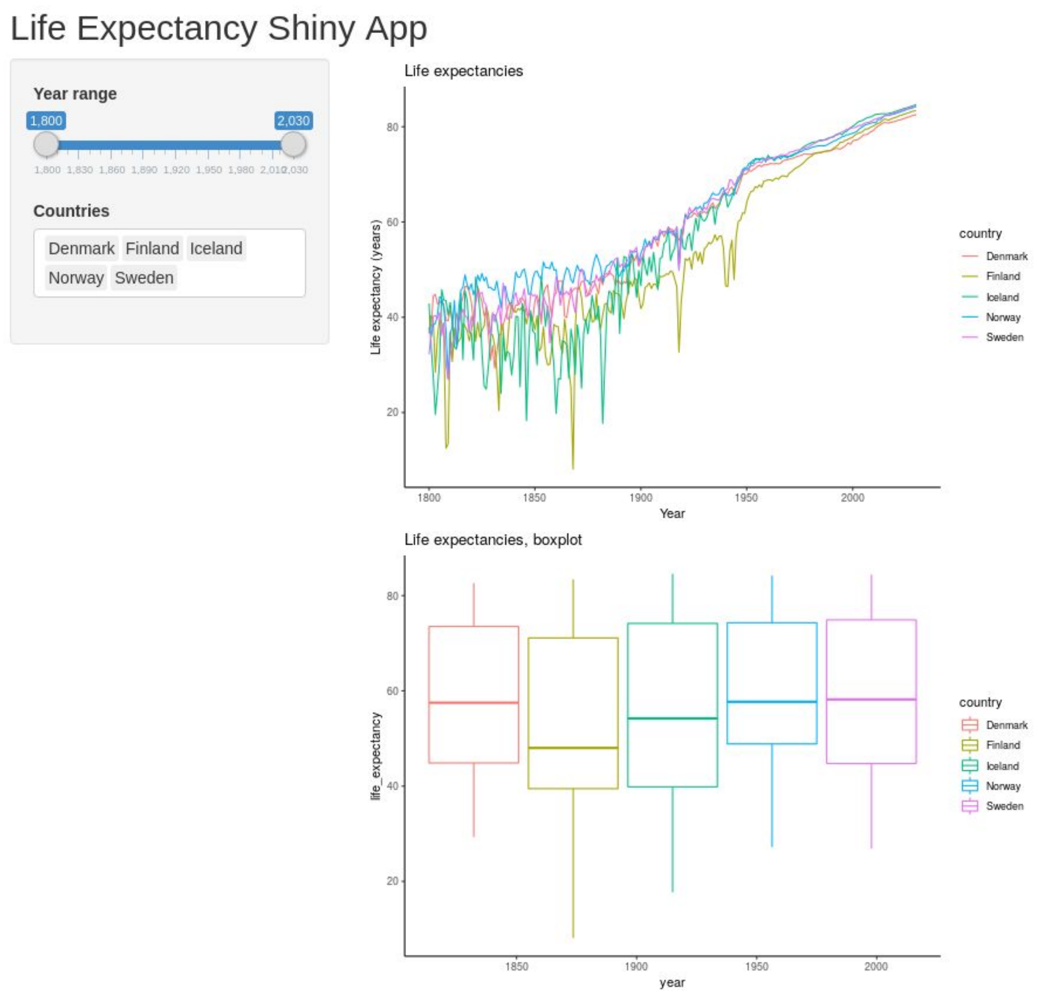
R-Shiny from scratch

Skåne R User Meetup, 20th April 2020
Jakob Willforss

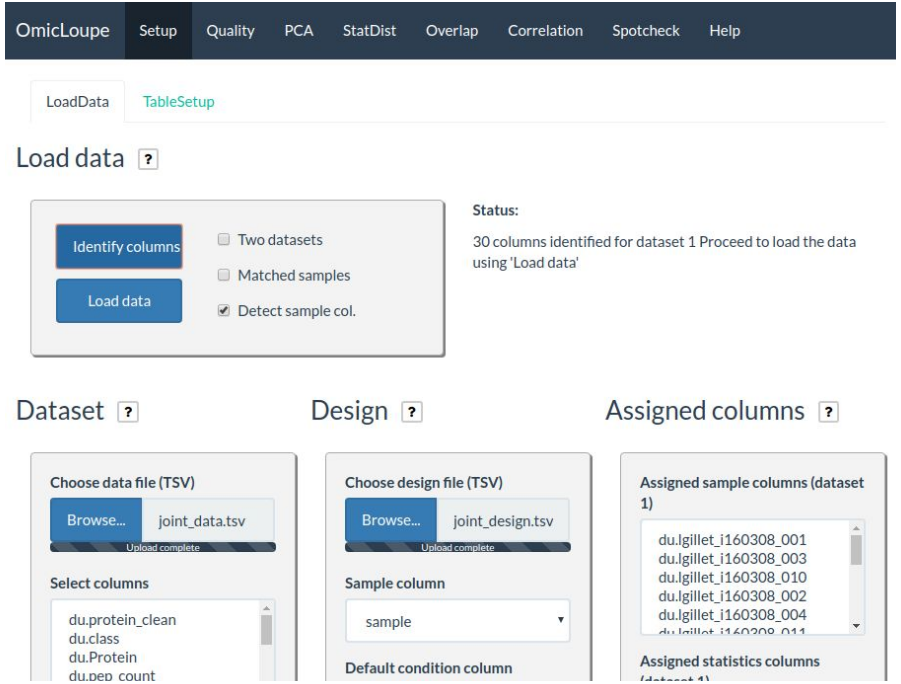
Skåne R User group
https://www.meetup.com/Skane-R-User-Group

github.com/Jakob37
linkedin.com/in/jakobwillforss

"**{shiny}** is an R package that makes it easy to build **interactive web apps straight from R**"



Life Expectancy Shiny App



Load data ?

Two datasets
Matched samples
Detect sample col. ✓

Status:

30 columns identified for dataset 1 Proceed to load the data using 'Load data'

Dataset ?

Choose data file (TSV)

joint_data.tsv

Upload complete

Select columns

du.protein_clean
du.class
du.Protein
du.pep_count

Design ?

Choose design file (TSV)

joint_design.tsv

Upload complete

Sample column

sample

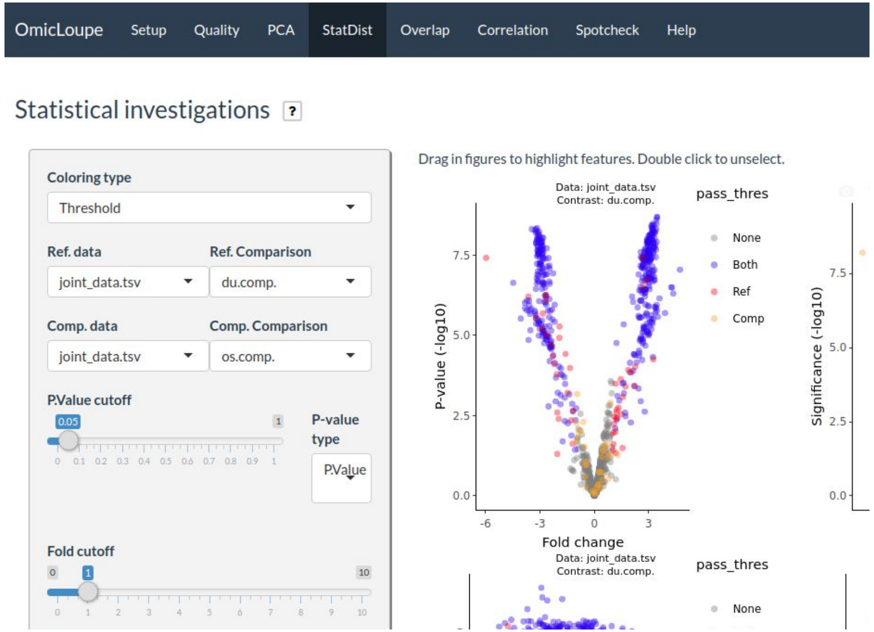
Default condition column

Assigned columns ?

Assigned sample columns (dataset 1)

du.lgillet_i160308_001
du.lgillet_i160308_003
du.lgillet_i160308_010
du.lgillet_i160308_002
du.lgillet_i160308_004

Assigned statistics columns (dataset 1)



Statistical investigations ?

Drag in figures to highlight features. Double click to unselect.

Coloring type: Threshold

Ref. data: joint_data.tsv

Ref. Comparison: du.comp.

Comp. data: joint_data.tsv

Comp. Comparison: os.comp.

P.Value cutoff: 0.05

P-value type: P.Value

Fold cutoff: 1

Why {shiny}

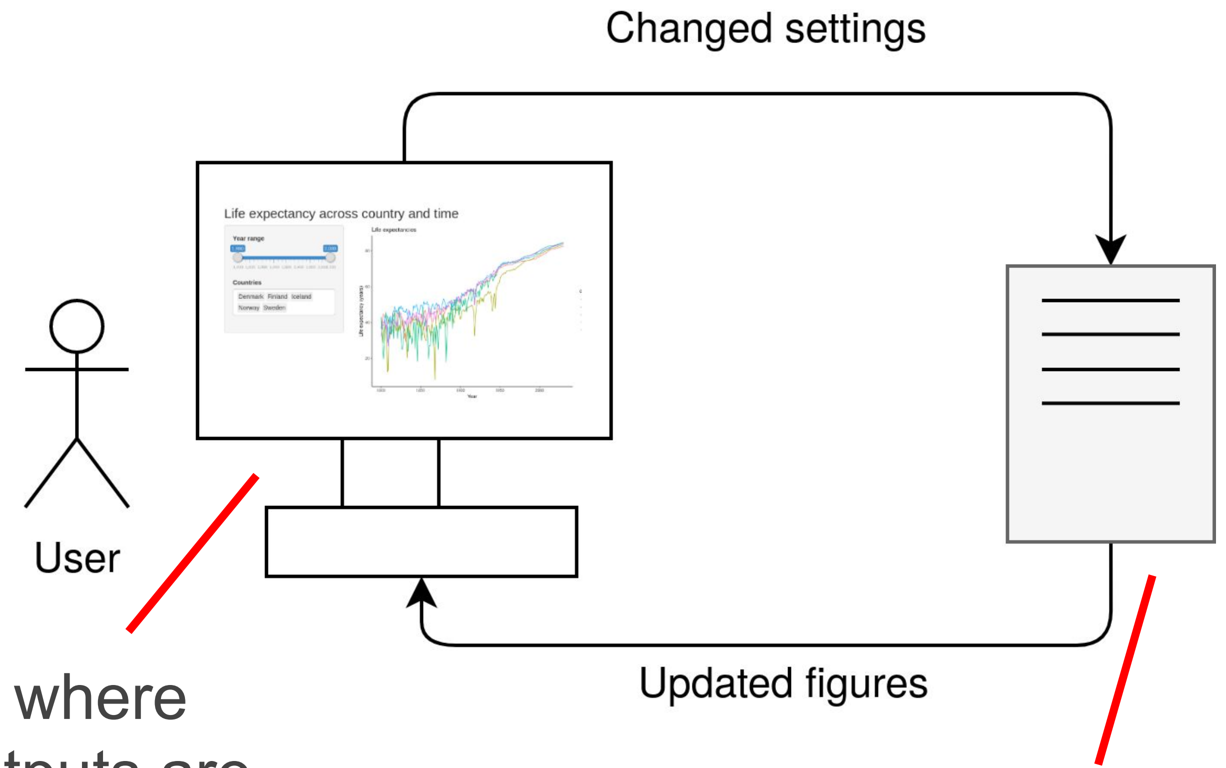
- Rapidly deploy R you analysis as web app
- Build without Linux- and web-programming knowledge
- … but is extendible with HTML, CSS, JavaScript
- **Easy to code, easy to deploy**

Some use cases

- Explore your data **interactively** without need to continuously rerun your code
- **Allow others** coworkers/boss/client **to explore** and draw conclusions themselves
- Setup **dashboards** to visualize incoming data in real time
- Build data analysis **applications**

The UI, server and reactivity

The UI / server structure

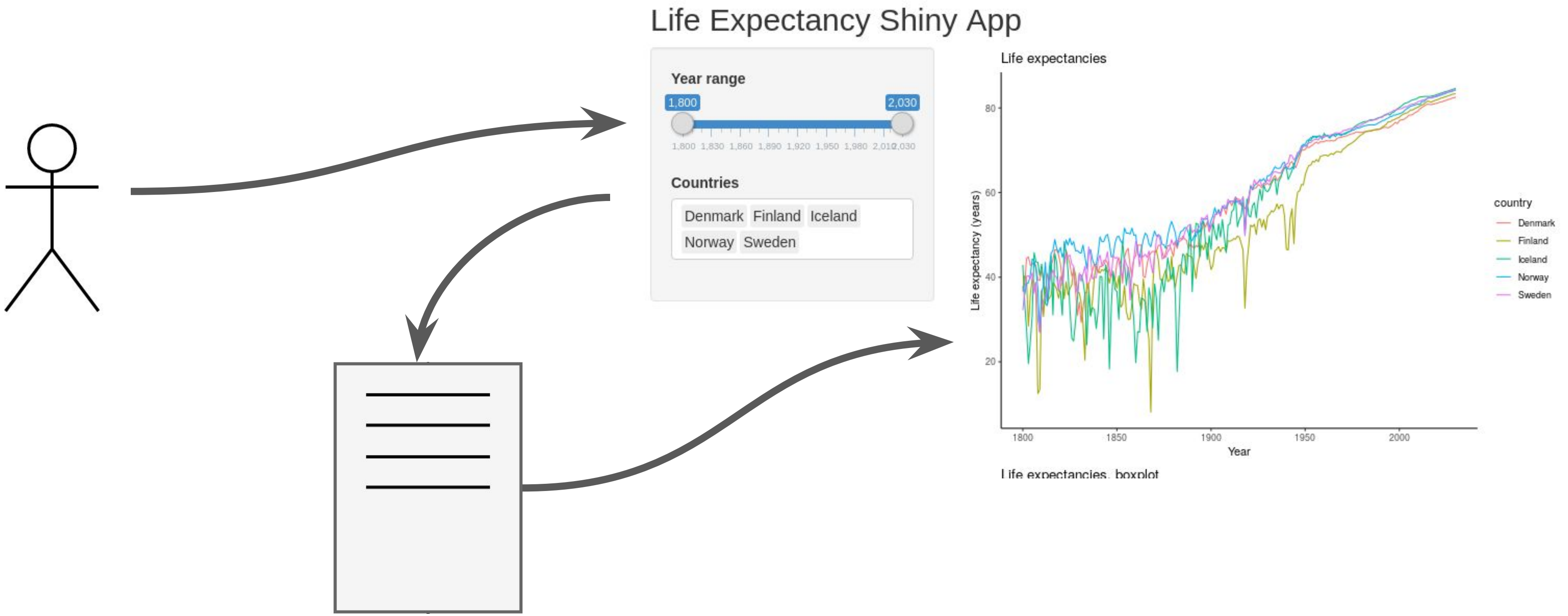


UI: What and where inputs and outputs are in the web page

Server: Generate outputs based on inputs

Reactive programming

1. User change setting


Life Expectancy Shiny App

Year range

1,800 — 2,030

Countries

Denmark Finland Iceland Norway Sweden

2. Code linked to that input is reexecuted

3. Figures linked to that code is updated

In **reactive programming** (and {shiny}), code is executed "**on demand**"

↕

In contrast to "common" programming (also called **imperative programming**) where things are executed **in order**

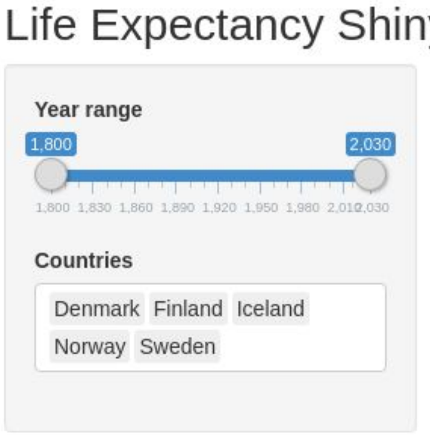
*Simplified!

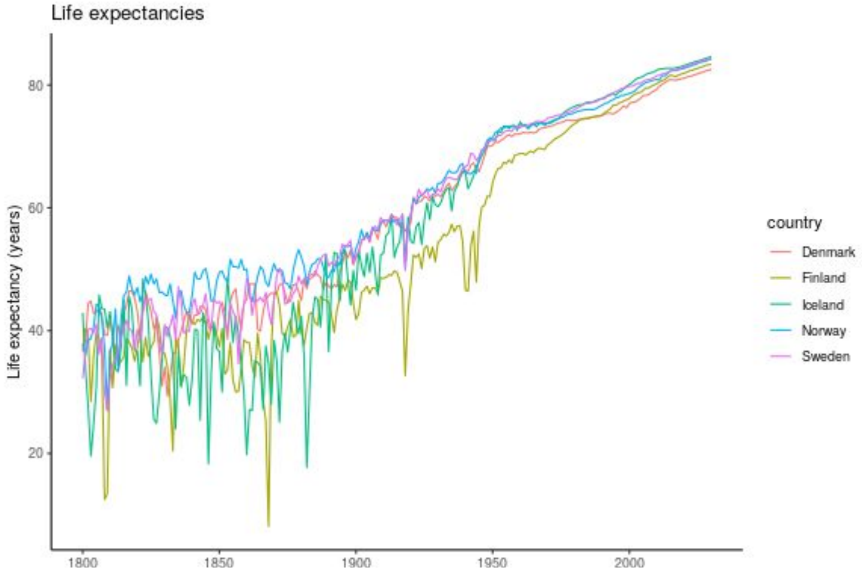
Illustrating reactive programming

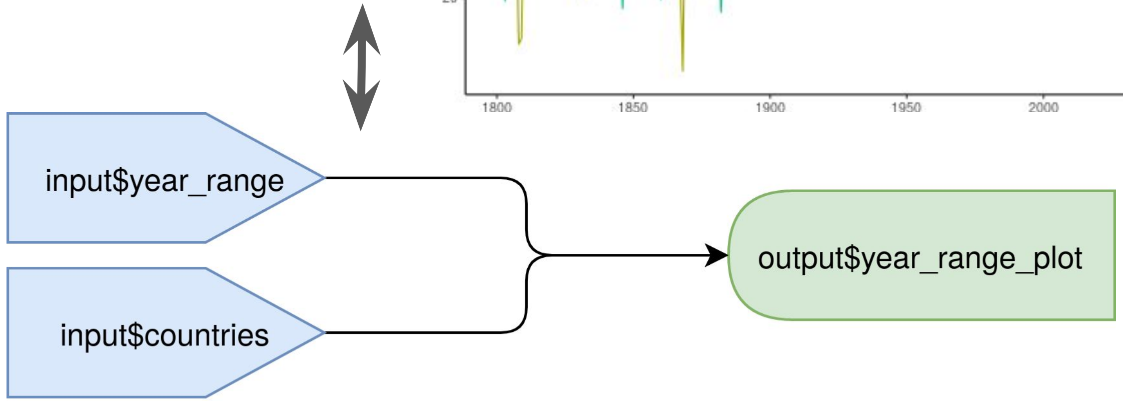
input$year_range

input$countries

output$year_range_plot







Today's workshop

- Gradual introduction to core aspects of {shiny}
- Example code and exercises
- At the end you will have built a fully functioning web app

Hands-on materials
https://jakob37.github.io/ShinyFromScratch

And please - feel free to ask questions! Any question you have likely many others have too.

Building a **{shiny}** app from scratch

A very minimal app

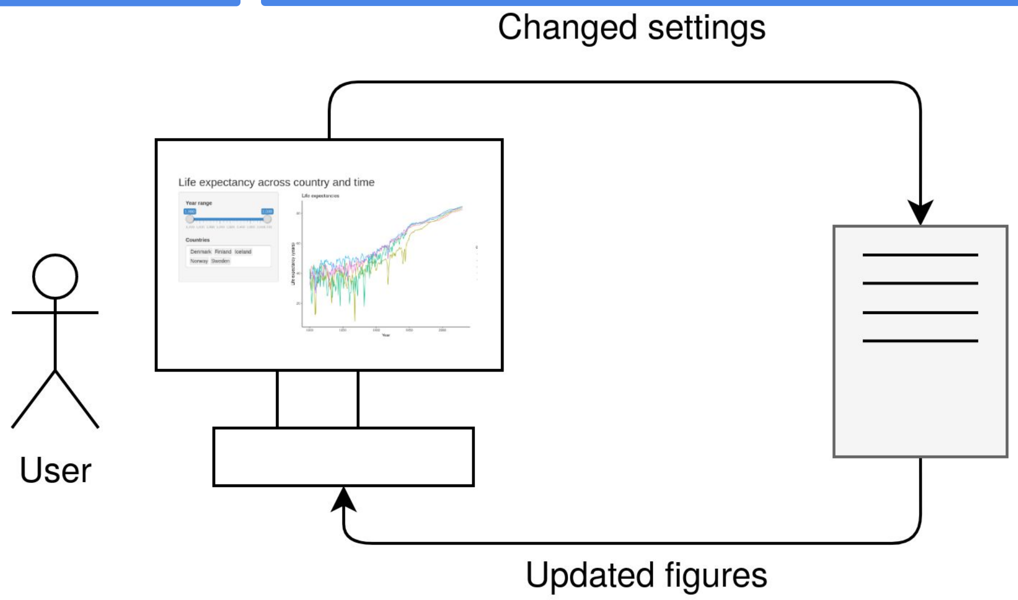
```
ui <- fluidPage()

server <- function(input, output) {}

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

UI

```
ui <- fluidPage()
```

Server

```
server <- function(input, output) {}
```



Create the app!

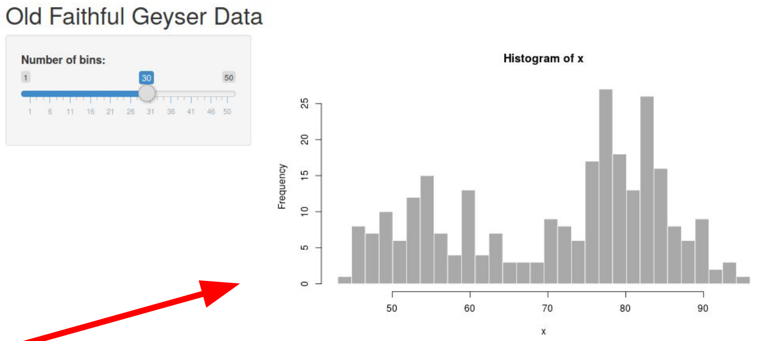
```
shinyApp(ui=ui, server=server)
```

The UI

!

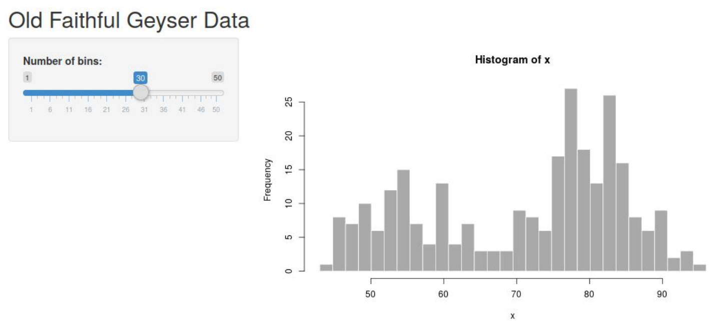
Take note of the ids "bins" and "distPlot", they will be used in the server side

```
ui <- fluidPage(

     titlePanel("Old Faithful Geyser Data"),

     sidebarLayout(
          sidebarPanel(
               sliderInput("bins",
                              "Number of bins:",
                              min = 1,
                              max = 50,
                              value = 30)
          ),

          mainPanel(
               plotOutput("distPlot")
          )
     )
)
```



The server

```
server <- function(input, output) {

  output$distPlot <- renderPlot({
    x <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
}
```



The code generating this figure

Hands-on time

Up until "Preparing a dataset"
(~row 104)

Don't hestitate to ask questions!

The dataset

GAPMINDER

- Life expectancy values in countries 1800-2030
- Pre-parsed from wide- to long- format to fit {ggplot}
- Feel free to use your own data!

Wide format

country	1950	2000
Sweden	65	80
Denmark	66	82

Long format

country	year	life
Sweden	1950	65
Sweden	2000	80
Denmark	1950	66
Denmark	2000	82

https://www.gapminder.org/data

Two slides on ggplot2 and dplyr

{ggplot2} Popular and flexible package for **building visuals** in R

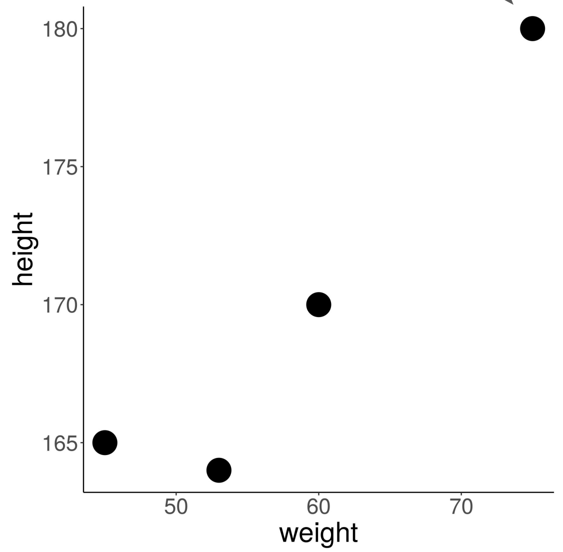
{dplyr} Intuitive data slicing functions (here we mostly use **filter**)

The **ggplot** command

```
ggplot(df, aes(x=height,y=weight)) + geom_point()
```

df

height	weight
180	75
165	45
170	60
164	53



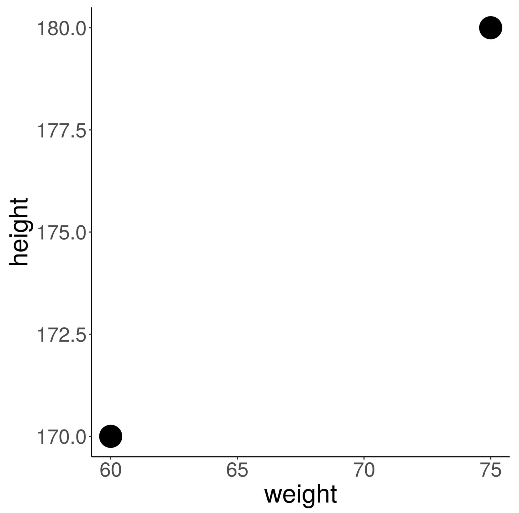
The **filter** command

```
df %>% filter(height >= 170)
```

df

height	weight
180	75
165	45
170	60
164	53

height	weight
180	75
170	60



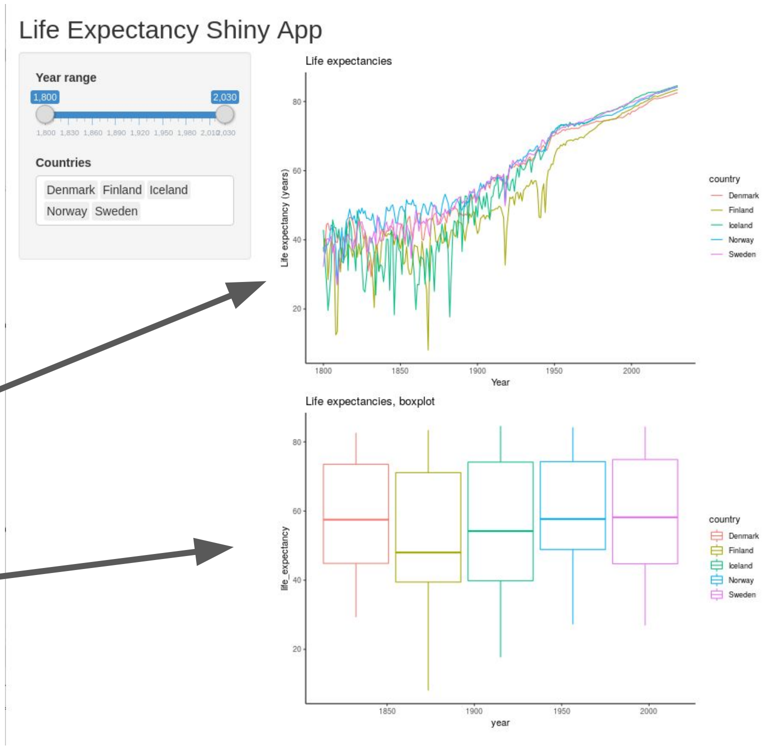
Hands-on time

Up until "Showing the data from multiple angles" (~row 418)

Don't hestitate to ask questions!

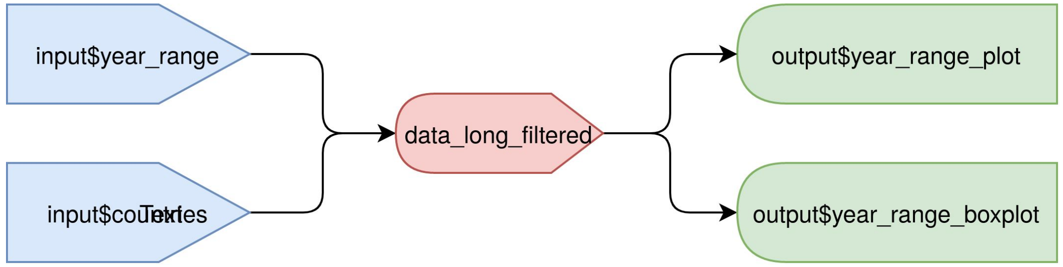
The **reactive** statement

```
server <- function(input, output) {

      data_long_filtered <- reactive({
            data_long %>%
                  filter(country %in% input$countries) %>%
                  filter(year >= input$year_range[1] &
                         year <= input$year_range[2])
      })

      output$year_range_plot <- renderPlot({
            ggplot(data_long_filtered(), … ) …
      })

      output$year_range_boxplot <- renderPlot({
            ggplot(data_long_filtered(), … ) …
      })

}
```


Life Expectancy Shiny App

Year range: 1,800 – 2,030

Countries: Denmark, Finland, Iceland, Norway, Sweden



(Final) Hands-on time

Don't hestitate to ask questions!

Thank you for participating!

For more R meetups, join the Skåne R User Group at:

https://www.meetup.com/Skane-R-User-Group

You find Jakob on:

GitHub: github.com/Jakob37
LinkedIn: linkedin.com/in/jakobwillforss