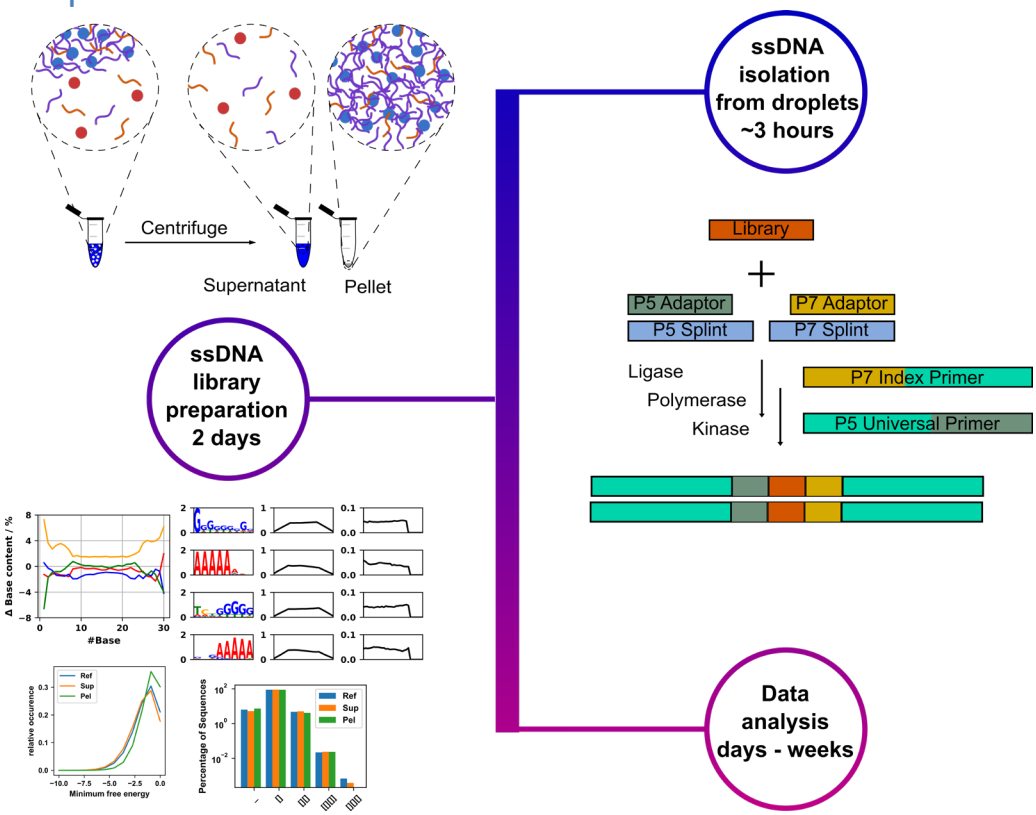
Recovery and deep sequencing of short ssDNA pools from transient, fuel-driven coacervate droplets.

[Anna-Lena Holtmannspoetter,[2,3] Corbin Machatzke,[1,3,4,*] Job Boekhoven[2] and Hannes Mutschler[1,5,**]]**
[1]TU Dortmund University, Otto-Hahn-Strasse 4a, 44227 Dortmund, Germany.
[2]Department of Bioscience, School of Natural Sciences, Technical University of Munich, Lichtenbergstrasse 4, 85748 Garching, Germany.
[3]Authors contributed equally
[4]Technical Contact
[5]Lead contact
*Correspondence: [Technical contact email]
**Correspondence: [Corresponding/Lead Contact email]

Summary

Complex coacervates droplets are synthetic cell models that can sequester and concentrate nucleic acids. This protocol describes time-resolved harvesting of transient, metabolically active complex coacervates, extraction of single-stranded DNA pools, sequence-independent adapter ligation and library construction for next-generation sequencing. We present a analysis pipeline to assess enrichment dynamics and sequence distributions. The protocol is adaptable to diverse nucleic acid containing droplet systems.

Graphical abstract



Before you begin

Rational fore method development.

We used this protocol to compare single-stranded DNA (ssDNA) that partitions into the polymer-rich phase of a transient, fuel-driven complex coacervate system with ssDNA remaining in the polymer-poor phase. Because sequence determinants of partitioning are difficult to predict, this protocol enables experimental determination of enrichment and depletion rules and dynamics.[1–3]

The workflow of this protocol is compatible with other synthetic and protocellular model systems if ssDNA recovery and library preparation are adjusted accordingly. The protocol may also be adapted to RNA-based systems, if the extraction is adjusted and strand-aware library steps follow after cDNA synthesis using reverse transcription.

After ssDNA recovery, we present a strand-specific, sequence-agnostic library preparation method for deep sequencing. Finally, we provide a modular analysis pipeline tailored to short (i.e. prebiotically credible) sequences [4–7], quantifying nucleotide composition, predicted intramolecular structure, and motif enrichment.

Innovation

We enable time-resolved recovery and deep sequencing of ssDNA from transient, fuel-driven complex coacervate droplets.[8,9] Because these droplets have limited lifetimes, we implement a rapid quenching and harvesting protocol to capture polymer-rich versus polymer-poor sequence pools with minimal bias. For library construction, we adapt the single reaction single-stranded library (SRSLY) protocol[10] to unknown and low-input amounts of input ssDNA to increase yields while preserving strand orientation and minimizing sequence bias.

On the computational side, we provide a modular analysis pipeline tailored to short oligonucleotides typical of synthetic coacervate droplets (~10–100 nt). Modules report quality metrics, nucleotide composition, intramolecular folding, and motif discovery. The pipeline promotes standardization across laboratories working on synthetic / protocell systems and can be extended to alternative nucleic acid backbones such as RNA with minimal changes to recovery and library preparation.

Taken together, the protocol delivers a reproducible end-to-end workflow for discovering and analyzing sequence determinants of partitioning into static and metabolically active droplet systems and related compartments.

Buffer Preparation

Timing: 30 minutes – 1 hour

1. Prepare 500 mM MES pH 5.3 stock solution
   Dissolve 53.3 g MES monohydrate in 400 mL nuclease free ddH_2O. Adjust the pH to 5.3 using HCl, then top up to 500 ml and mix thoroughly before use.

2. Prepare 41 mM polyU (charge concentration) stock solution
   Dissolve 12.55 mg polyuridylic acid potassium salt in 1 mL nuclease free ddH_2O.

3. Prepare 155 mM RG_3 peptide stock solution
   a. Dissolve 500 mg Ac-FRGRGRGD in 1 mL nuclease free ddH_2O
   b. Adjust pH with 1 M NaOH to pH 5.3 (~475 µL)
   c. Add nuclease free ddH_2O until total volume is equal to 2.475 mL

   CRITICAL: Do not prepare a stock solution with a higher peptide concentration! The peptide will precipitate over time if the concentration is too high.
   CRITICAL: Immediately adjust pH of peptide solution.
   CRITICAL: Always verify the quality of custom peptides. Confirm identity by mass spectrometry or demonstrate correct formation of complex coacervates under the conditions described in the Detailed Protocol, with no significant precipitation.

4. Prepare 5 M NaCl stock solution
   Dissolve 29.22 g NaCl in 70 mL nuclease free ddH_2O. Adjust volume to 100 mL after complete dissolution.

5. Prepare 400 mM EDC stock solution
   Dissolve 19.2 mg EDC in 230 µL nuclease free ddH_2O

CRITICAL: Always prepare the EDC solution fresh. Ideally, store EDC at -20 °C, under an inert gas.

6. Prepare 50% w/v PEG-8000 stock solution
7. *Dissolve 20 g of PEG-8000 in 23 ml of nuclease-free ddH_2O. If not all of the PEG-8000 dissolves under constant mixing and slight heat, carefully add more nuclease-free ddH_2O until it is fully dissolved and the total volume is 40 mL.*

Key resources table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Chemicals, peptides, and recombinant proteins		
RG_3 peptide (Ac-FRGRGRGD), >98%	Caslo	N/A
MES monohydrate	AppliChem	Cat# A1074,0500
Polyuridylic acid potassium salt	Sigma-Aldrich	Cat# P9528-25MG
Sodium Chloride	Sigma-Aldrich	Cat# S9888-2.5KG-M
1-ethyl-3-(3-dimethylaminopropyl) carbodiimide (EDC)	Sigma-Aldrich	Cat# E6383-5G
RNase Cocktail	Invitrogen	Cat# AM2286
PEG-8000	Carl Roth	Cat# 0263.1
T4 DNA ligase	New England Biolabs	Cat# M0202S
T4 Polynucleotide Kinase (PNK)	New England Biolabs	Cat# M0201S
NEBNext Ultra II Q5 Master Mix	New England Biolabs	Cat# M0544L
Critical commercial assays		
NEBNext Multiplex Oligos for Illumina (Index Primer Set)	New England Biolabs	Eg. E7500L or E7710L
Monarch Spin PCR & DNA cleanup kit	New England Biolabs	T1130L
NucleoMag NGS Clean-up and Size Selection	Macherey-Nagel	REF 744970.5
PhiX Control v3	Illumina	15017666
Oligonucleotides		
N30 library: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN	Integrated DNA Technologies	N/A
SRSLY P5 adapter: /5AmMC12/AC ACT CTT TCC CTA CAC GAC GCT CTT CCG ATC T	Integrated DNA Technologies	N/A
SRSLY P5 splint: /5AmMC6/NN NNN NNA GAT CGG AAG AGC GTC GTG TAG GGA AAG AGT GT/3AmMO/	Integrated DNA Technologies	N/A
SRSLY P7 adapter: /5Phos/AG ATC GGA AGA GCA CAC GTC TGA ACT CCA GTC A/3ddC/	Integrated DNA Technologies	N/A
SRSLY P7 splint: /5AmMC12/GT GAC TGG AGT TCA GAC GTG TGC TCT TCC GAT CTN NNN NNN /3AmMO/	Integrated DNA Technologies	N/A
SRSLY P5 adapter Cy5 tagged (for troubleshooting): /5Cy5/ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT	Integrated DNA Technologies	N/A
SRSLY P7 adapter FAM tagged (for troubleshooting): /5Phos/AG ATC GGA AGA GCA CAC GTC TGA ACT CCA GTC A/36-FAM/	Integrated DNA Technologies	N/A
Tagged RNA: /5Cy5/rArArArArCrCrArGrUrC	Integrated DNA Technologies	N/A

Software and algorithms		
Illumina Sequencing Analysis Viewer 2.4.7	Illumina	https://support.illumina.com/sequencing/sequencing_software/sequencing_analysis_viewer_sav.html
FastQC 0.12.1		https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
MultiQC 1.15		https://github.com/MultiQC/MultiQC
Cutadapt 4.3		https://cutadapt.readthedocs.io/en/stable/
Seqkit 2.4.0		https://bioinf.shenwei.me/seqkit/
Vienna RNA package, RNAfold 2.4.17		https://www.tbi.univie.ac.at/RNA/
RNAshapes 3.4		https://bibiserv.cebitec.uni-bielefeld.de/rnashapes
Python 3.8.8		https://www.python.org/
Meme Suite, streme 5.5.7		https://meme-suite.org/meme/
Other		
4150 TapeStation	Agilent	https://www.agilent.com/en/product/automated-electrophoresis/tapestation-systems/tapestation-instruments/4150-tapestation-system-297322
Illumina iSeq 100	Illumina	https://www.illumina.com/systems/sequencing-platforms/iseq.html
ProFlex PCR Systems or equivalent thermocycler	Thermo Fisher Scientific	Cat# 4483636
Micro Star 17R or equivalent centrifuge	VWR	Cat# 521-1647
DynaMag-2 or equivalent magnetic rack	Thermo Fisher Scientific	Cat# 12321D

Step-by-step method details

Droplet preparation and DNA recovery

Timing: 3 hours

First, we describe how to prepare fuel dependent complex coacervates in the presence of an ssDNA library. Next, we describe how to separate the phases and recover the ssDNA for downstream analysis.

1. Prepare Complex Coacervates and separate phases
    A. Thaw ssDNA library (and tagged RNA, if used) on ice.
    B. Prepare Complex Coacervates mix (900 µL)

Reagent	[stock]	[final]	V / µL
MES pH 5.3	500 mM	200 mM	400
RG3 peptide	155 mM	23 mM	150
polyU	41 mM	4.1 mM	100
N30 library	3 µg/µL	6 µg	3
Tagged RNA (optional)	100 µM	1 µM	10
ddH_2O			237 or 247

Note: Adding tagged RNA makes it easier to detect the pellet after centrifugation. This is not strictly necessary but is advised for the first trials. The sequence of the RNA is non-critical, as it will be removed by the RNase cocktail.

C. Add 100 µL 400 mM EDC solution to the complex coacervates master mix. Mix thoroughly by pipetting up and down.

Note: The solution should turn turbid within seconds of mixing the sample, which indicates coacervate formation. If the solution does not turn turbid, refer to Problem 1.

CRITICAL: From the addition of EDC until the supernatant is removed, it is imperative to follow all incubation times precisely and work swiftly. The fuel-dependent system is extremely short-lived

D. Incubate for 1 minute at room temperature (25 °C).
E. Spin for 3 minutes at 16.000 g in centrifuge set to 25 °C.

CRITICAL: The temperature should not exceed 30 °C during centrifugation. The lifetime of droplets is strongly affected by temperature. Lower temperatures will greatly extend lifetimes whereas higher temperatures may decrease them drastically.

F. After centrifugation, immediately remove all supernatant very thoroughly and store it for downstream analysis.

Note: Use a P1000 to remove most of the bulk of the supernatant without disturbing the pellet. Then, using a P10, remove as much liquid as possible from the pellet. Discard these small volumes.

G. Resuspend the pellet in 10 µL 5M NaCl.
H. After resuspension, solution should be clear. Add 40 µL ddH_2O.
I. Use the Monarch Spin PCR & DNA cleanup kit with the "small oligo" protocol per manufacturer's instructions to clean up 50 µL of pellet and 50 µL of supernatant:
    i. Add 100 µL Buffer BZ to solution.
    ii. Add 300 µL Isopropanol. Mix well by pipetting up and down. Do not vortex.
    iii. Transfer 450 µL solution onto a spin column and spin for 1 minute at 16,000 g. Discard flow-through.
    iv. Add 500 µL Buffer WZ to the spin column and spin for 1 minute at 16,000 g. Discard flow-through.
    v. Repeat step iv.
    vi. Transfer spin column to a fresh 1.5 mL sample tube. Spin empty for 1 minute at 16.000 g to remove residual buffer.
    vii. Transfer spin column to a fresh sample tube. Add 10 µL ddH_2O to the center of the matrix to elute DNA. Spin for 1 minute at 16,000 g, then discard spin column.

2. Digest polyU RNA from both samples.
    A. Add 1 µL RNase cocktail to both samples

    Note: Always keep enzyme mixes in -20 °C cooling blocks

    B. Incubate the samples in a thermocycler for 2 hours at 37 °C, followed by 10 minutes at 65 °C with the lid set to 105 °C.
    C. You can use 2 µL of the reaction mix to verify successful DNA isolation on a urea-PAGE gel (20% 19:1 acrylamide:bisacrylamide)

Note: 2 µL of the recovered, RNase digested ssDNA can be used for analysis via urea-PAGE gel (20% 19:1 acrylamide:bisacrylamide). If low yields are observed in this step, refer to Problem 2.

STOP: At this point it is possible to store the samples at -20 °C and continue the protocol at a later point.

Library preparation and sequencing

Timing: 2 days

In this step, we describe how to prepare the ssDNA for sequencing using sequence agnostic, orientation-specific splinted ligations. We then briefly describe how to sequence using an iSeq 100 although any Illumina-type sequencer compatible with the adapters can be used. If you are

proceeding with in-house sequencing, remember to thaw the sequencing reagents according to the manufacturer's instructions. This process often takes 9-18 hours before sequencing.

3. Ligation of isolated ssDNA to the adapter oligonucleotides via splinted ligation
    a. Thaw adapter and splint oligos on ice.
    b. Prepare 10 µL of P5 adapter and splint mix per ssDNA sample.

Reagent	[stock]	[final]	V / µL
T4 DNA Ligase reaction buffer	10x	1x	1
SRSLY P5 adapter	100 µM	10 µM	1
SRSLY P5 splint	100 µM	20 µM	2
ddH_2O			6

    c. Prepare 10 µL of P7 adapter and splint mix per ssDNA sample.

Reagent	[stock]	[final]	V / µL
T4 DNA Ligase reaction buffer	10x	1x	1
SRSLY P5 adapter	100 µM	10 µM	1
SRSLY P5 splint	100 µM	20 µM	2
ddH_2O			6

    d. Anneal oligonucleotides by heating the reaction mix to 95 °C for 10 seconds and then cooling them down to 10 °C at a ramp rate of 0.1 °C/s in a thermocycler.
    e. Prepare 78 µL Ligation mix per ssDNA sample. Mix well by pipetting up and down or vortexing.

Reagent	[stock]	[final]	V / µL
T4 DNA Ligase reaction buffer	10x	1x	8
P5 adapter and splint mix			9
P7 adapter and splint mix			9
ssDNA sample			9
PEG-8000	50%	20%	32
ddH_2O			11

    f. Add 1 µL (30 U) T4 DNA Ligase and 1 µL T4 PNK (10 U) to the reaction mix. Carefully mix by pipetting up and down.
    g. Incubate overnight (~16 hours) at 25 °C in a thermocycler with the lid set to "off".
    h. Incubate 10 minutes at 65 °C to inactivate proteins.
    i. Use the Monarch Spin PCR & DNA cleanup kit with the standard cleanup protocol to clean up 80 µL of each sample. Change ratio of Sample:Binding buffer to 3:1:
        i. Add 240 µL Buffer BZ to solution. Mix well by pipetting up and down. Do not vortex.
        ii. Transfer 320 µL solution onto a spin column and spin for 1 minute at 16,000 g. Discard flow-through.
        iii. Add 200 µL Buffer WZ to the spin column and spin for 1 minute at 16,000 g. Discard flow-through.

iv. Repeat step iv.
v. Transfer spin column to a fresh 1.5 mL sample tube. Spin empty for 1 minute at 16.000 g to remove residual buffer.
vi. Transfer spin column to a fresh 1.5 mL sample tube. Add 17 µL ddH_2O to the center of the matrix to elute DNA. Spin for 1 minute at 16,000 g, then discard spin column.

Note: 2 µL of the final cleaned up product can be used for analysis via Urea-PAGE gel (20% 19:1 acrylamide:bisacrylamide) or a TapeStation or Bioanalyzer. The adapters in this work are 33 nt and 34 nt long, meaning the expected ligated product is library length + 67 nt, in the case of N_{30} 97 nt. If low yields are observed, refer to Problem 3.

4. Addition of Index- and sequencing-primers via PCR
   A. Prepare the following PCR mix:

Reagent	[stock]	[final]	V / µL
adapter ligated DNA			15
NEBNext Ultra II Q5 Master Mix	2x	1x	25
Index Primer (NEB primer set)			5
Universal Primer (NEB primer set)			5

   B. Run the following PCR cycle:

Steps	Temperature	Time	Cycles
Initial Denaturation	98 °C	30 sec	1
Denaturation	98 °C	10 sec	
Annealing/Extension	65 °C	75 sec	15 cycles
Final extension	65 °C	300 sec	1
Hold	4 °C	forever	

CRITICAL: Do not use the same index primer for multiple samples! Make a note of which index primer is used for each sample and the correct index read of that primer. You will need to provide this information to the sequencer.

   C. Clean up using NucleoMag NGS Clean-up and size selection:
      i. Vortex NucleoMag solution until all beads are resuspended.
      ii. Add 45 µL bead solution to the finished PCR reaction, incubate 5 minutes.
      iii. Put on magnetic rack, incubate 5 minutes, remove supernatant.
      iv. Add 200 µL 80% Ethanol while on magnetic rack, incubate 30 seconds, remove supernatant.
      v. Repeat step iv. Insure that as much liquid as possible is removed.
      vi. Air dry beads for a maximum of 5 minutes.
      vii. Remove from magnetic rack. Add 33 µL of 0.1 TE buffer (10 mM Tris-HCl pH 8.0, 1 mM EDTA), mix well by pipetting up and down to resuspend beads. Incubate 2 minutes.
      viii. Put on magnetic rack, incubate 5 minutes.
      ix. Transfer 30 µL to a fresh 0.2 mL PCR tube.

D. Confirm success of the library preparation e.g using a TapeStation with a D1000 cassette if available.

5. Based on the TapeStation concentration measurements, dilute and mix your library for loading concentrations.
    A. For calculations, use the peak corresponding to the correct library length (for the same adapters and library length this peak is at 154 bp), as well as the peak corresponding to just an adapter dimer (for the same adapters this peak is at 124 bp) and the peak corresponding to a double insert (for the same adapters and library length this peak is at 184 bp) added together as the total sample library concentration.
    B. In low-binding tubes, dilute each sample library to 1 nM with 10 mM Tris pH 8.0.
    C. Mix 10 µL of each 1 nM sample library in a low-binding tube. This yields a 1 nM mixed sample library.
    D. Prepare the final loading library by preparing this last dilution step and adding in the PhiX Control Library:

Reagent	[stock]	[final]	V / µL
mixed sample library	1 nM	50 pM	5
PhiX Control V3	1 nM	50 pM	5
Tris pH 7.5	10 mM	10 mM	90

6. Prepare the iSeq 100 and sequencing cassette per manufacturer's instructions:
    A. Load the sequencing cassette with the flow cell and library into the iSeq 100:
        i. Thaw the flow cell for 30 minutes at room temperature.
        ii. Unpack thawed sequencing cassette.
        iii. Invert sequencing cassette 5 times.
        iv. Tap sequencing cassette on table 5 times.
        v. Pierce foil protecting the library loading well with a pipette tip.
        vi. With a fresh pipette tip, load 20 µL of the final loading library into the library loading well.
        vii. Insert the flow cell into the sequencing cassette.
        viii. Insert the sequencing cassette into the iSeq 100.
    B. On the sequencer, set up a new run in Local Run Manager:
        i. Select "Generate FastQ" as method.
        ii. Name your experiment.
        iii. Choose "Custom" as library preparation protocol.
        iv. Choose 1 Index primer read (P7) with read length of 6.
        v. Choose dual-read mode with a read length of 151 each.
        vi. In the sample table, add all the samples your prepared together with the corresponding Index read
    C. In the sequencing software, click on sequencing. Follow the instructions on the sequencer, choose the new experiment created in the step above, and start the sequencer.

Data analysis

Timing: days – weeks

This final step provides a workflow for analyzing the sequencing data. First, we describe the general downstream processing, followed by multiple optional analysis modules for your specific data.

7. Quality control of sequencing itself using Illumina Sequencing Analysis Viewer.
    A. Download or transfer the entire folder containing sequencing results onto a machine with Illumina Sequencing Analysis Viewer installed.
    B. Open Illumina Sequencing Analysis Viewer. The software is available under the following link: https://support.illumina.com/sequencing/sequencing_software/sequencing_analysis_viewer_sav.html)
    C. At the top, click "Browse" and navigate to the folder containing the sequencing results. Open that folder.
    D. On the "Analysis" screen, many relevant metrics can be found, like quality statistics for reads per cycle, sequence etc.
    E. Changing to "Imaging" and clicking the "Scatter Plot…" button allows plotting of different characteristics in a x/y scatter plot with each tile on the flow cell as a separate data point.
        i. Plot "% Occupied" on x-Axis.
        ii. Plot "% Pass Filter" on y-Axis.
        iii. You should see an approximately oval cloud of data points if the flow cell was properly loaded. If you instead see a diagonal line with x ≈ y, the flow cell was underloaded. A vertical band at % Occupied 95-100% indicates overloading.

        Note: If the number of reads per sample are unexpectedly low, refer to Problems 4 and 5.

8. Quality control of FASTQ files using FastQC. [11]

    CRITICAL: Quality control with FastQC should always be performed with raw FASTQ files even after adapter trimming or filtering.

    A. If applicable, unzip the FASTQ files with gzip.

```
>#!bin/bash
>gunzip *.fastq.gz
```

    B. Install FastQC and MultiQC in a new environment using conda.

```
>#!bin/bash
>conda create -n fastqc-env fastqc multiqc -c bioconda -c
conda-forge
```

    C. Activate the new environment and run FastQC.

```
>#!bin/bash
>conda activate fastqc-env
>fastqc *.fastq
```

D. FastQC generates .html outputs for each file it analyzed. You can either look at each of these individually, or run MultiQC to analyze them together.[12]

```
>#!bin/bash
>multiqc .
```

Note: MultiQC uses any FASTQ output in the specified folder to generate its own output, which is simply a combined report of all the individual FastQC reports. We advise using MultiQC only to group analysis examples of the same processing stage together. For example, do not use it to group samples before and after adapter trimming, as these files may differ significantly and their combined report may not be useful.

9. Remove adapters using cutadapt.[13]
   A. Install cutadapt in a new environment using conda.

```
>#!bin/bash
>conda create -n cutadapt-env cutadapt -c bioconda -c
conda-forge
```

B. Activate the new environment and run cutadapt for trimming of paired end reads.

```
#!/bin/bash
source "$(conda info --base)/etc/profile.d/conda.sh"

# Define an array of variable names
variables=("Ref" "Sup" "Pel")

# Loop through each variable
for var in "${variables[@]}"; do
    # Define input and output file names based on the variable
    input_file_1="../raw_data/${var}_R1.fastq"
    input_file_2="../raw_data/${var}_R2.fastq"
    trimmed_output_file_1="${var}_trim_1.fastq"
    trimmed_output_file_2="${var}_trim_2.fastq"

    # Run cutadapt to trim adapters
    conda activate cutadapt-env
    cutadapt -a "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" -A
"AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" --discard-untrimmed -o
"$trimmed_output_file_1" -p "$trimmed_output_file_2"
"$input_file_1" "$input_file_2"
    conda deactivate
    # Run FastQC on the trimmed output file
    conda activate fastqc-env
    fastqc "$trimmed_output_file_1"
    fastqc "$trimmed_output_file_2"
    conda deactivate
done

echo "Processing complete for all variables."
```

Note: This script assumes that you have the original FASTQ files stored in the folder "../raw_data" with the naming convention "{var}_RX.fastq" with X = 1 or 2 (1 being the first read, 2 the second read). Make sure that your data follows this format or change the above script for your needs.

Note: This script uses the adapter sequences P5 and P7 presented in this work for adapter trimming. Make sure to change the above sequences to match the ones you used when running the above script.

Note: The error tolerance for finding matching adapters is set to 0.2. This permits a total of 3 mismatches when searching for adapters.

C. Confirm if adapter trimming was successful by looking at the FastQC output.

10. Filter sequences for correct length using seqkit.[14]
    A. Install seqkit using conda.

```
>#!bin/bash
>conda create -n seqkit-env seqkit -c bioconda -c conda-forge
```

    B. Activate the new environment and run the following script to filter your sequencing reads by length.

```bash
#!/bin/bash
source "$(conda info --base)/etc/profile.d/conda.sh"

# Define an array of variable names
variables=("Ref" "Sup" "Pel")

# Loop through each variable
for var in "${variables[@]}"; do
    # Define input and output file names based on the variable
    input_file_1="../cut_adap/${var}_trim_1.fastq"
    output_file_1="${var}_fil_1.fastq"
    input_file_2="../cut_adap/${var}_trim_2.fastq"
    output_file_2="${var}_fil_2.fastq"

    # Run seqkit to filter for length
    conda activate seqkit-env
    seqkit seq -m 30 -M 30 ${input_file_1} > ${output_file_1}
    seqkit seq -m 30 -M 30 ${input_file_2} > ${output_file_2}
    conda deactivate

    # Run FastQC on the filtered output file
    conda activate fastqc-env
    fastqc "$output_file_1"
    fastqc "$output_file_2"
    conda deactivate
done

echo "Processing complete for all variables."
```

    C. Confirm if adapter trimming was successful by looking at the FastQC output.

Note: Following this, all analysis modules are optional and highly depend on the samples you aim to analyze. All scripts presented in this protocol are compatible with the data generated from sequencing the N_{30} library isolated above.

D. Generate FASTA files from your fastq files for faster data processing

```
>#!bin/bash
>seqkit fq2fa Ref_fil_1.fastq -o Ref_fil_1.fasta
```

Note: Use seqkit to process all files once. It is not possible to use a wildcard character here; you must therefore process all files manually.

CRITICAL: Since our sequencing data originated from ssDNA only, from now on we will only process the R1 reads, which correspond to the original sequence in the droplets.

11. Calculate and plot nucleotide content in Python.[15–18]
    A. Install and import the relevant python packages into a python environment.

```
>#!bin/bash
>conda install pandas -c conda-forge
>conda install matplotlib -c conda-forge
```

```
>#!/usr/bin/env python3
>import pandas as pd
>import numpy as np
>import matplotlib.pyplot as plt
```

B. Use the following python script to load the FASTA files into a pandas library in python.

```
#!/usr/bin/env python3

def read_fasta_to_dataframe(file_path):
    sequences = []

    with open(file_path, 'r') as file:
        for i, line in enumerate(file):
            if i % 2 == 1:  # Every second line contains the
sequence
                sequences.append(line.strip())

    # Create a DataFrame from the list of sequences
    df = pd.DataFrame(sequences, columns=['Sequence'])

    return df

variants = ["Ref", "Sup", "Pel"]

#Create an empty dictionary
df_raw = {}

#Store each dataframe generated by read_fasta_to_dataframe in
df_raw under its variants name
for variant in variants:
    file_path = f"../len_fil/{variant}_fil_1.fasta"
    df_raw[variant] = read_fasta_to_dataframe(file_path)
```

C. Calculate the nucleotide content per position for each sample.

```
#!/usr/bin/env python3

def calculate_nucleotide_content(df):
    # Get the maximum sequence length
    max_length = df['Sequence'].str.len().max()

    # Initialize a dictionary to store nucleotide counts for
each position
    position_counts = {nucleotide: [0] * max_length for
nucleotide in 'AGCT'}

    # Count the nucleotides at each position
    for sequence in df['Sequence']:
        for position, nucleotide in enumerate(sequence):
            if nucleotide in position_counts:
                position_counts[nucleotide][position] += 1

    # Convert counts to percentages
    position_percentages = {nucleotide: [] for nucleotide in
'AGCT'}
    for position in range(max_length):
        total_count = sum(position_counts[nuc][position] for nuc
in 'AGCT')
        if total_count > 0:
            for nucleotide in 'AGCT':
                position_percentages[nucleotide].append((positio
n_counts[nucleotide][position] / total_count) * 100)
        else:
            for nucleotide in 'AGCT':
                position_percentages[nucleotide].append(0)

    # Create a DataFrame from the position percentages
    percentage_df = pd.DataFrame(position_percentages)

    return percentage_df
#Create an empty dictionary
df_perc = {}
#Store each dataframe generated by calculate_nucleotide_content
in df_perc under its variants name
for variant in variants:
    df_perc[variant] =
calculate_nucleotide_content(df_raw[variant])
```

D. Plotting the nucleotide content per position for each sample in an individual plot.

```
#!/usr/bin/env python3

#Some generic settings to match the asthetics of this paper
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 9
plt.rcParams['font.weight'] = 'bold'

for variant, perc in df_perc.items():
    #Plots the lines for G, A, C and T
    plt.figure(figsize=(2.5, 2))
    plt.plot(perc.index + 1, perc["G"], label="G",
color="orange")
    plt.plot(perc.index + 1, perc["A"], label="A", color="red")
    plt.plot(perc.index + 1, perc["C"], label="C", color="blue")
    plt.plot(perc.index + 1, perc["T"], label="T",
color="green")

    #Labels and design
    plt.grid()
    plt.title(f"{variant} Content", weight="bold", fontsize=10)
    plt.xlabel("#Base", weight="bold", fontsize=10)
    plt.ylabel("Base content / %", weight="bold", fontsize=10)
    plt.yticks(np.arange(15, 40, 5), fontsize=9, weight='bold')

    #Save each figure individually
    plt.savefig(f"{variant}_content.pdf", bbox_inches="tight",
pad_inches=0.1)
    plt.show()
```

E. Creating the corresponding legend as a separate file.

```
#!/usr/bin/env python3

#Creating the legend as a seperate file
plt.figure(figsize=(4, 1))
handles = [
   plt.Line2D([0], [0], color='orange', lw=4),
   plt.Line2D([0], [0], color='red', lw=4),
   plt.Line2D([0], [0], color='blue', lw=4),
   plt.Line2D([0], [0], color='green', lw=4),
]
labels = ['G', 'A', 'C', 'T']
plt.legend(handles, labels, loc='center', fontsize=9, ncol=4)

# Turn off axes and save legend figure
plt.axis('off')
plt.savefig("legend_content.pdf", bbox_inches="tight")
```

F. Calculating the difference between each sample and the “Reference” sample to see the change in distribution before and after samples were taken.

```
#!/usr/bin/env python3

#Creating an empty dictionary
df_diff = {}

#Subtracting the Reference from each sample
for variant in variants:
    df_diff[variant] = df_perc[variant] - df_perc["Ref"]
```

G. Plotting the new difference once again in individual plots for each sample.

```
#!/usr/bin/env python3

#Plotting the df_diff for each sample.
for variant, diff in df_diff.items():
    plt.figure(figsize=(2.5, 2))
    plt.plot(diff.index + 1, diff["G"], label="G",
color="orange")
    plt.plot(diff.index + 1, diff["A"], label="A", color="red")
    plt.plot(diff.index + 1, diff["C"], label="C", color="blue")
    plt.plot(diff.index + 1, diff["T"], label="T",
color="green")

    plt.grid()
    plt.title(f"{variant} Difference", weight="bold",
fontsize=10)
    plt.xlabel("#Base", weight="bold", fontsize=10)
    plt.ylabel(r"$\Delta$ Base content / %", weight="bold",
fontsize=10)
    plt.yticks(np.arange(-8, 9, 4), fontsize=9, weight='bold')

    plt.savefig(f"{variant}_difference.pdf",
bbox_inches="tight", pad_inches=0.1)
```

12. Calculating folding and minimum-free energy values
    A. Install Vienna RNA package per suppliers' instructions:[19–21] https://www.tbi.univie.ac.at/RNA/documentation.html#install
    B. Use the following script to calculate the minimum free energy structure using RNAfold, then extract the minimum free energy from the result file.

Note: The flag -P DNA is parsed so we use DNA parameters for folding determination instead of RNA parameters. The parameters used by RNAfold are from Matthews et al.[22]

```bash
#!/bin/bash

file_list="file_list.txt"
rnafold_output_dir="rnafold_output"
mkdir -p $rnafold_output_dir

while IFS= read -r input_fasta; do
    filename=$(basename "$input_fasta")
    rnafold_output="$rnafold_output_dir/${filename%.fasta}_rnafold.txt
"
    deltaG_output="$rnafold_output_dir/${filename%.fasta}_deltaG.txt"
    RNAfold -i "$input_fasta" --noPS --noconv -P DNA >
"$rnafold_output"
    grep "(" "$rnafold_output" > "$deltaG_output"
    echo "Processed RNAfold and extracted deltaG values for
$output_fasta -> $rnafold_output"

    # Python processing block
    python3 <<EOF
# Define file paths directly from bash variables
deltaG_file = "${deltaG_output}"
dot_bracket_file =
"./${rnafold_output_dir}/${filename%.fasta}_dot_bracket.txt"
energy_file = "./${rnafold_output_dir}/${filename%.fasta}_energy.txt"

# Read input file and process line by line
with open(deltaG_file, 'r') as infile, \
     open(dot_bracket_file, 'w') as dot_out, \
     open(energy_file, 'w') as energy_out:
    for line in infile:
        # Extract first 60 characters (dot-bracket notation)
        dot_bracket = line[:30].strip()
        dot_out.write(dot_bracket + '\n')

        # Extract energy value starting from character 61 onward,
        # stripping spaces and brackets (retaining only numeric parts)
        energy_value_raw = line[31:]  # Start reading from character
61 onward
        energy_value = ''.join(c for c in energy_value_raw if
c.isdigit() or c == '.' or c == '-')
        energy_out.write(energy_value + '\n')

print(f"Generated {dot_bracket_file} and {energy_file}")
EOF

done < "$file_list"
```

Note: The script produces 4 output files: The regular RNAfold output named {var}_rnafold.txt; A file with the dot-bracket structure and minimum free energy values called {var}_deltaG.txt; and then files containing just the dot-bracket structure and just the energy values called {var}_dot_bracket.txt and {var}_energy.txt.

C. In a python environment, import the relevant python packages.

```
>#!/usr/bin/env python3
>import pandas as pd
>import matplotlib.pyplot as plt
>import numpy as np
```

D. Load the energy values for each sample from {var}_fil_1_energy.txt.

```
#!/usr/bin/env python3

file_list = ["Ref", "Sup", "Pel"]
df = {}

for file in file_list:
    df[file] =
pd.read_csv(f"rnafold_output/{file}_fil_1_energy.txt",
header=None)
    df[file].rename(columns={0:"values"}, inplace=True)
```

E. Sort the values into bins, count them and calculate the frequency of each bin with the following script.

```
#!/usr/bin/env python3

num_bins = 10
min_value = -10
max_value = 0
bin_edges = np.linspace(min_value, max_value, num_bins + 1)
x_values = np.linspace(min_value, max_value, num_bins + 2)

df_counts = {}
df_freq = {}

for file in file_list:
    df[file]["bin"] = np.digitize(df[file]["values"], bin_edges)
    df_counts[file] = np.bincount(df[file]["bin"])
    df_freq[file] = df_counts[file] / len(df[file])
```

F. Plot the MFE frequencies as a distribution plot.

```
#!/usr/bin/env python3

plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 9
plt.rcParams['font.weight'] = 'bold'

plt.figure(figsize=(2.5, 2.5))

for file in file_list:
    plt.plot(x_values, df_freq[file], label = file)

plt.legend(loc="upper left")
plt.xlabel(r"Minimum free energy", weight="bold", fontsize=10)
plt.ylabel("relative occurence", weight="bold", fontsize=10)

plt.savefig(fname="deltaG.pdf", bbox_inches='tight',
pad_inches=0.1)
plt.show()
```

G. Calculate the average MFE values per sample.

```
#!/usr/bin/env python3

df_avg = {}

for file in file_list:
    df_avg[file] = df[file]["values"].mean()
    print(f"{file}: {round(float(df_avg[file]), 3)}")
```

13. Discovering enriched motifs in samples compared to a control sample.
    A. Install the meme-suite per suppliers' instructions:[23,24] https://meme-suite.org/meme/doc/install.html
    B. Use streme from the meme suite to find enriched motifs in a sample compared to a reference

```
>#!/usr/bin/env python3
>streme -p ../len_fil/pel_fil_1.fasta -o streme_output -n
../len_fil/sup_fil_1.fasta -maxw 10 --rna
```

CRITICAL: It is essential to run this with the '—rna' flag! Using a DNA library instead will automatically force the algorithm to generate and search the reverse complement of each sequence which is not present in the original experiment. This will result in false positives that are the reverse complements of actual motifs.

Note: Adjusting the length of the discovered motifs can drastically increase or decrease the number of motifs found. For example, with a maxw of 20 we found a total of 10 statistically

significant motifs, whereas with a maxw of 10 we found over 20. This is due to the way the algorithm works: it removes parts of sequences that were found to carry a motif in order to avoid finding many variations of the same motif in the same sites.

Note: Please note that this algorithm is very computationally taxing as it was designed for far fewer, longer reads and is not capable of multi-threading. For a sample of 370,000 reads and a reference of 1,500,000 reads, we found that it took approximately one day on an i9 14900k processor and required approximately 25 GB of RAM.

C. From the output files, you can get the Motif information from the streme.txt file. Copy these (starting with the line "MOTIF 1…" and paste them into a new file called motifs.txt.
D. Import all relevant libraries in python. Install and that you are missing analogues to previously installed libraries.[25]

```
#!/usr/bin/env python3

import numpy as np
import logomaker
import pandas as pd
import matplotlib.pyplot as plt
import re
from collections import defaultdict
```

E. Use the following script to open both motifs.txt as well as sites.tsv for plotting of logos and positional information of logos.

```
#!/usr/bin/env python3

# Load the text file
with open("motifs.txt", "r") as f:
    raw_data = f.read()

# Regex to split motifs
motif_blocks = re.findall(r'(MOTIF\s+\d+.*?)(?=MOTIF\s+\d+|\Z)',
raw_data, re.DOTALL)

motifs = []

# Map index to base
base_order = ['A', 'C', 'G', 'T']

for block in motif_blocks:
    # Extract lines with matrix values only
    lines = block.strip().splitlines()
    matrix_lines = [line for line in lines if
re.match(r'^\s*\d*\.?\d+\s+', line)]

    motif = defaultdict(list)

    for line in matrix_lines:
        probs = list(map(float, line.strip().split()))
        for i, base in enumerate(base_order):
            motif[base].append(probs[i])

    motifs.append(dict(motif))

df = pd.read_csv("sites.tsv", sep="\t")
```

F. Calculate information content to plot logos with information content on the y-axis instead of nucleotide content (column 1 of final figure)

```
#!/usr/bin/env python3

def calculate_information_content(motif):
    ic = np.log2(4) + (motif * np.log2(motif)).sum(axis=1)
    ic = ic.fillna(0)  # Handle NaNs resulting from log2(0)
    return ic
```

G. Calculate positional hits for each motif for later plotting (column 3 of final figure).

```
#!/usr/bin/env python3

position_counts = defaultdict(lambda: defaultdict(int))
total_counts = defaultdict(int)

for _, row in df.iterrows():
    # Skip rows with missing start/end
    if pd.isna(row['site_Start']) or pd.isna(row['site_End']):
        continue

    motif = row['motif_ALT_ID']
    try:
        start = int(row['site_Start'])
        end = int(row['site_End'])
    except ValueError:
        continue  # skip if conversion fails

    # Only count positions within the 1–30 range
    for pos in range(start, end + 1):
        if 1 <= pos <= 30:
            position_counts[motif][pos] += 1

    total_counts[motif] += 1
```

H. Build a dataframe storing all information for column 3 (positional hits of motifs).

```
#!/usr/bin/env python3

# Create list of all unique motifs
motifs_ids = list(position_counts.keys())

# Build table for Column 1-3
data = []

for motif in motifs_ids:
    row = {'motif_ALT_ID': motif}
    total = total_counts[motif]
    row['total'] = total

    # Fill in positions 1–30
    for pos in range(1, 31):
        row[f'pos{pos}'] = position_counts[motif].get(pos, 0)

    data.append(row)

# Convert to DataFrame
result_df = pd.DataFrame(data)
```

I. Define a function for numerical sorting of Indexes (needed to correctly sort dataframes).

```
#!/usr/bin/env python3

def numeric_sort_index(df):
    # Extract the number from strings like "STREME-11"
    nums =
df.index.to_series().str.extract(r'(\d+)$').astype(int)[0]
    return df.iloc[nums.argsort()]
```

J. Build a dataframe for column 4 (Starting position of motifs).

```
#!/usr/bin/env python3

# Build df_start used for Column 4
df_start = (
    df.dropna(subset=['site_Start'])
      .assign(site_Start=lambda x: x['site_Start'].astype(int))
      .query('1 <= site_Start <= 30')
      .groupby(['motif_ALT_ID', 'site_Start'])
      .size()
      .unstack(fill_value=0)
)

# Ensure all positions exist
for pos in range(1, 31):
    if pos not in df_start.columns:
        df_start[pos] = 0
df_start = df_start[sorted(df_start.columns)]

# Sort index numerically
df_start = numeric_sort_index(df_start)
```

K. Build a dataframe for column 5 (end sites of motifs).

```
#!/usr/bin/env python3

# Build df_end used for Column 5
df_end = (
    df.dropna(subset=['site_End'])
      .assign(site_End=lambda x: x['site_End'].astype(int))
      .query('1 <= site_End <= 30')
      .groupby(['motif_ALT_ID', 'site_End'])
      .size()
      .unstack(fill_value=0)
)

for pos in range(1, 31):
    if pos not in df_end.columns:
        df_end[pos] = 0
df_end = df_end[sorted(df_end.columns)]
df_end = numeric_sort_index(df_end)
```

L. Calculate fractions of hits from the three dataframes generated above.

```
#!/usr/bin/env python3

# Fractions of hits at each position used for Column 3
result_fraction_df = result_df.set_index('motif_ALT_ID').copy()
for pos in range(1, 31):
    result_fraction_df[f'pos{pos}'] = (
        result_fraction_df[f'pos{pos}'] /
result_fraction_df['total'].replace(0, np.nan)
    ).fillna(0)

# Sort same as df_start/df_end
result_fraction_df = numeric_sort_index(result_fraction_df)

#Fractions of start and end used for Column 4 and 5

df_start_frac = df_start.div(df_start.sum(axis=1),
axis=0).fillna(0)
df_end_frac = df_end.div(df_end.sum(axis=1), axis=0).fillna(0)
```

M. Generate final dataframe that will store all data used for plotting.

```
#!/usr/bin/env python3
plot_data = []

# Loop through motifs in the already-sorted result_df
for mid in result_df['motif_ALT_ID']:
    plot_data.append({
        'id': mid,
        'logo_matrix':
pd.DataFrame(motifs[result_df['motif_ALT_ID'].tolist().index(mid
)]),
        'pos_fraction': result_fraction_df.loc[mid, [f'pos{p}'
for p in range(1, 31)]].values,
        'start_fraction': df_start_frac.loc[mid].values,
        'end_fraction': df_end_frac.loc[mid].values
    })
```

N. Plot a master figure with five columns per motif, with the first column being the logo with information content as y-axis, second column logo with nucleotide content as y-axis, third column showing positional hits (frequency that logo appears at each site), fourth column showing starting sites of logos and the fifth column showing ending positions of logos.

```
#!/usr/bin/env python3

# Number of motifs
num_motifs = len(plot_data)

# Create subplots
fig, axes = plt.subplots(num_motifs, 5, figsize=(5 * 35 / 25.4,
18 / 25.4 * num_motifs))
plt.subplots_adjust(wspace=0.5, hspace=0.7)

# Define the titles for each column
col_titles = [
    "Logo inf",
    "Logo nuc",
    "Motif pos",
    "Motif start",
    "Motif end"
]

# Position titles above the top row's axes
for col_idx, title in enumerate(col_titles):
    ax = axes[0][col_idx]
    ax.set_title(title, fontsize=12, fontweight='bold', pad=20)

# Loop over motifs
for i, motif_entry in enumerate(plot_data):
    row_axes = axes[i]  # 5 axes in a row
    motif_name = motif_entry['id']

    # Extract precomputed data
    df_logo = motif_entry['logo_matrix']
    pos_fraction = motif_entry['pos_fraction']
    start_fraction = motif_entry['start_fraction']
    end_fraction = motif_entry['end_fraction']

    # Calculate information content
    ic = calculate_information_content(df_logo)
    df_ic = df_logo.mul(ic, axis=0)

    color_scheme = {
        'A': 'red',
        'C': 'orange',
        'G': 'blue',
        'T': 'green'
    }
```

```
    # Column 1: Information content logo
    logomaker.Logo(df_ic, ax=row_axes[0],
color_scheme=color_scheme)
    row_axes[0].set_xticks([])
    row_axes[0].set_ylim(0, 2)

    # Column 2: Nucleotide content logo
    logomaker.Logo(df_logo, ax=row_axes[1],
color_scheme=color_scheme)
    row_axes[1].set_xticks([])
    row_axes[1].set_ylim(0, 1)

    # Column 3: Position hits
    row_axes[2].plot(range(1, 31), pos_fraction, color='black')
    row_axes[2].set_ylim(0, 1)
    row_axes[2].set_xlim(1, 30)
    row_axes[2].set_xticks([])

    # Column 4: Start positions
    row_axes[3].plot(range(1, 31), start_fraction,
color='black')
    row_axes[3].set_ylim(0, 0.1)
    row_axes[3].set_xlim(1, 30)
    row_axes[3].set_xticks([])

    # Column 5: End positions
    row_axes[4].plot(range(1, 31), end_fraction, color='black')
    row_axes[4].set_ylim(0, 0.1)
    row_axes[4].set_xlim(1, 30)
    row_axes[4].set_xticks([])

    # Y-axis styling
    for ax in row_axes:
        ax.tick_params(axis='y', labelsize=8)
        for tick in ax.get_yticklabels():
            tick.set_fontsize(8)
            tick.set_fontweight('bold')

# Save & show
plt.savefig("combined_motifs_all_in_one.pdf",
bbox_inches='tight')
plt.show()
```

14. Calculate abstracted structure of sequences and analyze them for enrichment of specific motifs.
    A. Install RNAshapes in a new environment using conda:[26]

```
>#!bin/bash
>conda create -n RNAshapes-env RNAshapes -c bioconda
```

    B. Run the following script to generate the abstracted structure for all samples.

```
#!/bin/bash
source "$(conda info --base)/etc/profile.d/conda.sh"

variables=("Ref" "Sup" "Pel")

conda activate RNAshapes-env
for var in "${variables[@]}"; do
    RNAshapes -c 0 < ../len_fil/${var}_fil_1.fasta >
${var}_abstract.txt
    echo "${var} finished computing"

done
conda deactivate

echo "Finished the script"
```

    C. Import all relevant libraries in python. Install and that you are missing analogues to previously installed libraries.

```
#!/usr/bin/env python3

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

D. Use the following script to load the data into a python dataframe.

```
#!/usr/bin/env python3

def RNAshaper_to_dataframe(file_path):
    energies = []
    dotbrackets = []
    abstracted_dotbrackets = []

    with open(file_path, 'r') as file:
        lines = file.readlines()
        total_lines = len(lines)
        # Loop through the file and look for header lines (those
starting with '>')
        for i in range(total_lines):
            if lines[i].startswith('>'):
                # Attempt to get the third line after the header
(i+2)
                if i + 2 < total_lines:
                    line = lines[i + 2].strip()
                    components = line.split()
                    # Make sure the line has exactly 3
components
                    if len(components) == 3:
                        try:
                            energy = float(components[0])
                            dotbracket = components[1]
                            abstracted_dotbracket =
components[2]

                            energies.append(energy)
                            dotbrackets.append(dotbracket)
                            abstracted_dotbrackets.append(abstra
cted_dotbracket)
                        except ValueError:
                            # Skip lines where energy is not a
float
                            continue
    df = pd.DataFrame({
        'Energy': energies,
        'DotBracket': dotbrackets,
        'Abstracted': abstracted_dotbrackets
    })
    return df
```

```
#!/usr/bin/env python3

variables = ("Ref", "Sup", "Pel")

df = {}

for var in variables:
    file_path = f"{var}_abstract.txt"
    df[var] = RNAshaper_to_dataframe(file_path)
```

E. Count the occurrences of each abstracted structures.

```
#!/usr/bin/env python3

def count_abstracted_structures(df):
    # Count occurrences of each unique structure in the
'Abstracted' column
    structure_counts = df['Abstracted'].value_counts()

    # Create a new DataFrame with the structure and its count
    result_df = pd.DataFrame({
        'Structure': structure_counts.index,
        'Occurrences': structure_counts.values
    })

    return result_df

df_abs = {}

for var in variables:
    df_abs[var] = count_abstracted_structures(df[var])
```

F. Plot the occurrences of different structures in a grouped bar graph.

```python
#!/usr/bin/env python3

# Dynamically collect all unique "Structure" values across all
df_abs DataFrames
all_categories = sorted(
    set().union(*[df_abs[var]["Structure"] for var in
variables]),
    key=lambda x: (len(x), x)  # First sort by length, then
alphabetically
)

# Reindex all DataFrames in df_abs to include all categories,
filling missing ones with 0
for var in variables:
    df_abs[var] =
df_abs[var].set_index("Structure").reindex(all_categories,
fill_value=0).reset_index()

bar_width = 0.8 / len(variables)  # Adjust bar width based on
the number of variables
x = np.arange(len(all_categories))

plt.figure(figsize=(2.5, 2))

# Loop through variables and plot each one
for i, var in enumerate(variables):
    plt.bar(
        x + (i - len(variables) / 2) * bar_width,  # Shift bars
for each variable
        df_abs[var]["Occurrences"] / len(df[var]) * 100,  #
Normalize occurrences by total sequences
        width=bar_width,
        label=var.capitalize()  # Use variable names as labels
    )

plt.xlabel("Abstracted Structures", size=10, fontweight="bold")
plt.ylabel("Percentage of Sequences", size=10,
fontweight="bold")
plt.xticks(x, all_categories, rotation=45)
plt.yscale("log")

plt.legend()

plt.savefig("Abstracted_Structures.pdf", bbox_inches="tight",
pad_inches=0.1)
plt.show()
```

G. Calculate the average stem length per sample and write it in a new dataframe.

```
#!/usr/bin/env python3

def count_stem_length(df):
    # Calculate the count of '(' for each entry in the
'DotBracket' column
    df['StemLength'] = df['DotBracket'].apply(lambda x:
x.count('('))
    return df

for var in variables:
    count_stem_length(df[var])

def count_stem_length_occurences(df):
    # Count occurrences of each unique structure in the
'Abstracted' column
    structure_counts = df['StemLength'].value_counts()

    # Create a new DataFrame with the structure and its count
    result_df = pd.DataFrame({
        'Length': structure_counts.index,
        'Occurrences': structure_counts.values
    })

    return result_df

df_stem = {}

for var in variables:
    df_stem[var] =
count_stem_length(df[var]).sort_values("Length")
```

H. Plot the density of stem length per sample in a density plot.

```
#!/usr/bin/env python3

# Calculate max_length dynamically from all df_stem DataFrames
max_length = max([df_stem[var]["Length"].max() for var in
variables])

plt.figure(figsize=(2.5, 2.5))

# Loop through each variable to plot its density
for var, color in zip(variables, ["purple", "blue", "green",
"orange", "red", "cyan", "brown"]):  # Add more colors as needed
    # Normalize occurrences by total sequences and plot the line
    plt.plot(
        df_stem[var]["Length"],
        df_stem[var]["Occurrences"] / len(df[var]) * 100,
        alpha=0.7,
        label=var.capitalize(),
        color=color
    )
    # Fill under the curve for better visualization
    plt.fill_between(
        df_stem[var]["Length"],
        df_stem[var]["Occurrences"] / len(df[var]) * 100,
        alpha=0.3,
        color=color
    )

# Labels and other configurations
plt.xlabel("Stem Length", size=10, fontweight="bold")
plt.ylabel("Percentage of Sequences", size=10,
fontweight="bold")
plt.xticks(np.arange(0, max_length + 1, 2))  # Dynamically
adjust x-ticks based on max_length
plt.legend()

# Save and show plot
plt.savefig("StemLength_Density.pdf", bbox_inches="tight",
pad_inches=0.1)
plt.show()
```

Expected outcomes

The results of successful isolation of ssDNA from droplets (Figure 1A) and library preparation (Figure 1B) can be verified on a urea-PAGE gel. The ssDNA isolated from complex coacervates should reveal a clear band (Figure 1C, lane 4). Similarly, the ligation reaction (Figure 1C, lane 5-7) and final PCR (Figure 1C, lane 8-10) should reveal a prominent product band (marked with a red star). Note that bands corresponding to the adapter dimers (no insert), as well as double insert products (concatemers) are expected (Figure 1C, lane 5-7). These consume a small fraction of reads in the sequencing data, but can be ignored as their yield is significantly lower compared to the target library (Figure 1C, lane 8-10). These reads will be removed during downstream filtering of sequencing results. With the library shown and a loading concentration of 12 pM, we obtained 2.8 million reads across three samples (Figure 1D). This comparable low yield is likely due to a high PhiX spike-in (50 pM) bit still provides high-quality data with sufficient depth for robust statistics.

Data analysis should yield figures similar to those in Figure 2. For example, base composition (Figure 2A), difference in minimum free energy of folding (Figure 2B), abstract secondary structure classes (Figure 2C) and average intramolecular hybridization length (Figure 2D). We particularly emphasize motif discovery using STREME,[23,24] which identifies enriched sequence motifs. In our data, we identified a total of 14 statistically significant motifs (Figure 3), using an e-value cutoff of ≤0.05.

We encourage researchers to further mine their dataset for system-specific results in their own dataset. In our associated research manuscript, base-composition results motivated targeted analyses of reads starting/ending in guanosines (Figure S6 in the associated research manuscript), and motif discovery guided searches for runs of G or A in droplet samples (Figure 2F in associated research manuscript). Follow-up experiments in the droplet system verified specific interactions between ssDNA sequences and RNA or peptide droplet components.

Limitations

This protocol is optimized for transient complex coacervates. Because protocell implementations vary widely, recovery steps are inherently system-specific and might deviate e.g. in static droplet systems or vesicle-based models. While these models can reuse the library preparation and analysis modules, alternative ssDNA extraction methods compatible with the system matrix must be developed. If RNA rather than ssDNA is used as nucleic acid component, both recovery and library preparation require re-optimization.

Our analysis pipeline is modular but not exhaustive. It provides a baseline: base composition, predicted folding energies, and motif discovery for short oligonucleotides. Researchers with different systems (longer sequences, double stranded sequences, RNA) may need to extend or develop additional modules to meet their specific analysis needs.

Troubleshooting

Problem 1:

Droplet solution does not turn turbid upon addition of EDC.

Potential solution:

- Make sure to use a fresh solution of dry EDC, as the compound is not stable in the presence of water. If powder the powder forms clumps, it may be too wet, and a new, pristine stock should be opened. For this reason, always store EDC in completely dry conditions.
- Confirm the identity and pH of the peptide stock. Impurities can result in poor or no droplet formation or cause the peptide to precipitate, particularly at low pH values. Always store at pH 5.2 at 4 °C.

Problem 2

Low recovery of ssDNA from the polymer-rich phase.

Potential solution:

- Recovery of ssDNA is a time-sensitive step due to the dependency of droplets on fuel. Work quickly and strictly adhere to the indicated incubation times.
- The droplet pellet formed after centrifugation can be very small and transparent. To confirm the presence of a pellet, consider adding a dye-labelled RNA oligo (e.g. a Cy5-tagged RNA) at a concentration of ~1 µM to the reaction mixture. This will enable identifying a pellet by fluorescence even in the absence of specific excitation. The RNA will later be digested by RNase, rendering this a non-issue for the subsequent application.
- If working with different droplet systems, ensure that your downstream clean-up protocol is compatible with the droplet material. Complex coacervates commonly contain poly-anions and poly-cations that can bind to the column material of spin-down columns from PCR clean-up kits. This may result in oversaturation of the column material due to its limited capacity and cause the loss of ssDNA material during loading and washing.

Problem 3:

Low ligation yields in the one-pot reaction with T4 DNA ligase and T4 PNK.

Potential solution:

- Depending on the system, carry over of droplet material may inhibit T4 DNA ligase. We found that poly-styrene sulfonate and dextran sulphate can both inhibit ligase activity, but they rea difficult to separate from the ssDNA due to their poly-anionic character. It is paramount that the ssDNA is recovered cleanly from your droplet sample.
- If your library contains fixed ends (even just one or two nucleotides), the P5 and/or P7 splint oligos can be modified to match these regions. This will significantly increase yields, as more compatible splints will match your library design.

Problem 4:

The files for each individual sample are almost empty, with a large number of reads ending up in the 'Undetermined' file.

Potential solution:

- Confirm that the correct index sequence was entered in the correct orientation in the sample table of the sequencer.
- If demultiplexing on the sequencer fails, it is possible to demultiplex manually using cutadapt. Follow the instructions at: https://cutadapt.readthedocs.io/en/stable/guide.html#demultiplexing
- Reducing the PhiX spike-in will reduce the number of 'Undetermined' reads, as all reads from the PhiX control will be stored here. However, going too low on the spike-in though might result in poor cluster formation and low-quality sequencing data.

Problem 5:

The sequencing resulted in a low number of reads with a high occupancy but low pass filter on the sequencing flow cell.

Potential solution:

- This is most likely due to overloading of the flow-cell. Verify the loaded concentration, ideally with an orthogonal method (e.g. using a TapeStation or a Qubit).
- Reduce the concentration loaded. We obtained good results with concentrations as low as 8 pM on the iSeq 100, despite the recommended loading concentration being between 50 pM and 200 pM.

Acknowledgments

The BoekhovenLab is grateful for support from the TUM Innovation Network – RISE, funded through the Excellence Strategy. This research was conducted within the Max Planck School Matter to Life, supported by the German Federal Ministry of Education and Research (BMBF) in collaboration with the Max Planck Society. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2094 – 390783311. The BoekhovenLab is grateful for support from the European Research Council through ERC Starting Grant 852187 and ERC Consolidator Grant 101124380.

Author contributions

CM worked on method design and development, as well as sample preparation, sequencing and bioinformatic data analysis. A-LH worked on sample preparation. JB and HM worked on conceptualization of the project. All authors worked together on writing and revising the manuscript. CM and A-LH have contributed equally and have the right to list their name first in their CV or any bibliographical list.

Declaration of interests

The authors declare no conflict of interest.

References

(1) Mirlohi, K.; Blocher McTigue, W. C. Coacervation for biomedical applications: innovations involving nucleic acids. *Soft Matter* **2024**, *21* (1), 8–26. DOI: 10.1039/D4SM01253D. Published Online: Dec. 18, 2024.

(2) Frankel, E. A.; Bevilacqua, P. C.; Keating, C. D. Polyamine/Nucleotide Coacervates Provide Strong Compartmentalization of Mg2+, Nucleotides, and RNA. *Langmuir* **2016**, *32* (8), 2041–2049. DOI: 10.1021/acs.langmuir.5b04462.

(3) Wollny, D.; Vernot, B.; Wang, J.; Hondele, M.; Safrastyan, A.; Aron, F.; Micheel, J.; He, Z.; Hyman, A.; Weis, K.; Camp, J. G.; Tang, T.-Y. D.; Treutlein, B. Characterization of RNA content in individual phase-separated coacervate microdroplets. *Nat Commun* **2022**, *13* (1), 2626. DOI: 10.1038/s41467-022-30158-1. Published Online: May. 12, 2022.

(4) Robertson, M. P.; Joyce, G. F. The origins of the RNA world. *Cold Spring Harbor Perspectives in Biology* **2012**, *4* (5), a003608. DOI: 10.1101/cshperspect.a003608. Published Online: May. 1, 2012.

(5) Orgel, L. E. Polymerization on the rocks: theoretical introduction. *Orig Life Evol Biosph* **1998**, *28* (3), 227–234. DOI: 10.1023/A:1006595411403.

(6) Horning, D. P.; Joyce, G. F. Amplification of RNA by an RNA polymerase ribozyme. *Proceedings of the National Academy of Sciences of the United States of America* **2016**, *113* (35), 9786–9791. DOI: 10.1073/pnas.1610103113. Published Online: Aug. 15, 2016.

(7) Wochner, A.; Attwater, J.; Coulson, A.; Holliger, P. Ribozyme-catalyzed transcription of an active ribozyme. *Science (New York, N.Y.)* **2011**, *332* (6026), 209–212. DOI: 10.1126/science.1200752#.

(8) Späth, F.; Donau, C.; Bergmann, A. M.; Kränzlein, M.; Synatschke, C. V.; Rieger, B.; Boekhoven, J. Molecular Design of Chemically Fueled Peptide-Polyelectrolyte Coacervate-Based Assemblies. *Journal of the American Chemical Society* **2021**, *143* (12), 4782–4789. DOI: 10.1021/jacs.1c01148. Published Online: Mar. 22, 2021.

(9) Donau, C.; Späth, F.; Sosson, M.; Kriebisch, B. A. K.; Schnitter, F.; Tena-Solsona, M.; Kang, H.-S.; Salibi, E.; Sattler, M.; Mutschler, H.; Boekhoven, J. Active coacervate droplets as a model for membraneless organelles and protocells. *Nat Commun* **2020**, *11* (1), 5167. DOI: 10.1038/s41467-020-18815-9. Published Online: Oct. 14, 2020.

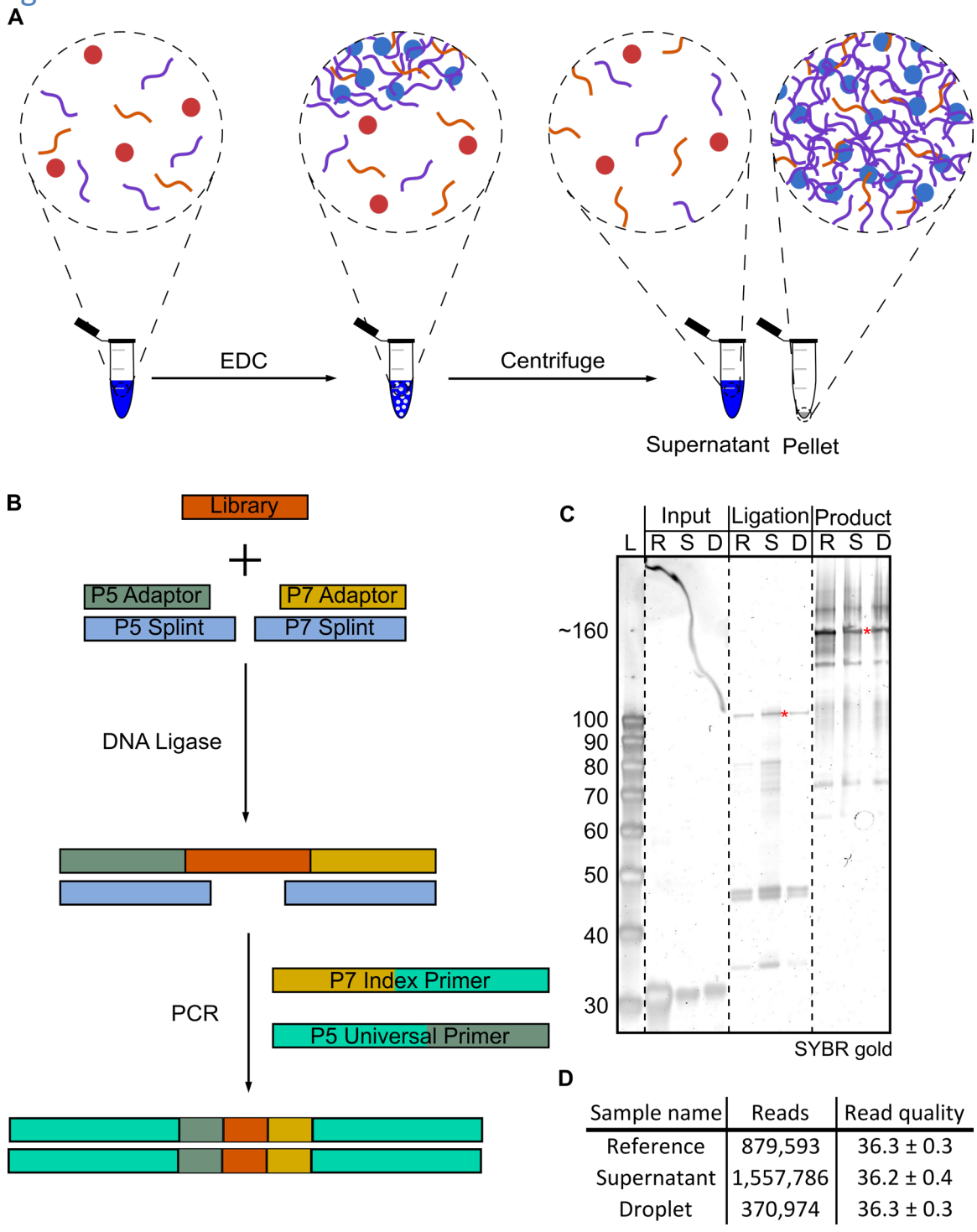
(10) Troll, C. J.; Kapp, J.; Rao, V.; Harkins, K. M.; Cole, C.; Naughton, C.; Morgan, J. M.; Shapiro, B.; Green, R. E. A ligation-based single-stranded library preparation method to analyze cell-free DNA and synthetic oligos. *BMC Genomics* **2019**, *20* (1), 1023. DOI: 10.1186/s12864-019-6355-0. Published Online: Dec. 27, 2019.

(11) *FastQC*, 2015. https://qubeshub.org/resources/fastqc.

(12) Ewels, P.; Magnusson, M.; Lundin, S.; Käller, M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics (Oxford, England)* **2016**, *32* (19), 3047–3048. DOI: 10.1093/bioinformatics/btw354. Published Online: Jun. 16, 2016.

(13) Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet j.* **2011**, *17* (1), 10. DOI: 10.14806/ej.17.1.200.
(14) Shen, W.; Sipos, B.; Zhao, L. SeqKit2: A Swiss army knife for sequence and alignment processing. *iMeta* **2024**, *3* (3), e191. DOI: 10.1002/imt2.191. Published Online: Apr. 5, 2024.
(15) Python Software Foundation. *Python*, 2025. https://www.python.org/downloads/.
(16) McKinney, W. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*; SciPy, 2010; pp 56–61. DOI: 10.25080/Majora-92bf1922-00a.
(17) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; Del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E. Array programming with NumPy. *Nature* **2020**, *585* (7825), 357–362. DOI: 10.1038/s41586-020-2649-2. Published Online: Sep. 16, 2020.
(18) Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9* (3), 90–95. DOI: 10.1109/MCSE.2007.55.
(19) Lorenz, R.; Bernhart, S. H.; Höner Zu Siederdissen, C.; Tafer, H.; Flamm, C.; Stadler, P. F.; Hofacker, I. L. ViennaRNA Package 2.0. *Algorithms for molecular biology : AMB* **2011**, *6*, 26. DOI: 10.1186/1748-7188-6-26. Published Online: Nov. 24, 2011.
(20) Schuster, P. *Fast folding and comparison of RNA secondary structures*, 2014.
(21) Hofacker, I. L.; Fekete, M.; Stadler, P. F. Secondary structure prediction for aligned RNA sequences. *Journal of molecular biology* **2002**, *319* (5), 1059–1066. DOI: 10.1016/S0022-2836(02)00308-X.
(22) Mathews, D. H.; Disney, M. D.; Childs, J. L.; Schroeder, S. J.; Zuker, M.; Turner, D. H. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences of the United States of America* **2004**, *101* (19), 7287–7292. DOI: 10.1073/pnas.0401799101. Published Online: May. 3, 2004.
(23) Bailey, T. L. STREME: accurate and versatile sequence motif discovery. *Bioinformatics (Oxford, England)* **2021**, *37* (18), 2834–2840. DOI: 10.1093/bioinformatics/btab203.
(24) Bailey, T. L.; Johnson, J.; Grant, C. E.; Noble, W. S. The MEME Suite. *Nucleic Acids Res* **2015**, *43* (W1), W39-49. DOI: 10.1093/nar/gkv416. Published Online: May. 7, 2015.
(25) Tareen, A.; Kinney, J. B. *Logomaker: Beautiful sequence logos in python*, 2019. DOI: 10.1101/635029.
(26) Janssen, S.; Giegerich, R. The RNA shapes studio. *Bioinformatics (Oxford, England)* **2015**, *31* (3), 423–425. DOI: 10.1093/bioinformatics/btu649. Published Online: Oct. 1, 2014.

Figures



Sample name	Reads	Read quality
Reference	879,593	36.3 ± 0.3
Supernatant	1,557,786	36.2 ± 0.4
Droplet	370,974	36.3 ± 0.3

Figure 1: ssDNA isolation and library preparation. A) Prior to the addition of fuel, peptide, polyU RNA and ssDNA molecules are mixed in solution, and do not form supramolecular assemblies. When fuel (EDC) is added, phase separation occurs and coacervate droplets form, that sequester ssDNA molecules from the polymer-poor phase. The phases are separated via centrifugation to

yield the polymer-poor and polymer-rich phases, from which ssDNA populations are isolated. B) Library preparation relies on the orientation-specific ligation of two adapter molecules to the recovered ssDNA oligonucletotides. After the ligation, the sequencing primers are added in a PCR using primers with overhangs. C) Urea-PAGE is performed following library preparation. R = Reference, S = Supernatant, D = Droplets. “Input” corresponds to samples obtained from the droplets, while the R sample is the naïve input library. ”Ligation” corresponds to the product of the first step in B and shows a clear band (marked with a red star) corresponding to the expected full-length ligation product and a weaker smear corresponding to the two partially adapter-ligated products. Bands at 45 nt correspond to the adapters and splint, and bands at ~30 correspond to unreacted input ssDNA. “Product” shows the full-length sequencing library, with the correct PCR band being marked with a red star. Multiple additional bands are visible, with two prominent bands roughly 30 nt above and below the main PCR band, corresponding to the no-insert (adapter dimer) and double insert ligation products. The smear and bands below correspond to Indexing primers.

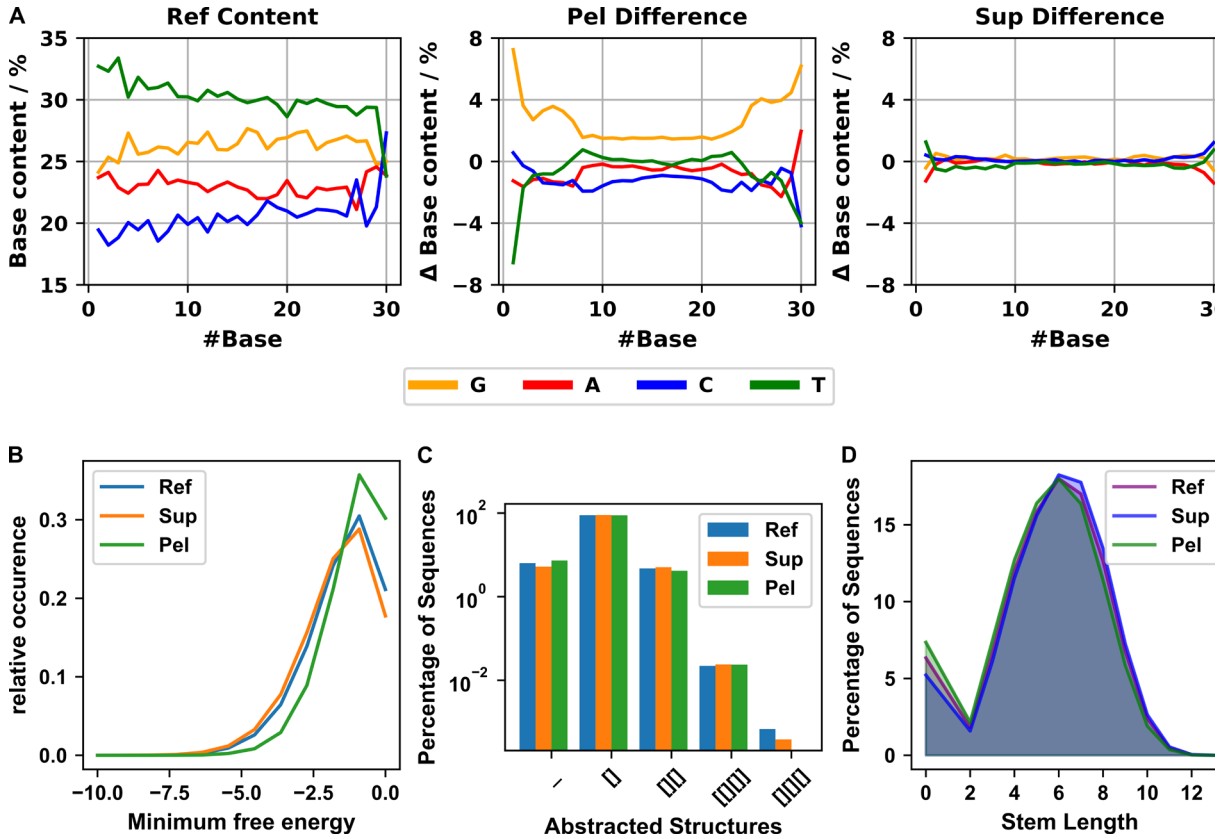


Figure 2: Results generated during data analysis. A) The base composition of the Reference sample enables the identification of biases in the naive library, most result from solid-phase synthesis (left panel). Difference in base composition between the sample from the polymer-rich phase (Pel = Droplet, middle panel) and the polymer-poor phase (Sup = Supernatant, right panel) B) Distribution of minimum free energies of folding for sequencing reads of each sample. C) Occurrence of different abstracted structures in each sample. D) Distribution of intramolecular stem lengths in each sample.

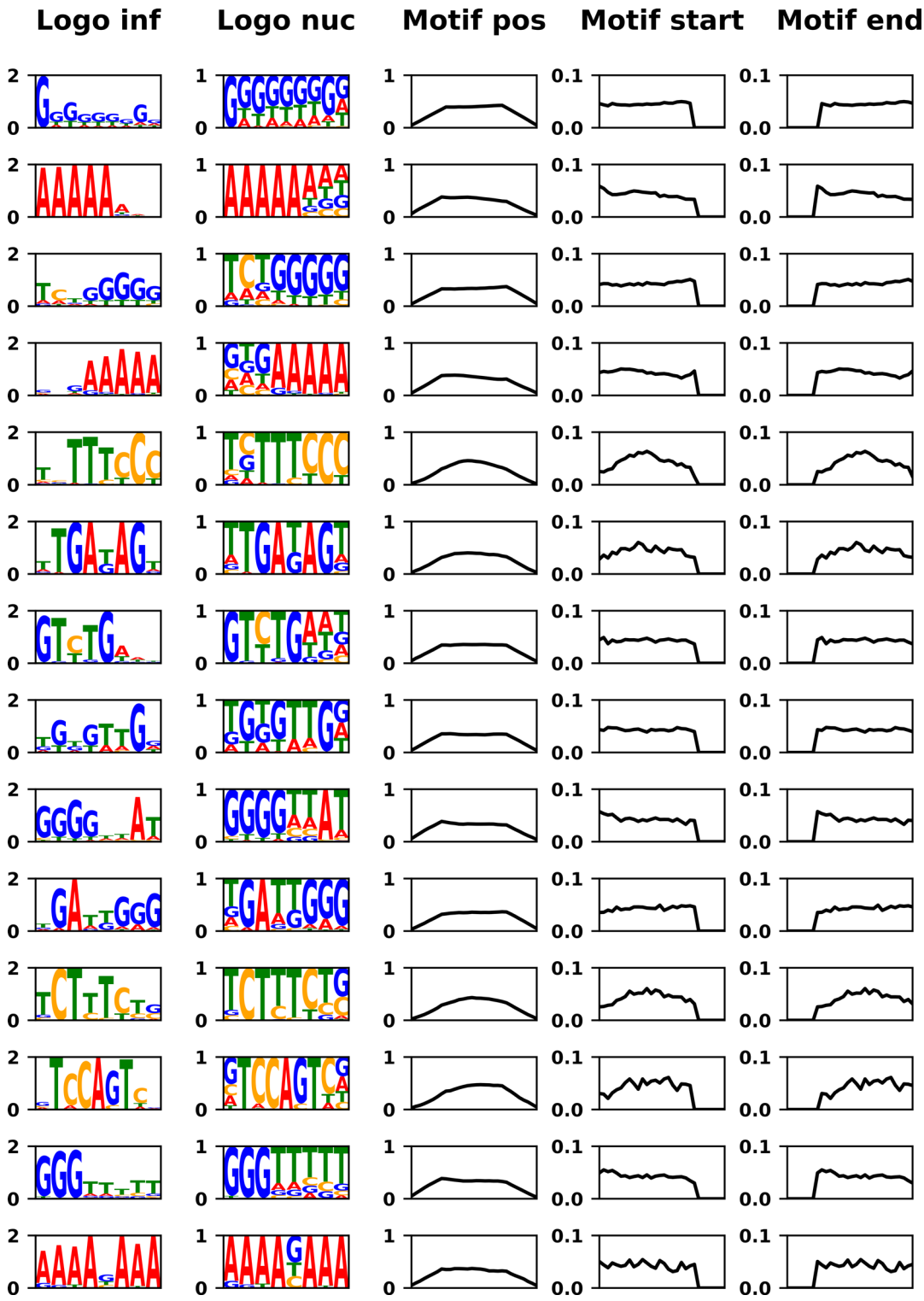


Figure 3: List showing all statistically significant motifs discovered with STREME.