Inferring the MoA of drugs based on barcode fingerprints.

Bastien CHASSAGNOL and Vera PANCALDI and Luca GRUMULATO and Marie-Pier SCOTT KELLY

2025-05-03

Contents

# Welcome

This is the online Quarto book project for reproducing: **Inferring the mechanism of action of new drugs through the analysis of the predetermined heterogeneous response to treatment of different subpopulations of cancer cells** poster, presented at NetBioMed 2022 conference.

## Introduction: Cell lines DNA Barcoding

* DNA barcoding is also used for inferring the species abundancies in environement samples, just replace the notion of species by cell lines. Similar statistical issues, with *zero-inflated* distributions.
* Steps:
  1. Transfection by virus.
  2. Clonal amplification, unique tagging per cell using MOI.
  3. Clone sizes are assumed to be proportional to the barcode abundances due to this 1-1 mapping of a barcode and a single cell.
* Pros DNA barcoding:
  + Better capture of cell population sizes
  + Better tracking of tagged clones.
* Cons DNA barcoding:
  + Lack of systematic reviews and benchmarks.
  + Mostly rely on bulk RNASeq analytical tools, not accounting for *drop-outs*. In particular, the assumptions that variance across tags is homogeneous, and abundancies follow a negative binomial distribution are quite controversial.

# 1. Data management: metadata correction and cleansing

## 1.1 Reproducibility

I list below the R packages required to reproduce the analyses.

## data wrangling  
library(dplyr)  
library(tidyr)  
library(stringr)  
library(readr)  
library(readxl)  
library(purrr)  
  
## reporting  
library(flextable)  
  
## plotting  
library(ggplot2)  
library(ComplexHeatmap)  
library(cowplot)  
library(grid)  
library(RColorBrewer)  
library(igraph)  
  
## auxiliary functions  
source("R/utils.R")  
today\_date <- "2025-04-28"  
## today\_date <- format(Sys.Date(), "%Y-%m-%d")  
  
## set the seed, for fixing generation of ComplexHeatmaps (dendogram clustering)  
set.seed(20)

* [Original Google Drive repository](https://drive.google.com/drive/folders/1pMSX4M4kHcDGxcsCzsMYHHM6VRvBHHUO).

## 1.2 Barcode counts cleaning

We early discarded exp130921\_in vivo.csv from the experiences

|  |
| --- |
| Listing 1.1: Update labelling of latest experience.  exp271221 <- readr::read\_csv("data/barcode-counts/exp271221.csv")  exp271221\_whole <- readr::read\_csv2("old-data-backup/data-raw/barcode-counts/exp211221cor.csv") |>   rename(barcode\_id = 1) selected\_dataset <- compare\_dataframes(exp271221, exp271221\_whole)  file.rename("data/barcode-counts/exp211221.csv", "data/barcode-counts/exp271221.csv") |

In Listing 1.1, we rename expression file exp211221.csv into exp271221.csv, relying on date annotations.

While the dose-response experience exp200921\_dose response osim exhibits distinct biological objective, it turned out that the 4 control biological replicates are shared with classical compound response comparison exp200921.csv. However, the number of sequenced barcode IDs differ in both experiences!! In addition, note that exp200921.csv has been tagged as discarded, except for the Controls!!

Code snippets in [Listing 1.2](#lst-281022) and [Listing 1.3](#lst-281022-time-course) aim at replacing unprecise labelling of barcode IDs (no concentration, no time duration, …) in the 2022 batch experiences.

|  |
| --- |
| Listing 1.2: Update labelling of latest experience.  ## extract old colnames exp281022\_expression <- readr::read\_csv("data/barcode-counts/exp281022.csv")  ## cat(colnames(exp281022\_expression), file = "281022\_old\_labels.txt", sep = "\n")  ## extract mapping old\_names with new\_names mapping\_old\_new\_281022 <- readxl::read\_excel("data/Table of compounds\_whole\_2025-04-28.xlsx",   sheet = "Sample Mapping 281022")  labels\_old\_new\_281022 <- setNames(object = mapping\_old\_new\_281022$OLD\_replicate\_label,   nm = mapping\_old\_new\_281022$NEW\_replicate\_label)  ## apply change labelling exp281022\_expression <- exp281022\_expression |>   dplyr::rename(dplyr::all\_of(labels\_old\_new\_281022))  readr::write\_csv(exp281022\_expression, "data/barcode-counts/exp281022.csv") |

|  |
| --- |
| Listing 1.3: Update time course experiences.  exp281022\_expression\_timecourse <- readr::read\_csv("data/barcode-counts/exp281022\_time course.csv")  ## cat(colnames(exp281022\_expression\_timecourse), file = "281022\_timecourse\_old\_labels.txt", sep = "\n")  ## update colnames ----  mapping\_old\_new\_time\_course <- read\_excel("data/Table of compounds\_whole\_2025-04-28.xlsx",   sheet = "Sample Mapping Time Course")  labels\_old\_new\_time\_course <- setNames(object = mapping\_old\_new\_time\_course$OLD\_replicate\_label,   nm = mapping\_old\_new\_time\_course$NEW\_replicate\_label)  ## apply change labelling exp281022\_expression\_timecourse <- exp281022\_expression\_timecourse |>   dplyr::rename(dplyr::all\_of(labels\_old\_new\_time\_course))  readr::write\_csv(exp281022\_expression\_timecourse,  "data/barcode-counts/exp281022\_time course.csv")  ## To evaluate whether controls are different between experiment 281022 and 281022 Time Course controls\_exp281022 <- exp281022\_expression |>   select(key\_barcode, dplyr::matches("^ctl")) |>   dplyr::inner\_join(exp281022\_expression\_timecourse |>   select(key\_barcode, dplyr::matches("^Ctrl")),  by="key\_barcode") cor(controls\_exp281022$ctl2, controls\_exp281022$Ctrl2) ## they strongly correlate with each other, but not a perfect match -> controls are indeed distinct |

### 1.2.1 TODO: barcode counts inconsistencies

Things to consider:

* Format used is French csv, instead of international CSV. CSV means for ‘comma-separated value’, maybe consider switching Excel settings.
* I rename the first colname of each expression file as barcode\_id, storing unequivocal tag.
* Some barcode counts exhibit final line index, for which reason?

### 1.2.2 Overview of barcode counts

We report in [Table 1.1](#tbl-overview-barcode-counts) a summary of the barcode counts (File location, batch\_id, and dimensions of the counts matrix) considered for the generation of the Heatmaps, and the construction of a drug-drug network.

## 1) summarise kept batches ----  
barcode\_files <- list.files("./data/barcode-counts/",  
 pattern = "exp.\*\\.csv",  
 full.names = TRUE)  
  
kept\_barcode\_summaries <- purrr::map(barcode\_files, function(filename) {  
 barcode\_counts <- readr::read\_csv(filename, show\_col\_types = FALSE)  
 experience\_name <- filename |> basename() |> tools::file\_path\_sans\_ext()  
 experience\_summary <- tibble::tibble(`Experience Name`= experience\_name,  
 Features= paste0("Barcode matrix contains: ",  
 nrow(barcode\_counts),  
 " unique barcode IDs, and ",  
 ncol(barcode\_counts)-1, " replicates."))  
 return(experience\_summary)  
}) |>  
 purrr::list\_rbind()  
flextable::flextable(kept\_barcode\_summaries) |>   
 bold(part="header")  
  
## 2) summarise discarded batches ----  
discarded\_files <- list.files("./data/barcode-counts/temp discarded/",  
 pattern = "exp.\*\\.csv",  
 full.names = TRUE)  
  
## change format + barcode name  
barcode\_discarded\_summaries <- purrr::map(discarded\_files, function(filename) {  
 barcode\_counts <- readr::read\_csv(filename, show\_col\_types = FALSE)  
 experience\_name <- filename |> basename() |> tools::file\_path\_sans\_ext()  
 experience\_summary <- tibble::tibble(`Experience Name`= experience\_name,  
 Features= paste0("Barcode matrix contains: ",  
 nrow(barcode\_counts),  
 " unique barcode IDs, and ",  
 ncol(barcode\_counts)-1, " replicates."))  
 return(experience\_summary)  
}) |>  
 purrr::list\_rbind()  
  
flextable::flextable(barcode\_discarded\_summaries) |>   
 bold(part="header")

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | | **Experience Name** | **Features** | | --- | --- | | exp010821 | Barcode matrix contains: 869714 unique barcode IDs, and 16 replicates. | | exp040821 | Barcode matrix contains: 869714 unique barcode IDs, and 16 replicates. | | exp070222 | Barcode matrix contains: 93132 unique barcode IDs, and 68 replicates. | | exp130921 | Barcode matrix contains: 627539 unique barcode IDs, and 24 replicates. | | exp151121 | Barcode matrix contains: 844434 unique barcode IDs, and 60 replicates. | | exp181021 | Barcode matrix contains: 813579 unique barcode IDs, and 59 replicates. | | exp200921 | Barcode matrix contains: 627539 unique barcode IDs, and 40 replicates. | | exp200921\_dose response osim | Barcode matrix contains: 1067031 unique barcode IDs, and 20 replicates. | | exp220322 | Barcode matrix contains: 93132 unique barcode IDs, and 66 replicates. | | exp271221 | Barcode matrix contains: 844434 unique barcode IDs, and 80 replicates. | | exp281022 | Barcode matrix contains: 315642 unique barcode IDs, and 22 replicates. | | exp281022\_time course | Barcode matrix contains: 398004 unique barcode IDs, and 51 replicates. | | exp300821 | Barcode matrix contains: 413335 unique barcode IDs, and 28 replicates. |   (a) Kept Batches | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | | **Experience Name** | **Features** | | --- | --- | | exp130921\_in vivo | Barcode matrix contains: 813579 unique barcode IDs, and 16 replicates. | | exp200921\_dose response osim | Barcode matrix contains: 813579 unique barcode IDs, and 20 replicates. |   (b) Removed batches | |

Table 1.1: Overview of the Barcoding Batches.

From [Table 1.1](#tbl-overview-barcode-counts), 13 barcode experiences are included in the analysis, whereas 2 batches/experiences are filtered out.

## 1.3 Phenotype metadata cleaning

* General question: what’s the difference between Run Date and Experiment Date?

Most of the R instructions reported in this section aim at rendering the Table of compounds file compliant with the tidy format, see [Tip 1.1](#tip-tidyformat). These data-wrangling operations are split in 4 steps:

1. In [Listing 1.4](#lst-process-pheno1), we homogenise Concentrations, Date and Duration to ISO standards, while dealing with erroneous missing values generated by fused Excel cells. Besides, we removed special, non-ascii characters that could not be processed by visualisation plots, such as NF−κB.
2. In [Listing 1.5](#lst-process-pheno2), we retrieve from each individual barcode counts profile, the individual replicates ID identifying unequivocally each replicate. We set apart replicates that could be mapped back to original Table of compounds file.
3. In [Listing 1.6](#lst-process-pheno3), we merge together in a comprehensive phenotype table the short Samples\_ID (short term referring to drugs listed in Compound column), the full Compound (complete name of drugs) and map each of them to its complete list of replicates (from 2 to 8). We save the output in data-derived.
4. Finally, in [Listing 1.7](#lst-check-pheno-data-1) and [Listing 1.8](#lst-check-pheno-data-2), we perform several posterior *data integrity checks* to verify if i) we were able to map each individual replicate ID to its original ID in Table of compounds, and ii) if the number of replicates reported in [Table of compounds](data/Table%20of%20compounds_whole_2025-04-28.xlsx) was consistent with the number of barcode profiles.

|  |
| --- |
| Listing 1.4: Coerce original Excel file reporting experimental design to tidy format.  pheno\_colnames <- c(Pathway = "Pathways", MoA = "...2",   Compound = "Samples", Replicates = "Replicates",  Concentrations = "Concentrations", Date = "Experiment date",  `Run date` = "Run date", Duration="Duration",   Kept="Kept", Comments = "Comments")  ## prune irrelevant colnames ---- ## data/Table of compounds\_stringent\_2025-04-18.xlsx pheno\_data <- readxl::read\_xlsx("data/Table of compounds\_whole\_2025-04-28.xlsx",  sheet = "Experimental Design",  col\_types = c(rep("text", 3), "numeric",   rep("text", 4), "logical", "text")) |>  dplyr::rename(dplyr::all\_of(pheno\_colnames)) |>  tidyr::fill(Pathway, MoA, .direction = "down")  pheno\_data\_discarded <- pheno\_data |>   dplyr::filter(!Kept)  ## Add batches and correct for fused Excel cells ----  pheno\_data <- pheno\_data |>   ## extract last 6 numbers  dplyr::mutate(Compound = if\_else(grepl("Control", MoA, ignore.case = TRUE),  MoA, Compound),  `Run date` = stringr::str\_sub(`Run date`, -6, -1),  Batch\_ID= paste0("exp", Date)) |>   tidyr::fill(Compound, .direction = "down") |>  ## remove unwanted replicates  dplyr::filter(Kept) |>   dplyr::mutate(Batch\_ID = dplyr::if\_else(Comments %in% c("Dose Response"),   "exp200921\_dose response osim", Batch\_ID)) |>   dplyr::mutate(Batch\_ID = dplyr::if\_else(Comments %in% c("Time Course"),   "exp281022\_time course", Batch\_ID),   Pathway = if\_else(is.na(Pathway), MoA, enc\_to\_ascii(Pathway))) |>   dplyr::select(- Comments, -Kept) |>   dplyr::distinct(.keep\_all = TRUE) ## remove duplicates  ## format concentrations, creating both 'Concentrations\_ID' and 'Concentrations' in Moles ---- pheno\_data <- pheno\_data |>   dplyr::mutate(Concentrations=if\_else(Compound=="Control", "000 uM", Concentrations)) |>   dplyr::mutate(Concentrations=if\_else(Compound=="Osimertinb+sorafenib", "015 uM", Concentrations)) |>  tidyr::separate\_wider\_delim(Concentrations, delim = " ", names = c("Concentrations Value", "Unit"), cols\_remove = FALSE) |>   dplyr::rename(OLD\_Concentrations=Concentrations) |>   dplyr::mutate(Unit = case\_when(  Unit %in% c("uM", "µM", "µg/ml") ~ "u",  Unit == "nM" ~ "n",   Unit %in% c("mg/kg", "mM") ~ "m",   )) |>   dplyr::mutate(Concentrations=parse\_number(`Concentrations Value`, trim\_ws = TRUE,   locale = locale(decimal\_mark = ",", grouping\_mark = "."))) |>   dplyr::mutate(Concentrations = case\_when(  Unit == "u" ~ Concentrations \*10^-6,  Unit == "n" ~ Concentrations \*10^-9,   Unit == "m" ~ Concentrations \*10^-3)) |>   dplyr::mutate(`Concentrations Value`=parse\_number(`Concentrations Value`, trim\_ws = TRUE,   locale = locale(decimal\_mark = ",", grouping\_mark = ".")) |>   format\_concentrations()) |>   tidyr::unite(col ="Concentrations\_ID", `Concentrations Value`, Unit, sep = "")  ## format Durations, creating both 'Duration\_ID' and 'Duration' as an integer value in Days ---- pheno\_data <- pheno\_data |>   ## extract last 6 numbers, as the only ones used for identification of samples downstream  dplyr::mutate(Duration\_ID = Duration,  Duration = parse\_number(Duration, trim\_ws = TRUE,   locale = locale(decimal\_mark = ".")),   Duration = dplyr::if\_else(grepl("m$", Duration\_ID),  Duration\*30, Duration)) |

|  |
| --- |
| Metadata inconsistencies and Conversion to UIC Units |
| * In Run date, when there was a range instead of a fixed Date, I kept only the end of the interval, as used for labelling colnames in barcode counts. * Homogenise units for the Concentrations colname: convert everything to moles. * Homogenise units for Date and Run date, using international ISO 8601 format: YYYY-MM-DD |

In [Listing 1.5](#lst-process-pheno2) and [Listing 1.6](#lst-process-pheno3), we identify which raw barcode counts matrices avalaible in folder barcode-counts are not reported in the global experimental dataset, and reciprocally. Finally, we only **keep experiences that are both reported in Table of compounds and present in barcode-counts folder.**.

|  |
| --- |
| Listing 1.5: Identify shared experience IDs between experimental design table (Table of compounds), and barcode-count counts profiles.  ## keep only relevant files  barcodes\_filenames <- list.files("./data/barcode-counts/",  pattern = "^exp.\*\\.csv$")  barcodes\_IDs <- readxl::read\_xlsx("data/Table of compounds\_whole\_2025-04-28.xlsx",  sheet="Batch Mapping") |>   select(Filename, Batch\_ID) |>   filter(!Batch\_ID %in% c("exp130921\_in vivo")) |>   mutate(Filename =paste0(Filename, ".csv"),   Date = stringr::str\_extract(Batch\_ID, "(?<=^exp)[[:digit:]]{6}")) |

To generate the final phenotype dataset, we only keep experiences that were reported both in Table of Compound and folder barcode-counts in [Listing 1.6](#lst-process-pheno3). Please ensure that Drugs Mapping sheet in Table of compounds is correct.

|  |
| --- |
| Listing 1.6: Map each compound to its full list of replicates. Generate a Batch\_ID to unequivocally ientify each run of experiences, and convert all run Dates to their international format.  ## step 1) keep only shared Batch\_ID experiences ---- barcodes\_IDs <- barcodes\_IDs |>  dplyr::semi\_join(pheno\_data, by="Batch\_ID") ## exp201021 is discarded, that's normal, we just want to keep the controls!! pheno\_data <- pheno\_data |>  dplyr::semi\_join(barcodes\_IDs, by="Batch\_ID")  ## step 2) extract individual replicate IDs directly from sample experiences ---- ## Filename <- "exp281022.csv"; Batch\_ID <- "exp281022"; Date <- "281022" ID\_mapping <- purrr::pmap(barcodes\_IDs, function(Filename, Batch\_ID, Date) {  ## build path  counts\_path <- file.path("data","barcode-counts",   paste0(Filename))    ## extract replicate names (reading the header)  replicates\_ID <- strsplit(readLines(counts\_path, n = 1), split = ",")[[1]] |>  str\_subset("\\S") |> ## remove empty strings and 'key'  str\_subset("barcode\_id", negate = TRUE)   ## ID: combination of letters, numbers and -, followed by '[1-4]\_', starting the sample's name  dataset\_ID <- tibble::tibble(Batch\_ID = Batch\_ID,   Date = stringr::str\_extract(replicates\_ID,  "(?<=\_exp)[[:alnum:]]{6}"),  Replicates\_ID = replicates\_ID,  `Run date` = stringr::str\_extract(replicates\_ID,  "(?<=\_run)[[:alnum:]]{6}(?=\_)"),  Concentrations\_ID = stringr::str\_extract(replicates\_ID,   "(?<=[1-8]\_)[[:digit:]]{3}[gmnux]{1}(?=\_exp)")) |>   ## account for second notation syntax for concentrations  dplyr::mutate(Concentrations\_ID = if\_else(is.na(Concentrations\_ID),   stringr::str\_extract(replicates\_ID,  "(?<=[1-8]\_)[0-9]{1}\\.[0-9]{2}[gmnux]{1}(?=\_exp)"),  Concentrations\_ID))    ## Deal with two time notations  if (grepl("Time",Batch\_ID, ignore.case = TRUE)) {  dataset\_ID <- dataset\_ID |>   dplyr::mutate (Samples\_ID = stringr::str\_extract(replicates\_ID,  "(?<=\_)[[[:alnum:]]\\-]+(?=[1-8]\_)"),   Duration\_ID=stringr::str\_extract(replicates\_ID,  "^[[[:alnum:]]\\.]{1,3}[dm](?=\_)"))  }  else {  dataset\_ID <- dataset\_ID |>   dplyr::mutate (Samples\_ID = stringr::str\_extract(replicates\_ID,  "^[[[:alnum:]]\\-]+(?=[1-8]{1}\_)"),   Duration\_ID="9d")  }    return(dataset\_ID) }) |>  purrr::list\_rbind()  ## deal with specific p42 and p43 ID\_mapping <- ID\_mapping |>   dplyr::mutate(Samples\_ID = if\_else(grepl("^p42", Replicates\_ID), "p42", Samples\_ID),   Samples\_ID = if\_else(grepl("^p43", Replicates\_ID), "p43", Samples\_ID))  ## remove irrelavant samples ID\_mapping <- ID\_mapping |>   dplyr::filter(!grepl("^p42|^p43|^CTRL", Replicates\_ID))   ## detect miscatech samples ## ID\_mapping\_missing <- ID\_mapping |> ## filter(if\_any(everything(), is.na))  ## step 3) map short compound IDs with full compound names ---- mapping\_compounds <- readxl::read\_xlsx("data/Table of compounds\_whole\_2025-04-28.xlsx",  sheet = "Drugs Mapping") ID\_mapping <- ID\_mapping |>   inner\_join(mapping\_compounds, by = "Samples\_ID") ## test <- ID\_mapping |> anti\_join(mapping\_compounds, by = "Samples\_ID") ## test2 <- mapping\_compounds |> anti\_join(ID\_mapping, by = "Samples\_ID") |

Finally, in [Listing 1.7](#lst-check-pheno-data-1), we map all replicates (one column in a given barcode count matrix) to its comprehensive metadata descripion (stored in Tables of compounds), using as **primary and foreign keys** the following 6 variables: Batch\_ID, Compound, Duration\_ID, Run date, Date, and Concentrations\_ID.

|  |
| --- |
| Listing 1.7: Extract discarded replicates  replicates\_discarded <- dplyr::anti\_join(ID\_mapping, pheno\_data,  by = c("Batch\_ID", "Compound", "Duration\_ID",  "Run date", "Date", "Concentrations\_ID"))  pheno\_data\_unmapped <- dplyr::anti\_join(pheno\_data,  ID\_mapping,  by = c("Batch\_ID", "Compound", "Duration\_ID",  "Run date", "Date", "Concentrations\_ID"))  pheno\_data\_unmapped |>   arrange(Compound, Date, Concentrations\_ID) |>   flextable() |>   bold(part = "header") |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Pathway** | **MoA** | **Compound** | **Replicates** | **Concentrations\_ID** | **OLD\_Concentrations** | **Date** | **Run date** | **Duration** | **Batch\_ID** | **Concentrations** | **Duration\_ID** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |   Table 1.2: Identify set of experiences reported in Table of compounds that could not be mapped against their corresponding Barcode counts replicates. We check this information with an anti\_join between original *phenotype* and *counts*, returning samples from Table of Compound that could not have been mapped back. |

In [Table 1.3](#tbl-check-pheno-data-2), we check discrepancies between the number of barcode replicates reported in Table of compounds with the number of replicates avalaible in barcode counts matrices. And it turned out that compound: XAV-939, Date: 220322, is reported with 4 replicates, while only 2 could be found in exp220322 barcode count matrix

|  |
| --- |
| Listing 1.8: Secound round of data wrangling quality controls on *counts* versus *phenotype data*, focusing on divergent number of replicates.  replicate\_inconsistencies <- pheno\_data |>   dplyr::inner\_join(ID\_mapping,  by = c("Batch\_ID", "Date",   "Compound", "Run date",  "Concentrations\_ID", "Duration\_ID")) |>  group\_by(Batch\_ID, Compound, Date, Replicates, Concentrations\_ID,Duration\_ID) |>  summarise(n=n()) |>  filter(n!=Replicates)  flextable(replicate\_inconsistencies) |>  autofit() |>  bold(part = "header") |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Batch\_ID** | **Compound** | **Date** | **Replicates** | **Concentrations\_ID** | **Duration\_ID** | **n** | | --- | --- | --- | --- | --- | --- | --- | | exp220322 | XAV-939 | 220322 | 4 | 080u | 9d | 2 |   Table 1.3: Secound round of data wrangling quality controls on *counts* versus *phenotype data*, focusing on divergent number of replicates. |

### 1.3.1 Save phenotype Metadata

By applying all the formatting analyses reported in [Section 1.3](#sec-metadata-phenotypes), we generate in [Table 1.4](#tbl-phenotype-metadata-saving) a global metadata phenotypic table, adhering to tidy principles [Tip 1.1](#tip-tidyformat).

|  |
| --- |
| Tip 1.1: Tidy Data Format (Key Principles) |
| 1. **Each variable has its own column**. 2. **Each observation has its own row** – Every row corresponds to one observation. 3. **Each value has its own cell** – Each cell contains a single, unique value.   This format streamlines data wrangling, and generally speaking, data analysis and visualisation. In other words, prefer simpler CSV formats for the experimental design, and avoid *cell fusion* in Excel documents. Finally, for the formatted tabular representation in documentations, I use flextable. |

|  |
| --- |
| Listing 1.9: Save the final metadata annotations for barcodes, using Today’s date for historical versioning.  ## step 4) Use international date formats ---- pheno\_data\_formatted <- pheno\_data |>   dplyr::inner\_join(ID\_mapping,  by = c("Batch\_ID", "Date",   "Compound", "Run date",  "Concentrations\_ID", "Duration\_ID")) |>   dplyr::select(Batch\_ID, Pathway, MoA,  Compound, Samples\_ID,   Replicates, Replicates\_ID,  Concentrations, Concentrations\_ID,   Date, `Run date`, Duration, Duration\_ID) |>   ## convert to ISO 1860 Date format  mutate(Date = lubridate::dmy(Date) |> format(),  `Run date`= lubridate::dmy(`Run date`),   ## artifically de-duplicate controls for exp, simplify the analytical workflow   Replicates\_ID = dplyr::if\_else(Batch\_ID %in% c("exp200921\_dose response osim") &  Compound=="Control",  gsub("exp200921\_", "exp200921\_dose\_response\_", Replicates\_ID),  Replicates\_ID))  ## 2) QC: guarantee uniqueness of the replicates pheno\_data\_deduplicated <- pheno\_data\_formatted |>   dplyr::distinct(Replicates\_ID, .keep\_all = TRUE) all.equal(pheno\_data\_deduplicated, pheno\_data\_formatted) ## [1] TRUE  ## 3) save as .xlsx file discarded and preserved replicates  barcode\_counts\_summaries <- list("kept counts" = kept\_barcode\_summaries,  "kept compounds" = pheno\_data,  "kept replicates" = pheno\_data\_formatted |>   select(Compound, Replicates\_ID),  "discarded counts" = barcode\_discarded\_summaries,   "discarded compounds"=pheno\_data\_discarded,   "discarded replicates" = replicates\_discarded)  openxlsx::write.xlsx(x = barcode\_counts\_summaries,  file = paste0("./data/logs/barcode\_metadata\_overview\_",  today\_date,".xlsx"),  overwrite = TRUE, asTable = TRUE)   ## 4) save the tidy phenotype metatada table ---- readr::write\_csv(pheno\_data\_formatted,  file = paste0("data/pheno\_data\_metadata\_", today\_date,".csv"))  flextable::flextable(head(pheno\_data\_formatted)) |>   autofit() |>   bold(part="header") |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Batch\_ID** | **Pathway** | **MoA** | **Compound** | **Samples\_ID** | **Replicates** | **Replicates\_ID** | **Concentrations** | **Concentrations\_ID** | **Date** | **Run date** | **Duration** | **Duration\_ID** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | exp010821 | Control | Control | Control | Contro | 4 | Contro1\_000u\_exp010821\_run180821\_03 | 0 | 000u | 2021-08-01 | 2021-08-18 | 9 | 9d | | exp010821 | Control | Control | Control | Contro | 4 | Contro2\_000u\_exp010821\_run180821\_04 | 0 | 000u | 2021-08-01 | 2021-08-18 | 9 | 9d | | exp010821 | Control | Control | Control | Contro | 4 | Contro3\_000u\_exp010821\_run180821\_05 | 0 | 000u | 2021-08-01 | 2021-08-18 | 9 | 9d | | exp010821 | Control | Control | Control | Contro | 4 | Contro4\_000u\_exp010821\_run180821\_06 | 0 | 000u | 2021-08-01 | 2021-08-18 | 9 | 9d | | exp040821 | Control | Control | Control | Contro | 4 | Contro1\_000u\_exp040821\_run180821\_25 | 0 | 000u | 2021-08-04 | 2021-08-18 | 9 | 9d | | exp040821 | Control | Control | Control | Contro | 4 | Contro2\_000u\_exp040821\_run180821\_26 | 0 | 000u | 2021-08-04 | 2021-08-18 | 9 | 9d |   Table 1.4 |

**Conclusion**: on a total of 544, 532 replicates are kept, and 12 replicates were removed.

# 2. Methods: normalise, identify DRBs and save as SummarizedExperiment objects

## 2.1 Reproducibility

I list below the R packages required to reproduce the analyses.

## data wrangling  
library(dplyr)  
library(tidyr)  
library(stringr)  
library(readr)  
library(readxl)  
library(openxlsx)  
library(purrr)  
  
## reporting  
library(flextable)  
  
# automated package linting  
library(xml2)  
library(downlit)  
  
## auxiliary functions  
source("R/utils.R")  
today\_date <- "2025-04-28"  
## today\_date <- format(Sys.Date(), "%Y-%m-%d")  
  
## set the seed, for fixing generation of ComplexHeatmaps (dendogram clustering)  
set.seed(20)

## 2.2 Preprocessing barcode counts

**Code colour**: Luca’s protocol, and Bastien’s comments and perspectives

### 2.2.1 **Background Noise Removal** and SummarizedExperiment stantardised storage

Eliminate barcodes for which the combined counts of the 4 controls per barcode are below a given threshold, here 5.

|  |
| --- |
| Listing 2.1: Remove background noise.  barcode\_files <- list.files("./data/barcode-counts/",  pattern = "exp.\*\\.csv",  full.names = TRUE) thresh\_background <- 4 ## ./data/pheno\_data\_metadata\_2025-04-18.csv pheno\_data <- readr::read\_csv(paste0("./data/pheno\_data\_metadata\_",  today\_date, ".csv"))  ## filename <- grep( "exp200921", barcode\_files, value = TRUE)[2] barcode\_counts\_aggregated <- purrr::map(barcode\_files, function(filename) {  barcode\_counts <- readr::read\_csv(filename)  experience\_name <- filename |> basename() |> tools::file\_path\_sans\_ext()    ## 1) explicitly change replicates name to enable 1-1 mapping ----  if (experience\_name=="exp200921\_dose response osim") {  barcode\_counts <- barcode\_counts |>   dplyr::rename\_with(  .fn = ~ gsub("exp200921\_", "exp200921\_dose\_response\_", .x),  .cols = starts\_with("Contro"))  }  replicates\_names <- setdiff(colnames(barcode\_counts), "barcode\_id")    ## 2) Remove lowly expressed barcodes ----  control\_index <- pheno\_data |>   dplyr::filter(Batch\_ID==experience\_name &   Compound=="Control" &  Replicates\_ID %in% replicates\_names) |>   dplyr::pull(Replicates\_ID)  signicant\_barcodes\_index <- which(rowSums(barcode\_counts[, control\_index]) > thresh\_background)  filtered\_barcode\_counts <- barcode\_counts[signicant\_barcodes\_index,]    message(paste("\n\nWe are considering experiment:", experience\_name, "with", nrow(filtered\_barcode\_counts), "barcode IDs kept.\n\n"))    return(filtered\_barcode\_counts) }) |>   purrr::reduce(~ inner\_join(.x, .y, by = "barcode\_id"))  ## Deal with specific duplicated colnames, resulting from control duplicates ## barcode\_counts\_aggregated <- barcode\_counts\_aggregated |> ## dplyr::select(!ends\_with(".y")) |>  ## dplyr::rename\_with(.fn = ~ gsub("\\.x$", "", .x), ## .cols = dplyr::ends\_with(".x"))  ## 3) keep only replicates present in pheno\_data barcode\_counts\_aggregated <- barcode\_counts\_aggregated |>   select(all\_of(c("barcode\_id", pheno\_data$Replicates\_ID))) |

* After filtering for background noise, and merging all experiences based on shared barcode IDs, 4629 unique barcode IDs are kept, for 532 samples, see [Listing 2.1](#lst-background-filtering) for details.
* Density plot of controls + justify why + check absence of outliers, to evaluate the relevance of this threshold, HTSFilter, comparison with existing filtering approaches.

### 2.2.2 **Normalisation**

* Normalize barcodes so that the total number of counts per sample is equal to 100,000. This operation is performed in [Listing 2.2](#lst-cpm-normalisation).

|  |
| --- |
| Listing 2.2: Normalise by .  ## 1) normalise by total number of counts ---- barcode\_counts\_normalised <- barcode\_counts\_aggregated |>   mutate(across(  .cols = where(is.numeric),   .fns = ~ (.x / sum(.x)) \* 1e5)) |

* Close to two existing normalisation methods: Counts Per Million (CPM) which additionally scales raw counts by total library size and multiplies by 1,000,000 and Total Count Scaling (TCS): Scales raw counts by the total number of reads (or mapped reads) in each sample, then multiplies by a fixed number (e.g., 100,000). Would compare other normalisation approaches + ignore biological or technical biases + generate MA plots for verifying the mean-variance correction trend + not suitable for DEGs analyses. Apply concatenation phase of all samples before normalising. Preprocessing and normalisation functions for transcriptomics: a general overview

### 2.2.3 Prepare SummarizedExperiment objects for normalised data

* Higher level of granularity in [Listing 2.3](#lst-SummarizedExperiment-replicates):

|  |
| --- |
| Listing 2.3: Simply compute the Pearson correlation score at the replicate level, and save the output as a SummarizedExperiment.  ## 2) prepare SummarizedExperiment inputs ---- ## Barcode counts at replicate level barcode\_normalised\_replicates\_mat <- barcode\_counts\_normalised |>   tibble::column\_to\_rownames("barcode\_id") |>   as.matrix()  ## Sample metadata pheno\_data\_replicates <- pheno\_data |>   tibble::column\_to\_rownames("Replicates\_ID")   ## Feature metadata barcode\_metadata <- tibble::tibble(barcode\_id = barcode\_counts\_normalised$barcode\_id) row.names(barcode\_metadata) <- barcode\_metadata$barcode\_id  ## 3) Buid and Save SummarizedExperiment se\_replicates <- SummarizedExperiment::SummarizedExperiment(  assays = list(counts = barcode\_normalised\_replicates\_mat),  rowData = barcode\_metadata,  colData = pheno\_data\_replicates) |

* Medium level of granularity in [Listing 2.4](#X39ca325e8a9dbe44a3fac819b3976f914168448), averaging over replicates:

|  |
| --- |
| Listing 2.4: Simply compute the Pearson correlation score at the Compound per Batch level.  barcode\_counts\_long <- barcode\_counts\_normalised |>   tidyr::pivot\_longer(-barcode\_id,   names\_to = "Replicates\_ID",  values\_to = "Barcode\_Counts") |>   dplyr::inner\_join(pheno\_data, by="Replicates\_ID")  barcode\_counts\_wider\_compound\_batch <- barcode\_counts\_long |>   tidyr::pivot\_wider(id\_cols = c(barcode\_id),   names\_from = c(Batch\_ID, Compound, Concentrations\_ID, Duration\_ID),  names\_sep = ":",  values\_from = Barcode\_Counts,   values\_fn=mean)  pheno\_compound\_batch <- tibble::tibble(original =setdiff(colnames(barcode\_counts\_wider\_compound\_batch),"barcode\_id")) |>  mutate(Batch\_ID = stringr::str\_split\_i(original, ":", 1),  Compound = stringr::str\_split\_i(original, ":", 2),   Concentrations\_ID = stringr::str\_split\_i(original, ":", 3),   Duration\_ID = stringr::str\_split\_i(original, ":", 4)) |>  inner\_join(pheno\_data |> select(Batch\_ID, Pathway, MoA, Compound,   Concentrations, Concentrations\_ID, Duration, Duration\_ID),   by = c("Batch\_ID", "Compound", "Concentrations\_ID", "Duration\_ID")) |>   dplyr::distinct() |>   dplyr::select(-Concentrations\_ID, -Duration\_ID) |>   tibble::column\_to\_rownames("original")   se\_compound\_batch <- SummarizedExperiment::SummarizedExperiment(  assays = list(counts = barcode\_counts\_wider\_compound\_batch |>   tibble::column\_to\_rownames("barcode\_id") |>   as.matrix()),  rowData = barcode\_metadata,  colData = pheno\_compound\_batch) |

* Low level of granularity in [Listing 2.5](#lst-SummarizedExperiment-compound), with one CC per compound:

|  |
| --- |
| Listing 2.5: Compute the Pearson correlation score, after averaging the barcode counts per compound.  barcode\_counts\_wider\_by\_compound <- barcode\_counts\_long |>   tidyr::pivot\_wider(id\_cols = c(barcode\_id),   names\_from = c(Compound),  values\_from = Barcode\_Counts,   values\_fn=mean)  pheno\_compound <- tibble::tibble(Compound = setdiff(colnames(barcode\_counts\_wider\_by\_compound), "barcode\_id")) |>  inner\_join(pheno\_data |> select(Pathway, MoA, Compound),   by = c("Compound")) |>   dplyr::distinct() |>   tibble::column\_to\_rownames("Compound")   se\_compound <- SummarizedExperiment::SummarizedExperiment(  assays = list(counts = barcode\_counts\_wider\_by\_compound |>   tibble::column\_to\_rownames("barcode\_id") |>   as.matrix()),  rowData = barcode\_metadata,  colData = pheno\_compound) |

saveRDS(se\_replicates, file = paste0("./results/compounds/se\_normalised\_per\_replicate\_",  
 today\_date, ".rds"))  
saveRDS(se\_compound\_batch, file = paste0("./results/compounds/se\_normalised\_per\_compound\_by\_batch\_",  
 today\_date, ".rds"))  
saveRDS(se\_compound, file = paste0("./results/compounds/se\_normalised\_per\_compound\_",  
 today\_date, ".rds"))

## 2.3 **Differential analyses** and Binarisation

1. Calculate mean of the 4 controls, followed by Differential Represented Barcode Analysis: binarise each replicate, assigning a 1 if Fold change is above 3 with respect to Mean value, and 0 otherwise

|  |
| --- |
| Listing 2.6: Compute the averaged value for control replicates, then binarise for each compound. A 1 denoting a significant positive enrichment for a given barcode\_id.  ## Step 1: Compute control means per barcode\_id and Batch ---- threshold\_FC <- 3 control\_means <- barcode\_counts\_long |>  filter(Compound == "Control") |>  group\_by(Batch\_ID, barcode\_id) |>  summarise(Control\_Mean = mean(Barcode\_Counts), .groups = "drop")  ## Step 2: Join control mean to full dataset ---- barcode\_discretised\_replicates <- barcode\_counts\_long |>  left\_join(control\_means,   by = c("Batch\_ID","barcode\_id"))  ## Step 3: Compute Fold Change and discretise ---- barcode\_discretised\_replicates <- barcode\_discretised\_replicates |>  filter(Compound != "Control") |>  mutate(Diff = Barcode\_Counts - Control\_Mean,  Barcode\_Counts = if\_else(Diff > threshold\_FC, 1, 0)) |>  ## Final cleanup   select(-Diff, -Control\_Mean) |>   dplyr::mutate(Barcode\_Counts = as.logical(Barcode\_Counts)) |

Batch effect correction for integrating across batches? // As Vera emphasised it out, why keeping only positive values? Negative are also interesting. Why not pairing p-values and fold-change (considering indeed really small sample sizes)? // All these operations can be performed with one run, using lm, and a model as such, Expr∼0+Gene+Batch+Drug, with a contr.treatment design matrix // Perform sensitivity analyses to evaluate the impact of threshold criteriaon for binarisation on downstream analyses. .

### 2.3.1 SummarizedExperiment of binarised markers

* SummarizedExperiment at the replicates level (controls being discarded)

## Step 4: Save as a SummarizedExperiment ----  
barcode\_discretised\_replicates\_mat <- barcode\_discretised\_replicates |>   
 tidyr::pivot\_wider(id\_cols = barcode\_id, names\_from = Replicates\_ID, values\_from = Barcode\_Counts)  
  
pheno\_replicates\_discretised <- pheno\_data |>   
 dplyr::filter(Replicates\_ID %in% barcode\_discretised\_replicates$Replicates\_ID) |>   
 tibble::column\_to\_rownames("Replicates\_ID")   
  
se\_discretised\_replicates <- SummarizedExperiment::SummarizedExperiment(  
 assays = list(binarised = barcode\_discretised\_replicates\_mat |>   
 tibble::column\_to\_rownames("barcode\_id") |>   
 as.matrix()),  
 rowData = barcode\_metadata,  
 colData = pheno\_replicates\_discretised)  
  
saveRDS(se\_discretised\_replicates,  
 file = paste0("./results/compounds/se\_discretised\_per\_replicate\_",  
 today\_date, ".rds"))

* Logical AND over replicates: drug by batch (and concentration and time).

|  |
| --- |
| Listing 2.7: Average the replicates, using a all operator.  ## 2) discretised correlation at the compound by batch level (averaging over replicates), with an all function barcode\_discretised\_compound\_batch <- barcode\_discretised\_replicates |>   tidyr::pivot\_wider(id\_cols = c(barcode\_id),   names\_from = c(Batch\_ID, Compound, Concentrations\_ID, Duration\_ID),  names\_sep = ":",  values\_from = Barcode\_Counts,   values\_fn=all)  pheno\_discretised\_compound\_batch <- pheno\_compound\_batch[setdiff(colnames(barcode\_discretised\_compound\_batch), "barcode\_id"), ]  se\_compound\_batch\_discretised <- SummarizedExperiment::SummarizedExperiment(  assays = list(discretised = barcode\_discretised\_compound\_batch |>   tibble::column\_to\_rownames("barcode\_id") |>   as.matrix()),  rowData = barcode\_metadata,  colData = pheno\_discretised\_compound\_batch)  saveRDS(se\_compound\_batch\_discretised,  file = paste0("./results/compounds/se\_discretised\_per\_compound\_by\_batch\_",  today\_date, ".rds")) |

* Average over compounds.

|  |
| --- |
| Listing 2.8: Aggregate at the compound level, assigning a 1 if barcode cell lines were found positiveliy enriched in all compounds.  ## 2) discretised correlation at the compound by batch level (averaging over replicates), with an all function barcode\_discretised\_compound <- barcode\_discretised\_replicates |>   tidyr::pivot\_wider(id\_cols = c(barcode\_id),   names\_from = c(Compound),  names\_sep = ":",  values\_from = Barcode\_Counts,   values\_fn=all)  pheno\_discretised\_compound<- pheno\_compound[setdiff(colnames(barcode\_discretised\_compound), "barcode\_id"), ]  se\_compound\_discretised <- SummarizedExperiment::SummarizedExperiment(  assays = list(discretised = barcode\_discretised\_compound |>   tibble::column\_to\_rownames("barcode\_id") |>   as.matrix()),  rowData = barcode\_metadata,  colData = pheno\_discretised\_compound)  saveRDS(se\_compound\_discretised,  file = paste0("./results/compounds/se\_discretised\_per\_compound\_",  today\_date, ".rds")) |

# 3. Results: Heatmaps and Drug networks, based on barcode fingerprints

## 3.1 Reproducibility

I list below the R packages required to reproduce the analyses.

## data wrangling  
library(dplyr)  
library(tidyr)  
library(stringr)  
library(purrr)  
  
## reporting  
library(flextable)  
  
## plotting  
library(ggplot2)  
library(ComplexHeatmap)  
library(cowplot)  
library(grid)  
library(RColorBrewer)  
library(igraph)  
  
## auxiliary functions  
source("R/utils.R")  
today\_date <- "2025-04-28"  
## today\_date <- format(Sys.Date(), "%Y-%m-%d")  
  
## set the seed, for fixing generation of ComplexHeatmaps (dendogram clustering)  
set.seed(20)

## 3.2 Compute Compound Similarities

* Actual computation of the correlation matrix using stats::cor.
* Yet, we have to try other correlation methods on the continuous space, or consider other metrics if working on the discrete space. Unbalanced Wassertein distance, or relatives, for distinct input and output dimensions.

### 3.2.1 Compound Heatmaps

* Some tutorials:
  + [Blend of useful resources on generating Heatmaps](https://www.linkedin.com/posts/%F0%9F%8E%AF-ming-tommy-tang-40650014_making-a-heatmap-is-an-essential-skill-for-activity-7317543424511868928-F25t):
  + [Mapping quantitative data to color](https://www.nature.com/articles/nmeth.2134), from Gehlenborg and Wong ([2012](#ref-gehlenborg2012nm)):
  + **Real-Life Examples** with ComplexHeatmap:
    - [Heatmap with transcriptomic expression](https://rpubs.com/crazyhottommy/heatmap_demystified)
    - [Heatmap with genomic expression](https://jokergoo.github.io/ComplexHeatmap-reference/book/more-examples.html)
    - [Genome-level Heatmap](https://jokergoo.github.io/ComplexHeatmap-reference/book/genome-level-heatmap.html)
    - [Plotting large heatmaps in R](https://gdevailly.netlify.app/post/plotting-big-matrices-in-r) and [*Rasterisation*, with thousands of genes to be considered](https://jokergoo.github.io/2020/06/30/rasterization-in-complexheatmap).
    - [Control size of Heatmaps](https://jokergoo.github.io/2020/05/11/set-cell-width/height-in-the-heatmap/)
  + **Real-life examples** with [ggplot2](https://github.com/theislab/atlas-feature-selection-benchmark/blob/b89fc0f66747062e6e1b4b35bd392b27ad035295/analysis/R/plotting.R#L8)
  + **Real-Life examples** with [funkyheatmap CRAN package](https://funkyheatmap.github.io/)

#### 3.2.1.1 Heatmap with Pearson correlation score

* Heatmap in **?@fig-heatmap-normalised-replicate** is generated at the replicate level:

## Read SummarizedExperiment object  
se\_normalised\_replicate <- readRDS(paste0("./results/compounds/se\_normalised\_per\_replicate\_", today\_date, ".rds"))  
se\_normalised\_replicate\_mat <- SummarizedExperiment::assay(se\_normalised\_replicate)  
se\_normalised\_pheno\_data\_replicate <- SummarizedExperiment::colData(se\_normalised\_replicate)  
se\_normalised\_pheno\_data\_replicate$log10Conc <- if\_else(se\_normalised\_pheno\_data\_replicate$Concentrations==0,   
 0,   
 -log10(se\_normalised\_pheno\_data\_replicate$Concentrations))  
## Compute global correlation score  
cor\_per\_replicate <- cor(se\_normalised\_replicate\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_replicate), 0, 1),  
 c("blue", "white", "red"))  
  
## Define custom colour Heatmap annotation  
## num\_batches <- length(unique(se\_normalised\_pheno\_data\_replicate$Batch\_ID))  
## num\_compounds <- length(unique(se\_normalised\_pheno\_data\_replicate$Compound))  
## num\_moa <- length(unique(se\_normalised\_pheno\_data\_replicate$MoA))  
##   
## batch\_col <- setNames(colorRampPalette(brewer.pal(8, "Dark2"))(num\_batches),   
## unique(se\_normalised\_pheno\_data\_replicate$Batch\_ID))  
## compound\_col <- setNames(colorRampPalette(brewer.pal(11, "Spectral"))(num\_compounds),   
## unique(se\_normalised\_pheno\_data\_replicate$Compound))  
## moa\_col <- setNames(colorRampPalette(brewer.pal(12, "Set3"))(num\_moa),   
## unique(se\_normalised\_pheno\_data\_replicate$MoA))  
  
  
## Heatmap annotation  
top\_ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_normalised\_pheno\_data\_replicate |>   
 as.data.frame() |>   
 dplyr::select(Batch\_ID, MoA, Compound))   
 ## col = list(Batch\_ID = batch\_col,   
 ## MoA = moa\_col,   
 ## Compound = compound\_col))  
  
row\_ha <- ComplexHeatmap::rowAnnotation(Concentration =  
 ComplexHeatmap::anno\_barplot(se\_normalised\_pheno\_data\_replicate$log10Conc),   
 Duration = paste0(se\_normalised\_pheno\_data\_replicate$Duration, "d"))  
  
## Generate Heatmap  
heatmap\_normalised\_replicates <- ComplexHeatmap::Heatmap(cor\_per\_replicate,   
 col=col\_heatmap\_scale,   
 name = "CC for all Replicates",   
 show\_row\_names = TRUE,  
 show\_column\_names = FALSE,   
 row\_dend\_reorder = TRUE,  
 column\_dend\_reorder = TRUE,   
 row\_names\_gp = gpar(fontsize = 2),   
 top\_annotation = top\_ha,   
 right\_annotation = row\_ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_normalised\_replicates\_", today\_date,".pdf"),   
 heatmap\_normalised\_replicates, dpi = 600, width = 20, height = 20)

* Heatmap in **?@fig-heatmap-normalised-compound-batch** is generated by averaging normalised barcode counts over replicates:

## Read SummarizedExperiment object  
se\_normalised\_compound\_batch <- readRDS(paste0("./results/compounds/se\_normalised\_per\_compound\_by\_batch\_", today\_date, ".rds"))  
se\_normalised\_compound\_batch\_mat <- SummarizedExperiment::assay(se\_normalised\_compound\_batch)  
se\_pheno\_data\_normalised\_compound\_batch <- SummarizedExperiment::colData(se\_normalised\_compound\_batch)  
## Compute global correlation score  
cor\_per\_compound\_by\_batch <- cor(se\_normalised\_compound\_batch\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_compound\_by\_batch), 0, 1),  
 c("blue", "white", "red"))  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_pheno\_data\_normalised\_compound\_batch |>   
 as.data.frame() |>   
 dplyr::select(Pathway))  
  
## Define Heatmap  
heatmap\_normalised\_compound\_batch <- ComplexHeatmap::Heatmap(cor\_per\_compound\_by\_batch,   
 col=col\_heatmap\_scale,   
 name = "CC for all Compound",   
 show\_row\_names = TRUE,  
 show\_column\_names = TRUE,   
 row\_dend\_reorder = TRUE,  
 column\_dend\_reorder = TRUE,   
 column\_names\_gp = gpar(fontsize = 8),  
 row\_names\_gp = gpar(fontsize = 8),   
 top\_annotation = ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_normalised\_compound\_by\_batch\_", today\_date,".pdf"),   
 heatmap\_normalised\_compound\_batch, dpi = 600, width = 20, height = 20)

* Heatmap in **?@fig-heatmap-global-compound** is generated at the compound level:

## Read SummarizedExperiment object  
se\_normalised\_compound <- readRDS(paste0("./results/compounds/se\_normalised\_per\_compound\_", today\_date, ".rds"))  
se\_normalised\_compound\_mat <- SummarizedExperiment::assay(se\_normalised\_compound)  
se\_normalised\_pheno\_data\_compound <- SummarizedExperiment::colData(se\_normalised\_compound)  
## Compute global correlation score  
cor\_per\_compound <- cor(se\_normalised\_compound\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_compound), 0, 1),  
 c("blue", "white", "red"))  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_normalised\_pheno\_data\_compound |>   
 as.data.frame() |>   
 dplyr::select(Pathway))  
  
## Define Heatmap  
heatmap\_normalised\_compound <- ComplexHeatmap::Heatmap(cor\_per\_compound,   
 col=col\_heatmap\_scale,   
 name = "CC for all Compound",   
 show\_row\_names = TRUE,  
 show\_column\_names = TRUE,   
 row\_dend\_reorder = FALSE,  
 column\_dend\_reorder = TRUE,   
 ## font size  
 row\_names\_gp = gpar(fontsize = 8),   
 top\_annotation = ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_normalised\_compound\_", today\_date,".pdf"),   
 heatmap\_normalised\_compound, dpi = 600, width = 20, height = 20)

#### 3.2.1.2 Heatmap with Pearson correlation and binarised values

* Heatmap at the replicates level in **?@fig-heatmap-binarised-replicates**:

## Read SummarizedExperiment object  
se\_binarised\_replicates <- readRDS(paste0("./results/compounds/se\_discretised\_per\_replicate\_", today\_date, ".rds"))  
se\_binarised\_replicates\_mat <- SummarizedExperiment::assay(se\_binarised\_replicates)  
se\_binarised\_pheno\_data\_replicate <- SummarizedExperiment::colData(se\_binarised\_replicates)  
## Compute global correlation score  
cor\_per\_replicates <- cor(se\_binarised\_replicates\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_replicates), 0, 1),  
 c("blue", "white", "red"))  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_binarised\_pheno\_data\_replicate |>   
 as.data.frame() |>   
 dplyr::select(Pathway, Batch\_ID, Compound))  
  
  
heatmap\_binarised\_replicates <- ComplexHeatmap::Heatmap(cor\_per\_replicates,   
 col=col\_heatmap\_scale,   
 name = "CC at binarised Compound Level",  
 show\_row\_names = TRUE,  
 show\_column\_names = TRUE,   
 row\_dend\_reorder = FALSE,  
 column\_dend\_reorder = TRUE,   
 ## font size  
 row\_names\_gp = gpar(fontsize = 3),   
 top\_annotation = ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_binarised\_replicates\_", today\_date,".pdf"),   
 heatmap\_binarised\_replicates, dpi = 600, width = 20, height = 20)

* Heatmap at the compound by batch level in **?@fig-heatmap-binarised-compound-by-batch** (aggregated over technical replicates):

## Read SummarizedExperiment object  
se\_binarised\_compound\_batch <- readRDS(paste0("./results/compounds/se\_discretised\_per\_compound\_by\_batch\_", today\_date, ".rds"))  
se\_binarised\_compound\_batch\_mat <- SummarizedExperiment::assay(se\_binarised\_compound\_batch)  
se\_binarised\_pheno\_data\_compound\_batch <- SummarizedExperiment::colData(se\_binarised\_compound\_batch)  
## Compute global correlation score  
cor\_per\_compound\_by\_batch <- cor(se\_binarised\_compound\_batch\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_compound\_by\_batch), 0, 1),  
 c("blue", "white", "red"))  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_binarised\_pheno\_data\_compound\_batch |>   
 as.data.frame() |>   
 dplyr::select(Pathway, MoA))  
  
  
heatmap\_binarised\_compound\_batch <- ComplexHeatmap::Heatmap(cor\_per\_compound\_by\_batch,   
 col=col\_heatmap\_scale,   
 name = "CC at binarised Compound Level",  
 show\_row\_names = TRUE,  
 show\_column\_names = TRUE,   
 row\_dend\_reorder = FALSE,  
 column\_dend\_reorder = TRUE,   
 ## font size  
 row\_names\_gp = gpar(fontsize = 8),   
 top\_annotation = ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_binarised\_compound\_by\_batch\_", today\_date,".pdf"),   
 heatmap\_binarised\_compound\_batch, dpi = 600, width = 20, height = 20)

* Heatmap at the compound level in **?@fig-heatmap-binarised-compound**.

## Read SummarizedExperiment object  
se\_binarised\_compound <- readRDS(paste0("./results/compounds/se\_discretised\_per\_compound\_", today\_date, ".rds"))  
se\_binarised\_compound\_mat <- SummarizedExperiment::assay(se\_binarised\_compound)  
se\_binarised\_pheno\_data\_compound <- SummarizedExperiment::colData(se\_binarised\_compound)  
## Compute global correlation score  
cor\_per\_compound <- cor(se\_binarised\_compound\_mat)  
  
## Define clustering colours  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_per\_compound), 0, 1),  
 c("blue", "white", "red"))  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_binarised\_pheno\_data\_compound |>   
 as.data.frame() |>   
 dplyr::select(Pathway, MoA))  
  
  
heatmap\_binarised\_compound <- ComplexHeatmap::Heatmap(cor\_per\_compound,   
 col=col\_heatmap\_scale,   
 name = "CC at binarised Compound Level",  
 show\_row\_names = TRUE,  
 show\_column\_names = TRUE,   
 column\_names\_gp = gpar(fontsize = 8),  
 row\_dend\_reorder = TRUE,  
 column\_dend\_reorder = TRUE,   
 row\_names\_gp = gpar(fontsize = 8),   
 top\_annotation = ha) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/compounds/heatmap\_binarised\_compound\_", today\_date,".pdf"),   
 heatmap\_binarised\_compound, dpi = 600, width = 20, height = 20)

### 3.2.2 Compound Networks

* Plot weighted undirected compound graphs with igraph::graph\_from\_adjacency\_matrix in **?@fig-igraph-drug**, filtering out pairwise connections between 2 drugs if the -value is not significant and absolute correlation score is below 0.8:

## 1) Compute correlation matrix and p-values ----  
cor\_res <- Hmisc::rcorr(se\_binarised\_compound\_mat, type = "pearson")  
cor\_binarised\_compound <- cor\_res$r  
pval\_binarised\_compound <- cor\_res$P  
  
adj\_mat <- cor\_binarised\_compound  
adj\_mat[pval\_binarised\_compound > 0.05/ncol(cor\_res)] <- 0  
  
## 2) build igraph adjacency matrix ----  
compound\_graph <- igraph::graph\_from\_adjacency\_matrix(adjmatrix=adj\_mat,   
 weighted = TRUE,  
 diag = FALSE,   
 mode = "undirected")  
## 3) igraph customisation ----  
compound\_graph$date <- today\_date  
E(compound\_graph)$color <- if\_else(E(compound\_graph)$weight >0, "red", "blue")  
E(compound\_graph)$weight <- abs(E(compound\_graph)$weight)  
  
layout\_graph <- igraph::layout\_with\_fr(compound\_graph,  
 niter = 500)  
  
vertex\_structure <- se\_binarised\_pheno\_data\_compound |>   
 as.data.frame() |>   
 dplyr::mutate(color = if\_else(Pathway=="EGFR", "green", "yellow"))  
  
igraph::V(compound\_graph)$color <- vertex\_structure$color  
  
pdf(paste0("figures/compounds/compound\_graph\_", today\_date, ".pdf"),   
 width = 16, height = 16, useDingbats = TRUE, compress = FALSE)  
plot(compound\_graph, layout = layout\_graph, vertex.size = 10,  
 ## vertex.label.dist= 2,  
 main = "Compound network interaction.  
 In green, drugs related with the EGFR mechanism,  
 and in yellow other pathways.  
 Edge widths are proportional to the coefficient of correlation.",   
 edge.width = igraph::E(compound\_graph)$weight \* 20,  
 vertex.label.cex = 1, curved = 0.2)  
dev.off()  
## png   
## 2  
  
## 4) export PNG and readable graph formats for Neo4J  
igraph::write\_graph(graph = compound\_graph,   
 file = paste0("./results/compounds/compound\_network\_", today\_date, ".graphml"),  
 format = "graphml")

## 3.3 Compute Cell Lines Similarities

### 3.3.1 Cell Line by Compound

se\_binarised\_compound\_batch\_fac <- apply(se\_binarised\_compound\_batch\_mat, 2, as.character)  
  
## Heatmap annotation  
ha <- ComplexHeatmap::HeatmapAnnotation(df = se\_binarised\_pheno\_data\_compound\_batch |>   
 as.data.frame() |>   
 dplyr::select(Batch\_ID, Pathway, Compound))  
  
heatmap\_cell\_lines\_by\_sample <- ComplexHeatmap::Heatmap(se\_binarised\_compound\_batch\_fac,   
 col = c("TRUE" = "red", "FALSE" = "blue"),   
 name = "Heatmap: cell-line by sample",  
 show\_row\_names = FALSE,  
 row\_dend\_reorder = TRUE,  
 ## clustering\_distance\_rows = "pearson",  
 show\_row\_dend = FALSE,  
 column\_dend\_reorder = TRUE,   
 show\_column\_names = TRUE,   
 ## clustering\_distance\_columns = "pearson",  
 column\_names\_gp = gpar(fontsize = 4),   
 top\_annotation = ha,   
 ## compression issues  
 use\_raster = TRUE) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/cell\_lines/heatmap\_cell\_lines\_by\_samples\_", today\_date,".pdf"),   
 heatmap\_cell\_lines\_by\_sample, dpi = 300, width = 20, height = 20)

### 3.3.2 Cell Line Correlation

## 1) evaluation of cell resistance  
present\_cells <- rowSums(se\_binarised\_compound\_batch\_mat)  
hist(present\_cells,   
 breaks = length(unique(present\_cells)),  
 main=NULL)  
  
## 2) filtered Heatmaps   
threshold\_cell\_prevalence <- 10 ## alternatively, remove cells that are not present in all samples  
se\_binarised\_compact <- se\_binarised\_compound\_batch\_mat[present\_cells >  
 threshold\_cell\_prevalence, ]  
cor\_cell\_lines <- cor(t(se\_binarised\_compact))  
col\_heatmap\_scale <- circlize::colorRamp2(c(min(cor\_cell\_lines, na.rm = TRUE),  
 0,   
 max(cor\_cell\_lines, na.rm = TRUE)),  
 c("blue", "white", "red"))  
  
## 3) generate Heatmap of correlation cell lines  
heatmap\_cell\_lines\_by\_cell\_lines <- ComplexHeatmap::Heatmap(cor\_cell\_lines,   
 col = col\_heatmap\_scale,   
 name = "Heatmap: cell-line by cell line",  
 show\_row\_names = FALSE,  
 row\_dend\_reorder = TRUE,  
 show\_row\_dend = TRUE,  
 column\_dend\_reorder = TRUE,   
 show\_column\_names = FALSE,   
 na\_col = "black",   
 use\_raster = TRUE,   
 raster\_resize\_mat = mean) |>   
 ComplexHeatmap::draw() |>   
 grid::grid.grabExpr()  
  
ggsave(paste0("figures/cell\_lines/heatmap\_cell\_lines\_by\_cell\_lines\_", today\_date,".pdf"),   
 heatmap\_cell\_lines\_by\_cell\_lines, dpi = 600, width = 20, height = 20)

|  |
| --- |
| Figure 3.1: Histogram of the distribution of cell resistance (for a given barcode ID, returns the total number of cell lines identified). |

# 4. Future perspectives: single-cell integration.

### 4.0.1 Alternative metrics

* [Compute correlations with strongly sparse vectors?](https://www.linkedin.com/posts/%F0%9F%8E%AF-ming-tommy-tang-40650014_1-you-have-a-clear-question-is-gene-a-activity-7322971683596734464-5BqV)

### 4.0.2 Pair Single Cell with (or without) Drug fingerprints

* [Single cell and compound response largest database](https://www.linkedin.com/posts/independent-data-lab_check-out-the-preprint-activity-7300471798029012993-YRjy)
* [Biologist perspective](https://docs.google.com/document/d/1UHD6IG9Rti2tD77zwHfSyh5XyTbnJ25S/edit)
  + Single-cell lineage capture across genomic modalities with CellTag-multi reveals fate-specific gene regulatory changes -> use of **single-cell lineage-tracing (scLT)**.
  + High-resolution, noninvasive single-cell lineage tracing in mice and humans based on DNA methylation epi-mutations.
* Ming Tommy Tang lists in [Single-cell LinkedIn post](https://www.linkedin.com/posts/%F0%9F%8E%AF-ming-tommy-tang-40650014_pylemur-activity-7320442513671008256-qLJC) trendy papers and tools for multi-sample, single-cell RNAseq differential expression analysis.

#### 4.0.2.1 Correct for Batch Effects

* [DESeq2-MultiBatch](https://www.biorxiv.org/content/10.1101/2025.04.20.649392v1): Batch Correction for Multi-Factorial RNA-seq Experiments, avalaible as open-source [GH repository](https://github.com/julienroyulaval/DESeq2-MultiBatch), from Roy, Monthony, and Torkamaneh ([2025](#ref-roy2025)).

#### 4.0.2.2 Drug-response

* [TRADE: Transcriptome-wide analysis of differential expression in perturbation atlases](https://www.nature.com/articles/s41588-025-02169-3), from Nadig et al. ([2025](#ref-nadig2025ng)). The paper notably reports how to model a compound-dose response curve with a **Hill equation**. Avalaible as an open-source GH repository [TRADEtools](https://github.com/ajaynadig/TRADEtools). [R TRADE Tutorial](https://www.rna-seqblog.com/trade-ranscriptome-wide-analysis-of-differential-expression/)

### 4.0.3 Barcode Differential Analysis

#### 4.0.3.1 bartools and BARtab

* Analysis of synthetic cellular barcodes in the genome and transcriptome with BARtab and bartools.

#### 4.0.3.2 DEBRA

* Pros DEBRA
  + Better characterisation of the mean-variance deviation -> between trended or shrinkage, trended is favoured.
* Cons DEBRA:
  + DEBRA does not account for outliers expression, nor zero-inflated counts -> recommendation of glmQLFit and glmQLFTest for routine GLM-based DE analyses, from [EdgeR: Explaning dispersion types to newbies](https://support.bioconductor.org/p/110273/).
  + Complex protocol for discarding lowly differentially expressed barcodes.
  + No available BioConductor/CRAN Repository, while latest [DEBRA GitHub update](https://github.com/YevhenAkimov/DEBRA_1.01) dates back more than 4 years.

#### 4.0.3.3 Combine Fold change and -value

* [Combine statistical discernibility (aka *significance*) with practical significance](https://www.linkedin.com/posts/adrianolszewski_how-to-combine-statistical-discernibility-activity-7310365190959632384-sNHs)

### 4.0.4 Drug clustering and mapping

* Use of graph clustering approaches? Like Louvain? + multiple case studies, how to combine them (2 vials of cell lines)?
* Compare with ATC prediction and clustering: [PDATC-NCPMKL-updated GH Repo](https://github.com/Lywhere/PDATC-NCPMKL-updated).

# Appendix A — Appendix: publish to standardised cell line repositories.

## A.1 Repository organisation and Data tracking

* [Project Code Organisation](https://www.linkedin.com/posts/danleedata_become-10-better-at-machine-learning-by-activity-7320797963549564928-Au1_)
* [Data Fingerprinting with MDA5 and Environment Snapshots: R and Code tracking](https://www.linkedin.com/posts/sebastian-rauschert-836760a0_datascience-reproducibleanalytics-commandline-activity-7312256950962896896-9JkE)
* Large datasets versioning:
  + [DataLad VS DVC](https://www.linkedin.com/posts/sebastian-rauschert-836760a0_install-datalad-activity-7307188424682127361-dJ_G)
  + [DataLad Versus DVC](https://www.linkedin.com/posts/sebastian-rauschert-836760a0_dataversioncontrol-dvc-datascience-activity-7314768876876058626-GVNF), and [DataLad Versus DVC V2](https://handbook.datalad.org/en/latest/beyond_basics/101-168-dvc.html#dvc)
  + [Git LFS VS Alembic](https://www.linkedin.com/posts/sebastian-rauschert-836760a0_reproducibility-databaseversioning-dataengineering-activity-7315522466259390466-zMun)
* [Homogenise Excel colnames, and other spreadsheet tips](https://www.linkedin.com/posts/%F0%9F%8E%AF-ming-tommy-tang-40650014_115-every-wet-lab-uses-spreadsheets-activity-7319725287603130368-9ILC)

## A.2 Publish cell lines on Web repositories

* [cellosaurus General Comments](https://www.expasy.org/resources/cellosaurus) and [cellosaurus Official Website](https://www.cellosaurus.org/), from([Bairoch 2018](#ref-bairoch2018jbt); [Robin, Capes-Davis, and Bairoch 2020](#ref-robin2020ijc)). Specifically, the CLASTR algorithm relies on *short-tandem repeat* patterns to map user-provided cell lines with the ones available in Cellosaurus database.
* [Bgee](https://www.bgee.org/).

# Bibliographic References

Bairoch, Amos. 2018. ‘The Cellosaurus, a Cell-Line Knowledge Resource’. *Journal of Biomolecular Techniques : JBT* 29 (2): 25–38. <https://doi.org/10.7171/jbt.18-2902-002>.

Gehlenborg, Nils, and Bang Wong. 2012. ‘Mapping Quantitative Data to Color’. *Nature Methods* 9 (8): 769–69. <https://doi.org/10.1038/nmeth.2134>.

Nadig, Ajay, Joseph M. Replogle, Angela N. Pogson, Mukundh Murthy, Steven A. McCarroll, Jonathan S. Weissman, Elise B. Robinson, and Luke J. O’Connor. 2025. ‘Transcriptome-Wide Analysis of Differential Expression in Perturbation Atlases’. *Nature Genetics*, April, 1–10. <https://doi.org/10.1038/s41588-025-02169-3>.

Robin, Thibault, Amanda Capes-Davis, and Amos Bairoch. 2020. ‘CLASTR: The Cellosaurus STR Similarity Search Tool - A Precious Help for Cell Line Authentication’. *International Journal of Cancer* 146 (5): 1299–1306. <https://doi.org/10.1002/ijc.32639>.

Roy, Julien, Adrian S. Monthony, and Davoud Torkamaneh. 2025. ‘DESeq2-MultiBatch: Batch Correction for Multi-Factorial RNA-seq Experiments’. 22 April 2025. <https://doi.org/10.1101/2025.04.20.649392>.