Supplemental Protocol 4

iPS2-seq design and analysis with catcheR

This Supplemental Protocol describes the following procedures:

1. Overview
2. Installation
3. Oligonucleotides design
4. Pooled cloning step 1 plasmid QC
5. Pooled cloning step 2 plasmid QC
6. Generation of cell-by-gene matrices
7. iPS2-10X-seq perturbation deconvolution
8. iPS2-sci-seq perturbation deconvolution
9. Barcode reassignment

Overview

catcheR is a comprehensive bioinformatic package for designing and analyzing iPS2-seq experiments. It comprises the following functions (Figure SP4.1):

1. *catcheR_design*, which designs oligonucleotides for Supplemental Protocol 1 - Design shRNA oligonucleotides, facilitating shRNA library cloning
2. *catcheR_step1QC*, which analyses the results of Supplemental Protocol 1 - Intermediate plasmid pool QC, assessing pooled cloning step 1 plasmids for barcode swaps
3. *catcheR_step2QC*, which analyses the results of Supplemental Protocol 1 - Final plasmid pool QC, assessing pooled cloning step 2 for shRNA representation
4. *catcheR_scicount*, which analyses 2-level indexing sci-RNA-seq data, facilitating the generation of gene expression matrix for iPS2-sci-seq experiments
5. *catcheR_scicatch*, which assigns shRNA perturbations to single nuclei transcriptomes obtained by Supplemental Protocol 2, enabling the primary analysis of iPS2-sci-seq
6. *catcheR_10Xcatch*, which assigns shRNA perturbations to single cell transcriptomes obtained by Supplemental Protocol 3, enabling the primary analysis of iPS2-10X-seq
7. *catcheR_scicatchQC* and *catcheR_10XcatchQC*, which use the outputs of *catcheR_ scicatch* and *catcheR_10Xcatch*, respectively, to fine-tune shRNA assignment thresholds
8. *catcheR_filtercatch*, which leverages on the output of *catcheR_scicatchQC* and *catcheR_10XcatchQC* to filter single nuclei/cell transcriptomes expressing a single shRNA
9. *catcheR_sortcatch*, which quality controls the cell-by-gene matrix based on the results of *catcheR_step1QC*, reassigning hPSC clones with barcode swaps to the correct shRNA
10. *catcheR_scinocatch* and *catcheR_10Xnocatch*, which identify cells expressing no shRNA in iPS2-sci-seq and iPS2-10X-seq experiments, respectively, adding them to the cell-by-gene matrix to be used as additional controls

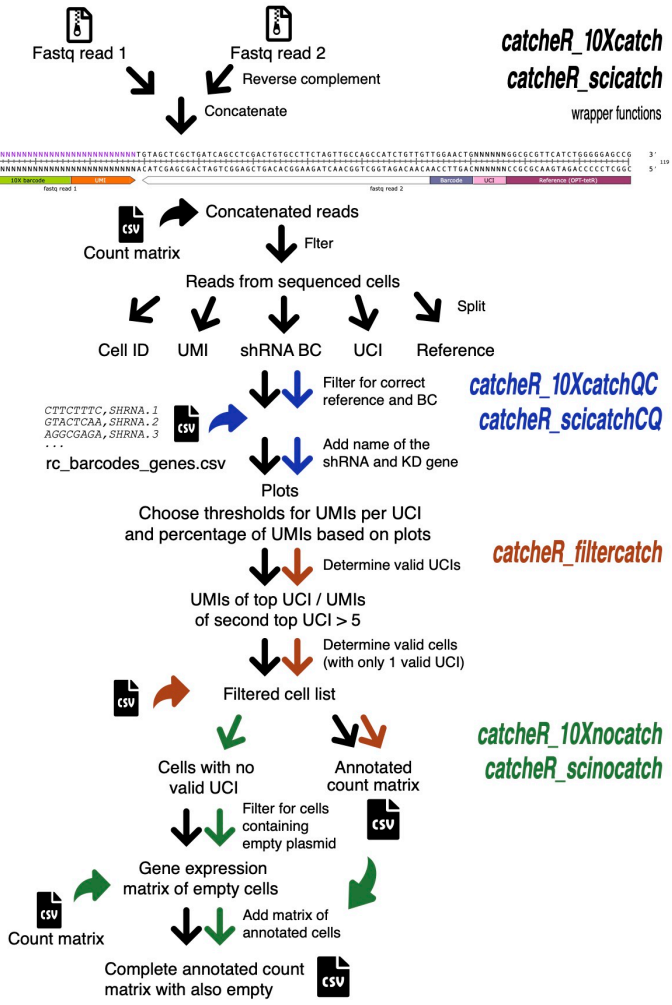


Figure SP4.1. Overview of catcheR
Logical relationship and main inputs/outputs for the main functions used for the catcheR analytical pipeline of iPS2-seq experiments. Functions used for oligo design and plasmid QC are not illustrated.

Installation

catcheR is available at https://github.com/alessandro-bertero/catcheR. The GitHub repository folder "scripts" contains all the bash and R scripts that can be run independently. However, in order to ensure reproducible analyses, we strongly recommend to install catcheR package from GitHub, since its functions run all the analysis inside Docker containers.

1. Install Docker engine, following the instructions at https://docs.docker.com/engine/install/
2. Install catcheR
   (a) In R (≥ 3.0.2) install *devtools*, if not already present

```
install.packages("devtools")
```

   (b) Install catcheR from GitHub

```
install_github("alessandro-bertero/catcheR")
```

   (c) Load catcheR in your R environment

```
library(catcheR)
```

Oligonucleotides design

This steps complements Supplemental Protocol 1 - Design shRNA oligonucleotides

1. In a new working folder, prepare the following files:

   (a) A comma-separated values (csv) file with three columns listing: (1) the forward oligos from the TRC shRNA library; (2) the corresponding barcodes (BC); (3) the shRNA names. Below is an example for the shRNA described in Figure SP1.1:

   ```
   CCGGCAAGTACTCCTTGCTGGATTGCTCGAGCAATCCAGCAAGGAGTACTTGTTTTTG,CAGTTCCA,SMAD2.1
   ...
   ```

   (b) (Optional) - A txt file with a newline-separated list of 5'-3' restriction sites, or other sequences, to be avoided in the shRNAs. By default these are SalI, SwaI and AscI:

   ```
   GTCGAC
   ATTTAAAT
   GGCGCGCC
   ```

2. Run *catcheR_design*:

   ```
   catcheR_design(
       group=c("docker","sudo"),
       folder,
       sequences,
       gibson.five = "AGTTCCCTATCAGTGATAGAGATCCC",
       gibson.three = "GTAGCTCGCTGATCAGC",
       fixed = "GTCGACATTTAAATGGCGCGCC",
       restriction.sites = NULL)
   ```

   catcheR_design arguments:

   (a) group: string with two options: sudo or docker, depending on the user group. For a detailed explanation of Docker user groups, see this page

   (b) folder: string with the working folder path

   (c) sequences: string with the csv file name from step 1a

   (d) gibson.five: (optional) string with the 5' Gibson homology

   (e) gibson.three: (optional) string with the 3' Gibson homology

   (f) fixed: (optional) character string with the multicloning site

   (g) restriction.sites: (optional) string with the txt file name from 1b

   Example usage:

   ```
   catcheR_design(
       group = "docker",
       folder = "path/to/folder",
       sequences = "filename.csv",
       restriction.sites = "filename.txt")
   ```

   catcheR_design outputs:

   (a) "output.txt", with the oligo sequences to be used for synthesis (Figure SP1.1)

   (b) "bad_oligos.txt", with the shRNAs with forbidden restriction enzyme sites (highlighted in lowercase characters); this information can be used to refine the shRNA list
   Example output for "bad_oligos.txt":

   ```
   AGTTCCCTATCAGTGATAGAGATCCCGGACATAATCACTGCGTAATCCTCagatctTACGCAGTGATTATGTCCTTTTTTTGT-
   CGACATTTAAATGGCGCGCCNNNNNNGCTGAAGAGTAGCTCGCTGATCAGC,GATA4
   ...
   ```

Pooled cloning step 1 plasmid QC

This steps complements Supplemental Protocol 1 - Intermediate plasmid pool QC

1. In a new working folder, prepare the following files:
   (a) Fastq/fq or fastq.gz files with demultiplexed read 1 from the NGS run
   (b) A tab-separated txt file with the shRNA names and their full sequences

```
SMAD2.1 GCAAGTACTCCTTGCTGGATTGCTCGAGCAATCCAGCAAGGAGTACTTG
...
```

   (c) (Optional) - A txt file with a newline-separated list of clones of interest as "BC_UCI"

```
CAAGAGCC_CATCGT
...
```

2. Run *catcheR_step1QC*:

```
catcheR_step1QC(
    group=c("docker","sudo"),
    folder,
    fastq.read1,
    DIs = 100,
    ratio = 10,
    plot.threshold = 2000,
    clones = NULL)
```

   catcheR_step1QC arguments:

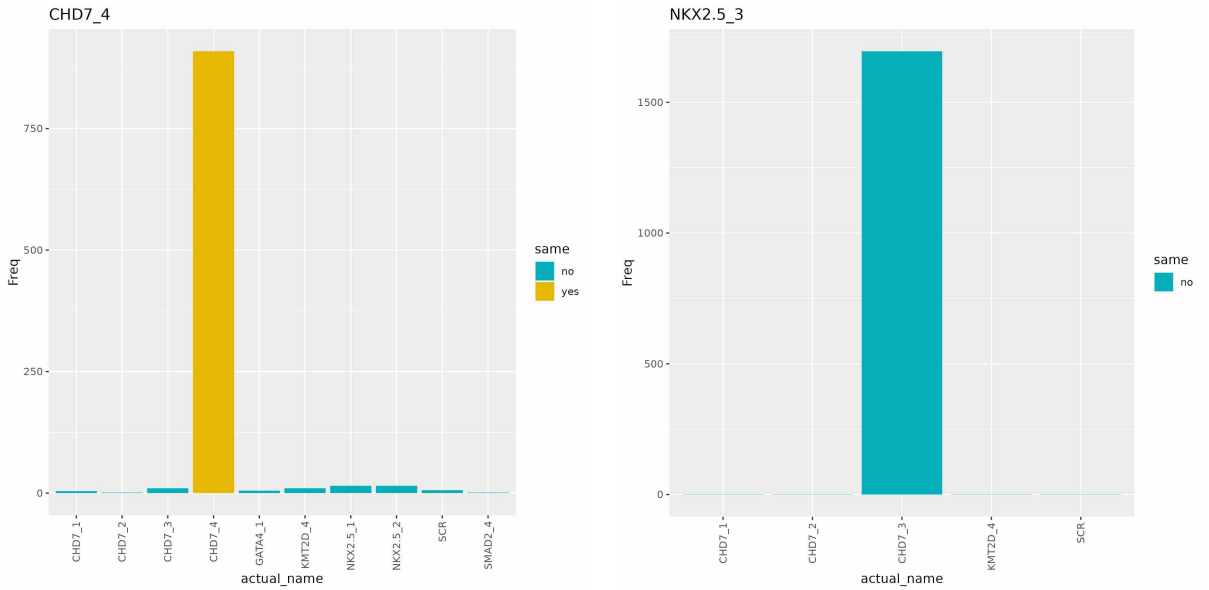
   (a) group: string with two options: sudo or docker, depending on the user group (info)
   (b) folder: string with the working folder path
   (c) fastq.read1: string with the read 1 filename from step 1a
   (d) DIs: integer of the minimum number of diversity indexes (DIs, pseudo-unique reads) of the most represented shRNA matching to a given UCI-BC; in combination with "ratio", it selects UCI-BCs for which it is possible to reliably assign an shRNA
   (e) ratio: integer of the minimum ratio between the number of DIs of the most represented and second most represented shRNAs matching to a given UCI-BC
   (f) plot.threshold: integer of the minimum number of DIs per UCI-BC for output 2b
   (g) clones: (optional) a string with the txt file from step 1b

   Example usage:

```
catcheR_step1QC(
    group = "docker",
    folder = "path/to/folder",
    fastq.read1 = "filename.fq",
    clones = "filename.txt")
```

   catcheR_step1QC key outputs:

   (a) "reliable_clones_swaps.txt", which lists UCI-BCs with reliable evidence of shRNA-barcode swap, and is used as input for Barcode reassignment
   (b) Bar chart of the number of DIs for each clone above "plot.threshold"
   (c) Bar chart of the number of DIs associated to each BCs, shRNAs, and reliable swaps
   (d) (Optional) - Text files and bar charts with the number of DIs for each shRNA matching to each clone of interest (Figure SP4.2)



(A) (B)

Figure SP4.2. Exemplary quantification of DIs for bacterial clones carrying the expected shRNA (A) or with robust evidence of a barcode swap that can be reassigned (B; Figure 1E). These graphs allow visual evaluation of appropriate thresholds for ”DI” and ”ratio”.

Pooled cloning step 2 plasmid QC

This steps complements Supplemental Protocol 1 - Final plasmid pool QC

1. In a new working folder, prepare the following files:
   (a) Fastq/fq or fastq.gz files with demultiplexed read 1 from the NGS run
   (b) ”rc_barcodes_genes.csv”, a csv file with two columns: (1) the shRNA BCs; (2) the matching shRNA names in the format ”GENE.shRNAID”

```
CAAGAGCC,SMAD2.1
...
```

   (c) (Optional) - A txt file with clones of interest (step 1a of the previous section)

2. Run *catcheR_step2QC*:

```
catcheR_step2QC(
    group=c("docker","sudo"),
    folder,
    fastq.read1,
    DIs = 1000,
    clones = NULL)
```

catcheR_step2QC arguments:

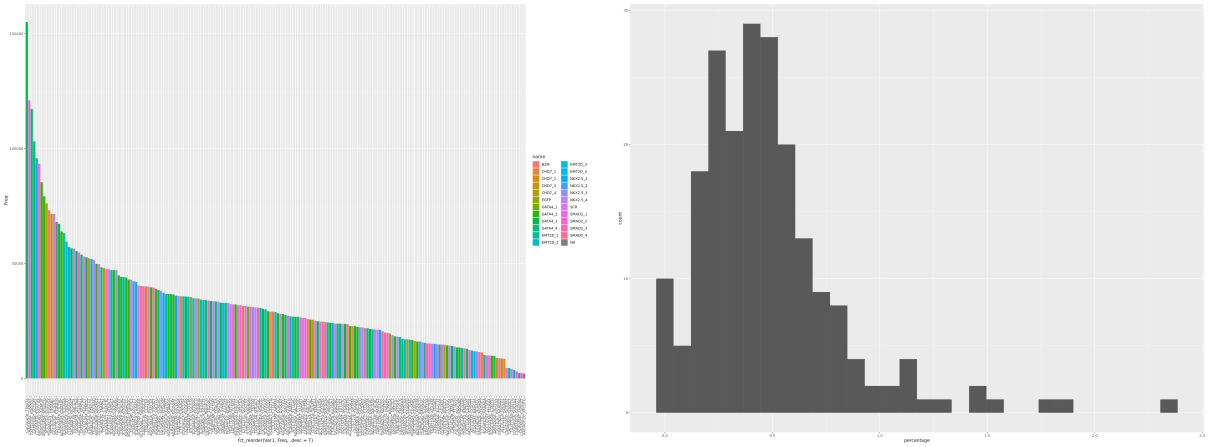
   (a) group: string with two options: sudo or docker, depending on the user group (info)
   (b) folder: string with the working folder path
   (c) fastq.read1: string with the read 1 file from step 1a
   (d) DIs: integer of the minimum number of DIs for a given UCI-BC; it selects reliably-measured UCI-BCs
   (e) clones: (optional) a string with the txt file from step 1c

Example usage:

```
catcheR_step2QC(
    group= "docker",
    folder = "path/to/working/folder",
    fastq.read1 = "filename.fq",
    clones = "filename.txt")
```

catcheR_step2QC key outputs:

(a) Pie charts of DIs per shRNA and per gene targets associated to each clone
(b) Text file listing all clones above the DI threshold
(c) Bar chart of the number of DIs for each clone above the DI threshold (Figure SP4.3A)
(d) Frequency histogram of the percentage of clones above the DI threshold for each shRNA (Figure SP4.3B)



(A) **(B)**

Figure SP4.3. Exemplary quantification of DIs for bacterial clones represented by more than 1.000 DIs (A), and of the frequency of bacterial clones for each shRNA. These graphs allow to confirm whether shRNAs are normally distributed in the final plasmid pool

Generation of cell-by-gene matrices

Obtain iPS2-10X-seq count matrices

1. Download and install cellranger from the 10X Genomics download resource page; alternatively, a Docker container with cellranger 7 can be pulled from our Docker repository
2. Demultiplex Illumina base calls with *cellranger mkfastq*, following 10X Genomics' manual. In the simple CSV sample sheet, include the 10X Genomics indexes used to generate the GEX, CMO, and UCI-BC libraries in Supplemental Protocol 3
3. Obtain cell-by-gene count matrices using *cellranger count* (single sample) or *cellranger multi* (multiplexed experiment).

   Note: count matrices from multiple samples from the same experiment can be aggregated with *cellranger aggr*, but keep the individual count matrices as these are required to run *catcheR_10Xcatch* (which requires GEX and UCI-BC fastq files to have the same library ID to match UCI-BC and cells). Our GitHub repository contains R scripts that can be used to subsequently integrate the outputs of *catcheR_10Xcatch* and *cellranger aggr*
4. Run *cell ranger matrix2csv* to transform the sparse count matrix in a dense csv matrix

After obtaining the count matrix, proceed to iPS2-10X-seq perturbation deconvolution.

Obtain iPS2-sci-seq count matrices

1. *catcheR_scicount* is a wrapper of the ”bbi-sci” pipeline developed by the Brotman Baty Institute for Precision Medicine, which was implemented and dockerized in catcheR to be used on any operating system.

2. Demultiplex Illumina base calls to fastq files

   (a) Create a ”SampleSheet.csv” file with as many sample rows as the PCR wells from Indexing PCR and NGS of Supplemental Protocol 2, where ”Sample_ID” is the well identified with format [A-H][01-12], and index and index2 are the i7 and i5 indexes used in the corresponding row and column (refer to Table SP2.1)

   (b) Run Illumina *bcl2fastq* following Illumina’s manual

   (c) Run fastQC to confirm the quality of the fastq files

3. In a new working folder:

   (a) Create the subfolder ”fastq”, and copy all the demultiplexed fastq.gz files. Ensure that all file names begin with the well coordinate (e.g. A01)

   (b) Create a tab separated txt file called ”sci-RNA-seq-8.RT.oligos” with the association between RT wells and RT barcode sequences (refer to Table SP2.1)

   ```
   A01 TTCTCGCATG
   ...
   ```

   (c) Create the subfolder ”GENOMES”, and copy the annotated genome (i.e., GRCh38)

4. Run *catcheR_scicount*:

   ```
   catcheR_scicount(
       group=c("docker","sudo"),
       folder,
       sample.name,
       UMI.cutoff)
   ```

   catcheR_scicount arguments:

   (a) group: string with two options: sudo or docker, depending on the user group (info)

   (b) folder: string with the working folder path

   (c) sample.name: string with the name of the experiment

   (d) UMI.cutoff: integer of the minimum number of UMI per nucleus needed to consider the single cell transcriptome valid

   Example usage:

   ```
   catcheR_scicount(
       group = "docker",
       folder = "path/to/file",
       sample.name = "experiment",
       UMI.cutoff = 500)
   ```

   catcheR_scicount outputs: cell-by-gene count matrix (sparse), knee-plot of UMI per cell (Figure S2F), and statistics files, which are all placed in the ”final-output” folder

5. Run *catcheR_scicsv* using the ”final-output” folder from *catcheR_scicount* as its only input (a folder containing the files ”gene.annotations”, ”cell.annotations” and ”UMI.count.matrix”):

   ```
   catcheR_scicsv(
       group=c("docker","sudo"),
       folder)
   ```

   .

   Output files are ”exp_mat.csv” and ”exp_mat_no0.csv” (filtered to exclude genes with 0 counts in all cells)

After obtaining the count matrix, proceed to iPS2-sci-seq perturbation deconvolution.

iPS2-10X-seq perturbation deconvolution

This section describes a typical analysis pipeline starting with *catcheR_ 10Xcatch* to perform a full analysis with automatic thresholds, after which *catcheR_10XcatchQC* can fine-tune the thresholds to re-filter cells with *catcheR_filtercatch*. Lastly, *catcheR_10Xnocatch* can add cells expressing no shRNA as additional controls to the final annotated cell-by-gene count matrix.

1. In a new working folder:
   (a) Copy the fastq/fq or fastq.gz files with demultiplexed read 1 and read 2 of the UCI-BC library (step 2 of Obtain iPS2-10X-seq count matrices)
   (b) Copy the cell-by-gene count matrix in csv format (step 4 of Obtain iPS2-10X-seq count matrices)
   (c) Create a csv file named “rc_barcodes_genes.csv” with two columns: (1) the shRNA BCs; (2) the matching shRNA names in the format ”GENE.shRNAID”

```
CAAGAGCC,SMAD2.1
...
```

2. Run *catcheR_10Xcatch* to execute the complete analysis with automatic thresholding:

```
catcheR_10Xcatch(
    group=c("docker","sudo"),
    folder,
    fastq.read1,
    fastq.read2,
    expression.matrix,
    reference = "GGCGCGTTCATCTGGGGGAGCCG",
    UCI.length = 6,
    threads = 2,
    percentage = 15,
    mode = "bimodal")
```

   catcheR_10Xcatch arguments:
   (a) group: string with two options: sudo or docker, depending on the user group (info)
   (b) folder: string with the working folder path
   (c) fastq.read1: string with the filename of read 1 fastq/fq or fastq.gz (step 1a)
   (d) fastq.read2: string with the filename of read 2 fastq/fq or fastq.gz (step 1a)
   (e) expression.matrix: string with the filename of the count matrix file csv (step 1b)
   (f) reference: string with the reverse complement of the sequence before the shRNA BC at the start of read 2 (default is the 3’ end of the OPTtetR cDNA; Figure S3F)
   (g) UCI.length: integer of the UCI lenght (default is 6)
   (h) threads: integer of the threads to be used for parallelization (default is 2)
   (i) percentage: integer of the percentage of UMIs supporting a given UCI over the total UMIs supporting all UCIs in a given cell; it is used as threshold to consider the UCI valid (Figure Figure SP4.4D); the recommended default is 15
   (j) mode: string with two options: bimodal or noise, defining the mode for automatic thresholding the minimum number of UMIs per UCI to consider the UCI valid.
      i. ”bimodal” (default) sets the threshold at the valley of the bimodal UMIxUCI distribution (Figure SP4.4C)
      ii. ”noise” sets the threshold at 1.35 * the number of UCIs supported by a single UMI; used if the UMIxUCI distribution is not bimodal

Example usage:

```
catcheR_10Xcatch(group = "docker",
    folder = "path/to/folder",
    fastq.read1 = "R1.fastq.gz",
    fastq.read2 = "R2.fastq.gz",
    expression.matrix = "filename.csv",
    threads = 12)
```

catcheR_10Xcatch key outputs (all inside the working folder):

(a) "log.txt" with the starting number of reads

(b) "log2.txt" with the number of cells, UMIs, UCIs, and the calculated thresholds of UMI per UCI for both "bimodal" and "noise" (only one is chosen based on the "mode" argument; the other one is provided for a potential reanalysis)

(c) Bar charts of UMI counts per shRNA and target gene (Figure SP4.4A and Figure SP4.4B)

(d) Frequency histograms of UCIs supported by a certain number of UMIs (UMIxUCI), to interpret the signal/noise of the experiment and possibly set a custom UMIxUCI threshold for subsequent reanalysis (Figure SP4.4C).

Note: here and below, copies of the same UCI expressed by different cells are plotted and analyzed separately

(e) Frequency histogram of UCIs supported by a certain fraction of UMIs over the total number of UMIs supporting all UCIs in a given cell (UMIpercentagexUCI), to further assess signal/noise and possibly adjust the default threshold (Figure SP4.4D)

(f) Dot plots that combine the UMIpercentagexUCI and UMIxUCI data on the x and y axes, respectively, with each dot representing one or more UCI (quantified by the dot size). Dot colors indicate either the number of valid UCIs (i.e., shRNAs) in the cell containing a given UCI (Figure SP4.4E), or whether said UCI is the only valid one in the relevant cell and is thus assigned to such cell (Figure SP4.4F)

(g) "log_part3.txt" with how many cells were assigned to a single shRNA or were filtered out due to zero or multiple shRNAs

(h) "silencing_matrix.csv" is the key output used for the secondary analyses: the cell-by-gene count matrix provided as input, filtered and annotated with the shRNA encoded in each cell. Cells names are modified as follows:

`cellID_UMIxUCI_BC_GENE_UCI`

Where:

i. cellID is the original cell name (i.e., the 16 bp of the 10X RT barcode)
ii. UMIsxUCI is the number of UMI associated with that perturbation
iii. BC is the shRNA barcode
iv. GENE is the shRNA target
v. UCI is the Unique Clonal Identifier (shared by all cells originating from the same hPSC clone)

Example of cell name after the analysis:

`TTCTAACCACAGTCGC_180_CGTGATGC_NKX2.5_ACAGTG`

This annotation can be leveraged with the scripts available on our GitHub repository to enable a variety of secondary analyses described in the manuscript, or using custom scripts for other types of secondary analyses

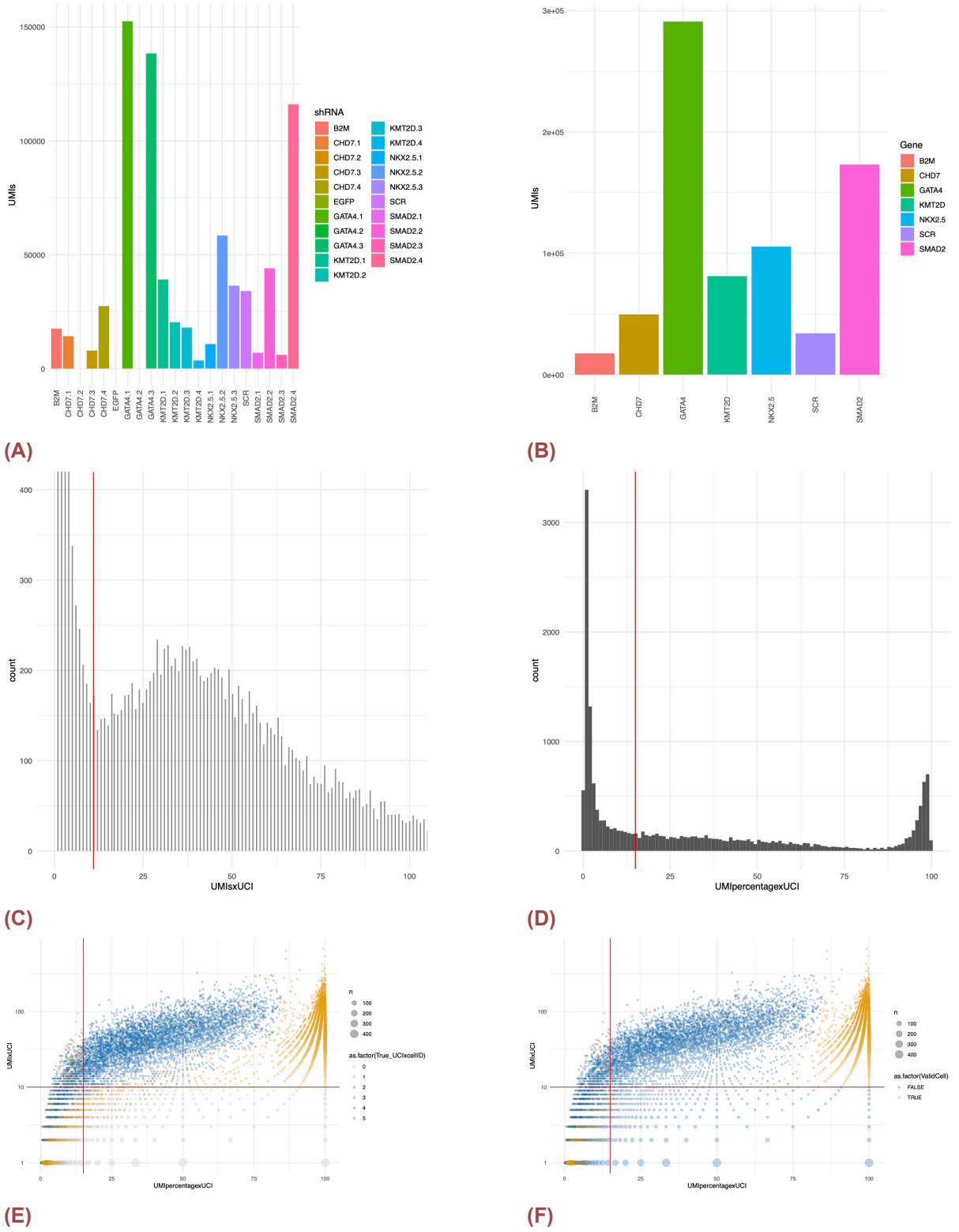


Figure SP4.4. Exemplary key outputs of a *catcheR_10Xcatch* analysis: ”barcode_distribution.pdf” (A); ”gene_distribution.pdf” (B); ”UMIxUCI_400_100.pdf” with the automatic UMIxUCI threshold based on ”bimodal” (C); ”percentage_of_UMIxUCI_dist.pdf” with the default 15% ”percentage” treshold (D); ”2D_percentage_of_UMIxUCI_UMI_count_trueorfalse.pdf” (E); and ”2D_percentage_of_UMIxUCI_UMI_ValidCells.pdf” (F)

OPTIONAL: fine tune iPS2-10X-seq perturbation assignment

1. In the same working folder used to run *catcheR_10Xcatch*, run *catcheR_10XcatchQC*, which uses the txt file outputs to regenerate the quality control plots described in Figure SP4.4 and suggest new thresholds to enable subsequent filtering

```
catcheR_10XcatchQC(
    group=c("docker","sudo"),
    folder,
    reference = "GGCGCGTTCATCTGGGGGAGCCG",
    mode = "bimodal")
```

Example usage:

```
catcheR_10XcatchQC(
    group = "docker",
    folder = "path/to/working/folder",
    mode = "noise")
```

catcheR_10XcatchQC arguments and outputs are the same as *catcheR_10Xcatch*

2. In the same working folder used to run *catcheR_10Xcatch*, run *catcheR_filtercatch*, which uses the output of *catcheR_10Xcatch* to filter cells based on custom thresholds

```
catcheR_filtercatch(
    group=c("docker","sudo"),
    folder,
    expression.matrix,
    UMI.count,
    percentage = 15)
```

Example usage:

```
catcheR_filtercatch(group = "docker",
    folder = "path/to/working/folder",
    expression.matrix = "filename.csv",
    UMI.count = 5)
```

catcheR_10XcatchQC argument *UMI.count* is an integer of the custom UMI threshold; the other arguments and the outputs are the same as *catcheR_10Xcatch*. ;

OPTIONAL: identify cells expressing no shRNA in iPS2-10X-seq

1. In the same working folder used to run *catcheR_10Xcatch*, run *catcheR_10Xnocatch*:

```
catcheR_10Xnocatch(
    group=c("docker","sudo"),
    folder,
    expression.matrix,
    threshold)
```

Example usage:

```
    catcheR_10Xnocatch(group = "docker",
    folder = "path/to/folder",
    expression.matrix = "filename.csv",
    threshold = 5)
```

catcheR_10Xnocatch argument *threshold* is an integer of the minimal UMI count for the ”empty” reference Figure SP4.3B needed to confidently identify it as having integrated said plasmid; all other arguments are the same as *catcheR_10Xcatch*

catcheR_10Xnocatch output is an updated ”silencing_matrix.csv” in which names of empty cells are modified as ”cellID_?_empty_NA_empty”.

iPS2-sci-seq perturbation deconvolution

This section describes a typical analysis pipeline similar to the one described in the previous section, but leveraging on the *catcheR_scicatch*, *catcheR_scicatchQC*, *catcheR_filtercatch*, and *catcheR_scinocatch* functions.

1. In a new working folder:

   (a) Create a subfolder called ”fastq”, and copy all the demultiplexed fastq.gz files (step 2 of Obtain iPS2-sci-seq count matrices). Ensure that all file names begin with the well coordinate (e.g. A01)

   (b) Copy the cell-by-gene expression matrix csv file obtained with *catcheR_scicount* function (step 4 of Obtain iPS2-sci-seq count matrices).

   (c) Create a csv file called “rc_barcodes_genes.csv” with two columns: (1) the shRNA BCs; (2) the matching shRNA names in the format ”GENE.shRNAID”

   ```
   CAAGAGCC,SMAD2 .1
   ...
   ```

   (d) Copy the text file ”sci-RNA-seq-8.RT.oligos” used by *catcheR_scicount* (step 3b of Obtain iPS2-sci-seq count matrices).

2. Run *catcheR_scicatch* to execute the complete analysis with automatic tresholding; the arguments are the same as for *catcheR_10Xcatch* (step 2 of iPS2-10X-seq perturbation deconvolution), except that no filenames are provided since fastq files must be in the ”fastq” subfolder in the working directory:

   ```
   catcheR_scicatch(
       group=c("docker","sudo"),
       folder,
       expression.matrix,
       reference = "GGCGCGTTCATCTGGGGGAGCCG",
       UCI.length = 6,
       threads = 2,
       percentage = 15,
       mode = "bimodal")
   ```

   Example usage:

   ```
   catcheR_scicatch(group = "docker",
       folder = "path/to/working/folder",
       expression.matrix = "filename.csv",
       threads = 12)
   ```

   catcheR_scicatch key outputs are the same as for *catcheR_10Xcatch* (step 2 of Generation of cell-by-gene matrices and Figure SP4.3), except that in ”silencing_matrix.csv” ”cell ID” indicates the PCR well and RT barcode ID (which can be leveraged to identify nuclei from different samples pooled on the same RT plate)

   Example of cell name after the analysis:

   ```
   P24__RT_27_7_GCCTGTGT_SCR_ACGGTC
   ```

   In addition, *catcheR_scicatch* provides two quality control outputs to evaluate experimental biases during sci-RNAseq library preparation: *demux* and *RT* detail about how many cells were identified from each row and column of the PCR and RT plates, respectively.

OPTIONAL: fine tune iPS2-sci-seq perturbation assignment

1. Run *catcheR_scicatchQC* following the steps described for *catcheR_10XcatchQC* in OPTIONAL: fine tune iPS2-10X-seq perturbation assignment: the functions share the same arguments and outputs, but *catcheR_scicatchQC* used the output of *catcheR_scicatch*

   Example usage:

```
catcheR_scicatchQC(
    group = "docker",
    folder = "path/to/working/folder",
    mode = "noise")
```

2. Run catcheR_filtercatch function as described in in OPTIONAL: fine tune iPS2-10X-seq perturbation assignment: this function also works on the output of *catcheR_scicatch* or *catcheR_scicatchQC*

OPTIONAL: identify cells expressing no shRNA in iPS2-sci-seq

1. Run *catcheR_scinocatch* following the steps described for *catcheR_10Xnocatch* in OPTIONAL: identify cells expressing no shRNA in iPS2-10X-seq: the functions share the same arguments and outputs, but *catcheR_scinocatch* uses the output of *catcheR_scicatch*

   Example usage:

```
catcheR_scinocatch(group = "docker",
    folder = "path/to/working/folder",
    expression.matrix = "filename.csv",
    threshold = 5)
```

Barcode reassignment

catcheR_sortcatch is an optional function that corrects the annotated cell-by-gene count matrix obtained with *catcheR_10Xcatch* or *catcheR_scicatch* reassigning the perturbation of any cell belonging to a hPSC clone with reliable evidence of a shRNA-barcode swap, based on the results of *catcheR_step1QC*.

1. In a new working folder:

   (a) Copy the annotated count matrix (i.e., ”silencing_matrix.csv”)

   (b) Copy the txt file with the list of UCI-BCs with reliable evidence of a shRNA-barcode swap (i.e. ”reliable_clones_swaps.txt”; step 2a of Pooled cloning step 1 plasmid QC)

2. Run *catcheR_sortcatch*:

```
catcheR_sortcatch(
    group=c("sudo","docker"),
    folder,
    expression.matrix,
    swaps)
```

   Example usage:

```
catcheR_sortcatch(
    group = "docker",
    folder = "path/to/working/folder",
    expression.matrix = "silencing_matrix.csv",
    swaps = "reliable_clones_swaps.txt")
```

catcheR_sortcatch arguments:

(a) group: string with two options: sudo or docker, depending on the user group (info)

(b) folder: string with the working folder path

(c) expression.matrix: string with the filename of the annotated count matrix csv

(d) swaps: a character string with the filename of the txt file (step 1b)

catcheR_sortcatch output is an updated annotated gene expression matrix csv file called ”silencing_matrix_updated.csv”, in which ”BC” and ”GENE” have been modified to reflect the actual shRNA encoded in each cell