IMPLEMENTACJA ŚRODOWISKA TESTOWEGO W JĘZYKU *R*

Jan Baumgart

Student 1 roku MU Informatyka

Uniwersytet Kazimierza Wielkiego

Wydział Matematyki, Fizyki i Techniki

e-mail: janek@ukw.edu.pl

Streszczenie: *W pracy przedstawione zostały biblioteki oraz ich zastosowanie do implementacji środowiska testowego algorytmów rojowych. Wykorzystane moduły pozwalają przygotować w pełni sprawne narzędzie w szybki sposób. Przedstawione środowisko testowe w tym artykule zostało przygotowane do wspomagania badań nad algorytmami rojowymi i może być wykorzystane jako zasób sieciowy jak i skrypt dostępny lokalnie. Metody wykorzystane w tym artykule mogą posłużyć do budowy środowisk testowych dla wielu innych nie związanych z algorytmami rojowymi sceneriów.*

Słowa kluczowe: *R, Shiny R, Shiny Dashboard, DT, Algorytmy, Środowiska testowe.*

Implementation of a test environment in *R*

Abstract: *The work presents libraries and their application for implementing the swarm algorithms test environment. The modules used allow you to prepare a fully functional tool quickly. The test environment presented in this article has been prepared to support research on swarm algorithms and can be used as a network resource as well as a locally available script. The methods used in this article can be used to build test environments for many other scenery not related to swarm algorithms.*

Keywords: *R, Shiny R, Shiny Dashboard, DT, Algorithms, Test enviroments.*

1. Wprowadzenie

Podczas rozwiązywania wielu problemów naukowych często trudnym zadaniem jest przedstawienie wyników w sposób przystępny. W każdym projekcie niezależnie czy w środowisku korporacyjnym czy naukowym nadchodzi moment prezentacji danych celem omówienia osiągniętych wyników. Dlatego często za kluczowe uważa się przygotowanie środowiska testowego które pozwoli na czytelną reprezentację efektów naszej pracy. Zaprojektowanie i przygotowanie odpowiedniego narzędzia niezależnie od wykorzystywanego środowiska czy omawianego problemu może czasem okazać się zadaniem zajmującym więcej czasu niż samo rozwiązanie problemu. Głównymi trudnościami jakie można napotkać na swej drodze są problemy z czytelną reprezentacją wyników czy zaprojektowanie prostego w obsłudze interfejsu użytkownika.

Pomysł rozwiązania problemu powstał i został rozwinięty podczas badań nad implementacją skierowanych liczb rozmytych w algorytmach rojowych. Interfejs został przygotowany w środowisku R ze względu na początkową implementację algorytmów rojowych aby zachować spójność badań i wystrzec się konieczności przepisywania aktualnie istniejących implementacji oraz aby zachować główną zaletę języka skryptowego R którą jest prędkość kompilacji oraz łatwość eliminacji błędów. Głównymi bibliotekami wykorzystywanymi podczas

przygotowywania omawianego środowiska są moduły Shiny, Shiny Dashboard oraz DT. Pozostałe omawiane moduły to biblioteki takie jak: ShinyCSSLoaders, ShinyJS oraz wiele innych modułów poprawiających czytelność i odbiór wyników pracy.

Przedstawione zrzuty ekranu narzędzia są przykładowym wykorzystaniem środowiska testowego przygotowanego właśnie z użyciem omówionych bibliotek dostępnych w środowisku R.

2. Środowisko R

Środowisko R jest bardzo często wykorzystywane do obliczeń statystycznych i ich wizualizacji. Głównymi powodami szerokiego zainteresowania omawianego języka jest dostępność (środowisko R jest dostępne jako Open Source na warunkach licencji GNU (*General Public License*), szybkość działania oraz elastyczność dostępnych narzędzi. Te główne zalety spowodowały że wiele firm zdecydowało się na wykorzystanie komercyjne środowiska R. Najlepszymi przykładami są firmy takie jak Facebook - R używany do analizy zachowań związanych ze zmianami statusu, Google - do mierzenia efektywności reklamy oraz przewidywań ekonomicznych oraz wiele innych korporacji takich jak Twitter, Microsoft, Uber, Airbnb, IBM. Jednak zanim język R znalazł swoje zastosowanie w świecie modeli biznesowych swoje początki język przeżył na uczelniach na całym świecie i tam króluje do dnia dzisiejszego. Został zaadaptowany na wiele różnych platform Unix oraz Windows i MacOS. R to nie tylko język programowania, to całe zintegrowane środowisko, zawierające oprogramowanie do wykonywania operacji na danych oraz reprezentacji wyników. R to optymalnie zaprojektowany i spójny system przygotowany do operacji na zbiorach danych. Zastosowanie R w projektach pozwala na ominięcie problemów związanych z innymi nieelastycznymi programami do operacjach nad danymi, problemów jak limitacje odczytu danych do wąskiego typu danych czy problem z brakiem narzędzi do reprezentacji wyników.

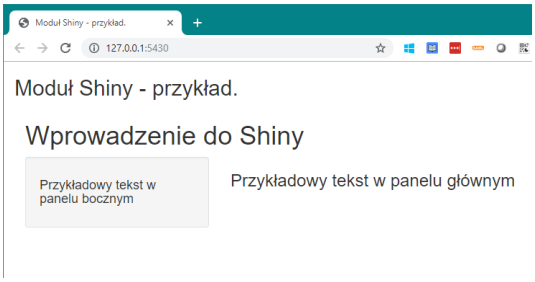
3. Biblioteka Shiny

Shiny to darmowa biblioteka przygotowana przez znanych w kręgu twórców oprogramowania RStudio. Wykorzystując omawiany moduł w bardzo prosty i przyjazny sposób można przygotować dynamiczne strony internetowe wykorzystując tylko język R bez konieczności nauki kolejnych języków programowania jak JavaScript czy języka znaczników HTML. Stworzone aplikacje z wykorzystaniem moduły Shiny są generowane dynamicznie co pozwala na zmianę treści strony podczas pracy skryptu. Omawiana biblioteka nie ogranicza i pozwala jednak na korzystanie z technologii takich jak HTML, CSS czy JavaScript. Moduł Shiny jest w pełni zgodny ze środowiskiem R, co pozwala na jego używanie na każdej maszynie obsługującej środowisko R. Dzięki zastosowaniu przez twórców, biblioteki HTTPUV która dostarcza komunikację dwukierunkową między serwerem a użytkownikiem, interakcja między stworzoną przez nas stroną a naszymi skryptami dokonywana jest natychmiastowo, a całą zaawansowana implementacja komunikacji obsługiwana jest właśnie przez moduł Shiny.

Poniżej przedstawiony został prosty przykład jak przygotować prostą aplikację z użyciem biblioteki Shiny. Dobrą metodyką jest stworzenie dwóch osobnych plików UI.R oraz Server.R, lecz celem ukazania łatwości przygotowania aplikacji przykłady kodu aplikacji przygotowanej będą w jednym pliku App.R.

```
#Komenda ładująca
#Shiny do naszego skryptu.
library(shiny)

#Zdefiniowanie strony
#UI dla naszego programu.
ui <- fluidPage(
  #Nadanie tytułu naszej stronie.
  titlePanel("Moduł Shiny - przykład."),

  #Nazwa nazwy panelu głównego.
  headerPanel("Wprowadzenie do Shiny."),

  #Zdefiniowanie panelu bocznego.
  sidebarPanel(
  #Przykładowy nagłówek w panelu bocznym.
  h4("Przykładowy tekst w panelu bocznym.")
             ),
  #Zdefiniowanie panelu głównego.
  mainPanel(
    h3("Przykładowy tekst w panelu głównym.")
  )
)
#Logikę naszego programu zawieramy
#w części server naszej aplikacji.
server <- function(input, output){
}

#Zdefiniowanie głównych funkcji
#dla biblioteki Shiny.
shinyApp(ui = ui, server = server)
```

Rysunek. 1 Przykładowy kod prostej aplikacji z wykorzystaniem modułu Shiny wraz z komentarzem. Jest to cały plik app.R. Źródło: własne.



Rysunek. 2 Reprezentacja aplikacji z powyżej załączonego kodu. Źródło: własne.

Jak widać na załączonych rysunkach przygotowanie aplikacji internetowej używając biblioteki Shiny jest bardzo szybkie i proste. Dodanie kolejnych elementów to kwestia pojedynczych linijek kodu.

4. Biblioteka Shiny Dashboard

Użycie biblioteki Shiny pozwala przy niskim nakładzie pracy na osiągnięcie bardzo ładnych lecz prostych interfejsów. W tym miejscu na pomoc przychodzi kolejna biblioteka rozbudowująca możliwości pakietu Shiny.

Moduł Shiny Dashboard jest pakietem pozwalającym w bardzo szybki i przy małym skomplikowaniu stworzyć zaawansowane widoki dla naszej aplikacji. Shinydashboard to pakiet R, którego zadaniem jest ułatwienie (jak sama nazwa wskazuje) budowania przejrzystych dashboardów (z ang. kokpitów) z Shiny.

```
library(shiny)
  #Załądowanie modułu ShinyDashboard
library(shinydashboard)
  #Konfiguracja DashboardPage jako domyślnego widoku.
ui <- dashboardPage(
  #Załadowanie domyślnych modułów.
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
server <- function(input, output) { }
shinyApp(ui, server)
```

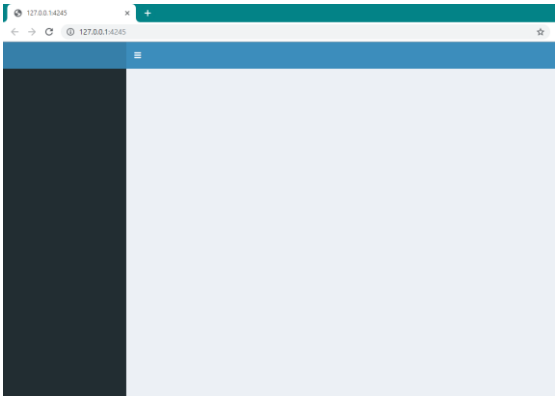
Rysunek. 3 Przykładowy kod prostej aplikacji z wykorzystaniem modułu Shiny oraz Shiny Dashboard wraz z komentarzem. Jest to cały plik app.R. Źródło: własne.

Część interfejsu użytkownika w aplikacji zbudowanej z Shiny Dashboard zawiera 3 podstawowe elementy zamknięte w dashboardPage:

1. dashboardHeader - elementy dostępne w nagłówku strony;
2. dashboardSidebar - elementy dostępne w panelu po lewej (domyślnie) stronie;
3. dashboardBody - elementy dostępne w głównym panelu aplikacji.

Aplikacja przygotowana z pomocą Shiny Dashboard jest bardzo ładna a przy tym zachowuje kompatybilność z elementami modułu Shiny w pełni co pozwala na bardzo efektowne i proste budowanie zaawansowanych aplikacji.

Przygotowanie aplikacji z wykorzystaniem opisanego modułu jest jeszcze prostsze a efekty jeszcze lepsze. Ponieważ korzystamy z wielu gotowych, przygotowanych szablonów. I tym sposobem w 8 linijkach kodu możliwe jest przygotowanie ładnie wyglądającego szablonu do naszej aplikacji.

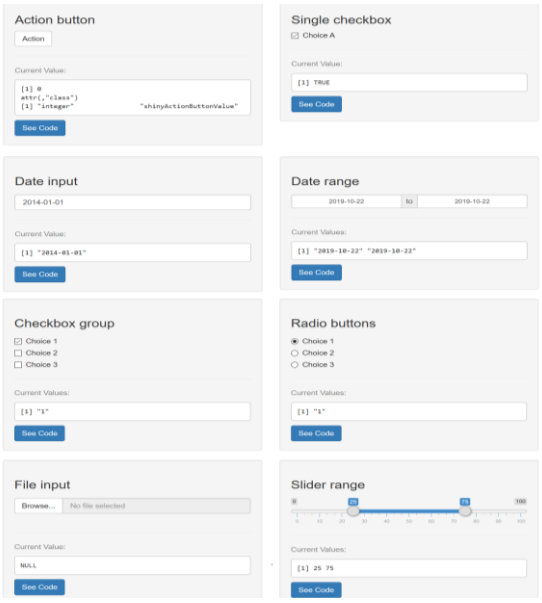


Rysunek. 4 Reprezentacja aplikacji z powyżej załączonego kodu z wykorzystaniem Shiny Dashboard. Źródło: własne.

Możliwość korzystania z elementów biblioteki Shiny jest w tym przypadku kluczowa, ponieważ dzięki połączeniu możliwości dwóch wymienionych bibliotek otrzymujemy narzędzie, które pozwoli nam zaprezentować wyniki naszej pracy w sposób bardzo profesjonalny w bardzo krótkim czasie. Jest to funkcjonalność, która w innych językach programowania wiązałaby się ze skomplikowaniem projektu, utratą kompatybilności czy innego rodzajami ograniczeń. Pakiet Shiny dla wielu jest głównym powodem wyboru R ponad język Python. Python pomimo łatwiejszej struktury kodu nie oferuje w tak prosty sposób reprezentacji wyników w tak szybki sposób. To w wielu przypadkach dyskwalifikuje Python podczas wyboru z zastosowań przy projektach bazujących na przetwarzaniu danych.

5. Widgety a możliwości

Shiny oferuje szereg dostępnych elementów graficznych do reprezentacji danych nazywanych ‘widżetami’. To właśnie widgety grają pierwsze skrzypce podczas budowania interfejsu. Widżety umożliwiają użytkownikom wysyłanie informacji do aplikacji. Widgety jak cała nasza aplikacja są dynamiczne oznacza to, że gdy użytkownik zmieni wartość na widżecie, wartość również się zmieni po stronie serwerowej naszej aplikacji. Pozwala to na dynamiczne zmiany konfiguracji naszych aplikacji oraz na bieżącą interakcje z naszą aplikacją, bez potrzeby przeładowywania strony.



Rysunek. 5 Przykłady niektórych widżetów dostępnych w Shiny. Źródło: https://shiny.rstudio.com/gallery/widget-gallery.html

Jednak nie musimy się ograniczać do ogromnej biblioteki widgetów Shiny, każda biblioteka R bazująca na HTML i JavaScript może zostać zaimplementowana w naszą aplikację. A takich modułów dodatkowych jest bardzo wiele.

Na chwilę uwagi zasługuje tu wykorzystywana wielokrotnie przeze mnie biblioteka DT (skrót od *DataTables*). Moduł poza skromną nazwą nie reprezentuję sobą nic skromnego. To przeogromna narzędzie pozwalające na wyświetlanie i pracę na tabelach. Przykładowe wykorzystanie tej tabeli znajdziemy w dokumentacji Shiny pod adresem: *https://gallery.shinyapps.io/012-datatables/* . Biblioteka DT poza podstawową obsługą wyświetlania danych pozwala również na ich segregację, paginację itp. co jest niezbędne przy pracy z wynikami w formie tabeli.

Show 10 entries Search:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.52	Ideal	D	VS2	61.4	56	1664	5.16	5.19	3.18
2	0.5	Very Good	F	SI1	62.3	60	1250	5.07	5.11	3.17
3	0.61	Ideal	G	VVS2	61.6	54	2242	5.45	5.49	3.37
4	0.36	Premium	G	VS2	62.5	58	756	4.55	4.51	2.83
5	0.7	Very Good	E	VS2	63.5	54	2889	5.62	5.66	3.58
6	0.56	Ideal	F	VS1	61.7	56	2016	5.32	5.28	3.27
7	1.19	Premium	E	I1	60.2	61	3572	6.91	6.87	4.15
8	0.52	Ideal	F	IF	60.6	57	2575	5.21	5.22	3.16
9	0.38	Ideal	E	IF	62.7	55	1433	4.61	4.67	2.91
10	0.51	Ideal	E	VS2	62.1	54	1608	5.13	5.15	3.19

Showing 1 to 10 of 1,000 entries Previous 1 2 3 4 5 … 100 Next

Rysunek. 6 Przykłady wykorzystania biblioteki DT. Źródło: własne.

O modułach dostępnych w R oraz współpracujących z biblioteką Shiny można pisać bez końca, a lista dostępnych ciągle rośnie z dnia na dzień. Z wartych wspomnienia warto zapoznać się z modułami takimi jak:

- ShinySense - Jest to to pakiet modułów, które pomagają Shiny obsłużyć moduły sprzętowe (m.in. nagrywać obraz z kamery internetowej, nagrywać dźwięk, przechwytywać dane akcelerometru).
- ShinyFiles - Przeglądarka systemu plików po stronie serwera dla Shiny.
- RegexSelect - Pozwala na wyszukiwanie wyrażeń regularnych w naszym zaznaczeniu. Wielokrotnie używana biblioteka do wyszukania szczegółowych danych.
- Waiter - Ekrany ładowania, niezbędne podczas pracy nad większymi zbiorami danych.
- Shinytoastr - Powiadomienia z aplikacji w formie pop up.
- Rintrojs - Moduł do tworzenia prezentacji krok po kroku i klikalnych wskazówek.
- plotly - Interaktywna generator wykresów 2D/3D. Wspiera specjalne wsparcie dla łączenia / podświetlania / filtrowania widoków.

oraz wiele innych bardzo ciekawych, wartych odkrycia bibliotek. Możliwości R są tak szerokie jak wybór dostępnych bibliotek, co może wydawać się nieskończone.

6. Budowa środowiska testowego

Celem projektu było zaprojektowanie i przygotowanie narzędzia. które pozwoliłoby testować nowo zaimplementowane algorytmy rojowe również z wykorzystaniem skierowanych liczb rozmytych pod kątem optymalizacji. Głównymi wyznacznikami podczas implementacji

rozwiązania była otwartość na rozwój, duża baza dostępnych algorytmów rojowych, brak ograniczeń implementacji nowych ustawień oraz możliwość przyjaznej reprezentacji danych. Największym wyzwaniem było przygotowanie narzędzia w sposób prosty do użytkowania, tak aby reprezentacja wyników była w przyjaznej użytkownikowi formie. Co miało na celu znacznie zachęcić do wykorzystywania narzędzia a tym samym zwiększyć zainteresowanie algorytmami rojowymi.

Środowisko testowe zostało przygotowane w sposób modularny tak aby można było je w przyszłości rozwijać w prosty sposób. Składa się z komponentów uruchamiających aplikację, komponentów interfejsu użytkownika, komponentów odpowiedzialnych za logikę środowiska testowego oraz plików serwowanych statycznie.

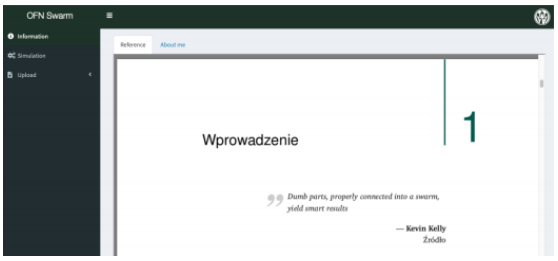
Komponent uruchamiający aplikację zawiera spis wszystkich ustawień globalnych. W tym komponencie również załadowane są wszystkie wykorzystywane przez nasz program biblioteki.

```
library(shiny)
library(shinydashboard)
library(shinycssloaders)
library(shinyjs)
library(dt)
library(waiter)
library(plotly)
```

Rysunek. 7 Zrzut ekranu bibliotek ładowanych z głównym komponencie środowiska testowego. Źródło: własne.

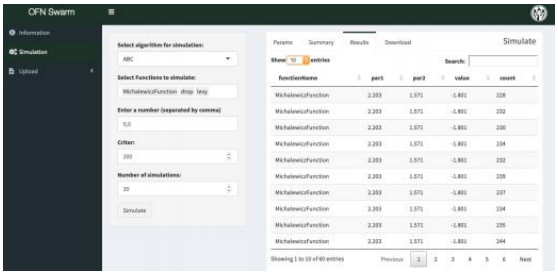
Moduły interfejsu użytkownika oraz logiki naszego środowiska testowego również zostały przygotowane tak, aby ich rozwój był prosty i dostępne nawet dla początkującego użytkownika. Każda dodana funkcja to dodanie nowego pliku w odpowiedniej lokalizacji oraz załadowanie w komponencie uruchamiającym.

W pierwszej zakładce, na stronie powitalnej umieszczona została dokumentacja projektu oraz krótkie wprowadzenie do algorytmów rojowych oraz problemów optymalizacji. Zostało to zrobione dzięki możliwości implementacji w języku R funkcji R Markdown i połączeniu dokumentacji przygotowanej w języku LaTeX za jej pomocą z naszym środowiskiem.



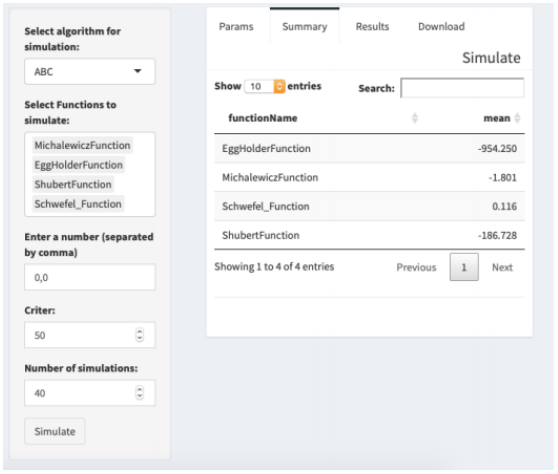
Rysunek. 8 Zrzut ekranu pierwszej strony oprogramowania.. Źródło: własne.

Drugi odnośnik w menu z prawej strony przenosi nas do zakładki testowania algorytmów na wybranych funkcjach matematycznych z wykorzystaniem wybranych algorytmów. Interfejs został przygotowany w taki sposób aby umożliwić testowanie algorytmów rojowych zarówno z wykorzystaniem skierowanych liczb rozmytych jak i w implementacji klasycznej, pozwala również na testowanie wielu algorytmów jednocześnie.



Rysunek. 9 Zrzut ekranu strony testowania w środowisku testowym. Źródło: własne.

Przygotowanie narzędzia w taki sposób pozwala na znaczne skrócenie czasu testowania algorytmów rojowych pod kątem optymalizacji oraz na szybkie i dokładne porównywanie wyników. Wynik jest reprezentowany za pomocą biblioteki DT oraz widgetu zakładek.



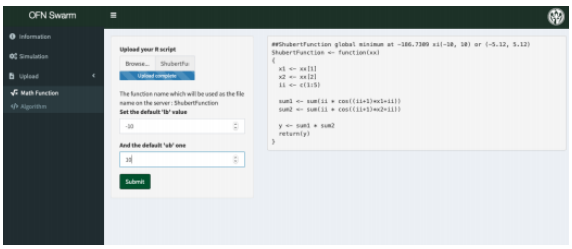
Rysunek. 10 Zrzut ekranu strony testowania w środowisku testowym. Przedstawienie wykorzystanych widgetów. Źródło: własne.

Zakładki pozwalają na wybór pomiędzy podstronami z wynikiem szczegółowym, podsumowaniem, wygenerowanym grafem oraz możliwością pobrania wyników w formacie CSV lub PDF. Co pozwala na późniejszą dalszą obróbkę danych lub przygotowanie obszernego raportu.

functionName	Z OFN	Bez OFN	U źródła
EggHolderFunction	-958.1018285	-954.2496102	-959.6507
MichalewiczFunction	-1.801303408	-1.80130341	-1.8013
Schwefel_Function	0.109273939	0.11634984	0
ShubertFunction	-186.7301957	-186.7284601	-186.7309

Tabela. 1 Przykład tabeli wyników wyeksportowanych do CSV. Źródło: własne.

Ostatnie dwie zakładki zostały dodane aby ułatwić rozwój narzędzia użytkownikowi końcowemu. Pozwalają na dodanie nowych algorytmów rojowych czy to z wykorzystaniem skierowanych liczb rozmytych czy też nie, pozwalają również na dodanie kolejnych nowych testowych funkcji matematycznych.



Rysunek. 12 Zrzut ekranu strony dodawania funkcji matematycznych. Źródło: własne.

Dzięki zastosowaniu takiej metody dodawania nowych algorytmów czy funkcji matematycznych nie ma konieczności w ingerowanie w pliki środowiska testowego. Cała praca operacyjna podczas testów może zostać wykonana z punktu widzenia interfejsu użytkownika. Pozwala to na wykorzystanie chmury czy zasobów sieciowych do przeprowadzania badań, przez co zasoby do testowania wcale nie muszą się ograniczać do zasobów naszego komputera.Twórcy Shiny R, poza przygotowaniem narzędzia idealnego do szybkiego tworzenia łatwo dostępnych interfejsów dla naszego oprogramowania w R przygotowali również prosty i skuteczny sposób publikacji naszego oprogramowania w formie strony internetowej dostarczając nam kolejny moduł biblioteki o nazwie Shiny Server. Właśnie dzięki połączeniu dobrych praktyk podczas projektowania narzędzia oraz dostępnych udogodnień przygotowanych przez społeczność języka R. Nasze środowisko testowe może być wykorzystane w nieskończenie wiele sposobów, dla różnych scenariuszy. Bez zmartwień o brak zasobów czy ograniczenia z punktu widzenia modyfikacji interfejsu.

7. Podsumowanie i wnioski

Dzięki zastosowaniu dostępnych, omówionych bibliotek języka R oraz narzędzi dostarczonych przez twórców R Studio takich jak Shiny R Server. Środowisko testowe przygotowane w środowisku R spełnia w pełni swoje założenia jako oprogramowanie, które można wielokrotnie wykorzystać i dalej rozwijać. Z przebiegu pracy jednoznacznie wynika że przygotowanie narzędzia bazując na języku R i dostępnych bibliotekach znacznie ułatwia przygotowanie środowisk testowych oraz ogólne przeprowadzenie badań przy problemach optymalizacyjnych a implementacja dodatkowych algorytmów czy funkcji matematycznych jest bardzo prosta.

Literatura

1. Matthew Campbell. „Shiny R Dashboards“. W: kw. 2019.
2. Leila Etaati. „Introduction to R“. W: czer. 2019
3. Beeley, Chris. Web application development with R using Shiny. Packt Publishing Ltd, 2013.
4. Allaire, J. "RStudio: integrated development environment for R." Boston, MA 770 2012.
5. Beeley, Chris, and Shitalkumar R. Sukhdeve. Web Application Development with R Using Shiny: Build stunning graphics and interactive data visualizations to deliver cutting-edge analytics. Packt Publishing Ltd, 2018.
6. Chang, Winston, and Barbara Borges Ribeiro. "Shinydashboard: create dashboards with'Shiny'." R package version 0.7. 2018.