Day 2 – Exercises

Seminar practice worksheets

All Day 2 exercises collected in one place, including the optional bonus customization challenge inspired by the ComplexHeatmap book.

Exercise – Explore Data

This exercise notebook follows the same steps as the guided exploration while letting you practice independently. It works on `second_day_part2/data/dataset2_subset_long.csv`, the three-genome version of the dataset (Akkermansia, Bacteroides, Turicimonas). Type your own code into the chunks below; the prompts explain what to do and the code blocks are left with `# TODO` comments so you can fill them in.

> **Tip:** Run `second_day_part2/00_prepare.Rmd` once to install packages, and `second_day_part2/data/00_prepare_da`
> only if you need to regenerate the three-genome dataset.

1. Load and preview the dataset

Goal: use `read.csv()` to load the file, then report the number of rows/columns and display the first few entries.

Think of this as your "sanity check" step. Confirm that the CSV path you expect actually exists, that the delimiter was read correctly (no shifted columns), and that the headers align with the guided walkthrough. Printing the first couple of lines also helps you spot accidental factor conversion or hidden `stringsAsFactors = TRUE` issues before they propagate into later tasks.

```
# TODO: load dataset2_subset_long.csv into an object named long_df
# TODO: print nrow(long_df) and ncol(long_df)
# TODO: call head(long_df)
```

Hint: remember to set `stringsAsFactors = FALSE` and `check.names = FALSE` so the column headers stay untouched.

2. Enumerate genomes and SNPs

Goal: list all genomes present, count how many rows each contributes, and count unique SNP identifiers per genome.

These counts will anchor your interpretation later. If one genome contributes far more rows, its patterns will dominate simple histograms. Likewise, the SNP totals tell you whether a genome appears sparse because it has fewer markers or because of experimental filtering. Capture both the raw row counts and the unique SNP counts so you can mention them when presenting results.

```
# TODO: unique() to list genomes
# TODO: table() to count rows per genome
# TODO: tapply() + length(unique()) to count SNPs per genome
```

Hint: model your solution after the formula `tapply(long_df$snp_id, long_df$Genome, ...)`.

3. Summaries by genome

Goal: compute summary statistics of `value` for each genome (min/median/mean, etc.) using `aggregate(value ~ Genome, ...)`. Also, show a quick `summary()` of the first few values to illustrate what the anonymous function does.

Interpreting a heatmap gets easier when you already know the numeric range per genome. Use this section to capture descriptive stats you can reuse in slides or discussion. Consider storing the aggregated summaries in an object (e.g., `genome_summary`) so you can sort by median or compare genomes later without rerunning the aggregation.

```
# TODO: sample_values <- long_df$value[1:10]; summary(sample_values)
# TODO: aggregate(value ~ Genome, data = long_df, function(x) summary(x))
```

Hint: read `value ~ Genome` as "value grouped by Genome". The column to the left of `~` must be numeric.

4. Focus on Turicimonas

Goal: subset the data frame to `Turicimonas_muris_YL45` only, show `head()` of the subset, and build a `table(mouse_id, day)` for that genome.

Turicimonas is the new genome compared with the two-genome walkthrough, so spend time understanding its footprint. After filtering, look at which mice and days have observations—do they align with the treatment timeline? You can also inspect a few `snp_ids` to see if they cluster by chromosome or appear randomly distributed.

```
# TODO: turicimonas_df <- long_df[long_df$Genome == 'Turicimonas_muris_YL45', ]
# TODO: head(turicimonas_df)
# TODO: table(turicimonas_df$mouse_id, turicimonas_df$day)
```

Hint: inspect `head(long_df$Genome == 'Turicimonas_muris_YL45')` to see the TRUE/FALSE mask created by the comparison.

5. Allele-frequency focus (Turicimonas vs all genomes)

Goal: plot the allele-frequency (`value`) histogram for `Turicimonas_muris_YL45`, then for the entire dataset. Report `summary()` for both vectors and note whether Turicimonas shows more high-frequency SNPs.

Go beyond the simple plot: compare medians, interquartile ranges, and extreme values. If distributions look different, jot down a short interpretation such as "Turicimonas peaks near 0.4 whereas Akkermansia spans 0–1 evenly." These observations will justify why adding a third genome matters in the downstream heatmap exercise.

```
# TODO: turicimonas_values <- long_df$value[long_df$Genome == 'Turicimonas_muris_YL45']
# TODO: hist(turicimonas_values, breaks = 20, main = 'Turicimonas AF', xlab = 'value')
# TODO: summary(turicimonas_values)
# TODO: hist(long_df$value, breaks = 30, main = 'All genomes AF', xlab = 'value')
# TODO: summary(long_df$value)
# Question: Does Turicimonas concentrate at higher or lower allele frequencies compared to the mix?
```

Hint: reuse the histogram idea from the guided notebook and jot down your interpretation.

6. Missing values

Goal: check whether the `value` column contains any `NA`s overall and per mouse/day combination.

Missing entries often indicate gaps in the sequencing schedule or quality filters that failed. Calculating `na_total` tells you whether the dataset is complete, while the `tapply()` step pinpoints exactly which mouse/day pairs are affected. If you do find NAs, note them explicitly so you can exclude or impute them consistently later.

```
# TODO: na_total <- sum(is.na(long_df$value))
# TODO: na_by_mouse_day <- with(long_df, tapply(value, list(mouse_id, day), function(x) sum(is.na(x))))
# TODO: print both objects and interpret (all zeros means no missing data).
```

Question: If you do detect NA values, which genome(s) or treatment groups do they belong to? State explicitly whether Turicimonas introduces any missing allele frequencies.

7. Treatment groups

Goal: use the `treatment_group` column to understand how samples are distributed across treatments.

Understanding group balance helps when you later draw stratified plots. Capture the total number of measurements per treatment, how many mice belong to each group, and whether every day has representation for each treatment. You can also look for edge cases (e.g., a mouse switching treatments) and document them here before they confuse you during plotting.

```
# TODO: list unique(long_df$treatment_group)
# TODO: table(long_df$treatment_group)
# TODO: table(long_df$mouse_id, long_df$treatment_group)
# TODO: subset long_df for a specific day (e.g., day 30) and inspect treatment groups
```

Hint: use `table()` for quick counts and `unique()` to see combinations of mouse/day/group.

8. Stretch idea

Push yourself to answer: "Which genome has the highest median value on day 30?" Outline your approach below before coding it up.

Treat this like a mini-analysis write-up: define the filter criteria, jot down which helper functions you will reuse (`tapply`, `aggregate`, or `dplyr`), and decide how you will report ties. If you have extra time, repeat for another day or compare medians vs means to see whether outliers influence your conclusion.

```
# Notes / pseudo-code:
# 1. Filter long_df to day == 30
# 2. Split by Genome and compute median(value)
# 3. Identify the maximum
```

These practice tasks mirror the guided walkthrough but force you to re-create the analysis yourself. Once satisfied, continue with `scripts/individual_notebooks/02_simple_heatmap.Rmd` to generate the heatmap.

Exercise – Simple Heatmap

This worksheet mirrors the guided heatmap build but uses the three-genome dataset (`second_day_part2/data/dataset2_sub`
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/individual_notebooks/02_simple_heatma`
> 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.

Exercise – Full Heatmap

This practice notebook guides you through rebuilding the "all genomes" heatmap (`dataset3_subset*.csv`). Compared with earlier exercises, you now juggle a larger matrix, multiple annotation layers, and custom ordering. Follow the prompts, fill in each `# TODO`, and keep notes on the design decisions you make.

> **Tip:** Run `scripts/individual_notebooks/00_prepare.Rmd` first so `ComplexHeatmap`, `circlize`, and helper packages are ready. If the dataset is missing, regenerate it via `data/00_prepare_dataset.Rmd`.

1. Packages, paths, helper settings

Set yourself up for success by loading packages, pointing to the dataset files, and defining constants you will re-use (e.g., the NA color and PDF output path). Add any palettes you plan to try so they are available later.

```
# TODO: library(ComplexHeatmap); library(circlize); library(viridisLite)
# TODO: subset_path <- file.path('..','data','dataset3_subset.csv')
#       long_path   <- file.path('..','data','dataset3_subset_long.csv')
#       pdf_path    <- file.path('..','pdf','04_full_heatmap_exercise.pdf')
# TODO: na_color <- '#dcdcdc'
# TODO: set.seed(...) if you plan to shuffle or sample
```

Reflection: Why do we keep both the wide and long versions even though the heatmap consumes only the wide table?

2. Load data and report missingness

Read both tables, print their dimensions, and log the total number of NA values in the `value` column. If NAs are present, produce a matrix (mouse vs day) that shows where they occur so you can make informed filtering choices later.

```
# TODO: wide_df <- read.csv(subset_path, check.names = FALSE, stringsAsFactors = FALSE)
# TODO: long_df <- read.csv(long_path, check.names = FALSE, stringsAsFactors = FALSE)
# TODO: cat('Wide rows x cols:', nrow(wide_df), ncol(wide_df), '\n')
# TODO: cat('Long rows x cols:', nrow(long_df), ncol(long_df), '\n')
# TODO: na_total <- sum(is.na(long_df$value))
# TODO: if (na_total > 0) { build tapply(...) to locate them }
```

Question: Does the larger dataset introduce NA clusters concentrated in a single mouse/treatment?

3. Build the numeric matrix

Construct the heatmap matrix using the sample columns, ensuring you keep row labels informative for debugging. Reuse or rebuild a `sample_meta` data frame that captures `mouse_id`, `day`, `treatment_group`, and `sample_id`. This metadata will power your annotations and ordering.

```
# TODO: sample_cols <- setdiff(names(wide_df), c('Genome','snp_id','Position'))
# TODO: mat <- as.matrix(wide_df[, sample_cols]); mode(mat) <- 'numeric'
# TODO: rownames(mat) <- paste(wide_df$Genome, wide_df$snp_id, sep = ' | ')
# TODO: sample_meta <- unique(long_df[, c('mouse_id','day','treatment_group')])
# TODO: sample_meta$sample_id <- paste(sample_meta$mouse_id, sample_meta$day, sep='-')
# TODO: sample_meta <- sample_meta[match(colnames(mat), sample_meta$sample_id), ]
```

Check: Use `stopifnot(identical(colnames(mat), sample_meta$sample_id))` to catch mismatches early.

4. Baseline heatmap and color experiments

Start with a minimal `Heatmap` (no clustering, no annotations) so you can verify the matrix orientation. Next, experiment with at least two color palettes by constructing separate `colorRamp2` functions. Document which palette you prefer and why (contrast, perceptual ordering, etc.).

```
# TODO: mins <- min(mat, na.rm = TRUE); maxs <- max(mat, na.rm = TRUE); mids <- (mins+maxs)/2
# TODO: palette_a <- circlize::colorRamp2(c(mins, mids, maxs), c('#0c2c84','#f7fbff','#b30000'))
# TODO: palette_b <- circlize::colorRamp2(c(mins, mids, maxs), viridisLite::viridis(3))
# TODO: Heatmap(mat, name='value', col=palette_a, na_col = na_color)
# TODO: Heatmap(mat, name='value', col=palette_b, na_col = na_color)
```

Reflection: Which palette makes low values easiest to distinguish once annotations are added?

5. Column annotations and ordering

Build annotations that explain treatment group and baseline/post-antibiotic status. Then define an ordering (e.g., order by `mouse_id`, then `day`) and apply it consistently to both the matrix and the annotation object. Draw the heatmap again to confirm columns appear in the intended sequence.

```
# TODO: sample_meta$post_ab <- ifelse(sample_meta$day == 0, 'baseline', 'post')
# TODO: col_ann <- HeatmapAnnotation(
#         treatment = sample_meta$treatment_group,
#         status = sample_meta$post_ab,
#         annotation_name_side = 'left'
#       )
# TODO: order_idx <- order(sample_meta$mouse_id, sample_meta$day)
# TODO: mat_ordered <- mat[, order_idx]
# TODO: col_ann_ordered <- col_ann[order_idx]
# TODO: Heatmap(mat_ordered, name='value', top_annotation = col_ann_ordered,
#               cluster_rows = FALSE, cluster_columns = FALSE, na_col = na_color)
```

Check: Are replicates or treatment switches now easier to spot?

6. Annotation enhancements, column splits, and readable row titles

Recreate the richer annotation stack from `03_heatmap_annotations.Rmd`, but now apply it to the full dataset. Ideas to try:

- Encode `treatment_group` with a named color vector so the legend matches your slide deck.

- Add a continuous day gradient using `anno_simple()` so viewers can track the time axis directly in the annotation bar.
- Overlay mouse IDs (rotated text or a thin color strip) to highlight replicate structure.
- Optionally, split rows by genome (`row_split`) to mimic the multi-panel layout from the guided notebook, and either shorten those labels or shrink the font so they do not overlap in the PDF.
- Add a `column_split` (e.g., by treatment group) so the plot is chunked into digestible vertical blocks in addition to the chronological ordering.

```
# TODO: treatment_cols <- c(...); status_cols <- c(...)
# TODO: day_col_fun <- circlize::colorRamp2(range(sample_meta$day), c('#f7fbff','#084594'))
# TODO: col_ann_rich <- HeatmapAnnotation(
#         treatment = anno_simple(sample_meta$treatment_group, col = treatment_cols),
#         status    = anno_simple(sample_meta$post_ab, col = status_cols),
#         day       = anno_simple(sample_meta$day, col = day_col_fun),
#         mouse     = anno_text(sample_meta$mouse_id, rot = 90, gp = grid::gpar(fontsize = 6)),
#         annotation_name_side = 'left'
#       )
# TODO: col_ann_rich <- col_ann_rich[order_idx]
# TODO: column_split <- factor(sample_meta$treatment_group[order_idx], levels = treatment_levels)
# TODO: row_split <- factor(wide_df$Genome, levels = unique(wide_df$Genome))
# TODO: row_titles <- gsub('_.*', '', levels(row_split)) # or set fontsize via row_title_gp
# TODO: Heatmap(mat_ordered, name='value', col = palette_a, top_annotation = col_ann_rich,
#               cluster_rows = TRUE, row_split = row_split, column_split = column_split,
#               na_col = na_color, show_row_names = FALSE, show_column_names = FALSE,
#               row_title = row_titles, row_title_gp = grid::gpar(fontsize = 9))
```

Reflection: Which annotation layer (treatment colors, day gradient, mouse IDs) helped the most when interpreting the plot? Keep notes for your presentation.

7. Row filtering via variance

Large matrices can hide structure. Compute per-row variance, keep the top 100 rows (or another threshold), and draw a diagnostic heatmap to see whether the higher-variance SNPs highlight patterns that were previously buried. Remember to reapply the column ordering and annotations.

```
# TODO: row_var <- apply(mat, 1, var, na.rm = TRUE)
# TODO: keep_idx <- order(row_var, decreasing = TRUE)[seq_len(min(100, nrow(mat)))]
# TODO: mat_topvar <- mat[keep_idx, order_idx]
# TODO: Heatmap(mat_topvar, name='value', top_annotation = col_ann_rich,
#               cluster_rows = TRUE, cluster_columns = FALSE, column_split = column_split,
#               na_col = na_color, row_split = row_split[keep_idx], show_row_names = FALSE,
#               row_title_gp = grid::gpar(fontsize = 9))
```

Reflection: Does variance filtering change the biological story you would tell?

8. Final polish and PDF export

Combine your favorite palette, annotations, column order, and a row clustering strategy into a final heatmap. Add a column title and tweak legend names if needed. Export the final figure to `pdf_path` so it can be shared outside the notebook.

```
# TODO: ht_final <- Heatmap(
#         mat_ordered,
#         name = 'value',
#         col = palette_a,
#         top_annotation = col_ann_rich,
#          cluster_rows = TRUE,
#          cluster_columns = FALSE,
#          column_split = column_split,
#          show_row_names = FALSE,
#          show_column_names = FALSE,
#          na_col = na_color,
#          row_split = row_split,
#          row_title = row_titles,
#          row_title_gp = grid::gpar(fontsize = 9),
#          column_title = 'All genomes (dataset3)'
#        )
# TODO: draw(ht_final)
# TODO: pdf(pdf_path, width = 11, height = 7); draw(ht_final); dev.off();
#        cat('Saved heatmap to', pdf_path, '\n')
```

Question: Which design choices would you highlight if you presented this plot in lab meeting (ordering rationale, annotation selection, etc.)?

9. Notes and extensions

Use this space to capture observations, alternative palettes, or next steps (e.g., adding row annotations, trying `row_split`, exporting PNGs).

```
# TODO: jot down thoughts, ideas, or additional code experiments
```

Once you're happy with the exercise output, compare it against the solution notebook (`scripts/individual_notebooks/04_f` to fill any gaps.

Bonus – Heatmap Customization

This optional worksheet extends the full heatmap workflow with decorations and custom legends inspired by the ComplexHeatmap book. You will work with the `dataset3_subset*.csv` files (all genomes, all treatment groups) and add three extras:

1. Highlight high-variance rows directly on the heatmap body with `decorate_heatmap_body()` (chapter 8).
2. Add a custom legend describing your highlight using `Legend()` (chapter 5).
3. Combine two heatmaps into a `HeatmapList` so you can show a derived metric alongside the main matrix (chapter 10).

> **Goal:** reinforce how decorations/legends/heatmap lists fit into your teaching notebook without making the code overwhelming. The prompts below keep the same "fill in the TODO" style as the other exercises.

1. Setup and pick two treatment groups

Load `ComplexHeatmap`, `circlize`, `viridisLite`, and `grid`. Read both `dataset3_subset.csv` (wide) and `dataset3_subset_long.csv` (long) from `second_day_part2/data`. Pick two treatment groups to compare (for example Control vs Ciprofloxacin) and build a sample metadata table with `sample_id`, `mouse_id`, `day`, and `treatment_group`.

```
# TODO: library(ComplexHeatmap); library(circlize); library(viridisLite); library(grid)
# TODO: subset_path <- file.path('..','..','data','dataset3_subset.csv')
# TODO: long_path   <- file.path('..','..','data','dataset3_subset_long.csv')
# TODO: wide_df <- read.csv(...); long_df <- read.csv(...)
# TODO: choose target_groups <- c('Control', 'Ciprofloxacin') (or any two)
# TODO: long_df$sample_id <- paste(long_df$mouse_id, long_df$day, sep='-')
# TODO: sample_meta <- unique(long_df[, c('sample_id','mouse_id','day','treatment_group')])
# TODO: sample_meta <- subset(sample_meta, treatment_group %in% target_groups)
```

Why? Decorations depend on knowing orientation and group membership. Keeping metadata tidy upfront simplifies the later steps.

Before moving on, explore the metadata you just built. Getting comfortable with the annotation inputs reduces surprises later.

```
# TODO: table(sample_meta$treatment_group)
# TODO: table(sample_meta$treatment_group, sample_meta$day)
# TODO: sample_meta <- sample_meta[order(sample_meta$treatment_group, sample_meta$mouse_id, sample_meta
# TODO: head(sample_meta)
```

Keep these summary outputs (or screenshots) handy so you can explain which mice and time points appear in the heatmap.

2. Build the base matrix and ordering

Create a numeric matrix that keeps only the selected samples. Order columns by `treatment_group`, then `mouse_id`, then `day`. Compute a sequential color scale spanning the observed range (e.g., via `colorRamp2()`), and record row-level information (genome name, row variance) that you will use later.

```
# TODO: sample_cols <- sample_meta$sample_id ordered as described above
# TODO: heatmap_matrix <- as.matrix(wide_df[, sample_cols]); mode(heatmap_matrix) <- 'numeric'
# TODO: rownames(heatmap_matrix) <- paste(wide_df$Genome, wide_df$snp_id, sep=' | ')
# TODO: row_genome <- wide_df$Genome
# TODO: row_var <- apply(heatmap_matrix, 1, var, na.rm = TRUE)
# TODO: define color_fun <- circlize::colorRamp2(...)
```

Checkpoint: print the range of the matrix and confirm you have rows > 0 before moving on.

3. Explore annotation-friendly summaries

Treat `sample_meta` as a small dataset worth exploring on its own. Summaries and quick visuals reinforce how treatments, mice, and days relate before you encode them in colors.

```
# TODO: print(table(sample_meta$treatment_group, sample_meta$mouse_id))
# TODO: print(table(sample_meta$treatment_group, sample_meta$day))
# TODO: annotation_df <- sample_meta[, c('sample_id','treatment_group','day')]
# TODO: head(annotation_df)
```

Optional: sketch a barplot of day counts per treatment (`barplot()` is plenty).

4. Column/row annotations

Reuse `sample_meta` to define column annotations for treatment group and day. Also build a row annotation that shows the genome label so you can scan which organisms contribute high-variance SNPs. Use distinct palettes for clarity.

```
# TODO: treatment_colors <- c(Control = '#1b9e77', Ciprofloxacin = '#d95f02') (adjust as needed)
# TODO: day_colors <- circlize::colorRamp2(seq(min(sample_meta$day), max(sample_meta$day), length.out =
# TODO: col_ann <- HeatmapAnnotation(...)
# TODO: row_ann <- rowAnnotation(...)
```

Preview the annotations before combining everything. A blank heatmap with only the column annotation is enough to sanity-check the palettes.

```r
# TODO: preview_mat <- matrix(0, nrow = 1, ncol = ncol(heatmap_matrix))
# TODO: preview_ht <- Heatmap(preview_mat,
#                              col = c('0' = '#ffffff'),
#                              top_annotation = col_ann,
#                              cluster_rows = FALSE,
#                              cluster_columns = FALSE,
#                              show_heatmap_legend = FALSE,
#                              show_row_names = FALSE,
#                              show_column_names = FALSE,
#                              column_title = 'Annotation preview')
# TODO: draw(preview_ht, annotation_legend_side = 'right')
```

You can create a similar preview for the row annotation by making a one-column matrix and assigning `right_annotation = row_ann`.

5. Highlight high-variance rows with decorations

Use `quantile(row_var, 0.9)` (or a threshold you prefer) to flag the top 10% most variable rows. Save the indices so you can mention them in the discussion or add legends later.

```r
# TODO: high_var_cut <- quantile(row_var, 0.9, na.rm = TRUE)
# TODO: highlight_rows <- which(row_var >= high_var_cut)
# TODO: ht_bonus <- Heatmap(..., name = 'bonus_heatmap', col = color_fun,
#                           top_annotation = col_ann, right_annotation = row_ann,
#                           column_split = factor(sample_meta$treatment_group, levels = target_groups),
#                           cluster_rows = FALSE, cluster_columns = FALSE,
#                           show_row_names = FALSE)
# TODO: draw(ht_bonus)
```

6. Add a companion heatmap and custom legends

Compute a row-wise statistic to show next to the main heatmap (e.g., difference between the two treatment means, number of times each row exceeded 0.5, etc.). Turn that vector into a single-column heatmap and combine it with the main heatmap via the `+` operator. Finally, create a `Legend()` describing the highlighted rows and pass it to `draw()` via `annotation_legend_list` or `heatmap_legend_list`.

```r
# TODO: control_mean <- rowMeans(heatmap_matrix[, sample_meta$treatment_group == target_groups[1], drop
# TODO: treated_mean <- rowMeans(heatmap_matrix[, sample_meta$treatment_group == target_groups[2], drop
# TODO: mean_delta <- treated_mean - control_mean
# TODO: delta_ht <- Heatmap(mean_delta, name = 'Δ mean', width = unit(1.2, 'cm'),
#                           col = circlize::colorRamp2(c(min(mean_delta), 0, max(mean_delta)),
#                                                      c('#4575b4', '#f7f7f7', '#d73027')),
#                           show_row_names = FALSE, cluster_rows = FALSE)
# TODO: combo <- ht_bonus + delta_ht
# TODO: highlight_legend <- Legend(title = 'Highlight', labels = 'Row variance  90th pct', legend_gp =
# TODO: draw(combo, heatmap_legend_list = list(highlight_legend), annotation_legend_side = 'right')
```

Experiment with other `Legend()` entries (e.g., for treatment colors) so the final plot clearly explains what the decoration means.

7. Reflection

Document what the decoration + companion heatmap revealed. Did the highlighted rows correspond to a particular genome? How would you explain the extra legend to students who are new to ComplexHeatmap? Jot down notes or screenshots for your Day 2 facilitation.

```
# TODO: cat('Observations: ...') or store `highlight_rows` for later use
```

You can now integrate this chunk into `day2_walkthrough.Rmd` or keep it as a bonus appendix depending on time.