
November 2024

R Shiny Masterclass Series - Advanced

Session 8

AI Tools, Programming sins and how to avoid them



EPI-interactive

Agenda

Session 1 | October 30th | Introduction, recap and responsive interfaces in R Shiny

Session 2 | November 1st | Advanced reactivity and UX considerations

Session 3 | November 5th | Useful R packages to extend core Shiny functionality

Session 4 | November 6th | Managing complexity: modularizing with the module pattern

Session 5 | November 8th | Advanced data sources and processing

Session 6 | November 12th | Automated report generation

Session 7 | November 13th | User authentication, Extended exercise

Session 8 | November 15th | AI Tools, Programming sins and how to avoid them

Today

- Review extended exercise
- AI Tools discussion
- Programming sins and how to avoid them
- Wrap up

Extended Exercise

Exercise (using /stage3)

- In *quarto-module.R*:
 - Generate quarto render function with inputs for html download
 - Inputs: filtered world data, filtered temperature data, both country inputs
 - Copy generated file for user to download
- In *quarto.qmd*:
 - Generate a section for each country, displaying the countries name, continent, region, subregion and flag
 - For each displayed plot in compare counties, generate a quarto section with a title and plot

Stage 3

R Shiny Advanced Masterclass

AUTHOR
Epi-interactive

PUBLISHED
March 25, 2024

Session 7

This exercise will allow you to practise the use of Quarto, integrating Quarto with R / R Shiny code, and exporting a PDF from an Quarto Markdown document through a Shiny application.

In separate paragraphs below, please insert the Quarto Markdown text described in the exercise. Feel free to include more additions to this content:

Today's date: 2024-03-25

Countries Summary

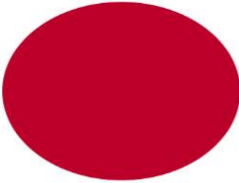
Canada

Continent: North America
Region: Americas
Subregion: Northern America

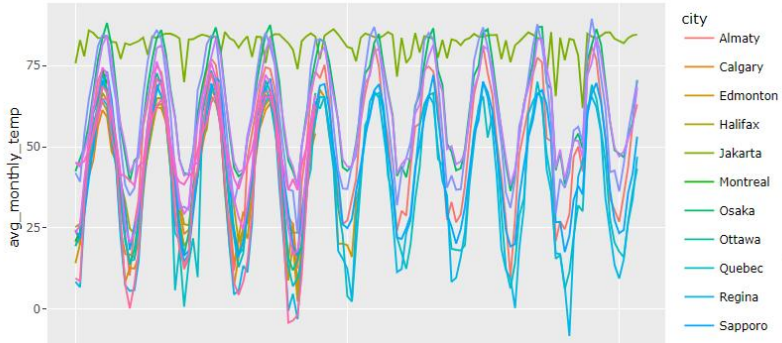


Japan

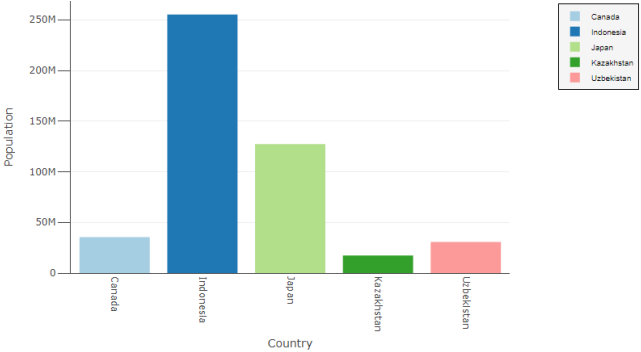
Continent: Asia
Region: Asia
Subregion: Eastern Asia



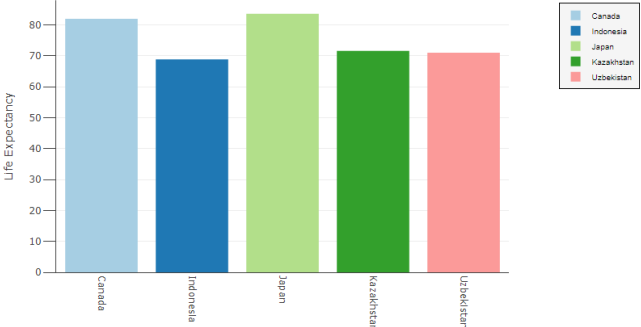
Average Monthly Temperature



Population



Life Expectancy



Exercise (using /challenge)

- In *global.R*
 - Create a dataframe with the credentials in it
 - This should use the ADMIN_USER, ADMIN_PW, and AUTH_DB_KEY in your *.env* file
 - Run the `create_db()` function to initialize the SQLite database
- In *server.R*
 - Put the credentials check and `auth_output` at the beginning of the `shinyServer` function
- In *ui.R*
 - Wrap the ui in the `secure_app()` function to ensure it loads the login screen first

AI tools

AI Tools

In many domains: chatbots, insurance, health, medical, travel, finance, translations, writing assistants.... [Medium.com](#)

Generative AI

*“At a high level, generative AI refers to a category of AI models and tools designed to create **new content, such as text, images, videos, music, or code**. Generative AI uses a variety of techniques—including neural networks and deep learning algorithms—to **identify patterns** and **generate new outcomes** based on them.”* [Github Blog](#)

Tools for data science / analytics

Level of abstraction

- Customer ► Waiter ► Chef ► Prep-cook

For R programming workflows

- GitHub Copilot in RStudio and Posit Workbench (since late September 2023)
- AI powered code utilities
- Online generative AI tools




Helpful Developer tools




AI Powered R Packages

Extending The RStudio / R workflow with AI

- OpenAI
- GPTStudio
- GPTTools
- Chattr

Submit 
Copilot (GitHub)

You can remove all the vowels from a character string in R using the `gsub()` function. Here's the code:

```
string <- "Hello, World!"  
string_without_vowels <- sub("[aeiou]", "", string, ignore.case = TRUE)  
string_without_vowels
```

This code will remove all vowels (both lowercase and uppercase) from the string and store the result in the variable `string_without_vowels`.

How can I remove all the vowels from a character string?

Hello! How can I assist you today with R programming and data modeling using tidyverse and tidymodels?

Hello there!

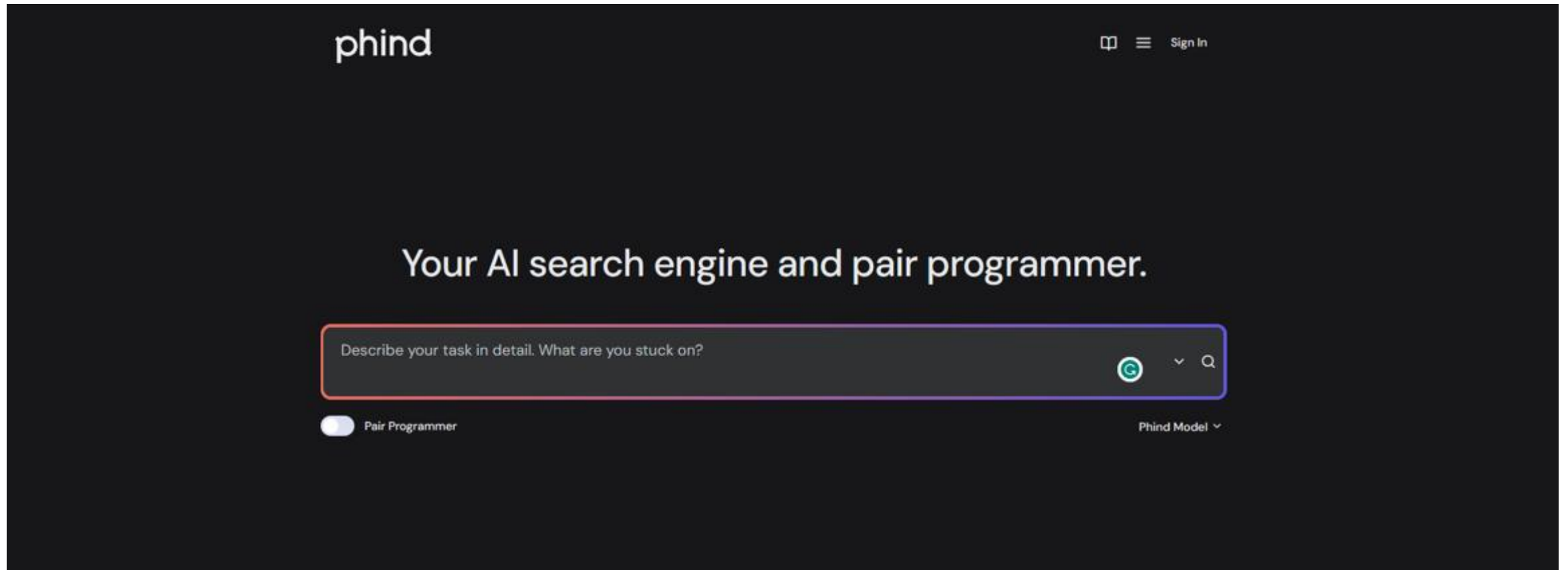
Useful tools

phind

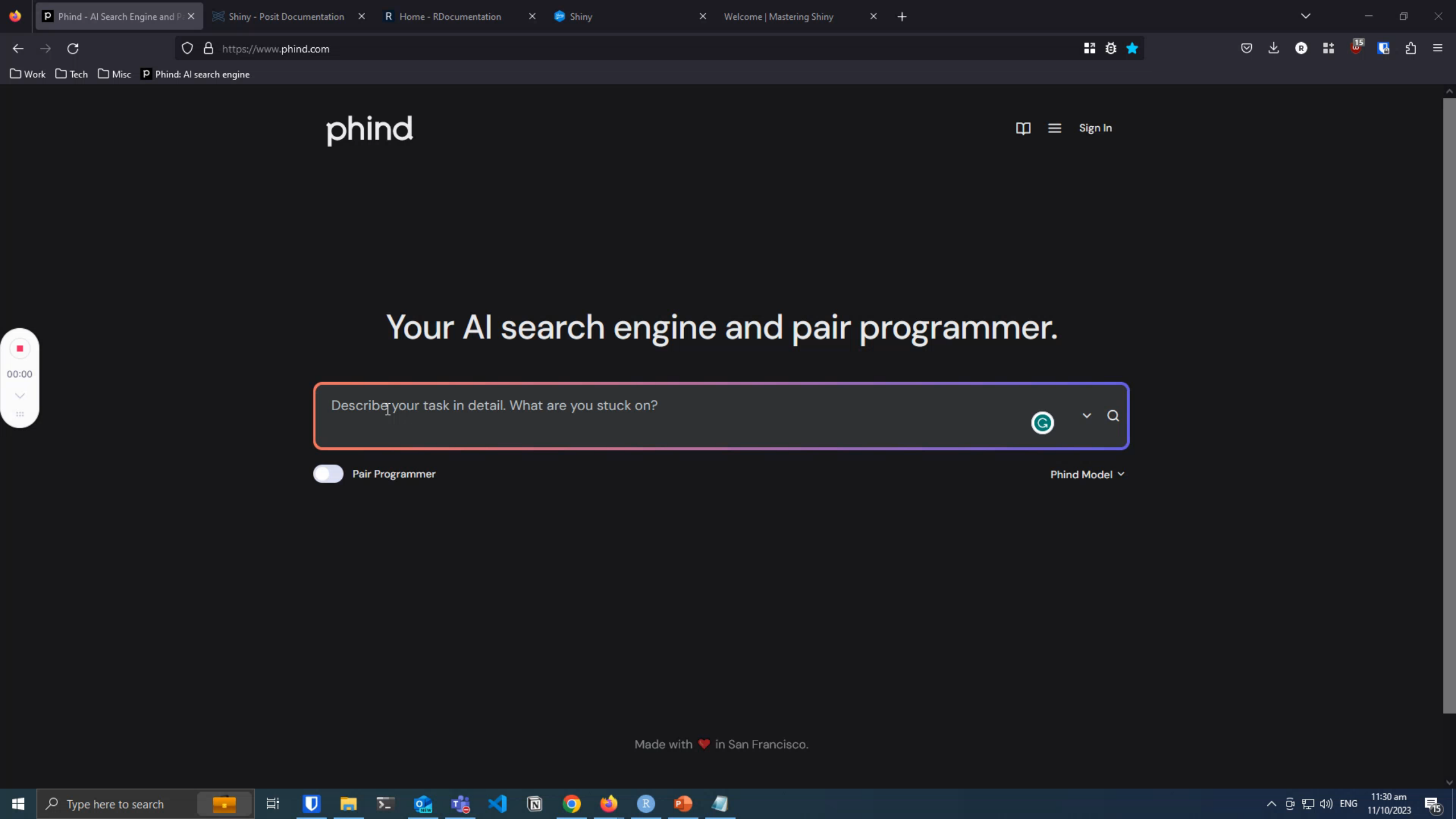


1. **Phind** – phind.com
 - Most helpful for questions about a specific task or function
 - Useful for searching docs and Stack Overflow
 - Has access to the internet and can be provided with specific sources
2. **Chat GPT** – chat.openai.com
 - More open ended than phind
 - Make further comments or questions to clarify results
3. **Copilot** – github.com/features/copilot
 - Autocompletes code
 - Improves coding practice
 - Open ended suggestion and prompts

Phind



<https://www.phind.com>



ChatGPT

- Fluid conversation between ChatGPT and the developer
- Allows clarification to get better results (without retyping the whole prompt)
- Useful to find out how to do something specific with technology that is well established (e.g. Regex)

How do I change the number of ticks on a plotly graph using the r package plotly?



In Plotly using R, you can control the number of ticks on an axis using the `tickmode`, `nticks`, or `tickvals` attributes within the layout settings for `xaxis` or `yaxis`.

Here's how you can do it:

Using `nticks`

If you want Plotly to automatically generate a specific number of ticks, you can use `nticks`:

R

Copy code

```
library(plotly)

# Sample data
x <- c(1, 2, 3, 4, 5)
y <- c(10, 15, 13, 17, 14)

# Plotly scatter plot
fig <- plot_ly(x = ~x, y = ~y, type = 'scatter', mode = 'lines+markers')

# Adjust the number of ticks on x-axis and y-axis
fig <- fig %>%
  layout(
    xaxis = list(nticks = 10), # Set number of ticks for x-axis
    yaxis = list(nticks = 5)   # Set number of ticks for y-axis
  )
```



Message ChatGPT

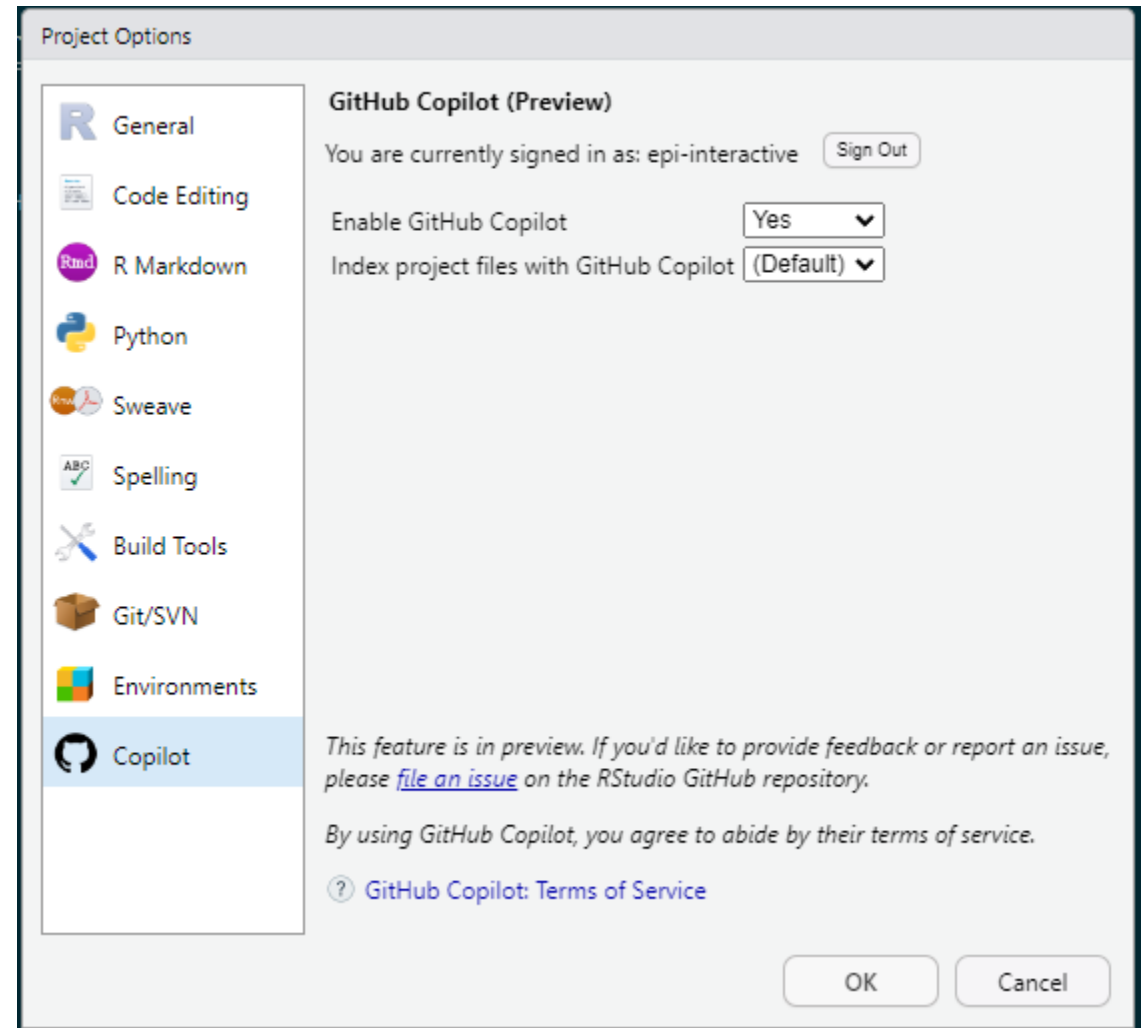


GitHub Copilot



Copilot in RStudio

- Available in RStudio and Posit Workbench 2023.09.0.
- Requires a GitHub account with an active Copilot subscription.
- Needs to be enabled in the project or global settings.



example-shiny - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

Addins

example-shiny

Files

Plots

Packages

ktop > AI > example-shiny

Name

..

.Rhistory

.Rproj.user

app.R

example-shiny.Rproj

preprocessing.R

00:00

Environment

History

Cc

248 MiB

R

Global Environment

Data

diamo... 53940 obs. ...

Functions

get_d... function ()

app.R

preprocessing.R

1

2

3

4

5

6

7

8

9

10

11

12

13

14

library(shiny)

ui <- fluidPage(
 "Hello, world!"
)

server <- function(input, output, session) {

}

shinyApp(ui, server)

Run App

1:15

(Top Level)

Copilot: No completions available.

R Script

Console

Terminal

Background Jobs

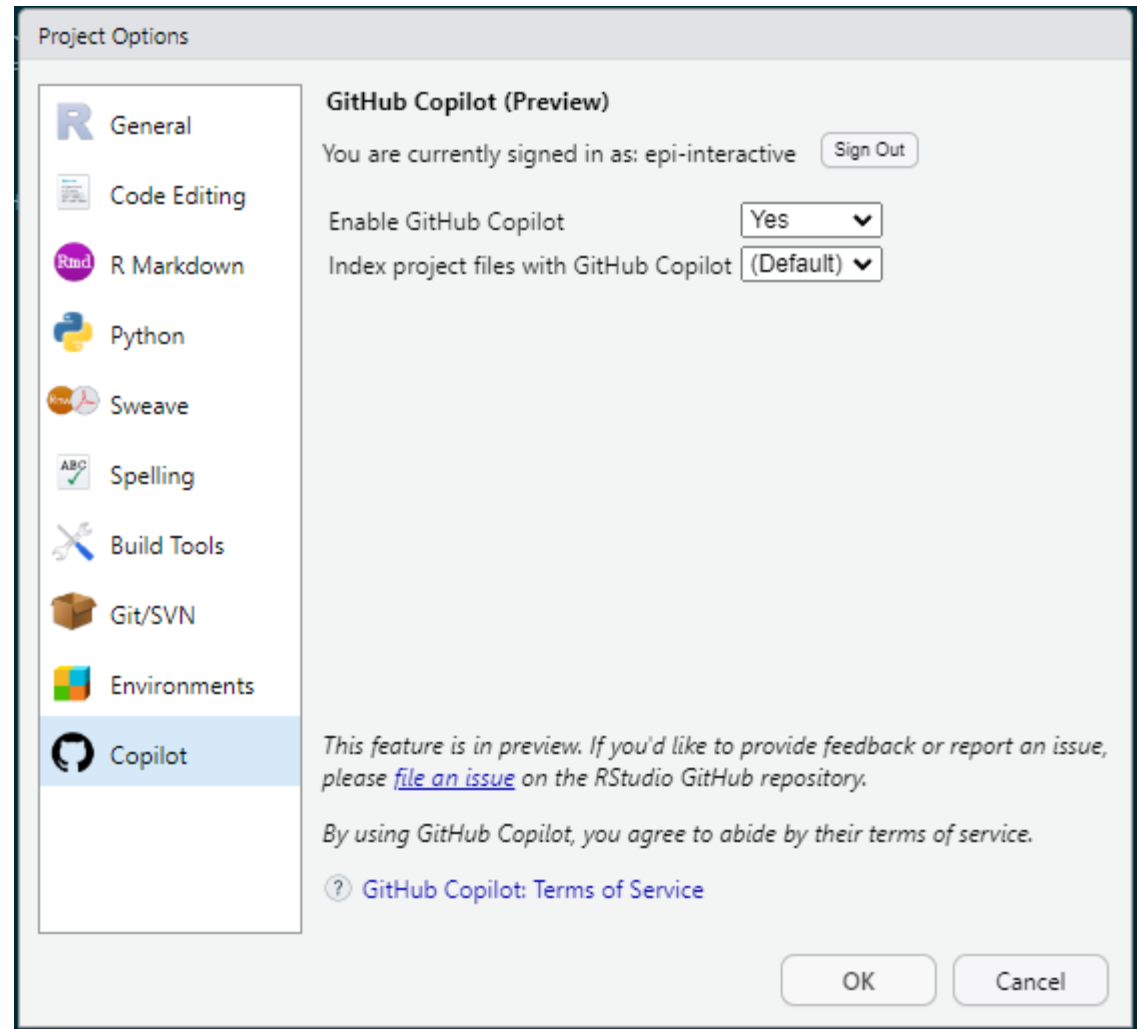
R 4.1.3 · C:/Users/Ben/Desktop/AI/example-shiny/

>


Type here to search

10:04 pm 11/10/2023

Copilot Privacy Concerns



GitHub Account Settings

 **Epi-interactive** (epi-interactive)
Your personal account

[Go to your personal profile](#)

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and plans

Emails

Password and authentication

Sessions

SSH and GPG keys

Organizations

Enterprises

Moderation

Code, planning, and automation

Repositories

Codespaces


Packages

Copilot

Pages

GitHub Copilot

Get started with Copilot

 You have an active Copilot subscription.

Select your IDE and follow the easy steps to get started with your AI pair-programmer.

Visual Studio Code

JetBrains

Visual Studio

Vim/Neovim

Suggestions matching public code

GitHub Copilot can allow or block suggestions matching public code. See [the GitHub Copilot documentation](#) to learn more.

Allow

☐ Allow GitHub to use my code snippets for product improvements *

Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features. More information in [Privacy FAQ](#).

Save

ⓘ

 It can take up to 30 minutes for the changes to take effect. Restart your code editor for the changes to take effect immediately.

Useful tools

phind



1. **Phind** – phind.com
 - Most helpful for questions about a specific task or function
 - Useful for searching docs and Stack Overflow
 - Has access to the internet and can be provided with specific sources
2. **Chat GPT** – chat.openai.com
 - More open ended than phind
 - Make further comments or questions to clarify results
3. **Copilot** – github.com/features/copilot
 - Autocompletes code
 - Improves coding practice
 - Open ended suggestion and prompts

What to look out for

- Privacy concerns
 - Copilot data sources for model training?
 - ChatGPT – data upload
- Hallucinations – accuracy can lack
- Limited training data (e.g. Chat GPT used to be time limited)

UL

Who is the sole survivor of the Titanic?



The last known survivor of the RMS Titanic, Millvina Dean, passed away on May 31, 2009. Millvina Dean was an infant when the Titanic sank in 1912, and she was the youngest passenger on board. She lived to be 97 years old and was the last remaining survivor of the tragic sinking at the time of her death.

Good for specific use cases! (be as specific as possible)

Exploring the AI toolbox



youtube.com/watch?v=NTMCFciE4j4

Programming Sins and How to Avoid Them

1. Using for loops in R & Shiny

- For loops are slow
- R has built-in functionality for iterating over data – the ‘apply’ family of functions (*lapply*, *sapply*, *tapply*, *vapply*)
- Using these functions instead will give us better performance

```
dat <- largedata
dat$maingenre <- ""
for(n in 1:nrow(largedata)) {
  dat[n, ]$maingenre <- ...
}
```



```
dat <- largedata
dat$maingenre <- ""
dat$maingenre <- sapply(dat$genres,
  function(x){ return(...) }
)
```

2. Repeated code

- Repeating code makes us more open to mistakes
- Can be hard to spot mistakes sometimes!
- Group code into functions, reactives, modules to avoid this
- Spot the difference?

```
data$exc <- rbind(data$exc,  
data$inc[input$included_items_rows_selected, ])  
data$inc <- data$inc[  
input$included_items_rows_selected,]
```

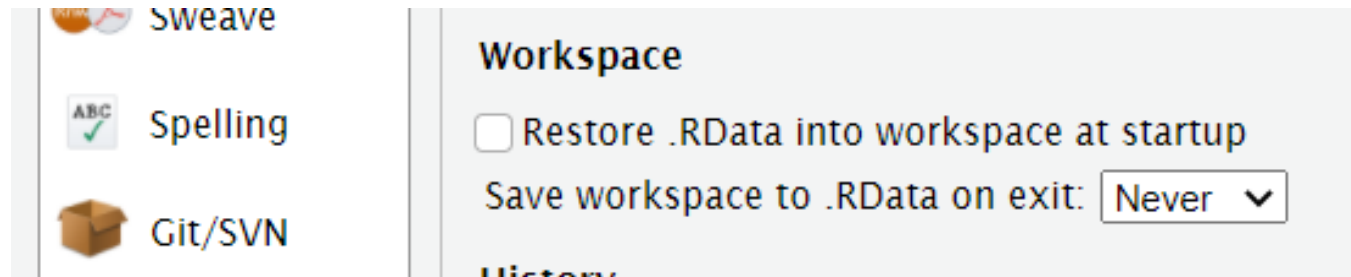
```
data$exc <- rbind(data$exc,  
data$inc[input$included_items_rows_selected, ])  
data$inc <- data$inc[  
-input$included_items_rows_selected,]
```

3. Colossal code files

- Large files are difficult to read and maintain
- Can easily 'get lost' in different sections
- Tightly coupled code difficult to update
- Want to keep ui.R / server.R as small as possible:
 - Separate files
 - Module pattern
- Do this early to avoid refactoring later!

4. Treating environment data like pets

- “Don’t rely on data in your environment that you couldn’t recreate on demand”
- Pre-processing scripts to generate data
- Posit Cloud setting to clear environment on startup
(*Tools > Global options*):



5. Not optimising your application

- Users have high expectations for software performance
- Avoid repeating expensive operations:
 - Load data once
 - Create reactivities for different stages
 - Move expensive operations to pre-processing
 - Minimise data updates
- Include some UX features:
 - Skeletons
 - Progress bars
 - Warnings for long processing times



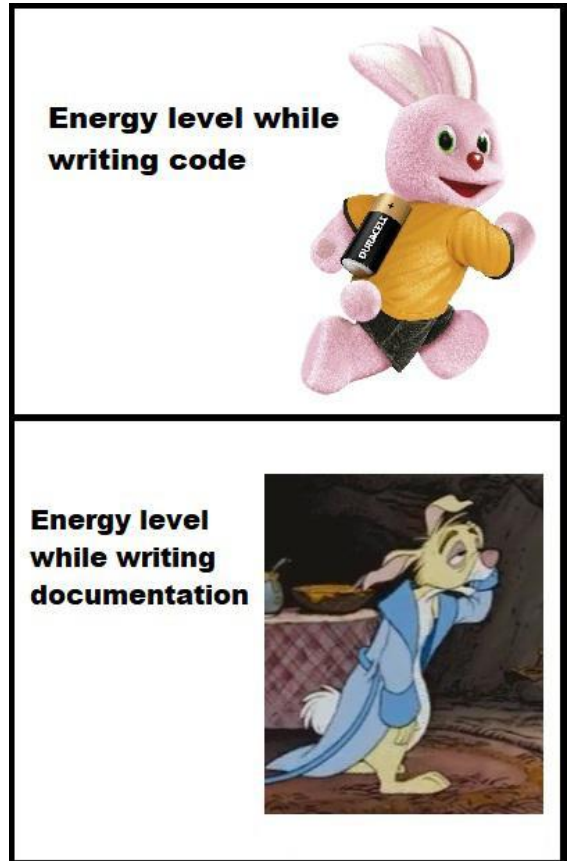
6. Neglecting version control

- Makes it harder to collaborate
- Harder to keep track of correct version
- Harder to roll back unwanted changes
- Vulnerable to random failure (PC crash, corrupted files etc)

- Cloud based version control makes collaboration simple
 - GitHub, GitLab, BitBucket etc
- All versions stored online, can roll back to any version needed
- Secure against random failures

7. No documentation

- Hard to return to undocumented code
- Hard to collaborate on undocumented code
- Comments should explain 'why', code should explain 'how'
- Write 'self-documenting' code
- Can also write full documentation (?help) using "roxygen2" package (structure dependent)
- Handover documentation / walk-through of key concepts



https://www.reddit.com/r/ProgrammerHumor/comments/bvxeku/im_only_20_done_with_this_documentation_and_im/

8. Too many dependencies

- Too many dependencies slows down project development
- Libraries come with other hidden dependencies
- Project will be more fragile, harder to maintain over time
- Minimise use of packages, look for existing functionality ***where possible***
- <https://www.tidyverse.org/blog/2019/05/itdependencies/>

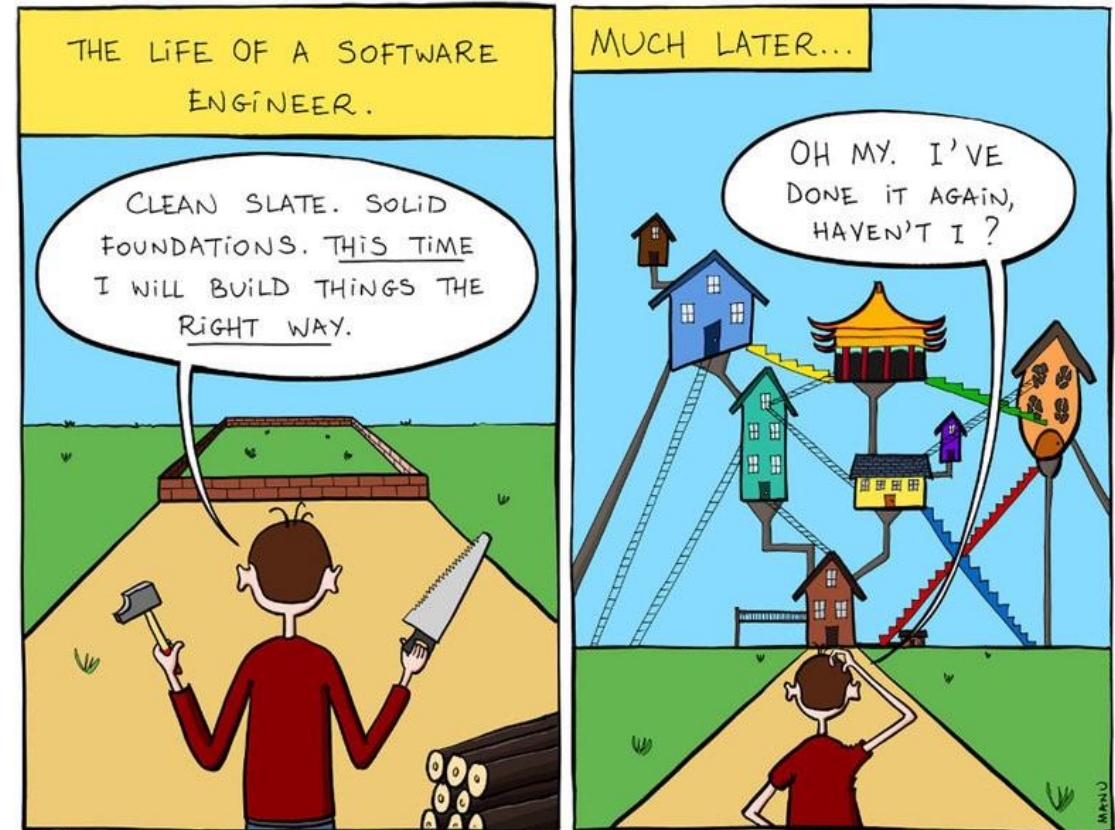
pkg	count
base	3151
NA*	792
htmltools	201
dplyr	130
shinyjs	34
DT	22
stringr	19
sf	16
plotly	8
tidyr	5
webshot	5
httr	2
png	2
readxl	2
base64enc	1

9. Not testing regularly

- Code changes can have unintended side effects
 - Some bugs only appear in specific environments
 - Bugs may be introduced by changes in external factors
 - Previously fixed bugs may re-emerge
-
- Have a dedicated testing step in development / publishing
 - Don't make assumptions
 - Test early, test often
 - Follow a test plan!

10. Developing without a plan

- Complexity of Shiny apps can escalate fast
- Without a plan, codebase can quickly become disorganised
- Plan for new features ahead of time
- Modular code
- Regularly review, refactor & tidy code



<https://plus.google.com/+ManuCornet/posts>

Open source considerations

Free software

- For free – no cost (*gratis*)
- Little or no restrictions (*libre*)

Richard Stallman on *libre*:

“Think of free as in free speech, not free beer”

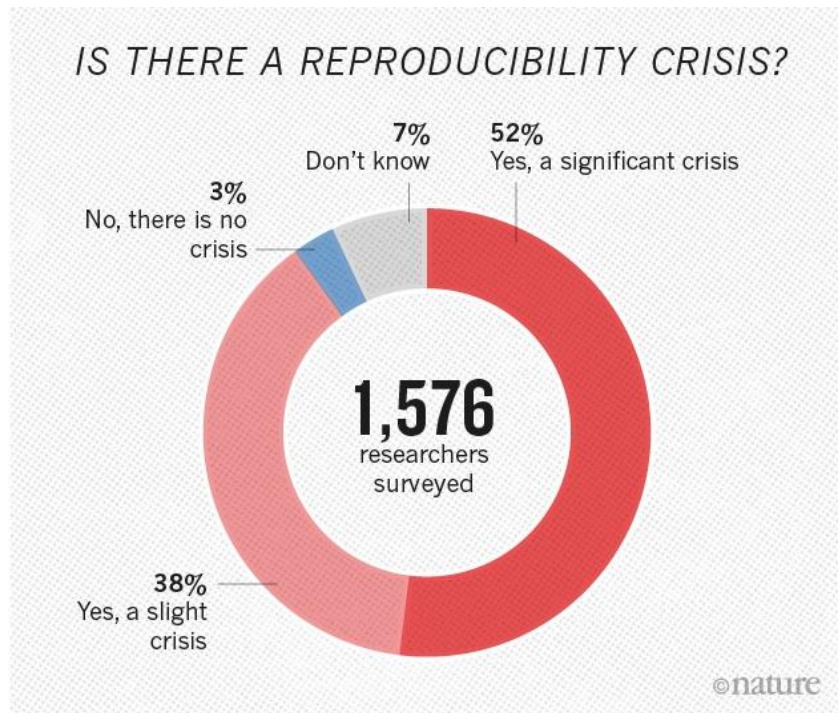
Both are important!

Four essential freedoms

1. The freedom to run the program as you wish, for any purpose.
2. The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
3. The freedom to redistribute copies so you can help others.
4. The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Reasons for using open source

Reproducibility



- Ability for others to run your software
- Proprietary software = less reproducible
- (Expensive) licence
- Vendor might go out of business

<https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970>

Reasons for using open source

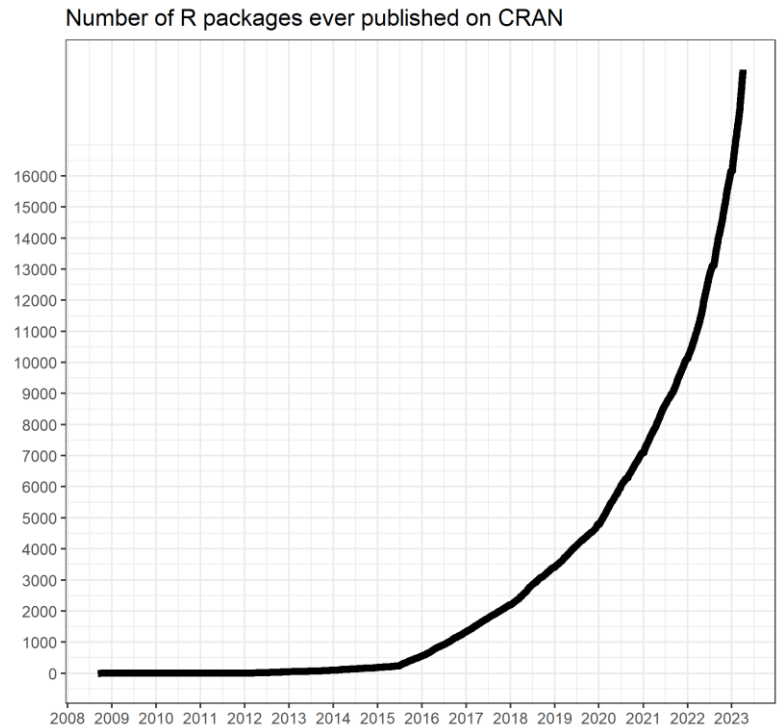
Resiliency

- Software companies come and go – tying research to vendors is risky
- Prices may rise dramatically
- Four freedoms guarantee this can't happen
- Example: MySQL

Reasons for using open source

Participation

- Single vendor vs community
- Science: requires in-depth methodological innovation – can't be provided by single vendor
- Low cost in creating and distributing software supports niches and even small communities



Currently ~20,600 packages

Reasons for using open source

Accessibility

- Supports data literacy and science – important for the global economy and research community
- All individuals should have access (students, professionals, researchers, ...)



R Shiny open source repositories

We develop innovative data-driven dashboards for better decision making, data sharing and teaching – open-source technology is at the heart of what we do, here are some coding samples we'd like to give back to the community.

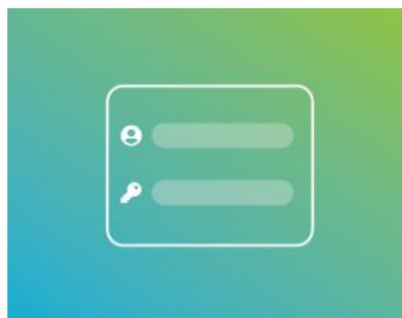
Looking for more? Contact us at info@epi-interactive.com

 **posit**™ Full Service Certified Partner

Progress Indicator



User Authentication



Datatable Overlay



Custom Theming



Dynamic PDF Generator



Choropleth Grid



Wrapping up

- Anything to clarify from our sessions?
 - What concepts have been most useful?
 - Any R Shiny projects planned?
-
- Certificates, Posit Cloud, learning platform, recordings
 - Feedback survey

Connect with us on [LinkedIn](#)

Thank you for joining the Masterclass!