

Walter R. Paczkowski

Business Analytics

Data Science for Business Problems

 Springer

Business Analytics

Walter R. Paczkowski

Business Analytics

Data Science for Business Problems

 Springer

Walter R. Paczkowski
Data Analytics Corp.
Plainsboro, NJ, USA

ISBN 978-3-030-87022-5 ISBN 978-3-030-87023-2 (eBook)
<https://doi.org/10.1007/978-3-030-87023-2>

© Springer Nature Switzerland AG 2021

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.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

I analyze business data—and I have been doing this for a long time. I was an analyst and department head, a consultant and trainer, worked on countless problems, written many books and reports, and delivered numerous presentations to all levels of management. I learned a lot. This book reflects insights I gained from this experience about *Business Data Analytics* that I want to share.

There are three questions you should quickly ask about this sharing. The first is obvious: “*Share what?*” The second logically follows: “*Share with whom?*” The third is more subtle: “*How does this book differ from other data analytic books?*” The first is about focus, the second is about target, and the third is about competitive comparison. So, let me address each question.

The Book’s Focus

My experience has been with practical business problems. When I finished my academic training with a Ph.D. in economics and a heavy statistics exposure, I immediately started my professional career with an AT&T internal consulting group, *The Analytical Support Center (ASC)*. I quickly learned that I needed both a theoretical, technical understanding of quantitative work—how to estimate a regression model, for example—and an understanding of how to deal with messy data beyond the nice, clean data sets I used as a graduate student. My time at the ASC was a great learning experience that I carried throughout my professional career at AT&T, including Bell Labs, and into my own consulting business. The lessons I learned were that good, solid data analysis for practical business problems requires:

1. A theoretical understanding of statistical, econometric, and (in the current era) machine learning methods
2. Data handling capabilities encompassing data organizing, preprocessing, and wrangling
3. Programming knowledge in at least one software language

These three components form a synergistic whole, a unifying approach if you wish, for doing business data analytics, and, in fact, any type of data analysis. This synergy implies that one part does not dominate any of the other two. They work together, feeding each other with the goal of solving only one overarching problem: how to provide decision makers with rich information extracted from data. Recognizing this problem was the most valuable lesson of all. All the analytical tools and know how must have a purpose and solving this problem is that purpose—there is no other.

I show this problem and the synergy of the three components for solving it as a triangle in Fig. 1. This triangle represents the almost philosophical approach I take for any form of business data analysis and is the one I advocate for all data analyses.

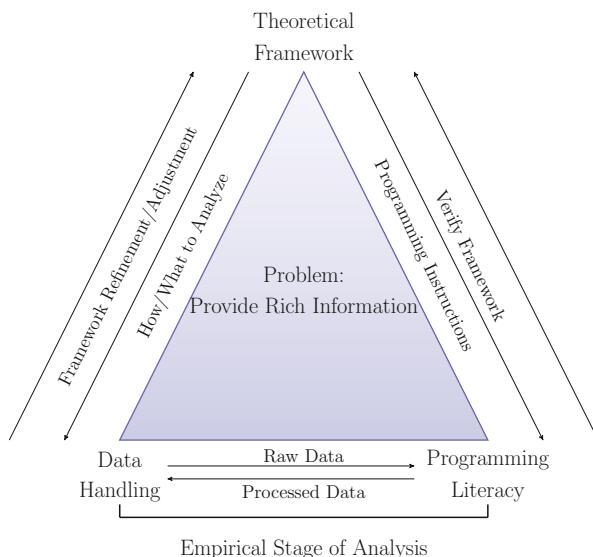


Fig. 1 The synergistic connection of the three components of effective data analysis for the overarching problem is illustrated in this triangular flow diagram. Every component is dependent on the others and none dominates the others. Regardless of the orientation of the triangle, the same relationships will hold

The overarching problem at the center of the triangle is not obvious. It is subtle. But because of its preeminence in the pantheon of problems any decision maker faces, I decided to allocate the entire first chapter to it. Spending so much space talking about information in a data analytics book may seem odd, but it is very important to understand why we do what we do, which is to analyze data to extract that rich information from data.

The theoretical understanding should be obvious. You need to know not just the methodologies but also their limitations so you can effectively apply them to solve a problem. The limitations may hinder you or just give you the wrong answers. Assume you were hired or commissioned by a business decision maker (e.g., a

CEO) to provide actionable, insightful, and useful rich information relevant for their problem. If the limitations of a methodology prevent you from accomplishing your charge, then your life as an analyst will be short-lived, to say the least. This will hold if you either do not know these limitations or simply choose to ignore them. Another methodological approach might be better, one that has fewer problems, or is just more applicable.

There is a dichotomy in methodology training. Most graduate-level statistics and econometric programs, and the newer Data Science programs, do an excellent job instructing students in the theory behind the methodologies. The focus of these academic programs is largely to train the next generation of academic professionals, not the next generation of business analytical professionals. Data Science programs, of which there are now many available online and “in person,” often skim the surface of the theoretical underpinnings since their focus is to prepare the next generation of business analysts, those who will tackle the business decision makers’ tough problems, and not the academic researchers. Something in between the academic and data science training is needed for successful business data analysts.

Data handling is not as obvious since it is infrequently taught and talked about in academic programs. In those programs, beginner students work with clean data with few problems and that are in nice, neat, and tidy data sets. They are frequently just given the data. More advanced students may be required to collect data, most often at the last phase of training for their thesis or dissertation, but these are small efforts, especially when compared to what they will have to deal with post training. The post-training work involves:

- Identifying the required data from diverse, disparate, and frequently disconnected data sources with possibly multiple definitions of the same quantitative concept
- Dealing with data dictionaries
- Dealing with samples of a very large database—how to draw the sample and determine the sample size
- Merging data from disparate sources
- Organizing data into a coherent framework appropriate for the statistical/econometric/machine learning methodology chosen
- Visualizing complex multivariate data to understand relationships, trends, patterns, and anomalies inside the data sets

This is all beyond what is provided by most training programs.

Finally, there is the programming. First, let me say that there is programming and then there is programming. The difference is scale and focus. Most people, when they hear about programming and programming languages, immediately think about large systems, especially ones needing a considerable amount of time (years?) to fully specify, develop, test, and deploy. They would be correct regarding large-scale, complex systems that handle a multitude of interconnected operations. Online ordering systems easily come to mind. Customer interfaces, inventory management, production coordination, supply chain management, price maintenance and dynamic pricing platforms, shipping and tracking, billing, and

collections are just a few components of these systems. The programming for these is complex to say the least.

As a business data analyst, you would not be involved in this type of programming although you might have to know about and access the subsystems of one or more of these larger systems. And major businesses are composed of many larger systems! You might have to write code to access the data, manipulate the retrieved data, and so forth, basically write programming code to do all the data handling I described above. And for this you need to know programming and languages.

There are many programming languages available. Only a few are needed for most business data analysis problems. In my experience, these are:

- *SQL*
- Python
- R

Julia should be included because it is growing in popularity due to its performance and ease of use. For this book, I will use Python because its ecosystem is strongly oriented toward machine learning with strong modeling, statistics, data visualization, and programming functionalities. In fact, its programming paradigm is clear to use, which is a definite advantage over other languages.

The Target Audience

The target audience for this book consists of business data analysts, data scientists, and market research professionals, or those aspiring to be any of these, in the private sector. You would be involved in or responsible for a myriad of quantitative analyses for business problems such as, but not limited to:

- Demand measurement and forecasting
- Predictive modeling
- Pricing analytics including elasticity estimation
- Customer satisfaction assessment
- Market and advertisement research
- New product development and research

To meet these tasks, you will have a need to know basic data analytical methods and some advanced methods, including data handling and management. This book will provide you with this needed background by:

- Explaining the intuition underlying analytic concepts
- Developing the mathematical and statistical analytic concepts
- Demonstrating analytical concepts using Python
- Illustrating analytical concepts with case studies

This book is also suitable for use in colleges and universities offering courses and certifications in business data analytics, data sciences, and market research. It could be used as a major or supplemental textbook.

Since the target audience consists of either current or aspiring business data analysts, it is assumed that you have or are developing a basic understanding of fundamental statistics at the “Stat 101” level: descriptive statistics, hypothesis testing, and regression analysis. Knowledge of econometric and market research principles, while not required, would be beneficial. In addition, a level of comfort with calculus and some matrix algebra is recommended, but not required. Appendices will provide you with some background as needed.

The Book’s Competitive Comparison

There are many books on the market that discuss the three themes of this book: analytic methods, data handling, and programming languages. But they do them separately as opposed to a synergistic, analytic whole. They are given separate treatment so that you must cover a wide literature just to find what is needed for a specific business problem. Also, once found, you must translate the material into business terms. This book will present the three themes so you can more easily master what is needed for your work.

The Book’s Structure

I divided this book into three parts. In Part **I**, I cover the basics of business data analytics including data handling, preprocessing, and visualization. In some instances, the basic analytic toolset is all you need to address problems raised by business executives. Part **II** is devoted to a richer set of analytic tools you should know at a minimum. These include regression modeling, time series analysis, and statistical table analysis. Part **III** extends the tools from Part **II** with more advanced methods: advanced regression modeling, classification methods, and grouping methods (*a.k.a.*, *clustering*).

The three parts lead naturally from basic principles and methods to complex methods. I illustrate this logical order in Fig. 2.

Embedded in the three parts are case study examples of business problems using (albeit, fictitious, fake, or simulated) business transactions data designed to be indicative of what business data analysts use every day. Using simulated data

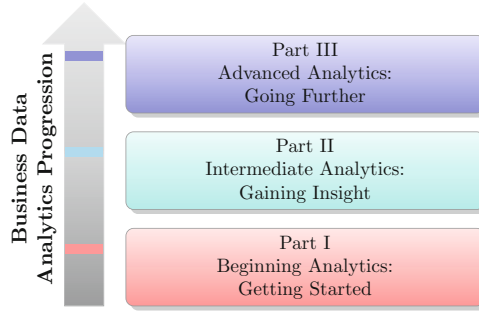


Fig. 2 This is a flow chart of the three parts of this book. The parts move progressively from basics to advanced. At the end of Part I, you should be able to do basic analyses of business data. At the end of Part II, you should be able to do regression and times series analysis. At the end of Part III, you should be able to do advanced machine learning work

for instructional purposes is certainly not without precedence. See, for example, Gelman et al. (2021). Data handling, visualization, and modeling are all illustrated using Python. All examples are in Jupyter notebooks available on Github.

Plainsboro, NJ, USA

Walter R. Paczkowski

Acknowledgments

In my last book, I noted the support and encouragement I received from my wonderful wife, Gail, and my two daughters, Kristin and Melissa, and my son-in-law, David. As before, my wife Gail encouraged me to sit down and just write, especially when I did not want to, while my daughters provided the extra set of eyes I needed to make this book perfect. They provided the same support and encouragement for this book, so I owe them a lot, both then and now. I would also like to say something about my two grandsons who, now at 5 and 9, obviously did not contribute to this book but who, I hope, will look at this one in their adult years and say “My grandpa wrote this book, too.”

Contents

Part I Beginning Analytics

1	Introduction to Business Data Analytics: Setting the Stage	3
1.1	Types of Business Problems	4
1.2	The Role of Information in Business Decision Making	5
1.3	Uncertainty vs. Risk	7
1.4	The Data-Information Nexus	9
1.4.1	Data and Information Confusion	10
1.4.2	The Data Component	10
1.4.3	The Extractor Component	15
1.4.4	The Information Component	21
1.5	Analytics Requirements	24
1.5.1	Theoretical Framework	25
1.5.2	Data Handling	27
1.5.3	Programming Literacy	28
1.5.4	Component Interconnections	30
2	Data Sources, Organization, and Structures	31
2.1	Data Dimensions: A Taxonomy for Defining Data	32
2.1.1	Taxonomy Component #1: Source	32
2.1.2	Taxonomy Component #2: Domain	38
2.1.3	Taxonomy Component #3: Levels	38
2.1.4	Taxonomy Component #4: Continuity	39
2.1.5	Taxonomy Component #5: Measurement Scale	40
2.2	Data Organization	42
2.2.1	External Database Structures	42
2.2.2	Internal Database Structures	45
2.3	Data Dictionary	55

3	Basic Data Handling	57
3.1	Case Studies	58
3.1.1	Case Study 1: Customer Transactions Data	58
3.1.2	Case Study 2: Measures of Order Fulfillment	59
3.2	Importing Your Data	61
3.2.1	Data Formats	61
3.2.2	Importing a CSV Text File into Pandas	63
3.2.3	Importing Large Files in Chunks	65
3.2.4	Checking Your Imported Data	67
3.3	Merging or Joining DataFrames	77
3.4	Reshaping DataFrames	79
3.5	Sorting a DataFrame	80
3.6	Querying a DataFrame	81
3.6.1	Boolean Operators and Indicator Functions	81
3.6.2	Pandas Query Method	83
4	Data Visualization: The Basics	85
4.1	Background for Data Visualization	85
4.2	Gestalt Principles of Visual Design	86
4.3	Issues Complicating Data Visualization	87
4.3.1	Human Visual Limitations	87
4.3.2	Data Visualization Tools	89
4.3.3	Types of Visuals	92
4.3.4	What to Look for in a Graph	92
4.4	Visualizing Spatial Data	97
4.4.1	Data Preparation	98
4.4.2	Visualizing Continuous Spatial Data	98
4.4.3	Visualizing Categorical Spatial Data	109
4.4.4	Visualizing Continuous and Categorical Spatial Data	112
4.5	Visualizing Temporal (Time Series) Data	115
4.5.1	Properties of Temporal (Time Series) Data	117
4.5.2	Visualizing Time Series Data	118
4.5.3	Times Series Complications	119
4.6	Faceted Plots	124
4.7	Appendix	126
4.7.1	Taylor Series Expansion for Growth Rates	126
5	Advanced Data Handling: Preprocessing Methods	127
5.1	Transformations	128
5.1.1	Linear Transformations	129
5.1.2	Nonlinear Transformations	136
5.1.3	A Family of Transformations	138
5.2	Encoding	141
5.2.1	Dummy or One-Hot Encoding	142
5.2.2	Patsy Encoding	146

- 5.2.3 Label Encoding 147
- 5.2.4 Binarizing Data 147
- 5.3 Dimension Reduction 150
- 5.4 Handling Missing Data 151
- 5.5 Appendix 153
 - 5.5.1 Mean and Variance of Standardized Variable 154
 - 5.5.2 Mean and Variance of Adjusted Standardized Variable... 154
 - 5.5.3 Unbiased Estimators of μ and σ^2 155

Part II Intermediate Analytics

- 6 OLS Regression: The Basics** 161
 - 6.1 Basic OLS Concept 162
 - 6.1.1 The Disturbance Term and the Residual 162
 - 6.1.2 OLS Estimation 163
 - 6.1.3 The Gauss-Markov Theorem 167
 - 6.2 Analysis of Variance 167
 - 6.3 Case Study 170
 - 6.3.1 Basic OLS Regression 170
 - 6.3.2 The Log-Log Model 170
 - 6.3.3 Model Set-up 172
 - 6.3.4 Estimation Summary 173
 - 6.3.5 ANOVA for Basic Regression 173
 - 6.3.6 Elasticities 173
 - 6.4 Basic Multiple Regression 175
 - 6.4.1 ANOVA for Multiple Regression 176
 - 6.4.2 Alternative Measures of Fit: AIC and BIC 178
 - 6.5 Case Study: Expanded Analysis 180
 - 6.6 Model Portfolio 184
 - 6.7 Predictive Analysis: Introduction 185
 - 6.7.1 Predicting vs. Forecasting 186
 - 6.7.2 Developing a Prediction 186
 - 6.7.3 Simulation Tool for Prediction Application 187
- 7 Time Series Analysis** 189
 - 7.1 Time Series Basics 189
 - 7.1.1 Time Series Definition 190
 - 7.1.2 Time Series Concepts 191
 - 7.2 Importing a Date/Time Variable 193
 - 7.3 The Data Cube and Time Series Data 193
 - 7.4 Handling Dates and Times in Python and Pandas 194
 - 7.4.1 Datetimes vs. Periods 195
 - 7.4.2 Aggregating Datetime Measures 196
 - 7.4.3 Converting Time Periods in Pandas 196
 - 7.4.4 Date-Time Mini-Language 198
 - 7.5 Some Calendrical Calculations 200

7.6	Time Series Generation Process: AR(1) Model	200
7.7	Visualization for $AR(1)$ Detection	203
7.8	Durbin-Watson Test Statistic	204
7.9	Lagged Dependent and Independent Variables	210
7.9.1	Lagged Independent Variable: $ARDL(0, 1)$	211
7.9.2	Lagged Dependent Variable: $ARDL(1, 0)$	211
7.9.3	Lagged Dependent and Independent Variables: $ARDL(1, 1)$	211
7.10	Further Exploration of Time Series Analysis	211
7.10.1	Step 1: Identification of a Model	214
7.10.2	Step 2: Estimation of the Model	219
7.10.3	Step 3: Validation of the Model	221
7.10.4	Step 4: Forecasting with the Model	222
7.11	Appendix	223
7.11.1	Backshift Operator	223
7.11.2	Useful Algebra Results	224
7.11.3	Mean and Variance of Y_t	224
7.11.4	Demeaned Data	225
7.11.5	Time Trend Addition	225
8	Statistical Tables	227
8.1	Data Preprocessing	227
8.2	Categorical Data	228
8.3	Creating a Frequency Table	229
8.4	Hypothesis Testing: A First Step	231
8.5	Cross-tabs and Hypothesis Tests	233
8.5.1	Hypothesis Testing	237
8.5.2	Plotting a Frequency Table	238
8.6	Extending the Cross-tab	245
8.7	Pivot Tables	247
8.8	Appendix	249
8.8.1	Pearson Chi-Square Statistic	249

Part III Advanced Analytics

9	Advanced Data Handling for Business Data Analytics	253
9.1	Supervised and Unsupervised Learning	253
9.2	Working with the Data Cube	255
9.3	The Data Cube and DataFrame Indexing	256
9.4	Sampling From a DataFrame	261
9.4.1	Simple Random Sampling (<i>SRS</i>)	262
9.4.2	Stratified Random Sampling	263
9.4.3	Cluster Random Sampling	264
9.5	Index Sorting of a DataFrame	264
9.6	Splitting a DataFrame: The Train-Test Splits	265
9.6.1	Model Tuning of Hyperparameters	266

- 9.6.2 Incorrect Use of Testing Data 268
- 9.6.3 Creating the Training/Testing Data Sets 269
- 9.6.4 Recombining the Data Sets 275
- 9.7 Appendix 276
 - 9.7.1 Primer on Random Numbers 276
- 10 Advanced OLS for Business Data Analytics 279**
 - 10.1 Link Functions: An Introduction 279
 - 10.2 Data Preprocessing 280
 - 10.2.1 Data Standardization for Regression Analysis 280
 - 10.2.2 One-Hot and Effects (or Sum) Encoding 282
 - 10.3 Case Study Application 284
 - 10.4 Heteroskedasticity Issues and Tests 289
 - 10.4.1 Heteroskedasticity Problem 291
 - 10.4.2 Heteroskedasticity Detection 292
 - 10.4.3 Heteroskedasticity Remedy 294
 - 10.5 Multicollinearity 296
 - 10.5.1 Digression on Multicollinearity 297
 - 10.5.2 Detection with *VIF* and the Condition Index 299
 - 10.5.3 Principal Component Regression and High-Dimensional Data 300
 - 10.6 Predictions and Scenario Analysis 301
 - 10.6.1 Making Predictions 301
 - 10.6.2 Scenario Analysis 302
 - 10.6.3 Prediction Error Analysis (*PEA*) 303
 - 10.7 Panel Data Models 309
- 11 Classification with Supervised Learning Methods 313**
 - 11.1 Case Study: Background 314
 - 11.2 Logistic Regression 314
 - 11.2.1 A Choice Interpretation 315
 - 11.2.2 Properties of this Problem 315
 - 11.2.3 A Model for the Binary Problem 316
 - 11.2.4 Case Study: Train-Test Data Split 319
 - 11.2.5 Case Study: Logit Model Training 320
 - 11.2.6 Making and Assessing Predictions 322
 - 11.2.7 Classification with a Logit Model 328
 - 11.3 K-Nearest Neighbor (*KNN*) 330
 - 11.3.1 Case Study: Predicting 333
 - 11.4 Naive Bayes 333
 - 11.4.1 Background: Bayes Theorem 333
 - 11.4.2 A General Statement 336
 - 11.4.3 The Naive Adjective: A Simplifying Assumption 337
 - 11.4.4 Distribution Assumptions 337
 - 11.4.5 Case Study: Naive Bayes Training 339

- 11.5 Decision Trees for Classification 339
 - 11.5.1 Partitioning by Constants 343
 - 11.5.2 Gini Index and Entropy 344
 - 11.5.3 Case Study: Growing a Tree 348
 - 11.5.4 Case Study: Predicting with a Tree 350
 - 11.5.5 Random Forests 351
- 11.6 Support Vector Machines 351
 - 11.6.1 Case Study: SVC Application 353
 - 11.6.2 Case Study: Prediction 353
- 11.7 Classifier Accuracy Comparison 355
- 12 Grouping with Unsupervised Learning Methods 357**
 - 12.1 Training and Testing Data Sets 358
 - 12.2 Hierarchical Clustering 359
 - 12.2.1 Forms of Hierarchical Clustering 359
 - 12.2.2 Agglomerative Algorithm Description 360
 - 12.2.3 Metrics and Linkages 361
 - 12.2.4 Preprocessing Data 362
 - 12.2.5 Case Study Application 362
 - 12.2.6 Examining More than One Solution 367
 - 12.3 K-Means Clustering 368
 - 12.3.1 Algorithm Description 368
 - 12.3.2 Case Study Application 369
 - 12.4 Mixture Model Clustering 371
- Bibliography 375**
- Index 381**

List of Figures

Fig. 1	The synergistic connection of the three components of effective data analysis for the overarching problem is illustrated in this triangular flow diagram. Every component is dependent on the others and none dominates the others. Regardless of the orientation of the triangle, the same relationships will hold	vi
Fig. 2	This is a flow chart of the three parts of this book. The parts move progressively from basics to advanced. At the end of Part I, you should be able to do basic analyses of business data. At the end of Part II, you should be able to do regression and times series analysis. At the end of Part III, you should be able to do advanced machine learning work	x
Fig. 1.1	This cost curve illustrates what happens to the cost of decisions as the amount of information increases. The Base Approximation Cost is the lowest possible cost you can achieve due to the uncertainty of all decisions. This is an amount above zero	6
Fig. 1.2	Data is the base for information which is used for decision making. The <i>Extractor</i> consists of the methodologies I will develop in this book to take you from data to information. So, behind this one block in the figure is a wealth of methods and complexities	11
Fig. 1.3	This is an example of a Data Cube illustrating the three dimensions of data for a manufacturer. As I noted in the text, more than three dimensions are possible, but only three can be visualized	13

Fig. 1.4 This is a DataFrame version of the Data Cube for the product return example. There are 288 rows. This example has a multilevel index representing the Data Cube. Each combination of the levels of three indexes is unique because each combination is a row identifier, and there can only be one identifier for each row 13

Fig. 1.5 This is a stylized Data Cube illustrating the three dimensions of data 14

Fig. 1.6 This illustrates three possible aggregations of the DataFrame in Fig. 1.4. Panel (a) is an aggregation over months; (b) is an aggregation over plants; and (c) is an aggregation over plants and products. There are six ways to aggregate over the three indexes 15

Fig. 1.7 This illustrates information about the structure of a DataFrame. The variable “supplier” is an object or text, “averagePrice” is a float, “ontime” is an integer, and “dateDelivered” is a datetime 20

Fig. 1.8 Not only does information have a quantity dimension that addresses the question “*How much information do you have?*”, but it also has a quality dimension that addresses the question “*How good is the information?*” This latter dimension is illustrated in this figure as varying degrees from Poor to Rich 23

Fig. 1.9 Cost curves for Rich Information extraction from data 25

Fig. 1.10 The synergistic connection of the three components of effective data analysis for business problems is illustrated in this triangular flow diagram. Every component is dependent on the others and none dominates the others. Regardless of the orientation of the triangle, the same relationships will hold 26

Fig. 1.11 Programming roles throughout the Deep Data Analytic process 28

Fig. 2.1 A data taxonomy. Source: Paczkowski (2016). Permission to use granted by SAS Press 33

Fig. 2.2 Measurement scales attributed to Stevens (1946). Source for this chart: Paczkowski (2016). Permission to use granted by SAS Press 41

Fig. 2.3 This is the Pandas code to create the supplier on-time DataFrame. The resulting DataFrame is shown 44

Fig. 2.4 This is the SQL code to select the on-time suppliers. The resulting DataFrame is shown. Notice that the query string, called “qry” in this example, contains the three verbs I mentioned in the text 44

Fig. 2.5 This is a simple DataFrame for state data 47

Fig. 2.6 States are categorized as technology talented or not. This shows that only 32% of the states are technology talented 48

Fig. 2.7 A two-sample t-test for a difference in the median household income for tech vs non-tech states shows that there is a statistical difference. Notice my use of the query statements 48

Fig. 2.8 This is a hierarchical structure of consumers and businesses. **(a)** Consumer structure. **(b)** Business structure 52

Fig. 2.9 This is a Python script to generate a data dictionary..... 56

Fig. 3.1 Importing a *CSV* file. The path for the data would have been previously defined as a character string, perhaps as *path = './Data/*. The file name is also a character string as shown here. The path and file name are string concatenated using the plus sign 64

Fig. 3.2 Reading a chunk of data. The chunk size is 5 records. The columns in each row in each chunk are summed 65

Fig. 3.3 Processing a chunk of data and summing the columns, but then deleting the first two columns after the summation 66

Fig. 3.4 Chunks of data are processed as in Fig. 3.3 but then concatenated into one DataFrame 66

Fig. 3.5 Display the *head()* of a DataFrame. The default is *n = 5* records. If you want to display six records, use *df.head(6)* or *df.head(n = 6)*. Display the tail with a comparable method. Note the “dot” between the “df” and “head()”. This means that the *head()* method is chained or linked to the DataFrame “df” 68

Fig. 3.6 This is a style definition for setting the font size for a DataFrame caption 68

Fig. 3.7 This is an example of using a style for a DataFrame 69

Fig. 3.8 Display the *shape* of a DataFrame. Notice that the *shape* does not take any arguments and parentheses are not needed. The shape is an attribute, not a method. This DataFrame has 730,000 records and six columns 69

Fig. 3.9 Display the column names of a DataFrame using the *columns* attribute 70

Fig. 3.10 These are some examples where an *NaN* value is ignored in the calculation 71

Fig. 3.11 These are some examples where an *NaN* value is not ignored in the calculation 71

Fig. 3.12 Two symbols are assigned an *NaN* value using Numpy’s *nan* function. The *id()* function returns the memory location of the symbol. Both are stored in the same memory location 72

Fig. 3.13 This illustrates counting missing values by the columns of a DataFrame. The top portion of the output shows the display from the *info()* method while the bottom portion shows the results from the *count()* in a DataFrame 73

Fig. 3.14 This illustrates a possible display of missing values for the four *POI* measures. The entire DataFrame was subsetted to the first 1000 records for illustrative purposes. Missing values were randomly inserted. This map visually shows that “documentation” had no missing values while “ontime” had the most 74

Fig. 3.15 This illustrates several different types of joins using Venn Diagrams. Source: Paczkowski (2016). Used with permission of SAS 78

Fig. 3.16 This illustrates merging two DataFrames on a common primary key: the variable “key.” Notice that the output DataFrame has only two records because there are only two matches of keys in the left and right DataFrames: key “A” and key “C”. The non-matches are dropped 78

Fig. 3.17 This illustrates melting a DataFrame from wide- to long-form using the final merged DataFrame from Fig. 3.16. The rows of the melted DataFrame are sorted to better show the correspondence to the DataFrame in Fig. 3.16 80

Fig. 3.18 This illustrates the unstacking of the DataFrame in Fig. 3.17 from long- to wide-form 80

Fig. 3.19 These are two example queries of the *POI* DataFrame. The first show a simple query for all records with a *FID* equal to 100. There are 1825 of them. The second show a more complex query for all records with a *FID* between 100 and 102, but excluding 102. There are 3650 records 83

Fig. 4.1 This is the structure for two figures in Matplotlib terminology. Panel (a) is a basic structure with one axis (ax) in the figure space. This is created using *fig, ax = plt.subplots()*. Panel (b) is a structure for 2 axes (ax1 and ax2) in a (1 × 2) grid. This is created using *fig, ax = plt.subplots(1, 2)*. Source: Paczkowski (2021b). Permission granted by Springer 91

Fig. 4.2 Four typical distributions are illustrated here. The top left is **left** skewed the top right is **right** skewed. The two bottom ones are symmetric. The lower right is **almost** uniform while the lower left is almost normal. The one on the lower left is the most desirable 94

Fig. 4.3 This is an example of the skewness test. This is a Z-test. A Z value less than zero indicates left skewness; greater than zero indicates right skewness. The p-value is used to test the Null Hypothesis skewness that the skewness is zero. Since the p-value is greater than 0.05, the Null Hypothesis of no skewness is not rejected 95

Fig. 4.4 This illustrates the effect of an outlier on a regression line. The left panel shows how the outlier pulls the line away from what appears to be the trend in the data. The right panel shows the effect on the line with the outlier removed 97

Fig. 4.5 This code shows how the data for the spatial analysis of the *POI* data are aggregated. This aggregation is over time for each *FID*. Aggregation is done using the *groupby* function with the mean function. Means are calculated because they are sensible for this data. The DataFrame is called *df_agg* 98

Fig. 4.6 This code shows how the data are merged. The new DataFrame is called *df_agg* 99

Fig. 4.7 Definitions of parts of a boxplot. Source: Paczkowski (2021b). Permission granted by Springer 99

Fig. 4.8 Boxplot for a single continuous variable 100

Fig. 4.9 Histogram for a single continuous variable 103

Fig. 4.10 Scatter plot for two continuous variables 104

Fig. 4.11 A contour plot of the same data used in Fig. 4.10 105

Fig. 4.12 A hex bin plot of the same data used in Fig. 4.10 106

Fig. 4.13 A scatterplot of the same data used in Fig. 4.10 but with a LOWESS smooth overlaid 108

Fig. 4.14 The same data used in Fig. 4.10 is used here to compare different extreme settings for the LOWESS span setting. The scatter points were omitted for clarity 109

Fig. 4.15 Parallel plot of the *POI* components for each of the four marketing regions. The Southern region stands out 110

Fig. 4.16 Choropleth map of mean *POI* data by U.S. states 111

Fig. 4.17 Our inability to easily decipher angles makes it challenging to determine which slice is largest for *Pie A* 112

Fig. 4.18 Bar Chart view of *Pie A* of Fig. 4.17. This is easier to read and understand. Market *B* clearly stands out 113

Fig. 4.19 Stacked bar chart 113

Fig. 4.20 Cross-tab of *POI* warning and store type 114

Fig. 4.21 *POI* mosaic graph 114

Fig. 4.22 Example of a heatmap 115

Fig. 4.23 Boxplot of a continuous variable conditioned on the levels of a categorical variable. The conditioning variable is location: Rural and Urban 115

Fig. 4.24 Faceted panel plot of a continuous variable conditioned on two categorical variables 116

Fig. 4.25 Bubble plot of *POI* by *Ontime* delivery sized by marketing region 116

Fig. 4.26 Time series classifications 117

Fig. 4.27 A single, continuous times series of annual data 119

Fig. 4.28 A single, continuous times series of annual data could be split into subperiods with a boxplot created for each subperiod 119

Fig. 4.29 A plot of the *Ontime POI* measure for the 2019–2020 subperiod. This is clearly nonstationary 120

Fig. 4.30 A first differenced plot of the monthly data in Fig. 4.29. This clearly has a constant mean so it is mean stationary as opposed to the series in Fig. 4.29 121

Fig. 4.31 This shows simulated data for an unlogged and logged versions of some data 122

Fig. 4.32 The monthly data for the document component of the *POI* measure plotted against itself lagged one period 123

Fig. 4.33 The average monthly damage *POI* data are plotted by months to show seasonality 123

Fig. 4.34 Scatter plot matrix for four continuous variables. Notice that there are 16(= 4 × 4) panels, each presenting a plot of a pair of variables 124

Fig. 4.35 Scatter plot matrix lower triangle of Fig. 4.34 125

Fig. 5.1 A randomly generated data set is standardized using (5.1.1) and (5.1.4). The means and standard deviations are calculated using Numpy functions 131

Fig. 5.2 This chart illustrates the Z-transformations in Fig. 5.1. Note the linear relationship between *X* and *Z* 132

Fig. 5.3 A randomly generated data set is standardized using the *sklearn* preprocessing package *StandardScaler*. Notice how the package is imported and the steps for the standardization. In this example, the data are first fit (i.e., the mean and standard deviation are first calculated) and then transformed by (5.1.1) using the single method *fit_transform* with the argument *df*, the DataFrame 133

Fig. 5.4 A randomly generated data set is standardized using (5.1.7) and (5.1.8) 134

Fig. 5.5 This chart illustrates the MinMax standardization in Fig. 5.4 134

Fig. 5.6 This illustrates the *sklearn* preprocessing function *MinMaxScaler* 135

Fig. 5.7 This is an example of the nonlinear transformation using (5.1.13) 137

Fig. 5.8 This is an example of the nonlinear odds transformation using (5.1.14) 138

Fig. 5.9 This illustrates the Box-Cox transformation on randomly simulated log-normal data 139

Fig. 5.10 This compares the histograms for the log-normal distribution and the Box-Cox transformation of that data 140

Fig. 5.11 This illustrates the Yeo-Johnson transformation alternative to the Box-Cox transformation. The same log-normally distributed data are used here as in Fig. 5.9 141

Fig. 5.12 This compares the histograms for the log-normal distribution, the Box-Cox transformation, and the Yeo-Johnson transformation of that data 142

Fig. 5.13 Several continuous or floating point number variables or features can be nominally encoded based on a threshold value. Values greater than the threshold are encoded as 1; 0 otherwise. In this example, the threshold is 5 148

Fig. 5.14 Several continuous or floating point number variables or features are ordinally encoded. Notice that the *fit_transform* method is used 149

Fig. 5.15 A missing value report function using the package *sidetable*. This function also relies on another function, *get_df_name* to retrieve the DataFrame name. An example report is in Fig. 5.16 152

Fig. 5.16 A missing value report function using the function in Fig. 5.15 153

Fig. 6.1 This is a comparison of the squared and absolute value of the residuals which are simulated. I used the Numpy *linspace* function to generate 1000 evenly spaced points between -5 and $+5$ with the end points included. Notice that the sum of the residuals is 0.0 165

Fig. 6.2 Panel (a) shows the raw data for unit sales of the living room blinds while Panel (b) shows the log transformed unit sales. The log transform is $\log(1 + Usales)$ to avoid any problems with zero sales. I use the Numpy log function: *log1p*. This function is the natural log by default 171

Fig. 6.3 A single variable regression is shown here. (a) Regression setup. (b) Regression results 174

Fig. 6.4 ANOVA table for the unit sales regression 174

Fig. 6.5 There calculations verify the relationship between the R^2 and the F-Statistic. I retrieved the needed values from the *regO1* object I created for the regression in Fig. 6.3 175

Fig. 6.6 A multiple variable regression is shown here. (a) Regression setup. (b) Regression results 182

Fig. 6.7 ANOVA table for the unit sales multiple regression model 182

Fig. 6.8 Correlation matrix showing very little correlation 183

Fig. 6.9 F-test showing no region effect 183

Fig. 6.10 You define the statistics to display in a portfolio using a setup like this 184

Fig. 6.11 This is the portfolio summary of the two regression models from this chapter 185

Fig. 6.12 This illustrates a framework for making predictions with a simulation tool 188

Fig. 7.1 The relationships among the four concepts are shown here 192

Fig. 7.2 The Data Cube can be collapsed by aggregating the measures for periods that were extracted from a datetime value using the accessor *dt*. Aggregation is done using the *groupby* and *aggregate* functions 193

Fig. 7.3 This function in this example, returns date as a datetime integer. This integer is the number of seconds since the Pandas epoch which is January 1, 1970. The Unix epoch is January 1, 1960 195

Fig. 7.4 These are consecutive dates, each written in a different format. Each format is a typical way to express a date. Pandas interprets each format the same way and produces the datetime value, which is the number of seconds since the epoch. The column labeled “Time Delta” is the day-to-day change. Notice that it is always 86,400 which is the number of seconds in a day 195

Fig. 7.5 The *groupby* method and the *resampling* method can be combined in this order: the rows of the DataFrame are first grouped by the *groupby* method and then each group’s time frequency is converted by the *resample* method 197

Fig. 7.6 The *groupby* method is called with an additional argument to the variable to group on. The additional argument is *Groupers* which groups by a datetime variable. This method takes two arguments: a key identifying the datetime variable and a frequency to convert to. The *Groupers* can be placed in a separate variable for convenience as I show here 198

Fig. 7.7 The *groupby* method is called with the *Groupers* specification only 198

Fig. 7.8 The furniture daily transactions data are resampled to monthly data and then averaged for the month. The rule is “M” for end-of-month, the object is *Tdate* and the aggregation is *mean* 199

Fig. 7.9 The residuals for a times series model of log unit sales on log pocket price are retrieved 203

Fig. 7.10 The residuals from Fig. 7.9 are plotted against time. A sine wave appearance is evident 204

Fig. 7.11 The residuals from Fig. 7.9 are plotted against their lagged values. Most of the points fall into the upper right quadrant suggesting positive autocorrelation based on Table 7.4. This graph can also be produced using the Pandas function *pd.plotting.lag_plot(series)* where “series” is the residual series 205

Fig. 7.12 The unit sales and pocket price data were resampled to a monthly frequency and then aggregated. The sum of sales would be zero for a particular month if there were no sales in that month. That zero value was replaced by NaN 208

Fig. 7.13 The resampled and aggregated orders data are checked for missing values. Notice that there are 21 records but 20 have non-null data 208

Fig. 7.14 The missing values are filled-in using the Pandas *Interpolate()* method 209

Fig. 7.15 The Durbin-Watson statistic is low, 1.387 209

Fig. 7.16 After the *GLS* correction, the Durbin-Watson statistic is improved only slightly to 1.399 210

Fig. 7.17 This illustrates the two time series plots instrumental in identifying a times series model. Panel (a) is an autocorrelation plot for 10 lags; (b) is a partial autocorrelation plot for the same lags. The shaded areas are the 95% confidence interval 214

Fig. 7.18 This illustrates the application of the Augmented Dickey-Fuller Test to the pocket price time series. Notice that the time series plot shows that the series varies around 1.6 on the log scale. This suggests Case II which includes a constant but no trend. The test suggests there is stationarity since the Null Hypothesis is that the series is nonstationary 220

Fig. 7.19 This illustrates the application of the *KPSS* Test to the pocket price time series. The time series plot in Fig. 7.18 suggests constant or level stationarity. The test suggests there is level stationarity 221

Fig. 7.20 The *AR(1)* model for the pocket price times series 221

Fig. 7.21 The *AR(1)* model is used to forecast the pocket price times series. In this case, I forecast 4-steps ahead, or four periods into the future 222

Fig. 7.22 These are the 4-steps ahead forecasts for the pocket prices. **(a)** Forecast values. **(b)** Forecast plot..... 223

Fig. 8.1 This illustrates the code to remap values in a DataFrame 228

Fig. 8.2 A Categorical data type is created using the *CategoricalDtype* method. In this example, a list of ordered levels for the *paymentStatus* variable is provided. The categorical specification is applied using the *astype()* method 230

Fig. 8.3 The variable with a declared categorical data type is used to create a simple frequency distribution of the recoded payment status. Notice how the levels are in a correct order so that the cumulative data make logical sense 231

Fig. 8.4 The variable with a declared categorical data type is used to create a simple frequency distribution, but this time subsetted on another variable, *region* 231

Fig. 8.5 This is the frequency table for drug stores in California. Notice that 81.2% of the drug stores in California are past due 232

Fig. 8.6 This illustrates a chi-square test comparing an observed frequency distribution and an industry standard distribution. The industry distribution is in Table 8.3. The Null Hypothesis is no difference in the two distributions. The Null is rejected at the $\alpha = 0.05$ level of significance 233

Fig. 8.7 This illustrates a basic cross-tab of two categorical variables. The payment status is the row index of the resulting tab. The argument, *margins = True* instructs the method to include the row and column margins. The sum of the row margins equals the sum of the column margins equals the sum of the cells. These sums are all equal to the sample size 234

Fig. 8.8 This illustrates a basic tab but with a third variable, “daysLate”, averaged for each combination of the levels of the index and column variables 235

Fig. 8.9 This is the Python code for interweaving a frequency table and a proportions table. There are two important steps: (1) index each table to be concatenated to identify the respective rows and (2) concatenate based on axis 0 236

Fig. 8.10 This is the result of interweaving a frequency table and a proportions table using the code in Fig. 8.9. This is sometimes more compact than having two separate tables 236

Fig. 8.11 This illustrates the Pearson Chi-Square Test using the tab in Fig. 8.7. The p-value indicates that the Null Hypothesis of independence should not be rejected. The Cramer’s V statistic is 0.0069 and supports this conclusion 239

Fig. 8.12 This illustrates a heatmap using the tab in Fig. 8.7. It is clear that the majority of Grocery stores is current in their payments 240

Fig. 8.13 This is the main function for the correspondence analysis of the cross-tab developed in Fig. 8.7. The function is instantiated with the number of dimensions and a random seed or state (i.e., 42) so that results can always be reproduced. The instantiated function is then used to fit the cross-tab 241

Fig. 8.14 The functions to assemble the pieces for the final correspondence analysis display are shown here. Having separate function makes programming more manageable. This is *modular programming* 242

Fig. 8.15 The complete final results of the correspondence analysis are shown here. Panel (a) shows the set-up function for the results and the two summary tables. Panel (b) shows the biplot 243

Fig. 8.16 This is the map for the entire nation for the bakery company 245

Fig. 8.17 The cross-tab in Fig. 8.7 is enhanced with the mean of a third variable, *days-late* 246

Fig. 8.18 The cross-tab in Fig. 8.17 can be replicated using the Pandas *groupby* function and the mean function. The values in the two approaches are the same; just the arrangement differs. This is a partial display since the final table is long 246

Fig. 8.19 The cross-tab in Fig. 8.17 is aggregated using multiple variables and aggregation methods. The *agg* method is used in this case. An aggregation dictionary has the aggregation rules and this dictionary is passed to the *agg* method 247

Fig. 8.20 The DataFrame created by a *groupby* in Fig. 8.18, which is a *long-form* arrangement, is pivoted to a *wide-form* arrangement using the Pandas *pivot* function. The DataFrame is first reindexed 248

Fig. 8.21 The *pivot_table* function is a more convenient way to pivot a DataFrame 248

Fig. 8.22 The *pivot_table* function is quite flexible for pivoting a table. This is a partial listing of an alternative pivoting of our data 249

Fig. 8.23 This illustrates the chi-square distribution for several value of k . Notice how the shape changes as k increases and begins to approach the standard normal curve 250

Fig. 9.1 There are several options for identifying duplicate index values shown here 257

Fig. 9.2 This illustrate how to convert a *DatetimeIndex* to a *PeriodIndex* 259

Fig. 9.3 Changing a *MultiIndex* to a new *MultiIndex* 260

Fig. 9.4 This is one way to query a *PeriodIndex* in a *MultiIndex*. Notice the `@`. this is used then the variable is in the environment, not in the *DataFrame*. This is the case with "x" 261

Fig. 9.5 This illustrates how to draw a stratified random sample from a *DataFrame* 263

Fig. 9.6 This illustrates how to draw a cluster random sample from a *DataFrame*. Notice that the Numpy *unique* function is used in case duplicate cluster labels are selected 264

Fig. 9.7 This schematic illustrates how to split a master data set 267

Fig. 9.8 This illustrates a general correct scheme for developing a model. A master data set is split into training and testing data sets for basic model development but the training data set is split again for validation. If the training data set itself is not split, perhaps because it is too small, then the trained model is directly tested with the testing data set. This accounts for the dashed arrows 267

Fig. 9.9 This illustrates a general incorrect scheme for developing a model. The test data are used with the trained model and if the model fails the test, it is retrained and tested again. The test data are used as part of the training process 269

Fig. 9.10 There is a linear trade-off between allocating data to the training data set and the testing data set. The more you allocate to the testing, the less is available for training 270

Fig. 9.11 As a rule-of-thumb, split your data into three-fourths training and one-fourth testing. Another is two-thirds training and one-third testing 270

Fig. 9.12 This is an example of a train-test split on simulated cross-sectional data 272

Fig. 9.13 This is an example of a train-test split on simulated time series data. Sixty monthly observations were randomly generated and then divided into one-fourth testing and three-fourths training. A time series plot shows the split and a table summarizes the split sizes 274

Fig. 9.14 This illustrates a master panel data set consisting of five cross-sectional units, each with three time periods and two measures (X and Y) for each combination. A random assignment of the cross-sectional units is shown. Notice that each unit is assigned with its entire set of time periods 275

Fig. 9.15 This illustrates how the master panel data set of Fig. 9.3 is split into the two required pieces. Notice that I set the training size parameter to 0.60 276

Fig. 9.16 This shows how to generate a random number based on the computer’s clock time. The *random* package is used 277

Fig. 9.17 This shows how to generate a random number based on a seed. I used 42 The *random* package is used 278

Fig. 9.18 This shows how to generate a random number based on seed and using the Numpy *random* package 278

Fig. 10.1 This is the code to aggregate the orders data. I had previously created a DataFrame with all the orders, customer-specific data, and marketing data 285

Fig. 10.2 This is the code to split the aggregate orders data into training and testing data sets. I used three-fourths testing and a random see of 42. Only the head of the training data are shown 285

Fig. 10.3 This is the code to set up the regression for the aggregated orders data. Notice the form for the formula statement 286

Fig. 10.4 This is the results for the regression for the aggregated orders data 287

Fig. 10.5 These are the regression results for simulated data. The two lines for the R^2 are the R^2 itself and the adjusted version 289

Fig. 10.6 Panel (a) is the unrestricted ANOVA table for simulated data and Panel (b) is the restricted version 290

Fig. 10.7 This is the manual calculation of the F-Statistic using the data in Fig. 10.6. The F-statistic here agrees with the one in Fig. 10.5 290

Fig. 10.8 This is the F-test of the two regressions I summarized in Fig. 10.5 290

Fig. 10.9 These are the signature patterns for heteroskedasticity. The residuals are randomly distributed around their mean in Panel (a); this indicates homoskedasticity. They fan out in Panel (b) as the X -axis variable increases; this indicates heteroskedasticity 293

Fig. 10.10 This is the residual plot for the residuals in Fig. 10.4 293

Fig. 10.11 These are the White Test results 295