Day 2 – 02 Simple Heatmap (Exercises)

Seminar practice worksheet

This worksheet mirrors the guided heatmap build but uses the three-genome dataset (`second_day_part2/data/dataset2_subs`
Work through the prompts, typing your own code where `# TODO` markers appear. The goal is to recreate the full pipeline: load data, explore it briefly, build the `ComplexHeatmap`, and save the PDF.

> **Tip:** Ensure packages from `00_prepare.Rmd` are installed, and regenerate the dataset via `data/00_prepare_dataset.Rmd` if needed. A worked key lives in `scripts/02_simple_heatmap_exercises_solution.R` check it only after trying on your own.

1. Load packages and define paths

Before you touch the data, bring the visualization libraries into scope and set up deterministic file paths. Reusing the same directory layout as the guided walkthrough avoids surprises when you wire this script into `render_all.sh`. In this chunk you should:

- Call `library(ComplexHeatmap)` and `library(circlize)` (add `viridisLite` or `RColorBrewer` if you plan to experiment with palettes).
- Build absolute paths for the wide, long, and PDF outputs using `file.path()` so the notebook works no matter which directory you knit from.
- Store the PDF name (`dataset2_heatmap.pdf` or similar) under `pdf_path`—you will reference it again during export.

```
# TODO: library(ComplexHeatmap); library(circlize)
# TODO: subset_path <- file.path('..','data','dataset2_subset.csv')
#       long_path <- file.path('..','data','dataset2_subset_long.csv')
#       pdf_path <- file.path('..','pdf','dataset2_heatmap.pdf')
```

Question: Why do we still need both the wide and long versions?

2. Load/inspect the data

Create `wide_df`/`long_df` with `read.csv(..., stringsAsFactors = FALSE)` and immediately sanity-check them:

- Print `dim(wide_df)` and `dim(long_df)` to ensure they match expectations.
- Use `head()` (or `dplyr::glimpse()`) so you know which columns exist before writing subset logic.
- If you regenerated the datasets recently, make sure the timestamp or SNP counts align with your notes so the rest of the exercise is reproducible.

```
# TODO: read the CSVs into wide_df and long_df (stringsAsFactors = FALSE)
# TODO: print their dimensions and call head() on each
```

3. Choose a treatment group subset

Decide which `treatment_group` you want to display (e.g., Control vs Ciprofloxacin). Filter the columns of the wide matrix so only samples from that group remain. Hint: use the long table to map `mouse_id` + `day` to sample IDs like `1683-0` before subsetting the wide table.

This step mimics the decision-making you would do for a figure panel:

1. Inspect `unique(long_df$treatment_group)` to remind yourself of the valid options.
2. Build a tidy metadata table with one row per sample (`mouse_id`, `day`, `treatment_group`, and a combined `sample_id`).
3. Use that metadata to pick the relevant columns from the wide table; this ensures your heatmap only contains the cohort you plan to discuss.
4. Keep the metadata tibble handy—you will reuse the ordering columns later.

```
# TODO: target_group <- 'Control'
# TODO: build sample_meta <- unique(long_df[, c('mouse_id','day','treatment_group')])
# TODO: sample_meta$sample_id <- paste(...)
# TODO: keep_samples <- sample_meta$sample_id[sample_meta$treatment_group == target_group]
# TODO: subset the wide_df columns with keep_samples
```

4. Quick summaries

Checkpoint statistics help you spot typos early. Record at least the following numbers before moving on:

- Count SNPs per genome using `table(wide_df$Genome)` so you know which genomes dominate the matrix.
- Build `with(long_df, table(mouse_id, day))` to verify every mouse/day pair is present (missing rows usually mean a filtering bug).
- Use `tapply(long_df$value, long_df$Genome, FUN = summary)` or custom `min/median/max` calls to understand the numeric range—this will inform your color palette.
- Optional: store these summaries in variables and print short interpretations (e.g., "Turicimonas has the fewest SNPs but the widest spread of values").

```
# TODO: add commands described above
```

5. Build the heatmap matrix

Transform the wide data frame into a numeric matrix ready for ComplexHeatmap:

- Drop the genome metadata columns so you only keep sample measurements.
- Convert the remaining data to a matrix and coerce to numeric (in case CSV import left character columns).
- Create informative row names such as `Genome | snp_id` to make debugging easier if you temporarily enable `show_row_names = TRUE`.
- Reconstruct `sample_meta` (or reuse the earlier version) directly from the column names, split into `mouse_id`/`day`, and define an ordering via `order(mouse_id, day)` so columns follow the experimental timeline.
- Apply that order to both `heatmap_matrix` and the metadata so everything stays synchronized.

```
# TODO: sample_cols <- setdiff(names(wide_df), c('Genome','snp_id','Position'))
# TODO: heatmap_matrix <- as.matrix(wide_df[, sample_cols]); mode(heatmap_matrix) <- 'numeric'
# TODO: rownames <- paste(wide_df$Genome, wide_df$snp_id, sep = ' | ')
# TODO: create sample_meta with mouse_id/day parsed from column names
# TODO: order columns by mouse/day and reorder heatmap_matrix accordingly
```

Hint: reuse the ordering logic from the guided notebook (`order()` on `mouse_id`, `day`, `sample_id`).

6. Colors and annotations

With the matrix ordered, design the visual cues that explain it:

- Derive `mins`, `mids`, and `maxs` (or quantiles) using `range()` so the color scale reflects the actual data spread.

- Build a palette via `circlize::colorRamp2()`—try both a diverging scheme and a sequential scheme to see which communicates the values better.
- Craft column annotations: treatment group, day/baseline status, or mouse ID. Store them in a `HeatmapAnnotation` so you can reuse the object across plot iterations.
- If you prefer explicit daylight vs post-treatment shading, add `annotation_legend_param` entries with descriptive titles.

```
# TODO: compute min/mid/max for the matrix (na.rm = TRUE)
# TODO: color_fun <- circlize::colorRamp2(...)
# TODO: build mouse/day annotation via HeatmapAnnotation()
```

Challenge yourself to switch the palette (e.g., use `RColorBrewer::brewer.pal`).

7. Draw and export the heatmap

Bring everything together in a final plot:

1. Instantiate `Heatmap()` with your matrix, `col = color_fun`, ordering, and `top_annotation`. Consider toggling `column_split`, `column_title`, or `column_names_rot` if it helps readability.
2. Call `draw(ht)` in the notebook to confirm the appearance before exporting.
3. Save a PDF by wrapping `pdf(pdf_path, width = 10, height = 6)` / `dev.off()` around `draw(ht)` (or use `ComplexHeatmap::draw()` followed by `ComplexHeatmap::save_pdf()` if you prefer).
4. Print a message stating where the file landed so future-you knows it worked.

```
# TODO: construct Heatmap(...) object with top_annotation, column_split, etc.
# TODO: draw() it in the notebook
# TODO: save to pdf_path (dir.create + pdf + draw + dev.off())
```

8. Reflection prompts

1. Do you notice any new patterns when Turicimonas is included?
2. How would you highlight just the Turicimonas rows (hint: `row_split`)?
3. What additional annotation (e.g., day as a gradient) could help the reader?

Document your answers below—future iterations of this course often borrow the best observations or styling tweaks, so write clearly enough that a teammate could reproduce your reasoning.

```
# TODO: jot down observations or extra code experiments
```

Once you have a working script, compare with the solution notebook and proceed back to the main workflow.