Day 2 – 01 Explore Data

Seminar plotting walk-through

We start the Day 2 session with the already prepared file `second_day_part2/data/dataset1_subset_long.csv`. This document simply shows how to open that file with base R commands and how to inspect what is inside before we move on to plotting. Run `second_day_part2/data/00_prepare_dataset.Rmd` first if you need to regenerate the CSV. Knit it from the repository root with:

```
R -e "rmarkdown::render('second_day_part2/scripts/01_explore_data.Rmd')"
```

1. Load the CSV

```
input_path <- file.path('..', 'data', 'dataset1_subset_long.csv')
long_df <- read.csv(input_path, stringsAsFactors = FALSE, check.names = FALSE)

cat('Rows:', nrow(long_df), '\\nColumns:', ncol(long_df), '\\n')
```

```
## Rows: 1536 \nColumns: 7 \n
```

2. Look at column names and a small preview

`colnames()` lists the headers exactly as they appear in the CSV and `head()` prints the first few rows. Encourage learners to read these outputs aloud so they begin to map column names to real-world meaning (genome name, SNP id, mouse ID, day, value, . . .).

```
colnames(long_df)
```

```
## [1] "Genome"          "snp_id"          "Position"        "value"
## [5] "mouse_id"        "day"             "treatment_group"
```

```
head(long_df)
```

```
##                         Genome      snp_id Position    value mouse_id day
## 1 Akkermansia_muciniphila_YL44  239840-C-G   239840 0.000000     1683   0
## 2 Akkermansia_muciniphila_YL44  241793-A-G   241793 0.049587     1683   0
## 3 Akkermansia_muciniphila_YL44  355328-A-T   355328 0.138182     1683   0
## 4 Akkermansia_muciniphila_YL44  356291-C-A   356291 0.000000     1683   0
## 5 Akkermansia_muciniphila_YL44 2351445-C-T  2351445 0.000000     1683   0
## 6   Bacteroides_caecimuris_I48 1601848-T-C  1601848 0.041609     1683   0
##   treatment_group
## 1         Control
## 2         Control
## 3         Control
## 4         Control
## 5         Control
## 6         Control
```

It can also help to peek at the SNP identifiers themselves to remind everyone what a typical label looks like:

```r
head(unique(long_df$snp_id), n = 5)
```

```
## [1] "239840-C-G"  "241793-A-G"  "355328-A-T"  "356291-C-A"  "2351445-C-T"
```

3. Which genomes are present?

We often start by asking "Which genomes are represented here?". `unique()` returns all distinct values, while `table()` counts how many rows belong to each genome.

```r
unique(long_df$Genome)
```

```
## [1] "Akkermansia_muciniphila_YL44" "Bacteroides_caecimuris_I48"
```

```r
table(long_df$Genome)
```

```
##
## Akkermansia_muciniphila_YL44   Bacteroides_caecimuris_I48
##                          320                         1216
```

Explain to students that each row contains the `Genome`, the unique SNP id, the mouse ID, the day of sampling, and the measured value.

4. How many SNPs per genome?

Counting the unique SNP identifiers per genome shows how much information we keep for each organism. The helper `tapply(values, groups, FUN)` applies a function to each subset of `values` defined by `groups`. Here we give it the vector of SNP ids and ask it to count how many unique ids (`length(unique(x))`) exist inside each genome group.

```r
snp_per_genome <- tapply(long_df$snp_id, long_df$Genome, function(x) length(unique(x)))
snp_per_genome
```

```
## Akkermansia_muciniphila_YL44   Bacteroides_caecimuris_I48
##                            5                          19
```

6. How many measurements per mouse and per day?

```r
table(long_df$mouse_id)
```

```
##
## 1683 1688 1692 1699
##  384  384  384  384
```

```r
table(long_df$day)
```

```
##
##  0  4  9 14 18 23 30 37 44 49 53 58 63 67 72 79
## 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
```

For a combined view you can also build a two-way table. Again we use `table()`, this time with `mouse_id` on rows and `day` on columns. Reading across a row shows how many time points we have for a given mouse.

```r
table(long_df$mouse_id, long_df$day)
```

```
##
##         0  4  9 14 18 23 30 37 44 49 53 58 63 67 72 79
##   1683 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
##   1688 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
```

```
##   1692 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
##   1699 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
```

Pick any cell and inspect those rows directly. For example, mouse 1683 on day 0 has the following measurements (all SNPs for that mouse/day):

```
subset_example <- long_df[long_df$mouse_id == '1683' & long_df$day == 0, ]
subset_example
```

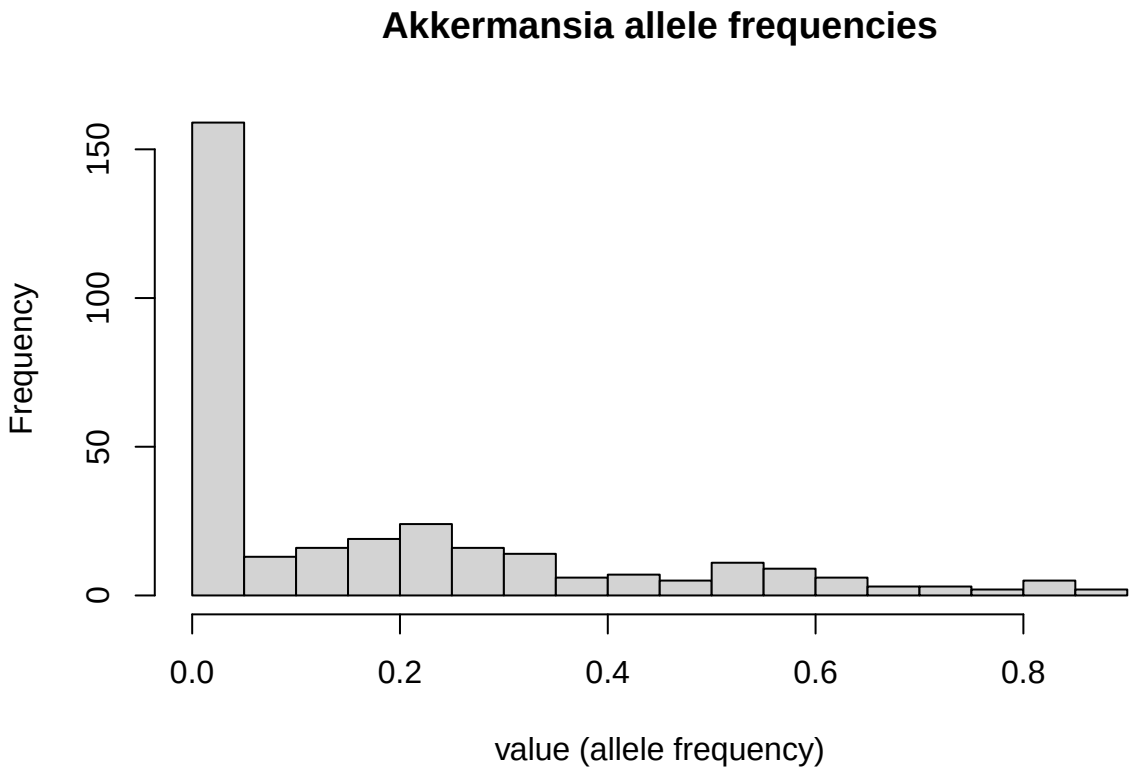
```
##                          Genome      snp_id Position    value mouse_id day
## 1  Akkermansia_muciniphila_YL44  239840-C-G   239840 0.000000     1683   0
## 2  Akkermansia_muciniphila_YL44  241793-A-G   241793 0.049587     1683   0
## 3  Akkermansia_muciniphila_YL44  355328-A-T   355328 0.138182     1683   0
## 4  Akkermansia_muciniphila_YL44  356291-C-A   356291 0.000000     1683   0
## 5  Akkermansia_muciniphila_YL44 2351445-C-T  2351445 0.000000     1683   0
## 6    Bacteroides_caecimuris_I48 1601848-T-C  1601848 0.041609     1683   0
## 7    Bacteroides_caecimuris_I48 1602949-A-G  1602949 0.000000     1683   0
## 8    Bacteroides_caecimuris_I48 2105155-C-T  2105155 0.113208     1683   0
## 9    Bacteroides_caecimuris_I48 2562529-T-C  2562529 0.000000     1683   0
## 10   Bacteroides_caecimuris_I48 2958703-C-A  2958703 0.000000     1683   0
## 11   Bacteroides_caecimuris_I48 2963503-C-A  2963503 0.000000     1683   0
## 12   Bacteroides_caecimuris_I48 2964459-G-T  2964459 0.000000     1683   0
## 13   Bacteroides_caecimuris_I48 2973657-A-G  2973657 0.000000     1683   0
## 14   Bacteroides_caecimuris_I48 2975719-A-G  2975719 0.000000     1683   0
## 15   Bacteroides_caecimuris_I48 2981479-T-G  2981479 0.000000     1683   0
## 16   Bacteroides_caecimuris_I48 2996527-C-A  2996527 0.000000     1683   0
## 17   Bacteroides_caecimuris_I48 2996770-A-G  2996770 0.000000     1683   0
## 18   Bacteroides_caecimuris_I48 3004527-A-T  3004527 0.000000     1683   0
## 19   Bacteroides_caecimuris_I48 3004659-G-A  3004659 0.000000     1683   0
## 20   Bacteroides_caecimuris_I48 3004706-A-G  3004706 0.000000     1683   0
## 21   Bacteroides_caecimuris_I48 3005011-C-A  3005011 0.000000     1683   0
## 22   Bacteroides_caecimuris_I48 3009589-C-T  3009589 0.000000     1683   0
## 23   Bacteroides_caecimuris_I48 3623124-G-A  3623124 0.173591     1683   0
## 24   Bacteroides_caecimuris_I48 4472955-A-G  4472955 0.000000     1683   0
##    treatment_group
## 1          Control
## 2          Control
## 3          Control
## 4          Control
## 5          Control
## 6          Control
## 7          Control
## 8          Control
## 9          Control
## 10         Control
## 11         Control
## 12         Control
## 13         Control
## 14         Control
## 15         Control
## 16         Control
## 17         Control
## 18         Control
## 19         Control
## 20         Control
```

```
## 21          Control
## 22          Control
## 23          Control
## 24          Control
```

6. What do the allele-frequency values look like?

Start by focusing on a single genome (Akkermansia) so the histogram is easy to interpret, then compare it against the combined dataset.

```
akk_values <- long_df$value[long_df$Genome == 'Akkermansia_muciniphila_YL44']
hist(akk_values, breaks = 20, main = 'Akkermansia allele frequencies', xlab = 'value (allele frequency)
```

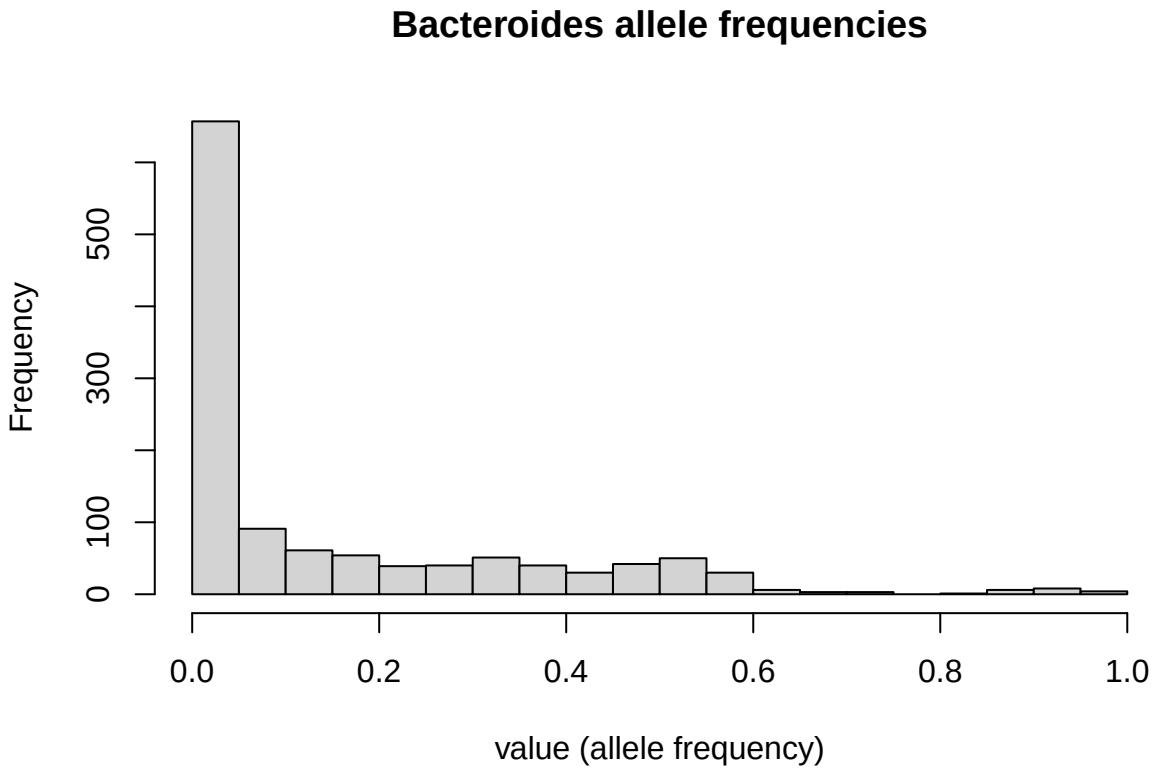



```
summary(akk_values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.05231 0.17419 0.27145 0.87125
```

Do the same for Bacteroides (the other genome in this dataset).

```
bacto_values <- long_df$value[long_df$Genome == 'Bacteroides_caecimuris_I48']
hist(bacto_values, breaks = 20, main = 'Bacteroides allele frequencies', xlab = 'value (allele frequenc
```

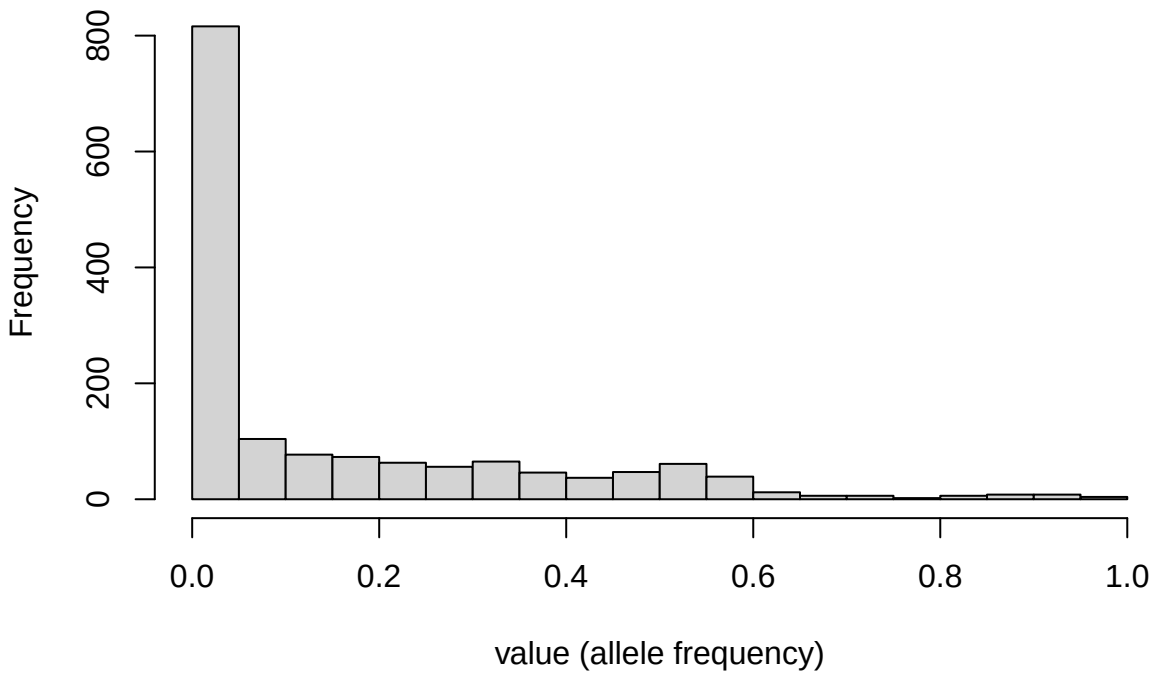



```
summary(bacto_values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.03383 0.14854 0.26792 0.99122
```

Now look at the complete long table to see how the mixture of genomes changes the distribution.

```
hist(long_df$value, breaks = 30, main = 'All genomes: allele-frequency histogram', xlab = 'value (allele
```




```r
summary(long_df$value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.03941 0.15388 0.27070 0.99122
```

To compare genomes explicitly, break the summary down by genome using `aggregate(value ~ Genome, ...)`. Read `value ~ Genome` as "value grouped by Genome". The column on the left must be numeric because we are calling `summary()` on it.

```r
value_by_genome <- aggregate(value ~ Genome, data = long_df, function(x) summary(x))
value_by_genome
```

```
##                         Genome value.Min. value.1st Qu. value.Median value.Mean
## 1 Akkermansia_muciniphila_YL44  0.0000000     0.0000000    0.0523090  0.1741893
## 2   Bacteroides_caecimuris_I48  0.0000000     0.0000000    0.0338305  0.1485411
##   value.3rd Qu. value.Max.
## 1     0.2714463  0.8712450
## 2     0.2679232  0.9912180
```

7. How many missing values?

Count how many `NA` values appear in `value` and, if any, locate them by mouse/day.

```r
na_total <- sum(is.na(long_df$value))
na_total
```

```
## [1] 0
```

```r
na_by_mouse_day <- with(long_df, tapply(value, list(mouse_id, day), function(x) sum(is.na(x))))
na_by_mouse_day
```

```
##      0 4 9 14 18 23 30 37 44 49 53 58 63 67 72 79
## 1683 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
## 1688 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 1692 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 1699 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

8. Treatment-group overview

The long table now contains a `treatment_group` column. Start by listing the distinct labels and counting how many rows belong to each group.

```
unique(long_df$treatment_group)
```

```
## [1] "Control"       "Ciprofloxacin" "Tetracyclin"   "Vancomycin"
```

```
table(long_df$treatment_group)
```

```
##
## Ciprofloxacin       Control   Tetracyclin    Vancomycin
##           384           384           384           384
```

Which treatment groups appear for each mouse? Use a cross-tabulation to reinforce how mice were assigned.

```
table(long_df$mouse_id, long_df$treatment_group)
```

```
##
##        Ciprofloxacin Control Tetracyclin Vancomycin
##   1683             0     384           0          0
##   1688           384       0           0          0
##   1692             0       0         384          0
##   1699             0       0           0        384
```

You can also inspect specific days. For example, day 30 only:

```
subset_day30 <- long_df[long_df$day == 30, c('mouse_id', 'treatment_group')]
unique(subset_day30)
```

```
##      mouse_id treatment_group
## 97       1683         Control
## 481      1688   Ciprofloxacin
## 865      1692     Tetracyclin
## 1249     1699      Vancomycin
```

9. Focus on a single genome (Akkermansia)

Subsetting is a great way to answer specific questions. Below we keep only `Akkermansia_muciniphila_YL44` (using a logical comparison inside the square brackets) and inspect the resulting table. This demonstrates how to focus on one organism without altering the original data frame.

Explain that `long_df$Genome == 'Akkermansia_muciniphila_YL44'` evaluates to a vector of TRUE/FALSE values—TRUE where the genome matches that text and FALSE everywhere else. When we place that logical vector inside `[...]`, R keeps only the TRUE rows.

```
akk_df <- long_df[long_df$Genome == 'Akkermansia_muciniphila_YL44', ]
cat('Rows for Akkermansia:', nrow(akk_df), '\n')
```

```
## Rows for Akkermansia: 320
```

```
head(akk_df)
```

```
##                          Genome      snp_id Position    value mouse_id day
## 1  Akkermansia_muciniphila_YL44  239840-C-G   239840 0.000000     1683   0
```

```
## 2  Akkermansia_muciniphila_YL44  241793-A-G   241793 0.049587     1683   0
## 3  Akkermansia_muciniphila_YL44  355328-A-T   355328 0.138182     1683   0
## 4  Akkermansia_muciniphila_YL44  356291-C-A   356291 0.000000     1683   0
## 5  Akkermansia_muciniphila_YL44 2351445-C-T  2351445 0.000000     1683   0
## 25 Akkermansia_muciniphila_YL44  239840-C-G   239840 0.000000     1683  14
##    treatment_group
## 1          Control
## 2          Control
## 3          Control
## 4          Control
## 5          Control
## 25         Control
```

```
# After subsetting we can still use `table()` to see how measurements are
# distributed for this genome alone.
table(akk_df$mouse_id, akk_df$day)
```

```
##
##        0 4 9 14 18 23 30 37 44 49 53 58 63 67 72 79
##   1683 5 5 5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   1688 5 5 5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   1692 5 5 5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   1699 5 5 5  5  5  5  5  5  5  5  5  5  5  5  5  5
```

```
head(long_df$Genome == 'Akkermansia_muciniphila_YL44')
```

```
## [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

Reading this out loud ("TRUE, TRUE, FALSE, ...") reinforces that the comparison creates a logical mask; only the TRUE rows survive when we subset with `[...]`.

Having these quick checks documented makes it easy to reassure learners that we understand the input file before moving into the plotting notebook (`second_day_part2/scripts/02_simple_heatmap.Rmd`).