2025 ■ Vol. 21 ■ no. 2

Creation of a Combined Violin and Box Plot Using R

Cynthia-Maria Kanaan [a] and Justine Bellavance [a,b]

[a]School of Psychology, University of Ottawa

[b]Département de psychologie, Université du Québec à Montréal

Abstract ■ The ability to effectively visualize and communicate complex data is crucial, especially when conveying research findings to audiences who may lack specialized training in data interpretation, such as the general public (Grierson et al., 2015; Nordmann et al., 2022). This article presents a comprehensive tutorial for creating an informative infographic using R Studio to visualize experimental psychology data. Focusing on a hypothetical concussion study that compares concussed athletes to healthy controls over three time points, we demonstrate how to generate a combined violin and box plot to effectively display reaction time data across these different groups and time points. This tutorial is designed to guide anyone (i.e., undergraduate students, researchers, clinicians) through the process of creating visually appealing and informative data representations. The article details the step-by-step procedure and creation process, explains the resulting infographic, and discusses its potential applications in experimental psychology.

Keywords ■ Plots, violin plots, box plots. **Tools** ■ R, ggplot.

✉ Jbell164@uottawa.ca

10.20982/tqmp.21.2.p032

Acting Editor ■ Denis Cousineau (Université d'Ottawa)

Reviewers
■ One anonymous reviewer

Introduction

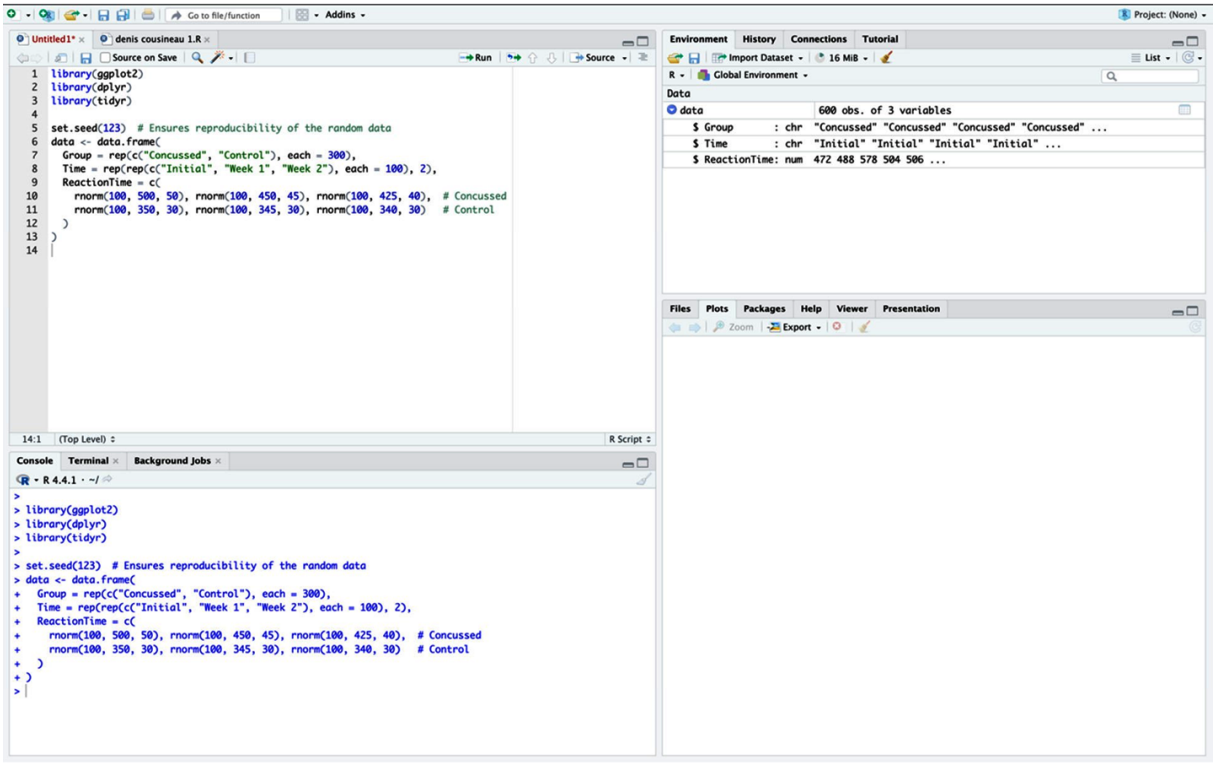
In the field of experimental psychology, research methodologies are becoming increasingly sophisticated, and datasets are growing larger (Grierson et al., 2015). Consequently, traditional data visualization methods (e.g., simple bar graphs, line plots) often fail to capture the nuanced distribution of complex and multifaceted findings (Grierson et al., 2015). This challenge is particularly pressing when trying to communicate results to audiences who may lack specialized training or education in data interpretation, such as the general public (Grierson et al., 2015; Nordmann et al., 2022). These limitations highlight the growing need for clear, concise, and informative data representation techniques (Grierson et al., 2015). Moreover, the replication crisis in psychology has sparked extensive discourse on the importance of adopting reproducible research practices (Nordmann et al., 2022).

One promising solution is the use of R, a powerful statistical programming language widely adopted in psychological research (R Core Team, 2024). R's code-based approach to data visualization aligns well with these concerns, allowing for easy sharing and replication of exact visualization methods, as well as flexible data analysis (Nordmann et al., 2022). R also provides researchers with near-complete autonomy over every aspect of graph creation, enabling highly customized and professional outputs (Nordmann et al., 2022).

By using code for visual representations of data (e.g., producing plots), researchers can easily modify and repurpose their methods, thus enhancing efficiency and consistency in their work (Nordmann et al., 2022). While coding is initially daunting, this flexibility is highly advantageous, having even led major media outlets such as BBC and The New York Times to adopt R as their primary data visualization tool (Nordmann et al., 2022).

Building on these advantages, this tutorial provides a step-by-step guide to creating infographics that combines violin plots and overlaid box plots using R. This approach was chosen as it offers a comprehensive view of data distribution while simultaneously displaying key summary statistics. Specifically, violin plots effectively illustrate the probability density of data at different values, while box plots show critical metrics such as median and interquartile range (Hintze & Nelson, 1998). Together, they provide a rich and informative representation of complex datasets, provinding nuances that might be obscured by simpler visualizations (Hintze & Nelson, 1998; Correll & Gleicher,

Figure 1 ■ Screenshot of the RStudio environment displaying the generated dataset of 600 reaction time measurements across 3 time points.



2014). Furthermore, we will include various tips for R Studio novices, such as layering plots and colour changes, as those are common problems encountered during R Studio usage.

As a novice to R, RStudio, and these visualization techniques ourselves, this tutorial is designed to be accessible to beginners while still offering insights for more experienced users. Our personal learnings and reflections will be detailed at the end of this article, providing additional perspective for those new to these methods.

Method for the creation of a combined violin – Box plot Infographic

The following methods are made using a macOS system. Users operating on different systems (e.g., Windows, Linux) may need to adjust certain steps accordingly.

Step 1 – Installing the Software

To begin this tutorial, you need to install R, a free and open-source statistical programming environment (R Core Team, 2024). You can download R can from The Comprehensive R Archive Network (CRAN) website (R Core Team, 2024). First, select the link corresponding to your operating system — Linux, macOS, or Windows. For macOS users, click on the "Download R for macOS" link. This action will direct you to a page listing available package versions tailored to specific hardware configurations and operating system versions. For example, a MacBook Pro from 2023 running the latest version of macOS Sonoma may require a different package than an older MacBook Air from 2017 with an earlier macOS version. It is generally advisable to select the latest release that is compatible with your operating system. After downloading the package, double-click the installer file and follow the on-screen prompts to complete the installation process.

Following this, we recommend installing RStudio, an integrated development environment (IDE) that significantly enhances the user experience when working with R (R Core Team, 2024). You can download RStudio from the posit website (posit.co/downloads/). As with R, you will need to select the appropriate installer for your operating system, click on it, and follow the prompts to complete the installation.

Once both R and RStudio are successfully installed,

launch RStudio where the expected layout upon opening includes the console (left), environment pane (top right), and working directory (bottom right).

Step 2 – Setting up the R Environment and Packages

To effectively visualize data in R, you will need to install and load three packages: `ggplot2` for creating plots (Wickham, 2016), `dplyr` for data manipulation (Wickham et al., 2023), and `tidyr` for data reshaping (Wickham et al., 2024). Begin by copying and pasting the following command into the console:

```
install.packages(c("ggplot2", "dplyr", "tidyr"))
```

Then, press the “run” icon located at the top right of the console; it will show the confirmation message that appears upon successful installation. This command installs all three packages simultaneously.

You will then need to open a new R script. Navigate to the top left corner of the RStudio interface, click the "New File" icon, located at the top left corner of the RStudio interface, and select the "R Script" option from the resulting dropdown menu. This action will open a new window for entering your commands. There, enter the following commands to load the necessary libraries into your R environment, making their functions available for use:

```
library(ggplot2)
library(dplyr)
library(tidyr)
```

Press the “Run” icon after each command or select all three commands together to execute them simultaneously. You should see indications in the console confirming that the libraries have been successfully loaded (see Figure 1).

Step 3 – Generating the Sample Data

In a real research scenario, you would import your actual data next. However, for the purposes of this tutorial, we will generate a simulated dataset representing reaction time measurements from a hypothetical concussion study (see Appendix A for the specific code sample). This dataset will include two groups: concussed athletes ($n = 300$) and healthy controls ($n = 300$), measured at three different time points (Initial, Week 1, and Week 2).

To generate your sample dataset, run the first listing of Appendix A. This code creates a dataframe with 600 observations, with each observation representing a reaction time measurement for one participant at one time point (see Figure 1). The `rnorm` function creates normally distributed reaction times for each group across different time points, simulating the variability typically observed in such studies.

It is worthy of noting that in this study, it is not required to know what the three measurements of a given participant are, but it might be needed when performing an ANOVA or other statistics. In this case, an additional command is needed to ID the participants in the different measures (see second listing of Appendix A for the code sample).

Step 4 – Generating the Plot

With your data prepared, you are now ready to create your visualization using `ggplot2`, a powerful package that simplifies the creation of complex and visually appealing plots. In this step, you will layer different graphical elements to build a combined violin plot with overlaid box plots.

First, enter the code from Listing 1 into RStudio.

This code initializes a ggplot object with your data and sets the aesthetic mappings for the x-axis (Time), y-axis (ReactionTime), and fill color (`scale_fill_brewer()`) based on the Group variable. The `geom_violin()` function adds violin plots to display the distribution of reaction times, while `geom_boxplot()` overlays box plots to present summary statistics. The `labs()` and `theme_minimal()` functions customizes the plot titles and axis labels for clarity, and the `position_dodge(0.7)` function visually separates the groups.

Then, to display your visualization, run the following command:

```
print(plot)
```

This command will render your plot in the RStudio viewer, allowing you to effectively compare reaction times between concussed athletes and control athletes across different time points (see Figure 2).

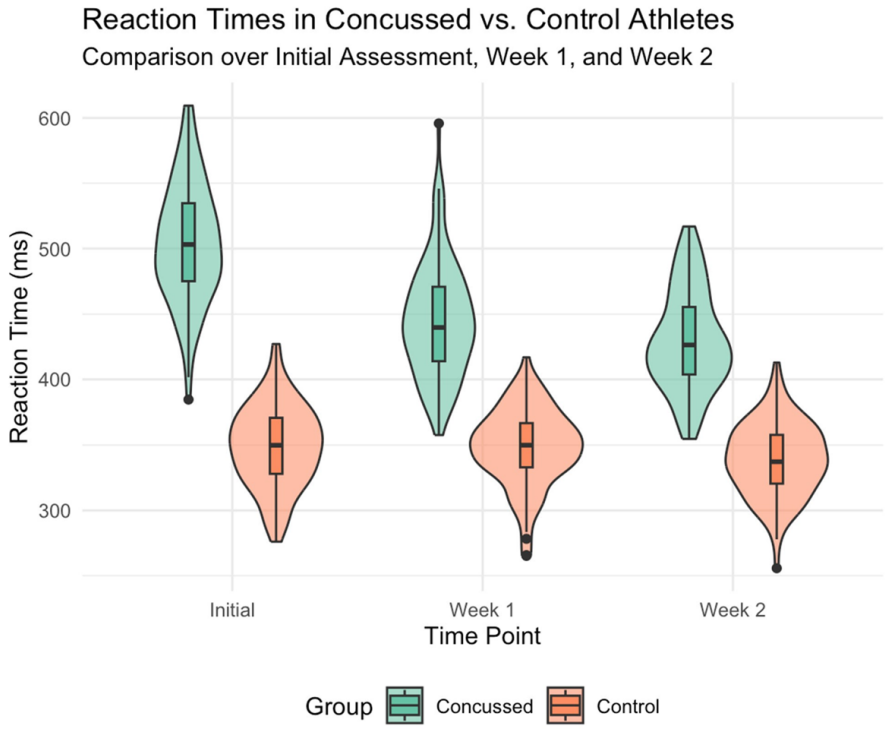
The resulting plot can be seen as a layered representation of data, where each component adds valuable information.

The first layer is a box plot (see Figure 3a). It is a widely used tool for summarizing the distribution of data, providing insights into key statistical properties such as symmetry, skewness, variance and outliers. The box plot is particularly useful for comparing multiple groups simultaneously, as it clearly shows the central tendency (median) and dispersion (interquartile range). However, the box plot visualization does not display the full density of the data, potentially obscuring patterns within the group distribution, which is an important limitation of this method.

To address this, the second layer introduces the violin plot (see Figure 3b). This technique enhances the visualization by illustrating the probability density of the data. Unlike box plots, violin plots depict the shape of the data distribution, revealing multimodal tendencies within a group that might otherwise go unnoticed.

By combining both representations, this visualization balances clarity and depth, offering a more comprehensive

Figure 2 ■ Final graph produced in R: violin plot with overlaid Box plots comparing reaction times between concussed athletes and healthy controls over 3 time points.



understanding of complex datasets. Both the key characteristics and the distribution density can be seen as a glance (see Figure 2). Researchers can quickly grasp summary statistics while also observing finer distributional details, making this approach particularly useful in fields where nuanced data interpretation is critical.

Layering representations makes the possibilities endless. For example, we can layer jittered dots over the combined violin and box plot (see Figure 4), to nuance the data even more.

Layering Different Plots

As mentioned previously, layering different types of visuals can be an interesting and effective way to present nuanced and comprehensive data. However, this approach requires specific coding knowledge and may sometimes create issues with the visuals themselves.

When combining a violin plot with a box plot, it is important have the right layering order. For example, in Figure 4b, the violin plot nearly hides the totality of the box plot. This is due to the transparency setting or the layering sequence of the violin plot. There are two possible solutions to address this issue.

Layer order

Firstly, adjusting the layer order will make the box plot more visible on the graph. To do this, simply arrange the commands in the correct sequence from Listing 2.

The key commands here are `geom_boxplot()` and `geom_violin()`. Their order will determine what remains visible in the graph. Since the violin plot is typically lareger than the box plot, it is recommended to place the box plot on top of the violin plot, by calling `geom_violin()` first in the console. This ensure the violin plot is layered underneath, resulting in a graph like shown in Figure 2, where both plots are clearly visible.

Transparency

When layering more than two different plots, as seen in Figure 4, left right panel, adjusting the layering order alone may not suffice to ensure all different elements remain visible. In such cases, controlling the transparency of each layer can help resolve the issue. This can be done using the `alpha` parameter, which should be included in the command for each plot (shown in bold), as seen in Listing 3.

To reduce the opacity of a visual, the `alpha` value must

Listing 1 ■ Code Used to Generate the Combined Violin and Box Plot Infographics.

```
library(ggplot2)
library(dplyr)
library(tidyr)

plot <- ggplot(data, aes(x = Time, y = ReactionTime, fill = Group)) +
  geom_violin(position = position_dodge(0.7), alpha = 0.6) +
  geom_boxplot(width = 0.1, position = position_dodge(0.7)) +
  labs(title = "Reaction Times in Concussed vs. Control Athletes",
       subtitle = "Comparison over Initial Assessment, Week 1, and Week 2",
       x = "Time Point",
       y = "Reaction Time (ms)",
       fill = "Group") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2") +
  theme(legend.position = "bottom")

print(plot)
```

be lowered. As shown in Figure 4, bottom panel, the right-handed graph uses an `alpha` of 0.8 for the jittered dots, making the box plot difficult to see. In contrast, the left-handed graph lowers `alpha` to 0.4 for the same dots, allowing the background layers to become more visible (as seen in Figure 4, top left panel). Adjusting the opacity of each layer helps ensure that all combined plots remain distinguishable.

When combining multiple plots, it is essential to maintain the visibility of each individual component. Both the layering order and transparency settings of each component play a crucial role in making the final graph clear and readable.

Changing the Colors of Graphs

Changing the Color Using Premade Color Packages

Some scientific journals have strict color requirements, which can be challenging to manage in certain programs. However, in RStudio, this issue can be easily addressed. To do so, you will need to install the `RColorBrewer` package, using the command `install.packages("RColorBrewer")`. Once installed, you can modify color palettes for your graphs. After installing this package, you can start changing color palettes for your graphs. In the following lines of codes, the command `scale_fill_brewer(palette= "")` controls the color scheme, with the specified palette name determining the color used.

```
#[...]
  theme_minimal() +
  scale_fill_brewer(palette = "Set2") +
  theme(legend.position = "bottom")
```

For example, changing the `Set2` color palette (see Figure 2) to `Accent` in the previously mentioned command alters the colors in the graph — shifting from orange and aquamarine to green and lilac, as shown in Figure 5a. There are multiple pre-made color palettes in `ColorBrewer` (See Figure 6 for all the available color palettes).

Using Individual Colors

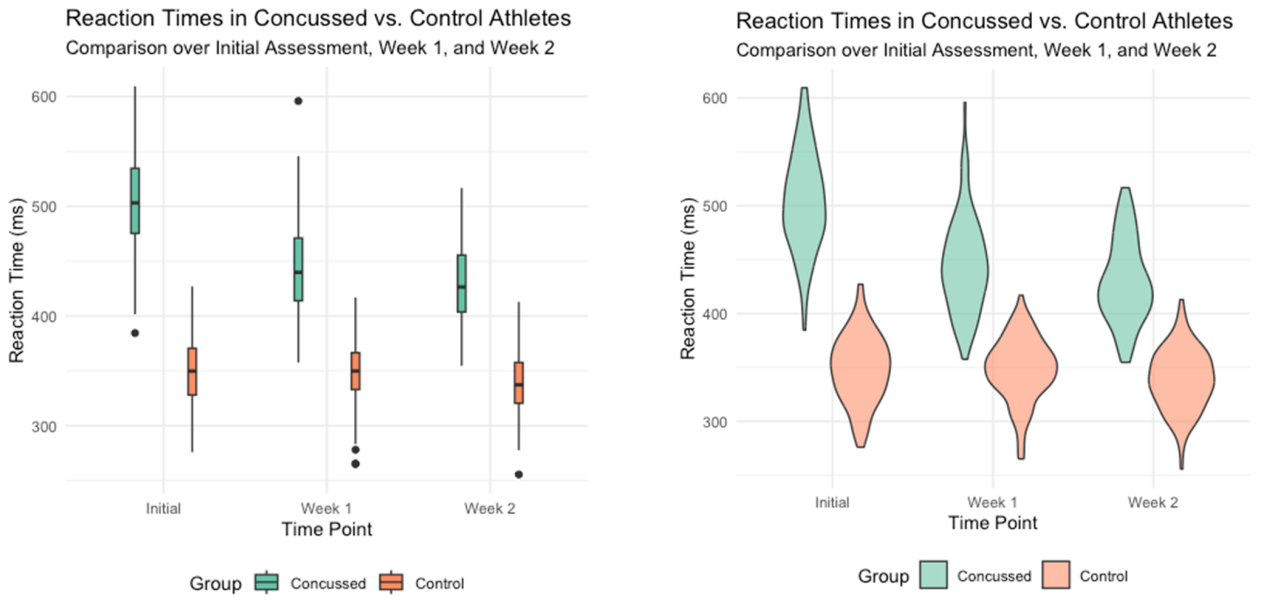
It is also possible to change colors individually, which can be especially useful when layering multiple plots. For example, when combining a violin plot, a box plot, and jittered dots (see Figure 4, left panel), distinguishing each element can be challenging. Adjusting the color palette or assigning unique colors to each plot enhances visualization and improves readability. To achieve this, install the package `ggnewscale` using the following command:

```
install.packages("ggnewscale")
```

Next, modify the `scale_fill_brewer(palette = "")` command in your code as follows, where the bolded text represent predefined color values. These color valules cann be found online or in Frazier's cheat sheet (Frazier, 2020).

A line of code for an individual graph, such as a a violin plot, should look like this, with the bolded line of code specifying the individual color change for each group.

Figure 3 ■ Individual Box plot visualization graph comparing reaction times between the two groups over 3 time points (left) and individual violin plot visualization graph comparing reaction times between the two groups over 3 time points.



Listing 2 ■ Code to obtain a layered plot

```
#[...]
p_final <- ggplot(data, aes(x = Time, y = ReactionTime, fill = Group)) +
  geom_violin(position = position_dodge(0.7), alpha = 0.6) +
geom_boxplot(width = 0.1, position = position_dodge(0.7), outlier.shape = NA, alpha
    = 0.8)
#[...]

#And not:
#
#[...]
#p_final <- ggplot(data, aes(x = Time, y = ReactionTime, fill = Group)) +
#    geom_boxplot(width = 0.1, position = position_dodge(0.7), outlier.shape = NA,
   alpha = 0.8) +
#    geom_violin(position = position_dodge(0.7), alpha = 0.6)
#[...]
```

Listing 3 ■ Code for manipulating transparency

```
#[...]
p_final <- ggplot(data, aes(x = Time, y = ReactionTime, fill = Group)) +
  geom_violin(position = position_dodge(0.7), alpha = 0.6) +
  geom_boxplot(width = 0.1, position = position_dodge(0.7), outlier.shape = NA,
   alpha = 0.8)
  geom_jitter(aes(color = Group),
              position = position_jitterdodge(jitter.width = 0.2, dodge.width = 0.7
   ),
              size = 1.5, alpha = 0.4) +
#[...]
```

```
library(ggnewscale2)

ggplot(data, aes(x = Time, y = ReactionTime,
    fill = Group)) +
  geom_violin(position = position_dodge(0.7),
    alpha = 0.6) +
  scale_fill_manual(values = c("Concussed" =
    "brown4", "Control" = "brown1"))
```

This line of code should be added after each command for every individual plot layer (see Appendix D for the full code). An example of the final graph can be seen in Figure 5.

However, be sure to include the command `new_scale_fill() +` between the first and second plot to allow the next plot layer to have its own scale. Additionally, for the jittered dots, use `scale_color_manual()` instead, as points are colored using the 'color' aesthetic rather than 'fill'.

Results

The resulting infographic (Figure 2) effectively demonstrates the tutorial's objectives by illustrating reaction times in concussed versus control athletes across multiple time points. It provides a comprehensive view of the simulated data, highlighting both distribution and central tendencies. The plot reveals that concussed athletes exhibit higher and more variable reaction times compared to healthy controls, reflecting the cognitive impairments and heterogeneous nature of concussion symptoms. Notably, there is a decreasing trend in reaction times for the concussed group over the two-week period, suggesting potential recovery, while the control group maintains stable reaction times throughout the hypothetical study.

This clear visual representation not only enhances understanding of complex data but also demonstrates the effectiveness of using `ggplot2` for creating informative graphics that facilitate interpretation of experimental results. The step-by-step process, from data preparation to final plot creation, showcases R's power and flexibility in handling multifaceted datasets, thus achieving the tutorial's goal of providing a practical educational tool for data visualization in psychological research.

References

Correll, M., & Gleicher, M. (2014). Bad for data, good for the brain: Knowledge-first axioms for visualization design.

Frazier, M. (2020). *R color cheatsheet*. https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf

Grierson, H., Corney, J., & Hatcher, G. (2015). Using visual representations for the searching and browsing of large, complex, multimedia data sets. *International Journal of Information Management*, *35*(2), 244–252. doi: 10.1016/j.ijinfomgt.2014.12.003.

Hintze, J. L., & Nelson, R. D. (1998). Violin plots: A box plot-density trace synergism. *The American Statistician*, *52*(2), 181–184. doi: 10.1080/00031305.1998.10480559.

Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. (2022). Data visualization using R for researchers who do not use R. *Advances in Methods and Practices in Psychological Science*, *5*(2), 2515245922107. doi: 10.1177/25152459221074654.

R Core Team. (2024). *R: The R project for statistical computing* [CREATION OF COMPLEX INFOGRAPHICS USING R]. Retrieved October 20, 2024, from https://www.r-project.org/

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis* [Springer-Verlag]. https://ggplot2.tidyverse.org/

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *dplyr: A grammar of data manipulation* [R package version 1.1.4]. https://CRAN.Rproject.org/package=dplyr

Wickham, H., Vaughan, D., & Girlich, M. (2024). *tidyr: Tidy messy data* [R package version 1.3.1]. https://CRAN.R-project.org/package=tidyr

Listing 4 ■ Code Used to Demonstrate Color Changing Capabilities Of R Studio.

```
library(ggplot2)
library(ggnewscale)  # Allows multiple fill scales

p_final <- ggplot(data, aes(x = Time, y = ReactionTime)) +
  geom_violin(aes(fill = Group),
              position = position_dodge(0.7), alpha = 0.6) +
  scale_fill_manual(values = c("Concussed" = "royalblue1", "Control" = "
   palevioletred1")) +
  new_scale_fill() +
  geom_jitter(aes(color = Group),
              position = position_jitterdodge(jitter.width = 0.2, dodge.width = 0.7
   ),
              size = 1.5, alpha = 0.8) +
  scale_color_manual(values = c("Concussed" = "lightsteelblue1", "Control" = "
   hotpink1")) +
   geom_boxplot(aes(fill = Group),
                width = 0.1,
                position = position_dodge(0.7),
                outlier.shape = NA,
                alpha = 0.8) +
  scale_fill_manual(values = c("Concussed" = "lightskyblue", "Control" = "pink1"))
   +
  labs(title = "Reaction Times in Concussed vs. Control Athletes",
       subtitle = "Comparison over Initial Assessment, Week 1, and Week 2",
       x = "Time Point",
       y = "Reaction Time (ms)",
       fill = "Group",
       color = "Group") +
  theme_minimal() +
  theme(legend.position = "bottom")

print(p_final)
```

Figure 4 ■ Layered violin and Box plot, with added jittered dots, comparing reaction times between the two groups over 3 time points (top left), combined violin and Box plot, with the violin plot layered over the Box plot (top right) and layered violin and Box plot, with added jittered dots, where the overlayered jittered dots are too opaque (bottom).

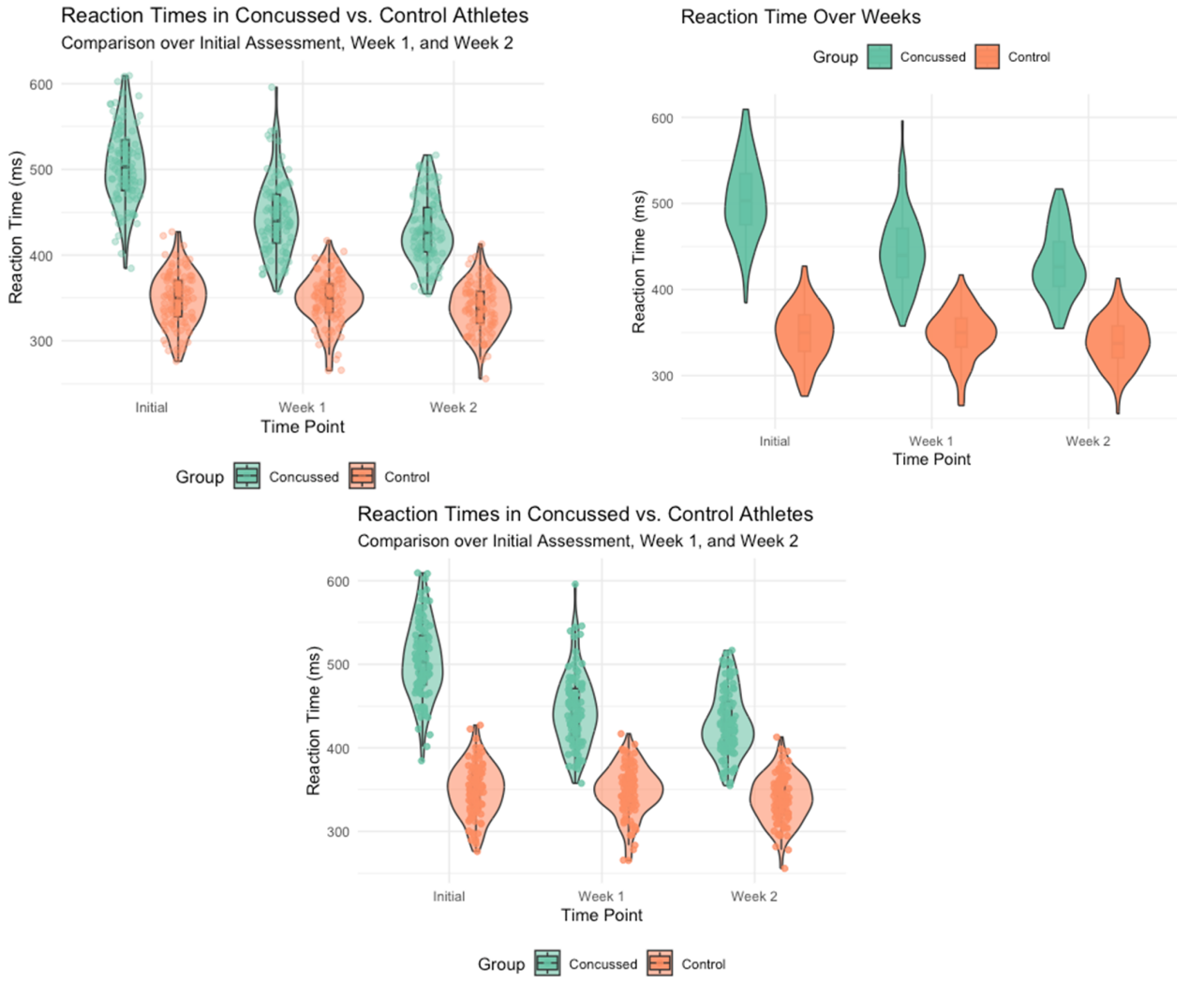


Figure 5 ■ Combined violin and Box plots, using `accent` color palette (left) with added jittered dots, with changed colors (right)

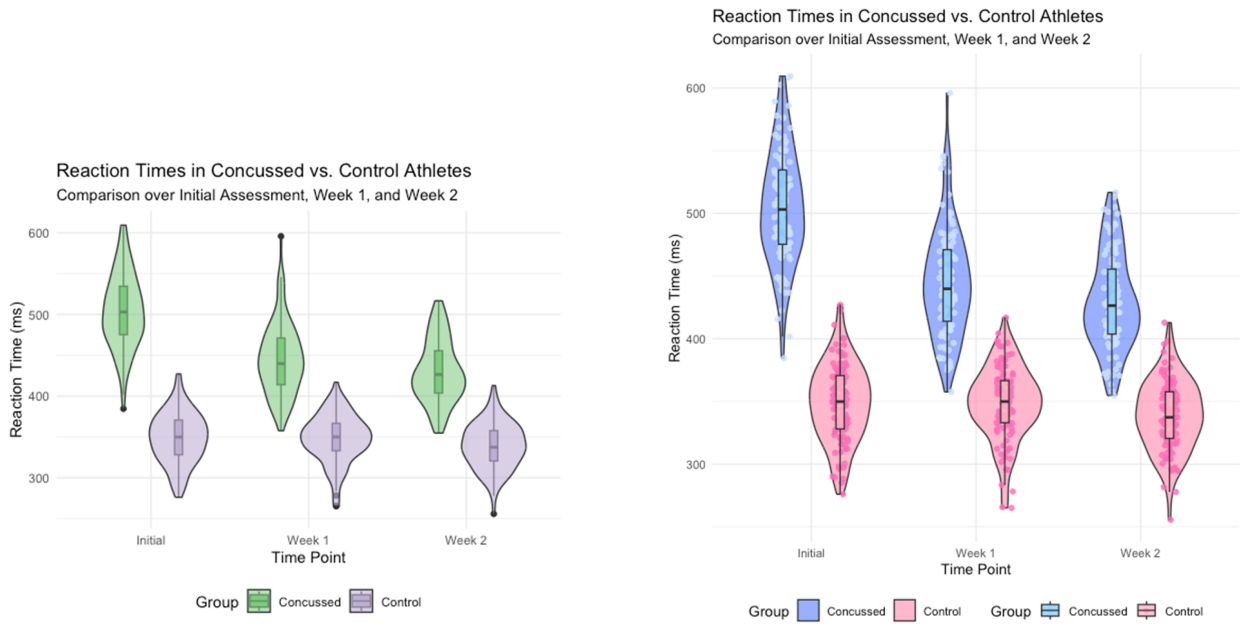
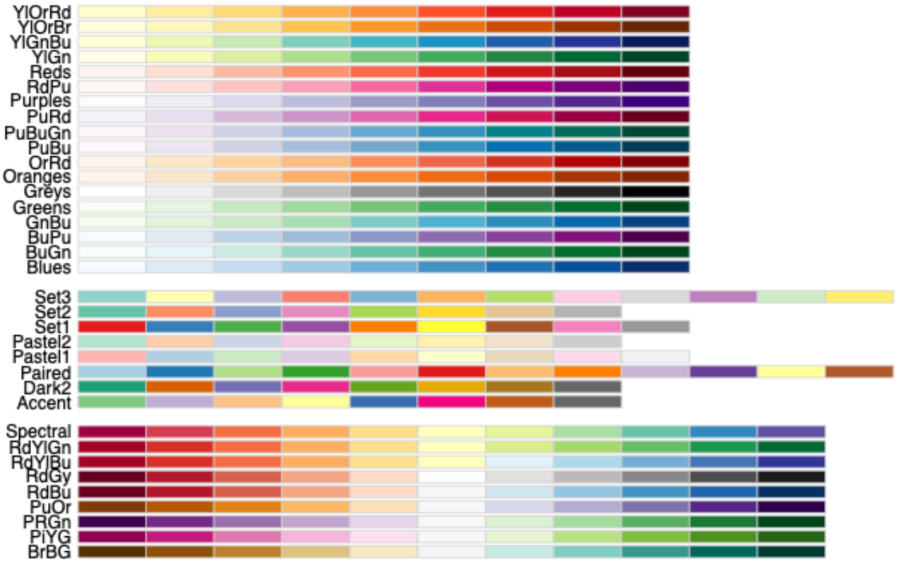


Figure 6 ■ Color palettes available in `ColorBrewer` package.



Appendix A: Code used to generate the randomized data.

This code generates simulated response times (RTs) in two groups, each measured three times.

```
library(ggplot2)
library(dplyr)
library(tidyr)

set.seed(123)  # Ensures reproducibility of the random data
data <- data.frame(
  Group = rep(c("Concussed", "Control"), each = 300),
  Time = rep(rep(c("Initial", "Week 1", "Week 2"), each = 100), 2),
  ReactionTime = c(
    rnorm(100, 500, 50), rnorm(100, 450, 45), rnorm(100, 425, 40),  # Concussed
    rnorm(100, 350, 30), rnorm(100, 345, 30), rnorm(100, 340, 30)   # Control
  )
)
```

The following is identical except that it adds identifiers to the participants, in case subsequent analyses would require them.

```
library(ggplot2)
library(dplyr)
library(tidyr)

set.seed(123)  # Ensures reproducibility of the random data
data <- data.frame(
  ID = c(rep(1:100, length.out=300), rep(101:200, length.out=300) ),
  Group = rep(c("Concussed", "Control"), each = 300),
  Time = rep(rep(c("Initial", "Week 1", "Week 2"), each = 100), 2),
  ReactionTime = c(
    rnorm(100, 500, 50), rnorm(100, 450, 45), rnorm(100, 425, 40),  # Concussed
    rnorm(100, 350, 30), rnorm(100, 345, 30), rnorm(100, 340, 30)   # Control
  )
)
```

Citation

Kanaan, C.-M., & Bellavance, J. (2025). Creation of a combined violin and box plot using R. *The Quantitative Methods for Psychology*, *21*(2), 32–42. doi: 10.20982/tqmp.21.2.p032.

Received: 09/12/2024 ~ Accepted: 20/02/2025