Patient Timeline Visualization: Presenting Individual and Population-Level Clinical Trial Journeys with R and D3.js

Winkle Lu | Clinical Statistical Programming · Data Visualization

More details

WinSual 溫究

BACKGROUND

When reviewing clinical trial data, analysts routinely need to view the **full patient cohort**, focus on **treatment windows**, highlight **subgroups** by clinical criteria, and filter by **data class**. Switching between these views across separate static plots interrupts review flow. Combining **D3.js with R** delivers all four capabilities in one self-contained HTML file — each transition handled **client-side**, with no server required.

Overview — Interactive Patient Timeline

Five event layers on a single time axis per patient: medication exposure (EX ▬), clinically significant lab values (LB ♦), adverse events by grade (AE ▲), disease progression (PD ⊣), and overall survival (OS —). Cohort can be filtered and sorted for case review; output is a self-contained HTML file — **no server needed**.

★1 Per-patient Dual Axis

Each patient row carries its own **calendar date axis** (top), independently derived from that patient's first dose date, while all rows share a common **Study Day axis** (bottom). Same x-position; different dates per row. Impossible with ggplot2's `sec_axis()`, which applies a single linear transform across the whole plot.

★2 Semantic Zoom

Scroll-wheel zoom changes both axis tick density *and* data visibility: at overview only Grade ≥3 AEs appear; zooming in ≥3× reveals all grades plus Lab (CS) markers. `ggplotly()` zoom is geometric only — it never alters **which data are shown**.

★3 Cross-layer Patient Highlight

Hovering any event highlights **every layer** for that patient (opacity 1) while all others fade to 8%. Shown here: patients with progressive disease (PD) highlighted. Plotly cannot link hover events across multiple traces without a Shiny server.

★4 Data Class Filter

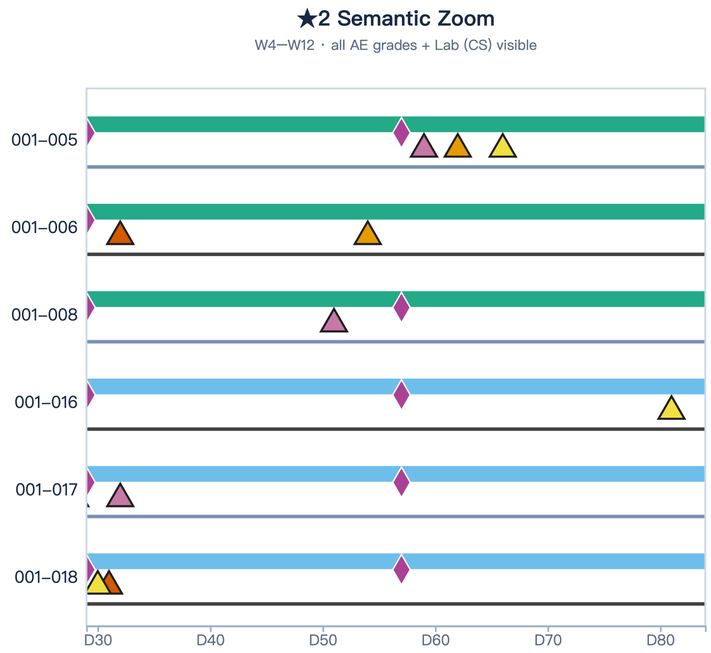
Clicking a legend pill [AE] [EX] [LB] [PD] [OS] toggles entire event classes. AE + Lab (CS) selected here: those markers stay at full opacity while medication bars and survival lines dim to 8%. All filtering runs **client-side in the browser**.

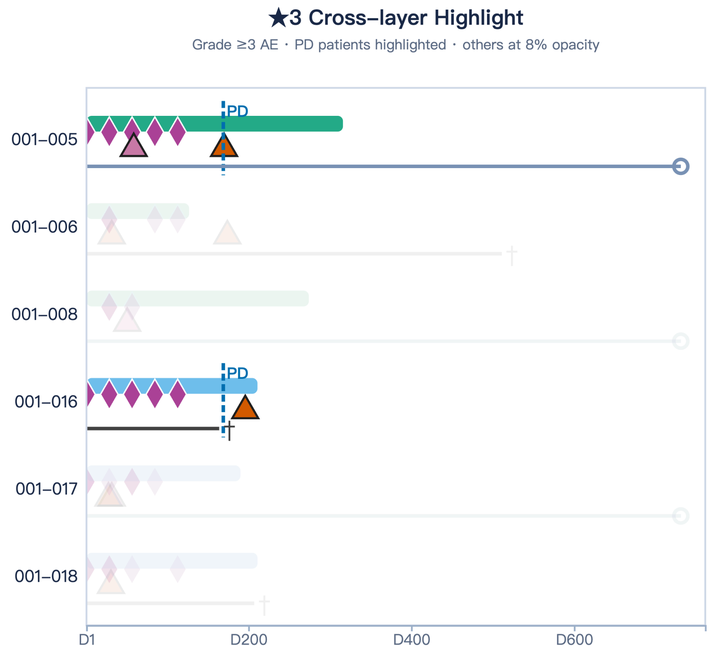
HONEST TRADE-OFFS

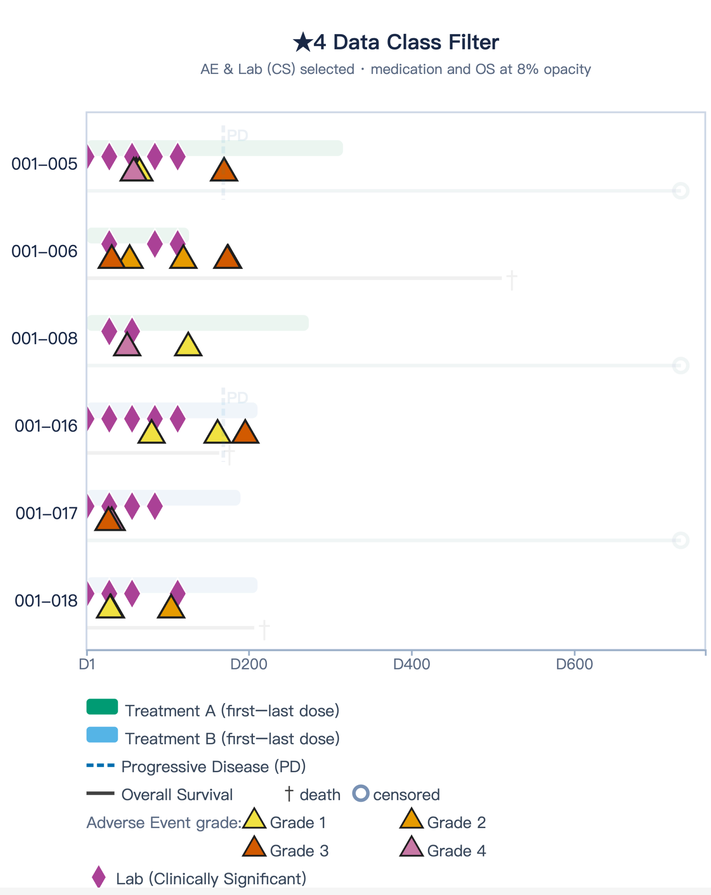
① **No regulatory validation pathway** — no published precedent for formal submissions; Best suited for exploratory review — not for submission-facing deliverables.

② **Steep learning curve** — every visual element is coded from SVG primitives; proficiency demands fluency in **both JavaScript and SVG**, uncommon in clinical programming teams.

③ **3–5× development cost** — production D3.js visuals require far more code than equivalent `ggplot2` charts; every update demands the same JavaScript expertise as the original build.







WORKFLOW

STEP 1 — SAS Clinical Data (Dummy Data)
Raw `.sas7bdat` files:
AE · DM · EX · LB · OS · Tumor

→

STEP 2 — R Data Pipeline
`haven` + `dplyr`
Select cohort · Compute study days
Bind event tables → **data.frame**

→

STEP 3 — r2d3 Bridge
`r2d3()` injects data as JSON
Passes **svg · width · height**
into the D3 script at render time

→

STEP 4 — D3.js v6 Rendering
SVG · Zoom · Tooltip · Filter
★1 Dual Axis · ★2 Semantic Zoom · ★3 Highlight · ★4 Filter

→

OUTPUT — Self-contained HTML
Single `.html` file
No server required
Poster · review · sharing ready