TPSA Rubric Evaluation Database and Interactive App

School of Economic, Political and Policy Sciences

EPPS 6354 Information Management

Oliver Jack Myers

Instructor: Dr. Karl Ho

May 5th, 2025

Table of Contents

1. Introduction

The Texas Public Safety Association (TPSA) is a nonprofit Career Technical Student Organization (CTSO) dedicated to providing leadership development and hands-on competitive experiences for high school students interested in public safety careers. TPSA organizes events spanning law enforcement, fire services, forensic science, legal studies, and corrections. Each event is evaluated using detailed rubrics assessing presentation quality, safety, protocol adherence, and professionalism. With over 32 distinct competitive events, TPSA faces significant challenges in effectively analyzing rubric criteria performance. TPSA staff, primarily volunteer high school teachers, lack sufficient time and resources to review and adjust rubrics across multiple conferences manually. Without a centralized database or visualization system, TPSA struggles to utilize student scoring data for rubric refinement efficiently.

To address these challenges, this project developed a centralized PostgreSQL database integrated with an interactive web shiny application. A relational database schema was designed and implemented using TPSA's existing CSV datasets on student scores, rubrics, and competitive events, conferences, rubric criteria and more. This involved creating tables within a pgAdmin-managed server, populating the database, and subsequently hosting it on the Railway cloud platform to ensure broader accessibility beyond a local environment. The core of the analytical interface is a web application built using the shiny package (Chang et al. 2024), which provides the

framework for user interaction and dynamic data display. This application allows users to select a specific TPSA conference via dropdown menus. Upon selection, SQL queries are dynamically constructed and executed against the database to retrieve relevant performance data. This interaction with the PostgreSQL database is managed through the DBI (Database Interface) package (R Special Interest Group on Databases (R-SIG-DB) et al. 2024), which provides a consistent framework for database communication, and the RPostgres driver (Wickham, Ooms, and Müller 2025), which facilitates the specific connection to PostgreSQL. To optimize database interactions and manage connections efficiently, particularly in a multi-user web application context, the pool package (Cheng, Borges, and Wickham 2024) was employed to create and manage a pool of database connections.

The application initially presents an overview of all competitive events for the selected conference, with data often summarized and manipulated using functions from the dplyr package (Wickham et al. 2023). Users can then drill down by selecting a specific competitive event. This action triggers further SQL queries, filtered by the chosen conference and event, to display a detailed breakdown of rubric criteria. This granular view is visualized using interactive bar charts generated by the highcharter package (Kunst 2022), which leverages the Highcharts JavaScript library for rich, customizable graphics. These charts display average student scores for each criterion, with bars color-coordinated to visually signal performance status (e.g., indicating criteria with notably low or high average scores). Complementing the graphical display,

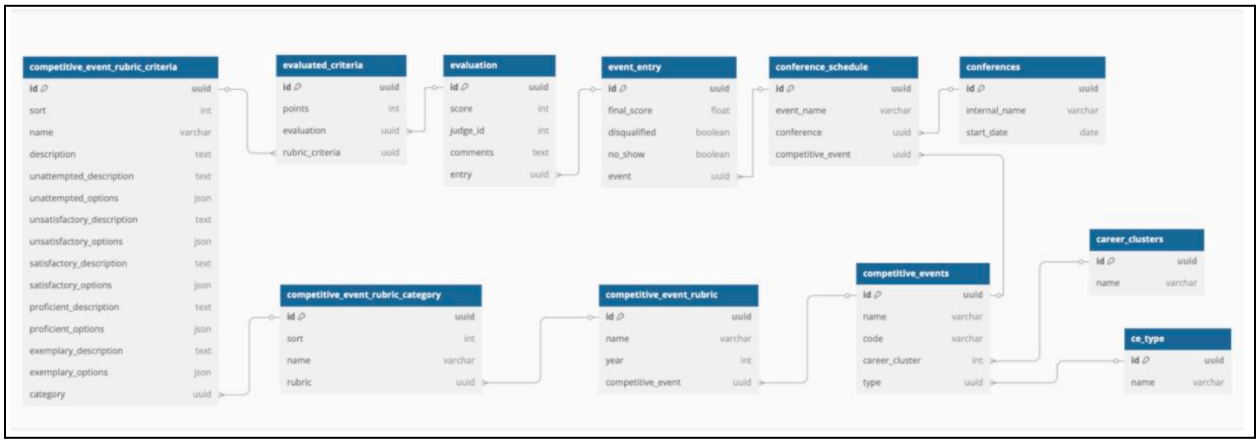
interactive data tables are rendered using the reactable package (Lin 2023), offering sortable and searchable tabular views of the criteria and scores. The shiny application's overall aesthetic and user experience are enhanced using the bslib package (Sievert, Cheng, and Aden-Buie 2025), allowing for the easy application of modern Bootstrap themes and custom styling. The resulting database and application provide TPSA with an efficient platform for analyzing event performance and rubric effectiveness, enabling staff to objectively assess competitive event rigor and track the impact of rubric modifications over time.

2. Data and Database Design

2.1 Data, Database Creation (pgAdmin) and Railway

Initial data included CSV files of student scores across various competitive events and conferences, acquired through collaboration with TPSA’s IT Director. This data was supplemented with rubric criteria and career clusters obtained from TPSA’s website. Information on event judges was later incorporated to provide comprehensive analytical capabilities. Using pgAdmin, a PostgreSQL database titled "Railway Server" was created to host the integrated dataset. The database was subsequently hosted on Railway, ensuring cloud accessibility and seamless integration with a Shiny application.

Figure 1: Schema



The database schema includes interconnected tables capturing details about conferences, competitive events, evaluation rubrics, student scores, and judges. The schema was carefully structured to allow efficient querying and comprehensive data retrieval for real-time analysis within the Shiny app.

Figure 2: Postgres Tables

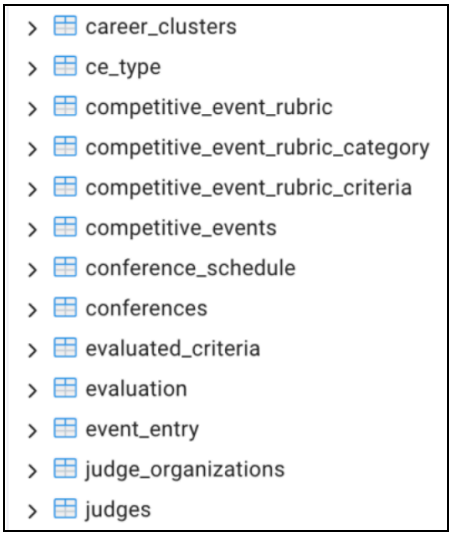
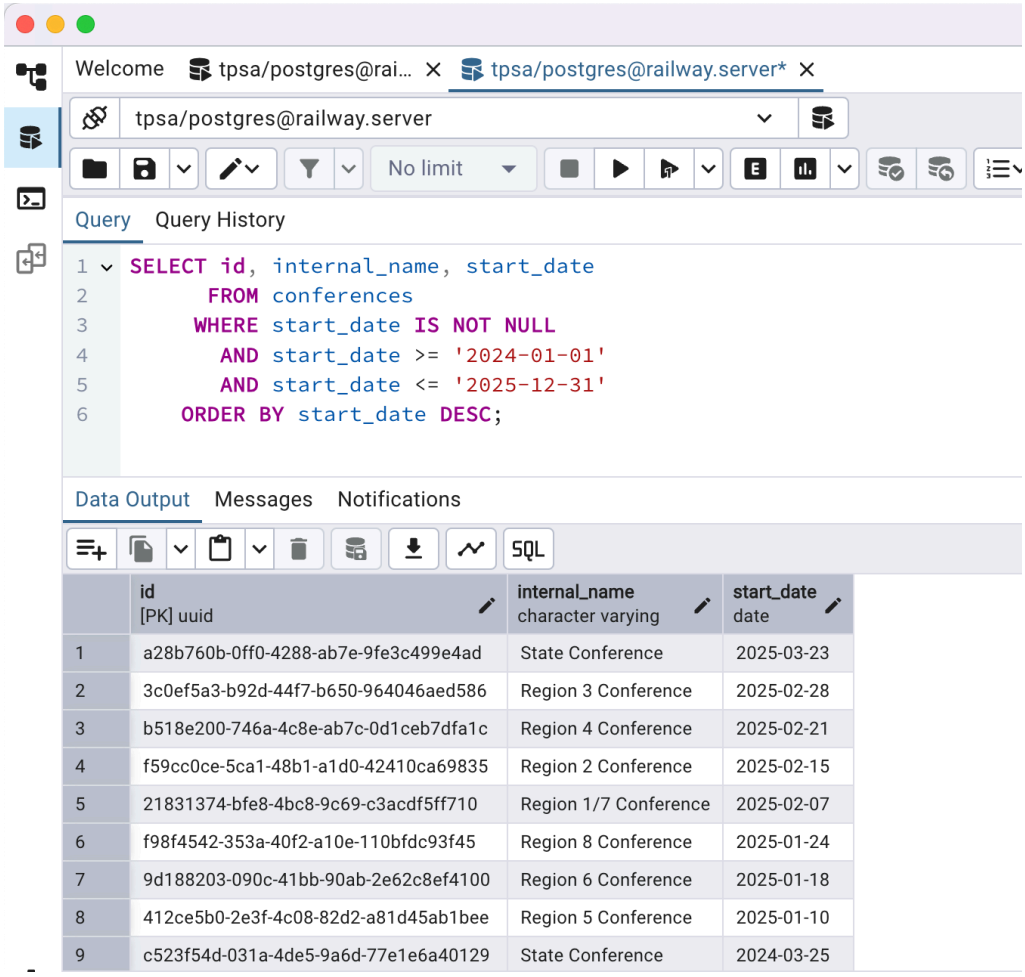


Figure 3: Example SQL Query in pgAdmin

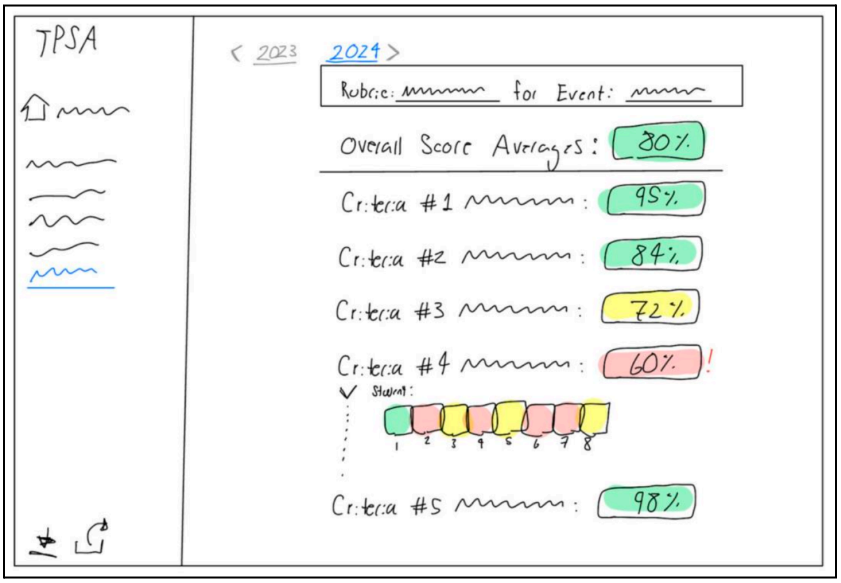


Here is the SQL logic (in pgAdmin) for selecting conference internal names and dates descending in the timeframe I have data for. This will work as the initial dropdown for the user to select the conference by name that they would like to see competitive event data for.

3. Shiny App Implementation

3.1 Initial UI Dashboard Sketch

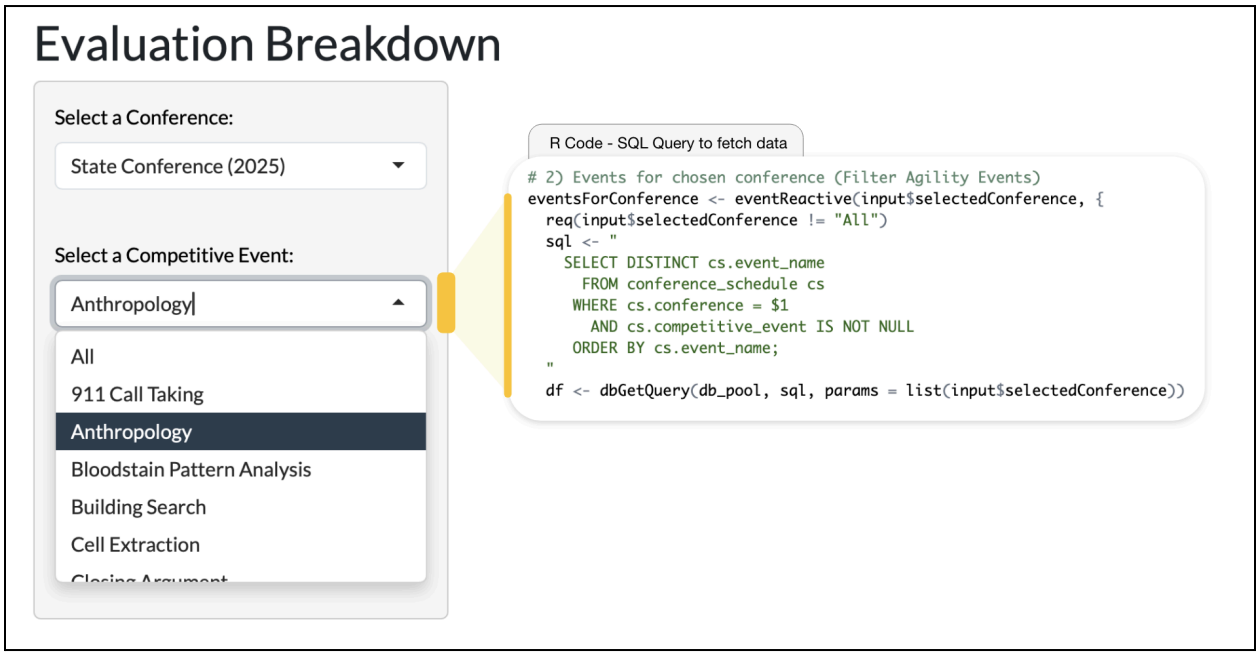
Figure 4: Initial UI Sketch



The Shiny application was designed to serve two primary functions: an overview dashboard showing overall competitive event performance per conference, and detailed views for individual competitive events to assess specific rubric criteria. The dashboard facilitates rapid identification of rubric criteria that may require adjustments based on aggregated student performance data. This interface enables TPSA staff to quickly discern event trends, adjust rubric difficulty appropriately, and measure the effects of rubric modifications over subsequent conferences.

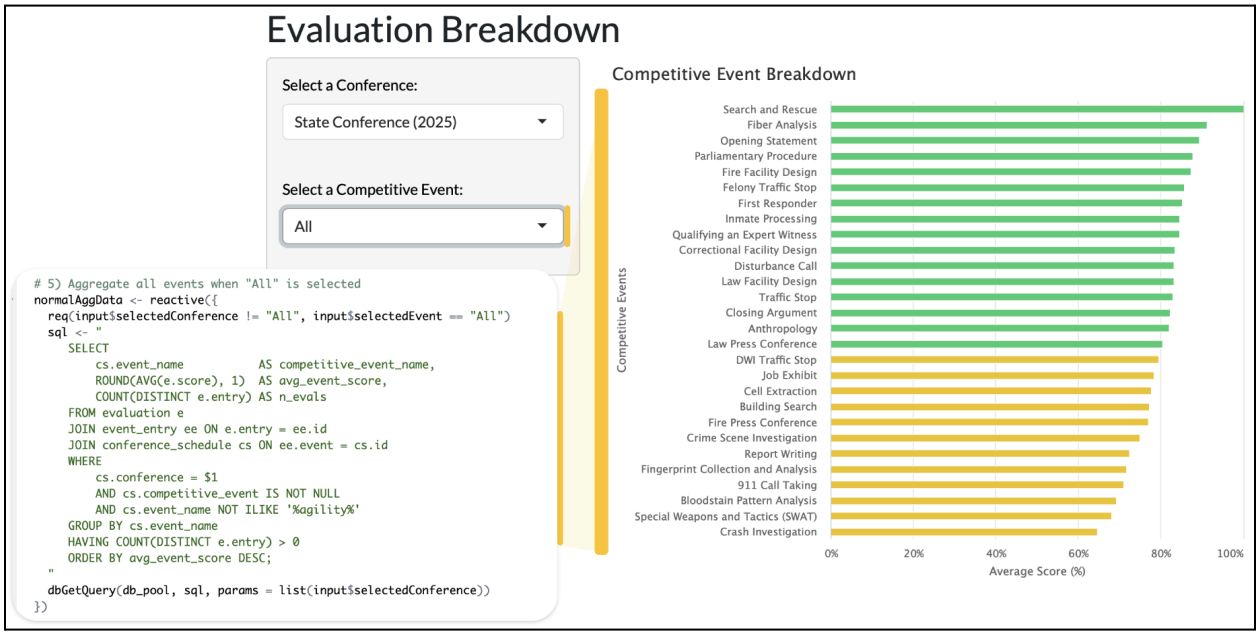
3.2 Shiny App Interactive Dashboard for viewing Conferences and Competitive Event Rubrics, and SQL Queries

Figure 5: SQL and UI for selecting a competitive event



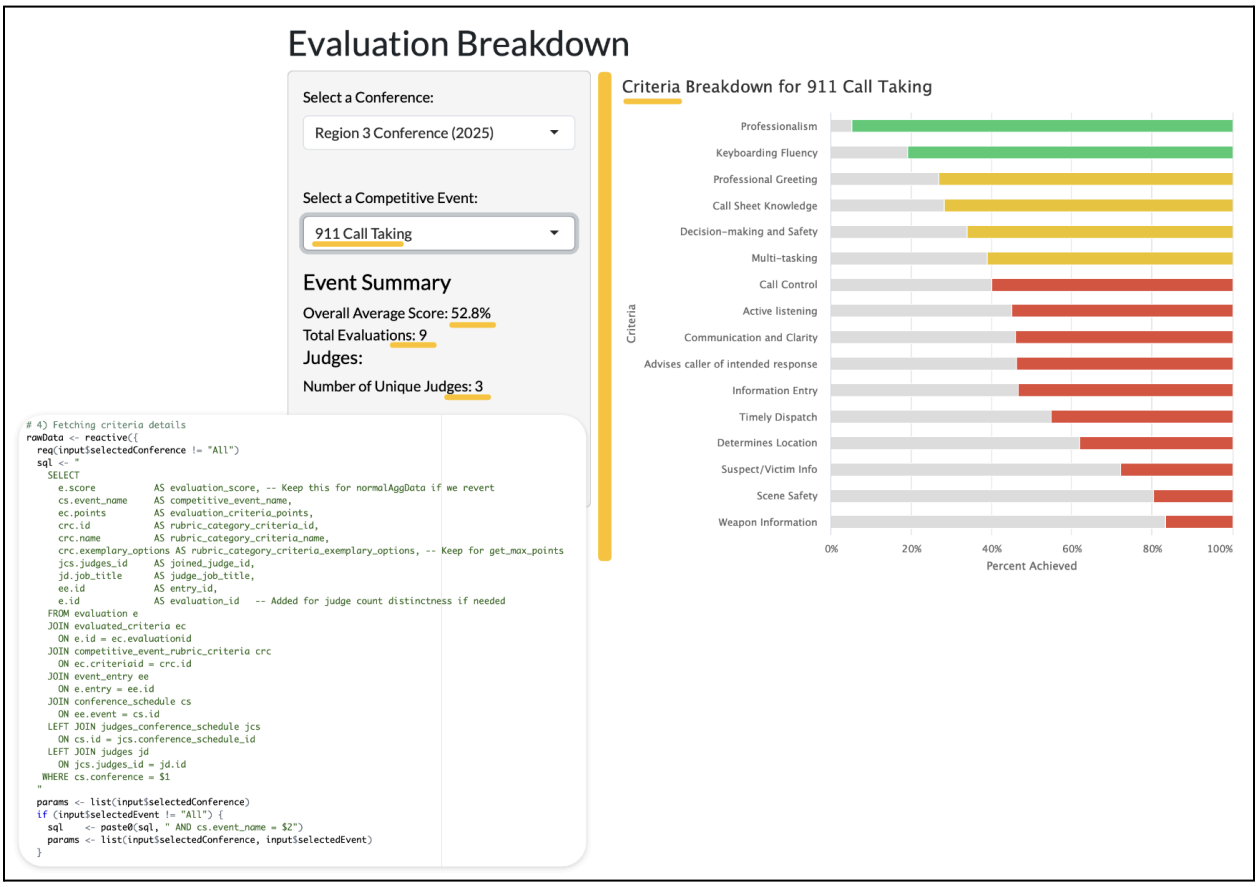
The dashboard begins by allowing users to select a specific conference from a dropdown populated by a SQL query returning conference names and dates within a specified range. Following conference selection, a second SQL query retrieves distinct competitive event names, excluding problematic events labeled "agility," thereby providing cleaner and more relevant analyses.

Figure 6: SQL and UI for selecting a competitive “All” and viewing the conference wide data



If users select "ALL" competitive events (the default setting), a query calculates the average score for each event, color-coding results—yellow for scores below 80% and red for scores below 60%—allowing for immediate visual identification of underperforming events. Selecting an individual competitive event triggers another SQL query, retrieving rubric criteria and average scores per criterion, providing deeper insights into rubric performance and facilitating targeted adjustments.

Figure 7: SQL and UI for Rubric criteria given a specific selected competitive event for a conference, displaying scores per criteria aggregated by all scores of that group.



The code above shows that when a user selects a particular competitive event from the 'select a competitive event' dropdown, my SQL query retrieves information from the database to present the rubric criteria and aggregate the average scores per criterion, finally allowing the potential TPSA Staff user to see which criteria are performing well or may be too easy, and which are performing poorly and may be too difficult.

4. Conclusion

4.1 Discussion (what I learned)

This iterative project provided valuable real-world experience in managing, modeling, and visualizing complex data. Collaboration with TPSA, particularly direct interactions with their CTO and attendance at their summer conference, deepened understanding of the organization's needs and informed dashboard development. This practical application of Shiny (Chang et al. 2024), bslib (Sievert, Cheng, and Aden-Buie 2025), and reactable (Lin 2023) packages significantly advanced data modeling skills, particularly in designing interactive visualizations and custom tooltips for enhanced user experience. Utilizing jsonlite (Ooms 2014) effectively resolved complexities in handling rubric scoring criteria.

4.2 Challenged Encountered

- Filtering out agility events as they had problematic scores that skewed the data and proved to be unhelpful.
- When the user is looking at a competitive event, the graphic becomes problematic in its current version. I had issues with getting the axis for the graphic to work correctly; it shows up on the right side instead of the left. When I

have made simple changes, it makes the object not show up at all. I will need to fix this in the future.

- Custom tool tips, I struggled with ggplot to get the graphic I wanted to show by breaking down the criteria and having custom tool tips initially. So, I used AI to help explore other options and found a really cool package called “highcharter” that gave me more flexibility, a sleeker UI design, and allowed me to use JavaScript for more customization of the tool tips.
- I was initially struggling to obtain “max points” for each of the rubric criteria. Each rubric criterion has its own set of points, which can vary significantly from rubric to rubric and even among different criteria. With the help of chatgpt, I was able to use jsonlite to debug this issue and retrieve the json data for the max points.

4.3 Future Steps

Next steps include presenting the evaluation breakdown tool to TPSA stakeholders for usability testing and feedback. Based on their input, we can refine the interface and analysis. We would also enhance the tool by incorporating predictive modeling to simulate changes in average scores for specific rubric criteria, forecasting their impact on overall event outcomes. Additionally, collaborating with TPSA, we could develop a feature to explore different 'cut-off' scores needed for state advancement, analyzing how various thresholds affect qualification rates and event competitiveness, thereby offering data-driven insights for future policy adjustments.

4.4 Important project links

Live APP Link: https://oliverjackmyers.shinyapps.io/TPSA_ScoreCard_Data/

Full Code Link:

https://oliverjackmyers.github.io/pages/EPPS_6354/projectproposal/EPPS_6354_projectproposal.html

References

Cheng, Joe, Barret Borges, and Hadley Wickham. 2024. pool: Object Pooling. R package version 1.0.4. https://CRAN.R-project.org/package=pool

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jonathan

Allen, Jeff McPherson, Alan Dipert, and Barbara Borges. 2024. shiny: Web Application Framework for R. R package version 1.10.0. https://CRAN.R-project.org/package=shiny

Kunst, Joshua. 2022. highcharter: A Wrapper for the 'Highcharts' Library. R package version 0.9.4. https://CRAN.R-project.org/package=highcharter

Lin, Greg. 2023. reactable: Interactive Data Tables for R. R package version 0.4.4. https://CRAN.R-project.org/package=reactable

Ooms, Jeroen. 2014. “The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects.” arXiv:1403.2805 [stat.CO]. https://arxiv.org/abs/1403.2805

R Special Interest Group on Databases (R-SIG-DB), Hadley Wickham, and Kirill Müller. 2024. DBI: R Database Interface. R package version 1.2.3. https://CRAN.R-project.org/package=DBI

Sievert, Carson, Joe Cheng, and Garrick Aden-Buie. 2025. bslib: Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'. R package version 0.9.0. https://CRAN.R-project.org/package=bslib

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. dplyr: A Grammar of Data Manipulation. R package version 1.1.4. https://CRAN.R-project.org/package=dplyr

Wickham, Hadley, Jeroen Ooms, and Kirill Müller. 2025. RPostgres: C++ Interface to PostgreSQL. R package version 1.4.8. https://CRAN.R-project.org/package=RPostgres

I. AI Appendex

Example AI Prompts:
a. Example Prompt:
“Can you change the broken-down criteria view so it behaves the same way as the ‘All’ selection? It isn’t working correctly for specific events.”

b. AI Response (Excerpt):

```
output$breakdownChart <- renderHighchart({
if (input$selectedEvent == "All") {
df <- normalAggData()
… # aggregate chart code
} else {
df <- aggData()
… # per-criterion chart code
}
})

output$breakdownTable <- renderReactable({
if (input$selectedEvent == "All") {
df <- normalAggData()
… # table of events
} else {
df <- aggData()
… # table of criteria
}
})
```

c. Example Prompt:
“For the 'All Events' bar chart in my Shiny app using highcharter, I want to distinguish performance levels visually. How can I dynamically color the bars based on the avg_event_score? For example, green for scores > 80%, yellow for 60-80%, and red for < 60%?"

AI Response (Excerpt):

```
color_vec <- ifelse(df$avg_event_score > 80, '#2ecc71',
ifelse(df$avg_event_score >= 60, '#f1c40f', '#e74c3c'))
```

Tools Used:

Google AI Studio (Gemini developer preview)

ChatGPT (paid subscription 4o, 04-mini-high)

Explanation of AI Tool Usage:
a. Chart Customization: AI was instrumental in identifying and implementing the "highcharter" package, enabling dynamic color coding, stacked bar charts, and interactive tooltips to enhance UI and user experience.
b. Reactive Data Pipelines: Leveraged AI for optimizing eventReactive blocks (normalAggData, aggData, rawData) to efficiently manage data retrieval and caching, significantly improving application responsiveness.
c. Debugging: Used AI extensively to debug existing R code and SQL queries, resolving logical errors, refining data filtering (notably removing problematic "agility" events), and addressing edge cases like handling NA values and empty datasets.

Reasons for AI Usage:
a. Efficiency: Significantly reduced development and debugging time.
b. Visual Enhancement: Improved app aesthetics through strategic visual elements such as color-coding, tooltips, and layout adjustments.
c. Complex Feature Implementation: AI facilitated the integration of advanced visualization features using Highcharts and Reactable, bridging gaps in initial technical expertise.
d. SQL Optimization: Enhanced efficiency of SQL queries to improve app performance.
e. Learning and Validation: AI explanations and validations deepened understanding of Shiny, R, and JavaScript integrations, improving overall development skillsets.