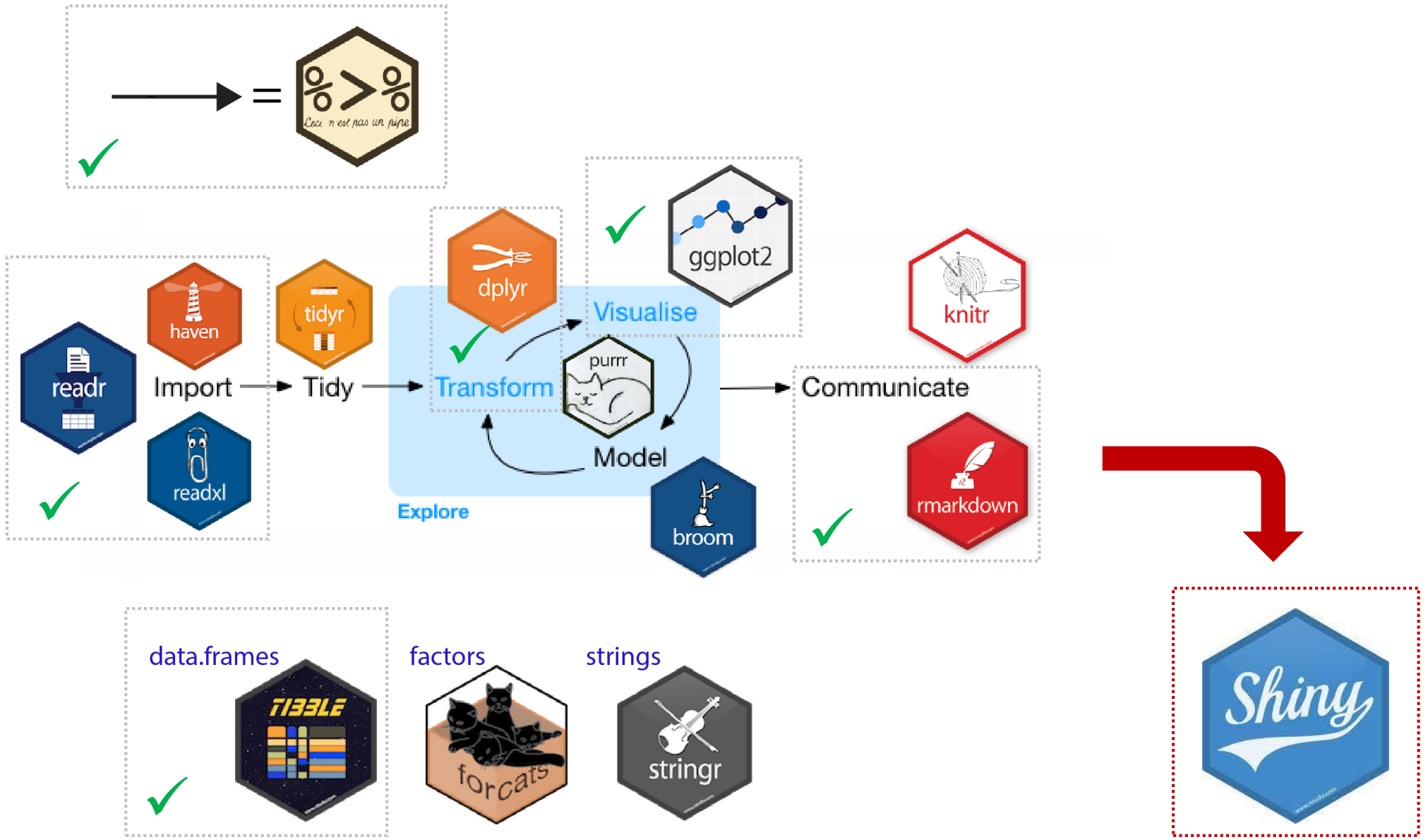
HOST
Hochschule Stralsund

Data Science

Deployment

Gero Szepannek


Ceci n'est pas un pipe
ggplot2
dplyr
tidyr
haven
readr
readxl
knitr
Visualise
purrr
Import
Tidy
Transform
Model
Communicate
Explore
broom
rmarkdown
data.frames
factors
strings
tibble
forcats
stringr
Shiny

A Student‘s Work from Last Year…

https://digitour.hochschule-stralsund.de/shiny/apps/app19/
User: user
Password: pass

Shiny

Basic Idea:

- Hide R code behind a Website
- …make (predefined) analyses available to non-technical users.

Shiny

Basic Idea:

- Hide R code behind a Website
- …make (predefined) analyses available to non-technical users.

- Structure of the code:
  1. Server
  2. User interface (UI)

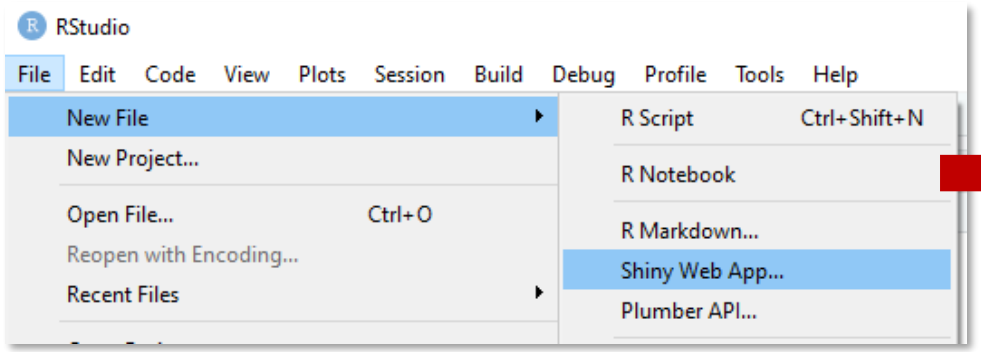
- Important concept;
  - Reactivity

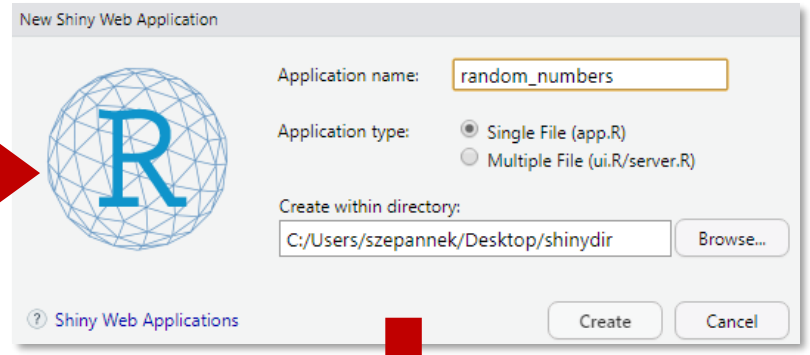
Task

Generate n random numbers and draw a histogram!

J. Smuller (2018): R in Projekten anwenden für Dummies, Wiley – VCH, chp. 4.

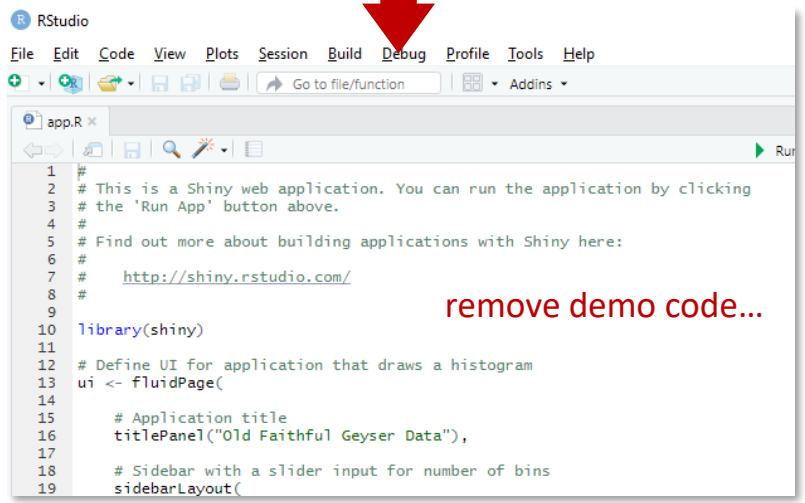
A Template…





Basic structure of a shiny app:

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



remove demo code…

User Interface: Design the Page

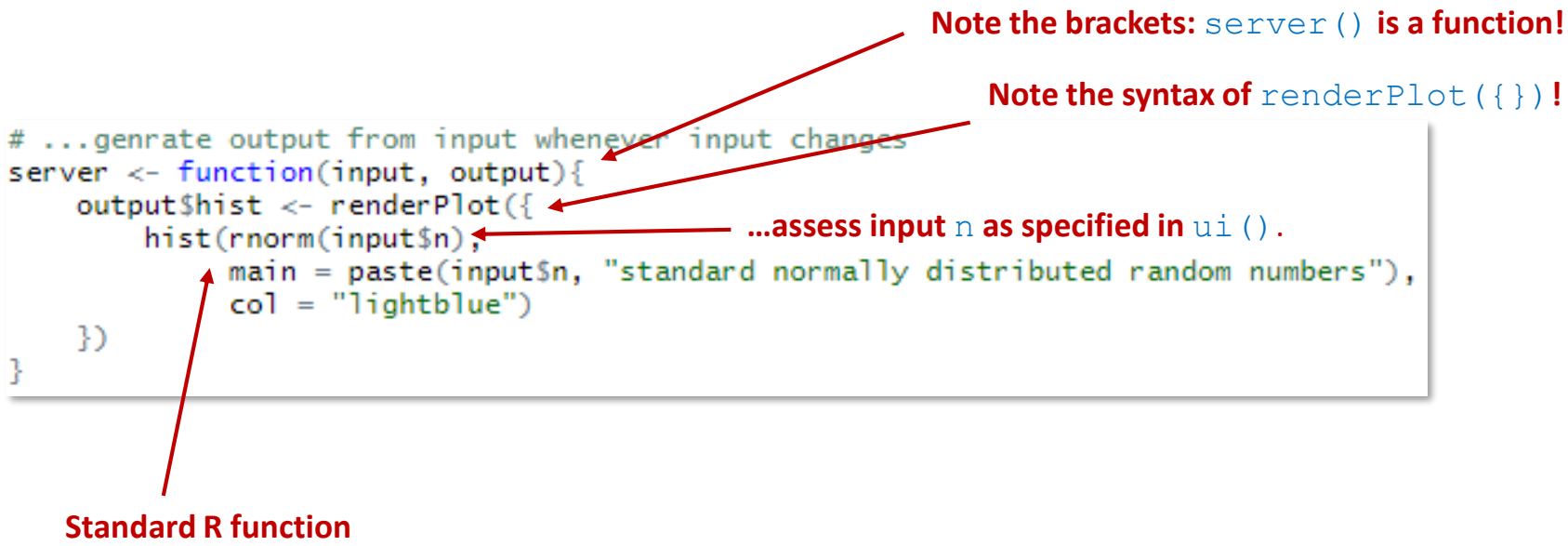
```
# design input and output window
ui <- fluidPage(                    # ...adapts with of the browser

    sliderInput(inputId = "n",      # ...can be assessed via input$n from server()
                label = "Anzahl an Zufallszahlen",
                value = 50, min = 5, max = 1000000),

    plotOutput("hist")              # ...creates a window for the output graphic
)
```

Don‘t miss the comma!
Note, both `sliderInput` and `plotOutput` are arguments to `fluidPage()`.

Server: Create Output from Input



…bringing all together

```
1  library(shiny)
2
3  # design input and output window
4  ui <- fluidPage(                  # ...adapts with of the browser
5
6      sliderInput(inputId = "n",    # ...can be assessed via input$n from server()
7                  label = "Anzahl an Zufallszahlen",
8                  value = 50, min = 5, max = 1000),
9
10     plotOutput("hist")            # ...creates a window for the output graphic
11 )
12
13 # ...genrate output from input whenever input changes
14 server <- function(input, output){
15     output$hist <- renderPlot({
16         hist(rnorm(input$n),
17              main = paste(input$n, "standard normally distributed random numbers"),
18              col = "lightblue")
19     })
20 }
21
22 shinyApp(ui = ui, server = server)  # ...runs app
```

Define User Interface

Define Server

Run App

Reactivity

```
1  library(shiny)
2
3  # design input and output window
4  ui <- fluidPage(                 # ...adapts with of the browser
5
6      sliderInput(inputId = "n",   # ...can be assessed via input$n from server()
7                  label = "Anzahl an Zufallszahlen",
8                  value = 50, min = 5, max = 1000),
9
10     plotOutput("hist")           # ...creates a window for the output graphic
11 )
12
13 # ...genrate output from input whenever input changes
14 server <- function(input, output){
15
16     histdata <- reactive({
17         rnorm(input$n)
18         })
19
20     output$hist <- renderPlot({
21         hist(histdata(),
22              main = paste(input$n, "standard normally distributed random numbers"),
23              col = "lightblue")
24     })
25
26 }
27
28 shinyApp(ui = ui, server = server)  # ...runs app
```

Note again the syntax of `reactive({})`!

Replace server code

`histdata()` **is called as a function.**

Via `({})` a **reactive context** is defined, i.e. it will be executed whenever the input (as given from the UI) changes.
… `histdata()` is a reactive variable, i.e. It has to be called just as a function whenever the input changes (therefore the bracket s: `()`).

Exercise

Change the app and modify…

1. …the color of the histogram
2. …and further change the distribution of the random numbers into uniform `runif()` !