R project for statistical computing - Animint2

Google Summer of Code 2025

Project Title - “Advancing Animint2 - Testing, Rendering and Gallery port”

Project Idea Page :
https://github.com/rstats-gsoc/gsoc2025/wiki/Animated-interactive-ggplots

About Me :

Name : Suhaani Agarwal
County : India
Time Zone : IST (UTC + 5:30)
Language : English

I am currently in my first year of B.Tech in Computer Science and Artificial Intelligence at Indira Gandhi Delhi Technical University for Women (IGDTUW). With an SGPA of 9.43 in my first semester, I have built a strong academic foundation and developed a keen interest in data visualization, open-source development, and web technologies.

As part of my curriculum, I was introduced to R programming early on and have since actively explored its applications in data analysis and visualization. My experience with Animint2 includes reading its manual, exploring its existing galleries, and understanding its core differences from tools like gganimate, plotly, and htmlwidgets. This has given me a deep understanding of how Animint2 facilitates rich, layered, and linked interactivity. Additionally, I have worked on designing sample tests for Animint2 and have a solid grasp of its testing framework using Chromote, ensuring smooth and reliable functionality.

I possess strong proficiency in HTML, CSS, JavaScript, and Git/GitHub, enabling robust implementation of interactive features and effective project maintenance. My experience with JavaScript libraries like tippy.js, TailwindCSS, and PixiJS enhances UI development and rendering performance. I'm skilled in data.table

for efficient data manipulation in Animint2 and GitHub Actions for CI/CD workflow automation. My knowledge of package dependencies and core DSA principles allows me to optimize rendering processes, resolve technical issues, and streamline development workflows.

Beyond technical expertise, I have a strong track record in competitive development, having won multiple hackathons where I applied skills like JavaScript, debugging, and optimizing interactive visualizations. My experience in testing, debugging, and developing dynamic web components will help improve Animint2's interactivity.

With my technical skills, I am confident that I can contribute meaningfully to Animint2 by optimizing its functionality, improving user experience, and strengthening its long-term sustainability.

Contact Information :

Postal Address : H130 3rd floor, Shivaji Park, West Punjabi Bagh, Delhi-110026
Telephone : +91 8800348580
Email : suhaani0707@gmail.com
Website : suhaani_agarwal.co.in
Communication Channels : Video Conferencing (Google Meet , Zoom Meet) , Email , Call
Links :

- Linked In Profile
- Github Profile

Contributor Affiliation :

Institute : Indira Gandhi Delhi Technical University for Women (IGDTUW)
Program : B-Tech in Computer Science and Artificial Intelligence
Stage of completion : 1st year (currently in 2nd semester)
Contact to verify : Ms. Kiran Malik - +91 99111 15455

Availability :

My end-semester exams fall in the 2nd and 3rd week of May, but since my preparation is nearly complete, I will only need about 1 hour per day for revision, which will not impact my work during the community bonding period (May 8 – June 1).
Once the coding period officially begins on June 2, I will have no other commitments and will be fully dedicated, contributing **40–50 hours per week.**

Mentors :

Evaluating Mentor
Name : Toby Dylan Hocking
Email : toby.hocking@r-project.org

Co-Mentor
Name : Yufan Fei
Email : yufanfei8@gmail.com

I have been in contact with the evaluating mentor, Sir Toby Dylan Hocking, since submitting test solutions for the easy, medium, medium-hard, and hard Animint2 GSoC application tests. After sharing my initial solutions, I received valuable feedback and made necessary improvements based on his guidance. For the hard test, I implemented a test-renderer for the Newton-Raphson method and revised it to include proper interaction testing, as he suggested.
I also reached out regarding issues in the Animint2 repository, such as outdated dependencies of plyr and tooltip functionality, and have been actively working on resolving them under his guidance. Throughout this process, I have regularly communicated with him, discussing improvements, test strategies, and implementation details. Additionally, I was invited to the Animint organization as a member, which has allowed me to contribute more efficiently.
More details about my test solutions and other contributions are mentioned later in the proposal.

Project Details

Introduction

The goal of this GSoC project is to enhance Animint2 by implementing new features, improving visualization layouts, porting and refining the gallery, fixing key rendering issues, and optimizing the package's interactivity.

Features to be implemented :

1) Layout ggplots in a HTML table:

Analysis of issue #115 and PR#153:

The fundamental challenge is implementing a reliable grid layout system for ggplots in animint2, where plots can be positioned according to user specifications rather than just rendering sequentially. After examining PR #153 and related discussions, I've understood that this solution involves:

Three Key Attributes:

rowspan: Vertical span of a plot (default=1), colspan: Horizontal span of a plot (default=1) and last_in_row: Boolean flag to end current table row

HTML Table Structure:

1. Outer table created when layout attributes are present
2. Simple rules for row/column management:
3. New <tr> before first plot and after plots with last_in_row=TRUE
4. Each plot gets a <td> in current row
5. No automatic grid size calculations

Building on the Existing Foundation:

The current PR provides an excellent starting point for implementing table-based plot layouts. While the core structure is in place, we can enhance it to create a more robust and polished implementation. My goal is to build upon this foundation to develop a complete solution that:

- Accurately renders plots in the specified table/grid format

- Properly implements all spanning behaviors
- Maintains clean visual separation between elements

Proposed Implementation Approach:

- R Side Implementation (z_animintHelpers.R)

I'll extend the theme attribute handling to properly extract and validate the layout parameters. The R code will process the theme_animint() specifications and pass them to the compiler. This includes:

- Validating that rowspan/colspan are positive integers
- Converting last_in_row to a boolean flag
- Ensuring backward compatibility with existing plots

- JavaScript Implementation (animint.js)

The core rendering logic will be enhanced with a lightweight table manager that:

- Creates the outer table structure when layout attributes are present
- Tracks current row and column positions during rendering
- Properly applies rowspan/colspan attributes to table cells
- Automatically handles row breaks based on either Explicit last_in_row markers , Column span requirements or Rowspan commitments from previous rows

- The table rendering function may follow this pattern: (only an idea of how the code can look like)

```
function renderTableLayout(plots) {
  let currentRow = null;
  let columnsUsed = 0;
  for (let i = 0; i < plots.length; i++) {
    if (!currentRow || columnsUsed >= MAX_COLUMNS) {
      currentRow = startNewRow();
      columnsUsed = 0;
    }
    const plot = plots[i];
    const cell = createTableCell(currentRow, plot);
    renderPlotContent(cell, plot);

    columnsUsed += plot.colspan || 1;
    if (plot.last_in_row) columnsUsed = MAX_COLUMNS;
  }
}
```

- Visual Structure - The implementation will ensure:

- Proper cell padding and spacing
- Contained plot elements (including legends)
- Clean borders between cells

---

2) Gallery Porting and Enhancement

I have thoroughly examined both the old and new Animint2 galleries, comparing the visualizations in each and identifying those that have not yet been ported. While doing so, I have taken into account the work done by Nhi Truong in GSoC 2024, as well as the issues she raised and the PRs she submitted—including one that has not been merged yet. I have studied these issues in detail and have compiled a structured list of visualizations that still need to be ported.

To streamline the porting process, I have categorized my findings as follows:

1) Visualizations in the Old Gallery That Have Not Yet Been Ported:

> **Visualizations with missing files:** 12, 25, 26 (not to be ported)
> **Requires code updates (from animint to animint2):** 27 (Temperatures in Montreal office and outside)
>
> **Visualizations with raised issues (needing code updates):**
>
> > 22 – AB lines out of range (requires fixing Issue #142 first)
> >
> > 6 – Gradient descent for regression (issue needs to be resolved)
> > 34 – Related issues documented in PeakSegFPOP-paper
>
> **Similar visualizations that can be ported together:** (issues)
> > 9, 18 – Resolve raised issues and port together
> > 10, 21 – Similar structure, can be ported in one step
> > 4, 5 – Gradient descent for neural network and linear model (related)
>
> **Other pending visualizations to be ported:** 35, 51, 52, 53, 56

2) Visualizations in the Unmerged PR (PR #14)

This PR includes visualizations that have already been created but require issue resolution before being merged:

30 – Cost/slack minimization over all feature thresholds for max margin interval tree models
31 – Demo of Constrained Pruned Dynamic Programming Algorithm
36 – Demo of constrained Pruned DPA
41 – WorldBank data viz for Animint paper
42 – Max margin interval regression for supervised segmentation model selection
43 – Montreal Bikes
45 – WorldBank facets + select multiple countries
55 – Temperature Maps
58 – Tornado select multiple states

Enhancements to the Newly Ported Visualizations

I have taken detailed notes on my understanding of which visualizations have already been ported, which ones are similar to each other, and which were created by Nhi but not yet integrated due to unresolved errors. I plan to resolve these issues and port the missing visualizations accordingly.

Additionally, many visualizations do not have **demonstration videos** showing their typical interactions. I will create these videos in a comprehensive and explanatory manner. I will also add videos for the new visualizations that I port to ensure clarity in their functionality.

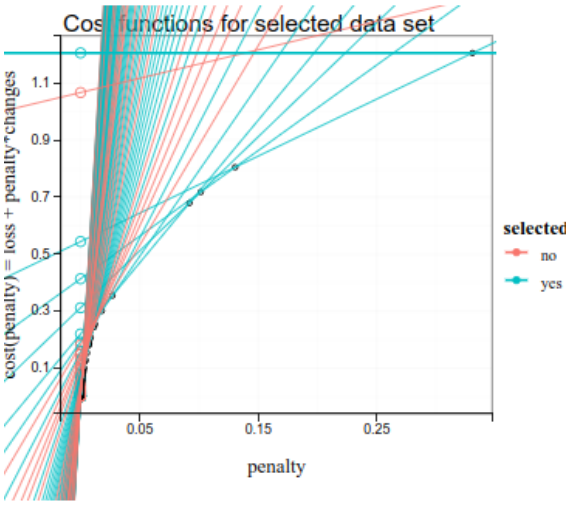
Since the older visualizations do not contain guided tours and interactive tooltips, I will integrate these features into all the new visualizations I port to enhance usability and interactivity. Furthermore, my proficiency in creating new visualizations is demonstrated by my easy and medium test solutions, and my understanding of how the gallery works is evident from my medium-hard test solution.

---

3) Tests and bugfix for geom_abline

The core problem occurs when ablines extend beyond the visible plot area in animint2 visualizations, particularly noticeable in the changepoint model selection example. The lines appear unbounded, continuing indefinitely outside the plot's axes limits. This

creates visual clutter and makes the plot difficult to interpret.



Problem Analysis of geom_abline Implementation

After carefully examining the changepoint model selection visualization (particularly the cost functions plot in the bottom-right panel), I've identified a fundamental limitation in how geom_abline currently handles line rendering. The HTML output reveals lines with extreme coordinates (e.g., x2="4945.74" when the visible area is much smaller), confirming that ablines are being drawn as infinite lines without regard to plot boundaries.

Core Problems Identified:

Unbounded Line Rendering:

- Currently takes slope and intercept but no boundary constraints
- Uses fixed extreme x-values (-167 to 4945 in the example) regardless of actual plot range
- Creates lines that extend far beyond visible area

Mathematical Oversights:

- Doesn't calculate intersections with plot boundaries
- Fails to handle edge cases (vertical/near-vertical lines)
- No consideration for dynamic plot ranges in interactive visualizations

Visual Consequences:

- Cluttered appearance as lines shoot off to infinity
- Particularly problematic with steep slopes (common in cost functions)

- Wasted rendering resources on invisible line segments

Proposed Solution Approach

1. Mathematical Line Clipping:

I plan to implement a boundary-aware rendering system that:

- Takes current x/y ranges from the plot
- Calculates exact intersection points with all four plot edges
- Returns only the visible segment of each line
- Handles special cases mathematically:

```
# Pseudo-code for boundary calculation
calculate_visible_segment <- function(slope, intercept, x_range, y_range) {
  # Find intersections with left/right/top/bottom boundaries
  # Return only the segment that falls within the plot area
  # Special handling for vertical/horizontal cases
}
```

2. Enhanced Rendering Pipeline:

The geom_abline implementation will be modified to:

- Accept plot range information from the coord system
- Process lines through the clipping algorithm
- Only render the visible portions
- Maintain all existing aesthetics and interactive features

3. I will implement comprehensive renderer tests that validate geom_abline behavior across edge cases (including vertical lines and dynamic plot ranges), with specific focus on the changepoint model visualization that revealed the original issue.

---

4) Move scale range calculation from compiler to the renderer

My Understanding of the Issue:

The core problem lies in how scale ranges (axis domains) are currently computed in animint2 when users interact with plots (via clickSelects or showSelected). Here's the breakdown:

Current System (Problematic Workflow)

- Compiler Pre-Computes Scale Ranges (R Side)

- The compiler side tries to predict all possible data subsets that could arise from user interactions.
- Uses compute_domains() (in z_animint.R) to calculate min/max for each possible subset.
- Stores these pre-computed ranges in plot.json.

- Limitations:

- Cannot anticipate all possible combinations of user selections (combinatorial explosion).
- Only works for predefined subsets, not dynamic filtering.
- Static snapshots of data (built_data) mean some ranges are incorrect or missing.

- Renderer Just Applies Pre-Computed Ranges (JS Side)

- The JS update_scales() function in animint.js simply reads the pre-computed ranges from plot.json.
- Problem: If a user interaction creates a new subset that wasn't pre-computed, the scales don't adjust properly.

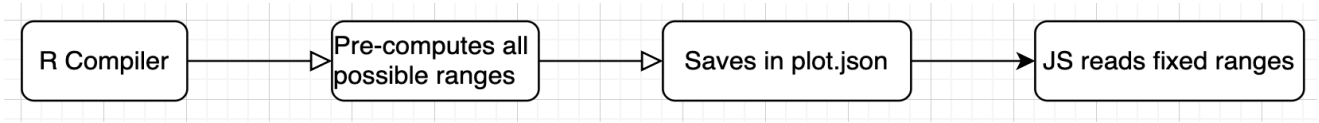
- Why Move to Renderer?

- The renderer already has all the data (via TSV files).
- It can dynamically compute the best scale for any user selection.
- Avoids the combinatorial explosion problem in the R compiler.
- More accurate "zoom-to-fit" behavior for any subset.

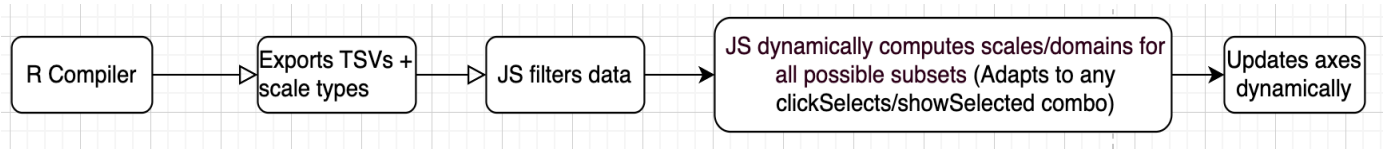
Proposed Implementation Approach:

1. Workflow (Moving Logic to JS Renderer)

Before:



After:



2. Key Files to Modify + What functions will be written?

Compiler Side (R)

- z_animint.R

- Remove: compute_domains() and get_domain() (no longer pre-compute ranges).
- Modify animint2dir() to:
    - Export axis metadata (variable names, scale types, transformations).
    - Preserve panel/facet layouts (for multi-plot support).
    - Skip writing axis_domains to plot.json.

- helper-plot-data.R

    - Retain data export logic (TSV files for each geom).
    - Ensure all necessary columns (e.g., x, y, PANEL) are exported.

Renderer Side (JavaScript)

- animint.js

- Refactor update_scales(): Replace pre-computed range lookup with dynamic computation and Use D3's d3.extent() for min/max calculations.
- New Helper Methods: getDomainsForGeom() – Computes ranges per geom and aggregateDomains() – Merges ranges across geoms.
- New JS Class: ScaleManager : will handle per-plot scale state including two methods: init() – Sets up scales from plot.json and update() – Recalculates domains on interaction.

Additional Features:

5) Smart Box and Label Placement for Animint2 Using Quadratic Programming

My understanding of Issue #174

Right now, when we add labeled boxes to plots in Animint2 (like annotation boxes at the end of lines in a line chart), they don't align neatly and often overlap. The current approach uses basic positioning rules that fail when multiple boxes need to be placed close together. In R, we have a better solution using quadratic programming (through the quadprog package) that carefully calculates optimal box positions to maintain alignment and prevent overlaps, but this capability is missing in Animint2's JavaScript implementation.

My Implementation Plan:

Add the QP Solver:

- I'll integrate the albertosantini/quadprog JavaScript library (a well-maintained port of R's quadprog)
- This will handle the mathematical optimization for box placement

Create New Geom: geom_aligned_boxes:

- This will work with aesthetics like:
- ```
  aes(x, y, label, color, fill,
      alignment = "horizontal"/"vertical",
      position = "top"/"bottom"/"left"/"right")
  ```
- It will create properly aligned boxes containing labels

Smart Placement System:

- Before rendering boxes, the system will:
  - Calculate each box's exact dimensions (based on text content and styling)
  - Use quadratic programming to determine optimal positions where:
    - Boxes stay perfectly aligned in columns/rows
    - No boxes overlap
    - Each box maintains proper connection to its data point
    - All boxes stay within plot boundaries
  ```

- The placement will automatically adjust during user interactions like zooming

Testing:

I'll create test cases showing:

- Basic alignment (verifying boxes stay in perfect columns/rows)
- Overlap prevention (even with many crowded boxes) by quadprog.js
- Dynamic adjustment during interactions

---

6) Code coverage on github actions (#122)

Problem: Animint2 lacks code coverage metrics for its R and JavaScript components, making it hard to track tested code and ensure reliability.

Implementation Approach

For the R code, I'll integrate the covr package with our GitHub Actions workflow to generate detailed coverage reports. This involves proper configuration to handle our package structure while excluding deprecated code. For JavaScript, as discussed with the mentor, I'll implement coverage tracking using Chrome's native V8 coverage collection through Chromote, combined with puppeteer-to-istanbul for converting the coverage data to Istanbul format. The solution will include modifying our test infrastructure to collect coverage data from browser-executed code and combining it with R coverage metrics. I'll set up unified reporting through Codecov.io to give developers a clear picture of overall test coverage.

Expected Impact: Implementing coverage tracking will improve code quality, prevent regressions, and help maintain robust visualizations.

Detailed Timeline:

Community Bonding Period (May 8 - June 1)

Goals: Deep dive into implementation plans, start early coding (gallery ports + geom_abline fix), and align with mentors.

Week 1 (May 8 - May 14)

Research:

- ➢ Test PR #153 (HTML table layout) locally, identify edge cases (e.g., overlapping spans).
- ➢ Study geom_abline bug (#142) and clipping algorithms.

Coding:

- ➢ Port 2-3 gallery plots (eg. 41, 55).
- ➢ Record demo videos for some ported visualizations.

Week 2 (May 15 - May 21)

Goal: Port 4-5 visualizations from the end of the list :

- ➢ Target: #51, #52, #53, #56 (basic plots)
- ➢ Tasks:
  - Update syntax from animint1 → animint2 (wherever applicable)
  - Verify interactions (clickSelects/hover)
  - Record demonstration videos

Week 3 (May 22 - May 28) - geom_abline Bugfix Implementation

Research & Debugging

- ➢ Analyze geom_abline.R to identify why lines extend beyond plot bounds
- ➢ Reproduce issue in test environment

Implement Fix

- ➢ Modify geom_abline.R to enforce plot boundaries (xlim/ylim)
- ➢ Handle edge cases (vertical/near-vertical lines)

Testing

- ➢ Write new renderer tests for bounded lines
- ➢ Verify interactive features still work

Port Visualization #22 (based on geom_abline issue)

- ➢ Apply fix to AB lines visualization
- ➢ Confirm proper clipping

Week 4 (May 29 - June 1)

Priority Ports:

- ➢ #27 (Montreal temps): Modernize animint1 syntax
- ➢ #34 (PeakSegFPOP): Test with new line rendering
- ➢ Record demo videos for ports.

Phase 2 Preparation

- ➢ Finalize implementation plans with mentors for:
  - HTML table layouts
  - Scale computation migration
  - Code coverage
  - QP based alignment
  - Any additional features/improvements requested by mentors
- ➢ Create detailed technical roadmap for coding phase after testing and researching on every feature.

Phase 1: Core Features (June 2 - July 18)

Midterm Deliverables:

- ➢ HTML Table Layout: Full implementation with renderer tests.
- ➢ geom_abline Fix (completed in bonding period)
- ➢ Gallery: 15+ ports with videos.
- ➢ Code Coverage: Live on Codecov.

Week 5-6 (June 2 - June 13) – Code Coverage Implementation

- ➢ R Coverage :
  - Configure covr for core R functions
  - Set up GitHub Actions workflow
  - Exclude deprecated code from coverage
- ➢ JS Coverage :

        - Implement V8 coverage collection via Chromote
        - Convert to Istanbul format using puppeteer-to-istanbul
    - ➢ Integrate with Codecov.io
    - ➢ Document setup in project wiki

Week 7-9 (June 14 - July 4) – HTML Table Layout

Implement responsive grid system using:

- ➢ User-defined rowspan/colspan
- ➢ Automatic row management (last_in_row)
- ➢ Aspect ratio preservation
- ➢ Handle edge cases

Post-implementation:

- ➢ Create renderer tests (normal and edge cases)
- ➢ Write usage documentation

Week 9-11 (June 28 - July 18) – Finalization

Wrap-up:

- ➢ Complete any pending table layout fixes / code coverage integration
- ➢ Port 2-3 showcase visualizations
- ➢ Finalize all tests

Documentation:

- ➢ Wiki pages for:
    - Table layout syntax
    - Codecov integration guide

Demo videos for new features

Mentor Requests:

- ➢ Address any additional features/fixes identified during review
- ➢ Optimize performance if needed
- ➢ Polish UI/UX elements

Phase 2: Advanced Features (July 19 - August 25)

Final Deliverables:

- ➢ Scale Computation Migration: Move range logic from R compiler to JS renderer.
- ➢ Smart Box Placement: QP-based alignment.
- ➢ Gallery Completion: All remaining plots ported and gallery finalization.

Week 12-14 (July 19 - Aug 8) – Scale Migration

- ➢ R Modifications: Remove compute_domains(), export scale metadata (e.g., variable names, log transforms).
- ➢ JS Side: Refactor update_scales() to dynamically compute ranges.

Week 15-16 (Aug 9 - Aug 22) – Smart Box Placement

- ➢ Integrate quadprog-js for optimization.
- ➢ Implement geom_aligned_boxes with no overlaps and Alignment in columns/rows.

Week 17 (Aug 23 - Aug 25) – Gallery Finalization

- ➢ Port remaining plots.
- ➢ Finalize all demo videos.

Final Week (Aug 25 - Sept 1)

- ➢ Polish: Fix edge cases, ensure tests cover all features.
- ➢ Documentation:
- ➢ Add examples for new features (HTML layout, box placement)
- ➢ Update wiki with coverage guidelines and gallery usage.
- ➢ Submit: Final PRs and evaluations.

Code Quality Assurance & Testing Approach

I will implement a rigorous testing framework. For every feature implemented, I will write comprehensive tests that validate both the core functionality and edge cases. All renderer tests will simulate real user interactions including click events (using clickID functionality) and hover behaviors where required, testing the before and after HTML of the simulated behaviour. I will also create detailed documentation for every new feature implemented, explaining its usage, parameters, and examples.

Commit Strategy & Development Discipline

I plan to maintain a high-frequency commit schedule to ensure transparency and incremental progress. For major features, I expect 10-15 atomic commits, broken down into implementation, testing, and refinement phases. Smaller fixes will require 3-5 commits. My goal is 1-2 meaningful commits per day, each representing a logical, self-contained improvement (e.g., "Added rowspan support in HTML table renderer").

Progress Tracking & Mentor Communication

I will maintain extremely transparent communication through multiple channels:

- ➢ A dedicated PR titled "Suhaani's GSoC Progress" where I'll post **daily updates** including:
  - Completed tasks with commit links
  - Current challenges and solutions tried
  - Next steps and implementation plans
  - Any questions needing mentor input
  - Daily progress summary messages
- ➢ Immediate availability for Google Meet calls whenever mentors want to discuss implementation approaches, review designs, or troubleshoot issues.

Contingency Plan for Risk Mitigation

To handle unforeseen challenges, I will:

- ➢ Maintain a backup branch with stable code at all times
- ➢ Flag issues early-if a task takes >1 day without progress, I will immediately seek guidance
- ➢ Prioritize flexibly-if deadlines are at risk, I will defer non-core features (with mentor approval)

Tests and contributions

The qualification tests submitted : (tests link) , (solutions link)

1) Easy test :
Solution:
Diamond Dataset Visualization: Github Pages Link
Source Code: easy_test_solution/main.R

2) Medium test :
Solutions:

1. Newton Raphson Root Demonstration: Github Pages Link
   Source Code: medium_test_newtons_method_code/newtons_root.R

2. Bisection Method of Root Finding: Github Pages Link
   Code Link: medium_test_solution/main.R

3. Demonstration of Central Limit Theorem: Github Pages Link
   Code Link: medium_test_solution.R

4. Demonstration of Central Limit Theorem (Exponential Distribution Example): Github Pages Link
   Code Link: medium_test_solution.R

3) Medium-Hard Test Solution:
Gallery Link: suhaani-animint-gallery
Code Link: suhaani-anaimint-gallery/index.Rmd

4) Hard Test Solution:
PR Link : https://github.com/animint/animint2/pull/190
Video link : Demonstration Video

Contributions to Animint2:

I have contributed to animint2 through two significant pull requests that demonstrate my understanding of the codebase:

- ➢ Replacing plyr occurrences with data.table (#188)
  - Resolving issue #165 , replacing all plyr dependencies with data.table
  - Resolved test failures.
  - Currently finalizing remaining issues in helper function (cdata) through mentor discussions
- ➢ Tooltip Implementation with Tippy.js (#191)
  - Replaced SVG title tooltips with modern Tippy.js implementation
  - Integrated Tippy.js into animint2's rendering pipeline
  - Updated existing tooltip tests for new implementation
  - Developed innovative hover simulation tests using Chromote's automation capabilities

- Created comprehensive tests validating tooltip content and interactive behaviors
- Currently working on resolving the Chromote timeout error in CI tests.

After GSoC:

Beyond GSoC, I am deeply invested in animint2's growth and plan to remain an active contributor to the project. My work during the summer will serve as a foundation for long-term involvement—whether through refining the features I implement, addressing future issues, or mentoring new contributors. The blend of R, JavaScript, and interactive visualization in animint2 aligns perfectly with my interests, and I am excited to continue collaborating with the community to push the project's boundaries. I envision contributing not just code but also documentation, tutorials, and outreach to help expand animint2's adoption. This isn't just a summer project for me; it's the beginning of a sustained commitment to a tool I'm truly passionate about.