# 37 Building a Bibliometric Shiny App

*R. Heller · Date: 2026-05-09 · Last changed: 2026-10-31*

## 37.1 Learning objectives

After completing this chapter, you will be able to:

- Design a Shiny app architecture for bibliometric analysis
- Implement reactive data fetching from OpenAlex
- Build interactive UI components: dropdowns, sliders, and tabbed panels
- Create server-side logic for citation analysis and network visualisation
- Deploy a Shiny app to shinyapps.io or a self-hosted server

## 37.2 Setup

```
library(tidyverse)
library(shiny)
library(openalexR)
library(glue)

set.seed(20260509)

source(here::here("R", "sci_palette.R"))
```

## 37.3 Conceptual background

Shiny transforms R scripts into web applications. Users interact through widgets (dropdowns, sliders, buttons) and see results update in real time. For bibliometrics, Shiny apps serve two audiences: researchers who want to explore data without writing code, and decision-makers who need accessible analytics dashboards.

A Shiny app has two components: the **UI** (user interface, defining layout and inputs) and the **server** (the R logic that reacts to user input and generates output). **Reactive programming** connects inputs to outputs: when a user changes a dropdown, only the computations that depend on that input are re-executed.

For bibliometric apps, the typical architecture involves:

1. **Input panel**: Select a journal, institution, or search query; choose date range and sample size.
2. **Data fetching**: Reactively query OpenAlex based on user selections.
3. **Analysis tabs**: Trend charts, citation distributions, author rankings, and optionally network visualisation.
4. **Export**: Download buttons for data (CSV) and figures (PNG).

`biblioshiny()` from the `bibliometrix` package (Aria and Cuccurullo 2017) provides a ready-made Shiny interface for many standard analyses. Building a custom app gives full control over the analysis pipeline and user experience.

Deployment options include shinyapps.io (hosted by Posit, free tier available), Posit Connect (enterprise), and self-hosted via Shiny Server (open source).

## 37.4 Worked example

### 37.4.1 App structure

We demonstrate the key components of a bibliometric Shiny app. The code below shows the UI and server logic separately for clarity.

### 37.4.2 UI definition

```
ui ← fluidPage(
  titlePanel("Bibliometric Explorer"),

  sidebarLayout(
    sidebarPanel(
      textInput("journal_id", "OpenAlex Source ID:",
                value = "S148561398"),
      sliderInput("year_range", "Publication Years:",
                  min = 2015, max = 2023, value = c(2020, 2023)),
      numericInput("sample_size", "Sample Size:", value = 200,
                   min = 50, max = 1000),
      actionButton("fetch", "Fetch Data", class = "btn-primary"),
      hr(),
      downloadButton("download_csv", "Download CSV")
    ),

    mainPanel(
      tabsetPanel(
        tabPanel("Trends",
                 plotOutput("trend_plot"),
                 tableOutput("summary_table")),
        tabPanel("Citations",
                 plotOutput("cite_dist"),
                 plotOutput("cite_boxplot")),
        tabPanel("Data",
                 DT::dataTableOutput("data_table"))
      )
    )
  )
)
```

### 37.4.3 Server logic

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

  works_data ← eventReactive(input$fetch, {
    withProgress(message = "Fetching from OpenAlex...", {
      fetch_openalex(
        entity = "works",
        primary_location.source.id = input$journal_id,
        from_publication_date = paste0(input$year_range[1], "-01-01"),
        to_publication_date = paste0(input$year_range[2], "-12-31"),
        type = "article",
        options = list(sample = input$sample_size, seed = 42)
      ) ▷
        transmute(
          id, title = display_name,
          year = year(publication_date),
          cited_by_count, oa_status
        )
    })
  })

  output$trend_plot ← renderPlot({
    req(works_data())
    works_data() ▷
      count(year) ▷
      ggplot(aes(x = factor(year), y = n)) +
      geom_col(fill = palette_sci(1)) +
      labs(x = "Year", y = "Publications") +
      theme_sci()
  })

  output$cite_dist ← renderPlot({
    req(works_data())
    ggplot(works_data(), aes(x = cited_by_count)) +
      geom_histogram(binwidth = 5, fill = palette_sci(1), colour = "white") +
      labs(x = "Citations", y = "Count") +
      theme_sci()
  })

  output$cite_boxplot ← renderPlot({
    req(works_data())
    works_data() ▷
      filter(!is.na(oa_status)) ▷
      ggplot(aes(x = oa_status, y = cited_by_count + 1)) +
      geom_boxplot(fill = palette_sci(1), alpha = 0.7) +
      scale_y_log10() +
      labs(x = "OA Status", y = "Citations (log)") +
      theme_sci()
  })

  output$summary_table ← renderTable({
    req(works_data())
    works_data() ▷
      summarise(
        Articles = n(),
        `Mean Citations` = round(mean(cited_by_count), 1),
        `Median Citations` = median(cited_by_count),
        `OA Rate` = scales::percent(mean(oa_status ≠ "closed", na.rm = TRUE))
      )
  })

  output$data_table ← DT::renderDataTable({
```

```
    req(works_data())
    works_data() ▷
      select(title, year, cited_by_count, oa_status) ▷
      arrange(desc(cited_by_count))
  })

  output$download_csv ← downloadHandler(
    filename = function() paste0("biblio_data_", Sys.Date(), ".csv"),
    content = function(file) write_csv(works_data(), file)
  )
}
```

### 37.4.4 Running the app

```
shinyApp(ui = ui, server = server)
```

### 37.4.5 biblioshiny as an alternative

```
# bibliometrix provides a ready-made Shiny interface:
# library(bibliometrix)
# biblioshiny()
```

## 37.5 Diagnostics and interpretation

- **Responsiveness**: API calls can take several seconds. Use `withProgress()` to show loading indicators.
- **Error handling**: Wrap API calls in `tryCatch()` to display user-friendly error messages instead of stack traces.
- **Memory**: Each user session loads data into R memory. For apps with many concurrent users, pre-aggregate data or use database backends.
- **Testing**: Use `shinytest2` to automate testing of Shiny apps.

## 37.6 Limitations and responsible use

## 37.7 Limitations and responsible use

- **Shiny apps are not analyses.** They enable exploration but do not substitute for rigorous, documented analytical pipelines. Use Shiny for discovery, not for producing final results.
- **API rate limits.** Live OpenAlex queries from a Shiny app can hit rate limits if many users fetch data simultaneously. Use caching or pre-fetched data for production apps.
- **Maintenance.** Shiny apps require ongoing maintenance: API changes, package updates, and server costs. Plan for long-term sustainability before deploying.

- **Interpretive guardrails.** Non-technical users may misinterpret metrics. Build in tooltips, descriptions, and warnings — especially around citation-based indicators (Hicks et al. 2015; American Society for Cell Biology 2012).

## 37.8 Common pitfalls

## 37.9 Common pitfalls

- **Fetching data on every input change.** Use `eventReactive()` with an action button instead of `reactive()` to prevent re-fetching on every slider adjustment.
- **Not validating inputs.** Check that the journal ID exists and the date range is valid before making API calls.
- **Monolithic server functions.** Break the server logic into modules for maintainability as the app grows.
- **Deploying without testing.** Test with realistic data volumes before deploying. An app that works with 100 records may fail with 10,000.

## 37.10 Exercises

1. **Add network tab.** Extend the app with a tab that builds and displays a co-authorship network using `visNetwork`.
2. **Caching.** Implement server-side caching so that repeated queries with the same parameters return instantly.
3. **Deployment.** Deploy the app to shinyapps.io. Test with different journal IDs and date ranges.
4. **biblioshiny comparison.** Run `biblioshiny()` and compare its features with your custom app. What does biblioshiny offer that your app does not?

## 37.11 Solutions

Solutions are provided in E.

## 37.12 Further reading

- Aria and Cuccurullo (2017) — `bibliometrix` and `biblioshiny()` for turnkey bibliometric exploration.
- Priem et al. (2022) — OpenAlex API for reactive data fetching.
- Hicks et al. (2015) — The Leiden Manifesto: designing interfaces that prevent metric misuse.
- American Society for Cell Biology (2012) — DORA: responsible assessment in interactive tools.

## 37.13 Session info

```
sessionInfo()
```

```
#> R version 4.6.1 (2026-06-24)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 24.04.5 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK
version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
#>  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
#>  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
#> [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] shiny_1.15.0              DT_0.34.0
#>  [3] plotly_4.12.1             uwot_0.2.5
#>  [5] Matrix_1.7-5              word2vec_0.4.1
#>  [7] stm_1.3.10                topicmodels_0.2-17
#>  [9] quanteda.textstats_0.97.2 visNetwork_2.1.4
#> [11] ggraph_2.2.2              tidygraph_1.3.1
#> [13] igraph_2.3.4              quanteda_4.5.1
#> [15] pdftools_3.9.1            arrow_26.0.1.1
#> [17] bibliometrix_5.5.0        RefManageR_1.4.0
#> [19] bib2df_1.1.2.0            rcrossref_1.2.1
#> [21] gt_1.4.0                  tidytext_0.4.3
#> [23] glue_1.8.1                openalexR_3.1.2
#> [25] lubridate_1.9.5           forcats_1.0.1
#> [27] stringr_1.6.0             dplyr_1.2.1
#> [29] purrr_1.2.2               readr_2.2.0
#> [31] tidyr_1.3.2               tibble_3.3.1
#> [33] ggplot2_4.0.4             tidyverse_2.0.0
#>
#> loaded via a namespace (and not attached):
#>   [1] bibtex_0.5.3           RColorBrewer_1.1-3     rstudioapi_0.19.0
#>   [4] jsonlite_2.0.0         magrittr_2.0.5         modeltools_0.2-25
#>   [7] farver_2.1.2           rmarkdown_2.32         fs_2.1.0
#>  [10] vctrs_0.7.3            memoise_2.0.1          askpass_1.2.1
#>  [13] base64enc_0.1-6        htmltools_0.5.9        contentanalysis_1.1.1
#>  [16] curl_8.1.0             broom_1.0.13           janeaustenr_1.0.0
#>  [19] cellranger_1.1.0       sass_0.5.0             bslib_0.12.0
#>  [22] htmlwidgets_1.6.4      tokenizers_0.3.0       plyr_1.8.9
#>  [25] httr2_1.3.0            cachem_1.1.0           dimensionsR_0.0.3
#>  [28] mime_0.13              lifecycle_1.0.5        pkgconfig_2.0.3
#>  [31] R6_2.6.1               fastmap_1.2.0          digest_0.6.39
#>  [34] patchwork_1.3.2        shinycssloaders_1.1.0  rprojroot_2.1.1
#>  [37] RSpectra_0.16-2        crosstalk_1.2.2        SnowballC_0.7.1
#>  [40] labeling_0.4.3         urltools_1.7.3.1       timechange_0.4.0
#>  [43] mgcv_1.9-4             polyclip_1.10-7        httr_1.4.9
#>  [46] compiler_4.6.1         here_1.0.2             bit64_4.8.6
#>  [49] withr_3.0.3            backports_1.5.1        S7_1.0.0
#>  [52] viridis_0.6.5          ggforce_0.5.0          MASS_7.3-65
#>  [55] bibliometrixData_0.3.0 tools_4.6.1            otel_0.3.0
#>  [58] stopwords_2.3          zip_3.0.2              httpuv_1.6.17
#>  [61] rentrez_1.2.4          nlme_3.1-169           promises_1.5.0
#>  [64] grid_4.6.1             stringdist_0.9.17      reshape2_1.4.5
```

```
#>  [67] generics_0.1.4         gtable_0.3.6           tzdb_0.5.0
#>  [70] rscopus_0.9.0          ca_0.72                data.table_1.18.6.1
#>  [73] hms_1.1.4              xml2_1.6.0             utf8_1.2.6
#>  [76] ggrepel_0.9.8          pillar_1.11.1          nsyllable_1.0.1
#>  [79] vroom_1.7.1            later_1.4.8            splines_4.6.1
#>  [82] tweenr_2.0.3           lattice_0.22-9         FNN_1.1.4.1
#>  [85] bit_4.6.0              tidyselect_1.2.1       tm_0.7-20
#>  [88] miniUI_0.1.2           downlit_0.4.5          knitr_1.52
#>  [91] gridExtra_2.3.1        NLP_0.3-3              bookdown_0.48
#>  [94] stats4_4.6.1           curl_1.6.0             xfun_0.62
#>  [97] graphlayouts_1.2.5     matrixStats_1.5.0      humaniformat_0.6.0
#> [100] stringi_1.8.9          qpdf_1.4.1             yaml_2.3.12
#> [103] evaluate_1.0.5         httpcode_0.3.0         cli_3.6.6
#> [106] xtable_1.8-8           jquerylib_0.1.4        Rcpp_1.1.2
#> [109] readxl_1.6.0           triebeard_0.4.1        XML_3.99-0.25
#> [112] parallel_4.6.1         assertthat_0.2.1       pubmedR_1.1.0
#> [115] slam_0.1-56            viridisLite_0.4.3      scales_1.4.0
#> [118] crayon_1.5.3           openxlsx_4.2.9         rlang_1.3.0
#> [121] fastmatch_1.1-8
```