University of Zurich[UZH]

Tell me and I will forget, show me and I will remember, involve me and I will understand

How to create interactive dashboards

Dr. Markus Meierer | markus.meierer@business.uzh.ch --- Patrick Bachmann | patrick.bachmann@business.uzh.ch

Learning goals

After this lecture you should be able to:

1. Run a shiny app and know how its structure
2. Deploy a shiny app
3. Build a user interface for a shiny app

What is a Shiny app and how to run it?

What is dashboard?

- In information technology, a dashboard is a user interface that, somewhat resembling an automobile's dashboard, **organizes and presents information in a way that is easy to read.** However, a computer dashboard is more likely to be interactive than an automobile dashboard (unless it is also computer-based).

- To some extent, most graphical user interfaces (GUIs) resemble a dashboard. However, some product developers consciously employ this metaphor (and sometimes the term) so that the user instantly recognizes the similarity.

http://searchcio.techtarget.com/definition/dashboard

Why interactive dashboards?

- With dashboards managers have access to
  1. Up-to-date reports on latest figures (**presentation**)
  2. Self-service analytics tools which enable data **exploration** to look at the impact of contextual factors by using interactive dashboard elements
- However, dashboards are by no means a panacea. Be aware of **possible traps that might lead to wrong decisions**:
  1. Importance trap (Focus on relevant metrics!)
  2. Context trap (Context implies on what to focus, e.g. absolute or relative figures.)
  3. Causality trap (Is a relationship based on correlation or causation?)

https://hbr.org/2017/01/3-ways-data-dashboards-can-mislead-you

What is Shiny?

High-level summary:

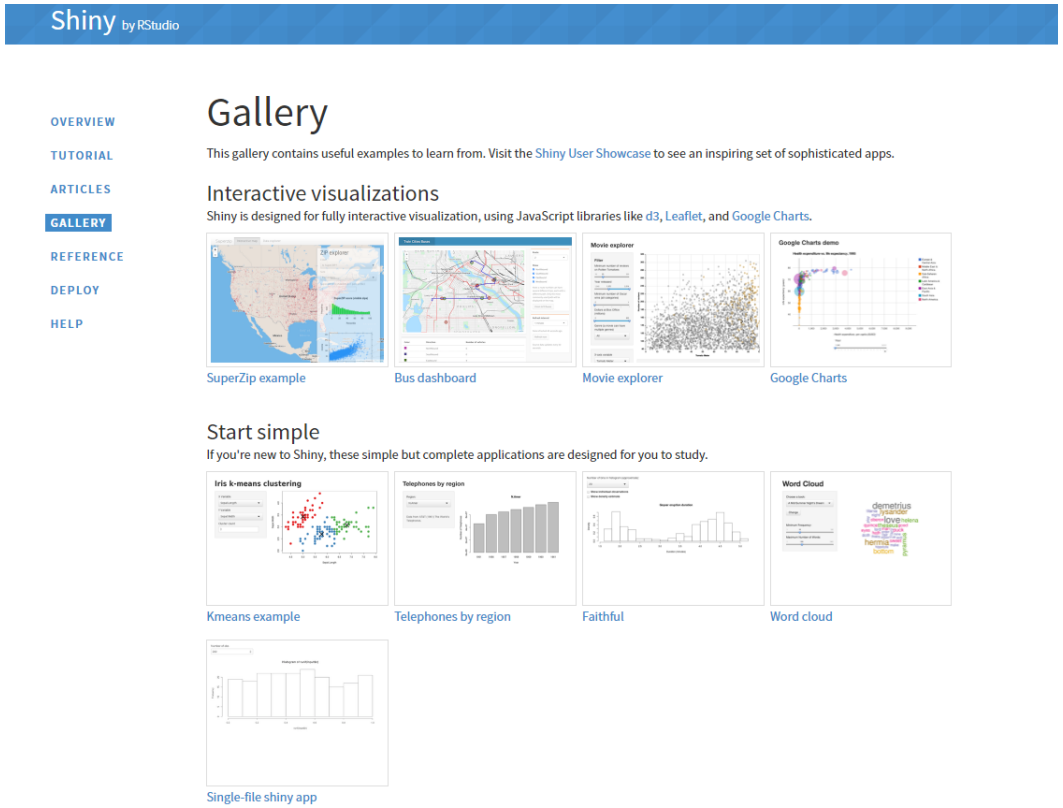
- Shiny is a platform for creating interactive R programs embedded into a webpage.
- Shiny is made by the folks at Rstudio.

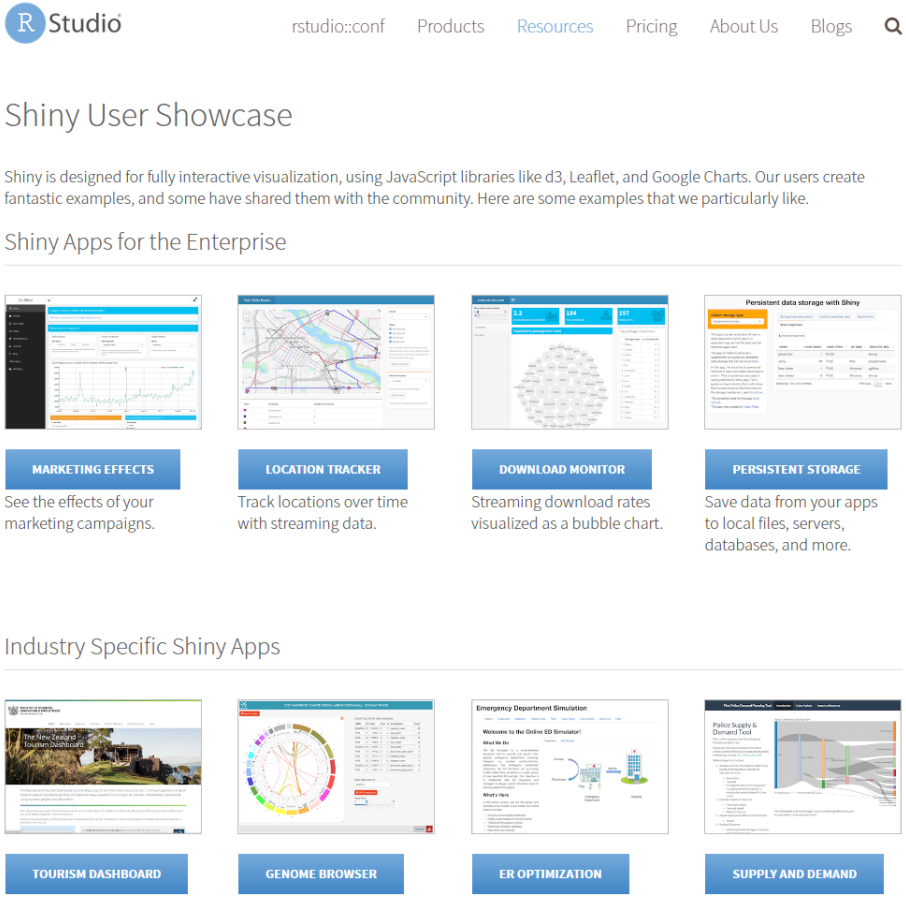
Some more details:

- A shiny app is a webpage (user interface – UI) connected to a computer running a live R session (server).
- Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code)

http://shiny.rstudio.com/images/shiny-cheatsheet.pdf

What can Shiny do? Look at (1) the Shiny gallery and (2) the Shiny user showcases





https://shiny.rstudio.com/gallery/, https://www.rstudio.com/products/shiny/shiny-user-showcase/

Requirement: Install Shiny (and optionally, be open a learn a tiny bit of web programming)

1. Download and install the R package `shiny`

2. Optional: **Web programming**:

   Shiny doesn't really require it, but as with all web programming, a little knowledge of HTML, CSS, and JS is helpful:

   - HTML gives a web page structure and sectioning as well as markup instructions
   - CSS gives the style
   - JS for interactivity

It's not too complicated too write your own Shiny app: The simplest possible app has 4 lines of code

1 line for …

… loading the R package **`shiny`**

… creating the **user interface** (HTML page)

… for the **backend** (i.e. how to build and rebuild the R objects displayed in the UI)

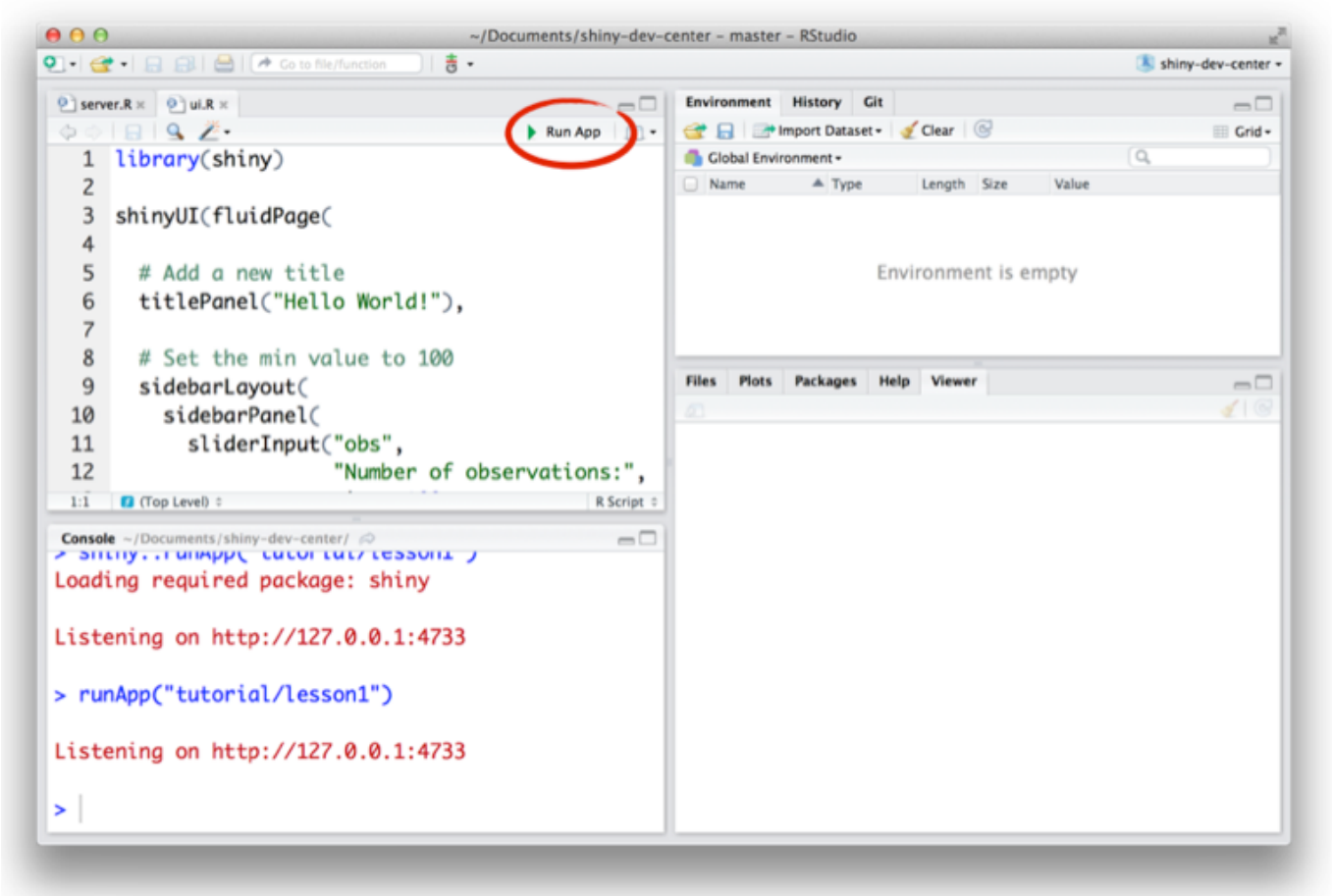
… **running the app**

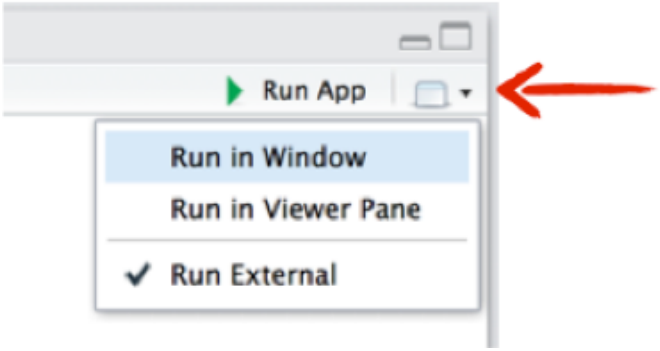
Minimal Example

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

http://zevross.com/blog/2016/04/19/r-powered-web-applications-with-shiny-a-tutorial-and-cheat-sheet-with-40-example-apps/

In Rstudio you can decided if your Shiny App should run within RStudio or in an external browser





There are many other options how to run your Shiny app locally

a. If you're developing in RStudio, and have a multi-page app you can open either `ui.R` or `server.R` and click on **Run App in the top right.**

b. For a single page app you can use the function `shinyApp()` where you specify the ui and the server, then **Run App appears in the top right.**

c. For a single or multi-page app you can use the **function** **`runApp()`** where you specify the directory your `app.R` or `ui.R` / `server.R` files are housed in.

Stop a Shiny App by pressing the Stop button

There are versions of the `runApp()` function that are designed to run your Shiny app if the files are hosted on Github or any other webspace (i.e. `runGitHub()`, `runGist()`, and `runURL()`)

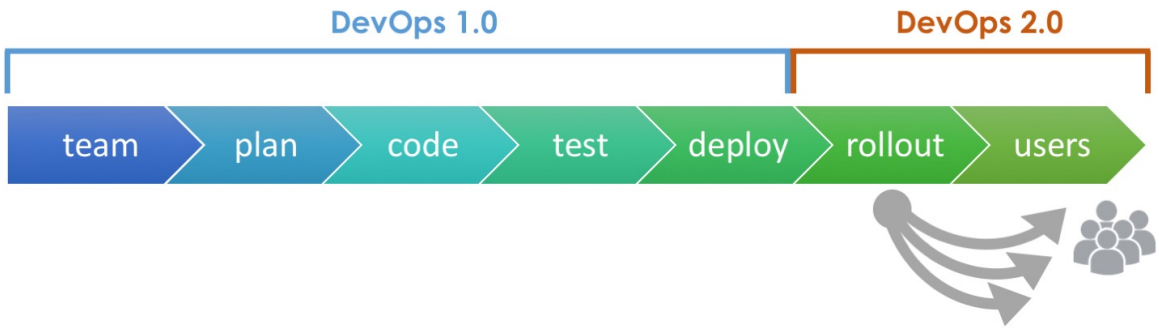
Exercise
What is a Shiny app and how to run it?

1. Install and load the R package “shiny”.
2. Create and run your first Shiny app.
   a) Setup a Shiny app by using the Rstudio menu “New File” → “Shiny Web App …”
   b) Create a single file Shiny app, name it “My_very_first_Shiny_app”, specify a folder, and press CREATE. Use the sample Shiny app code that RStudio loads when creating a new Shiny app.
   c) Run you app in your “Viewer Pane”.

Deploy a Shiny app

What is deployment and why should you bother?

- To deploy (from the French employer) is "**to spread out or arrange strategically**".
- Long used in the context of military strategy, it has now gained currency in information technology. In its IT context, **deployment encompasses all the processes involved in getting new software or hardware up and running properly in its environment**, including installation, configuration, running, testing, and making necessary changes.
- The word **implementation** is sometimes used to mean the same thing.



http://blog.launchdarkly.com/devops2

http://whatis.techtarget.com/definition/deploy

You have 2 fundamental options to deploy your app

1. Deploy your Shiny app for local use as a desktop application
2. Deploy your Shiny app on a (Web/Intranet) server

(1) Deploy your Shiny app for local use as a desktop application: The R community provides a solution

2016/04/12

Desktop DeployR

I'm going to be giving a talk this Thursday at my local R/Data Science Meetup about my method for deploying self contained desktop R applications. Since my original post on the subject (over 2 years ago!) I've made **many** of improvements thanks to the many useful comments I received and my own "dog-fooding".

So many in fact that the framework is a project in its own right, which I'm calling *DesktopDeployR*. This post will officially document the changes I've made and share the project to the greater R/Data Science community.

If you haven't already, I recommend reading my original post to understand the fundamentals of how such deployments are done.

For the impatient, the TL;DR summary is: on a Windows system, use R-Portable and Windows Scripting Host to launch a Shiny app in a user's default web browser.

https://oddhypothesis.blogspot.ch/2016/04/desktop-deployr.html

(2) Deploy your Shiny app on a (Web/Intranet) server

1. Running on **shinyapps.io**

   This site is managed by RStudio and is free for small apps with limited visits and scales up in paid versions.

2. Running your **own Shiny Server Open Source**

   There is a free, open source version of the Shiny server that you can run on, for example, Amazon Web Services or your own server.

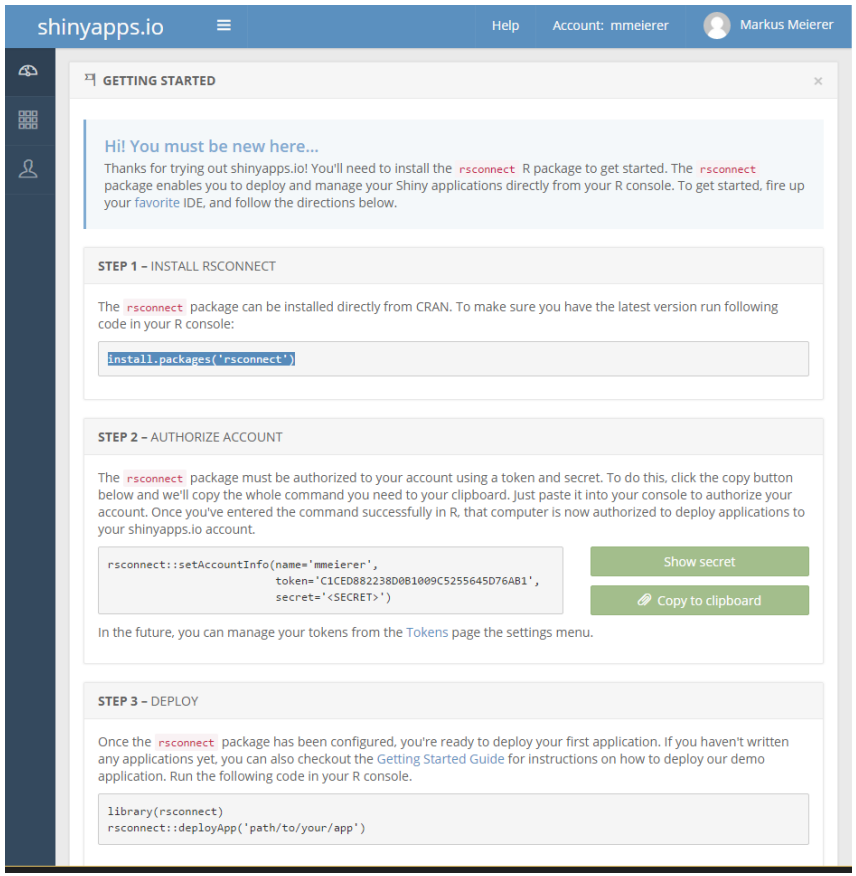
3. Running **Shiny Server Pro**

   RStudio also sells a yearly subscription to Shiny Server Pro that provides security, admin and other enhancements when compared to the open source version.

http://zevross.com/blog/2016/04/19/r-powered-web-applications-with-shiny-a-tutorial-and-cheat-sheet-with-40-example-apps/

Shinyapps.io

1. Create a (free) **account** at www.shinyapps.io
2. After setting up your account, follow the steps in **the "Getting Started" tutorial** (click on the "dashboard" icon)
    1. Install the `rsconnect` package
    2. Authorize your account by specifying your name, token, and secret (see code snippet in the "Getting strated" tutorial)
3. Use the **Publish icon** in the Rstudio IDE or run `rsconnect::deployApp("<path to directory>")`



Exercise
Deploy a Shiny app

1. Create a free account on www.shinyapps.io and pick any name for your personal subdomain.
2. Login to www.shinyapps.io and follow the “Getting Started” tutorial to setup RStudio thus you can upload Shiny Apps to the platform.
3. Publish your Shiny app “My_very_first_Shiny_app” on www.shinyapps.io.

Understand the structure of a Shiny app

It's not too complicated too write your own Shiny app: The simplest possible app has 4 lines of code

1 line for …

… loading the R package **`shiny`**

… creating the **user interface** (HTML page)

… for the **backend** (i.e. how to build and rebuild the R objects displayed in the UI)

… **running the app**

Remember this slide?

Minimal Example

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

http://zevross.com/bl… …vered-web-applications-with-shiny-a-tutorial-and-cheat-sheet-with-40-example-apps/

Every Shiny app is structured into 2 parts: (1) user interface and (2) backend

The distinction between input and output objects will be explained on the next slide

Add inputs to the UI with ***Input()** functions

Add outputs with ***Output()** functions

Tell server how to render outputs with R in the server function. To do this:

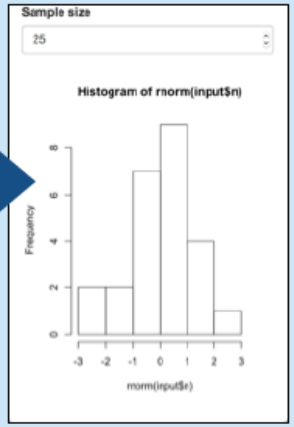
1. Refer to outputs with **output$<id>**
2. Refer to inputs with **input$<id>**
3. Wrap code in a **render*()** function before saving to output

```
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

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



Save your template as **app.R**. Alternatively, split your template into two files named **ui.R** and **server.R**.

```
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

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

```
# ui.R
fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
```

```
# server.R
function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
```

ui.R contains everything you would save to ui.

server.R ends with the function you would save to server.

No need to call **shinyApp()**.

Shiny Cheat Sheet (2016)

Further, you should be able to distinguish between (1) input objects and (2) output objects

- **Outputs** can be any object that R creates and that we want to display in our app.
  - Type-specific placeholders for each output need to be added to the user interface (e.g. a table or map).
  - The actual form of the output is defined on the server side (e.g. data shown in the table or geographic region shown on the map).
- **Inputs** introduce interactivity, i.e. they enable users to request customized outputs.
  - Input objects are added to the user interface (e.g. dropdown lists).
  - The state of input objects is then passed on and used on the server side (e.g. field of dropdown list)

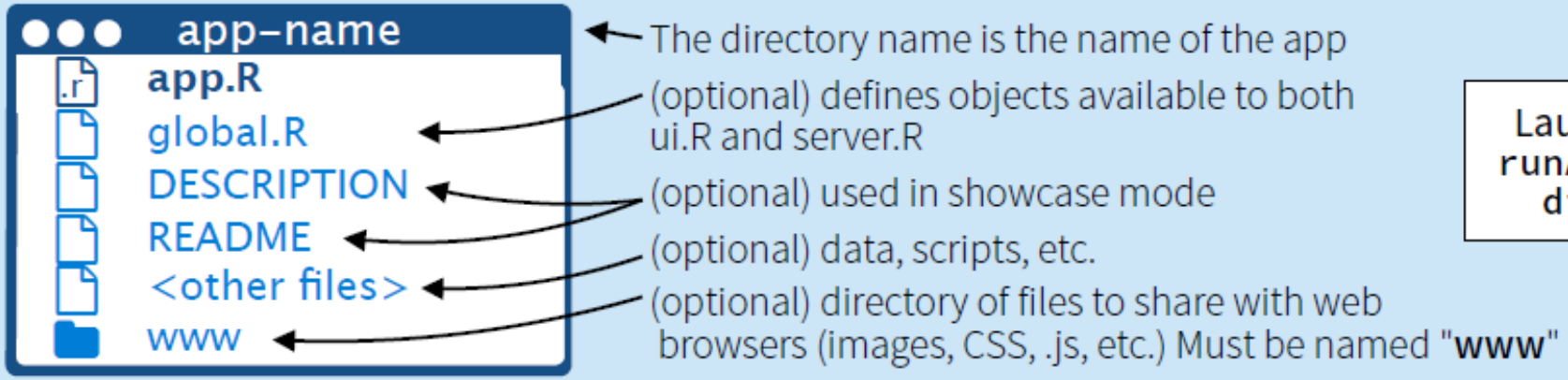
Without INPUTS a Shiny app resembles more a static R markdown report.

Steps to build an interactive Shiny app

1. Start a Shiny app with the RStudio **template** to sets up the UI and server part.
2. In the UI part, setup the **general structure and design** of the Shiny app (.e.g., fluidPage())
3. In the UI part, setup the **input widgets**
4. In the UI part, setup **placeholders for the outputs**
5. In the server part, **define the outputs** – in particular, how the inputs shape the output

How to store the files of a Shiny project?

Save each app as a directory that contains an **app.R** file (or a **server.R** file and a **ui.R** file) plus optional extra files.



Launch apps with `runApp(<path to directory>)`

Shiny Cheat Sheet (2016)

Exercise
Understand the structure of a Shiny app

Be smart, execute first the R script and check what the code does.

1. Use the R script “create_map_NO_SHINY.R” to create a Shiny app, which displays the location of the customers listed in the CSV file on a map. For the moment, the interactivity of the Shiny app is limited to the built-in zoom function of the map.

   - Start writing a Shiny app “customer map”, i.e. first, add the relevant libraries.
   - Define the <u>user interface</u> part, thus the output is defined as a leaflet map.
   - Define the <u>server part</u> based on the existing code to create a app,. This includes among others, (1) the rendering of the map object and (2) the assignment of a name under which it is provided as an output.

University of Zurich[UZH]

Tell me and I will forget, show me and I will remember, involve me and I will understand

How to create interactive dashboards?

Dr. Markus Meierer | markus.meierer@business.uzh.ch --- Patrick Bachmann | patrick.bachmann@business.uzh.ch

Learning goals

After this lecture you should be able to:

1. Run a shiny app and know how its structure
2. Deploy a shiny app
3. Build a user interface for a shiny app