Shiny Application

from package development to server deployment

Cervan Girard

28 octobre 2018

ThinkR

Vincent Guyader

Codeur Fou, formateur et expert logiciel R

Diane Beldame

Dompteuse de ~~dragons~~ données, formatrice logiciel R

Colin Fay

Data Scientist & R Hacker

Sébastien Rochette

Modélisateur, Formateur R, Joueur de cartographies

Cervan Girard

Data Scientist, Spécialiste du jonglage avec les Serveurs, Docker et ShinyProxy

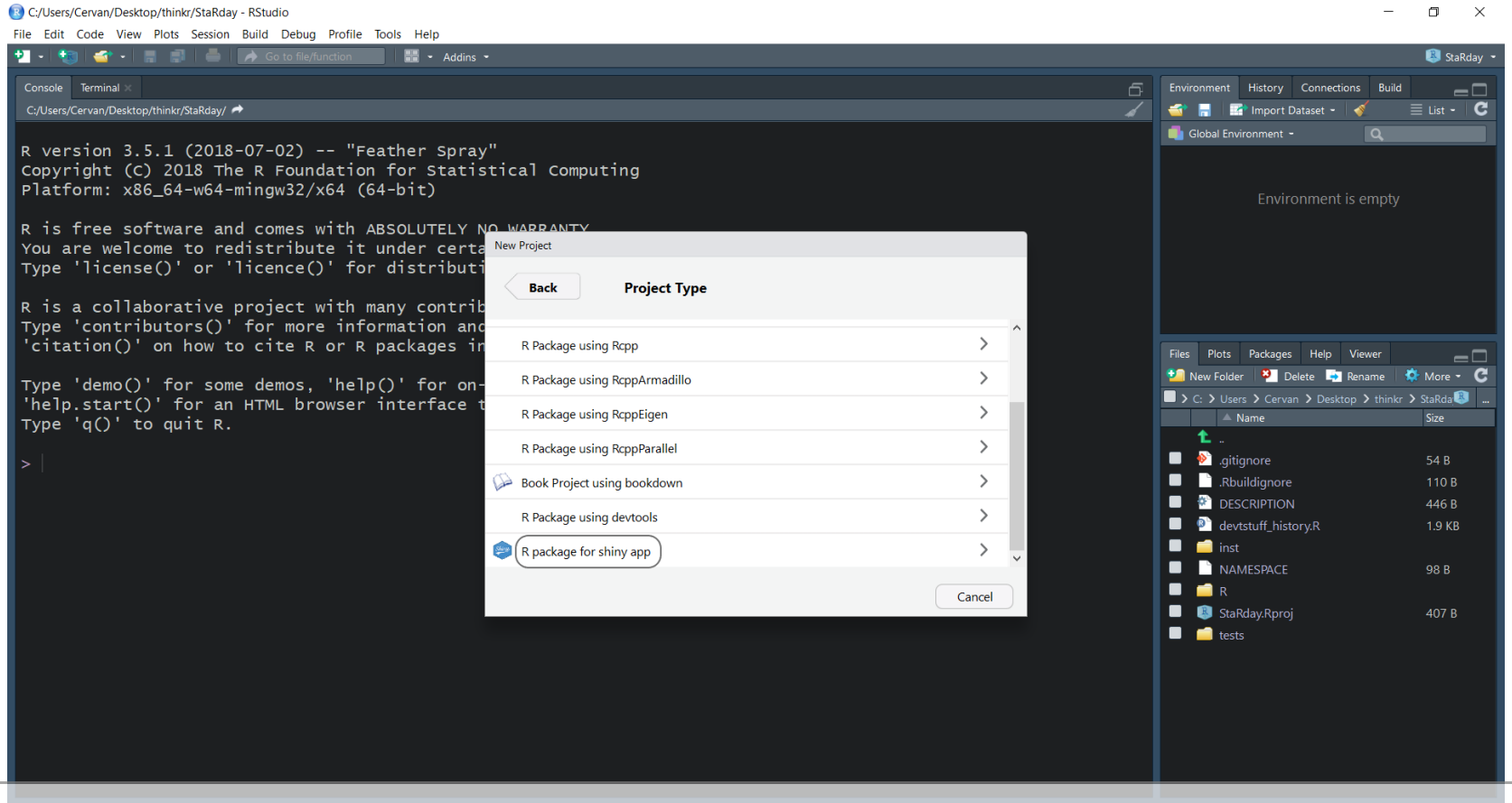
Context

- Maintain your app (more than one thousand of lines for ui.R and server.R)
- Working together on the same app (more than 4 or 5 developer)
- Not really easy without git
- Back and forth to see the changes in the application
- Coding the application too quickly
- How to deploy your application ?

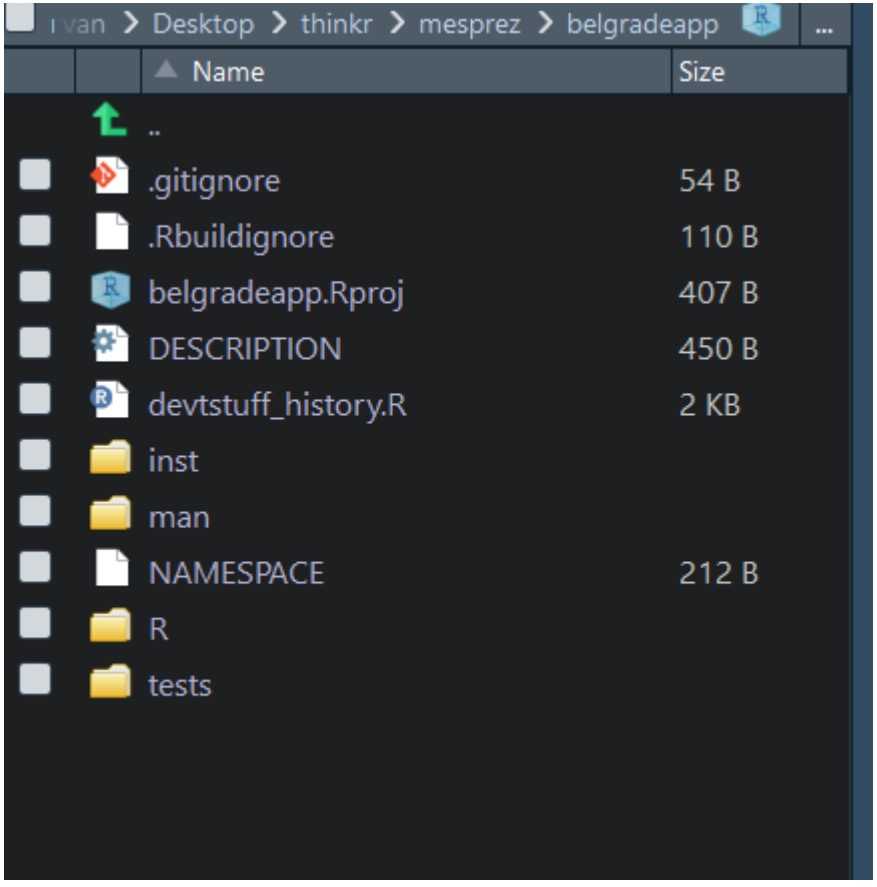
Download the package and install it.

Find the package ThinkR-open/shinytemplate

After that, create a new shiny app with this template :



What can be found inside this package



devstuff_history.R

Why do we use devstuff_hisory.R?

> So we have an history of development code

Example:

```
usethis::use_build_ignore("devstuff_history.R")
# Dependencies
usethis::use_package("shiny")
usethis::use_package("DT")
usethis::use_package("stats")
usethis::use_package("graphics")
usethis::use_package("glue")
# For data
usethis::use_data_raw()
# If you want to use the MIT licence, code of conduct, lifecycle badge,
and README
usethis::use_mit_license(name = "ThinkR")
```

In the R folder

Files:

..	
app_prod.R	176 B
app_server.R	470 B
app_ui.R	302 B
mod_first.R	674 B
mod_second.R	823 B
mod_third.R	323 B
onload.R	219 B
run_app.R	238 B
zzz.R	51 B

- app_prod
- mod_first (we use modules)
- app_server and app_ui
- run_app

app_prod file:

```
if(app_prod()){
  cat("Hey! this is prod mode")
}else{
  cat("Hey, this is dev mode :)
!")
}
```

```
> options(app.prod=FALSE)
> # TRUE = production mode,
> #FALSE = development mode
> shiny::runApp('inst/app')
Loading required package: shiny

Listening on http://127.0.0.1:5789
Hey, this is dev mode :) !
```

Always use shinymodules

Motivations:

Easier to maintain your app, to organize your project, and to call your code in another app.

UI

```
mod_firstUI <- function(id){
  ns <- NS(id)

  fluidPage(
    h3("Choose your data"),
    fluidRow(
      div(
        selectInput(ns("data"),
label = "" , choices =
c("iris","mtcars"))
      )
    )
  )
}
```

Server

```
mod_first <- function(input,
output , session, r){

  my_data <- reactive({
      switch(input$data,
             "iris" = iris,
             "mtcars" = mtcars)
  })

  output$my_summary <-
renderPrint({
   summary(my_data())
  })
}
```

In the R folder

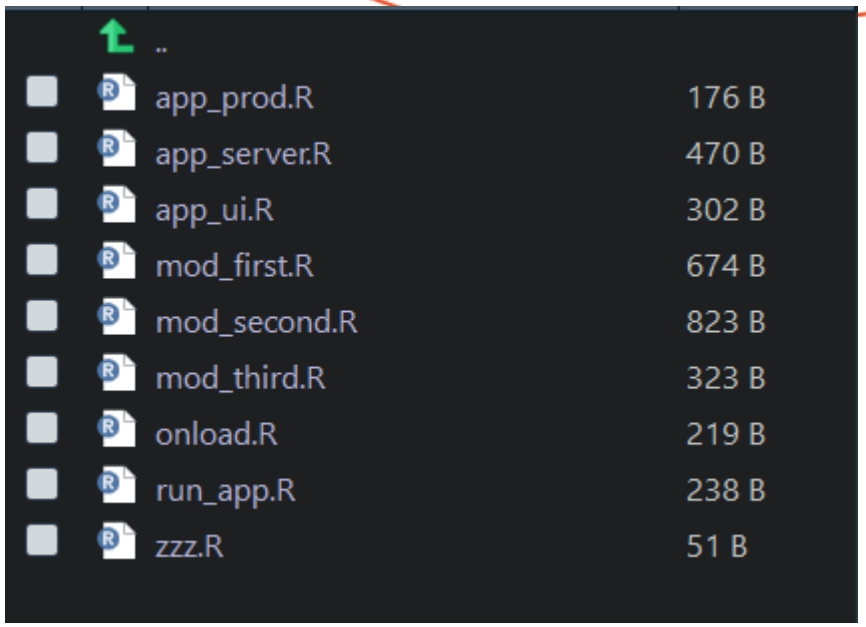
app_server

```
app_server <- function(input,
output,session) {

  r <- reactiveValues()

  if(app_prod()){
    cat("Hey! this is prod mode")
  }else{
    cat("Hey, this is dev mode :)
!")
  }

  callModule(mod_first,"first",r
= r)
}
```

app_ui()

```
app_ui <- function() {

  fluidPage(
    div(
    h2("My application", style =
"text-align:center"),
    tabsetPanel( id = "my_panel",
      tabPanel("data",
                mod_firstUI(id =
"first")),
    style = "text-align:center")
    )

  )
}
```

In the R folder



run_app.R

```
run_app <- function() {
  shinyApp(ui = app_ui(), server = app_server)
}
```

In the inst folder

```
-app
  -server.R
  -ui.R
```

> You don't need to modify them

ui.R:

```
belgradeapp:::app_ui()
```

server.R:

```
belgradeapp:::app_server
```

In the inst folder

We also find a `run_dev` folder.

`run_dev.R`:

```
# This script allow you to quick clean your R session
# update documentation and NAMESPACE, localy install the package
# and run the main shinyapp from 'inst/app'
.rs.api.documentSaveAll() # close and save all open file
try(suppressWarnings(lapply(paste("package:",
names(sessionInfo()$otherPkgs), sep = ""),
                            detach, character.only = TRUE, unload =
TRUE)), silent = TRUE)
rm(list=ls(all.names = TRUE))
devtools::document('.')
devtools::load_all('.')

options(app.prod=FALSE) # TRUE = production mode, FALSE = development mode
shiny::runApp('inst/app')
```

Working process at ThinkR

First phase: UI validation

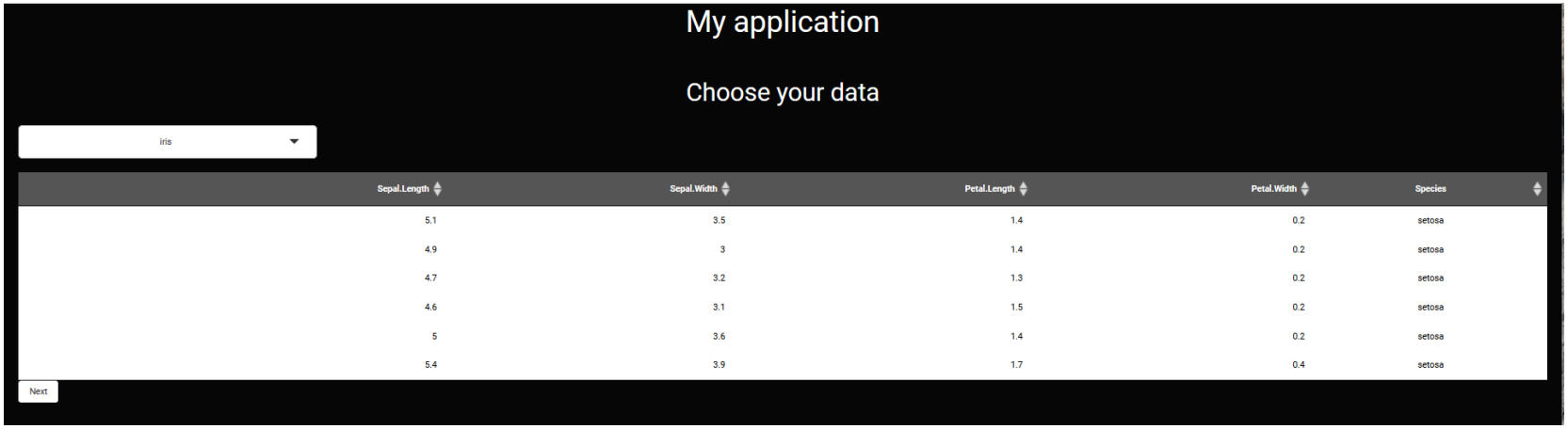
Second phase: vignettes

Third phase: module coding

First phase : UI validation

Why this first phase?

- Visual validation
- Allows a better understanding of the customer's request
- Better overall vision of the project



Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Second phase: Vignettes

Don't code your shiny app too fast!

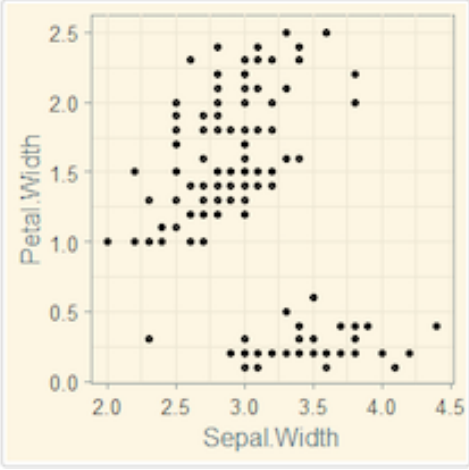
- Easy to develop a Rmd document with an example
- Propose several visual renderings with advantages and disadvantages

> Very useful for graphical rendering and expected tables

Graphics proposal

2018-10-19

Frist proposal :



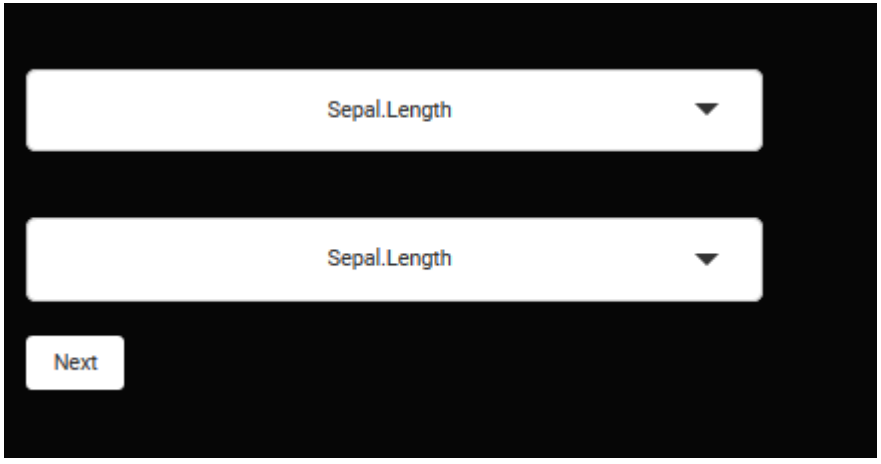
Third phase : module coding

- Easier thanks to first and second phase
- With git and modules, it's easy to work separately to avoid conflicts

> When you code your module, use run_dev.R

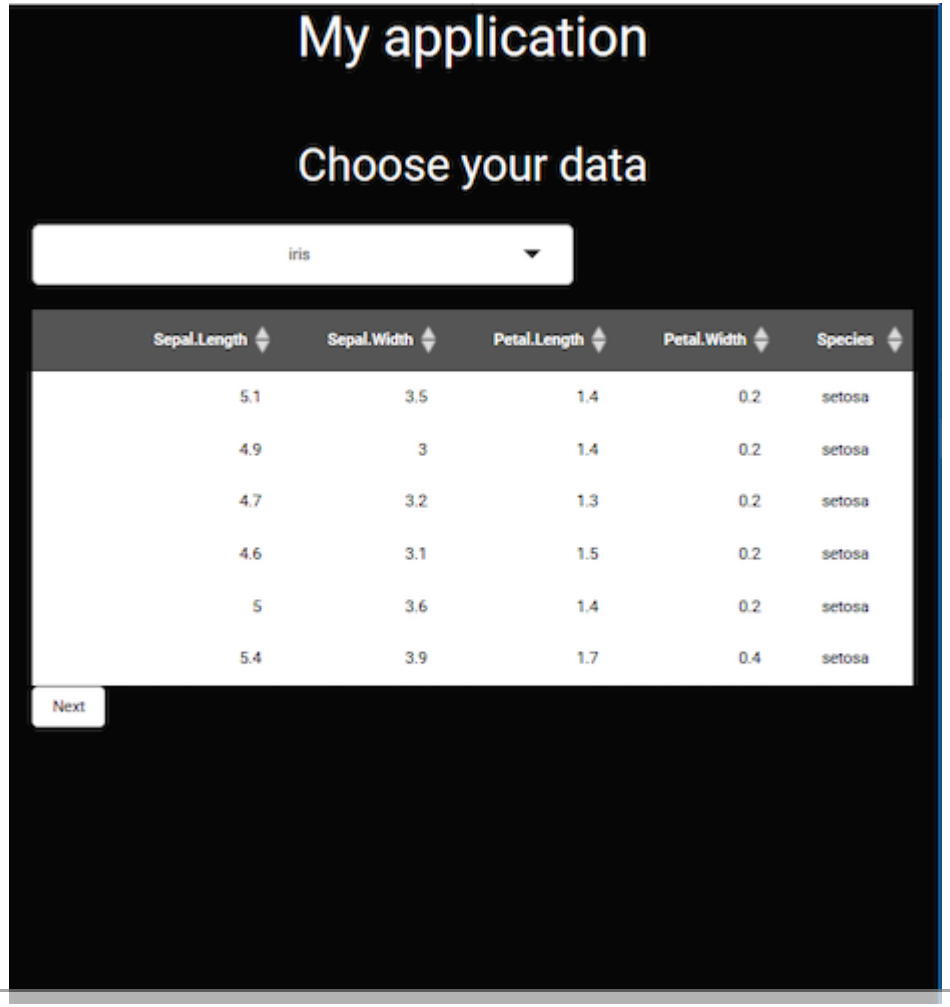
- It saves time
- No need to install your package each time

App with "Next button" :



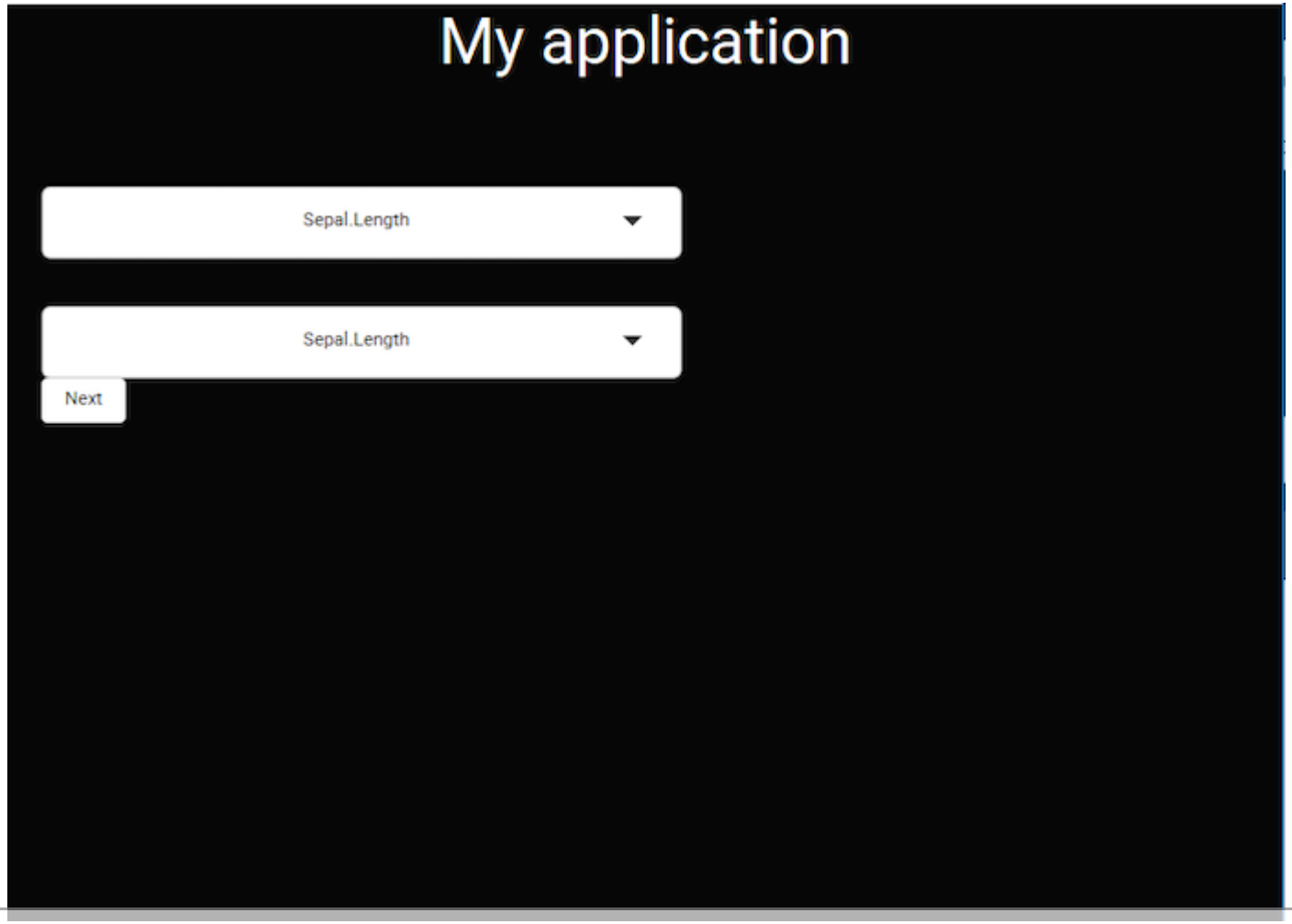
App with "next button"

First page:



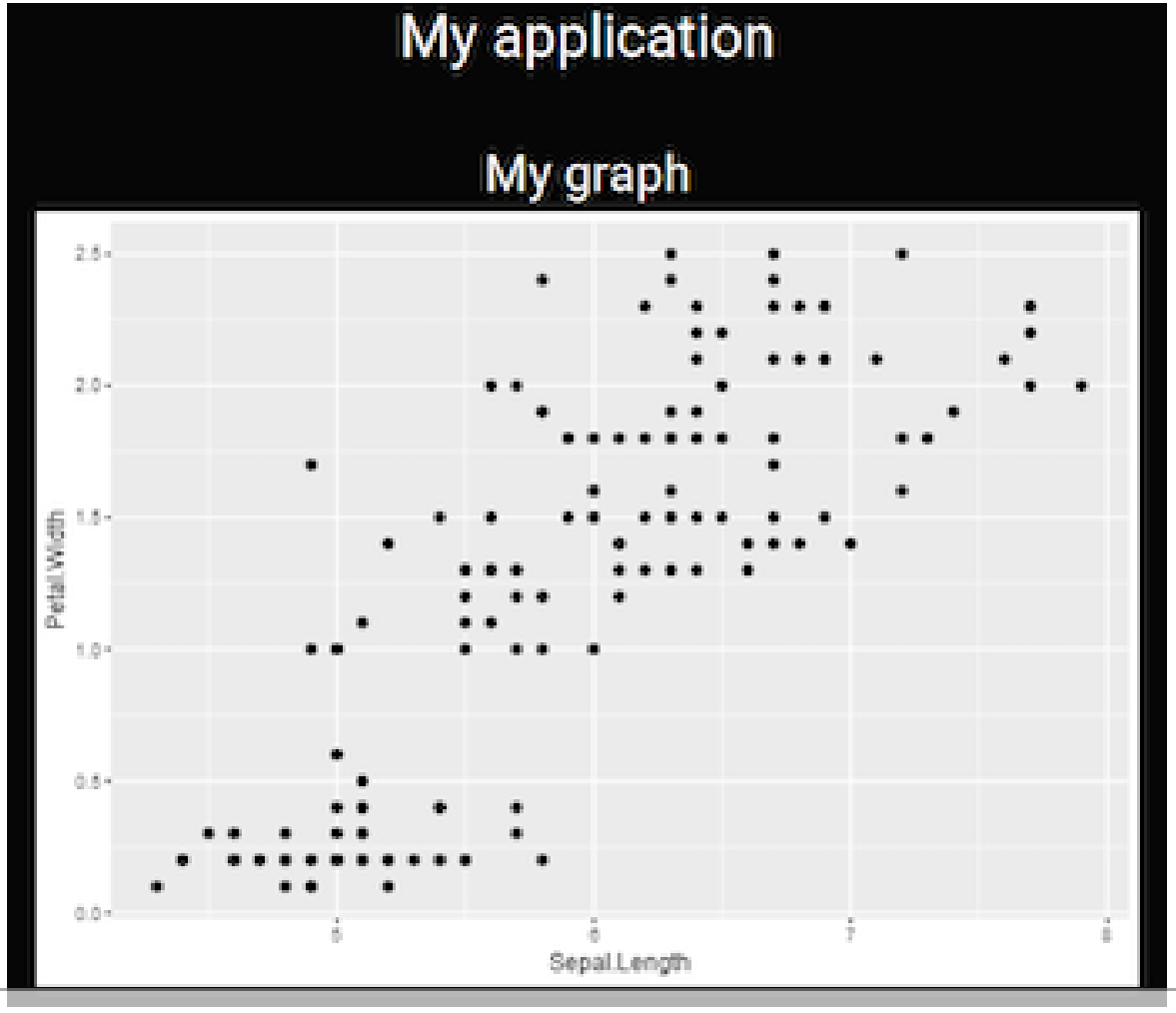
App with "next button"

Second page:



App with "next button"

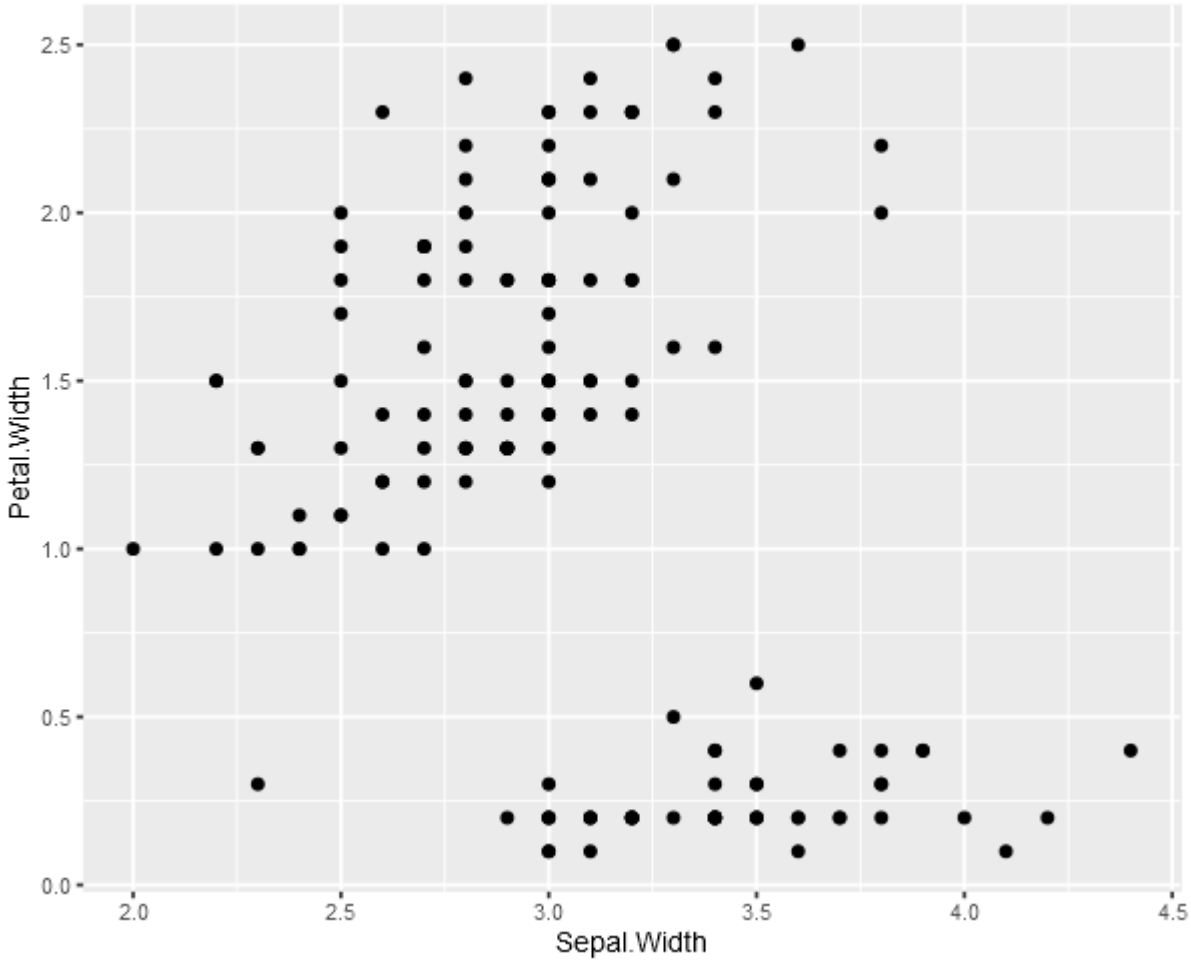
Third page:



Example for the third module:

```
.rs.api.documentSaveAll()
suppressWarnings(lapply(paste('package:',names(sessionInfo()$otherPkgs),sep=""),

rm(list=ls(all.names = TRUE))
devtools::document('.')
devtools::load_all('.')
options(app.prod=FALSE)
library(shiny)
library(DT)
if (interactive()){
  ui <- fluidPage(
    mod_thirdUI("test")
  )
  server <- function(input, output, session) {
   r <- reactiveValues(x = "Sepal.Width",y = "Petal.Width", data = iris)
   callModule(mod_third,"test", r = r)
    }
  shinyApp(ui, server)
}
```

My graph



Deploy your application

Shinyproxy

Shinyproxy

Docker

First thing to do: install Docker!

What is docker?

Why we use shinyproxy:

- Allows to launch one container per person
- Just needs Docker
- All docker advantages

Build our Shinyapp image

We need a dockerfile:

```
FROM rocker/r-ver:3.4
RUN apt-get update
RUN apt-get install -y libssl-dev libssh2-1-dev libcurl4-openssl-dev

RUN R -e "install.packages('DT')"
RUN R -e "install.packages('pacman')"
RUN R -e "pacman::p_load(shiny,  stats,    dplyr,    tidyr,    magrittr,
ggplot2)"


COPY belgradeapp*.tar.gz  /belgradeapp.tar.gz

RUN R -e "install.packages('belgradeapp.tar.gz', repos = NULL, type =
'source')"

COPY Rprofile.site /usr/local/lib/R/etc

EXPOSE 3838
CMD ["R", "-e belgradeapp::run_app()"]
```

Build our Shinyapp image

Rprofile.site file:

```
local({
   options(shiny.port = 3838, shiny.host = "0.0.0.0")
})
```

We also need our package.tar.gz.

Put everything in the same folder!

Then:

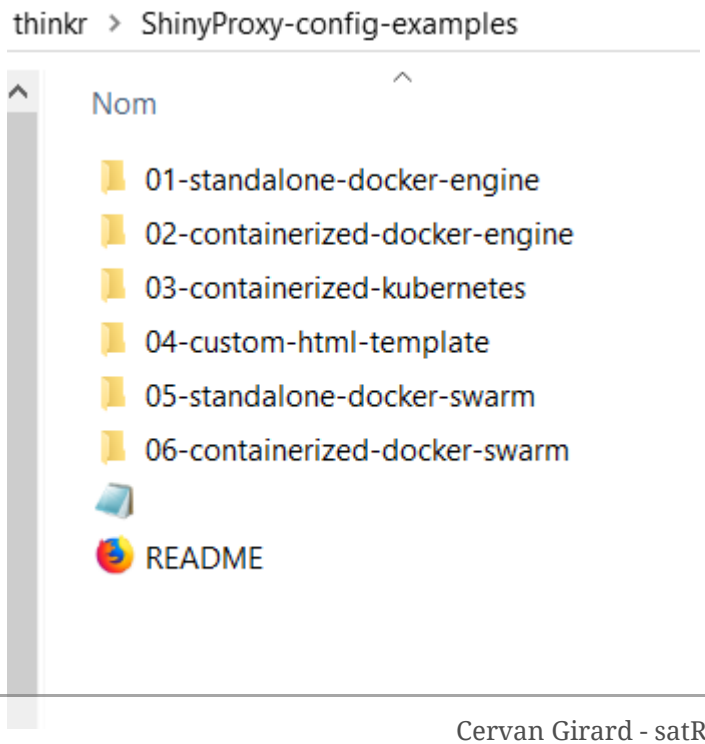
```
cd my-app/

docker build -t mon_app .
```

```
PS C:\Users\Cervan\Desktop\my_app> docker build -t mon_app .
Sending build context to Docker daemon  43.52kB
Step 1/11 : FROM rocker/r-ver:3.4
3.4: Pulling from rocker/r-ver
05d1a5232b46: Downloading [==========================================>     ]  41.33MB/45.31MB
38a78ed7cc6f: Downloading [==========>                                     ]  41.47MB/205.7MB
```

Get shinyproxy

On github:

```
cd thinkr/

git clone https://github.com/openanalytics/ShinyProxy-config-examples.git
```



Shinyproxy and docker

Dockerfile

```
FROM openjdk:8-jre

RUN mkdir -p /opt/shinyproxy/
RUN wget https://www.shinyproxy.io/downloads/shinyproxy-2.0.5.jar -O
/opt/shinyproxy/shinyproxy.jar
COPY application.yml /opt/shinyproxy/application.yml

WORKDIR /opt/shinyproxy/
CMD ["java", "-jar", "/opt/shinyproxy/shinyproxy.jar"]
```

Shinyproxy and docker

yaml

```yaml
proxy:
  port: 8080
  authentication: simple
  admin-groups: admins
  users:
  - name: jack
    password: password
    groups: admins
  - name: jeff
    password: password
  docker:
    url: http://localhost:2375
  specs:
  - id: 01_hello
    display-name: Hello Application
    description: Application which demonstrates the basics of a Shiny app
    container-cmd: ["R", "-e", "shinyproxy::run_01_hello()"]
    container-image: openanalytics/shinyproxy-demo
```

Yaml

Let's modify this file:

```yaml
proxy:
  port: 8080
  authentication: none
  admin-groups: admins
  docker:
    internal-networking: true
  specs:
  - id: Belgrade_application
    display-name: Belgradeapp
    description: Application which demonstrates the basics of a Shiny app
    container-cmd: ["R", "-e", "belgradeapp::run_app()"]
    container-image: mon_app
    container-network: sp-example-net

logging:
  file:
    shinyproxy.log
```

Last step

Deploy our app!!!

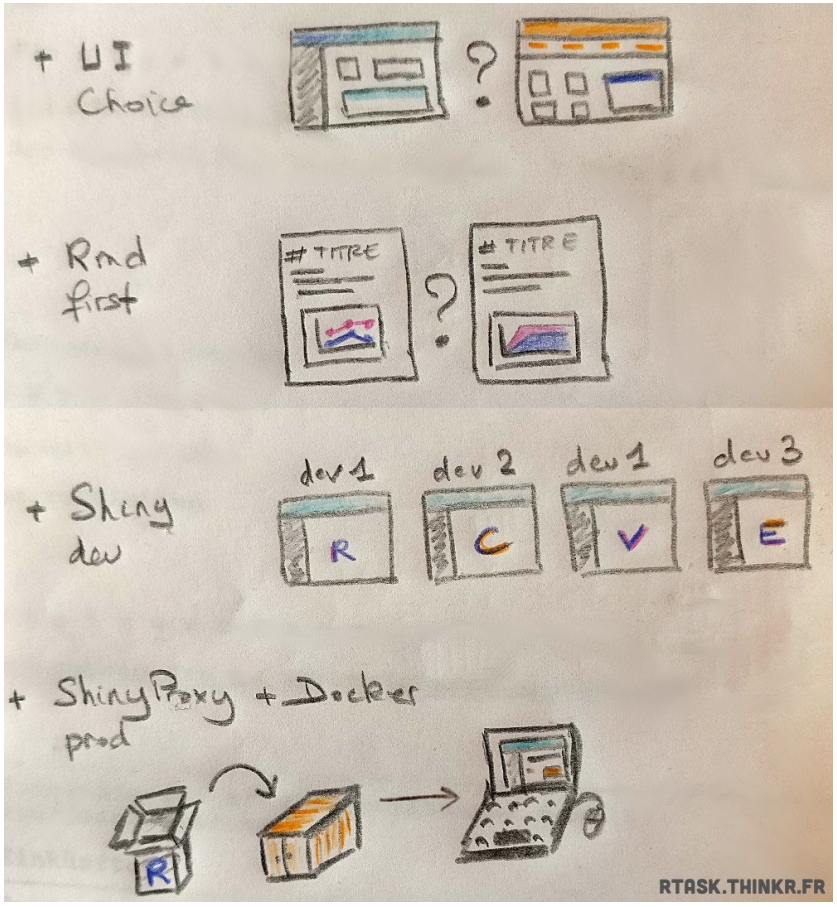
We also need folder wtih a dockerfile and application.yaml file.

```
### Create docker network
sudo docker network create sp-example-net

###Build the image
docker build -t mon_ShinyProxy .

### Run it
sudo docker run -d -v /var/run/docker.sock:/var/run/docker.sock --net sp-example-net -p 8080:8080 mon_ShinyProxy
```

ShinyProxy

- Belgradeapp
  Application which demonstrates the basics of a Shiny app

Conclusion


+ UI
Choice
+ Rmd
first
TITRE
TITRE
+ Shiny
dev
dev 1
dev 2
dev 1
dev 3
R
C
V
E
+ ShinyProxy + Docker
prod
R
RTASK.THINKR.FR

Would you come to Paris?

satRday Paris

23th February 2019

AgroParisTech

Thank you!

Cervan Girard

cervan@thinkr.fr

http://thinkr.fr