safetyGraphics Techincal Intro

2022-04-15

Agenda

- Intros
- Technical Intro
  - Contributor Guidelines
  - GitHub Tour
  - Package Ecosystem
- Subteams
- Next Steps
- Q&A

Contributor Guidelines

https://github.com/SafetyGraphics/SafetyGraphics.github.io/blob/master/CONTRIBUTING.md

Interactive Safety Graphics Contribution Guidelines

This page is under development. We'd love your help making it better - PRs with updates welcome!

Overview

This page provides guidelines for technical contributors to the ASA/DIA Interactive Safety Graphics (ISG) working group. For a non-technical overview, check out our home page. You can also always ask us questions or comments by posting to the safetyGraphics discussion.

Getting Started

Hello! Thank you for your interest in contributing to this project! Here's a quick guide to getting started:

1. **Review the Technical Requirements** - Most of our programming is done in R and Javascript, and we assume a basic understanding of GitHub and Git. We've provided a few links to a few technical resources and tutorials below.
2. **Read about the Project** - Learn a bit about safetyGraphics by reviewing the rest of this document and checking out the Vignettes, Papers and Presentations listed in the Resources Section below.
3. **Join the Mailing List** - Take 2 minutes to fill out this form to let us know you're interested; we'll reach out to discuss.
4. **Start Contributing** - Look through the Good First Issues to find a topic to work on. You can also post to the safetyGraphics discussion board to discuss options. Once you find a topic, you can get to work on your first Pull Request.

Technical Overview

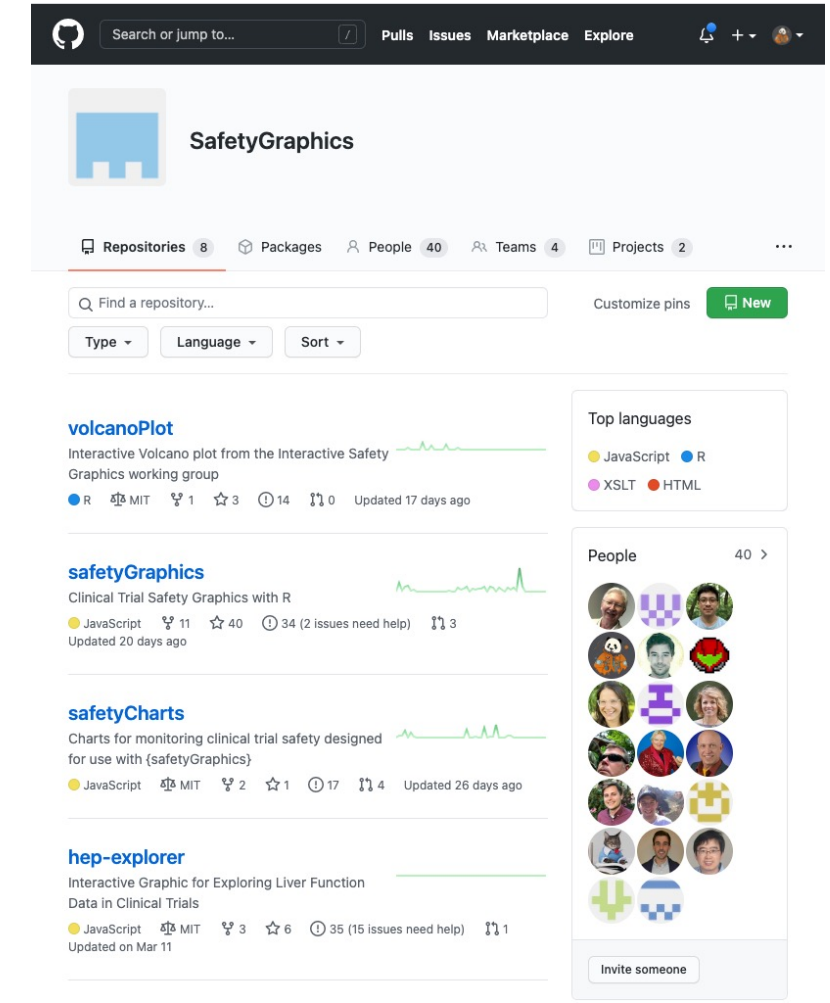
The safetyGraphics project consists of several GitHub repositories that are maintained in the safetyGraphics GitHub organization. The safetyGraphics package provides a platform for loading data, customizing settings and creating interactive graphics for monitoring clinical trial safety. However, the safetyGraphics platform really doesn't do much by iteslf! In fact, none of the content on the Charts tab is actually found in the safetyGraphics package; the default charts live in the safetyCharts package. safetyCharts has over a dozen charts that are configured to work with {safetyGraphics}, but can also easily be used independently. This vignette provides describes this relationship in more detail along with extensive technical details about the creation of custom renderers.

Open and Transparent Online Collaboration

https://github.com/SafetyGraphics

We use a public GitHub site for:

- Source Code Storage
- Version Control
- Testing/QC infrastructure
- Issue tracking
- Release Tracking
- Discussion boards
- Meeting Minutes
- Project Management (Kanban boards, etc.)
- Web-site hosting
- Continuous Integration



safetyGraphics Package Ecosystem

The {safetyGraphics} platform displays charts from {safetyCharts}.

- {safetyGraphics} – Reusable Shiny App for safety monitoring
- {safetyData} – Sample AdAM and SDTM datasets for examples and testing.
- {safetyCharts} – Reusable stand-alone charts for safety monitoring
- Domain-specific Charts – {BRForestPlot}, {qtexplorer}, {volcanoPlot}

safetyGraphicsApp() – Configurable Inputs

Study-Specific Inputs

- domainData – Domain-level Study Data
- mapping – List identifying the key columns/fields in your data

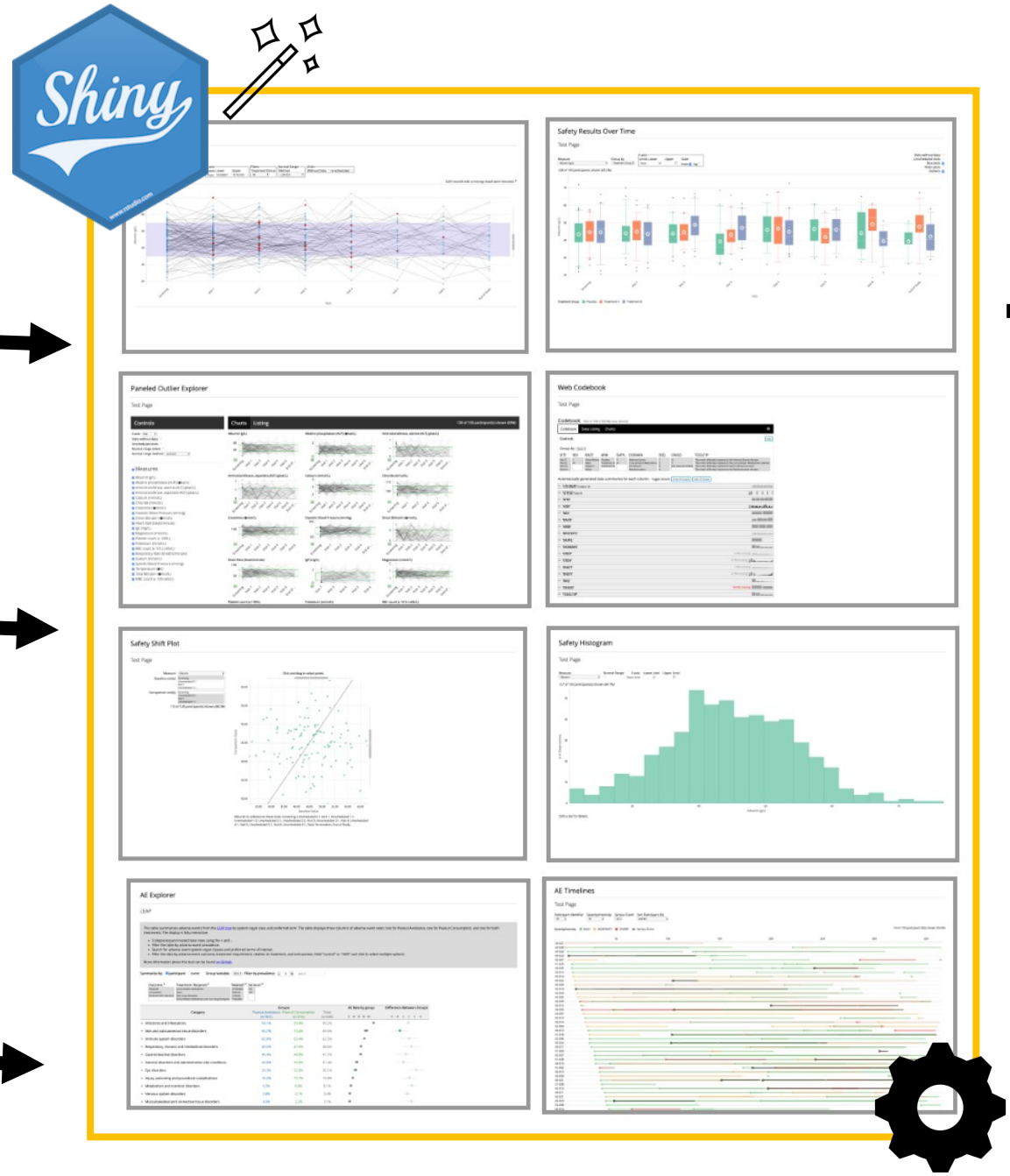
General Inputs used across multiple studies

- charts – Define the charts used in the app.
- meta – Metadata table with info about required columns and fields

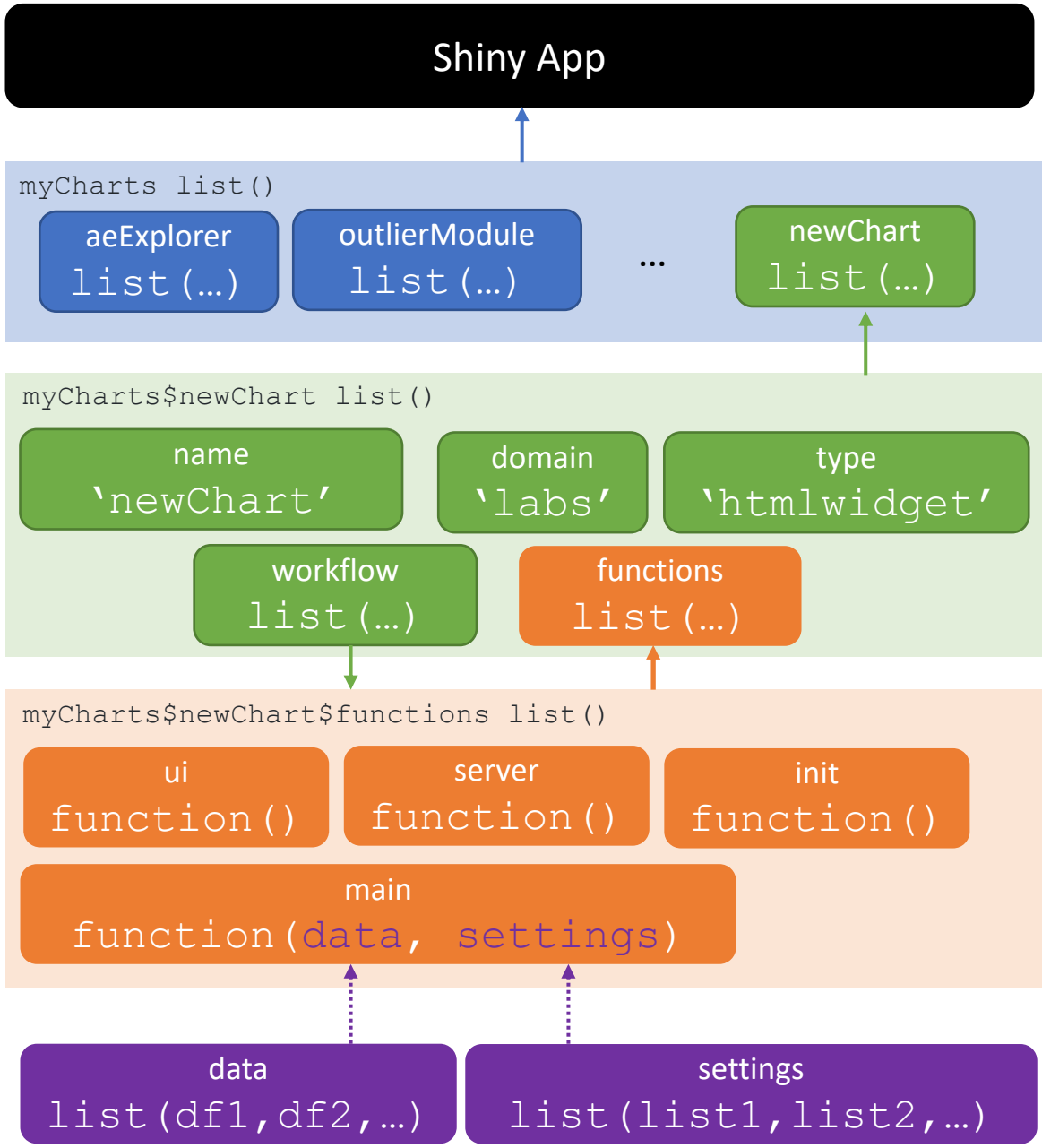
Domain Data

Charts

Metadata + Mapping



Interactive Graphics




```
# Start safetyGraphics Shiny App with custom charts
safetyGraphicsApp(charts=myCharts)
```

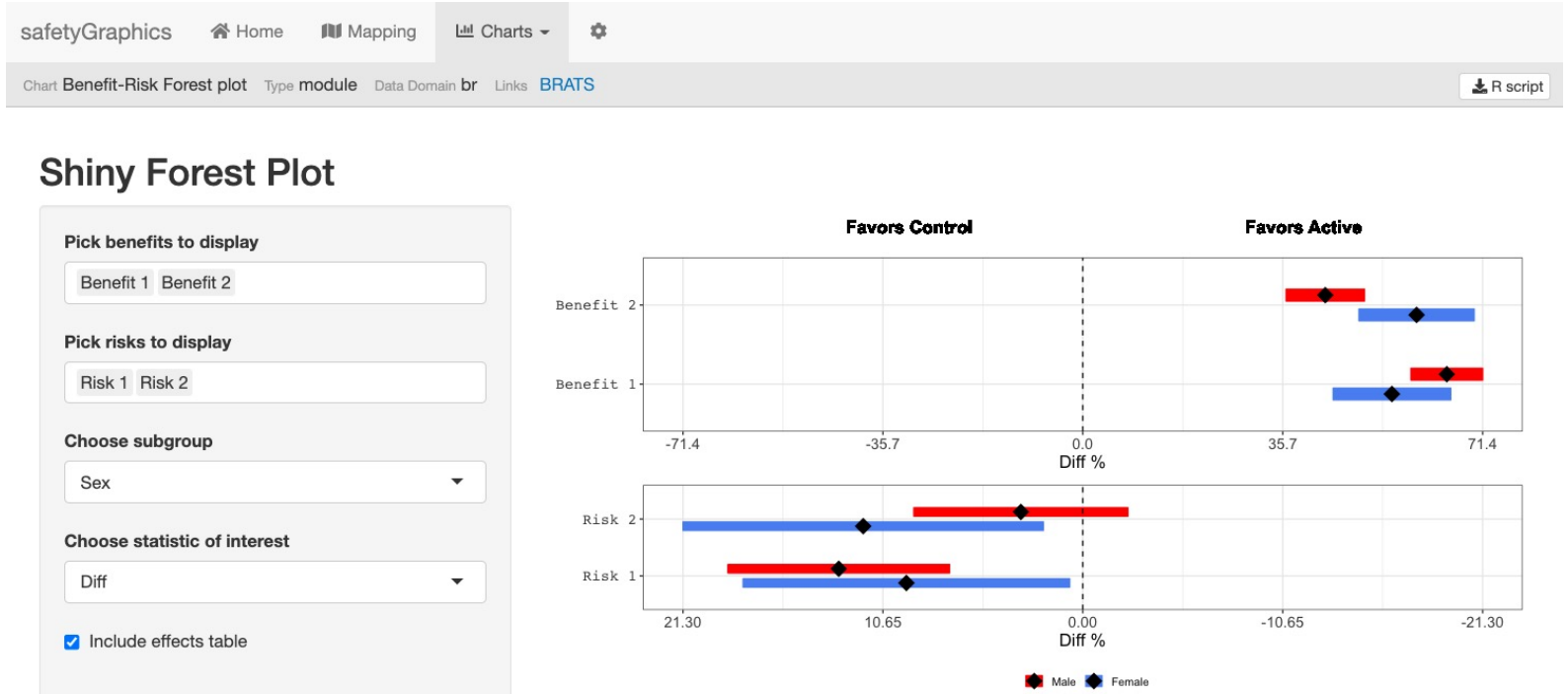
```
# Make list of all charts in specific location
# Specify 'path' or 'package' - 'safetyCharts' = default
myCharts<-makeChartConfig()
```

```
# Add a Single Chart (usually saved as yaml)
# Automatically called for yaml in 'path'/'package'
myCharts$newChart<-read_yaml(path)
```

```
# Bind list of functions to a chart
# Specified by chart$workflow, chart$type & chart$package
# Automatically called for yaml in 'path'/'package'
myCharts$newChart<-
prepareChart(myCharts$newChart)
```

```
# Data and Setting inputs used in main() and init()
# reactive() in shiny app, but not in exported code.
```

{safetyGraphics} v2.1 simplifies chart import



		Treatment Comparison		Active			
Benefit/Risk	**Sex**	**Diff %**	**95% CI**	**n**	**N**	**Raw %**	
Benefit 1	Female	55.2	(44.7, 65.7)	75	115	65.2	1
Benefit 1	Male	65	(58.6, 71.4)	225	300	75	2
Benefit 2	Female	59.6	(49.3, 69.9)	80	115	69.6	1
Benefit 2	Male	43.3	(36.3, 50.3)	160	300	53.3	2
Risk 1	Female	9.4	(0.7, 18.1)	20	115	17.4	
Risk 1	Male	13	(7.1, 18.9)	63	300	21	1
Risk 2	Female	11.7	(2.1, 21.3)	25	115	21.7	1

```
BRDemo<- function(data=BRForestPlot::demo){
   br_chart <- makeChartConfig(packages='BRForestPlot')
   safetyGraphicsApp(
     domainData=list(br=data),
     charts=br_chart
    )
}
```

safetyGraphics – Domain-specific Subteams

- QT/ECG Recruiting!
- Benefit Risk Recruiting!
- Hepatic Explorer (js)
- Nephrotoxicity Explorer
- AE Volcano plot

Next Steps

- Attend the next full team meeting on 6/17
- Attend the technical team meeting in May (Date TBD)
- Send me (jwildfire@gilead.com) the following:
  - Your GitHub Username
  - Subgroup preference (QT? Risk-Benefit??? Other?)