Dashboards for Clicker Data

INFO 4100 Learning Analytics

Aaradhyaa Gyawali

Part 1: Design Planning

Motivation

Instructors who use iClickers sit on a rich behavioral dataset — every attendance record, every clicker response, every quiz score — but rarely have a way to see it all together in a meaningful way. At the same time, students often have no visibility into how their own engagement patterns relate to their performance, or how they compare to peers. This project builds two interactive dashboards to close both gaps, using three years of clicker and quiz data from a Cornell Business Intelligence course (2016–2018, 123 students, 2,423 session records).

Dashboard Design

Two distinct user types drove two distinct designs, students and the instructor and the dashboard is design to make the data usable for both.

Student dashboard

The target user is an individual student with varying technical background who wants honest, actionable feedback, not raw numbers that are hard to contextualize. The central design question was: ***how do I know if I'm on track?*** Every visualization was designed to answer that question directly, by placing the student's data alongside a class benchmark. The dashboard surfaces attendance rate, average quiz score, and average sentiment as top-line stats, then provides four plots: quiz score trajectory overlaid on the class average, per-session clicker engagement colored by attendance, sentiment (temperature) responses over time, and a skill background comparison against class medians. A student selector dropdown makes the dashboard fully interactive. Any of the 123 students can explore their own record without accessing others'.

Instructor dashboard

The target user is the course instructor, who needs a rapid read on class health at the start of each week. The design prioritized breadth over depth: six panels covering attendance trends, quiz score trajectories with error bars, overall score distributions, grad vs. undergrad performance gaps, session-level clicker engagement, and a scatter plot of prior skill background against average quiz performance. The last panel directly addresses a recurring pedagogical question: does it actually matter whether students come in with prior SQL or database experience?

Wireframe

The instructor dashboard opens with a four-box summary row (total students, attendance rate, average quiz score, average sentiment), followed by three rows of paired plots progressing from high-level trends to distributional detail to predictive relationships. The student dashboard opens with the student selector, then a three-box summary row, then two rows of paired plots moving from performance to engagement to sentiment to background. Both layouts prioritize the most actionable information at the top.

Part 2: Dashboard Layout

The following defines the UI structure for both dashboard tabs — what panels appear, in what order, and what content each panel expects from the server.

```
instructor_dash = tabItem(
  tabName = "instructor",
  h2("Instructor Dashboard"),

  # Row 1: Top-line summary stats
  fluidRow(
    infoBoxOutput("inst.info1"),
    infoBoxOutput("inst.info2"),
    infoBoxOutput("inst.info3"),
    infoBoxOutput("inst.info4")
  ),

  # Row 2: Attendance and quiz score trends over sessions
  fluidRow(
    box(title = "Attendance Rate by Session", width = 6,
        plotOutput("inst.plot1", height = 280)),
```

```
    box(title = "Average Quiz Score by Quiz Number", width = 6,
        plotOutput("inst.plot2", height = 280))
  ),

  # Row 3: Score distribution and group comparison
  fluidRow(
    box(title = "Distribution of Quiz Scores", width = 6,
        plotOutput("inst.plot3", height = 280)),
    box(title = "Quiz Performance: Grad vs. Undergrad", width = 6,
        plotOutput("inst.plot4", height = 280))
  ),

  # Row 4: Engagement and skill background
  fluidRow(
    box(title = "Clicker Engagement by Session", width = 6,
        plotOutput("inst.plot5", height = 280)),
    box(title = "Prior Skill Score vs. Avg Quiz Score", width = 6,
        plotOutput("inst.plot6", height = 280))
  )
)

student_dash = tabItem(
  tabName = "student",
  h2("Student Dashboard"),

  # Student selector
  fluidRow(
    box(width = 4,
        selectInput("student_id", "Select Student ID:",
                    choices = NULL))  # populated in server
  ),

  # Row 1: Individual summary stats
  fluidRow(
    infoBoxOutput("stu.info1"),
    infoBoxOutput("stu.info2"),
    infoBoxOutput("stu.info3")
  ),

  # Row 2: Quiz score trajectory and session engagement
```

```
  fluidRow(
    box(title = "Your Quiz Score vs. Class Average", width = 6,
        plotOutput("stu.plot1", height = 280)),
    box(title = "Your Clicker Engagement by Session", width = 6,
        plotOutput("stu.plot2", height = 280))
  ),

  # Row 3: Sentiment over time and skill profile
  fluidRow(
    box(title = "Your Sentiment (Temperature) Over Sessions", width =
6,
        plotOutput("stu.plot3", height = 280)),
    box(title = "Your Skill Background vs. Class Median", width = 6,
        plotOutput("stu.plot4", height = 280))
  )
)
```

Part 3: Data Pre-processing

The two raw datasets — `experience` (one row per student, skill survey responses) and `quiz` (one row per student per session) — are joined and aggregated here into the summary structures needed by each visualization. Key decisions: clicker completion rate is undefined when zero questions were asked in a session, so those cases are excluded rather than treated as zero engagement; student-level averages are computed before joining to the experience table to avoid row duplication.

```
# --- Instructor-level aggregations ---

# Attendance rate per session
session_attendance <- quiz %>%
  group_by(session_number) %>%
  summarise(
    attendance_rate = mean(attended, na.rm = TRUE),
    avg_clicker_completion = mean(
      ifelse(total_possible_clicker > 0,
             total_completed_clicker / total_possible_clicker, NA),
      na.rm = TRUE),
    avg_temp = mean(avg_t_clicker, na.rm = TRUE),
```

```
    n_students = n_distinct(student_key)
  )

# Average quiz score per quiz number
quiz_avg <- quiz %>%
  filter(!is.na(quiz_score)) %>%
  group_by(quiz_number) %>%
  summarise(
    avg_score = mean(quiz_score, na.rm = TRUE),
    sd_score  = sd(quiz_score, na.rm = TRUE),
    n         = n()
  )

# Grad vs undergrad quiz performance
group_quiz <- quiz %>%
  left_join(experience %>% select(student_key, prog), by =
"student_key") %>%
  filter(!is.na(quiz_score), !is.na(prog))

# Skill score vs avg quiz score (student level)
student_avg_quiz <- quiz %>%
  group_by(student_key) %>%
  summarise(avg_quiz = mean(quiz_score, na.rm = TRUE))

skill_vs_quiz <- experience %>%
  left_join(student_avg_quiz, by = "student_key") %>%
  filter(!is.na(avg_quiz), !is.na(skill_survey_score))

# Overall stats for infoBoxes
overall_attendance   <- round(100 * mean(quiz$attended, na.rm = TRUE))
overall_avg_quiz     <- round(mean(quiz$quiz_score, na.rm = TRUE), 1)
overall_avg_temp     <- round(mean(quiz$avg_t_clicker, na.rm = TRUE),
1)
total_students       <- n_distinct(quiz$student_key)
```

Part 4: Server Logic and Visualizations

All plots are rendered reactively. The student dashboard plots update instantly when a new student ID is selected via the dropdown. Reactive guards (`req()`) prevent errors when the selector hasn't populated yet on first load.

```
server = function(input, output, session) {

  # Populate student selector with available IDs
  updateSelectInput(session, "student_id",
                    choices = sort(unique(quiz$student_key)))

  # ----------------------------------------------------------
  # INSTRUCTOR DASHBOARD
  # ----------------------------------------------------------

  output$inst.info1 = renderInfoBox({
    infoBox("Total Students", total_students,
            icon = icon("users"), color = "blue")
  })

  output$inst.info2 = renderInfoBox({
    infoBox("Avg Attendance", paste0(overall_attendance, "%"),
            icon = icon("check-circle"), color = "green")
  })

  output$inst.info3 = renderInfoBox({
    infoBox("Avg Quiz Score", paste0(overall_avg_quiz, " / 20"),
            icon = icon("graduation-cap"), color = "purple")
  })

  output$inst.info4 = renderInfoBox({
    infoBox("Avg Sentiment", paste0(overall_avg_temp, " / 5"),
            icon = icon("smile"), color = "yellow")
  })

  # Plot 1: Attendance rate by session
  output$inst.plot1 = renderPlot({
    ggplot(session_attendance, aes(x = session_number, y =
attendance_rate)) +
```

```
      geom_line(color = "#2c7bb6", linewidth = 1) +
      geom_point(color = "#2c7bb6", size = 2) +
      geom_hline(yintercept =
mean(session_attendance$attendance_rate),
                 linetype = "dashed", color = "gray50") +
      scale_y_continuous(labels = scales::percent_format(accuracy =
1),
                         limits = c(0, 1)) +
      labs(x = "Session Number", y = "Attendance Rate",
           caption = "Dashed line = overall average") +
      theme_minimal(base_size = 13) +
      theme(panel.grid.minor = element_blank())
  })

  # Plot 2: Avg quiz score by quiz number
  output$inst.plot2 = renderPlot({
    ggplot(quiz_avg, aes(x = quiz_number, y = avg_score)) +
      geom_col(fill = "#5e3c99", alpha = 0.8) +
      geom_errorbar(aes(ymin = avg_score - sd_score,
                        ymax = avg_score + sd_score),
                    width = 0.3, color = "gray40") +
      scale_x_continuous(breaks = 1:10) +
      scale_y_continuous(limits = c(0, 20)) +
      labs(x = "Quiz Number", y = "Average Score (out of 20)",
           caption = "Error bars show ± 1 SD") +
      theme_minimal(base_size = 13) +
      theme(panel.grid.minor = element_blank())
  })

  # Plot 3: Distribution of quiz scores
  output$inst.plot3 = renderPlot({
    quiz %>%
      filter(!is.na(quiz_score)) %>%
      ggplot(aes(x = quiz_score)) +
      geom_histogram(binwidth = 1, fill = "#1a9641", color = "white",
alpha = 0.85) +
      geom_vline(xintercept = overall_avg_quiz,
                 linetype = "dashed", color = "red", linewidth = 1) +
      labs(x = "Quiz Score (out of 20)", y = "Count",
           caption = "Red dashed line = class average") +
```

```
      theme_minimal(base_size = 13)
  })

  # Plot 4: Grad vs undergrad quiz performance
  output$inst.plot4 = renderPlot({
    group_quiz %>%
      ggplot(aes(x = prog, y = quiz_score, fill = prog)) +
      geom_boxplot(alpha = 0.75, outlier.alpha = 0.4) +
      scale_fill_manual(values = c("GRAD" = "#d7191c", "UGRAD" =
"#2c7bb6")) +
      labs(x = "Student Group", y = "Quiz Score (out of 20)",
           fill = "Group") +
      theme_minimal(base_size = 13) +
      theme(legend.position = "none")
  })

  # Plot 5: Clicker engagement by session
  output$inst.plot5 = renderPlot({
    session_attendance %>%
      filter(!is.na(avg_clicker_completion)) %>%
      ggplot(aes(x = session_number, y = avg_clicker_completion)) +
      geom_col(fill = "#fdae61", alpha = 0.85) +
      geom_hline(yintercept =
mean(session_attendance$avg_clicker_completion,
                                    na.rm = TRUE),
                 linetype = "dashed", color = "gray40") +
      scale_y_continuous(labels = scales::percent_format(accuracy =
1),
                         limits = c(0, 1)) +
      labs(x = "Session Number", y = "Avg Clicker Completion Rate",
           caption = "Dashed line = overall average") +
      theme_minimal(base_size = 13)
  })

  # Plot 6: Prior skill score vs avg quiz score
  output$inst.plot6 = renderPlot({
    ggplot(skill_vs_quiz, aes(x = skill_survey_score, y = avg_quiz,
                              color = prog)) +
      geom_point(alpha = 0.7, size = 2.5) +
      geom_smooth(method = "lm", se = TRUE, color = "gray30",
```

```
                  linetype = "dashed", linewidth = 0.8) +
      scale_color_manual(values = c("GRAD" = "#d7191c", "UGRAD" =
"#2c7bb6"),
                         na.value = "gray60") +
      labs(x = "Skill Survey Score (sum)", y = "Avg Quiz Score",
           color = "Group",
           caption = "Dashed line = linear trend") +
      theme_minimal(base_size = 13)
  })

  # ------------------------------------------------------
  # STUDENT DASHBOARD (reactive to selected student)
  # ------------------------------------------------------

  stu_quiz <- reactive({
    req(input$student_id)
    quiz %>% filter(student_key == input$student_id)
  })

  stu_exp <- reactive({
    req(input$student_id)
    experience %>% filter(student_key == input$student_id)
  })

  output$stu.info1 = renderInfoBox({
    att <- round(100 * mean(stu_quiz()$attended, na.rm = TRUE))
    infoBox("Your Attendance", paste0(att, "%"),
            icon = icon("calendar-check"), color = "green")
  })

  output$stu.info2 = renderInfoBox({
    avg <- round(mean(stu_quiz()$quiz_score, na.rm = TRUE), 1)
    infoBox("Your Avg Quiz Score", paste0(avg, " / 20"),
            icon = icon("star"), color = "purple")
  })

  output$stu.info3 = renderInfoBox({
    avg_t <- round(mean(stu_quiz()$avg_t_clicker, na.rm = TRUE), 1)
    infoBox("Your Avg Sentiment", paste0(avg_t, " / 5"),
            icon = icon("smile"), color = "yellow")
```

```r
  })

  # Student Plot 1: Quiz score trajectory vs class average
  output$stu.plot1 = renderPlot({
    stu_scores <- stu_quiz() %>%
      filter(!is.na(quiz_score)) %>%
      group_by(quiz_number) %>%
      summarise(score = mean(quiz_score, na.rm = TRUE))

    ggplot() +
      geom_line(data = quiz_avg, aes(x = quiz_number, y = avg_score),
                color = "gray60", linewidth = 1, linetype = "dashed")
+
      geom_point(data = quiz_avg, aes(x = quiz_number, y = avg_score),
                 color = "gray60", size = 2) +
      geom_line(data = stu_scores, aes(x = quiz_number, y = score),
                color = "#d7191c", linewidth = 1.2) +
      geom_point(data = stu_scores, aes(x = quiz_number, y = score),
                 color = "#d7191c", size = 3) +
      scale_x_continuous(breaks = 1:10) +
      scale_y_continuous(limits = c(0, 20)) +
      labs(x = "Quiz Number", y = "Score (out of 20)",
           caption = "Red = you    |    Gray dashed = class average") +
      theme_minimal(base_size = 13)
  })

  # Student Plot 2: Clicker engagement by session
  output$stu.plot2 = renderPlot({
    stu_eng <- stu_quiz() %>%
      mutate(completion = ifelse(total_possible_clicker > 0,
                                 total_completed_clicker /
total_possible_clicker,
                                 NA))

    ggplot(stu_eng, aes(x = session_number, y = completion,
                        fill = as.factor(attended))) +
      geom_col(alpha = 0.85) +
      scale_fill_manual(values = c("0" = "#d7191c", "1" = "#2c7bb6"),
                        labels = c("Absent", "Present"),
                        name = "Attendance") +
```

```
      scale_y_continuous(labels = scales::percent_format(accuracy =
1),
                         limits = c(0, 1)) +
      labs(x = "Session Number", y = "Clicker Completion Rate") +
      theme_minimal(base_size = 13)
  })

  # Student Plot 3: Sentiment (temperature) over sessions
  output$stu.plot3 = renderPlot({
    stu_temp <- stu_quiz() %>% filter(!is.na(avg_t_clicker))

    ggplot(stu_temp, aes(x = session_number, y = avg_t_clicker)) +
      geom_hline(yintercept = 3, linetype = "dashed", color =
"gray70") +
      geom_line(color = "#fdae61", linewidth = 1.1) +
      geom_point(aes(color = avg_t_clicker), size = 3) +
      scale_color_gradient2(low = "#d7191c", mid = "#fdae61", high =
"#1a9641",
                            midpoint = 3, limits = c(0, 5), name =
"Temp") +
      scale_y_continuous(limits = c(0, 5), breaks = 0:5) +
      labs(x = "Session Number", y = "Avg Temperature Response (0–5)",
           caption = "Dashed line = neutral (3)") +
      theme_minimal(base_size = 13)
  })

  # Student Plot 4: Skill background vs class median
  output$stu.plot4 = renderPlot({
    skill_cols <- c("database_score", "sql_score", "programing_score",
                    "stored_proc_score", "etl_score",
"data_vis_score",
                    "requirement_gather_score")

    class_medians <- experience %>%
      summarise(across(all_of(skill_cols), median, na.rm = TRUE)) %>%
      pivot_longer(everything(), names_to = "skill", values_to =
"class_median")

    stu_skills <- stu_exp() %>%
      select(all_of(skill_cols)) %>%
      pivot_longer(everything(), names_to = "skill", values_to =
```

```
"your_score")

    combined <- left_join(stu_skills, class_medians, by = "skill") %>%
      mutate(skill = str_replace(skill, "_score", "") %>%
               str_replace_all("_", " ") %>%
               str_to_title())

    combined %>%
      pivot_longer(c(your_score, class_median),
                   names_to = "who", values_to = "score") %>%
      mutate(who = recode(who,
                          "your_score"   = "You",
                          "class_median" = "Class Median")) %>%
      ggplot(aes(x = reorder(skill, score), y = score, fill = who)) +
      geom_col(position = "dodge", alpha = 0.85) +
      scale_fill_manual(values = c("You" = "#d7191c",
                                   "Class Median" = "#2c7bb6")) +
      scale_y_continuous(limits = c(0, 5), breaks = 0:5) +
      coord_flip() +
      labs(x = NULL, y = "Score (0–5)", fill = NULL) +
      theme_minimal(base_size = 13) +
      theme(legend.position = "top")
  })
}
```

Part 5: Launch and Evaluation

The chunk below assembles the UI and launches the Shiny app. Run it interactively in RStudio rather than knitting, the dashboard opens in the Viewer pane and can be expanded to full browser view.

```
dhead = dashboardHeader(title = "BI Clicker Dashboard")

dside = dashboardSidebar(
  sidebarMenu(
    menuItem("Instructor View", tabName = "instructor",
             icon = icon("chalkboard-teacher")),
    menuItem("Student View",    tabName = "student",
```

```
              icon = icon("user-graduate"))
  )
)

dbody = dashboardBody(
  tabItems(
    instructor_dash,
    student_dash
  )
)

ui = dashboardPage(dhead, dside, dbody)

shinyApp(ui, server)
```

Evaluation

Student dashboard

The fully reactive student selector is the feature that makes this dashboard genuinely useful rather than just illustrative. It means any student can use the same tool without needing separate outputs generated for each person. The quiz score overlay on the class average was deliberately kept simple: one red line, one gray dashed line. Adding more visual complexity (percentile bands, confidence intervals) would have made it harder to read quickly, which is the opposite of what a student checking in before an exam needs.

Given more time, two additions would meaningfully improve the student view. First, a "sessions at risk" flag, automatically highlighting sessions where the student was physically present but had near-zero clicker engagement, would make disengagement patterns visible rather than requiring the student to spot them manually in the bar chart. Second, a percentile rank infoBox would give students a more intuitive sense of where they stand than a raw score out of 20.

Instructor dashboard

The six-panel layout was designed so that the instructor can answer the most common questions, like Is attendance dropping? Which quiz was hardest? Are grad students outperforming undergrads?, without clicking through multiple tabs. The skill background scatter plot (prior survey score vs. average quiz performance) is the most analytically

interesting panel: if the relationship is weak, it suggests the course is successfully leveling the playing field regardless of prior experience.

The most valuable addition would be a filterable student-level table below the aggregate plots, allowing the instructor to move from noticing a pattern (e.g., attendance dropped in session 8) to identifying which specific students were absent and may need outreach. A year-over-year filter (2016 / 2017 / 2018) would also let the instructor evaluate whether course changes between semesters had measurable effects on engagement or performance.

Technical Notes

The most important data handling decision was in the clicker completion rate calculation. Dividing `total_completed_clicker` by `total_possible_clicker` produces `NaN` in sessions where no clicker questions were asked, treating those as zero completion would misrepresent students who simply had no opportunity to respond. These cases are excluded via `ifelse()` and `na.rm = TRUE` throughout. The same conservative approach applies to the temperature (sentiment) plots, where sessions with no temperature questions are excluded rather than imputed.

Reflection

The central challenge of this project was editorial. Both datasets are rich enough to support dozens of visualizations, and the harder problem was deciding what *not* to show. For the student dashboard, the guiding principle was: show only what a student can act on. For the instructor dashboard, it was: show the patterns that are invisible in a gradebook but visible in behavioral data. Clicker data is especially interesting for this because it captures engagement in real time, not just outcomes — a student can score well on a quiz while consistently disengaging mid-session, and only the behavioral log reveals that.

Building the reactive student selector also required thinking carefully about Shiny's execution model, understanding that `reactive()` blocks re-run on input change, and that `req()` is necessary to prevent the server from trying to filter on a NULL student ID before the UI has rendered. These are small details but they reflect a broader principle: interactive data tools fail silently in ways that static reports don't, so defensive programming matters more.

Technical Stack

R, tidyverse (dplyr, ggplot2, tidyr, stringr), Shiny, shinydashboard, scales

AI Assistance

ChatGPT was used to debug specific ggplot2 syntax — particularly the `scale_color_gradient2` gradient configuration in the sentiment plot and the `scales::percent_format` axis formatting. All design decisions, feature logic, and written analysis are original.