

Grid 2050 – User Manual

Sarah Troise^a, M. Granger Morgan^a

^aCarnegie Mellon University Department of Engineering and Public Policy

Abstract

This paper introduces Grid2050, a web-based decision support framework that allows users to build and assess US low-carbon electricity generation portfolios for 2050. Although base assumptions from various sources are integrated in the tool to support a default portfolio mix, the framework encourages users to alter assumptions about the techno-economic performance and socio-political acceptance of different technologies to create their own portfolios. To support that process, the framework includes built-in analytics to estimate deviations from base assumptions, a capability to explore alternative assumptions about technological learning and scale-up, and the premiums that users are willing to incur to avoid or deploy different technologies. This suite of features offers a way to benchmark the consistency and feasibility of users' energy system preferences. Its use should help inform energy system planning studies and regulatory interventions to bend the global warming curve.

Table of Contents

| | |
|---|----------|
| 1. System Requirements and Licenses..... | 3 |
| 2. Download and installation..... | 4 |
| 3. Guide for Using R Shiny Apps..... | 5 |
| 3.1 Getting Started: | 5 |
| 3.2 Structure of a Shiny Application: | 5 |
| 3.3 Running a Shiny Application Locally: | 5 |
| 3.4 Deploying a Shiny Application:..... | 6 |
| 3.5 Best Practices and Tips: | 6 |
| 3.6 Additional Resources: | 7 |
| 4. Possible Customizations..... | 8 |
| 4.1 Adding a new technology | 8 |
| 4.2 Using as a regional tool | 9 |

1. System Requirements and Licenses

Grid2050 can be found for web use at this [link](#). To edit and customize Grid2050 to fit a specific project, the following software tools are required:

- **R.** The Grid2050 tool is coded in R, so to work with the code, R is required. Grid2050 was built using R version 4.3.1, but it is programmed to be compatible with older and newer versions. Below is a table of all of the dependencies and the versions used to develop the tool.

| Dependency | Version |
|-----------------------|---------|
| Base R | 4.3.1 |
| shiny | 1.7.4.1 |
| shinydashboard | 0.7.2 |
| ggplot2 | 3.4.4 |

- **HTML.** Several input files for the Grid2050 tool are written in HTML.

2. Download and installation

This section describes how to download and install our code from GitHub. To get started follow these steps:

1. Navigate to the GitHub repository containing the tools code (<https://github.com/troises19/Grid2050>)
2. Once there, locate the "Code" button, usually displayed in green near the top right of the repository page. Click on the dropdown arrow next to it, and select "Download ZIP." This action will initiate the download of a compressed zip folder containing the entire codebase of the tool.
3. After the download is complete, locate the downloaded zip file on your computer and extract its contents to a folder of your choice.
 - a. RStudio is recommended to work with the files

Please refer to the next section about how to navigate and explore R Shiny applications

3. Guide for Using R Shiny Apps

R Shiny is a powerful web application framework for R, designed to make it easy to build interactive web applications directly from R. This user manual will guide you through the basic steps of creating and deploying a Shiny application, which will help you get everything out of Grid2050. To view and use Grid2050 as only a webpage, go to this link (<https://troises19.shinyapps.io/Grid2050/>). The rest of this section is for if you wish to edit and customize the Grid2050 system

3.1 Getting Started:

Before you start building Shiny applications, make sure you have R and RStudio installed on your system. You can download and install R from <https://cran.r-project.org/> and RStudio from <https://www.rstudio.com/products/rstudio/download/>. Additionally, ensure you have the Shiny package installed by running `install.packages("shiny")` in your R console.

3.2 Structure of a Shiny Application:

A Shiny application consists of two main components: the UI (User Interface) and the server logic. In Grid2050, these components are in the same file, but you can also build shiny apps with them in two separate files.

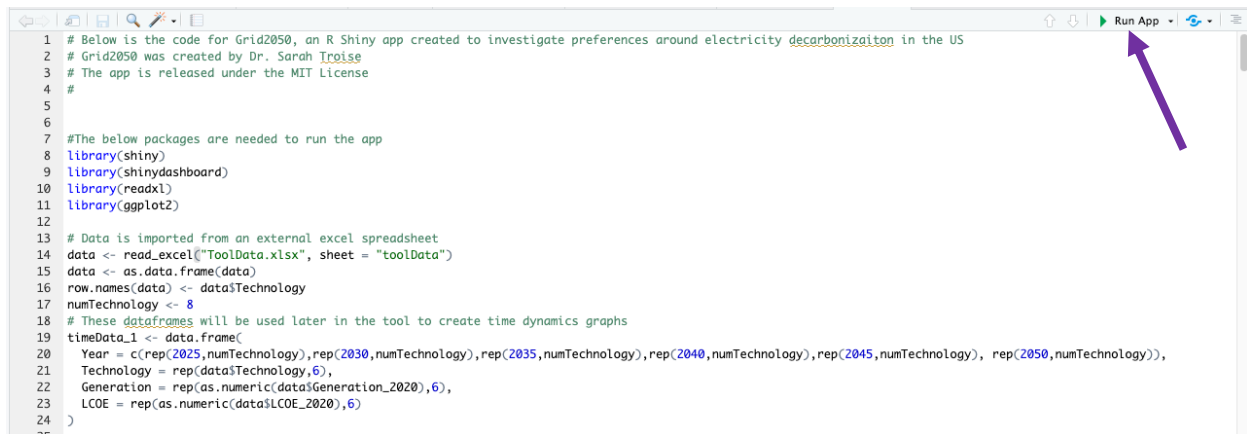
UI: The UI component defines how the web application will look. It's created using functions like `fluidPage()`, `sidebarLayout()`, `plotOutput()`, `textInput()`, etc., to create various elements such as layouts, inputs, outputs, and widgets.

Server Logic: The server logic defines how the application behaves and responds to user input. It's created using functions like `server <- function(input, output) { ... }`, where `input` represents the user inputs and `output` represents the outputs displayed to the user.

3.3 Running a Shiny Application Locally:

To run a Shiny application locally, follow these steps:

- Write your UI and server logic code in a single R script (e.g., `app.R`).
- Save the script in a directory.
- Set your working directory to the directory containing your script.
- Run the application using the `runApp()` function: `shiny::runApp("path/to/your/app.R")`.
 - If you are using RStudio, you can also click the “Run App” button



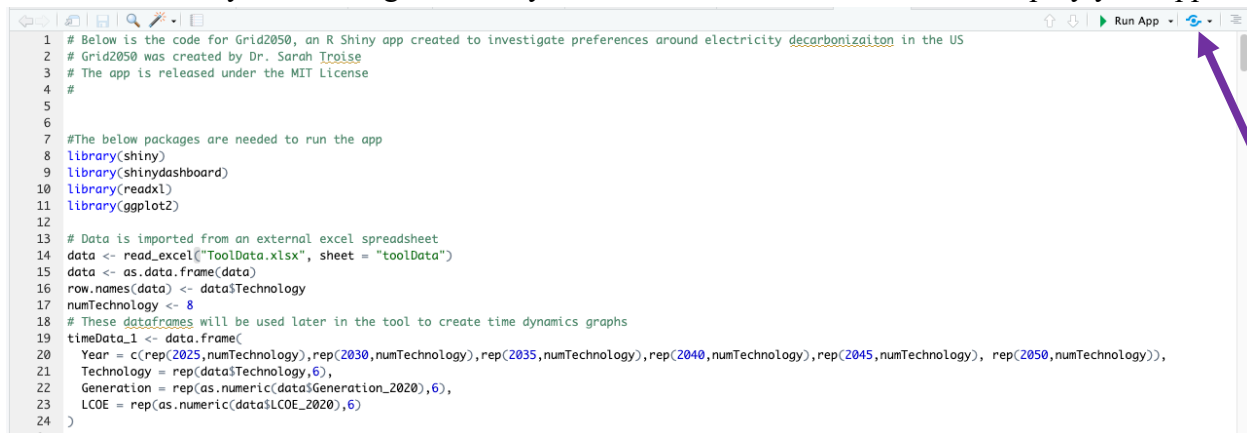
- Your default web browser should open, displaying your Shiny application.

3.4 Deploying a Shiny Application:

You can deploy your Shiny application to various platforms like shinyapps.io, RStudio Connect, or your own server.

Here's a general guide to deploying on shinyapps.io:

- Sign up for a shinyapps.io account: <https://www.shinyapps.io/admin/#/signup>.
- Install the `rsconnect` package: `install.packages("rsconnect")`.
- Deploy your application using the `rsconnect::deployApp()` function
 - If you are using RStudio, you can also click the blue button to deploy your app



3.5 Best Practices and Tips:

- Keep your UI code clean and organized for easier maintenance.
- Minimize unnecessary computations in the server logic to improve performance.
- Test your application thoroughly on different devices and browsers.
- Consider user experience and design principles when designing your UI.
- Regularly update your Shiny and R packages to benefit from the latest features and improvements.

3.6 Additional Resources:

- Official Shiny documentation: <https://shiny.rstudio.com/tutorial/>
- RStudio Shiny Cheatsheet: <https://shiny.rstudio.com/images/shiny-cheatsheet.pdf>
- Shiny Developer Center: <https://shiny.rstudio.com/>
- RStudio Community: <https://community.rstudio.com/c/shiny/>

4. Possible Customizations

This section of the user manual presents instructions for customizing Grid2050 to suit specific needs and preferences. There are numerous ways to adapt the tool to answer different research questions, but we have presented a few below. These instructions give an overview of how to make the proposed changes.

4.1 Adding a new technology

These instructions detail how to modify the selection of technologies available for the portfolios within Grid2050 to include additional technologies.

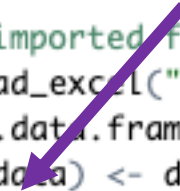
1. First, you will need to add the technology-specific data to “ToolData.csv” for the following values:

- Current LCOE
- Current generation capacity (TWhrs)
- Maximum growth rate, if applicable

2. Next, you will need to update the “**numTechnology**” value in code base to match the new number of technologies you are including

```
#The below packages are needed to run the app
library(shiny)
library(shinydashboard)
library(readxl)
library(ggplot2)

# Data is imported from an external excel spreadsheet
data <- read_excel("ToolData.xlsx", sheet = "toolData")
data <- as.data.frame(data)
row.names(data) <- data$Technology
numTechnology <- 8
```



3. Third, you will need to add a **tabPanel()** to the assumptions tab.

```
tabItem(tabName = "assume",
  tabsetPanel(
    tabPanel("Assumptions",
      column(width = 12, style='padding:20px',
        br(),
        includeHTML("assumption.html"))),
    tabPanel("CCS",
      column(width = 12, style='padding:20px',
        br(),
        includeHTML("CCS.html"))),
```

It should take the following form


```

tabPanel("TechnologyName",
column(width = 12, style='padding:20px',
br(),
includeHTML("TechnologyName.html"))),

```

4. Finally, write the “TechnologyName.html” file with the assumptions of your new technology

4.2 Using as a regional tool

These instructions detail how to modify the Grid2050 tool to model a specific region or state.

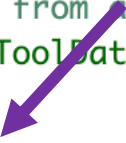
1. First, you would need to update “ToolData.csv” to reflect the cost and generation of each technology in the region you are aiming to model.
2. You would need to update the base electricity demand to be the demand of your chosen region.

```

# Data is imported from an external excel spreadsheet
data <- read_csv("ToolData.csv")

US_demand <- 4243
#Creates a dataframe format to be used in the tool
data <- as.data.frame(data)

```



3. The data in the HTML file’s for each technology would need to be updated to reflect the new base assumptions of your chosen region
4. Additional information should be added to the instruction page with a description of your region