Structural analysis of CODEX CTCL data with Kasumi

Jovan Tanevski

2025-02-20

```
library(kasumi)
#> Kasumi is able to run computationally intensive functions
#>   in parallel. Please consider specifying a future::plan(). For example by running
#>   future::plan(future::multisession) before calling Kasumi functions.
library(tidyverse)
#> ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
#> ✔ dplyr     1.1.4     ✔ readr     2.1.5
#> ✔ forcats   1.0.0     ✔ stringr   1.5.1
#> ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
#> ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
#> ✔ purrr     1.0.4
#> ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
#> ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
library(readxl)
```

Download Supplementary Data 1 from the publication Phillips, D., Matusiak, M., Gutierrez, B.R. et al. **Immune cell topography predicts response to PD-1 blockade in cutaneous T cell lymphoma**. *Nat Commun* 12, 6726 (2021). https://doi.org/10.1038/s41467-021-26974-6

```
if(!file.exists("CTCL.xlsx")){
  download.file("https://static-content.springer.com/esm/art%3A10.1038%2Fs41467-021-26974-6/Medic
}
```

Read data, select from the data the pre-treatment responder (Group 1) and non-responder (Group 2) samples (spot ids). Represent each sample by the one-hot encoded cell types (all.cells.ctcl) and their corresponding positions (all.positions.ctcl)

```
ctcl <- read_xlsx("CTCL.xlsx", skip=2)

spots <- ctcl %>%
  filter(Groups %in% c(1, 2)) %>%
  pull(Spots) %>%
  unique()

outcome <- ctcl %>%
  filter(Groups %in% c(1, 2)) %>%
  group_by(Spots, Patients) %>%
  summarize(Groups = Groups[1], .groups = "drop")

cts <- ctcl %>%
  pull(ClusterName) %>%
  unique()

all.cells.ctcl <- spots %>% map(\(id){
  ctcl %>%
    filter(Spots == id) %>%
    pull(ClusterName) %>%
    map(~ .x == cts) %>%
    rlist::list.rbind() %>%
    `colnames<-`(make.names(cts)) %>%
    as_tibble(.name_repair = "unique")
})

names(all.cells.ctcl) <- spots

all.positions.ctcl <- spots %>% map(\(id){
  ctcl %>%
    filter(Spots == id) %>%
    select(X, Y) %>%
    `colnames<-`(c("x", "y"))
})

names(all.positions.ctcl) <- spots
```

Alternatively each sample can be represented by the mean marker abundances in each cell

```
panel <- colnames(ctcl)[10:68][-c(53,56,58)]

all.cells.ctcl <- spots %>% map(\(id){
  ctcl %>%
    filter(Spots == id) %>%
    select(all_of(panel)) %>%
    rename_with(make.names)
})
```

For each sample, create a view composition consisting of an intraview capturing the identity of each cell as a one-hot encoded vector and a paraview capturing the cell-type composition of the 10 nearest neighbors. We prefix the names of the features (cell types) in the paraview to allow prediction of a cell type as a function of the number of cells with the same cell type in its neighborhood. We run Kasumi for each sample and store all results in the same database. We select a window size of 400px, and overlap of 50% and we require at least 20 cells per window. Windows with less than 20 cells will be ignored. Note that we are bypassing the modeling of the intraview and we are training only models predicting the cell type from the cell type composition in the neighborhood. If not defined otherwise (by defining a future::plan) Kasumi will run without parallelization. Consider defining a plan to speed up the modeling.

You can skip this step by downloading the results databases generated for the manuscript and the corresponding results objects from Zenodo (https://doi.org/10.5281/zenodo.14891956) for reproducing the results from the manuscript.

```
# future::plan(future::multisession, workers = 8)
if(!file.exists("CTCLct400.sqm")){

  as.character(spots) %>% walk(\(id){
    ct <- all.cells.ctcl[[id]]
    pos <- all.positions.ctcl[[id]]

    kasumi.views <- create_initial_view(ct) %>% add_paraview(pos, 10, family = "constant", prefix

    suppressWarnings(
      run_kasumi(kasumi.views, pos, window=400, overlap=50, id, "CTCLct400.sqm", minu=20, bypass.
    )
  })

}
```

Alternatively, if working with marker abundance representations the view composition should be somewhat different. The intraview captures the marker abundances per cell and the paraview captures the weighted sum of the abundances in the broader tissue structure, e.g., in a radius of 100px. Here we also model the intraview since we want to also capture the marker relationships within each cell. The downstream analysis proceeds in a similar way as for the cell type scenario.

```
seq_along(spots) %>% walk(\(id){
  expr <- all.cells.ctcl[[id]]
  pos <- all.positions.ctcl[[id]]

  kasumi.views <- create_initial_view(expr) %>% add_paraview(pos, 100, family = "gaussian")

  suppressWarnings(
    run_kasumi(kasumi.views, pos, window=400, overlap=50, id, "CTCLexpr400.sqm", minu=20)
  )
})
```

Collect the results, extract the relationship-based representation of windows, cluster and aggregate the clusters per sample. Filter non-persistent clusters.

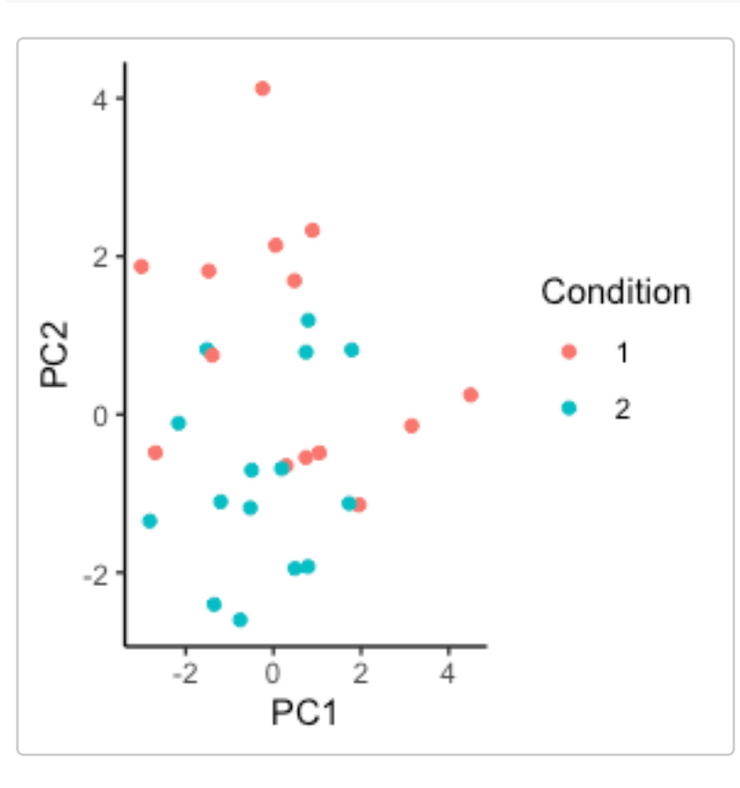
Note that the results of the Leiden clustering via the igraph package can be different for different versions of igraph. For the results reported in the manuscript igraph version 1.5.1 was used. Also, due to slight differences in implementation between the prototype and benchmarking implementation of Kasumi, and the R package implementation, the results may differ.

In both cases, while the performance and the interpretation reported in the manuscript can be reproduced, the clustering parameters might require slight adjustment.

```
# future::plan(future::multisession, workers = 8)

kasumi.results <- collect_results("CTCLct400.sqm")

# First representation - relationships
kasumi.representation <- extract_representation(kasumi.results)

# Second representation - clusters
kasumi.clusters <- extract_clusters(kasumi.representation, "leiden", 0.4, 0.9)

# Cluster composition
kasumi.agg <- aggregate_clusters(kasumi.clusters)

# Third representation - persistent cluster composition -
# clusters that are present in at least 5 samples
persistent.clusters <- persistent_clusters(kasumi.agg, 5)
kasumi.persistent <- kasumi.agg %>% select(id, all_of(persistent.clusters))
```

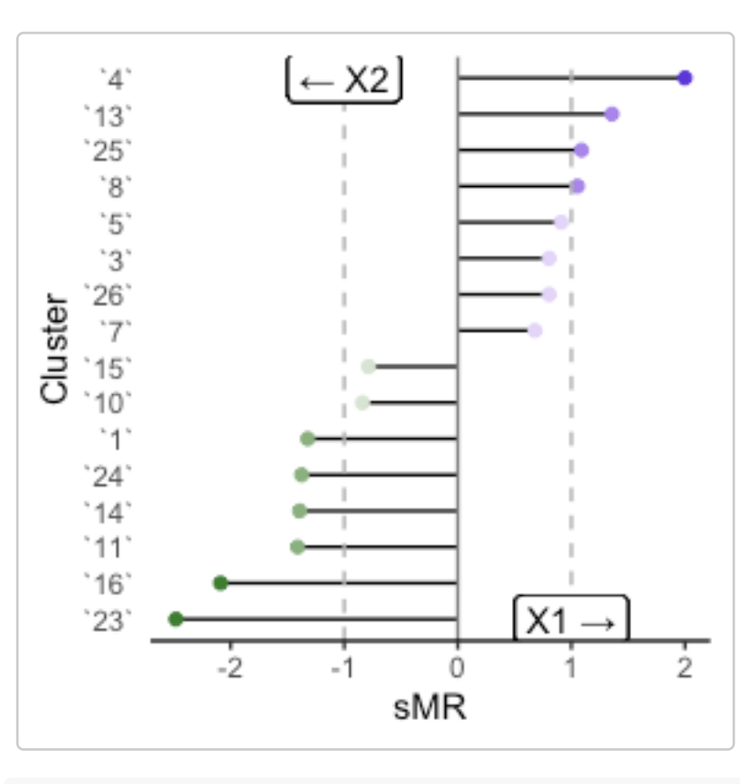
The Kasumi representation can be used for downstream unsupervised or supervised analysis. For example we can start by visualizing a lower dimensional map (PCA) of the representation to identify groups of samples.

```
# Extract condition information
target <- kasumi.persistent %>% mutate(id = str_remove(id, "sample")) %>%
  left_join(outcome %>% mutate(Spots = as.character(Spots)), by = c("id" = "Spots")) %>% pull(Grc

kasumi.pca <-prcomp(kasumi.persistent %>% select(-id), scale. = TRUE)

to.plot <- data.frame(kasumi.pca$x[,c(1,2)]) %>% mutate(Condition = as.factor(target))

ggplot(to.plot, aes(x = PC1, y = PC2, color = Condition)) + geom_point() + theme_classic()
```



In a supervise setting to evaluate whether the representation is associated with a clinical observation we can calculate the 10-fold cross validated performance of a linear classifier separating the samples into responders and non-responders based on the learned Kasumi representation. We can next estimate the importance of each cluster for the classification task by calculating the signed model reliance.

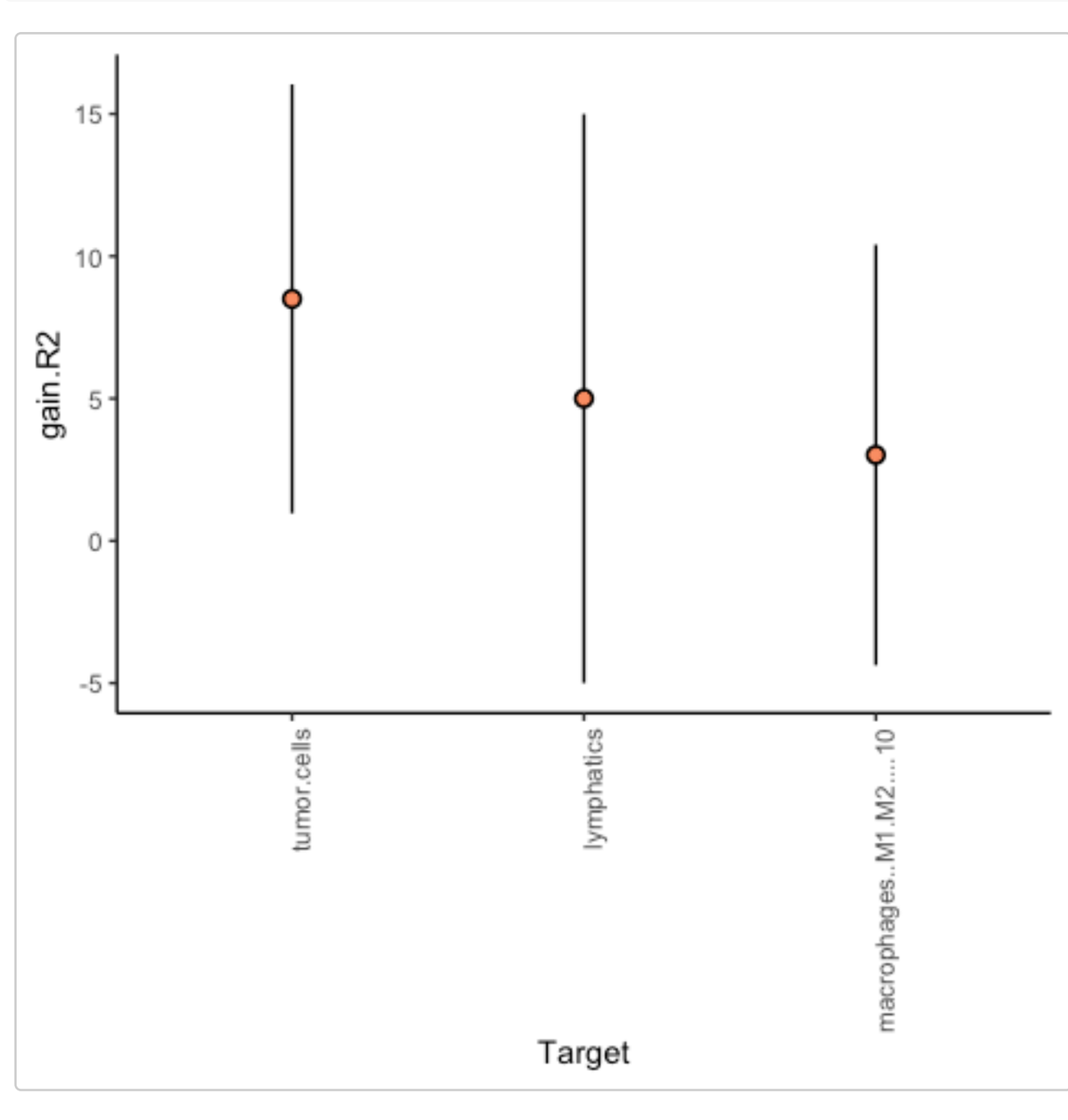
```
# Run logistic regression
kasumi.roc <- downstream_classify(kasumi.persistent, make.names(target))
#> Loading required package: lattice
#>
#> Attaching package: 'caret'
#> The following object is masked from 'package:purrr':
#>
#>     lift


# Signed model reliance
sMR(kasumi.persistent, as.factor(make.names(target)))
#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
#> AUC: 0.866666666666667
```



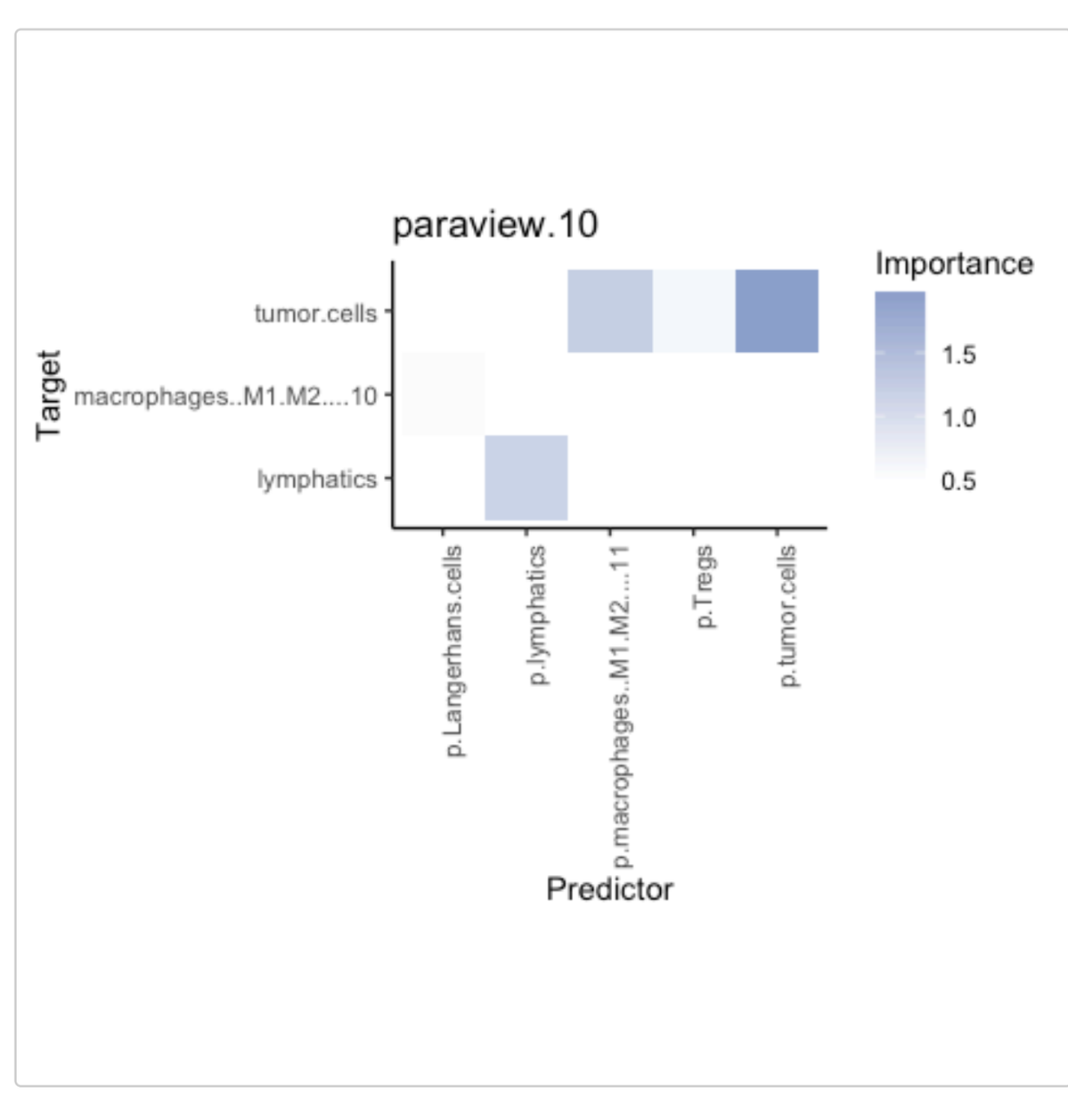

```
#> # A tibble: 16 × 2
#>    Cluster    sMR
#>    <fct>    <dbl>
#>  1 `1`     -1.32
#>  2 `3`      0.804
#>  3 `4`      2
#>  4 `5`      0.911
#>  5 `7`      0.679
#>  6 `8`      1.05
#>  7 `10`    -0.839
#>  8 `11`    -1.41
#>  9 `13`     1.36
#> 10 `14`    -1.39
#> 11 `15`    -0.786
#> 12 `16`    -2.09
#> 13 `23`    -2.48
#> 14 `24`    -1.37
#> 15 `25`     1.09
#> 16 `26`     0.804
```

Guided by the outcome of the classification task and the importance of the clusters for predicting the condition, individual clusters can be explored for the underlying cell type relationship patterns they capture.

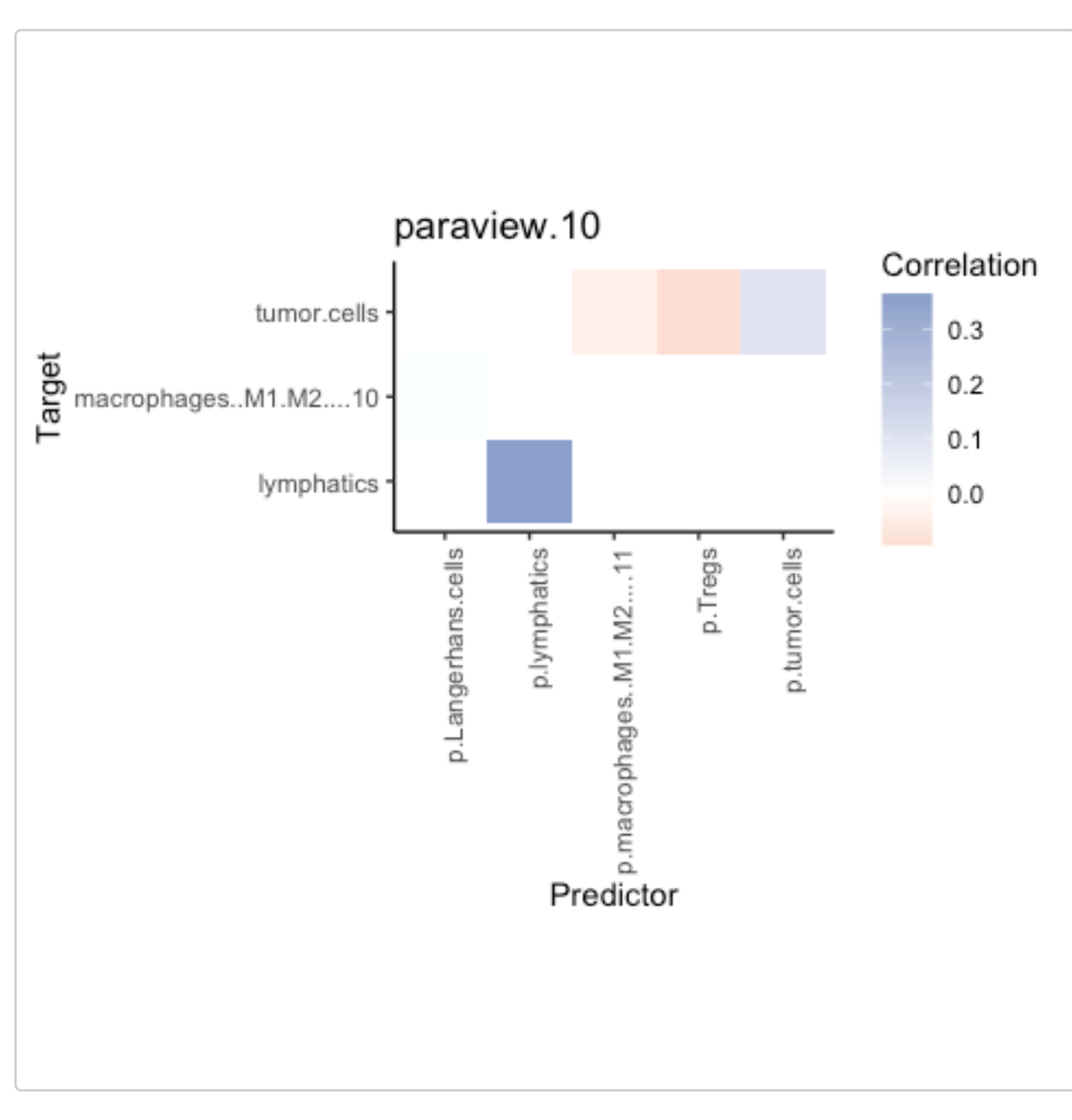
```
cluster.23  <- collect_kasumi_cluster(kasumi.clusters, "23", "CTCLct400.sqm")
#>
#> Collecting improvements
#>
#> Collecting contributions
#>
#> Collecting importances
#>
#> Aggregating

plot_improvement_stats(cluster.23, trim = 1)
```




```
plot_interaction_heatmap(cluster.23, "paraview.10", trim = 1, cutoff = 0.5, clean = TRUE)
```




```
plot_interaction_heatmap(cluster.23, "paraview.10", trim = 1, cutoff = 0.5, clean = TRUE, correla
```



In more recent versions, Kasumi also offers the possibility to perform an analysis of the clusters of sliding windows across samples represented by the composition of cell types or markers (WCC) as an alternative to the relationship based representation. Kasumi stores information about the window composition in the results database. The steps to follow are same as above. The difference is only in the function used to collect the results.

```
kasumi.wcc <- collect_wcc("results_database.sqm")
```

Note that the databases available on Zenodo do not contain a window composition table since they were generated with an older version of Kasumi. If interested in reproducting the results from the manuscript related to WCC follow the analysis steps in the Kasumi implementation at https://github.com/saezlab/kasumi_bench.

Here is the output of `sessionInfo()` at the point when this document was compiled:

```
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.3.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;
LAPACK version 3.12.0
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> time zone: Europe/Berlin
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] caret_7.0-1     lattice_0.22-6  readxl_1.4.3    lubridate_1.9.4
#>  [5] forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4     purrr_1.0.4
#>  [9] readr_2.1.5     tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1
#> [13] tidyverse_2.0.0 kasumi_0.99.7
#>
#> loaded via a namespace (and not attached):
#>  [1] tidyselect_1.2.1     timeDate_4041.110    farver_2.1.2
#>  [4] blob_1.2.4           fastmap_1.2.0        pROC_1.18.5
#>  [7] digest_0.6.37        rpart_4.1.24         timechange_0.3.0
#> [10] lifecycle_1.0.4      survival_3.8-3       RSQLite_2.3.9
#> [13] magrittr_2.0.3       compiler_4.4.2       rlang_1.1.5
#> [16] sass_0.4.9           tools_4.4.2          utf8_1.2.4
#> [19] igraph_1.5.1         yaml_2.3.10          data.table_1.16.4
#> [22] knitr_1.49           labeling_0.4.3       bit_4.5.0.1
#> [25] plyr_1.8.9           withr_3.0.2          stats4_4.4.2
#> [28] nnet_7.3-20          grid_4.4.2           colorspace_2.1-1
#> [31] future_1.34.0        globals_0.16.3       scales_1.3.0
#> [34] iterators_1.0.14     MASS_7.3-64          cli_3.6.4
#> [37] rmarkdown_2.29       generics_0.1.3       rlist_0.4.6.2
#> [40] rstudioapi_0.17.1    future.apply_1.11.3  reshape2_1.4.4
#> [43] tzdb_0.4.0           DBI_1.2.3            cachem_1.1.0
#> [46] proxy_0.4-27         splines_4.4.2        assertthat_0.2.1
#> [49] parallel_4.4.2       cellranger_1.1.0     vctrs_0.6.5
#> [52] hardhat_1.4.1        Matrix_1.7-2         jsonlite_1.8.9
#> [55] hms_1.1.3            bit64_4.6.0-1        listenv_0.9.1
#> [58] foreach_1.5.2        gower_1.0.2          jquerylib_0.1.4
#> [61] recipes_1.1.1        glue_1.8.0           parallelly_1.42.0
#> [64] codetools_0.2-20     stringi_1.8.4        gtable_0.3.6
#> [67] munsell_0.5.1        furrr_0.3.1          pillar_1.10.1
#> [70] htmltools_0.5.8.1    ipred_0.9-15         lava_1.8.1
#> [73] R6_2.6.1             mistyR_1.99.11       evaluate_1.0.3
#> [76] memoise_2.0.1        bslib_0.9.0          class_7.3-23
#> [79] Rcpp_1.0.14          nlme_3.1-167         prodlim_2024.06.25
#> [82] xfun_0.50            ModelMetrics_1.2.2.2 pkgconfig_2.0.3
```