Hybridisation Chain Reaction (HCR) probe designer for *Drosophila* transcripts: the instruction manual

Jeffrey Y Lee[1], Ilan Davis[1*]

[1] School of Molecular Biosciences, University of Glasgow, Glasgow, United Kingdom
[*] Corresponding author ilan.davis@glasgow.ac.uk

Abstract

Fluorescent RNA in situ hybridisation (FISH) allows quantification and localisation of RNA transcripts by visualising individual RNA molecules in single cells. Hybridisation Chain Reaction (HCR) is a popular and open FISH method with detection signal amplification that builds upon the development of catalytic DNA hairpins for highly amplified and multiplexed RNA imaging. While the third generation of the protocol is very well developed, the limited availability of HCR probe designer tools remains a key constraint. Here, we present a user manual for our open-source HCR probe designer tool, available on GitHub, for *Drosophila* transcripts, providing a complete pipeline from cDNA sequence to probe selection and initiator attachment. The resulting oligonucleotides can be synthesised at low cost in 96-well plate formats, and our generalised algorithm can be easily adapted to other species.

Introduction

The third-generation HCR utilises a split-probe approach, in which the catalytic initiator platform only forms when both probes, of a pair, hybridise to the target transcript, thereby suppressing background amplification (Choi, Beck and Pierce, 2014; Choi *et al.*, 2018). A single transcript is targeted by an array of split-probes, with the recommended minimum of 5 pairs for sufficient fluorescent signal, while each pair forms a chain reaction scaffold (Fig. 1). We developed an HCR probe designer that scans the target RNA sequence for optimal thermodynamic properties for hybridisation with the fewest potential off-target hits (Fig. 1). Subsequently, the selected probe sequences are split into two halves and the designer allows attachment of user-selected HCR initiators for multi-colour experiments. The HCR probe designer, written in R, is provided in a flexible *Quarto* markdown format that is easily accessible in RStudio, Visual Studio Code or in headless environments for use on a remote server. Our designer is highly computationally efficient and is practical to use for probe design against *Drosophila* transcripts on a standard laptop computer (Intel Core i5, 8GB RAM, minimally). As BLAST alignment presents a key bottleneck, a larger amount of RAM is recommended for mammalian transcriptomes. In the sections below, we provide a manual for installation and usage of the designer software.

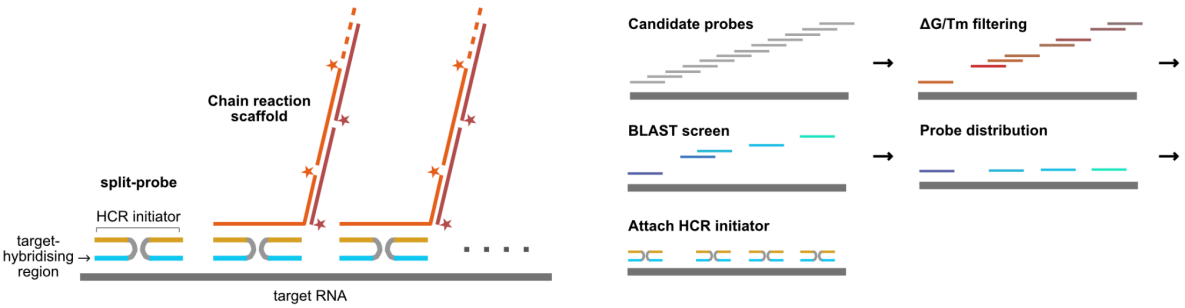

Figure 1: HCR amplification schematics (Left). Probe design overview (Right).

Installation

R and RStudio installation

Install R from CRAN network: https://cran.r-project.org/
Install RStudio: https://posit.co/products/open-source/rstudio/
Install Quarto: https://quarto.org/docs/get-started/

BLAST installation

Install BLAST binary: https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
Check that *R* can find the installed *BLAST*.

```
Sys.which("blastn")
# should say something like "/usr/local/ncbi/blast/bin/blastn"
```

If *R* cannot find *BLAST*, add the *BLAST* installation path to the *R* system path.

```
Sys.setenv(PATH = paste(Sys.getenv("PATH"), path/to/blast/bin, sep = .Platform$path.sep))
```

R package installation

Install *R* wrapper for *BLAST*.

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

BiocManager::install(c("Biostrings", "GenomicFeatures", "GenomicRanges", "Rsamtools", "IRanges",
"rtracklayer", "biomaRt"))

# install.packages("devtools")
devtools::install_github("HajkD/metablastr", build_vignettes = TRUE, dependencies = TRUE)
```

Install other necessary *R* packages.

```
install.packages(c("tidyverse", "furrr", "patchwork", "valr"))
```

Instruction manual

Follow the steps outlined in the Quarto document *HCRv3_probe_design_workbook.qmd*. The guide below provides detailed instructions for each step. The workbook uses relative paths for use with the RProject/RStudio workflow. If RStudio is not being used, change input and output paths to absolute paths as necessary. Upon successful installation, the environment code block in the workbook should load all necessary libraries and functions. In this guide, we use *Drosophila* gene *Cip4* (FBgn0035533) as a working example.

1. Prepare tiled candidate probes

Initially, we generate an array of candidate probes from the target RNA sequence using a 52-nt sliding window. This 52-nt chunk will later be broken down into two 25-nt split-probes (Dardani *et al.*, 2022) that form the target regions for hybridisation. The designer takes a FASTA file as an input, which can either be a DNA (contains T) or an RNA (contains U) sequence. It is very important to consider at this stage whether you wish to detect all mRNA isoforms (constitutive) or select isoforms (isoform-specific) of a gene. Depending on the experimental design, the user can use transcript architecture to their advantage by choosing the sequence used to design probes (e.g. splicing isoforms, 3'UTR isoforms, retained introns). When detecting all isoforms, tools such as *ClustalW* can be used to find constitutive sequences. Additionally, it is important to consider exon-intron junctions carefully, as probes that span splice junctions will fail to detect pre-mRNAs. This can be avoided by either (i) putting character 'N' between exons in case of a single fasta entry; or (ii) having each exon as separate fasta entries contained in a single FASTA file, which can be easily downloaded from the UCSC browser (Fig. 2).

Assign file path (`fasta_file`) and create a name (`target_name`) for the probe set. This code block will output sliding windows of candidate probes across the transcript (Fig. 2).

```
## ---- Input FASTA and name
fasta_file <- "./input/Dmel_cip4-exon.fa"
target_name <- "cip4-exon"

## ---- Get target RNA sequence
target_fasta <- readBStringSet(fasta_file)
target_raw <- paste(target_fasta, collapse = "N")

## ---- Prepare target RNA sequence
target <- clean_pasted_seq(target_raw)
target_sequence_total_length <- nchar(target)

## ---- Extract candidate probes into a dataframe
candidate_probes <- generate_candidate_probes(target_seq   = target,
                                              oligo_length = 52)
## ---- Inspect
inspect_probe_prep(candidate_probes)
```

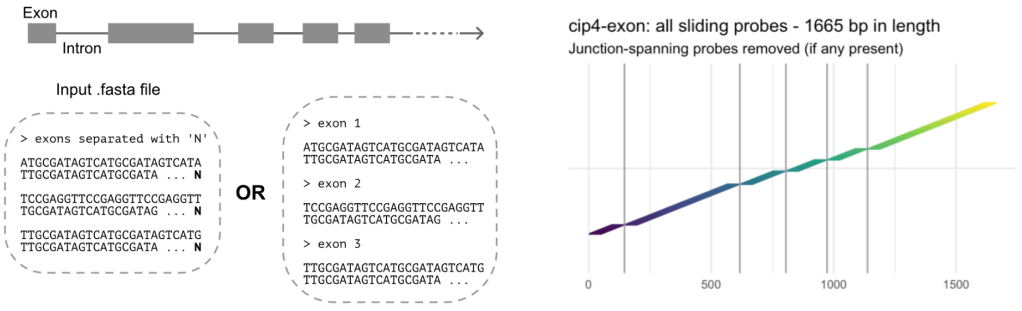

Figure 2: Input FASTA format for designing exonic probes (Left). Sliding window of candidate probes (Right). Vertical lines indicate exon-intron junctions.

2. Calculate thermodynamic and sequence composition parameters for potential oligo probes

This section calculates thermodynamic parameters (melting temperature, Gibb's free energy and GC content) for each of many individual oligos within a sliding window. The parameters are calculated based on the HCR hybridisation buffer conditions, as previously described (Choi *et al.*, 2018).

You should not change the parameters, unless you are using alternative hybridisation conditions (Raj *et al.*, 2008; Yang *et al.*, 2017; Choi *et al.*, 2018). The following code calculates the thermodynamic parameter values for each sliding window (Fig. 3).

```
## ---- Set the hybridisation condition
temperature <- 37
Na <- 0.3
oligo_conc <- 5e-5

## ---- Get thermodynamic parameters
thermodynamics <- get_thermodynamic_parameters(candidate_probes, temperature, Na, oligo_conc)

## ---- Annotate candidate sequences
candidate_probes_annotated <- annotate_probes(candidate_probes, thermodynamics)

## ---- Generate exploratory plots
c("Tm", "dG", "GC_content") %>% generate_exploratory_plots()
```

Figure 3: Thermodynamics parameters of candidate probes.

3. Filter based on thermodynamics and sequence composition parameters to select optimal probes

After calculating the thermodynamic parameters for many candidate probes, they are filtered for optimal thermodynamic and GC-content parameters. Default filtering parameters are provided below. Optimal Gibb's free energy (dG) has been previously demonstrated to have greater influence on DNA:RNA hybridisation efficiency than melting temperature (Tm) (Tsanov *et al.*, 2016; Acheampong *et al.*, 2022). Therefore, when a lenient filtering is required, it is advisable to widen the Tm or GC-content filter ranges while keeping the target dG as close as possible to -60 kcal/mol. Additionally, base composition filters, using *Oligostan* parameters, can avoid selection of regions capable of forming internal hairpins (Tsanov *et al.*, 2016).

The user should set the filtering parameters, as necessary, within the permitted ranges: dG [-110, -50]; Tm [10, 100], GC [0.3, 0.8]. However, it is recommended to first try to alter Tm and GC-content, before making changes to the dG target range. Running the following code will output probes that pass the selected thermodynamic parameters (Fig. 4).

```
## ---- Set filtering parameters
target_dG <- -60
target_dG_halves <- -32
dG_range <- c(-75, -50)
Tm_range <- c(30, 90)
```

```
GC_range <- c(0.4, 0.7)
pass_a_comp       <- FALSE
pass_c_comp       <- FALSE
pass_a_stack      <- TRUE
pass_c_stack      <- TRUE
pass_c_spec_stack <- TRUE

## ---- Filter based on thermodynamic/nucleotide compositions
candidate_probes_filtered <- filter_candidate_probes(candidate_probes_annotated)

## ---- Inspect filtered probes
plot_inspection(candidate_probes_filtered, "Tm")
```

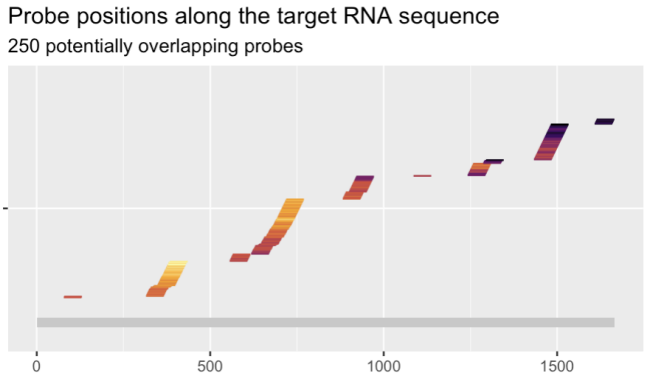

Figure 4: Candidate probes filtered for thermodynamics criteria.

4. BLAST screen

Next, the filtered candidate probes are further screened for potential off-target hits, which would lead to spurious fluorescent signals. Candidate sequences are filtered using *short-blastn* against a custom *Drosophila melanogaster* transcriptome database, supplied as a gzipped file in this repository, which can be unzipped to a directory of choice. The database has been created using cDNA, ncRNA and intron sequences from *D.mel* ENSEMBL version99 data. Each candidate will be BLASTed against the *D.mel* transcriptome, so a transcript-to-gene ID conversion table is also provided (*Dmel_tx2gene_ENSEMBL_v99.csv*) for easy interpretation of the BLAST output. The first time this code is run, it may take a long time to complete (~ 10 mins), as it initialises the BLAST database in your directory.

Assign directory paths to the unzipped BLAST database file (`blast_file`) and the tx2gene file (`tx2gene_csv`). BLAST match to the transcriptome database is determined by the `evalue_cutoff` parameter. Then, the stringency is controlled by the `max_blast_matches` parameter where the lowest value of 1 would allow BLAST matches to only one gene, i.e. our gene of interest. This level of stringency is fine for *D. mel*, but may need to be adjusted for other species with greater transcriptome diversity. The BLAST output is available (`blast_output`) in a table format, which is useful should the user wish to cross-reference tissue/cell-specific RNA-seq data for custom filtering criteria. This code block outputs an intermediary histogram of BLAST gene matches that can be used to optimise the `max_blast_matches` parameter, in case the screen fails to recover any candidate sequences (Fig. 5). Upon completion, screened probes' placements will be shown along the target transcript (Fig. 5).

```
## ---- Set params
blast_file <- "~/Documents/BLAST/Dmel/Dmel_BLASTdb_ens99.fa"
tx2gene_csv <- "./data/Dmel_tx2gene_ENSEMBL_v99.csv" # To convert tx_id to gene_id
evalue_cutoff <- 0.1 # smaller the value the more lenient
max_blast_matches <- 1 # Max number of BLAST hits
allowOverlapBreakRegion <- TRUE # Allow overlap over the 25/25 break region?
```

```
## ---- BLAST nucleotide
blast_output <- run_blastn_short(df = candidate_probes_filtered, db = blast_file, tx2gene =
tx2gene_csv)
blast_summary <- summarise_blast_output(blast_output, allowOverlapBreakRegion)

ggplot(blast_summary, aes(n_matches)) + geom_histogram(binwidth = 1)

## ---- Screen candidate probes
blast_screened <- screen_with_blast_summary(candidate_probes_filtered, max_blast_matches,
blast_summary)
candidate_probes_screened <- blast_screened[[1]]
merge_df <- blast_screened[[2]]

## ---- Inspect BLAST screened probes
plot_inspection(candidate_probes_screened, "Tm") +
  geom_linerange(data = merge_df,
      aes(x = "", ymin = start, ymax = end),
      colour = "gray40", size = 2,
      inherit.aes = FALSE)
```

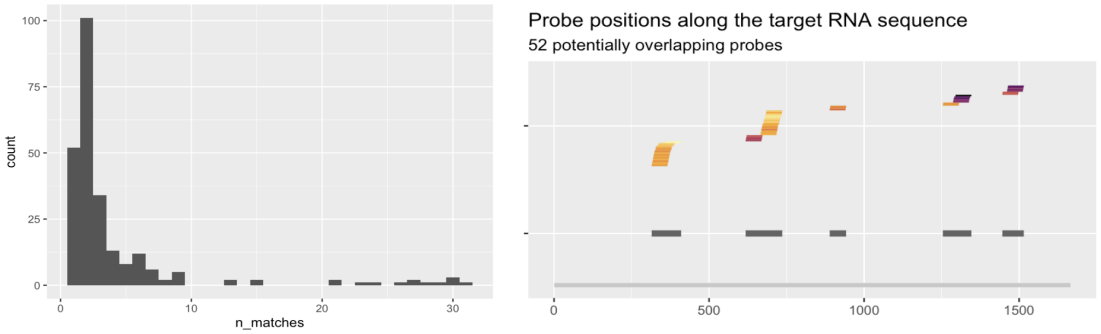

Figure 5: BLAST screen outputs.

5. Distribute and select probes

In this section, we select the best combination of candidate probes that went through the two rounds of filtering. As the candidate probes were initially generated via sliding windows, we flatten the overlapping regions covered by the filtered probes (Fig. 5). Next, we optimise the probe combination that yields the maximum number of non-overlapping probes. To do this, the following script will iteratively distribute probes to minimise the sum of deltaG deviations in each flattened region while fitting as many probes as possible.

Here, `probe_spacing` parameter can be set to define minimum nucleotide distance between probes. A value of 3 is suggested for this parameter. The code will output the final set of probes (Fig. 6). A minimum set of 5 non-overlapping probes is recommended. If fewer than 5 probes are outputted in the final set, then the user can try more lenient filtering criteria in the above sections. The resulting HCR signal will be brighter with more split-pairs, however a maximum of 33 pairs are recommended for cost purposes. Ordering less than 13 pairs should be done with IDT (or equivalent) 96-well plate format, while more than 13 pairs can be ordered using IDT oPools service at fixed cost (max 33 pairs for the cheapest oPools configuration).

```
## ---- Set params
probe_spacing <- 3
max_probe_pairs <- 33 # 33 is max pair count for IDT oPools cheap configuration

## ---- Get final non-overlapping screened candidate probes
candidate_probes_final <- distribute_overlapping_probes(candidate_probes_screened, merge_df,
```

```
probe_spacing)
candidate_probes_final <- cull_excess_pairs(candidate_probes_final, max_probe_pairs)

## ---- Inspect
plot_inspection(candidate_probes_final, "Tm") +
  geom_linerange(data = merge_df,
      aes(x = "", ymin = start, ymax = end),
      colour = "gray40", size = 2,
      inherit.aes = FALSE)
```

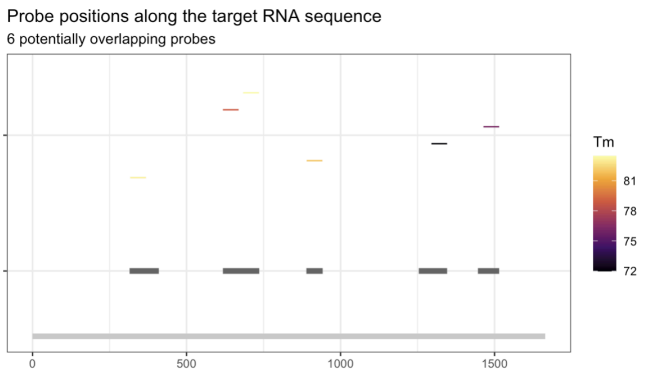

Figure 6: DeltaG-optimised non-overlapping probes

6. Attach HCR-B initiator sequences

The 52-mers in the final probe set are split into 25-nt halves and the chosen HCR-B initiator sequences are attached to the corresponding halves. The centre two nucleotides are removed. 12 HCR-B initiators (B1-5, B7, B9-10, B13-15, B17) are available in the current version of the designer, and the user should select the initiator based on their multi-colour experimental design.

- B1-5, Choi 2014 ACS Nano
- B7, B9-10, B13-15, B17, Y Wang 2020 BioRxiv

```
## ---- Choose hairpin for multiplexing
b_identifier <- "B1"

## ---- Attach hairpin sequences and save
probe_details <- attach_hcr_initiatior(candidate_probes_final, b_identifier = b_identifier)
probes <- get_final_probe_sequences(probe_details)
```

7. Save probe sequences and other outputs

In this section, the final split-probe sets and other associated design configurations are exported (Fig. 7). The code creates a folder in the output directory with the name of the probe set and the date of creation. In the output directory, five files will be created (see below for details). The '...probes.csv' file contains the initiator-attached split-probe sequences in a table format, which can be copy-pasted to a typical oligo ordering sheet. The oligos can be ordered at minimum synthesis scales (~25 nmol) with standard desalting purifications. For reproducibility, every filtering parameter used for the design and the raw BLAST output are exported in the output directory.

```
probes.txt          : final probe sequences to order
probes.pdf          : placing of probes along the target RNA sequence
details.csv         : thermodynamics and other calculated values of the final probes
```

```
params.txt          : parameters used for the design
rawblastoutput.csv  : Raw blast output of candidate probes
```

Only define the output directory (`output_dir`) if not using the RProject (when absolute paths are used).

```
## ---- Create output directory
output_dir <- paste0("./output/", Sys.Date(), "_", target_name, "_", "HCR", b_identifier, "/")
if(!dir.exists(output_dir)){dir.create(output_dir)}

## ---- Export outputs
export_outputs(output_dir, probe_details, probes, blast_output)

## ---- A plot showing plot distribution along target RNA sequence
plot_final_probes(probe_details, "Tm")
ggsave(paste0(output_dir, target_name, "_", "HCR", b_identifier, "_probes.pdf"),
       width = 7, height = 4)
```

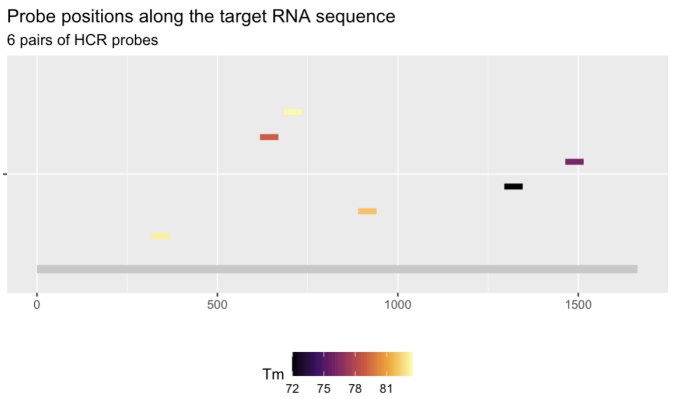

Figure 7: Final set of HCR probes.

Acknowledgements

We thank Tatjana Sauka-Spengler for helpful discussion to construct the probe design algorithm. We are grateful to Julia Olivares Abril, Tai Chaiamarit, and Jana Joha for testing out the initial version of the designer.

References

Acheampong, K.K. *et al.* (2022) 'Subcellular Detection of SARS-CoV-2 RNA in Human Tissue Reveals Distinct Localization in Alveolar Type 2 Pneumocytes and Alveolar Macrophages', *mBio*, 13(1), pp. e03751-21. Available at: https://doi.org/10.1128/mbio.03751-21.

Choi, H.M.T. *et al.* (2018) 'Third-generation in situ hybridization chain reaction: multiplexed, quantitative, sensitive, versatile, robust', *Development*, 145(12), p. dev165753. Available at: https://doi.org/10.1242/dev.165753.

Choi, H.M.T., Beck, V.A. and Pierce, N.A. (2014) 'Next-Generation in Situ Hybridization Chain Reaction: Higher Gain, Lower Cost, Greater Durability', *ACS Nano*, 8(5), pp. 4284–4294. Available at: https://doi.org/10.1021/nn405717p.

Dardani, I. *et al.* (2022) 'ClampFISH 2.0 enables rapid, scalable amplified RNA detection in situ', *Nature Methods*, 19(11), pp. 1403–1410. Available at: https://doi.org/10.1038/s41592-022-01653-6.

Raj, A. *et al.* (2008) 'Imaging individual mRNA molecules using multiple singly labeled probes', *Nat Methods*. 2008/09/23 edn, 5(10), pp. 877–9. Available at: https://doi.org/10.1038/nmeth.1253.

Sullivan, D.K. *et al.* (2023) 'kallisto, bustools, and kb-python for quantifying bulk, single-cell, and single-nucleus RNA-seq', *bioRxiv: The Preprint Server for Biology*, p. 2023.11.21.568164. Available at: https://doi.org/10.1101/2023.11.21.568164.

Thompson, M.K., Kiourlappou, M. and Davis, I. (2020) 'Ribo-Pop: simple, cost-effective, and widely applicable ribosomal RNA depletion', *RNA*, 26(11), pp. 1731–1742. Available at: https://doi.org/10.1261/rna.076562.120.

Tsanov, N. *et al.* (2016) 'smiFISH and FISH-quant – a flexible single RNA detection approach with super-resolution capability', *Nucleic Acids Research*, 44(22), p. e165. Available at: https://doi.org/10.1093/nar/gkw784.

Yang, L. *et al.* (2017) 'Single molecule fluorescence in situ hybridisation for quantitating post-transcriptional regulation in Drosophila brains', *Methods*. 2017/06/28 edn, 126, pp. 166–176. Available at: https://doi.org/10.1016/j.ymeth.2017.06.025.