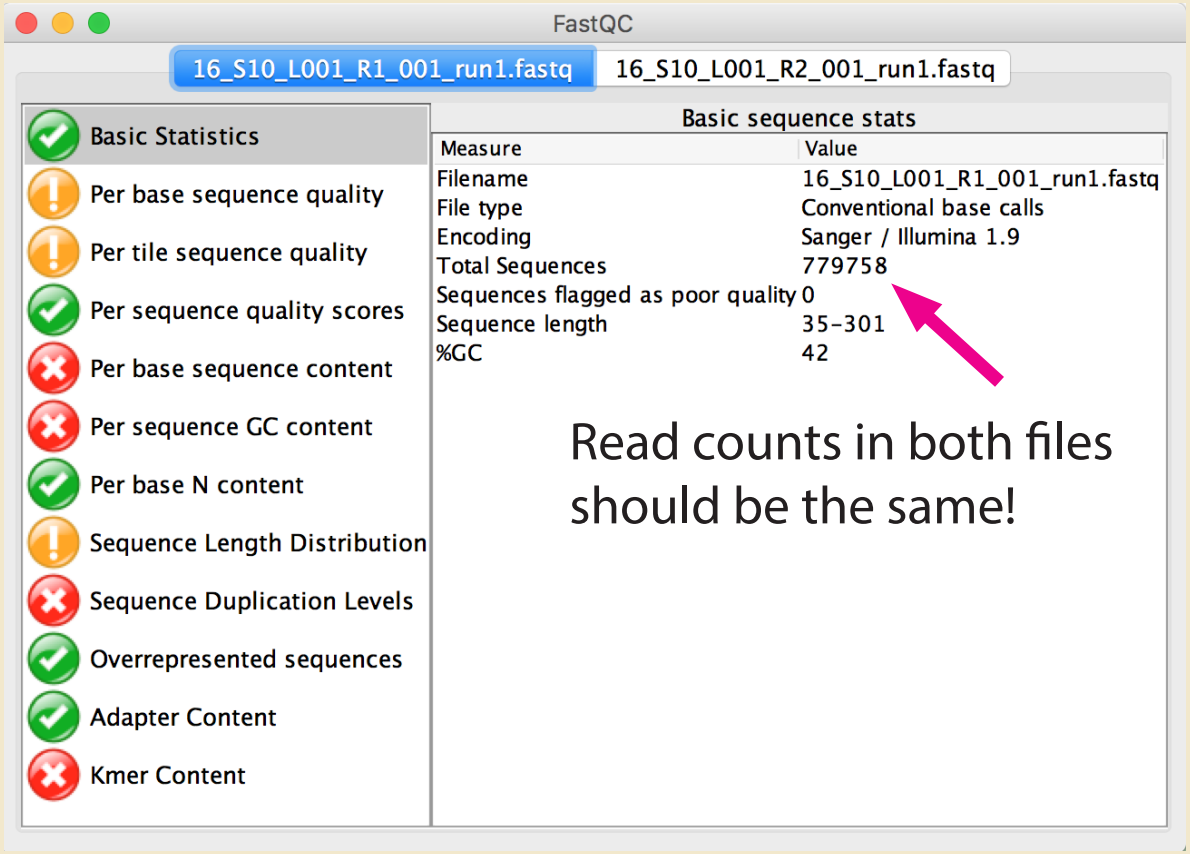
- Here: No index needed, inline tagging

2) FastQC - quality check

- Quick quality control of HTS raw data (can load sevaral files at once, also .gz)
- Warnings do not always fit metabarcoing data (e.g. Duplication levels)

- Caution! Builds averages for higher possitions

- Tip: All plots can be exported as images



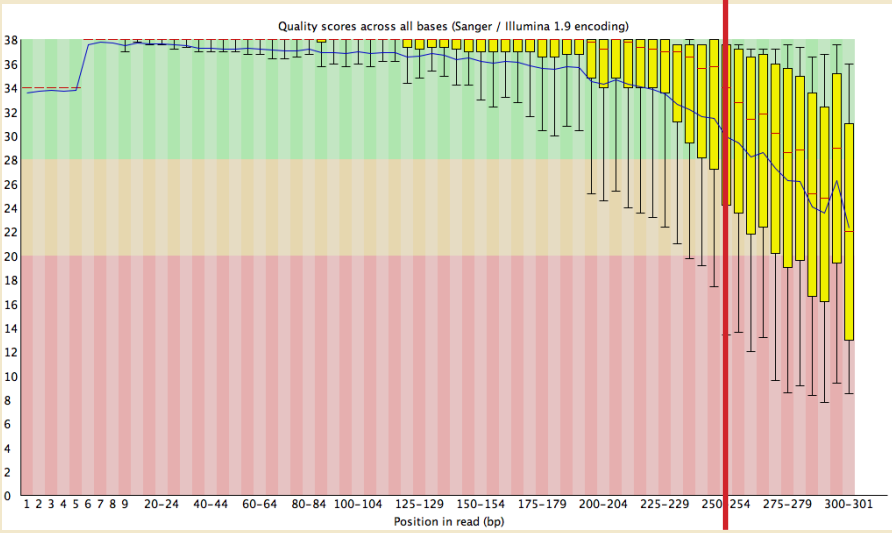
2) FastQC - Read quality

- Read 1 has a higher quality than read 2, deteriorates towards the end
- Currently 300 bp not needed, 250 bp sufficient
- Hopefully improves in the future

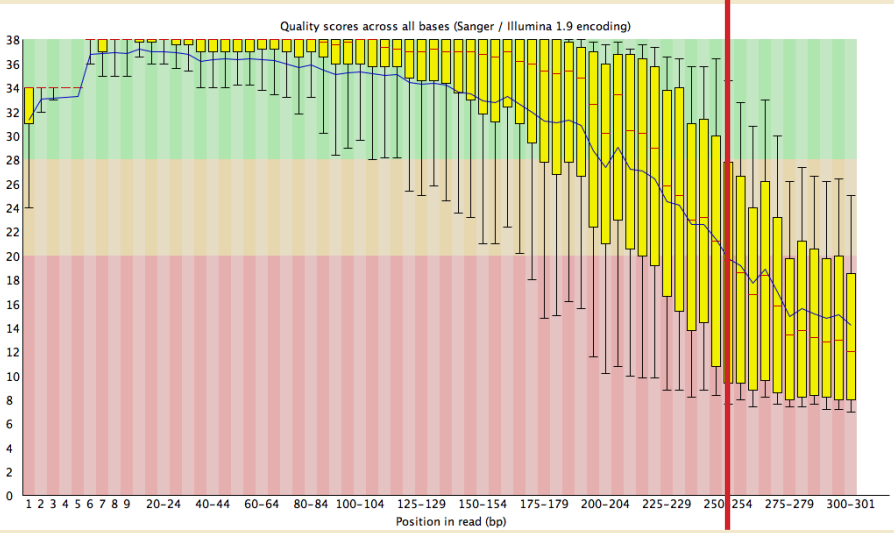
Phred Quality Score	Base call accuracy
10	90%
20	99%
30	99.9%
40	99.99%
50	99.999%
60	99.9999%

en.wikipedia.org/wiki/Phred_quality_score

Read 1



Read 2

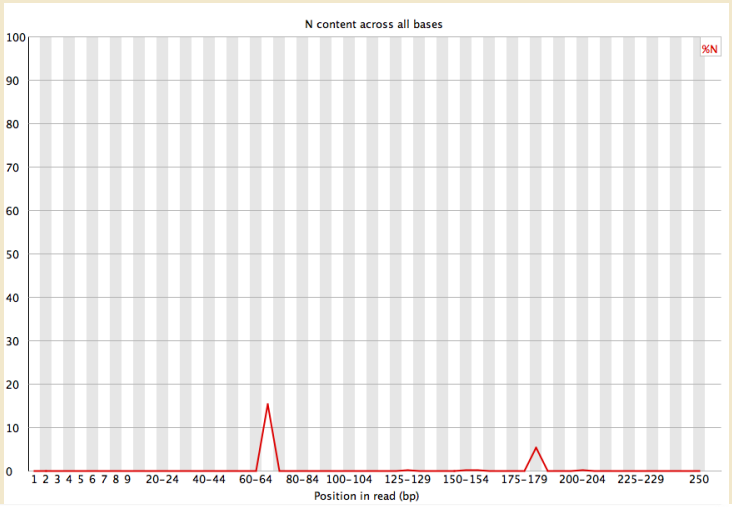


2) FastQC - Other issues

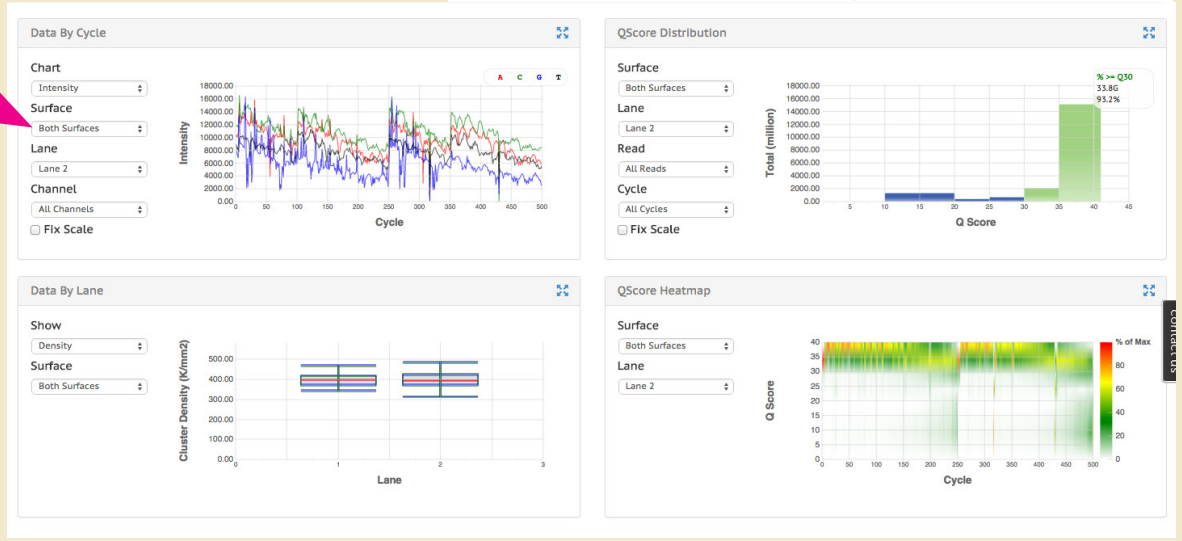
- N counts from HiSeq run (air bubble), did cause N reads
- More problems / sollutions: QCfail.com

QCFAIL.com

Articles about common next-generation sequencing problems



- If possible get Illumina base space screen shot!



- Alternatives e.g. HTQC Yang et al. 2013 BMC

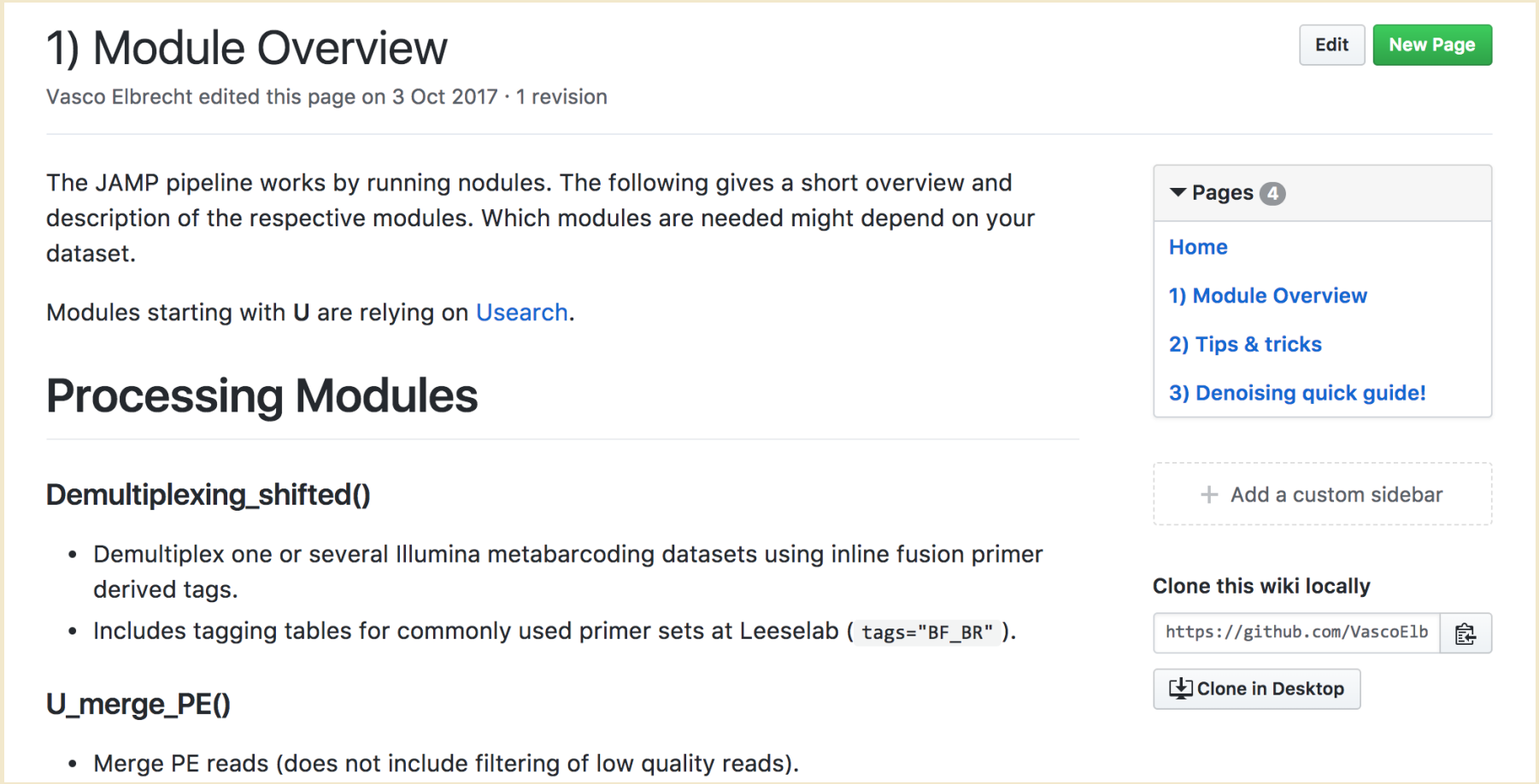
The JAMP pipeline

- Modular R metabarcoding package, needs:
  - Cutadapt
  - Usearch / Vsearch
- https://github.com/VascoElbrecht/JAMP
- Get to know R, try the `swirl` package

```
Installing JAMP (needs cutadapt, usearch, vsearch)

setwd("~/Documents/UNI_und_VORLESUNGEN/GitHub/JAMP/")

install.packages(c("bold", "XML", "seqinr"), dependencies=T)
install.packages("JAMP", repos = NULL, type="source")

library("JAMP") #v0.34
library("bold")
library("XML")
```

Not oficially supported, but wiki available

- https://github.com/VascoElbrecht/JAMP/wiki/1)-Module-Overview



1) Module Overview

Edit | New Page

Vasco Elbrecht edited this page on 3 Oct 2017 · 1 revision

The JAMP pipeline works by running nodules. The following gives a short overview and description of the respective modules. Which modules are needed might depend on your dataset.

Modules starting with **U** are relying on Usearch.

Processing Modules

Demultiplexing_shifted()

- Demultiplex one or several Illumina metabarcoding datasets using inline fusion primer derived tags.
- Includes tagging tables for commonly used primer sets at Leeselab (`tags="BF_BR"`).

U_merge_PE()

- Merge PE reads (does not include filtering of low quality reads).

Pages 4

Home

1) Module Overview

2) Tips & tricks

3) Denoising quick guide!

+ Add a custom sidebar

Clone this wiki locally

https://github.com/VascoElb

Clone in Desktop

2) Sample demultiplexing

- Demultiplexing using JAMP!

```
# Set your folder
setwd("~/Documents/UNI_und_VORLESUNGEN/GitHub/JAMP/")

library("JAMP") #v0.34
```

- File for tagging combinations + file names to save samples are needed!

```
Demultiplexing_shifted(
file1="../16_S10_L001_R1_001_run1.fastq",   # File 1
file2="../16_S10_L001_R2_001_run1.fastq",   # File 2
tags="../_converter/indexe_1.csv",          # index table
combinations="../_converter/combos_1.csv") # sample table
                                            # all need "../"
```

- Good chance to think about useful sample sorting / grouping

2) Tagging (many different strategies)

- 7 samples, all with the same 6 bp in-line index, followed by the forward or reverse primer (illumina TruSeq adapters ligated)
- Same index for each replicate (forward and reverse)

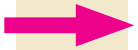
	Primer.label	Primer.sequence
1	mlCOIintF-Index1	AGACGCGGWACWGGWTGAACWGTWTAYCCYCC
2	mlCOIintF-Index2	AGTGTAGGWACWGGWTGAACWGTWTAYCCYCC
3	mlCOIintF-Index3	ACTAGCGGWACWGGWTGAACWGTWTAYCCYCC
4	mlCOIintF-Index4	ACAGTCGGWACWGGWTGAACWGTWTAYCCYCC
5	mlCOIintF-Index5	ATCGACGGWACWGGWTGAACWGTWTAYCCYCC
6	mlCOIintF-Index6	ATGTCGGGWACWGGWTGAACWGTWTAYCCYCC
7	mlCOIintF-Index7	ATAGCAGGWACWGGWTGAACWGTWTAYCCYCC
8	jgHCO-Index1	AGACGCTAIACYTCIGGRTGICCRAARAAYCA
9	jgHCO-Index2	AGTGTATAIACYTCIGGRTGICCRAARAAYCA
10	jgHCO-Index3	ACTAGCTAIACYTCIGGRTGICCRAARAAYCA
11	jgHCO-Index4	ACAGTCTAIACYTCIGGRTGICCRAARAAYCA
12	jgHCO-Index5	ATCGACTAIACYTCIGGRTGICCRAARAAYCA
13	jgHCO-Index6	ATGTCGTAIACYTCIGGRTGICCRAARAAYCA
14	jgHCO-Index7	ATAGCATAIACYTCIGGRTGICCRAARAAYCA

2) Index & sample table (folder `_converter`)

- Demultiplexing based on `barcode`, 6 bp removed
- Only if fully matched sample `ID` in combination file
  - Files can be renamed, but should end with `r1.txt` or `r2.txt`

indexe_1.csv

barcode	rm	ID	comment
AGACGC	6	i1	AGACGCGGWACWGGWTGAACWGTWTAYCCYCC
AGTGTA	6	i2	AGTGTAGGWACWGGWTGAACWGTWTAYCCYCC
ACTAGC	6	i3	ACTAGCGGWACWGGWTGAACWGTWTAYCCYCC
ACAGTC	6	i4	ACAGTCGGWACWGGWTGAACWGTWTAYCCYCC
ATCGAC	6	i5	ATCGACGGWACWGGWTGAACWGTWTAYCCYCC
ATGTCG	6	i6	ATGTCGGGWACWGGWTGAACWGTWTAYCCYCC
ATAGCA	6	i7	ATAGCAGGWACWGGWTGAACWGTWTAYCCYCC



combos_1.csv

ID	File1	File2
i1_i1	L1_r1.txt	L1_r2.txt
i2_i2	L2_r1.txt	L2_r2.txt
i3_i3	L3_r1.txt	L3_r2.txt
i4_i4	L4_r1.txt	L4_r2.txt
i5_i5	L5_r1.txt	L5_r2.txt
i6_i6	L6_r1.txt	L6_r2.txt
i7_i7	L7_r1.txt	L7_r2.txt

2) Check abundance!

- Sanity chack, look at your in between data:
  Lots of problems, artefacts or undocumented software changes!

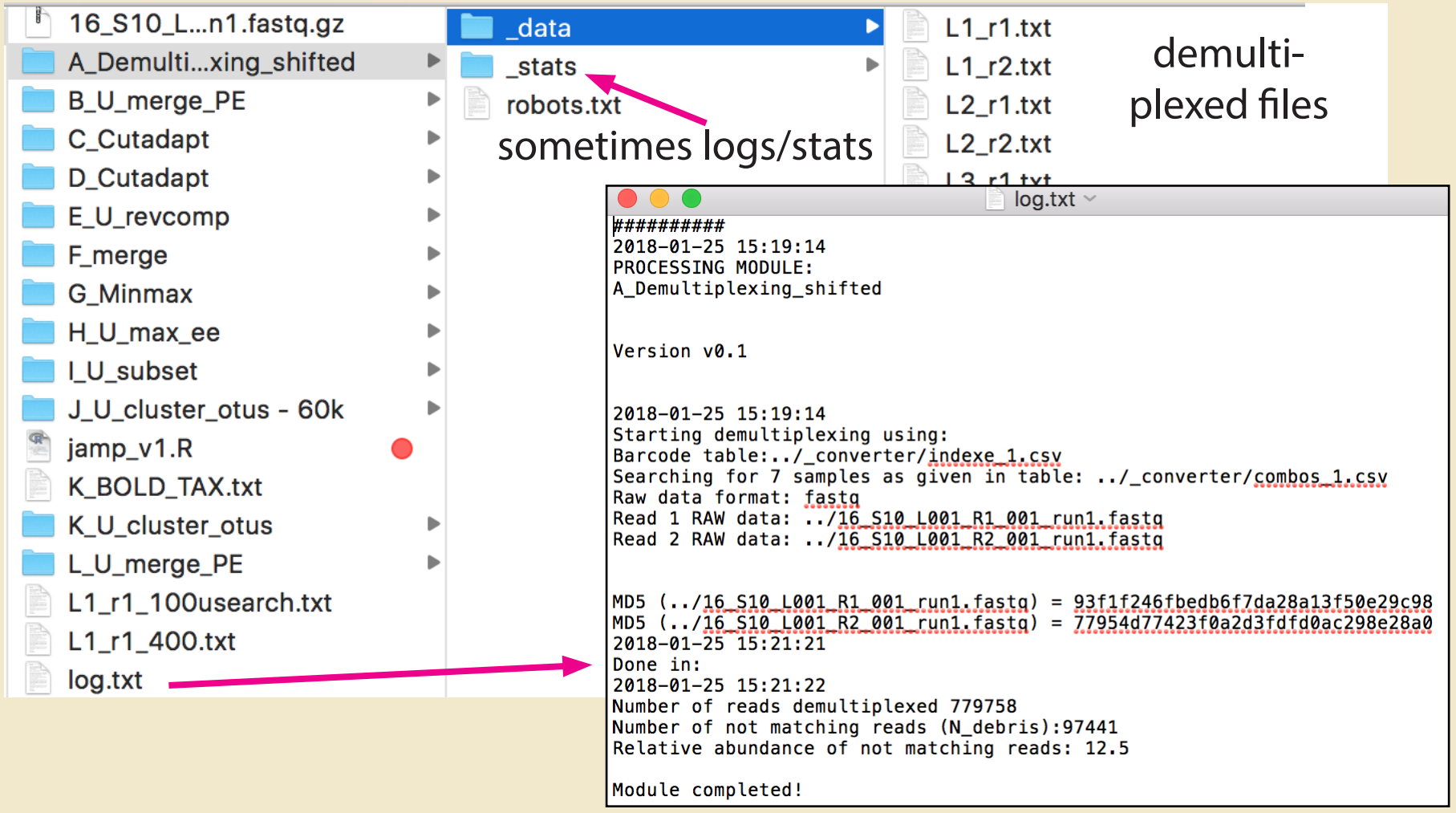
- Load things into Geneious, look at log files, summary statistics

- Big data sets: subset reads e.g. using the Terminal or Usearch

```
cd File/path                                          # set folder
head -n 400 A_Demultiplexing_shifted/_data/L1_r1.txt > L1_r1_400.txt

usearch -fastx_subsample A_Demultiplexing_shifted/_data/L1_r1.txt
-fastqout L1_r1_100usearch.txt -sample_size 100 -sizein -sizeout
```

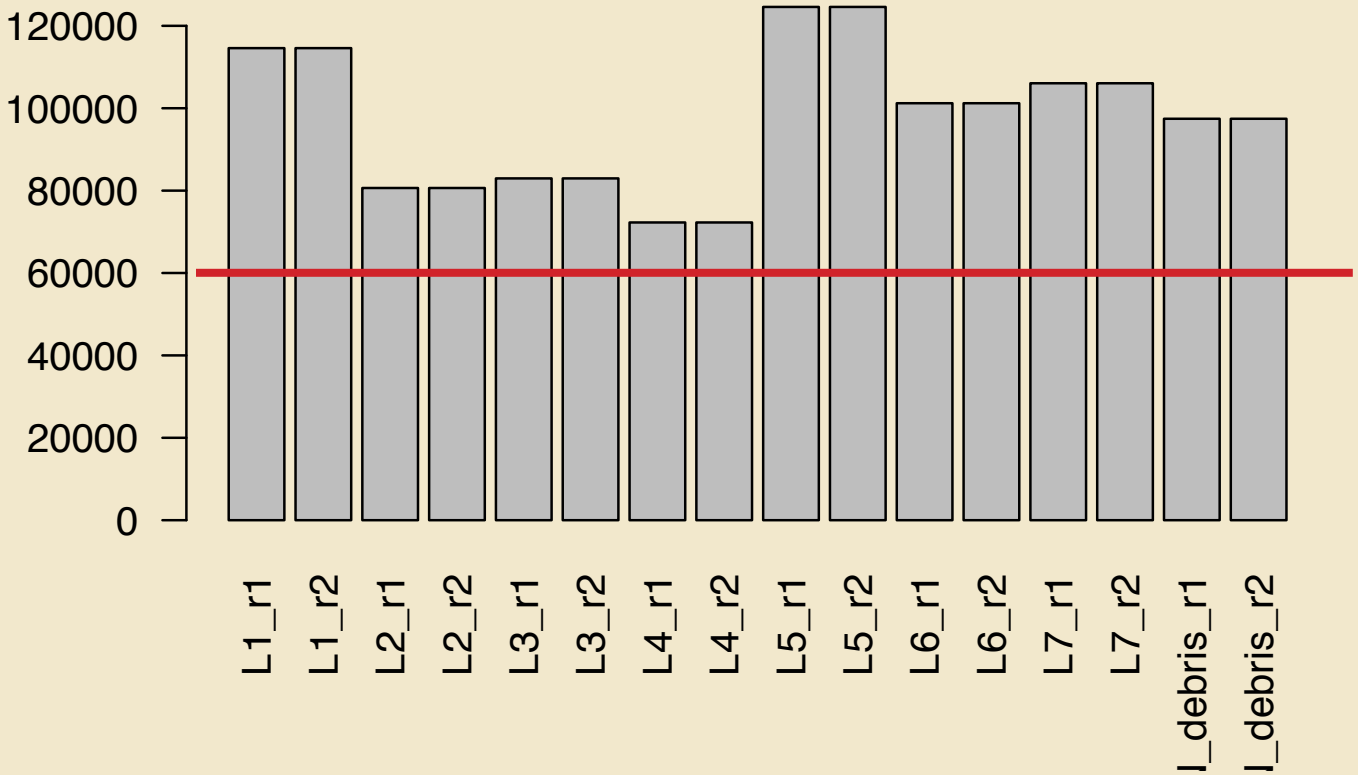
- JAMP provides you with statistics
  of proportions of reads loast in each step

JAMP folder structure



2) Read abundance after multiplexing

- Samples should have similar sequencing depth
  Only cluster a read subset, or use relative abundance in OTU table!

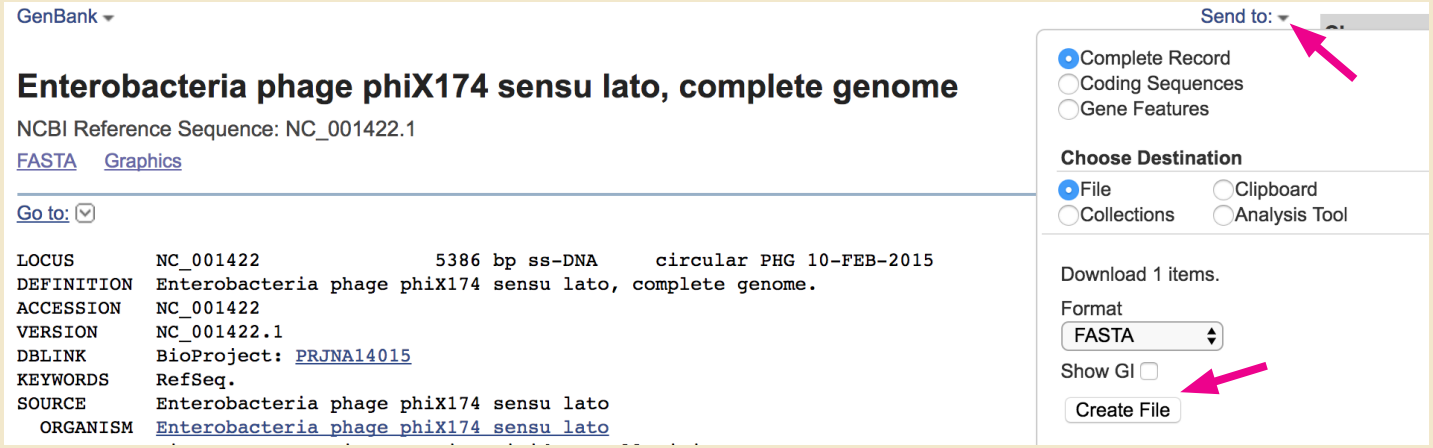


Different sequencing depth = differ-ent sampling efford!

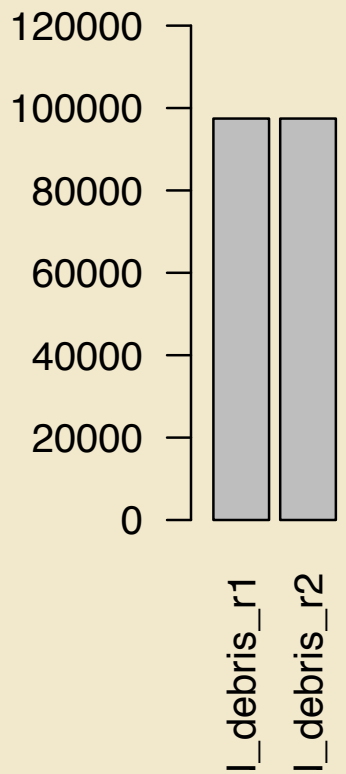
Potential bias!

2) Optional: Check for PhiX

- Virus genome, often spiked into Illumina sequencing runs to increase sequence diversity = improved throughput / quality
- https://www.ncbi.nlm.nih.gov/nucleotide/NC_001422




```
#Map reads against PhiX in R

system2("usearch", "-usearch_global A_Demultiplexing_
shifted/_data/N_debris_r1.txt -db PhiX.fasta -id 0.9
-strand both -blast6out PhiX_table.txt")
```



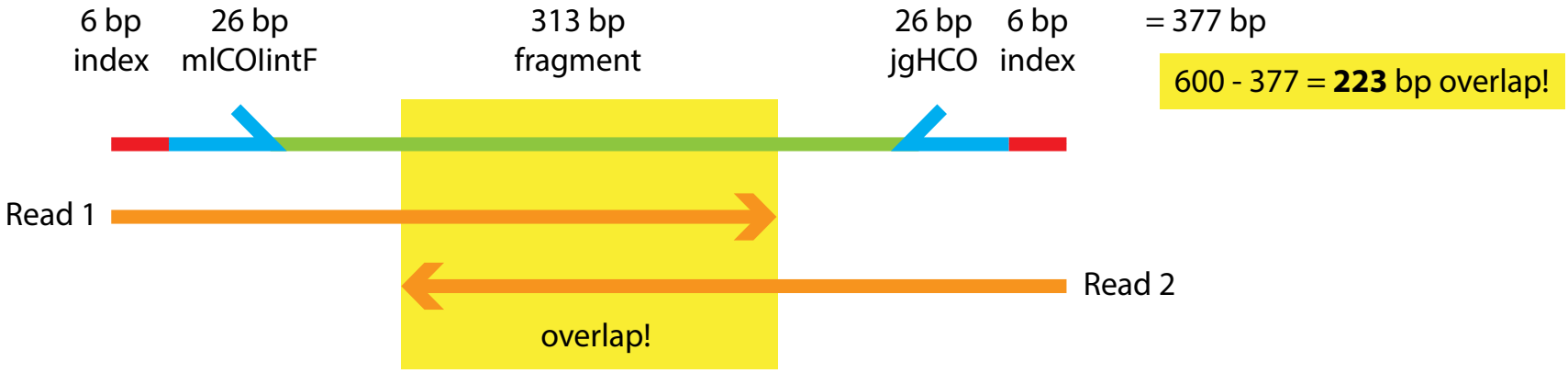
2) Optional: Results

- Only little PhiX used

```
> system2("usearch", "-usearch_global A_Demultiplexing_shifted/
_data/N_debris_r1.txt -db PhiX.fasta -id 0.9 -strand both -blast6out
PhiX_table.txt")
usearch v10.0.240_i86osx32, 4.0Gb RAM (17.2Gb total), 8 cores
(C) Copyright 2013-17 Robert C. Edgar, all rights reserved.
http://drive5.com/usearch

License: luckylion07@googlemail.com

00:00 1.5Mb   100.0% Reading PhiX.fasta
00:00 1.5Mb   100.0% Masking (fastnucleo)
00:00 2.4Mb   100.0% Word stats
00:00 2.4Mb   100.0% Alloc rows
00:00 2.5Mb   100.0% Build index
00:13 55Mb    100.0% Searching N_debris_r1.txt, 2.3% matched
>
```

3) Paired end merging (read 1+2)

- MiSeq 300 bp PE run = 233 bp overlap!
- Absolute minimum ~30 bp, but not recommended due to low sequence quality in read 2 (larger overlap can compensate for that) Eren et al. 2013, Plos ONE



A + T = A

- Do not use paired end merging as a quality filtering step
- Many sequencing errors towards end of reads, but usually reflected in Phred scores: Confilcts = higher quality base used, but score reduced!

3) Paired end merging using JAMP

- Usearch `fastq_mergepairs` integrated in JAMP `U_merge_PE()`
- Will automatically detect files from previous folder!

```
U_merge_PE(fastq_pctid=75) # allows 25% mismatches in overlap
```

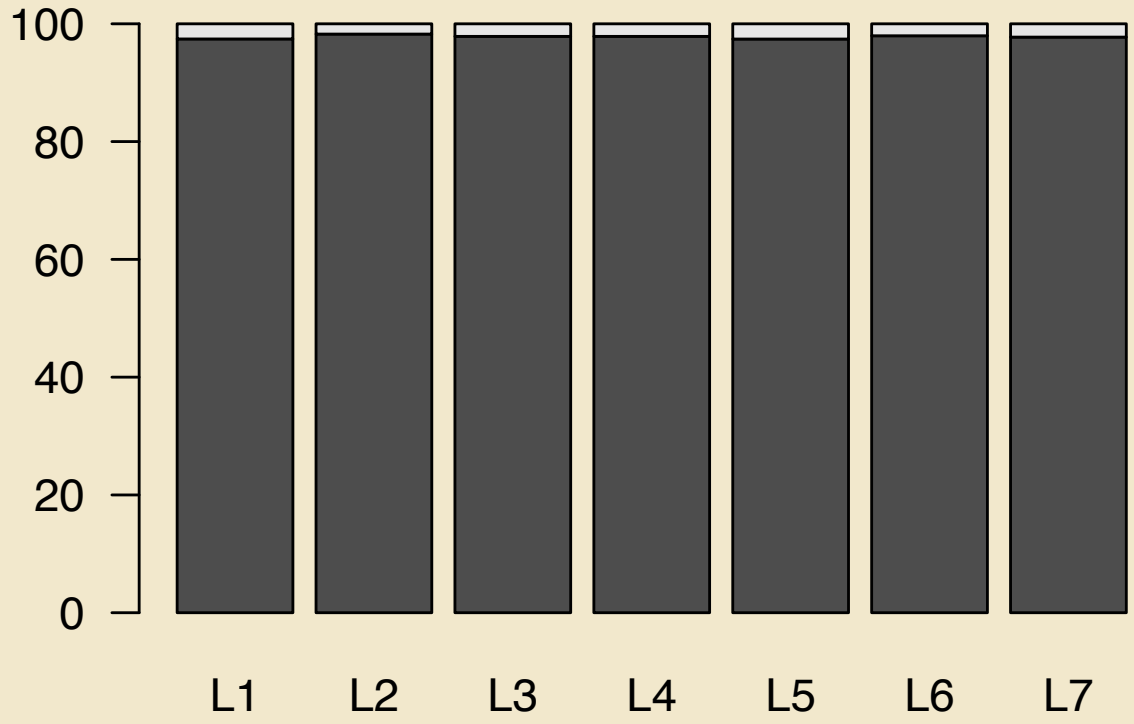
- IMO: Overlap / mismatch number could but should not be used for quality filtering! That an extra step!

```
Under the hood:
usearch -fastq_mergepairs Reads_r1.txt -reverse Reads_r1.txt
-fastqout Reads_PE.txt -report "report.txt" -fastq_maxdiffs 99
-fastq_pctid 75 -fastq_trunctail 0
```

- Tip: Read the Usearch documentation, it's well written and a great source of metabarcoding wisdom
- However, changes to Usearch happen often undocumented. Thus check in each step if unusual many reads are discarded (`_stats/` folder)

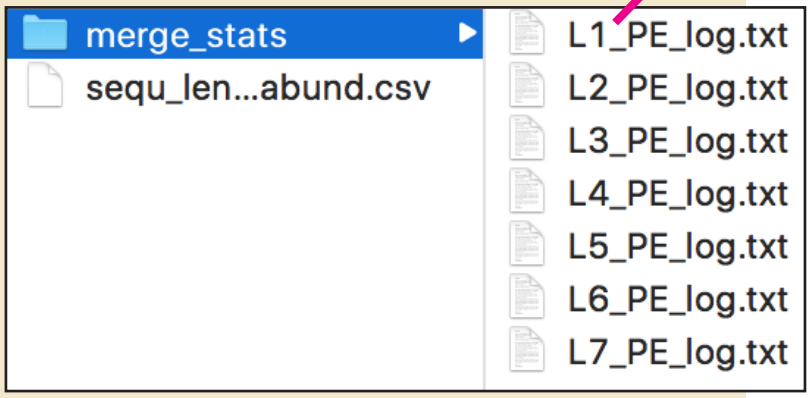
3) Reads lost in PE

- Most sequences should be successfully merged!
- In this case 2.21% (SD = 0.3%) sequences are discarded, lets look at the log



3) PE merging log

- Can be found in the `_stats` folder
- To many differences = sequencing errors




```
Merge
  Fwd ../A_Demultiplexing_shifted/_data/L1_r1.txt
  Rev ../A_Demultiplexing_shifted/_data/L1_r2.txt
  Keep read labels
  111644 / 114589 pairs merged (97.4%)

Merged length distribution:
        62  Min
       364  Low quartile
       365  Median
       365  High quartile
       548  Max

Totals:
    114589  Pairs (114.6k)
    111644  Merged (111.6k, 97.43%)
      1378  Alignments with zero diffs (1.20%)
      2653  Too many diffs (> 99) (2.32%)
       292  No alignment found (0.25%)
         0  Alignment too short (< 16) (0.00%)
       393  Staggered pairs (0.34%) merged & trimmed
    224.93  Mean alignment length
    364.04  Mean merged length
      1.68  Mean fwd expected errors
      5.98  Mean rev expected errors
      0.59  Mean merged expected errors
```

4) Primer removal with Cutadapt

- Primers should be removed before applying quality filtering
- Inosine has to be replaced with N

```
Cutadapt(forward="GGWACWGGWTGAACWGTWTAYCCYCC", # mlCOIintF
reverse="TANACYTCNGGRTGNCCRAARAAYCA") # jgHCO, I replaced with N
```

- Under the hood, removing forward from the left and reverse from the right of the sequence.
- Two separate times, so all not trimmed sequences are discarded

```
#in console
cutadapt -g ^FORWARD_PRIMER -o _data/temp.txt imput_file.txt -f fastq
--discard-untrimmed

cutadapt -a RVERSE_PRIMER$ -o _data/L1_PE_cut.fastq _data/temp.txt
-f fastq --discard-untrimmed
```

4) Cutadapt log file

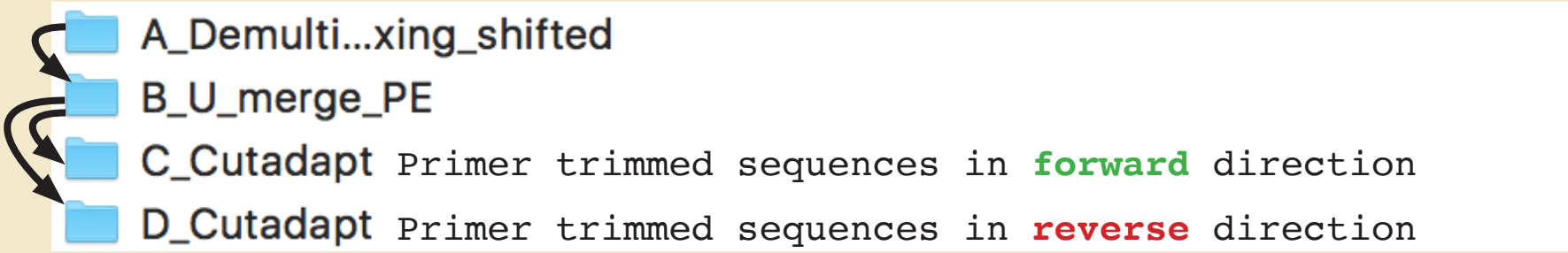
- Allows for 10% errors in the matched primer sequence
- We run the program twice to remove forward and reverse primer
- In the first step only around 50% of the forward primer is detected
- Library prep ligates Y on both ends of the dsDNA
  = 2 strands, one red in forward direction and the other in reverse direction first!

```
This is cutadapt 1.9 with Python 2.7.10
Command line parameters: -g ^GGWACWGGWTGAACWGTWTAYCCYCC -o _data/temp.txt ../B_U_merge_PE/
_data/L1_PE.fastq -f fastq --discard-untrimmed
Trimming 1 adapter with at most 10.0% errors in single-end mode ...
Finished in 3.07 s (28 us/read; 2.18 M reads/minute).

=== Summary ===

Total reads processed:                111,644
Reads with adapters:                   57,427 (51.4%)
Reads written (passing filters):       57,427 (51.4%)

Total basepairs processed:    40,643,245 bp
Total written (filtered):     19,408,400 bp (47.8%)

=== Adapter 1 ===

Sequence: GGWACWGGWTGAACWGTWTAYCCYCC; Type: anchored 5'; Length: 26; Trimmed: 57427 times.

No. of allowed errors:
0-9 bp: 0; 10-19 bp: 1; 20-26 bp: 2

Overview of removed sequences
length  count   expect  max.err error counts
24      100     0.0     2       0 0 100
25      970     0.0     2       0 941 29
26      55812   0.0     2       54401 1380 31
27      361     0.0     2       0 343 18
28      184     0.0     2       0 0 184

This is cutadapt 1.9 with Python 2.7.10
Command line parameters: -a TGRTTYTTYGGNCAYCCNGARGTNTA$ -o _data/L1_PE_cut.fastq _data/
temp.txt -f fastq --discard-untrimmed
Trimming 1 adapter with at most 10.0% errors in single-end mode ...
Finished in 2.48 s (43 us/read; 1.39 M reads/minute).

=== Summary ===

Total reads processed:                 57,427
Reads with adapters:                   57,271 (99.7%)
Reads written (passing filters):       57,271 (99.7%)

Total basepairs processed:    19,408,400 bp
Total written (filtered):     17,869,262 bp (92.1%)

=== Adapter 1 ===

Sequence: TGRTTYTTYGGNCAYCCNGARGTNTA; Type: anchored 3'; Length: 26; Trimmed: 57271 times.

No. of allowed errors:
0-9 bp: 0; 10-19 bp: 1; 20-26 bp: 2
```

4) Cutadapt again (but with primers switched)

- We use Cutadapt again, but this time start with the reverse primer followed by the forward primer
- Important: Use the PE merged sequences, using the `list.files` function in R

```
# trim reads in different orrientation
PEmerged <- list.files("~/Documents/UNI_und_VORLESUNGEN/14 Guelph/1
TEACHING/2018 metabarcoding course/1 JAMP/B_U_merge_PE/_data", full.
names=T)

Cutadapt(files= PEmerged,
forward="TANACYTCNGGRTGNCCRAARAAYCA", # jgHCO
reverse="GGWACWGGWTGAACWGTWTAYCCYCC") # mlCOIintF
```



5) Reverse complement

- Let's use Usearch to reverse complement the sequences in reverse orientation

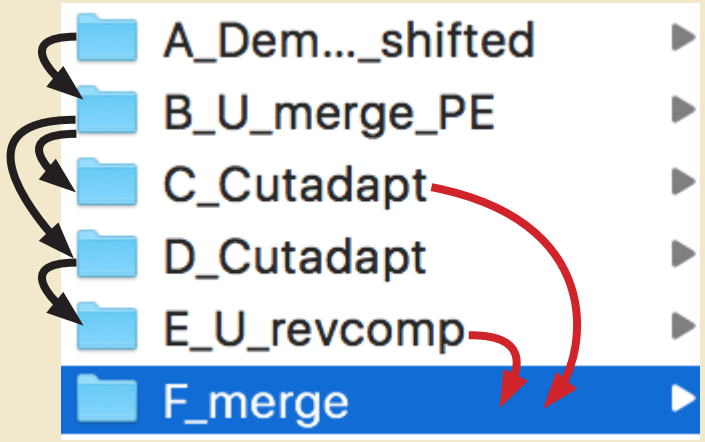
```
U_revcomp(RC=T) # RC can also be a list of sequences c(F,T,F,T)
```

- Under the hood

```
usearch -fastx_revcomp Input_file.txt -fastqout Output_file.txt
-label_suffix _RC

# No _stats data available

00:00 35Mb    100.0% Processing
```

6) Merge files from C & E

- We merge both read sets into one file using R and `cat`



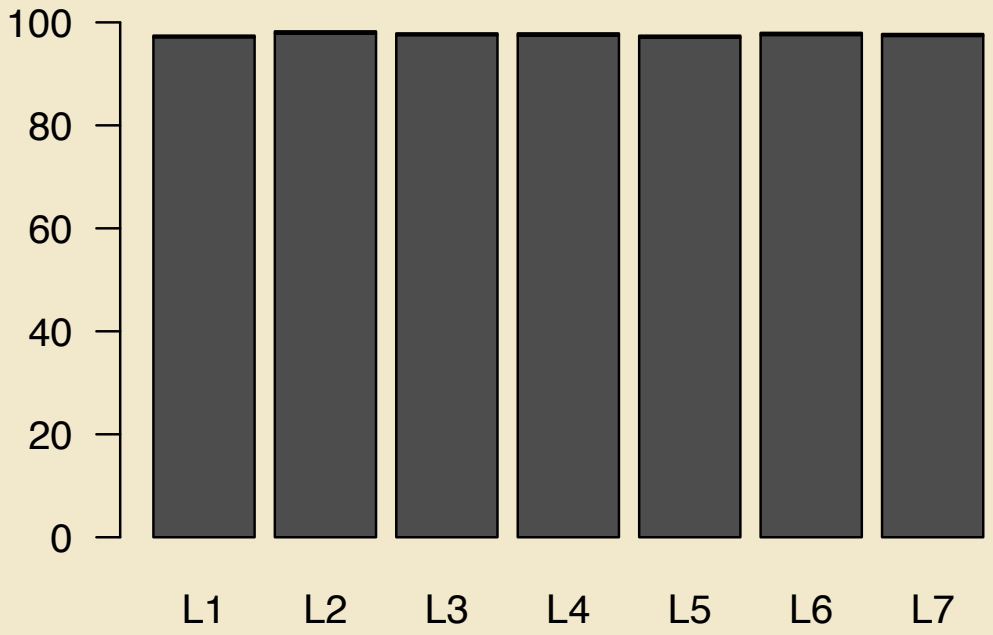

```
dir.create("F_merge/_data", recursive=T)

FW <- list.files("C_Cutadapt/_data",
full.names=T)
RC <- list.files("E_U_revcomp/_data",
full.names=T)

for (i in 1:length(FW)){
system2("cat", paste(FW[i], " ", RC[i], " > F_merge/_data/",
sub("E_U_revcomp/_data/(.*_).*", "\\1", RC[i]), "merged.txt",
sep=""))
}

cat(file="log.txt", append=T, c("\nPROCESSING MODULE:", "F_merge"),
sep="\n")
```

6) Reads lost in primer trimming

- Folder `F_merge` contains now adapter timed reads, in forward orientation
- Having all reads in one direction is very useful! (clustering errors, BOLD ident.)

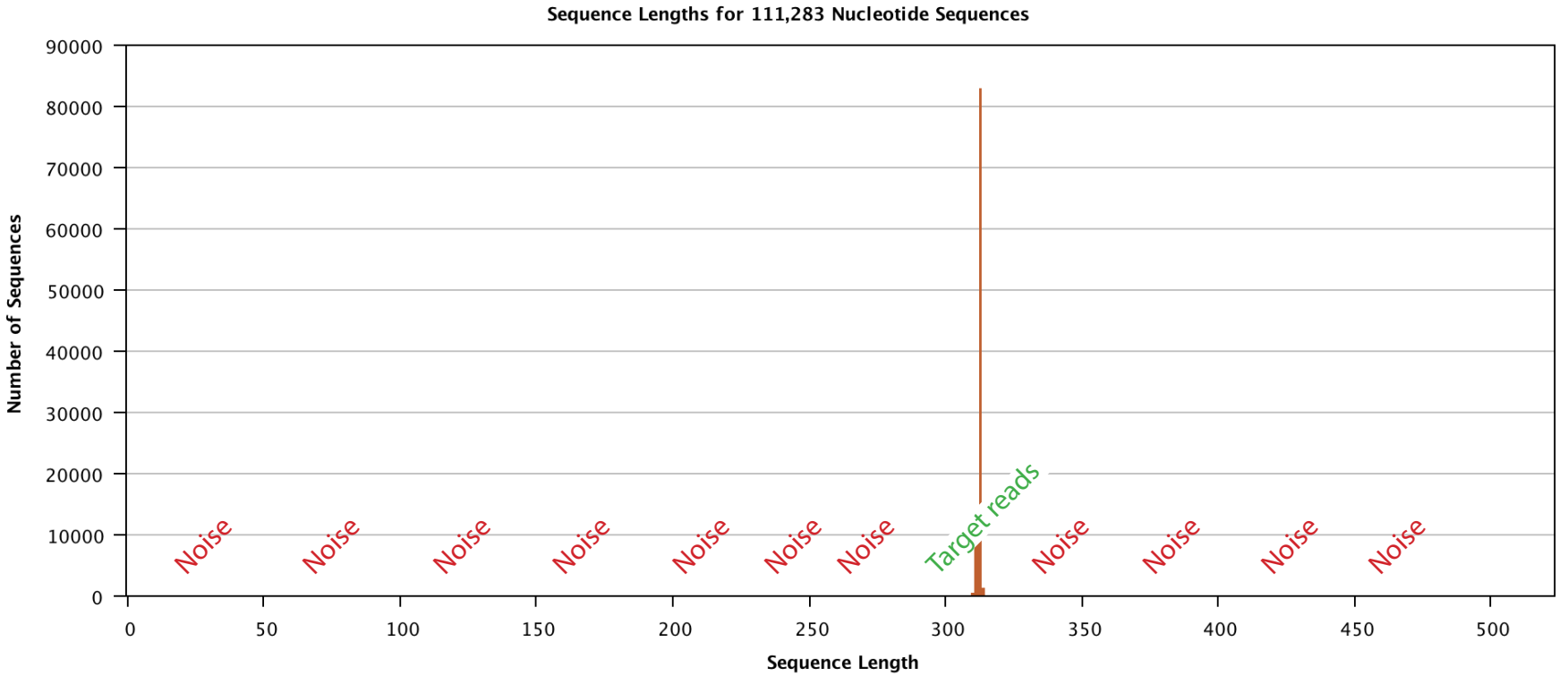
```
C_Cutadapt/_stats/cut_pass.csv + D_Cutadapt/_stats/cut_pass.csv
```

- In this case 0.38% (SD = 0.05%) sequences are discarded



7) Length filtering using Cutadapt

- The trimmed sequences should have a length of 313 bp
- However there can always be unspecific amplicons and artefacts
- Import sample `L1` from `F_merge` into Geneious



7) Length filtering, JAMP command

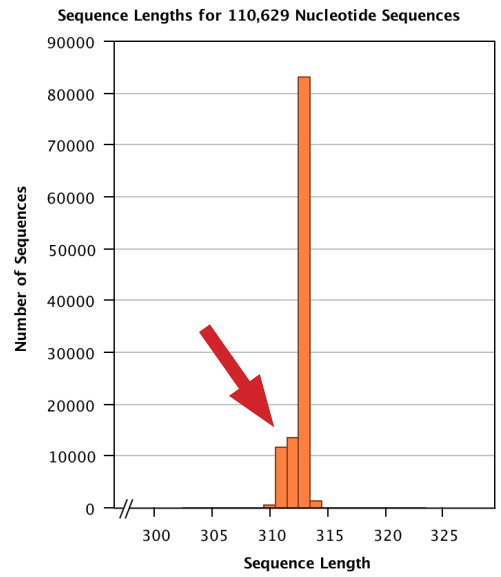
- We expect 313 bp, but also keep reads +/- 10 bp length

```
Minmax(min=(313-10), max=(313+10))
```

- Under the hood

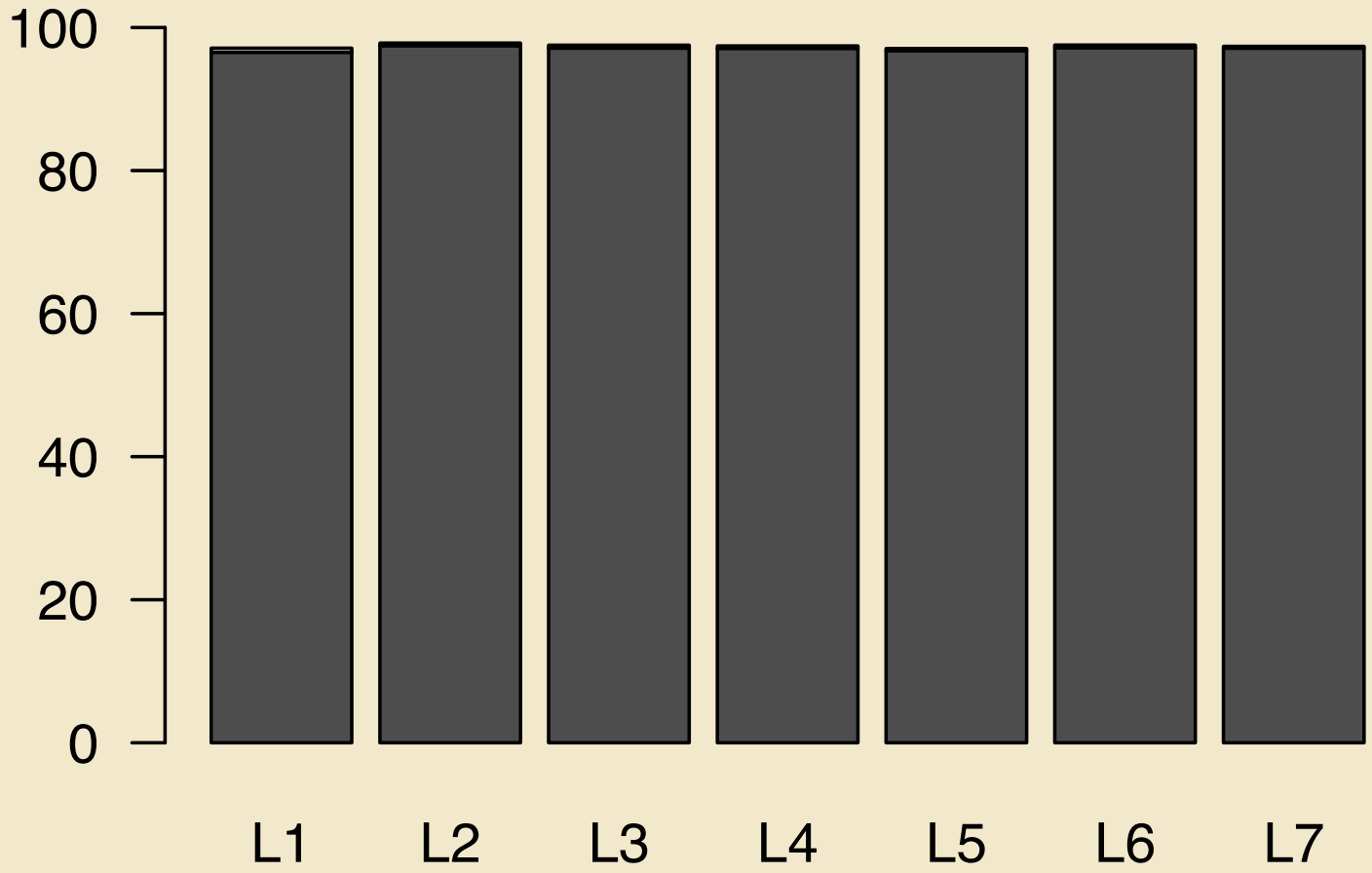
```
cutadapt input_file.txt -o otuput_file.txt -f fasta -m 303 -M 323
```

```
This is cutadapt 1.9 with Python 2.7.10
Command line parameters: ../F_merge/_data/L1_PE_merged.txt -o _data/
L1_PE_merged.txt -f fastq -m 303 -M 323
Trimming 0 adapters with at most 10.0% errors in single-end mode ...
Finished in 0.64 s (6 us/read; 10.45 M reads/minute).

=== Summary ===

Total reads processed:                111,283
Reads with adapters:                        0 (0.0%)
Reads that were too short:                497 (0.4%)
Reads that were too long:                 157 (0.1%)
Reads written (passing filters):      110,629 (99.4%)

Total basepairs processed:    34,731,820 bp
Total written (filtered):     34,588,804 bp (99.6%)
```



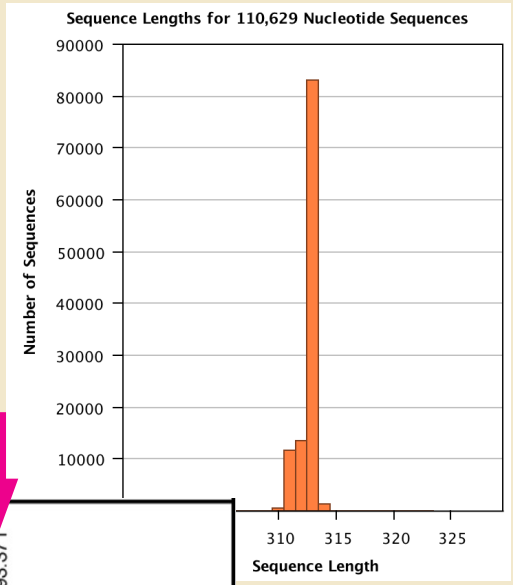
7) Reads lost in length filtering

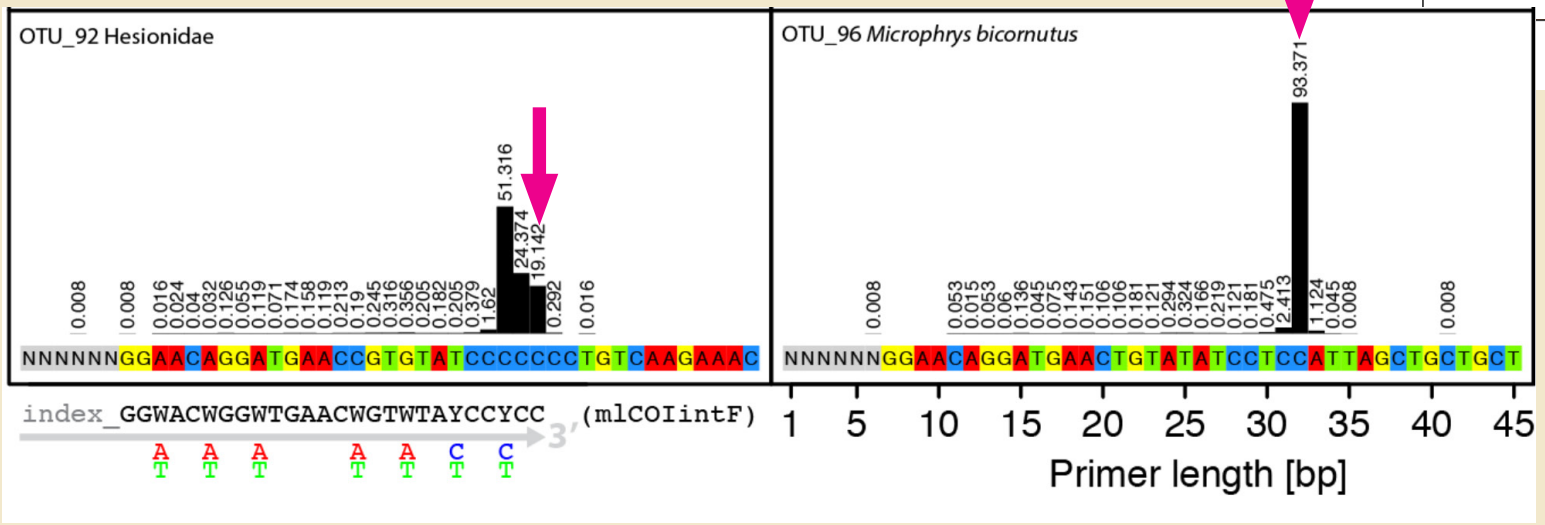
- In this case 0.38% (SD = 0.10%) sequences are discarded



7) Remaining length variation = Primer Slippage

- Highly degenerated primers, in low complexity regions Can slip forward! (mlCOIintF, BF1 & BF2 affected)
- Can be mitigated by e.g. GC clamp, or the low complexity region not extending in amplification direction
- Consider template DNA + flanking regions in primer design (species specific)





8) Quality filtering (max expected errors)

- Do never use average Phred scores!
  = Logarithmic error probabilities!

- Use expected errors instead!
  Edgar & Flyvbjerg (2015), Sequence analysis

  - Phred scores are converted into error probabilities
  - Sum of these probabilities:
    ee = expected errors

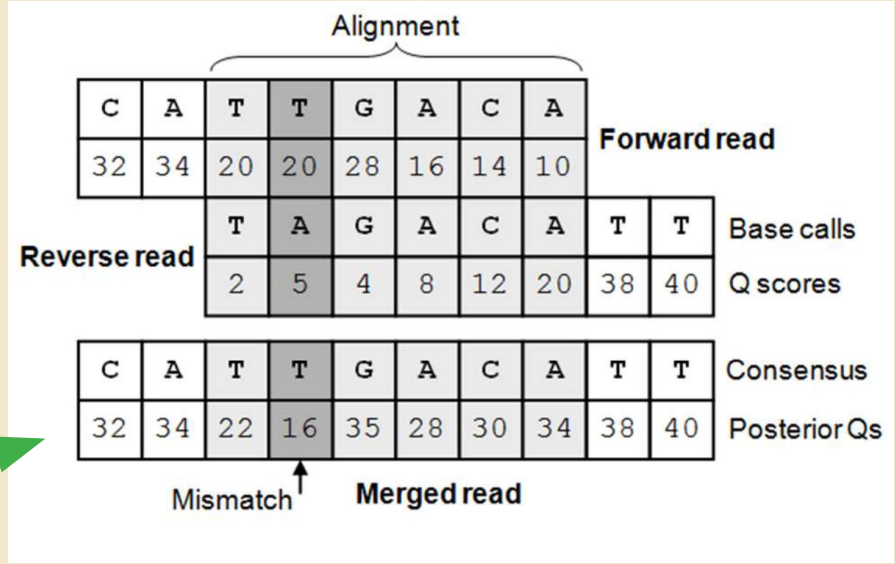
- 150% = 1.5 bases in the sequence are expected to be wrong

- Usearch does adjust Phred errors scores when PE merging!

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

en.wikipedia.org/wiki/Phred_quality_score



Edgar & Flyvbjerg (2015), Sequence analysis

8) Expecter error filtering implementation

- Using the U_max_ee function in R we discard all reads with ee > 1.
- Filse are now saved as fasta, no longer fastq

```
U_max_ee(max_ee=1)
```

- Under the hood

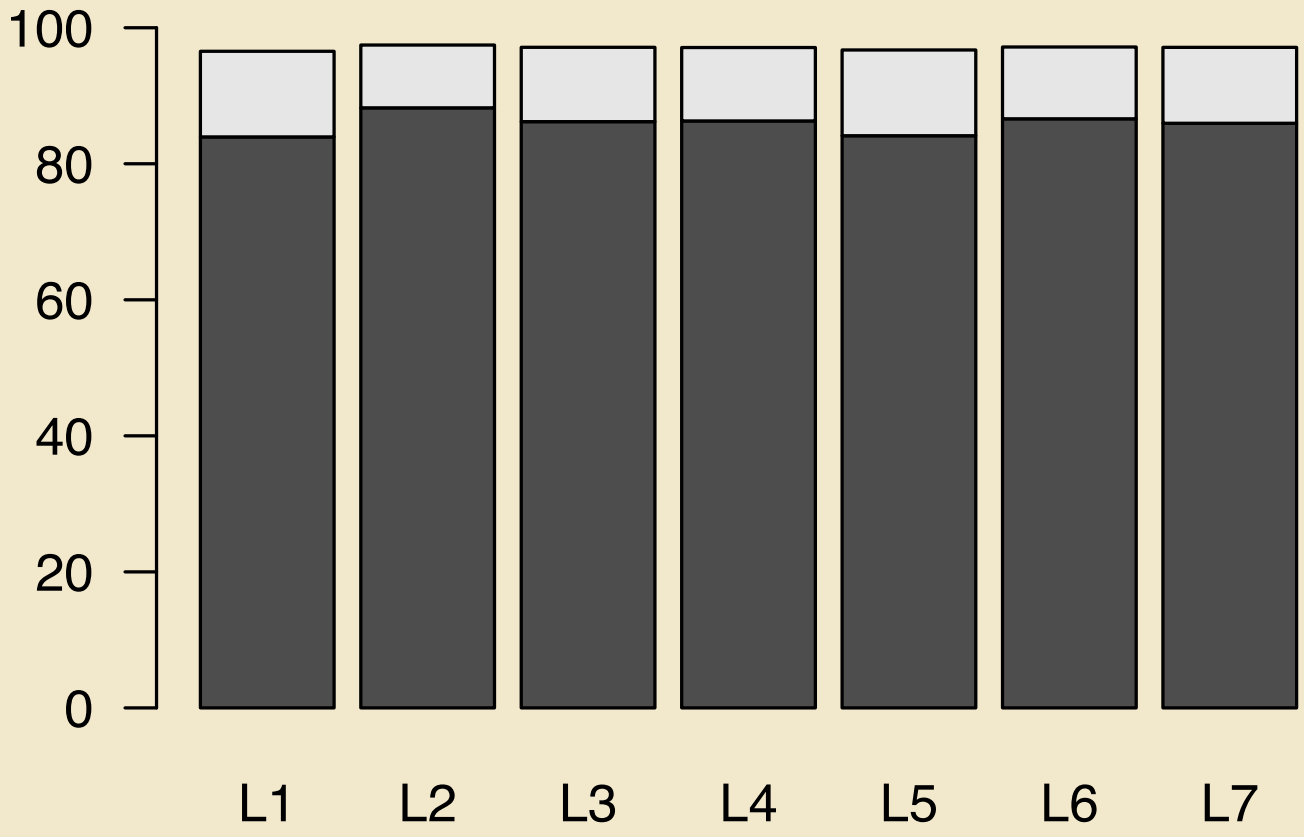
```
usearch -fastq_filter input_file.txt -fastaout output_dile.txt
-fastq_maxee 1 -fastq_qmax 60
```

- Reads passing

```
00:04 2.1Mb   100.0% Filtering, 87.0% passed
    110629  Reads (110.6k)
     14429  Discarded reads with expected errs > 1.00
     96200  Filtered reads (96.2k, 87.0%)
```

8) Reads discarded in expected error filtering

- In this case 11.13% (SD = 1.18%) sequences are discarded



9) Optional: Subsample to same sequencing depth

- In cases of very diferent sequencing depth it can be useful to subsample all samples to the same sequencing depth
- Here, subsampling to 60.000 sequences

```
U_subset(sample_size=60000)
```

- Under the hood

```
Usearch -fastx_subsample input_file.txt -fastaout
output_file.txt -sample_size 60000 -sizein -sizeout
```

```
00:00 1.4Mb     0.1% Counting seqs
00:00 36Mb    100.0% Counting seqs
00:00 36Mb      0.1% Sampling
00:00 36Mb      0.0% Sampling
00:01 36Mb     15.5% Sampling
00:02 36Mb     80.7% Sampling
00:02 36Mb    100.0% Sampling
```

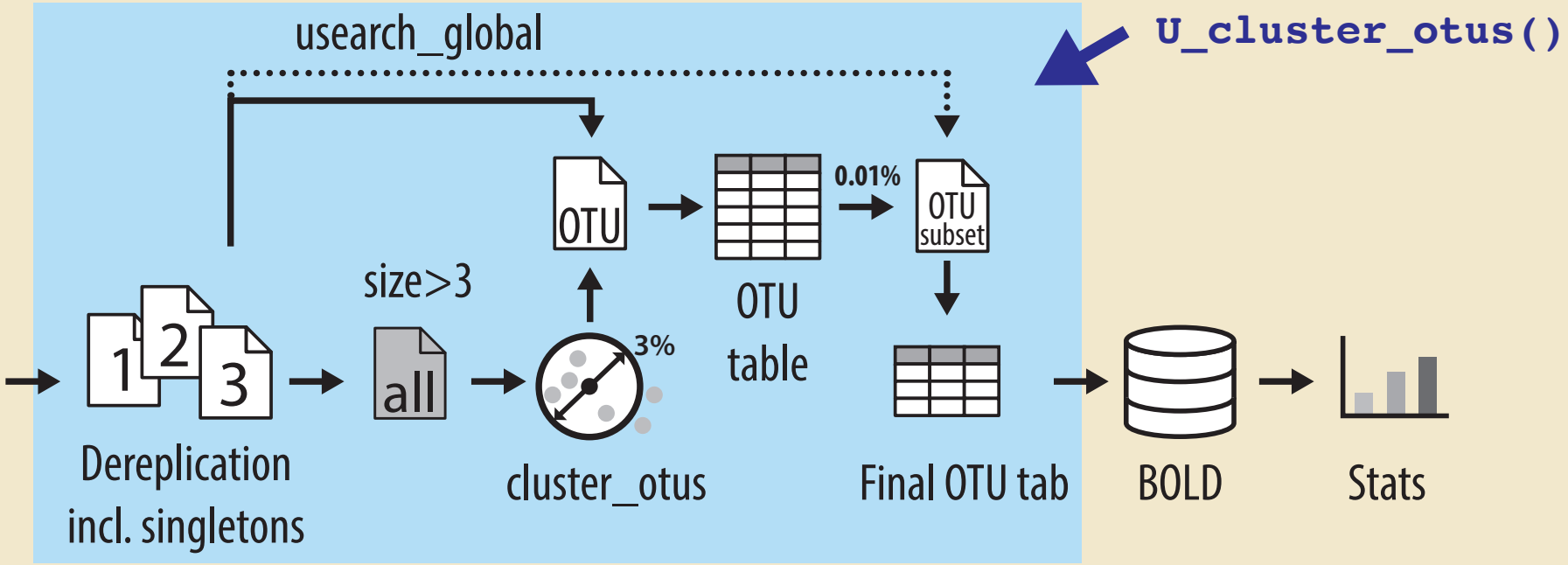
max_ee_stats....

	Sample	Sequ_count
1	L1	96200
2	L2	71139
3	L3	71514
4	L4	62369
5	L5	104808
6	L6	87665
7	L7	91166

10) OTU clustering

- Using the subsampled data (60k)

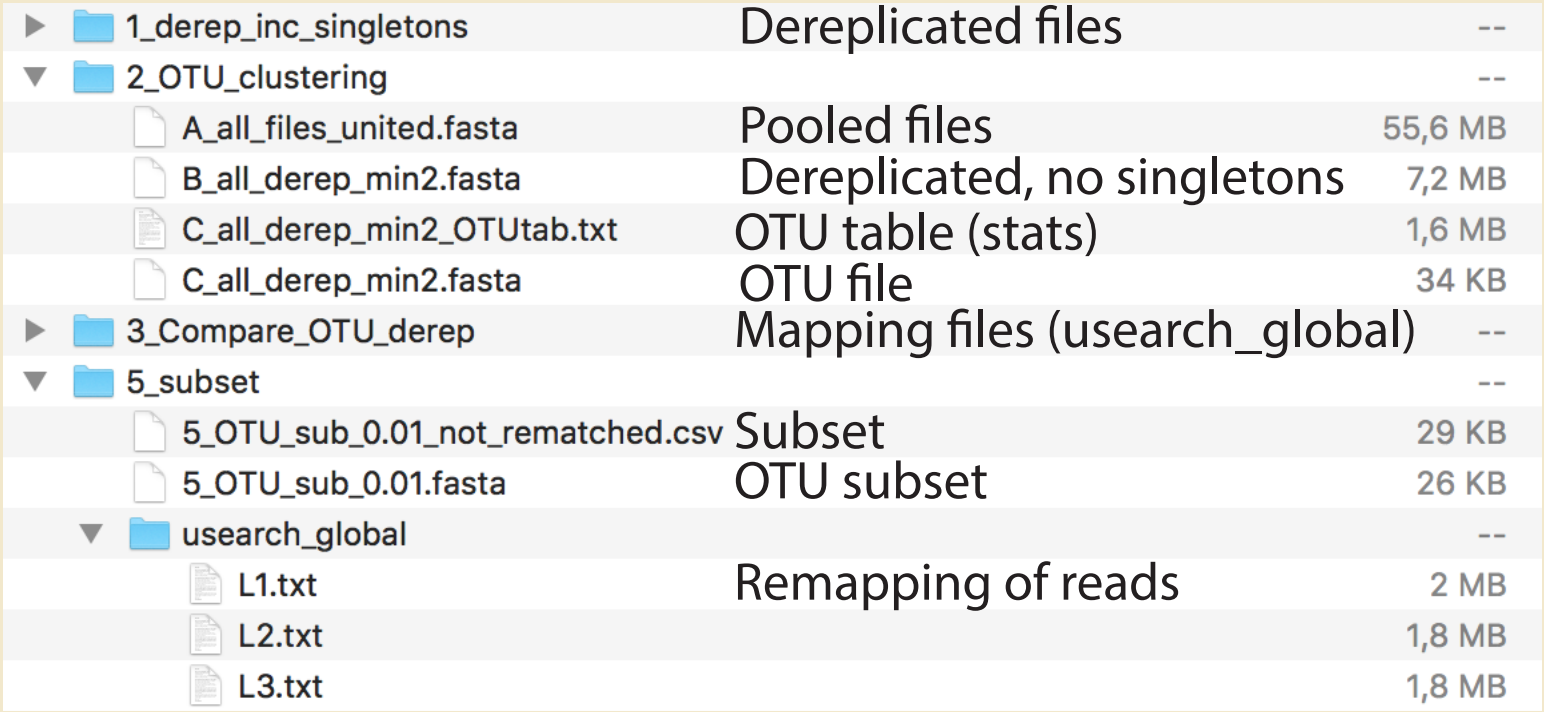
```
U_cluster_otus(filter=0.01)

file.rename("J_U_cluster_otus", "J_U_cluster_otus - 60k")
```



10) OTU clustering - under the hood

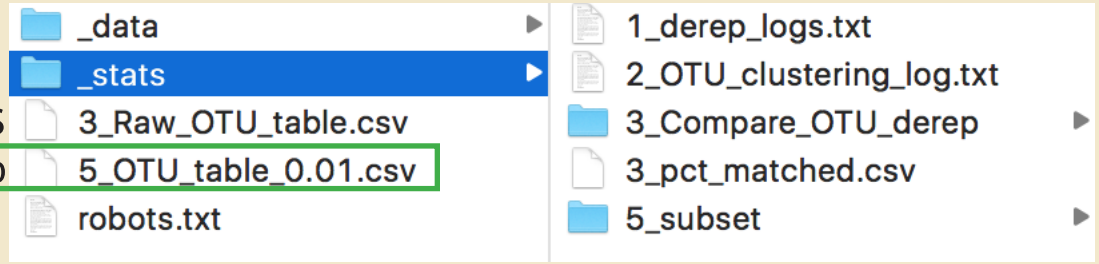
```
usearch -fastx_uniques input_file.txt -fastaout Output_file.txt
-sizeout # Dereplicate!

cat file1.txt file2.txt file3.txt > pooled_files.txt # merge files

vsearch -derep_fulllength pooled_files.txt -output derep_size2.txt
-sizein -sizeout -minuniquesize 2 # dereplicate, discard singletons

usearch -cluster_otus derep_size2.txt -otus OTUs.txt -uparseout
OTU_tab -relabel OTU_ -strand plus # OTU clustering + chimera remov.

usearch -usearch_global file1.txt -db OTUs.txt -strand plus -id 0.97
-blast6out hits_file1.txt -maxhits 1 # mapp each file against OTUs

# Subsetting OTUs below filter=0.01 % abundance
# filterN=1 discard OTUs with are in less than one sample

usearch -usearch_global file1.txt -db OTU_subset.txt -strand plus -id
0.97 -blast6out hits_file1.txt -maxhits 1 # re-map reads against OTUs
```

10) OTU clustering - files generated



Unfiltered OTUs

subset 0.01%



Log files

10) OTU clustering - log.txt file

```
##########
2018-01-25 16:28:01
PROCESSING MODULE:
J_U_cluster_otus
Version v0.1


7 files are dereplicated (incl. singletons!):
1_derep_inc_singletons/L1_PE_derep.fasta
1_derep_inc_singletons/L2_PE_derep.fasta
1_derep_inc_singletons/L3_PE_derep.fasta
1_derep_inc_singletons/L4_PE_derep.fasta
1_derep_inc_singletons/L5_PE_derep.fasta
1_derep_inc_singletons/L6_PE_derep.fasta
1_derep_inc_singletons/L7_PE_derep.fasta


7 dereplicated files where merged (inc singleotns) into file:
"_data/2_OTU_clustering/A_all_files_united.fasta"
Total number of sequences (not dereplicated): 149693

United sequences are dereplicated with minuniquesize = 2 into
a total of 100095 unique sequences.
File prepared for OTU clustering: "B_all_derep_min2.fasta"

Clustering reads from "B_all_derep_min2.fasta
minuniquesize = 2
strand = plus
Chimeras discarded: 78
OTUs written: 106 -> file "C_all_derep_min2.fasta"
```

Chimera removal in cluster_otus better than in stand alone removal

```
Comparing 7 files with dereplicated reads (incl. singletons)
against OTUs "C_all_derep_min2.fasta" using "usearch_global".

L1_PE_derep.fasta - 97.4% reads matched
L2_PE_derep.fasta - 97.6% reads matched
L3_PE_derep.fasta - 97.1% reads matched
L4_PE_derep.fasta - 97.2% reads matched
L5_PE_derep.fasta - 97.5% reads matched
L6_PE_derep.fasta - 97.5% reads matched
L7_PE_derep.fasta - 97.5% reads matched


OTU table generated (including OTU sequences):
3_Raw_OTU_table.csv
Discarding OTUs with below 0.01% abundance across at least 1
out of 7 samples.
Discarded OTUs: 25 out of 106 discarded (23.58%)


Remapping 7 files (incl. singletons) against subsetted OTUs
"_data/5_subset/5_OTU_sub_0.01.fasta" using "usearch_global".

L1_PE_derep.fasta - 97.2% reads matched
L2_PE_derep.fasta - 97.5% reads matched
L3_PE_derep.fasta - 97.0% reads matched
L4_PE_derep.fasta - 97.1% reads matched
L5_PE_derep.fasta - 97.4% reads matched
L6_PE_derep.fasta - 97.4% reads matched
L7_PE_derep.fasta - 97.4% reads matched


Subsetted OTU table generated (0.01% abundance in at least 1
sample): _data/5_subset/
2018-01-25 16:28:46

Module completed!
```

10) OTU clustering - OTU table!

5_OTU_table_0.01.csv Open with LibreOffice

sort	ID	L1	L2	L3	L4	L5	L6	L7	sequ
1	OTU_1	5629	5701	5480	5603	5649	5299	5500	TCTGGCCGGTAATCTAGCCCACGC
2	OTU_2	6903	7476	8099	7262	7569	7473	7082	CCTTGCAAGCAACATTGCTCATGC
3	OTU_3	3523	4504	2627	3341	3000	2557	2857	ATTAGCAGCTGGAATTGCTCACAG
4	OTU_4	2157	2536	2798	2503	2637	2136	2297	TCTTTCTAGTAGAATTGCCCATACC
5	OTU_5	2785	2567	2812	2283	2477	2518	2148	ATTAGCTGCTGCTATTGCTCATGCA
6	OTU_6	2820	374	2565	3389	3405	2969	3242	ACTATCAGGGCCAGTAGCCCACGC
7	OTU_7	2699	2073	1842	2673	2621	2680	2634	ACTTTCAGCAGGTATTGCCCATGC
8	OTU_8	2185	3410	1327	2189	2083	1759	2286	TCTCTCAAGTAACATCGCCCATGC
9	OTU_9	2390	1636	1540	1839	2216	3676	1575	TCTTGCTGGCAATTTAGCCCATGC
								1859	

11) OTU clustering, without subsetting

```
no_subset <- list.files("~/Documents/UNI_und_VORLESUNGEN/14 Guelph/1
TEACHING/2018 metabarcoding course/1 JAMP/H_U_max_ee/_data", full.
names=T)

U_cluster_otus(files= no_subset, filter=0.01)
```

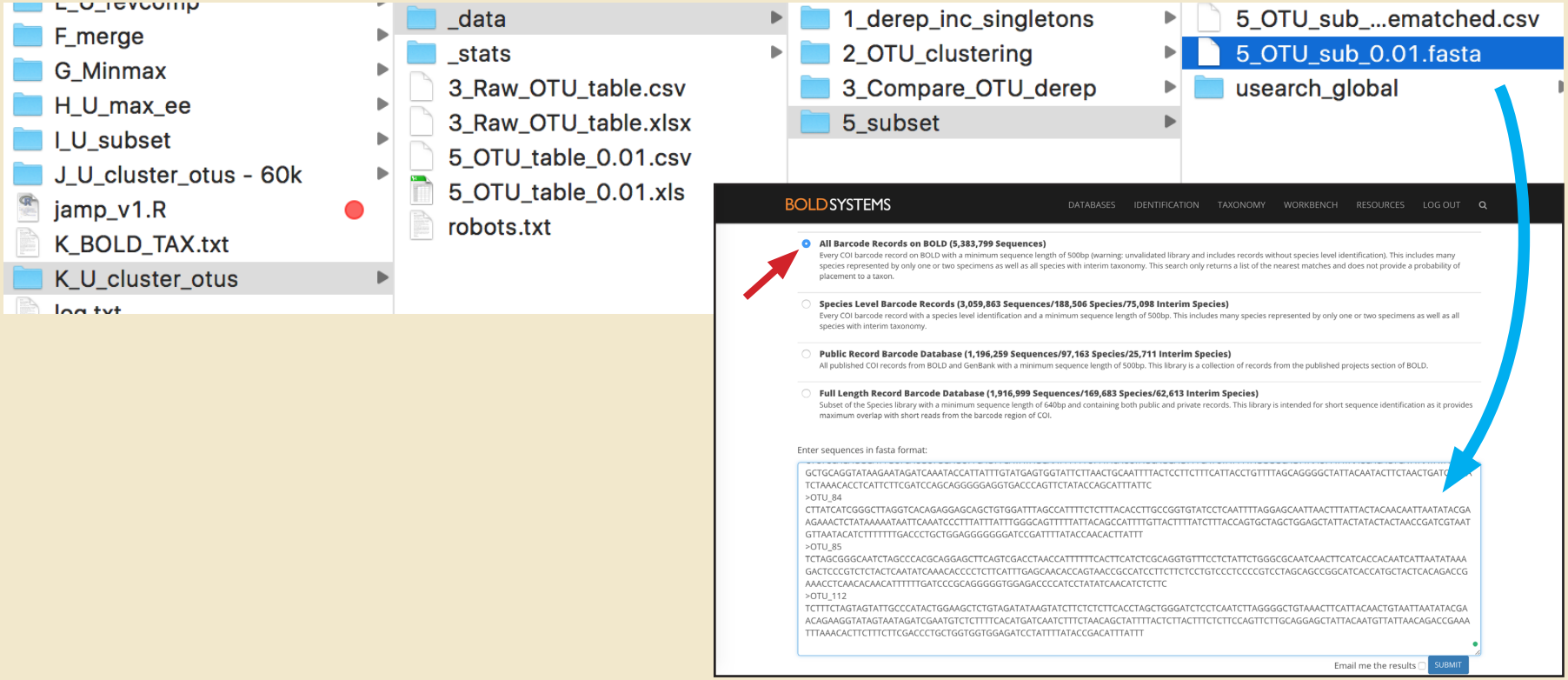
- Use relative abundance, to compare samples, due to (R script)
- OTU table number of reads mappes, also below `filter=0.01`

OTU numbers	filter	final
• Subsampling	106	81
• No subsampling	112	78

- Both approaches walid

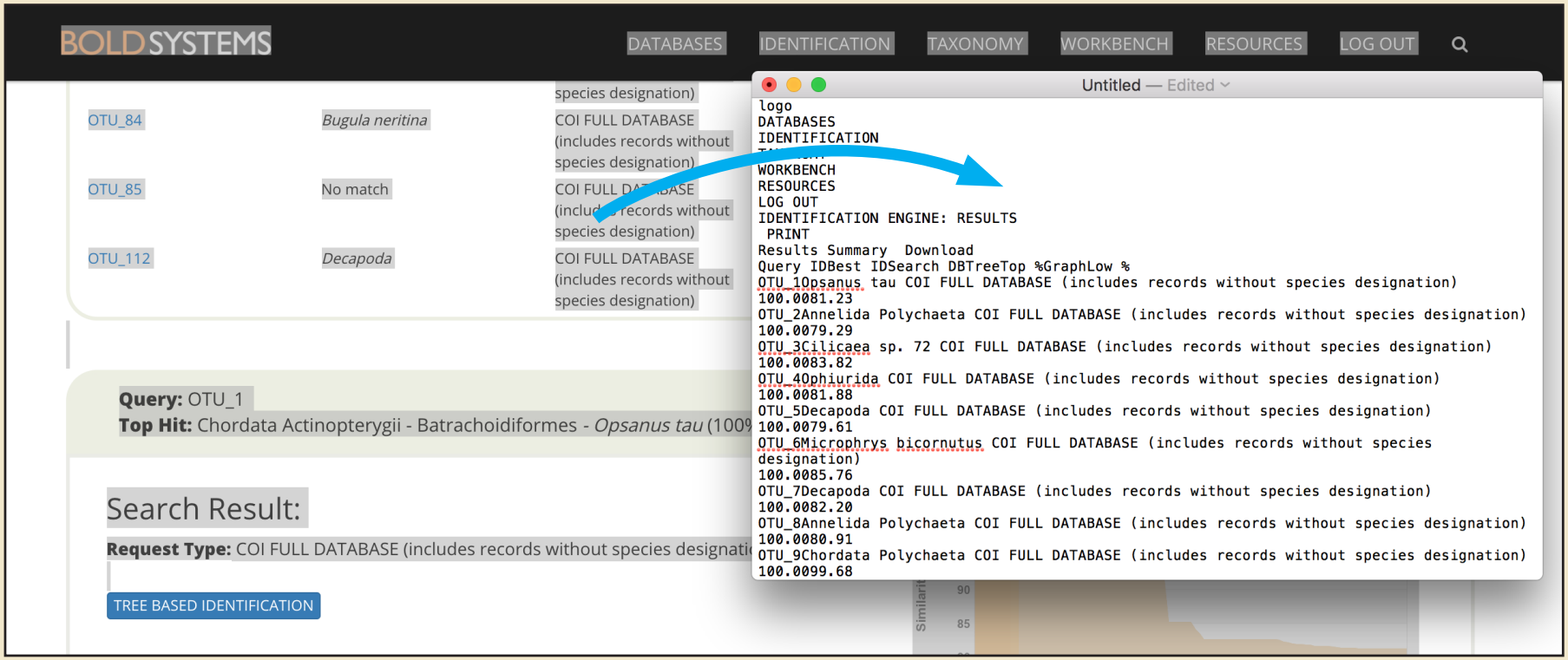
12) OTU identification with the BOLD website

- Do not use the BOLD API or R package for sequence identification!
  Only querying against the bold website will give access to private data!
- 500 sequences at a time, registration required



12) OTU identification with the BOLD website

- Using Google chrome: Wait till the full website is loaded (scroll down)
- Select all text, copy paste into txt file! `cmd + a`, `cmd + c`, `cmd + v`




```
Bold_web_hack(file="K_BOLD_TAX.txt")
```

12) OTU identification with the BOLD website

- Merge the OTU table with the taxa table in Excel
- Minimum match in % `MM=c(0.98, 0.95, 0.90, 0.85)`
  (species, genus, family and order)
- Takes the most common name as hit.
- Percent match not the best approach, but easy ;)

K_BOLD_TAX_taxonomy.csv — Open with Libre

OTU_ID	Phylum	Class	Order	Family	Genus	Species	Similarity	Status
OTU_1	Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Private
OTU_2	Annelida	Polychaeta	Phyllodocida	Nereididae	Ceratonereis	sp.	98.61	Private
OTU_3	Arthropoda	Malacostraca	Isopoda	Sphaeromatidae	Cilicaea	sp. 72	100	Published
OTU_4	Echinodermata	Ophiuroidea	Ophiurida	NA	NA	NA	100	Published
OTU_5	Arthropoda	Malacostraca	Decapoda	Hippolytidae	NA	NA	100	Published
OTU_6	Arthropoda	Malacostraca	Decapoda	Majidae	Microphrys	bicornutus	100	Published
OTU_7	Arthropoda	Malacostraca	Decapoda	Alpheidae	Alpheus	NA	100	Published
OTU_8	Annelida	Polychaeta	NA	NA	NA	NA	100	Private
OTU_9	Chordata	Actinopterygii	Gobiiformes	Gobiidae	Coryphopterus	glaucofraenum	100	Published
OTU_10	Arthropoda	Malacostraca	Decapoda	Panopeidae	Panopeus	occidentalis	100	Published
OTU_11	Annelida	Polychaeta	Phyllodocida	Syllidae	NA	NA	100	Private
OTU_12	Annelida	Polychaeta	Phyllodocida	Polynoidae	Lepidonotus	humilis	100	Early-Release
OTU_13	Arthropoda	Malacostraca	Decapoda	Panopeidae	Dyspanopeus	sayi	100	Published
OTU_14	Chordata	Actinopterygii	Blenniiformes	Blenniidae	Hypleurochilus	geminatus	99.68	Early-Release
OTU_15	Annelida	Polychaeta	Phyllodocida	Hesionidae	Hesione	NA	97.89	Private
		Gastropoda	Sacoglossa					

OTU_1.csv — Open with LibreC

Phylum	Class	Order	Family	Genus	Species	Similarity	Status
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Private
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Private
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Early-Release
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	100	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus		99.68	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	99.68	Private
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	99.68	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	tau	99.68	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	pardus	98.06	Private
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	pardus	98.06	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	beta	93.47	Published
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	pardus	93.2	Private
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	beta	92.88	Early-Release
Chordata	Actinopterygii	Batrachoidiformes	Batrachoididae	Opsanus	beta	[illegible]	Early-Release
	Actinopterygii	Bat[illegible]		Opsanus			

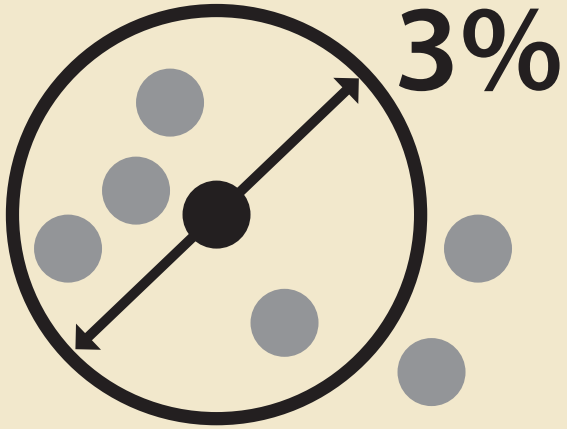
12) Set reads below 0.01% to 0

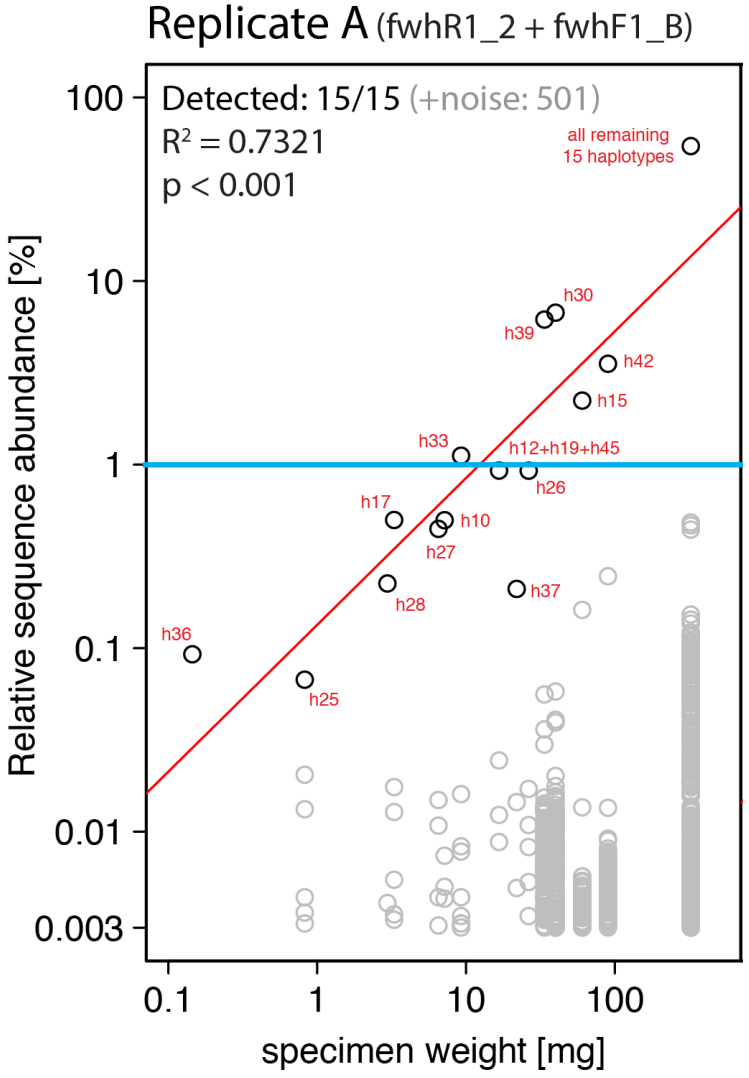
- Not done in JAMP, subsetting parameters can depend on replicates etc

```
# setting read counts below 0.01% to 0
data <- read.csv("K_U_cluster_otus/5_OTU_table_0.01.csv", stringsAsFactors=F)
colSum <- colSums(data[,3:9])
i <- 3
for(i in 3:9){
keep <- data[-nrow(data),i]/colSum[i-2]*100>=0.01 # find values blow 0.01
data[nrow(data),i] <- data[nrow(data),i]+sum(data[,i][!keep]) # below 0.01
data[,i][!keep] <- 0 # set lavues to 0
}
write.csv(data, "K_U_cluster_otus/data_cleaned.csv", row.names=F)

data2 <- data
#relative abundance
for(i in 3:9){
data2[,i] <- round(data2[,i]/colSum[i-2]*100, 4) #converte data into relative
 abundance
}
write.csv(data2, "K_U_cluster_otus/data_cleaned_rel.csv", row.names=F)
```

Bonus: Metabarcoding on haplotype level

- What about genetic diversity?

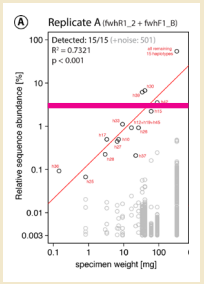


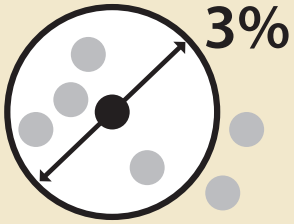


Elbrecht et al. 2018, *PeerJ Preprint*

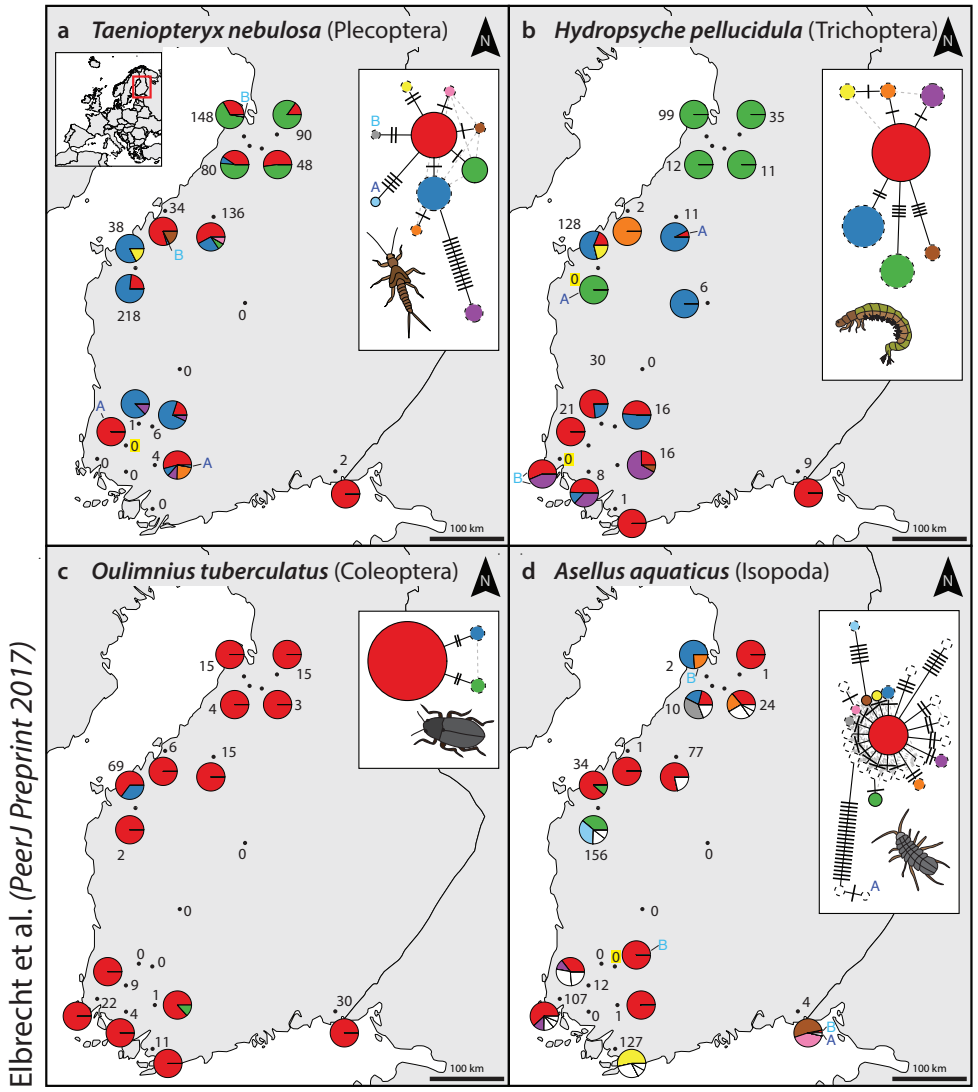
Rough estimate of Genetic connectivity

- Works with most high quality metabarcoding data
- Denoising instead of clustering



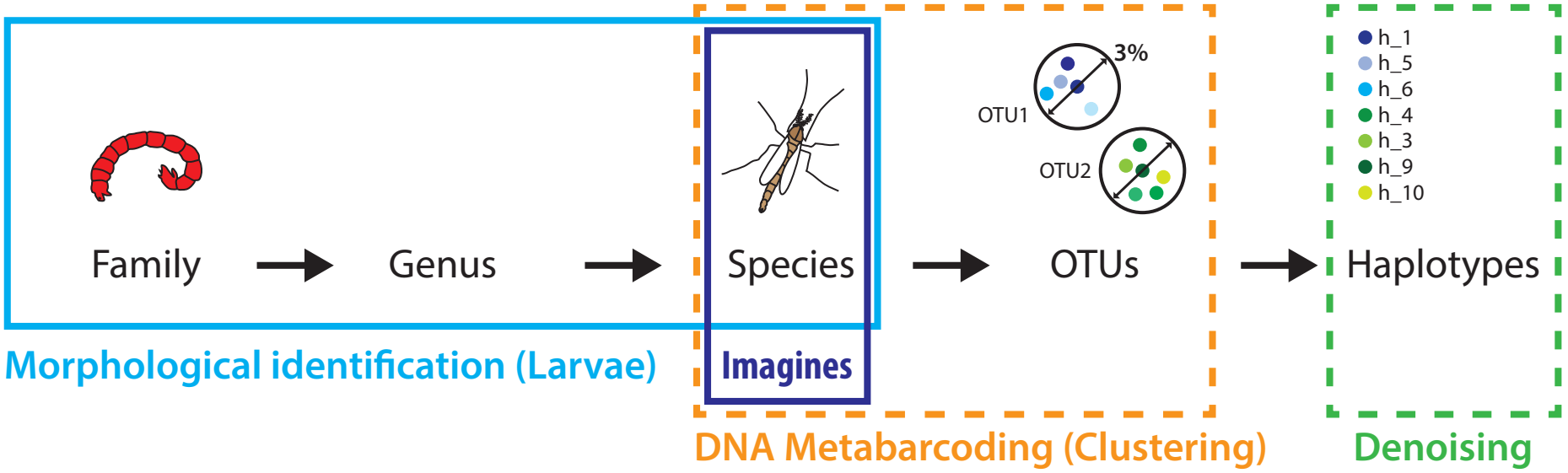


- Genetic fingerprints on ecosystem level!
- Genetic diversity and connectivity of populations



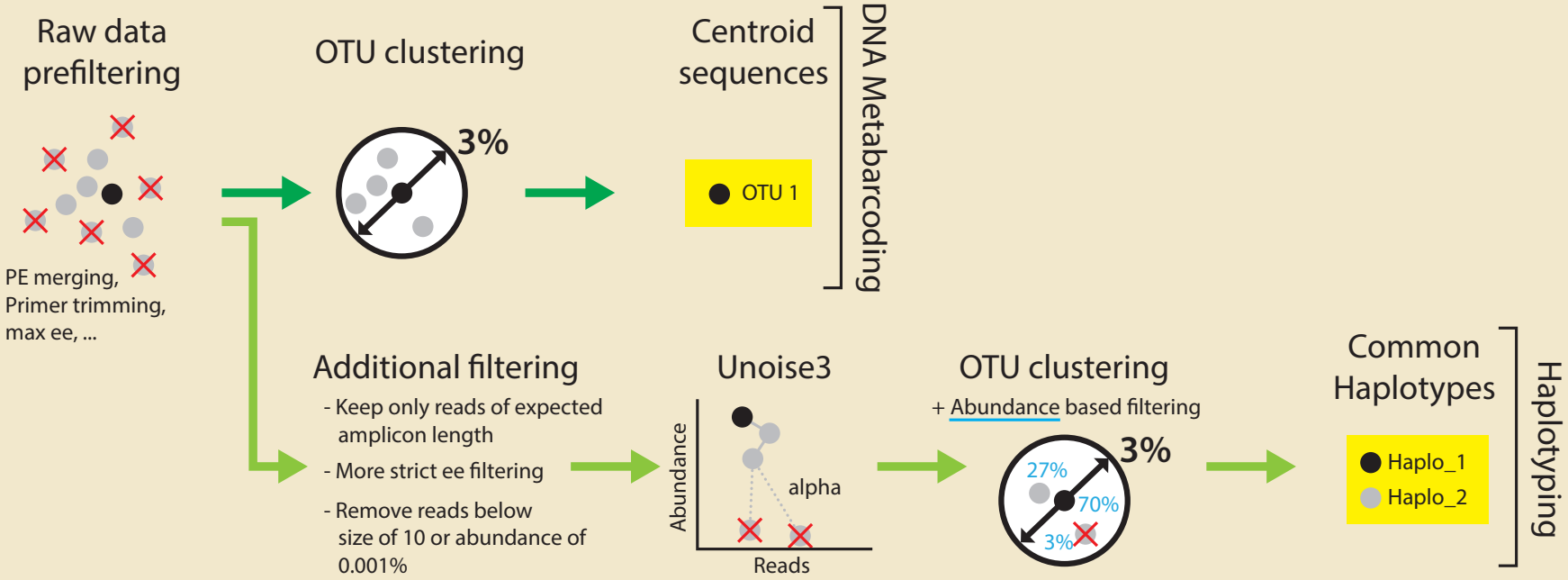
Taxonomic resolution

Taxonomic resolution with morphologycal and DNA based identification methods



Denoising with JAMP

- The leray data set is a bit to low on sequencing depth



- Documentation: https://github.com/VascoElbrecht/JAMP/wiki/3)-Denoising-quick-guide!

Data pre filtering

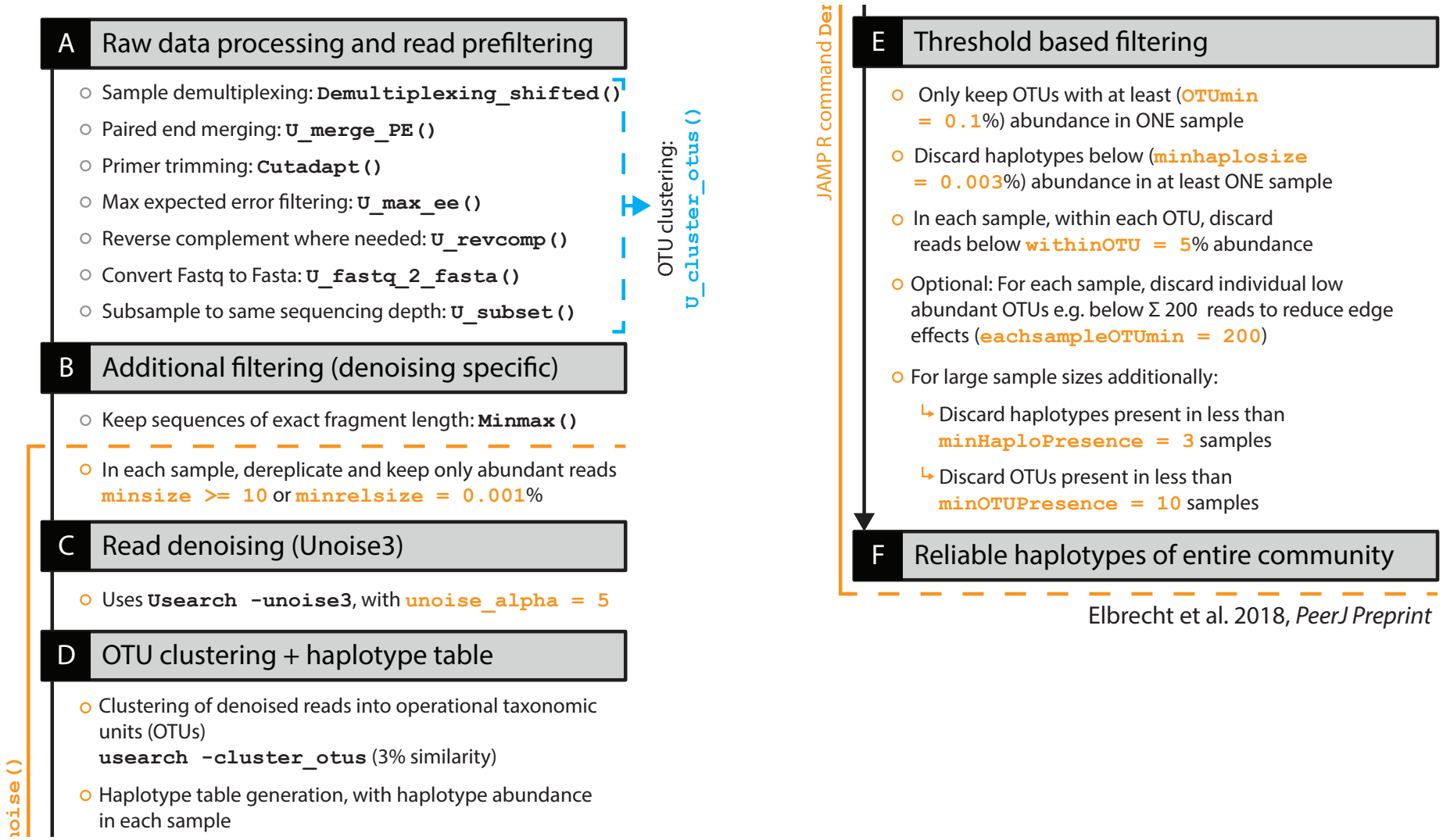
- The Leray data set is a bit to low on sequencing depth (not ideal)
- Only keep sequences of 313 bp length
- Strict EE filtering

```
# from merged data (Include full path):
no_subset <- list.files("~/Documents/UNI_und_VORLESUNGEN/14 Guelph/1
TEACHING/2018 metabarcoding course/1 JAMP/F_merge/_data", full.
names=T)

Minmax(file=no_subset, min=313, max=313)

U_max_ee(max_ee=0.2)
```

- Denoising

```
Denoise(files="latest",  strategy="unoise", unoise_alpha=5,
minsize=10, minrelsize=0.0001, OTUmin=0.01, minhaplosize=0.003,
withinOTU=5, eachsampleOTUmin=NULL, minHaploPresence=1,
minOTUPresence=1, renameSamples="(.*_.*)_cut.*")
```

Denoising pipeline - settings



Elbrecht et al. 2018, *PeerJ Preprint*

There will be false positives and negatives

- Haplotyping only works on high abundant OTUs (sufficient sequencing depth)
- Within each OTU, only high abundant haplotypes are retained
- Leray dataset, one specimen per species,
  but sometimes more haplotypes per OTU
  - Bycatch (other organisms, gut content)
  - Within species variability
  - Sequencing errors (Main source of false positives)