bidux: Behavioral Insight Design for Shiny Dashboards

Jeremy R. Winget, PhD[a,*]

[a] *Independent Researcher, Chicago, 60660*

Abstract

Dashboards often degrade into sprawling filter farms: impressive on paper, but opaque to users, brittle in use, and shaped by improvisation rather than evidence. `bidux` is an R package that brings a behavioral-science workflow to Shiny development. It formalizes five stages (Interpret, Notice, Anticipate, Structure, Validate) and optionally links them to telemetry so that UI decisions are driven by observed cognitive friction rather than guesswork. This article describes the design motivations, theoretical grounding, architecture, and usage sketch of `bidux`, and situates it within the UX/HCI/decision-making literature.

Keywords: R, shiny, behavioral science, UX, human-computer interaction

1. Summary

Dashboards rarely lack features. What they often lack is *insight*, a bridge between domain logic and how real humans use the interface. I created `bidux` to plug that gap. It embeds a behavioral-science workflow into existing Shiny applications, helping teams move from gut-driven layout decisions to structured, auditable, cognitively grounded UI changes.

At its simplest, `bidux` takes a compact set of design inputs (e.g. a central question, user personas, friction notes) and produces a recommended roadmap: which UI levers to tune (e.g. reduce filter count, adopt progressive disclosure, reorder content zones) and why. If you have telemetry, `bidux` can highlight the biggest bottlenecks in the user journey, linking interventions to measurable gains.

`bidux` is designed for R/Shiny practitioners who care about usability, but can't always hire a UX specialist. It augments, rather than replaces, existing Shiny best practices with a disciplined, theory-informed workflow.

[*] Corresponding author
Email address: `contact@jrwinget.com` (Jeremy R. Winget, PhD)

2. Statement of Need

In my consulting and dashboard audits, I repeatedly encountered dashboards where drop-downs proliferated, click counts ballooned, and user engagement dropped fast. Many dashboards are constructed by functional experimenters, domain experts, or statisticians, rarely by designers. When heuristics or UX reviews arise, they tend to be applied ad hoc, not baked into the development process.

Researchers and practitioners alike need a middle path: a lightweight, repeatable, theory-informed workflow for designing dashboards that respect real human cognitive constraints (choice overload, poor scanning, information scent failure). `bidux` addresses precisely that gap. It helps democratize behavioral design in dashboards by making the core decisions auditable, revisitable, and repeatable.

Moreover, in research contexts, dashboards often come with a need for interpretability and justification: stakeholders want to see *why* a UI change was made, not just that it improved engagement. `bidux` provides that rationale trail.

3. Theoretical & Empirical Foundations

I won't review every HCI or design book here. Instead, I anchor two theoretical pillars that `bidux` leans on heavily:

3.1. Choice overload & decision friction

The paradox that more options can *hurt* decision making is well documented. A meta-analysis of 99 experimental conditions (N = 7,202) identified that large option sets systematically degrade satisfaction, increase regret, and more frequently lead to choice deferral, particularly when decision tasks are complex or user preferences uncertain (i.e. moderate or ambivalent) (Chernev et al., 2015). The effect is moderated by task difficulty, preference certainty, and cognitive load.

More recent work argues that many traditional tests of choice overload are underpowered and that richer data often confirm that the phenomenon is more common than earlier studies suggested (Dean et al., 2022).

This supports a core design rule: fewer competing controls or clearer defaults often reduce user paralysis and friction.

3.2. Visual hierarchy & scanning behavior

Empirical eye-tracking studies have repeatedly shown that users scan screens in an "F-pattern", starting top-left, moving horizontally, then vertically and again horizontally, especially in Western reading contexts (Djamasbi et al., 2011; Group, 2017). Information buried low or far right is often missed entirely.

Further, layout complexity increases scattering of attention; simpler, well-ordered layouts help preserve the scanning trajectory (Djamasbi et al., 2011). More recent GUI studies reaffirm that users' visual search in graphical interfaces still reflect reading-based scanning heuristics (Putkonen et al., 2025).

Thus, interface structure matters, not just number of options, but where and how they appear relative to the scanning eye.

4. Design & Architecture of `bidux`

Below I describe the mental model (workflow), key modules, and API sketch (high level).

4.1. Behavioral design workflow (five stages)

1. **Interpret** Define the *central question* your dashboard should support, describe context (tension, trade-offs), articulate 1-3 personas (users and their motivations).
2. **Notice** Record concrete design problems or usage observations (e.g. "users ignore side filters", "clicks drop off after two filters"). Optionally map each to behavioral theory (e.g. "choice overload", "search cost").
3. **Anticipate** For each design problem, sketch likely cognitive failures (e.g. overload, choice deferral, ordering bias) and propose candidate mitigations (defaults, guardrails, pre-filtering, grouping).
4. **Structure** Lay out a candidate UI hierarchy (e.g. "first meaningful view," high-salience zones, prioritized filters). Internally, `bidux` leverages a lookup table that maps behavioral risk recommended Shiny interventions (accordion/progressive disclosure, reorder panels, sticky headers, minimal defaults, grouped controls, etc).
5. **Validate** Check consistency, perform lightweight heuristics (e.g. "does the first view align with the central question?"). If telemetry is available, identify the largest drop-offs in the user funnel and re-prioritize design fixes. Export a summary "rationale + intervention" report for stakeholders.

4.2. Key properties & architectural choices

- **Telemetry-optional**: `bidux` functions well without usage logs; but if metrics like page views, click events, filter application rates are present, `bidux` ingests them to spotlight bottlenecks.
- **Auditable outputs**: every design decision is stored in tidy data frames so you can trace "Why did I reorder filters?" back to a behavioral rationale.
- **Override-friendly**: users may override the default mapping of behavioral issues → UI interventions; `bidux` emphasizes transparency rather than blind automation.
- **Composable**: `bidux` is meant to be used alongside existing UI toolkits (e.g. `bslib`, `shiny.semantic`) and testing frameworks, not replace them.

5. Example Use Sketch

Here's a minimal R pipeline illustrating how `bidux` might be used in practice. (Note: this is illustrative; see package docs for full usage.)

```r
library(bidux)

plan <- bid_interpret(
  central_question = "How can we improve awareness?",
  data_story = list(
    hook = "Users are missing key messaging",
    context = "Ad campaigns are running but engagement is low",
    tension = "Too many competing visuals",
    resolution = "Simplify and prioritize key elements"
  ),
  user_personas = list(
    list(
      name = "Data Analyst",
      goals = "Needs to quickly find patterns in data",
      pain_points = "Gets overwhelmed by too many tables",
      technical_level = "Advanced"
    ),
    list(
      name = "Executive",
      goals = "Wants high-level insights at a glance",
      pain_points = "Limited time to analyze detailed reports",
      technical_level = "Basic"
    )
  )
) |>
  bid_notice(
    problem = "Too many filters",
    evidence = "Users never applied more than 2"
  ) |>
  bid_anticipate() |>
  bid_structure() |>
  bid_validate()

print(plan$rationale)
print(plan$interventions)
```

If telemetry data exists, you can inject it before `bid_structure()` to highlight drop-points in the application funnel and adjust priorities accordingly.

6. Illustrative Application

In a dashboard I built for an (undisclosed) internal research project, `bidux` guided me to (a) collapse 12 filter inputs into 4 default yes/no toggles, (b) reorder the primary outcome display to appear above controls, and (c) adopt accordion panels for secondary filters. After deployment, heatmap and click metrics (outside the purview of this paper) suggested that >50% more users applied a first filter and revisit rates rose modestly. The point is not the magnitude, but consistency of direction: you get *design with justification*, not magic.

7. Availability & Maturity

`bidux` is available under an OSI-approved MIT license on CRAN and GitHub. The repository includes function documentation, vignettes, unit tests, and contribution guidelines. Upon acceptance I will issue a curated release and archive via Zenodo, minting a software DOI.

8. Conclusion & Future Directions

`bidux` is a tool for the mid-ground: between naïve defaulted UI wrappers and full UX redesign. It embeds behavioral reasoning into the Shiny workflow so that layout choices, control counts, and content ordering aren't whimsical; they're auditable and revisitable.

Future work includes:

- Richer telemetry modules (e.g. session path mining, A/B test support)
- Integration with browser instrumentation (JavaScript hooks)
- Empirical validation (e.g. randomized UI experiments using `bidux` vs baseline dashboards)
- Support for non-Shiny front ends or hybrid R + JS dashboards

I hope `bidux` lowers the bar for better dashboards in research software and helps teams build interfaces that respect human constraints without discarding their domain logic.

9. Acknowledgements

Thanks to the Posit and Shiny community for feedback on early prototypes, and to early users who shared telemetry and use cases (with permission). I also acknowledge conversations with UX colleagues who patiently listened to dashboard horror stories.

References

Chernev, A., Böckenholt, U., Goodman, J., 2015. Choice overload: A conceptual review and meta-analysis. Journal of Consumer Psychology 25, 333–358. doi:10.1016/j.jcps.2014.08.002.

Dean, M., Ravindran, D., Stoye, J., 2022. A better test of choice overload. Working Paper / Manuscript URL: https://arxiv.org/abs/2212.03931. preprint; applied new tests showing stronger evidence of choice overload missed by earlier work.

Djamasbi, S., Siegel, M., Tullis, T., 2011. Visual hierarchy and viewing behavior: An eye tracking study, in: Human-Computer Interaction and Management: Contemporary Research and Practice. Springer, pp. 73–85.

Group, N.N., 2017. F-shaped pattern of reading on the web. NN/g Articles URL: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/.

Putkonen, A., et al., 2025. Understanding visual search in graphical user interfaces. Unpublished / preprint Preprint study of visual search in GUIs confirming scanning heuristics.