Paper RE08

From Static to Interactive: A Comparative Evaluation of ggplot2 and Plotly in Clinical Data Visualisation

Mugdha Joshi, Eli Lilly, Bengaluru, India

ABSTRACT

High-quality data visualization is essential in the clinical research and healthcare analytics domains, where clear communication of complex statistical findings can directly impact patient outcomes, regulatory decisions, and operational planning. Within the R ecosystem, ggplot2 and Plotly represent two dominant approaches to visualization—each with distinct advantages for clinical applications. ggplot2 offers robust, reproducible, and publication-ready graphics that align closely with statistical reporting standards common in regulatory submissions and peer-reviewed medical journals. In contrast, Plotly introduces interactive capabilities that are increasingly valuable for exploratory data analysis, stakeholder presentations, and real-time dashboarding, especially in translational research and operational contexts. Our evaluation includes case studies involving survival analysis data to highlight strengths and limitations in practical scenarios and we assess both tools based on usability, flexibility, customization, interactivity, and performance across diverse visualization tasks.

INTRODUCTION

High-quality data visualization is essential in clinical research because it turns complex, high-dimensional trial data into clear insights that can guide decisions impacting patient outcomes and regulatory strategy. In areas like oncology and cardiology, where time-to-event outcomes (e.g. survival times) are common, effective graphs help researchers and regulators quickly discern patterns—such as differences in survival between treatment groups—leading to better-informed decisions on safety and efficacy. Regulatory agencies also emphasize well-constructed figures; clear visual reports facilitate compliance audits and submissions by ensuring data is communicated transparently and correctly. In operational contexts, interactive dashboards enable real-time monitoring (for example, tracking patient enrollment or safety signals across sites), improving trial efficiency and patient safety through rapid data-driven adjustments.

R is a popular environment for clinical statisticians due to its rich ecosystem of graphics packages that support these needs. R's visualization ecosystem includes both static and interactive tools, from base R and lattice (for static trellis plots) to modern libraries like ggplot2 for static, publication-ready graphics and Plotly for interactive web graphics. Other tools like Shiny (for web dashboards), highcharter, and esquisse further expand R's capabilities. In practice, ggplot2 and Plotly have emerged as dominant choices in R for, respectively, static and interactive visualization paradigms. In the clinical domain, ggplot2 is widely used for creating the static figures found in regulatory submissions and journal articles, whereas Plotly underpins many interactive visualizations in exploratory data analysis and trial monitoring dashboards. We will discuss each tool's approach, illustrate their use on a case study (Kaplan–Meier survival curves), compare their strengths and weaknesses (including usability, flexibility, customization, interactivity, and performance), explore alternative methods (like esquisse and lattice), and conclude with recommendations tailored to clinical research applications.

WHY STATIC VS. INTERACTIVE? - USE CASES IN CLINICAL RESEARCH

Different stages and stakeholders in clinical research demand different visualization styles. In regulatory and publication settings, static visuals are the norm – for instance, a fixed Kaplan–Meier plot in a PDF report – because regulators and journal readers require self-contained, reproducible figures in standard formats. For these cases, interactivity is less important than consistency and adherence to reporting standards. ggplot2 excels here by producing high-resolution static graphics suitable for printed reports and articles, and its plots can be exactly reproduced as part of an R analysis pipeline. In fact, static tables and figures (so-called TLGs: Tables, Listings, and Graphs) are currently the only graphics accepted in regulatory submissions.

On the other hand, interactive visualization is invaluable in exploratory analysis and operational dashboards. During data exploration or while communicating with clinicians and trial managers, the ability to zoom into details, hover for exact values, and dynamically filter data can uncover insights that static plots might obscure. Plotly addresses this need by adding rich interactivity—hover tooltips, panning/zooming, clickable legend entries to toggle data series, etc. For example, a safety monitoring dashboard built with R Shiny might use Plotly graphs to allow medical monitors to examine adverse event rates by selecting subsets of patients dynamically. Such interactive tools engage end-users (who may not be R coders) and allow real-time interrogation of the data, which is increasingly valuable in translational research and pharma operations. Thus, ggplot2 and Plotly serve complementary roles in clinical data visualization: ggplot2 for static, rigorous reporting and Plotly for interactive exploration and communication.

ggplot2 OVERVIEW AND STRENGTHS

ggplot2 is an R package based on The Grammar of Graphics (Wilkinson, 2005), which provides a declarative framework to build plots layer by layer. In ggplot2, the user maps data variables to visual aesthetics (axes, color, size, etc.) and adds geometric layers (points, lines, bars, etc.) plus scale definitions, faceting, and theme elements. This layered approach yields enormous flexibility: for instance, one can overlay a non-parametric smoother on a scatterplot or facet a histogram by treatment group with just a few lines of code. Because of its rich layering and theming system, ggplot2 is well suited for complex, multi-variable graphics and meticulous customization. It integrates tightly with the tidyverse (dplyr, tidyr, etc.), allowing data transformations to be piped directly into plotting.

In clinical research, ggplot2's strengths align with requirements for reproducibility and high-quality output. Plots are generated via code (important for audit trails and methods sections), and the exact appearance can be controlled or theme-ed to meet publication guidelines (e.g., journal-specific styles). The package is widely adopted in the R community and by pharmaceutical statisticians – it's considered a de facto standard for static graphics, with "thousands of downloads a day" reflecting its popularity. Notably, regulatory agencies often expect analyses to be accompanied by static figures; with ggplot2, a statistician can produce these figures programmatically, ensuring consistency across outputs. Indeed, ggplot2 is "universally accepted as the package for graphics" (static) in many pharma organizations. Examples of ggplot2's use include creating Kaplan–Meier curves, forest plots for subgroup analyses, and dose–response curves for clinical pharmacology, all of which can be precisely tailored to highlight the scientific messages. The focus on non-interactive output means ggplot2 lacks built-in interactivity – plots are static images (e.g., PNG, PDF) – but this is by design for reliability and alignment with reporting standards.

Example: Using ggplot2 to plot a survival curve (Kaplan–Meier estimate) by group might involve computing a survival object with the survival package, then using ggplot() + geom_step() to draw the survival steps and geom_ribbon() for confidence intervals. The result is a polished static graph. Specialized extensions like survminer::ggsurvplot() even add clinical annotations (median survival, risk tables) automatically, leveraging ggplot2 under the hood – we will see this in the case study. Overall, ggplot2's key advantages for clinical research are its robustness, flexibility, and reproducibility, which make it ideal for peer-reviewed contexts and regulatory compliance.

PLOTLY OVERVIEW AND STRENGTHS

Plotly is an R package (built on the plotly.js JavaScript library) that enables interactive data visualization in R. Unlike static plotting libraries, Plotly creates graphics that users can directly interact with in a web browser or RStudio viewer. Every Plotly graph is essentially an HTML/JavaScript widget. In R, one can either use Plotly's R API (plot_ly() or higher-level functions) to create charts or convert an existing ggplot2 object to interactive form with ggplotly(). The interactive features include: hover tooltips (displaying data values when the cursor hovers over a point), zoom and pan controls on the axes, clickable legend entries to show/hide groups, and even custom JavaScript callbacks for more advanced interactivity. These capabilities are extremely useful for exploring data. For example, in an interactive safety scatterplot, one can hover to see each outlier's subject ID and relevant values – something impossible in a static image without overplotting text.

From a technical standpoint, Plotly graphs are rendered in modern web browsers, which means they can be integrated into R Shiny apps, markdown reports, or standalone HTML files. This makes Plotly a go-to choice for R Shiny dashboards used in clinical trial monitoring and medical data exploration. Because Plotly is built on web technology, it is not limited to R: Plotly's approach is language-agnostic (with APIs in Python, JS, etc.), which facilitates sharing interactive visuals beyond R (for instance, embedding a Plotly graph in a lab's webpage or internal portal). In translational research settings, Plotly can be used to build interactive genomic data explorers or patient profile review tools, where users need to dynamically query and visualize subsets of the data.

Plotly's strengths lie in its interactivity and immediate engagement. Especially for stakeholders like clinicians or project managers who may not be familiar with R coding, an interactive Plotly graphic allows them to investigate the data on their own: e.g., zoom into a specific time range of a survival curve to see early separations between treatment arms, or toggle off a comparator arm to focus on a particular group's trajectory. Plotly still produces attractive visuals, but compared to ggplot2, there may be slightly less fine control on aesthetics (though one can certainly style Plotly graphs extensively). Plotly graphs are excellent for presentations and dashboards, where the goal is to encourage audience interaction and exploration. In an exploratory analysis context—say, investigating a high-dimensional biomarker dataset—Plotly can be invaluable for quickly identifying patterns by interacting with the plot (brushing, zooming, etc.), whereas static plots would require generating many separate figures.

One trade-off to note is that because Plotly's interactivity comes via JavaScript in the browser, there is some performance overhead for very large datasets or very complex multi-layer plots. Plotly transmits data to the browser and draws using SVG or WebGL; thus, tens of thousands of points can strain the browser. (We will discuss performance more later.) In general, however, for moderate data sizes typical in clinical trials (hundreds or thousands of subjects), Plotly's performance is quite acceptable, and the "rich interactive options" it provides are considered worth the slight extra load.

In summary, Plotly is best for adding interactivity to R visuals – making data storytelling more dynamic and engaging, especially in contexts like internal decision meetings, live demos, and interactive reports. It complements ggplot2: one can develop a static

ggplot2 figure and then convert it to Plotly to allow interactive probing, combining the strengths of both, a strategy we explore below.

CASE STUDY: KAPLAN-MEIER SURVIVAL CURVES

To concretely compare ggplot2 and Plotly, we consider a case study of survival analysis, a common scenario in clinical research (particularly in oncology). Survival analysis deals with time-to-event data (e.g., time until death, relapse, hospitalization) and often uses Kaplan–Meier (KM) curves to estimate the survival function over time. These curves are ubiquitous in clinical trial reports because they clearly communicate not only if a treatment improves survival but when differences between groups emerge. Here, we use the NCCTG Lung Cancer Dataset (available in R's survival package) as an example, which contains survival times for patients with advanced lung cancer along with covariates like sex, age, and ECOG performance score. In our example, we will plot Kaplan–Meier curves stratified by patient gender (male vs. female) to illustrate how ggplot2 and Plotly handle the task. This mirrors a typical analysis where one might want to see if there are sex-based differences in survival.

Data and setup: The lung dataset comprises each patient's survival time in days and an indicator of status (censored or event occurred). For simplicity, we'll recreate the essential scenario: two groups (Male and Female), each with a set of survival times with censoring. The true data shows, for example, median survival for certain subgroups, but here we focus on the visualization aspect rather than the specific numeric outcomes.

Kaplan–Meier Plot using ggplot2 (Static)

In base R, one might use the survfit() function to compute the survival curve, and plot() it; ggplot2 allows a more flexible and styled approach. The survminer package offers ggsurvplot(), a convenient wrapper that takes a survfit object and produces a ggplot2-based KM plot with additional annotations. Using ggplot2 (via survminer), we can create a static KM plot by gender that includes confidence intervals and even a risk table below the plot showing the number of patients at risk over time.

This yields a publication-quality figure with minimal effort. Figure 1 below illustrates a Kaplan–Meier curve by gender generated with ggplot2. Each step down represents an event (death) and tick marks (if shown) indicate censored observations. The risk table (often included in static KM plots) would list how many subjects remain at risk at various time points for each group; ggplot2 (via survminer) can add this automatically in the static output. The static plot is styled according to the chosen theme (here a minimal theme with clean gridlines). Crucially, this static ggplot2 version is well-suited for inclusion in a clinical study report or journal article – it's clear, labeled, and non-interactive (ensuring that what the viewer sees is exactly what was intended, with no dependence on software).

ggsurvplot() (in the survminer package) is a high-level function built on ggplot2 that automates many survival plot details. It adds medians, confidence bands, risk tables, and even p-values to KM curves with one call. Under the hood it produces a list of ggplot objects (the main plot and the table) that can be further customized. This makes it ideal for quick clinical reports. However, because it builds complex layered plots, converting the result to Plotly can sometimes be tricky (e.g., tooltips for the risk table may not behave as expected).

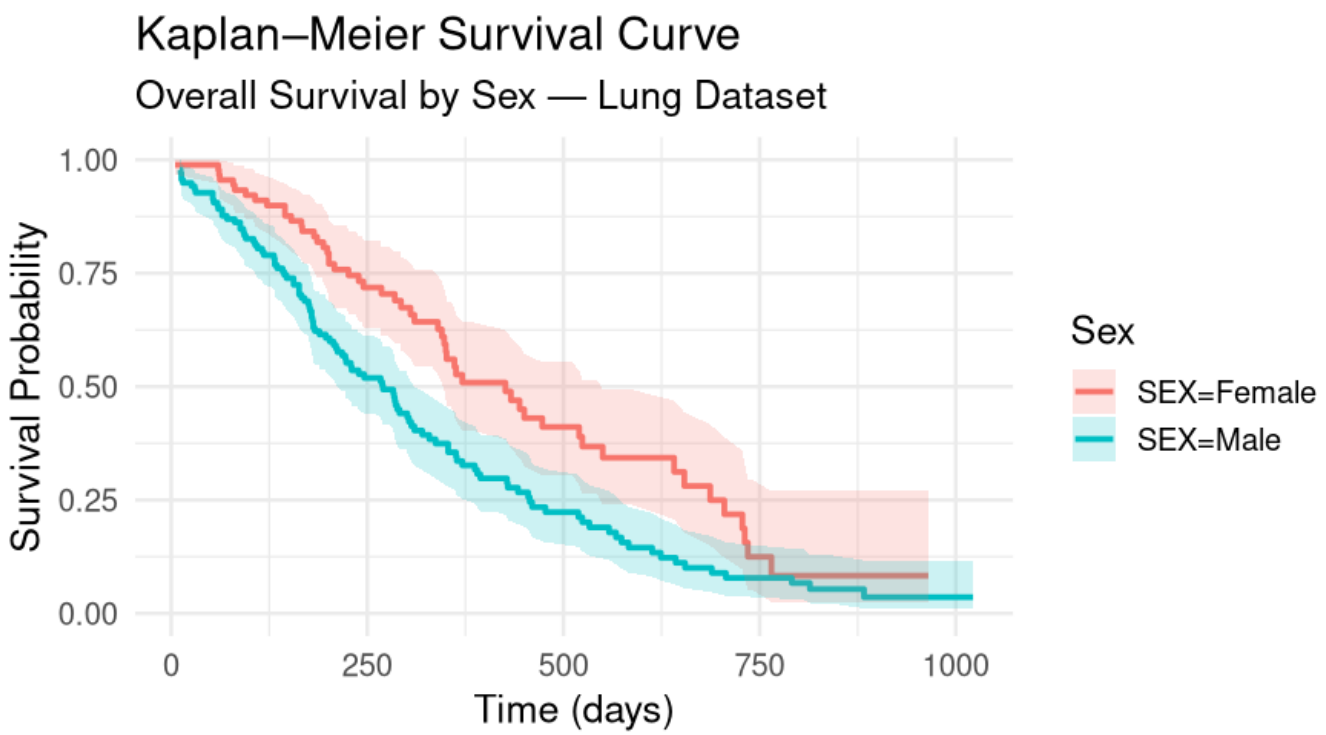

Figure 1: Kaplan–Meier survival curves by patient gender, generated with ggplot2 (static). Each step indicates a drop in survival probability when an event occurs. Such static plots are publication-ready, and survminer's ggsurvplot function can add annotations like risk tables and p-values automatically.

In Figure 1, we see the static KM curves for males (blue) and females (orange). With ggplot2, we had full control over the appearance: for example, we could apply a clean theme, set line types or colors to be colorblind-friendly, and label the axes as "Time (days)" and "Survival Probability". We could also add a shaded band for the 95% confidence interval around each step curve if desired. The static nature means that if one wants to know the exact survival percentage at a given time, they might need to read it off the y-axis gridlines or refer to a table. For instance, one might report "The 6-month survival was 55% in males vs 60% in females, p=0.40" (assuming the plot had a p-value computed). All that information can be embedded as text on the static figure or its caption. Notably, static KM plots like this are the standard in trial results – every detail is fixed and can be verified in print.

From an analytical perspective, this ggplot2 approach ensures reproducibility: anyone with the data and code can regenerate the exact same figure. The code can be part of an R Markdown or a script that also computes the survival statistics, meaning the figure is always consistent with the numeric results (no manual editing). This is highly valued in regulated environments.

Kaplan–Meier Plot using Plotly (Interactive)

Now, consider making the KM plot interactive using Plotly. There are two main ways:

Directly using Plotly's API (plot_ly or add_lines) on the survival data, or

Converting the ggplot2 object to Plotly with ggplotly().

In this case, converting the ggplot (from survminer) is tempting; however, as noted, the complex ggplot object with risk table might not convert perfectly to Plotly. An alternative is to use a package like GGally::ggsurv(), which generates a simpler ggplot2 survival plot (without extra tables) that is easier to convert. The ggsurv() function essentially creates a clean ggplot of the KM curves that one can then pipe into plotly::ggplotly() for interactivity. This approach yields an interactive plot that looks like the static one but has tooltips and other interactive features. A practical tip used by some analysts is: for interactive, conference-ready KM curves, prefer building a simple ggplot (via ggsurv or manual) and then using ggplotly; for quick internal reports where static is fine, ggsurvplot is great.

In our example, we constructed an interactive Plotly version of the KM plot by directly plotting the survival step functions for each gender. Figure 2 shows a screenshot of this interactive KM curve. In the live interactive version, a user could hover their mouse over any point on the male or female survival curves to see a tooltip with the exact survival probability and time at that point. They could also click the legend (e.g., click "Female" to toggle that curve off) to focus on one group, or use the zoom box tool to examine the first few months of the curves in detail. The interactive plot thus conveys the same overall information as the static one, but with additional exploratory power – for instance, one can inspect that around 200 days, the female survival probability is about 30% (from the tooltip) without needing to interpolate from the static graph.

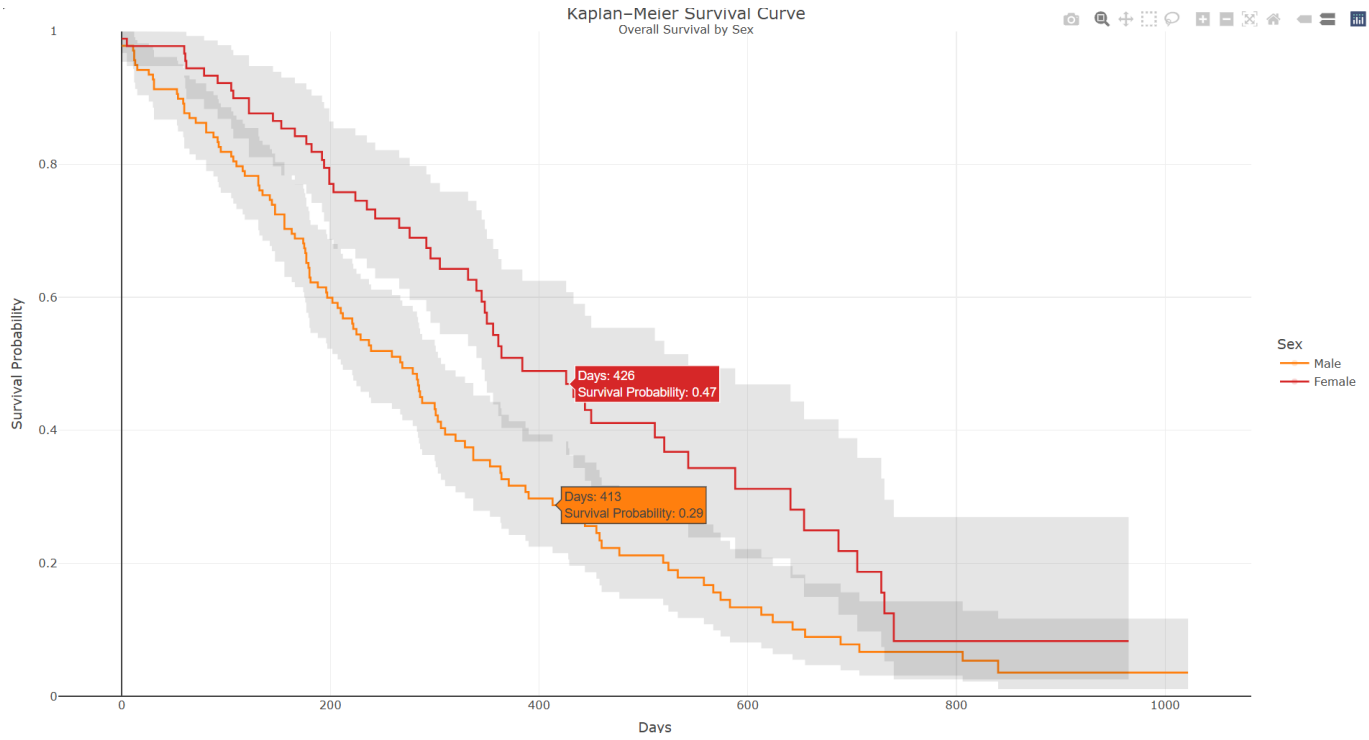

Figure 2: Interactive Kaplan–Meier plot by gender (Plotly). In the live interactive chart, users can hover to see exact values (e.g., "Female: 30% at 200 days"), zoom in on regions of interest, and toggle series via the legend. This interactivity aids exploration, especially in understanding when differences between groups occur (which one can see by zooming into the early part of the curve).

In the context of clinical data, what does this interactivity give us? It provides insight into details that a static plot might leave to the imagination. For example, if the curves cross at some point, an interactive plot allows the user to pinpoint the crossing time and survival percentages, whereas a static plot would require visual estimation or separate analysis. Interactive KM plots are also useful in meetings or live presentations: a presenter can demonstrate, say, how restricting the time axis to the first year reveals a divergence in survival that isn't as apparent when looking at the entire 5-year span. Moreover, Plotly allows integrating multiple plots with linked brushing (though not shown here) – for instance, one could have a second plot of, say, cumulative hazard or number at risk that updates as one zooms on the KM plot.

Challenges/Cons: While Plotly is powerful, there are a few considerations: (a) The interactive plot needs a medium (HTML, Shiny app, etc.) to be viewed – a PDF printout of Figure 2 would lose the interactivity. So these plots are mainly for digital consumption. (b) Converting very complex ggplot2 objects can sometimes lead to quirks. In our case, if we had included the risk table in ggplot2, ggplotly() might render the table in an odd way or produce an overly busy tooltip. A known issue is that tooltip text for confidence bands or censoring ticks may not translate cleanly when using ggplotly() on a ggsurvplot object. Therefore, a bit of extra work (like manually preparing a tidy data frame of survival estimates and using plot_ly) can give a cleaner result. In our example, we prepared the data (time and survival probability for each gender) and used Plotly's Scatter trace with "step" mode to draw the KM steps. This gave us full control to customize the hover text and omit any extraneous info.

In summary, the ggplot2 KM plot (static) is ideal for reporting definitive results, whereas the Plotly KM plot (interactive) is excellent for exploring the data and communicating with an engaged audience (e.g., in an interactive report for investigators). Both plots tell the same core story here (perhaps that there is no large difference between male and female survival in this dataset), but the interactive one allows confirming and investigating that story more dynamically.

COMPARISON OF ggplot2 AND PLOTLY

Having walked through the case study, we can generalize the key differences between ggplot2 and Plotly for clinical data visualization tasks. The table below summarizes these tools across important criteria:

Criteria	ggplot2	Plotly
Usability	Widely known in R community	Intuitive, but JS-oriented
Flexibility	Very flexible via extensions	Strong for dashboards
Customization	Themes, layers, aesthetics	Layout, hover, and callbacks
Interactivity	None (static)	Rich (zoom, hover, filters)
Performance	Lightweight	Slightly heavier on large data

Table 1: Comparison of ggplot2 and Plotly for clinical data visualization. (Note: Both tools are under active development, so gaps are continually closing – e.g., new R packages can bridge some functionality between static and interactive modes.)

In essence, ggplot2 offers greater fine-grained control and is the go-to for static, precise outputs, whereas Plotly offers interactivity and user engagement at the cost of some complexity in customization and potential performance considerations on very large datasets. It's worth noting that the two can be combined: as seen, one strategy is to design with ggplot2 and then use Plotly to add interactivity (ggplotly). This hybrid approach can yield the best of both worlds, though it requires testing to ensure the conversion works well for the given plot.

PERFORMANCE CONSIDERATIONS IN DETAIL

One concern often raised is how each tool scales with data complexity. In a simple runtime test using the microbenchmark package, converting a ggplot to Plotly (ggplotly) was found to be about 23 times slower than generating an equivalent plot directly with Plotly's API for a simple scatterplot. With a more complex plot (20k points), ggplotly() was ~143× slower than plot_ly() (mean ~0.39 seconds vs 0.0027 seconds per plot in that test). This underscores that if performance is critical (e.g., in a dashboard updating multiple plots simultaneously), using Plotly's native interface or simplifying the ggplot before conversion is beneficial. For most clinical trial plots, which rarely involve tens of thousands of points, performance is usually acceptable in either framework, but interactive plots might need extra care if, for example, showing very large datasets such as high-frequency IoT patient monitoring data.

Plotly can leverage WebGL for scatterplots (toWebGL() in R) to handle larger point clouds, and in newer Plotly versions, performance has improved. Still, the take-home message is that static plots are inherently lightweight, while interactive plots carry a performance penalty proportional to the complexity of interactivity. In practical terms: a static adverse event histogram with 100 categories poses no issue in ggplot2; an interactive version is also fine, but if you had an interactive Manhattan plot with 1e5 SNP p-values, you would need to be mindful of Plotly's limits or consider sampling/aggregating data for smooth interaction.

BRIDGING STATIC AND INTERACTIVE: HYBRID APPROACHES

Rather than viewing ggplot2 and Plotly as mutually exclusive, many workflows in clinical data science use them in combination. A common pattern is: develop with ggplot2, then enhance with Plotly. For example, one might create a complex faceted ggplot of laboratory values over time, ensure it's publication-ready, and then use ggplotly() to allow colleagues to mouse-over points in an interactive report. This approach can work well, but as noted, it has pitfalls if the ggplot object includes elements that don't translate well (e.g., annotations). Another approach is to use Plotly only for specific interactive needs – for instance, a ggplot2 static figure in the protocol and a Plotly version in an internal exploratory analysis document.

The case study highlighted a specific hybrid strategy for survival curves: using GGally::ggsurv() to get a ggplot object for the KM curve and then ggplotly() to convert it. The reason this is recommended is that ggsurv produces a relatively simple ggplot (just the steps), whereas ggsurvplot produces a composite of multiple plots (curve + table) which ggplotly may not handle cleanly. In general, when converting ggplot2 to Plotly:

Keep the ggplot as simple as necessary (add fancy annotations after conversion using Plotly's API if needed).

Use the tooltip argument in ggplotly() to control what appears on hover (so you can remove unneeded info).

Be cautious with unit transformations (log scales, date axis) as sometimes the numeric values appear in tooltips unless formatted.

Plotly's strength in making a static plot interactive with minimal code is evident, and tools like plotly::ggplotly have opened the door for many R users to embrace interactivity without rewriting all their plotting code. Just one line – ggplotly(p) – can turn a static ggplot p into an interactive chart. This lowers the barrier significantly, and is likely why Plotly has gained a strong following in the R community for interactive charts. The downside, as we saw, is potential performance impact and occasional conversion

glitches. Therefore, it's a balancing act: if a plot is simple and time is short, ggplotly is fantastic; if a plot is complex or must be highly performant in a dashboard, investing time to use plot_ly directly or simplifying the ggplot object is worthwhile.

PRACTICAL RECOMMENDATIONS FOR CLINICAL RESEARCHERS

Different visualization tasks call for different tools. Based on our evaluation, here are scenario-based recommendations for clinical research contexts:

Regulatory reports & submission figures: Use ggplot2. Static ggplot2 plots are ideal for inclusion in clinical study reports, electronic submissions, and regulatory briefing documents. They ensure reproducibility and are easily stored as image files. For example, a static forest plot of subgroup hazard ratios in an FDA briefing book should be made with ggplot2 (possibly with packages like forestplot or custom ggplot code), not an interactive widget. Static plots also satisfy archiving requirements.

Manuscripts for Peer-Reviewed Journals: Use ggplot2. Journal guidelines and the need for consistent styling make ggplot2 the safer choice. You can exactly match the journal's color palette or font requirements. Interactive content is rarely accepted in medical journals (some offer it as supplementary material, but static is primary). So, for Kaplan–Meier curves, adverse event bar charts, etc., ggplot2 is recommended when preparing publication figures.

Exploratory data analysis and internal reviews: Use Plotly (and Shiny if needed). When the goal is to understand the data rather than present final results, interactivity can accelerate insights. For instance, in a team meeting to sift through genomic correlation matrices or safety signal detection, Plotly plots allow drilling into details. Combining Plotly with Shiny can create powerful interactive dashboards for clinical data monitoring (e.g., a real-time dashboard of patient enrollment by site). The development time is often offset by the rich feedback you get from the data.

Interactive dashboards (e.g., pharmacovigilance, operations): Use Plotly. Dashboards that will be used by clinicians or project managers benefit enormously from interactive plots. Plotly integrates well with Shiny and other frameworks to build such dashboards. For example, a safety dashboard might have interactive Kaplan–Meier plots for different adverse events, where the user can select an event from a dropdown and see the KM plot update. This dynamic exploration is only feasible with interactive graphics.

Mixed static/interactive workflows (e.g., R Markdown reports for sponsors): Consider a hybrid approach. You can include ggplot2 images for static viewing and add a link to an interactive version or embed a Plotly widget in the HTML version of the report. The presentation recommended a "ggplotly hybrid" – you might generate a static ggplot2 figure and also provide an interactive version using Plotly for those who want to explore. For instance, an HTML report could show a static figure followed by a sentence: "Interactive version of Figure available here," which opens a Plotly chart. This way, both needs are met.

Ultimately, ggplot2 and Plotly are complementary. Many clinical trial analyses might begin with exploratory Plotly visuals to find the story in the data, and end with polished ggplot2 figures to communicate that story in a formal report. An analyst might use Plotly to interactively identify an outlier patient, then label that patient in a final static ggplot2 figure for the report. [From Stati...nteractive | PowerPoint]

OTHER VISUALISATION METHODS: ESQUISSE AND LATTICE

While ggplot2 and Plotly are our focus, it's worth noting a couple of alternative R visualization tools and their niches:

Esquisse: This is an R package that provides a GUI drag-and-drop interface for building ggplot2 plots. It's essentially an add-in for RStudio that lets you select a dataset, then interactively choose x/y variables, colors, geoms, etc., and it will generate the ggplot2 code for you. Use case: Esquisse is fantastic for beginners learning ggplot2 or for analysts who want to prototype a plot quickly without writing code. In a clinical company, a medical reviewer with minimal R experience could use Esquisse to quickly visualize data (say, drag "dose" to x-axis, "response" to y-axis, choose a boxplot) and then hand off the resulting code to a stats programmer for refinement. It promotes exploratory analysis in a user-friendly way. However, Esquisse is not typically used for creating final plots in automated workflows – rather, it's a helper tool. Once the ggplot2 code is exported, that code can be integrated into reports. In summary, Esquisse lowers the barrier to creating ggplot2 visuals by generating reproducible code through a GUI (which is arguably better than point-and-click in Excel, since you can reuse the code). This tool is more of interest internally (for interactive data exploration and training) and less so in formal deliverables.

Lattice: Lattice is an older graphics system in R (inspired by Trellis graphics from S-PLUS) for creating multi-panel plots based on formulas. It was the go-to for advanced R graphics before ggplot2. Lattice excels at displaying relationships across conditioning variables with a single function call – for example, plotting histograms of a measurement for every study site, arranged in a grid. It automatically handles spacing and common scales across panels. In clinical research, one might use lattice for things like subject-level spaghetti plots faceted by treatment or adverse event rate plots by subgroup, especially if quick visualization is needed without building a complex ggplot2 object. However, lattice is less flexible and less popular today. Customizing lattice plots (e.g., to add annotations or tweak a single panel) can be cumbersome compared to ggplot2's grid system. Additionally, lattice graphics are static (no interactivity). Some legacy scripts in pharma companies still use lattice (and base R plots) for certain output, but most new development has shifted to ggplot2. One advantage of lattice is speed and simplicity for certain plots – e.g., the xyplot() function can produce a matrix of scatterplots in one line. But considering modern

needs, lattice's limitations in fine-tuning and extension mean it's often bypassed in favor of ggplot2 unless one is modifying existing code or specifically needs trellis-style multi-panel output with minimal code.

In summary, Esquisse is a helpful GUI on top of ggplot2 for interactive chart building (useful for learning or quick EDA), and lattice is an older static plotting system good for multi-panel plots but largely supplanted by ggplot2 in flexibility. Neither offers the interactivity of Plotly; Esquisse actually often ends with you using ggplot2 (so it's complementary), and lattice remains static-only and less extensible. For a clinical researcher, it's useful to know these exist: Esquisse might be introduced in a training workshop to help non-coders visualize data, and lattice might appear in legacy code or certain specialized plots in some packages.

CONCLUSION AND RECOMMENDATIONS

ggplot2 and Plotly each have distinct strengths, and together they cover the spectrum of visualization needs in clinical research. If we distill our evaluation:

Use ggplot2 for reproducibility and compliance: when you need a trusted, exact figure for a CSR (Clinical Study Report) or publication, ggplot2 is the best choice. It ensures that your visuals adhere to analysis specifications and can be regenerated exactly from the data. Regulatory and QA teams appreciate the traceability of a ggplot2 graphic, because it ties into the analysis code and outputs. The static nature of ggplot2 plots is a feature here, not a bug – it freezes the graphic as a record of the analysis result, which is important for audits and methods validation.

Use Plotly for communication and exploration: when insight and engagement are the goals – be it exploring the data yourself, or communicating findings to a team in an interactive session – Plotly shines. Its ability to let users dive into the data dynamically can lead to new discoveries (for example, noticing an outlier's tooltip reveals a data entry error, or interacting with a subgroup plot to see a pattern only in younger patients). In an era where data is abundant, interactive visualization can accelerate understanding, and Plotly provides that in R with relatively little friction.

Combine both across the project lifecycle: one can imagine a workflow where during data cleaning and exploration, you use Plotly to identify quirks in the data; during analysis and modeling, you use ggplot2 to create interim reports and model diagnostics; and in the reporting phase, you again use ggplot2 for the final polished figures, but perhaps also provide interactive supplements to stakeholders for deeper dives. This combination approach leverages each tool at the appropriate phase. For instance, an interactive efficacy dashboard might be used internally to decide the key messages, and then static ggplot2 figures of those key efficacy endpoints appear in the final submission.

In the end, the choice isn't ggplot2 or Plotly – for a savvy analyst, it's often ggplot2 and Plotly. They are complementary tools in the toolbox of clinical data visualization. One offers the solid reliability and polish required for official communications; the other offers innovation and engagement for deep data understanding. Our recommendation is to use each in the context where it adds the most value, and to be aware of their limitations. Use ggplot2 when you must be sure (and need a static record), use Plotly when you aim to explore or explain (and need interactivity), and use both when a project can benefit from an interactive exploratory phase followed by static confirmatory visuals. By doing so, clinical researchers can ensure that they extract maximum insight from their data and communicate it in the most effective manner possible.

REFERENCES

1. Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*
2. Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*
3. Emily C. Zabor, "Survival Analysis in R". (https://www.emilyzabor.com/tutorials/survival_analysis_in_r_tutorial.html)
4. Survminer cheatsheet (https://rpkgs.datanovia.com/survminer/survminer_cheatsheet.pdf)
5. Carson Sievert, 2019, "Interactive web-based data visualization with R, plotly, and shiny". Available at https://plotly-r.com/index.html
6. FDA guidance on data visualisation for clinical review reports

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Mugdha Joshi

Company: Eli Lilly

Address: 1st Floor, Building Primrose(7B) Wing B. Embassy Tech Village, Outer Ring Road, Devarabisanahalli, Bengaluru-560103

Work Phone: 8080477879

Email: joshi_mugdha@lilly.com