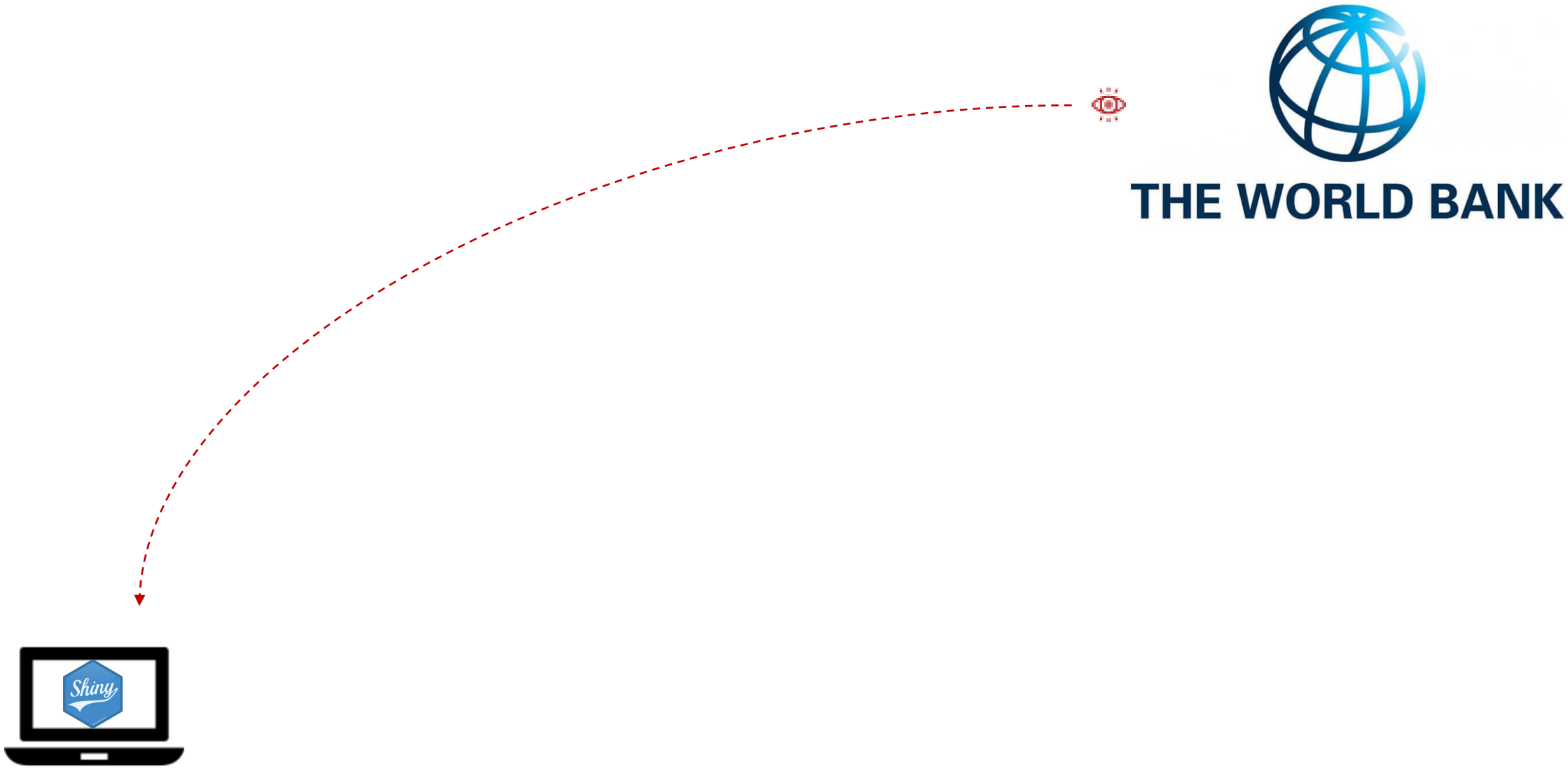
Shiny Days

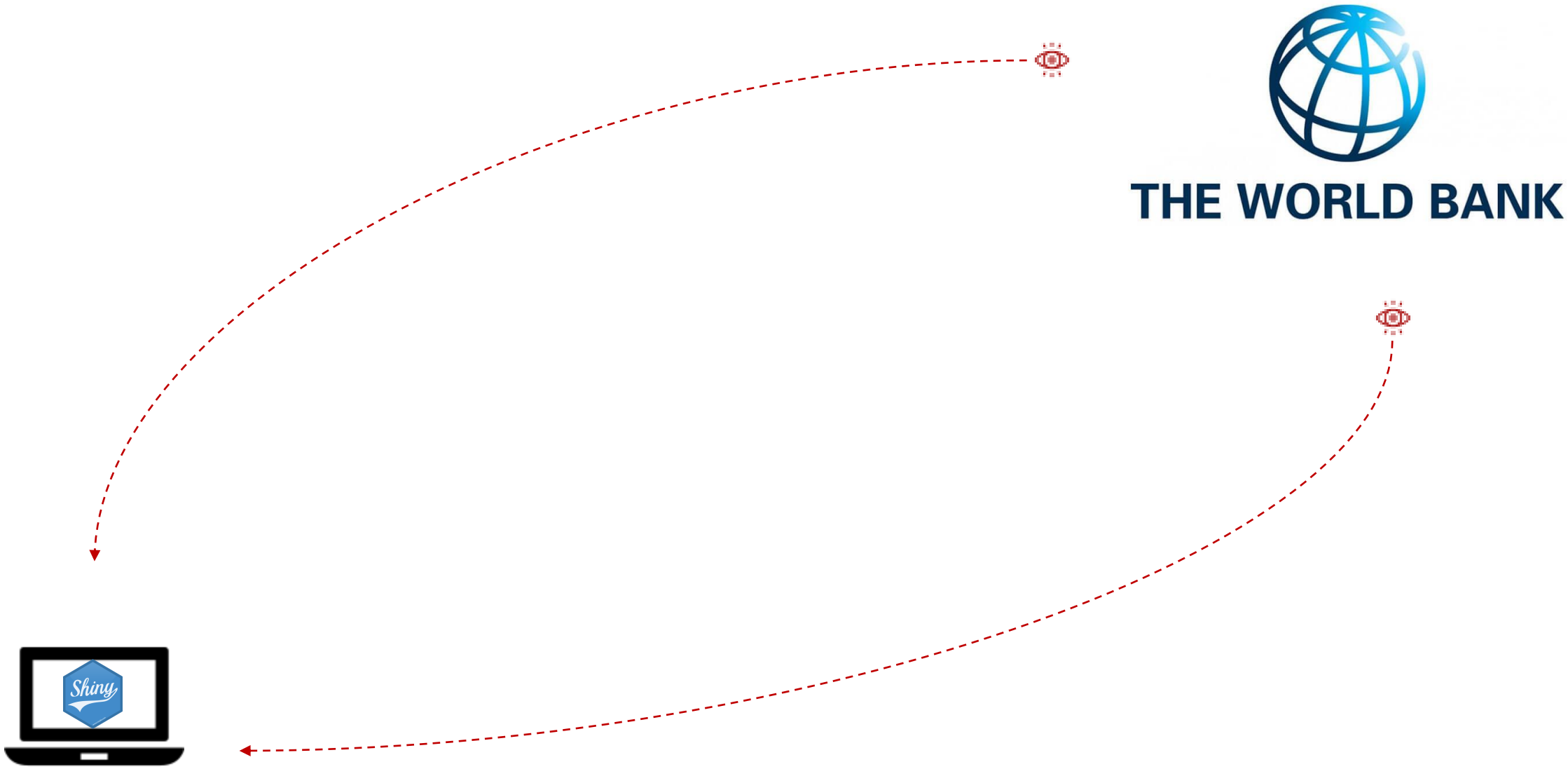
Performance & Maintenance

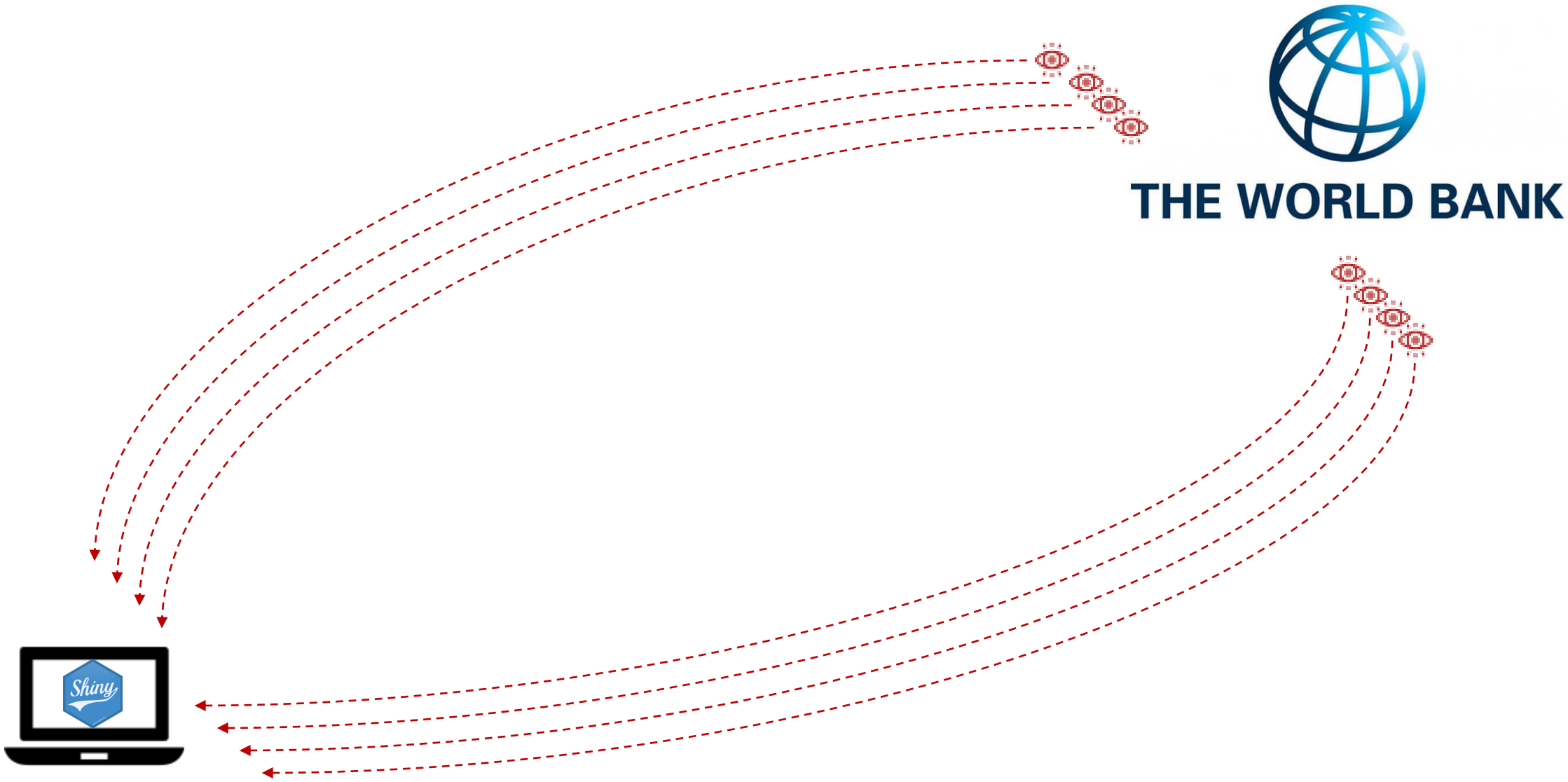
Once upon a time...


Shiny

THE WORLD BANK

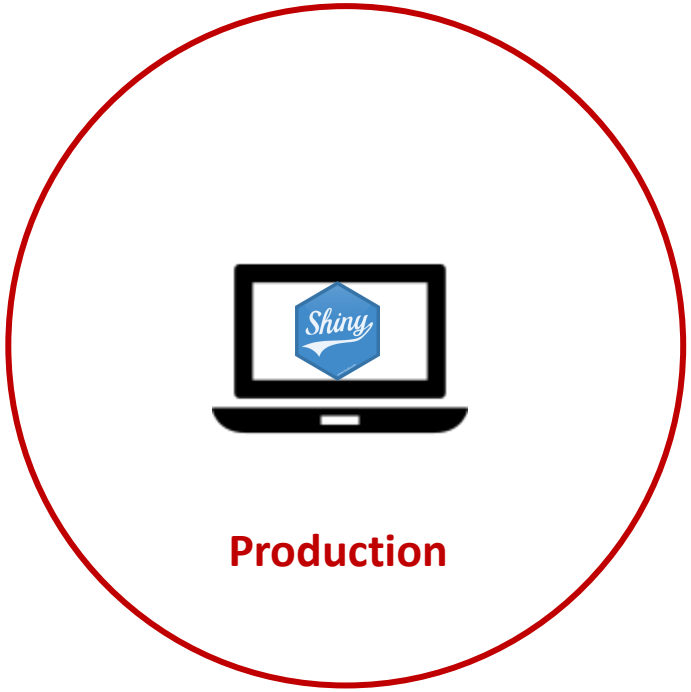

THE WORLD BANK
Shiny


THE WORLD BANK
Shiny

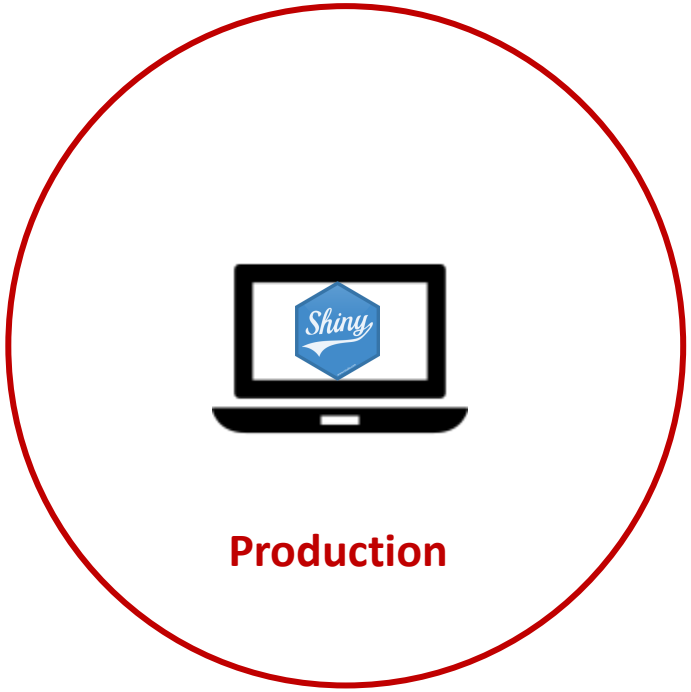

THE WORLD BANK
Shiny

Production

✓ Run on a dedicated server



- ✓ Run on a dedicated server
- ✓ Need to be always available




flickr.com/photos/68865613@N02/14938156538

Things will happen

Things **will** happen

✓ More connections

- ✓ More connections
- ✓ Requests for modifications

Today’s workshop

- **Performance:**
  - Is your app fast enough to handle multiple concurrent users?

Today’s workshop

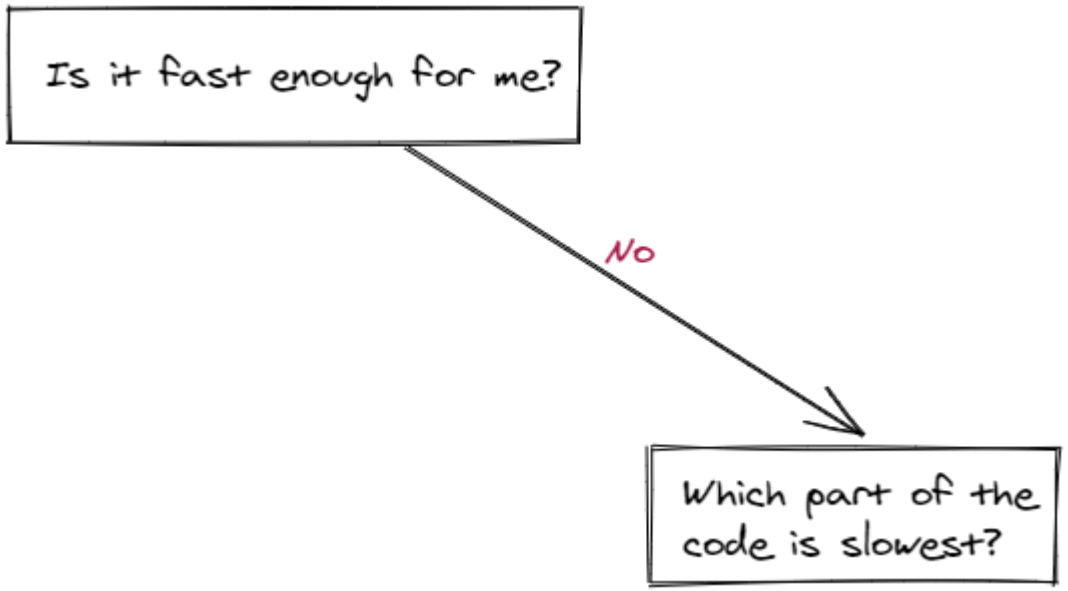
- **Performance:**
  - Is your app fast enough to handle multiple concurrent users?

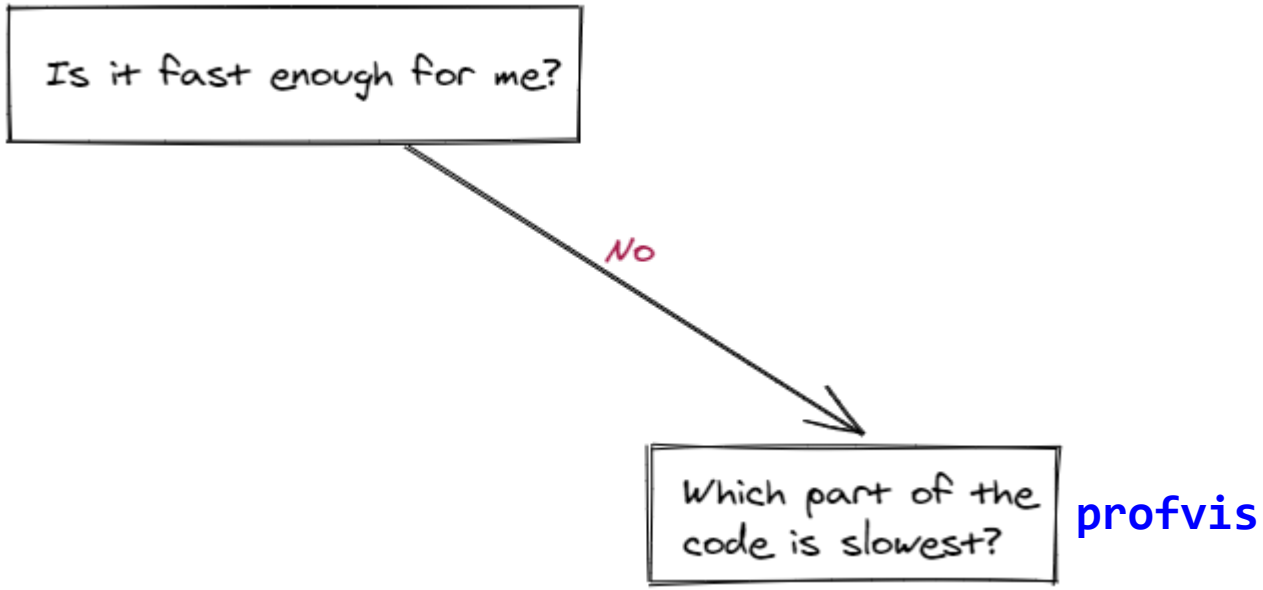
- **Maintenance:**
  - Is your app easy to modify?
  - Can you modify it without breaking it?

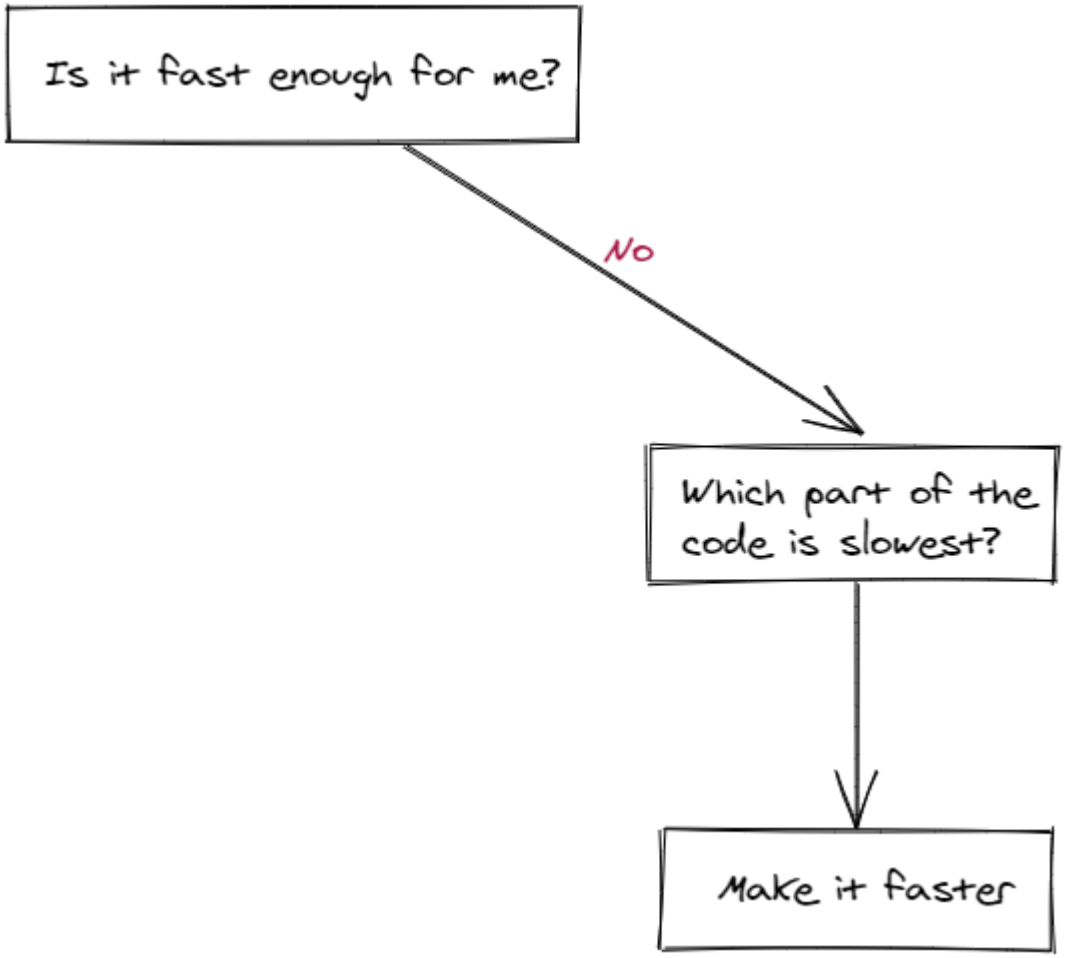
Performance

Demo

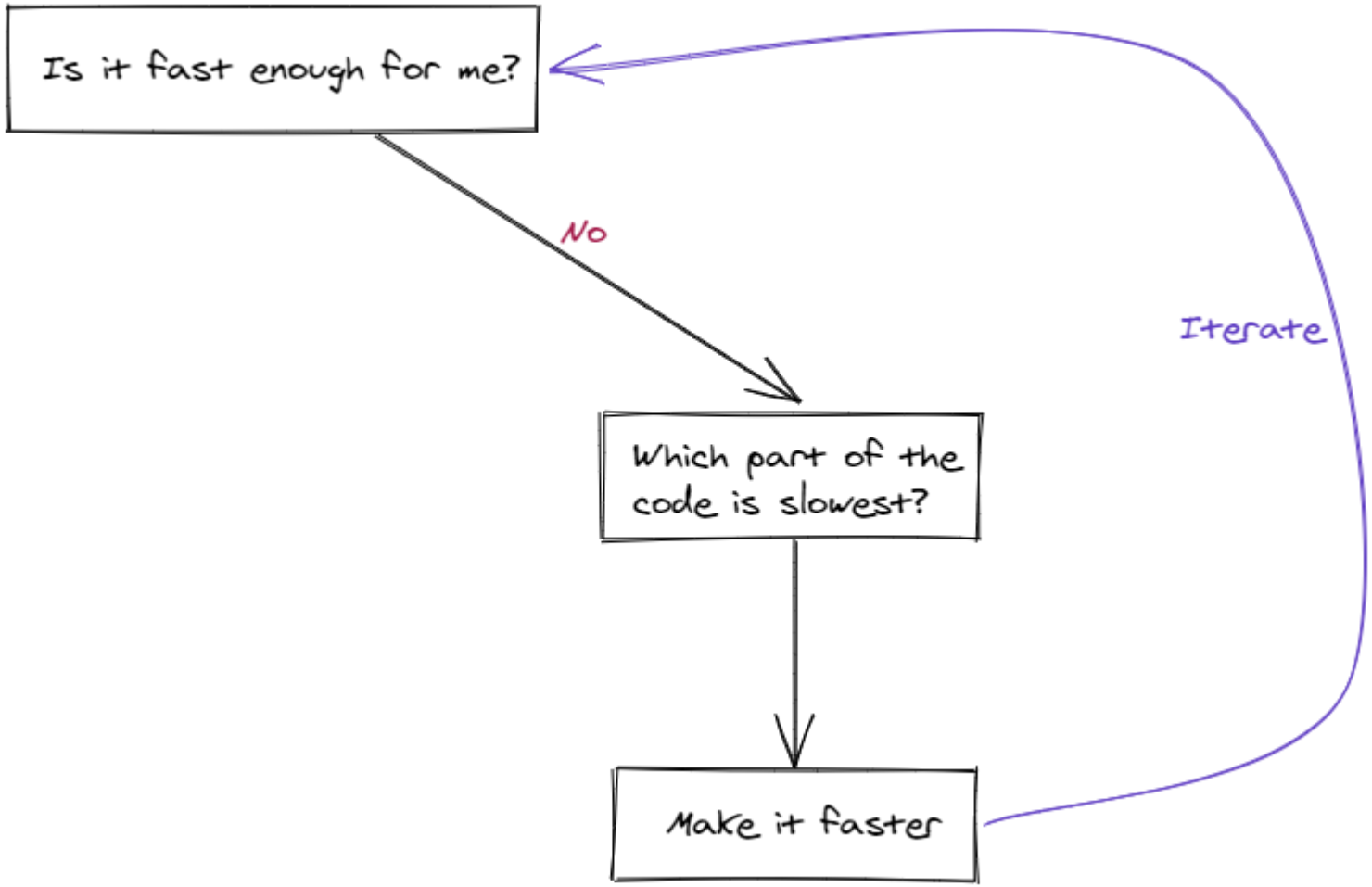
Is it fast enough for me?

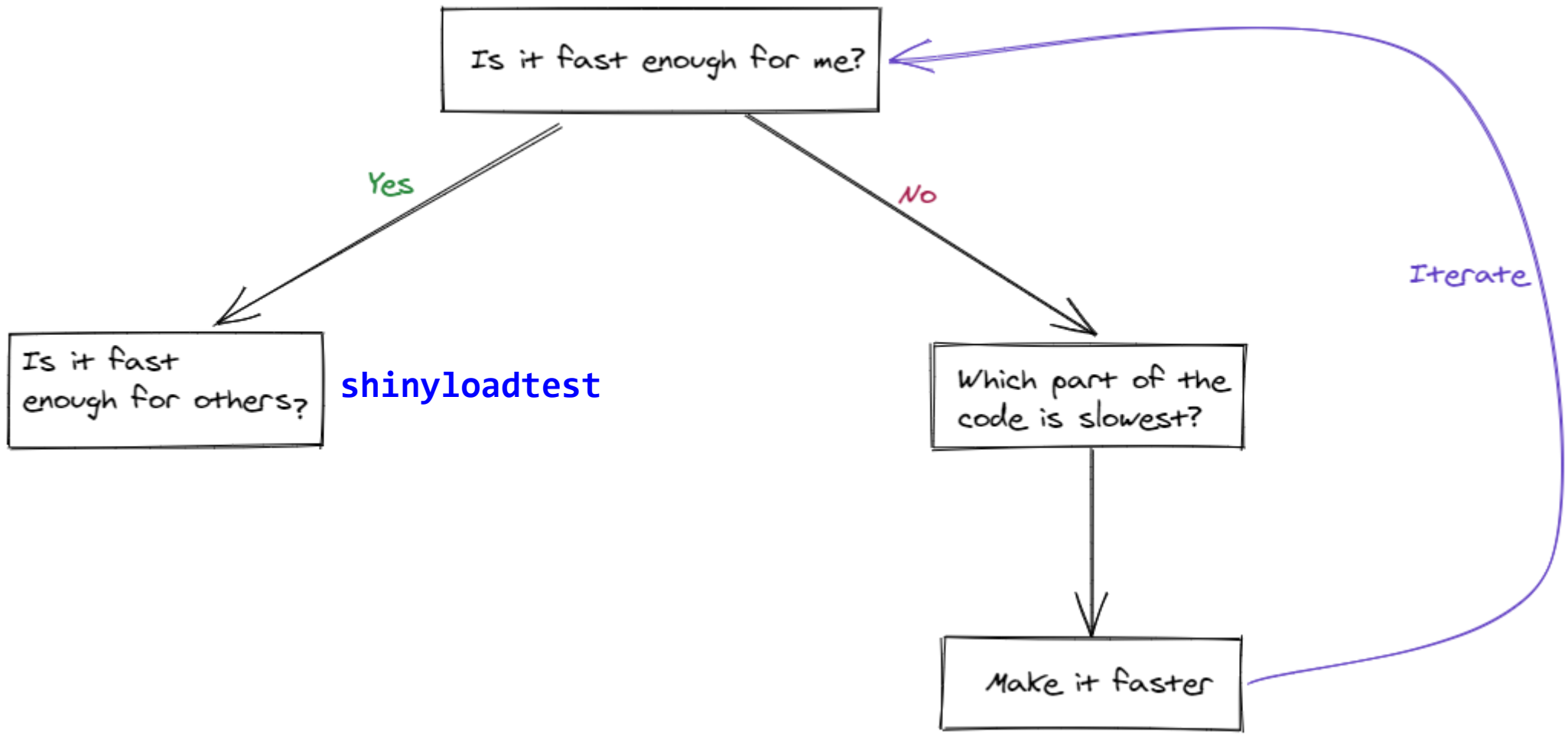

Is it fast enough for me?
No
Which part of the code is slowest?

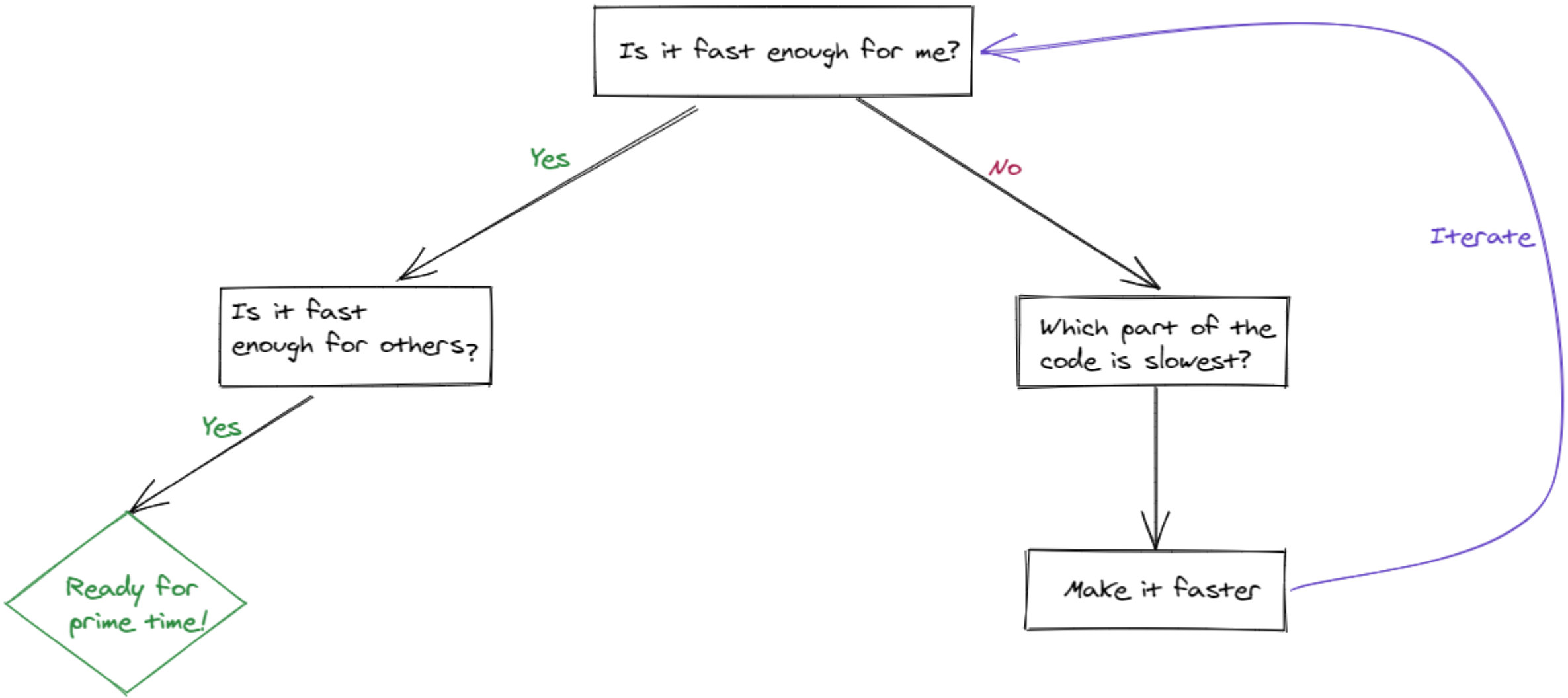

Is it fast enough for me?
No
Which part of the code is slowest?
profvis

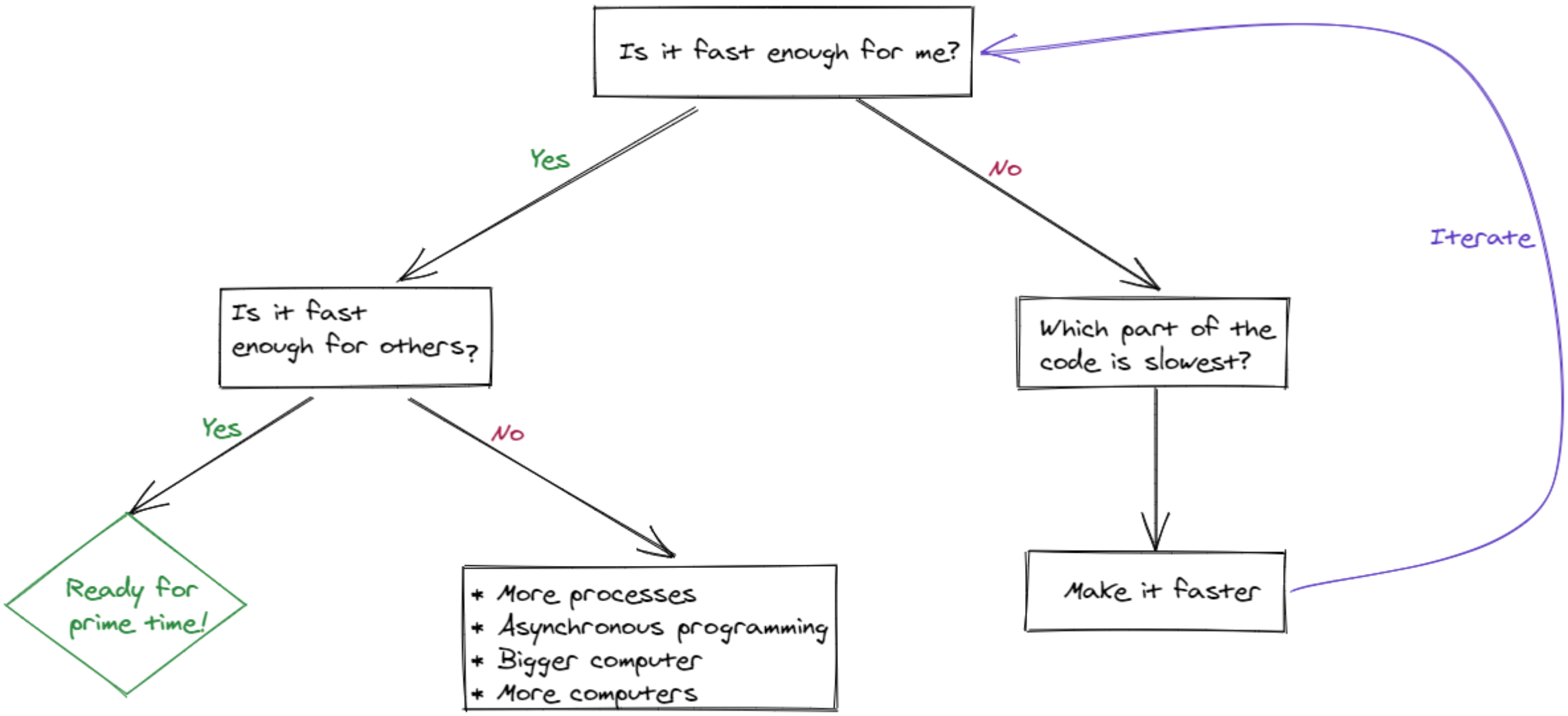


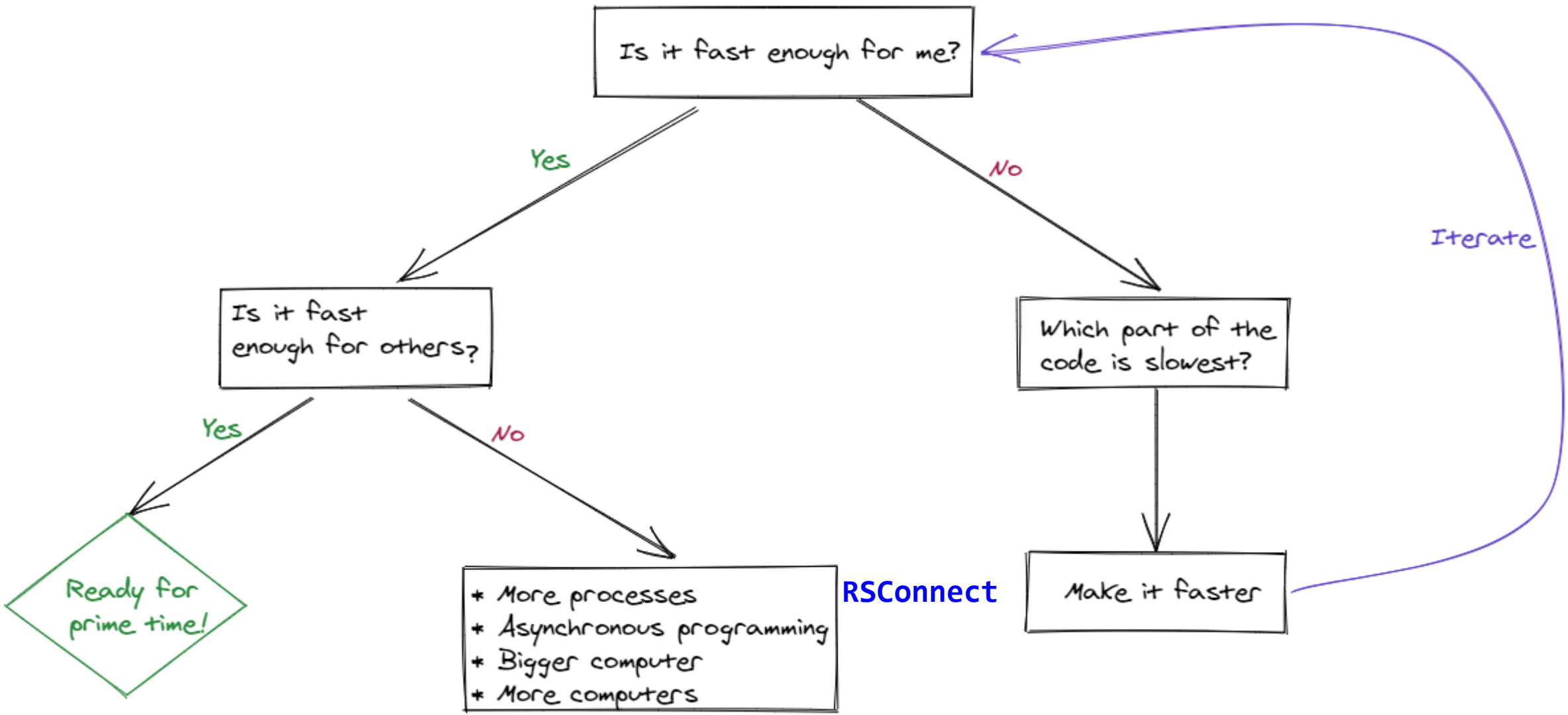
microbenchmark

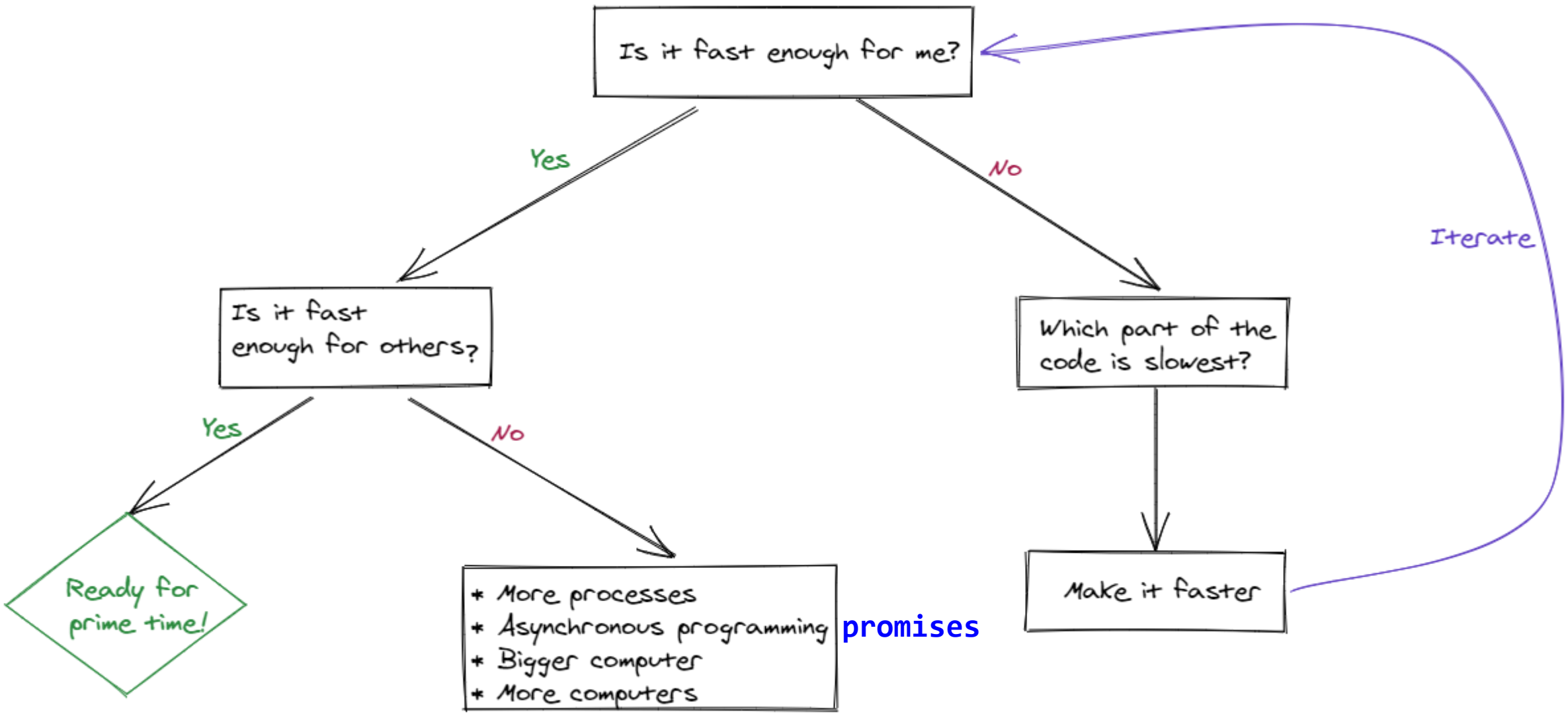

Is it fast enough for me?
No
Which part of the code is slowest?
Make it faster
Iterate


Is it fast enough for me?
Yes
No
Is it fast enough for others?
shinyloadtest
Which part of the code is slowest?
Make it faster
Iterate


Is it fast enough for me?
Yes
No
Is it fast enough for others?
Which part of the code is slowest?
Iterate
Yes
Ready for prime time!
Make it faster

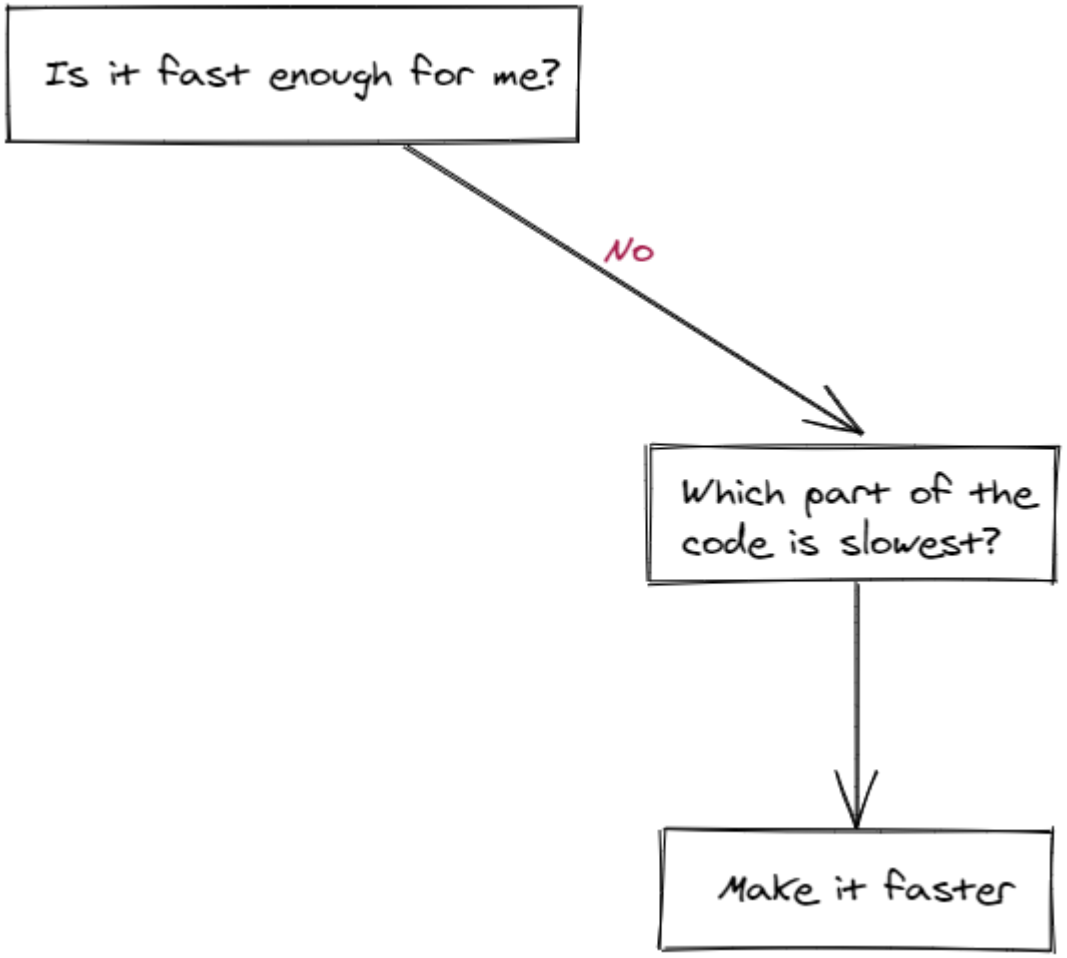

Is it fast enough for me?
Yes
No
Is it fast enough for others?
Which part of the code is slowest?
Iterate
Yes
No
Ready for prime time!
* More processes
* Asynchronous programming
* Bigger computer
* More computers
Make it faster


Is it fast enough for me?
Yes
No
Iterate
Is it fast enough for others?
Which part of the code is slowest?
Yes
No
Ready for prime time!
* More processes
* Asynchronous programming
* Bigger computer
* More computers
RSConnect
Make it faster


Is it fast enough for me?
Yes
No
Is it fast enough for others?
Which part of the code is slowest?
Iterate
Yes
No
Ready for prime time!
* More processes
* Asynchronous programming
* Bigger computer
* More computers
promises
Make it faster

Demo:
Profiling

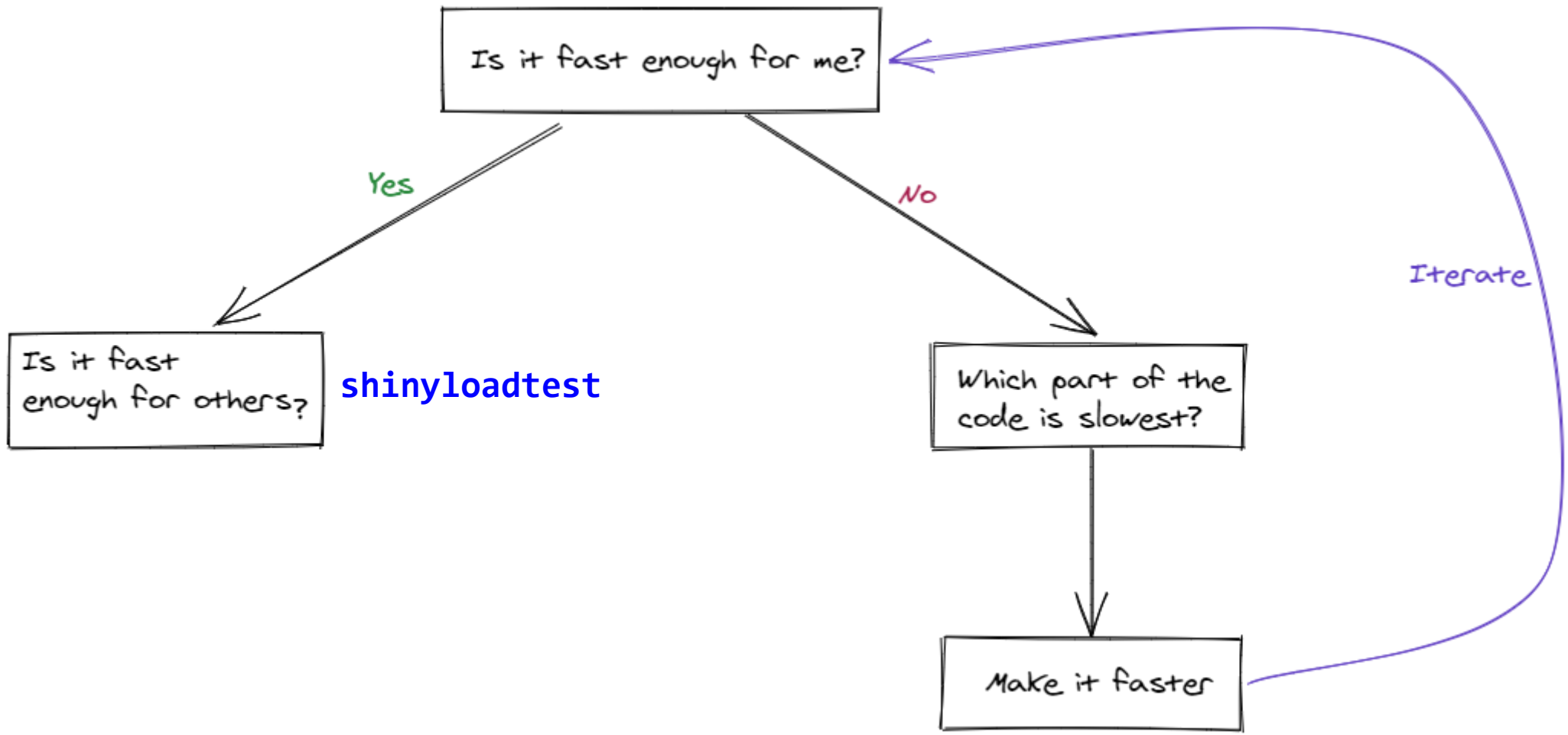
Your turn!
Profiling



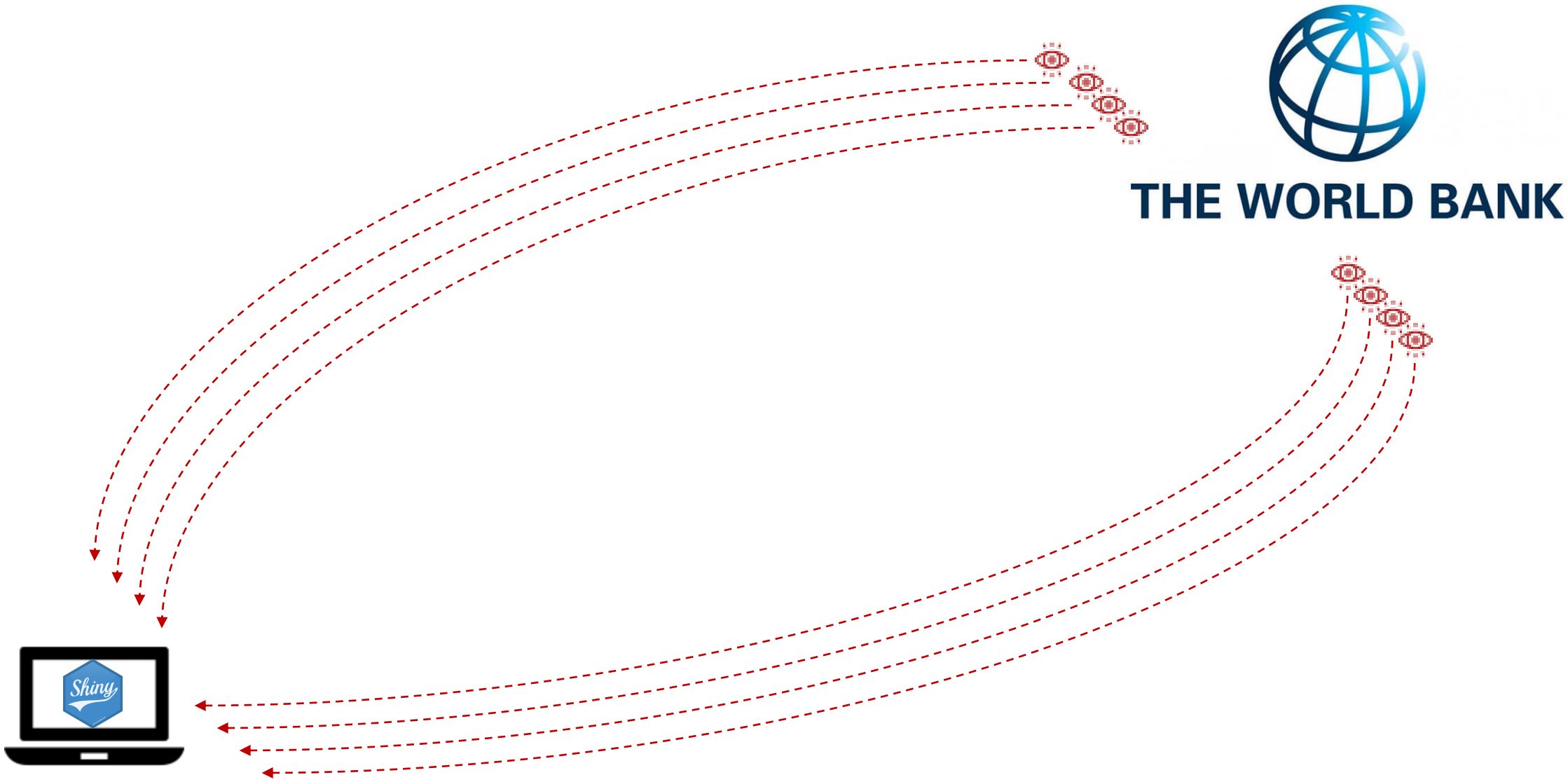
microbenchmark

Demo:
Benchmarking

Your turn!
Benchmarking


Is it fast enough for me?
Yes
No
Is it fast enough for others?
shinyloadtest
Which part of the code is slowest?
Make it faster
Iterate

THE WORLD BANK


THE WORLD BANK
Shiny

1 - Record a typical user session for the app

```
shinyloadtest::record_session('http://my_shiny_app/url')
```

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

THE WORLD BANK

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

Run the java application

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

THE WORLD BANK

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

Output of step 1

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

Shiny app URL

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

Number of users to simulate

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

Duration of the simulation

1 - **Record** a typical user session for the app

2 - Replay the session in parallel, simulating many simultaneous users accessing the app

```
java -jar shinycannon-1.1.0-45731f0.jar recording.log
http://w0lxopshyprd1b.worldbank.org:3939/shiny_test/
--workers 3 --loaded-duration-minutes 2
--output-dir run1
```

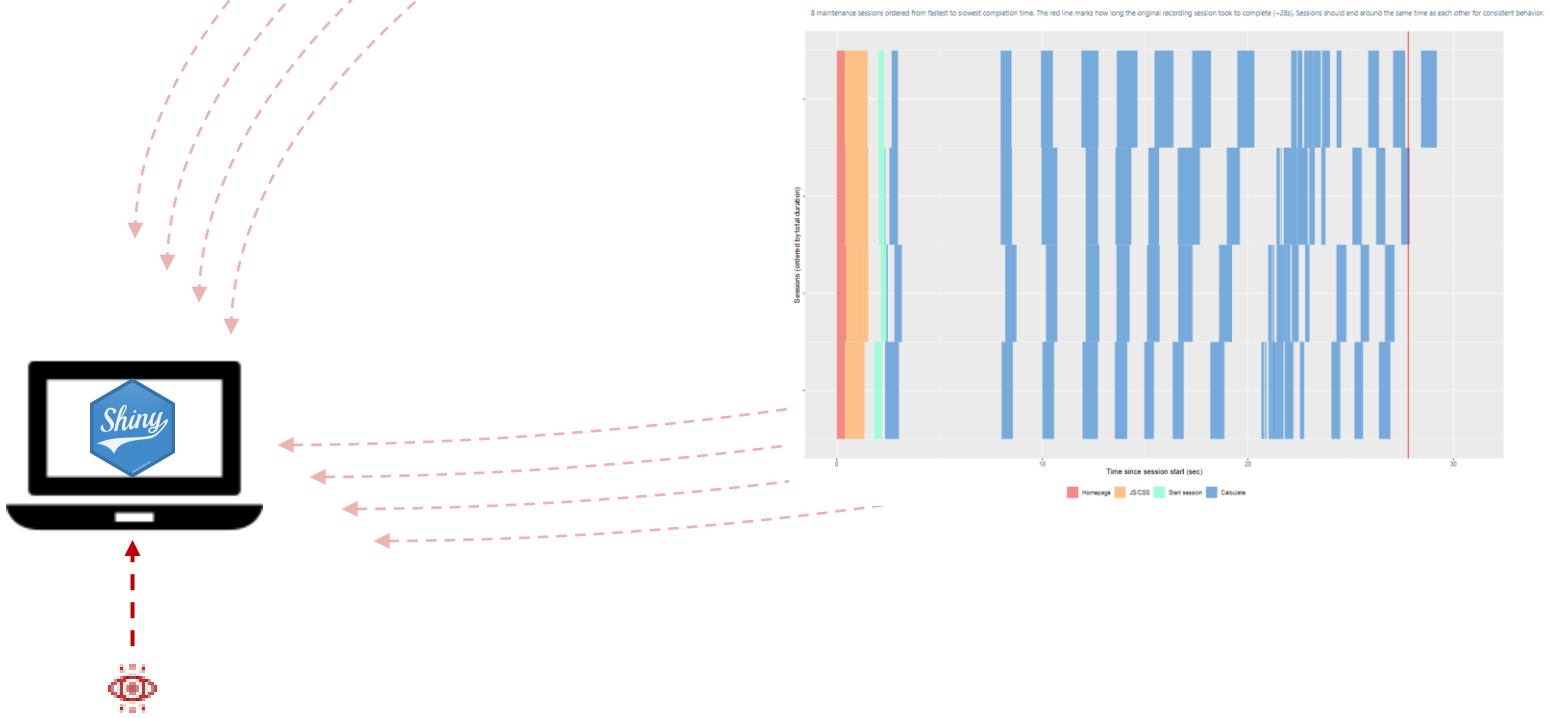
Directory where the results will be saved

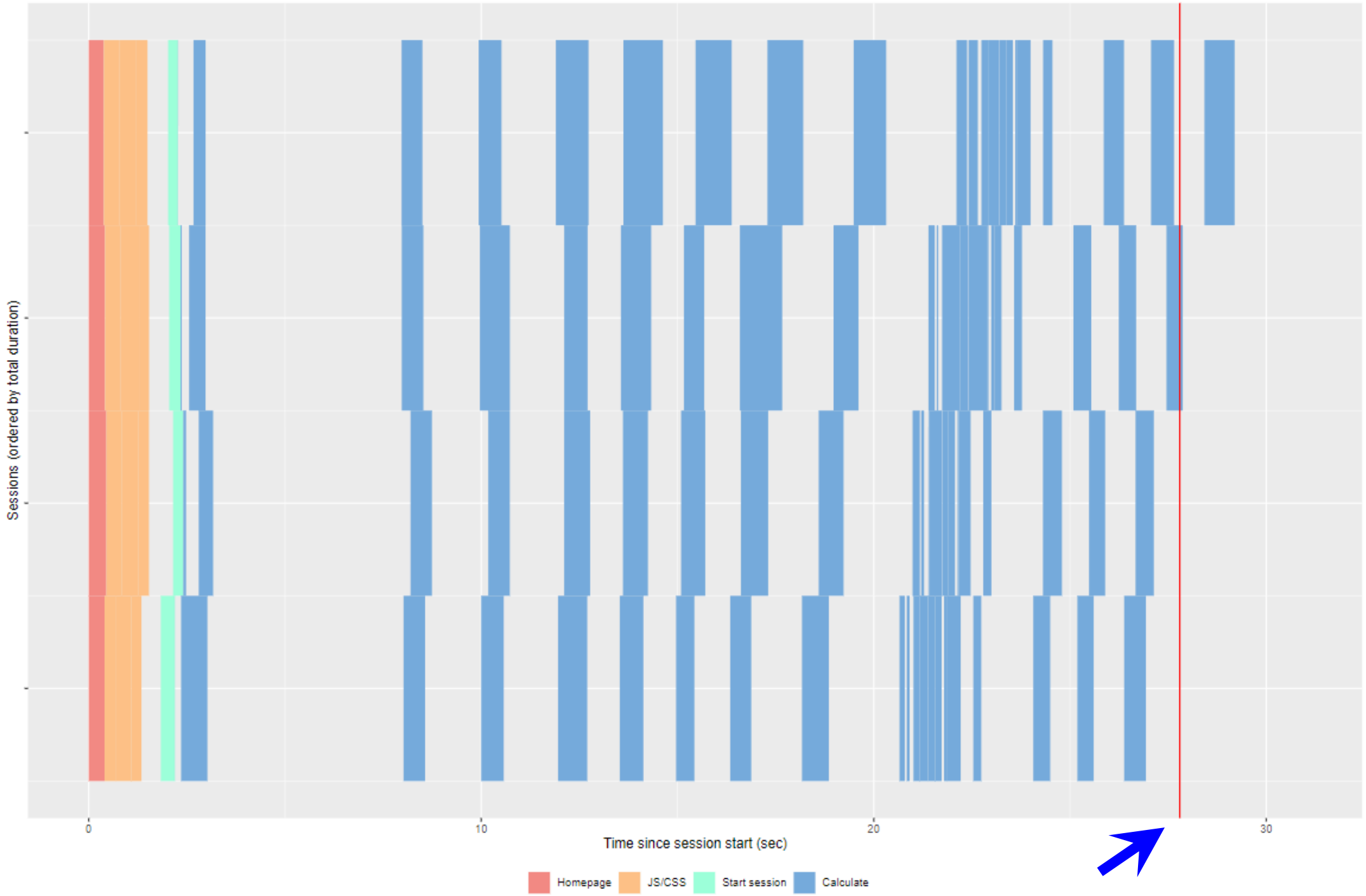
1 - **Record** a typical user session for the app

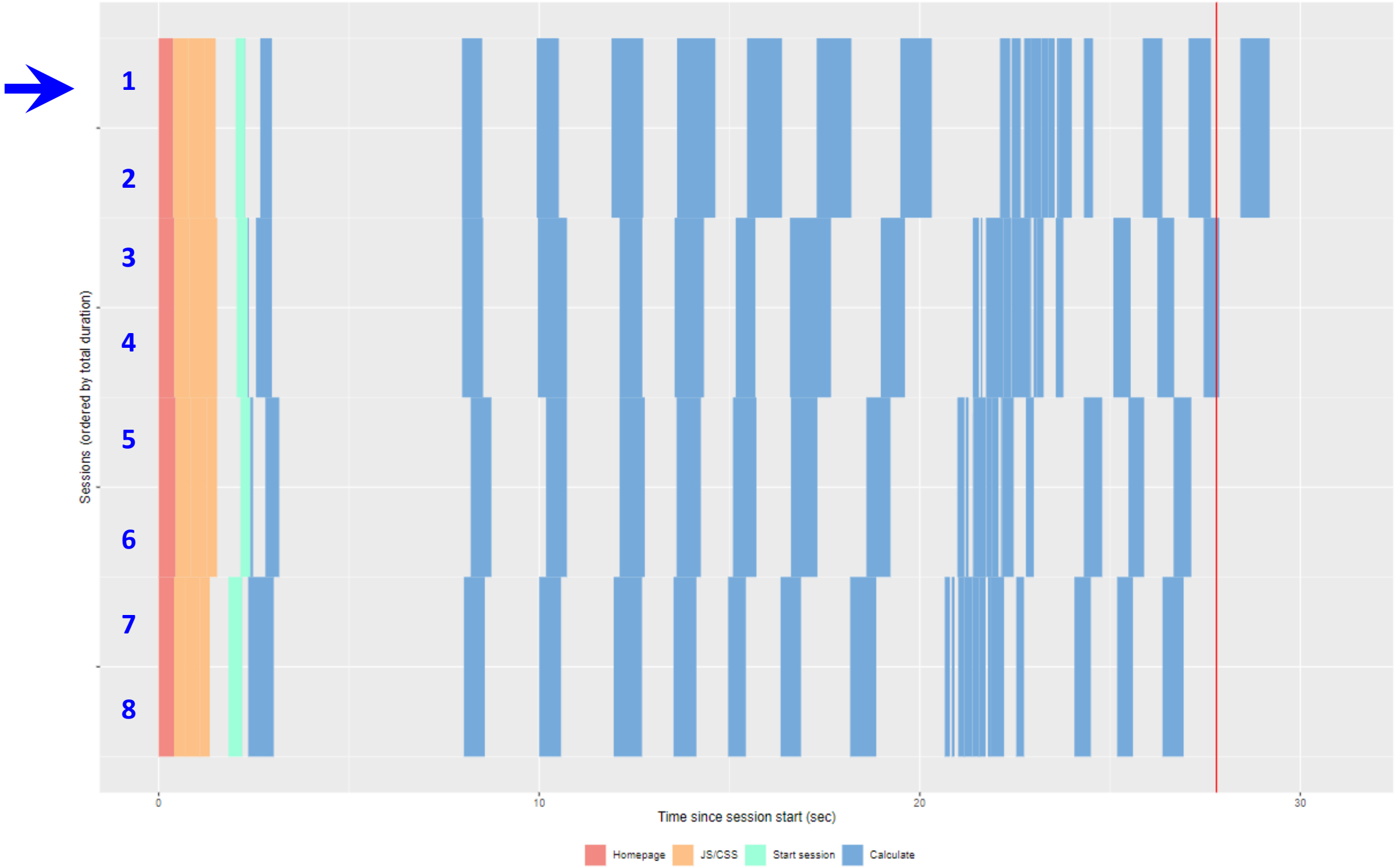
2 - **Replay the session in parallel, simulating many simultaneous users accessing the app**

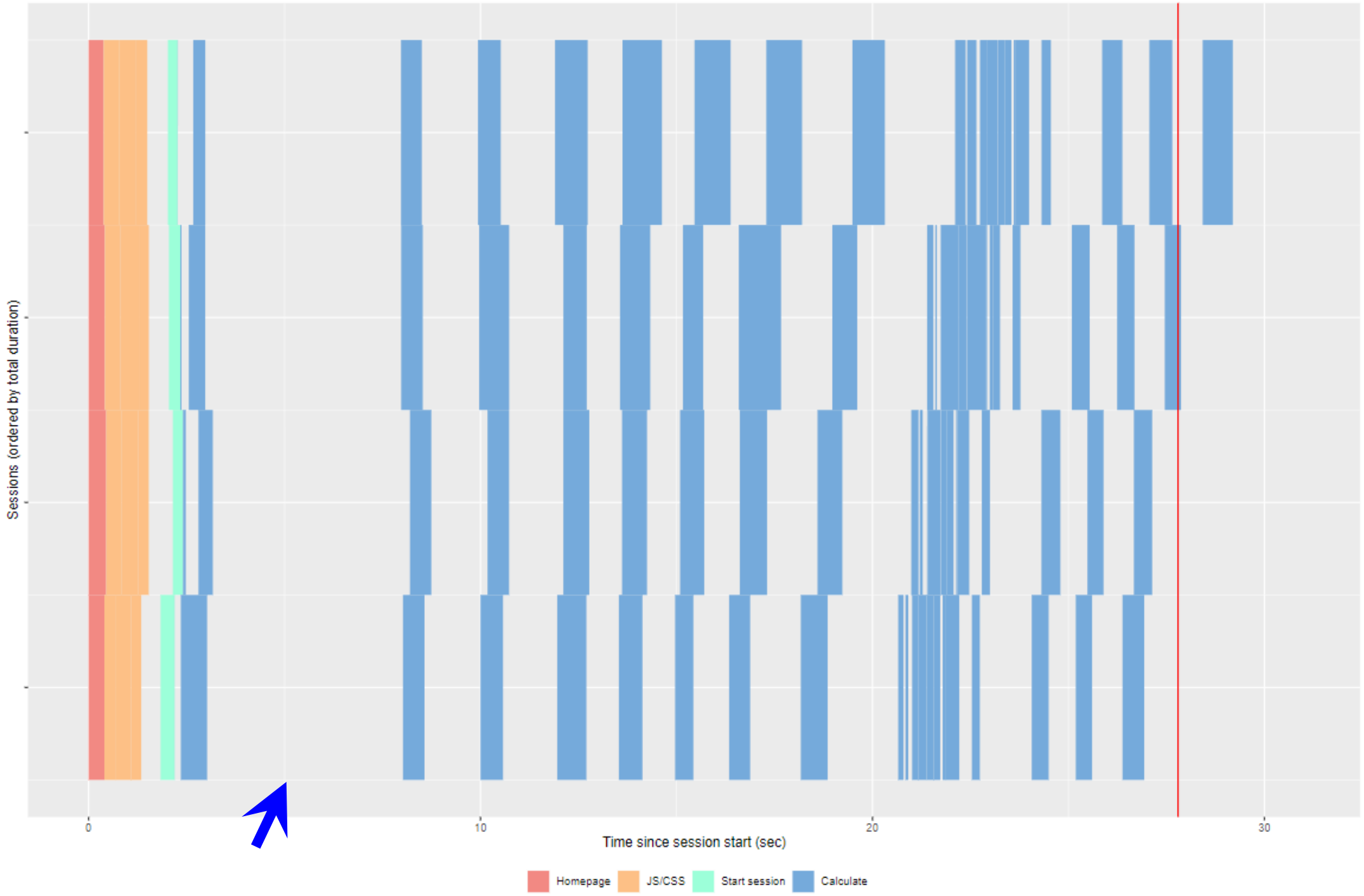
3 - Analyze results

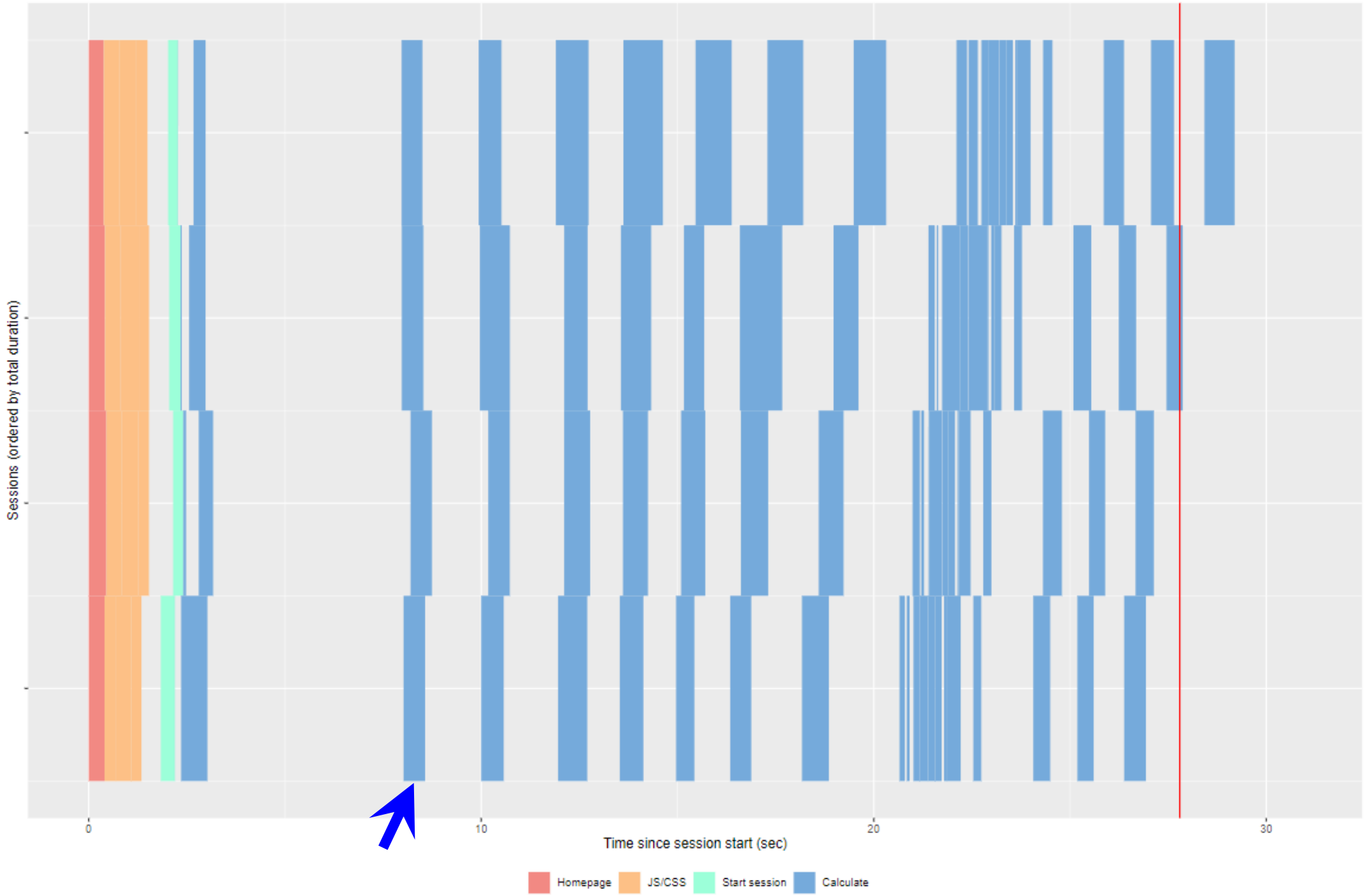
```
df <- shinyloadtest::load_runs("run1")
shinyloadtest::shinyloadtest_report(df, "run1.html")
```



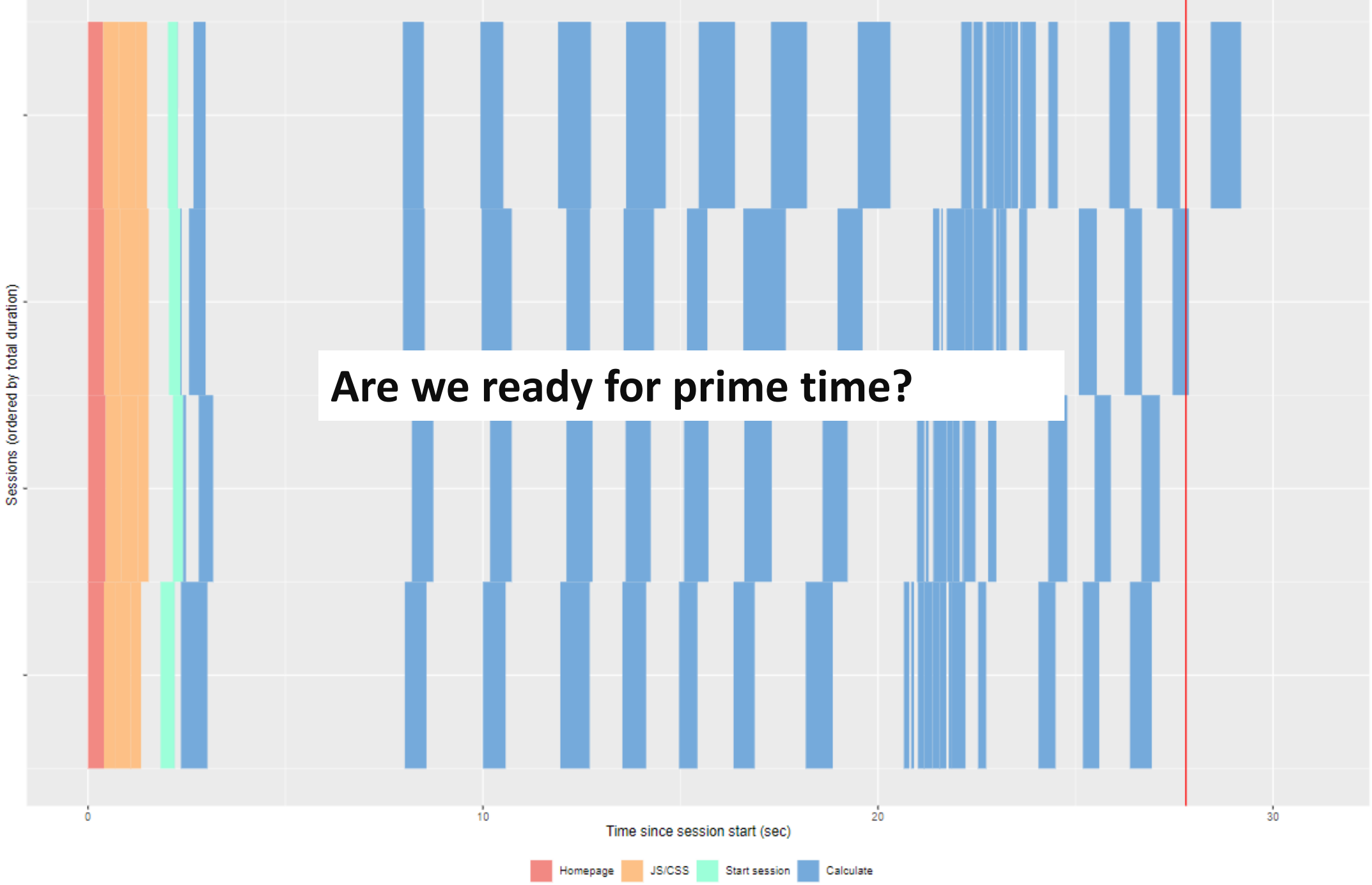

Sessions (ordered by total duration)
0
10
20
30
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate


1
2
3
4
5
6
7
8
Sessions (ordered by total duration)
0
10
20
30
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate

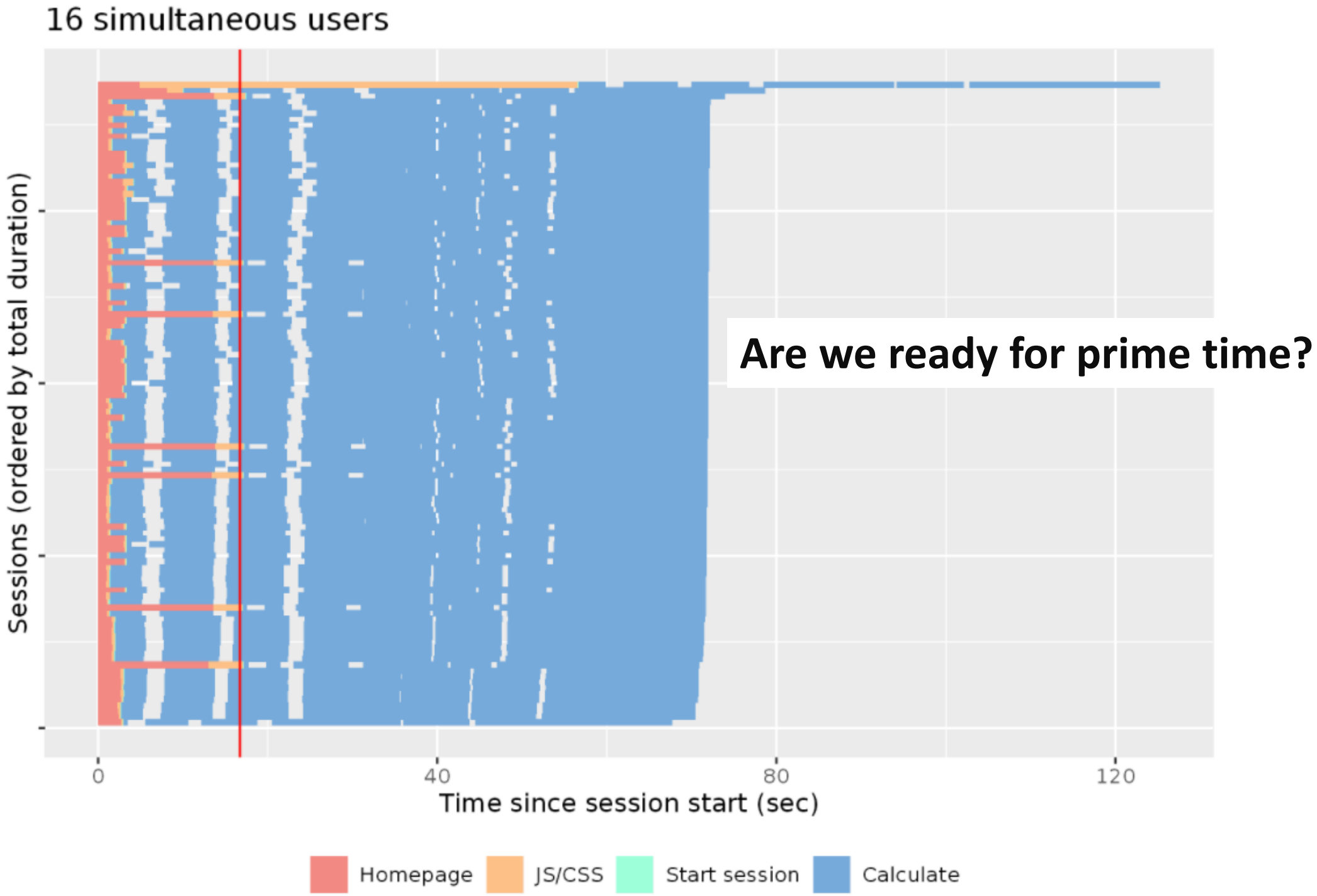

Sessions (ordered by total duration)
0
10
20
30
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate


Sessions (ordered by total duration)
0
10
20
30
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate

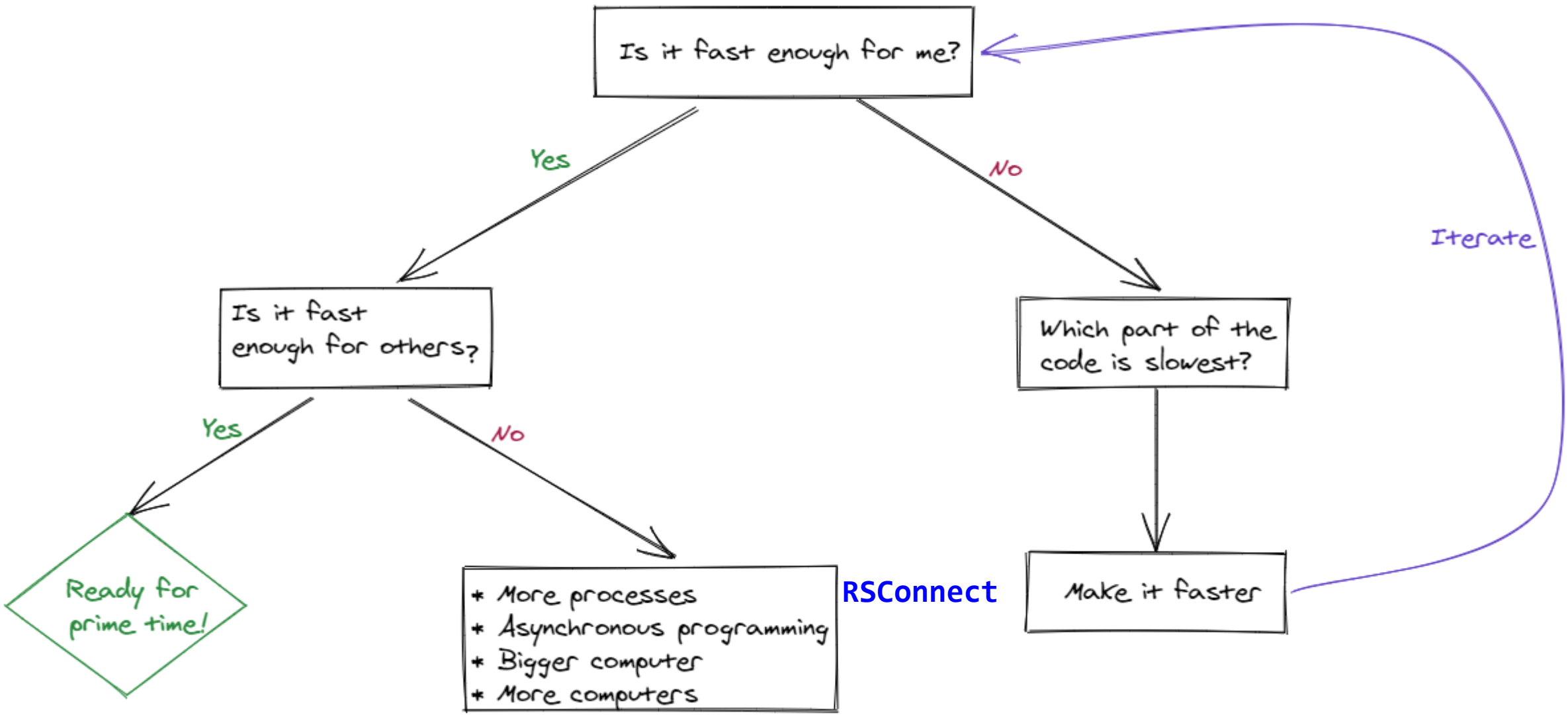
Demo:
Load-testing

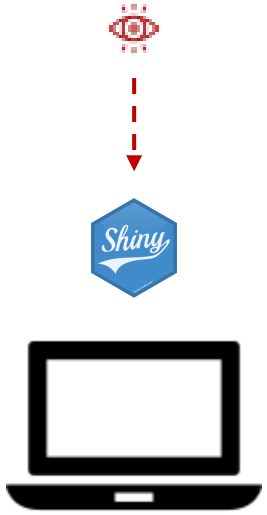


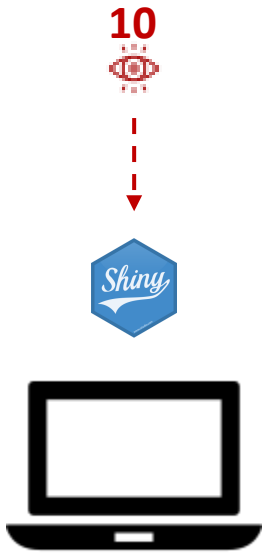
Are we ready for prime time?

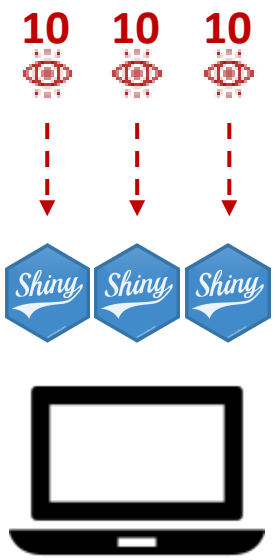


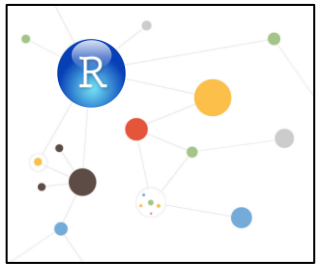
Are we ready for prime time?


Is it fast enough for me?
Yes
No
Iterate
Is it fast enough for others?
Which part of the code is slowest?
Yes
No
Ready for prime time!
* More processes
* Asynchronous programming
* Bigger computer
* More computers
RSConnect
Make it faster

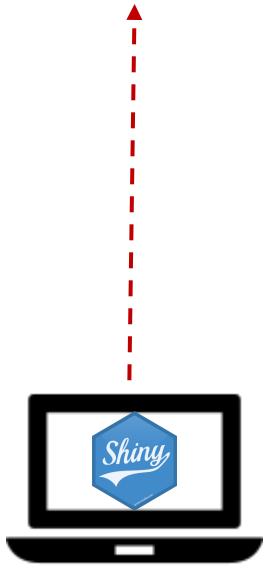

Shiny

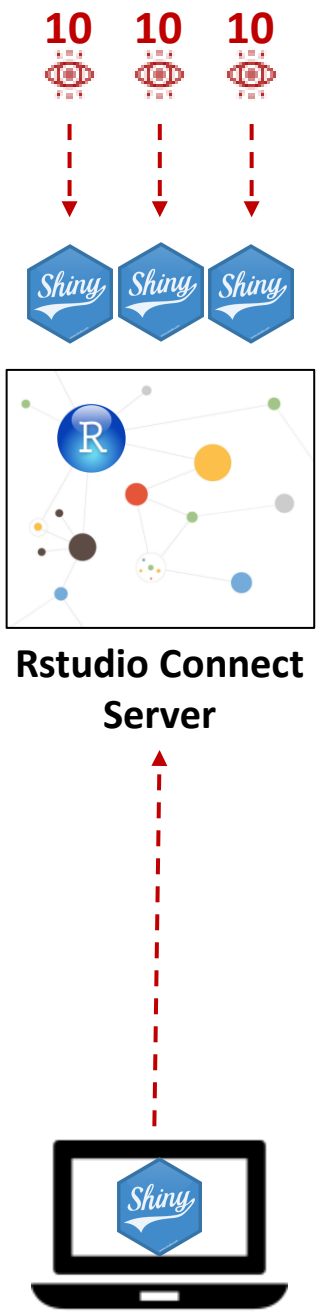

10
Shiny


10
10
10
Shiny
Shiny
Shiny



Rstudio Connect Server

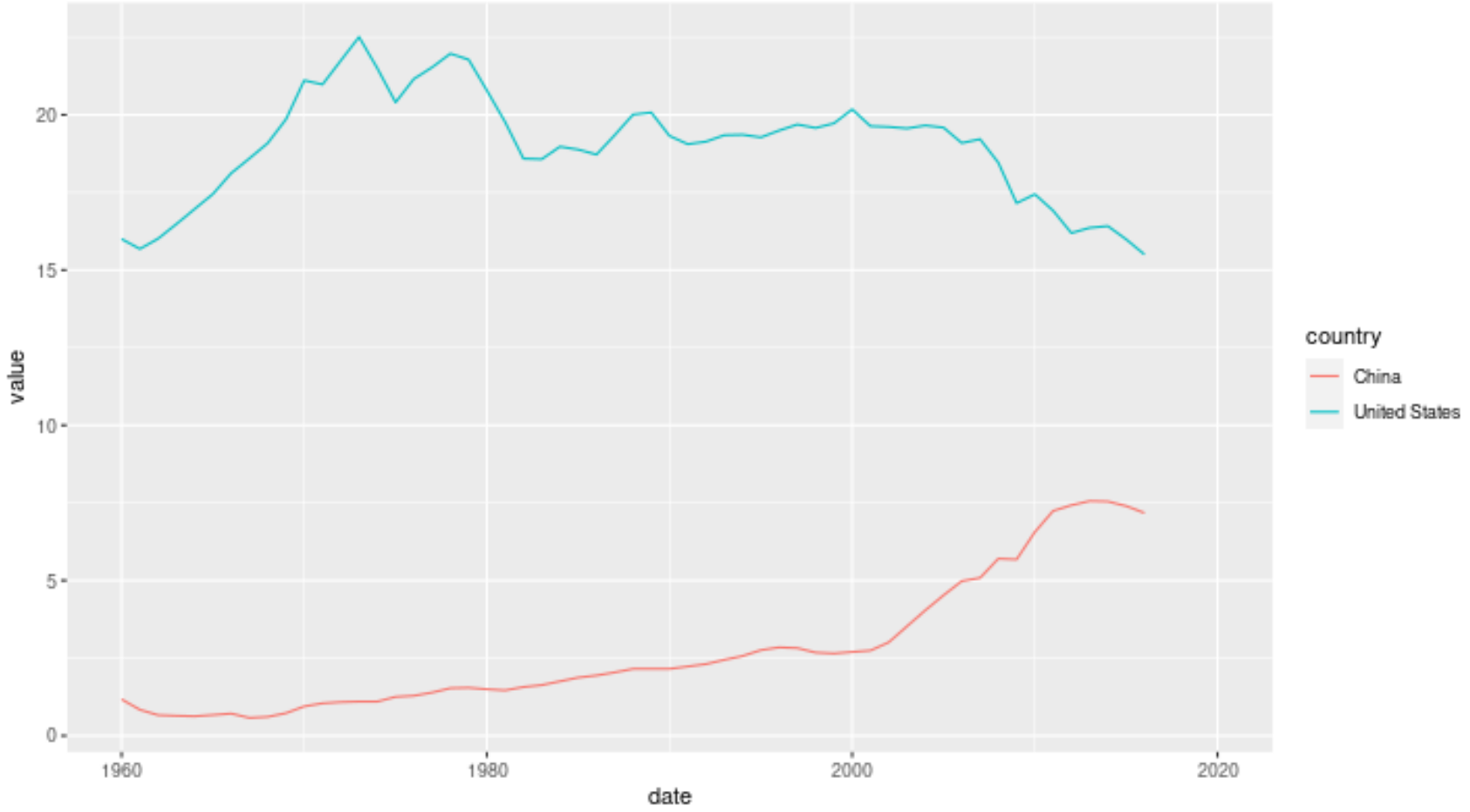



10
10
10
Shiny
Shiny
Shiny
R
Rstudio Connect Server
Shiny

CO2 emissions (metric tons per capita)

Select countries

China United States



Info Access Runtime Schedule Tags Vars Lo

Runtime settings

Use server defaults

Max processes

5

Min processes

1

Max connections per process

15

Load factor: 0.50

0 1

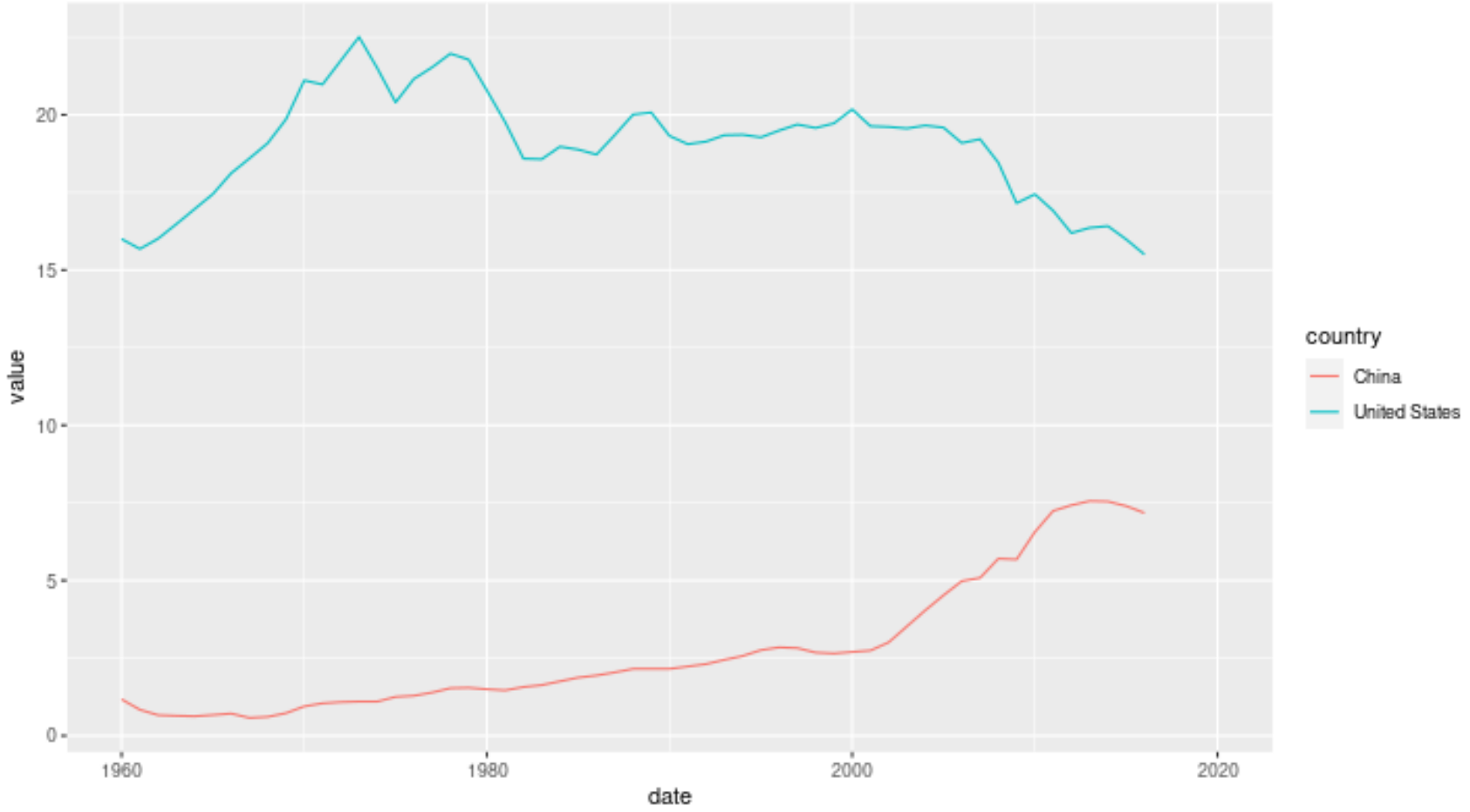
Idle Timeout per process

5

CO2 emissions (metric tons per capita)

Select countries

China United States



Info Access Runtime Schedule Tags Vars Lo

Runtime settings

☐ Use server defaults

Max processes

5

Min processes

1

Max connections per process

15

Load factor: 0.50

0 1

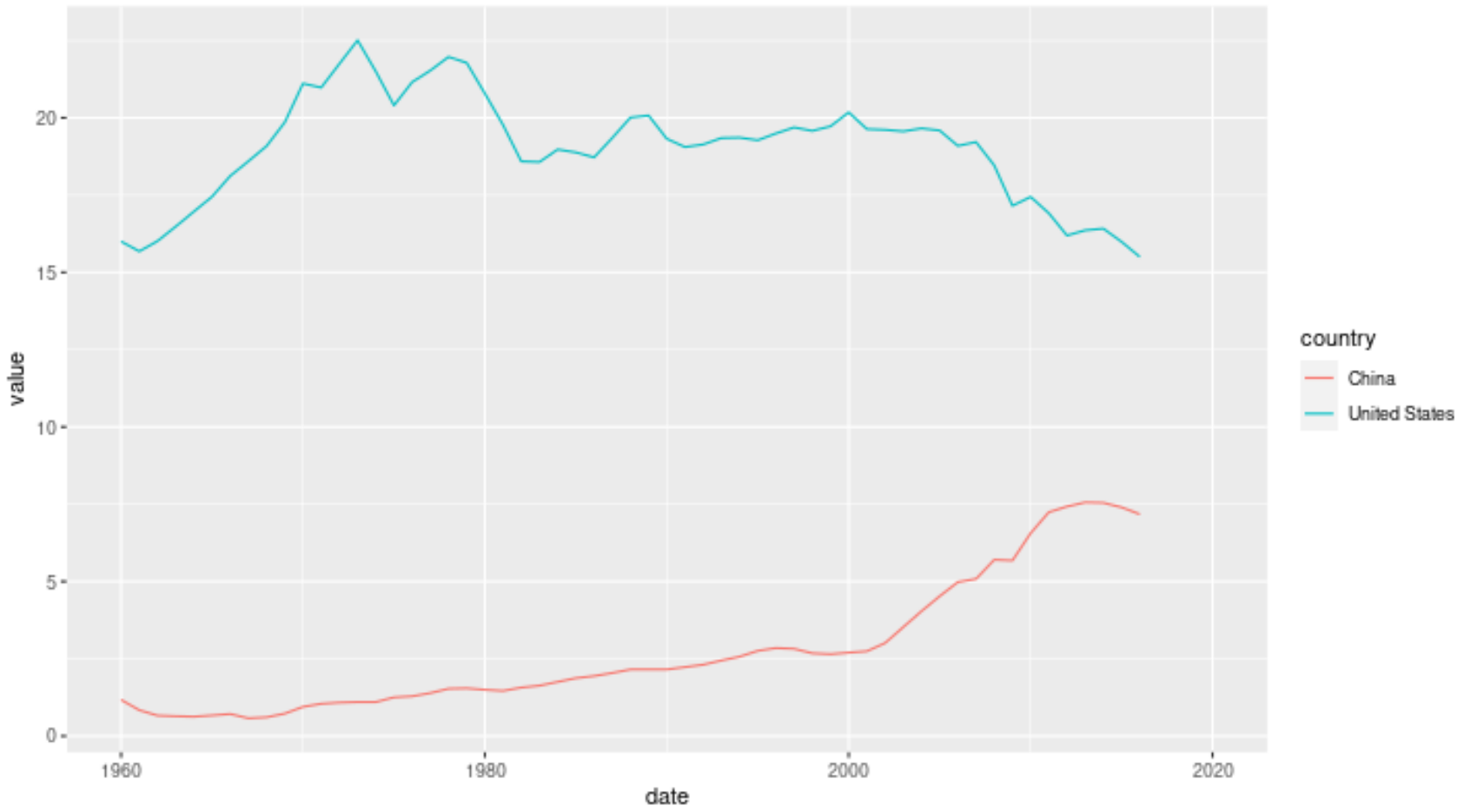
Idle Timeout per process

5

CO2 emissions (metric tons per capita)

Select countries

China United States



Info Access Runtime Schedule Tags Vars Lo

Runtime settings

Use server defaults

Max processes

5

Min processes

1

Max connections per process

15

Load factor: 0.50

0 1

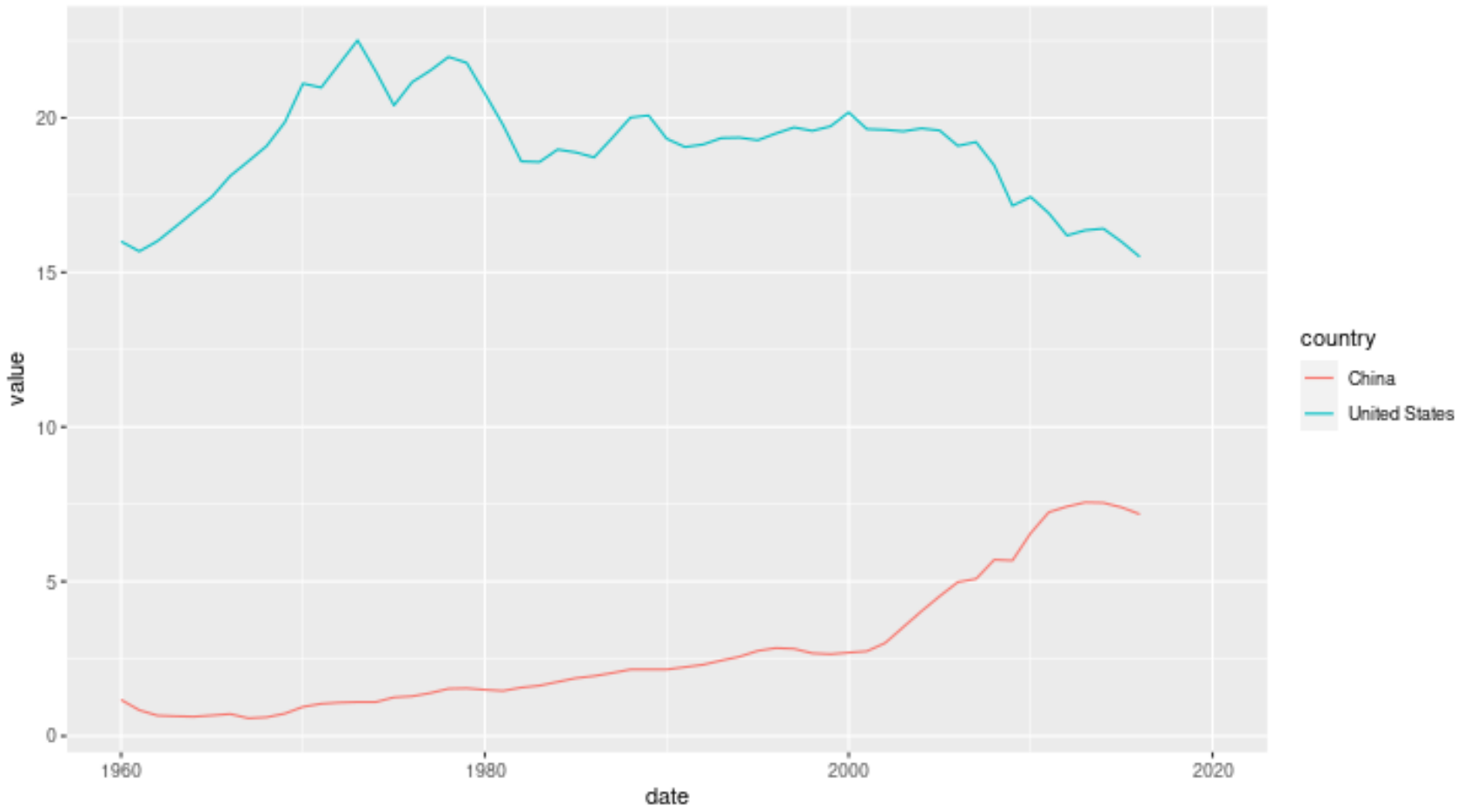
Idle Timeout per process

5

CO2 emissions (metric tons per capita)

Select countries

China United States



Info Access Runtime Schedule Tags Vars Lo

Runtime settings

Use server defaults

Max processes

5

Min processes

1

Max connections per process

15

Load factor: 0.50

0 1

Idle Timeout per process

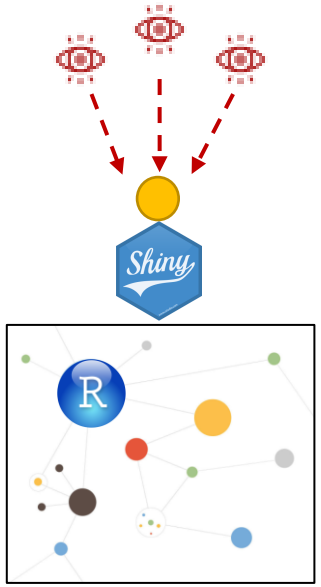
5

Scoping

```r
library(shiny)
library(ggplot2)


country_list <- readr::read_rds("input/country_list.rds")

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Number of confirmed covid cases"),

    # Sidebar with a select input
    sidebarLayout(
        sidebarPanel(
            selectInput("country",
                        "Select countries",
                        choices = country_list,
                        selected = c("China", "United States"),
                        multiple = TRUE)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("plot")
        )
    )
)

# Define server logic
server <- function(input, output) {

    df <- readr::read_csv("input/covid_small.csv")

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```

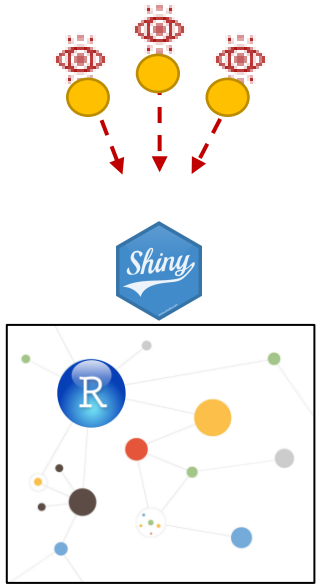
```r
library(shiny)
library(ggplot2)


country_list <- readr::read_rds("input/country_list.rds")

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Number of confirmed covid cases"),

    # Sidebar with a select input
    sidebarLayout(
        sidebarPanel(
            selectInput("country",
                        "Select countries",
                        choices = country_list,
                        selected = c("China", "United States"),
                        multiple = TRUE)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("plot")
        )
    )
)

# Define server logic
server <- function(input, output) {

    df <- readr::read_csv("input/covid_small.csv")

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
library(shiny)
library(ggplot2)

df <- readr::read_csv("input/covid_small.csv")
country_list <- readr::read_rds("input/country_list.rds")

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Number of confirmed covid cases"),

    # Sidebar with a select input
    sidebarLayout(
        sidebarPanel(
            selectInput("country",
                        "Select countries",
                        choices = country_list,
                        selected = c("China", "United States"),
                        multiple = TRUE)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("plot")
        )
    )
)

# Define server logic
server <- function(input, output) {

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```



Rstudio Connect Server

```r
library(shiny)
library(ggplot2)

df <- readr::read_csv("input/covid_small.csv")
country_list <- readr::read_rds("input/country_list.rds")

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Number of confirmed covid cases"),

    # Sidebar with a select input
    sidebarLayout(
        sidebarPanel(
            selectInput("country",
                        "Select countries",
                        choices = country_list,
                        selected = c("China", "United States"),
                        multiple = TRUE)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("plot")
        )
    )
)

# Define server logic
server <- function(input, output) {

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```



Rstudio Connect Server

```r
library(shiny)
library(ggplot2)


country_list <- readr::read_rds("input/country_list.rds")

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Number of confirmed covid cases"),

    # Sidebar with a select input
    sidebarLayout(
        sidebarPanel(
            selectInput("country",
                        "Select countries",
                        choices = country_list,
                        selected = c("China", "United States"),
                        multiple = TRUE)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("plot")
        )
    )
)

# Define server logic
server <- function(input, output) {

    df <- readr::read_csv("input/covid_small.csv")

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```

Improving performance

Some low-hanging fruit techniques

Optimization techniques:

- ✓ **Move all pre-processing outside of your app**

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ **Use efficient data formats**

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ Use efficient data formats
- ✓ **Use caching (Very easy as of shiny 1.6)**

Without cache

```r
# Define server logic
server <- function(input, output) {

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}
```

Without cache

```r
# Define server logic
server <- function(input, output) {

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    })
}
```

With cache

```r
# Define server logic
server <- function(input, output) {

    output$plot <- renderPlot({
        x    <- df[df$country_name %in% input$country, ]

        ggplot(x, aes(x = date,
                      y = total_confirmed,
                      group = country_code,
                      color = country_name)) +
            geom_line()

    }) %>%
        bindCache(input$country)
}
```

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ Use efficient data formats
- ✓ Use caching (Very easy as of shiny 1.6)
- ✓ **Leverage RSConnect features**

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ Use efficient data formats
- ✓ Use caching (Very easy as of shiny 1.6)
- ✓ Leverage RSConnect features
- ✓ **(Use efficient computations)**

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ Use efficient data formats
- ✓ Use caching (Very easy as of shiny 1.6)
- ✓ Leverage RSConnect features
- ✓ (Use efficient computations)
- ✓ **(Use reactive expressions carefully)**

Optimization techniques:

- ✓ Move all pre-processing outside of your app
- ✓ Use efficient data formats
- ✓ Use caching (Very easy as of shiny 1.6)
- ✓ Leverage RSConnect features
- ✓ (Use efficient computations)
- ✓ (Use reactive expressions carefully)
- ✓ **(Use asynchronous programming)**

Not enough memory?

flickr.com/photos/bachand/1086041678

Demo:
arrow

Your turn!

arrow

Thank you!

Resources

Overview

- Mastering Shiny book: https://mastering-shiny.org/
- Rstudio shiny references: https://shiny.rstudio.com/articles/

Performance

- Profiling: https://shiny.rstudio.com/articles/profiling.html
- Benchmarking: https://rdpeng.github.io/RProgDA/profiling-and-benchmarking.html
- Load-testing: https://rstudio.github.io/shinyloadtest/
- Scoping: https://shiny.rstudio.com/articles/scoping.html
- Caching: https://shiny.rstudio.com/articles/caching.html
- Asynchronous programming: https://shiny.rstudio.com/articles/async.html
- RstudioConnect settings: https://docs.rstudio.com/connect/user/content-settings/#content-runtime

Maintenance

- R packages: https://r-pkgs.org/
- Unit tests: https://r-pkgs.org/tests.html
- Golem framework: https://thinkr-open.github.io/golem/
- Modules: https://shiny.rstudio.com/articles/modules.html