INVENTING FOR LIFE

Web Application Development in R (Shiny) vs Python (Dash)

Danfeng Fu, MSD

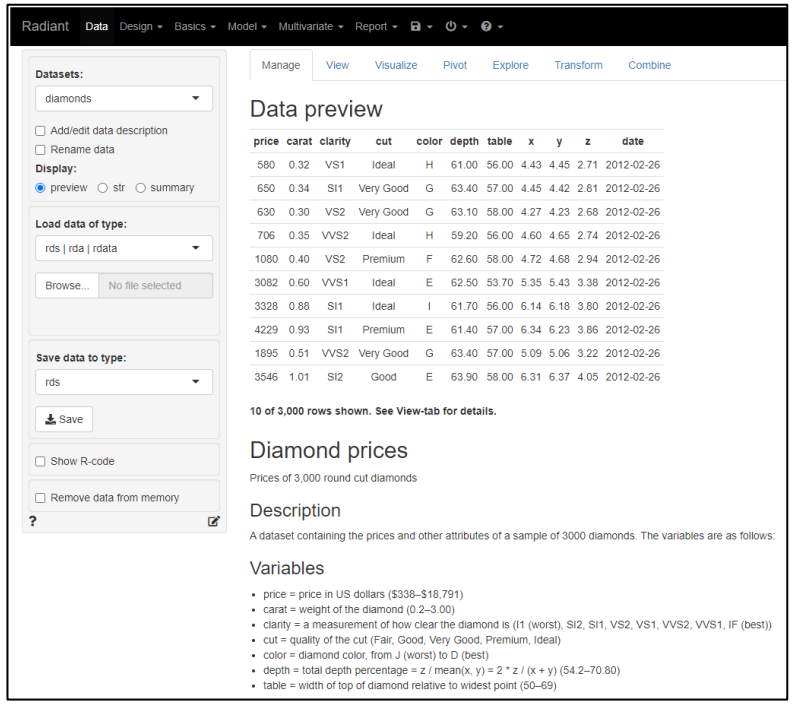
Oct, 2022

Contents

Overview

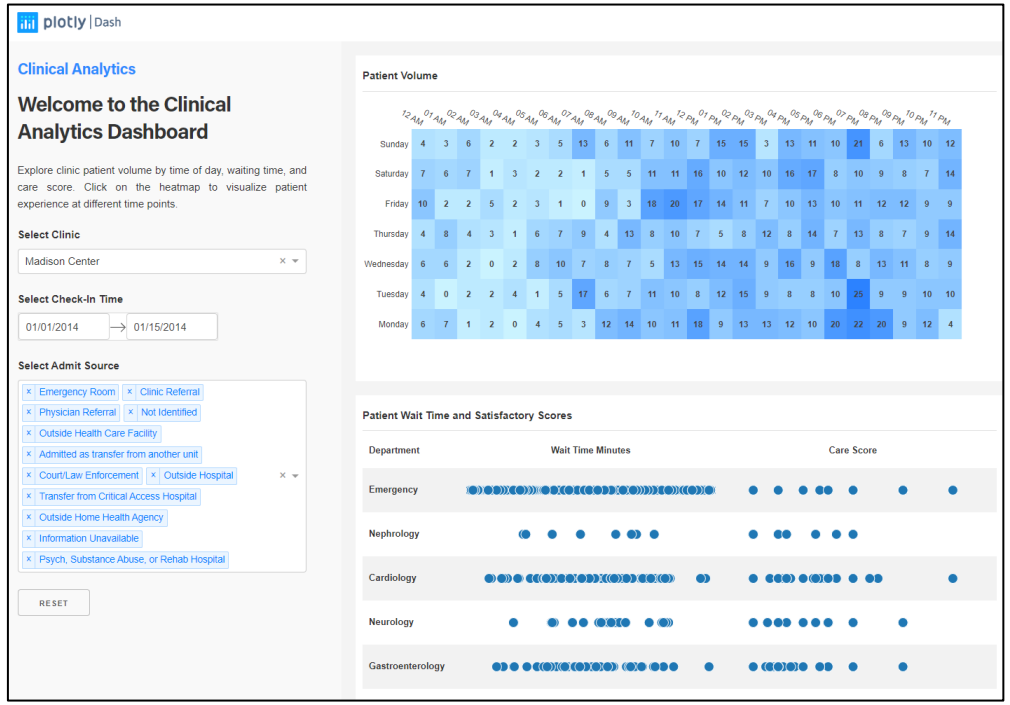
R Shiny App

Shiny application developed in R ecosystem



Python Dash App

Dash application developed in Python ecosystem



Similarities and Differences in Development

Which to choose?

Benefits of Web Application

- Fast deployments
- Easy access (anywhere from a web browser)
- More convenience
  - Combined utilities (portal of multiple small utilities)
- Lower development costs

Develop tools for

- **Data analysis**
- **Data exploration**
- **Visualization**
- **Modelling**
- **Instrument control**
- **Reporting**

Ref: https://budibase.com/blog/web-application-development/

Web Application Development Flow

R Shiny

Python Dash

Contents

App Content Development Key Aspects

UI – Components

Action Button

Number of bins:

1 30 50

1 6 11 16 21 26 31 36 41 46 50

Select input

Cylinders

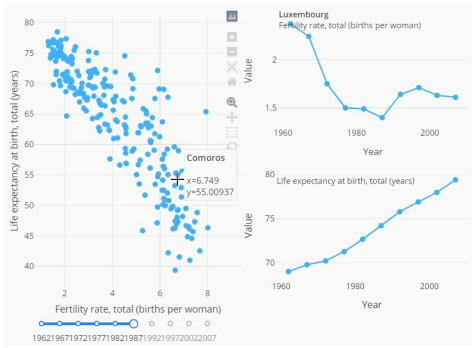
Numeric input

0.632

Server – UI Reactivity

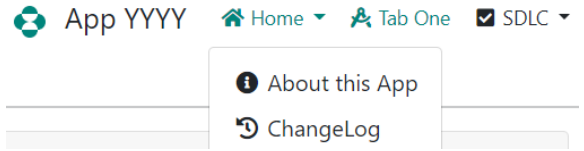
5

x^2 25



UI – Layout, Theme & Style

Layout & Theme



Style

SPEC Folder Path

SPEC Path

Server – User Defined Functions

```python
def function1(input1):
    output1 = "This is a function in Python"
    return output1
```

```r
function1 <- function(input1) {
  output1 <- "This is a function in R"
}
```

Basic Structure for app.R & app.py

R Shiny - app.R

```
library(shiny)

ui <- fluidPage( "Hello, world!")

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

shinyApp(ui, server)
```

User Defined Functions

Python Dash - app.py

```
from dash import Dash, html

app = Dash(__name__)

server = app.server

app.layout = html.Div([html.H2("Hello, world!")])

app.run_server(debug=True)
```

```
output$greeting <- renderText({

paste0("Hello ", input$name, "!") })
```

Reactive Events

```
@app.callback(
    Output(component_id='my-output', component_property='children'),
    Input(component_id='my-input', component_property='value')
)
def update_output_div(input_value):
    return f'Hello, {input_value}!'
```

Contents

UI - Components Available

UI Layout

absolutePanel() fixedPanel()
bootstrapPage() basicPage()
column()
conditionalPanel()
fillPage()
fillRow() fillCol()
fixedPage() fixedRow()
fluidPage() fluidRow()
helpText()
icon()
navbarPage() navbarMenu()
navlistPanel()
sidebarLayout() sidebarPanel()
mainPanel()
tabPanel() tabPanelBody()
tabsetPanel()
titlePanel()
inputPanel()
flowLayout()
splitLayout()
verticalLayout()
wellPanel()
withMathJax()

UI Inputs

actionButton() actionLink()
checkboxGroupInput()
checkboxInput()
dateInput()
dateRangeInput()
fileInput()
numericInput()
radioButtons()
selectInput() selectizeInput()
varSelectInput() varSelectizeInput()
sliderInput() animationOptions()
submitButton()
textInput()
textAreaInput()
passwordInput()
updateActionButton() updateActionLink()
updateCheckboxGroupInput()
updateCheckboxInput()
updateDateInput()
updateDateRangeInput()
updateNumericInput()
updateRadioButtons()
updateSelectInput() updateSelectizeInput() updateVarSelectInput() updateVarSelectizeInput()
updateSliderInput()
updateTabsetPanel() updateNavbarPage() updateNavlistPanel()
insertTab() prependTab() appendTab() removeTab()
showTab() hideTab()
updateTextInput()
updateTextAreaInput()
updateQueryString()
getQueryString() getUrlHash()

UI Outputs

htmlOutput() uiOutput()
imageOutput() plotOutput()
dataTableOutput()
outputOptions()
textOutput() verbatimTextOutput()
downloadButton() downloadLink()
Progress
withProgress() setProgress() incProgress()
modalDialog() modalButton()
urlModal()
showModal() removeModal()
showNotification() removeNotification()

html components

tags p() h1() h2() h3() h4() h5() h6() a() br() div() span() pre() code() img() strong() em() hr() tag()
HTML()
includeHTML() includeText() includeMarkdown()
includeCSS() includeScript()
singleton() is.singleton()
tagList()
tagAppendAttributes() tagHasAttribute() tagGetAttribute()
tagAppendChild() tagAppendChildren() tagSetChildren() tagInsertChildren()
validateCssUnit()
withTags()
htmlTemplate()
bootstrapLib()
suppressDependencies()
insertUI() removeUI()
markdown()

R Shiny

Open Source Component Libraries

Checklist
Clipboard
ConfirmDialog
ConfirmDialogProvider
DatePickerRange
DatePickerSingle
Download
Dropdown
Graph
Input
Interval
Link
Loading
Location
LogoutButton
Markdown
RadioItems
RangeSlider
Slider
Store
Tab
Tabs
Textarea
Tooltip
Upload

Dash Core Components
Dash HTML Components
Dash DataTable
Dash Bio
Dash DAQ
Dash Image Annotations
Dash Canvas
Dash Slicer
Dash Cytoscape
Dash VTK
Dash Bootstrap Components
Dash Community Components

Python Dash

Enterprise Component Libraries
Creating Your Own Components

UI - Components Available

UI Inputs

actionButton() actionLink()
checkboxGroupInput()
checkboxInput()
dateInput()
dateRangeInput()
fileInput()
numericInput()
radioButtons()
selectInput() selectizeInput()
varSelectInput() varSelectizeInput()
sliderInput() animationOptions()
submitButton()
textInput()
textAreaInput()
passwordInput()

UI Outputs

htmlOutput() uiOutput()
imageOutput() plotOutput()
dataTableOutput()
outputOptions()
textOutput() verbatimTextOutput()
downloadButton() downloadLink()
Progress
withProgress() setProgress() incProgress()
modalDialog() modalButton()
urlModal()
showModal() removeModal()
showNotification() removeNotification()

html components

Removed Items are
- **Layout related**
- **Reactivity related**

R Shiny

Open Source Component Libraries

Checklist
Clipboard
ConfirmDialog
ConfirmDialogProvider
DatePickerRange
DatePickerSingle
Download
Dropdown
Graph
Input
Interval
Link
Loading
Location
LogoutButton
Markdown
RadioItems
RangeSlider
Slider
Store
Tab
Tabs
Textarea
Tooltip
Upload

Enterprise Component Libraries

Dash Core Components
Dash HTML Components
Dash DataTable
Dash Bio
Dash DAQ
Dash Image Annotations
Dash Canvas
Dash Slicer
Dash Cytoscape
Dash VTK
Dash Bootstrap Components
Dash Community Components

Python Dash

Creating Your Own Components – more flexibility

Contents

Reactivity Realization

R Shiny

reactive() is.reactive()
observe()
observeEvent() eventReactive()
reactiveVal()
reactiveValues()
bindCache()
bindEvent()
reactiveValuesToList()
is.reactivevalues()
isolate()
invalidateLater()
debounce() throttle()
reactlog() reactlogShow() reactlogReset()
reactiveFileReader()
reactivePoll()
reactiveTimer()
getDefaultReactiveDomain() withReactiveDomain() onReactiveDomainEnded()
freezeReactiveVal() freezeReactiveValue()

renderPlot()

renderCachedPlot()

renderPrint() renderText()

dataTableOutput() renderDataTable()

renderImage()

renderTable()

renderUI()

downloadHandler()

createRenderFunction() quoToFunction() installExprFunction()

Pros: Built-in Reactive Functions
Cons: Difficult to properly use

Python Dash

One output One callback

```
@app.callback(
    Output( id=, property=),
    Input1=(id=, property=),
    Input2=(id=, property=),
)
def function1(input1,input2):
    Some events
    return output
```

Pros: Easy to control each reactivity
Cons: No Built-in Reactive Functions

Render Reactive Outputs

1. R Shiny

- Paired UI- Reactive Function

UI	Reactive Function
textOutput()	renderText() or renderPrint()
checkboxInput()	updateCheckboxInput()
dataTableOutput()	renderDataTable()

```
ui = fluidPage( fluidRow( column(12,
       dataTableOutput('table')
) ) ),

server = function(input, output) {
    output$table <- renderDataTable(iris)
}
```

2. Python Dash

- User specified callback details

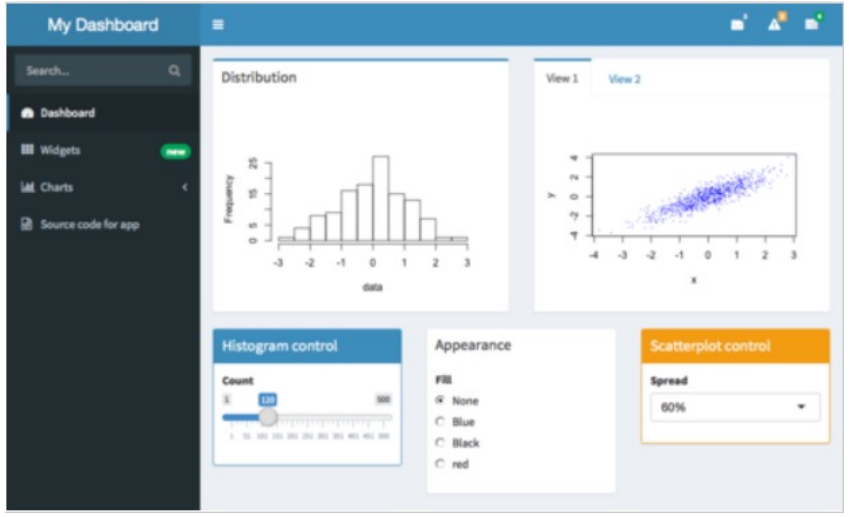
```
@app.callback(
    Output('out_datatable', 'data'),
    Output('out_datatable', 'columns'),
    Input('input1', 'data'),
    Input('input2',"value"),
)
def update_datatable(data,value):
    #iris is a Pandas Dataframe
    outdata = iris.to_dict("records")
    outcol = [{"name": i,"id": i,} for i in iris.columns]
    return outdata, outcol
```

Information in red box need to be taken care of by users themselves in Python Dash. These are additional steps than R Shiny because lack of built-in functions.

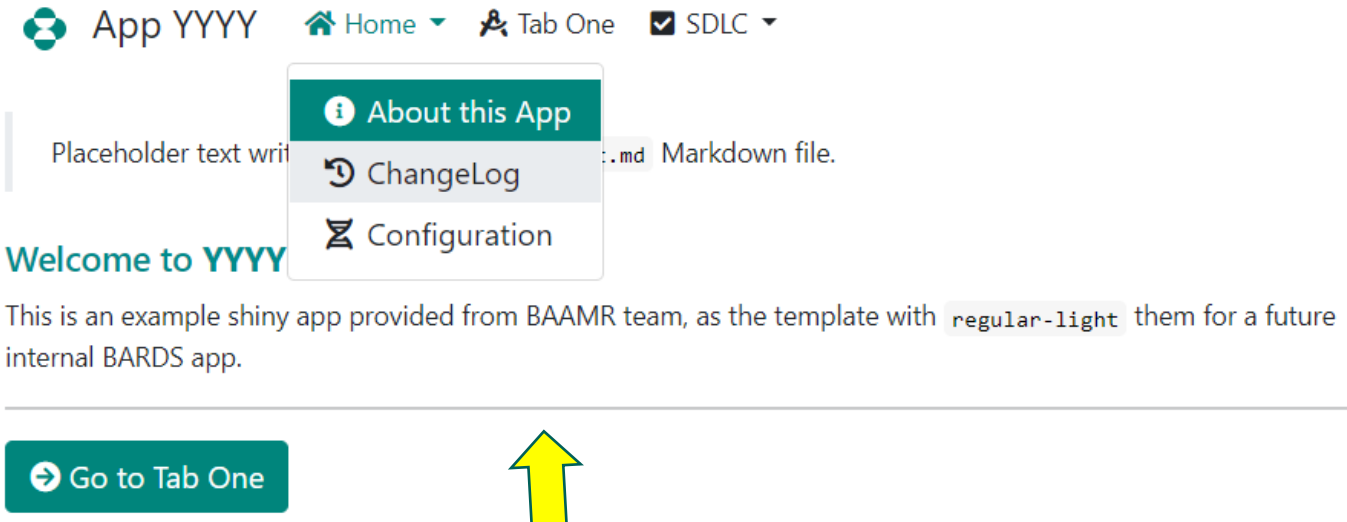
Contents

Layout – Application Structure

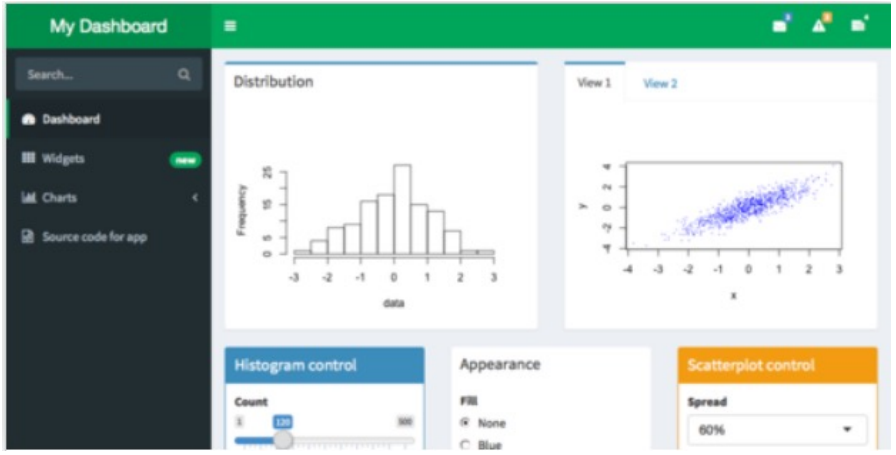
Sidebar Layout



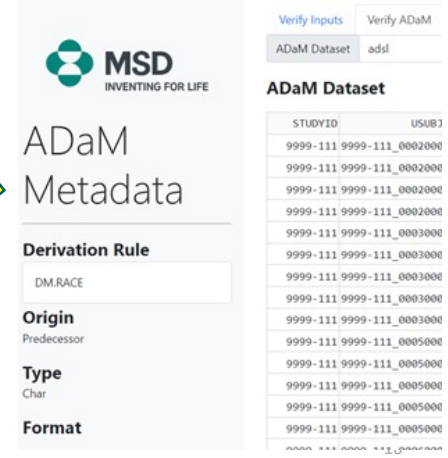
TabPanel Layout



Different Theme



Company Customized Theme



Layout Design

- R Shiny

1. Shiny
   - sidebarLayout
     - sidebarPanel
2. **shinydashboard**

```
ui <- dashboardPage(
  dashboardHeader(title = "Basic dashboard"),
  dashboardSidebar(),
  dashboardBody(
```

3. **shinydashboard**Plus

Built-in Layout Components

Features (sample)	sdb	sdbPlus
right sidebar (controlbar)	❌	✅
semi collapsible sidebar (sidebar mini)	❌	✅
expand on hover sidebar	❌	✅
closable boxes	❌	✅
box sidebar	❌	✅
get box state on the server (open, closed, …)	❌	✅
control sidebars on the server	❌	✅
dashboard user dropdown	❌	✅
theme selector	❌	✅
social box	❌	✅
user box	❌	✅
control AdminLTE options	❌	✅
seamlessly customize appearance	❌	✅
beautiful preloaders	❌	✅
scroll to top button!	❌	✅

- Python Dash

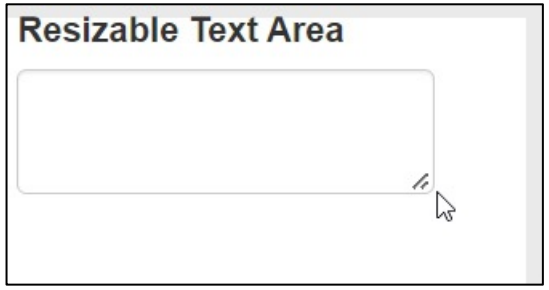
```
tabs = html.Div(
    dbc.Tabs([
        dbc.Tab(tab1_content, label="tab1"),
        dbc.Tab(tab2_content, label="tab2"),
        dbc.Tab(tab3_content, label="tab3"),
    ]),
    style=CONTENT_STYLE)

app.layout = html.Div([sidebar, tabs])
```

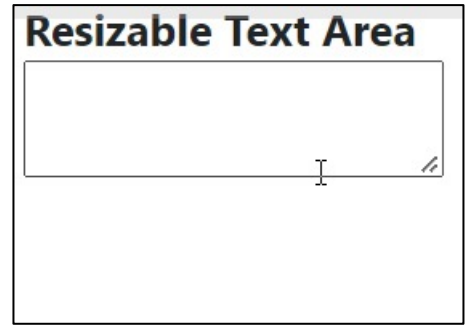
Self Customized Layout & Style

```
SIDEBAR_STYLE = {
    "position": "fixed",
    "top": 0,
    "left": 0,
    "bottom": 0,
    "width": "18rem",
    "padding": "2rem 1rem",
    "background-color": "#f8f9fa",
    "overflow-y":"auto"
}
```

Style - Customize UI Component



Max-height = 400px ??



- R Shiny

```
textAreaInput(
    InputId="textarea-id",
    label,
    value = "",
    width = NULL,
    height = NULL,
    cols = NULL,
    rows = NULL,
    placeholder =NULL,
    resize = NULL)
```

No control of max-height

Include a CSS file

```
textarea-id
{
    width: auto;
    max-height: 400px;
}
```

- Python Dash

```
dcc.Textarea(
    id="textarea-id",
    style={
            "width":"100%",
            "max-height":“400px"}
),
```

Component setting allows definition of styles directly

Contents

Server - User Defined Functions

Whether user defined functions satisfy the needs in the application?

1. **Strength in R**

   ➢ mainly used for statistical analysis

   - Available statistical analysis packages
   - Data visualization
   - Large dataset handling

```r
function1 <- function(input1) {
  output1 <- "This is a function in R"
}
```

2. **Strength in Python**

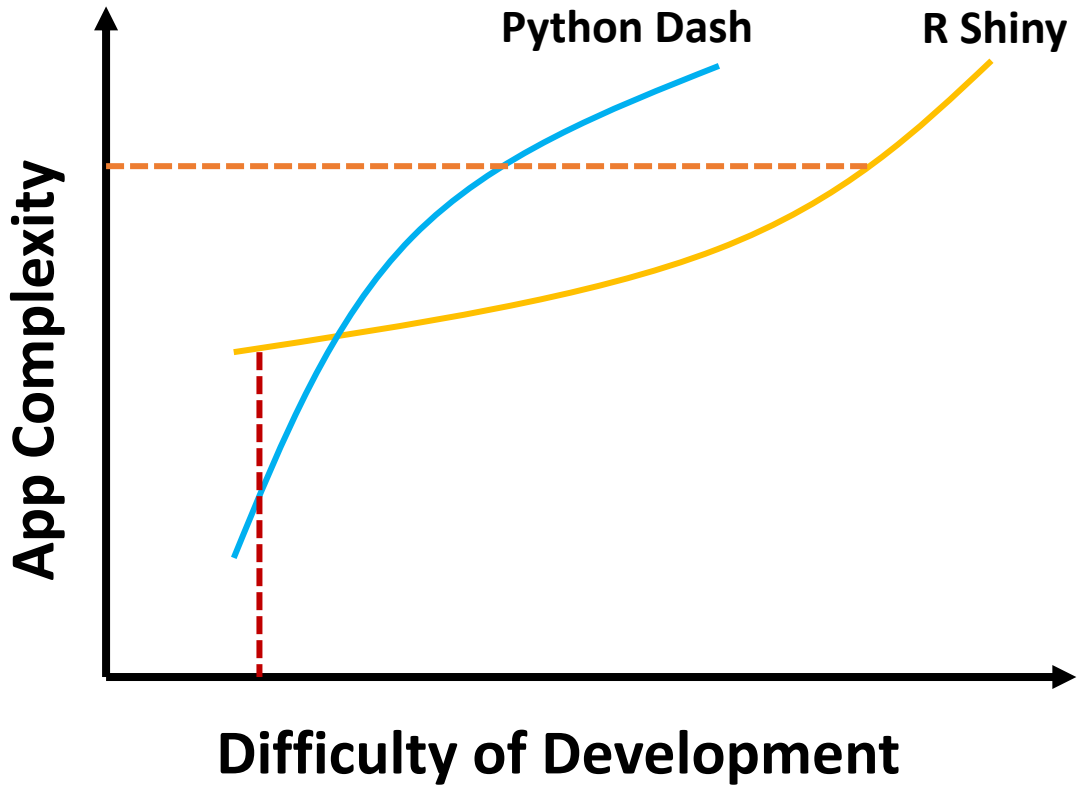
   ➢ more general approach to data wrangling

   - NLP
   - Data mining
   - Web scraping
   - Machine learning

```python
def function1(input1):
    output1 = "This is a function in Python"
    return output1
```

Contents

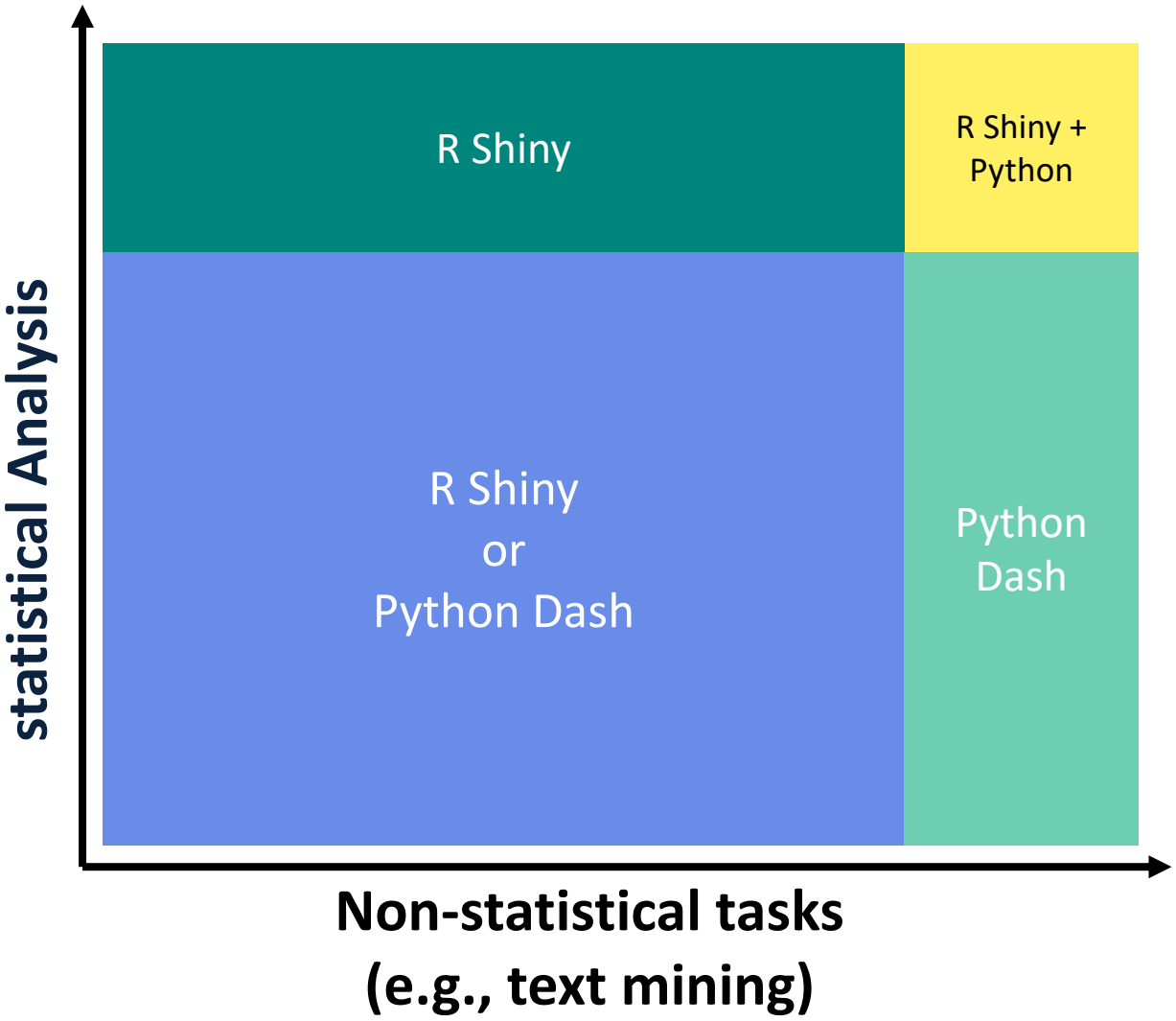
Summary - Difficulty vs. Complexity



Recommendation –

- **It is relatively easy for beginners to use R Shiny even to develop complex web applications**
  - **Included pre-defined modules but are difficult to customize**
- **For an expert, it's easier to use Python Dash to develop more complex web applications than R Shiny**
  - **Lack of pre-defined modules but allows flexibility to customize**

Discussion - Needs vs. Capability



1. More statistical analysis needs
   R Shiny
2. More Non-statistical tasks
   Python Dash
3. Others
   - Fewer statistical analysis needs
     R Shiny or Python Dash
   - More statistical analysis needs + Functionality only available in Python
     R Shiny + Python (reticulate)

INVENTING FOR LIFE

Thank you