mistyR and SpatialExperiment/SingleCellExperiment

true true

2021-10-25

Introduction

mistyR can be used to analyze spatial omics data sets stored in *SpatialExperiment* object with just a couple of functions. In this vignette we demonstrate how to build a user friendly workflow starting from data preprocessing, through running *mistyR*, to analysis of results, i.e., the spatial interactions between markers stored in alternative assays and specific locations.

The functions provided in this notebook can be adapted to the user preference but the main objective is to exploit as much as possible the flexibility of workflow creation from *mistyR* and object manipulation from *SpatialExperiment* and `BiocStyle::Biocpkg("SingleCellExperiment")`.

```
# MISTy
library(mistyR)
library(future)

# SpatialExperiment
library(SpatialExperiment)
library(SingleCellExperiment)
library(SummarizedExperiment)

# data manipulation
library(Matrix)
library(tibble)
library(dplyr)
library(purrr)

# normalization
library(sctransform)

# resource
library(progeny)

# plotting
library(ggplot2)

# setup parallel execution
plan(multisession)
```

The skeleton of mistyR pipelines

For user convenience and to facilitate the use of *mistyR* and *SpatialExperiment*, `run_misty_spe()` is a function describing a general skeleton of a mistyR workflow for analysing a 10x Visium slide given in a SpatialExperiment object. The function allows for:

1) Defining a number of assays/views to be used in the model.
2) Defining the type of spatial context for each view and their parameters.
3) Defining the specific assay and features to be used for the view creation of each view.
4) Defining the specific spots where the model will be built.

```
run_misty_spe <- function(slide,
                          # SpatialExperiment object with spatial omics data.
                          view.assays,
                          # Named list of assays for each view.
                          view.features = NULL,
                          # Named list of features/markers to use.
                          # Use all by default.
                          view.types,
                          # Named list of the type of view to construct
                          # from the assay.
                          view.params,
                          # Named list with parameters (NULL or value)
                          # for each view.
                          spot.ids = NULL,
                          # spot IDs to use. Use all by default.
                          out.alias = "results"
                          # folder name for output
) {

  # Extracting geometry
  geometry <- as.data.frame(spatialData(slide)) %>%
    select(array_row, array_col)

  # Extracting data
  view.data <- map(view.assays,
    extract_spe_data,
    geometry = geometry,
    slide = slide
  )

  # Constructing and running a workflow
  build_misty_pipeline(
    view.data = view.data,
    view.features = view.features,
    view.types = view.types,
    view.params = view.params,
    geometry = geometry,
    spot.ids = spot.ids,
    out.alias = out.alias
  )
}
```

Extracting specific information from SpatialExperiment objects

These are helper functions that allow to extract from *SpatialExperiment* objects specific assays and transform them into `tibble` which is a preferred format for *mistyR*.

```
# Extracts data from an specific assay from a SpatialExperiment object
# and aligns the IDs to the geometry
extract_spe_data <- function(slide,
```

```
                              assay,
                              geometry) {
  data <- altExp(slide, assay) %>%
    assay() %>%
    t() %>%
    as_tibble(rownames = NA)

  return(data %>% dplyr::slice(match(rownames(.), rownames(geometry))))
}


# Filters data to contain only features of interest
filter_data_features <- function(data,
                                 features) {
  if (is.null(features)) features <- colnames(data)

  return(data %>% rownames_to_column() %>%
    select(rowname, all_of(features)) %>% rename_with(make.names) %>%
    column_to_rownames())
}
```

View creation

This helper function wraps the three options available by default for view creation in *mistyR* with additional features that allow for creating views for specific spots.

```
# Builds views depending on the paramaters defined
create_default_views <- function(data,
                                 view.type,
                                 view.param,
                                 view.name,
                                 spot.ids,
                                 geometry) {
  view.data.init <- create_initial_view(data)

  if (!(view.type %in% c("intra", "para", "juxta"))) {
    view.type <- "intra"
  }

  if (view.type == "intra") {
    data.red <- view.data.tmp$data %>%
      rownames_to_column() %>%
      filter(rowname %in% spot.ids) %>%
      select(-rowname)
  } else if (view.type == "para") {
    view.data.tmp <- view.data.init %>%
      add_paraview(geometry, l = view.param)

    data.ix <- paste0("paraview.", view.param)
    data.red <- view.data.tmp[[data.ix]]$data %>%
      mutate(rowname = rownames(data)) %>%
      filter(rowname %in% spot.ids) %>%
      select(-rowname)
  } else if (view.type == "juxta") {
```

```
    view.data.tmp <- view.data.init %>%
      add_juxtaview(
        positions = geometry,
        neighbor.thr = view.param
      )

    data.ix <- paste0("juxtaview.", view.param)
    data.red <- view.data.tmp[[data.ix]]$data %>%
      mutate(rowname = rownames(data)) %>%
      filter(rowname %in% spot.ids) %>%
      select(-rowname)
  }

  if (is.null(view.param) == TRUE) {
    misty.view <- create_view(
      paste0(view.name),
      data.red
    )
  } else {
    misty.view <- create_view(
      paste0(view.name, "_", view.param),
      data.red
    )
  }

  return(misty.view)
}
```

Building a mistyR pipeline and running the model

This wrapper function `build_misty_pipeline()` allows for building automatically a *mistyR* workflow from a list of data frames, with specified spatial context, parameters, features and locations.

```
# Builds automatic MISTy workflow and runs it
build_misty_pipeline <- function(view.data,
                                 view.features,
                                 view.types,
                                 view.params,
                                 geometry,
                                 spot.ids = NULL,
                                 out.alias = "default") {

  # Adding all spots ids in case they are not defined
  if (is.null(spot.ids)) {
    spot.ids <- rownames(view.data[[1]])
  }

  # First filter the features from the data
  view.data.filt <- map2(view.data, view.features, filter_data_features)

  # Create initial view
  views.main <- create_initial_view(view.data.filt[[1]] %>%
    rownames_to_column() %>%
    filter(rowname %in% spot.ids) %>% select(-rowname))
```

```
  # Create other views
  view.names <- names(view.data.filt)

  all.views <- pmap(list(
    view.data.filt[-1],
    view.types[-1],
    view.params[-1],
    view.names[-1]
  ),
  create_default_views,
  spot.ids = spot.ids,
  geometry = geometry
  )

  pline.views <- add_views(
    views.main,
    unlist(all.views, recursive = FALSE)
  )


  # Run MISTy
  run_misty(pline.views, out.alias)
}
```

Basic visualization function

Finally, we adapted the function `plotMolecules()` from *ggspavis* to visualize features from custom assays for the purposes of the use case.

```
plotMolecules_adapted <- function(spe,
                                  molecule = NULL,
                                  x_coord = "array_col",
                                  y_coord = "array_row",
                                  palette = NULL,
                                  alt_assay = "lognorm") {
  if (is.null(palette)) {
    palette <- "yellow"
  }
  if (!is.null(palette) && length(palette) == 1) {
    palette <- c("black", palette)
  }
  df_plot <- spatialData(spe)[, c(x_coord, y_coord), drop = FALSE]
  mRNA_counts <-
    as.numeric(assay(altExp(spe, alt_assay))[molecule, , drop = FALSE])
  stopifnot(length(mRNA_counts) == nrow(df_plot))

  df_plot <- cbind(df_plot, expression = mRNA_counts)

  df_plot <- as.data.frame(df_plot) %>% mutate(array_row = array_row * -1)

  p <- ggplot(
    df_plot,
    aes_string(x = x_coord, y = y_coord, color = "expression")
```

```
  ) +
    geom_point(size = 2.5) +
    scale_color_gradient(
      low = palette[1],
      high = palette[2], trans = "sqrt"
    ) +
    coord_fixed() +
    ggtitle(molecule) +
    theme_void()
  p
}
```

Use case

As an example, we will analyze a 10X Visium spatial gene expression dataset of one breast cancer section (Block A Section 1) available here [https://support.10xgenomics.com/spatial-gene-expression/datasets]. We assume that the required files Feature/cell matrix HDF5 (filtered) and the Spatial imaging data (extracted) are in a folder named 'breast_A_1' in the current working directory.

We will explore the spatial interactions of the Hypoxia pathway responsive genes with the Estrogen pathway responsive genes. To this end we will use the model matrix with top significant genes for each pathway from the package *progeny* and the previously described functions.

Please note that the *SpatialExperiment* function `read10xVisium()` requires a fixed file structure that we create programmatically for this e xample. Furthermore, in this example we dropped all repeated symbols, however, the user must define what's the best solution for their analysis.

Loading and normalizing the data sets

```
# Load and normalize using SCT
folder <- "breast_A_1"

# rename to create the required file structure
file.rename(
  "breast_A_1/V1_Breast_Cancer_Block_A_Section_1_filtered_feature_bc_matrix.h5",
  "breast_A_1/filtered_feature_bc_matrix.h5"
)
#> [1] TRUE

spe <- read10xVisium(folder, type = "HDF5", data = "filtered",
                     images = "lowres")


# normalize data
# counts are of class DelayedMatrix which is incompatible with vst
sct.data <- vst(as(counts(spe), "dgCMatrix"), verbosity = 0)$y

# Dropping duplicates
spe <- spe[!duplicated(rowData(spe)), ]

# Getting relevant genes
gene.dict <- as_tibble(rowData(spe), rownames = NA) %>%
  rownames_to_column("ENSEMBL") %>%
```

```
  filter(ENSEMBL %in% rownames(sct.data))

# Re-naming normalized data with gene symbols
sct.data <- sct.data[gene.dict %>% pull("ENSEMBL"), colnames(spe)]
rownames(sct.data) <- gene.dict %>% pull(symbol)

# undo file rename
file.rename(
  "breast_A_1/filtered_feature_bc_matrix.h5",
  "breast_A_1/V1_Breast_Cancer_Block_A_Section_1_filtered_feature_bc_matrix.h5"
)
#> [1] TRUE
```

Filtering genes that are expressed in at least 5% of the spots

```
coverage <- rowSums(sct.data > 0) / ncol(sct.data)
slide.markers <- names(coverage[coverage >= 0.05])
```

Defining Hypoxia and Estrogen responsive genes

For this simple example we will pick the top 15 most significantly responsive genes of each pathway from the model matrix from the package `progeny`.

```
estrogen.footprints <- getModel(top = 15) %>%
  rownames_to_column("gene") %>%
  filter(Estrogen != 0, gene %in% slide.markers) %>%
  pull(gene)

hypoxia.footprints <- getModel(top = 15) %>%
  rownames_to_column("gene") %>%
  filter(Hypoxia != 0, gene %in% slide.markers) %>%
  pull(gene)
```

Note that for this use case we assume that all normalizations and assays used by `mistyR` are defined as **alternative experiments with identical names in the assay and the experiment**.

```
sct.exp <- SummarizedExperiment(sct.data[slide.markers, ])
assayNames(sct.exp) <- "SCT"
altExp(spe, "SCT") <- sct.exp
```

Defining the parameters of the workflow

In this example we will explain the expression of hypoxia responsive genes in terms of three different views:

1) Main (intrinsic) view (containing genes to be predicted): intrinsic expression of `hypoxia.footprints`
2) Paraview - hypoxia genes: expression of hypoxia markers in a significance radius of 10 spots
3) Paraview - estrogen genes: expression of estrogen markers in a significance radius of 10 spots

```
# Define assay for each view
view.assays <- list(
  "main" = "SCT",
  "para.hypoxia" = "SCT",
  "para.estrogen" = "SCT"
)
```

```
# Define features for each view
view.features <- list(
  "main" = hypoxia.footprints,
  "para.hypoxia" = hypoxia.footprints,
  "para.estrogen" = estrogen.footprints
)

# Define spatial context for each view
view.types <- list(
  "main" = "intra",
  "para.hypoxia" = "para",
  "para.estrogen" = "para"
)

# Define additional parameters (l in the case of paraview)
view.params <- list(
  "main" = NULL,
  "para.hypoxia" = 10,
  "para.estrogen" = 10
)

misty.out <- "vignette_model_spe"
```

Run MISTy pipeline and collect results

Now that we have preprocessed the data and have decided on a question to analyze, we can create and run a *mistyR* workflow.

```
misty.results <- run_misty_spe(
  slide = spe,
  view.assays = view.assays,
  view.features = view.features,
  view.types = view.types,
  view.params = view.params,
  spot.ids = NULL, # Using the whole slide
  out.alias = misty.out
) %>%
  collect_results()
#>
#> Generating paraview
#>
#> Generating paraview
#>
#> Training models
#>
#> Collecting improvements
#>
#> Collecting contributions
#>
#> Collecting importances
#>
#> Aggregating
```

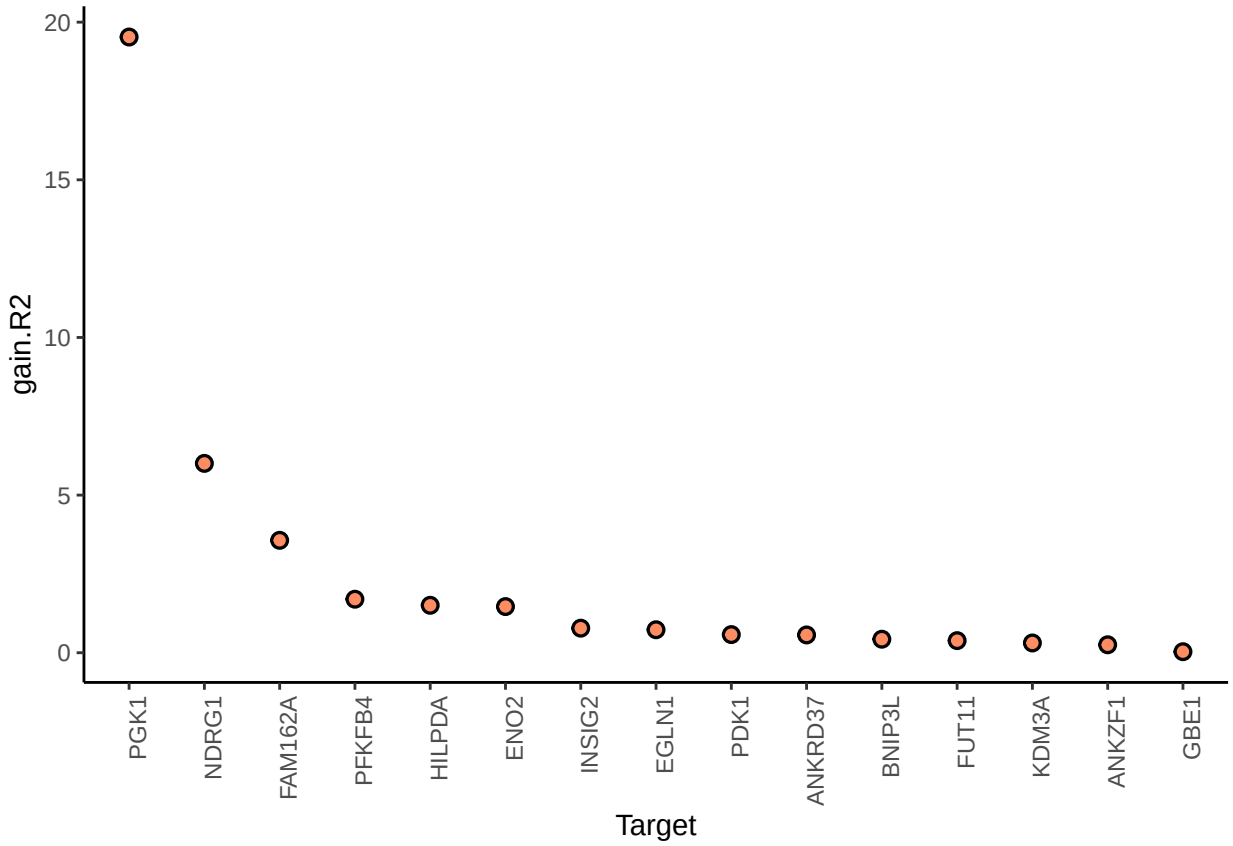
Interpretation and downstream analysis

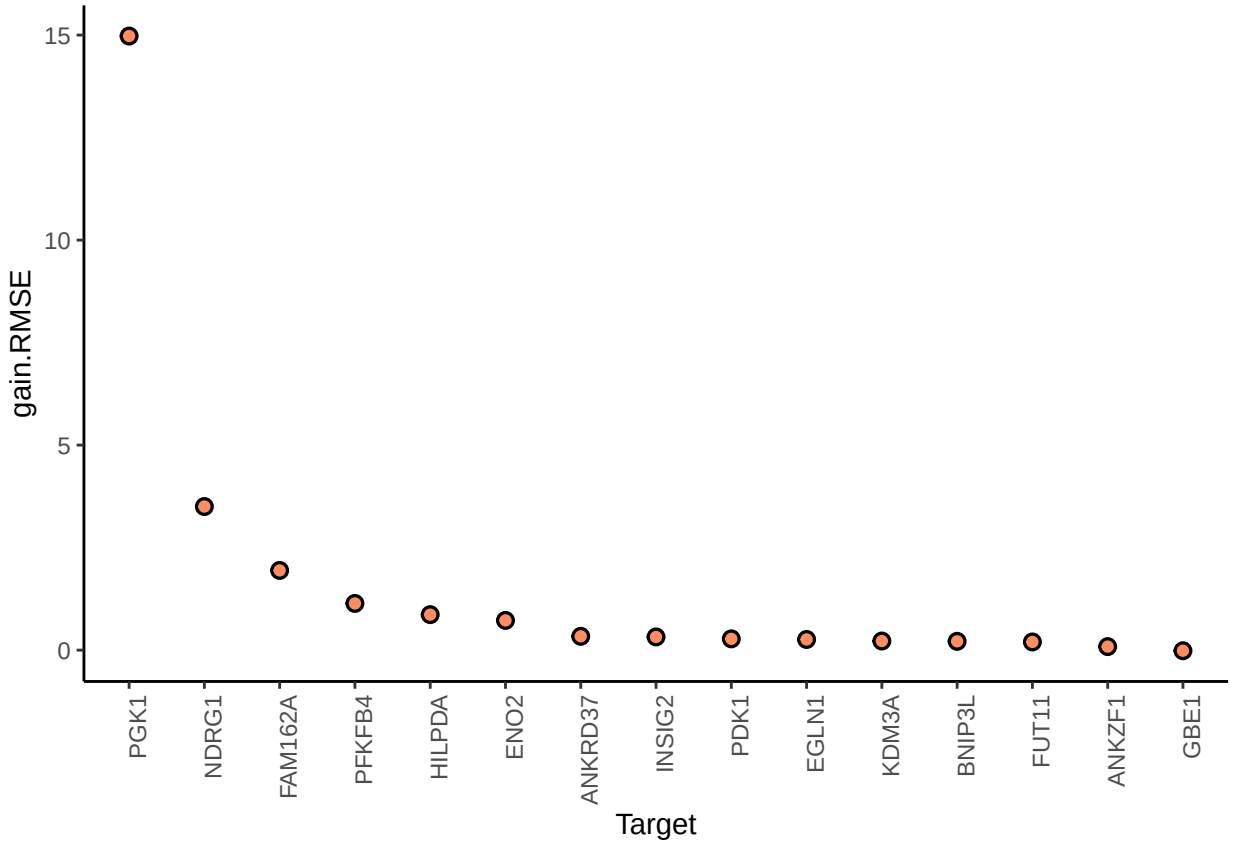
MISTy gives explanatory answers to three general questions:

1. How much can the broader spatial context explain the expression of markers (in contrast to the intraview)?

This can be observed in the gain in R2 (or RMSE) of using the multiview model in contrast to the single `main` view model.

```
misty.results %>%
  plot_improvement_stats("gain.R2") %>%
  plot_improvement_stats("gain.RMSE")
```


gain.R2
20
15
10
5
0
PGK1
NDRG1
FAM162A
PFKFB4
HILPDA
ENO2
INSIG2
EGLN1
PDK1
ANKRD37
BNIP3L
FUT11
KDM3A
ANKZF1
GBE1
Target


gain.RMSE
15
10
5
0
PGK1
NDRG1
FAM162A
PFKFB4
HILPDA
ENO2
ANKRD37
INSIG2
PDK1
EGLN1
KDM3A
BNIP3L
FUT11
ANKZF1
GBE1
Target

In this example, PGK1 is a marker whose expression can be explained better by modeling the broader spatial context around each spot.

We can further inspect the significance of the gain in variance explained, by the assigned p-value of improvement based on cross-validation.

```
misty.results$improvements %>%
  filter(measure == "p.R2") %>%
  arrange(value)
```

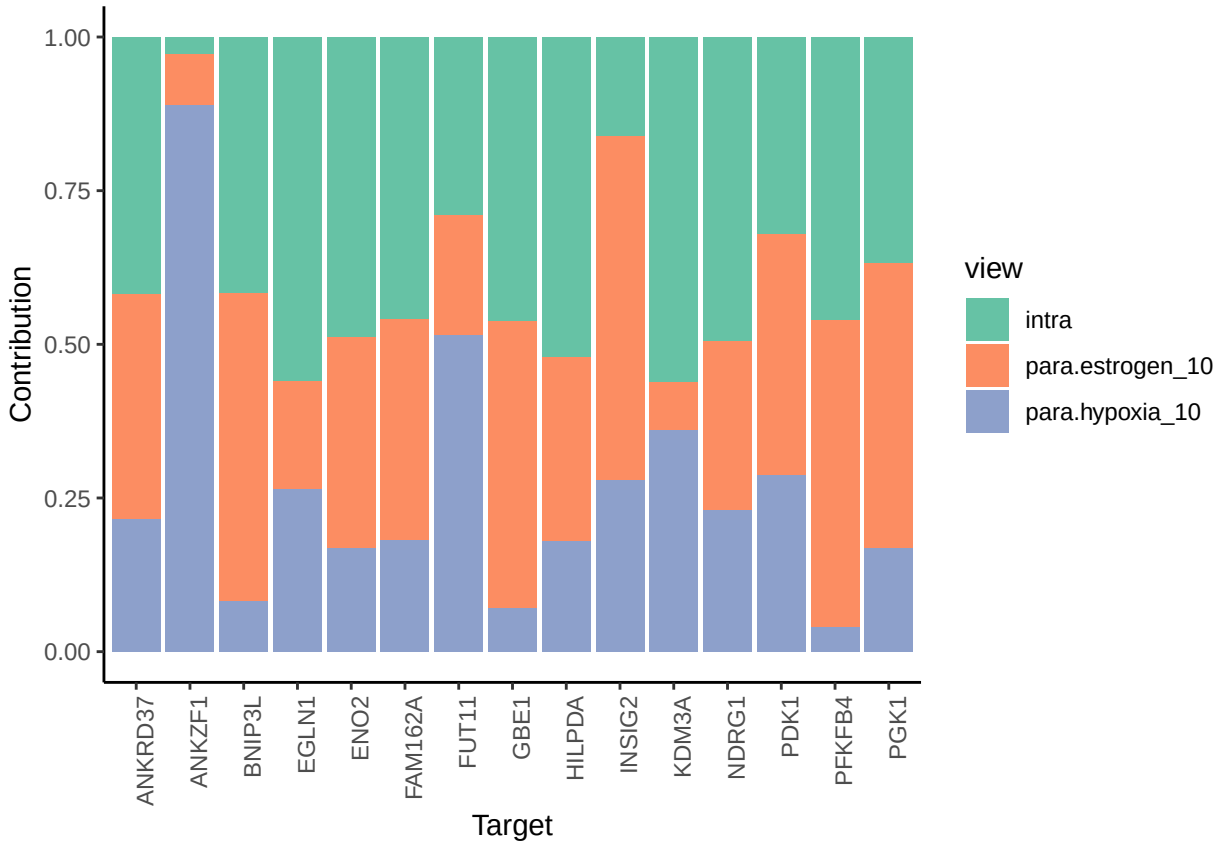
target	sample	measure	value
PGK1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0000767
FUT11	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0025297
ANKZF1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0101437
INSIG2	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0106670
PDK1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0358697
ENO2	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0650752
NDRG1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0711562
BNIP3L	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0722652
FAM162A	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.0796018
PFKFB4	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.1484128
EGLN1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.1521163
ANKRD37	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.1689154
HILPDA	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.1970744
KDM3A	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.2369597
GBE1	/tmp/RtmpfIImoi/Rbuild7407141443e76/mistyR/vignettes/vignette_model_shape	p.R2	0.2731579

In general, the significant gain in R2 can be interpreted as the following:

"We can better explain the expression of marker X, when we consider additional views, other than the intrinsic view."

2.How much do different view components contribute to explaining the expression?

```
misty.results %>% plot_view_contributions()
```




```
misty.results$contributions.stats %>% filter(target == "PGK1")
```

target	view	mean	fraction	p.mean	p.sd
PGK1	intra	0.4904152	0.3684802	0	NA
PGK1	para.estrogen_10	0.6167053	0.4633701	0	NA
PGK1	para.hypoxia_10	0.2237925	0.1681496	0	NA

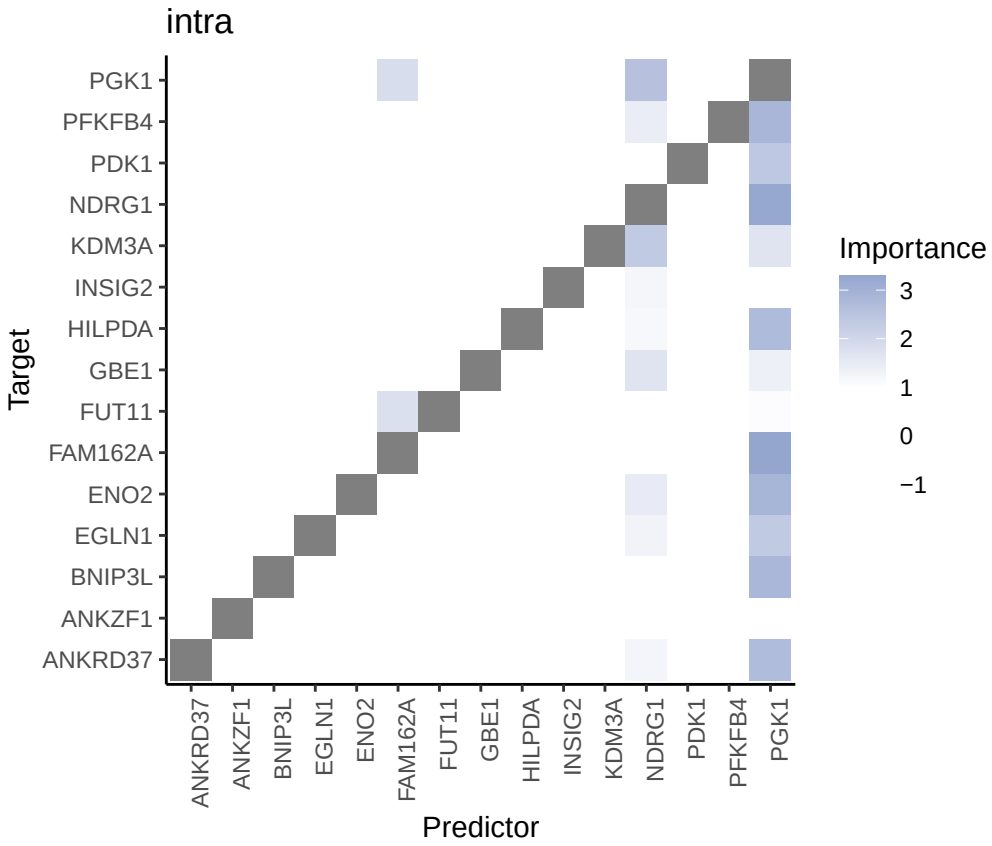
In the case of PGK1, we observe that around 37% of the contribution in the final model comes from the expression of other markers of hypoxia intrinsically or from the broader tissue structure. The rest (63%) comes from the expression of estrogen and hypoxia responsive genes from the broader tissue structure.

3.What are the specific relations that can explain the contributions?

To explain the contributions, we can visualize the importances of markers coming from each view separately as predictors of the expression of the intrinsic markers of hypoxia.

First, the intrinsic importances of the hypoxia markers.

```
misty.results %>% plot_interaction_heatmap(view = "intra")
```



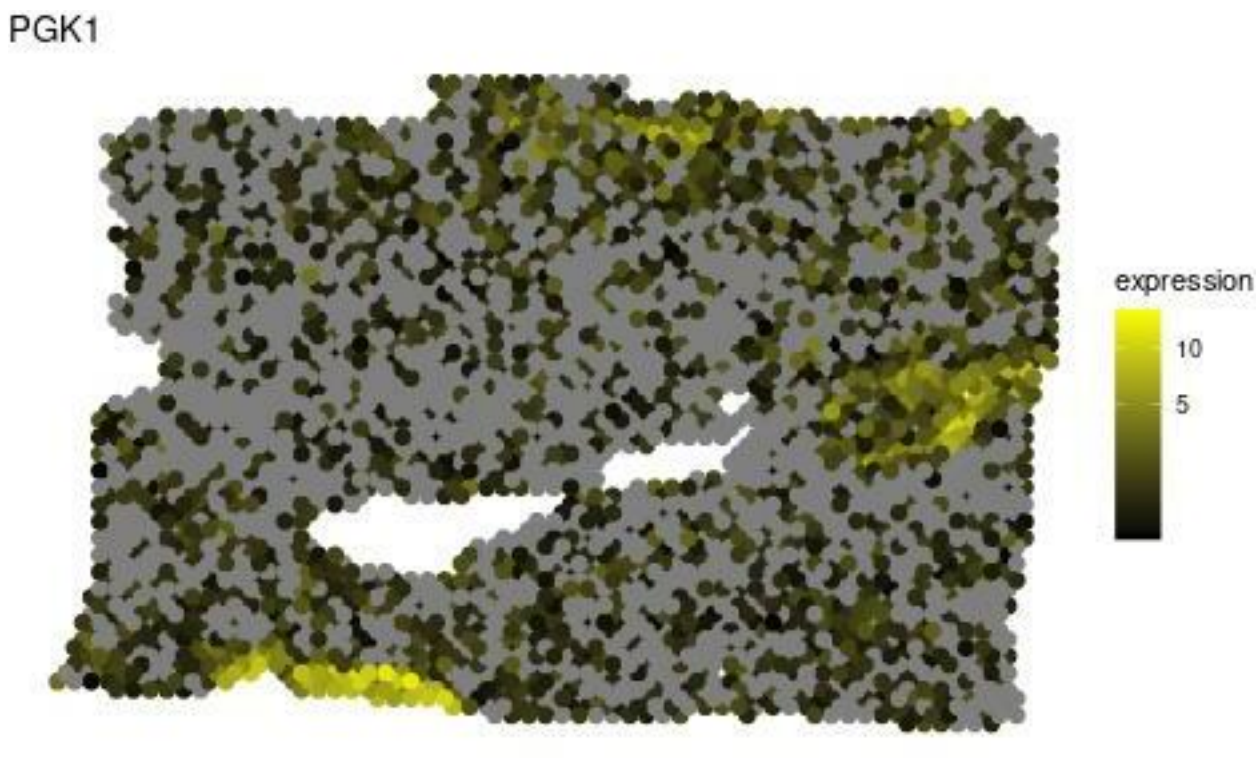
These importances are associated to the relationship between markers in the same spot. Let's pick the best predictor of PGK1 to confirm this:

```
misty.results$importances.aggregated %>%
  filter(view == "intra", Target == "PGK1") %>%
  arrange(-Importance)
```

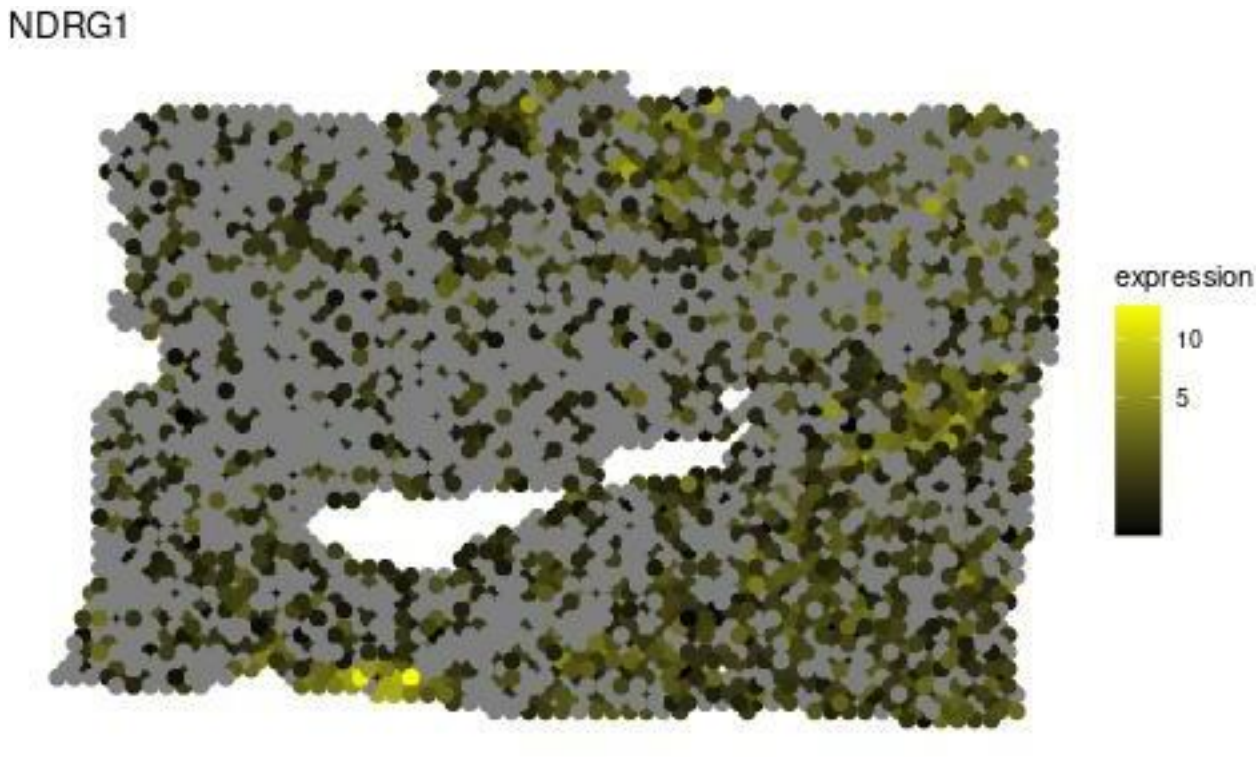
view	Predictor	Target	Importance	nsamples
intra	NDRG1	PGK1	2.5594726	1
intra	FAM162A	PGK1	1.8568832	1
intra	PFKFB4	PGK1	0.2742552	1
intra	HILPDA	PGK1	0.1953518	1
intra	ENO2	PGK1	0.0836979	1
intra	EGLN1	PGK1	-0.3324574	1
intra	ANKRD37	PGK1	-0.4088625	1
intra	KDM3A	PGK1	-0.5086982	1
intra	GBE1	PGK1	-0.5366770	1
intra	BNIP3L	PGK1	-0.5486009	1
intra	INSIG2	PGK1	-0.5544326	1
intra	ANKZF1	PGK1	-0.6321345	1
intra	FUT11	PGK1	-0.7139603	1
intra	PDK1	PGK1	-0.7338372	1
intra	PGK1	PGK1	NA	1

```
plotMolecules_adapted(spe,
  molecule = "PGK1",
```

```
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```

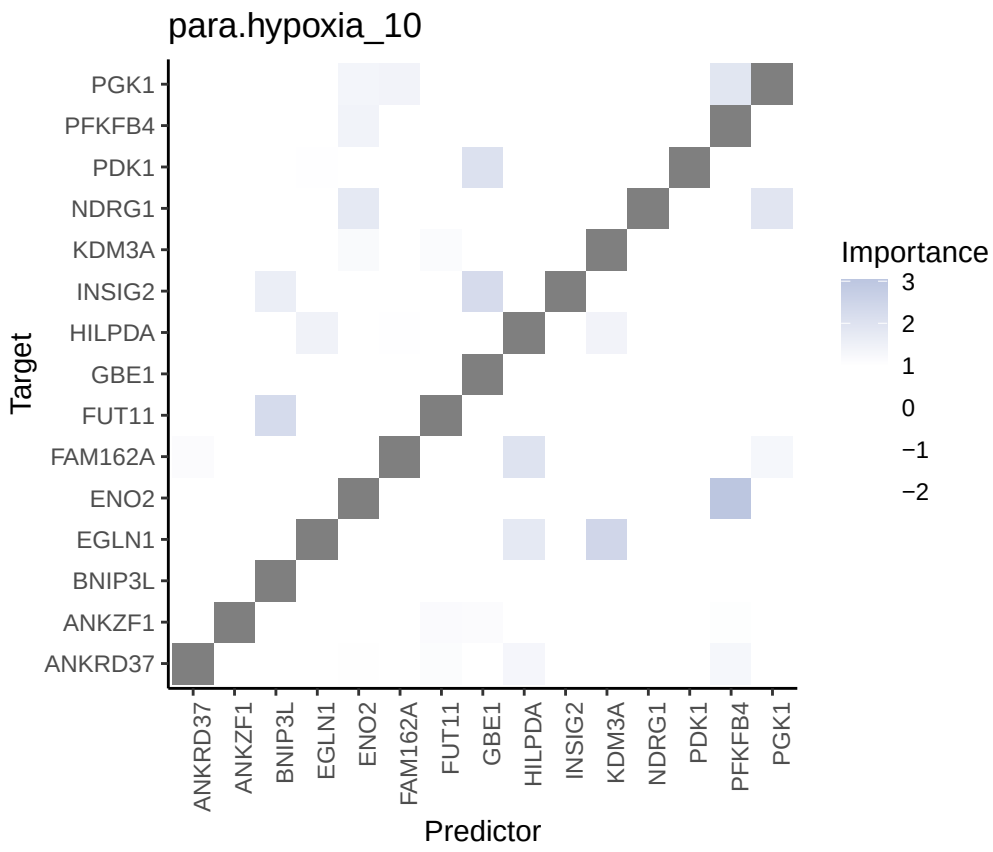



```
plotMolecules_adapted(spe,
  molecule = "NDRG1",
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```



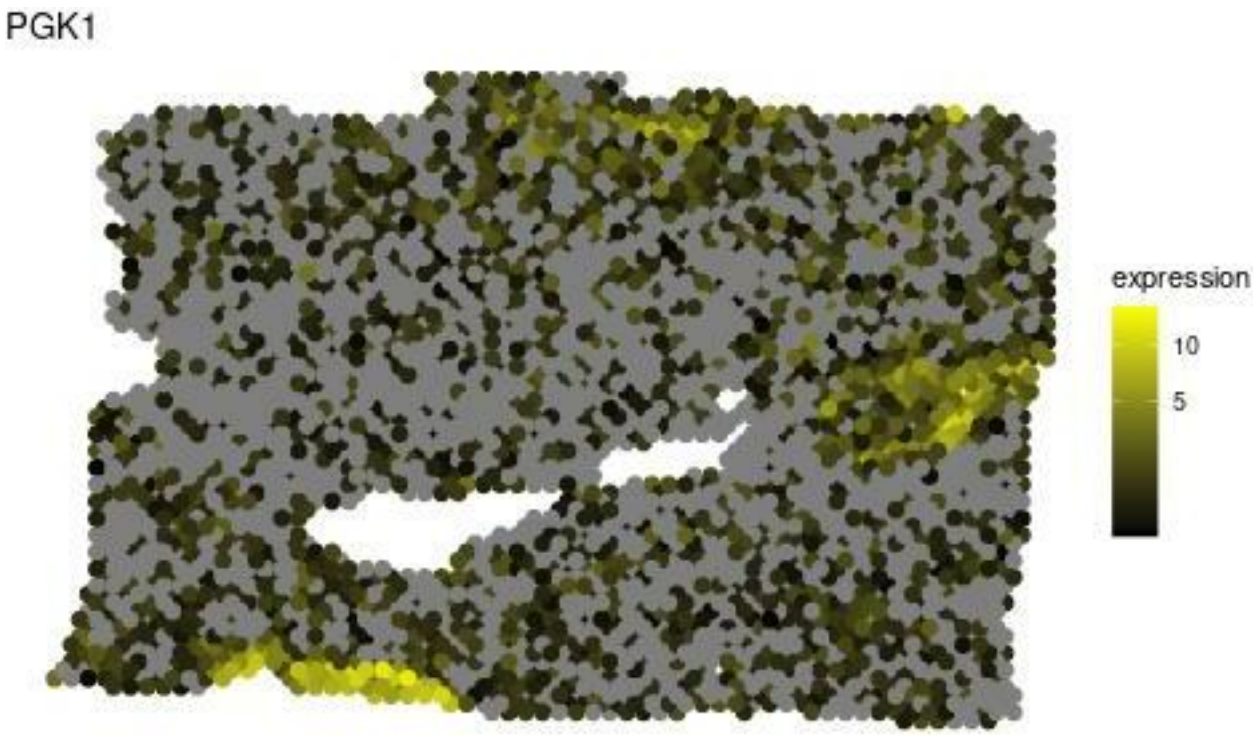
Second, the paraview importances of the hypoxia markers.

```
misty.results %>% plot_interaction_heatmap(view = "para.hypoxia_10")
```

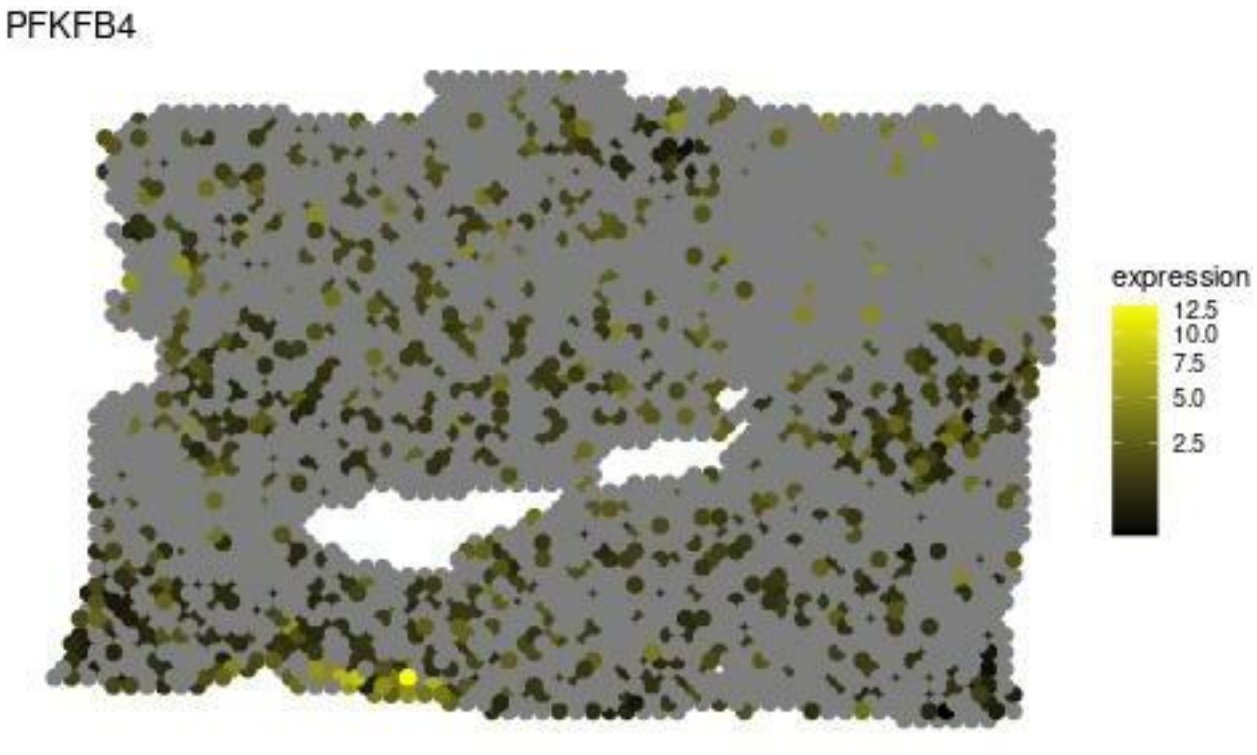


These importances are associated to the relationship between markers in the spot and markers in the neighborhood (controlled by our parameter l).

```
plotMolecules_adapted(spe,
  molecule = "PGK1",
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```



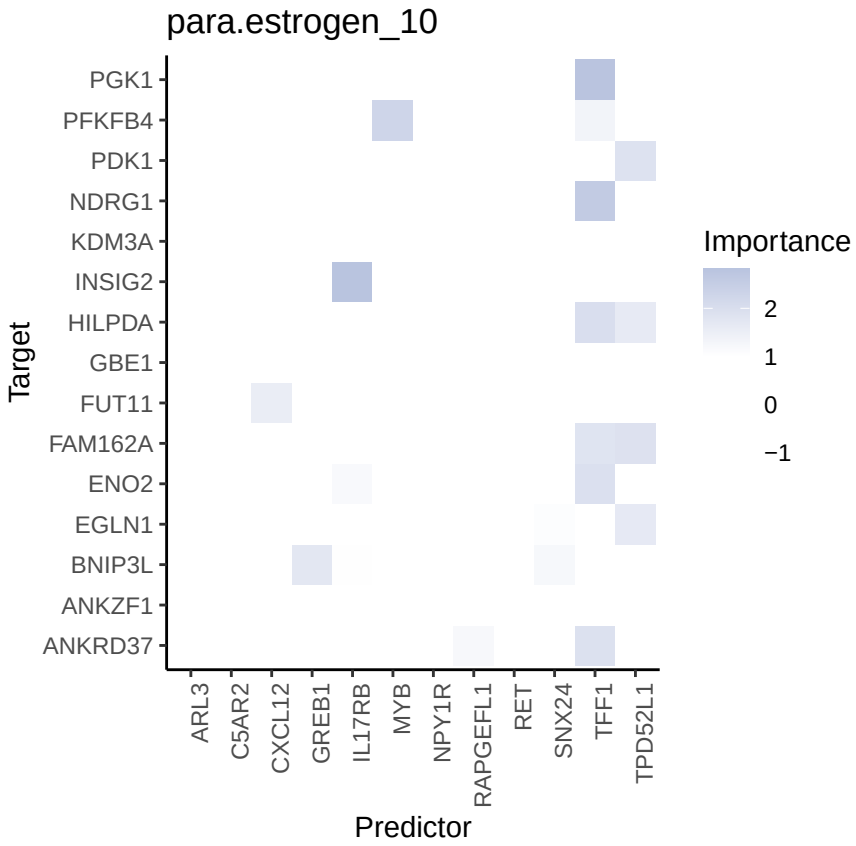

```
plotMolecules_adapted(spe,
  molecule = "PFKFB4",
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```



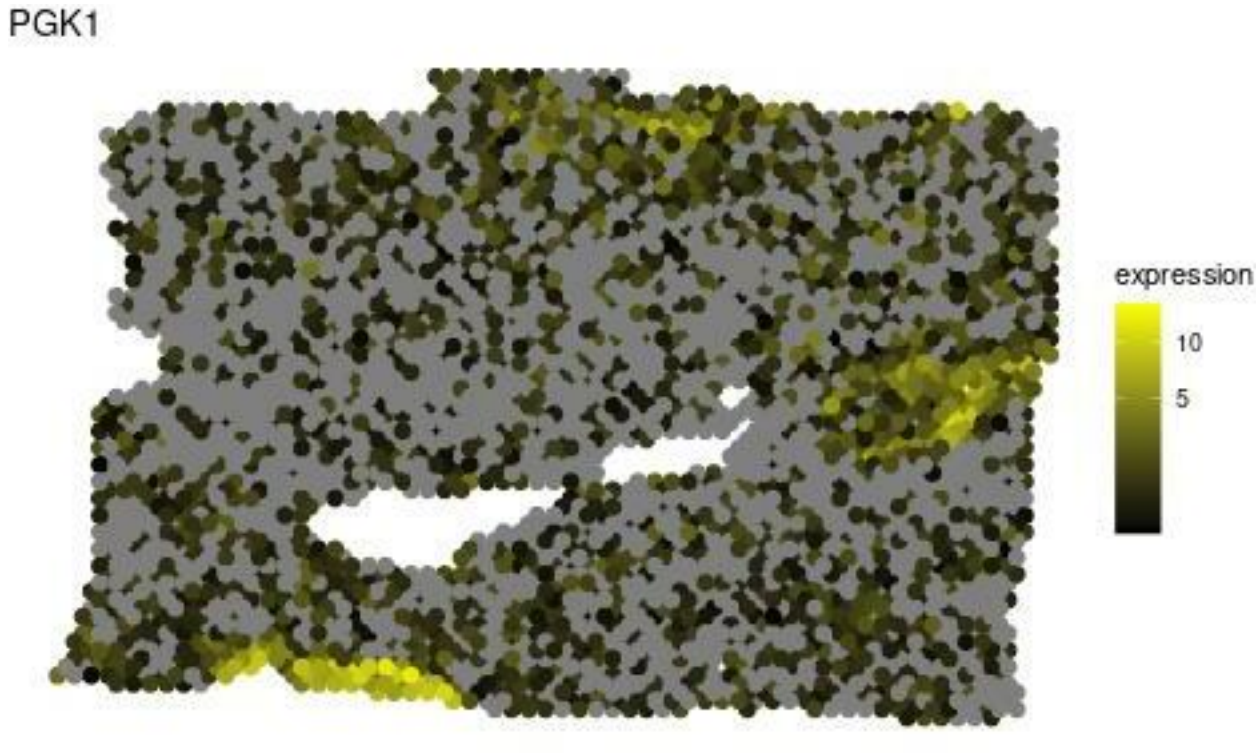
As expected, the expression of PFKFB4 (the best predictor from this view) in the neighborhood of each spot allows to explain the expression of PGK1.

Finally, the paraview importances of the estrogen markers. We will inspect the best predictor in this view.

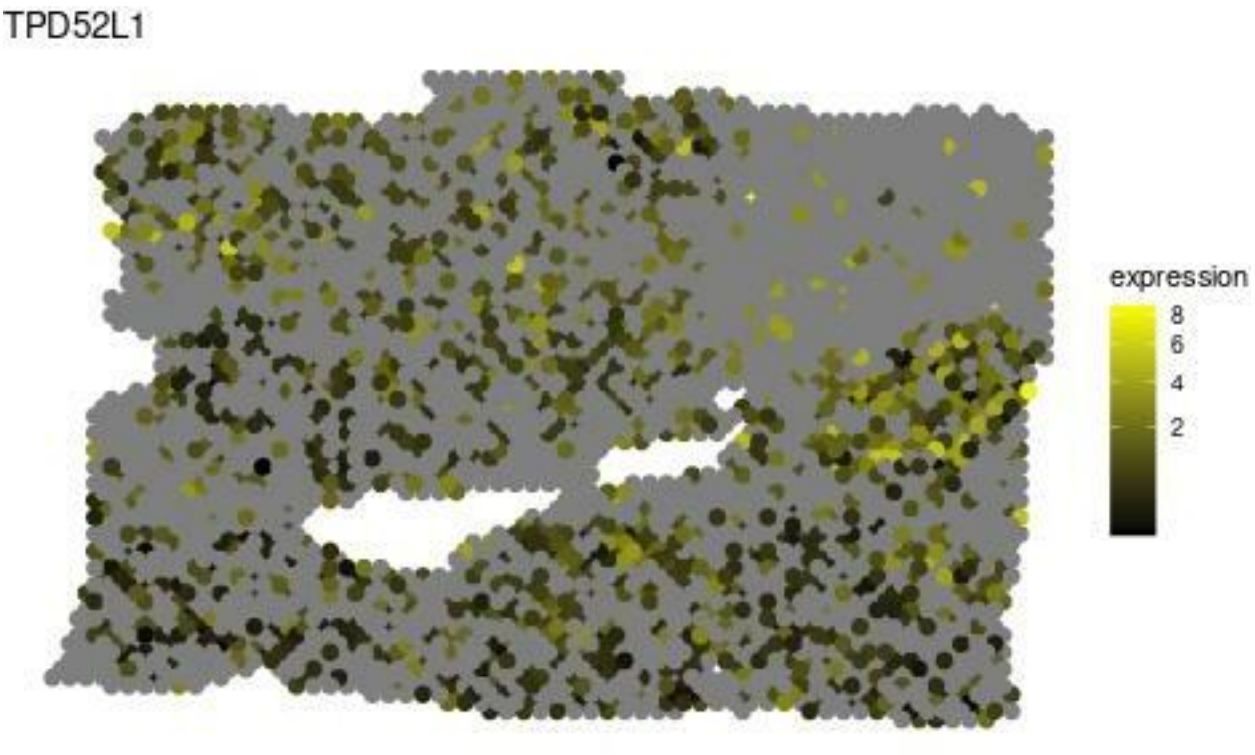
```
misty.results %>% plot_interaction_heatmap(view = "para.estrogen_10")
```




```
plotMolecules_adapted(spe,
  molecule = "PGK1",
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```




```
plotMolecules_adapted(spe,
  molecule = "TPD52L1",
  x_coord = "array_col",
  y_coord = "array_row",
  alt_assay = "SCT"
)
```



It is visible that in some areas the local expression of TPD52L1 overlaps with the areas with the highest expression of PGK1.

Important notes

- The relationships captured in the importances are not to assumed or interpreted as linear or casual.
- 1-to-1 importances between predictor and markers should always be interpreted in the context of the other predictors, since training MISTy models is multivariate predictive task.

Other use cases

The shown example is not the only way to use mistyR to analyze spatial transcriptomics data. Similar and complementary workflows can be constructed to describe different aspects of biology, for example:

- Spatial interactions between pathway activities and putative ligands, as shown here.
- Spatial interactions between cell-state lineage markers and putative ligands, as shown here.
- Spatial interactions between cell-type abundances leveraging deconvolution methods and creating descriptions of cell colocalization and tissue architecture.

Additionally, *mistyR* through the function `collect_results()` allows you to group the results of multiple slides, allowing for a more robust, integrative or comparative analysis of spatial interactions.

See also

More examples

```
browseVignettes("mistyR")
```

Online articles

Publication

Jovan Tanevski, Attila Gabor, Ricardo Omar Ramirez Flores, Denis Schapiro, Julio Saez-Rodriguez (2020). Explainable multi-view framework for dissecting inter-cellular signaling from highly multiplexed spatial data. bioRxiv. doi: 10.1101/2020.05.08.084145

Session info

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

```
#> R version 4.1.1 (2021-08-10)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 20.04.3 LTS
#>
#> Matrix products: default
#> BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
#> LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_GB              LC_COLLATE=C
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] progeny_1.15.5              sctransform_0.3.2
#>  [3] tibble_3.1.5                Matrix_1.3-4
#>  [5] SpatialExperiment_1.3.4     SingleCellExperiment_1.15.2
#>  [7] SummarizedExperiment_1.23.5 Biobase_2.53.0
#>  [9] GenomicRanges_1.45.0        GenomeInfoDb_1.29.10
#> [11] IRanges_2.27.2              S4Vectors_0.31.5
#> [13] BiocGenerics_0.39.2         MatrixGenerics_1.5.4
#> [15] matrixStats_0.61.0          ggplot2_3.3.5
#> [17] distances_0.1.8             purrr_0.3.4
#> [19] dplyr_1.0.7                 future_1.22.1
#> [21] mistyR_1.1.14               BiocStyle_2.21.4
#>
#> loaded via a namespace (and not attached):
#>  [1] bitops_1.0-7              filelock_1.0.2
#>  [3] RColorBrewer_1.1-2        tools_4.1.1
#>  [5] bslib_0.3.1               utf8_1.2.2
#>  [7] R6_2.5.1                  HDF5Array_1.21.0
#>  [9] DBI_1.1.1                 colorspace_2.0-2
#> [11] rhdf5filters_1.5.0        withr_2.4.2
#> [13] gridExtra_2.3             tidyselect_1.1.1
#> [15] compiler_4.1.1            cli_3.0.1
```

```
#> [17] DelayedArray_0.19.4       labeling_0.4.2
#> [19] bookdown_0.24             sass_0.4.0
#> [21] scales_1.1.1              stringr_1.4.0
#> [23] digest_0.6.28             rmarkdown_2.11
#> [25] R.utils_2.11.0            XVector_0.33.0
#> [27] pkgconfig_2.0.3           htmltools_0.5.2
#> [29] parallelly_1.28.1         sparseMatrixStats_1.5.3
#> [31] limma_3.49.5              fastmap_1.1.0
#> [33] highr_0.9                 rlang_0.4.12
#> [35] DelayedMatrixStats_1.15.4 jquerylib_0.1.4
#> [37] farver_2.1.0              generics_0.1.0
#> [39] jsonlite_1.7.2            BiocParallel_1.27.17
#> [41] R.oo_1.24.0               RCurl_1.98-1.5
#> [43] magrittr_2.0.1            rlist_0.4.6.2
#> [45] scuttle_1.3.1             GenomeInfoDbData_1.2.7
#> [47] Rhdf5lib_1.15.2           Rcpp_1.0.7
#> [49] munsell_0.5.0             fansi_0.5.0
#> [51] lifecycle_1.0.1           R.methodsS3_1.8.1
#> [53] furrr_0.2.3               edgeR_3.35.3
#> [55] stringi_1.7.5             yaml_2.2.1
#> [57] MASS_7.3-54               zlibbioc_1.39.0
#> [59] plyr_1.8.6                rhdf5_2.37.4
#> [61] grid_4.1.1                ggrepel_0.9.1
#> [63] dqrng_0.3.0               parallel_4.1.1
#> [65] listenv_0.8.0             crayon_1.4.1
#> [67] lattice_0.20-45           beachmat_2.9.1
#> [69] locfit_1.5-9.4            magick_2.7.3
#> [71] knitr_1.36                pillar_1.6.4
#> [73] igraph_1.2.7              rjson_0.2.20
#> [75] future.apply_1.8.1        reshape2_1.4.4
#> [77] codetools_0.2-18          glue_1.4.2
#> [79] evaluate_0.14             data.table_1.14.2
#> [81] BiocManager_1.30.16       vctrs_0.3.8
#> [83] gtable_0.3.0              tidyr_1.1.4
#> [85] assertthat_0.2.1          xfun_0.27
#> [87] DropletUtils_1.13.4       globals_0.14.0
#> [89] ellipsis_0.3.2
```