PowerBI Overview

As mentioned previously, PowerBI is used to represent interactive insights from various data sources visually. It is a perfect tool for reporting, which is a re easy to build and maintain. PowerBI is also one of the easiest tools to get started with, as only a couple of crash courses should get you up to speed in nc to learn, but the drag and drop GUI interface is considered to be easier to learn than R, at least for business folks.

But here's one of the most important selling points – **PowerBI just looks good out of the box**. You don't have to be an expert to produce great-looking v requires much more manual labor to produce great-looking dashboards. Manual work isn't necessarily bad, as there's no limit to what you can do if you're drag and drop tools are more than enough most of the time.

> Looking for some good-looking Shiny dashboard examples? Check out Appsilon's Shiny Demo Gallery.

For every pro, there's a con, and PowerBI is not an exception. **One of the most significant disadvantages of PowerBI is that it is read-only.** As a use make decisions and save them in a database directly. **Also, PowerBI doesn't have an accessible source code.** You can only edit fields in WYSIWYG r start but difficult to maintain. Having no source code makes it nearly impossible to have proper version control, automatically test logic, or collaborate on l

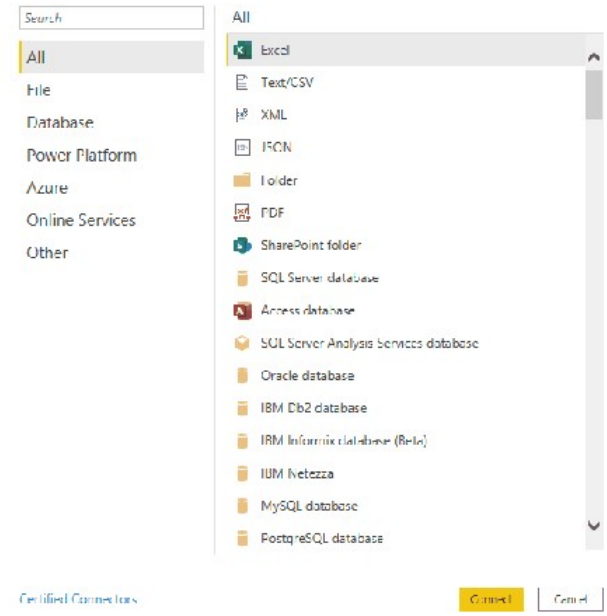
PowerBI comes in a few different flavors, listed below:

- *PowerBI Desktop* – an application you can download and install on your computer. Available only **for Windows**. It has powerful data analysis capal data sources. It is used to perform analysis, create visualizations, and to create reports.
- *PowerBI Service (Pro)* – web application. It is used to create visualizations and reports. The biggest selling point is dashboards -and they are easy results thanks to the collaboration mode.
- *PowerBI Mobile* – mobile application for both Android and iOS. It is used only to access your data from anywhere, not to perform analysis.

Most of our attention today will focus on the PowerBI Desktop variant.

Connectivity

PowerBI comes with many built-in connection types categorized into *Files*, *Databases*, *Power Platform*, *Azure*, and *Online services*, and is, without any do contestant – Tableau. As of late 2020, you find these connection options:

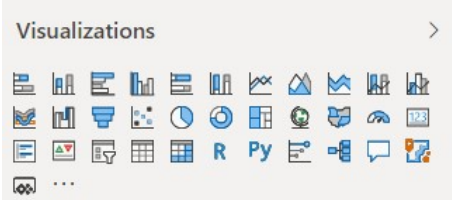


Put simply, PowerBI doesn't come short when it comes to connectivity. On the other side of the equation, R Shiny uses R as the programming language o any source that R can. A simple Google search will yield either a premade library or an example of API calls for any data source type. R Shiny can someti specific sources. Still, we've found multiple examples of PowerBI handling domain-specific data sources, such as CAD files.

Winner (Connectivity): Tie

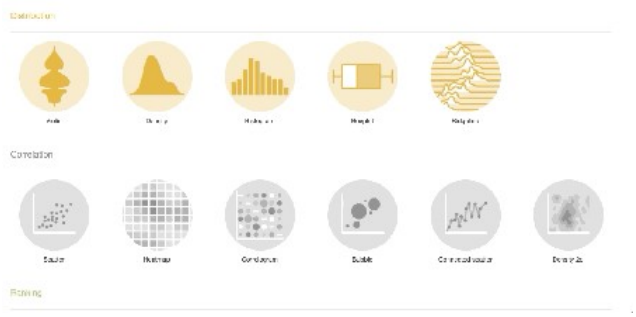
Chart Types

PowerBI provides basic visualization options – bar, line, area, scatter, and pie charts, with a couple of fancier types such as maps, treemaps, funnels, and below for a full list:



Still, it might be more than enough for most use cases, as more sophisticated statistical plots don't appear too often in production dashboards. Also, you r "Py" icons. This means you can use R and Python charts in PowerBI, with just one caveat – the source code of these charts is not under version control.

In R Shiny, you can use any visualization library that's available in R, such as `ggplot2` and `plotly`. Here's an overview of the types of visualizations you



GGplot2 options; Source: https://www.r-graph-gallery.com

Plotly options; Source: https://plotly.com/r/

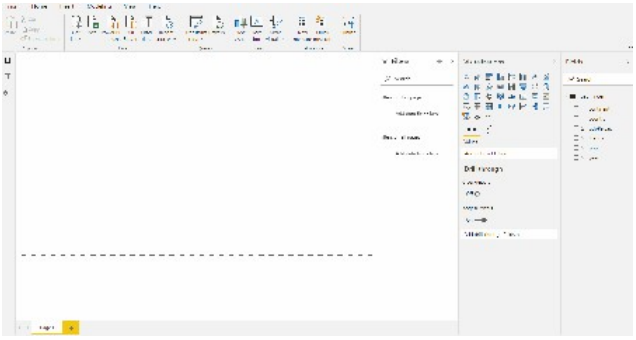
Declaring a winner in this department is a no-brainer. Sure, Shiny can do a lot, but PowerBI can do all of that and much more. PowerBI can handle its own

Winner (Chart Types): PowerBI

Ease of Use: Simple Charts

We'll now get our hands dirty with recreating the same visualization in both PowerBI and Shiny. For demonstration purposes, we'll use the Gapminder dat you're following along. The goal is to create a simple line chart, comparing average life expectancy over time per continent.

Let's start with PowerBI. We've imported the dataset and created the visualization with the following steps:



PowerBI was designed to be easy to use for people from all backgrounds, making this simple chart a no-brainer to implement.

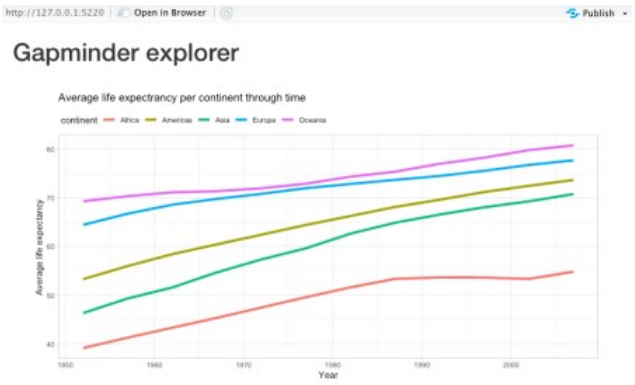
Replicating the same in R Shiny is quite a different story, as we need to write actual code. The dataset is available in R through the `gapminder` package, import the provided CSV. Shinyintroduces some boilerplate code – an apparent downside for this simple chart but negligible for larger, real-world projects.

Here's the code:

```
library(gapminder)
library(ggplot2)
library(dplyr)
library(shiny)

ui <- fluidPage(
  titlePanel("Gapminder explorer", windowTitle = NULL),
  plotOutput("line")
)

server <- function(input, output) {
  output$line <- renderPlot({
    data <- gapminder %>%
      group_by(year, continent) %>%
```

```
      summarise(avgLifeExp = mean(lifeExp))

    ggplot(data, aes(x = year, y = avgLifeExp, color = continent)) +
      geom_line(size = 1.5) +
      ggtitle("Average life expectrancy per continent through time") +
      labs(x = "Year", y = "Average life expectancy") +
      theme_light() +
      theme(
        plot.margin = unit(c(2, 1, 1, 1), "cm"),
        plot.title = element_text(vjust = 13),
        legend.position = c(0.25, 1.07), legend.direction = "horizontal"
      )
  })
}

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

And here are the results:



As it was the case with Tableau, there's no point in using R Shiny to produce single-chart and non-interactive dashboards. This is one of the areas where intuitive to use. Also, charts in R require a bit of customization to look decent and interpretable, which comes out of the box with PowerBI.

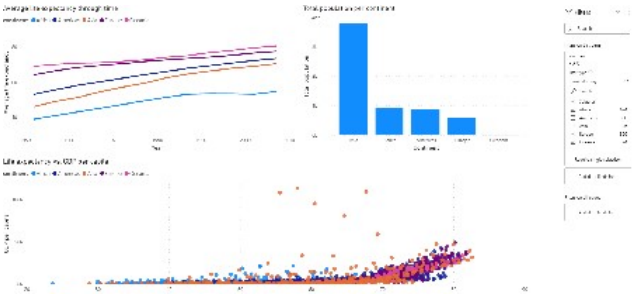
Winner (Simple Charts): PowerBI

Ease of Use: Simple Dashboards

Making a simple dashboard in PowerBI was easy – a couple of clicks here and there, a couple of filters, and we're ready to go. The goal was to display th

- Average life expectancy through time – as a line chart
- Total population per continent – only in the most recent year (2007), represented as a bar chart
- Life expectancy vs. GDP per capita – simple scatter plot, colored by continent

The entire dashboard can then be filtered by continent – by selecting a single one or multiple. Here's what we've managed to create in PowerBI:



To implement roughly the same dashboard in R Shiny, we need to write some R code. Once again, R is a relatively easy language to learn, so we don't se savvy users.

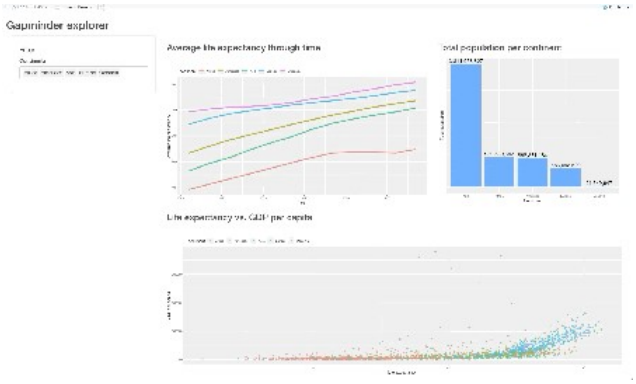
The following code recreates the dashboard we had in PowerBI:

```
library(shiny)
library(gapminder)
library(dplyr)
library(ggplot2)

ui <- fluidPage(
  titlePanel("Gapminder explorer", windowTitle = NULL),
  sidebarPanel(
    width = 3,
    tags$h4("Filter"),
```

```
    selectInput(
      inputId = "continentSelect",
      label = "Continents",
      choices = c("Africa", "Americas", "Asia", "Europe", "Oceania"),
      selected = c("Africa", "Americas", "Asia", "Europe", "Oceania"),
      multiple = TRUE
    )
  ),
  mainPanel(
    width = 9,
    fluidRow(
      column(7, tags$div(
        tags$h3("Average life expectancy through time"),
        plotOutput("line")
      )),
      column(5, tags$div(
        tags$h3("Total population per continent"),
        plotOutput("bar")
      ))
    ),
    fluidRow(
      column(12, tags$div(
        tags$h3("Life expectancy vs. GDP per capita"),
        plotOutput("scatter")
      ))
    )
  )
)

server <- function(input, output) {
  output$line <- renderPlot({
    dataLifeExp <- gapminder %>%
      filter(continent %in% input$continentSelect) %>%
      group_by(year, continent) %>%
      summarise(avgLifeExp = mean(lifeExp))

    ggplot(dataLifeExp, aes(x = year, y = avgLifeExp, color = continent)) +
      geom_line(size = 1.5) +
      labs(x = "Year", y = "Average life expectancy") +
      theme(
        plot.margin = unit(c(2, 0, 0, 0), "cm"),
        legend.position = c(0.25, 1.05),
        legend.direction = "horizontal"
      )
  })

  output$bar <- renderPlot({
    popPerContinent <- gapminder %>%
      filter(continent %in% input$continentSelect) %>%
      mutate(maxYear = max(year)) %>%
      filter(year == maxYear) %>%
      select(continent, pop) %>%
      group_by(continent) %>%
      summarise(total = sum(pop))

    ggplot(popPerContinent, aes(x = reorder(continent, -total), y = total)) +
      geom_bar(stat = "identity", fill = "#519bff") +
      labs(x = "Continent", y = "Total population") +
      geom_text(aes(label = formattable::comma(total, digits = 0)), vjust = -0.3, size = 5) +
      theme(
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank()
      )
  })

  output$scatter <- renderPlot({
    lifeExpWithGdp <- gapminder %>%
      filter(continent %in% input$continentSelect) %>%
      select(continent, gdpPercap, lifeExp)

    ggplot(lifeExpWithGdp, aes(x = lifeExp, y = gdpPercap, color = continent)) +
      geom_point(alpha = 0.7) +
      labs(x = "Life expectancy", y = "GDP per capita") +
      theme(
```

```
        plot.margin = unit(c(2, 0, 0, 0), "cm"),
        legend.position = c(0.15, 1.05),
        legend.direction = "horizontal"
      )
  })
}

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

And here's what the dashboard looks like:



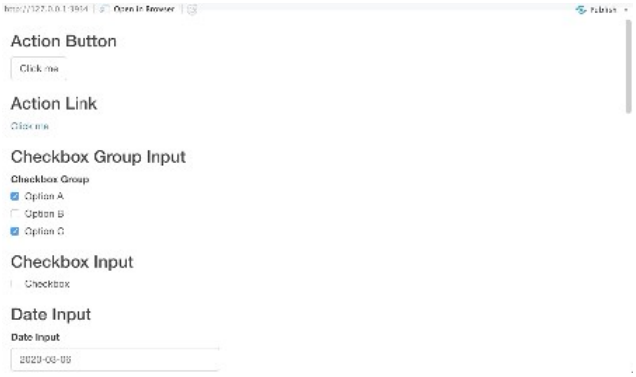
We'll work on visual appearance later, so don't worry too much about that. It's hard to announce a clear winner here, but R Shiny's dashboard feels more make a simple dashboard much faster with PowerBI. For this reason, we'll declare PowerBI the winner for simple dashboards, but only for ease of use. R require a sophisticated dashboard.

Winner (Simple Dashboards): PowerBI by a nose

User Input and Interactivity

As mentioned in the overview section, PowerBI is read-only. Sure, you can click on the various filters to include/exclude some data from the chart, but tha think inputs are essential in creating interactive dashboards, so a full web framework like Shiny annihilates its opponent in this department.

Shiny has a wide array of inputs – from text fields and buttons to dropdowns and modals. Here's everything Shiny provides:



With all of these options, if arranged right, there's no task too difficult for Shiny to solve. If you still aren't sold on the importance of interactivity, here are a reconsider:

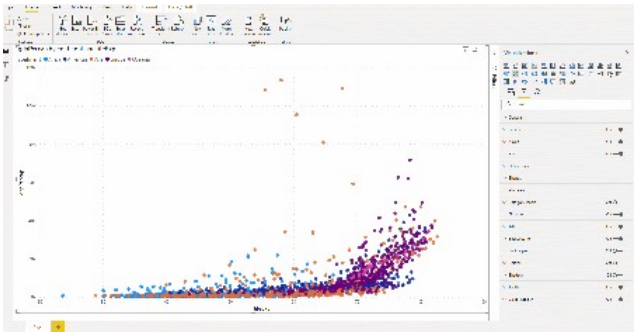
- The *file input* component allows us to upload a custom dataset and perform exploration in the browser
- The *text input* and *password input* fields allow us to build complete web forms – think authentication
- The *var select input* allows us to quickly select column(s) of interest from a dataset

Winner (User Input and Interactivity): R Shiny

Visual Styling

Tweaking the looks and feels of visuals in drag and drop tools like PowerBI or even Tableau was never their strong point. These tools are designed to lool agrees on what classifies as *good*, so it's nice to have options.

We've found that you can tweak a decent amount of things in PowerBI. Here's a rough overview:



The story is quite different with R Shiny. You can embed custom CSS styles by creating a `www` folder right where your Shiny app is. The CSS files are then

Learn More:How to Use CSS to Style Your R Shiny Dashboards

To connect the two, you have to put `theme = main.css` inside the `fluidPage` in the Shiny app. That's all. In just a couple of minutes with CSS, we've dashboard quite a bit:

Here's the source code for the completed dashboard. The winner of this battle is obvious, once again. PowerBI looks good out of the box, sure, but there's look perfect. R Shiny is extremely versatile when it comes to visual styling (in the hands of the right developer).

Winner (Visual Styling): R Shiny

Conclusion

The final results are in:

- PowerBI – 3 points
- R Shiny – 2 points
- Tie – 1 point

By our count, PowerBI took the lead by a single point for most general use-cases. Of course, choosing the appropriate tool isn't as simple as counting to 5 required. PowerBI is great when you need something relatively simple and aren't worried too much about the looks and overall feel of the dashboard. For overthrows PowerBI. PowerBI just can't compare with the customizability that Shiny offers, especially if you need to create an enterprise application.