



Application shiny avec {golem}

User 2021

2021-07-07

Vincent Guyader - ThinkR

Programme

- Partie 00 & 01: Introduction
- Partie 02: développer une application avec golem
- Partie 03: Tester et déployer son application

PART 00 - INTRODUCTION

Vincent Guyader & Cervan Girard

Data Scientist, R expert.



- <https://rtask.thinkr.fr>
- <https://github.com/ThinkR-open>
- https://twitter.com/thinkr_fr
- Formateur
- Certificateur :
 - R niveau 1 - Utilisateur – Analyse de données
 - R niveau 2 - Développeur – Création de packages
 - R niveau 3 - Développeur – Conception d'interfaces Shiny

<https://thinkr.fr/formation-au-logiciel-r/>

Logistique

Utilisation de Zoom

The screenshot shows the Zoom control bar with several callout boxes providing instructions:

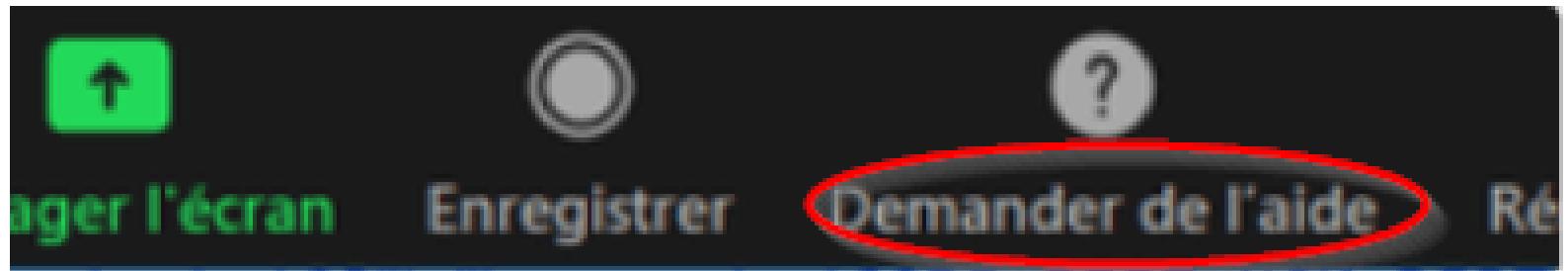
- Pour contrôler la caméra** (To control the camera) points to the video camera icon.
- Pour afficher la messagerie instantanée et poser des questions** (To display instant messaging and ask questions) points to the message bubble icon.
- Pour contrôler le micro et le haut-parleur** (To control the microphone and speaker) points to the microphone and speaker icons.
- Lever la main, Lever le pouce en l'air** (Raise your hand, give a thumbs up) points to the hands icon.

The control bar includes the following icons from left to right: Muet, Démarrer vidéo, Sécurité, Participants (with a count of 1), Sondages, Converser, Partager l'écran, Enregistrer, Diviser en groupe, Finir la réunion, and a registration button.

A floating window titled "Conversation de groupe Zoom" is also visible, showing a message: "Envoyer à : Tout le monde" (Send to: Everyone), "Fichier" (File), and the message "Bonjour, quelles sont vos attentes ?" (Hello, what are your expectations?). Below the control bar, there are icons for "Écran" (Screen), "Tableau blanc" (Whiteboard), and "iPhone / iPad". At the bottom, there are links for "Demander la réunion - Zoom - M...", "Distanciel", and "Bonnes pratiques - PowerPoint".

Utilisation de Zoom

Pour appeler à l'aide le formateur depuis une session isolée



Pratique

- Merci de couper votre micro pendant les présentations
- Merci de partager votre écran pendant les TD pour que nous puissions vous aider
- Si la webcam utilise trop votre bande passante, vous pouvez la couper
 - Merci de la remettre en marche pour les interactions avec les formateurs
 - Pour ne pas avoir l'impression de parler à un mur, les formateurs pourront demander à 2-3 personnes de conserver leur webcam allumée.

- En cas de blocage en plein écran
 - Double clic sur la plein écran devrait revenir en petite fenêtre
 - Sinon, appuyer sur "Échap"
- En cas de "perte" de la barre pour couper son micro/webcam
 - En mode plein écran, elle apparaît en haut
 - Passer la souris sur le petit bandeau vert avec le nom de votre ordinateur ou votre nom, le menu devrait apparaître.
- Astuce
 - Maintenir la touche espace pour ouvrir temporairement le micro

Rendez vous sur <http://slido.com> avec le code #golem

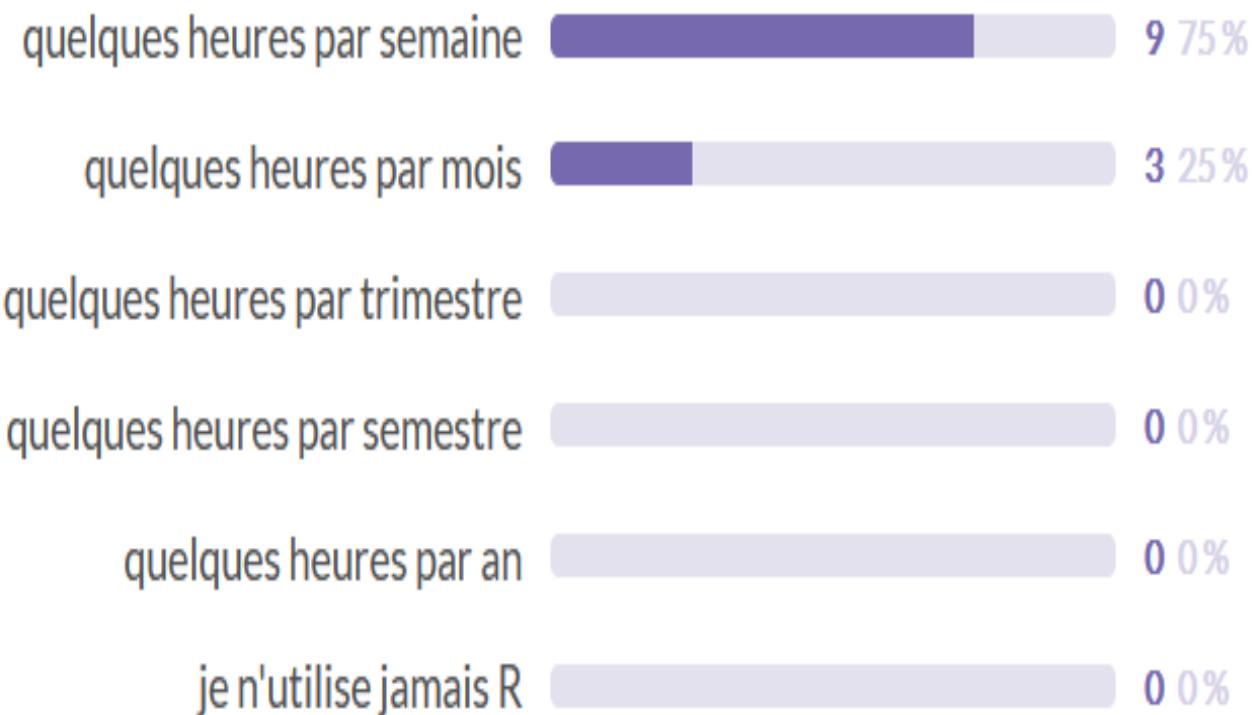
- Pour centraliser vos questions
- Pour centraliser vos réponses

A propos de vous

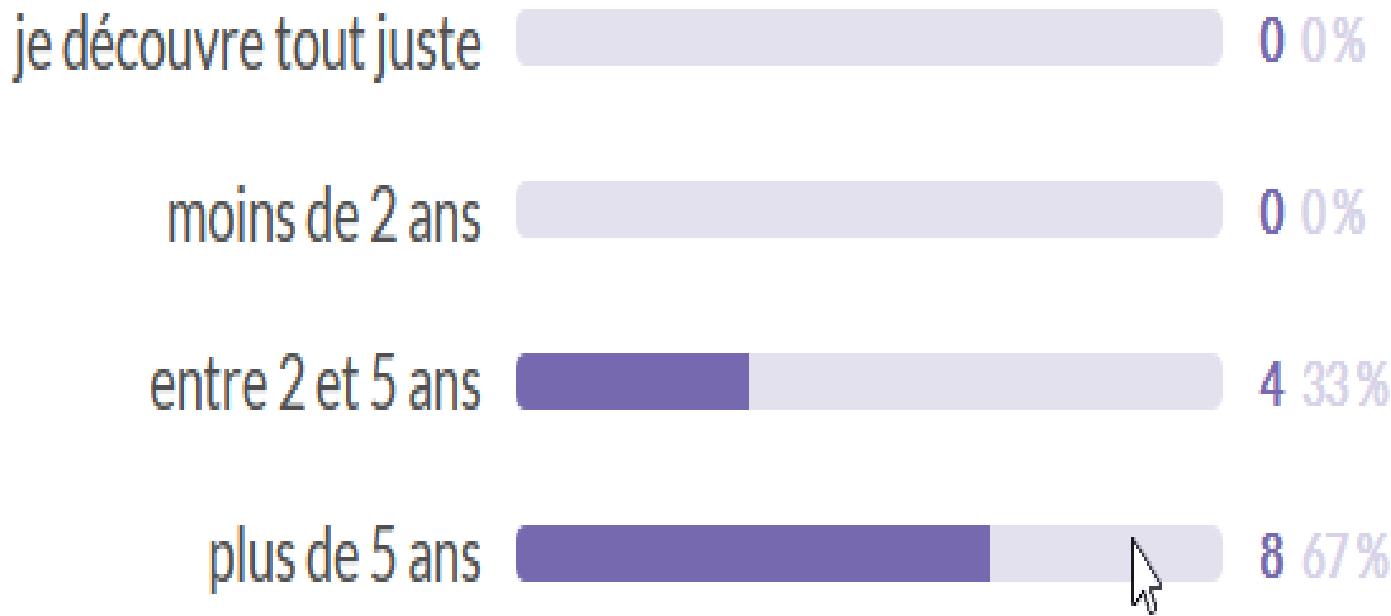
-> Regardons vos réponses au questionnaire de préparation

A propos de vous

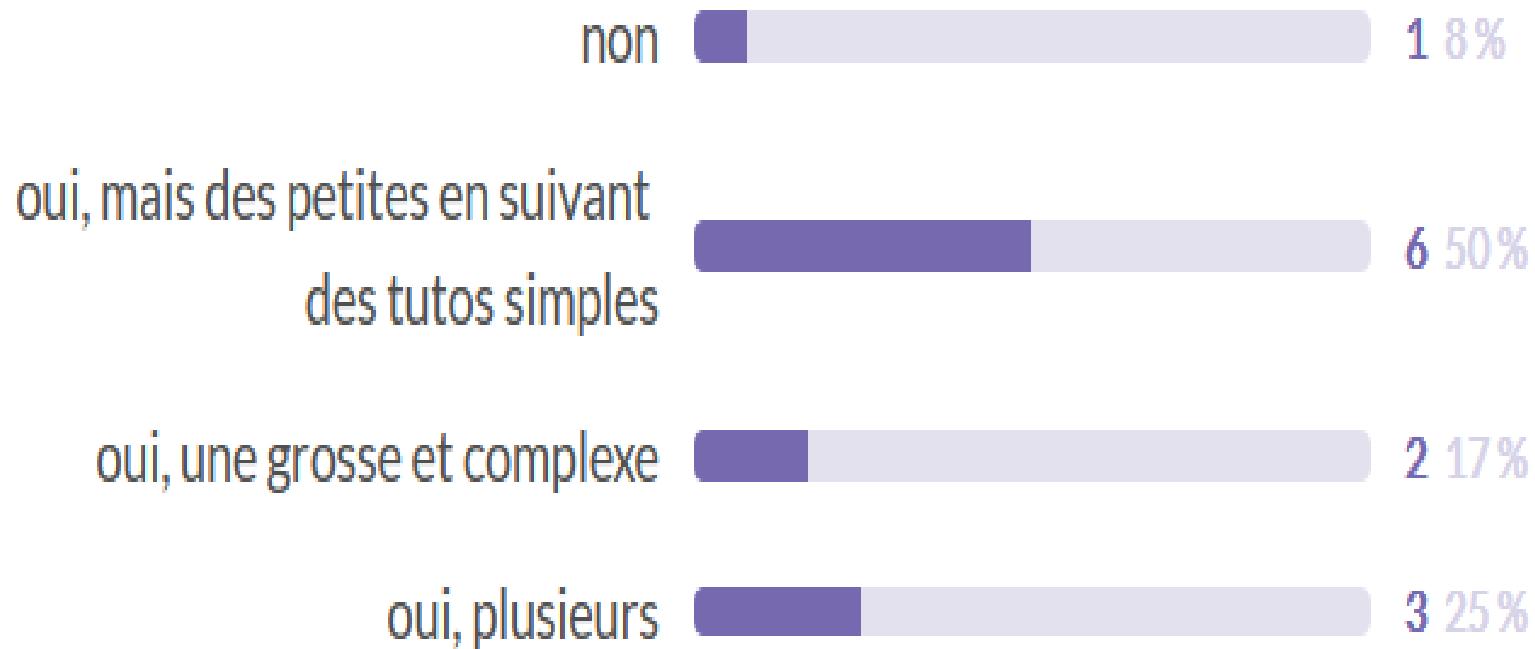
Combien de temps consacrer-vous à la programmation à R au quotidien?



Depuis combien de temps utilisez-vous R ?



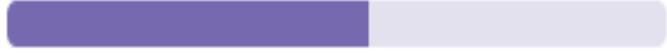
Avez-vous déjà développé une application Shiny ?



A propos de vous

Si on vous parle de "Shiny Module" ?:

aucune idée de ce que c'est  3 27%

je sais ce que c'est, mais je
n'utilise pas vraiment  6 55%

j'ai essayé, mais je n'ai pas réussi  1 9%

j'en utilise régulièrement  1 9%

je ne fais que ça  0 0%

A propos de vous

Si on vous parle du package {Golem} :

je ne connais pas du tout, je viens
pour découvrir



j'ai déjà essayé de m'y mettre
mais je débute ou n'ai pas encore
réussi.

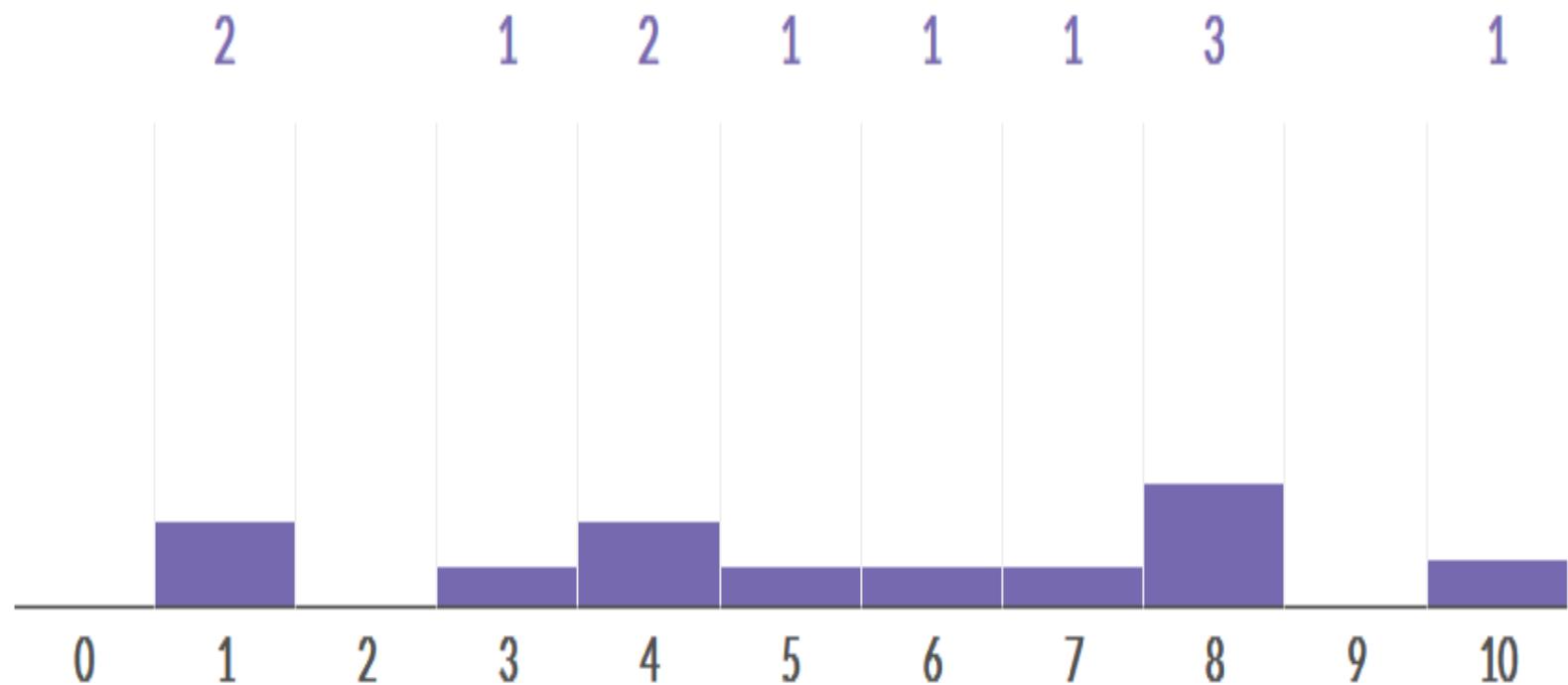


c'est d'ores et déjà mon standard
de développement



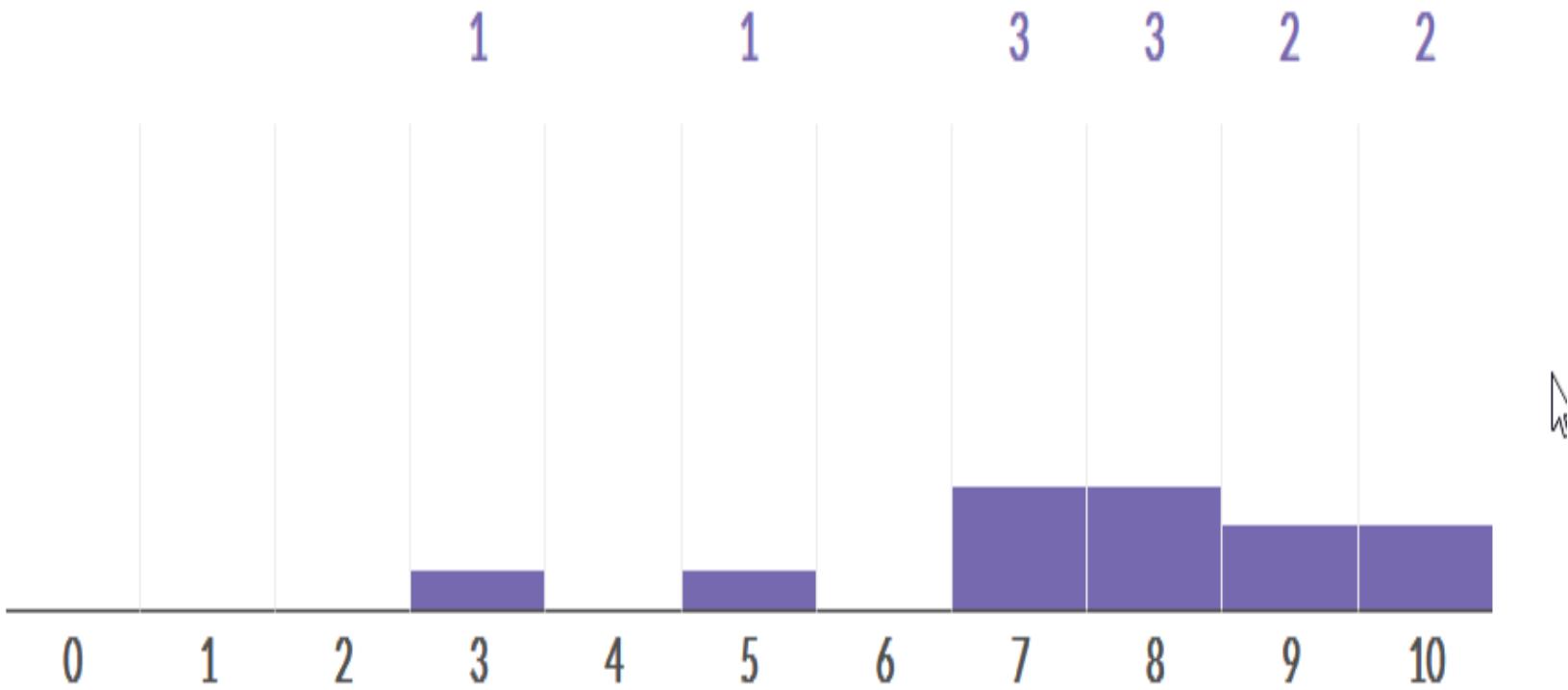
A propos de vous

Evaluez vos compétences. Savoir développer un package R de A à Z



A propos de vous

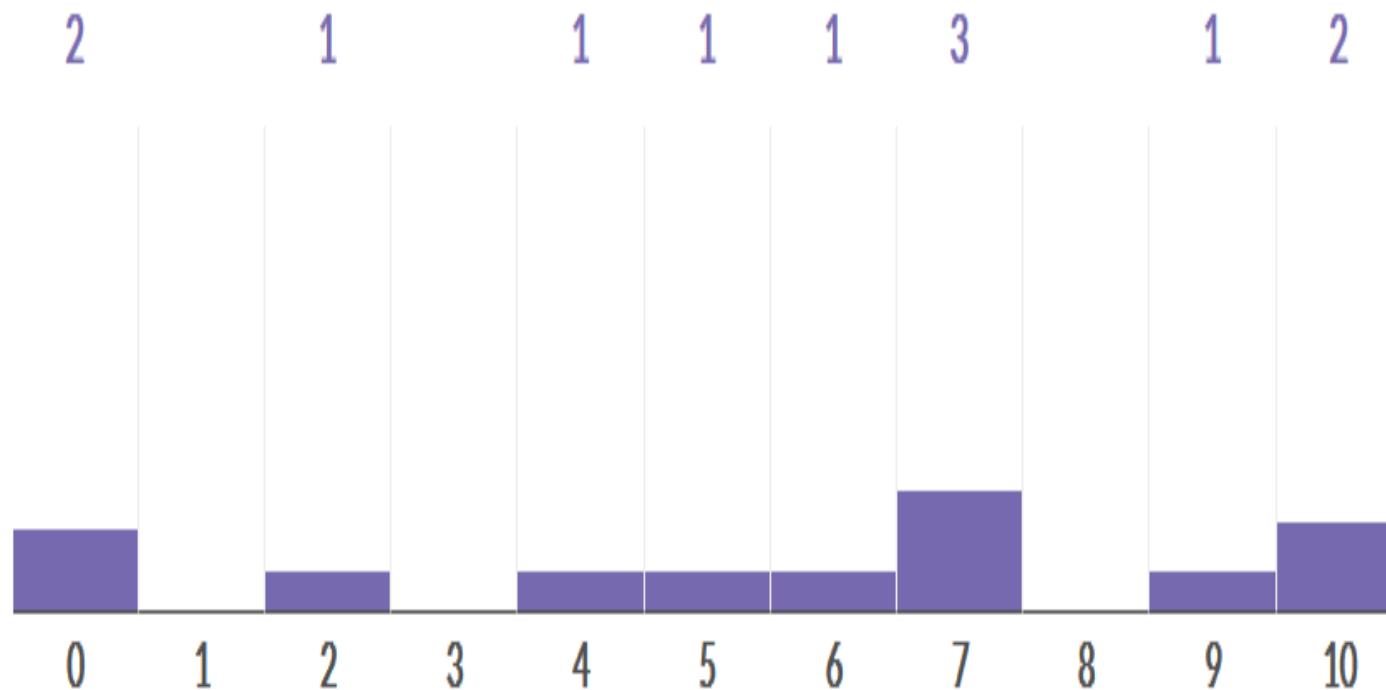
Evaluez vos compétences. Savoir concevoir un graphique avec ggplot2



A propos de vous

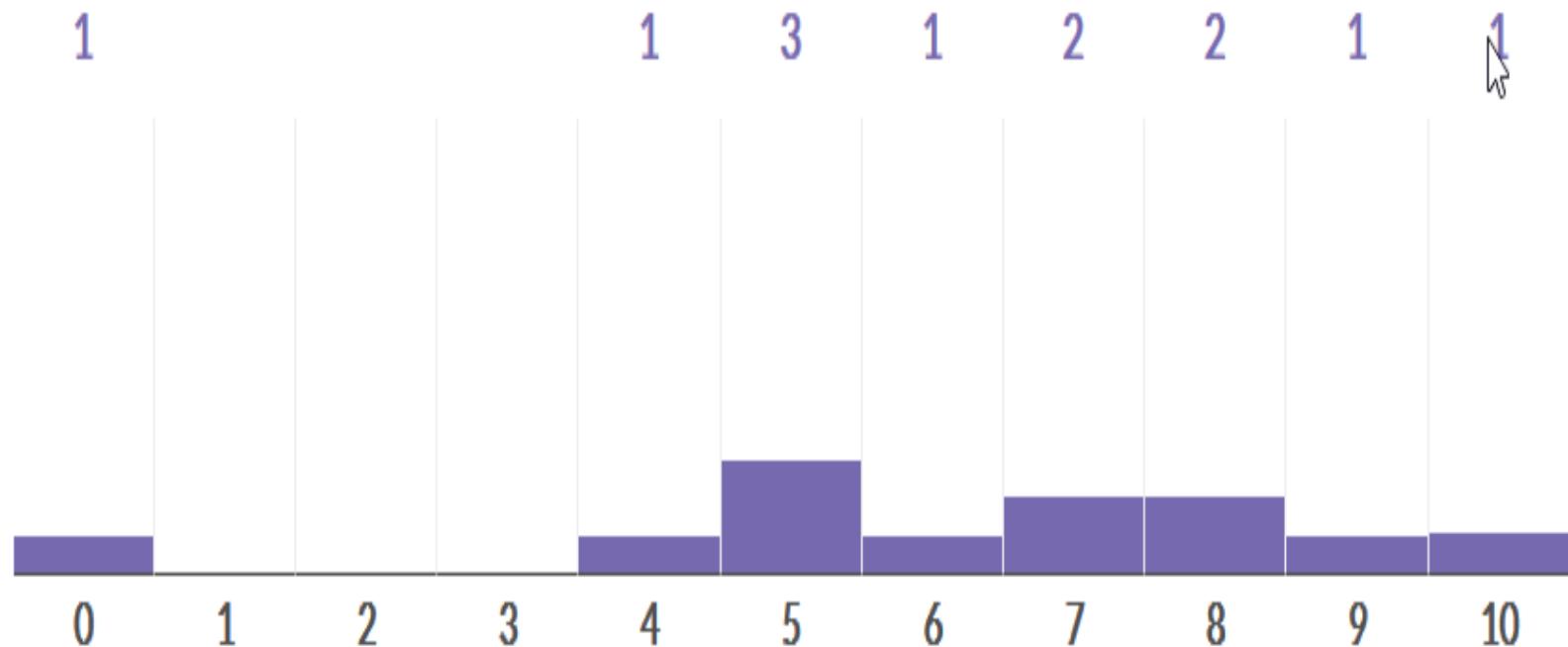
W

Evaluez vos compétences. Comprendre le concept de reactivité et le lien entre server et UI



A propos de vous

Evaluez vos compétences. Savoir positionner et mettre en place une interaction avec un élément graphique (bouton, menu ...)

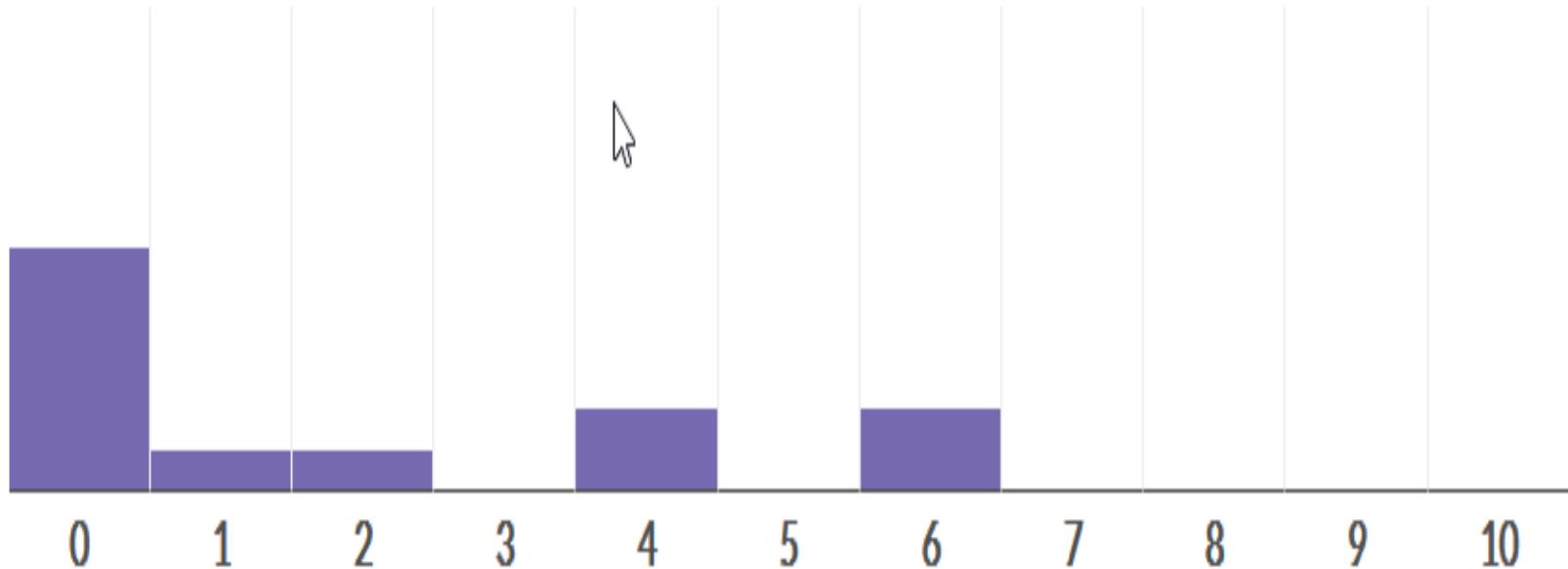


A propos de vous

Evaluez vos compétences. Savoir construire des modules Shiny

6 1 1

2 2

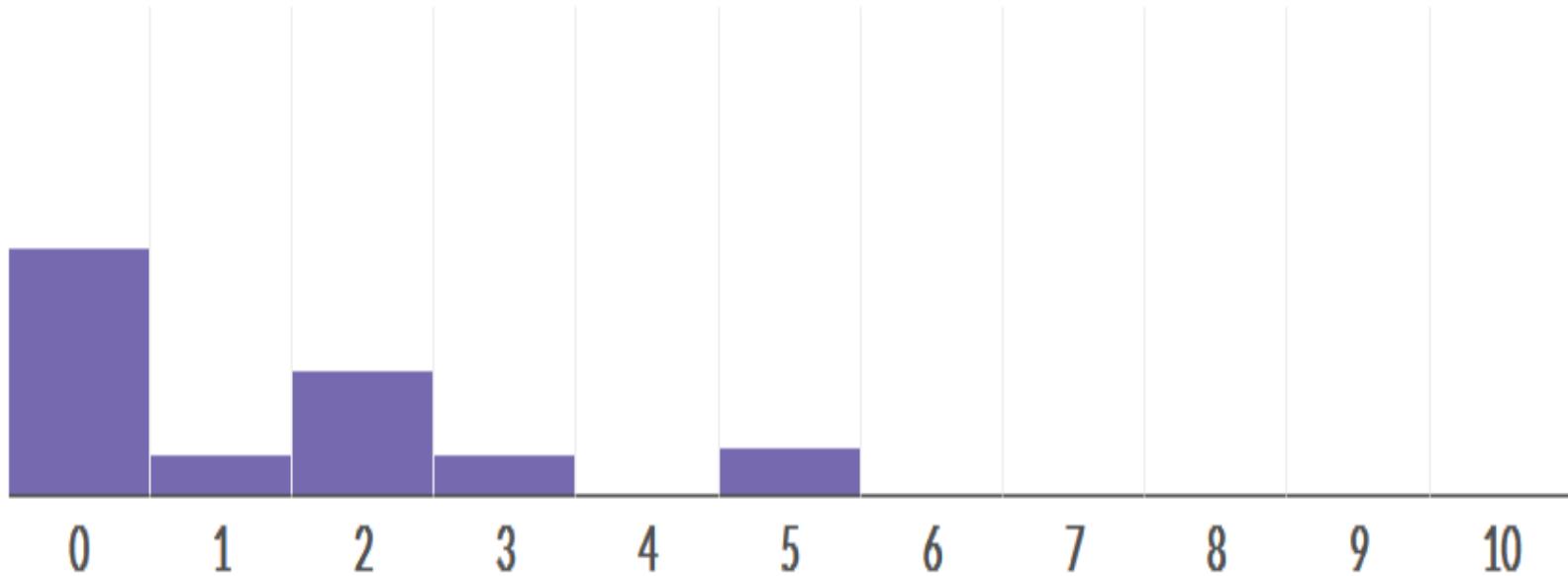


A propos de vous

Evaluez vos compétences. Orchestrer la communication entre modules

6 1 3 1

1



A propos de vous

Evaluez vos compétences. Être capable d'encapsuler une application shiny dans un package R, documenté, maintenable et durable



PART 01 - A propos de {golem}



Créer, maintenir et déployer facilement une application Shiny

```
golem <- cranlogs::cran_downloads(  
  "golem",  
  from = "2019-08-01"  
)  
sum(golem$count)
```

```
[1] 81724
```

Pourquoi {golem}?

Pourquoi utiliser {golem}?

- Automatiser les tâches ennuyeuses répétitives
- Travailler avec des outils fiables
- Gagner du temps
- Simplifier le déploiement
- standardiser le travail en équipe

A propos de {golem} chez ThinkR:

- Ancien produit interne, utilisé quotidiennement
- On avait besoin d'un outil fiable pour déployer chez nos clients
- Encourage l'usage de R en production

Une application Shiny dans un package

Checklist d'une application "prod-ready"

- Dispose de Métadonnées (`DESCRIPTION`)
- Découpée en fonction (`R/`)
- Testée (`tests/`)
- Des dependances explicites (`NAMESPACE`)
- Documentée (`man/` & `vignettes`)

Une application Shiny dans un package

Tout ce que vous connaissez du développement de package fonctionne avec `{golem}`.

Tout particulièrement:

-  La documentation
-  Les tests

Comprendre {golem}

⚠ Warning ⚠

```
packageversion("golem")
```

```
[1] '0.3.1'
```

Si ce n'est pas le cas (mais ca doit l'être), l'instruction suivante installera la bonne version

```
remotes::install_github(  
  "thinkr-open/golem"  
)
```

Creer un {golem}

<https://connect.thinkr.fr/user2021> / Ex 1

New Project

Back Project Type

- R Package using RcppEigen >
- R Package using RcppParallel >
-  Book Project using bookdown >
- R Package using devtools >
-  Package for Shiny App using golem >
-  New Plumber API Project >
- Simple R Markdown Website >

Create a new
Package for Shiny
App using golem

Cancel

02 : 00

Comprendre {golem}

```
fs::dir_tree("golem")
```

```
golem
├── DESCRIPTION
├── NAMESPACE
├── R
│   ├── app_config.R
│   ├── app_server.R
│   ├── app_ui.R
│   └── run_app.R
└── dev
    ├── 01_start.R
    ├── 02_dev.R
    ├── 03_deploy.R
    └── run_dev.R
└── inst
    ├── app
    │   └── www
    │       └── favicon.ico
    └── golem-config.yml
└── man
    └── run_app.Rd
```

Comprendre {golem}

les fichiers classiques d'un package (i.e. pas spécifique à `{golem}`):

- `DESCRIPTION`: metadonnées
- `NAMESPACE`: les fonction exportée et celles importée depuis un autre 
- `R/`: fonction (dans `{golem}` tout est fonction)
- `man/`: la documentation

R/app_server.R

```
#' The application server-side
#'
#' @param input,output,session Internal parameters for {shiny}.
#'     DO NOT REMOVE.
#' @import shiny
#' @noRd
app_server <- function( input, output, session ) {
  # Your application server logic

}
```

- La partie server de votre application
- peut être vu comme le remplacement du fichier `server.R`
- Fait appel aux éventuels `callModule()` / `module_server()` de votre application

```
#' @param request Internal parameter for `shiny`.  
#' DO NOT REMOVE.  
#' @import shiny  
#' @noRd  
app_ui <- function(request) {  
  tagList(  
    # Leave this function for adding external resources  
    golem_add_external_resources(),  
    # Your application UI logic  
    fluidPage(  
      h1("golex")  
    )  
  )  
}
```

- La partie UI de votre application
- Votre UI se place apres la ligne `# Your application UI logic`

R/app_ui.R

```
golem_add_external_resources <- function(){

  add_resource_path(
    'www', app_sys('app/www')
  )

  tags$head(
    favicon(),
    bundle_resources(
      path = app_sys('app/www'),
      app_title = 'golex'
    )
    # Add here other external resources
    # for example, you can add shinyalert::useShinyalert()
  )
}
```

- Se chargera pour vous d'intégrer vos éventuels fichiers CSS et JS nécessaires à vos applications

R/run_app.R

```
run_app <- function(  
  onStart = NULL,  
  options = list(),  
  enableBookmarking = NULL,  
  uiPattern = "/",  
  ...  
) {  
  with_golem_options(  
    app = shinyApp(  
      ui = app_ui,  
      server = app_server,  
      onStart = onStart,  
      options = options,  
      enableBookmarking = enableBookmarking,  
      uiPattern = uiPattern  
    ),  
    golem_opts = list(...)  
  )  
}
```

- `run_app()` Lance l'application. **⚠️** Vous n'avez pas à modifier ce fichier **⚠️**

```
golem::detach_all_attached()  
# rm(list=ls(all.names = TRUE))  
  
# Document and reload your package  
golem::document_and_reload()  
  
# Run the application  
run_app()
```

- Ces lignes de codes permettent de visualiser l'application en train d'être développée

sans subir la lourdeur de l'installation complète d'un package

<https://connect.thinkr.fr/user2021> / Ex 2 & Ex 3

05:00

A propos du dossier dev/

A propos du dossier dev/

```
fs::dir_tree("golex/dev")
```

```
golex/dev
├── 01_start.R
├── 02_dev.R
├── 03_deploy.R
└── run_dev.R
```

4 fichiers essentiels :

- `01_start.R`: à compléter et lancer une seule fois au début du projet
- `02_dev.R`: à maintenir et utiliser jours après jours
- `03_deploy.R`: pour la mise en prod le jour J
- `run_dev.R`: pour voir votre application

- Remplir le fichier `DESCRIPTION`

```
golem::fill_desc(  
  pkg_name = "golex", # The Name of the package containing the App  
  pkg_title = "PKG_TITLE", # The Title of the package containing the App  
  pkg_description = "PKG_DESC.", # The Description of the package containing the App  
  author_first_name = "AUTHOR_FIRST", # Your First Name  
  author_last_name = "AUTHOR_LAST", # Your Last Name  
  author_email = "AUTHOR@MAIL.COM", # Your Email  
  repo_url = NULL # The (optional) URL of the GitHub Repo  
)
```

| <https://connect.thinkr.fr/user2021> / Ex 4

04 : 00

Dans ce fichier beaucoup de lignes de code vous sont proposées. Il est fortement recommandé d'en lancer certaines, tandis que d'autre ne doivent être lancées qu'en connaissance de cause.

- A lancer de maniere (*quasiment*) obligatoire:

- `usethis::use_mit_license(name = "Golem User")`

vous pouvez choisir une autre licence

- `golem::use_recommended_tests()`

Met en place des tests unitaires "de base"

A lancer de manière facultative:

Les appels classiques à `{usethis}`

```
usethis::use_readme_rmd( open = FALSE )
usethis::use_code_of_conduct()
usethis::use_lifecycle_badge( "Experimental" )
usethis::use_news_md( open = FALSE )
# usethis::use_git()
```

D'autres instructions utiles :

- `golem::use_recommended_deps()`
- `golem::use_favicon()`
- `golem::use_utils_ui()` & `golem::use_utils_server()`

| <https://connect.thinkr.fr/user2021> / Ex 5.1 et Ex 5.2

10:00

PART 02 - DEV

Structurer son application

Logique Metier vs logique applicative

- logique applicative: Ce qui fait que votre application est une application reactive
- logique metier: les algorithmes et fonctions qui rendent votre application utile à votre secteur d'activité

Tâchons de garder ces concepts bien séparés!

Structurer son application

Convention de nommage:

- `app_*`: les fonctions qui définissent l'infrastructure globale de votre application
- `fct_*`: les fonctions relatives à la logique métier
- `mod_*`: fichier avec 1 module (ui + server)
- `utils_*`: fonctions outils, transverses à vos modules/fonctions
- `*_ui_*` / `*_server_*`: distingue les parties UI et SERVER

Pourquoi s'embêter?

-  : `R/connect.R`
-  : `R/summary.R`
-  : `R/plot.R`
-  : `R/odbc.R`

Pourquoi s'embêter?

- : `R/connect.R`
- : `R/summary.R`
- : `R/plot.R`
- : `R/odbc.R`
- : `R/fct_connect.R`
- : `R/mod_summary.R`
- : `R/utils_plot.R`
- : `R/fct_odbc.R`

Pourquoi s'embêter?

La séparation de la logique métier et de la logique applicative est cruciale :

- Il est plus facile de travailler sur les fonctions lorsqu'elles ne sont pas dans l'application (vous n'avez pas besoin de relancer l'application à chaque fois).
- Vous pouvez tester votre logique métier avec les outils de test classiques.
- Vous pouvez utiliser les fonctions de la logique métier en dehors de l'application.

```
golem::add_fct("helpers")
golem::add_utils("data_ui")
```

✓ File created at R/fct_helpers.R

- Go to R/fct_helpers.R

✓ File created at R/utils_data_ui.R

- Go to R/utils_data_ui.R

initialise les fichiers `fct_*` et `utils_*`

| <https://connect.thinkr.fr/user2021> / Ex 6

10:00

Maintenant on peut construire une premiere application simple

| <https://connect.thinkr.fr/user2021> / Ex 7

15:00

Les Shiny modules

A propos des Shiny Module

- "Morceaux" d'application
- Composé de 2 parties : une UI et une Server
- S'assure de l'unicité des ID nécessaire au bon fonctionnement de votre application
- Permet de factoriser votre application

Un million de bouton validate

```
library(shiny)
ui <- function(request){
  fluidPage(
    sliderInput(inputId = "choice1",label = "choice 1",min = 1,max = 10,value = 5),
    actionButton(inputId = "validate1",label = "validate choice 1"),
    sliderInput(inputId = "choice2",label = "choice 2",min = 1,max = 10,value = 5),
    actionButton(inputId = "validate2",label = "validate choice 2"),
    sliderInput(inputId = "choice3",label = "choice 3",min = 1,max = 10,value = 5),
    actionButton(inputId = "validate3",label = "validate choice 3")
  )
}

server <- function(input, output, session){
  observeEvent( input$validate1 , { print(input$choice1) })
  observeEvent( input$validate2 , { print(input$choice2) })
  observeEvent( input$validate3 , { print(input$choice3) })
}
shinyApp(ui, server)
```

Les Shiny Module

Notre premier module:

```
# dans le ui le seul truc important c'est le ns() systématique pour tous les ID.
mod_validate_ui <- function(id){
  ns <- NS(id)
  tagList(
    sliderInput(ns("choice"), "Choice", 1, 10, 5),
    actionButton(ns("validate"), "Validate Choice")
  )
}

mod_validate_server <- function(id){
  moduleServer( id, function(input, output, session){
    observeEvent(input$validate , {
      print(input$choice)
    })
  })
}
```

Les Shiny Module

On peut utiliser le module plusieurs fois:

```
library(shiny)
ui <- function(request){
  fluidPage(
    mod_validate_ui("id_1"),
    mod_validate_ui("id_2"),
    mod_validate_ui("id_3")
  )
}

# notez l'usage obligatoire de callModule ici.
server <- function(input, output, session){
  mod_validate_server("id_1")
  mod_validate_server("id_2")
  mod_validate_server("id_3")
}

shinyApp(ui, server)
```

Les Shiny Module

```
id <- "id_1"  
ns <- NS(id)  
ns("choice")
```

```
[1] "id_1-choice"
```

```
mod_validate_ui <- function(id, butname){  
  ns <- NS(id)  
  tagList( actionButton(ns("validate"), butname) )  
}
```

```
mod_validate_ui("id_1", "Validate Choice")  
mod_validate_ui("id_2", "Validate choice, again")
```

```
<button id="id_1-validate" type="button" class="btn btn-default action-  
button">validate choice</button>
```

```
<button id="id_2-validate" type="button" class="btn btn-default action-  
button">validate choice, again</button>
```

```
golem:::add_module( name = "my_first_module")
```

Initialise un squelette de module au bon endroit dans votre golem

```
#' my_first_module UI Function
#'
#' @description A shiny Module.
#'
#' @param id,input,output,session Internal parameters for {shiny}.
#'
#' @noRd
#'
#' @importFrom shiny NS tagList
mod_my_first_module_ui <- function(id){
  ns <- NS(id)
  tagList(
    )
}
```

```
#' my_first_module Server Functions
#'
#' @noRd
mod_my_first_module_server <- function(id){
  moduleServer( id, function(input, output, session){
    ns <- session$ns

  })
}

## To be copied in the UI
# mod_my_first_module_ui("my_first_module_ui_1")

## To be copied in the server
# mod_my_first_module_server("my_first_module_ui_1")
```

⚠ DON'T FORGET THE NS() IN THE UI ⚠

Colin Fay 🌐
 @_ColinFay

Iteration 7729 of:

- forgetting to put a ns() in a Shiny Module
- launching the app
- staring at the app for 15 seconds before understanding

IF I HAD A PENNY FOR EVERY TIME
I FORGOT A NS() IN A SHINY MODULE

8:53 PM · Nov 19, 2019 · Twitter Web App

A vous

Construisez un module qui :

- Présente un sliderInput
- Affiche la valeur sélectionnée par l'utilisateur via un TextOutput

| <https://connect.thinkr.fr/user2021> / Ex 8

15 : 00

Comprendre les NAMESPACE et les dépendances

le NAMESPACE

- L'un des fichier les plus important de votre package
- \triangleleft on ne l'edite jamais à la main \triangleright
- indique comment votre package intéragit avec les autre packages
- indique les fonctions exportée depuis votre package

Qu'est ce qu'une dépendance

- Votre application utilise des packages externes, au moins `{shiny}` !
- pour chaque fonction vous devez expliciter les packages utilisé pour son execution (*il est interdit d'utiliser `library` ou `require`*)
- On utilise `@import` (le pacakge entier) et `@importFrom` (pour importer une fonction spécifique).

golem rajoute la dependance à `{shiny}` automatiquement

:

```
#' @importFrom shiny NS tagList
```

Qu'est ce qu'une dépendance

Exemple

```
#' @import magrittr
#' @importFrom stats na.omit

mean_no_na <- function(x){
  x <- x %>% na.omit()
  sum(x)/length(x)
}
```

- On peut utiliser `import` ou `importFrom`.
- Il vaut mieux d'utiliser `importFrom`, pour éviter les conflits de nom .
- A faire pour **TOUTES** les fonctions.

| ça prend du temps, mais c'est rentable!

Qu'est ce qu'une dépendance

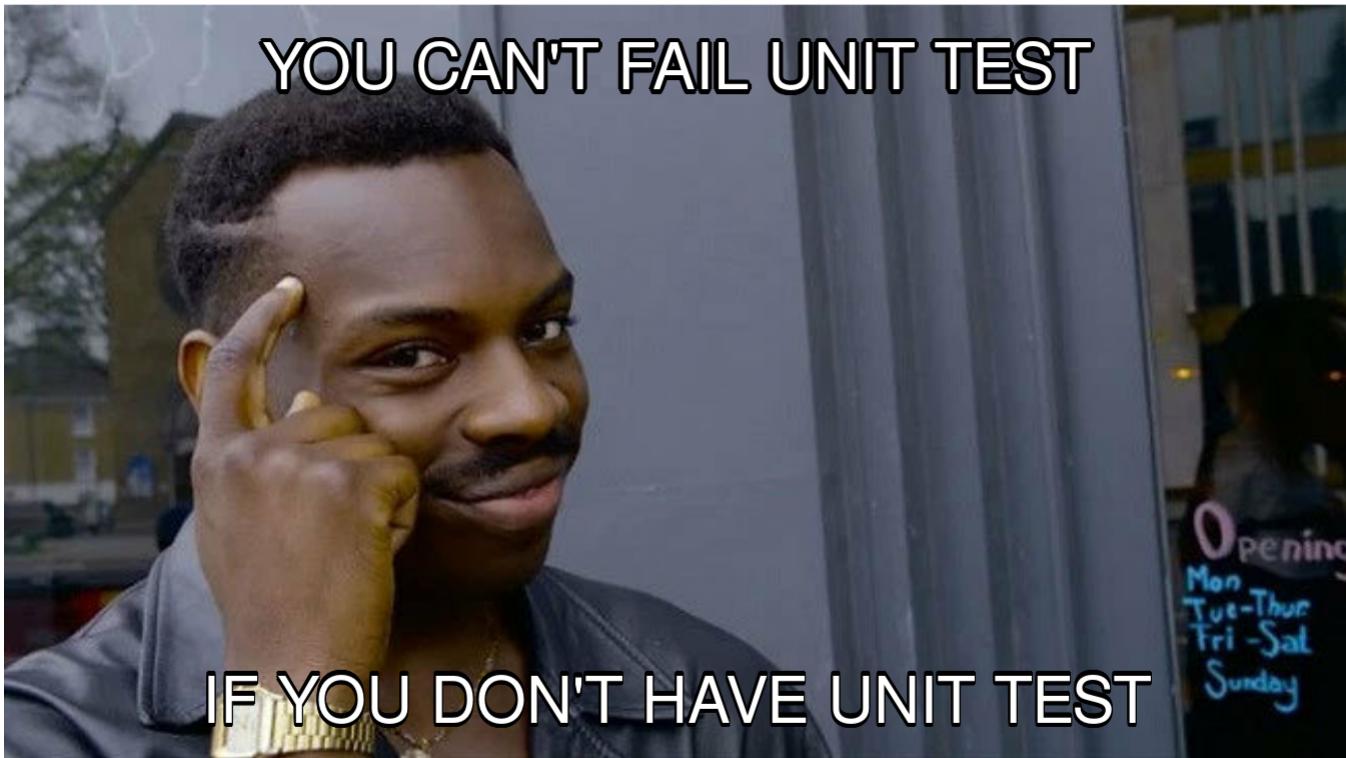
- Le fichier `DESCRIPTION` contient les dépendances
- Elles sont ajoutées dans `DESCRIPTION` avec:

```
attachment::att_amend_desc() # en lieu et place de multiple usethis::use_package()
```

PART 04 Tester et déployer

Tout code non testé, va casser.

Les tests unitaires



Pourquoi automatiser les tests?

- Gagner du temps!
- Travailler à plusieurs
- Pouvoir transférer le projet
- Assurer une stabilité à long terme de l'application

Que tester ?

- Se concentrer sur les tests de logique métier : il est plus important de vérifier que l'algorithme est toujours précis que de tester l'interface utilisateur.
- Comme nous avons séparé la logique métier de la logique applicative, nous utilisons des outils de test classique de développement de packages.
- On va utiliser `{testthat}` 

Mettre en place des tests

dev/01_start.R

```
## Init Testing Infrastructure ----  
## Create a template for tests  
golem::use_recommended_tests()
```

Les tests de base

dev/02_dev.R

```
## Tests ----  
## Add one line by test you want to create  
usethis::use_test( "app" )
```

Pour rajouter ses propres tests

Shiny dans un

-> tirer partie des outils de tests à disposition

```
test_that("The meaning of life is 42", {  
  expect_equal(  
    meaning_of_life(),  
    42  
)  
})
```



Test du front

{shinytests}

- 🚗 Test les variation d'apparence de votre application

puppeteer

- 🛠 Outil en ligne de commande pour simuler une session web, en NodeJS

{crrry}

- 🚕 Package R pour controler une app {shiny}

Test de charge

{shinyloadtest}

- 🚧: package R + Cli pour generer et relancer des test de charge

{dockerstats}

- 📈: récupere stat docker depuis R

Deployer son application

Deployer son application

- Une fois que tout est bien testé, on peut déployer avec `{golem}`
- On déploie sur un server, ou en tant que `tar.gz` avec
`pkgbuild::build()`

```
## RStudio ----  
## If you want to deploy on RStudio  
related platforms  
golem::add_rstudioconnect_file()  
golem::add_shinyappsi_file()  
golem::add_shinyserver_file()
```

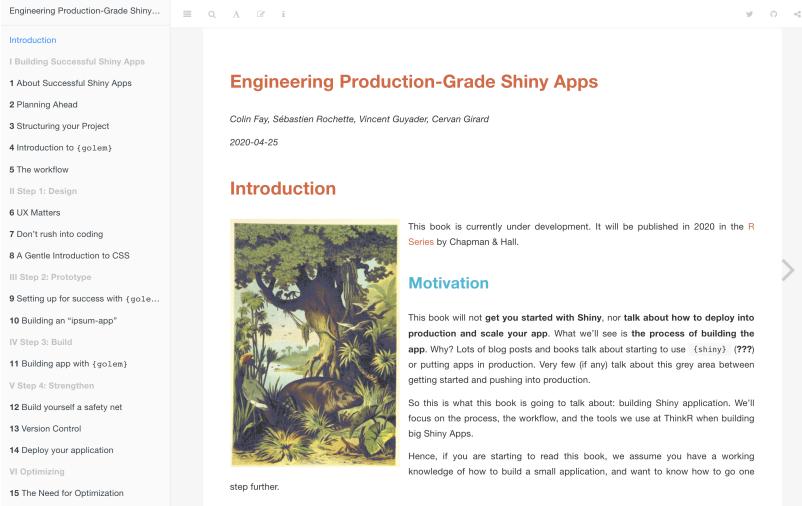
```
## Docker ----  
## If you want to deploy via a generic  
Dockerfile  
  
golem::add_dockerfile()  
## If you want to deploy to ShinyProxy  
golem::add_dockerfile_shinyproxy()  
## If you want to deploy to Heroku  
golem::add_dockerfile_heroku()
```

| <https://connect.thinkr.fr/user2021> / Ex 9

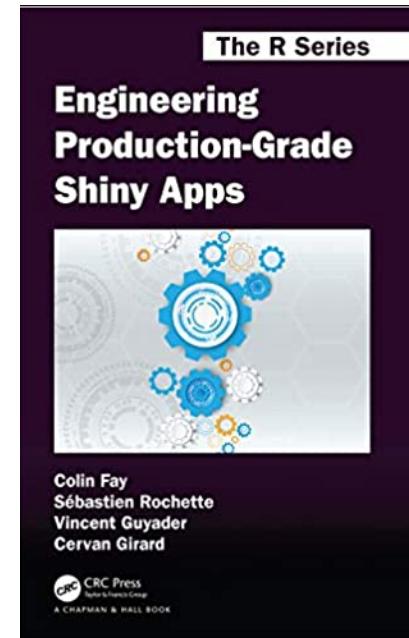
15:00

Online : <http://engineering-shiny.org>

format papier : <https://amzn.to/366XZdK>



The screenshot shows the homepage of the engineering-shiny.org website. The left sidebar contains a table of contents with 15 numbered sections from 'Introduction' to 'The Need for Optimization'. The main content area features a title 'Engineering Production-Grade Shiny Apps' by Colin Fay, Sébastien Rochette, Vincent Guyader, Cervan Girard, published on 2020-04-25. Below the title is a section titled 'Introduction' with a paragraph about the book's purpose and a small illustration of a tree. A 'Motivation' section follows, containing a paragraph and another illustration of a tree.



Merci, des Questions?

Vincent Guyader & Cervan Girard

Nous contacter ?

- http://twitter.com/thinkr_fr
- <https://thinkr.fr/>
- <https://rtask.thinkr.fr/>

A propos de {golem}

- <https://golemverse.org/>
- <http://engineering-shiny.org>
- <https://github.com/ThinkR-open/golem>