SCHARP

Paper OS22
PHUSE US-Connect 2026

Building for the Long Haul: Managing Scope, Refactoring, and CI/CD in Internal R Packages

Blazej (Moni) Neradilek (bneradil@fredhutch.org),
Chenchen Yu,
Huijun An

Statistical Center for HIV/AIDS Research and Prevention (SCHARP) at Fred Hutch, Seattle, USA

Fred Hutch Cancer Center

1 Context of our report figure tools

2 Our scope creep

3 Dev team expansion

4 Results & Impact

5 Key Takeaways

Our context

SCHARP

- SCHARP at Fred Hutch
- HIV Vaccine Trials Network, phase 1-3 trials (mostly phase 1s)
- 800+ immunological reports since 2007*
- Statistical analysts: from R beginners to 20+ years of experience

- **Needed: standardized, reliable, and effective reporting tools at scale**

Presentation focus

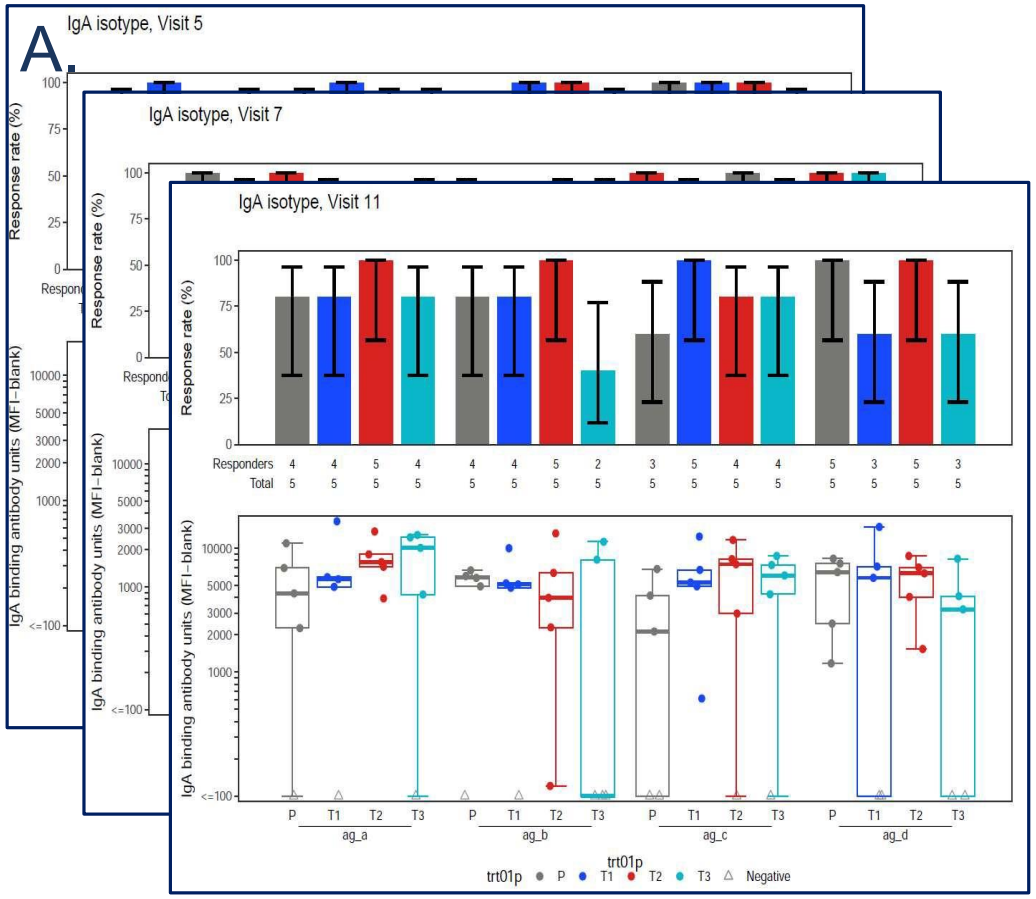
- Standard figure functions
- First in ad-hoc scripts
- Then internal packages (hvtnReports and hvtnFigures)

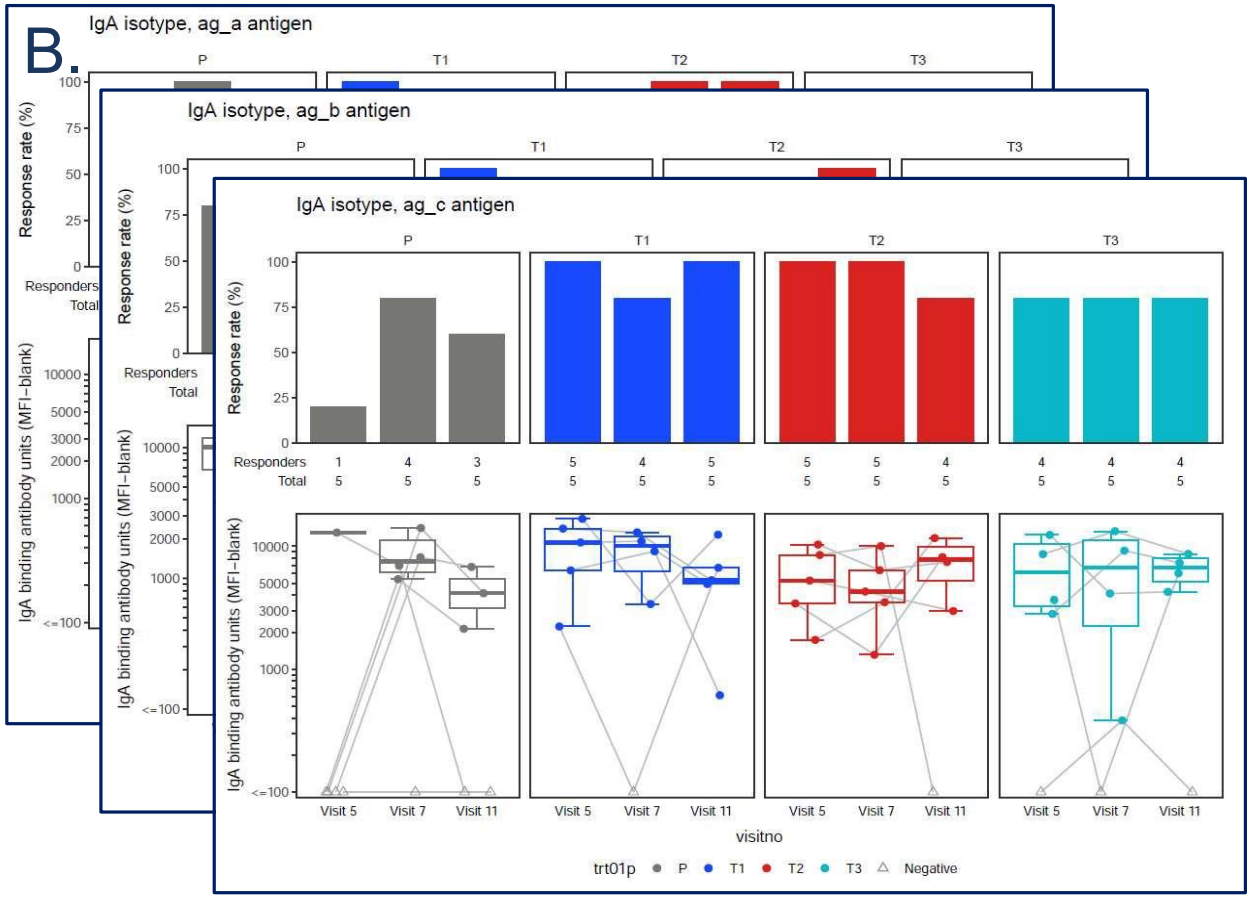
* Limited to reports that utilize our standard report figure tools

Our standard figure types

Core features: Multi-page, Box and/or line
Clustering (e.g. A)
Row faceting (e.g. B)
Aligned response and magnitude plots
Common (single) legend
Structure: *`ggpubr::ggarrange` lists (`ggplot2` embedded)*

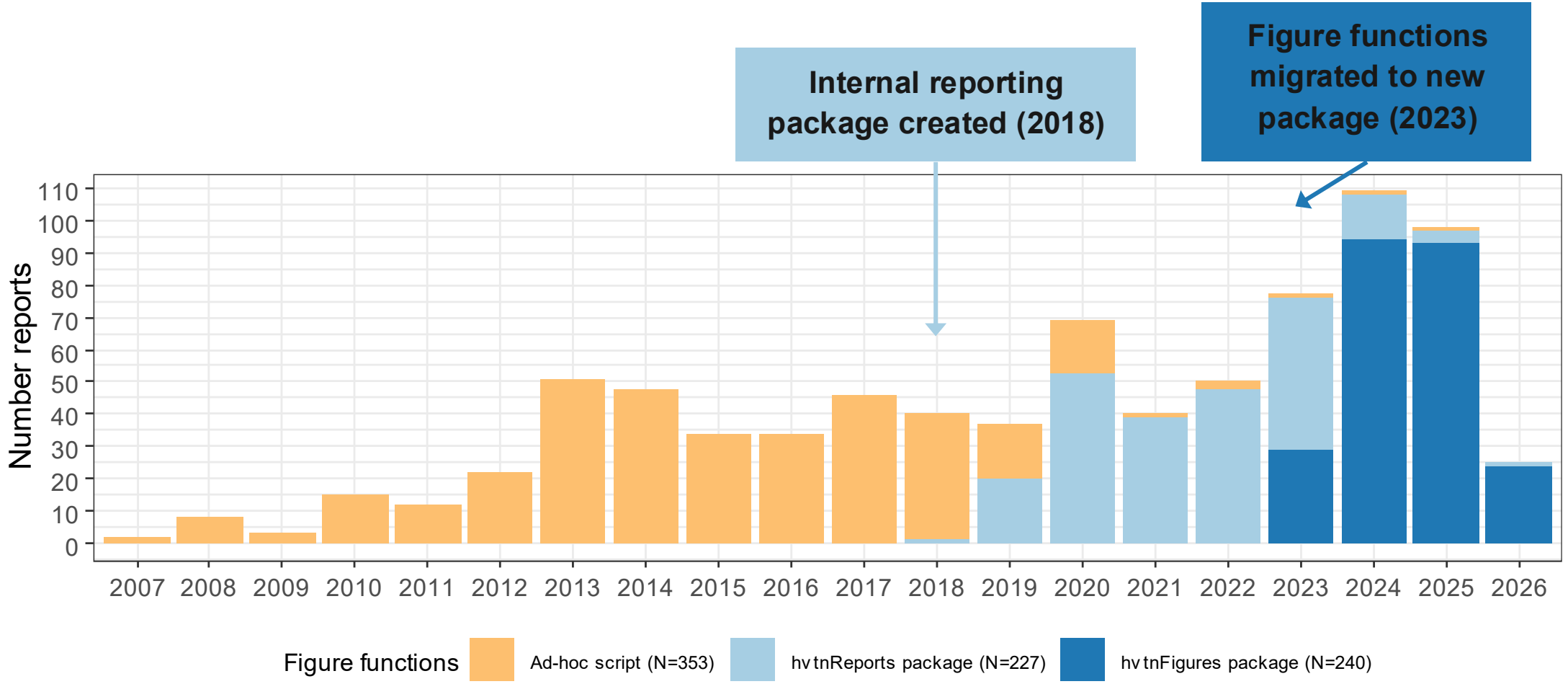
SCHARP





From ad-hoc scripts to packages

SCHARP



Why internal packages?

SCHARP

Ad-hoc scripts

- ✗ Version confusion (no easy propagation of changes)
- ✗ Namespace conflicts
- ✗ No tests

R packages

- ☑ Versioning (DESCRIPTION, Git)
- ☑ Standards (functions, documentation, tests, internal standards)
- ☑ Environment and dependency management (API boundary)
- ☑ Tests and CI/CD
- ☑ Ecosystem of tools (roxygen2, devtools, testhat, GH, …)

Scope creep
(Success breeds requests, which breeds complexity)

SCHARP

New features

- Truncated y-axes (lower and upper cutoffs)
- Colored (categorical) x-axis labels
- Aligned text showing response summary statistics
- Superimposed group means (and their confidence interval bars)
- Control which geoms (boxes, points, lines) are shown and in which order (multiple feature evolutions)
- Control drawing order of responder and non-responder points
- Outputting captions to show outside of the plot with LaTeX (next to figures and in list of figures)
- Output structured data from figures (for validation and further analysis/reporting)
- Control of boxplot widths (multiple feature evolutions)
- Additional `ggplot2` operations on figure objects

Needed Improvements

- Avoiding overlapping plot elements
- Better default font sizes
- Dropping unused factor levels (x-axis and legend variables)
- Better argument interface

Code complexity

- 600-700 lines per function
- 60+ arguments!

Expanding Team

Challenge: New features and improvements became harder given code complexity (especially for new team members)

Solution 1 - Strategic refactoring

SCHARP

- **Dedicated package** (`hvtnFigures` separation)
- **Consolidation** (`box_plot()` + `line_plot()` (repeated code) → `pt_plot()`)
- **Decomposition**
  - Helpers: `create_plot_data()` (data wrangling) + `create_plot()` (1 page)
  - Geom-specific functions (layers)
  - 600+ code lines (1 function) → 11-310 lines (6 functions)
  - (More in progress – e.g. reducing argument load)
- **Extensibility**
  - Decompose/recompose functions → ggplot2-style extensibility
  - Functional architecture enables user customization

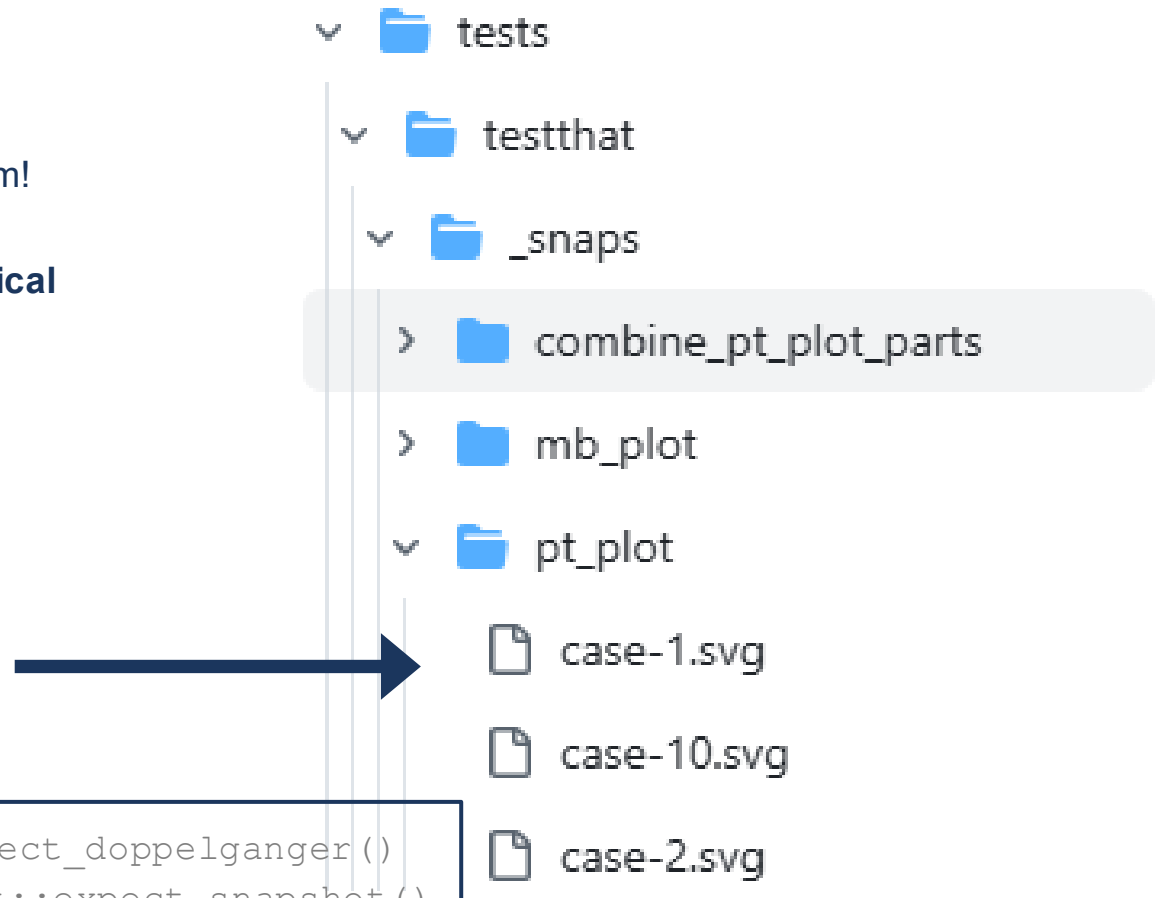
Solution 2 – Unit tests for figures

SCHARP

- Initially
  - Visual inspection of tests in PDF
  - Compare summaries vs. tables
  - Didn't scale well with more features and a larger team!
- More rigorous CI/CD needed
  - **`{vdiffr}` (Visual Regression Testing and Graphical Diffing)**
  - GitHub Actions

`test-pt_plot.R`

```
test_that(
  "Typical PT report boxplot",
  {case <- pt_plot(dat=lum…, test.pages = 1)
   expect_doppelganger("Case 1", case)
  })
…
```



`vdiffr::expect_doppelganger()`
(like `testthat::expect_snapshot()`
but for SVG images instead of R object)

Solution 3: From Solo Developer to Team

SCHARP

Challenge

- **Single developer (2018-2022) → six developers (2025)**
- New developers: range of R package dev experience
- **R package dev skills**
  - Mindset shift: analysis → engineering
  - Package structure
  - Package tools (e.g. `devtools`, `usethis`, `roxygen2`)
  - Dependencies and namespace
  - Testing
  - Documentation
  - Git and GitHub proficiency
- **Complexity**

Solution

Skills/Learning

- R package book (Wickham and Bryan, 2023)
- Weekly team learning of Git (3.5 months)
- Mentoring
- Gradual contribution "ladder": documentation, code comments → easier/new functions → core functions

Technical improvements

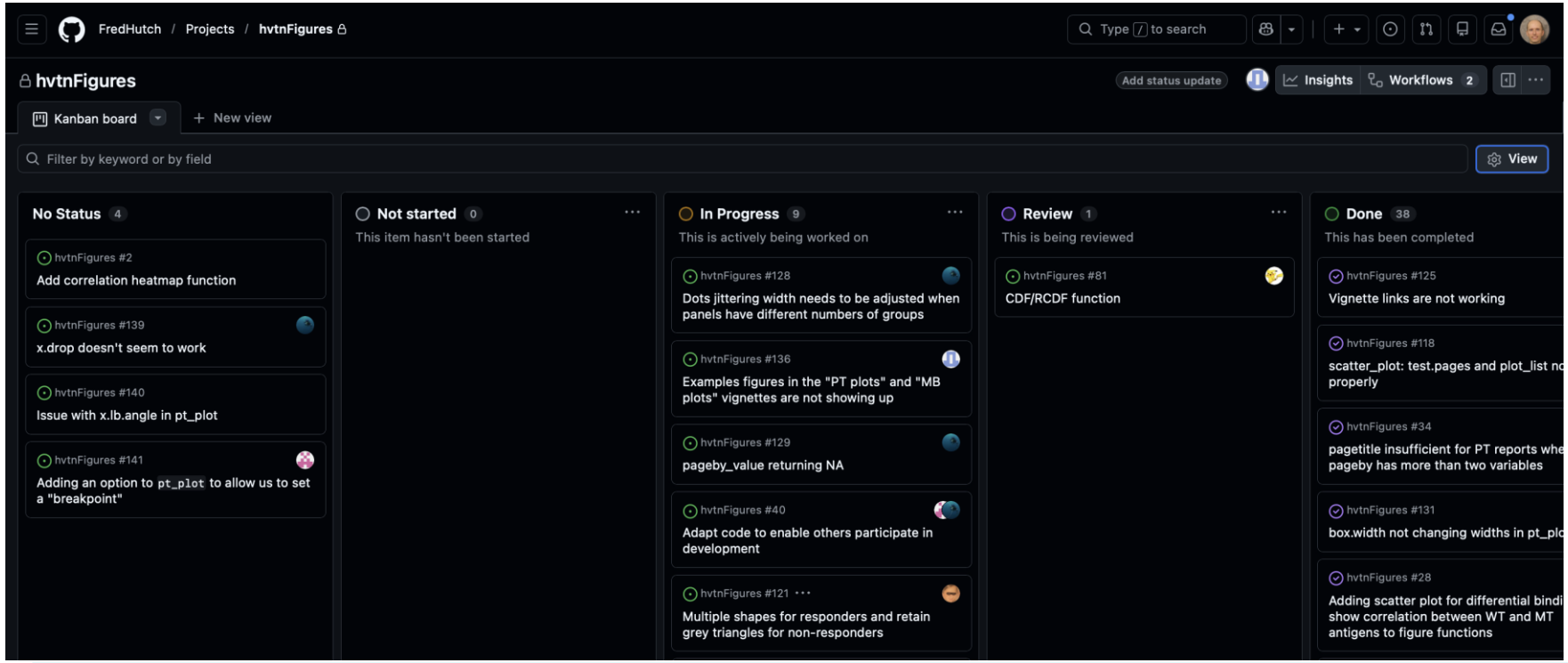
- Improved code comments and naming
- Developer vignette (more complex aspects of function designs)
- Refactoring (incl. code readability)
- Testing

Solution 3: From Solo Developer to Team

SCHARP

Project management

- Huddles (all devs – weekly, core – 1x/2 weeks)
- GitHub Kanban (Projects view for issues) has been especially useful



Results & Impact

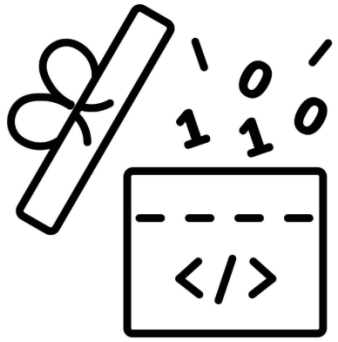
SCHARP

- 70-110 reports/year in each of last three years
- 1 → 6 contributors
  - % PRs by package creator: 92% in 2022-2023 → 49% in 2025
  - % issues assigned to package creator: 89% in 2022-2023 → 33% in 2025
- New features and improvements (faster implementation)
  - 15 PRs in 2022 → 37 PRs in 2025
- Better user-developer feedback
- Growth opportunity for both new and original developers

Key Takeaways

SCHARP

Technical



- **Start with packages early** (if you have reusable patterns)
- **Refactor proactively** (before complexity becomes unmanageable)
- **Design for extensibility** (functional architecture)
- **Automate testing** (`vdiffr` for plots)
- **Version rigorously** (map commits to releases)

Team

- **Users as developers** (domain knowledge + motivation)
- **Invest in new developers** (documentation, mentorship, learning groups)
- **Start small** (documentation → simple functions → core contributions)
- **Manage transparently** (GitHub issues/Kanban, regular huddles)
- **Balance innovation and stability** (CI/CD + version releases)

Thank you!

SCHARP

Acknowledgements

- Anthony Williams (original `hvtnReports` creator)
- Shannon Grant (liaison between report stakeholders and dev team; original `hvtnReports` author)
- Shiyu Chen, Jia Jin Kee, Kevin Gillespie (most recent `hvtnFigures` contributors)

Contact: Blazej (Moni) Neradilek, bneradil@fredhutch.org