A Short Introduction to R

R, RStudio, and User-Written Libraries

Introduction to R

- **Definition of R:** R is an open-source language and environment focused on statistical computing and graphics. It is well-suited for data manipulation, exploration, and advanced analytics, making it a natural choice for experimental design.(Mannarswamy et al. 2010)
- **Interpretation and Scripting:** R can be used interactively at the console or through scripts, facilitating reproducible research.
- **Data Structures:** R supports a variety of data structures, including vectors, matrices, data frames, and lists.

RStudio IDE

- **Overview of RStudio:** RStudio is an Integrated Development Environment (IDE) that simplifies coding in R. It provides a console, source editor, environment tab, and plots pane in one interface.
- **Project-Based Workflow:** Encourages best practices: keep project files, scripts, and data together. This is especially helpful for organizing experimental design projects.

Installing and Using Packages

- **CRAN and Other Repositories:** R packages are typically installed from CRAN. Some specialized packages may come from GitHub or other repositories.
- **User-Written Libraries:** Researchers often share R libraries for custom functions. For instance, specialized DOE packages or domain-specific modeling tools.
- **Installing/Loading Packages:** Use `install.packages("<pkgname>")`, then `library(<pkgname>)` to load them. This makes functions available for use in your scripts.

```r
# Example: Installing and loading a package
# Example: Installing and loading a package
install.packages("AlgDesign")  # from CRAN
library(AlgDesign)

# Installing from GitHub requires the 'remotes' package:
# install.packages("remotes")
# remotes::install_github("someUser/someRepo")
```

Data Types

Basic Data Types

- **Numerical:** Floating-point (e.g., `3.14`) or integer (e.g., `1L`). Often represent factor levels.
- **Character:** Strings used for labeling treatments or levels.
- **Logical:** Boolean values (TRUE/FALSE) for conditions and subsetting.

Data Structures

- **Vectors:** One-dimensional collections of elements of the same type.
- **Matrices:** Two-dimensional collections of the same type, crucial for design matrices in modeling.
- **Lists:** Ordered collections that can contain different data types (e.g., model objects).
- **Data Frames / Tibbles:** Tabular data with named columns; in the tidyverse, tibbles (tbl_df) are the default.

```r
library(tibble)

# Creating a tibble

df <- tibble( run_id = 1:5, factorA = c(-1, -1, 0, 1, 1), factorB = c(0, 1, -1, 0, 1), respo

print(df)
str(df)

# Changing the type of a column

df$response <- as.integer(df$response) df
```

Reading and Writing Data

Basic I/O Functions

Reading Data:
- Base R: `read.csv()`, `read.table()`, `readRDS()`.
- Tidyverse readr: `readr::read_csv()`.
Writing Data:
- Base R: `write.csv()`, `saveRDS()`.
- Tidyverse readr: `readr::write_csv()`.

Working Directory and Paths

Working Directory:
- `getwd()` shows the current directory; `setwd("path/to/dir")` sets it.
- Use RStudio Projects for automatic path management.

Relative vs. Absolute Paths:
- `file.path()` constructs paths; `here::here()` is useful for project paths.
- Helps keep projects organized and reproducible.

```
# Reading a CSV file using readr
library(readr) design_data \<- read_csv("my_design_data.csv")

# Writing a CSV file
write_csv(design_data, "my_output.csv")
```

Operations with Vectors and Matrices

Vectorized Arithmetic

- **Element-wise Operations:** +, -, *, / apply element by element.
- **Recycling Rules:** If vectors differ in length, the shorter one is recycled (be mindful of unintended consequences).

Matrix Operations

- **Creation:** `matrix(data, nrow, ncol)`, `rbind()`, `cbind()`.
- **Multiplication:** `%*%` for matrix multiplication (vital for computing X XX X in experimental design).
- **Transpose/Inverse:** `t(M)` and `solve(M)`.

```
# Vector operations
x <- 1:5
y <- c(2, 4, 6, 8, 10)

x + y # element-wise addition
x * y # element-wise multiplication

# Matrix operations
A <- matrix(1:4, nrow = 2, ncol = 2)
B <- matrix(c(2, 0, 0, 2), nrow = 2, ncol = 2)

A %*% B # Matrix multiplication
solve(B) # Matrix inverse
```

Logical Operators

Basic Logical Operators

- **Comparison Operators:** `<`, `>`, `<=`, `>=`, `==`, `!=`.
- **Logical Operators:** `&`, `|`, `!` for AND, OR, NOT.

Vectorized Comparisons

Operations happen element-by-element. Use `any()` or `all()` to combine results.

Subset Selection

- **In base R**, `[ ]` or `subset()`.
- **In the tidyverse**, `dplyr::filter()` is frequently used.

```r
# Vector comparisons
v <- c(1, 3, 5, 2, 4)
v > 2

# Subset selection with dplyr
library(dplyr)
filtered_data <- df %>%
  filter(factorA > 0 & response >= 15)
filtered_data
```

Base R Graphics

Base Plotting System

- `plot()`: Quick scatterplots; set arguments like `xlab`, `ylab`, `main`.
- **Add Lines/Points:** `lines()`, `points()`, `abline()`, etc.

Histograms, Bar Plots, Boxplots

- `hist()`: Frequency histograms.
- `barplot()`: For categorical or factor-based data.
- `boxplot()`: Summaries of numeric data distributions.

```r
# Example base R plot
plot(df$factorA, df$response,
     xlab = "Factor A Level",
     ylab = "Response",
     main = "Response vs. Factor A")
abline(h = mean(df$response), col = "red", lty = 2)
```

Selected R Libraries (plot3D, mix.DOE, and an Optimization Library)

plot3D

- **Purpose:** Functions like `scatter3D()`, `surf3D()`, and `contour3D()` for 3D visualization.

- **Use Case:** Plotting 3-factor response surfaces or 2-factor surfaces plus response dimension.

```
# Demo: 3D surface plotting

library(plot3D)
x <- seq(-1, 1, 0.1)
y <- seq(-1, 1, 0.1)
grid <- expand.grid(x = x, y = y)

# Sample quadratic response

z_vals <- with(grid, 15 + 3*x - 2*y - x\^2 - y\^2 + 2*x*y)
z_mat <- matrix(z_vals, nrow = length(x), ncol = length(y))

persp3D(x = x, y = y, z = z_mat,
        xlab = "X", ylab = "Y", zlab = "Response")
```

mix.DOE

- **Purpose:** Tools specialized for mixture designs (e.g., simplex-lattice, simplex-centroid).
- **Integration:** Often used with candidate sets for mixture experiments in combination with other DOE packages.

Optimization Library (Example: AlgDesign or skpr)

- **AlgDesign:** Classic package with `optFederov()` for constructing D-, I-, or other optimal designs.
- **skpr:** Provides coordinate-exchange algorithms for D-, I-, A-, and G-optimal designs. Also includes evaluation functions and FDS plots.
- **Workflow:**
  –Create a candidate set (possibly via tidyverse).
  –Run optimization function.
  –Evaluate and visualize the resulting design.

```r
# Minimal example using AlgDesign for an I-optimal design

library(AlgDesign)
candidate_set <- data.frame( x1 = c(-1, -1, 0, 1, 1),
                             x2 = c(0, 1, 0, 1, 0))

opt_result <- optFederov( frml = ~ x1 + x2 + I(x1^2) + I(x2^2) + x1:x2,
                          data = candidate_set,
                          nTrials = 5,
                          criterion = "I" )
opt_result$design
opt_result$I
```

Summary & Transition

In this chapter, you learned the essential R concepts and tools that will underpin all subsequent chapters on I-optimal designs. You now have a foundation in:

- **R basics** (IDE, package management, data types, data I/O)
- **Data structures** (vectors, matrices, tibbles)
- **Core operations** (vectorized arithmetic, matrix multiplication)
- **Graphics** (base plots, plus a glance at 3D visualizations)
- **Key libraries** (including plot3D, mix.DOE, and an optimization package for DOE)
  “‘

Mannarswamy, Aravind, Stuart H. Munson-McGee, Robert Steiner, and Charles L. Johnson. 2010. “Optimal Designs for Constant-Heating-Rate Differential Scanning Calorimetry Experiments for Polymerization Kinetics:*n*th-Order Kinetics.” *Journal of Applied Polymer Science* 117 (4): 2133–39. https://doi.org/10.1002/app.31406.