Web Applications for Cost Analysis Use: R Shiny vs Python Plotly Dash

Eric J. Hagee
John W. Maddrey

TECOLOTE
R E S E A R C H

Outline

- Introduction to Web Apps

- Project Description and Term Definition

- R Shiny vs Python Dash

  - Community Usage and Resources

  - File Structure

  - Layout, Styling, and Graphics

  - Reactivity

  - Enterprise Deployment

- Conclusions
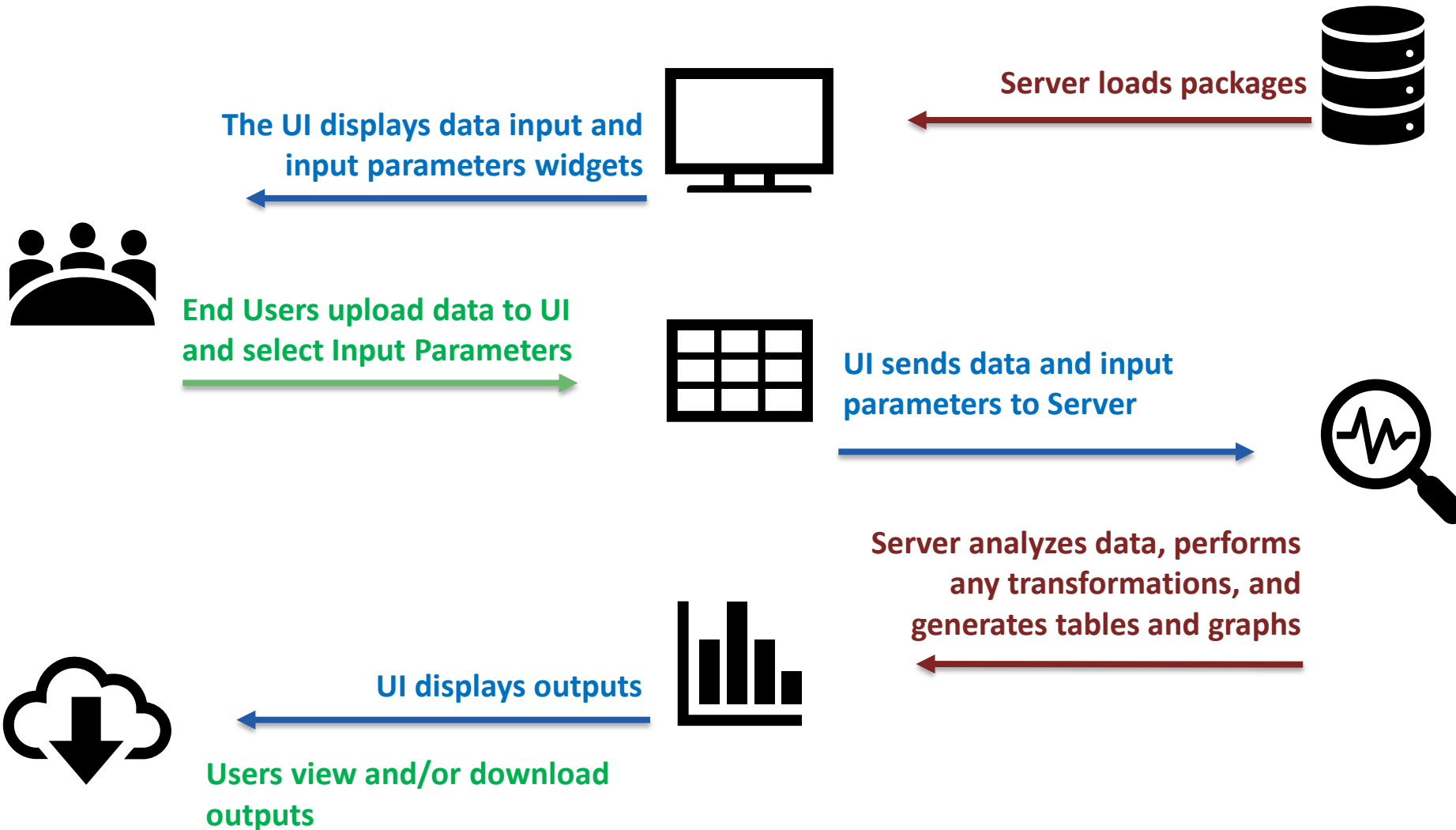
What are Web Apps and Why do we Care?

- The availability of data and increase in computer power has created interest in collation of data, often in "real-time"

- Web browsers use modern programming languages to render information in html format like web pages

    - Transforms code into html, rendered as a web page

    - Can present data and information with custom visualizations and annotation

    - Use of programming languages can deftly handle data and produce sophisticated graphics

    - **Web Applications can be viewed by multiple users in a web browser without any other applications**

General Architecture

End Users   UI   Server

Server loads packages

The UI displays data input and input parameters widgets

End Users upload data to UI and select Input Parameters

UI sends data and input parameters to Server

Server analyzes data, performs any transformations, and generates tables and graphs

UI displays outputs

Users view and/or download outputs

Project Description

- The programming languages R and Python are the two main open source data environments

- Each has packages that enable creation of web apps
  - R – Shiny
  - Python – Plotly Dash

- Created an inflation application in each package to illustrate comparisons between languages and packages
  - The intent was to demonstrate basic capabilities, so app is simple by design
  - Public inflation data is readily available and relevant to wider discussions and specifically to cost analysis
  - The application has several pages that discuss inflation definitions and inflation measures and graphs of historical data, as well as an inflation calculator and relevant reference links
  - The code utilizes the same basic format and calculations, but differs in syntax and any package-specific organization

Inflation Measures Review

Inflation is the increase of prices over time

Government bodies use inflation measures to track inflation

These are typically a basket of goods (a group of items, often typically used by a household)

The Consumer Price Index (CPI) from the BLS is used to adjust tax brackets and social security payments

The Personal Consumption Expenditures (PCE) index is used by the Fed to adjust interest rates

Core inflation does not include highly variable food and fuel prices, and is useful to track alongside total inflation

Comparisons – Community and General Considerations
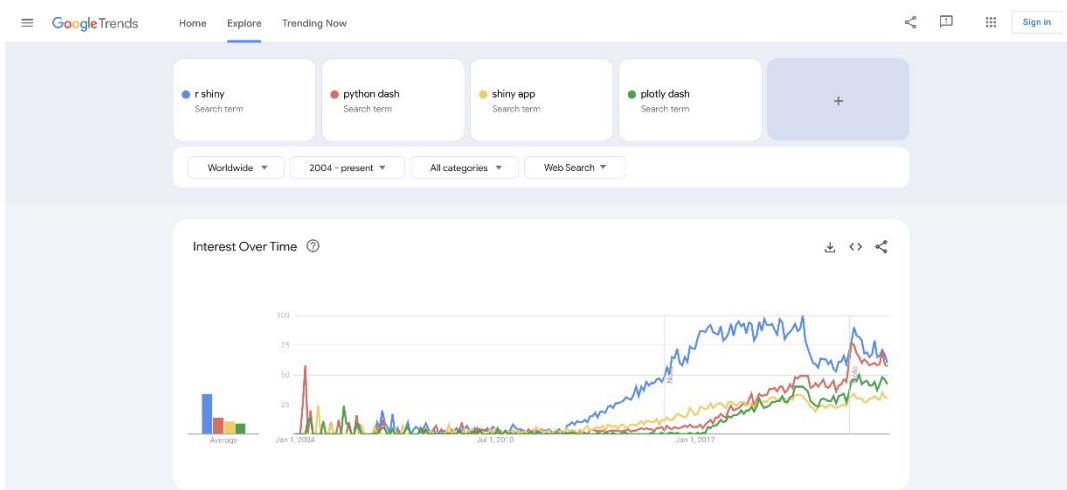
Worldwide Search Trends
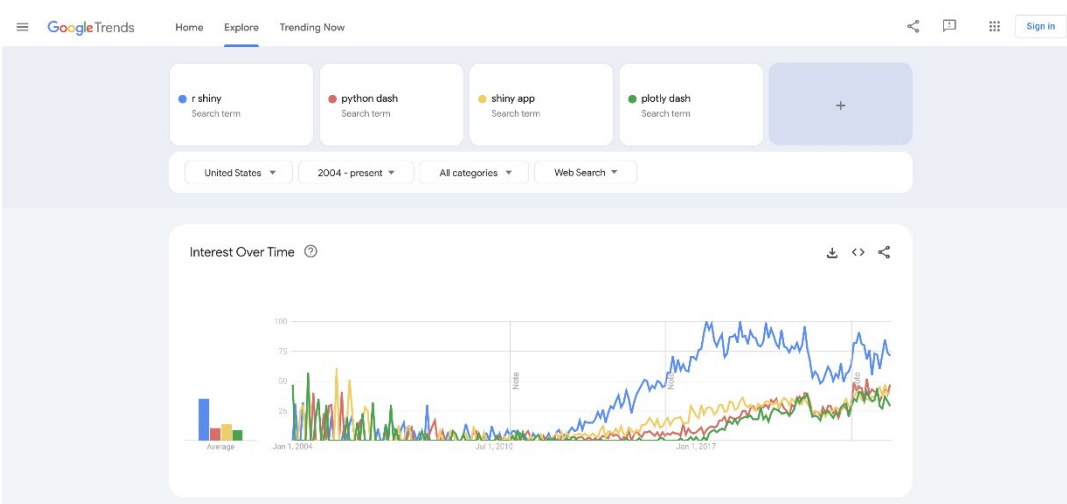
United States Search Trends

R Shiny has more historical usage, though Python Dash is catching up
(More worldwide than in the United States)

Comparisons – Community and General Considerations

- **Community:**
  - R Shiny was founded the same year as Plotly, out of an existing effort
  - R Shiny has more historical usage and is thus more established
  - More community resources on for example, Stack Exchange
    - Questions 6 to 1 on Stack Exchange
    - Numbers are more even on recent questions

- **Documentation:**
  - Good Documentation on Both
  - R Shiny has better introduction and lots of articles
  - Python Dash has more intermediate specific examples that and more organized function documentation

- **Language Usage:**
  - Python is more widely used than R (5 to 1 questions on Stack Exchange)
  - Python is more of a programming language than R, but R is made for statistical modeling

- **The Twist!: Both Shiny and Dash are developing a corresponding Python and/or R version**

Comparisons – File Structure

Python Dash File Structure
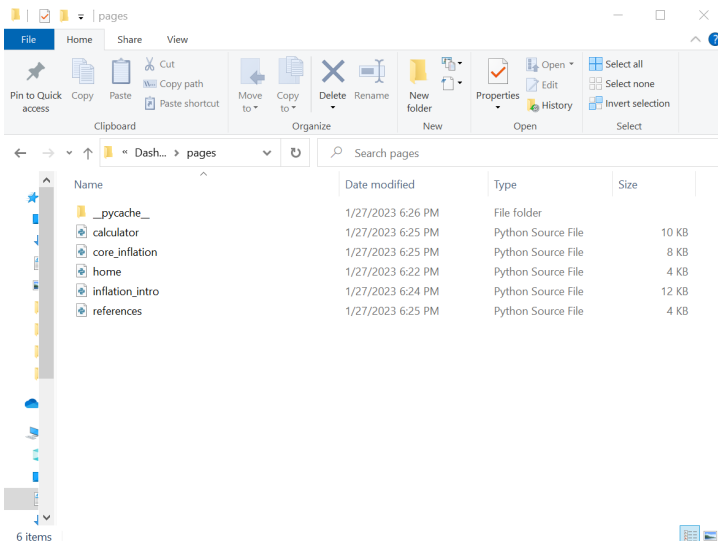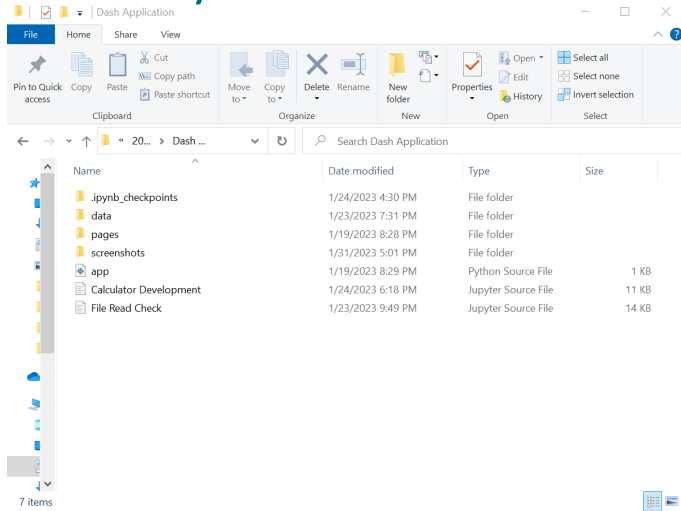
R Shiny File Structure
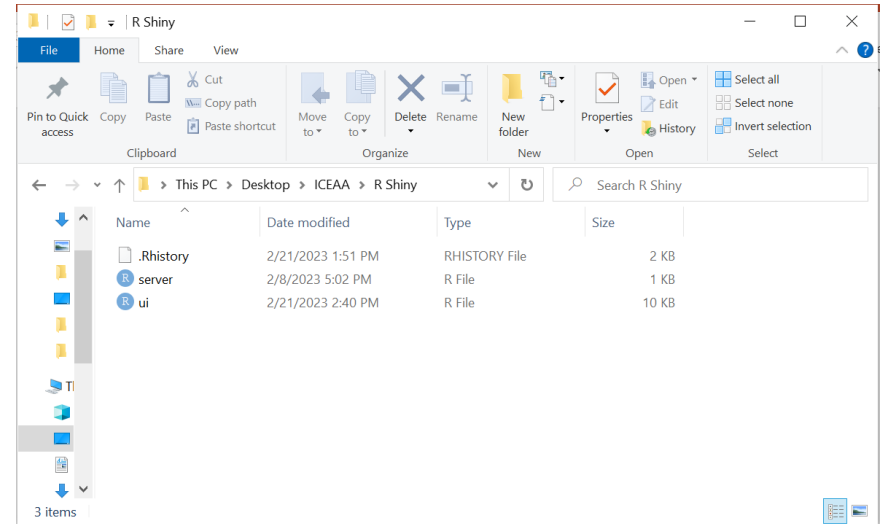
Python Dash -- One main file (app.py) with each tab its own executable file called by app.py

R Shiny – Two files, one for the user interface (ui.R) and one for the server and reactivity (server.R)

Comparisons – File Structure and Implementation

Python Dash	R Shiny
App file (can be named anything) handles interaction	ui.R and server.R interact to create a reactive environment
Call app file to run	Run app in directory with ui.R and server.R
One python file for each page	Tab layout all in ui.R
Calculations and data for each page in appropriate file	Calculations/data all in server.R
Reactivity available by callbacks using original functions	Reactivity attached to elements by wrapping output in specific pre-made functions

Page Files

Python Dash Implementation

app.py

User Display

ui.R

server.R

R Shiny

User Display
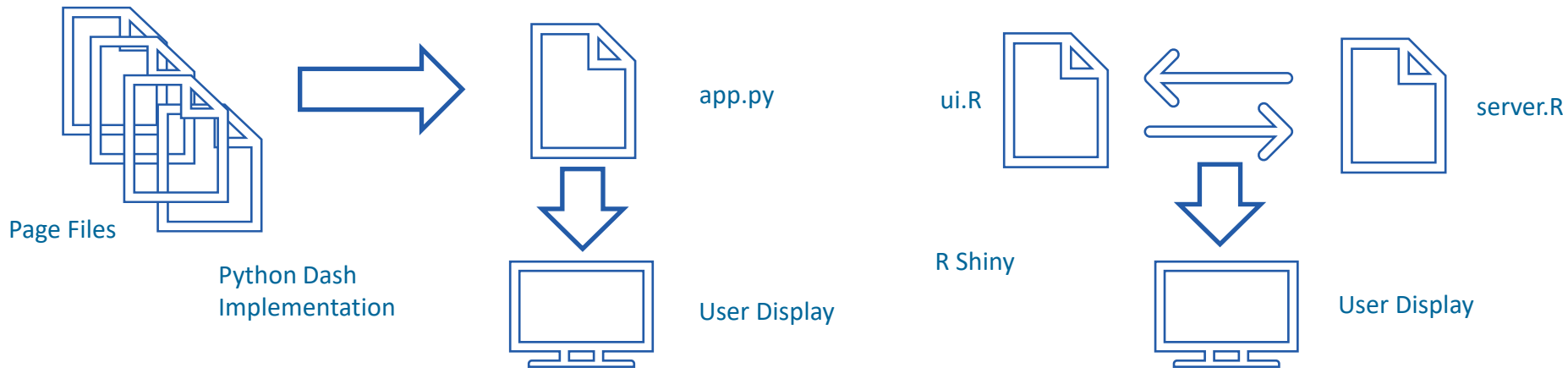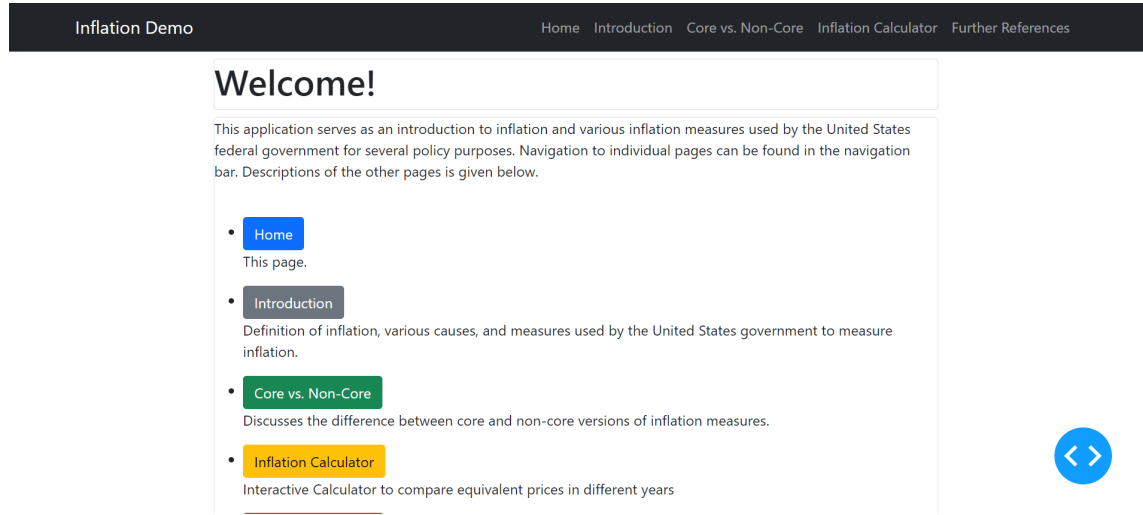
Home Page Implementation
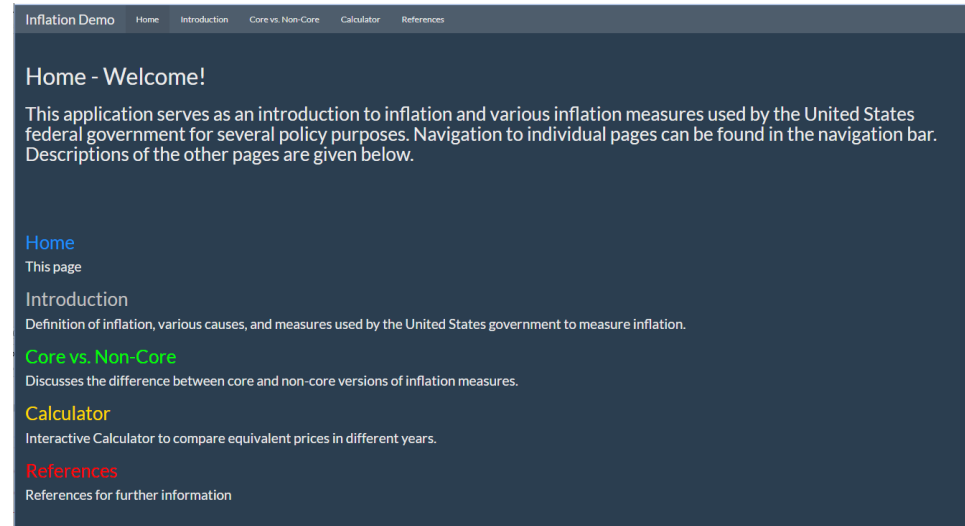
Python Dash

R Shiny

The home page in both apps serves as a link to both content pages. Python Dash gives a more modern web page look, particularly when using the Dash Bootstrap package.

Home Page Implementation: Python Dash

App.py

Home.py

Home Page Implementation: R Shiny

ui.R

```
###########################
######### UI.R #########
###########################
library(shiny)
library(shinyBS)
library(shinythemes)
library(shinyWidgets)
library(shinydashboard)


navbarPage("Inflation Demo",
          theme = shinytheme("superhero"),
          tabPanel("Home",
                  h2("Home - Welcome!"),
                  h3("This application serves as an introduction to inflation and various inflation measures used by the United States federal government for several policy purposes.  Navigation to individual pages can
                  br(),
                  br(),
                  br(),
                  h3("Home", style = "color:dodgerblue"),
                  h4("This page"),
                  h3("Introduction", style = "color:silver"),
                  h4("Definition of inflation, various causes, and measures used by the United States government to measure inflation."),
                  h3("Core vs. Non-Core", style = "color:lime"),
                  h4("Discusses the difference between core and non-core versions of inflation measures."),
                  h3("Calculator", style = "color:gold"),
                  h4("Interactive Calculator to compare equivalent prices in different years."),
                  h3("References", style = "color:red"),
                  h4("References for further information")
                  ),
```
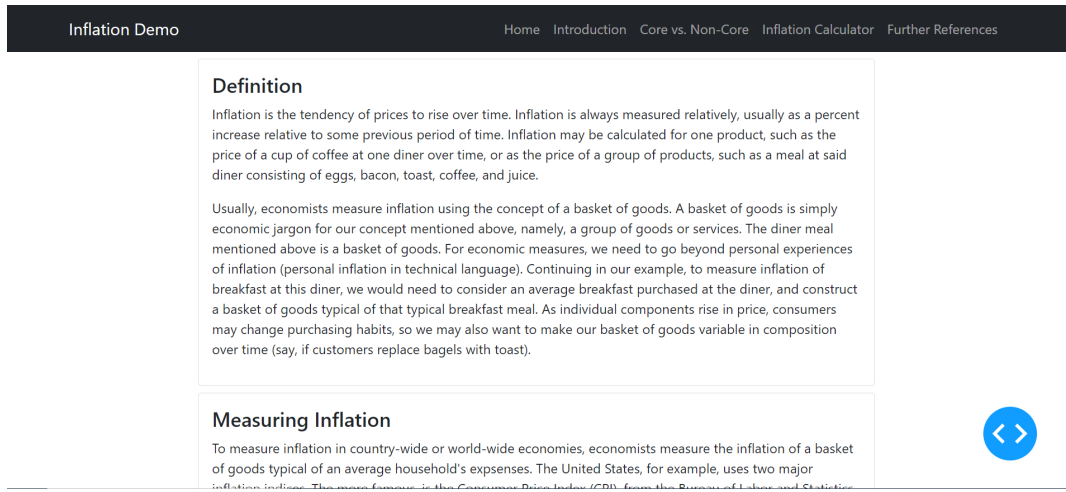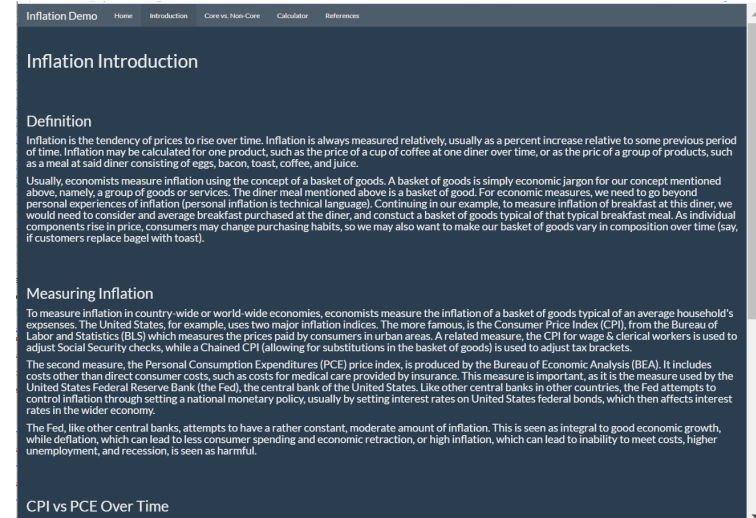
Capabilities: Basic Web Page Layout

Python Dash

R Shiny

The Introduction and Core vs. Non-Core pages provide information on inflation definitions and measures, demonstrating basic layout, formatting, and graphics.
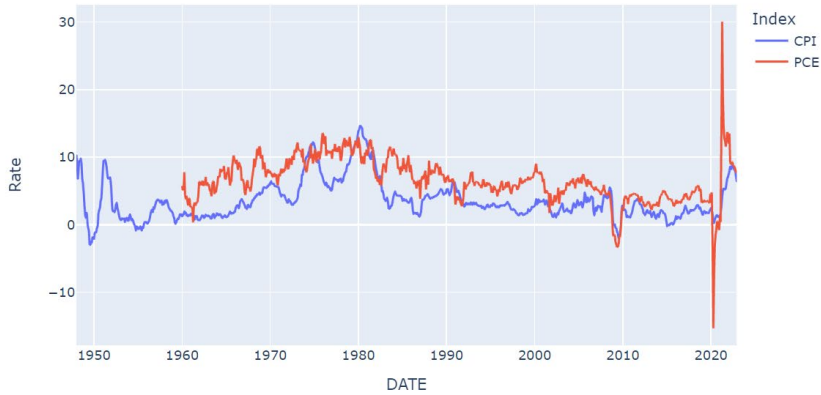
In Python Dash, the Dash Bootstrap package extends ease of formatting using a grid system.

Formatting, layout, and themes are more built-in in R Shiny. Themes in either case allow for consistent format of the page or pages.

Capabilities: Graphics

Python Dash

R Shiny

Capabilities: Graphics Implementation

Python Dash

```python
def cpi_pce():

    # read data
    cpi_df = pd.read_csv("./data/CPI.csv")
    pce_df = pd.read_csv("./data/PCE_monthly.csv")

    # reset column names
    cpi_df.columns = ['DATE', 'CPI']
    pce_df.columns = ['DATE', 'PCE']

    # change data type
    cpi_df['DATE'] = pd.to_datetime(cpi_df['DATE'])
    pce_df['DATE'] = pd.to_datetime(pce_df['DATE'])
```

```python
# Calculate Rates
cpi['CPI_Inflation_Rate'] = ((cpi['CPI'] - cpi['CPI_prev_yr'])/cpi['CPI_prev_yr'])*100
pce['PCE_Inflation_Rate'] = ((pce['PCE'] - pce['PCE_prev_yr'])/pce['PCE_prev_yr'])*100

# Merge Rates
inflation = cpi.merge(pce, on = 'DATE', how = 'left')

# Select rates only
inflation = inflation.loc[:, ['DATE', 'CPI_Inflation_Rate', 'PCE_Inflation_Rate']]
inflation.columns = ['DATE', 'CPI', 'PCE']

# reshape data for graphing
inflation = inflation.melt(id_vars = ['DATE'], value_name = 'Rate', var_name = 'Index')

# return plotly figure for graphing
fig = px.line(inflation, x = 'DATE', y = 'Rate', color = 'Index')

return fig
```

```python
card_graph = dbc.Card(
    dbc.CardBody(
        [
            html.H4("CPI vs PCE Over Time", className="card-title"),
            dcc.Graph(figure = cpi_pce()),
        ]
    ),
)
```

All code in inflation_intro.py. Layout code (bottom) calls function that returns a graph from the data (parts in top two pictures).

R Shiny

server.R

```r
cpi_csv = "CPI.csv"
pce_csv = "PCE_monthly.csv"
pCore_csv = "PCE_core_monthly.csv"
cCore_csv = "CPI_core_seasonally_adjusted.csv"

shinyServer(function(input, output) {
  #Read in CPI and PCE monthly data
  cpi_df <- read.csv(cpi_csv)
  pce_df <- read.csv(pce_csv)
  pCore_df <- read.csv(pCore_csv)
  cCore_df <- read.csv(cCore_csv)

  #Set Column names, convert date from character string to date format
  names(cpi_df) <- c("DATE","CPI")
  names(pce_df) <- c("DATE","PCE")
  names(pCore_df) <- c("DATE", "Pcore")
  names(cCore_df) <- c("DATE", "Ccore")
  cpi_df$DATE<-as.Date(cpi_df$DATE,tryFormats = c("%Y-%m-%d"))
  pce_df$DATE<-as.Date(pce_df$DATE,tryFormats = c("%Y-%m-%d"))
  pCore_df$DATE<-as.Date(pCore_df$DATE,tryFormats = c("%Y-%m-%d"))
  cCore_df$DATE<-as.Date(cCore_df$DATE,tryFormats = c("%Y-%m-%d"))

  #create plot and send to UI
  output$CPIvPCE <- renderPlot({
    p1<- ggplot(data=rate1, aes(x=DATE)) +
      geom_line(aes(y=CPIRate,colour = "blue")) +
      geom_line(aes(y=PCERate, colour = "red")) +
      labs(x="Date",y="Rate", title="CPI versus PCE Over Time",colour = "Yearly Inflation Rates") +
      theme(legend.position = c(0,1),legend.justification = c(0,1)) +
      scale_colour_hue(labels = c("CPI Rate", "PCE Rate"))

    p1
  })
})
```

ui.R

```r
h3("CPI vs PCE Over Time"),
plotOutput("CPIvPCE"),
br(),
```

Data load, processing, and graph rendering to ui element done in server.R. In ui.R, the plot element is declared.

Capabilities: Styling

Python Dash

R Shiny

The bottom of the Introduction page demonstrates styling capability. Both Python Dash and R Shiny have functions to produce many possible html formatting and styling tags.

Demonstrated here are bullet lists and markdown for equations. Other options include hyperlinks, video, images, and more.

Capabilities: Styling Implementation

Python Dash Bullet List

```python
detail_exchange = "On a national level, a devaluation of a country's currency can cause inflation.  " + \
    "This is due to increased costs of imports, and a stimulation of local demand due to a shift in " + \
    "consumer behavior."

detail_expectations = "As with any economic behavior, human behavior that affects the economy is " + \
    "due in part to human expectations of outcome.  For example, increased wages in anticipation of " + \
    "inflation contributes to continued inflation."

card3 = dbc.Card(
    dbc.CardBody(
        [
            html.H4("Causes of Inflation", className="card-title"),
            html.P(paragraph_causes),
            html.Ul([
                html.Li(detail_demand_pull),
                html.Li(detail_cost_push),
                html.Li(detail_exchange),
                html.Li(detail_expectations),
            ]),
        ]
    ),
)
```

Python Dash Equation (Markdown)

```python
paragraph_6 = "To be technical, an inflation index is the price of a specified basket of goods at a " +
    "specific period of time.  This price might be normalized, so that one period of time is 1, and " +
    "everything else is normalized to that time, such as inflation tables used in United States " + \
    "government appropriations.  Commonly, inflation is presented as a rate, a percentage increase " + \
    "compared to some previous time.  An inflation rate can be caclulated:"

paragraph_7 = "Commonly, rates are a percent increase from a previous year, though the BLS, for " + \
    "instance, reports the CPI as increase over the previous month.  In any case, to combine two rates, "
    "say, two years to get the total increase from year zero to year two, the rates need to be multiplie

equation_1 = dcc.Markdown('''
    $${{New Price - Old Price} \over Old Price} \cdot 100\%$
    ''', mathjax = True)

equation_2 = dcc.Markdown('''
    $Total Rate = (({{1 + {{Rate_1} \over {100\%}}}) \cdot ({1 + {{Rate_2} \over {100\%}}}) - 1)) \cdot
    ''', mathjax = True)

card4 = dbc.Card(
    dbc.CardBody(
        [
            html.H4("Calculating Inflation Rates", className="card-title"),
            html.P(paragraph_6),
            equation_1,
            html.P(paragraph_7),
            equation_2,
        ]
    ),
```

R Studio Equation (Markdown)

```r
h4("To be technical, an inflation index is the price of a specified basket of goods at a specific period of time.  This price might be normalized, so that one period of time is 1, and everything else is normalized to th
withMathJax(helpText("  $$\frac{New Price - Old Price}{Old Price} \\cdot 100\\%$$")),
h4("Commonly, rates are a percent increase from a previous year, though the BLS, for instance, reports the CPI as increase over the previous month.  In any case, to combine two rates, say, two years to get the total in
helpText("  $$Total Rate = \\left(\\left(1+\\frac{Rate_1}{100\\%}\\right))\\cdot\\left(1+\\frac{Rate_2}{100\\%}\\right)-1\\right)\\cdot100\\%$$")
);
```

Comparisons – Basic Functionality

Category	Python Dash	R Shiny
Result	Outputs HTML from programming language	Outputs HTML from programming language
File Structure	Runs an app.py file	Needs ui.R (for ui) and server.R (for reactivity)
Page Implementation	Pages folder with a .py file for each page (can have tabs on a page)	Essentially runs multiple pages as tabs
Layout, Formatting, Themes	Can build on grid system more easily with Dash Bootstrap	Grid system built into layout components; Themes already selectable
Styling	General styling (text, lists, markdown for equations)	General styling (text, lists, markdown for equations)
Graphics	Good with inbuilt Plotly (better than Matplotlib)	Ggplot2 is better than Base R (probably worse)
Miscellaneous	Can script in IDE, run on command line; Slower to load with multiple pages; Callback fails in runtime	Can develop/run in RStudio IDE; Sometimes error causes an annoying crash

Comparisons – Reactive Programming

The Inflation Calculator escalates a dollar amount from one month to another using four different inflation measures. The above image is a gif playable in Slide Show mode.

Comparisons – Reactive Programming Implementation

Python Dash

R Shiny

Data Read and Input Handling

Layout and User Interface

Callback to Render Calculator Output

ui.R

User Interface Layout

server.R

Calculator Output Rendering

In Python Dash, when the button is pushed, the callback runs a function to display the output, calling another function that reads the input and handles the data.

In R Shiny, the interface is defined in ui.R, with the output mapped and rendered by server.R. One difference from the graph, is that a button is used to only display output if clicked.

Comparisons – Implementing on an Enterprise Level

Comparisons – Implementing on an Enterprise Level

■ Commercial cloud-hosting options exist for both languages/packages

- Dash Design Kit from Plotly (for Dash)
- Posit Connect from Posit (for Shiny)

■ Some open source options also exist, such as the free ShinyServer for R and Shiny

■ Internal hosting is also an option (e.g., Tecolote Analytics Portal)

■ Organizational choice will depend on available blend of resources:

- Technical
- Human
- Financial
- Security

Internal hosting
Tecolote Analytics Portal (TAP)

- An internal (intranet only) web portal that hosts Web Applications such as R-Shiny and Python-Dash web apps.

- These applications are developed by analysts across the company for use by a broader audience
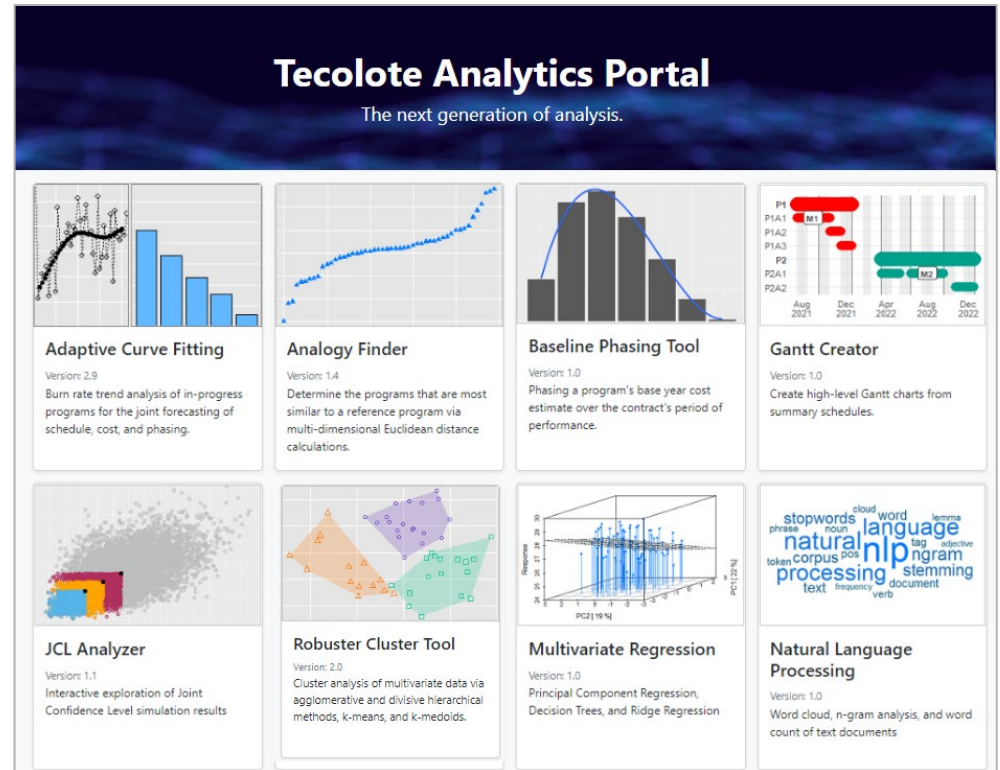
Conclusions

- Shiny in R and Dash in Python can both be used for similar web application functionality
  - Shiny is good for clean dashboards and loading data applicable for multiple tabs at once
  - Dash and Python is good for modularity and real web page/dashboard development

- Differences in some capability/file structure
  - Dash allows more complex file structure, which can be better organized, but may have loading speed implications
  - More extensive styling functionality available with additional packages in Dash that is somewhat inherent to Shiny
  - Plotly comes with Dash and has inbuilt high-level graphics

- R and Python's inherent language design creates differences
  - R is created for statistics, so has many inbuilt statistical functions
  - Python is designed as a programming language
    - Capabilities accessed by additional packages
    - Graphics capability arguably more advanced

- Best option will depend on user experience, IT capability, enterprise hosting options, etc.

Further Discussion and Future Thoughts

- Web applications are great as live dashboards
  - Professionally transitioning to a more virtual society
  - Pull live data from various sources
  - Online tools for modeling
  - Reactive programming

- Web applications do require effort to develop and possibly maintain
  - The more reactive the program or complex the styling, the longer this is
  - Version updates may be important
  - Enterprise-level hosting and security are challenging

- In some instances, BI tools (i.e. Tableau, Power BI, etc.) may be more advantageous

- Web forms are more complex than dashboards

Acknowledgements

- Pablo Barajas, Dr. Michael Schiavoni, Dr. Daniel Newkirk

- Tecolote Research, Inc.

- ICEAA

- You, the viewers

References

- Some introductions to inflation:

- Investopedia

- International Monetary Fund

- European Central Bank

- Some further information on causes of inflation:

- Reserve Bank of Australia

- Additionally, information on various inflation measurements:

- Brookings

- Investopedia

- San Francisco Federal Reserve

- The historical United States inflation data was sourced from:

- Federal Reserve Economic Data