



Advanced Analytics in Power BI with R and Python

Ingesting, Transforming, Visualizing

—
Ryan Wade

Apress®

Advanced Analytics in Power BI with R and Python

Ingesting, Transforming, Visualizing

Ryan Wade

Apress®

Advanced Analytics in Power BI with R and Python: Ingesting, Transforming, Visualizing

Ryan Wade
Indianapolis, IN, USA

ISBN-13 (pbk): 978-1-4842-5828-6
<https://doi.org/10.1007/978-1-4842-5829-3>

ISBN-13 (electronic): 978-1-4842-5829-3

Copyright © 2020 by Ryan Wade

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484258286. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

I'd like to thank my colleagues at BlueGranite. It is an honor to be part of a team of very smart and talented individuals. Iron sharpens Iron.

I'd like to thank the neighborhood I grew up in, the west side of Michigan City, IN. My experience growing up there is priceless. I want to thank Patrick Leblanc, Mico Yuk, Terry Morris, Dr. Brandeis Marshall, and Dr. Sydeaka Watson. Each of you has attributes that I admire much, and you all mentor me from afar via your examples.

I'd like to thank all my teammates from little league to college. Athletics has been one of my best life teachers, and it was an honor to go through the process with you all. I look up to and respect many of you. I'd like to thank my former coaches, especially my high-school and collegiate coaches. The tough challenges you placed me in caused me to develop grit that has helped me in other aspects of my life.

Because of that, I will always be indebted to you.

I'd like to thank my extended family. It takes a village to raise a child, and I have benefited from the support of many family members, and I appreciate that. I'd like to thank my mentees, Tim Adams Jr. and Camille Little. Both of you have a rare combination of very high intellect and very likable personalities. In a short time, roles will switch, and you will become the mentor, and I will be the mentee! I'd like to thank my brothers from Michigan City and my brothers that I met while playing football in Louisville. I experienced blood, sweat, and tears with you all. We jumped off the porch and became men together. No matter what, we will be brothers for life! I'd like to thank my first cousins. We grew up like brothers and sisters and shared many fond memories. Some of you had my back in a time of need. That will never be forgotten. I want to thank my siblings, Paulette (RIP), Stephanie, Tina, and Luke, and my nephew Junebug, who was raised like my brother. When we needed each other, we always had each other's back. Let's stay that way. I'd like to thank my mom and dad, Luther and Ernestine Wade. I appreciate your love, support, and sacrifice for me and my siblings. It is much appreciated! In your words, daddy, "I love you, and that is always."

Last but not least, I'd like to thank God. I appreciate all the talents you have given me. I will show my appreciation by using them to the fullest so that I can be a benefit to my community.

Table of Contents

About the Authorxvii

About the Technical Reviewerxix

Acknowledgmentsxxi

Introductionxxiii

Part I: Creating Custom Data Visualizations Using R..... 1

Chapter 1: The Grammar of Graphics 3

 Steps to build an R custom visual in Power BI 4

 Step 1: Configure Power BI..... 4

 Step 2: Drag the “R custom visual” icon to the Power BI canvas..... 4

 Step 3: Define the data set 5

 Step 4: Develop the visual in your default R IDE..... 5

 Step 5: Use the following template to develop your visual..... 6

 Step 6: Make the script functional..... 7

 Recommended steps to create an R visual using ggplot2..... 7

 Step 1: Load the required packages that you will need for your script..... 8

 Step 2: Make any required adjustments to the data set..... 9

 Step 3: Initiate the creation of the visualization with the ggplot() function 10

 Step 4: Add desired geom(s)..... 10

 Step 5: Define your titles, subtitles, and caption 12

 Step 6: Make any necessary changes to the x and y axis 13

 Step 7: Apply themes if needed 16

 Step 8: Use the theme() function to change any specific non-data elements 17

 Bonus step: Specifying specific colors for your points in your scatter plot..... 20

TABLE OF CONTENTS

The importance of having “tidy” data 23

Popular geoms 24

Controlling aesthetics with scales 32

Themes built into ggplot2 34

Using R visuals in the Power BI service..... 35

Helper packages for ggplot2..... 35

Summary..... 36

Chapter 2: Creating R Custom Visuals in Power BI Using ggplot2..... 39

Callout chart..... 40

 Step 1: Acquire the necessary data..... 41

 Step 2: Create a slicer based on the year in the Filter pane..... 42

 Step 3: Configure the R visual in Power BI 43

 Step 4: Export data to R Studio for development..... 43

 Step 5: Load the required packages..... 45

 Step 6: Create the variables needed for the data validation test..... 46

 Step 7: Create the data validation test 46

 Step 8: Add additional columns to your data set that is required for the
 custom R visual 47

 Step 9: Create the variables that will be used for the dynamic portions of the chart 48

 Step 10: Start building the chart by defining the ggplot() function..... 51

 Step 11: Add a column chart layer to the R visual..... 52

 Step 12: Add a text layer to the R visuals 52

 Step 13: Modify the y axis 54

 Step 14: Convert chart from a vertical column chart to horizontal bar chart 55

 Step 15: Add a dynamic annotation to the R visuals 56

 Step 16: Add the dynamic titles and caption to the R visual 58

 Step 17: Remove labels from x axis and y axis 59

 Step 18: Remove legend..... 60

 Step 19: Change the look and feel of the visual using theme_few()...... 61

 Step 20: Center align titles 62

 Step 21: Add code to Power BI 64

Bubble chart.....	66
Step 1: Acquire the necessary data.....	67
Step 2: Load the data into Power BI	68
Step 3: Create a filter slicer based on the year	68
Step 4: Do the initial R visual configuration.....	68
Step 5: Export data to R Studio for development.....	69
Step 6: Load the required packages.....	69
Step 7: Create the variables needed for the data validation test.....	69
Step 8: Create the data validation test	70
Step 9: Define the colors for the conferences and conference divisions	70
Step 10: Dynamically define the chart titles.....	71
Step 11: Create the chart's data set	71
Step 12: Start the chart by defining the ggplot function	72
Step 13: Add the layer for your bubble chart using the geom_point geom	73
Step 14: Add labels to the bubble chart.....	74
Step 15: Change the color of the bubble's border and change the color of the bubble's fill	76
Step 16: Create the ggtitle.....	77
Step 17: Set the theme	78
Step 18: Add code to Power BI	78
Forecast	80
Step 1: Acquire the necessary data.....	81
Step 2: Create a slicer based on quarterback	83
Step 3: Configure the R visual	83
Step 4: Export data to R Studio for development.....	83
Step 5: Load the required packages to the script.....	83
Step 6: Create the variables needed for the data validation test.....	84
Step 7: Create the data validation test	84
Step 8: Create the dynamic chart title	85
Step 9: Create the data set that is needed to generate the forecast.....	85

TABLE OF CONTENTS

Step 10: Generate the forecast..... 86

Step 11: Generate the plot..... 86

Step 12: Add code to Power BI 88

Line chart with shade 89

 Step 1: Acquire the necessary data..... 92

 Step 2: Load the data into Power BI 93

 Step 3: Create the report slicers..... 94

 Step 4: Configure the R visual 94

 Step 5: Export data to R Studio for development..... 94

 Step 6: Load the required packages to the script..... 95

 Step 7: Create the variables needed for the data validation test..... 95

 Step 8: Create the data validation test 96

 Step 9: Create a new data frame based on the *dataset* data frame..... 97

 Step 10: Create the variables that will be used for the dynamic portions of the chart 97

 Step 11: Create the data sets needed for background shade 98

 Step 12: Create the data sets needed for line chart..... 102

 Step 13: Create a named character vector that will be used to color the shades..... 103

 Step 14: Start the chart by defining the ggplot function 103

 Step 15: Add a layer to create the background shade..... 104

 Step 16: Add a line chart based on the statistic selected..... 105

 Step 17: Reshade the background using pre-determined colors based on the
political party..... 106

 Step 18: Format the y axis based on the statistic selected..... 106

 Step 19: Add labels to the x and y axis..... 106

 Step 20: Add the dynamic titles and caption to the custom R visuals..... 107

 Step 21: Apply a theme based on *The Economists* publication 107

 Step 22: Add code to Power BI 107

Map 109

 Step 1: Acquire the necessary data..... 111

 Step 2: Load the data into Power BI 113

 Step 3: Create a slicer based on state in the Filter pane..... 113

 Step 4: Configure the R visual 113

Step 5: Export data to R Studio for development.....	113
Step 6: Load the required packages.....	113
Step 7: Create the variables needed for the data validation test.....	114
Step 8: Create the data validation test	114
Step 9: Create the variables for the chart titles.....	115
Step 10: Add the quintile column to the data set.....	115
Step 11: Define the colors that will be used to shade the map.....	116
Step 12: Define the ggplot() function.....	116
Step 13: Add the map layer	117
Step 14: Format the x and y axis.....	118
Step 15: Color the counties based on their quintile.....	118
Step 16: Improve the approximation of the selected state.....	119
Step 17: Add the dynamic titles and caption to the custom R visuals.....	120
Step 18: Apply the theme_map() theme.....	120
Step 19: Add code to Power BI	120
Quad chart	121
Step 1: Acquire the necessary data.....	124
Step 2: Load the data into Power BI	125
Step 3: Create a slicer for game type and period	126
Step 4: Configure an R visual on the report canvas.....	126
Step 5: Export data to R Studio for development.....	126
Step 6: Load the required packages.....	127
Step 7: Create the variables needed for the data validation test.....	127
Step 8: Create the data validation test	128
Step 9: Create the chart titles.....	128
Step 10: Add additional columns to the data set.....	129
Step 11: Start the chart by defining the ggplot function	129
Step 12: Use the geom_point() geom to plot the players on the plot.....	130
Step 13: Add the labels for each quadrant	130
Step 14: Draw vertical and horizontal lines through the x and y axis	131
Step 15: Add quadrant labels to the chart.....	131

TABLE OF CONTENTS

Step 16: Add labels for the x and y axis	134
Step 17: Add the dynamic titles and caption to the custom R visual.....	134
Step 18: Add a theme	135
Step 19: Perform last minute cleanup	135
Step 20: Add code to Power BI	138
Adding regression line	140
Step 1: Acquire the necessary data.....	142
Step 2: Load the data into Power BI	142
Step 3: Configure the R visual	142
Step 4: Export data to R Studio for development.....	142
Step 5: Load the required packages.....	143
Step 6: Create the variables needed for the data validation test.....	143
Step 7: Create the data validation test	144
Step 8: Start the chart by initializing the ggplot function	144
Step 9: Add a scatter plot layer to the R visual.....	144
Step 10: Add regression line layer to the R visuals	145
Step 11: Add a title to the chart.....	146
Step 12: Change the chart's theme	147
Step 13: Perform some last minute cleanup	147
Step 14: Add code to Power BI	147
Part II: Ingesting Data into the Power BI Data Model Using R and Python	149
Chapter 3: Reading CSV Files	151
Dynamically combining files.....	151
Example scenario	152
Picking the rolling 24 months using R.....	152
Picking the rolling 24 months using Python	163
Filtering rows based on a regular expression.....	171
Leveraging regular expressions via R	171
Leveraging regular expressions via Python.....	173

Chapter 4: Reading Excel Files	177
Reading Excel files using R.....	178
Step 1: Import the tidyverse and readxl package	179
Step 2: Create the shell of the combine_sheets function.....	179
Step 3: Get the name of the sheets you need to combine in your function from the specified Excel workbook.....	179
Step 4: Convert the character vector built in Step 3 to a named character vector	180
Step 5: Use the mapr_df function to combine the sheets into a single data frame	181
Step 6: Return the data frame	182
Step 7: Set the working directory to the location where the Excel files are located.....	183
Step 8: Assign the file names to the excel_file_paths variable.....	183
Step 9: Use the map_dfr function to apply the combine_sheets function to each file in the working directory.....	183
Step 10: Copy the R script and paste it into the R editor via GetData in Power BI	184
Reading Excel files using Python	184
Step 1: Import the os and pandas library	185
Step 2: Create the shell of the combine_sheets function.....	185
Step 3: Create an Excel object based on the workbook located at the path specified in the excel_file_path variable.....	186
Step 4: Create a list of the sheet names in the workbook specified by the excel_file_path variable	186
Step 5: Use the read_excel method of pandas to read the data from each sheet into one data frame	186
Step 6: Return the data frame held in df as the output from the combine_sheets function	187
Step 7: Set the working directory to the location that contains the Excel data you want to combine	187
Step 8: Get the list of the file names in the current working directory and assign it to the excel_file_paths variable	187
Step 9: Create an empty data frame and name it combined_workbooks	188
Step 10: Create the shell of the for loop.....	188
Step 11: Combine all the data in each sheet into one data frame using the combine_sheets function	189

TABLE OF CONTENTS

Step 12: Append the combined_workbook data frame to the combined_workbooks data frame	189
Step 13: Copy the Python script and paste it into the Python editor via GetData in Power BI	189
Chapter 5: Reading SQL Server Data	191
Adding AdventureWorksDW_StarSchema database to your SQL Server instance.....	191
Reading SQL Server data into Power BI using R.....	192
Step 1: Create a DSN to the SQL Server database	193
Step 2: Create a log table in SQL Server	198
Step 3: Start developing the R script to load DimDate	199
Step 4: Create a variable to hold the name of the table you want to import	199
Step 5: Create a variable to hold the sql statement that will be used to return the table.....	200
Step 6: Create a connection to SQL Server.....	200
Step 7: Retrieve the data from SQL Server and store it in a data frame	200
Step 8: Get the current time	201
Step 9: Get the number of records that were read	201
Step 10: Create a one record R data frame that contains the information you want to log.....	201
Step 11: Insert the information you gathered in Step 10 into the history log table.....	202
Step 12: Close your connection	203
Step 13: Copy the script into Power BI	203
Step 14: Create a script to load DimProduct based on the ReadLog_DimDate.R script	204
Step 15: Create a script to load DimPromotion based on the ReadLog_DimDate.R script.....	205
Step 16: Create a script to load DimSalesTerritory based on the ReadLog_DimDate.R script.....	206
Step 17: Create a script to load FactInternetSales based on the ReadLog_DimDate.R script.....	207
Reading SQL Server data using Python	208
Step 1: Create a DSN to the SQL Server database	208
Step 2: Create a log table in SQL Server	208
Step 3: Begin creating the script to load the DimDate table	209

Step 4: Create a variable that holds the name of the table that you want to read into Power BI	210
Step 5: Create a connection to the database using sqlalchemy.....	210
Step 6: Read in the contents of the DimDate table and store it as a data frame in the df_read variable	210
Step 7: Get the current date and time and store the information into the datstamp variable.....	210
Step 8: Calculate the number of records in the DimDate table	211
Step 9: Create a one record pandas data frame that contains the information you want to log	212
Step 10: Insert the information you gathered in Step 9 into the history log table.....	212
Step 11: Copy the script into Power BI	213
Step 12: Create a script to load DimProduct based on the ReadLog_DimDate.py script	214
Step 13: Create a script to load DimPromotion based on the ReadLog_DimDate.py script	215
Step 14: Create a script to load DimSalesTerritory based on the ReadLog_DimDate.py script	216
Step 15: Create a script to load FactInternetSales based on the ReadLog_DimDate.py script	217
Chapter 6: Reading Data into the Power BI Data Model via an API	219
Reading Census data into Power BI via an API using R	219
Step 1: Get a personal Census API key	220
Step 2: Load the necessary R packages.....	220
Step 3: Identify the variables you want to return from your data set	221
Step 4: Create a character vector of the tables that contains the variables you want to return.....	222
Step 5: Configure the get_acs function	222
Step 6: Give the variables (columns) meaningful names.....	223
Step 7: Copy the script into Power BI	224
Reading Census data into Power BI via an API using Python.....	225
Step 1: Get a personal Census API key	226
Step 2: Load the necessary Python libraries	226
Step 3: Identify the variables you want to return in your data set.....	226

TABLE OF CONTENTS

Step 4: Create a variable that is based on the list variables you want..... 228

Step 5: Create a list of tuples that contains the geographies you want to in
your data set..... 229

Step 6: Retrieve the data using the *censusdata.download()* function 229

Step 7: Reset the index of the data frame created in Step 6..... 230

Step 8: Define the new column names..... 230

Step 9: Rename columns..... 231

Step 10: Copy the script into Power BI 231

Summary..... 232

Part III: Transforming Data Using R and Python 233

Chapter 7: Advanced String Manipulation and Pattern Matching 235

Masking sensitive data 236

 Masking sensitive data in Power BI using R..... 236

 Masking sensitive data in Power BI using Python..... 241

Counting the number of words and sentences in reviews..... 246

 Counting the number of words and sentences in reviews in Power BI using R..... 246

 Counting the number of words in reviews in Power BI using Python..... 249

Removing names that are in an invalid format 251

 Removing names that are in an invalid format in Power BI using R 251

 Removing names that are in an invalid format in Power BI using Python..... 254

Identifying patterns in strings based on conditional logic 259

 Identifying patterns in strings based on conditional logic in Power BI using R..... 261

 Identifying patterns in strings based on conditional logic in Power BI using Python..... 264

Summary..... 268

Chapter 8: Calculated Columns Using R and Python 269

Create a Google Geocoding API key 270

 Step 1: Log into the Google console 270

 Step 2: Set up a billing account..... 271

 Step 3: Add a new project..... 271

 Step 4: Enable Geocoding API..... 273

Geocode the addresses using R.....	275
Geocode the addresses using Python.....	277
Calculate the distance with a custom function using R.....	281
Calculate the distance with a custom function using Python	283
Calculate distance with a pre-built function in Power BI using R.....	287
Calculate distance with a pre-built function in Power BI using Python	289
Summary.....	291
Part IV: Machine Learning and AI in Power BI Using R and Python...	293
Chapter 9: Applying Machine Learning and AI to Your Power BI Data Models	295
Apply machine learning to a data set before bringing it into the Power BI data model.....	296
Predicting home values using R	297
Predicting home values using Python	300
Using pre-built AI models to enhance Power BI data models	304
Set up Cognitive Services in Azure.....	305
The Data Science Virtual Machine (DSVM)	306
Performing sentiment analysis in Microsoft Cognitive Services via Python.....	307
Applying AI to your Power BI data model using services other than Microsoft Cognitive Services	314
Configuring the Tone Analyzer service in IBM Watson.....	314
Writing the Python script to perform the tone analysis.....	317
Chapter 10: Productionizing Data Science Models and Data Wrangling Scripts	329
Predicting home values in Power BI using R in SQL Server Machine Learning Services	330
Build the R script that adds the model to SQL Server	330
Use SQL Server Machine Learning Services with R to score the data	333
Predicting home values in Power BI using Python in SQL Server Machine Learning Services	340
Create the script needed to add Python model to SQL Server	340
Use SQL Server Machine Learning Services with Python from Power BI to score the data.....	345

TABLE OF CONTENTS

Performing sentiment analysis in Power BI using R in SQL Server Machine Learning Services 351

 Add pre-built R models to SQL Server Machine Learning Services using PowerShell..... 351

 Use pre-built R sentiment model in SQL Server Machine Learning Services to score data in Power BI 353

Performing sentiment analysis in Power BI using Python in SQL Server Machine Learning Services 361

 Add pre-built Python models to SQL Server Machine Learning Services..... 361

 Use pre-built Python sentiment model in SQL Server Machine Learning Services to score data in Power BI 363

Calculating distance in Power BI using R in SQL Server Machine Learning Services 369

 Step 1: Make sure dplyr is loaded in SSMLS..... 369

 Step 2: Launch SSMS and connect to a SQL Server..... 369

 Step 3: Add the CalculateDistance database to the server..... 370

 Step 4: Add the stored procedure that will calculate the distances 371

 Step 5: Call the Power BI procedure from Power BI 374

Calculating distance in Power BI using Python in SQL Server Machine Learning Services..... 375

 Step 1: Launch SSMS and connect to a SQL Server..... 376

 Step 2: Add the CalculateDistance database to the server..... 376

 Step 3: Add the stored procedure that will calculate the distances 377

 Step 4: Call the Power BI procedure from Power BI 380

Index..... 383

About the Author



Ryan Wade is a data analytic professional with over 20 years of experience. His education and work experience enable him to have a holistic view of analytics from a technical and business viewpoint. He has an MCSE with an emphasis on BI reporting and Microsoft R. He has an advanced understanding of R, Python, DAX, T-SQL, M, and VBA. He knows how to leverage those programming languages for on-prem and cloud-based data analytics solutions using the Microsoft Data Platform.

Ryan is a data analytics enthusiast, and he has spoken at R meetups, Python meetups, SQLSaturdays, TDWI Conference, BDPA Conference, and PASS Summit about various data analytics topics. He is the developer of a comprehensive online course for ExcelTv showing how to implement R in Power BI for advanced data analytics and data visualization.

About the Technical Reviewer

Aaditya Maruthi works as a Senior Database Engineer for a reputed organization. Having over ten years of experience in RDMS systems like Microsoft SQL Server and Oracle, he worked extensively on Microsoft technologies like the SSAS, SSRS, SSIS, and Power BI.

Aaditya is also a Certified AWS Solutions Architect Associate.

Acknowledgments

I want to thank the technical editors, Mike Huffer and Aaditya Maruthi. The feedback that you all gave me was very valuable and much appreciated. I'd also like to thank Jonathan Gennick and Jill Balzano. I appreciate your patience and help. I would not have been able to complete the process without your guidance. I also want to thank both of you for keeping things in perspective. There was so much I wanted to include, but that would have taken way too long to write. You helped me decide what was important, which helped us finish the book in a reasonable time.

Introduction

Microsoft Power BI is considered by many to be the premier self-service business intelligence tool on the market. In recent years, it has passed up formidable tools such as QlikView and Tableau to gain the number one spot. One of the reasons why it is considered such a great self-service business intelligence tool is because it is more than just a visualization tool. Built into Microsoft Power BI are

- The *DAX* expression and query language that enables you to interrogate your Power BI data model using complex business logic in a fast and efficient way
- A data wrangling tool called *Power Query* that enables you to shape and transform your data into a form that is conducive for data analysis
- The *Vertipaq* engine that efficiently stores the data in a way that is optimized for reporting and performs complex calculations fast and efficiently
- Pre-packaged, interactive visualizations that enable you to present your data in ways that are easily understood by report consumers

So, you may ask the question, with all these features why would you need to leverage programming languages such as R and Python in Power BI? The answer is to fill in the few areas where the native tools fall short. A few examples are

- Creating custom visualizations in a relatively easy way
- Applying data science to your Power BI data models without the need of *Power BI Premium*
- Performing advanced string manipulations using advanced techniques that are not available in *Power Query* or *DAX*
- Interacting with *Microsoft Cognitive Services* without the need of *Power BI Premium*

INTRODUCTION

- Communicating with third-party data APIs to enrich your Power BI data models in an efficient way
- And many more

This book covers how to leverage R and Python to bring the added functionality listed earlier to your Power BI solutions. R is a perfect complement to Power BI because it is a language written specifically for data analytics. Data analysts have been using R to perform tasks like data wrangling and data visualization for decades. Given that, features that may not be available in Power BI might have been in R for some time.

Python has become a very popular programming language in data analytics over the last decade. One of the features that make Python so attractive is that it is not only great for data analytics, but it is great for general programming tasks as well. Communicating with APIs is a breeze with Python, but the same task is very clunky using Power Query.

These features of R and Python make them perfect companions to Power BI. This book will cover some recipes that illustrate the preceding features. The recipes will include detailed steps along with verbose descriptions so that you will get a clear understanding of how they work. Before you get started using the recipes, you need to configure your environment. Let's go over those configurations.

Configure your Azure environment

Different parts of *Azure* will be used throughout the book. *Microsoft Cognitive Services* will be used to apply artificial intelligence to your Power BI data models. Also, the *Data Science Virtual Machine (DSVM)* is the recommended development environment for the book. The *DSVM* is optional but highly recommended. In order to work many of the examples in this book, you will need an environment configured with many tools such as

- SQL Server 2017 or later
- SQL Server Machine Learning Services 2017 or later with R and Python enabled
- The Anaconda distribution of Python
- The R programming language
- R Studio

- VS Code
- Power BI Desktop

Manually configuring an environment with these resources can be a challenge, but if you use the *DSVM*, most of the configuration is handled for you. In the following sections, you will learn how to set up *Azure* so that you can consume *Microsoft Cognitive Services* and you will also learn how to spin up a *DSVM*.

Sign up for Azure

Sign up for Azure here: <https://azure.microsoft.com/en-us/free/>. You get 12 months free for selected services plus \$200 in credit during your first month!

Sign up for Microsoft Cognitive Services

Microsoft Cognitive Services will be used to perform sentiment analysis inside of Power BI using Python. You first need to set up the *Microsoft Cognitive Services* in *Azure* before you can call it from Power BI. Here are the required steps to set up the service:

1. Log in to your Azure Portal.
2. Type *Cognitive Services* in the search box, then press Enter.
3. The preceding action should take you to the *Cognitive Service* signup page. Click the *Create* button to initiate the signup process.
4. Fill in the following information:
 - Name
 - Subscription
 - Location
 - Pricing tier
 - Resource group
5. Click the *I confirm I have read and understood the notice below* check box.
6. Click the *Create* button.

Note that there is a cost to use *Microsoft Cognitive Services*. To get pricing information, go to your *Microsoft Cognitive Services* resource and type *Pricing tier* in the search box, then click it in the results. Select the pricing tier you want to use. Doing so will take you to a page that will give you pricing information based on your usage and *Azure* region. The exercise in this book that uses *Microsoft Cognitive Services* is relatively inexpensive, and you will have more than enough credits in your first month to cover the cost.

Create a Data Science Virtual Machine (DSVM)

The preferred setup is to start with a *Data Science Virtual Machine (DSVM)* and add the resources that do not come pre-installed in the *DSVM* to it. This setup is highly preferred because the amount of time it takes to fully configure your environment in the way that is needed to do every exercise in the book can be lengthy and challenging. Using the *DSVM* enables you to configure an environment in minutes that could take you many days of trial and error if you tried to do it yourself. Instructions will be provided in the next section if you decide to configure your own environment. Here are the instructions to set up the *DSVM*.

Steps to create a DSVM in Azure

1. Go to <https://portal.azure.com>. If prompted, sign in using the credentials created in Step 1.
2. Click *Create a resource* in the upper left.
3. In the search box, type *Data Science Virtual Machine - Windows 2019*.
4. Click the *Create* button. This action will cause a form to appear that you need to fill out to configure the *DSVM*. You will land on the *Basics* tab. The following steps will tell you how to fill out the form.
5. *Subscription*: Select the subscription you want to use. It should default to the subscription that you set up in Step 1.

6. *Resource group*: If you already have a resource group in your tenant you want to use, select it. Otherwise, create a new one for your DSVM">.
7. *Virtual machine name*: The name you want your DSVM to have.
8. *Region*: An Azure region close to you.
9. *Image*: Make sure Data Science Virtual Machine – Windows 2016 is selected.
10. *Size*: I use B4ms because it is the cheaper option for the 16 gigs of ram options. RAM is important for R, Python, and Power BI.
11. *Username*: <"Create a username">.
12. *Password*: <"Create password">.
13. *Confirm password*: <"Confirm password">.
14. Click *Next: Disks* >.
15. *OS disk type*: Select Standard SSD. This option is sufficient for what we are doing.
16. Click *Next: Networking*>.
17. Make sure all the required fields are filled out. You can identify the required fields with a *. They should be populated with a default value; if not, click the Create New link below them and accept the default settings.
18. Click *Next: Management* >.
19. Accept defaults and click *Next: Advanced* >.
20. Accept defaults and click *Next: Tags* >.
21. Click *Next: Review + create* >.
22. You will see a summary of the DSVM that you configured. You will also see the cost to run the DSVM. If you agree with the configuration, click the *Create* button.

INTRODUCTION

Make sure to stop your DSVM after every use so that you don't incur unnecessary costs.

As a safeguard, you should set up an auto-shutdown that will shut down your DSVM if it is still running past a specified time. Do the following steps to set up auto-shutdown:

1. Go into your DSVM machine in the *Azure Portal*.
2. Type *auto-shutdown* in the search box, then select it in the list.
3. Enable auto-shutdown by selecting *On* in the *Enabled* button.
4. Select the time you would like to automate the shutdown in the *Scheduled shutdown* textbox.
5. Choose the time zone you want to base the time on in the *Time zone* combo box.
6. If you want notification to be sent to you that lets you know that your DSVM will be shutting down, you can turn on the *Send notification before auto-shutdown*. The notification will be sent to the email that you put in the *Email address* textbox.

Configure R in DSVM

You will be using a different distribution of R than the one installed. The distribution of R that we will use is *Microsoft R Open (MRO)*. This distribution of R is totally compatible with distribution on CRAN, but it comes with enhancements that improve the performance of certain types of calculations plus many additional tools. Perform the following steps to download the *MRO* in your DSVM:

1. Get the version of R that is being used in the Power BI service.
You can find that information in the following Microsoft documentation: <https://docs.microsoft.com/en-us/power-bi/visuals/service-r-visuals>.
2. Open up a browser in the DSVM and go to the following site: <https://mran.microsoft.com/open>. Two browsers are pre-installed in the DSVM. They are *Microsoft Edge* and *Firefox*.

3. Click the *Download* button on the right and you will be taken to the download page. Once on the download page, click the *Past Releases* link which is located on the right section of the page. Clicking the link will take you to a page that has links to all previous versions of *Microsoft R Open*. Click the link for the version that the Power BI service is using.
4. Choose the download for *Windows*.
5. Execute the download.
6. Open R Studio in the *DSVM*.
7. Select *Tools* ► *Global Options*. Verify that the MRO distribution you just installed is selected. If it is not, click the *Change...* button and you should see it as one of the options. Select it, then click *OK*.

Configure Python in DSVM

One of the benefits you gain with the DSVM is you get a distribution of Python pre-installed that is perfect for data analytics. The name of the distribution is *Anaconda*. The *Anaconda* distribution of Python comes pre-installed with over 1500 libraries that are popular in data analytics. It also comes with a package manager and environment management system named *conda*. Installing libraries via *conda* is preferred because of how *conda* manage package dependencies. The environment management system in *conda* makes it easy to create an isolated copy of a specific version of Python with specific versions of Python libraries in it.

Let's create a dedicated Python environment in the *DSVM* for this book and name it *pbi*. To do so, you need to perform the following steps:

1. Log into the *DSVM*.
2. Open the command prompt by clicking in the search bar next to the *Windows* sign and type *cmd*.
3. Type the following code to create a *conda* environment named *pbi* based on Python 3.7:

```
conda create -n pbi python=3.7
```

INTRODUCTION

The decision to use python 3.7 was based on information obtained from the following Microsoft documentation: <https://docs.microsoft.com/en-us/business-applications-release-notes/october18/intelligence-platform/power-bi-service/pervasive-artificial-intelligence-bi/python-service>. According to the documentation, the Power BI service is compatible with Python 3.x so the current 3.x versions of Python should be compatible.

Now we have a Python environment that we can use for our Python development in this book.

Configuring SQL Server Machine Learning Services in DSVM

Several examples of the book require the use of *SQL Server Machine Learning Services* (SSMLS). SSMLS provides tools that enable you to perform advanced analytics inside the database using R, Python, and some tools that make it easier to work with big data. SSMLS also offers some pre-trained models built by Microsoft that you can leverage. The preceding features are not part of the default features, but SSMLS is enabled by default in the DSVM. If you are not using the DSVM, you will have to enable it, and instructions of how to do so can be found here: <https://docs.microsoft.com/en-us/sql/machine-learning/install/sql-machine-learning-services-windows-install?view=sql-server-ver15>. The pre-trained models that will be used in the book can be added to your instance of SQL Server as a post-task installation. Go to this URL to get instructions of how to install the pre-trained models as a post-task installation: <https://docs.microsoft.com/en-us/sql/machine-learning/install/sql-pretrained-models-install?view=sql-server-ver15>.

Installing R packages

Some of the R scripts in this book may contain R packages that are not installed on your machine. Installing packages in R is simple and straightforward. The following code installs a popular R package named *data.table* via the R console:

```
install.packages("<package name>")
```

There will be times when you may want to install multiple packages at once. For instance, you may want to install the R package *data.table* and *dplyr* together. You can accomplish that task by creating a character vector that contains two elements, one for *data.table* and another for *dplyr*, and assign the results to a variable named *pkgs*. Then you would pass that variable to the *install.packages()* function as illustrated here:

```
pkgs <- c("data.table", "dplyr")
install.packages(pkgs)
```

The *R character vector* data type is a one-dimensional array of the character data type. You will learn more about this data types as well as other R data types throughout the book.

When you are creating R visuals in Power BI, you need to be cognizant of which version of the package is being used by the Power BI service. You can get a list of all the available R packages in the Power BI service along with their version at this URL: <https://docs.microsoft.com/en-us/power-bi/service-r-packages-support>.

The *install.packages()* will download the most recent version of the package from the repository you are using if you are using the distribution of R from *CRAN*. If you are using *Microsoft R Open*, it will install the most recent package based on the *snapshot* date. Both methods may result in a version being installed that is not the same as the one in the service. To download the version of a package that is being used in the service, you first need to get the package version from the page located at the preceding URL, then you need to use the *devtools* package to install it. Here is an example of using the *devtools* package to install *ggplot2 0.9.1* from CRAN:

```
library(devtools)
install_version(
  "ggplot2",
  version = "0.9.1",
  repos = "http://cran.us.r-project.org")
```

Installing Python libraries

There are multiple ways you can install libraries in Python. You will use two methods in this book, *conda* install and *pip* install. Installing libraries in Python is not as easy and straightforward as it is installing packages in R. In this book, you will use the *conda* prompt to install Python libraries. Perform the following steps to install Python a library using *conda*:

1. Go to the *Windows search bar* located on the lower right next to the *Windows* icon.
2. Type the word *Anaconda* and the *Anaconda Prompt* should appear in the returned list. Click it to launch the *Anaconda Prompt*.
3. Activate the environment that you are using for Power BI by typing the following code in the command prompt:

```
conda activate "<environment name>"
```

It is highly recommended to use an environment for the Python development associated with this book. Instructions of how to create one are located in the next section.

4. Install the package using *conda* with the following code:

```
conda install <"package name">
```

So, if you were installing *pandas*, the code would be as follows:

```
conda install pandas
```

If you wanted to install *pandas* 1.0.4, you would use the following code:

```
conda install pandas=1.0.4
```

Not all packages are available for a *conda* install. Go to the following URL to get a list of packages that can be installed using *conda* in Python 3.6: https://docs.anaconda.com/anaconda/packages/py3.6_win-64/. One of the packages that will be used

in this book is not available in conda but is available in *PyPI*. The name of the package is *CensusData*. You must use *pip* to install the *CensusData* library as illustrated here:

```
pip install CensusData
```

Configure Power BI in DSVM

There are several configurations that you need to do in the *Power BI Desktop* to enable R and Python. Those steps are covered in detail in the book's code repository in *GitHub*. You will also find instructions on how to create and use *conda* environment in Python. Here is the URL to the book's code repository: <https://github.com/Apress/advanced-analytics-in-power-bi-w-r-and-python>.

Alternative setup

That *DSVM* is the optimal way to go, but it may not be an option for you. If that is the case, you will have to manually install the required software. Here are links to the required software that you need to install:

- Manually Install Power BI: www.microsoft.com/en-us/download/details.aspx?id=58494
- Manually Install R Studio: <https://rstudio.com/products/rstudio/download/>
- Manually Install Microsoft R Open: <https://mran.microsoft.com/download>
- Manually Install Anaconda: www.anaconda.com/products/individual
- Manually Install VS Code: <https://code.visualstudio.com/download>
- Manually Install SQL Server 2019 Developer: www.microsoft.com/en-us/sql-server/sql-server-downloads