Visualizing Pathways with Sunburst Plots

Gowtham A Rao

March 24, 2025

Contents

1 Introduction

After running pathway analyses with **CohortPathways**, you'll often want to visualize the resulting sequences of events in a clear, interactive format. A **sunburst plot** is a popular choice because it shows the hierarchical nature of treatment or event sequences in a circular layout. This vignette demonstrates how to:

1. Use **CohortPathways** to generate pathway results.
2. Map numeric codes to meaningful labels.
3. Convert the pathway table into a suitable hierarchical structure (JSON).
4. Create and export sunburst plots using the `sunburstR` package.

1.1 Prerequisites

1. A recent version of R (4.0+ recommended).

2. A working installation of **CohortPathways** (from GitHub, as of writing).

3. The following packages installed:

```r
install.packages(c("htmlwidgets", "d3r", "sunburstR", "here"))
```

4. A results object from a successful pathway run, e.g.:

```r
library(CohortPathways)

# Hypothetical example
cohortPathwaysResults <- executeCohortPathways(
  connectionDetails      = connectionDetails,
  cohortDatabaseSchema   = "my_results_schema",
  cohortTableName        = "my_cohort",
  targetCohortIds        = c(123, 456),
  eventCohortIds         = c(789, 1011, 1213),
  allowRepeats           = TRUE,
  maxDepth               = 3,
  collapseWindow         = 30
)
```

1.2 1. Inspecting the CohortPathways Output

The `executeCohortPathways()` function returns a **list** of data frames. Two key data frames for our purpose are:

- **`pathwaysAnalysisPathsData`**: Summaries of each unique pathway sequence.
  Typically includes columns like `pathwayAnalysisGenerationId`, `targetCohortId`, `step1`, `step2`, `step3`, . . . , and `countValue`.
- **`pathwayAnalysisCodesLong`** (or `pathwayAnalysisCodesData`): Contains the numeric "codes" for each event cohort, which let us translate internal numeric IDs into more descriptive labels.

```r
names(cohortPathwaysResults)
#> [1] "pathwayAnalysisStatsData"   "pathwaysAnalysisPathsData"
#> [3] "pathwaysAnalysisEventsData" "pathwaycomboIds"
#> [5] "pathwayAnalysisCodesLong"   "isCombo"
#> [7] "pathwayAnalysisCodesData"
```

1.3 2. Mapping Numeric Codes to Meaningful Labels

In many pathway analyses, you'll see numeric values like `2`, `4`, `6` in columns like `step1`, `step2`, etc., which can be cryptic. Let's create a small mapping table that converts each numeric code to a human-readable label.

1.3.1 Example: A Toy Scenario with Aspirin and Clopidogrel

Below is an illustrative example. Adapt this code to your own actual event cohorts and naming:

```r
library(dplyr)

# For demonstration, suppose your analysis found codes 2,4,6,8, etc.
codesData <- cohortPathwaysResults$pathwayAnalysisCodesLong %>%
  distinct(code) %>%
  mutate(codeName = dplyr::case_when(
    code == 2  ~ "Aspirin+Clopidogrel",
    code == 4  ~ "Clopidogrel (no Aspirin)",
    code == 8  ~ "Aspirin (no Clopidogrel)",
    code == 6  ~ "Clopidogrel (no Aspirin) → Aspirin+Clopidogrel",
    code == 10 ~ "Aspirin (no Clopidogrel) → Aspirin+Clopidogrel",
    code == 12 ~ "Clopidogrel (no Aspirin) → Aspirin (no Clopidogrel)",
    code == 14 ~ "Aspirin (no Clopidogrel) → Aspirin+Clopidogrel → Clopidogrel (no Aspirin)",
    TRUE ~ NA_character_
  ))
```

- Here, `codeName` strings are hypothetical. Your real scenario might map to medication names, condition categories, procedure types, etc.

1.4 3. Joining Labels with Pathway Steps

In `pathwaysAnalysisPathsData`, columns like `step1`, `step2`, `step3`, etc. contain numeric codes. We can **left join** these columns to our `codesData` mapping so the final data frame has descriptive labels.

```r
pathsData <- cohortPathwaysResults$pathwaysAnalysisPathsData %>%
  # Omit columns not needed for the sunburst
  select(-pathwayAnalysisGenerationId, -targetCohortId) %>%

  # Join and rename step1
  left_join(codesData, by = c("step1" = "code")) %>%
  rename(step1Name = codeName) %>%

  # Join and rename step2
  left_join(codesData, by = c("step2" = "code")) %>%
  rename(step2Name = codeName) %>%

  # Join and rename step3
  left_join(codesData, by = c("step3" = "code")) %>%
  rename(step3Name = codeName) %>%

  # ... similarly for step4, step5, etc. if your analysis had more steps ...

  # Keep only relevant columns for sunburst
  select(step1Name, step2Name, step3Name, countValue)
```

Here we assume `maxDepth = 3`. If you had more steps, you'd join similarly for `step4`, `step5`, and so forth.

1.5 4. Creating a Hierarchical JSON Structure

The **sunburstR** package needs data in a nested JSON format or a single-character "path string" format. The `d3r` package helps convert data frames to a nested JSON structure.

```r
library(d3r)

pathsDataJson <- d3_nest(
  pathsData,
  value_cols = "countValue"
)
```

- `value_cols` = `"countValue"` tells `d3_nest()` to treat `countValue` as the size/count field in the hierarchy.

- The resulting `pathsDataJson` is a JSON string representing the nested structure of steps 1 → steps 2 → steps 3.

1.6 5. Rendering the Sunburst Plot

With **sunburstR**, you have two main options: `sunburst()` or `sund2b()`. Both create an interactive sunburst chart in your R session or R Markdown document.

1.6.1 5.1 Using `sunburst()`

```r
library(sunburstR)

sunburstPlot <- sunburst(
  data       = pathsDataJson,
  width      = "80%",     # optional, can be numeric or string
  height     = 600,       # optional
  valueField = "countValue",
  legend     = list(w = 490, h = 50, r = 100, s = 5), # customizing legend layout
  count      = TRUE       # show the counts in the hover
)

sunburstPlot
```

- In an interactive R session, the chart pops up in your viewer pane (or browser, depending on the IDE).
- In an R Markdown document, the chart will appear embedded.

1.6.2 5.2 Using `sund2b()`

```r
sunburstPlotd2b <- sund2b(
  data       = pathsDataJson,
  width      = "80%",
  height     = 600,
  valueField = "countValue",
  rootLabel  = "Patients with at least one pathway"
)

sunburstPlotd2b
```

- `rootLabel` is the label at the center of the sunburst, describing the root node.

1.7 6. Saving the Plot as HTML

You can store the interactive widget in an HTML file using **htmlwidgets**:

```r
library(htmlwidgets)
library(here)      # convenient for file paths

saveWidget(
  widget = sunburstPlot,
  file   = here("results", "cohortPathways", "sunburstPlot.html")
)

saveWidget(
  widget = sunburstPlotd2b,
  file   = here("results", "cohortPathways", "sunburstPlotd2b.html")
)
```

These files can be viewed in any modern browser, shared internally with colleagues, or embedded in a dashboard.

1.8 Summary

- **CohortPathways** produces a sequence-oriented dataset that's ideal for hierarchical visualization.
- Using `sunburstR`:
  1. **Map** numeric codes to descriptive labels.
  2. **Flatten** or *nest* the data to reflect the transitions across `step1 → step2 → step3...`.
  3. **Convert** the table to nested JSON (via `d3r::d3_nest()`) or a path string format.
  4. **Plot** the data with `sunburst()` or `sund2b()`.
  5. **Save** the widget as HTML to share or archive.

This approach creates an interactive, drill-down visualization that helps stakeholders and researchers quickly grasp how patients move through different treatments or events over time.