

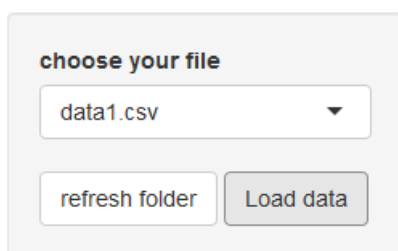
Making a shiny app that can be modified/updated by external people

Sometimes, you may want to have a shiny app whose code and data can be updated or modified directly by external people. This may enhance engagement and ownership.

This document presents one way to do so with a minimal example.

We are going to make the following app:

Super Shiny App!



Name	Gender	Super.power	Score.of.something
Sam	Male	Magnetic force	4
Dave	Male	Self-duplication	2
Emily	Female	Kim Jung Gi-ness	2
Lucia	Female	Mafia superstrength	4
Faheem	Male	Gravity manipulation	2
Carlos	Male	Mind-control	2

This app allows the user to visualize the first 6 lines of whatever data that has been placed in a drobox folder. It is pretty useless, but that's ok. The great stuff will come from you!

The first thing we need is a way to access dropbox from R. To do so, we will use the package rdrop2:

```
library(rdrop2)
token <- drop_auth()
saveRDS(token, file = "path_to_shiny_server_folder/token.rds")
```

Here we are asking dropbox for a token that grants access to our dropbox folders and we then save this token on the shiny app folder of our shiny server. You don't want to give it to external people. Keep it safe!

app.R

Then we want to create our file app.R. We will place it in the same folder on our shiny server. Here is the code inside this file:

```
library(shiny)
library(rdrop2)

token <- readRDS("token.rds")

folder_path <- "path_to_dropbox_folder"
```

We load the two necessary libraries, we read and save the token that gives access to our dropbox, and we define the path of the dropbox folder where the data (and scripts of the app) will be placed. The idea is to share this folder with external people so that they can do whatever they want with the app.

```
drop_download(paste0(folder_path, 'myServer.R'), dtoken=token, overwrite = T)
drop_download(paste0(folder_path, 'myUI.R'), dtoken=token, overwrite = T)
drop_download(paste0(folder_path, 'functions.R'), dtoken=token, overwrite = T)
```

Using the token, we download from our dropbox the files myServer.R, myUI.R and functions.R that we will soon create and that some external people may have modified. Warning: As is, this is probably not super safe. You probably want to check these files before downloading them, because someone good in IT could probably do quite a bit of damage if they wanted. But that's fine here, for our example.

```
source("functions.R")

ui <- shinyUI({
  source('myUI.R', local=TRUE)$value
})

server <- function(input, output, session) {
  source("myServer.R", local=TRUE)
}

shinyApp(ui = ui, server = server)
```

We finally run these files to load the functions inside functions.R, build the user interface defined in myUI.R and build the server side defined in myServer.R... before running the app.

myUI.R

The code for the user interface, which is inside the file myUI.R is the following:

```
fluidPage(
  titlePanel("Super Shiny App!"),

  sidebarLayout(

    sidebarPanel(
      selectInput("choose_file", "choose your file", choices = "spotcheck.csv"),
      actionButton("refresh", "refresh folder"),
      actionButton("load_data", "Load data")
    ),
    mainPanel(
      tableOutput("super_table")
    )
  )
)
```

It has a selection element (choose_file), two buttons (refresh and load_data) and one output (super_table)

Because we (may) want it to be modifiable by an external person and we place the file myUI.R in the dropbox folder.

myServer.R

The code for the server side will contain the three chunks that follow:

```
observe({
  input$refresh
```

```
files<-drop_dir(folder_path, dtoken = token)$name
updateSelectInput(session, "choose_file", choices = files)
})
```

The chunk tells the server that when the button “refresh” is clicked, it should observe what’s in the dropbox folder and update the “choose_file” selection element accordingly

```
data <- reactive({
  input$load_data
  raw_data <- drop_read_csv(paste0(folder_path,input$choose_file), dtoken = token)
  process(raw_data)
})
```

Here we tell the server that when the button “load_data” is pressed, it should read and process the data selected by the user. `isolate()` is necessary to prevent the server from reading the file before the button is pressed.

```
# show data
output$super_table <- renderTable({
  data()
})
```

Finally we ask the server to show the processed data

functions.R

for this exemple, functions.R is a totally useless file:

```
process<-function(myData){
  return(head(myData))
}
```

... but maybe you’ll find better things to put inside.

the data

Last thing, you need to have some data in the folder. Here is the data I used ;)

```
read.csv("data1.csv")
```

##	Name	Gender	Super.power	Score.of.something
## 1	Sam	Male	Magnetic force	4
## 2	Dave	Male	Self-duplication	2
## 3	Emily	Female	Kim Jung Gi-ness	2
## 4	Lucia	Female	Mafia superstrength	4
## 5	Faheem	Male	Gravity manipulation	2
## 6	Carlos	Male	Mind-control	2
## 7	Shannon	Female	Telekinesis	3
## 8	Nicolas	Male	Light reader	2
## 9	Tina	Female	Color emperor	2
## 10	Alex	Male	Catalan elasticity	3