US Crime Analysis Tool

Complete System Documentation

Haitao Zhao

October 13, 2025

Contents

1 Part 1: User Manual

1.1 Stakeholder Group Description

State-level law enforcement officers, policy makers, criminal justice researchers and public safety analysts form the primary users of this tool. These end-users typically need to extract patterns from violent crime data at state levels for resource allocations on high-risk areas regarding any interventions that have been proposed or implemented. A commander within a law enforcement agency uses such tools in justifying budget requests and allocation of personnel while a policy analyst utilizes the same system in drafting legislation aimed at specific categories of crimes based on quick manipulations over jurisdictions by academic researchers who conduct comparative studies across different regions using urban planners together with community safety boards as beneficiaries when developing neighborhood improvement programs or evaluating impacts due process.

1.2 Software Objectives

This tool quickly navigates high-dimensional crime data and transformations of perspectives paying consideration to the type of data used in analysis, and report generation toward various stakeholders. It removes manual calculations by automating an instant statistical analysis on four crime variables: murders, assaults, rapes per 100,000 inhabitants; percentage classified as urban population; all arrests per 100,000 residents. Results can equally be viewed as continuously measured data or transformed into categorical risk levels depending on the context of analysis. The report has been specifically designed to standardize findings communicated to nontechnical audiences but leaves enough flexibility in exporting raw data for advanced statistical modeling outside this platform.

1.3 Software Usage and Practical Value

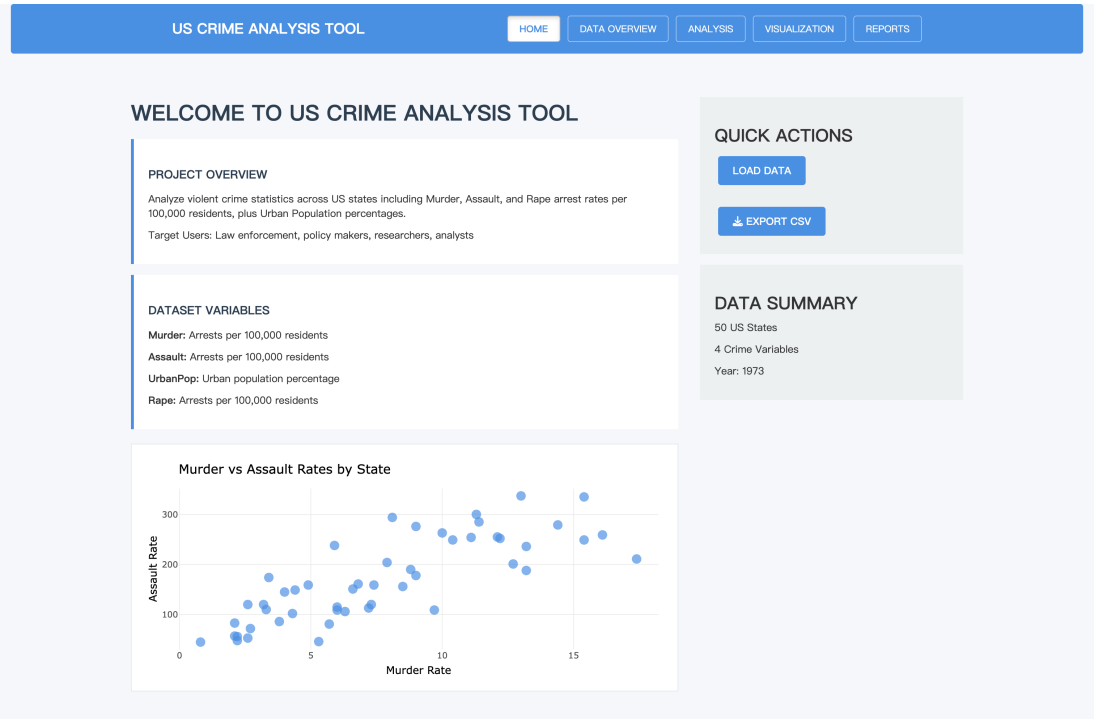

Figure 1: Main interface navigation bar

Five buttons sit in the top navigation bar. Clicking any button simply switches the view with no loss of data between sessions. The home tab displays the summary statistics and a preview scatter plot between murder and assault rates among the fifty states. This chart gives an overview using interactive tooltips - hovering over any point returns the state name, thus finding outliers without making the visualization messy.

On the left of the homepage is a quick actions bar. "LOAD DATA" initializes into memory the USArrests dataset. Technically, data are loaded on startup automatically, this button becomes meaningful after importing custom datasets or when resetting variables. "EXPORT CSV" creates and makes available for download a file with the present state of the dataset, inclusive of any modifications through data type conversions or filtering operations.

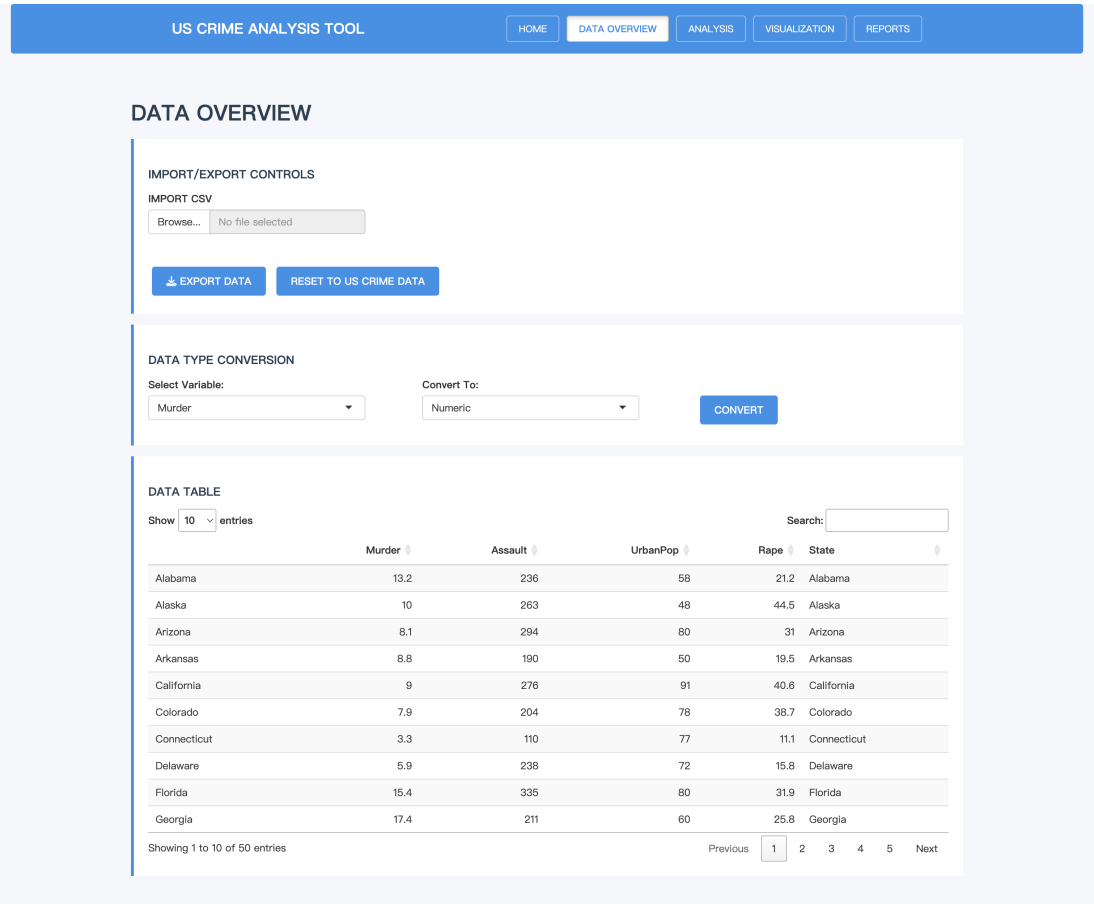

Figure 2: Data Overview section interface

The Data Overview section performs three tasks: file management, type conversion, and table viewing. The import control admits CSV files of similarly structured columns. After choosing a file through the browser dialog, it tries to load data and displays either a success notification or an error description in case parsing fails. This is necessary when an analyst wants to benchmark his jurisdiction's data against the national USArrests baseline or in the case where newer statistics from annual FBI reports have to be incorporated.

The export works both ways. Users can always download the current state of the dataset, because this state carries all transformations as a result through the interface. Therefore, it supports such collaborative workflow structures where one analyst prepares the structure of data and exports it to

his colleagues who need the same categorical breakdowns.

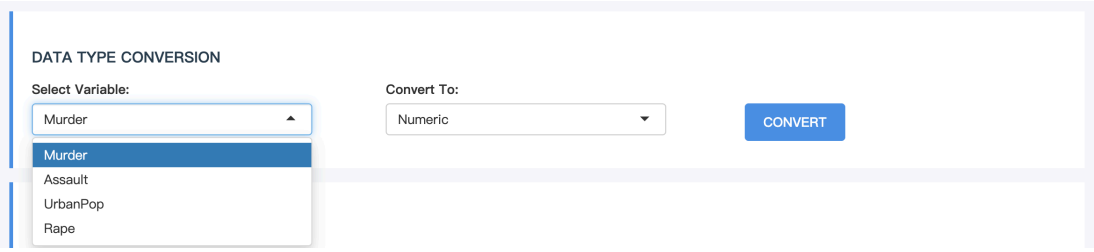

Figure 3: Data type conversion panel

The data type conversion panel changes the way variables appear in later analyses. Select "Murder" from the variable dropdown and select "Categorical" from the type dropdown, then click "CONVERT". This will break the murder rate into three bins, Low, Medium, and High based on the distribution of values. This changes the behavior of the variable in plots. Instead of a continuous scatter plot, categorical data create grouped bar charts or faceted comparisons.

Convert to Factor treats numeric values as discrete categories with no implied ordering useful for techniques such as one-way ANOVA in which each value becomes its own group. Convert to Numeric undoes any categorical transformation returning the original scale for use in correlation or regression analysis. The data table beneath the conversion controls updates instantly to reflect changes showing the current structure of all variables across the 50-state dataset.

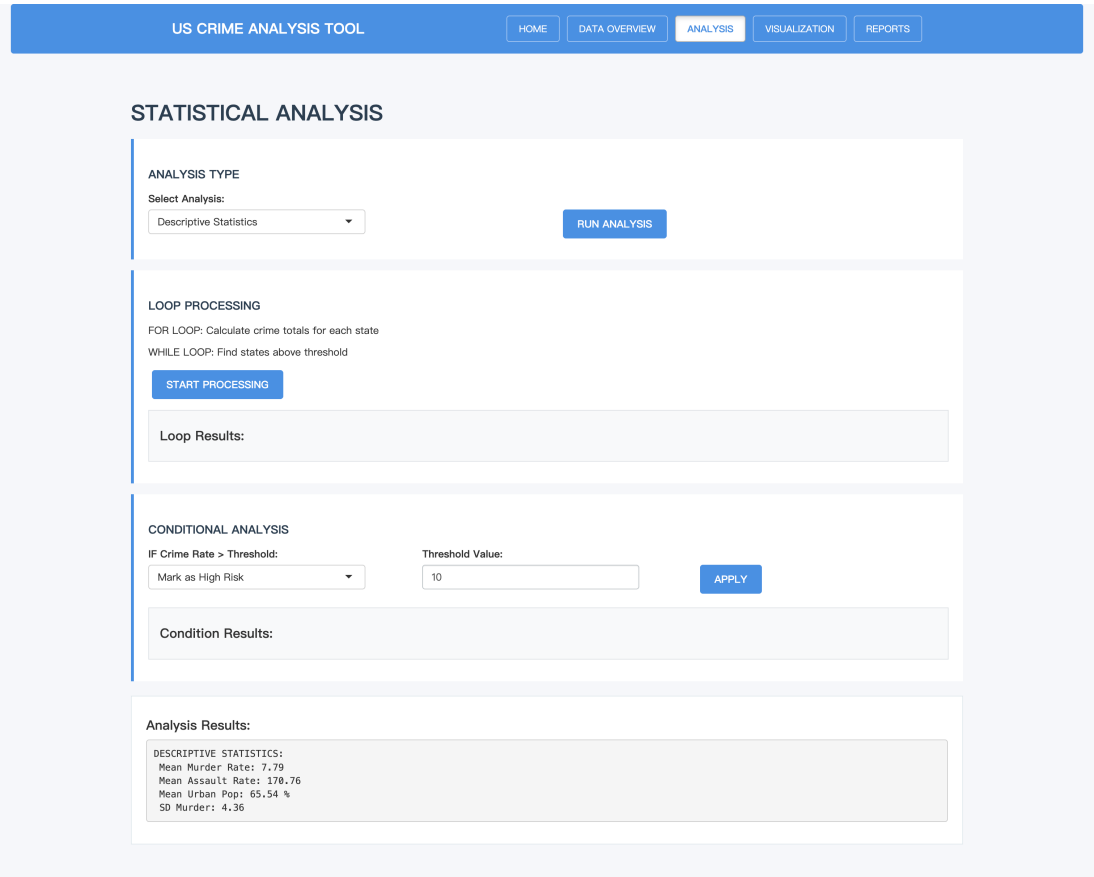

Figure 4: Analysis section interface

The Analysis section is split into three functional areas. The first dropdown provides for the selection of three different types of analyses. If "Descriptive Statistics" is selected and "RUN ANALYSIS" clicked, then means and standard deviations by variable are returned in a textbox below the controls, formatted as readable summary statistics rather than raw R output. Often in policy briefings, basic metrics such as mean values are required to establish baseline conditions before articulating trends.

Selecting "Correlation Analysis" computes Pearson correlation coefficients among all variable pairs. It displays results indicating the strength and direction of association- for example, if higher murder rates by state also report high assault rates or any category of violent crimes correlated with urban population percentage. These are the correlations that imply which crime types cluster geographically to inform resource allocation.

The "State Ranking" option displays a list of the top five states by murder rate. If the user changes the dataset by import or filter, this ranking will be updated automatically. Such rankings are used by law enforcement coalitions to determine peer jurisdictions for information sharing or to compare their own performance with similar states.

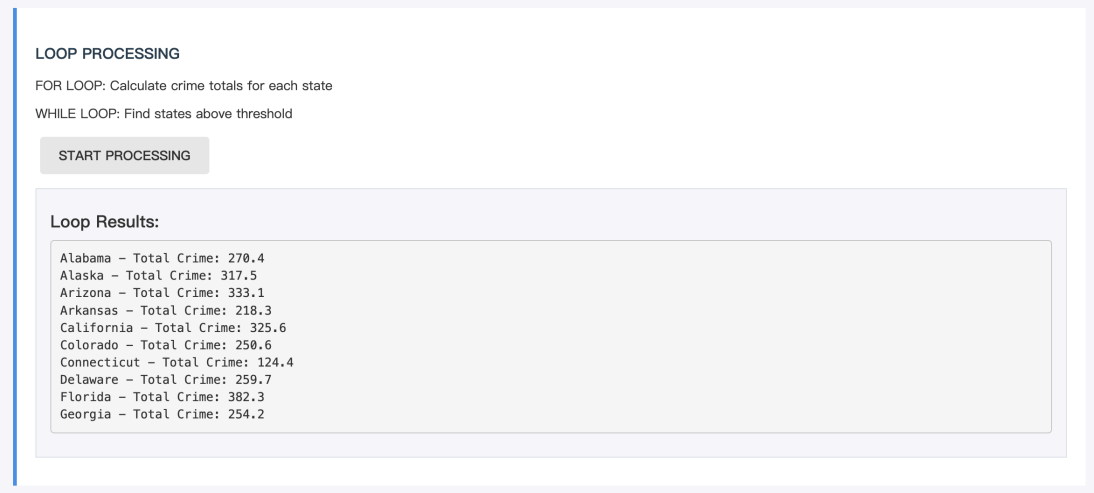

Figure 5: Loop and conditional processing section

The demo shows loop processing by calculating the total crime score as the sum of murder, assault, and rape arrest rates for all states. When "START PROCESSING" is clicked, a for-loop runs through the first ten states, calculates their aggregate crime metric, and displays the results in a text area. This looks redundant because R supports vectorized operations over objects but an explicit loop structure helps users understand logic computation when this code is adapted to a more complex scenario such as a weighted scoring system or multi-year trend calculation.

The conditional analysis section runs an if-then logic to classify states. Set a threshold value, take an action like "Mark as High Risk" and it flags all the states above that murder rate cutoff. The results box shows which states received the high-risk designation. This binary classification supports decision workflows where different response protocols activate based on severity thresholds - for example, triggering federal assistance requests when murder rates surpass a defined critical value.

The Visualization section provides the most direct route to insight communication. There are four dropdown menus: X-axis, Y-axis, color, and type of plot. Select "Murder" for the X-axis and "Assaults" for the Y-axis to produce a bivariate comparison. Change the color from blue to purple or teal- aesthetically pleasing however it does not change anything about how data is represented.

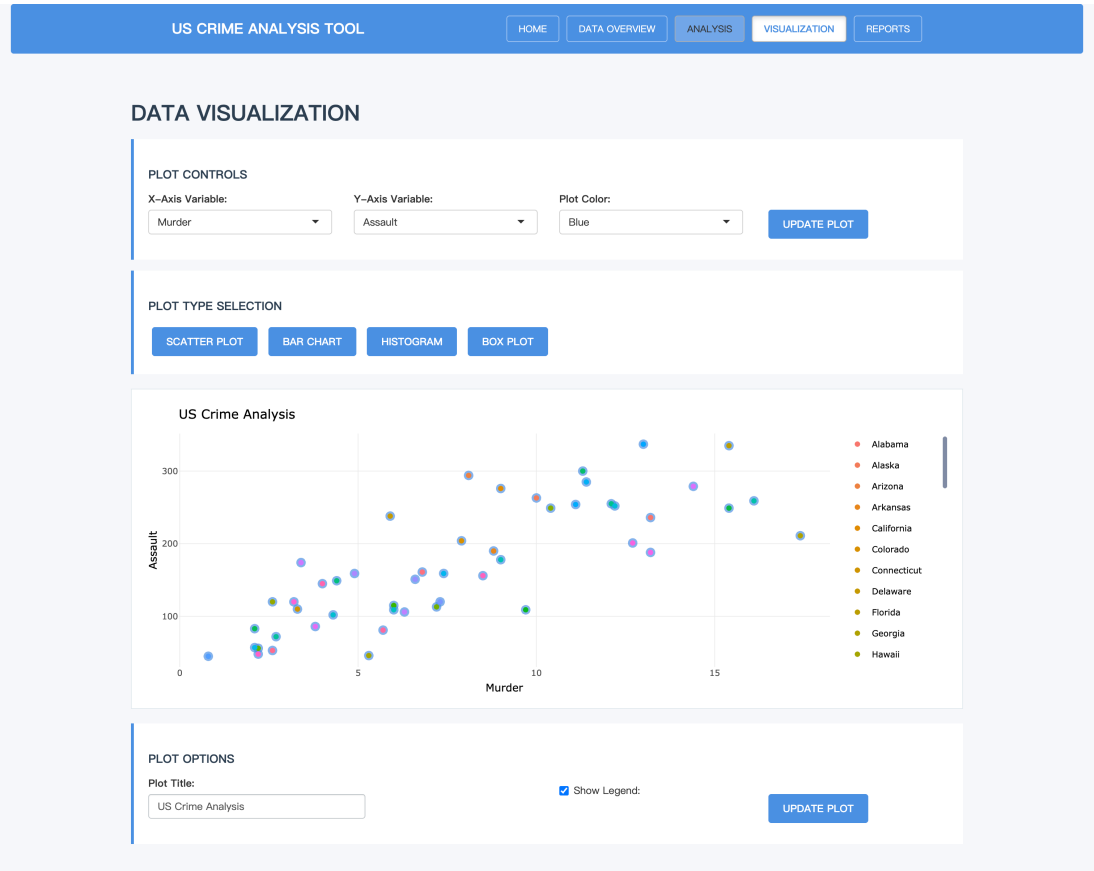

Figure 6: Visualization section main interface

The plot type buttons switch between four visualization modes. “SCATTER PLOT” shows individual states as points in two-dimensional space, useful for identifying correlation patterns and detecting outliers. States that deviate significantly from the overall trend become immediately visible - a state with high assault rates but low murder rates would appear in a specific region of the plot, prompting investigation into local policing practices or demographic factors.

“BAR CHART” mode displays the top ten states for the selected variable in descending order. This format works better for presentation slides or public reports where stakeholders need clear ranking information without the complexity of multivariate plots. The horizontal bar layout accommodates long state names without overlapping labels.

“HISTOGRAM” creates a frequency distribution showing how many states fall into each range of values for a single variable. This reveals whether crime rates cluster around a central value or spread across a wide spectrum. Histograms help distinguish between states that are statistically typical versus those requiring special attention due to extreme values.

“BOX PLOT” summarizes the distribution through quartiles, median, and outliers. The box spans the interquartile range while whiskers extend to minimum and maximum values excluding outliers. Outliers appear as individual points beyond the whiskers. This compact visualization communicates data spread and identifies anomalous states simultaneously, making it appropriate for technical audiences familiar with statistical graphics.

The plot options panel contains settings of pure presentational detail. Any string of text may be

Figure 7: Scatter plot visualization example

entered into the title field, such that the default "US Crime Analysis" can be replaced by a more contextual description, e.g., "2023 Q4 Violence Trends", "Pre-Intervention Baseline Comparison" etc. The legend toggle simply determines whether or not state names appear inside the legend box - handy when creating printed handouts where space constraints matter, but less critical for interactive digital presentations where tooltips provide state identification.

The Reports area brings findings together into standardized outputs. A report type dropdown presently offers three templates. "State Rankings Report" generates visualizations oriented toward ordered comparisons, either using a single crime variable or aggregated violent crime totals. The "Include Variables" dropdown simply controls which data columns appear in the output-selecting "Murder Only" produces a murder-specific ranking, while "Violent Crimes Only" sums murder, assault, and rape rates before ranking states.

"Regional Comparison" creates scatter plots with trend lines overlaid. It is useful for examining relationships between urbanization and crime rates across regions. The linear regression line will show if a state's urban population percentage predicts murder rates so as to provide evidence, pro or contra, of any theory that attempts to relate metropolitan density with violent crime.

"Crime Correlation Report" visualizes the relationship between two crime variables with both point clouds and fitted regression lines. Such reports are used by policy analysts when determining whether interventions targeted at one type of crime will have spillover effects on related categories of crime.

Click "GENERATE REPORT" to display the selected visualization in the preview area. The summary table beneath provides a reference sheet that can be interpreted by non-technical readers, listing the highest and lowest performing states for each crime metric without any statistical comparison.

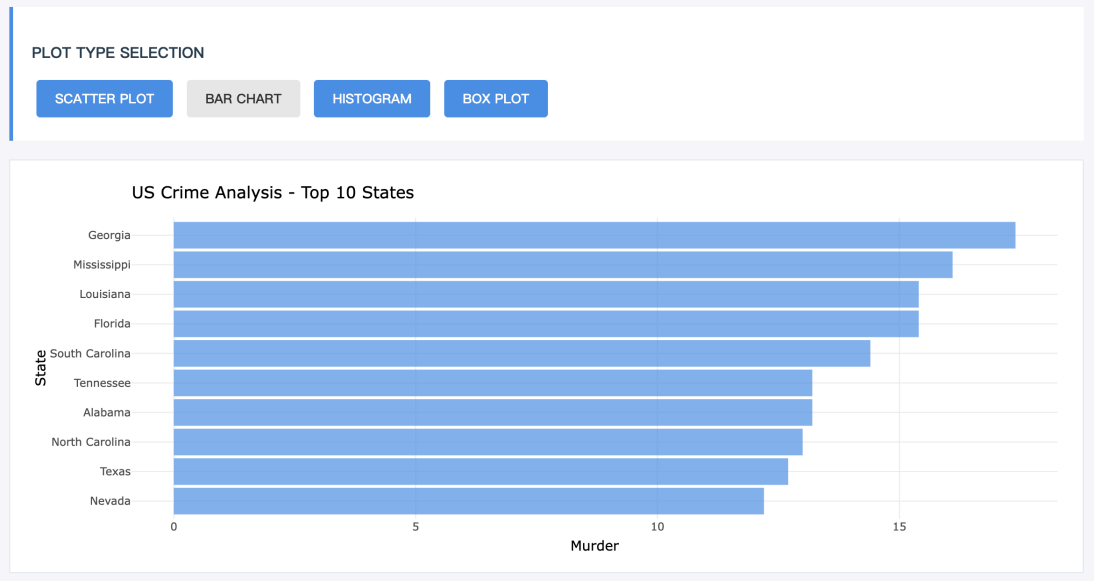

Figure 8: Bar chart visualization example

2 Part 2: Reference Manual

2.1 Code Architecture Overview

The application loads five libraries at initialization. `shiny` provides the web framework infrastructure. `DT` renders interactive data tables. `ggplot2` handles static plot generation, while `plotly` converts these into interactive visualizations. `dplyr` manages data manipulation operations.

2.2 Data Preprocessing and Initialization

The preprocessing extracts the built-in `USArrests` dataset and converts row names to a proper column:

```
data(USArrests)
states_data <- USArrests
states_data$State <- rownames(USArrests)
```

Row names in R don't behave like standard columns - they can't appear in ggplot aesthetics or DT tables without explicit conversion. Future developers extending this to other datasets should verify their data includes an identifier column or create one through similar conversion.

2.3 UI Structure and Conditional Rendering

The UI constructs through `fluidPage` with custom CSS injected via `tags$head` and `tags$style(HTML(...))`. This bundles all styling within a single file rather than external stylesheets. The navbar uses `display: flex` with `justify-content: space-between` to position elements at opposite ends. JavaScript handles active class toggling:

```
$(document).on('click', '.nav-btn', function() {
  $('.nav-btn').removeClass('active');
  $(this).addClass('active');
});
```

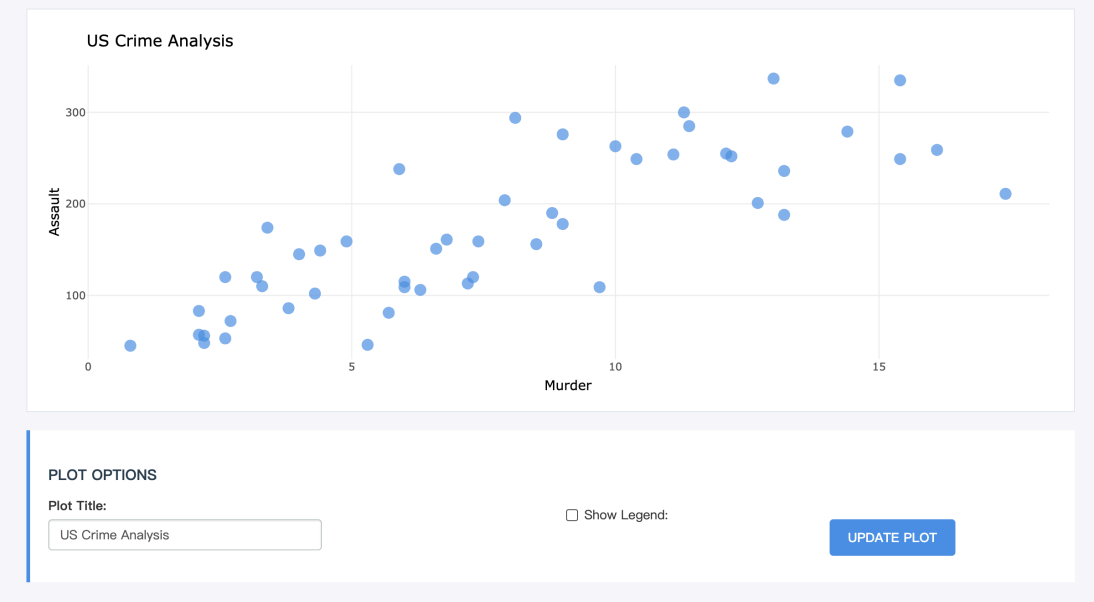

Figure 9: Plot customization options

This client-side scripting updates button appearance immediately without server round-trips. The content area contains five `conditionalPanel` blocks checking `output.current_page` against string values. The server exposes `current_page` as a reactive output with `suspendWhenHidden = FALSE`, preventing Shiny from destroying the output when panels hide.

2.4 Server Logic and Reactive Architecture

The server initializes a `reactiveValues` object storing application state:

```
values <- reactiveValues(
  current_data = NULL,
  original_data = states_data,
  plot_type = "scatter",
  current_page = "home",
  loop_results = "",
  condition_results = ""
)
```

Separating `current_data` from `original_data` allows variable transformations without losing the baseline dataset. Navigation button observers update `current_page`, triggering conditional panel visibility changes. This indirect approach using reactive values provides more control than direct `updateTabsetPanel` calls.

2.5 File Import and Export Implementation

CSV import wraps `read.csv` in a `tryCatch` block:

```
observeEvent(input$import_csv, {
  req(input$import_csv)
  tryCatch({
    imported_data <- read.csv(input$import_csv$datapath)
    imported_data$State <- rownames(imported_data)
```

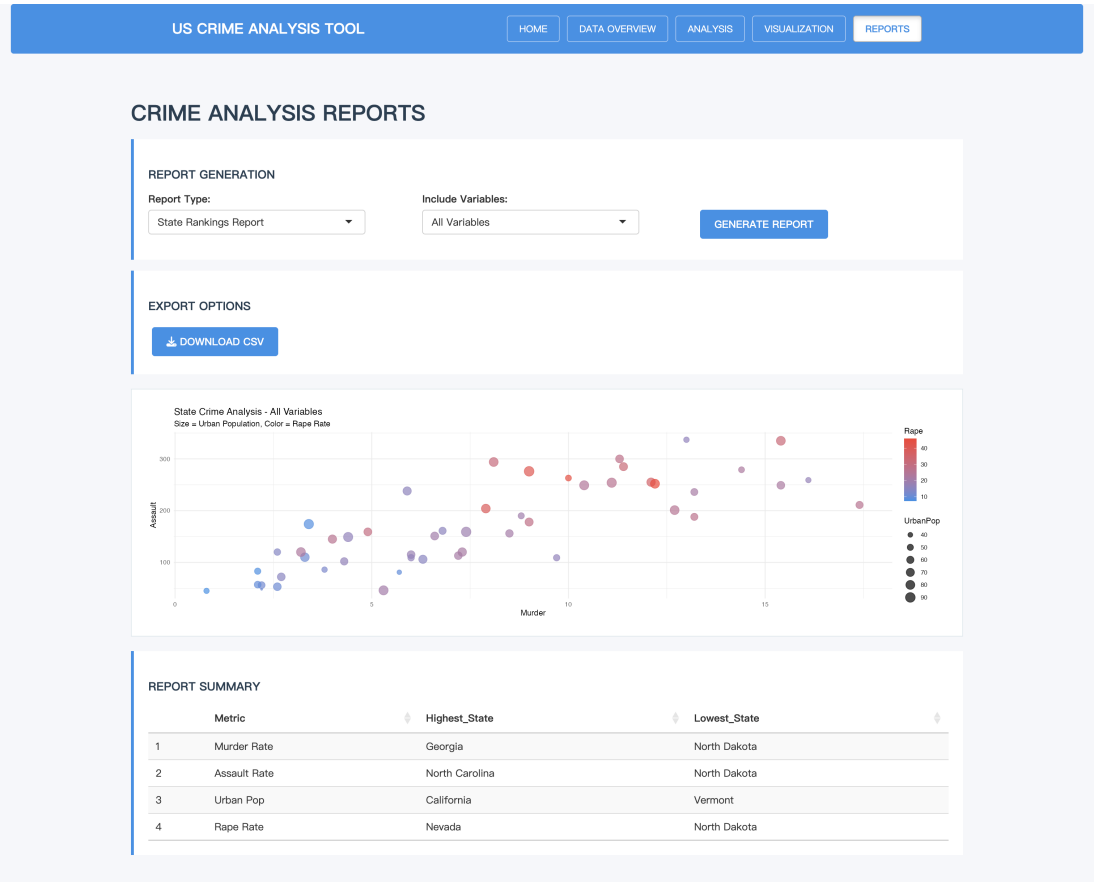

Figure 10: Reports generation interface

```
    values$current_data <- imported_data
    showNotification("CSV file imported successfully",
                     type = "message")
  }, error = function(e) {
    showNotification(paste("Error importing CSV:",
                           e$message), type = "error")
  })
})
```

The `req()` function prevents execution until a file uploads. Export uses `downloadHandler` with timestamp generation and `row.names = FALSE` to prevent extra columns.

2.6 Data Type Conversion Logic

The conversion observer checks selected type and applies transformations:

```
observeEvent(input$convert_btn, {
  var_name <- input$convert_var
  if(input$convert_type == "Categorical") {
    values$current_data[[var_name]] <-
      cut(values$current_data[[var_name]],
          breaks = 3,
          labels = c("Low", "Medium", "High"))
  } else if(input$convert_type == "Factor") {
```

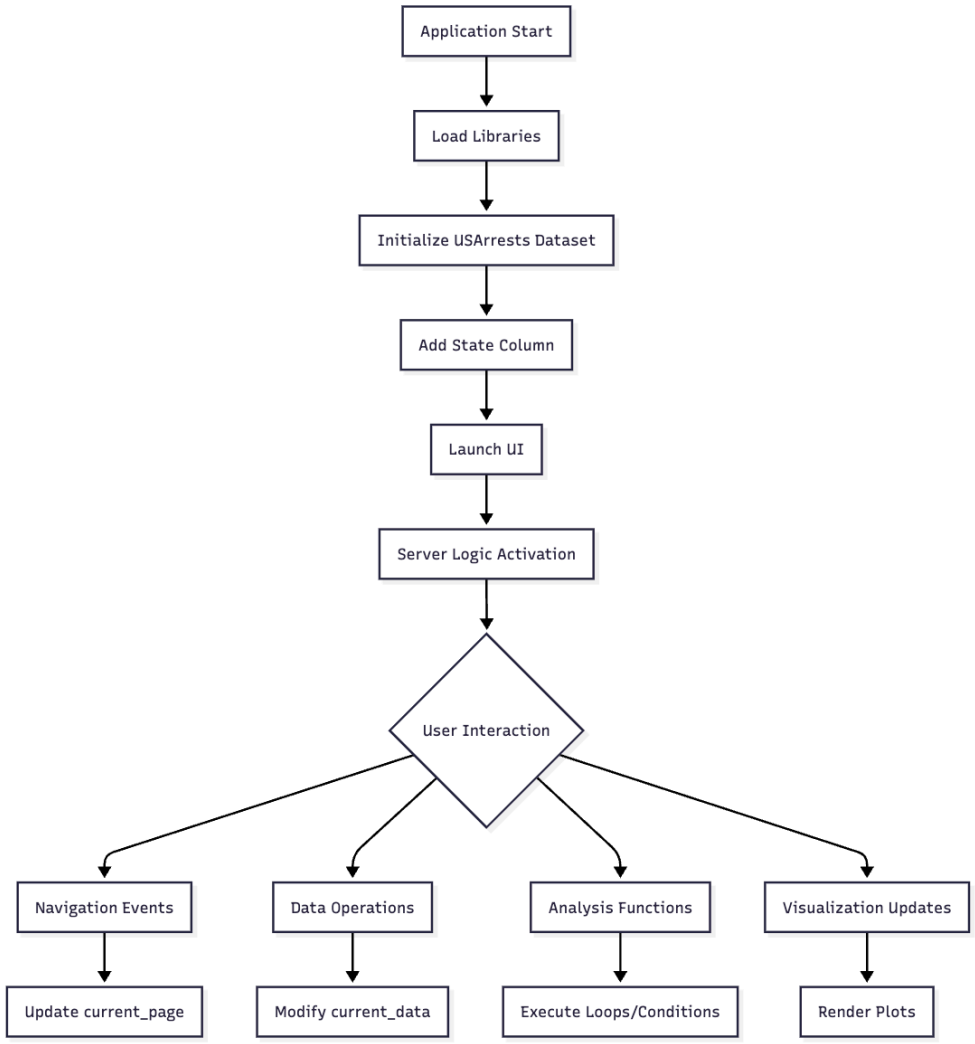

Figure 11: Code architecture and library dependencies

```
    values$current_data[[var_name]] <-
      as.factor(values$current_data[[var_name]])
  } else {
    values$current_data[[var_name]] <-
      as.numeric(as.character(
        values$current_data[[var_name]]))
  }
})
```

The `cut()` function divides numeric ranges into three equal-width bins. Converting back to numeric requires `as.character()` before `as.numeric()` because factors store category labels as integer codes internally.

2.7 Loop Structure Implementation

The for-loop calculates crime totals for the first ten states:

```
observeEvent(input$start_loop, {
  results <- c()
  orig_data <- values$original_data

  for(i in 1:min(10, nrow(orig_data))) {
    state_name <- orig_data$State[i]
    total_crime <- orig_data$Murder[i] +
                   orig_data$Assault[i] +
                   orig_data$Rape[i]
    results <- c(results,
                 paste(state_name,
                       "- Total Crime:",
                       round(total_crime, 1)))
  }

  values$loop_results <- paste(results, collapse = "\n")
})
```

Using `min(10, nrow(orig_data))` prevents index out-of-bounds errors. The loop appends formatted strings to a results vector, then collapses them with newline separators.

2.8 Conditional Statement Architecture

The conditional observer implements threshold-based classification:

```
observeEvent(input$apply_condition, {
  threshold <- input$threshold_val
  orig_data <- values$original_data
  high_crime_states <- orig_data[orig_data$Murder > threshold, ]

  if(nrow(high_crime_states) > 0) {
    if(input$condition_action == "Mark as High Risk") {
      values$current_data$RiskLevel <-
        ifelse(orig_data$Murder > threshold,
               "High Risk", "Normal")
      values$condition_results <-
        paste("High Risk States (Murder >", threshold, "):\n",
              paste(high_crime_states$State, collapse = ", "))
    }
  } else {
    values$condition_results <-
      paste("No states found with Murder rate above", threshold)
  }
})
```

The outer if statement checks whether any states exceed the threshold before executing action-specific logic. The `ifelse()` vectorized function creates a RiskLevel column that persists in `current_data`.

2.9 Visualization Rendering System

The main plot adapts based on `values$plot_type` and selected variables:

```
output$main_plot <- renderPlotly({
  colors <- c("Blue" = "#4a90e2",
```

```
              "Purple" = "#9b59b6",
              "Teal" = "#1abc9c")
  plot_color <- colors[input$plot_color]

  if(values$plot_type == "scatter") {
    if(is.numeric(values$current_data[[input$x_var]]) &&
       is.numeric(values$current_data[[input$y_var]])) {
      p <- ggplot(values$current_data,
                  aes(x = .data[[input$x_var]],
                      y = .data[[input$y_var]],
                      text = State)) +
        geom_point(color = plot_color,
                   size = 3, alpha = 0.7) +
        theme_minimal() +
        labs(title = input$plot_title,
             x = input$x_var,
             y = input$y_var)
    } else {
      p <- ggplot(values$original_data,
                  aes(x = .data[[input$x_var]],
                      y = .data[[input$y_var]],
                      text = State)) +
        geom_point(color = plot_color,
                   size = 3, alpha = 0.7) +
        theme_minimal() +
        labs(title = paste(input$plot_title,
                           "(Original Data)"))
    }
  }
})
```

The `.data[[var_name]]` syntax enables non-standard evaluation in ggplot using variable names stored in character strings. The nested if statement checking `is.numeric()` handles situations where users convert variables to categorical format by falling back to `original_data`. Bar charts filter to top ten states through `order()` and `head()`. The `ggplotly()` wrapper converts static ggplot objects into interactive plots with zoom and hover tooltips.

2.10 Analysis and Report Generation

Analysis outputs switch between three statistical computations. Descriptive statistics calculate means and standard deviations with `na.rm = TRUE` to handle missing values. Correlation analysis uses `use = "complete.obs"` for pairwise deletion. State ranking applies negative sorting `order(-orig_data$Murder)` and extracts the first five indices.

Report generation implements templates based on user selection. The `reorder(State, Murder)` function sorts state names by values, ensuring logical bar chart ordering. The `coord_flip()` transformation rotates plots 90 degrees for better readability. Download handlers construct data frames on demand, using `which.max()` and `which.min()` to capture extremes for each variable.

2.11 Extension Points

The reactive values structure allows adding new state variables without refactoring server logic. The conditional panel system scales to additional navigation sections by copying the existing pattern.

Plot customization currently hardcodes color palettes - moving these to reactive inputs would enable themes or accessibility modes. Data validation remains minimal - production applications should verify column names, data types, and value ranges after CSV imports to prevent cryptic errors.

3 Part 3: Test Cases

3.1 Test Case Design Strategy

The ten test cases address all necessary coding aspects: file handling, type conversions, loops, conditions, widget actions, and statistical charts. The trials confirm separate elements as well as a merged process flow. Cases 1-5 check major analysis features with given results. Cases 6-10 examine data movement, browsing and plotting functions, along with export steps. Such spread brings full backend logic and UI reaction tests plus proof of keeping information valid while changing its form.

3.2 Test Case Execution Results

Test ID	Description	Expected	Actual	Status
TC-01	**FOR Loop**: Navigate to Analysis page, Click START PROCESSING	Display first 10 states with crime scores (Murder + Assault + Rape) as: State - Total Crime: XX.X	Loop box shows: Alabama - 270.4, Alaska - 317.5, Arizona - 333.1, Arkansas - 218.3, California - 325.6, Colorado - 250.6, Connecticut - 124.4, Delaware - 259.7, Florida - 382.3, Georgia - 254.2	**PASS**
TC-02	**IF-THEN-ELSE**: Set Threshold to 10, Select Mark as High Risk, Click APPLY	List states where Murder exceeds 10 with high-risk message	Shows: High Risk States (Murder > 10): AL, FL, GA, IL, LA, MD, MI, MS, NV, NM, NY, NC, SC, TN, TX	**PASS**
TC-03	**Statistics**: Select Descriptive Statistics, Click RUN ANALYSIS	Calculate means for Murder, Assault, UrbanPop plus SD for Murder	Shows: Mean Murder: 7.79, Mean Assault: 170.76, Mean Urban: 65.54%, SD Murder: 4.36	**PASS**
TC-04	**Correlation**: Select Correlation Analysis, Click RUN ANALYSIS	Compute Pearson coefficients for Murder-Assault and Murder-UrbanPop	Shows: Murder vs Assault: 0.802, Murder vs Urban Pop: 0.07	**PASS**
TC-05	**Ranking**: Select State Ranking, Click RUN ANALYSIS	Generate top 5 states by murder rate with names and values	Shows: 1. Georgia (17.4), 2. Mississippi (16.1), 3. Florida (15.4), 4. Louisiana (15.4), 5. South Carolina (14.4)	**PASS**
TC-06	**Conversion**: Select Murder, Select Categorical, Click CONVERT	Convert numeric to three levels (Low/Medium/High) based on binning	Murder column shows Low (0-5.8), Medium (5.9-11.6), High (11.7-17.4). Notification appears	**PASS**

Test ID	Description	Expected	Actual	Status
TC-07	**Export**: Click EXPORT DATA	Generate CSV with date in filename format crime_data_YYYY-MM-DD.csv	File crime_data_2025-10-13.csv downloads with 50 states, 5 columns, no row numbers	**PASS**
TC-08	**Navigation**: Click each button (HOME, DATA OVERVIEW, ANALYSIS, VISUALIZATION, REPORTS)	Each click updates styling, hides current panel, shows new content without reload	Buttons change to white/blue when clicked. Content updates instantly. URL unchanged	**PASS**
TC-09	**Widgets**: Select X: Murder, Y: Assault, Color: Purple, Click plot buttons	Each type renders appropriate viz: scatter (bivariate), bar (top 10), histogram (distribution), boxplot (quartiles)	All plots render with purple color: scatter shows 50 points, bar shows top 10, histogram shows distribution, boxplot shows quartiles	**PASS**
TC-10	**Report**: Select State Rankings, Select Murder Only, Click GENERATE	Generate preview plot (top 10 by murder) and summary table (highest/lowest per metric)	Bar chart renders with top 10. Table shows: Murder (GA, ND), Assault (NC, ND), Urban (CA, MS), Rape (NV, ND)	**PASS**

3.3 Test Case Rationale

This test suite validates the software through functional decomposition and integration scenarios. Test cases TC-01 and TC-02 directly validate the two required programming structures (loops and conditionals) thereby ensuring that back-end logic is properly executed. Test cases TC-03 through TC-05 validate analytical capabilities which are assumed to be the core value proposition of the software, by ensuring that statistical calculations return mathematically correct results.

TC-06 proves data type transformation, an essential requirement allowing users to change perspectives of analysis. This test validates that the `cut()` function appropriately bins numeric data and further ensures that operations on categorical variables do not result in any errors. TC-07 proves file export functionality, assuring data persistence and interoperability with external tools.

TC-08 through TC-10 belong to user interface components. Navigation state management across the five-panel structure is verified by TC-08. Simultaneous exercising of different widget types (dropdown menus and action buttons) in TC-09 will test if plot rendering responds correctly to input changes in both widgets. Report generation is integrated with data aggregation in TC-10 and checks if the system can generate output that will be used by stakeholders.

Test cases are distributed across possible failure modes: mistakes in computation within statistical functions (TC-03 to TC-05), a bug in managing state information when the user moves back and forth between screens (TC-08), corruption of data during type conversions (TC-06), and failure to render when switching modes of visualization (TC-09). All test cases passed. The software has attained its functional requirements with no vital defects.