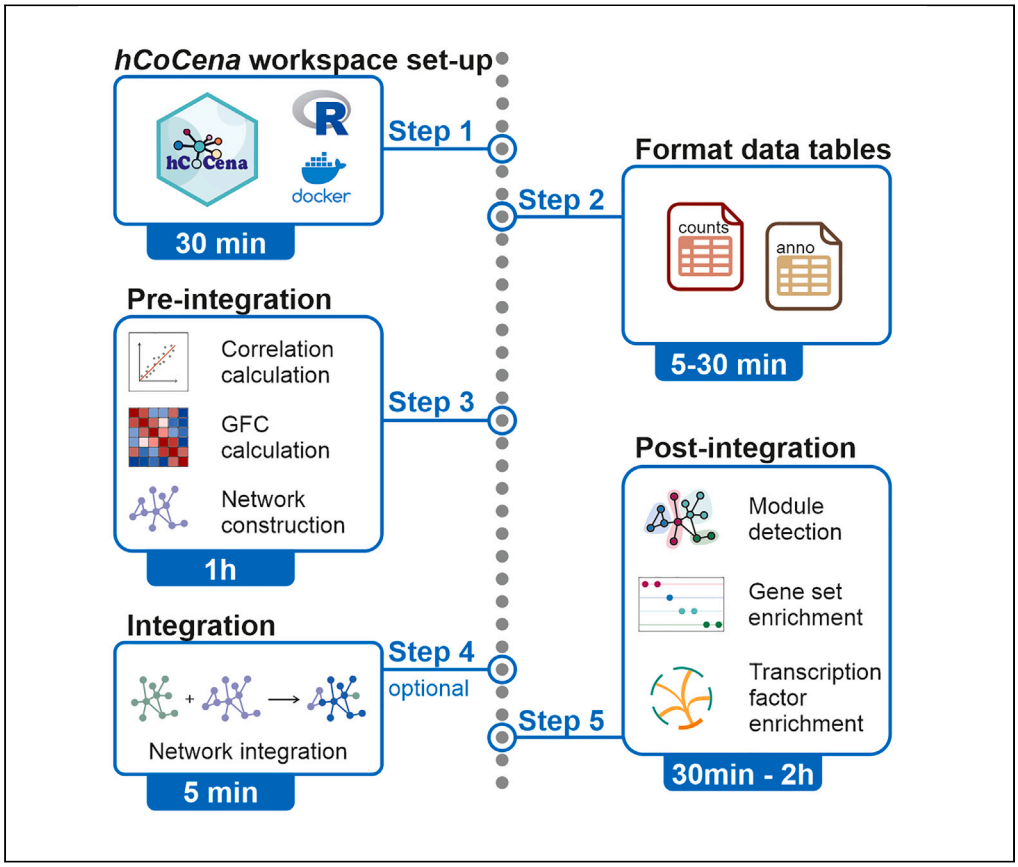
STAR Protocols

CellPress
OPEN ACCESS

Protocol

hCoCena: A toolbox for network-based co-expression analysis and horizontal integration of transcriptomic datasets



As the number and complexity of transcriptomic datasets increase, there is a rising demand for accessible and user-friendly analysis tools. Here, we present *hCoCena*: horizontal construction of co-expression networks and analysis, a toolbox facilitating the analysis of a single dataset, as well as the joint analysis of multiple datasets. We describe steps for workspace setup, formatting tables, data processing, and network integration. We then detail procedures for gene clustering, gene set enrichment analysis, and transcription factor enrichment analysis.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Lisa Holsten, Kilian Dahm, Marie Oestreich, Matthias Becker, Thomas Ulas

lisa.holsten@dzne.de (L.H.)
thomas.ulas@uni-bonn.de (T.U.)

Highlights

Calculation of gene expression correlations for network construction

Network-based integration of multiple transcriptomic datasets

Clustering of genes into modules with similar expression patterns across conditions

In-depth module characterization to elucidate biological changes across conditions

Holsten et al., STAR Protocols 5, 102922
March 15, 2024
https://doi.org/10.1016/j.xpro.2024.102922

STAR Protocols

CellPress
OPEN ACCESS

Protocol

hCoCena: A toolbox for network-based co-expression analysis and horizontal integration of transcriptomic datasets

Lisa Holsten,[1,2,3,4,6,*] Kilian Dahm,[1,2,4] Marie Oestreich,[1,5] Matthias Becker,[1,5] and Thomas Ulas[1,2,3,7,*]

[1]Systems Medicine, German Center for Neurodegenerative Diseases (DZNE), 53127 Bonn, Germany

[2]PRECISE Platform for Single Cell Genomics and Epigenomics, DZNE, and University of Bonn, 53127 Bonn, Germany

[3]Genomics and Immunoregulation, Life & Medical Sciences (LIMES) Institute, University of Bonn, 53115 Bonn, Germany

[4]Department of Pediatrics, University Hospital Würzburg, 97080 Würzburg, Germany

[5]Modular High-Performance Computing and Artificial Intelligence, German Center for Neurodegenerative Diseases (DZNE), 53127 Bonn, Germany

[6]Technical contact

[7]Lead contact

*Correspondence: lisa.holsten@dzne.de (L.H.), thomas.ulas@uni-bonn.de (T.U.)
https://doi.org/10.1016/j.xpro.2024.102922

SUMMARY

As the number and complexity of transcriptomic datasets increase, there is a rising demand for accessible and user-friendly analysis tools. Here, we present *hCoCena* (horizontal construction of co-expression networks and analysis), a toolbox facilitating the analysis of a single dataset, as well as the joint analysis of multiple datasets. We describe steps for workspace setup, formatting tables, data processing, and network integration. We then detail procedures for gene clustering, gene set enrichment analysis, and transcription factor enrichment analysis.

For complete details on the use and execution of this protocol, please refer to Oestreich et al.[1]

BEFORE YOU BEGIN

Transcriptomic datasets are continuously increasing in number and complexity, causing a growing demand for data analysis tools facilitating both the comprehensive analysis of a single dataset and the integrated analysis of multiple datasets. To address this need, we have developed *hCoCena* (horizontal construction of co-expression networks and analysis), a comprehensive toolbox for network-based gene co-expression analysis.[1]

hCoCena enables researchers to analyze a single transcriptomic dataset or multiple datasets in a unified analysis. The tool is suitable for the analysis and integration of transcriptomic data measuring the same entity (genes) across different sample spaces (data types), which has been referred to as horizontal data integration.[2] These data types include microarray, bulk RNA-sequencing (RNA-seq), and single-cell RNA-seq data aggregated into pseudobulks. Moreover, it can integrate datasets with varying numbers of genes and samples through a transformation-based integration approach based on gene co-expression networks. *hCoCena* is designed to meet the requirements of beginners and experienced data analysts by offering extensive documentation on the one hand, and high customizability on the other hand.

The following protocol outlines the specific steps of the *hCoCena* workflow, which is applied to demonstrate the horizontal integration and joint analysis of microarray and bulk RNA-seq data from macrophages treated with IFN-gamma (IFN-γ), Interleukin-4 (IL-4), or untreated (baseline).

The microarray and RNA-seq data were originally published by Xue et al. (2014)[3] and Beyer et al. (2012),[4] respectively, and pre-processed by Oestreich et al. (2022).[1] The processed datasets, analysis scripts, as well as detailed documentation, can be found in the *hCoCena* GitHub repository: https://github.com/MarieOestreich/hCoCena.

Throughout the protocol, text in squared brackets defines placeholders that need to be replaced by the user.

Prerequisites/systems requirement

1. Windows, Linux, or macOS systems that run Docker environments.
2. 64-bit processor.
3. 4 GB system RAM (8 GB recommended).

> ***Note:*** We recommend running the *hCoCena* analysis within an R Studio session based on the provided Docker image, which does not require the pre-installation of any R packages on your system. If you choose the local installation of *hCoCena*, it is essential to install R, R Studio, and all the necessary packages and their dependencies. Refer to Troubleshooting 1 for details.

Setting up an *hCoCena* workspace

Timing: 30 min

hCoCena is provided as an R package that runs on any operating system compatible with R. To simplify the setup of your R session, as well as to ensure the reproducibility of your *hCoCena* analysis, we provide an *hCoCena* Docker image. The Docker image is built on the latest *hCoCena* package version (v1.1.1) and includes all required packages and their dependencies. For a local installation of *hCoCena* refer to the instructions outlined in Troubleshooting 1.

4. Install Docker on your Windows system:
   a. Set up the Windows Subsystem for Linux 2 (WSL 2) and a Linux distribution:
      i. Install WSL2 by following the installation guide: https://learn.microsoft.com/en-us/windows/wsl/install. Ensure that the default version is WSL2 and not WSL1 before you proceed.
      ii. Install Ubuntu (or another Linux distribution) by following the installation guide: https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overview.
   b. Set up the Docker Desktop WSL2 backend by following the installation guide: https://docs.docker.com/desktop/wsl/.
   c. Create a Docker Desktop user account.
   d. Log into Docker Desktop via Ubuntu:
      i. Open the Ubuntu terminal.
      ii. Log in with your Docker Desktop user credentials:

```
>docker login
```

   e. Verify the installation of Docker Desktop by running:

```
>docker --version
```

> ⚠ CRITICAL: If your installation was successful, the Docker version will be displayed on the console. If no output is generated or if you receive an error message, please consult the

troubleshooting section of Docker desktop: https://docs.docker.com/desktop/troubleshoot/overview/.

Note: If you are using a different operating system, please refer to https://www.docker.com/ for installation instructions.

5. Start a Docker container with an R Studio session for your *hCoCena* analysis:
   a. Download the latest version of the *hCoCena* Docker image via the Ubuntu terminal:

```
>docker pull mo126/hcocena:v1.1.1
```

   b. Start a Docker container with the downloaded Docker image using the 'docker run' command and by replacing the text in squared brackets:
      i. Specify the port on your local host to which the port within the Docker container (8787) should be mapped to.
      ii. Specify a password to access the Docker container.
      iii. Mount a local directory in which you want to save the analysis results. This local directory will be mounted to "home/rstudio/output" within your Docker container.

```
>docker run -dp [YOUR_PORT]:8787 \
-e USER=rstudio \
-e PASSWORD=[YOUR_PASSWORD] \
--name hcocena_analysis \
-v [PATH_TO_LOCAL_DIRECTORY]:/home/rstudio/output \
mo126/hcocena:v1.1.1
```

   c. Open the R Studio session from your browser via the following address: http://localhost:[YOUR PORT].
   d. Log in with the username ("rstudio") and the password you have set in Step 5b.

Note: You can check the tag of the latest *hCoCena* Docker image on Docker hub (see Key resources table).

Note: Your port number can be any number between 1024 and 65535. However, you can only use each port number for one running Docker container. If you want to use multiple Docker containers in parallel, you need to use distinct port numbers on your local host, whereas the port number in your Docker container (8787) stays the same.

Format your count and annotation table

Timing: 5–30 min

hCoCena requires a count table, as well as a matching annotation table. You can upload both as delimiter-separated files or provide them as data frames within the R environment. The annotation file needs to contain at least the variable of interest (VOI) that you want to use for the grouping of your samples in the *hCoCena* analysis.

6. Format your count table:
   a. Format your count table for the upload from a file (Figure 1A):

A Expression table for upload in file format

GENE_SYMBOL	Sample_1	Sample_2	Sample_3	...
ACSL6	0.28	0.49	0.36	
CYBB	11.79	8.73	9.84	
TOP2A	3.53	1.21	4.65	
⋮				

B Expression table for upload from the environment

	Sample_1	Sample_2	Sample_3	...
ACSL6	0.28	0.49	0.36	
CYBB	11.79	8.73	9.84	
TOP2A	3.53	1.21	4.65	
⋮				

C Annotation table

	VOI	Var_1	Var_2	...
Sample_1	Group_A	Stim_1	28	
Sample_2	Group_B	Stim_1	37	
Sample_3	Group_B	Stim_2	31	
⋮				

Figure 1. Example of the expression and annotation table

(A) Example expression table with exemplary data for upload in file format.

(B) Example expression table with exemplary data for upload from the R environment.

(C) Example annotation table. VOI, variable of interest.

i. Ensure that your genes are in rows and your samples in columns.
ii. Include one column containing your gene symbol information.

b. Format your count table to be loaded from the R environment (Figure 1B):
i. Ensure that your genes are in rows and your samples in columns.
ii. Assign the gene symbol as rownames and ensure that there are no additional columns beside the Sample IDs.

7. Format your annotation table (Figure 1C):
a. Ensure that your annotation table contains the Sample IDs in rows and the metadata variables in columns.
b. Match the order of your Sample IDs to those of your Sample IDs in the columns of your count table.

⚠ CRITICAL: The rownames of the annotation file need to match the column names of the gene expression table.

⚠ CRITICAL: The annotation table needs to contain the VOI. If you are aiming at integrating multiple datasets, ensure that the column name of this VOI is identical in all annotation tables.

Note: We recommend importing data in a logarithmic scale that has been both variance-stabilized and, if needed, batch-corrected. For the processing of your data before the *hCoCena* workflow, we recommend the DESeq2 vignette by Michael I. Love, Simon Anders, and Wolfgang Huber: https://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Microarray and RNA-seq data (processed)	Oestreich et al.[1]	https://github.com/MarieOestreich/hCoCena/tree/main/STAR_protocol
Reference data files	Liberzon et al.[5]; The Gene Ontology Consortium[6]; Kaneshisa et al.[7]; Gillespie et al.[8]	https://github.com/MarieOestreich/hCoCena/tree/main/reference_files
Software and algorithms		
ChEA3	Keenan et al.[9]	https://maayanlab.cloud/chea3/
circlize v0.4.15	Gu et al.[10]	https://cran.r-project.org/package=circlize
clusterProfiler v4.10.0	Wu et al.[11]	https://bioconductor.org/packages/release/bioc/html/clusterProfiler.html
ComplexHeatmap v2.18.0	Gu et al.[12]	https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html
Cytoscape v3.10.1	Shannon et al.[13]	http://cytoscape.org; RRID: SCR_003032
Docker Desktop v4.25.2	Merkel[14]	https://www.docker.com/products/docker-desktop; RRID: SCR_016445
dplyr v1.1.4	Wickham et al.[15]	https://cran.r-project.org/package=dplyr
ggplot2 v3.4.4	Wickham et al.[16]	https://cran.r-project.org/package=ggplot2
hCoCena v1.1.1	Oestreich et al.[1]	https://github.com/MarieOestreich/hCoCena, https://doi.org/10.5281/zenodo.10475493
igraph v1.5.1	Csardi et al.[17]	https://cran.r-project.org/package=igraph
leidenAlg v1.1.2	Kharchenko et al.[18]	https://cran.r-project.org/package=leidenAlg
R v4.3.2	R Foundation for Statistical Computing[19]	https://cran.r-project.org/; RRID: SCR_001905
RCy3 v2.22.1	Gustavsen et al.[20]	https://bioconductor.org/packages/release/bioc/html/RCy3.html
RStudio v 2023.09.1 Build 494	Posit Team[21]	https://posit.co/products/open-source/rstudio/; RRID: SCR_000432
Other		
Code to reproduce the analysis	This publication	https://github.com/MarieOestreich/hCoCena/tree/main/STAR_protocol
mo126/hCoCena docker image v1.1.1	Oestreich et al.[1]	https://hub.docker.com/r/mo126/hcocena

STEP-BY-STEP METHOD DETAILS

In the following protocol, we will exemplify all steps of the *hCoCena* workflow which are graphically summarized in Figure 2. The entire R code to run an *hCoCena* analysis is provided in the form of R markdown files on GitHub (see key resources table). The main analysis steps are outlined within the 'hCoCena_main.Rmd' file, while all optional steps are outlined in the 'hCoCena_satellite.Rmd' file. The files adjusted for this showcase can be found within the respective sub-folder "STAR_protocol" on GitHub (see key resources table). You can execute the optional steps either when they are suggested within the protocol or at any later time point during the analysis.

The following commands are R commands that should be executed within the R Session you have started in the section before you begin.

Pre-integration phase

Timing: 1 h

This first phase of the *hCoCena* workflow includes all analysis steps that lead up to the actual integration procedure of the datasets. This includes the loading of the *hCoCena* package, the setup of the workspace and *hCoCena* object, defining all global (dataset-independent) and dataset-specific settings, the import of datasets, and the processing of the data for integration.

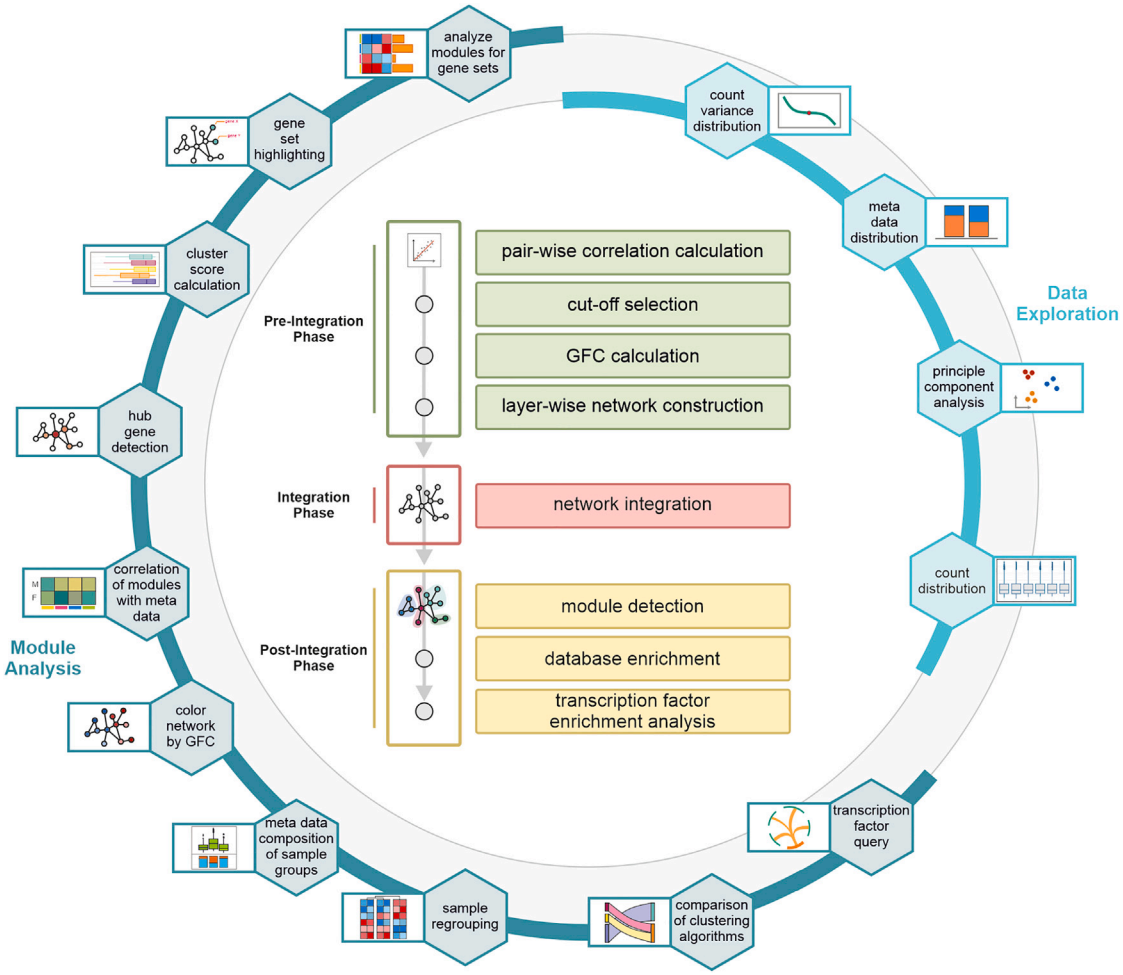


Figure 2. Schematic *hCoCena* overview

The main steps of the analysis workflow are depicted in the center. The surrounding circle illustrates optional functions. Optional functions are divided into data exploration functions that enable a first impression of the data, while the other functions are part of the module analysis and can only be applied once the co-expression modules have been determined in the main analysis. Figure reprinted in an adapted version with permission from Oestreich et al., 2022.

⚠ CRITICAL: Even if you do not want to integrate datasets, you still need to perform all the steps outlined within this workflow for your single dataset.

1. Load the *hCoCena* package:

```
>library(hcocena)
```

2. Create the *hCoCena* object ("hcobject"):

```
>init_object()
```

Note: You do not need to specify any variables within the 'init_object' function.

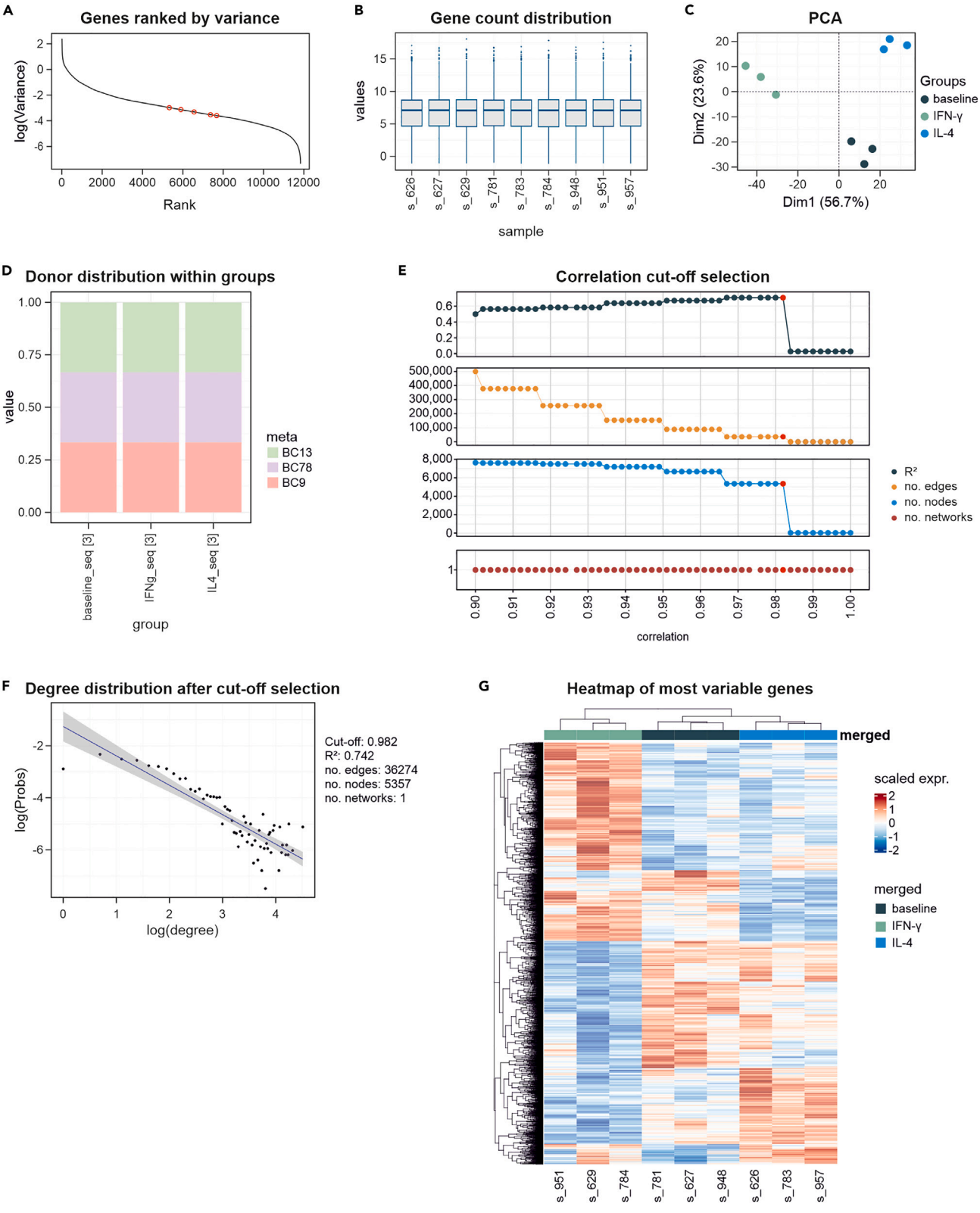

A
Genes ranked by variance
log(Variance)
Rank
B
Gene count distribution
values
sample
s_626
s_627
s_629
s_781
s_783
s_784
s_948
s_951
s_957
C
PCA
Dim2 (23.6%)
Dim1 (56.7%)
Groups
baseline
IFN-γ
IL-4
D
Donor distribution within groups
value
group
meta
BC13
BC78
BC9
baseline_seq [3]
IFNg_seq [3]
IL4_seq [3]
E
Correlation cut-off selection
correlation
R²
no. edges
no. nodes
no. networks
F
Degree distribution after cut-off selection
log(Probs)
log(degree)
Cut-off: 0.982
R²: 0.742
no. edges: 36274
no. nodes: 5357
no. networks: 1
G
Heatmap of most variable genes
merged
scaled expr.
merged
baseline
IFN-γ
IL-4

Figure 3. Pre-integration phase

(A) Genes ranked by variance, with red points indicating inflection points calculated for the RNA-seq dataset.

(B) Boxplots depicting gene expression values per sample within the RNA-seq dataset. Hinges correspond to the first and third quantiles, whiskers extend to the 1.5 * interquartile range.

(C) Principal component analysis (PCA) based on all genes of the RNA-seq dataset, with points colored by the variable of interest (VOI; merged).

(D) Barplots displaying the distribution of a metadata variable of interest ("donors") across the groups of the VOI ("merged") within the RNA-seq dataset.

(E) Correlation cut-off statistics for a defined range of cutoffs (0.90–1.00).

(F) Visualization of the logarithmic degree distribution and linear regression after applying the correlation cutoff on the RNA-seq dataset. Confidence interval is displayed in grey.

(G) Heatmap of mean expression values for network genes of the RNA-seq dataset scaled per row. Rows and columns are clustered by the complete linkage method based on the Euclidean distance.

Note: The 'hcobject' is designed to store all intermediate results of the *hCoCena* analysis. If you want to learn more about the structure and how to access specific data slots, please refer to the *hCoCena* Wiki on GitHub (see Key resources table).

3. Set up the working directory:
   a. Load the exemplary data and define the paths to where your expression data (count table), metadata (annotation table), and reference files are stored, as well as where the output of the *hCoCena* workflow should be saved.

```
>load("start_envo.RData")

>init_wd(dir_count_data = FALSE,

  dir_annotation = FALSE,

  dir_reference_files = ~/reference files/",

  dir_output = ~/output/")
```

Note: In this example, we are loading the count and annotation table from an existing R environment 'start_envo.RData', which is included in the Docker image, but can also be downloaded from the GitHub repository (see key resources table). Since we are loading the count and annotation table from an existing R environment, we set the respective paths in the function's arguments to FALSE. If you do not want to start from an existing R environment, skip the 'load' function and define the respective paths to your files.

Note: The reference files will be used e.g. for transcription factor and gene set enrichment analyses. If you are using the recommended Docker image, the folder "reference_files" will be loaded to your R Session automatically. For more information regarding the reference files, refer to Step 4 or the Wiki of the GitHub repository (see key resources table).

   b. Ensure that all defined paths exist:

```
>check_dirs()
```

Note: If one or multiple of the defined paths do not exist, a corresponding error message will be printed. To get help solving this error, please refer to Troubleshooting 2.

   c. Specify the name of the output folder that will be created within the previously specified output directory and in which the results of the analysis will be stored:

```
>init_save_folder(name = "STAR_protocol_output")
```

4. Define the names of the datasets, the respective count and annotation files, as well as the reference files to enable their import and correct storage within the 'hcobject':
   a. To assign names to your datasets and to specify the names of the count and annotation table, provide a named list of vectors with two elements - the first element is the name of the count table and the second element is the name of the annotation table. The list names will become the names of the datasets used throughout the *hCoCena* analysis.

```
>define_layers(list(Array = c("data_array", "annotation_array"),

RNA_Seq = c("data_seq", "annotation_seq"))
```

   b. Define the name of the transcription factor (TF) reference file, as well as all gene set files you want to utilize in the downstream analysis:

```
>set_supp_files(Tf = "TFcat.txt",

Hallmark = "h.all.v2023.1.Hs.symbols.gmt")
```

   Note: *hCoCena* supports Gene Set Enrichment Analysis (GSEA) based on the Gene Ontology (GO), Hallmark, Kyoto Encyclopedia of Genes and Genomes (KEGG), Reactome databases, and custom gene sets supplied in either CSV or GMT format. CSV files need to include two columns, with the first labeled as "term" and the second as "gene."

   c. Finally, load and store all defined datasets within the 'hcobject':

```
>read_data()

>read_supplementary()
```

   Note: If any expression dataset contains genes with no variance across all samples, these genes will be automatically removed. The original (unfiltered) expression data are stored in the "set_counts_unfiltered" slot of the *hCoCena* object.

   Note: If you are loading your count and/or annotation data from files, you need to specify additional information within the 'read_data' function. Please refer to the function's documentation for more information on parameters and settings.

   Note: If the loading of your data was not successful, please refer to Troubleshooting 3.

5. Define the global (dataset-independent) settings for the *hCoCena* analysis:

organism: Specify the organism based on your dataset. *hCoCena* currently supports human and mouse data. Set the parameter to "human" or "mouse", respectively.

control_keyword: Specify the name of your control group. The control group serves as the baseline reference for calculating the group-fold change (GFC). In cases where you lack control samples or wish to treat your control samples as a distinct group throughout the analysis, you can set this parameter to "none".

variable_of_interest: Specify the name of the column in your annotation table(s) that contain(s) the information for grouping the samples in the subsequent analysis.

min_nodes_number_for_network/min_nodes_number_for_cluster: Specify the minimum gene count required for creating a network or gene module. Networks and gene modules containing fewer genes will be omitted.

Range_GFC: Specify the range of the scaled GFC values.

layout_algorithm: Select the layout algorithm for network visualization. You can choose between "layout_with_fr" and "cytoscape". Cytoscape offers multiple layout algorithms but its local installation is required. The default layout algorithm in Cytoscape is the "Prefuse Force Directed Layout". Refer to Troubleshooting 4 for Cytoscape installation instructions.

data_in_log: If you provide your gene expression data in log scale, set this parameter to TRUE.

```
>set_global_settings(organism = "human",
  control_keyword = "none",
  variable_of_interest = "merged",
  min_nodes_number_for_network = 25,
  min_nodes_number_for_cluster = 25,
  range_GFC = 2.0,
  layout_algorithm = "cytoscape",
  data_in_log = T)
```

Note: *hCoCena* currently supports human and mouse data. However, the specification of the organism is only important for the analyses related to TFs. All the other functionalities of *hCoCena* are independent of the organism parameter and can be applied to data of any other organism.

Note: The selection of the layout algorithm solely impacts the visualization of the network and has no impact on the clustering or any other downstream results.

⚠ CRITICAL: If you want to integrate two datasets, both datasets need to derive from the same organism, the variable of interest needs to be present in both annotation tables, and the count data needs to be provided in the same scale (log or non-log, respectively).

Optional: You can define user-specific color vectors to be used for the visualization of the metadata variables:

```
>color_voi <-
  c("IL4_array" = "#00A1D5FF", "IFNg_array" = "#79AF97FF",
    "baseline_array" = "#374E55FF", "IL4_seq" = "#00A1D5FF",
    "IFNg_seq" = "#79AF97FF", "baseline_seq" = "#374E55FF")
>color_donor <-
  c("BC9" = "#FBB4AE", "BC11" = "#B3CDE3",
    "BC13" = "#CCEBC5", "BC78" = "#DECBE4")
```

Optional: In Step 6, you will need to define the number of top-variable genes to include in the downstream analysis. Setting this cutoff is valuable for excluding genes with very low variance, thereby eliminating technical noise and accelerating downstream calculations. You can determine the number of top-variable genes for each dataset in a data-driven manner by identifying the inflection point of the genes ranked by their variances (Figure 3A):

```
>suggest_topvar()
```

6. Define the layer-specific (dataset-dependent) settings for the *hCoCena* analysis:

top_var: Specify the number of top variable genes for each dataset, which will be used in the downstream analysis. We recommend setting this parameter using the 'suggest_topvar' function. However, if you prefer not to filter genes before performing correlation calculations, you can set this variable to "all".

min_corr: Specify the minimum correlation threshold between a pair of genes to be considered for the cut-off selection.

range_cutoff_length: Specify the number of distinct correlation cutoffs to investigate in the range extending from 'min_corr' up to a maximum correlation value of 1 during the data processing part I (Step 7).

print_distribution_plots: If you want to generate node-degree distribution plots for all tested cutoffs, you can set this parameter to "TRUE".

```
>set_layer_settings(top_var = c(7700, 7664),
   min_corr = c(0.9, 0.9),
   range_cutoff_length = c(50, 50),
   print_distribution_plots = c(F, F))
```

Note: You need to specify the variables for each dataset within a vector.

Note: The minimum correlation argument within this function is only used as a lower limit for the calculation of correlation cutoff statistics. The actual correlation cutoff will be defined after careful consideration of the resulting network parameters in Step 7.

Optional: You have the option to visualize the count distribution for every sample in each dataset to identify outliers or significant differences between samples or datasets (Figure 3B):

```
>plot_sample_distributions(plot_type = "boxplot", log_2 = F, plot = T)
```

Optional: In addition, you have the option to perform a principal component analysis (PCA) to detect trends, clusters, and outliers in each dataset based on all genes ("all") (Figure 3C). After you have executed Step 7, you can additionally perform the PCA on the top variable genes ("topvar") or the genes included in the network ("network").

```
>PCA(which = "all", color_by = c("merged", "merged"), cols = color_voi)
```

Optional: Moreover, you can visualize the distribution of continuous and categorical metadata variables among the groups of the VOI. You need to specify the dataset ("layer") for which you wish to plot the distribution, along with the name and the class of the metadata variable you are interested in (Figure 3D).

```
>meta_plot(set = 2, group_col = "merged", meta_col = "Donor", type = "cat", cols = color_donor)
```

7. Data processing:
   a. Filter the datasets for the most variable genes as defined in Step 6. Subsequently, compute the correlation for every pair of genes within each dataset:

```
>run_expression_analysis_1(corr_method = "spearman")
```

Note: You can choose between the "pearson", "spearman", and "rho" correlation calculation methods by specifying the 'corr_method' variable accordingly.

   b. Filter both datasets based on a suitable correlation cut-off to remove all connections between genes where the correlation value is below the chosen cut-off.
      i. Determine a suitable correlation cut-off by examining the statistics of the resulting network (Figure 3E):

```
>plot_cutoffs(interactive = T)
```

Note: The objective is to identify a correlation cut-off that results in a network with a high R^2-value, indicative of scale-free topology, a high number of nodes (genes) to minimize the loss of information, and a limited number of edges (connections) to enhance gene clustering into distinct modules. If none of the cutoffs seem suitable, consider decreasing the lower limit for the correlation cutoff chosen in Step 6.

      ii. Set the correlation cut-off for each dataset:

```
>set_cutoff(cutoff_vector = c(0.982, 0.982))
```

      iii. Confirm the scale-free topology by visualizing the logarithmic degree distribution and the linear regression for each dataset after applying the respective correlation cut-off (Figure 3F):

```
>plot_deg_dist()
```

Note: While biological networks are generally thought to follow scale-free topology,[22] experience has shown that many transcriptomic datasets may not perfectly conform to this principle.

      iv. Visualize the scaled expression of all genes remaining in the downstream analysis for each sample and dataset in a heatmap (Figure 3G). This helps you to verify that the set of genes selected for the downstream analysis accurately represents your samples and groups. Additionally, this function computes the GFC for each gene and each group within each dataset, either in comparison to the control group defined in Step 5 or to the mean of all groups if no control group was defined as in our example.

```
>run_expression_analysis_2(cols = list("merged" = color_voi))
```

Integration phase

Timing: 5 min

Following the successful processing of both datasets, this second phase of the *hCoCena* workflow proceeds with the integration of the datasets for a joint downstream analysis.

8. Integrate the two previously constructed networks into a single merged network based on the union ("u") or the intersection ("i") of the two networks. Moreover, for edges that exist in both networks with varying edge weights, decide whether to incorporate the minimum ("min"), mean ("mean", or maximum ("max") edge weight into the integrated network.

```
>build_integrated_network(mode = "u", multi_edges = "min")
```

Note: If you select the integration based on the union, all nodes and edges will be retained in the integrated network, allowing you to inspect shared as well as unique features. If you select the intersection, only nodes and edges that exist in both networks will be conserved in the integrated network, limiting your analyses to shared features.

⚠ CRITICAL: You need to execute this function, even if you are only analyzing a single dataset. In this case, you do not need to specify any variables.

Post-integration phase

Timing: 30 min to 2 h

During this third phase of the *hCoCena* workflow, you have the option to perform various analyses based on the integrated network. These analyses encompass tasks such as identifying gene modules with similar expression patterns, creating network visualizations, conducting GSEA, and performing TFEA.

9. Module detection:
   a. Define clusters (modules) of genes with strong co-expression using a clustering algorithm of your preference:

```
>cluster_calculation(cluster_algo = "cluster_leiden",
  no_of_iterations = 2,
  resolution = 0.1,
    partition_type = "ModularityVertexPartition",
    max_cluster_count_per_gene = 1)
```

Note: We recommend the use of the Leiden clustering algorithm. Nonetheless, *hCoCena* also supports additional clustering algorithms. For more details, refer to the function's documentation or the Wiki on GitHub (see Key resources table).

Note: The default clustering resolution of the Leiden algorithm is 0.1. By adjusting the resolution, you can modulate the number of modules, with an increase resulting in more modules, and a decrease resulting in fewer modules.

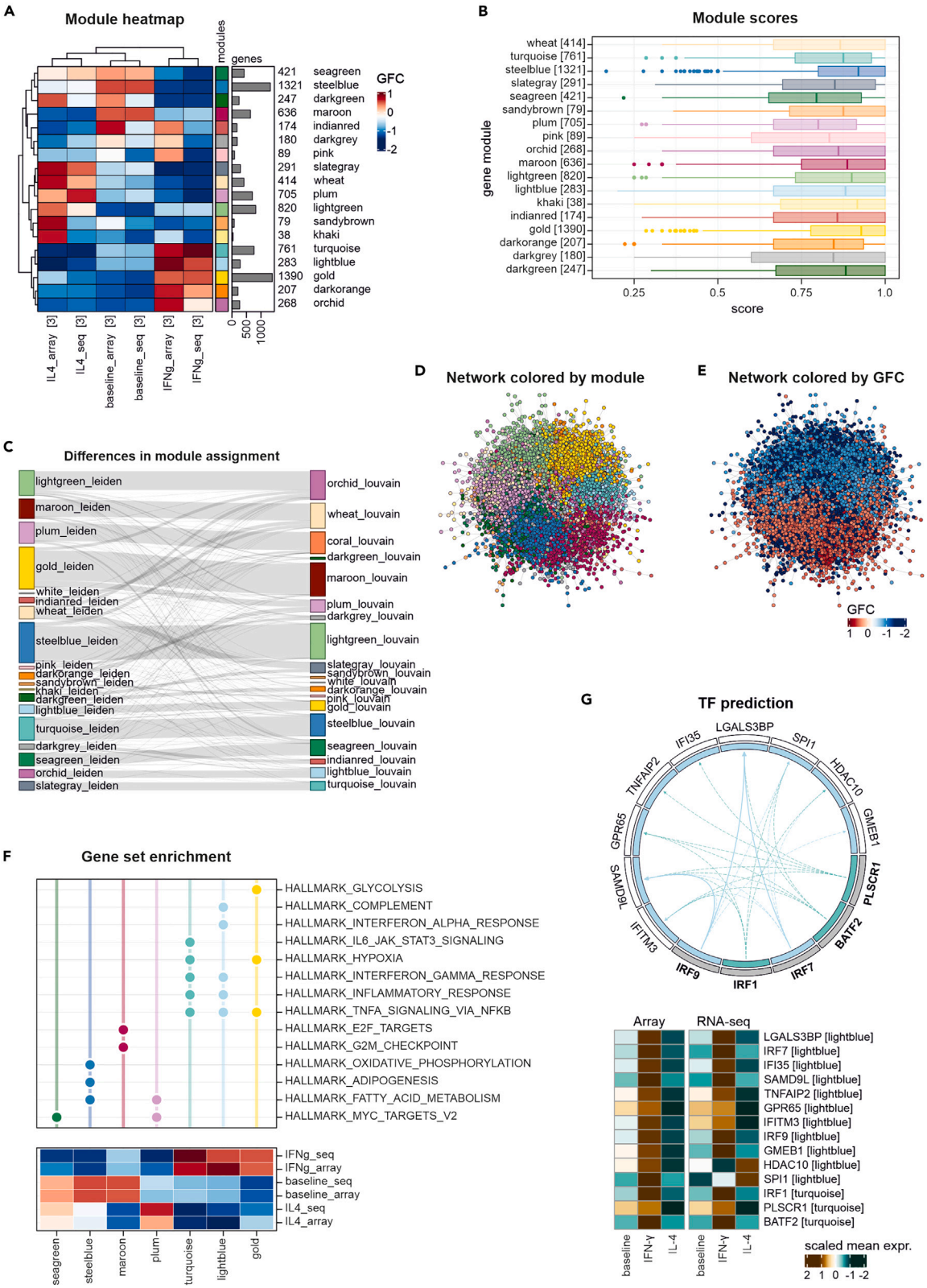

A
Module heatmap
modules
genes
GFC
1
0
-1
-2
421 seagreen
1321 steelblue
247 darkgreen
636 maroon
174 indianred
180 darkgrey
89 pink
291 slategray
414 wheat
705 plum
820 lightgreen
79 sandybrown
38 khaki
761 turquoise
283 lightblue
1390 gold
207 darkorange
268 orchid
0
500
1000
IL4_array [3]
IL4_seq [3]
baseline_array [3]
baseline_seq [3]
IFNg_array [3]
IFNg_seq [3]
B
Module scores
gene module
wheat [414]
turquoise [761]
steelblue [1321]
slategray [291]
seagreen [421]
sandybrown [79]
plum [705]
pink [89]
orchid [268]
maroon [636]
lightgreen [820]
lightblue [283]
khaki [38]
indianred [174]
gold [1390]
darkorange [207]
darkgrey [180]
darkgreen [247]
0.25
0.5
0.75
1.0
score
C
Differences in module assignment
lightgreen_leiden
maroon_leiden
plum_leiden
gold_leiden
white_leiden
indianred_leiden
wheat_leiden
steelblue_leiden
pink_leiden
darkorange_leiden
sandybrown_leiden
khaki_leiden
darkgreen_leiden
lightblue_leiden
turquoise_leiden
darkgrey_leiden
seagreen_leiden
orchid_leiden
slategray_leiden
orchid_louvain
wheat_louvain
coral_louvain
darkgreen_louvain
maroon_louvain
plum_louvain
darkgrey_louvain
lightgreen_louvain
slategray_louvain
sandybrown_louvain
white_louvain
darkorange_louvain
pink_louvain
gold_louvain
steelblue_louvain
seagreen_louvain
indianred_louvain
lightblue_louvain
turquoise_louvain
D
Network colored by module
E
Network colored by GFC
GFC
1 0 -1 -2
F
Gene set enrichment
HALLMARK_GLYCOLYSIS
HALLMARK_COMPLEMENT
HALLMARK_INTERFERON_ALPHA_RESPONSE
HALLMARK_IL6_JAK_STAT3_SIGNALING
HALLMARK_HYPOXIA
HALLMARK_INTERFERON_GAMMA_RESPONSE
HALLMARK_INFLAMMATORY_RESPONSE
HALLMARK_TNFA_SIGNALING_VIA_NFKB
HALLMARK_E2F_TARGETS
HALLMARK_G2M_CHECKPOINT
HALLMARK_OXIDATIVE_PHOSPHORYLATION
HALLMARK_ADIPOGENESIS
HALLMARK_FATTY_ACID_METABOLISM
HALLMARK_MYC_TARGETS_V2
IFNg_seq
IFNg_array
baseline_seq
baseline_array
IL4_seq
IL4_array
seagreen
steelblue
maroon
plum
turquoise
lightblue
gold
G
TF prediction
LGALS3BP
SPI1
HDAC10
GMEB1
PLSCR1
BATF2
IRF7
IRF1
IRF9
IFITM3
SAMD9L
GPR65
TNFAIP2
IFI35
Array
RNA-seq
LGALS3BP [lightblue]
IRF7 [lightblue]
IFI35 [lightblue]
SAMD9L [lightblue]
TNFAIP2 [lightblue]
GPR65 [lightblue]
IFITM3 [lightblue]
IRF9 [lightblue]
GMEB1 [lightblue]
HDAC10 [lightblue]
SPI1 [lightblue]
IRF1 [turquoise]
PLSCR1 [turquoise]
BATF2 [turquoise]
baseline
IFN-γ
IL-4
baseline
IFN-γ
IL-4
scaled mean expr.
2 1 0 -1 -2

Figure 4. Post-integration phase I

(A) Module heatmap displaying the mean group fold-change (GFC) per group and gene module. Module sizes are indicated by numbers and barplots on the right side. Rows and columns are clustered by the complete linkage method based on the Euclidean distance.

(B) Boxplot illustrating the module score for each gene within the corresponding gene module. Hinges correspond to the first and third quantiles, whiskers extend to the 1.5 * interquartile range.

(C) Alluvial plot visualizing differences in module assignment between the Leiden and Louvain algorithms.

(D) Network visualization with genes colored according to their module.

(E) Network visualization with genes colored by the GFC of a selected group of the VOI ("baseline_RNA_seq").

(F) DJ plot illustrating the top 5 significantly enriched terms (if available) for each selected module based on the selected gene set ("Hallmark"). Significance was determined by Benjamini-Hochberg correction with an adjusted p-value cutoff of 0.1.

(G) Top: Circos plot displaying the result of the transcription factor (TF) enrichment for the module of interest ("light blue"). The outer circle highlights TFs in gray and targets in white. The inner circle indicates the module the respective gene was assigned to. Arrows connect TFs to their targets, with solid lines indicating the presence and dashed lines indicating the absence of predicted connections within the network. Bottom: Heatmap visualizing the mean expression values of the top predicted TFs and their top targets scaled per row.

Note: Clustering will be executed with the specified number of iterations. Only genes consistently assigned to the same gene module in all iterations (if max_cluster_count_per_gene = 1) will be considered for the downstream analysis.

b. Visualize the mean GFC of all genes within the respective modules for each group in a heatmap (Figure 4A):

```
>plot_cluster_heatmap()
```

Optional: You can export the clustering result to a text file:

```
>export_clusters()
```

Optional: You can import a clustering model as a text file with two columns. The first column should contain the gene symbol, while the second column should contain the module color:

```
>import_clusters(file = "[YOUR_FILE].txt")
```

Optional: To evaluate the degree to which each gene belongs to its assigned module, you can generate a boxplot illustrating the ratio of each gene's connections to genes within the same module to its overall number of connections (Figure 4B). The higher the module score, the more discrete is your clustering.

```
>get_module_scores()
```

Note: If the module scores are low, you may inspect the outcome upon clustering with a different resolution (only available for the Leiden algorithm; see Step 9) or another community detection algorithm (see optional step below).

Optional: You can assess the outcome of all clustering algorithms supported by *hCoCena* using alluvial plots (Figure 4C) that visualize the differences in module assignment for each gene, as well as by PCA of the module's gene expression. If you identify a clustering algorithm that describes the structure of your data better, you can update the clustering algorithm accordingly:

```
>algo_alluvial()

>PCA_algo_compare()

>update_clustering_algorithm(new_algo = "[ALGORITHM_OF_CHOICE]")
```

10. Generate a network visualization where nodes are colored according to their assigned module:
    a. Unfortunately, direct communication with Cytoscape from the Docker container is not possible. Therefore, you need to export the network to your output folder:

```
>export_to_local_folder()
```

    b. Open Cytoscape:
        i. Press "Import Network From File System" and select the "network.txt" file within your output folder, which was generated by the above function.
        ii. Set the "from" column to "Source Node" and the "to" column to "Target Node", then press "ok".
        iii. The default layout algorithm ("Prefuse Force Directed Layout") will be applied automatically, but you can apply any other algorithm available in Cytoscape.
    c. To export the layout information, open a local R Studio session:
        i. Install the RCy3 package:

```
>BiocManager::install(package = "RCy3")
```

        ii. Import the layout from Cytoscape into your local R session.

```
>layout <- base::as.matrix(RCy3::getNodePosition())
```

        iii. Save the layout matrix in your *hCoCena* output folder by adapting the path accordingly:

```
>utils::write.csv(x = layout,

file = base::paste0("[path_to_local_output_dir]",

"/network_layout.csv"), row.names = T)
```

    d. Now, proceed within your Docker container to import the layout and save it within the hcobject:

```
>import_layout_from_local_folder()
```

    e. Visualize the network (Figure 4D):

```
>plot_integrated_network(layout =

hcobject[["integrated_output"]][["cluster_calc"]][["layout"]])
```

Note: Instead of re-running these steps, you can import the file "cytoscape_session_STAR.cys", which we saved within the "STAR_protocol" folder, into Cytoscape. In addition, you can directly import the network layout into your R session within the Docker container by copying the "network_layout.csv" file from the "STAR_protocol" folder into your *hCoCena* output folder and executing Step d.

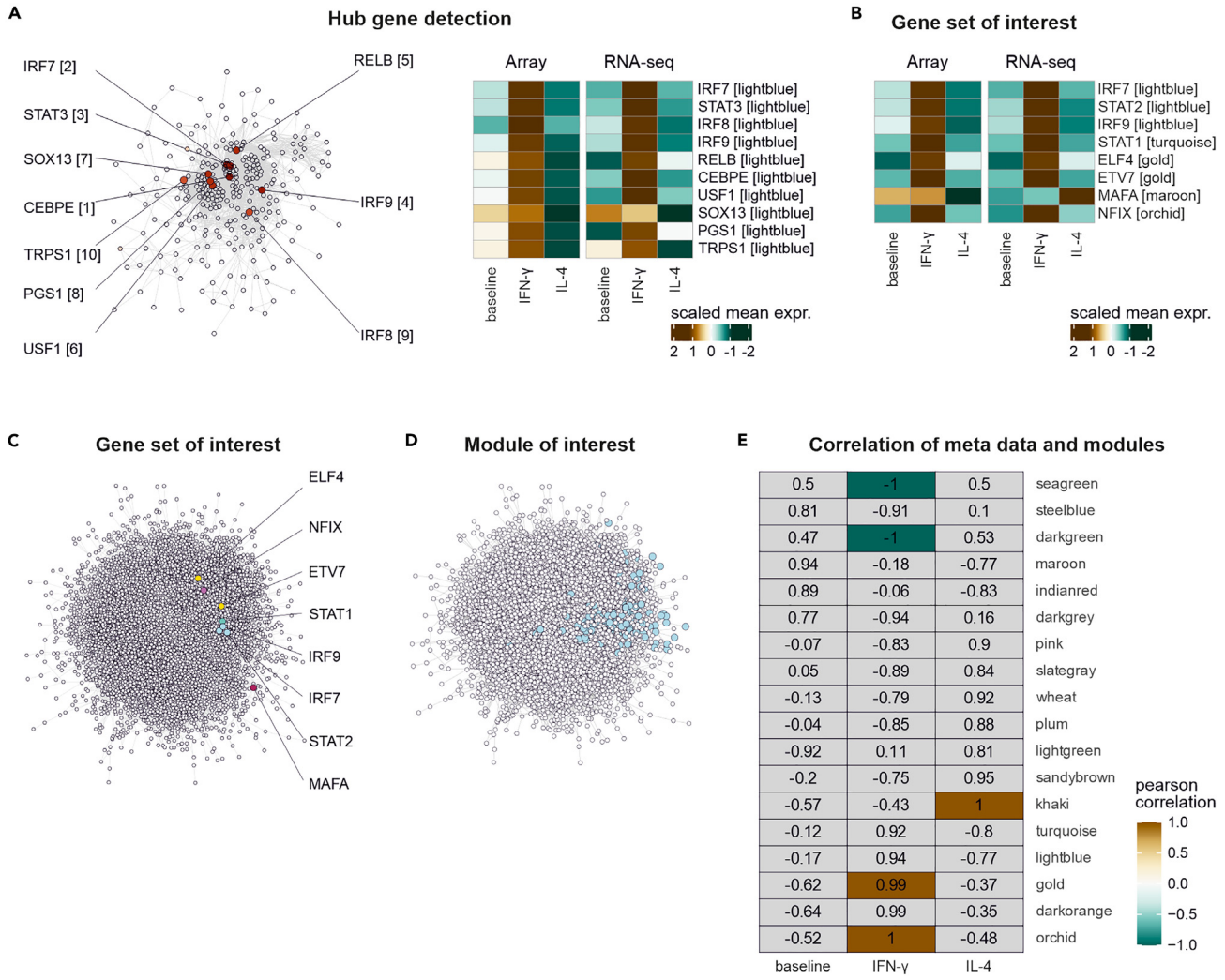


Figure 5. Post-integration phase II

(A) Module hub gene detection for the module of interest ("light blue"). Left: Network visualization of the selected module, highlighting hub genes with nodes colored by degree of centrality. Numbers represent centrality rank. Right: Heatmap illustrating the mean expression values of identified hub genes across each group within the VOI and each dataset and scaled per row.

(B) Heatmap illustrating mean expression values of a gene set of interest ("IFNγ-induced signature") across each group of the VOI ("merged") and each dataset and scaled per row.

(C) Network visualization highlighting genes of interest from a specific gene set of interest ("IFNγ-induced signature"). Genes are colored according to the module they are assigned to.

(D) Network visualization highlighting all genes of a selected cluster of interest ("lightblue").

(E) Heatmap displaying Pearson correlation values between the fraction of samples of the respective groups of the defined meta variable ("Phenotype") across the groups of the VOI ("merged") and the mean gene expression in the respective module. Correlations with a p-value > 0.1 are shown in gray.

Note: If you are working within a local R session, please refer to the "hCoCena_satellite_STAR-protocol.Rmd" file for the respective code. This file is available within the "STAR_protocol" folder in the GitHub repository (see Key resources table).

Note: If you do not want to use Cytoscape for network visualization, you can skip Steps a-d, set the network layout in Step 5 to "layout_with_fr", and run the function in e without defining the "layout" variable. This will only impact the network visualization, but not the gene clustering.

Optional: You can also visualize the network for each group with the nodes colored by the GFC within the respective group (Figure 4E):

```
>plot_GFC_network()
```

11. To connect the detected gene modules with functional terms, you can perform GSEA of gene modules using a database of your preference, e.g., "Hallmark", either for all modules or only selected modules (Figure 4F):

```
>functional_enrichment(

gene_sets = c("Hallmark"),

clusters = c ("seagreen", "steelblue", "turquoise",

"plum", "maroon", "lightblue", "gold"),

top = 5, padj = "BH", qval = 0.1)
```

Note: *hCoCena* supports the enrichment based on GO, Hallmark, KEGG, and Reactome, as well as user-defined enrichment databases. However, the respective reference files need to be specified and loaded within Step 4. If you want to add another reference file, you can adjust the code in Step 4 and re-run the respective code chunks in Step 4.

Note: You can specify the number of top enriched terms to return, the method to be used for multiple-testing correction, e.g. Benjamini-Hochberg ("BH"), as well as the q-value cutoff.

Note: If you receive an error message upon executing the 'functional_enrichment' function, refer to Troubleshooting 5.

12. Identify potential upstream regulators of the identified gene modules by performing a TFEA based on ChEA3 (Figure 4G):

```
>TF_overrep_module(clusters = "lightblue", topTF = 5, topTarget = 5)
```

Note: You can perform the TFEA for all or only selected gene modules. Moreover, you can define the number of TFs, as well as the number of target genes for each TF to return.

Optional: Instead of performing the TFEA separately for each module, you can also perform the analysis on the entire network and then query the result for any TF of interest:

```
>TF_overrep_network()

>check_tf("[TF_OF_INTEREST]")
```

Optional: You can define the hub genes for each module (Figure 5A). Hub genes are central genes of their respective modules that are computed using a combined ranking based on weighted degree centrality, weighted closeness centrality, and weighted betweenness centrality. You can filter the results for TFs based on the TF file you uploaded in Step 4.

```
>find_hubs(clusters = "lightblue", top = 10, tree_layout = F,
TF_only = "TF", save = T, plot = T)
```

Note: The calculation of hub genes is computationally extensive and might take a few minutes to run, depending on the number of nodes and edges in the respective gene module.

Optional: If you are interested in the expression of a specific gene set across groups, you can visualize the genes in a heatmap for each dataset. In our example, we visualize an IFNγ-induced gene signature described by Xue et al., 2014[3] (Figure 5B).

```
>visualize_gene_expression(genes = c("MAFA","IRF9","IRF7","STAT2", "STAT1","NFIX","ELF4",
"ETV7"),
name = "Geneset_IFNg")
```

Optional: If you are interested in where a specific gene or set of genes is located within your network, you can highlight and label them within the network (Figure 5C):

```
>highlight_geneset(genes = c("MAFA","IRF9","IRF7","STAT2",
"STAT1","NFIX","ELF4","ETV7"),
name = "Geneset_IFNg")
```

Optional: If you want to highlight a specific module within the network, you can color a single module within the network (Figure 5D):

```
>colour_single_cluster(cluster = "lightblue")
```

Optional: In addition to the described GSEA, you can also perform a module profiling. To do so you can either provide a CSV file or choose one of the databases defined in Step 4. Within the CSV file, the column names need to represent categories (e.g. cell types), and the columns need to contain genes representative of that category (e.g. cell type marker genes). For more information on the file structure, refer to the "hCoCena_satellite.Rmd" script in the GitHub repository (see Key resources table). The results of the enrichment will be visualized as a stacked barplot along the module heatmap. Refer to the function's documentation for additional information on how to set the respective variables within the function.

```
>user_specific_cluster_profiling()
>plot_cluster_heatmap()
```

Optional: Moreover, you can add categorical and numerical metadata annotations to the module heatmap. You can specify whether the data should be converted into percentages or kept as absolute values with the 'type' parameter.

```
> col_anno_categorical(variables = c("stimuli", "Phenotype"),
variable_label = "Stimulation",
type = "percent")
> plot_cluster_heatmap()
```

Optional: Besides adding the metadata information to the module heatmap, you can also calculate the Pearson correlation between numeric and the levels of categorical metadata variables with the mean gene expression of module genes (Figure 5E). You can specify the method to be used for multiple-testing correction, e.g. Benjamini-Hochberg ("BH"). For more details refer to the functions' documentations.

```
>meta_correlation_cat(set = 2, meta = "Phenotype",
p_val = 0.1, padj = "none")
```

Optional: If you notice that an alternative grouping variable might be more suitable to explain the differences in the data, you can change the grouping parameter. We recommend rerunning the entire *hCoCena* workflow, should you decide to use this alternative variable.

```
> change_grouping_parameter(group_by = "[VARIABLE_OF_CHOICE]")
```

Optional: If your current VOI does not align with the clustering patterns observed in the heatmap generated by 'run_expression_analysis_2' or within the PCA, you can assign new group labels based on the hierarchical clustering of the scaled mean expression of all genes, the network genes, or the gene modules. This approach enables a data-driven analysis, potentially revealing new subgroups within your data. For additional information, refer to the function's documentation.

```
>cut_hclust()
```

13. To ensure reproducibility, save all essential information and parameter configurations:

```
>write_session_info()
```

EXPECTED OUTCOMES

The *hCoCena* workflow offers support for the selection of variables used for the generation of weighted gene co-expression networks, enables the integration of these networks, and supports comprehensive downstream analyses. These analyses are based on gene modules identified through community detection algorithms that are not strongly dependent on the original data type. Using these gene modules, *hCoCena* facilitates GSEA, and TFEA, as well as various optional network visualizations, and metadata correlation analyses.

Results of the *hCoCena* workflow are provided in the form of visualizations that can be directly displayed in RStudio and/or saved in the format of PDF files, as well as EXCEL files, which are saved within the specified output directory. In addition, all results are stored within the 'hcobject' which is highly interactive and enables the simple integration of results into a broader analysis pipeline.

QUANTIFICATION AND STATISTICAL ANALYSIS

A detailed description of *hCoCena* functions and methods is reported in the original publication.[1] Additionally, please refer to the R markdown files and the package source code for an in-depth description of statistical methods (see Key resources table).

LIMITATIONS

While *hCoCena* can perform horizontal integration of transcriptomic datasets of varying sequencing types and dimensions through its transformation-based integration approach, the tool cannot perform data corrections for batch variables within the datasets. Therefore, any batch correction should be carried out before importing the data into *hCoCena*. Nonetheless, *hCoCena* serves as a valuable tool to quickly identify any outliers or batches in the data.

Furthermore, *hCoCena* does not currently support the analysis of single-cell RNA-seq data. However, it can be applied to single-cell RNA-seq data that has been aggregated into pseudobulks. Moreover, up-to-date *hCoCena* only supports human and mouse data.

For the GSEA, *hCoCena* is highly reliable on public databases. While these databases offer the great advantage of quickly linking genes with functional terms, they also carry the risk of oversimplification and potential misinterpretation.

TROUBLESHOOTING

Problem 1

Your system does not fulfill the required hardware prerequisites to successfully run Docker containers (related to Setting up an hCoCena workspace).

Potential solution

- Download and install R version 4.3.2 from: https://cran.r-project.org/. *hCoCena* is expected to work with more recent R versions, however, compatibility testing for these versions has not been conducted.
- Download and install RStudio version 2023.09 + 494 from: https://posit.co/products/open-source/rstudio/. *hCoCena* is expected to work with more recent RStudio versions, however, compatibility testing for these versions has not been conducted.
- Download the R script "install_hcocena.R" from the *hCoCena* GitHub repository (see Key resources table).
- Source the script to install the *hCoCena* R package and all required dependencies.
- Download the folders "reference_files" and "STAR_protocol" from the GitHub repository.

Problem 2

One or multiple paths to the directories to the count table, annotation table, reference files, and output folder you defined do not exist (related to Step 3.).

Potential solution

- Verify that you have provided the paths to the folders where your count table, annotation table, and reference files are located, respectively, without including the file names at the end of the path. If you are using a Windows machine, you can find the folder path by navigating into the specific folder and then clicking on the Address Bar at the top of the File Explorer. Make sure that in your R code, you use backslashes in the defined paths.
- Ensure that the output directory you defined exists. A new folder in which your analysis results will be stored throughout the analysis will only be created in Step 3c.

- For more detailed information, refer to the GitHub documentation (see Key resources table) or the functions documentation in R:

```
>?init_wd
```

Problem 3

Your data import was not successful (related to Step 4).

Potential solution

Errors during data import can be caused by various reasons. We try to provide solutions to the most common issues.

For the import of count and annotation tables.

- Verify that you have specified either the correct file names or data frame name within the 'define_layers' function depending on whether you aim to upload the data from files or the R environment. When uploading files, it is essential to include the file format extension at the end of the file name. For additional details, refer to the function's documentation:

```
>?define_layers
```

- If you aim to upload your data from files, ensure that they are located within the respective folders you specified in Step 3.
- If you aim to upload your data from files, further ensure that you have specified all necessary variables within the 'read_data' function, namely the separator of the count ("sep_counts") and/or annotation file ("sep_anno"), the column name of the column containing the gene symbol information in the count file ("gene_symbol_col"), as well as the column name of the column containing the sample ID information in the annotation file ("sample_col"), and whether the count or annotation file have rownames ("count_has_rn" or "anno_has_rn").
- Ensure that you have provided the data in the correct format. Refer to the section Format your count and annotation table for details.
- Verify that the sample IDs within the count file match the sample IDs in the annotation file. They need to be provided in the same order and need to contain the same IDs.

For the import of reference files.

- Ensure that you have specified the correct file names, including the file format extension within the 'set_supp_files' function.
- The TF reference files need to include two columns with gene symbols within the first column and the TF annotation in the second column. HALLMARK, GO, KEGG, and REACTOME files need to be provided in GMT format. User-defined enrichment files can either be provided in GMT or CSV format. CSV files should contain two columns, with the first column ("term") holding the pathway names and the second column ("gene") containing the gene name. Exemplary reference files can be found within the references folder on GitHub (see Key resources table).
- If you aim to upload your data from files, ensure that they are located within the respective folder you specified in Step 3.

Problem 4

You need to install Cytoscape for network visualization (related to Step 10).

Potential solution

- Download and install Cytoscape version 3.10.1: https://cytoscape.org/. *hCoCena* is expected to work with more recent R versions, however, compatibility testing for these versions has not been conducted.

Problem 5

You receive an error message upon executing the 'functional_enrichment' function (related to Step 11).

Potential solution

- Verify that you have uploaded all the required reference files for the "gene_sets" you have specified in the 'functional_enrichment' function. You specified and uploaded these reference files in Step 4 using the 'set_supp_files' and 'read_supplementary' functions. In our example, we have only specified the Hallmark file within the 'set_supp_files' function, which limits functional enrichment to the Hallmark gene set. To perform the enrichment analysis based on the GO, KEGG, Reactome, or a user-defined gene set, you need to specify the respective file names in 'set_supp_files' and then upload these files by re-executing the 'read_supplementary' function within Step 4.
- If you have not downloaded the files from the GitHub repository (see Key resources table), verify that you have uploaded the files either in GMT format or as a two-column CSV file with the column names "term" and "gene". Additionally, ensure that you have provided a reference file with gene symbols and no other gene identifier.
- For more detailed information, refer to the GitHub documentation (see Key resources table) or the functions documentation in R:

```
>?set_supp_files

>?read_supplementary
```

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Thomas Ulas (thomas.ulas@uni-bonn.de).

Technical contact

Technical questions on executing this protocol should be directed to and will be answered by the technical contact, Lisa Holsten (lisa.holsten@dzne.de).

Materials availability

This study did not generate new unique reagents.

Data and code availability

The code generated during this study is available on GitHub (see Key resources table). The *hCoCena* package source code was previously reported[1] and is available alongside extensive documentation on GitHub and Zenodo (see Key resources table). The datasets used for the exemplary analysis were previously published[3,4] and pre-processed.[1,3] The processed datasets are publicly available in the GitHub repository as "start_envo.RData" (see Key resources table).

ACKNOWLEDGMENTS

L.H. is supported by the DFG (UL 521/1-1). K.D. is supported by the DFG (HA 6409/5-1). M.O. is supported by the HGF Helmholtz AI grant Pro-Gene-Gen (ZT-I-PF5-23). M.B. is supported by the BMBF and EU grant PriSyn (16KISA030) and the HGF Helmholtz AI grant Pro-Gene-Gen (ZT-I-PF5-23). T.U. is a member of the excellence cluster ImmunoSensation (EXC 2151) and is funded by the DFG (UL 521/1-1 and 521/4-1).

AUTHOR CONTRIBUTIONS

Conceptualization, L.H. and T.U.; methodology, L.H., K.D., M.O., and T.U.; software, L.H., K.D., and M.O.; investigation, L.H.; writing – original draft, L.H.; writing – review and editing, L.H., K.D., M.O., M.B., and T.U.; visualization, L.H.; supervision, T.U.; funding acquisition, M.B. and T.U.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Oestreich, M., Holsten, L., Agrawal, S., Dahm, K., Koch, P., Jin, H., Becker, M., and Ulas, T. (2022). hCoCena: horizontal integration and analysis of transcriptomics datasets. Bioinformatics *38*, 4727–4734. https://doi.org/10.1093/bioinformatics/btac589.

2. Ulfenborg, B. (2019). Vertical and horizontal integration of multi-omics data with miodin. BMC Bioinf. *20*, 649. https://doi.org/10.1186/s12859-019-3224-4.

3. Xue, J., Schmidt, S.V., Sander, J., Draffehn, A., Krebs, W., Quester, I., De Nardo, D., Gohel, T.D., Emde, M., Schmidleithner, L., et al. (2014). Transcriptome-based network analysis reveals a spectrum model of human macrophage activation. Immunity *40*, 274–288. https://doi.org/10.1016/j.immuni.2014.01.006.

4. Beyer, M., Mallmann, M.R., Xue, J., Staratschek-Jox, A., Vorholt, D., Krebs, W., Sommer, D., Sander, J., Mertens, C., Nino-Castro, A., et al. (2012). High-resolution transcriptome of human macrophages. PLoS One *7*, e45466. https://doi.org/10.1371/journal.pone.0045466.

5. Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J.P., and Tamayo, P. (2015). The Molecular Signatures Database (MSigDB) hallmark gene set collection. Cell Syst. *1*, 417–425. https://doi.org/10.1016/j.cels.2015.12.004.

6. Gene Ontology Consortium (2021). The Gene Ontology resource: enriching a GOld mine. Nucleic Acids Res. *49*, D325–D334. https://doi.org/10.1093/nar/gkaa1113.

7. Kanehisa, M., and Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res. *28*, 27–30. https://doi.org/10.1093/nar/28.1.27.

8. Gillespie, M., Jassal, B., Stephan, R., Milacic, M., Rothfels, K., Senff-Ribeiro, A., Griss, J., Sevilla, C., Matthews, L., Gong, C., et al. (2022). The reactome pathway knowledgebase 2022. Nucleic Acids Res. *50*, D687–D692. https://doi.org/10.1093/nar/gkab1028.

9. Keenan, A.B., Torre, D., Lachmann, A., Leong, A.K., Wojciechowicz, M.L., Utti, V., Jagodnik, K.M., Kropiwnicki, E., Wang, Z., and Ma'ayan, A. (2019). ChEA3: transcription factor enrichment analysis by orthogonal omics integration. Nucleic Acids Res. *47*, W212–W224. https://doi.org/10.1093/nar/gkz446.

10. Gu, Z., Gu, L., Eils, R., Schlesner, M., and Brors, B. (2014). circlize Implements and enhances circular visualization in R. Bioinformatics *30*, 2811–2812. https://doi.org/10.1093/bioinformatics/btu393.

11. Wu, T., Hu, E., Xu, S., Chen, M., Guo, P., Dai, Z., Feng, T., Zhou, L., Tang, W., Zhan, L., et al. (2021). clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. Innovation *2*, 100141. https://doi.org/10.1016/j.xinn.2021.100141.

12. Gu, Z. (2022). Complex heatmap visualization. iMeta *1*. https://doi.org/10.1002/imt2.43.

13. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. *13*, 2498–2504. https://doi.org/10.1101/gr.1239303.

14. Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux J. *2014*, 2.

15. Wickham, H., François, R., Henry, L., Müller, K., and Davis, V. (2023). dplyr: A Grammar of Data Manipulation (CRAN).

16. Wickham, H. (2016). ggplot2 - Elegant Graphics for Data Analysis, 2nd ed. (Springer International Publishing). https://doi.org/10.1007/978-3-319-24277-4.

17. Csárdi, G., Nepusz, T., Müller, K., Horvát, S., Traag, V., Zanini, F., and Noom, D. (2023). igraph for R: R interface of the igraph library for graph theory and network analysis. Zenodo. https://doi.org/10.5281/zenodo.8240644.

18. Peter, K., Petukhov, V., Wang, Y., and Biederstedt, E. (2023). leidenAlg: Implements the Leiden Algorithm via an R Interface (CRAN).

19. R Foundation for Statistical Computing (2022). R: The R Project for Statistical Computing. https://www.r-project.org/.

20. Gustavsen, J.A., Pai, S., Isserlin, R., Demchak, B., and Pico, A.R. (2019). RCy3: Network biology using Cytoscape from within R. [version 3; peer review: 3 approved]. F1000Res *8*, 1774. https://doi.org/10.12688/f1000research.20887.3.

21. Posit team (2023). RStudio: Integrated Development Environment for R (Posit Software, PBC).

22. Barabási, A.L., and Oltvai, Z.N. (2004). Network biology: understanding the cell's functional organization. Nat. Rev. Genet. *5*, 101–113. https://doi.org/10.1038/nrg1272.