**R-driven Power BI: Demo notes**

## Importing data

## 

From a *csv* file:

* Get Data > Text/CSV
* Browse to your file

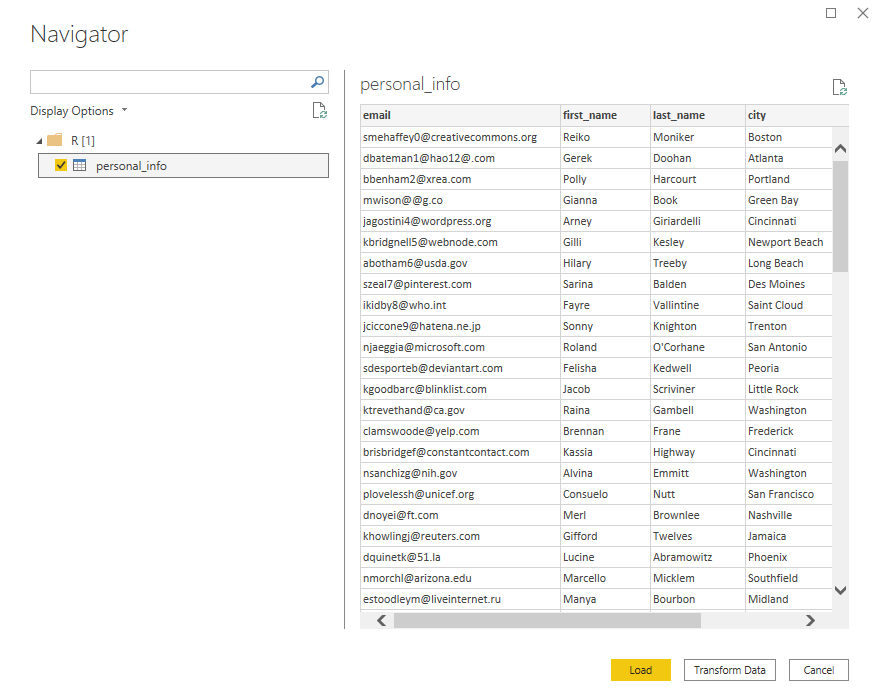
From an R Script:

* Get Data > More > R Script.
* A script box will appear. Read in the file. We will import a *csv*  file, but this is a great way to read in API data.

library(tidyverse)

personal\_info <- read\_csv('https://raw.githubusercontent.com/stringfestdata/satrday-r-power-bi/main/data/personal\_info.csv')

personal\_info

* From here you can check on the data frame you want and load it in.  
  

## Viewing relationships

One of Power BI’s most celebrated features is its relational data modeler. Select “Model” view and confirm that Power BI has automatically-detected a one-to-one relationship between contestants and personal\_info using the email column.

* Click
* Browse to your file

## Data profiling and ETL

Power BI also has an in-built ETL and data profiling tool called Power Query. This is another place where R scripts can be used.

* Go to Home > Transform data
* Select the contestants table
* Go to View and check on the Data Preview options
* We can see that some of our columns contain missing values. Power BI does not have a statistically-informed method to handle missing values, so this could be another use case. We will look at using regular expressions.

## Checking for invalid email addresses

* Keep the contestants table selected
* Go to Transform > Run R script
* As the script mentions, dataset is the de facto name of this data frame.
* Run the following script. Power BI does best when you assign the results to a brand-new data frame name.

is\_email <- function(x) {

grepl("\\<[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,}\\>", as.character(x), ignore.case=TRUE)

}

output <- dataset

output$is\_valid\_email <- is\_email(output$email)

* You will see the resulting script and table name in the Applied Steps menu

## Performing the paired-samples t-test

* We are going to end up with a table containing the “tidied” results of the test, so it’s best to duplicate the query so we have a second table (thus not impacting the original data)
* We’ll run the paired samples t-test and put the results into a table:

library(tidymodels)

contestants\_t <- tidy(t.test(dataset$pre, dataset$post, paired = TRUE, rm.na = TRUE))

* The results are now in the table.

## Inserting a visualization

Go to Home > Close & Apply to exit out of Power Query. We will now insert some visualizations into the report.

* We can insert a Table or Multi-row Card to insert the model parameters if we’d like. Drag one of these visualizations to the report and check on the fields you want to include.
* Power BI includes many common visualizations, but not everything. There are some great add-ins to make more custom visualizations, but why not use R? Choose R from the visualization menu.
* As you click on different fields, they will be added to the dataset data frame to plot.

library(tidyverse)

library(CGPfunctions)

dataset %>%

na.omit() %>%

group\_by(cohort) %>%

summarise(pre = round(mean(pre), 0), post = round(mean(post), 0)) %>%

pivot\_longer(cols = c(pre, post), names\_to = 'period', values\_to = 'score') %>%

newggslopegraph(period, score, cohort,

Title = 'Pre vs post scores by cohort',

SubTitle = 'for satRday Columbus :)',

DataTextSize = 4,

Caption = '',

WiderLabels = TRUE,

DataLabelPadding = .01)

* You can open the script in RStudio here and then run the visualization
* Remove the plot title by going to the paint icon to the right and checking off “Title.”

## Making the visualization interactive

By default, as we interact with one field in the Power BI report, all other instances of that field change.

For example, we can insert a table in the report counting the number of emails that are valid and invalid. As we click on the rows, our visualization is altered.