Block by block: An introduction to {blockr}

Karma Tarap (Bristol Myers Squibb)

Section 1

1. Why We Built {blockr}

1. Why We Built {blockr}

At Bristol-Myers Squibb (BMS), we faced several challenges in developing Shiny applications:

- **Inconsistent R Capabilities**
- **Duplication of Effort**
- **Styling and Consistency**
- **Scalability and Extensibility**
- **App Persistence**
- **Validation Hurdles**

Section 2

2. What is {blockr}?

2. What is {blockr}?

{blockr} is an R package designed to democratize data analysis by providing a flexible, intuitive, and code-free approach to building data pipelines.
It allows users to create powerful data workflows using pre-built blocks that can be easily connected, all without writing a single line of code.

Section 3

3. {blockr} 101

3.1 Demographic Analysis in Clinical Trials

How are demographics distributed across treatment groups?

```
data(adsl)

adsl |>
  filter(!is.na(TRT01P) & !is.na(SEX)) |>
  ggplot(
    aes(
      x = TRT01P,
      fill = SEX
    )
  ) +
  geom_bar(position = "fill") +
  labs(
    x = "Treatment",
    y = "Proportion",
    fill = "Sex",
    title = "Demographic Distributions by Treatment",
    subtitle = "Distribution of Sex by Treatment Group"
  ) +
  theme_minimal()
```

3.2 Demographic Analysis in Clinical Trials (cont.)

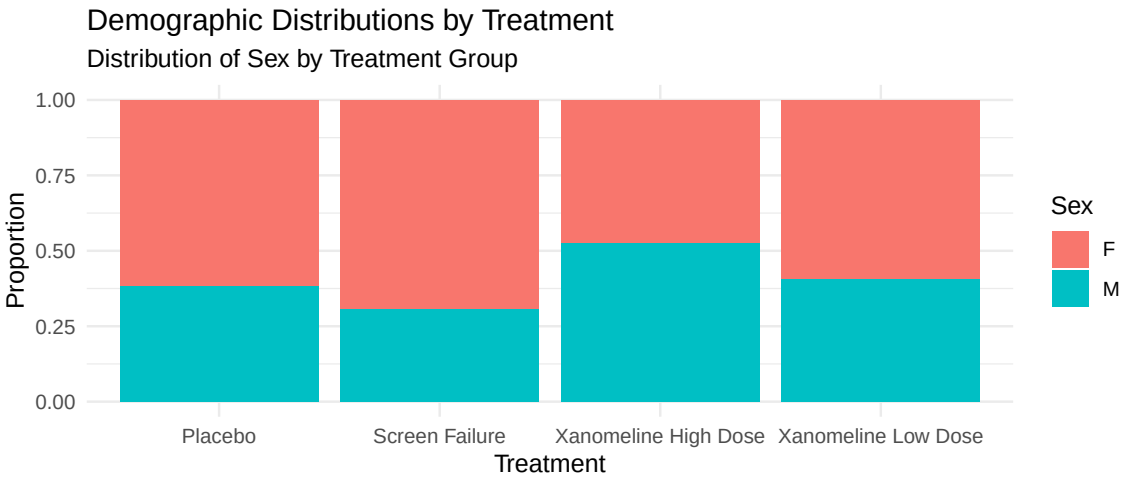
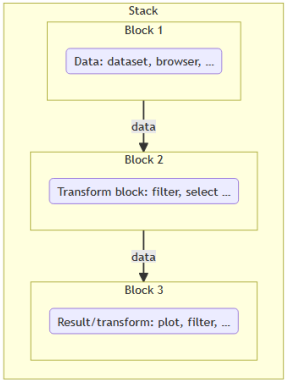


Figure 1: Distribution of Sex by Treatment Group

3.3 The stack: a data analysis recipe



- Collection of instructions, **blocks**
- Blocks can be varied: from **data import** to **wrangling/visualization**.

3.4 Demo

3.5 How much code would it take with Shiny?

```
library(shiny)
library(bslib)
library(ggplot2)
library(pharmaverseadam)
library(dplyr)

data(adsl)

shinyApp(
  ui = page_fluid(
    layout_sidebar(
      sidebar = sidebar(
        selectInput(
          "trt",
          "Treatment Variable",
          choices = c("TRT01P", "TRT01A"),
          selected = "TRT01P"
        ),
        selectInput(
          "demographic",
          "Demographic Variable",
          choices = c("SEX", "RACE", "ETHNIC"),
          selected = "SEX"
        )
      ),
      plotOutput("plot")
    )
  ),
  server = function(input, output, session) {
    ...
```

3.6 How much code would it take with Shiny? (cont.)

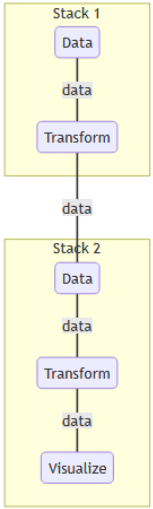
```
...
  server = function(input, output, session) {

    output$plot <- renderPlot({
      adsl |>
        filter(!is.na(!!sym(input$trt)) & !is.na(!!sym(input$demographic))) |>
        ggplot(aes(x = !!sym(input$trt), fill = !!sym(input$demographic))) +
        geom_bar(position = "fill") +
        labs(
          x = "Treatment",
          y = "Proportion",
          fill = input$demographic,
          title = paste("Distribution of", input$demographic, "by", input$trt),
          subtitle = "Demographic Distribution by Treatment Group"
        ) +
        theme_minimal()
    })
  }
)
```

3.7 It's much easier with blockr

```
library(blockr)
new_stack(
  data_block = new_dataset_block("adsl", "pharmaverseadam"),
  filter_block = new_filter_block("TRT01P", "is.na", FALSE),
  filter_block2 = new_filter_block("SEX", "is.na", FALSE),
  plot_block = new_ggplot_block("TRT01P", fill = "SEX"),
  layer_block = new_geombar_block(position = "fill")
)
serve_stack(stack)
```

1. Create the stack.
2. Import data.
3. Filter out missing values.
4. Set up the plot structure.
5. Add a bar layer with fill position.
6. Serve a Shiny app.

3.8 Connecting stacks: The workspace



- A *workspace* is a collection of independent or interdependent stacks
- *Special* blocks allow references to existing blocks
- This allows for the creation of sophisticated linear pipelines

3.9 How do I create a workspace?

```
library(blockr)
# Creates an empty workspace
set_workspace(                                    (1)
  stack_1 = new_stack()                           (2)
  stack_2 = new_stack()
)
serve_workspace()                                 (3)
```

(1) Initialise.
(2) Optional: add stacks.
(3) Serve Shiny app.

Section 4

4. Create your own blocks supermarket

4.1 Today's mission

Create a new **lm()** block:

```
lm(height ~ weight, data = adsl)
```

4.2 Create new blocks: lm block 1/4

```
new_lm_block <- function(y = character(), predictor = character(), ...) {
}
```
①

① Create the `constructor`.

4.3 Create new blocks: lm block 2/4

```
new_lm_block <- function(y = character(), predictor = character(), ...) {

  all_cols <- function(data) colnames(data)

  fields <- list(
    y = new_select_field(y, all_cols, type = "name"),
    predictor = new_select_field(predictor, all_cols, type = "name")
  )

  new_block(
    fields = fields,

  )
}
```

① ② ③

② Construct columns dynamically.

③ Add field(s): **y** and **predictor** columns (`type` allows to pass in cols as name instead of strings.)

4.4 Create new blocks: lm block 3/4

```
new_lm_block <- function(y = character(), predictor = character(), ...) {

  all_cols <- function(data) colnames(data)

  fields <- list(
    y = new_select_field(y, all_cols, type = "name"),
    predictor = new_select_field(predictor, all_cols, type = "name")
  )

  new_block(
    fields = fields,
    expr = quote({
      model <- lm(data = data, formula = .(y) ~ .(predictor))
      broom::tidy(model)
    }),

  )
}
```

④

④ Provide expression: use `quote` and pass field name with `.(field_name)`.

4.5 Create new blocks: lm block 4/4

```
new_lm_block <- function(y = character(), predictor = character(), ...) {

  all_cols <- function(data) colnames(data)

  fields <- list(
    y = new_select_field(y, all_cols, type = "name"),
    predictor = new_select_field(predictor, all_cols, type = "name")
  )

  new_block(
    fields = fields,
    expr = quote({
      model <- lm(data = data, formula = .(y) ~ .(predictor))
      broom::tidy(model)
    }),
    ...,
    class = c("lm_block", "transform_block")
  )
}
```

5. Give it the correct classes: `transform_block` + a custom class.

4.6 The registry: the blocks supermarket

- **Information** about blocks.
- **Shared** between block packages.

4.7 Filling the supermarket with block

```
register_lm_block <- function(pkg) {
  register_block(
    constructor = new_lm_block,
    name = "lm block",
    description = "Create a linear model block",
    classes = c("lm_block", "transform_block"),
    input = "data.frame",
    output = "data.frame",
    package = pkg
  )
}

# Put in zzz.R
.onLoad <- function(libname, pkgname) {
  register_lm_block(pkgname)
  invisible(NULL)
}
```

4.8 Other blockr examples

Section 5

5. We need you!

5.1 Getting started

1. Install

```
pak::pak("blockr-org/blockr")

library(blockr)
serve_stack(new_stack())
```

2. Read the doc at https://blockr-org.github.io/blockr/index.html.
3. Enjoy!

5.2 Use blocks and build dashboards

Share dashboards with your teams to **speed up** data analysis

5.3 Create blocks to help your data scientists

You're an **advanced** R developer, you can **extend** blockr!

5.4 Our team

Karma Tarap	John Coene	Nicolas Bennett, Christoph Sax, David Granjon
Bristol Myers Squibb	The Y Company	cynkra
https://www.bms.com/	https://the-y-company.com/	https://cynkra.com/