

The Fundamentals of People Analytics

With Applications in R

Craig Starbuck

2022-10-26

Contents

Dedication	1
Foreword	3
Preface	5
Getting Started	7
Guiding Principles	8
Pro-Employee Thinking	8
Quality	8
Prioritization	9
Tooling	10
Data Sets	11
Employees	11
Turnover Trends	12
Survey Responses	13
4D Framework	13
Introduction to R	17
Getting Started	17
Installing R	17
Installing R Studio	17
Installing Packages	18
Loading data	18

Case Sensitivity	19
Help	19
Objects	19
Comments	20
Testing Early and Often	21
Vectors	21
Matrices	26
Factors	28
Data Frames	29
Lists	31
Loops	31
User-Defined Functions (UDFs)	32
Graphics	33
Review Questions	36
 Introduction to SQL	 37
Basics	37
Aggregate Functions	40
Joins	42
Subqueries	48
Virtual Tables	49
Window Functions	50
Common Table Expressions (CTEs)	53
Review Questions	54
 Research Design	 57
Research Questions	57
Research Hypotheses	57
Internal and External Validity	58
Research Methods	58
Research Designs	59
Review Questions	63

Measurement & Sampling	65
Variable Types	65
Independent Variables (IV)	65
Dependent Variables (DV)	66
Control Variables (CV)	66
Moderating Variables	66
Mediating Variables	67
Endogenous vs. Exogenous Variables	68
Measurement Scales	69
Discrete Variables	69
Continuous Variables	69
Sampling Methods	70
Probability Sampling	70
Non-Probability Sampling	75
Sampling & Nonsampling Error	76
Sampling Error	76
Nonsampling Error	78
Scale Reliability and Validity	80
Reliability	80
Validity	81
Review Questions	83
Data Preparation	85
Data Extraction	85
Data Architecture	86
Data Screening & Cleaning	89
Missingness	92
Outliers	94
Low Variability	95
Inconsistent Categories	96
Data Binning	97

One-Hot Encoding	97
Feature Engineering	100
Review Questions	101
Descriptive Statistics	103
Univariate Analysis	103
Measures of Central Tendency	103
Measures of Spread	107
Bivariate Analysis	117
Covariance	117
Correlation	119
Review Questions	126
Statistical Inference	129
Introduction to Probability	129
Probability Distributions	129
Conditional Probability	137
Central Limit Theorem	140
Confidence Intervals	144
Hypothesis Testing	150
Alpha	151
Type I & II Errors	151
p-Values	152
Bonferroni Correction	152
Statistical Power	153
Review Questions	156
Analysis of Differences	157
Parametric vs. Nonparametric Tests	157
Differences in Discrete Data	158
Differences in Continuous Data	163
Review Questions	187

Linear Regression	189
Sample Size	192
Simple Linear Regression	192
Multiple Linear Regression	202
Moderation	210
Mediation	214
Review Questions	217
Linear Model Extensions	219
Model Comparisons	219
Hierarchical Regression	221
Multilevel Models	223
Polynomial Regression	225
Review Questions	234
Logistic Regression	237
Binomial Logistic Regression	238
Multinomial Logistic Regression	243
Ordinal Logistic Regression	249
Review Questions	253
Predictive Modeling	255
Cross-Validation	257
Model Performance	258
Bias-Variance Tradeoff	260
Tree-Based Algorithms	260
Predictive Modeling	262
Classification	262
Forecasting	272
Review Questions	277

Unsupervised Learning	279
Factor Analysis	280
Exploratory Factor Analysis (EFA)	281
Confirmatory Factor Analysis (CFA)	287
Clustering	292
<i>K</i> -Means Clustering	293
Hierarchical Clustering	296
Review Questions	300
Data Visualization	301
Best Practices	301
Color Palette	301
Chart Borders	302
Zero Baseline	303
Intuitive Layout	303
Preattentive Attributes	305
Step-by-Step Visual Upgrade	306
Step 1: Build Bar Chart with Defaults	306
Step 2: Remove Legend	307
Step 3: Assign Colors Strategically	307
Step 4: Add Axis Titles and Margins	308
Step 5: Add Left-Justified Title	309
Step 6: Remove Background	310
Step 7: Remove Axis Ticks	310
Step 8: Mute Titles	312
Step 9: Flip Axes	312
Step 10: Sort Data	314
Visualization Types	314
Tables	315
Heatmaps	317
Scatterplots	317

Line Graphs	319
Slopegraphs	323
Bar Charts	324
Combination Charts	329
Waterfall Charts	330
Waffle Charts	331
Sankey Diagrams	332
Pie Charts	336
3D Visuals	339
Elegant Data Visualization	341
Review Questions	341
Data Storytelling	345
Know Your Audience	345
Production Status	346
Structural Elements	346
TL;DR	347
Purpose	347
Methodology	348
Results	349
Limitations	350
Next Steps	350
Appendix	351
Q&A	351
Review Questions	351
Bibliography	353
Appendix	357
4D Framework	357
Discover	357
Design	360

Develop	361
Deliver	362
Data Visualization	363
Step-by-Step Visual Upgrade	363
Tables	369
Heatmaps	370
Scatterplots	371
Line Charts	373
Slopegraphs	376
Bar Charts	378
Combination Charts	382
Waterfall Charts	383
Waffle Charts	385
Sankey Diagrams	386
Pie Charts	388

Dedication

To my wife and children:

Casey, Ava, and Callum

and to my parents:

John and Fonda Starbuck

for their indelible love, support, and encouragement

Foreword

Preface

Twenty years ago, I was an unlikely author of this book. My first statistics course in college was dreadful. On day one, my professor entered our lecture hall and shared some grim stats: “Based on historical data, half of you won’t make it to the midterm and of those who do, half won’t receive a passing grade in the end.” This was both discouraging and motivating. Stats was a required course for my major so failure wasn’t an option; I had to pass. I attended weekly study sessions with classmates and studied a lot independently to learn the material. I saw no applications for statistics to anything I planned to do with my degree, so the course was reduced to memorization of equations. I passed the course with a B, and I was determined to never open another stats book.

You may be wondering what changed to motivate authoring a book involving this insufferable subject. The short answer is that I discovered the very important applications to a discipline I truly love, people analytics. The practical applications were altogether absent from my undergraduate stats course. As I began to think about complex and nuanced challenges in social science contexts, it became clear that I would not only need to reengage with stats; I would need to develop an authentic appreciation for the discipline. Over the past decade, I have taken the journey of ‘relearning’ statistics and developing a deep understanding of how statistical methodologies can be applied to various organizational problem statements.

My purpose in writing this book is to help make this content – which may unfortunately be intimidating to many – both accessible and exciting. In addition to my roles in people analytics, I have taught a graduate-level business analytics course for Finance and MBA students for many years and have developed several teaching strategies through this experience that have proven successful in demystifying statistical concepts. In addition to applying these strategies, this book makes a unique contribution in curating what I consider to be the most salient topics for people analytics practitioners along with step-by-step instructions on the technical implementations. There are many texts available for deeper treatments of individual subjects covered in this book but as of this writing, none organize within a single text both theoretical and applied instruction spanning the whole of the people analytics lifecycle.

Thus, this book represents my earnest attempt to provide a concise – yet adequately comprehensive – treatment of the concepts and methods I've found to be most important for success in people analytics. My hope is that this book will ignite within you the same passion for analytics I have discovered over the past decade.

Getting Started

People analytics is the evidence-based practice of surfacing actionable insights from data to help people and organizations thrive. Relative to other functions such as Finance and Marketing, data-informed decisions in the talent domain is a recent concept. People analytics has significant potential to transform organizations by surfacing subtle barriers to success and optimizing for both the employee experience and shareholder value.

The importance of the future of work, employee well-being, candidate and employee experience, worker productivity and collaboration, diversity, equity, inclusion, and belonging (DEIB), and retention of critical talent has resulted in companies making growing investments in people analytics capabilities. The nature of workforce challenges is increasingly too complex and nuanced for traditional HR skills, and the ability to extract intelligence from workforce data to inform strategic talent decisions is critically important.

Many organizations struggle to progress beyond basic operational reporting and dashboards, but reports and dashboards are not people analytics; they merely help inform what questions to ask based on unanticipated observations (e.g., surprising changes and trajectories) which are often the impetus for people analytics projects. The ability to think critically and reason through the available data to properly frame problems and theoretical explanations are perhaps the most essential skills for success in people analytics.

This book will cut through the fluff and teach you how to do stuff. Knowledge of concepts is futile without an understanding of how to apply them to people analytics use cases. The goal of this book is not to boil the ocean by implementing an exhaustive set of analysis methods in every tool. This book is guided by a goal of optimizing for the *fewest* number of concepts and applications required to successfully design and execute the *majority* of projects in the people analytics domain. While this is a technical book, it indexes more heavily on concepts and practical applications to people analytics than on mathematical underpinnings.

Whether you are a people leader, individual contributor, or aspiring analytics practitioner, this book is for you. This book will serve as a guide through the analytics lifecycle, curating the key concepts and applications germane to

common questions and hypotheses within people analytics and providing a repeatable framework for successful analytics projects.

Guiding Principles

Among the many principles guiding how analytics teams operate, there are three that I have found to be universally applicable and fundamental to the success of an analytics capability.

Pro-Employee Thinking

With great power comes great responsibility.

‘Pro-employee’ thinking is addressed first and for good reason. People analytics has the power to improve the lives of people in meaningful ways. Whether we are shedding light on an area of the business struggling with work-life balance or identifying developmental areas of which a group of leaders may be unaware, people analytics ideally improves employee well-being and effectively, the success of the business. It is important to embrace a pro-employee philosophy, as newfound knowledge could also have damaging repercussions if shared with the wrong people or if findings are disseminated without proper instruction on how to interpret and act.

One way to error on the side of caution when considering whether to disseminate insights is to ask the following: “With this knowledge, could the recipient act in a manner that is inconsistent with our pro-employee philosophy?” If the answer to this question is not a clear *no*, discuss with your HR, legal, and privacy partners and together, determine how best to proceed. The decision may be to not share the findings with the intended audience at all or to develop a proper communication and training plan to ensure there is consistency in how recipients interpret the insights and act in response. Employment Law and Data Privacy Counsel are our friends, and it is important to build strong relationships with these critical partners.

Quality

Garbage in, garbage out.

Never compromise quality for greater velocity. If quality falls to the bottom of the priority list, all other efforts are pointless. It is unlikely that requestors of data and analytics will ever ask us to take longer to prepare the information. The onus is on us as analytics professionals to level set on a reasonable timeline for analyses based on many factors that can impact the quality of analyses and insights. A single instance of compromised quality can have lasting damage

on the reputation of the analytics function and cause consumers of insights to view all findings as suspect. Be sure quality is consistently a top value and guard your team's reputation at all costs. If stakeholders lose trust, there will likely be additional data requests for validation; this is wasteful to both you and your user community and detracts from the bigger story that needs to be communicated.

Trustworthy results are highly dependent on the quality of data in source systems. If tight controls do not exist within source applications to support data integrity, downstream data cleaning efforts can only go so far in delivering reliable and valid findings. It is often the analysts who identify data integrity issues due to the nature of their work; therefore, close relationships should be formed with source application owners to put into place validation rules to proactively prevent the entry of erroneous data or at the very least, exception/audit reports to identify and address the issues once they are observed.

Prioritization

If everything is a priority, nothing is a priority.

Analyses should always have a strong value proposition – a clear expectation of how an analysis will support a General Manager, People Partner, Salesperson, or other member of the organization. Nothing in this book will increase the value of an analysis no one needs. There should be a clear business need and commitment to action before implementing an analysis. While curiosity is important, it is not a justification for an analysis.

It is crucial to be relentless about prioritizing strategically important projects with ‘measurable’ impact over merely interesting questions that few care to answer. According to the Pareto Principle, 80% of outcomes (or outputs) result from 20% of causes (or inputs). In analytics, it is important to be laser focused on identifying the 20% of inputs that will result in disproportionate value creation for stakeholders. There are some general customer-oriented questions I have found to be helpful for the intake process to optimize the allocation of time and resources:

1. Does this support a company or departmental objective? If not, why should this be prioritized over something else?
2. Who is the executive sponsor? Really important projects will have an executive-level sponsor.
3. What quantitative and/or qualitative data can be provided as a rationale for this request? Is there data to support doing this, or is the problem statement rooted merely in theories and anecdotes?
4. Will this mitigate risk or enable opportunities?
5. What actions can or will be taken as a result of this analysis?
6. What is the *scale* of impact (# of impacted people)?

7. What is the *depth* of impact (minimum -> significant)?
8. Is this a dependency or blocker for another important deliverable?
9. What is the impact of not doing (or delaying) this?
10. What is the request date? Is there flexibility in this date and/or scope of the request (e.g., what does MVP look like)?

These questions can be weighted and scored as well to support a more automated and algorithmic approach to prioritization.

Tooling

Applications in this book are demonstrated in R, which is open-sourced statistical and data visualization software that can be downloaded free of charge. It is incredibly powerful, and there is a package (or at least the ability to easily create one) for every conceivable statistical method and data visualization. R is also widely used in highly regulated environments (e.g., clinical trials).

As of this writing, R Markdown – the dynamic document creator in which this book is written – allows for coding in 56 different languages! Therefore, debating whether to use Python, Julia, or other software is unproductive; we need not sacrifice the advantages of other languages by choosing one. All code has been written such that it is fully reproducible should you choose to implement the analysis methods on your machine (and I highly recommend you do).

Software such as R allows analysts to organize and annotate steps of the analytical process in a manner that is both logical and reproducible. End-to-end analytics workflows (data extraction -> wrangling -> cleaning -> analysis -> visualization) can be fully automated and executed in R without opening the script, mitigating the risk of inadvertently modifying data or formulas. This is one of the many reasons tools like Excel will not be covered in this book.

Please note that while R basics are covered, this is not a book on how to code. An introductory programming course is highly recommended, and this is one of the best investments you can make for a successful career in analytics. The ability to write code is now table stakes for anyone in an analytics-oriented field, as this is the best way to develop reproducible analyses. Coding is to analytics professionals what typing was for Baby Boomers decades ago; a lack of coding proficiency is a major limiting factor on one's potential in this field.

The goal of the code provided in this book is not to represent the most performant, succinct, or productionizable approaches. The code herein is intended only to facilitate understanding and demonstrate how concepts can be implemented in people analytics settings. The most performant approaches are often at odds with more intuitive alternatives. Programming expertise is important for optimizing these approaches for production applications.

Data Sets

Several data sets are leveraged throughout this book to demonstrate how to implement various analysis methods. All data sets are available in the R package named `peopleanalytics`. Instructions on installing this package and loading the data sets will be covered in Chapter @ref(r-intro).

Employees

The primary data set used in this book is `employees`, which contains information on active and terminated employees. Fields are defined in the data dictionary provided below:

- `employee_id`: Unique identifier for each employee
- `active`: Flag set to *Yes* for active employees and *No* for inactive employees
- `stock_opt_lvl`: Stock option level
- `trainings`: Number of trainings completed within the past year
- `age`: Employee age in years
- `commute_dist`: Commute distance in miles
- `ed_lvl`: Education level, where 1 = *High School*, 2 = *Associate Degree*, 3 = *Bachelor's Degree*, 4 = *Master's Degree*, and 5 = *Doctoral Degree*
- `ed_field`: Education field associated with most recent degree
- `gender`: Gender self-identification
- `marital_sts`: Marital status
- `dept`: Department of which an employee is a member
- `engagement`: Employee engagement score measured on a 4-point Likert scale, where 1 = *Highly Disengaged* and 4 = *Highly Engaged*
- `job_lvl`: Job level, where 1 = *Junior* and 5 = *Senior*
- `job_title`: Job title
- `overtime`: Flag set to *Yes* if the employee is nonexempt and works overtime and *No* if the employee does not work overtime
- `business_travel`: Business travel frequency
- `hourly_rate`: Hourly rate calculated irrespective of hourly/salaried employees
- `daily_comp`: Hourly rate * 8
- `monthly_comp`: Hourly rate * 2080 / 12
- `annual_comp`: Hourly rate * 2080
- `ytd_leads`: Year-to-date (YTD) number of leads generated for employees in Sales Executive and Sales Representative positions
- `ytd_sales`: Year-to-date (YTD) sales measured in USD for employees in Sales Executive and Sales Representative positions
- `standard_hrs`: Expected working hours over a two-week payroll cycle
- `salary_hike_pct`: The percent increase in salary for the employee's most recent compensation adjustment (whether due to a standard merit increase, off-cycle adjustment, or promotion)

- **perf_rating:** Most recent performance rating, where 1 = *Needs Improvement*, 2 = *Core Contributor*, 3 = *Noteworthy*, and 4 = *Exceptional*
- **prior_emplr_cnt:** Number of prior employers
- **env_sat:** Environment satisfaction score measured on a 4-point Likert scale, where 1 = *Highly Dissatisfied* and 4 = *Highly Satisfied*
- **job_sat:** Job satisfaction score measured on a 4-point Likert scale, where 1 = *Highly Dissatisfied* and 4 = *Highly Satisfied*
- **rel_sat:** Colleague relationship satisfaction score measured on a 4-point Likert scale, where 1 = *Highly Dissatisfied* and 4 = *Highly Satisfied*
- **wl_balance:** Work-life balance score measured on a 4-point Likert scale, where 1 = *Poor Balance* and 4 = *Excellent Balance*
- **work_exp:** Total years of work experience
- **org_tenure:** Years at current company
- **job_tenure:** Years in current job
- **last_promo:** Years since last promotion
- **mgr_tenure:** Years under current manager
- **interview_rating:** Average rating across the interview loop for the onsite stage of the employee's recruiting process, where 1 = *Definitely Not* and 5 = *Definitely Yes*

Most of these fields have also been broken into separate topical data sets to support data wrangling examples in Chapter @ref(sql-intro): **benefits**, **demographics**, **job**, **payroll**, **performance**, **prior_employment**, **status**, **survey_response**, and **tenure**.

Turnover Trends

The **turnover_trends** data set contains trailing 12-month turnover rates for each month across a five-year period. Fields are defined in the data dictionary provided below:

- **year:** Integer representing the year, which ranges from 1 (earliest) to 5 (most recent)
- **month:** Integer representing the month, which ranges from 1 (January) to 12 (December)
- **job:** Job title
- **level:** Job level, where 1 = *Junior* and 5 = *Senior*
- **remote:** Flag set to *Yes* for a remote worker and *No* for a non-remote worker
- **turnover_rate:** monthly turnover rate, calculated by dividing the termination count into the average headcount (beginning headcount + ending headcount / 2) for the respective month

Survey Responses

The `survey_responses` data set contains responses to various survey items. Each observation represents a unique anonymized survey respondent.

- `belong`: Belonging score measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `effort`: Discretionary Effort score measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `incl`: Inclusion score measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `eng_1`: Engagement score on item 1 of 3 measured on a 5-point Likert scale, where 1 = *Highly Disengaged* and 5 = *Highly Engaged*
- `eng_2`: Engagement score on item 2 of 3 measured on a 5-point Likert scale, where 1 = *Highly Disengaged* and 5 = *Highly Engaged*
- `eng_3`: Engagement score on item 3 of 3 measured on a 5-point Likert scale, where 1 = *Highly Disengaged* and 5 = *Highly Engaged*
- `happ`: Happiness score measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `psafety`: Psychological Safety score measured on a 7-point Likert scale, where 1 = *Highly Unfavorable* and 7 = *Highly Favorable*
- `ret_1`: Retention score on item 1 of 3 measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `ret_2`: Retention score on item 2 of 3 measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `ret_3`: Retention score on item 3 of 3 measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*
- `ldrshp`: Senior Leadership score measured on a 5-point Likert scale, where 1 = *Highly Unfavorable* and 5 = *Highly Favorable*

4D Framework

In practical analytics settings, we generally operate with respect to five primary constraints: timeliness, client expectation, accuracy, reliability, and cost (Bartlett, 2013). Adherence to a lightweight framework over hastily rushing into an analysis full of assumptions generally lends to better outcomes that respect these constraints. A framework ensures (a) the problem statement is understood and well-defined; (b) relevant literature and prior research are reviewed; (c) the measurement strategy is sound; (d) the analysis approach is suitable for the hypotheses being tested; and (e) results and conclusions are valid and communicated in a way that resonates with the target audience. This chapter will outline a recommended framework as well as other important considerations that should be reviewed early in the project.

It is important to develop a clear understanding of the key elements of research. Scientific research is the systematic, controlled, empirical, and critical investigation of natural phenomena guided by theory and hypotheses about the presumed relations among such phenomena (Kerlinger & Lee, 2000). In other words, research is an organized and systematic way of finding answers to questions. If you are in the business of analytics, I encourage you to think of yourself as a research scientist – regardless of whether you are wearing a lab coat or have plans to publish.

As we will discover when exploring the laws of probability in a later chapter, there is a 1 in 20 chance of finding a significant result when none exists. Therefore, it is important to remain disciplined and methodical to protect against backward research wherein the researcher mines data for interesting relationships or differences and then develops hypotheses which they know the data support. There have been many examples of bad research over the years, which often presents in the form of p-hacking or data dredging: the act of finding data to confirm what the researcher wants to prove. This can occur by running an exhaustive number of experiments to find one that supports the hypothesis, or by using only a subset of data which features the expected patterning.

Academics at elite research institutions are often under immense pressure to publish in top-tier journals that have a track record of accepting new groundbreaking research over replication studies or unsupported hypotheses, and incentives have unfortunately influenced some to compromise integrity. As my PhD advisor told me many years ago, an unsupported hypothesis – while initially disappointing given the exhaustive literature review that precedes its development – is a meaningful empirical contribution given theory suggests the opposite should be true.

If you participated in a science fair as a child, you are likely already familiar with the scientific method. The scientific method is the standard scheme of organized and systematic inquiry, and this duly applies to people analytics practitioners in the promotion of robust analyses and recommendations.

An important feature of the Scientific Method, as reflected in Figure @ref(fig:sci-method), is that the process never ends. New knowledge resulting from hypothesis testing prompts a new set of questions and hypotheses, which initiates a new lifecycle of scientific inquiry.

Over the years, I have adapted the scientific method into a curtailed four-step framework to promote a rigorous and disciplined approach to the end-to-end analytical process. This framework is summarized in Figure @ref(fig:4d-framework), and the four steps are (a) Discover, (b) Design, (c) Develop, and (d) Deliver.

The 4D framework, for which there is a detailed checklist in Chapter @ref(appendix), provides a method of structuring analytics projects to ensure they are anchored in well-defined problem statements and proactively consider

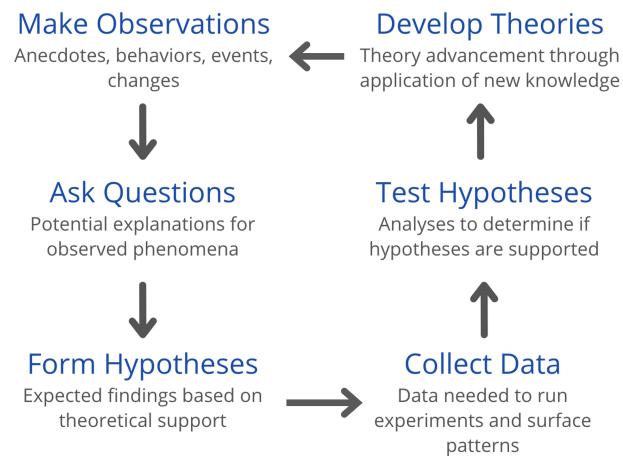


Figure 1: The Scientific Method

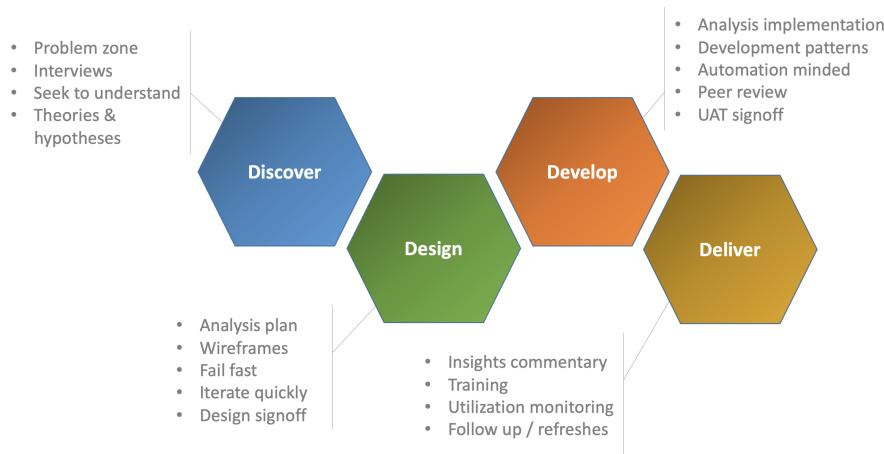


Figure 2: 4D Framework

the key elements at each stage of the analytics lifecycle to increase the likelihood of meaningful ROI.

Introduction to R

This chapter covers how to install R, R Studio, and required packages for replicating examples in this book. This chapter also covers R basics such as objects, data structures, and data types that are fundamental to working in R. In addition, many common functions will be covered in this chapter, and many more will be introduced throughout later chapters.

Getting Started

This section will cover some fundamental concepts and best practices for working in R.

Installing R

R can be compiled and ran on a variety of platforms including UNIX, Windows, and MacOS. R can be downloaded here: <https://www.r-project.org/>

When installing R, you will need to select a CRAN mirror. The **Comprehensive R Archive Network (CRAN)** is a network of servers around the world that store identical, current versions of code and documentation for R. You should select the CRAN mirror nearest you to minimize network load.

Installing R Studio

R Studio, which is being rebranded **Posit** to reflect the scientific research it endeavors to support (i.e., testing *posited* hypotheses), is an **Integrated Development Environment (IDE)** for R. This IDE provides a console with syntax editing that is helpful for debugging code as well as tools for plotting, history, and workspace management. Both open source and commercial editions are available, but the open-source option is sufficient for replicating everything in this book.

R Studio can be downloaded here: <https://www.rstudio.com/products/rstudio/download/#download>

Installing Packages

This book will utilize libraries from many R packages, and all are available on CRAN. The line of code below can be executed within either the R console or IDE to install all at once:

```
install.packages(c("peopleanalytics", "tidyverse", "corrplot",
←  "psych", "moments", "ggpubr", "ggdist", "GGally",
←  "networkD3", "sqldf", "caret", "car", "reshape2", "effsize",
←  "lmtest", "equatiomatic", "pwr", "nnet", "MASS", "brant",
←  "lme4", "lmerTest", "rpart", "rpart.plot", "lavaan",
←  "lavaanPlot", "factoextra", "cluster"), dependencies = TRUE,
←  repos = "http://cran.us.r-project.org")
```

Loading data

To load the data sets for this book from the `peopleanalytics` package, we need to load the library using the `library()` function and then load the data using the `data()` function.

```
# Load library
library(peopleanalytics)

# Load data set named "employees"
data("employees")
```

To view a list of available data sets, execute `data(package = "peopleanalytics")`.

Files stored locally, or hosted on an online service such as GitHub, can be imported into R using the `read.table()` function. For example, the following line of code will import a comma-separated values file named `employees.csv` containing a header record (row with column names) from a specified GitHub directory, and then store the data in a R object named `data`:

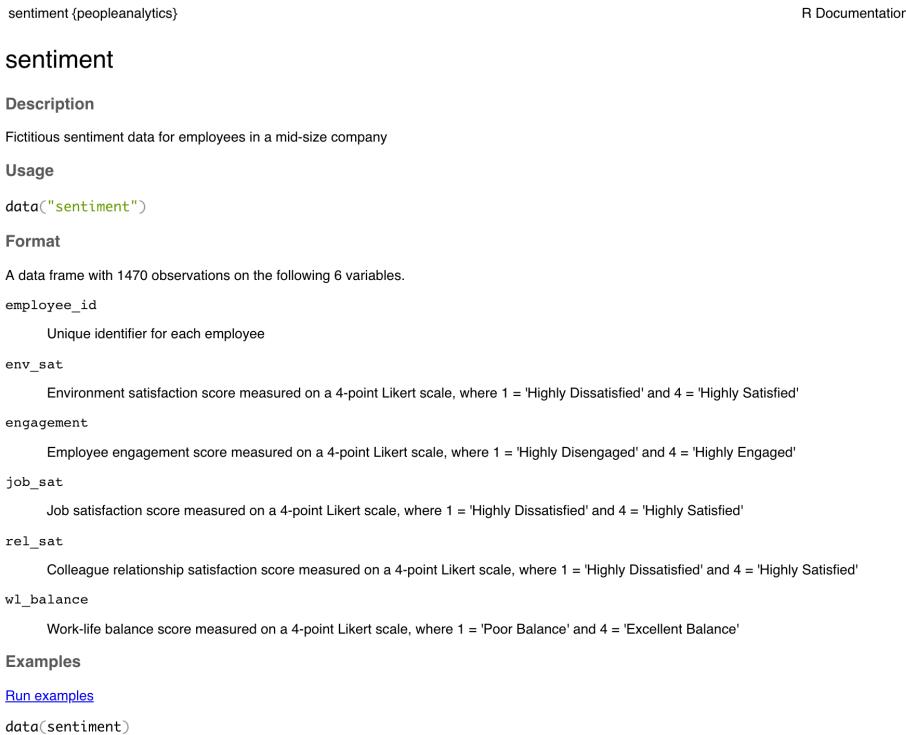
```
# Load data from GitHub file
data <- read.table(file =
←  'https://raw.githubusercontent.com/crstarbuck/peopleanalytics_book/master/data/emp
←  header = TRUE, sep = ",")
```

Case Sensitivity

It's important to note that everything in R is case-sensitive. When working with functions, be sure to match the case when typing the function name. For example, `Mean()` is not the same as `mean()`; since `mean()` is the correct case for the function, capitalized characters will result in an error when executing the function.

Help

Documentation for functions and data is available via the `?` command or `help()` function. For example, `?sentiment` or `help(sentiment)` will display the available documentation for the `sentiment` data set, as shown in @ref(fig:r-help).



The screenshot shows the R documentation for the `sentiment` data set. At the top, it says "sentiment {peopleanalytics}" and "R Documentation". Below that, the title is "sentiment". It includes sections for "Description" (Fictitious sentiment data for employees in a mid-size company), "Usage" (`data("sentiment")`), "Format" (A data frame with 1470 observations on the following 6 variables), and "Examples" (with a link to "Run examples"). Under "Format", there are detailed descriptions for each variable: `employee_id` (Unique identifier for each employee), `env_sat` (Environment satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'), `engagement` (Employee engagement score measured on a 4-point Likert scale, where 1 = 'Highly Disengaged' and 4 = 'Highly Engaged'), `job_sat` (Job satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'), `rel_sat` (Colleague relationship satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'), and `wl_balance` (Work-life balance score measured on a 4-point Likert scale, where 1 = 'Poor Balance' and 4 = 'Excellent Balance').

```

sentiment {peopleanalytics}
R Documentation

sentiment

Description
Fictitious sentiment data for employees in a mid-size company

Usage
data("sentiment")

Format
A data frame with 1470 observations on the following 6 variables.

employee_id
    Unique identifier for each employee

env_sat
    Environment satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'

engagement
    Employee engagement score measured on a 4-point Likert scale, where 1 = 'Highly Disengaged' and 4 = 'Highly Engaged'

job_sat
    Job satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'

rel_sat
    Colleague relationship satisfaction score measured on a 4-point Likert scale, where 1 = 'Highly Dissatisfied' and 4 = 'Highly Satisfied'

wl_balance
    Work-life balance score measured on a 4-point Likert scale, where 1 = 'Poor Balance' and 4 = 'Excellent Balance'

Examples
Run examples

data(sentiment)

```

Figure 3: R documentation for sentiment data set

Objects

Objects underpin just about everything we do in R. An object is a container for various types of data. Objects can take many forms, ranging from simple

objects holding a single number or character to complex structures that support advanced visualizations. The assignment operator `<-` is used to assign values to an object, though `=` serves the same function.

Let's use the assignment operator to assign a number and character to separate objects. Note that non-numeric values must be enveloped in either single ticks `' '` or double quotes `""`:

```
obj_1 <- 1
obj_1
```

```
## [1] 1
```

```
obj_2 <- 'a'
obj_2
```

```
## [1] "a"
```

Several functions are available for returning the type of data in an object, such as `typeof()` and `class()`:

```
typeof(obj_2)
```

```
## [1] "character"
```

```
class(obj_2)
```

```
## [1] "character"
```

Comments

The `#` symbol is used for commenting/annotating code. Everything on a line that follows `#` is treated as commentary rather than code to execute. This is a best practice to aid in quickly and easily deciphering the role of each line or block of code. Without comments, troubleshooting large scripts can be a more challenging task.

```
# Assign a new number to the object named obj_1
obj_1 <- 2

# Display value in obj_1
obj_1
```

```
## [1] 2

# Assign a new character to the object named obj_2
obj_2 <- 'b'

# Display value in obj_2
obj_2

## [1] "b"
```

In-line code comments can also be added where needed to reduce the number of lines in a script:

```
# Assign a new number to the object named obj_1
obj_1 <- 3

obj_1 # Display value in obj_1
```

```
## [1] 3

# Assign a new character to the object named obj_2
obj_2 <- 'c'

obj_2 # Display value in obj_2

## [1] "c"
```

Testing Early and Often

A best practice in coding is to run and test your code frequently. Writing many lines of code before testing will make debugging issues far more difficult and time-intensive than it needs to be.

Vectors

A **vector** is the most basic data object in R. Vectors contain a collection of data elements of the same data type, which we will denote as x_1, x_2, \dots, x_n in this book, where n is the number of observations or length of the vector. A vector may contain a series of numbers, set of characters, collection of dates, or logical TRUE or FALSE results.

In this example, we introduce the combine function `c()`, which allows us to fill an object with more than one value:

```
# Create and fill a numeric vector named vect_num
vect_num <- c(2,4,6,8,10)
```

```
vect_num
```

```
## [1] 2 4 6 8 10
```

```
# Create and fill a character vector named vect_char
vect_char <- c('a','b','c')
```

```
vect_char
```

```
## [1] "a" "b" "c"
```

We can use the `as.Date()` function to convert character strings containing dates to an actual date data type. By default, anything within single ticks or double quotes is treated as a character, so we must make an **explicit type conversion** from characters to dates. Remember that R is case-sensitive. Therefore, `as.date()` is not a valid function; the D must be capitalized.

```
# Create and fill a date vector named vect_dt
vect_dt <- c(as.Date("2021-01-01"), as.Date("2022-01-01"))
```

```
vect_dt
```

```
## [1] "2021-01-01" "2022-01-01"
```

We can use the sequence generation function `seq()` to fill values between a start and end point using a specified interval. For example, we can fill `vect_dt` with the first day of each month between 2021-01-01 and 2022-01-01 via the `seq()` function and its `by = "months"` argument:

```
# Create and fill a date vector named vect_dt
vect_dt <- seq(as.Date("2021-01-01"), as.Date("2022-01-01"), by =
  'months')
```

```
vect_dt
```

```
## [1] "2021-01-01" "2021-02-01" "2021-03-01" "2021-04-01" "2021-05-01"
## [6] "2021-06-01" "2021-07-01" "2021-08-01" "2021-09-01" "2021-10-01"
## [11] "2021-11-01" "2021-12-01" "2022-01-01"
```

We can also use the `:` operator to fill integers between a starting and ending number:

```
# Create and fill a numeric vector with values between 1 and 10
vect_num <- 1:10
```

```
vect_num
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

We can access a particular element of a vector using its index. An **index** represents the position in a set of elements. In R, the first element of a vector has an index of 1, and the final element of a vector has an index equal to the vector's length. The index is specified using square brackets, such as `[5]` for the fifth element of a vector.

```
# Return the value in position 5 of vect_num
vect_num[5]
```

```
## [1] 5
```

When applied to a vector, the `length()` function returns the number of elements in the vector, and this can be used to dynamically return the last value of vectors.

```
# Return the last element of vect_num
vect_num[length(vect_num)]
```

```
## [1] 10
```

Vectorized Operations

Vectorized operations (or vectorization) underpin mathematical operations in R and greatly simplify computation. For example, if we need to perform a mathematical operation to each data element in a numeric vector, we do not need to specify each and every element explicitly. We can simply apply the operation at the vector level, and the operation will be applied to each of the vector's individual elements.

```
# Create a numeric vector named x and fill with values between 1
→ and 10
```

```
x <- 1:10
```

```
# Add 2 to each element of x
```

```
x_plus2 <- x+2
```

```
x_plus2
```

```
## [1] 3 4 5 6 7 8 9 10 11 12
```

```
# Multiply each element of x by 2
```

```
x_times2 <- x*2
```

```
x_times2
```

```
## [1] 2 4 6 8 10 12 14 16 18 20
```

```
# Square each element of x
```

```
x_sq <- x^2
```

```
x_sq
```

```
## [1] 1 4 9 16 25 36 49 64 81 100
```

Many built-in arithmetic functions are available and compatible with vectors:

```
# Aggregate sum of x elements
```

```
sum(x)
```

```
## [1] 55
```

```
# Count of x elements
```

```
length(x)
```

```
## [1] 10
```

```
# Square root of x elements
sqrt(x)

## [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
## [9] 3.000000 3.162278

# Natural logarithm of x elements
log(x)

## [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
## [8] 2.0794415 2.1972246 2.3025851

# Exponential of x elements
exp(x)

## [1] 2.718282 7.389056 20.085537 54.598150 148.413159
## [6] 403.428793 1096.633158 2980.957987 8103.083928 22026.465795
```

We can also perform various operations on multiple vectors. Vectorization will result in an implied ordering of elements, in that the specified operation will be applied to the first elements of the vectors and then the second, then third, etc.

```
# Create vectors x1 and x2 and fill with integers
x1 <- 1:10
x2 <- 11:20

# Store sum of vectors to new x3 vector
x3 <- x1 + x2

x3
```

```
## [1] 12 14 16 18 20 22 24 26 28 30
```

Using vectorization, we can also evaluate whether a specified condition is true or false for each element in a vector:

```
# Evaluate whether each element of x is less than 6, and store
→ results to a logical vector
logical_rslts <- x<6

logical_rslts

## [1] TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
```

Matrices

A **matrix** is like a vector in that it represents a collection of data elements of the same data type; however, the elements of a matrix are *arranged into a fixed number of rows and columns*.

We can create a matrix using the `matrix()` function. Per `?matrix`, the `nrow` and `ncol` arguments can be used to organize like data elements into a specified number of rows and columns.

```
# Create and fill matrix with numbers
mtrx_num <- matrix(data = 1:10, nrow = 5, ncol = 2)

mtrx_num
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

We can leverage shorthand for function calls. As long as the argument values are in the correct order per the documentation, the argument names are not required. Per `?matrix`, the first argument is `data`, followed by `nrow` and then `ncol`. Therefore, we can achieve the same result using the following:

```
# Create and fill matrix with numbers
mtrx_num <- matrix(1:10, 5, 2)

mtrx_num
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

Several functions are available to quickly retrieve the number of rows and columns in a rectangular object like a matrix:

```
# Return the number of rows in mtrx_num
nrow(mtrx_num)

## [1] 5

# Return the number of columns in mtrx_num
ncol(mtrx_num)

## [1] 2

# Return the number of columns and rows in mtrx_num
dim(mtrx_num)

## [1] 5 2
```

The `head()` and `tail()` functions return the first and last pieces of data, respectively. For large matrices (or other types of objects), this can be helpful for previewing the data:

```
# Return the first five rows of a matrix containing 1,000 rows
→ and 10 columns
head(matrix(1:10000, 1000, 10), 5)

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1 1001 2001 3001 4001 5001 6001 7001 8001 9001
## [2,]    2 1002 2002 3002 4002 5002 6002 7002 8002 9002
## [3,]    3 1003 2003 3003 4003 5003 6003 7003 8003 9003
## [4,]    4 1004 2004 3004 4004 5004 6004 7004 8004 9004
## [5,]    5 1005 2005 3005 4005 5005 6005 7005 8005 9005

# Return the last five rows of a matrix containing 1,000 rows and
→ 10 columns
tail(matrix(1:10000, 1000, 10), 5)

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [996,]   996 1996 2996 3996 4996 5996 6996 7996 8996 9996
## [997,]   997 1997 2997 3997 4997 5997 6997 7997 8997 9997
## [998,]   998 1998 2998 3998 4998 5998 6998 7998 8998 9998
## [999,]   999 1999 2999 3999 4999 5999 6999 7999 8999 9999
## [1000,]  1000 2000 3000 4000 5000 6000 7000 8000 9000 10000
```

Using vectorization, we can easily perform matrix multiplication.

```
# Create 3x3 matrix
matrix(1:9, 3, 3)

##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9

# Multiply each matrix value by 2
matrix(1:9, 3, 3) * 2

##      [,1] [,2] [,3]
## [1,]    2    8   14
## [2,]    4   10   16
## [3,]    6   12   18
```

Factors

A **factor** is a data structure containing a finite number of categorical values. Each categorical value of a factor is known as a **level**, and the levels can be either ordered or unordered. This data structure is a requirement for several statistical models we will cover in later chapters.

We can create a factor using the `factor()` function:

```
# Create and fill factor with unordered categories
education <- factor(c("undergraduate", "post-graduate",
→ "graduate"))

education

## [1] undergraduate post-graduate graduate
## Levels: graduate post-graduate undergraduate
```

Since `education` has an inherent ordering, we can use the `ordered` and `levels` arguments of the `factor()` function to order the categories:

```
# Create and fill factor with unordered categories
education <- factor(education, ordered = TRUE, levels =
  c("undergraduate", "graduate", "post-graduate"))

education

## [1] undergraduate post-graduate graduate
## Levels: undergraduate < graduate < post-graduate
```

The ordering of factors is critical to a correct interpretation of statistical model output as we will cover later.

Data Frames

A *data frame* is like a matrix in that it organizes elements within rows and columns but unlike matrices, data frames can store multiple types of data. Data frames are often the most appropriate data structures for the data required in people analytics.

A data frame can be created using the `data.frame()` function:

```
# Create three vectors containing integers (x), characters (y),
  and dates (z)
x <- 1:10
y <- c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j')
z <- seq(as.Date("2021-01-01"), as.Date("2021-10-01"), by =
  'months')

# Create a data frame with 3 columns (vectors x, y, and z) and 10
  rows
df <- data.frame(x, y, z)

df

##      x y       z
## 1    1 a 2021-01-01
## 2    2 b 2021-02-01
## 3    3 c 2021-03-01
## 4    4 d 2021-04-01
## 5    5 e 2021-05-01
## 6    6 f 2021-06-01
## 7    7 g 2021-07-01
```

```
## 8   8 h 2021-08-01
## 9   9 i 2021-09-01
## 10 10 j 2021-10-01
```

The structure of an object can be viewed using the `str()` function:

```
# Describe the structure of df
str(df)
```

```
## 'data.frame':   10 obs. of  3 variables:
## $ x: int  1 2 3 4 5 6 7 8 9 10
## $ y: chr  "a" "b" "c" "d" ...
## $ z: Date, format: "2021-01-01" "2021-02-01" ...
```

Specific columns in the data frame can be referenced using the infix operator `$` between the data frame and column names:

```
# Return data in column x in df
df$x
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

Another method that allows for efficient subsetting of rows and/or columns is the `subset()` function. The example below illustrates how to subset `df` using conditions on `x` and `y` while only displaying `z` in the output. The logical operator `|` is used for OR conditions, while `&` is the logical operator for AND conditions:

```
# Return z values for rows where x is at least 7 OR y is a, b, or
# ~ c.
subset(df, x >= 7 | y %in% c('a', 'b', 'c'), select = z)
```

```
##           z
## 1 2021-01-01
## 2 2021-02-01
## 3 2021-03-01
## 7 2021-07-01
## 8 2021-08-01
## 9 2021-09-01
## 10 2021-10-01
```

Lists

Lists are versatile objects that can contain elements with different types and lengths. The elements of a list can be vectors, matrices, data frames, functions, or even other lists.

A list can be created using the `list()` function:

```
# Store vectors x, y, and z as well as df to a list
lst <- list(x, y, z, df)

str(lst)

## List of 4
## $ : int [1:10] 1 2 3 4 5 6 7 8 9 10
## $ : chr [1:10] "a" "b" "c" "d" ...
## $ : Date[1:10], format: "2021-01-01" "2021-02-01" ...
## $ :'data.frame':   10 obs. of  3 variables:
##   ..$ x: int [1:10] 1 2 3 4 5 6 7 8 9 10
##   ..$ y: chr [1:10] "a" "b" "c" "d" ...
##   ..$ z: Date[1:10], format: "2021-01-01" "2021-02-01" ...
```

Unlike vectors, accessing elements of a list requires double brackets, such as `[[3]]` for the third element.

```
# Return data from the third element of lst
lst[[3]]

## [1] "2021-01-01" "2021-02-01" "2021-03-01" "2021-04-01" "2021-05-01"
## [6] "2021-06-01" "2021-07-01" "2021-08-01" "2021-09-01" "2021-10-01"
```

Loops

In many cases, the need arises to perform an operation a variable number of times. Loops are available to avoid the cumbersome task of writing the same statement many times. The two most common types of loops are `while` and `for` loops.

Let's use a `while` loop to square integers between 1 and 5:

```
# Initialize variable
i <- 1
```

```

# Using a 'while' loop, square the values 1 through 5 and print
→ results to the screen
# 'i' is a variable that takes on a value between 1 and 5 for the
→ respective loop
while (i < 6) {

  print(i^2)
  i <- i + 1 # increment i by 1

}

## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25

```

With a **while** loop, we needed to initialize the variable **i** as well as increment it by 1 within the loop. With a **for** loop, we can accomplish the same goal with less code:

```

# Using a 'for' loop, square the values 1 through 5 and print
→ results to the screen
for (i in 1:5) {

  print(i^2)

}

## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25

```

User-Defined Functions (UDFs)

Though many built-in functions are available, R provides the flexibility to create our own.

Functions can be an effective alternative to loops. For example, here is a basic function that achieves the same goal as our **while** and **for** loop examples (i.e., squaring integers 1 through 5):

```
# Create a function named square.val() with one argument (x) that
→ squares given x values
square.val <- function(x) {

  x^2
}

# Pass integers 1 through 5 into the new square.val() function
→ and display results
square.val(1:5)

## [1] 1 4 9 16 25
```

While many projects warrant UDFs and/or loops, we do not actually need either to square a set of integers thanks to vectorization. As you gain experience writing R code, you will naturally discover ways to write more performant and terse code:

```
# Square integers
(1:5)^2

## [1] 1 4 9 16 25
```

Graphics

While base R has native plotting capabilities, we will use more flexible and sophisticated visualization libraries such as `ggplot2` in this book. `ggplot2` is one of the most versatile and popular data visualization libraries in R.

We can load the `ggplot2` library by executing the following command:

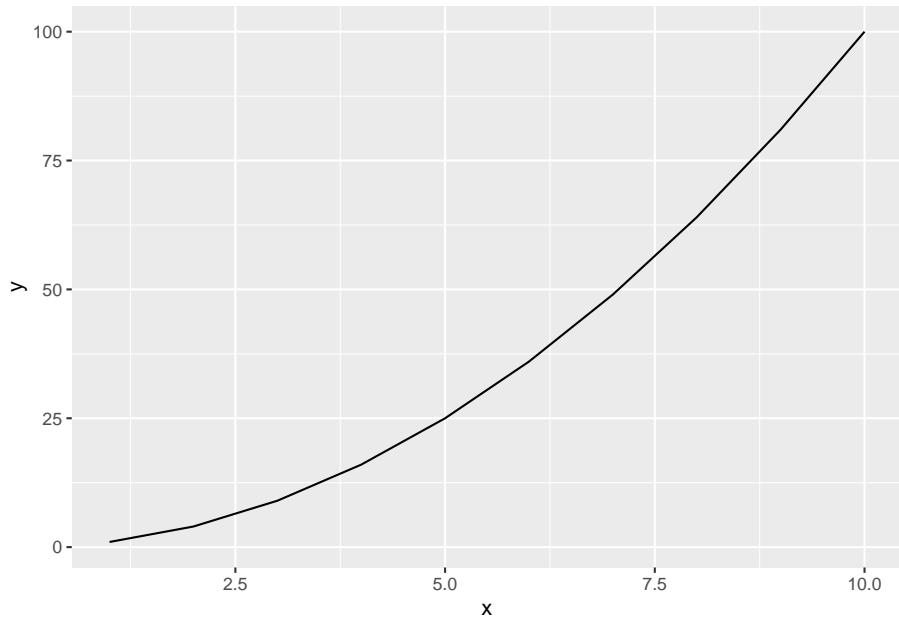
```
# Load library
library(ggplot2)
```

When working with functions beyond what is available in base R, entering `::` between the library and function names is a best practice for efficient coding. R Studio will provide a menu of available functions within the specified library upon typing `library_name::`.

The `ggplot2` library contains many types of visualizations. For example, we can build a line chart to show how the values of vector `x` relate to values of vector `y` in a data frame named `data`:

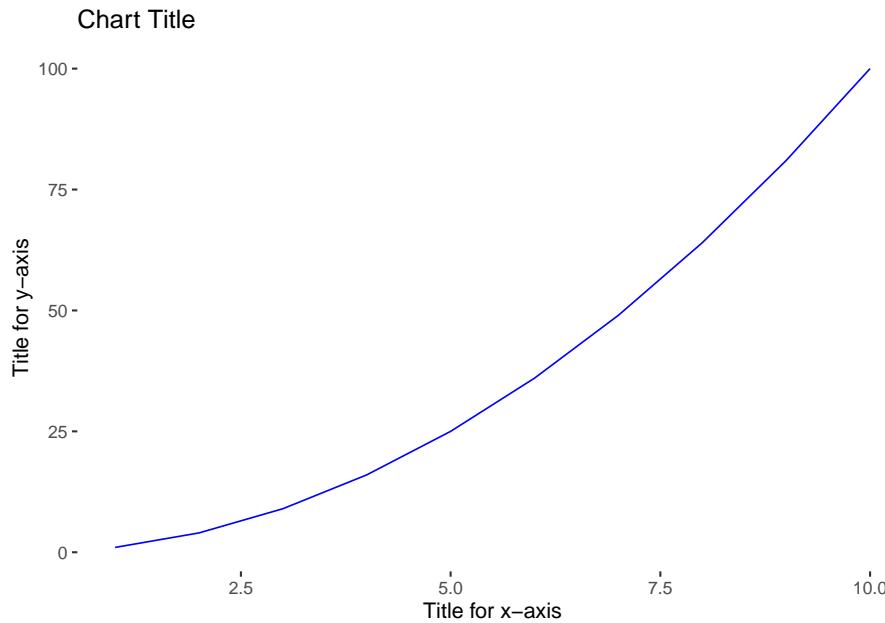
```
# Create data frame containing two related columns, combining
# columns via the cbind() function
x <- 1:10
y <- (1:10)^2
data <- as.data.frame(cbind(x, y))

# Produce line chart
ggplot2::ggplot(data, aes(x = x, y = y)) +
  ggplot2::geom_line()
```



Furthermore, we can use `ggplot` parameters and themes to adjust the aesthetics of visuals:

```
# Produce line chart
ggplot2::ggplot(data, aes(x = x, y = y)) +
  ggplot2::geom_line(size = .4, colour = "blue") + # Reduce line
# thickness and change color to blue
  ggplot2::theme(panel.background = element_blank()) + # Remove the
# default grey background
  ggplot2::labs(title = 'Chart Title', # Assign a chart title
               x = 'Title for x-axis', # Title the x-axis
               y = 'Title for y-axis') # Title the y-axis
```



```
ggplot2::theme(plot.title = element_text(hjust = 0.5)) # Center  
← plot title
```

```
## List of 1  
## $ plot.title:List of 11  
##   ..$ family      : NULL  
##   ..$ face        : NULL  
##   ..$ colour       : NULL  
##   ..$ size         : NULL  
##   ..$ hjust        : num 0.5  
##   ..$ vjust        : NULL  
##   ..$ angle        : NULL  
##   ..$ lineheight   : NULL  
##   ..$ margin        : NULL  
##   ..$ debug         : NULL  
##   ..$ inherit.blank: logi FALSE  
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"  
##   - attr(*, "class")= chr [1:2] "theme" "gg"  
##   - attr(*, "complete")= logi FALSE  
##   - attr(*, "validate")= logi TRUE
```

Review Questions

1. What is the difference between a factor and character vector?
2. What is vectorization?
3. How do data frames differ from matrices?
4. Does executing the `Sum()` function achieve the same result as executing `sum()`?
5. Does `seq(1,10)` return the same output as `1:10`?
6. Is the following command sufficient for creating a vector recognized by R as having three *dates*: `dates <- c("2022-01-01", "2022-01-02", "2022-01-03")`.
7. How are `while` and `for` loops different?
8. If vectors `x1` and `x2` each hold 100 integers, how can we add each element of one to the respective element of the other using a single line of code?
9. How are slots in a list object referenced?
10. What are some examples in which a user-defined function (UDF) is needed?

Introduction to SQL

Structured Query Language (SQL) is the most common language used to extract and wrangle data contained in a relational database. SQL is an essential skill for anyone working in analytics.

Basics

There are three main *clauses* in a SQL query: (a) **SELECT**, (b) **FROM**, and (c) **WHERE**. The **SELECT** and **FROM** clauses are required, though the optional **WHERE** clause is frequently needed.

- **SELECT**: Specifies the columns to include in the output
- **FROM**: Specifies the table(s) in which the relevant data are contained
- **WHERE**: Specifies the rows to search

Despite the clauses being ordered as shown above (**SELECT** then **FROM** then **WHERE**), the **FROM** clause is the first to execute since we first need to identify the relevant table(s) before filtering rows and selecting columns. The **SELECT** clause is the last to execute.

Additional clauses are available for grouping and sorting data.

- **GROUP BY**: Specifies the columns by which data should be grouped when using aggregate functions
- **HAVING**: Specifies conditions for filtering rows based on aggregate functions
- **ORDER BY**: Specifies how data should be sorted in the output

When implementing aggregate functions in a **SELECT** clause, such as counting, summing, or averaging a numeric field, all other non-aggregated fields must be included in the **GROUP BY** clause.

When working with large datasets, it is best to filter records on the database side to avoid reading superfluous records into an analytics tool such as R only

to then filter data to the relevant subset. For example, if we are performing an analysis on employees in the Research & Development department, we should ideally filter to this subset on the database side rather than loading data on the entire workforce and then paring down to the relevant records within R. Fewer records can help enhance the performance of R scripts – especially when R is running on a local machine, such as a laptop, rather than on a more powerful server.

Though it is important to execute SQL queries directly on the database to minimize the amount of data read into R, we will use the `sqldf` library within R to demonstrate the mechanics of a SQL query for easily replicable examples. The `sqldf` library allows us to write SQL to query data frames via an embedded database engine (SQLite by default). In a practical setting, we would pass a string containing the SQL query, execute it directly against the database, and then store the query's results to an object within R. While the syntax of SQL may vary by database, the core structure of queries is universal.

First, let's load the data sets:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")
data("benefits")
data("demographics")
data("job")
data("payroll")
data("performance")
data("prior_employment")
data("sentiment")
data("status")
data("tenure")

# Return row and column counts
dim(employees)
```

```
## [1] 1470    36
```

Next, we will apply the `sqldf()` function to our data frame to extract specific rows and columns. In addition to the `SELECT`, `FROM`, and `WHERE` clauses, we will use the `LIMIT` clause to limit the number of rows that are displayed given the data frame's size ($n = 1,470$). In a practical setting, the `LIMIT` clause is only used for efficient data profiling and troubleshooting, as we would not want to arbitrarily truncate a data set used for analysis.

A best practice in writing SQL is to capitalize the names of clauses and functions and to use separate lines and indentation to make the SQL statements more readable:

```
# Load library
library(sqldf)

# Store SQL query as a character string using the paste()
# function
sql_string <- paste("SELECT
    employee_id
FROM
    employees
WHERE
    dept = 'Research & Development'
LIMIT 10")

# Execute SQL query
sqldf::sqldf(sql_string)

##   employee_id
## 1      1002
## 2      1003
## 3      1004
## 4      1005
## 5      1006
## 6      1007
## 7      1008
## 8      1009
## 9      1010
## 10     1011
```

This query returned a list of employee ids for employees in the Research & Development department.

To optimize query performance, it is important to order conditions in the `WHERE` clause beginning with the condition that will exclude the largest number of records. Conditions are executed sequentially, and each subsequent condition must evaluate all records that remain following any preceding filtering. Limiting the number of records that must be searched when evaluating each condition will reduce the time it takes the query to return results. For example, if two conditions are needed and one excludes 5,000 records while the other excludes 10, the condition that excludes 5,000 records should be listed first in the `WHERE` clause.

Aggregate Functions

Next, let's take a look at average organization tenure by job for those in the Research & Development department:

```
# Store SQL query as a character string
sql_string <- paste("SELECT
                      job_title,
                      AVG(org_tenure)
                  FROM
                      employees
                  WHERE
                      dept = 'Research & Development'
                  GROUP BY
                      job_title")  
  
# Execute SQL query
sqldf::sqldf(sql_string)
```

```
##                                     job_title AVG(org_tenure)
## 1 Healthcare Representative          8.618321
## 2 Laboratory Technician            5.019305
## 3 Manager                          13.673077
## 4 Manufacturing Director          7.600000
## 5 Research Director                10.937500
## 6 Research Scientist               5.113014
## 7 Vice President                  9.500000
```

There are 7 distinct job titles among employees in the Research & Development department, and the average organization tenure for these ranges from 5 to 13.7 years.

Since there could be a small number of employees in certain jobs, in which case average organization tenure may not be as meaningful, we can use the `COUNT(*)` function to count the number of rows for each group. In this case, `COUNT(*)` will return the number of employees in each job in the Research & Development department. We can also assign column aliases via `AS` in the `SELECT` clause to assign different names to the output fields:

```

        FROM
          employees
        WHERE
          dept = 'Research & Development'
        GROUP BY
          job_title
        ORDER BY
          job_title")

# Execute SQL query
sqldf::sqldf(sql_string)

##           job_title employee_cnt avg_org_tenure
## 1 Healthcare Representative      131     8.618321
## 2 Laboratory Technician       259     5.019305
## 3 Manager                      52    13.673077
## 4 Manufacturing Director      145     7.600000
## 5 Research Director            80    10.937500
## 6 Research Scientist           292     5.113014
## 7 Vice President                 2     9.500000

```

The output shows that there are only 2 Vice Presidents in the Research & Development department, while other job titles are much more prevalent.

Since relatively few employees are Vice Presidents, let's use the HAVING clause to only show average organization tenure for Research & Development department jobs with more than 10 employees. We can also use the ROUND() function to truncate average organization tenure to one significant digit:

```

# Store SQL query as a character string
sql_string <- paste("SELECT
                      job_title,
                      COUNT(*) AS employee_cnt,
                      ROUND(AVG(org_tenure), 1) AS avg_org_tenure
                    FROM
                      employees
                    WHERE
                      dept = 'Research & Development'
                    GROUP BY
                      job_title
                    HAVING
                      COUNT(*) > 10
                    ORDER BY
                      job_title")

```

```
# Execute SQL query
sqldf::sqldf(sql_string)
```

```
##          job_title employee_cnt avg_org_tenure
## 1 Healthcare Representative           131        8.6
## 2     Laboratory Technician           259        5.0
## 3             Manager                 52       13.7
## 4 Manufacturing Director            145        7.6
## 5      Research Director              80       10.9
## 6   Research Scientist             292        5.1
```

Joins

In a practical setting, the required data are rarely contained within a single table. Therefore, we must query multiple tables and join them together.

Figure @ref(fig:data-schemas) illustrates how worker, position, and recruiting schemas may be related. For example, a candidate submits a job application to a posted requisition, which is connected to an open position Finance approved as part of the company's workforce plan; when the selected candidate is hired, they become a worker with one or many events (hire, promotion, transfer, termination) and activities (learning, performance appraisals, surveys) during their tenure with the company.

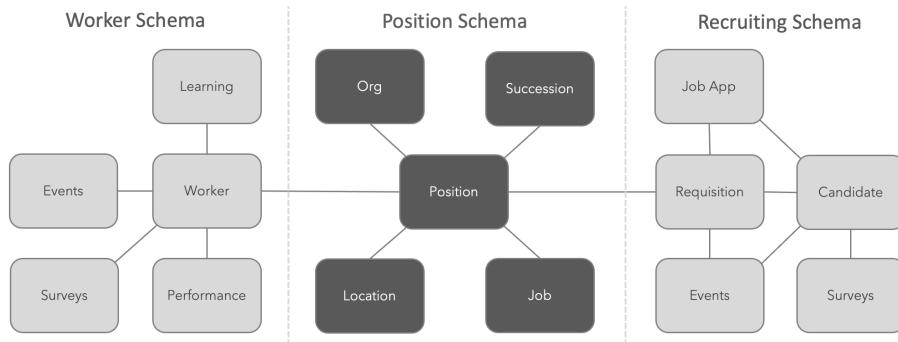


Figure 4: Related tables organized within schemas

Tables are related using a set of keys. Each table needs a **Primary Key (PK)**, which is a unique identifier for each row in the table. A PK may be a single column or multiple columns; a multi-column PK is known as a *composite key*. It is generally best to leverage non-recyclable system-generated ids for PKs. A

Foreign Key (FK) is a column whose values correspond to the values of a PK in another table. **Referential integrity** is the logical dependency of a FK on a PK, and this is an important concept in the context of relational data structures. Referential integrity constraints protect against orphaned FK values in child tables by deleting PK values from an associated parent table.

Figure @ref(fig:erd) shows an **Entity Relationship Diagram (ERD)** that depicts PK/FK relationships among the Position, Worker, and Requisition tables. Notice that the PK for each table shown in Figure @ref(fig:data-schemas) is listed as a FK in related tables.

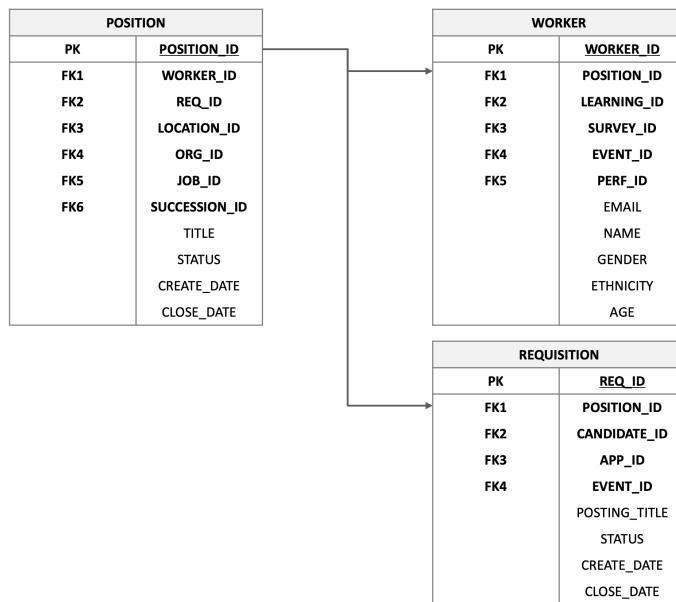


Figure 5: Entity Relationship Diagram (ERD)

With knowledge of the keys required to connect records across tables, there are several methods of joining the tables. Figure @ref(fig:sql-joins) illustrates SQL join types using Venn diagrams:

Both the join type and keys for related tables need to be specified in the SQL statement. The structure of SQL queries for each method of joining **Table A** and **Table B** is represented in the following code blocks:

LEFT INCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# LEFT OUTER JOIN B
# ON A.Key = B.Key
```

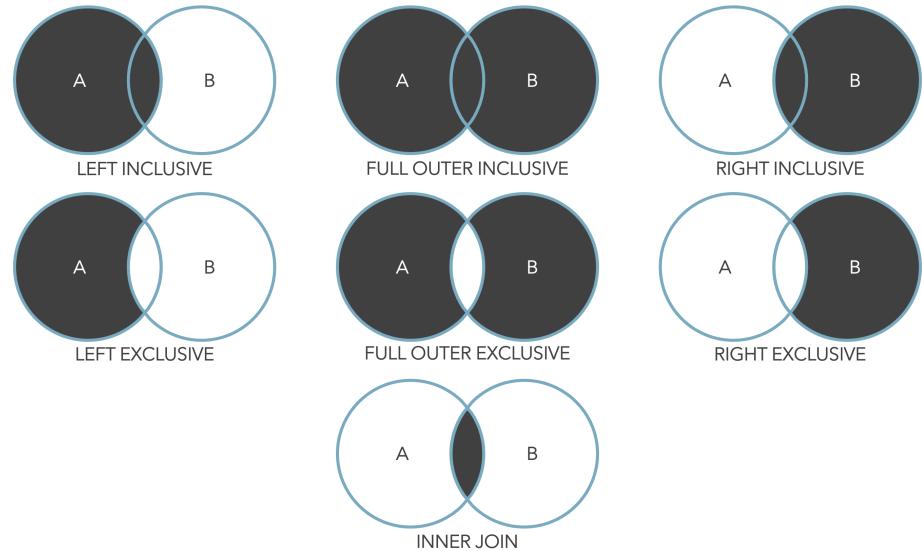


Figure 6: Types of SQL joins

LEFT EXCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# LEFT OUTER JOIN B
# ON A.Key = B.Key
# WHERE B.Key IS NULL
```

FULL OUTER INCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# FULL OUTER JOIN B
# ON A.Key = B.Key
```

FULL OUTER EXCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# FULL OUTER JOIN B
# ON A.Key = B.Key
# WHERE A.Key IS NULL OR B.Key IS NULL
```

RIGHT INCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# RIGHT OUTER JOIN B
# ON A.Key = B.Key
```

RIGHT EXCLUSIVE

```
# SELECT [Output Field List]
# FROM A
# LEFT OUTER JOIN B
# ON A.Key = B.Key
# WHERE A.Key IS NULL
```

INNER JOIN

```
# SELECT [Output Field List]
# FROM A
# INNER JOIN B
# ON A.Key = B.Key
```

To illustrate how SQL joins work, we will leverage three of the data sets used to produce the consolidated `employees` data set that will be leveraged throughout this book: `job`, `tenure`, and `demographics`. In a people analytics context, employee id is often the PK since this identifier should not be shared by two or more employees – past, present, or future. Email or network id may also be a suitable PK. We will use the `employee_id` column in each of the three data frames to facilitate joins.

Let's query these data frames to return the average organization tenure and average commute distance for employees in the Research & Development department, grouped by jobs with more than 10 employees. To accomplish this, we will leverage an `INNER JOIN`, which will return records only for employee ids which are present in all three data frames. For example, if a record exists in `demographics` and `tenure` for a particular employee id, but there is no corresponding record in `job`, that employee id would not be included in the output.

```
# Store SQL query as a character string
sql_string <- paste("SELECT
    job_title,
    COUNT(*) AS employee_cnt,
    ROUND(AVG(org_tenure), 1) AS
    avg_org_tenure,
    ROUND(AVG(commute_dist), 1) AS
    avg_commute_dist
    FROM `bigdata-udacity`."
    "employees AS emp
    INNER JOIN `bigdata-udacity`."
    "demographics AS demo
    ON emp.employee_id = demo.employee_id
    INNER JOIN `bigdata-udacity`."
    "tenure AS ten
    ON emp.employee_id = ten.employee_id
    WHERE emp.department = 'Research & Development'
    GROUP BY job_title
    HAVING employee_cnt > 10")
```

```

        FROM
          demographics
        INNER JOIN
          tenure
        ON
          demographics.employee_id =
←  tenure.employee_id
        INNER JOIN
          job
        ON
          demographics.employee_id =
←  job.employee_id
      WHERE
        dept = 'Research & Development'
      GROUP BY
        job_title
      HAVING
        COUNT(*) > 10
      ORDER BY
        job_title")
# Execute SQL query
sqldf::sqldf(sql_string)

```

	job_title	employee_cnt	avg_org_tenure	avg_commute_dist
## 1	Healthcare Representative	131	8.6	9.8
## 2	Laboratory Technician	259	5.0	9.4
## 3	Manager	52	13.7	7.2
## 4	Manufacturing Director	145	7.6	9.5
## 5	Research Director	80	10.9	8.4
## 6	Research Scientist	292	5.1	9.0

Note that the `INNER JOIN` in this SQL query was structured such that both `tenure` and `job` were joined to `demographics` via the `employee_id` column. We could have instead joined `job` to `tenure` since we joined `tenure` to `demographics`; this would have achieved the same result since all employee ids exist in each of the three data frames.

If it were possible for all employee ids to exist in `demographics` but not in either `tenure` or `job`, we could leverage a `LEFT JOIN` to ensure all records from `demographics` are included in the output irrespective of whether they have matches in `tenure` or `job`:

```
# Store SQL query as a character string
sql_string <- paste("SELECT
    job_title,
    COUNT(*) AS employee_cnt,
    ROUND(AVG(org_tenure), 1) AS
→ avg_org_tenure,
    ROUND(AVG(commute_dist), 1) AS
← avg_commute_dist
    FROM
        demographics
    LEFT JOIN
        tenure
    ON
        demographics.employee_id =
← tenure.employee_id
    LEFT JOIN
        job
    ON
        demographics.employee_id =
← job.employee_id
    WHERE
        dept = 'Research & Development'
    GROUP BY
        job_title
    HAVING
        COUNT(*) > 10
    ORDER BY
        job_title")

# Execute SQL query
sqldf::sqldf(sql_string)
```

	job_title	employee_cnt	avg_org_tenure	avg_commute_dist
## 1	Healthcare Representative	131	8.6	9.8
## 2	Laboratory Technician	259	5.0	9.4
## 3	Manager	52	13.7	7.2
## 4	Manufacturing Director	145	7.6	9.5
## 5	Research Director	80	10.9	8.4
## 6	Research Scientist	292	5.1	9.0

In this case, if `demographics` is the base data set which contains all employee ids (i.e., the ‘LEFT’ data set), it is important for `tenure` and `job` to be joined to it. Joining `job` to `tenure` may result in information loss if an employee id exists in `demographics` and `job` but not in the intermediate `tenure` data set.

When integrating data within R, the `tidyverse` provides a more efficient and parsimonious method of joining many data sets using various join types. Within this framework, components are chained together via the `|>` operator. Though slightly less efficient, the legacy `%>%` operator could be used as an alternative. The example below joins nine data sets into a single `employees` data frame using a left join on the `employee_id` column:

```
# Load library
library(tidyverse)

employees <- list(demographics,
                   status,
                   benefits,
                   job,
                   payroll,
                   performance,
                   prior_employment,
                   sentiment,
                   tenure) |>
  purrr::reduce(left_join, by = "employee_id")
```

Subqueries

Subqueries are queries nested within other queries. Subqueries are often referred to as **inner queries**, while the main queries are referred to as **outer queries**.

For example, if we are interested in performing an analysis on employees with more than a year of organization tenure, we can use a subquery to pass a list of employee ids that meet this criterion into the outer query for filtering. In this case, we would not need to include `tenure` in the join conditions of our main query:

```
# Store SQL query as a character string
# Note: Since employee_id exists in multiple data sets, we would
#       need to specify the data set to include this field in the
#       outer query
sql_string <- paste("SELECT
                      job_title,
                      COUNT(*) AS employee_cnt,
                      ROUND(AVG(commute_dist), 1) AS
                      avg_commute_dist
                 FROM
                   demographics
```

```

        LEFT JOIN
          job
        ON
          demographics.employee_id =
→  job.employee_id
      WHERE
        demographics.employee_id IN (SELECT
→  employee_id FROM tenure WHERE org_tenure > 1)
      AND
        dept = 'Research & Development'
      GROUP BY
        job_title
      HAVING
        COUNT(*) > 10
      ORDER BY
        job_title")

# Execute SQL query
sqldf::sqldf(sql_string)

```

	job_title	employee_cnt	avg_commute_dist
## 1	Healthcare Representative	118	9.7
## 2	Laboratory Technician	198	9.6
## 3	Manager	49	6.9
## 4	Manufacturing Director	133	9.7
## 5	Research Director	74	8.3
## 6	Research Scientist	238	9.2

Virtual Tables

An alternative to a subquery is creating a **virtual table** in the FROM clause. When using an INNER JOIN to connect `demographics` to the virtual table `ids`, which provides a list of employee ids for those with more than a year of organization tenure, any records in `demographics` or `job` that do not relate to employees with at least a year of organization tenure will be dropped. This is true even though a LEFT JOIN is used to join `job` to `demographics` since records in `demographics` will be filtered based on `employee_id` matches in the virtual table. With this approach, our WHERE clause is limited to the `department = 'Research & Development'` condition:

```

# Store SQL query as a character string
sql_string <- paste("SELECT
  job_title,

```

```

        COUNT(*) AS employee_cnt,
        ROUND(AVG(commute_dist), 1) AS
← avg_commute_dist
    FROM
        demographics
    LEFT JOIN
        job
    ON
        demographics.employee_id =
← job.employee_id
    INNER JOIN
        (SELECT employee_id FROM tenure WHERE
← org_tenure > 1) ids
    ON
        demographics.employee_id =
← ids.employee_id
    WHERE
        dept = 'Research & Development'
    GROUP BY
        job_title
    HAVING
        COUNT(*) > 10
    ORDER BY
        job_title")

# Execute SQL query
sqldf::sqldf(sql_string)

```

```

##                job_title employee_cnt avg_commute_dist
## 1 Healthcare Representative           118          9.7
## 2     Laboratory Technician           198          9.6
## 3             Manager                 49          6.9
## 4     Manufacturing Director         133          9.7
## 5     Research Director               74          8.3
## 6   Research Scientist              238          9.2

```

As you can see, the output of the query using a virtual table matches the results from the preceding approach that utilized a subquery.

Window Functions

Window functions are used for performing calculations over a set of rows without collapsing the records. Unlike the aggregate functions we've covered,

window functions do not collapse rows into a single value; the calculated value is returned for each of the rows over which the calculation is performed.

For example, we can assign an organization tenure rank by Research & Development job using the `RANK()` and `OVER()` functions in the `SELECT` clause. The `PARTITION BY` argument functions like a `GROUP BY` clause but without collapsing rows, while the `ORDER BY` argument sorts the records in ascending (`ASC`) or descending (`DESC`) order for proper ranking:

```
# Store SQL query as a character string
# Limit output to 10 records since query does not collapse
→ records
sql_string <- paste("SELECT
    demographics.employee_id,
    job_title,
    commute_dist,
    RANK () OVER (PARTITION BY job_title ORDER
→ BY commute_dist DESC) AS commute_dist_rank
    FROM
        demographics
    LEFT JOIN
        job
    ON
        demographics.employee_id =
→ job.employee_id
    INNER JOIN
        (SELECT employee_id FROM tenure WHERE
→ org_tenure > 1) ids
    ON
        demographics.employee_id =
→ ids.employee_id
    WHERE
        dept = 'Research & Development'
    ORDER BY
        job_title,
        commute_dist_rank
    LIMIT 10")

# Execute SQL query
sqldf::sqldf(sql_string)

##   employee_id      job_title commute_dist commute_dist_rank
## 1      2325 Healthcare Representative      29             1
## 2      1414 Healthcare Representative      28             2
## 3      1010 Healthcare Representative      27             3
## 4      1573 Healthcare Representative      27             3
```

## 5	2200 Healthcare Representative	26	5
## 6	1730 Healthcare Representative	25	6
## 7	1833 Healthcare Representative	25	6
## 8	1993 Healthcare Representative	25	6
## 9	2415 Healthcare Representative	25	6
## 10	1164 Healthcare Representative	24	10

Notice that in the case of commute distance ties, the `RANK()` function assigns the same rank and then adds the number of ties to that rank to determine the rank for the next highest value of commute distance.

We can also treat this query as a virtual table, and then filter on the derived `commute_dist_rank` field to return the highest commute distance for each job. We can add a `DISTINCT()` function in the `SELECT` clause to collapse jobs for which there are more than one employee with the max commute distance, and display the number of ties for each using the `COUNT(*)` function:

```
# Store SQL query as a character string
sql_string <- paste("SELECT
    DISTINCT(job_title) AS job_title,
    COUNT(*) AS employee_count,
    commute_dist
FROM
    (SELECT
        demographics.employee_id,
        job_title,
        commute_dist,
        RANK () OVER (PARTITION BY job_title
← ORDER BY commute_dist DESC) AS commute_dist_rank
    FROM
        demographics
    LEFT JOIN
        job
    ON
        demographics.employee_id =
    → job.employee_id
    INNER JOIN
        (SELECT employee_id FROM tenure WHERE
    → org_tenure > 1) ids
    ON
        demographics.employee_id =
    → ids.employee_id
    WHERE
        dept = 'Research & Development'
    ORDER BY
        job_title,
```

```

        commute_dist_rank) tbl
WHERE
    tbl.commute_dist_rank = 1
GROUP BY
    job_title")

# Execute SQL query
sqldf::sqldf(sql_string)

##          job_title employee_count commute_dist
## 1 Healthcare Representative           1         29
## 2     Laboratory Technician           6         29
## 3             Manager               2         29
## 4 Manufacturing Director            4         29
## 5      Research Director            3         28
## 6   Research Scientist            4         29
## 7 Vice President                  1          8

```

Common Table Expressions (CTEs)

An alternative to the virtual table approach is to use a **common table expression (CTE)**, which is the result set of a query that exists temporarily and only for use in a larger query. Like the virtual table example, CTEs do not persist data in objects or tables; the data exist only for the duration of the query.

```

# Store SQL query as a character string
sql_string <- paste("WITH max_commute_job
AS
(SELECT
    demographics.employee_id,
    job_title,
    commute_dist,
    RANK () OVER (PARTITION BY job_title
    ORDER BY commute_dist DESC) AS commute_dist_rank
    FROM
        demographics
    LEFT JOIN
        job
    ON
        demographics.employee_id =
    job.employee_id
    INNER JOIN
        (SELECT employee_id FROM tenure WHERE
    org_tenure > 1) ids

```

```

    ON
        demographics.employee_id =
    ←  ids.employee_id
        WHERE
            dept = 'Research & Development'
        ORDER BY
            job_title,
            commute_dist_rank)

    SELECT
        DISTINCT(job_title) AS job_title,
        COUNT(*) AS employee_count,
        commute_dist
    FROM
        max_commute_job
    WHERE
        commute_dist_rank = 1
    GROUP BY
        job_title")

# Execute SQL query
sqldf::sqldf(sql_string)

##          job_title employee_count commute_dist
## 1 Healthcare Representative           1         29
## 2      Laboratory Technician          6         29
## 3             Manager               2         29
## 4   Manufacturing Director          4         29
## 5      Research Director           3         28
## 6   Research Scientist           4         29
## 7      Vice President             1          8

```

Review Questions

1. What two clauses must always be present in a SQL query?
2. What SQL clause is executed first at run time?
3. To optimize the performance of a SQL query, how should conditions in the WHERE clause be ordered?
4. How do aggregate functions differ from window functions in SQL?
5. What is a subquery?

6. What is the difference between an INNER JOIN and LEFT JOIN?
7. What is the purpose of a common table expression (CTE)?
8. What does the PARTITION BY function do?
9. Why is it important for queries to limit records on the database side before reading into R?
10. In which clause are filter conditions applied to aggregate functions (e.g., COUNT(*) > 5, AVG(salary) < 100000)?

Research Design

The importance of appropriate research methods and designs cannot be overstated. Research methods and designs help us achieve an accurate understanding of various phenomena and ensure conclusions are justified.

Research Questions

Research questions are fundamental to all research projects. Research questions help focus the study, determine the appropriate methodology, and guide each stage of inquiry, analysis, and reporting. Some examples of research questions germane to people analytics include:

- Q_1 : Why has there been an increase in attrition over the past quarter?
- Q_2 : How equitable are promotion nominations across the organization?
- Q_3 : Are there meaningful differences in the favorability of experiences for remote vs. non-remote employees?
- Q_4 : Do new joiners have the training and resources they need to be successful?
- Q_5 : What portion of team performance is attributable to leadership effectiveness?

Research Hypotheses

Research hypotheses are testable statements about the expected outcome of a research project or experiment.

- H_1 : Manager satisfaction is a significant predictor of voluntary attrition.
- H_2 : Promotion nomination rates are not significantly different by gender and ethnicity.
- H_3 : Employee experience favorability is not significantly different between remote and non-remote workers.

- H_4 : New hire training perceptions are positively associated with onboarding experience favorability.
- H_5 : Leadership effectiveness perceptions explain significant variation in team performance.

Internal and External Validity

Internal validity refers to the extent to which confounding variables are controlled. In other words, internal validity reflects the *robustness* of the study.

For example, if a study finds a significant relationship between work location and attrition but considers no other factors or explanations, this would *not* be a robust study. Work location may emerge significant because certain roles for which attrition is higher are more concentrated in one or more geographies. It could also be the case that the company has made acquisitions in new geographies, and the acquired employees have significantly different experiences (and attrition rates) relative to non-acquired employees.

Confounding variables are critically important in the context of internal validity. A confounding variable is an extraneous variable whose presence impacts the variables being studied such that results do not reflect the actual relationships. Studies with weak internal validity often result in spurious associations that *confound* the true relationship between two variables, leading to invalid conclusions and recommendations.

External validity refers to the extent to which study conclusions will hold in other contexts (for other people, in other places, at other times). *Randomization* is fundamental to our ability to generalize and apply findings to other groups or contexts.

If we survey employees to understand sentiments about recent changes in business strategy but exclude groups for which there may be different impacts or perceptions, conclusions about the collective sentiment would be suspect at best.

Research Methods

There are three major categories of research methods: (1) *quantitative*, (2) *qualitative*, and (3) *mixed methods*.

1. Quantitative

- Addresses *what* questions
- Utilizes numerical data (e.g., surveys, systems)
- Primarily deductive
- Used to test hypotheses

- Involves statistical analyses
- More objective
- More generalizable

2. Qualitative

- Addresses *how* and *why* questions
- Utilizes text data (e.g., focus groups, interviews, open-ended feedback)
- Primarily inductive
- Used to formulate theory or hypotheses
- Involves organizing data into categories or themes
- More subjective
- Less generalizable

3. Mixed Methods

- Integrates the strengths of both quantitative and qualitative methods within a single study, often leading with qualitative approaches to build theory and hypotheses followed by quantitative methods to test hypotheses

Research Designs

In addition to determining whether a quantitative, qualitative, or mixed methods study is most appropriate, researchers also need to decide on the type of study within each of these three. **Research designs** are the types of inquiry within quantitative, qualitative, and mixed methods approaches that issue specific direction for the research procedures (Creswell, 2018). There are multiple taxonomies for research designs, and we will simplify to the most common types.

Within the quantitative category, there are three types of designs: (a) *experimental*, (b) *quasi-experimental*, and (c) *non-experimental*. As shown in Figure @ref(fig:res-designs), it is important to understand the centrality of randomization in this decision.

Experimental Research

Experimental research is concerned with causal (internal) validity. Randomized experimental designs provide the most rigor with regard to causal validity. However, in social science research contexts, true experiments often are not possible due to ethical considerations.

For example, if we were interested in understanding the causal effect leadership quality has on employee engagement, based on a hypothesis that poor leadership decreases employee engagement, we would need to randomly assign employees to one of two groups that are identical on the basis of all variables that could theoretically explain why employees vary in their levels of engagement. Then,

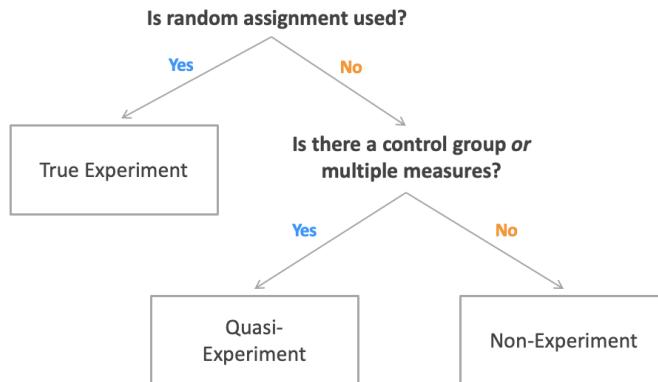


Figure 7: Quantitative research designs

we would need to manipulate the variable of interest (leadership quality) to evaluate if the group of employees subjected to poor leadership (treatment group) reports significantly different levels of engagement relative to the group of employees for whom leadership quality has not been manipulated (control group). In a practical setting, it would of course be unethical to purposefully subject employees to poor leadership with the expectation of reducing engagement – and consequently, productivity, retention, and impact to the organization.

Clinical trials are a common setting for true experiments, as isolating the effects of an experimental drug can be a matter of life or death. In a randomized clinical trial, patients are randomly assigned to an experimental group (patients who receive the drug) or control group (patients who receive a placebo). To protect against placebo effects biasing the results, patients do not know if they receive the experimental treatment or the placebo. Done correctly, these experiments have the highest level of internal validity.

Another example of an experimental design is **A/B testing**. A/B testing is often performed in the context of website optimization, in which two or more versions of the site are shown to customers to identify which version impacts key success metrics more positively. In a people analytics context, we may create two versions of a dashboard and *randomly* assign the permissioned users to each. We could then assess whether utilization rates, average usage time, repeat usage, among other success measures are significantly different between the two groups of users to inform which design is most effective.

In experimental research, it is important to consider the potential influence of the **Hawthorne Effect**, which refers to the tendency of some individuals to modify their behavior in response to the awareness that they are being observed. This term was coined during experiments at Western Electric's factory in the Hawthorne suburb of Chicago in the late 1920s and early 1930s. One of many studies conducted to understand how work environments effect productivity was

known as the “Illumination Experiment”. During this study, researchers experimented with a number of lighting levels in a warehouse in which workers made electrical relays. The researchers found that any change in the lighting – even when introducing poor lighting – led to favorable changes in output. However, these productivity gains disappeared once the attention faded (Roethlisberg & Dickson, 1939).

In a people analytics context, if we inform employees that we are going to monitor productivity over a period of time, it is likely that at least some employees will attempt to modify their behavior in order to increase productivity levels. After all, higher productivity is generally regarded as an ideal across companies and industries. In this case, manipulating some aspect of the typical work context to study a treatment effect, such as flexible work arrangements, may be impacted by this phenomenon; that is, observed differences in productivity may not be attributable to flexible work arrangements but merely due to employees knowing they are being observed.

Quasi-Experimental Research

Quasi-experimental research is an experiment in which participants cannot be randomly assigned.

In the case of our leadership quality example, a quasi-experiment may examine engagement differences between two groups of employees who rate their leader either favorably (Group A) or unfavorably (Group B). A key limitation of this approach is that the groups may be different in important ways beyond leader perception incongruities. For example, Group A employees may be concentrated within a single department, whereas Group B employees may span all other departments. This would indicate that the difference in leadership – and presumably engagement – is driven by factors unique to the department, making it more challenging to isolate the effects of leadership quality on engagement. Perhaps the department with unfavorable leader perceptions has seen significant attrition, or the department is largely first-time people leaders in need of coaching and support.

Another example of quasi-experiments is a pretest-posttest setting in which there are multiple measures. Random assignment could be used in pretest-posttest contexts, in which case this would be characterized as a true experiment, but often this approach is implemented without random assignment. For example, we could test the hypothesized effect of leadership quality on engagement via a pretest-posttest approach. If leaders are selected for a leadership development workshop, we could survey the leaders’ teams and collect data on leader effectiveness perceptions and self-reported engagement prior to (baseline) and after the workshop. It is unlikely that leaders were selected for this workshop by a random process; there were likely criteria driving the selection, such as leaders who were identified as critical talent or who achieved a certain performance level. If this study finds that improvements in leadership effectiveness correlate with improvements in engagement, there would be *some* evidence

favoring investments in leadership development; however, this would not be sufficient evidence for a causal effect.

Though quasi-experiments are not as robust as true experiments, they are usually more feasible in a people analytics context. True experiments control for confounding variables by way of the research design (randomization ensures equivalent groups), while these factors must be controlled statistically in quasi-experimental contexts. In Chapter @ref(lm), we will discuss how to model relationships among multiple variables in order to study how one variable influences another while holding constant variables that may influence the outcome but are not the primary focus of the research.

Non-Experimental Research

Unlike experimental and quasi-experimental designs, **non-experimental research** does not involve the manipulation of a variable. The goal of non-experiments is not to provide evidence for causal effects, but to study measured variables as they naturally occur and disentangle patterns in the data.

Given the potential for alternative explanations of any observed differences or relationships, non-experimental research tends to have lower internal validity than experimental and quasi-experimental designs. As we have discussed, it is often not possible or ethical to manipulate aspects of a person's work context or to randomly assign people to groups. In addition, the nature of research questions does not always warrant experiments. In these cases, one of three non-experimental approaches may be considered: (a) *cross-sectional*, (b) *correlational*, and (c) *observational*.

Cross-sectional research compares two or more natural groups of people. For example, we may examine differences in engagement between employees in the Engineering department relative to employees in the Product department. In this case, we would neither manipulate one's department to determine how department influences engagement, nor randomly assign employees to these departments. Department membership exists apart from the research, so these naturally occurring groups can be leveraged for comparisons. There are of course many examples of naturally occurring groups that we would not manipulate, such as gender, ethnicity, generation, education, job family, job level, location, and tenure band. When participant characteristics are used to create groups, these variables are sometimes referred to as *experimenter-selected* – rather than *experimenter-manipulated*.

Correlational research involves studying the statistical relationship between two variables without manipulating some aspect of a person's natural context. The relationship between leadership quality and engagement could be evaluated using correlational research. However, we would be unable to leverage a correlational design to test a hypothesis positing a causal effect of leadership quality on engagement. We would be limited to understanding how leadership quality and engagement covary; that is, to what extent a change in one variable is associated with a change in the other. Engagement may tend to increase as leadership

quality increases, but a correlational design does not lend to understanding the direction of causal influence – if such an effect exists.

Observational research refers to studies in which the researcher gathers information without research subjects being explicitly involved in the recording of data. Collecting data from the company's Human Capital Management (HCM) system could be an observational research method. For example, if we access data on terminations to determine the rate of attrition over a specified period, we would not need to interfere by asking past or present employees for this information. We would also do so without manipulating any aspect of the ordinary environment, tagging people to naturally occurring or artificially created groups, or evaluating the association of attrition with another variable. The reality is that such an approach would not be too actionable, however, as this would offer no understanding of what may be influencing attrition or how attrition varies across departments, jobs, locations, or other theoretically-relevant dimensions.

Review Questions

1. What type of research method and design would be best suited for a study aiming to understand the effect of stay interviews on employee attrition?
2. Why are quasi-experiments less rigorous than true experiments?
3. When evaluating the effectiveness of a new program, what are some reasons an experimental design would not be implemented?
4. What is the role of research questions?
5. What is the role of research hypotheses?
6. What is the difference between internal and external validity, and why are these concepts important in research?
7. What is an example of a mixed methods study?
8. What is the key difference between experimental and non-experimental research designs?
9. What are the differences between cross-sectional, correlational, and observational non-experimental designs?
10. How can the Hawthorne Effect impact the integrity of an experiment?

Measurement & Sampling

This chapter will cover variable types, measurement scales, errors, sampling methods, and scale reliability and validity.

Variable Types

The framing of variables in research hypotheses guides the treatment of each in our analyses. This section explores the function of *independent*, *dependent*, *control*, *moderating*, and *mediating* variables as well as the broader classification of these variables as either *endogenous* or *exogenous*.

Independent Variables (IV)

An **Independent Variable (IV)** is a variable which is assumed to have a direct effect on another variable. IVs are sometimes referred to as *predictors*, *factors*, *features*, *antecedents*, or *explanatory* variables, and these terms will be used interchangeably throughout this book.

Let's consider a scenario in which we wish to examine whether an inclusion training program given to a random sample of leaders has a positive effect on team-level belonging. In a true experimental design, participation in the inclusion training would be the only difference between the treatment (teams whose leaders who participate in the training) and the control (teams whose leaders who do not participate in the training). In this case, inclusion training participation is the IV (the variable we are manipulating).

IVs are also present in non-experimental designs. For example, we may survey employees and ask them to rate how inclusive their leader is and also provide self-reported belonging ratings. In this context, leader inclusiveness (rather than an inclusion training) is the IV. If we find that average team-level belonging scores tend to be higher when leader inclusiveness scores are higher, this *may* indicate that leader inclusion has some influence on team-level belonging. Of course,

there could be alternative explanations for any observed differences in team-level belonging, which is why experimental designs tend to provide stronger evidence for an IV's effect.

Dependent Variables (DV)

A **Dependent Variable (DV)** is a variable that is *dependent* on the IV. DVs are also referred to as *outcome*, *response*, or *criterion* variables.

In our leader inclusion example, team-level belonging is the DV since this variable is assumed to depend on the level of team leaders' inclusiveness. It's important to note that regardless of a study's results, it is the positioning of the variables in the study's hypotheses (rooted in theory) that determines the type of variable. If we hypothesize that leader inclusion training has a positive effect on team-level belonging, but the study finds no such effect, the inclusion intervention is still the IV and team-level belonging the DV.

Control Variables (CV)

A **Control Variable (CV)** is a variable that is held constant in research. The unchanging state of a CV allows us to understand the extent to which the IV has a unique and independent effect on the DV.

In an experimental context, control variables represent a researcher's attempt to control for alternative explanations so that the IV's main effect on the DV can be isolated. In our leader inclusion example, we would seek to determine whether the reason for any observed differences in team-level belonging can be attributed to leader inclusion training rather than other factors that theoretically may explain the differences. In this setting, we should ensure that the two groups are similar with respect to characteristics such as gender and ethnicity, since underrepresented groups (URGs) may have different experiences independent of any training programs. Gender and ethnicity would be CVs in this case.

While we control for the effects of alternative explanations by way of the research design in an experimental context, we will discuss ways to control for these statistically in Chapter @ref(lm) for correlational designs. CVs are equally important in experimental and non-experimental contexts.

Moderating Variables

A **Moderating Variable** influences the strength of the effect of an IV on a DV. The effect of a moderating variable is often referred to as an *interaction effect* or in the context of a model, an *interaction term*.

Moderating variables may augment (strengthen) or attenuate (weaken) the effect one variable has on another. Interactions are widely understood in the context

of medications; one drug may independently have a positive effect on one's health but when combined with another medication, the interaction can behave very differently – and may even be lethal. In our inclusive leadership example, we may find that the strength of the training's effect on team-level belonging varies based on a leader's span of control (SoC). It stands to reason that leaders with a lower SoC may find it easier to cultivate a climate of inclusivity and belonging, while leaders with a higher SoC may have more scope/projects and team dynamics to manage and may find it more difficult to consistently apply strategies covered during the training.

Interactions between variables are vital to our understanding of nuance and complexity in the dynamics influencing outcomes in organizational settings. Chapter @ref(lm) will cover how to test whether an interaction is meaningful or observed merely due to chance.

Mediating Variables

A **Mediating Variable** explains the *how* or *why* of an IV's effect on a DV. It may be helpful to think of mediating variables as a “go-between” – a part of the causal pathway of an effect.

In the case of inclusive leadership training, effects on belonging are likely not the result of the training itself. More likely, the training raised awareness and helped leaders develop strategies to cultivate team inclusivity. One strategy endorsed during the training may be participative decision making, and the implementation of this strategy may explain **why** the training has a positive effect on team-level belonging. There may be multiple mediators of any observed effect of the intervention on team-level belonging depending on training outcomes.

Variables in these more complex relationship paths are sometimes characterized as having **proximal** or **distal** effects. Proximal effects are those which *directly* influence an outcome, such as the impact of participative decision making on team belonging. Distal effects are upstream effects that *indirectly* influence an outcome, such as inclusion training participation.

Mediating variables may fully or partially mediate the relationship between an IV and DV. **Full mediation** indicates that the mediator fully explains the effect; in other words, without the mediator in the model, there is no relationship between an IV and DV. **Partial mediation** indicates that the mediator partially explains the effect; that is, there is still a relationship between an IV and DV without the mediator in the model. Partial mediation indicates that there are additional explanations for *how* or *why* observed effects exist, such as the implementation of additional team inclusion strategies influencing team belonging. In Chapter @ref(lm), we will discuss how to test for both full and partial mediation.

Translating research hypotheses into conceptual models of hypothesized relationships is helpful in visually representing the function of each variable in the

study. Figure @ref(fig:concept-mdl) illustrates how each type of variable is depicted using our inclusive leadership example:

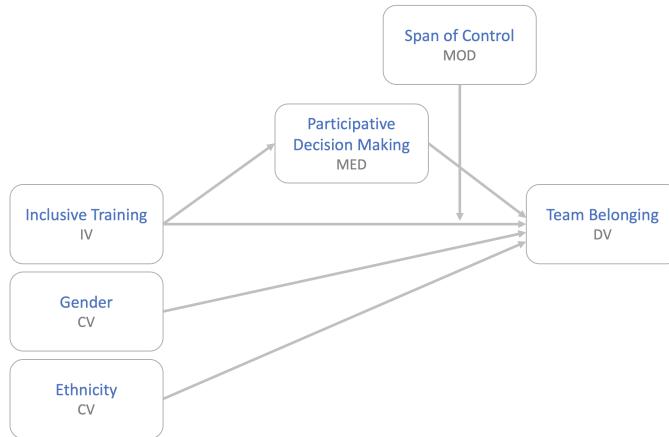


Figure 8: Conceptual model of hypothesized relationships among variables

Endogenous vs. Exogenous Variables

In the context of regression models, which will be covered in later chapters, variables are sometimes categorized as either *endogenous* or *exogenous*. Endogenous and exogenous variables are especially important in econometrics and economic modeling in which analysts seek to understand causal factors influencing outcomes.

An **endogenous variable** is a dependent variable whose values can be determined or explained based on the other variables in a statistical model. In other words, values of the dependent variable change predictably with values of other variables in the model. By contrast, an **exogenous variable** is an independent variable on which other variables in the model have no direct or systematic impact.

For example, if we are interested in studying the factors influencing year-to-date (YTD) sales among salespeople, we may consider an independent variable such as years of sales experience. If YTD sales is a function of one's sales experience, YTD sales is an endogenous variable since its values can be explained – at least in part – by the values of the independent factor (sales experience). Since one's experience in sales is independent of other variables in the model (i.e., YTD sales does not influence a person's years of experience), experience is an exogenous variable.

Measurement Scales

Measurement scales are used to categorize and quantify variables. There are two major categorizations – *discrete* and *continuous* – and these, together with the research hypotheses, help determine appropriate types of analyses to perform.

Discrete Variables

Discrete variables are also known as *categorical* or *qualitative* variables. Categorical variables have a finite or countable number of values associated with them, and these can be further categorized as either *nominal* or *ordinal*.

Nominal

A **nominal** variable is one with two or more categories for which there is no intrinsic ordering to the categories. Examples of nominal variables include office locations, departments, and teams. A **dichotomous variable** is a specific type of nominal variable which has only two unordered categories. Examples of dichotomous variables include people leader vs. individual contributor, active vs. inactive status, and remote worker vs. non-remote worker.

Ordinal

An **ordinal** variable is like a nominal variable with one important difference: ordinal variables have *ordered* categories. Examples of ordinal variables include education levels, job levels, and survey variables measured on Likert-type scales.

Continuous Variables

Continuous variables are also known as *quantitative* variables. Continuous variables can assume any real value in some interval, and these can be further categorized as either *interval* or *ratio* variables.

Interval

Variables measured on an **interval** scale have a natural order and a quantifiable difference between values but no absolute zero value. Examples include SAT scores, IQ scores, and temperature measured in Fahrenheit or Celsius (but not Kelvin). In these examples, 0 is either not an option (i.e., SAT and IQ) or does not represent the absence of something (e.g., 0 degrees is a temperature).

Ratio

Variables measured on a **ratio** scale have the same properties as data measured on an interval scale with one important difference: ratio data have an absolute zero value. Examples include compensation, revenue, and sales; a zero in these contexts is possible and would indicate a true absence of something.

Sampling Methods

The goal of research is to understand a population based on data from a subset of population members. In practice, it is often not feasible to collect data from every member of a population, so we instead calculate **sample statistics** to estimate **population parameters**.

While the population represents the entire group of interest, the **sampling frame** represents the subset of the population to which the researcher has access. In an ideal setting, the population and sampling frame are the same, but they are often different in practice. For example, a professor may be interested in understanding student sentiment about a new school policy but only has access to collect data from students in the courses she teaches. In this case, the entire student body is the population but the students she has access to (those in the courses she teaches) represent the sampling frame. The sample is the subset of the sampling frame that ultimately participates in the research (e.g., those who complete a survey or participate in a focus group).

Sampling methods are categorized as either *probability* or *non-probability*, and this section will cover the types and implementations within each.

Probability Sampling

Probability sampling can help us gain insight into the *probable*. Probability sampling is intended to facilitate inferences since data collected through random selection methods is more likely to be representative of the population and protects against over-indexing on participants with similar qualities in the sample.

It is important to understand the centrality of **randomness** in probability sampling. Randomization protects against subjective biases, self-validates the data, and is the key ingredient that defines the representative means of extracting information (Kahneman, 2011). Sample data that are not representative of the population of interest can lend to anomalies – mere coincidences. While non-random data can be leveraged for directionally correct insights, randomness is required to make inferences about a broader population with a reasonable degree of confidence.

Let's consider an example from Kahneman (2011) in which six babies are born in sequence at a hospital. The gender of these babies is of course random and independent; the gender of one does not influence the gender of another. Consider the three possible sequences of girls (G) and boys (B) below:

- BBBGGG
- GGGGGG
- BGBBGB

Though it may initially be counter-intuitive, since the events are independent and the outcomes (B and G) are (approximately) equally likely, any possible sequence of births is as likely as any other.

Sample size (colloquially referred to as the n -count) is another important factor in sampling as this can have a material influence on the representativeness of sample data – and consequently, the veracity of results and conclusions based on them. As we will explore further in Chapter @ref(inf-stats), as the sample size increases, so too does our confidence that estimates based on sample data reflect population parameters.

To illustrate the effects of sample sizes, let's consider a hypothetical study in which the promotion rate in an organization is found to be lowest in divisions that are primarily software engineers, low diversity, small, and geographically dispersed. Which of these characteristics might offer an explanation? Let's assume that this study also found that the divisions with highest promotion rates have identical characteristics: software engineers, low diversity, small, and geographically dispersed. The key piece of information here is that the divisions are *small*.

Small samples yield extreme results more often than large samples. Small samples neither cause nor prevent outcomes; they merely allow the incidence of the outcome to be much higher (or much lower) than it is in the larger population (Kahneman, 2011).

Simple Random Sampling

Simple random sampling is a method in which each member of the population has the same probability of being selected for a sample. An example of simple random sampling is randomly selecting a specified number (or percent) of employees from the workforce to participate in a survey without regard for tenure, department, level, or other characteristics.

We can use the `sample()` function in R to randomly select from a vector of elements:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")

# Set seed for reproducible results
set.seed(1234)

# Sample 10 employees randomly
n = 10
sample(employees$employee_id, n, replace = F)
```

```
## [1] 2308 2018 2125 2004 1623 1905 1645 1934 1400 1900
```

The `replace = F` argument in the `sample()` function indicates that we want to sample *without* replacement (i.e., we do not want an employee to be selected twice in the sample). If we draw multiple names from a hat without replacement, we will not put names back into the hat once they are drawn; each has a chance of being selected only once.

While sampling *with* replacement would not make sense for applications such as pulse survey participation, as we would not want a given employee to take a survey multiple times, `replace = T` can be applied if the application requires it (e.g., a lottery in which employees can win more than once).

Stratified Random Sampling

Stratified random sampling is a sampling method in which the population is first divided into *strata*. Then, a simple random sample is taken from each *stratum* – a homogeneous subset of the population with similar characteristics with regard to the variable of interest. The combined results constitute the sample.

To ensure samples do not comprise a larger proportion of employees from a particular department, education level, tenure band, generational cohort, or other variable deemed useful in explaining differences in response scores, researchers can randomly select members for each stratum based on the proportion in the respective stratum in the larger population. For example, if 30% of all employees are in the Engineering department, the researcher could *randomly* select a calculated number of employees from the Engineering department such that 30% of employees in the sample come from this department.

Let's demonstrate stratified random sampling in R by sampling $\frac{1}{2}$ of employees within each department. First, we can review counts of employees for each department using the `group_by()` and `summarise()` functions from the `dplyr` library.

```
# Load library
library(dplyr)

# Return employee counts by department
employees |>
dplyr::group_by(dept) |>
dplyr::summarise(n = dplyr::n())

## # A tibble: 3 x 2
##   dept             n
##   <chr>          <int>
## 1 Human Resources     63
```

```
## 2 Research & Development    961
## 3 Sales                      446
```

Next, we will randomly select $\frac{1}{2}$ of employees within each department using the `group_by()` and `sample_frac()` functions from the `dplyr` library. We will store the selected employees' records in a data frame and then query it to validate that the counts are roughly $\frac{1}{2}$ the total count observed for each department.

```
# Obtain and store stratified random sample
strat_sample <- employees |>
  dplyr::group_by(dept) |>
  dplyr::sample_frac(size = .5)

# Return sample counts by department
strat_sample |>
dplyr::group_by(dept) |>
dplyr::summarise(n = dplyr::n())

## # A tibble: 3 x 2
##   dept              n
##   <chr>           <int>
## 1 Human Resources     32
## 2 Research & Development 480
## 3 Sales                  223
```

Cluster Sampling

Cluster sampling is a sampling method often used in market research in which the population is first divided into clusters. Then, a simple random sample of clusters is taken. All the members of the selected clusters together constitute the sample. Unlike stratified random sampling, it is the clusters that are selected at random – not the individuals. It is assumed that each cluster by itself is representative of the population (i.e., each cluster is heterogeneous).

Employees may be partitioned into clusters based only on their geographic region, for example. Since there is not further partitioning on other variables, each cluster is expected to be heterogeneous on the basis of variables other than geographic region – unless geography is related to other variables (e.g., call center employees are all located in a company's Pacific Northwest region). By selecting a random set of clusters, the combination of employees across the selected clusters is expected to be representative of the population.

Let's demonstrate how to implement cluster sampling in R:

```

# Randomly assign each employee to 1 of 10 clusters (groups)
employees$cluster <- sample(1:10, size = nrow(employees), replace
←   = T)

# Randomly select 5 clusters
clusters <- sample(unique(employees$cluster), size = 5, replace =
←   F)

# Store cluster sample
clust_sample <- employees |>
  dplyr::filter(cluster %in% clusters)

# Display dimensions of the cluster sample object
dim(clust_sample)

```

[1] 748 37

Systematic Sampling

Systematic sampling involves selecting sample members from a population according to a random starting point and a fixed, periodic interval known as a sampling interval. The sampling interval is computed by taking the population size and dividing it by the desired sample size. The resulting number is the interval at which population members are selected for the sample.

For example, if there are 10,000 employees and our desired sample size is 500, the sampling interval is 20. Therefore, we would select every 20th employee for our sample. It is important that the sequence does not represent a standardized pattern that would bias the data; this process needs to be random. For example, if the employee id generated by the HCM system increases with time, we would expect employees with longer tenure to have lower employee ids while new joiners would have higher employee ids. Ordering employees by employee id prior to selection could bias the sample on the basis of variables related to tenure (e.g., aggressive periods of hiring in a particular department).

Let's walk through the step-by-step process for implementing the systematic sampling procedure in R:

```

# Specify desired sample size
n = 100

# Determine population size
N = nrow(employees)

# Compute sampling interval, rounding up to nearest whole number
←   via the ceiling() function

```

```

si = ceiling(N/n)

# Randomly select value between 1 and the sampling interval for
# starting value
strt = sample(1:si, 1)

# Increment starting value by the sampling interval until the
# desired sample size is achieved, and hydrate index vector
# with the selected indices
index = seq(strt, strt + si * (n - 1), si)

# Store systematic sample
syst_sample <- employees[index, ]

```

Non-Probability Sampling

Non-probability sampling can help us gain insight into the *possible*. The main difference between non-probability and probability sampling is that non-probability sampling does not involve random selection and probability sampling does. Therefore, we cannot make inferences based on data collected through non-probability sampling methods since the sample is unlikely to be representative of the population.

Convenience (Accidental) Sampling

Convenience sampling is the most common type of nonprobabilistic sampling. This sampling method involves taking samples that are conveniently located around a specific location (physical or virtual).

If we were to study employee sentiment about new benefit plans by polling employees walking through the lobby of a particular office building one morning, this would represent convenience sampling. Aside from the risk of employees sharing socially desirable responses in such a setting and invalidating the results, a major shortcoming of this approach is that we are only capturing the sentiment of those who happen to walk into one particular building during one limited window of time. This would not capture the sentiment of those working remotely, working in another office location, on PTO, taking a sick day, attending an offsite conference or meeting, or stuck in traffic and running late.

Quota Sampling

Quota sampling is a nonprobabilistic sampling method in which researchers assign quotas to a group of people to create subgroups of individuals that reflect the characteristics of the population. This is nonprobabilistic since researchers *choose* the sample rather than randomly selecting it.

If the characteristics of the employee population are known, the researcher polling employees in the office lobby about benefit plans could collect some additional information (e.g., department, job, tenure) to achieve a commensurate proportion of each in the sample. If 30% of all employees are in the Engineering department, the researcher could assign a quota – such as 3 in every 10 participants – to *choose* a sample in which 30% of employees come from the Engineering department.

Purposive (Judgmental) Sampling

The main goal of **purposive sampling** is to construct a sample by focusing on characteristics of a population that are of interest to the researcher. Purposive sampling is often used in qualitative or mixed methods research contexts in which a smaller sample is sufficient. Since it is a nonprobabilistic sampling method, purposive sampling is highly prone to researcher bias.

For example, the People Team may be interested in understanding what is top-of-mind for employees in order to design a survey with relevant items. The team may choose people to participate in focus groups to surface qualitative themes – not for the purpose of generalizing findings but to guide survey item selection efforts.

Sampling & Nonsampling Error

Sampling and nonsampling errors are general categorizations of biases and error in research (Albright & Winston, 2016). It is important to understand these and proactively mitigate the risks they present to research integrity.

Sampling Error

Sampling error is the inevitable result of basing inferences on a random sample rather than the entire population. The two main contributors to sampling error are the size of the sample and variation in the underlying population. The risk of sampling error decreases as the sample size approaches the population size; however, it is usually not feasible to gain information from the entire population, so sampling error is generally a concern.

Selection Bias

Selection bias is the bias introduced by a non-random method of selecting data for analysis, which can systematically skew results in a particular direction. Selection bias may result in observed relationships or differences that are not due to true relationships or differences in the underlying populations, but to the way in which participants or data were selected for the research.

A type of selection bias that is especially important to consider in people analytics is **survival bias**. In people analytics, survival bias is the logical error

of focusing only on people who made it past some selection process while overlooking those who did not. For example, to gain an accurate understanding of the number of employees who *survive* to each tenure milestone (e.g., 1 year, 2 years, 3 years, etc.), we need to study both active and inactive people to avoid biased results. We may find a significant drop in the percent of active employees who survive from 3 to 4 years, for example, but without information on inactive employees we do not know if this is a function of hiring (i.e., relatively few hired more than 3 years ago) or due to a spike in attrition beyond 3 years of tenure.

The work of a mathematician named Abraham Wald during World War II is a classic example of survival bias. Wald was a member of the Statistical Research Group (SRG) at Columbia University that examined damage to returning aircraft (Wald, 1943). Rather than focusing on the areas with damage, Wald recommended a different way of looking at the data, suggesting that the reason certain parts of planes were not covered in bullet holes was because planes that were shot in these areas did not return. In other words, locations with bullet holes represented locations that could sustain damage and still return home. This insight led to armor being reinforced in areas with no bullet holes.

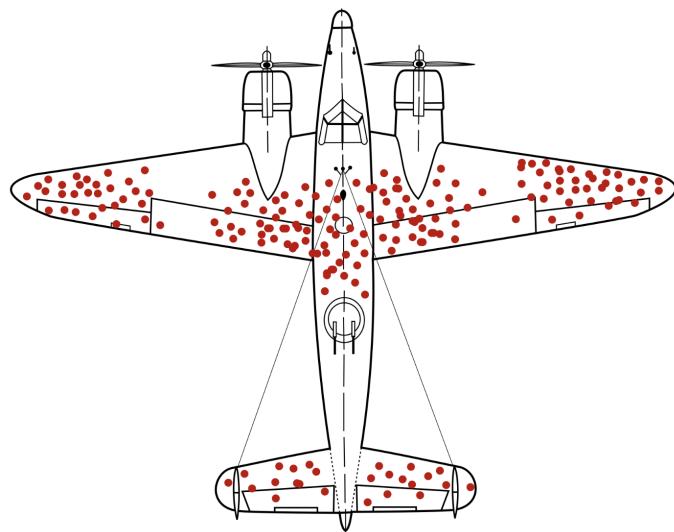


Figure 9: Hypothetical data for damaged portions of returning WWII planes.
Image courtesy of Wikimedia Commons.

Missing data can sometimes be more valuable than the data we have, and it is critical to promote a representative data generative process to prevent biased selection and results.

Nonsampling Error

There are many types of **nonsampling error** that can invalidate results beyond the sampling procedure, and we will focus on several that are particularly germane to people analytics: *nonresponse bias, nontruthful responses, measurement error, and voluntary response bias.*

Nonresponse Bias

As discussed in the context of sampling error, we usually do not have access to information on entire populations of interest, so we must consider the possibility that those for whom we are missing data may have common qualities, perceptions, or opinions that differ from those for whom we do have data. This is known as **nonresponse bias**.

Surveys are a staple in the set of data sources leveraged for people analytics. While survey data provide unique attitudinal and perceptive signals that can help explain future behavior and events, surveys tend to be far more susceptible to nonsampling error than other data sources. If we administer an employee experience survey to the entire organization and receive a 60% response rate, the reality is that we do not know how the 40% of nonrespondents would have responded. It is possible that nonrespondents represent highly disengaged employees, in which case their responses may have materially influenced results and conclusions in an unfavorable direction. It is also possible that nonrespondents did not participate because they were busy, away on vacation, cynical to the confidentiality language in the communications, or any number of other reasons which may or may not have resulted in significantly different feedback relative to respondents.

As an aside, nonrespondents can actually function as an important variable in analyses. In some contexts, nonrespondents may be at greater risk of voluntary attrition than those who respond unfavorably on a stay intentions survey item such as, "I plan to be working here in one year." Evaluating response rate distributions across departments, divisions, roles, and other dimensions may be an insightful endeavor.

Nonresponse bias is not limited to surveys. For example, self-reported demographics such as gender and ethnicity may not be disclosed by all employees in the HCM system. This can bias segmentations based on these categorical dimensions. While there are strategies to address this, such as visual ID or applying models trained to infer missing values (which may be necessary to fulfill EEOC reporting requirements), there may still be error in the imputed values.

Nontruthful Responses

While high response rates may reduce nonresponse bias, this isn't always something to celebrate. Organizations that incentivize participation in surveys often do so at the risk of people responding in socially desirable ways and providing **nontruthful responses** to achieve some defined target. If an employee has

an unhealthy relationship with his manager but does not trust that managers will not have access to individual-level responses, the employee may decide to indicate on the survey that everything is highly favorable to help the team win the month of casual days leadership promised. This can skew and invalidate results.

While survey participation should be strongly encouraged since higher response rates can mitigate the risk of certain types of bias and increase confidence that the survey is representative of the collective organization's sentiments, incentivizing participation can be dangerous.

Measurement Error

Measurement error relates to errors stemming from confusing questions, survey fatigue, and low-quality scales used to measure multidimensional psychological constructs. The field of **psychometrics** is a vast scientific discipline concerned with the development of assessment tools, measurement instruments, and formalized models to understand latent psychological constructs such as engagement, belonging, purpose, and wellbeing using observable indicators.

The measurement method can affect observed data either by changing the underlying construct of interest, or by distorting the measurement process without impacting the construct itself (Spector, 2006). **Common method variance (CMV)**, also known as **monomethod bias**, relates to a widely held belief that relationships between variables measured using the same method are inflated. The idea that the measurement method itself introduces a degree of variance in measures has been cited in the organizational literature for decades, and it is raised almost exclusively when cross-sectional, self-reported surveys are utilized. Though controversial, there is generally a consensus that where it is possible to do so, it is preferable to leverage multiple measurement methods.

My doctoral dissertation research explored how implicit voice theories, which are deep-seated beliefs about the risks involved in speaking up to those higher in the organizational hierarchy (e.g., negative career consequences), influenced the extent to which individual contributors actually speak up in prosocial ways to their leaders (Starbuck, 2016). Since the individual contributors are best placed to provide information on the implicit beliefs they maintain, the IV was measured using cross-sectional self-reports. At the same time, I surveyed the immediate supervisor for each individual contributor and asked them to rate each of their direct reports using a leader-directed voice scale; these supervisor-reports of leader-directed voice were used as the DV in this study. To investigate CMV, which was a tangential interest to the primary research objective, I also included the leader-directed voice scale (using self-reported language) on the survey administered to individual contributors.

For the 1,032 employees from whom I collected data in an investment firm context (individual contributors: $n = 696$; supervisors: $n = 336$), I was surprised to find only a weak correlation between self-reported voice and supervisor-reported voice ($r = .26$, $p < .01$). On average, self-reported voice was higher with less

variation ($\bar{x} = 5.91$, $s = 1.15$) relative to supervisor-reported voice ($\bar{x} = 5.69$, $s = 1.34$). Interestingly, there was support for almost none of the hypothesized relationships when supervisor-reported voice was positioned as the DV, though most were supported when self-reported voice was substituted as the DV in post-hoc analyses.

Given the prevalence of monomethod self-reports in the social sciences, the influence of CMV is an important consideration.

Scale Reliability and Validity

While an exhaustive treatment of psychometrics is beyond the scope of this book, *reliability* and *validity* are two broad sets of methods designed to increase the robustness of psychological instrumentation which will be reviewed in this section.

It may be helpful to consider a weight scale to understand differences between reliability and validity. A weight scale is designed to provide an accurate measurement of one's weight, and we expect measurements to be consistently accurate over time. If a 150 lb. person steps onto a weight scale and receives a reading of 180 lbs., the scale is not valid as the person actually weighs 30 lbs. less than the reading. If the person steps onto the scale a second time moments later and receives a reading of 200 pounds, the scale is not reliable either (inconsistent measurements).

Figure @ref{fig:reli-vali} visually depicts differences between reliability and validity. As researchers, it is critical to measure what we intend to measure (validity) and do it with consistency (reliability). Survey items with poor psychometric properties can lend to invalid conclusions due to measurement error. Even slight adjustments to validated instrumentation – such as changing the number of scale anchors (e.g., increasing from a 5-point to 7-point Likert scale), tweaking item language, or modifying which items are included in a composite scale – generally warrant reliability and validity analyses.

Reliability

Reliability describes the quality of measurement (i.e., the consistency or repeatability of measures). Types of reliability include:

- **Inter-Rater or Inter-Observer Reliability:** the degree to which different raters/observers give consistent estimates of the same phenomenon.
- **Test-Retest Reliability:** the consistency of a measure from one time to another.
- **Parallel-Forms Reliability:** the consistency of the results of two tests constructed in the same way from the same content domain.

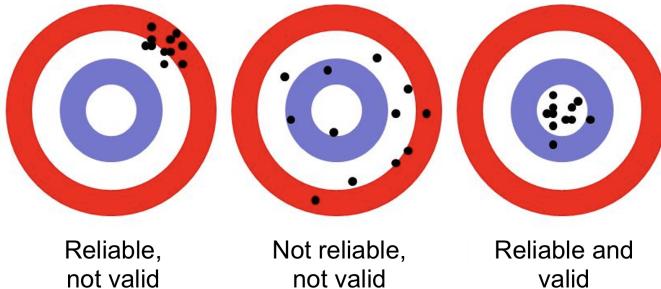


Figure 10: Visual depiction of reliability and validity

- **Internal Consistency Reliability:** the consistency of results across items within a test.

Validity

Validity describes how well a concept was translated into a functioning and operating reality (operationalization). There are four main types of validity: (a) *face validity*, (b) *content validity*, (c) *construct validity*, and (d) *criterion-related validity*:

Face validity

Face validity is an assessment of how valid a measure appears on the surface. In other words, face validity represents whether the measurement approach *on its face* is a good translation of the construct. This is the least scientific method of validity and should never be accepted on its own merits.

Content Validity

Content validity is a somewhat subjective assessment of whether a measure covers the full content domain. Content validity relies on people's perceptions to measure constructs that would otherwise be difficult to measure.

For example, a panel of experts may gather to discuss the various dimensions of a theoretical construct. The psychometrician may then use this information to develop survey items that tap these dimensions to achieve a comprehensive measure of the construct.

Construct Validity

In social science, constructs are often measured using a collection of related indicators that together, cover the various dimensions of the theoretical idea. Constructs may manifest in a set of behaviors, which provide evidence for their existence. **Construct validity** represents the degree to which a collection

of indicators and behaviors – the **operationalization** of the concept – truly represents theoretical constructs.

Psychological safety, a belief that a context is safe for interpersonal risk-taking (Edmondson, 1999), has no direct measure. However, there are indicators and behaviors that are helpful in understanding the extent to which an environment is psychologically safe. We may ask employees whether they are able to bring up problems to decision makers or whether it is safe to take risks on their team. Based on the theoretical conception of psychological safety, these would be helpful (though not collectively exhaustive) indicators of the construct in an organizational setting.

Construct validity can be partitioned into two types:

- **Convergent validity:** the degree to which the operationalization is similar to (converges on) other operationalizations to which it theoretically should be similar.
- **Discriminant validity:** the degree to which the operationalization is *not* similar to (diverges from) other operationalizations to which it theoretically should not be similar.

A **nomological network** is central to providing evidence for a measure's construct validity. The nomological network is an idea developed by Cronbach and Meehl (1955) to represent the constructs of interest, their observable manifestations, as well as the interrelationships among them. The term “nomological” is derived from the Greek word meaning “lawful”; therefore, the nomological network aims to make clear what a construct means so that laws (nomologicals) can be applied. Simply put, the nomological network attempts to link the conceptual and theoretical realm to the observable one to provide a practical methodology for assessing a measure's construct validity.

If psychological safety theory suggests the construct should be positively associated with leader openness and negatively related to employee withholding (silence), we can use validated measures of openness and withholding to test for these theoretical relationships with psychological safety and substantiate construct validity.

Criterion-Related Validity

Criterion-related validity, sometimes referred to as *instrumental validity*, describes how well scores from one measure are adequate estimates of performance on an outcome measure (or *criterion*).

There are two types of criterion-related validity:

- **Predictive validity:** the operationalization's ability to predict something it should theoretically be able to predict.

- **Concurrent validity:** the operationalization's ability to distinguish between groups that it should theoretically be able to distinguish between.

If psychological safety should positively influence employee voice, there would be support for predictive validity if we find that employees who report more favorable perceptions of psychological safety are more willing to speak up. If we administer a new scale to measure psychological safety and *at the same time* (concurrently) administer an existing, validated measure of the same construct, highly correlated results would lend support for the new measure's concurrent validity.

For detailed instruction on the survey scale development process, see DeVellis (2012).

Review Questions

1. What are the differences between parameters and statistics?
2. How does a sampling frame differ from a sample?
3. How does cluster sampling differ from stratified random sampling?
4. What is the primary benefit of probabilistic sampling methods over non-probabilistic sampling?
5. Is nonresponse bias only applicable in the context of surveys?
6. What type of variable influences the strength of the effect one variable has on another?
7. 100 randomly selected employees in the Marketing department of an organization participated in a survey on career pathing for marketing professionals. What is the sample and what is the population in this case?
8. How does the meaning of zero differ between interval and ratio-scaled variables?
9. Can discrete variables have more than 2 values?
10. What are some examples of nonprobabilistic sampling methods?

Data Preparation

To begin a data analysis, we must first extract, combine, organize, and clean the requisite data. As depicted in Figure @ref(fig:analytics-tasks), these data preparation tasks account for a large part of the work analytics professionals do.

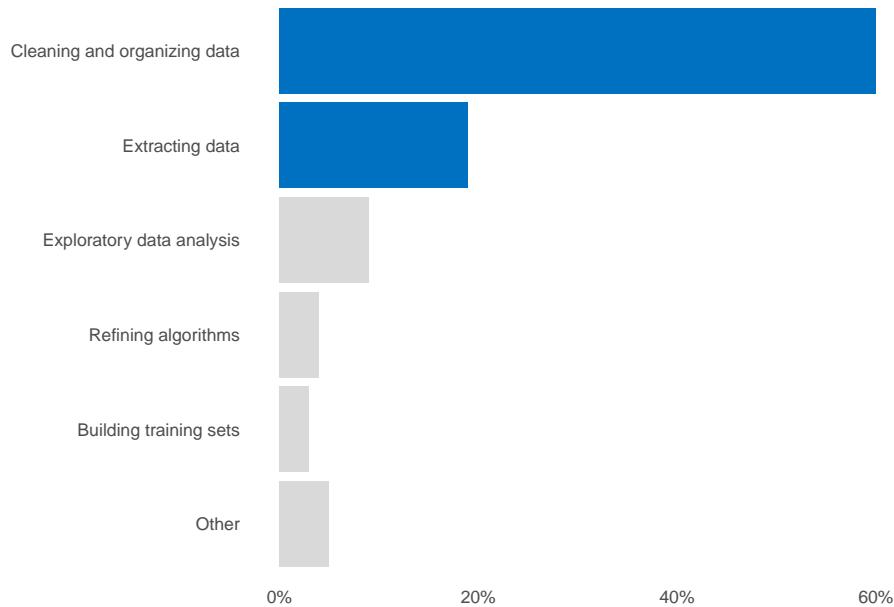


Figure 11: How analytics professionals spend their time

Data Extraction

To properly and efficiently extract data – often through the use of SQL as covered in Chapter @ref(sql-intro) – it is important to first understand some

common ways in which data are stored and structured.

Data Architecture

Data are generally extracted directly from the source systems in which they are generated or from downstream repositories such as a *data lake*, *data warehouse*, or *data mart*.

Data Lake

A **data lake** stores myriad types of data – both structured and unstructured – in its native format until needed. **Structured data** refers to data that fits a predefined data model, such as hire dates formatted as MM/DD/YYYY or zip codes stored as a five-digit string. **Unstructured data** has no predefined data model, such as audio and video files, free-form performance review comments, emails, or digital exhaust from messaging tools; it is difficult to structure this type of data within a set of related tables.

Data are often stored in a data lake so that they are available when use cases are defined for them, at which time a data model is designed and developed outside the data lake to facilitate the requirements.

Data Warehouse

Data in a **data warehouse (DW)** are structured and organized into schemas of related tables *based on business requirements and use cases*. The main difference between a data lake and data warehouse is the type of data they are designed to store. A DW is designed to support analytics across large collections of data, such as transactional data (e.g., point-of-sale systems), point-in-time snapshots (e.g., month-end close reports), survey responses, spreadsheets, and more.

A DW can contain many different types of tables, but this chapter will focus on the two most common which are known as *Type 1* and *Type 2* tables. These tables are sometimes referred to as **slowly changing dimensions (SCD)**.

A **Type 1 table** is created on a regular cadence (usually daily or monthly) and contains no history – only current values. For example, a Type 1 table may contain the latest known attributes for each active and terminated worker such as job, location, and manager. Type 1 tables are sometimes archived and appended to prior snapshots with an effective date, and this design has utility when a view of the workforce is required as of a past date or when an analysis calls for querying across multiple point-in-time snapshots (e.g., computing trailing 12-month attrition rates using average monthly headcount).

It is important to note that leveraging snapshots for trending analyses has some notable deficiencies given the large number of retroactive transactions processed in HCM systems that are not captured if prior snapshots are not updated. Below are some examples:

- Org changes for which incorrect manager assignments are later identified and corrected
- Back-dated compensation changes
- Job profile attribute updates, resulting in incorrect values across snapshots prior to the update date
- Edits to hire, promotion, transfer, and termination events after completing the business process in the system (e.g., delayed start date)

While a past date may be set as the effective date for these transactions, snapshots would incorrectly indicate that the change was effective on the date they were entered into the system (i.e., when the value first changed across snapshots), resulting in misalignment with the system of record. This can result in inaccurate metrics related to headcount, hires, career moves, and terminations in a given period. This type of data leakage can quickly become a larger issue as the size and complexity of the workforce grows. Even a few discrepancies relative to what managers see in the source system can create mistrust in data solutions and hamstring progress up the analytics value chain.

A **Type 2 table** is a table in which a new record is inserted when a change occurs for one or more specified dimensions. Jobs, managers, and locations are examples of slowly changing dimensions but unlike the Type 1 table which contains only the latest information, the Type 2 table houses a *start date* and *end date* for each worker and dimension to facilitate reporting and analysis on changes to attribute values over time. This concept of storing attribute values for the period of time during which they were effective is known as **effective dating**, and the inclusion of effective-dated logic in queries is fundamental to how data are accurately extracted for a particular date of interest.

Figure @ref(fig:type-2-tbl) illustrates the design of a Type 2 SCD for an active worker's job, manager, and location changes. As the data show, worker 123 was promoted from Data Analyst to Sr. Data Analyst 1.5 years after joining, began reporting to their original manager (456) after a short stint reporting to someone else (789), and has worked remotely throughout their entire tenure with the company.

Note that rows where `end_date = '12/31/9999'` indicate *current attributes* for active workers. For inactive workers, `end_date` would be set to the worker's termination date for rows that represent *last known attributes*:

Constructing a view of the workforce as of a particular effective date involves selecting rows where the effective date is on or after `start_date` and on or prior to `end_date`. In a SQL query, this logic can be specified in the `WHERE` clause, which defines the rows to search.

To construct a view of the last known attributes for each worker in this table, we could select rows where the current date is on or after `start_date` and on or prior to `end_date` for active workers and then select the most recent rows (`max end_date`) for each inactive worker. However, using a Type 1 table simplifies

employee_id	attribute_name	attribute_value	start_date	end_date
123	Job	Data Analyst	2/1/2020	8/1/2021
123	Job	Sr. Data Analyst	8/2/2021	12/31/9999
123	Manager	456	2/1/2020	5/15/2020
123	Manager	789	5/16/2020	10/8/2020
123	Manager	456	10/9/2020	12/31/9999
123	Location	Remote	2/1/2020	12/31/9999

Figure 12: Type 2 SCD

this task since each dimension value is stored in a separate column and there is only one row per employee. Figure @ref(fig:type-1-tbl) shows how the current record for worker 123 would look in a Type 1 SCD:

employee_id	hire_date	termination_date	manager_id	job	location
123	2/1/2020		456	Sr. Data Analyst	Remote

Figure 13: Type 1 SCD

Data Mart

A **data mart** is a subset of a DW designed to easily deliver specific information to a certain set of users on a particular subject or for a well-defined use case. For example, in a people analytics context a diversity data mart could be developed to better isolate and secure restricted data such as gender and ethnicity. This data may be used to support diversity descriptives and trends for a limited audience approved by Data Privacy and Legal counsel based on legitimate business needs.

Database Normalization

Normalization is the process of partitioning data into multiple tables to reduce data redundancy and promote data integrity. Conversely, **denormalization** combines data into a single table to facilitate easier and faster data retrieval.

The tables used to explain SQL joins in Chapter @ref(sql-intro) are examples of normalized data. Normalized tables introduce more complexity for analysts since data organized across tables need to be joined together to create a flat data structure that is easier to work with for analytics. However, normalized tables have a key advantage in accounting for past-dated changes since the latest data are retrieved from the various tables when needed rather than leveraging immutable snapshots that only reflect data as of the date and time they were created. For example, if a worker snapshot was created yesterday, and today a change is processed in the system to rename location id MA123 from **Cambridge Office** to **Boston HQ** with an effective date of yesterday, a static worker snapshot created yesterday would show **Cambridge Office** as the location while querying normalized tables would incorporate the updated **Boston HQ** location name.

One way to address the shortcomings of snapshots without the data engineering overhead is to perform destructive loads. **Destructive loads**, sometimes referred to as a **truncate and reload** approach, involves destroying prior snapshots and rebuilding them for each effective date. For example, if there is a policy that retroactive changes cannot be processed in the system prior to the past six months, a destructive load could be performed for a rolling six months of snapshots to ensure they reflect any past-dated worker events and non-worker attribute changes (e.g., department, location, job, position).

Modern Data Infrastructure

Though a deep treatment of data architecture is beyond the scope of this book, it is important to acknowledge the significant advancements in infrastructure and computation – and the important implications for analytics – since SCD architectures were first introduced by Ralph Kimball decades ago. These developments have greatly improved the efficiency with which analysts can translate data into information and insight.

With modern cloud environments, the significant investment associated with humans designing, developing, and maintaining these complex architectures is often difficult to justify given how inexpensive storage and compute have become. Increasingly, the heavy computational tasks have migrated out of data pipelines and into the analytics layer wherein analytics teams have more flexibility and control. Today, daily snapshots containing *all* current and historical records can be copied and stored within partitioned DW tables for a negligible increase in storage costs, and this greatly simplifies data pipeline complexity and engineering support requirements. This changing dynamic has given rise to a new breed of data engineers focused on optimizing the heavy computation requirements of analytics teams.

Data Screening & Cleaning

Once data are extracted and organized in a flat data structure, the initial data review process can commence. This process is often referred to as **exploratory data analysis (EDA)**. EDA involves investigating patterns, completeness, anomalies, and assumptions using summary statistics and graphical representations. An analytics ideal is *unimpeachable data quality*, which is to say that the rigor of upstream business processes and downstream data screening is such that stakeholders become confident enough to channel more energy towards actioning on results than questioning the quality. Suspect data quality is often surfaced during the initial EDA step and affords the opportunity to address and avoid stakeholders discounting results during later phases of the project.

A handy function in base R for initial data screening is `summary()`. This function returns measures of central tendency (mean and median) and spread (min, max, and 1st/3rd quartiles) for each numeric variable.

```

# Load library
library(peopleanalytics)

# Load data
data("employees")

# Summarize df
summary(employees)

```

	employee_id	active	stock_opt_lvl	trainings
##	Min. :1001	Length:1470	Min. :0.0000	Min. :0.000
##	1st Qu.:1368	Class :character	1st Qu.:0.0000	1st Qu.:2.000
##	Median :1736	Mode :character	Median :1.0000	Median :3.000
##	Mean :1736		Mean :0.7939	Mean :2.799
##	3rd Qu.:2103		3rd Qu.:1.0000	3rd Qu.:3.000
##	Max. :2470		Max. :3.0000	Max. :6.000
##				
##	age	commute_dist	ed_lvl	ed_field
##	Min. :18.00	Min. : 1.000	Min. :1.000	Length:1470
##	1st Qu.:30.00	1st Qu.: 2.000	1st Qu.:2.000	Class :character
##	Median :36.00	Median : 7.000	Median :3.000	Mode :character
##	Mean :36.92	Mean : 9.193	Mean :2.913	
##	3rd Qu.:43.00	3rd Qu.:14.000	3rd Qu.:4.000	
##	Max. :60.00	Max. :29.000	Max. :5.000	
##				
##	gender	marital_sts	dept	engagement
##	Length:1470	Length:1470	Length:1470	Min. :1.00
##	Class :character	Class :character	Class :character	1st Qu.:2.00
##	Mode :character	Mode :character	Mode :character	Median :3.00
##				Mean :2.73
##				3rd Qu.:3.00
##				Max. :4.00
##				
##	job_lvl	job_title	overtime	business_travel
##	Min. :1.000	Length:1470	Length:1470	Length:1470
##	1st Qu.:1.000	Class :character	Class :character	Class :character
##	Median :2.000	Mode :character	Mode :character	Mode :character
##	Mean :2.064			
##	3rd Qu.:3.000			
##	Max. :5.000			
##				
##	hourly_rate	daily_comp	monthly_comp	annual_comp
##	Min. : 30.00	Min. :240.0	Min. : 5200	Min. : 62400
##	1st Qu.: 48.00	1st Qu.:384.0	1st Qu.: 8320	1st Qu.: 99840
##	Median : 66.00	Median :528.0	Median :11440	Median :137280

```

##   Mean    : 65.89    Mean    :527.1    Mean    :11421    Mean    :137054
##  3rd Qu.: 83.75    3rd Qu.:670.0    3rd Qu.:14517    3rd Qu.:174200
##  Max.    :100.00    Max.    :800.0    Max.    :17333    Max.    :208000
##
##      ytd_leads      ytd_sales      standard_hrs salary_hike_pct perf_rating
##  Min.   :11.00      Min.   :15496     Min.   :80       Min.   :11.00    Min.   :3.000
##  1st Qu.:45.00      1st Qu.:56997     1st Qu.:80       1st Qu.:12.00    1st Qu.:3.000
##  Median :59.00      Median :73505     Median :80       Median :14.00    Median :3.000
##  Mean   :55.84      Mean   :77124     Mean   :80       Mean   :15.21    Mean   :3.154
##  3rd Qu.:65.00      3rd Qu.:96002     3rd Qu.:80       3rd Qu.:18.00    3rd Qu.:3.000
##  Max.   :95.00      Max.   :281499    Max.   :80       Max.   :25.00    Max.   :4.000
##  NA's   :1061       NA's   :1061      NA's   :80       NA's   :25.00    NA's   :4.000
##
##      prior_emplr_cnt env_sat      job_sat      rel_sat
##  Min.   :0.000      Min.   :1.000     Min.   :1.000     Min.   :1.000
##  1st Qu.:1.000      1st Qu.:2.000     1st Qu.:2.000     1st Qu.:2.000
##  Median :2.000      Median :3.000     Median :3.000     Median :3.000
##  Mean   :2.693      Mean   :2.722     Mean   :2.729     Mean   :2.712
##  3rd Qu.:4.000      3rd Qu.:4.000     3rd Qu.:4.000     3rd Qu.:4.000
##  Max.   :9.000      Max.   :4.000     Max.   :4.000     Max.   :4.000
##
##      wl_balance      work_exp      org_tenure      job_tenure
##  Min.   :1.000      Min.   : 0.00     Min.   : 0.000     Min.   : 0.000
##  1st Qu.:2.000      1st Qu.: 6.00     1st Qu.: 3.000     1st Qu.: 2.000
##  Median :2.000      Median :10.00     Median : 5.000     Median : 3.000
##  Mean   :1.841      Mean   :11.28     Mean   : 7.032     Mean   : 4.229
##  3rd Qu.:2.000      3rd Qu.:15.00     3rd Qu.: 9.000     3rd Qu.: 7.000
##  Max.   :2.000      Max.   :40.00     Max.   :72.000     Max.   :18.000
##
##      last_promo      mgr_tenure      interview_rating
##  Min.   : 0.000      Min.   : 0.000     Min.   :2.000
##  1st Qu.: 0.000      1st Qu.: 2.000     1st Qu.:3.600
##  Median : 1.000      Median : 3.000     Median :4.100
##  Mean   : 2.188      Mean   : 4.123     Mean   :3.989
##  3rd Qu.: 3.000      3rd Qu.: 7.000     3rd Qu.:4.500
##  Max.   :15.000      Max.   :17.000     Max.   :5.000
##

```

Note that fields with `NA` values contain missing values. Also, by default `employee_id` is treated as an integer in R, which is why descriptive statistics appropriate for numeric data are provided. Despite the absence of characters, `employee_id` should be treated as a character string since we will not perform any arithmetic operations using these ids.

Missingness

Before considering whether and how to handle missing data, it is important to distinguish between *structural missingness* and *informative missingness* (Kuhn & Johnson, 2013).

Structural missingness relates to data that are missing for a logical reason. For example, we would not expect a new joiner with a few days of tenure to have a performance score. Likewise, we would not expect an active employee who is not a rehire to have a termination date. Therefore, it would not make sense to define a value to address missing data in these cases.

Informative missingness relates to missing data that is informative regarding an outcome of interest. For example, in a survey context we may find a relationship between missing values on manager effectiveness questions and unfavorability on a psychological safety scale. This may indicate that employees who are fearful of retaliation are uncomfortable providing honest feedback about their managers, while employees who feel it is safe to speak up about issues are more comfortable responding in prosocial ways.

In some cases, we have the luxury of simply removing observations with missing values and using the remaining complete cases for analysis – assuming there are relatively few observations with missing values and no systematic missingness patterns that could bias analyses. However, since we are often working with wide datasets containing relatively few observations in a people analytics setting, this may not be feasible. As we will cover in later chapters, sample size considerations are fundamental to achieving adequate power in statistical testing, so case removal is only possible with larger datasets.

Data imputation refers to the methods by which missing data are replaced with substituted values when case removal is not appropriate. The most common data imputation method is replacing missing values with a descriptive statistic such as the mean, median, or mode based on available data. For example, if most employees have an age in the system, the average, median, or most frequent age could be used in place of the cases with a missing age. To be more precise, the average, median, or most frequent age of those with *similar values* for variables believed to correlate with the missing variable may be used (e.g., similar years of experience, job, level). We would expect there to be less variability in age within a well-defined segment relative to the entire employee population, so this would likely be a more accurate estimate of an individual's actual age.

Let's evaluate the `employees` data frame for missing `annual_comp` values using the logical `is.na()` function, and return values of variables relevant in determining one's annual compensation. The `subset()` function can be used to select a subset of data from a data frame.

```

# Store original annual comp for sample employee
orig_comp <- subset(employees, employee_id == '2176', select =
  ↪ annual_comp)

# Create a NA in lieu of annual comp for illustrative purposes
employees[employees$employee_id == '2176', 'annual_comp'] <- NA

# Return relevant employee characteristics where annual comp is
# missing
subset(employees, is.na(annual_comp), select = c(employee_id,
  ↪ job_title, job_lvl))

##      employee_id          job_title job_lvl
## 1176       2176 Manufacturing Director      2

```

Next, we will impute the average value of `annual_comp` based on employees with the same values for the relevant variables. The `sapply()` function can be used in conjunction with the `mean()` function to apply the average to the subsetted data frame. The `sapply()` function is a member of a broader set of `apply()` functions in R, and the `s` indicates that the result of applying the specified function is a *scalar* object that holds a single value, such as a number (mean value in this case).

```

# Return average annual comp for employees with similar
# characteristics, excluding employees with missing comp values
imputed_comp <- sapply(subset(employees, job_title ==
  ↪ 'Manufacturing Director' & job_lvl == 2, select =
  ↪ annual_comp), mean, na.rm = TRUE)

# Impute missing comp for relevant segment
employees[employees$employee_id == '2176', 'annual_comp'] <-
  ↪ imputed_comp

# Display absolute difference between original and imputed comp
round(abs(orig_comp - subset(employees, employee_id == '2176',
  ↪ select = annual_comp)), 0)

##      annual_comp
## 1176       1169

```

While this approach should help in demonstrating the mechanics of imputing a missing value on a case-by-case basis, a more scalable solution is needed for data with a large number of missing values across employees with different values of

these variables. There are more sophisticated methods of data imputation that involve models to estimate missing values, such as *linear regression* which will be introduced in Chapter @ref(lm). Modeling techniques leverage a similar approach to the method outlined above in that the target values of cases with similar characteristics to those with missing values are used to aid estimation. **Multiple imputation** builds upon this approach by combining the information from multiple data sets imputed using different methods with a goal of minimizing the potential bias introduced by a singular method of imputation.

Outliers

The treatment of outliers is one of the most enduring and pervasive methodological challenges in organizational research. A literature review by Aguinis, Gottfredson, and Joo (2013) uncovered 14 unique definitions of outliers, 39 outlier identification techniques, and 20 different ways of addressing them. Appropriate methods for defining and addressing outliers are domain-specific, and there are many important considerations that should inform whether and how outliers should be handled.

The water crisis in Flint, Michigan is a tragic example of a series of statistical mishaps involving poor sampling methodology and outlier handling. As the story goes, Flint stopped paying the Detroit Water and Sewer Department to source water from Lake Huron and began sourcing it from the Flint River as a cost-cutting measure in April 2014 (Langkjær-Bain, 2017). Residents of Flint began showing signs of lead poisoning, and authorities denied residents' claims that their tap water was to blame – despite some extreme cases in which the tap water was colored orange.

Water companies routinely add chemicals to water to prevent pipe corrosion which can cause lead to seep into drinking water. In Flint's hurry to switch water sources, they failed to address the fact that the Flint River is naturally high in chloride – a chemical that corrodes pipes. According to the Lead and Copper Rule (LCR) of 1991, lead consumption should not exceed 15 parts per billion (ppb) in more than 10% of homes tested – though no quantity of lead is considered safe to ingest. If the 90th percentile value for sampled homes is greater than 15 ppb, action is required.

Two initial samples of tap water were taken from a concerned resident's home; one measured 104 ppb (6X higher than the LCR threshold) and the other measured 397 ppb (25X higher than the LCR threshold). Authorities dismissed these samples as outliers, citing old lead pipes in the resident's home. Authorities collected samples of their own and despite federal guidelines requiring $n \geq 100$ samples, an under-powered analysis was performed using only 71 samples. Of the 71 samples, two with levels above the 15 ppb threshold were discarded, and the removal of these outliers resulted in aggregate lead levels falling beneath the action threshold.

In the end, the tenacity of the growing number of residents with health concerns resulted in new samples being analyzed by a team of researchers at Virginia Tech University. Researchers found that the 90th percentile value among the sample of households – which included homes with non-lead pipes and water filtration systems – was 26.8 ppb and the highest individual sample was 158 ppb! The city switched back to the Lake Huron water source in October 2015 (18 months later), and a state of emergency was declared. The State of Michigan has brought numerous criminal charges against state and local officials which include misconduct in office, tampering with evidence, willful neglect of duty, and various counts of conspiracy. Residents also launched a series of class action lawsuits against the Governor (Langkjær-Bain, 2017).

This may seem like a dramatic appeal, but the importance of investigating outliers cannot be overstated. Simply discarding outliers may truly be a grave mistake! If outliers are attributable to measurement error, it may be appropriate to discard them. If outliers represent properly measured values, they should be investigated. As we will discuss further in Chapter @ref(desc-stats), a common method of outlier detection is identifying values which fall outside the following interval:

$$I = Q1 - 1.5 * IQR; Q3 + 1.5 * IQR$$

Low Variability

Variables with **low variability** often do not provide sufficient information for identifying patterns in data. For example, if we are interested in using information on stock options to understand why employees vary in their levels of retention risk, but find that the employee stock purchase plan (ESPP) terms are identical for nearly all employees, including a stock option variable in the analysis is unlikely to provide any meaningful signal.

When working with survey data, checking for **straightlining** should be an early data screening step. Straightlining refers to a constant response across all survey items, which may be evidence that the respondent lost motivation or was not attentive and thoughtful when taking the survey. Since straight-line responses may influence results, it is often best to discard these cases – especially when the sample size is adequately large for the planned analyses without them. If the same response is given for both positively and negatively worded versions of a question (e.g., comparing “I plan to be here in a year” to “I do not plan to be here in a year”), which we expect to be inversely related, this gives added support for discarding these responses.

Fields with low variability can be easily identified using descriptive statistics from the `summary()` function. If the `Min` and `Max` values are equal, there is no variability in the field’s values. Based on the following descriptives, we should remove `standard_hrs` from the data:

```
# Return descriptives to understand distribution of standard
→ hours
summary(employees$standard_hrs)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	80	80	80	80	80	80

Given that the data dictionary in Chapter @ref(getting-started) indicates performance ratings range from 1 to 4, the following descriptives should raise red flags:

```
# Return descriptives to understand distribution of standard
→ hours
summary(employees$perf_rating)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	3.000	3.000	3.000	3.154	3.000	4.000

Assuming not everyone in the company is a stellar performer (i.e., only Note-worthy and Exceptional ratings), we may be working with a partial data set that could bias analyses. This may be due to poor integrity of performance data in the source system or repository from which the data were pulled, or the query written to extract data from the source may be flawed.

Inconsistent Categories

Inconsistent categories impact aggregation and trending by categorical dimensions. It is often necessary to create mappings based on logical rules in order to standardize dimension values across time. In the case of reorgs, a department may be disbanded, renamed, or integrated into one or multiple other departments. Therefore, when working with historical data, records may contain legacy department names that do not align with the current organizational taxonomy. Mapping from former to current departments may require logic based on manager ids, divisions, job profiles, or other variables depending on the nature of reorgs over time.

Job architecture projects often introduce the need for mappings as well. Jobs and levels may completely change for all employees with a job architecture revamp, in which case trending along job and level dimensions (e.g., attrition by job or level over multiple years) is only possible with logic that clarifies how legacy jobs and levels map to those in the new career framework.

Changes to allowable values in source systems often result in inconsistent categorical data over time. For example, the education field may switch from a

free-form text field in which employees can enter any value (e.g., B.S., B.A., BS, BA, Bachelor of Science, Bachelor of Arts, Bachelor's, Bachelors, Bachelor's Degree, Bachelor Degree, undergraduate degree, 4-year degree, four-year degree) to a standardized solution in which there is a clean and well-defined set of allowable values from which employees can choose (e.g., Bachelor's Degree, Master's Degree, Doctoral Degree). This warrants either a one-time historical cleanup upon implementing the allowable values or downstream logic to tidy up data for analytics. A best practice is to address data quality issues upstream (e.g., in the source system) to avoid duplicative data cleaning procedures across downstream applications.

Data Binning

Data binning refers to the process by which larger high-level groups of values are defined and constructed. As a general rule, extremely granular categories should be avoided – especially when there is no theoretical basis for such categories facilitating a project's objectives or deepening insights. Where the n -count is expected to be consistently low for a defined categorical bin, it is usually best to define a larger bin. For example, a variable measuring highest level of educational attainment that contains 9th, 10th, 11th, and 12th grade categories may be converted into higher-level “High School Not Completed” and “High School Completed” bins.

For modeling applications, it is important to let the algorithm determine the cutpoints for numeric data in relation to the outcome. For example, if organization tenure is measured in years, arbitrarily defining bin sizes of ‘Less Than 1 Year’, ‘1-5 Years’, and ‘More Than 5 Years’ will likely result in information loss. Any variability *within* these bins that may be useful in explaining variance in the outcome would be lost with such wide bins. The machine learning (ML) models that will be covered in Chapter @ref(pred-mod) are great for algorithmically determining cut points for binning numeric data across descriptive, diagnostic, and predictive projects alike.

One-Hot Encoding

One-hot encoding, also known as **dummy coding**, involves transforming a categorical variable into numeric values on which statistical procedures can be performed. For EDA, this is not required, as counts and percent of total metrics can be calculated on these dimensions for descriptive purposes. However, for modeling applications, unordered categorical variables must be converted into $k - 1$ variables, where k is the number of categories, using binary (1/0) coding.

Understanding how categorical data are coded is critical to a correct interpretation of output. For example, if a remote work variable exists with “Remote” or

“Non-Remote” values, we may code “Remote” values as 1 and “Non-Remote” values as 0. We could then evaluate the statistical relationship of this transformed categorical variable with other numeric variables.

If an unordered categorical variable has more than 2 values, we must create a separate 1/0 field for each value and omit one category for use as a reference group. As we will cover in Chapter @ref(lm), one of several assumptions in linear regression is that independent variables are not collinear; that is no pair of independent variables is highly correlated. Without an omitted category, each of the one-hot encoded fields will be perfectly correlated with the others. That is, when the field representing category A is 1, the fields for other categories will always be 0. As illustrated in Figure @ref(fig:onehot-encoding), by omitting a category there will be cases when all fields have a 0 value (i.e., rows where the value is the omitted category), which will reduce the strength of the bivariate correlations.

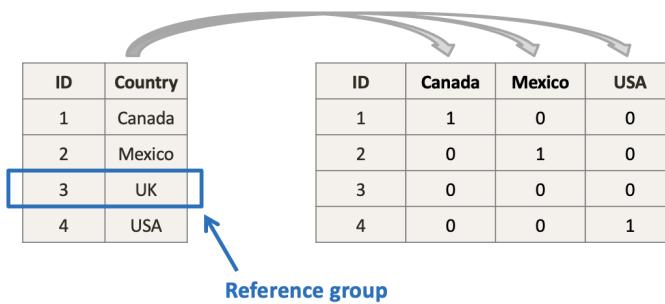


Figure 14: One-hot encoding

For a categorical variable with only two values, the `ifelse()` function can be leveraged to assign values:

```
# Return unique values of gender field with unique() function
unique(employees$gender)
```

```
## [1] "Female" "Male"

# Gender one-hot encoding
employees$gender_ohe <- ifelse(employees$gender == 'Female', 1,
                                0)

# Preview records
head(subset(employees, select = c(employee_id, gender_ohe)))
```

```
##   employee_id gender_ohe
## 1         1001      1
## 2         1002      0
## 3         1003      0
## 4         1004      1
## 5         1005      0
## 6         1006      0
```

For variables with more than 2 unordered categories, we can leverage the `model.matrix()` function for one-hot encoding. Let's illustrate be encoding locations. As we can see, Human Resources is the smallest department ($n = 63$) in these data:

```
# Return counts by department
employees |> dplyr::count(dept, sort = TRUE)
```

```
##           dept   n
## 1 Research & Development 961
## 2          Sales 446
## 3 Human Resources  63
```

By default, the `model.matrix()` function will produce a matrix of 1/0 values for $k - 1$ categories. The first column in the matrix is an intercept column containing a value of 1 for each row to ensure linear independence, and the default behavior results in the first value of the factor being the omitted group. For more flexibility over which value is omitted, we can drop the intercept using `-1` in the first argument passed to `model.matrix()` and then choose the reference group for the analysis in a subsequent step.

```
# Department one-hot encoding
dept_ohe <- model.matrix(~dept-1, data = employees)

# Preview data
head(dept_ohe)
```

```
##   deptHuman Resources deptResearch & Development deptSales
## 1                  0                      0                  1
## 2                  0                      1                  0
## 3                  0                      1                  0
## 4                  0                      1                  0
## 5                  0                      1                  0
## 6                  0                      1                  0
```

We will drop the department with the lowest n rather than the more arbitrary method based on the first value of the factor. Since departments are coded as either 1 or 0, we can use the `colSums()` function to sum each column and the `which.min()` function to identify which has the lowest sum (i.e., smallest department by employee count).

```
# Drop department with lowest sum (lowest n-count)
dept_ohe <- dept_ohe[, -which.min(colSums(dept_ohe))]

# Preview refined one-hot encoded data
head(dept_ohe)

##   deptResearch & Development deptSales
## 1                      0          1
## 2                      1          0
## 3                      1          0
## 4                      1          0
## 5                      1          0
## 6                      1          0
```

As expected, the Human Resources department was dropped via the n -count selection criterion. We can now integrate these one-hot encoded fields into the original data frame for analysis.

```
# Combine employees and matrix containing one-hot encoded
# → departments
employees <- cbind(employees, dept_ohe)

# Drop original department field
employees <- subset(employees, select = -c(dept))
```

Feature Engineering

Level one people analytics tends to utilize only the delivered fields from the HCM system (e.g., location, job profile, org tenure), but a good next step is to derive smarter variables from these fields. These can then be used to cut data differently or as inputs in models. Below are some examples of how basic data available in the HCM system can be transformed into new variables that provide different information. This can be easily accomplished using the arithmetic functions we have covered.

- Number of jobs per unit of tenure (larger proportions tend to see greater career pathing)

- Office/remote worker (binary variable dummy coded as 1/0)
- Local/remote manager (binary variable dummy coded as 1/0)
- Hire/Rehire (binary variable dummy coded as 1/0)
- Hired/acquired (proxy for culture shock effects)
- Gender isolation (ratio of employee's gender to number of the same within immediate work group)
- Generation isolation (comparison of age bracket to most frequent generational bracket within immediate work group)
- Ethnic isolation (ratio of employee's ethnicity to number of the same within immediate work group)
- Difference between employee and manager age
- Percentage change between last two performance appraisal scores (per competency and/or overall)
- Team and department quit outbreak indicators (ratio of terms over x months relative to average headcount over x months)
- Industry experience (binary or length in years)

Review Questions

1. What are the differences between data lakes, data warehouses, and data marts?
2. What is the difference between a Type 1 and Type 2 table in a DW?
3. In what ways has modern cloud computing influenced data architecture?
4. Why is it dangerous to address missing values without domain knowledge of how the data are generated?
5. How can missing values be addressed when impacted records cannot be eliminated from a data set?
6. When is one-hot encoding required for categorical variables?
7. When one-hot encoding a categorical variable with more than two categories, why is an omitted category important?
8. When binning numeric data, what are some considerations in determining the size of each bin?
9. Why should variables with low to no variability be dropped?
10. Where are validation rules ideally situated to limit downstream data cleaning tasks and ensure consistent categorical dimension values?

Descriptive Statistics

This chapter reviews essential univariate and bivariate analysis concepts that underpin the more complex statistical methods in subsequent chapters of this book. Univariate and bivariate analyses can be either descriptive or inferential; this chapter will cover descriptive techniques while Chapter @ref(inf-stats) will cover inferential methods.

Descriptive statistics are rudimentary analysis techniques that help describe and summarize a variable's data in a meaningful way. Descriptive statistics do not allow us to draw any conclusions beyond the available data but are helpful in interpreting the data at hand.

Univariate Analysis

Univariate analysis is the simplest form of statistical analysis, which explores each variable independently.

There are two categories of univariate analyses: (a) **measures of central tendency** describe the central position in a set of data; and (b) **measures of spread** describe how dispersed the data are.

Measures of Central Tendency

Mean

Perhaps the most intuitive measure of central tendency is the **mean**, which is often referred to as the average. The mean of a sample is denoted by \bar{x} and is defined by:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

The population mean is denoted by μ and is defined by:

$$\mu = \frac{\sum_{i=1}^n x_i}{N}$$

The mean of a set of numeric values can be calculated using the `mean()` function in R:

```
# Fill vector x with integers
x <- c(1, 2, 3, 3, 100, 200, 300)

# Calculate average of vector x
mean(x)
```

```
## [1] 87
```

Median

The **median** represents the midpoint in a sorted vector of numbers. For vectors with an even number of values, the median is the average of the middle two numbers; it is simply the middle number for vectors with an odd number of values. When the distribution of data is skewed or there is an extreme value, the median *may* be a better measure of central tendency.

The `median()` function in R can be used to handle the sorting and midpoint selection:

```
# Calculate median of vector x
median(x)
```

```
## [1] 3
```

In this example, the median is only 3 while the mean is $\bar{x} = 87$. Large deltas between mean and median values provide important information about the distribution of data.

Here, a single value has significant leverage on these measures of central tendency. To demonstrate, let's eliminate one instance of 3 from the vector and recalculate the mean and median:

```
# Fill vector x1 with integers
x1 <- c(1, 2, 3, 100, 200, 300)
```

```
# Calculate mean of vector x1
mean(x1)
```

```
## [1] 101
```

```
# Calculate median of vector x1
median(x1)
```

```
## [1] 51.5
```

By removing a single value from this vector, the mean increased from $\bar{x} = 87$ to $\bar{x} = 101$ and the median from 3 to 51.5!

Note that differences in mean and median values for `x` and `x1` are *not* due to an extreme value (outlier), as 3 is similar to half of the values in the vector. However, in some cases extreme values may be the cause of large discrepancies between mean and median values since the mean can be sensitive to extreme values. Consider the following set of values:

```
# Fill vector x2 with integers
x2 <- c(1, 2, 3, 4, 5, 1000)
```

```
# Calculate mean of vector x2
mean(x2)
```

```
## [1] 169.1667
```

```
# Calculate median of vector x2
median(x2)
```

```
## [1] 3.5
```

In this case, the value of 1000 has a significant influence on the mean ($\bar{x} = 169.2$) but the median of 3.5 is representative of the middle of values in this vector.

The reality is that both the mean and median can be misleading – and even inappropriate. It is important to understand how the data are distributed around these centers. It would not be too useful to calculate median organization tenure, for example, for a hyper-growth company that has hired the majority of its workforce in the past few months; long-tenured employees would be lost in this metric.

The larger the n -count, the less influential an extreme value will be on \bar{x} . As we will learn in Chapter @ref(inf-stats), sample size is fundamental to our ability to achieve precise estimates of population parameters based on sample statistics.

While the focus of this section is central tendency, it is important to recognize that outlying values are often the more actionable data points in an analysis since these cases may represent those with significantly different experiences relative to the average employee. Understanding the *distribution* of data is critical, and the spread of data around measures of central tendency will receive considerable attention throughout this book.

Mode

The **mode** is the most frequent number in a set of values.

While `mean()` and `median()` are standard functions in R, `mode()` returns the internal storage mode of the object rather than the statistical mode of the data. We can easily create a function to return the statistical mode(s):

```
# Fill vector x2 with integers
x3 <- c(1, 2, 3, 3, 100, 200, 300, 300)

# Create function to calculate statistical mode(s)
stat.mode <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}

# Return mode(s) of vector x3
stat.mode(x3)
```

```
## [1] 3 300
```

In this case, we have a bimodal distribution since both 3 and 300 occur most frequently.

Range

The **range** is the difference between the maximum and minimum values in a set of numbers.

The `range()` function in R returns the minimum and maximum numbers:

```
# Return lowest and highest values of vector x
range(x)
```

```
## [1] 1 300
```

We can leverage the `max()` and `min()` functions to calculate the difference between these values:

```
# Calculate range of vector x
max(x, na.rm = TRUE) - min(x, na.rm = TRUE)

## [1] 299
```

In people analytics, there are many conventional descriptive metrics – largely counts, percentages, and averages cut by various time (e.g., day, month, quarter, year) and categorical (e.g., department, job, location, tenure band) dimensions. Here is a sample of common measures:

- Time to Fill: average days between job requisition posting and offer acceptance
- Offer Acceptance Rate: percent of offers extended to candidates that are accepted
- Pass-Through Rate: percent of candidates in a particular stage of the recruiting process who passed through to the next stage
- Progress to Goal: percent of approved positions that have been filled
- cNPS/eNPS: candidate and employee NPS (-100 to 100)
- Headcount: counts and percent of workforce across worker types (employee, intern, contingent)
- Diversity: counts and percent of workforce across gender, ethnicity, and generational cohorts
- Positions: count and percent of open, committed, and filled seats
- Hires: counts and rates
- Career Moves: counts and rates
- Turnover: counts and rates (usually terms / average headcount over the period)
- Workforce Growth: net changes over time, accounting for hires, internal transfers, and exits
- Span of Control: ratio of people leaders to individual contributors
- Layers/Tiers: average and median number of layers removed from CEO
- Engagement: average score or top-box favorability score

Measures of Spread

Variance

Variance is a measure of variability in the data. Variance is calculated using the average of squared differences – or deviations – from the mean.

Variance of a population is defined by:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{N}$$

Variance of a sample is defined by:

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

It is important to note that since differences are squared, the variance is always non-negative. In addition, we cannot compare these squared differences to the arithmetic mean since the units are different. For example, if we calculate the variance of annual compensation measured in *USD*, variance should be expressed as *USD*² while the mean exists in the original *USD* unit of measurement.

In R, the sample variance can be calculated using the `var()` function:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")

# Calculate sample variance for annual compensation
var(employees$annual_comp)

## [1] 1788038934
```

Sample statistics are the default in R. Since the population variance differs from the sample variance by a factor of $s^2(\frac{n-1}{n})$, it is simple to convert output from `var()` to the population variance:

```
# Store number of observations
n = length(employees$annual_comp)

# Calculate population variance for annual compensation
var(employees$annual_comp) * (n - 1) / n

## [1] 1786822581
```

Standard Deviation

The **standard deviation** is simply the square root of the variance.

The standard deviation of a population is defined by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{N}}$$

The standard deviation of a sample is defined by:

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

Since a squared value can be converted back to its original units by taking its square root, the standard deviation expresses variability around the mean in the variable's original units.

In R, the sample standard deviation can be calculated using the `sd()` function:

```
# Calculate sample standard deviation for annual compensation
sd(employees$annual_comp)
```

```
## [1] 42285.21
```

Since the population standard deviation differs from the sample standard deviation by a factor of $s\sqrt{\frac{n-1}{n}}$, it is simple to convert output from `sd()` to the population standard deviation:

```
# Calculate population standard deviation for annual compensation
sd(employees$annual_comp) * sqrt((n - 1) / n)
```

```
## [1] 42270.82
```

Quartiles

A **quartile** is a type of quantile that partitions data into four equally sized parts after ordering the data. Each quartile is equally sized with respect to the number of data points – not the range of values in each. Quartiles are also related to **percentiles**. For example, Q1 is the 25th percentile – the value at or below which 25% of values lie. Percentiles are likely more familiar than

quartiles, as percentiles show up in the height and weight measurements of babies, performance on standardized tests like the SAT and GRE, among other things.

The **Interquartile Range (IQR)** represents the difference between Q3 and Q1 cut point values (the middle two quartiles). The IQR is sometimes used to detect extreme values in a distribution; values less than $Q1 - 1.5 * IQR$ or greater than $Q3 + 1.5 * IQR$ are generally considered outliers.

In R, the `quantile()` function returns the values that bookend each quartile:

```
# Return quartiles for annual compensation
quantile(employees$annual_comp)
```

```
##      0%     25%     50%     75%    100%
## 62400  99840 137280 174200 208000
```

Based on this output, we know that 25% of people in our data earn annual compensation of 99,840 USD or less, 137,280 USD is the median annual compensation, and 75% of people earn annual compensation of 174,200 USD or less.

We can also return a specific percentile value using the `probs` argument in the `quantile()` function. For example, if we want to know the 80th percentile annual compensation value, we can execute the following:

```
# Return 80th percentile annual compensation value
quantile(employees$annual_comp, probs = .8)
```

```
##      80%
## 180960
```

In addition, the `summary()` function returns several common descriptive statistics for an object:

```
# Return common descriptives
summary(employees$annual_comp)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 62400   99840  137280  137054  174200  208000
```

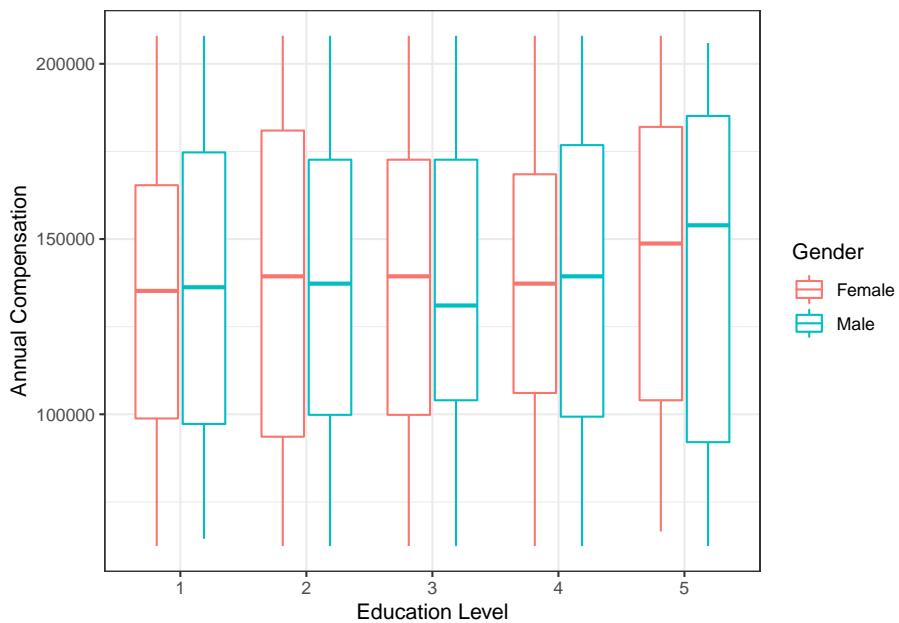
Box plots are a common way to visualize the distribution of data. Box plots are not usually found in presentations to stakeholders, since they are a bit more

technical and often require explanation, but these are very useful to analysts for understanding data distributions during the EDA phase.

Let's visualize the spread of annual compensation by education level and gender using the `geom_boxplot()` function from the `ggplot2` library:

```
# Load library
library(ggplot2)

# Produce box plots to visualize compensation distribution by
# education level and gender
ggplot2::ggplot(employees, aes(x = as.factor(ed_lvl), y =
# annual_comp, color = gender)) +
ggplot2::geom_boxplot() +
ggplot2::labs(x = "Education Level", y = "Annual Compensation") +
ggplot2::guides(col = guide_legend("Gender")) +
ggplot2::theme_bw()
```



Box plots can be interpreted as follows:

- Horizontal lines represent median compensation values.
- The box in the middle of each distribution represents the IQR.
- The end of the line above the IQR represents the threshold for outliers in the upper range: $Q3 + 1.5 * IQR$.

- The end of the line below the IQR represents the threshold for outliers in the lower range: $Q1 - 1.5 * IQR$.
- Data points represent outliers: $x > Q3 + 1.5 * IQR$ or $x < Q1 - 1.5 * IQR$.

While box plots are pervasive in statistically-oriented disciplines, they can be misleading. Figure @ref(fig:boxplot-barchart-compare) illustrates how information about the shape of a distribution can be lost on a box plot. The range with the highest frequency (0-9) is not as obvious in the box plot relative to the bar chart.

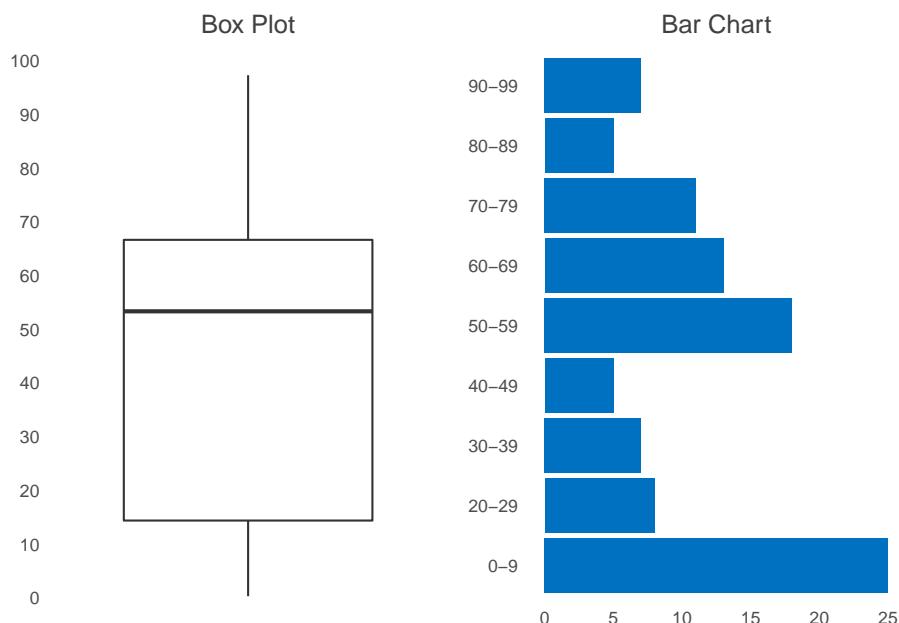


Figure 15: The number range with the highest frequency (0-9) is not as apparent with a box plot (left) relative to the bar chart (right).

Box plot alternatives such as **violin plots**, **jittered strip plots**, and **raincloud plots** are often more helpful in understanding data distributions. Figure @ref(fig:boxplot-alt) shows the juxtaposition of a raincloud plot against a box plot. While it may seem like an oxymoron, in this case the spread of data is clearer in the rain.

Skewness

Skewness is a measure of the horizontal distance between the mode and mean – a representation of symmetric distortion. In most practical settings, data are not normally distributed. That is, the data are skewed either positively (right-tailed distribution) or negatively (left-tailed distribution). The coefficient of

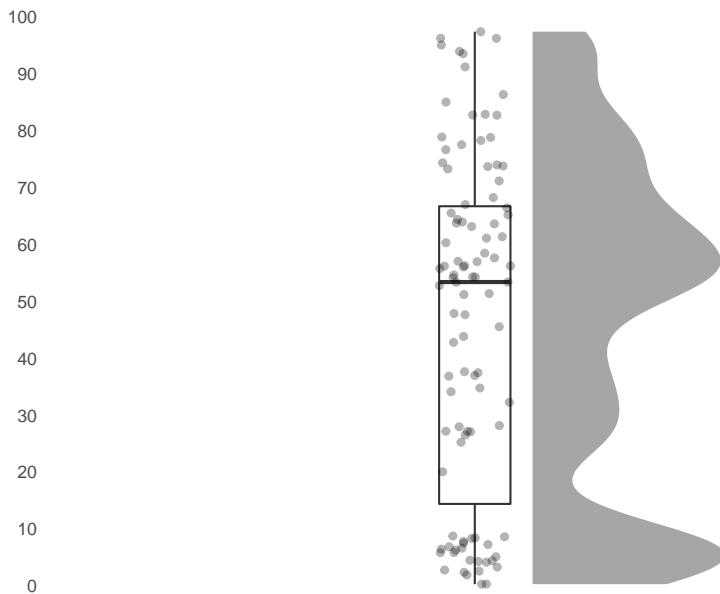


Figure 16: Raincloud plot superimposed on a box plot to illustrate the data distribution.

skewness is one of many ways in which we can ascertain the degree of skew in the data. The skewness of sample data is defined as:

$$Sk = \frac{1}{n} \frac{\sum_{i=1}^n (x_i - \bar{x})^3}{s^3}$$

A positive skewness coefficient indicates positive skew, while a negative coefficient indicates negative skew. The order of descriptive statistics can also be leveraged to ascertain the direction of skew in the data:

- Positive skewness: mode < median < mean
- Negative skewness: mode > median > mean
- Symmetrical distribution: mode = median = mean

Figure @ref(fig:skewness) illustrates the placement of these descriptive statistics in each of the three types of distributions:

The magnitude of skewness can be determined by measuring the distance between the mode and mean relative to the variable's scale. Alternatively, we can simply evaluate this using the coefficient of skewness:

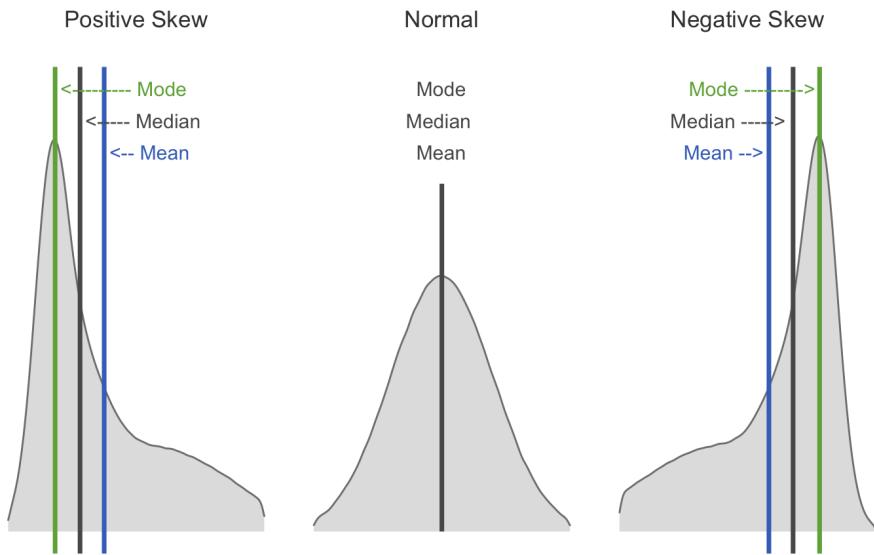


Figure 17: Skewness

- If skewness is between -0.5 - 0.5, the data are considered symmetrical.
- If skewness is between -0.5 and -1 or 0.5 and 1, the data are moderately skewed.
- If skewness is < -1 or > 1, the data are highly skewed.

Since there is not a base R function for skewness, we can leverage the moments library to calculate skewness:

```
# Load library
library(moments)

# Calculate skewness for org tenure, rounded to two significant
# figures via the round() function
round(moments::skewness(employees$org_tenure), 2)
```

[1] 2.27

Statistical Moments, after which this library was named, play an important role in specifying the appropriate probability distribution for a set of data. Moments are a set of statistical parameters used to describe the characteristics of a distribution. Skewness is the third statistical moment in the set; hence the sum of cubed differences and cubic polynomial in the denominator of the formula above. The complete set of moments comprises: (1) expected value or mean, (2) variance and standard deviation, (3) skewness, and (4) kurtosis.

We can verify that the `skewness()` function from the `moments` library returns the expected value (per the aforementioned formula) by validating against a manual calculation:

```
# Store components of skewness calculation
n = length(employees$org_tenure)
x = employees$org_tenure
x_bar = mean(employees$org_tenure)
s = sd(employees$org_tenure)

# Calculate skewness manually, rounded to two significant figures
# via the round() function
round(1/n * (sum((x - x_bar)^3) / s^3), 2)

## [1] 2.27
```

A skewness coefficient of 2.27 indicates that organization tenure is positively skewed. We can visualize the data to confirm the expected right-tailed distribution:

```
# Produce histogram to visualize sample distribution
ggplot2::ggplot() +
  ggplot2::aes(employees$org_tenure) +
  ggplot2::labs(x = "Organization Tenure", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..), fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
  ggplot2::theme_bw()
```

Kurtosis

While skewness provides information on the symmetry of a distribution, **kurtosis** provides information on the heaviness of a distribution's tails ("tailedness"). Kurtosis is the fourth statistical moment, defined by:

$$K = \frac{1}{n} \frac{\sum_{i=1}^n (x_i - \bar{x})^4}{s^4}$$

Note that the quartic functions characteristic of the fourth statistical moment are the only differences from the skewness formula we reviewed in the prior section (which featured cubic functions).

The terms **leptokurtic** and **platykurtic** are often used to describe distributions with light and heavy tails, respectively. "Platy-" in platykurtic is the same

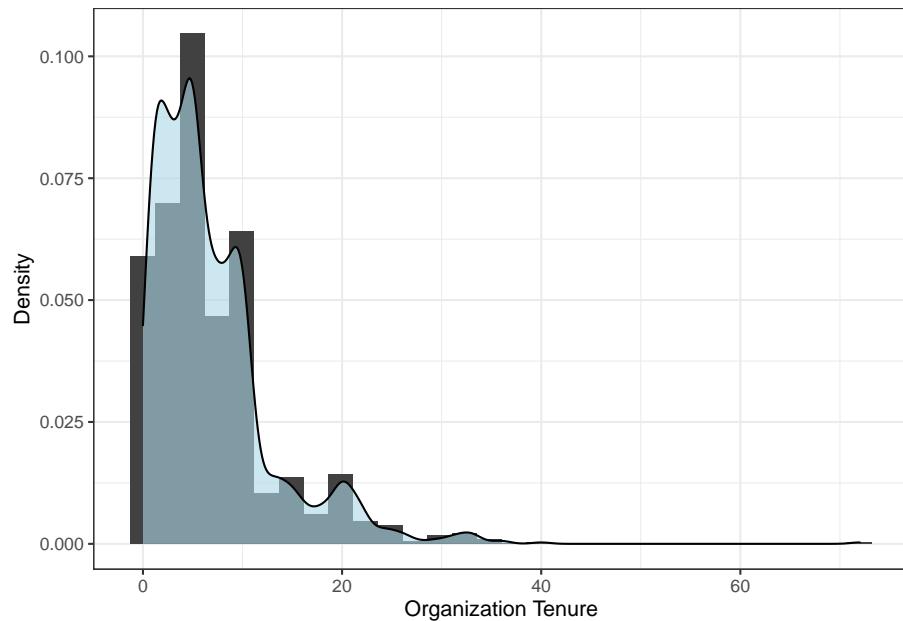


Figure 18: Organization tenure distribution

root as “platypus”, and I’ve found it helpful to recall the characteristics of the flat platypus when characterizing frequency distributions as platykurtic (wide and flat) vs. its antithesis, leptokurtic (tall and skinny). The normal (or Gaussian) distribution is referred to as a **mesokurtic** distribution in the context of kurtosis.

Figure @ref(fig:kurtosis) illustrates the three kurtosis categorizations:

Kurtosis is measured relative to a normal distribution. Normal distributions have a kurtosis coefficient of 3. Therefore, the kurtosis coefficient is greater than 3 for leptokurtic distributions and less than 3 for platykurtic distributions.

The moments library can also be used to calculate kurtosis in R:

```
# Calculate kurtosis for org tenure, rounded to one significant
→ figure
round(moments::kurtosis(employees$org_tenure), 1)
```

```
## [1] 13.4
```

We can verify that the `kurtosis()` function returns the expected value (per the aforementioned formula) by validating against a manual calculation:

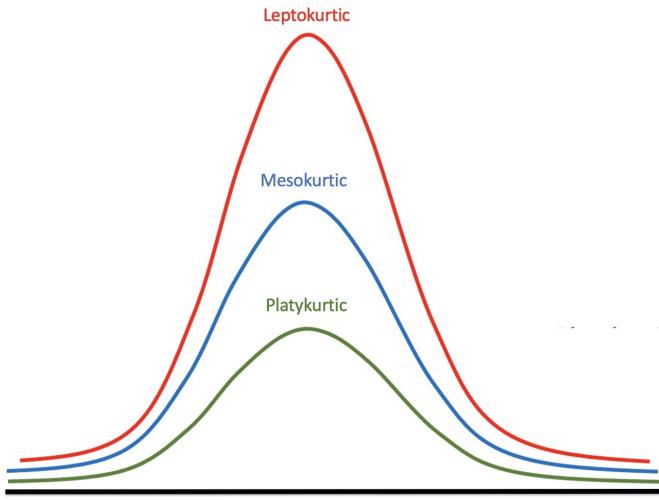


Figure 19: Kurtosis

```
# Calculate kurtosis manually, rounded to one significant figure
round(1/n * (sum((x - x_bar)^4) / s^4), 1)
```

```
## [1] 13.4
```

Our kurtosis coefficient of 13.4 indicates a leptokurtic distribution which is supported by the visual in Figure @ref(fig:org-tenure-distribution).

It is important not to characterize a distribution based on a single isolated metric; we need the complete set of statistical moments to fully understand the distribution of data.

Bivariate Analysis

As we covered, univariate analysis explores a *single* variable. This section will cover **bivariate analysis**, which explores statistical relationships between *two* variables.

Covariance

While variance provides an understanding of how values for a single variable vary, **covariance** is an unstandardized measure of how two variables vary together. Values can range from $-\infty$ to $+\infty$, and these values can be used to

understand the direction of the linear relationship between variables. Positive covariance values indicate that the variables vary in the same direction (e.g., tend to increase or decrease together), while negative covariance values indicate that the variables vary in opposite directions (e.g., when one increases, the other decreases, or vice versa).

Covariance of a sample is defined by:

$$cov_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n - 1}$$

It's important to note that while covariance aids our understanding of the direction of the relationship between two variables, we cannot use it to understand the strength of the association since it is unstandardized. Due to differences in variables' units of measurement, the strength of the relationship between two variables with large covariance could be weak, while the strength of the relationship between another pair of variables with small covariance could be strong.

In R, we can compute the covariance between a pair of numeric variables by passing the two vectors into the `cov()` function:

```
# Calculate sample covariance between annual compensation and age
← using complete observations (missing values will cause issues
← if not addressed)
cov(employees$annual_comp, employees$age, use = "complete.obs")
```

```
## [1] 9381.677
```

In this example, using the default method the covariance between annual compensation and age is 9,381.7. The positive value indicates that annual compensation is generally higher for older employees and lower for younger employees.

Just as we multiplied the sample variance by $(n - 1)/n$ to obtain the population variance, we can apply the same approach to convert the sample covariance returned by `cov()` to the population covariance:

```
# Calculate population covariance between annual compensation and
← age
cov(employees$annual_comp, employees$age, use = "complete.obs") *
← (n - 1) / n
```

```
## [1] 9375.295
```

Rather than looking at isolated pairwise relationships, we can produce a covariance matrix to surface pairwise associations among many variables by passing a data frame or matrix object into the `cov()` function:

```
# Generate a covariance matrix among select continuous variables
cov(subset(employees, select = c("annual_comp", "age",
  ↪ "org_tenure", "job_tenure", "prior_emplr_cnt",
  ↪ "commute_dist")), use = "complete.obs")
```

	annual_comp	age	org_tenure	job_tenure
## annual_comp	1.788039e+09	9381.6772019	-3921.9601469	-3693.1960749
## age	9.381677e+03	83.4550488	17.9255146	7.0467503
## org_tenure	-3.921960e+03	17.9255146	39.7967987	16.9797312
## job_tenure	-3.693196e+03	7.0467503	16.9797312	13.1271220
## prior_emplr_cnt	2.340406e+03	6.8377387	-1.8547177	-0.8213802
## commute_dist	1.067158e+04	-0.1248728	0.7746438	0.5535206
## prior_emplr_cnt		commute_dist		
## annual_comp	2340.4057552	10671.5790741		
## age		6.8377387	-0.1248728	
## org_tenure		-1.8547177	0.7746438	
## job_tenure		-0.8213802	0.5535206	
## prior_emplr_cnt		6.2400490	-0.5923586	
## commute_dist		-0.5923586	65.7212510	

Using the default Pearson method, the `cov()` function will return sample variances for each variable down the diagonal, since covariance is not applicable in the context of a variable with itself. We can confirm by producing the variance for age:

```
# Return sample variance for age
var(employees$age)
```

```
## [1] 83.45505
```

As expected, the variance for age ($s^2 = 83.5$) matches the value found in the age x age cell of the covariance matrix.

Correlation

Correlation is a scaled form of covariance. While covariance provides an unstandardized measure of the direction of a relationship between variables, correlation provides a standardized measure that can be used to quantify both the

direction and strength of bivariate relationships. Correlation coefficients range from -1 to 1, where -1 indicates a perfectly negative association, 1 indicates a perfectly positive association, and 0 indicates the absence of an association. **Pearson's product-moment correlation coefficient r** is defined by:

$$r_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

In R, Pearson's r can be calculated using the `cor()` function:

```
# Calculate the correlation between annual compensation and age
cor(employees$annual_comp, employees$age, use = "complete.obs")
```

```
## [1] 0.02428654
```

While we already know that the relationship between annual compensation and age is positive based on the positive covariance coefficient, Pearson's r of .02 indicates that the strength of the positive association is weak ($r = 0$ represents the absence of a relationship). Though there are no absolute rules for categorizing the strength of relationships, as thresholds often vary by domain, the following is a general rule of thumb for interpreting the strength of bivariate associations:

- Weak = Absolute value of correlation coefficients between 0 and .3
- Moderate = Absolute value of correlation coefficients between .4 and .6
- Strong = Absolute value of correlation coefficients between .7 and 1

There are several correlation coefficients, and the measurement scale of x and y determine the appropriate type:

Pearson's r can be used when both variables are measured on continuous scales or when one is continuous and the other is dichotomous (point-biserial correlation).

When one or both variables are ordinal, we can leverage **Spearman's ρ** or **Kendall's τ** , which are both standardized nonparametric measures of the association between one or two rank-ordered variables. Let's look at Spearman's ρ , which is defined as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

We can override the default Pearson method in the `cor()` function to implement a specific form of rank correlation using the `method` argument:

Measurement Scale		Correlation Coefficient
x	y	
Continuous	Continuous	Pearson's Product Moment
Continuous	Dichotomous	Point-Biserial
Continuous	Ordinal	Spearman or Kendall Rank
Dichotomous	Dichotomous	Phi, Contingency
Ordinal	Dichotomous	Rank-Biserial
Ordinal	Ordinal	Spearman or Kendall Rank

Figure 20: Proper applications of correlation coefficients

```
# Calculate the correlation between job level and education level
→ using Spearman's method
cor(employees$job_lvl, employees$ed_lvl, method = "spearman", use
→ = "complete.obs")
```

[1] 0.1074192

The ρ coefficient of .11 indicates that the positive association between job level and education level is weak. We could also pass `method = "kendall"` to this `cor()` function to implement Kendall's τ .

The **Phi Coefficient** (ϕ), sometimes referred to as the **mean square contingency coefficient** or **Matthews correlation** in ML, can be used to understand the association between two dichotomous variables.

For the 2x2 table for two random variables x and y depicted in Figure @ref(fig:phi-tbl), the ϕ coefficient is defined as:

$$\phi = \frac{(AD - BC)}{\sqrt{(A + B)(C + D)(A + C)(B + D)}}$$

To illustrate, let's examine whether there is a relationship between gender and performance after transforming performance from its ordinal form to a dichotomous variable (high vs. low performance). We can leverage the `psych` library to calculate ϕ in R:

	$y = 0$	$y = 1$
$x = 0$	A	B
$x = 1$	C	D

Figure 21: 2x2 table for random variables x and y

```
# Set females to 1 and everything else to 0
employees$gender_code <- ifelse(employees$gender == 'Female', 1,
                                0)

# Set stock options to 1 if level > 0
employees$stock_option_code <- ifelse(employees$stock_opt_lvl >
                                         0, 1, 0)

# Create a 2x2 contingency table
contingency_tbl <- table(employees$gender_code,
                           employees$stock_option_code)

# Calculate the Phi Coefficient between dichotomous variables
psych::phi(contingency_tbl)

## [1] -0.01
```

ϕ is essentially 0, which means stock options are distributed equitably across gender categories (good news!). While there are not differences in the proportion of males and females who receive at least some stock options, examining whether there is equity in the amount of stock grants and refreshes may be a good next step.

A correlation matrix can be produced to surface associations among many variables by passing a dataframe or matrix object into the `cor()` function:

```
# Generate a correlation matrix among select continuous variables
cor(subset(employees, select = c("annual_comp", "age",
                                "org_tenure", "job_tenure", "prior_emplr_cnt",
                                "commute_dist")), use = "complete.obs")
```

	annual_comp	age	org_tenure	job_tenure	prior_emplr_cnt
## annual_comp	1.0000000	0.02428654	-0.01470248	-0.02410622	0.02215688

```

## age           0.02428654  1.00000000  0.31104359  0.21290106   0.29963476
## org_tenure    -0.01470248  0.31104359  1.00000000  0.74288567  -0.11769547
## job_tenure    -0.02410622  0.21290106  0.74288567  1.00000000  -0.09075393
## prior_emplr_cnt 0.02215688  0.29963476  -0.11769547  -0.09075393   1.00000000
## commute_dist   0.03113059  -0.00168612  0.01514695  0.01884500  -0.02925080
##                           commute_dist
## annual_comp     0.03113059
## age             -0.00168612
## org_tenure      0.01514695
## job_tenure      0.01884500
## prior_emplr_cnt -0.02925080
## commute_dist    1.00000000

```

Based on this correlation matrix, most pairwise associations are weak with the exception of the relationship between `org_tenure` and `job_tenure` ($r = .7$). The values down the diagonal are 1 because these represent the correlation between each variable with itself. You may also notice that the information above and below the diagonal is identical and, therefore, redundant.

A great R library for visualizing correlation matrices is `corrplot`. Several arguments can be specified for various visual representations of the relationships among variables, as illustrated in Figure @ref(fig:corrplot-comp):

```

# Store correlation matrix to object M
M <- cor(subset(employees, select = c("annual_comp", "age",
  → "org_tenure", "job_tenure", "prior_emplr_cnt",
  → "commute_dist")), use = "complete.obs")

# Visualize correlation matrix
corrplot::corrplot(M, method = "color",
  type = "upper", order = "hclust", # Apply
  → hierarchical clustering for ordering
  → coefficients above the diagonal
  addCoef.col = "black", # Add correlation
  → coefficient
  tl.col = "grey", tl.srt = 45, # Label color
  → and rotation
  diag = FALSE # Hide correlation coefficient on
  → the principal diagonal
)

```

The `GGally` library produces a variety of useful information, including correlation coefficients, bivariate scatterplots, and univariate distributions, as illustrate in Figure @ref(fig:ggpairs-comp):

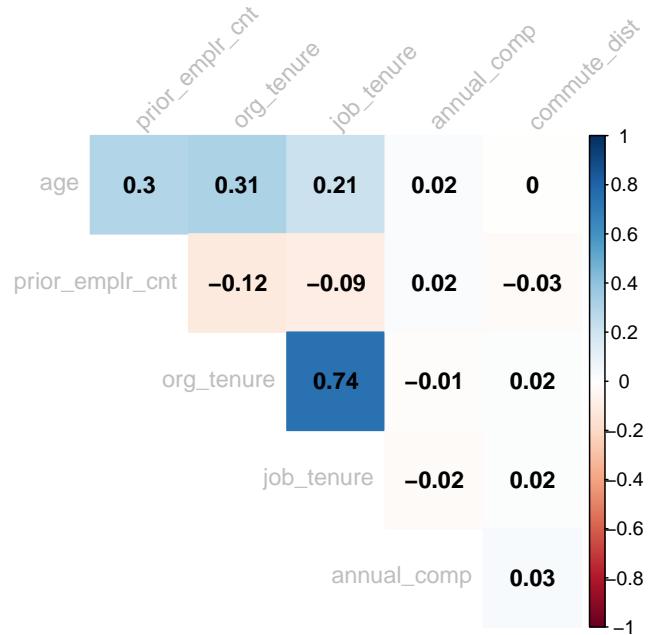


Figure 22: Corrplot correlation matrix

```
# Visualize correlation matrix
GGally::ggpairs(subset(employees, select = c("annual_comp",
  ~ "age", "org_tenure", "job_tenure", "prior_emplr_cnt",
  ~ "commute_dist")))
```

We may find that these bivariate associations look quite different for certain business areas or jobs, assuming departments and jobs were created at different points in the company's history. There is often a lot of noise in data at the broader company level, so understanding the nature and nuance of associations is important.

A classic example of this is a statistical phenomenon known as **Simpson's Paradox**, which is particularly common in the social sciences. Simpson's Paradox occurs when a correlation is present in subsets of data but disappears or reverses when the subsets are combined. The prototypical case is a study of gender discrimination at the University of California, Berkeley (Bickel, Hammel, & O'Connell, 1975). The overall data indicated that men were more likely than women to gain admission to the university's graduate programs, though there was no evidence of bias in any individual department. Upon closer evaluation, researchers found that women were more likely to apply to departments with lower acceptance rates while men tended to apply to less selective departments. The more nuanced relationships, such as the association between gender and the

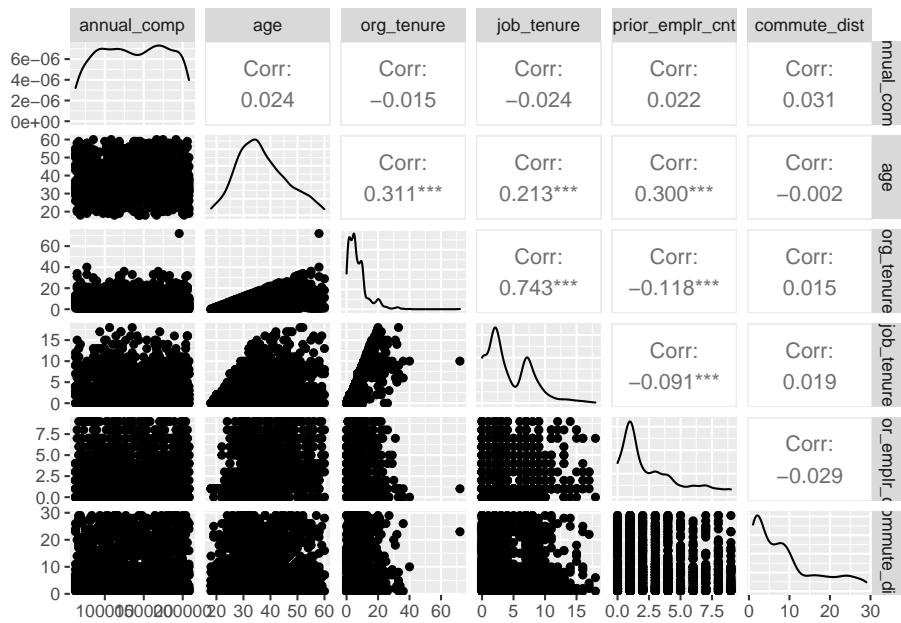


Figure 23: GGpairs bivariate correlations and data distributions

partitioning variable (department) in this example, can lead to incorrect conclusions when examining relationships only at the broader population level. We will explore how to control for this in the context of linear regression beginning in Chapter @ref(lm).

Finally, it is important to remember that correlation is not causation. Correlations can be spurious (variables related by chance), and drawing conclusions based on bivariate associations alone – especially in the absence of sound theoretical underpinnings – can be dangerous.

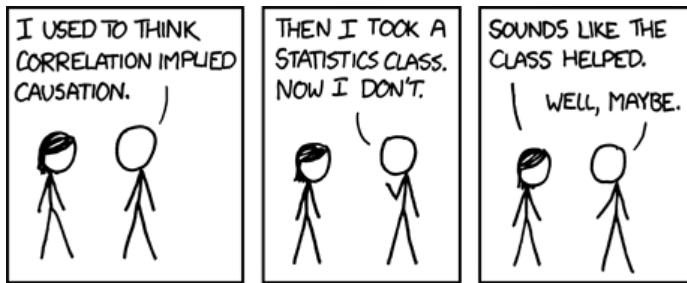


Figure 24: Correlation comic (Source: www.explainxkcd.com)

Figures @ref(fig:spur-corr-1) and @ref(fig:spur-corr-2) are two examples of

nearly perfect correlations between variables for which there is likely no true direct association:

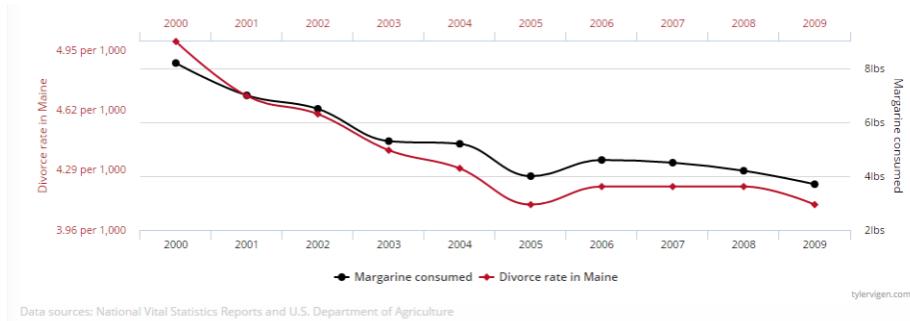


Figure 25: Correlation between Maine divorce rate and margarine consumption ($r = .99$)

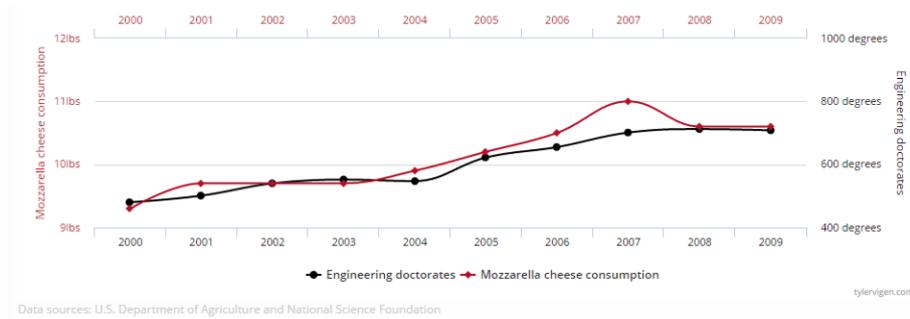


Figure 26: Correlation between mozzarella cheese consumption and civil engineering doctorate conferrals ($r = .96$)

Neither covariance nor correlation alone are sufficient for determining whether an observed association in sample data is also present in the population. For this, we need to graduate from descriptive to inferential statistics.

Review Questions

1. How does the mean and median compare with respect to sensitivity to extreme values (outliers)?
2. What does the standard deviation tell us about the spread of data, and how does it compare to the variance?
3. How does the order of the mean, median, and mode differ between positively and negatively skewed distributions?

4. Do large covariance coefficients always indicate strong bivariate associations? Why or why not?
5. What information is represented in box plots?
6. Do quartiles relate to percentiles?
7. What type of correlation coefficient should be used when evaluating the relationship between a pair of rank-ordered variables?
8. What type of correlation coefficient should be used when evaluating the relationship between a pair of dichotomous variables?
9. How would you characterize the shape of platykurtic, leptokurtic, and mesokurtic distributions?
10. When using the Pearson method, what do the values down the diagonal of a *covariance* matrix represent?

Statistical Inference

The objective of **inferential statistics** is to make inferences – with some degree of confidence – about a population based on available sample data. Several related concepts are fundamental to this goal and will be covered here.

Introduction to Probability

Randomness and uncertainty exist all around us. In **probability theory**, random phenomena refer to events or experiments whose outcomes cannot be predicted with certainty (Pishro-Nik, 2014). If you've taken a course in probability, there is a good chance you have considered the case of a fair coin flip – one of the most intuitive applications of probability. In the absence of information on how the coin is flipped, we cannot be certain of the outcome. What we can be certain of is that with a large number of coin flips, the proportion of heads will become increasingly close to 50%, or $\frac{1}{2}$.

The **Law of Large Numbers (LLN)** is an important theorem for building an intuitive understanding of how probability relates to the statistical inference concepts we will cover. In the case of a fair coin flip, it is possible to observe many consecutive heads by chance. This is because small samples can lend to anomalies. However, as the number of flips increases, we will undoubtedly observe an increasing number of tails; we expect a roughly equal number of heads and tails with a large enough number of flips.

Probability Distributions

Probability distributions are statistical functions that yield the probability of obtaining possible values for a random variable. Probabilities range from 0 to 1, where the probability of a definite event is 1 and the probability of an impossible event is 0. The **empirical probability (or experimental probability)** of an event is the fraction of times it occurred relative to the total number of repetitions. Since a probability distribution defines the likelihood of observing

all possible outcomes of an event or experiment, the sum of all probabilities for all possible values must equal 1.

For example, let's look at how org tenure is distributed across employees. We can understand the general shape of the distribution using descriptive statistics:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")

# Produce descriptive stats for org tenure
summary(employees$org_tenure)

##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
##    0.000   3.000   5.000    7.032   9.000  72.000
```

Comparing the higher mean value of 7.03 to the median value of 5 indicates there are larger values skewing the mean upward which is further evidenced in the large delta between Q3 and max values.

Beyond descriptives, visuals are often helpful in understanding a variable's distribution. As shown in Figure @ref(fig:org-tenure-dist), it is clear that org tenure is positively skewed, and understanding the shape (or spread) of this distribution enables us to identify which values are most likely in order to estimate the likelihood of different results:

```
# Load library
library(ggplot2)

# Visualize org tenure distribution
ggplot2::ggplot() +
  ggplot2::aes(employees$org_tenure) +
  ggplot2::labs(x = "Org Tenure", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..), fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
  ggplot2::theme_bw()
```

You can likely imagine the shape of probability distributions for many common events. If we consider the probability of employees exiting an organization, the outcome is binary. That is, employees either leave or stay; there are no options between these extremes. However, the distribution of performance scores will likely look quite different. Most organizations have expected – or even forced – distributions in which an average rating is awarded most frequently and low

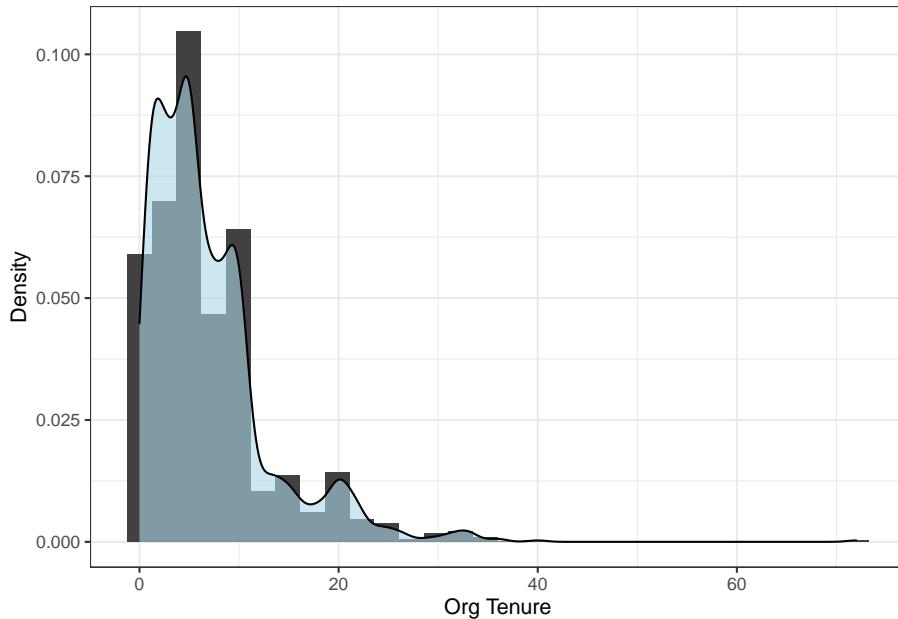


Figure 27: Organization tenure distribution

and high performance ratings less frequently. This would start to look more like a bell curve as the number of performance levels increases.

Just as we grouped variables into discrete and continuous categories in Chapter @ref(measure-sampl), this is also how probability distributions are categorized. If you read Chapter @ref(measure-sampl), you likely already have some a priori expectations about the characteristics of discrete and continuous distributions.

The shape of a probability distribution is defined by parameters, which represent its essential properties (e.g., measures of central tendency and spread). These probability distributions underpin the many types of statistical tests covered in this book.

Discrete Probability Distributions

Discrete probability distributions, also known as **Probability Mass Functions (PMF)**, can be leveraged to model different types of nominal and ordinal variables. Some common discrete distributions include:

- **Bernoulli**: probability of success or failure for a *single* observation with two outcomes
- **Binomial**: number of successes and failures in a *sequence* of independent observations with two outcomes (collection of Bernoulli trials)

- **Multinomial**: generalization of the binomial distribution for observations with more than two outcomes
- **Negative Binomial (Pascal)**: a version of the binomial distribution for a *fixed* number of observations (this is positively skewed despite what the name might suggest)
- **Poisson**: probability of a given number of events occurring over a specified period
- **Geometric**: special case of the negative binomial distribution that repeats observations until a success is observed (rather than a fixed number of times)

Several functions are available in R to simulate PMFs. The precise shape of a distribution depends on the parameters, but we will simulate and visualize these common PMFs to illustrate differences in the general shape of each. First, let's simulate the distributions by drawing 1,000 random values from each with a specified set of parameters:

```
# Set seed for reproducible random distribution
set.seed(1234)

# Simulate bernoulli distribution
bernoulli_dist <- rbinom(1000, 1, prob = .5)

# Simulate binomial distribution
# Notice the important difference relative to the Bernoulli
# simulation (100 trials vs. 1)
binomial_dist <- rbinom(1000, 100, prob = .5)

# Simulate negative binomial distribution
nbinomial_dist <- rnbinom(1000, 100, prob = .5)

# Simulate multinomial distribution with varying probabilities
# per level
multinomial_dist <- rmultinom(1000, 4, prob = c(.4, .3, .2, .6))

# Simulate poisson distribution
poisson_dist <- rpois(1000, 10)

# Simulate geometric distribution
geometric_dist <- rgeom(1000, prob = .2)
```

Next, we will visualize each distribution:

```

# Load library
library(ggpubr)

# Create user-defined function (UDF) to simplify probability
# distribution visualization
# Function arguments:
#   (1) data = object containing random distribution values
#   (2) type = 'discrete' or 'continuous' probability
#   distribution
#   (3) title = name of distribution
dist.viz <- function(data, type, title) {

  if (type == "discrete"){

    # Discrete distribution
    viz <- ggplot2::ggplot() +
      ggplot2::aes(data) +
      ggplot2::labs(title = paste(title), x = "x", y =
    "count") +
      ggplot2::geom_histogram(fill = "#414141") +
      ggplot2::theme_bw() +
      ggplot2::theme(plot.title = element_text(hjust = 0.5))

  } else {

    # Continuous distribution
    viz <- ggplot2::ggplot() +
      ggplot2::aes(data) +
      ggplot2::labs(title = paste(title), x = "x", y =
    "density") +
      ggplot2::geom_histogram(aes(y = ..density..), fill =
    "#414141") +
      ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
      ggplot2::theme_bw() +
      ggplot2::theme(plot.title = element_text(hjust = 0.5))
  }

  return(viz)
}

# Call UDF to build visualizations and store to objects
p_bernoulli <- dist.viz(data = bernoulli_dist, type = "discrete",
  title = "Bernoulli")
p_binomial <- dist.viz(data = binomial_dist, type = "discrete",
  title = "Binomial")

```

```

p_nbinomial <- dist.viz(data = nbinomial_dist, type = "discrete",
                         title = "Negative Binomial")
p_multinomial <- dist.viz(data = multinomial_dist, type =
                           "discrete", title = "Multinomial")
p_poisson <- dist.viz(data = poisson_dist, type = "discrete",
                       title = "Poisson")
p_geometric <- dist.viz(data = geometric_dist, type = "discrete",
                          title = "Geometric")

# Display distribution visualizations
ggpubr::ggarrange(p_bernoulli, p_binomial, p_nbinomial,
                   p_multinomial, p_poisson, p_geometric,
                   ncol = 3, nrow = 2)

```

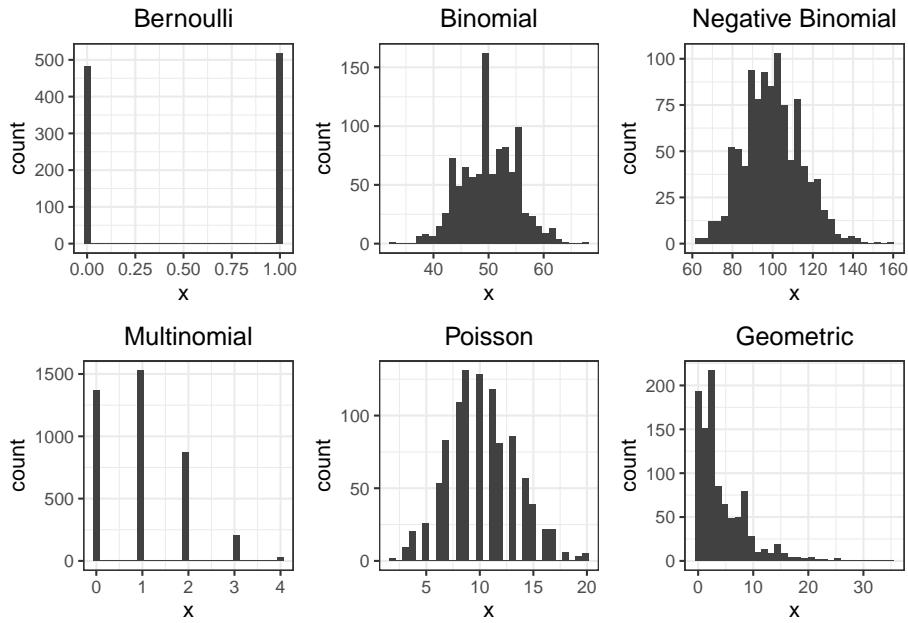


Figure 28: Discrete probability distributions

Continuous Probability Distributions

Continuous probability distributions, also known as **Probability Density Functions (PDF)**, can be leveraged to model different types of interval and ratio variables. Some common continuous distributions include:

- **Normal (Gaussian):** distribution characterized by a mean and standard deviation for which the mean, median, and mode are equal

- **Uniform:** values of a random variable with equal probabilities of occurring
- **Log-Normal:** normal distribution of log-transformed values
- **Student's t:** similar to the normal distribution but with thicker tails (approaches normal as n increases)
- **Chi-Square:** similar to the t distribution in that the shape approaches normal as n increases
- **F:** developed to examine variances from random samples taken from two independent normal populations

The normal distribution is colloquially known as a *bell curve*. It is important to note that a *normal distribution* and *standard normal distribution* are not one-and-the-same. The standard normal distribution is a special case of the normal distribution which has no free parameters; its parameters are always $\mu = 0$ and $\sigma = 1$. The parameters of a normal distribution are unspecified, and μ and σ can take on values other than 0 and 1, respectively.

A number of functions are available in R to simulate PDFs. While the precise shape of a distribution is parameter dependent, we will simulate and visualize these common PDFs to illustrate differences in the general shape of each. Let's first draw 1,000 random values from each distribution with a specified set of parameters:

```
# Set seed for reproducible random distribution
set.seed(1234)

# Simulate normal distribution
normal_dist <- rnorm(1000, mean = 50, sd = 5)

# Simulate log-normal distribution
lnormal_dist <- rlnorm(1000, meanlog = 0, sdlog = 1)

# Simulate uniform distribution
uniform_dist <- runif(1000, min = 1, max = 100)

# Simulate student's t distribution
t_dist <- rt(1000, df = 5)

# Simulate chi-square distribution
chisq_dist <- rchisq(1000, df = 5)

# Simulate F distribution
f_dist <- rf(1000, df1 = 5, df2 = 200)
```

Next, we will visualize each distribution. Since these continuous distributions are probability *density* functions, we will superimpose density plots over each:

```

# Call UDF to build visualizations and store to objects
# Note that as long as the arguments are in the order specified
#   in the function (see our UDF definition above), the argument
#   names do not need to be specified. To illustrate, we will
#   drop the argument names from these function calls:
p_normal <- dist.viz(normal_dist, "continuous", "Normal")
p_lnormal <- dist.viz(lnormal_dist, "continuous", "Log-Normal")
p_uniform <- dist.viz(uniform_dist, "continuous", "Uniform")
p_t <- dist.viz(t_dist, "continuous", "Student's T")
p_chisq <- dist.viz(chisq_dist, "continuous", "Chi-Square")
p_f <- dist.viz(f_dist, "continuous", "F")

# Display distribution visualizations
ggpubr::ggarrange(p_normal, p_lnormal, p_uniform, p_t, p_chisq,
  p_f,
  ncol = 3, nrow = 2)

```

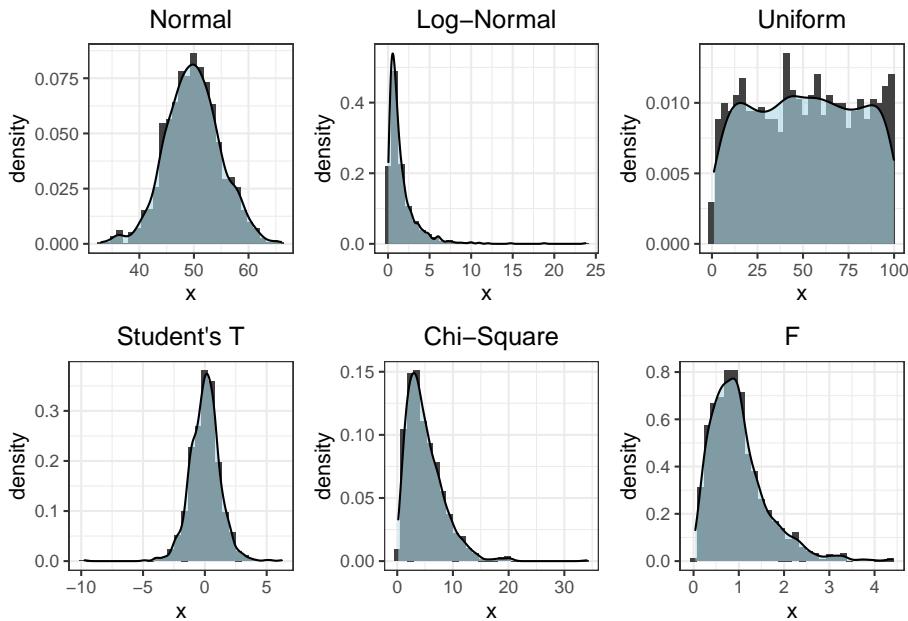


Figure 29: Continuous probability distributions

The distribution of data is critically important in statistics. The accuracy of many statistical tests is based on assumptions rooted in underlying data distributions, and violating these assumptions can result in serious errors due to misaligned probability distributions. Though there are many more discrete and continuous probability distributions, we will leverage several of these common

types to assess the likelihood of differences, effects, and associations in later chapters of this book.

Conditional Probability

Conditional probability reflects the probability conditioned on the occurrence of a previous event or outcome. For example, we may find that the proportion of heads is greater or less than $\frac{1}{2}$ with a large number of fair coin flips when the coin is consistently heads up when flipped. The outcome is, therefore, conditioned on the fixed – rather than random – positioning of the coin when flipped.

Formally, **Bayes' Theorem (alternatively, Bayes' Rule)** states that for any two events A and B wherein the probability of A is not 0 ($P(A) \neq 0$):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

where:

- A = an event
- B = another event
- $P(A|B)$ = conditional probability that event A occurs, given event B occurs (posterior)
- $P(B|A)$ = conditional probability that event B occurs, given event A occurs (likelihood)
- $P(B)$ = normalizing constant, constraining the probability distribution to sum to 1 (evidence)
- $P(A)$ = probability event A occurs before knowing if event B occurs (prior)

Bayes' Rule allows us to predict the outcome more accurately by conditioning the probability on known factors rather than assuming all events operate under the same conditions. Bayes' Rule is pervasive in people analytics, as the probability of outcomes can vary widely when conditioned on a person's age, tenure, education, job, perceptions, relationships, and many other factors. For example, if we consider a company with 100 terminations over a 12-month period and average headcount of 1,000, the probability of attrition not conditioned on any other factor is 10%, or $\frac{1}{10}$. Aside from trending this probability over time to identify if overall attrition is becoming more or less of a concern, this isn't too helpful at the company level. However, if we condition the probability of attrition on an event – such as a recent manager exit – and find that the probability of attrition among those whose manager has left in the last six months is 70%, or $\frac{7}{10}$, this is far more actionable (and concerning).

The **Monty Hall Problem** is an excellent example of how our intuition is often at odds with the laws of conditional probability.

In the classic game show, Let's Make a Deal, Monty Hall asks contestants to choose one of three closed doors. Behind one door is a prize while the other two doors contain nothing. After the contestant selects a door, Monty opens one of the other two doors which does not contain a prize. At this point, there are two closed doors: the door the contestant selected and another for which the content remains unknown. All that is known at this point is that the prize is behind one of the two closed doors.

It is at this juncture that Monty introduces a twist by asking if the contestant would like to switch doors. Most assume that the two closed doors have an equal (50/50) chance of containing the prize, because we generally think of probabilities as independent, random events. However, this is incorrect. Contestants who switch from their original selection have a 66% chance (rather than 50%) of winning. This may be counterintuitive, because the brain wants to reduce the problem to a simple coin flip. There is a major difference between the Monty Hall problem and a coin flip; for two outcomes to have the same probability, randomness and independence are required. In the case of the Monty Hall problem, neither assumption is satisfied.

When all three doors are closed, each has the same probability of being selected. The probability of choosing the door with a prize is .33. Monty's knowledge of the door containing the prize does not impact the probability of selecting the winning door. This is because the choice is completely random given we have no information that would increase the probability of a door containing the prize. The process is no longer random when Monty uses his insider knowledge about the prize's location and opens a door he knows does not contain the prize. The probabilities change. Since Monty will never show the door containing the prize, he is careful to always open a door that has nothing behind it. If he was not constrained by the requirement to not reveal the prize's location and instead chose to open one of the remaining doors at random, the probabilities would be equal (and he may end up opening the door that contains the prize).

Seeing is believing, so let's prove this with a simulation in R:

```
# Set seed for reproducible simulations
set.seed(12345)

# Set number of simulations
trials = 10000

# Store switch/keep decisions
decisions = c("switch", "keep")

# Store integer for each door
doors = 1:3

# Initialize empty data frame for results
```

```

results = NULL

for (n in 1:trials){

  for (decision in decisions){

    # Select correct door
    correct_door <- sample(doors, 1, replace = T)

    # Contestant chooses a door at random
    selected_door <- sample(doors, 1, replace = T)

    # Open door that was neither selected by the contestant nor
    # contains the prize
    # Choose one door to open if multiple remain without the
    # prize (i.e., the contestant didn't initially select the
    # door containing the prize)
    remaining_doors <- which(!doors == correct_door & !doors ==
    selected_door)
    open_door <- sample(remaining_doors, 1, replace = T)

    # Contestant makes decision to switch doors or keep with the
    # originally selected door
    selected_door <- ifelse(decision == "switch", which(!doors ==
    selected_door & !doors == open_door), selected_door)

    # Store results in data frame
    results <- rbind(results, cbind.data.frame(
      trial = n,
      decision = decision,
      result = ifelse(correct_door ==
        selected_door, "win", "lose")))
  }
}

# Calculate percentage difference in wins for switch vs. keep
# decisions
switch_wins <- nrow(results[results$decision == "switch" &
  results$result == "win", ]) / nrow(results) * 100
keep_wins <- nrow(results[results$decision == "keep" &
  results$result == "win", ]) / nrow(results) * 100
round((switch_wins - keep_wins) / keep_wins * 100, 0)

```

[1] 45

As we can see, wins occur nearly 50% more often when contestants switch doors. This exercise hopefully demonstrates the importance of conditional probability and statistical assumptions like randomness. Also, if ever you find yourself playing Let's Make a Deal, switch doors.

Central Limit Theorem

The **Central Limit Theorem (CLT)** is a mainstay of statistics and probability and fundamental to understanding the mechanics of statistical inference.

Coined by a French-born mathematician named Abraham De Moivre in the 1700s, the CLT states that given a sufficiently large sample size, the average of independent random variables tends to follow a normal (or Gaussian) distribution as the number of samples increases. The distribution of sample means approaches a normal distribution regardless of the shape of the population distribution from which the samples are drawn. This is important because the normal distribution has properties that can be used to test the likelihood that an observed value, difference, or relationship in a sample is also present in the population.

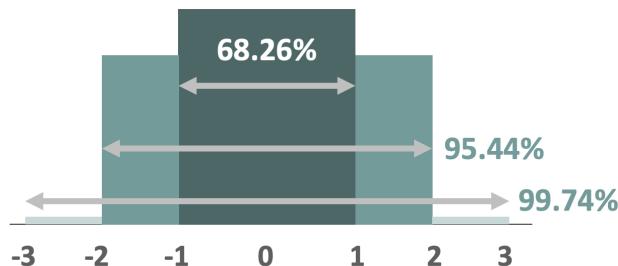


Figure 30: The Empirical Rule

Let's begin with an intuitive example of CLT. Imagine that we have a reliable way to measure how fun a population is on a 100-point scale, where 100 indicates maximum fun (life of the party) and 1 indicates maximum boringness. Consider that a small statistics conference is in progress at a nearby convention center, and there are 40 statisticians in attendance. In a separate room at the same convention center, there is also a group of 40 random people (non-statisticians) who are gathered to discuss some less interesting topic. Our job is to walk into one of the rooms and determine – based on the *fun* factor alone – whether we have entered the statistics conference or the other, less interesting gathering of non-statisticians.

Instinctively, we already know the statisticians will be more fun than the other group. However, let's assume we need the mean fun score and standard de-

viation of these two groups for this example. The group of statisticians have, on average, a fun score of 85 with a standard deviation of 2, while the group of non-statisticians are a bit less fun with a mean score of 65 and a standard deviation of 4. With a known population mean and standard deviation, the **standard error (SE)** – the standard deviation of sample means – provides the ability to calculate the probability that the sample (the room of 40 people) belongs to the population of interest (fellow statisticians).

The SE is defined by:

$$SE = \frac{\sigma}{\sqrt{n}}$$

Herein lies the beauty of the CLT: roughly 68 percent of sample means will lie within one standard error of the population mean, roughly 95 percent within two standard errors of the population mean, and roughly 99 percent within three standard errors of the population mean. Therefore, any room whose members have an average fun score that is not within two standard errors of the population mean (between 81 and 89 for our statisticians) is statistically unlikely to be the group of statisticians for which we are searching. This is because in less than 5 in 100 cases could we randomly draw a *reasonably sized* sample of statisticians with average funniness so extremely different from the population average.

Because small samples lend to anomalies, we could – by chance – select a single person who happens to fall in the tails (extremely boring or extremely fun); however, as the sample size increases, it becomes more and more likely that the observed average reflects the average of the larger population. It would be virtually impossible (in less than 1 in 100 cases) to draw a random sample of statisticians from the population with average funniness that is not within three standard errors of the population mean (between 79 and 91). Therefore, if we find that the room of people have an average fun score of 75, we will likely have far more fun in the other room!

Let's now see the CLT in action by simulating a random uniform population distribution from which we can draw random samples. Remember, the shape of the population distribution does not matter; we could simulate an Exponential, Gamma, Poisson, Binomial, or other distribution and observe the same behavior.

```
# Set seed for reproducible random distribution
set.seed(1234)

# Generate uniform population distribution with 1000 values
# ranging from 1 to 100
rand.unif <- runif(1000, min = 1, max = 100)
```

```
# Calculate population mean
mean(rand.unif)

## [1] 51.22007

# Calculate population variance
N = length(rand.unif)
var(rand.unif) * (N - 1) / N

## [1] 830.3155

# Produce histogram to visualize population distribution
ggplot2::ggplot() +
  ggplot2::aes(rand.unif) +
  ggplot2::labs(x = "x", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..), fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
  ggplot2::theme_bw()
```

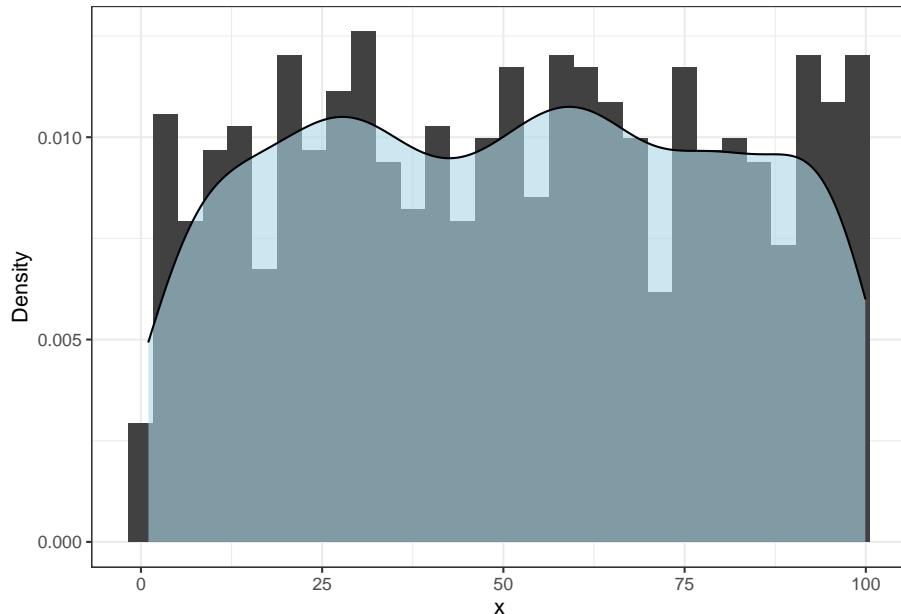


Figure 31: Uniform population distribution ($N = 1000$)

As expected, these randomly generated data are uniformly distributed. Next, we will draw 100 random samples of various sizes and plot the average of each.

```

# Define number of samples to draw from population distribution
samples <- 10000

# Populate vector with sample sizes
sample_n <- c(1:5,10,25,50)

# Initialize empty data frame to hold sample means
sample_means = NULL

# Set seed for reproducible random samples
set.seed(456)

# For each n, draw random samples
for (n in sample_n) {

  for (draw in 1:samples) {

    # Store sample means in data frame
    sample_means <- rbind(sample_means, cbind.data.frame(
      n = n,
      x_bar = mean(sample(rand.unif, n,
                           replace = TRUE, prob = NULL))))
  }
}

# Produce histograms to visualize distributions of sample means,
# grouped by n-count
sample_means |> ggplot2::ggplot() +
  ggplot2::aes(x = x_bar, fill = n) +
  ggplot2::labs(x = "x-bar", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..),
  fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha =
  0.6) +
  ggplot2::theme_bw() +
  ggplot2::facet_wrap(~n)

```

Per the CLT, we can see that as n increases, the sample means become more normally distributed.

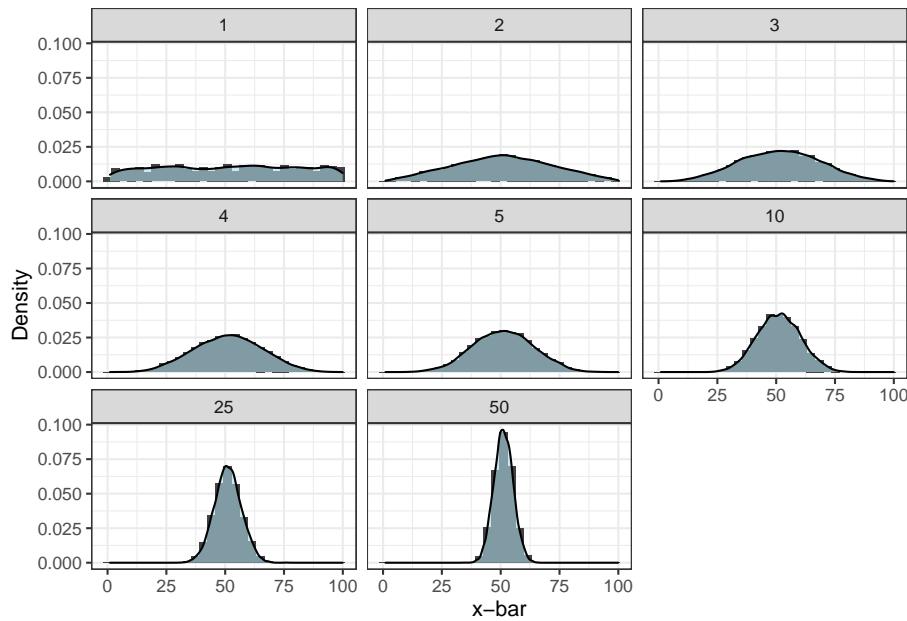


Figure 32: Distribution of 10,000 sample means of varied size

Confidence Intervals

A **Confidence Interval (CI)** is a range of values that likely contains the value of an unknown population parameter. These unknown population parameters are often μ or σ , though we will also leverage CIs in later chapters for regression coefficients, proportions, rates, and differences.

If we draw random samples from a population, we can compute a CI for each sample. Building on the CLT, for a given confidence level (usually 95%, though 99% or 90% are sometimes used), the specified percent of sample intervals is expected to include the estimated population parameter. For example, for a 95% CI we would expect 19 in every 20 (or 95 in every 100) intervals across the samples to include the true population parameter. This is illustrated in Figure @ref(fig:conf-int):

It is important to note that CIs should not be applied to the distribution of sample values; CIs relate to *population parameters*. A common misinterpretation of a CI is that it represents an interval within which a certain percent of sample values exists. Because this misinterpretation is so prevalent, there is a good chance you will be tested on your understanding of CIs when applying to positions involving statistical analyses!

The standard error is fundamental to estimating CIs. While the standard devi-

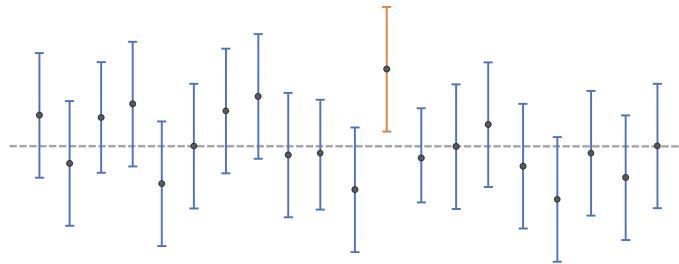


Figure 33: Intervals for 95 percent confidence level

ation is a measure of variability for a random variable, the variability captured by the SE reflects how well a sample represents the population. Since sample statistics will approach the actual population parameters as the size of the sample increases, the SE and sample size are inversely related; that is, the SE decreases as the sample size increases.

Since the CLT is fundamental to inferential statistics, let's validate that our simulated distribution of sample means adheres to the properties of normally distributed data per the Empirical Rule:

```
# Store sample means with n = 50
x_bars <- sample_means[sample_means$n == 50, "x_bar"]

# Store sample size
n <- length(x_bars)

# Calculate percent of sample means within +/- 2 SEs
length(subset(x_bars, x_bars < mean(x_bars) + 2 * sd(x_bars) &
  x_bars > mean(x_bars) - 2 * sd(x_bars))) / n * 100

## [1] 95.35
```

95% of sample means are within 2 SEs, which is what we expect per the characteristics of the normal distribution.

```
# Calculate percent of sample means within +/- 3 SEs
length(subset(x_bars, x_bars < mean(x_bars) + 3 * sd(x_bars) &
  x_bars > mean(x_bars) - 3 * sd(x_bars))) / n * 100

## [1] 99.79
```

Nearly all of the sample means are within 3 SEs, indicating that it would be highly unlikely – nearly impossible even – to observe a sample mean *from the same population* that falls outside this interval.

Now, let's illustrate the relationship between CIs and standard errors using sample data from our uniform population distribution. In our example, both μ and σ are known and our sample size n is at least 30; therefore, we can use a **Z-test** to calculate the 95% CI. A z score of 1.96 corresponds to the 95% CI for a two-tailed distribution; that is, we are looking for significantly different values in either the larger or smaller direction. The 95% CI represents the range of values we would expect to include μ in at least 95 of 100 random samples taken from the population.

The CI in this case is defined by:

$$CI = \bar{x} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

Let's randomly take $n = 100$ from the population, and compute sample statistics to estimate the 95% CI:

```
# Set seed for reproducible random samples
set.seed(456)

# Sample 100 values from uniform population distribution
x <- sample(rand.unif, 100, replace = TRUE, prob = NULL)

# Calculate 95% CI
ci95_lower_bound <- mean(x) - 1.96 * (sd(x) / sqrt(100))
ci95_upper_bound <- mean(x) + 1.96 * (sd(x) / sqrt(100))
```

```
# Print lower bound for 95% CI
ci95_lower_bound
```

```
## [1] 47.90733
```

```
# Print upper bound for 95% CI
ci95_upper_bound
```

```
## [1] 58.98773
```

Our known μ is 51.2, which is covered by our 95% CI (47.9 - 59.0). Per the CLT, in less than 5% of cases would we expect to draw a random sample from the population that results in a 95% CI which does not include μ . Note that

our CI narrows with larger samples since our confidence that the range includes μ increases with more data.

Next, let's look at a 99% CI. We will enter 2.576 for z :

```
# Calculate 99% CI
ci99_lower_bound <- mean(x) - 2.576 * (sd(x) / sqrt(100))
ci99_upper_bound <- mean(x) + 2.576 * (sd(x) / sqrt(100))
```

```
# Print lower bound for 99% CI
ci99_lower_bound
```

```
## [1] 46.16612
```

```
# Print upper bound for 99% CI
ci99_upper_bound
```

```
## [1] 60.72893
```

Like the 95% CI, this slightly wider 99% CI (46.2 - 60.7) also includes our μ of 51.2.

If σ is not known, and/or we have a small sample ($n < 30$), we need to use a **t-test** to calculate the CIs. In a people analytics setting, the reality is that population parameters are often unknown. For example, if we knew how engagement scores vary in the employee population, there would be no need to survey a sample of employees and make inferences about said population.

As we will see, the *t*-test underpins many statistical tests and models germane to the people analytics discipline since we are often working with small datasets, so it is important to understand the mechanics. As shown in Figure @ref(fig:t-distribution), the *t* distribution is increasingly wider and shorter relative to the normal distribution as the sample size decreases; this is also characteristic of the sampling distribution of means for smaller samples we observed in our CLT example. Specifically, **degrees of freedom (df)** is used to determine the shape of the probability distribution. Degrees of freedom represents the number of observations in the data that are free to vary when estimating statistical parameters, which is a function of the sample size ($n - 1$). For example, if we could choose 1 of 5 projects to work on each day between Monday and Friday, we would only be able to *choose* 4 out of the 5 days; on Friday, only 1 project would remain to be selected, so our degrees of freedom (the number of days in which we have a choice between projects) would be 4.

When estimating the CI for smaller samples, we need to leverage the wider, more platykurtic *t* distribution to achieve greater accuracy. Therefore, the CI for a two-tailed test in this case is defined by:

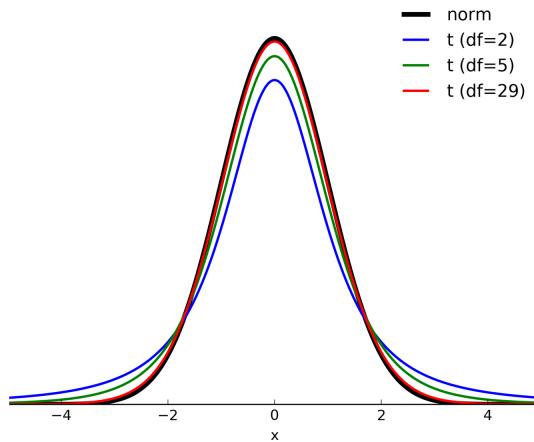


Figure 34: t distribution shape by degrees of freedom

$$CI = \bar{x} \pm t_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

Let's compare CIs calculated using a t -test to those calculated using the Z-Test. While a fixed z score can be used for each CI level when $n > 30$, the t statistic varies based on both the CI level and df . Though R will determine the correct t statistic for us, let's reference the table shown in Figure @ref(fig:t-crit) to manually lookup the t statistic:

For illustrative purposes, let's draw a smaller sample of $n = 25$ from our uniform population distribution and calculate the 95% CI using the t statistic from the table ($df = 24$). The t statistic for this CI and df is 2.064:

```
# Set seed for reproducible random samples
set.seed(456)

# Sample 25 values from uniform population distribution
x <- sample(rand.unif, 25, replace = TRUE, prob = NULL)

# Calculate 95% CI
ci95_lower_bound <- mean(x) - 2.064 * (sd(x) / sqrt(25))
ci95_upper_bound <- mean(x) + 2.064 * (sd(x) / sqrt(25))

# Print lower bound for 95% CI
ci95_lower_bound
```

cum.prob	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.995}$
one-tail	0.1	0.05	0.025	0.005
two-tails	0.2	0.1	0.5	0.01
df				
1	3.078	6.314	12.706	63.657
2	1.886	2.920	4.303	9.925
3	1.638	2.353	3.182	5.841
4	1.533	2.132	2.776	4.604
5	1.476	2.015	2.571	4.032
6	1.440	1.943	2.447	3.707
7	1.415	1.895	2.365	3.499
8	1.397	1.860	2.306	3.355
9	1.383	1.833	2.262	3.250
10	1.372	1.812	2.228	3.169
11	1.363	1.796	2.201	3.106
12	1.356	1.782	2.179	3.055
13	1.350	1.771	2.160	3.012
14	1.345	1.761	2.145	2.977
15	1.341	1.753	2.131	2.947
16	1.337	1.746	2.120	2.921
17	1.333	1.740	2.110	2.898
18	1.330	1.734	2.101	2.878
19	1.328	1.729	2.093	2.861
20	1.325	1.725	2.086	2.845
21	1.323	1.721	2.080	2.831
22	1.321	1.717	2.074	2.819
23	1.319	1.714	2.069	2.807
24	1.318	1.711	2.064	2.797
25	1.316	1.708	2.060	2.787
26	1.315	1.706	2.056	2.779
27	1.314	1.703	2.052	2.771
28	1.313	1.701	2.048	2.763
29	1.311	1.699	2.045	2.756
30	1.310	1.697	2.042	2.750
Z	1.282	1.645	1.960	2.576
	80%	90%	95%	99%
	Confidence Interval			

Figure 35: Critical values of Student's t distribution

```
## [1] 35.24305

# Print upper bound for 95% CI
ci95_upper_bound
```

```
## [1] 59.60959
```

As expected, the 95% CI using the t statistic is much wider (35.2 - 59.6), acknowledging the increased uncertainty in estimating population parameters given the limited information in this smaller sample. To increase our confidence to the 99% level, the interval widens even further (30.9 - 63.9):

```
# Calculate 99% CI
ci99_lower_bound <- mean(x) - 2.797 * (sd(x) / sqrt(25))
ci99_upper_bound <- mean(x) + 2.797 * (sd(x) / sqrt(25))
```

```
# Print lower bound for 99% CI
ci99_lower_bound
```

```
## [1] 30.91633
```

```
# Print upper bound for 99% CI
ci99_upper_bound
```

```
## [1] 63.93631
```

Hypothesis Testing

Hypothesis testing is how we leverage CIs to test whether a significant difference or relationship exists in the data. Sir Ronald Fisher invented what is known as the null hypothesis, which states that there is no relationship/difference; disprove me if you can! The null hypothesis is defined by:

$$H_0 : \mu_A = \mu_B$$

The objective of hypothesis testing is to determine if there is sufficient evidence to reject the null hypothesis in favor of an alternative hypothesis. The null hypothesis always states that there is *nothing* of significance. For example, if we want to test whether an intervention has an effect on an outcome in a population, the null hypothesis states that there is no effect. If we want to

test whether there is a difference in average scores between two groups in a population, the null hypothesis states that there is no difference.

An alternative hypothesis may simply state that there is a difference or relationship in the population, or it may specify the expected direction (e.g., Population A has a significantly *larger* or *smaller* average value than Population B; Variable A is *positively* or *negatively* related to Variable B). Therefore, alternative hypotheses are defined by:

$$H_A : \mu_A \neq \mu_B$$

$$H_A : \mu_A < \mu_B$$

$$H_A : \mu_A > \mu_B$$

Alpha

The **alpha** level of a hypothesis test, denoted by α , represents the probability of obtaining observed results due to chance if the null hypothesis is true. In other words, α is the probability of rejecting the null hypothesis (and therefore claiming that there is a significant difference or relationship) when in fact we should fail to reject it because there is insufficient evidence to support the alternative hypothesis.

α is often set at .05 but is sometimes set at a more rigorous .01, depending upon the context and tolerance for error. An α of .05 corresponds to a 95% CI (1 - .05), and .01 to a 99% CI (1 - .01). With non-directional alternative hypotheses, we must divide α by 2 (i.e., we could observe a significant result in either tail of the distribution), while one-tailed tests position the rejection region entirely within one tail based on what is being hypothesized.

At the .05 level, we would conclude that a finding is statistically significant if the chance of observing a value at least as extreme as the one observed is less than 1 in 20 if the null hypothesis is true. Recall that we observed this behavior with our simulated distribution of sample means. While we could observe more extreme values by chance with repeated attempts, in less than 1 in every 20 times would we expect a 95% CI that does not capture μ . Moreover, in less than 1 in every 100 times should we expect a sample with a 99% CI that does not capture μ .

Type I & II Errors

A **Type I Error** is a false positive, wherein we conclude that there is a significant difference or relationship when there is not. A **Type II Error** is a

false negative, wherein we fail to capture a significant finding. α represents our chance of making a Type I Error, while β represents our chance of making a Type II Error. I once had a professor explain that committing a Type I error is a shame, while committing a Type II error is a pity, and I've found this to be a helpful way to remember what each type of error represents.

	H_0 True	H_0 False
Reject H_0	Type I Error	Correct Rejection
Fail to Reject H_0	Correct Decision	Type II Error

Figure 36: Type I and II errors

p-Values

In statistical tests, the **p-value** is referenced to determine whether the null hypothesis can be rejected. We generally rely on the availability of a **theoretical null distribution** to obtain a p -value associated with a particular test statistic. The p -value represents the probability of obtaining a result at least as extreme as the one observed if the null hypothesis is true. As a general rule, if $p < .05$, we can confidently reject the null hypothesis and conclude that the observed difference or relationship was unlikely a chance observation.

While statistical significance helps us understand the probability of observing results by chance when there is no difference or effect in the population, it does not tell us anything about the size of the difference or effect. Analysis should never be reduced to inspecting p -values; in fact, p -values have been the subject of much controversy among researchers and practitioners in recent years. Later chapters will cover how to interpret results of statistical tests to surface the story and determine if there is anything ‘practically’ significant among statistically significant findings.

Bonferroni Correction

One caveat when leveraging a p -value to determine statistical significance is that when multiple testing is performed – that is, multiple tests using the same sample data – the probability of a Type I error increases by a factor equivalent to the number of tests performed. Though there is not agreement among statisticians about how (or even whether) the p -value threshold for statistical significance needs to be adjusted to account for this increased risk, we will cover a conservative approach known as the **Bonferroni Correction** to mitigate this risk.

Thus far, we have only discussed statistical significance in the context of a **per analysis error rate** – that is, the probability of committing a Type I error for a single statistical test. However, when two or more tests are being conducted on the same sample, the **familywise error rate** is an important factor in determining statistical significance. The familywise error rate reflects the fact that as we conduct more and more analyses on the same sample, the probability of a Type I error across the set (or family) of analyses increases. The familywise error rate can be calculated by:

$$\alpha_{FW} = 1 - (1 - \alpha_{PC})^C,$$

where c is equal to the number of comparisons (or statistical tests) performed, and α_{PC} is equal to the specified per analysis error rate (usually .05). For example, if $\alpha = .05$ per analysis, the probability of a Type I error with three tests on the same data increases from 5% to 14.3%: $1 - (1 - .05)^3 = .143$.

The most common method of adjusting the familywise error rate down to the specified per analysis error rate is the Bonferroni Correction. To implement this correction, we can simply divide α by the number of analyses performed on the dataset – such as $\alpha/3 = .017$ in the case of three analyses with $\alpha = .05$. This means that for each statistical test, we must achieve $p < .017$ to report a statistically significant result. An alternative which allows us to achieve the same number of statistically significant results is to multiply the unadjusted per analysis p -values for each statistical test by the number of tests. For example, if we run three statistical tests and receive $p = .014$, $p = .047$, and $p = .125$, we would achieve one significant result with the first method ($p < .017$) as well as with the alternative since the first statistical test satisfies the per analysis error rate ($p < .05$): $p = .014 * 3 = .042$.

Perneger (1998) is one of many who oppose the use of the Bonferroni Correction, suggesting that these “adjustments are, at best, unnecessary and, at worst, deleterious to sound statistical inference.” The Bonferroni Correction is controversial among researchers because while applying the correction reduces the chance of a Type I error, it also increases the chance of a Type II error. Because this correction makes it more difficult to detect significant results, it is rare to find such a correction reported in published research – though research often involves multiple testing on the same sample. Perneger suggests that simply describing the statistical tests that were performed, and why, is sufficient for dealing with potential problems introduced by multiple testing.

Statistical Power

Whereas α is the probability of a Type I error, **Beta** β is the opposite: the probability of accepting H_0 when it is false (Type II error).

β is related to the *power* of the analysis, which is calculated by $1 - \beta$ and reflects our ability to detect a difference or relationship if one exists. If a study has 80% power, for example, it has an 80% chance of detecting an effect if one actually exists in the population. **Power analysis** helps with defining the optimal n -count for detecting a population effect in sample data (i.e., correctly rejecting a false H_0). Increasing the power of a statistical test decreases the probability that we will fail to detect a significant effect present in the population.

At this point, it should be intuitive that larger samples increase our chances of detecting significant results when they exist. As we observed in the *t*-test example, CIs for small samples ($n < 30$) are quite wide relative to those for large samples; therefore, the power of the analysis to detect significance is limited given how different the values of x must be to observe non-overlapping CIs.

Before diving into the mechanics of power analysis, it is important to understand the three important – and interrelated – considerations in hypothesis tests that influence whether effects are real or a product of random sampling error:

- **Effect size:** Larger differences and stronger relationships are less likely random sampling error
- **Sample size:** Larger samples can detect smaller differences and weaker relationships (though they may be too small or weak to be meaningful)
- **Variability:** Greater variability in the data is likely to result in differences that are attributable to random sampling error

Power analysis may be thought of as an optimization problem. The goal is to achieve a large enough sample size to detect meaningful effects – but not wastefully large as data collection can be expensive – whilst protecting against an underpowered analysis with a low probability of detecting an important effect (Type II error).

To estimate the sample size needed to achieve a given power level, one must use domain expertise to specify parameters. The effect size parameter varies based on the statistical test but when in doubt, we can use Cohen's (1988) conventional effect sizes which are defined in Figure @ref(fig:conv-eff-size-tbl). The effect size for a particular test can also be retrieved using the `cohen.ES()` function. For example, the following command returns 0.25 as the medium effect size threshold for ANOVA: `cohen.ES(test = "anov", size = "medium")`.

Let's illustrate by calculating the sample size required for a one-way ANOVA that involves four groups. We will set $\alpha = .05$ and specify an 80% chance ($power = .8$) of detecting a moderate population effect. We can leverage the `pwr` library in R to perform power analysis. Executing `?pwr` will provide package documentation that clarifies what function to execute to calculate the sample size requirement for various statistical tests.

Test	Effect Size		
	Small	Medium	Large
Tests for Proportions (<i>p</i>)	0.2	0.5	0.8
Tests for Means (<i>t</i>)	0.2	0.5	0.8
Chi-Square (<i>chisq</i>)	0.1	0.3	0.5
Correlation (<i>r</i>)	0.1	0.3	0.5
ANOVA (<i>anov</i>)	0.1	0.25	0.4
General Linear Model (<i>f2</i>)	0.02	0.15	0.35

Figure 37: Cohen's conventional effect sizes by statistical test

```
# Load library
library(pwr)

# Calculate sample size for one-way ANOVA
# k = 4 indicates we plan to compare 4 groups
# f = .25 indicates we wish to detect a medium effect
pwr::pwr.anova.test(k = 4, f = .25, sig.level = .05, power = .8)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##              k = 4
##              n = 44.59927
##              f = 0.25
##      sig.level = 0.05
##              power = 0.8
##
## NOTE: n is number in each group
```

The power analysis for this one-way ANOVA shows that we need a minimum of $n = 45$ *within each* of the four groups to achieve an 80% chance of detecting a medium population effect across the four groups when setting $\alpha = .05$.

Review Questions

1. What are some examples of a null hypothesis?
2. What is the difference between Type I and Type II errors?
3. What is the primary purpose of inferential statistics, and how does it differ from descriptive statistics?
4. What is the Central Limit Theorem (CLT), and why is it important?
5. Is randomness a requirement for probabilistic methods? Why or why not?
6. What does the Bonferroni Correction seek to achieve?
7. What is a confidence interval (CI)?
8. What are some examples of how the context influences what level of confidence is appropriate for statistical significance testing? Should we always use a 95 CI?
9. When population parameters are unknown, which test would be appropriate for testing the following null hypothesis: $\mu_A = \mu_B$?
10. According to the Empirical Rule, 95% of normally distributed data lie within how many standard deviations of the mean?

Analysis of Differences

There are many statistical tests that can be used to test for differences *within* or *between* two or more groups. This chapter will cover common contexts for differences in people analytics and the tests applicable to each.

Parametric vs. Nonparametric Tests

In the context of data measured on a continuous scale (quantitative), we will cover parametric tests along with their nonparametric counterparts. When the hypothesis relates to average (mean) differences and n is large, **parametric tests** are preferred as they generally have more statistical power. **Nonparametric tests** are **distribution-free tests** that do not require the population's distribution to be characterized by certain parameters, such as a normal distribution defined by a mean and standard deviation. Nonparametric tests are great for qualitative data since the distribution of non-numeric data cannot be characterized by parameters.

Beyond ensuring the data were generated from a random and representative process as discussed in Chapter @ref(measure-sampl), as well as following the data screening procedures outlined in Chapter @ref(data-prep) (e.g., addressing concerning outliers), parametric tests of differences *generally* feature three key assumptions:

1. **Independence:** Observations within each group are independent of each other
2. **Homogeneity of Variance:** Variances of populations from which samples were drawn are equal
3. **Normality:** Residuals must be normally distributed (with mean of 0) within each group

While homogeneity of variance assumes the variances across multiple groups are equal, parametric tests are generally robust to violations of equal variances when the sample sizes are large. Also, you may recall that μ and σ are sufficient

to characterize a population distribution when data are situated symmetrically around the mean. However, the mean can be sensitive to outliers so if outliers are present in the data, the median may be a better way of representing the data's center (i.e., nonparametric tests); just remember that the use of nonparametric tests requires hypotheses to be modified to adjust for *median* – rather than *mean* – centers.

You may be wondering whether the magical elixir that is the CLT, which we covered in Chapter @ref(inf-stats), influences our ability to utilize parametric tests with respect to the assumption of normality. It's important to remember that the normal distribution properties under the CLT relate to the *sampling distribution of means* – not to the distribution of the population or to the data for an individual sample. The CLT is important for estimating population parameters, but it does not transform a population distribution from nonnormal to normal. If we know the population distribution is nonnormal (e.g., ordinal, nominal, or skewed data), nonparametric tests should be considered. This is why we used Spearman's correlation coefficient – a nonparametric test – in Chapter @ref(desc-stats) to evaluate the relationship between job level and education; these ordinal data are not normally distributed in the population.

Differences in Discrete Data

Nonparametric tests are generally best when working with data measured on a discrete scale since these data do not come from normally distributed populations. The two most commonly used tests to analyze variables measured on a discrete scale are the nonparametric *Chi-square test* and *Fisher's exact test*.

	Chi-Square Test	Fisher's Exact Test
Sample Size	Large	Small
Accuracy	Approximate	Exact
Groups	2 or more	2
Interpretation	Pearson residuals	Odds ratio

Figure 38: Chi-square and Fisher exact test criteria for discrete data

Both tests organize data within 2x2 **contingency tables** which enables us to understand interrelations between variables.

Chi-Square Test

The **Chi-Square Test of Independence** evaluates patterns of observations to determine if categories occur more frequently than we would expect by chance. The Chi-square statistic is defined by:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i},$$

where O_i is the observed value, and E_i is the expected value.

H_0 states that each variable is independent of one another (i.e., there is no relationship). In addition to the χ^2 test statistic, df for the contingency table defined by $df = (\text{rows} - 1) * (\text{columns} - 1)$, is required to determine whether we reject or fail to reject H_0 .

While there is not consensus on the minimum sample size for this test, it is important to note that the χ^2 statistic follows a chi-square distribution *asymptotically*. This means we can only calculate accurate p -values for larger samples, and a general rule of thumb is that the expected value for each cell needs to be at least 5. The challenge with small n -counts is illustrated in Figure @ref(fig:chisq-dist); the chi-square distribution approaches a vertical line as df drops below 5.

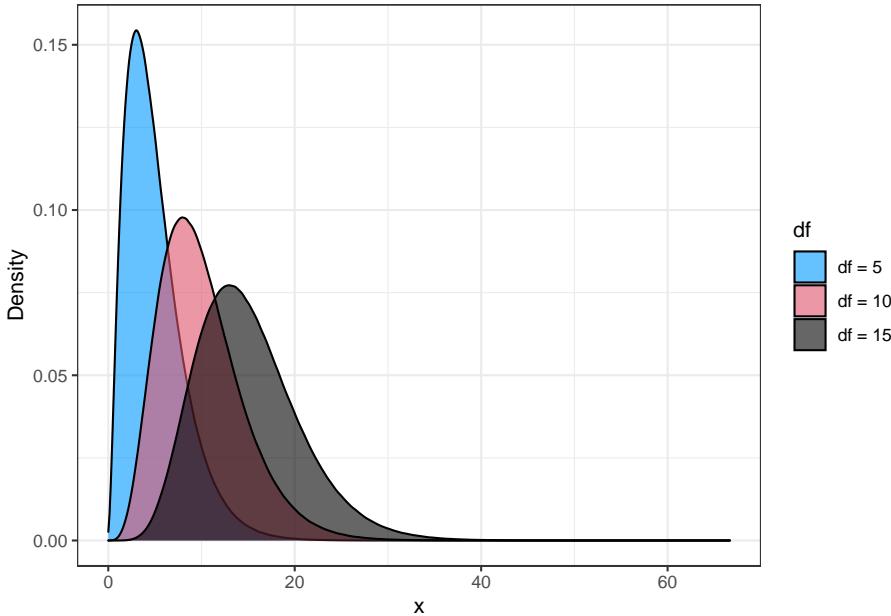


Figure 39: Chi-square distributions by degrees of freedom (df)

We will demonstrate how to perform a chi-square test of independence by evaluating whether exit rates are independent of whether an employee works overtime.

Let's first construct a 2x2 contingency table using the `table()` function:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")

# Create contingency table
cont_tbl <- table(employees$active, employees$overtime)

cont_tbl

##          No Yes
## No    110 127
## Yes   944 289
```

A **mosaic plot** is a great way to visualize the delta between expected and observed frequencies for each cell. This can be produced using the `mosaicplot()` function from the `graphics` library:

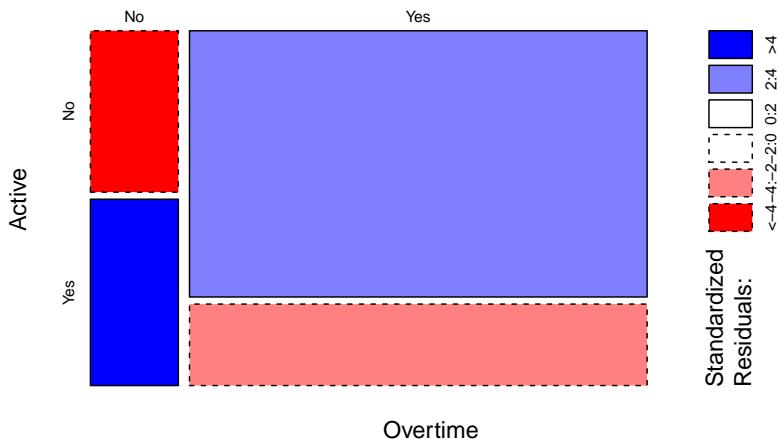


Figure 40: Mosaic plot of residuals for overtime x active status

In Figure @ref(fig:mosaic-plot), blue indicates that the observed value is higher than the expected value, while red indicates that the observed value is lower than the expected value. Based on this plot, there appears to be some meaningful patterns and departures from expected values in both the high and low directions. There are more inactive employees than expected in the *overtime* group, and more active employees than expected in the *no overtime* group. These large standardized residuals are indicative of meaningful relationships between the two categorical variables.

Let's run the chi-square test of independence to determine whether these residuals are statistically significant. This test can be performed in R by passing the contingency table into the `chisq.test()` function:

```
# Perform chi-square test of independence
chisq.test(cont_tbl)

##
## Pearson's Chi-squared test with Yates' continuity correction
##
## data: cont_tbl
## X-squared = 87.564, df = 1, p-value < 2.2e-16
```

Based on the results, exit rates are not independent of overtime ($\chi^2(1) = 87.56, p < .05$). Therefore, there is a statistically significant relationship between an employee working overtime and the rate at which they change from active to inactive statuses – confirming what was evident in the mosaic plot.

Fisher's Exact Test

When the sample size is small, **Fisher's Exact Test** can be used to calculate the *exact* p -value rather than the approximation characteristic of many statistical tests such as the chi-square test.

H_0 for Fisher's exact test is the same as H_0 for the chi-square test of independence: There is no relationship between the two categorical variables (i.e., they are independent). We can perform Fisher's exact test using the `fisher.test()` function in R:

```
# Perform Fisher's exact test
fisher.test(cont_tbl)

##
## Fisher's Exact Test for Count Data
##
## data: cont_tbl
## p-value < 2.2e-16
```

```
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
## 0.1969101 0.3572582
## sample estimates:
## odds ratio
## 0.2654384
```

Note the *odds ratio* shown in this output. The **odds ratio** represents the ratio of positive to negative cases, which is the ratio of overtime for active and inactive workers in this example. The odds ratio is defined by:

$$OR = \frac{a * d}{b * c}$$

An odds ratio of 1 indicates no difference in overtime frequency between active and inactive workers. Figure @ref(fig:contingency-tbl) illustrates the cells for the odds ratio calculation for the 2x2 contingency table of overtime for active and inactive workers.

	No Overtime	Overtime	
Inactive	a	b	(a + b)
Active	c	d	(c + d)
(a + c)	(b + d)	n	

Figure 41: 2x2 contingency table of overtime for active and inactive workers

Since the 95% *CI* for the odds ratio does not include 1, we reject the null hypothesis and conclude that exit rates are related to working overtime; this is consistent with results from the chi-square test of independence. Since overtime was indicated far more often for inactive workers than for active workers, it is no surprise that the denominator of our ratio is larger than the numerator (i.e., $OR < 1$).

As discussed in Chapter @ref(desc-stats), we can produce a ϕ coefficient to understand the strength of the association by passing the contingency table into the `phi` function from the `psych` library:

```
# Calculate the Phi Coefficient
psych::phi(cont_tbl)
```

```
## [1] -0.25
```

The relationship between active status and overtime is negative, and the strength of the relationship is weak ($\phi = -.25$).

Another common method of measuring the strength of the association between two categorical variables is **Cramer's V**, which ranges from 0 (no association) to 1 (strong association). In the interest of not muddying the waters with an exhaustive set of alternative methods, the implementation won't be covered.

Differences in Continuous Data

A variety of parametric and nonparametric tests are available for evaluating differences between variables measured on a continuous scale. Figure @ref(fig:continuous-tests) provides a side-by-side of these parametric and corresponding nonparametric tests of differences.

	Parametric	Nonparametric
1 DV 2 Categories Independent Samples	Independent <i>t</i> Test	Mann-Whitney U Test
1 DV 2 Categories Dependent Samples	Paired <i>t</i> Test	Wilcoxon Signed Rank Test
1 DV >2 Categories Independent Samples	One-Way ANOVA	Kruskal-Wallis H Test
1 DV >2 Categories Dependent Samples	Repeated Measures ANOVA	Friedman Test

Figure 42: Parametric and nonparametric tests of differences for continuous data

Independent Samples t-Test

When evaluating differences between two independent samples, social psychology researchers generally select from two tests: *Student's t-test* and *Welch's t-test*. There are other alternatives, such as *Yuen's t-test* and a *bootstrapped t-test*, but these are less commonly reported in scholarly social science journals and will not be covered in this book.

The **Student's t-test**, which was introduced in Chapter @ref(inf-stats), is a parametric test whose assumptions of equal variances seldom hold in people analytics. **Welch's t-test** is generally preferred to the Student's *t*-test because

it has been shown to provide better control of Type 1 error rates when homogeneity of variance is not met, whilst losing little robustness (e.g., Delacre, Lakens, & Leys, 2017). When n is equal between groups, the Student's t -test is known to be robust to violations of the equal variance assumption, as long as n is sufficiently high to accurately estimate parameters and the underlying distribution is not characterized by high skewness and kurtosis.

Let's explore the mechanics of independent samples t -tests. Figure @ref(fig:mean-group-diff) illustrates mean differences (MD) for nine Welch's t -tests based on random sample data generated from independent normal populations. Remember that statistical power increases with a large n , as t distributions approximate a normal distribution with larger df . In the context of analysis of differences, this translates to an increase in the likelihood of detecting statistical differences in the means of two distributions. Note that for the two cases where both MD and n are relatively small, mean differences are not statistically significant ($p \geq .05$). You may also notice that as the t -statistic approaches 0, statistical differences become less likely since a smaller t -statistic indicates a smaller difference between mean values.

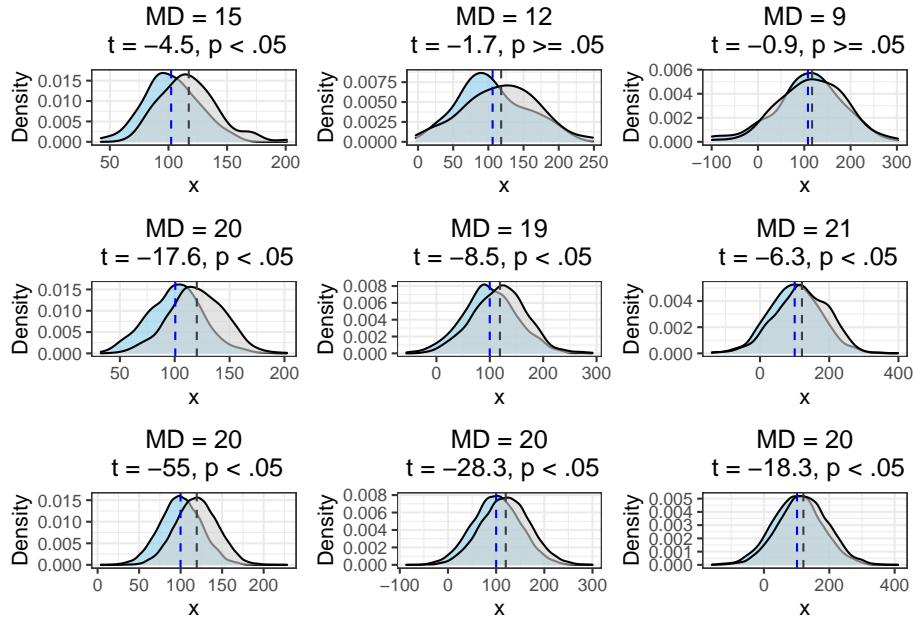


Figure 43: Density plots comparing distributions with a mean of 100 (blue) and 120 (grey) with sample sizes of 100, 1,000, and 10,000 (rows) and standard deviations of 25, 50, and 75 (columns). Dashed vertical lines represent distribution means.

Next, we will walk through the steps involved in performing Welch's t -test. Let's first visualize the distribution of data for each group using box plots:

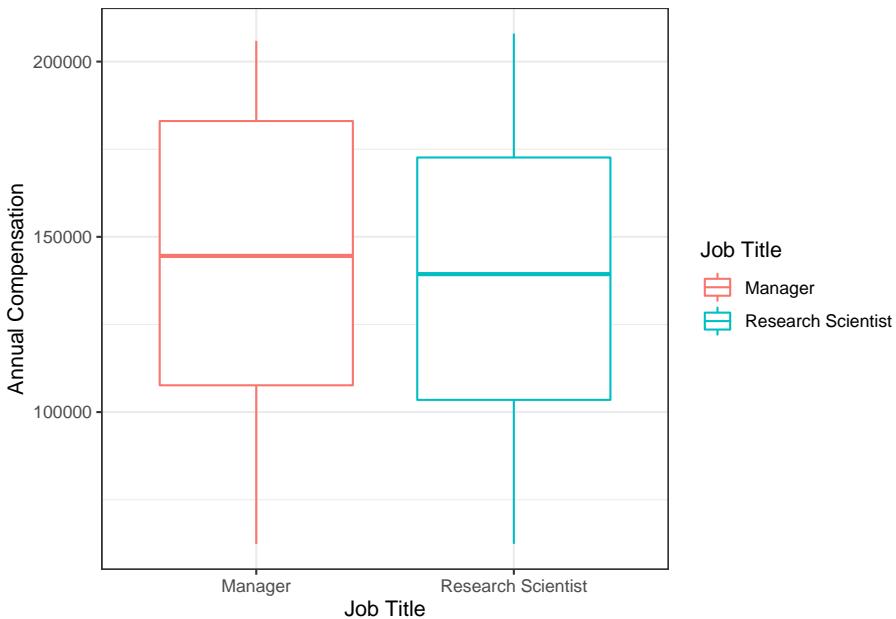


Figure 44: Annual compensation distributions for managers and Research Scientists

While the median – rather than the mean which is being evaluated with Welch's t -test – is shown in these box plots, this is a great way to visually inspect whether there are meaningful differences in the distribution of data between groups (in addition to identifying outliers). We could of course use density plots or histograms as an alternative. As we can see, annual compensation for employees with a Manager job title tends to be slightly higher than for those with a Research Scientist job title, and the variance in annual compensation appears to be fairly consistent between the groups.

There are several alternatives to a visual inspection of normality, such as the χ^2 **Goodness-of-Fit test** (Snedecor & Cochran, 1980), **Kolmogorov-Smirnov (K-S) test** (Chakravarti, Laha, & Roy, 1967), or **Shapiro-Wilk test** (Shapiro & Wilk, 1965). The general idea is consistent for each of these tests: compare observed data to what would be expected if data are sampled from a normally distributed population. The χ^2 Goodness-of-Fit test compares the *count* of data points across the range of values relative to what would be expected in each for a sample with the same dimensions taken from a normal distribution. For example, if data are sampled from a normally distributed population, it follows that roughly half the values should exist below the mean and half above the mean. The K-S test evaluates how the observed *cumulative* distribution compares to the properties of a normal *cumulative* distribution. The Shapiro-Wilk test is based on *correlations* between observed and expected data.

We will test for normality using the Shapiro-Wilk test. The null hypothesis for the Shapiro-Wilk test is that the data are normally distributed, so a high *p*-value indicates that the assumption of normality is satisfied (i.e., failure to reject the null hypothesis of normally distributed data). We can use the `with()` function together with the `shapiro.test()` function to run this test in R:

```
# Compute Shapiro-Wilk test of normality for each group
with(employees, shapiro.test(annual_comp[job_title ==
  →  'Manager']))
```



```
## 
##  Shapiro-Wilk normality test
##
## data: annual_comp[job_title == "Manager"]
## W = 0.93546, p-value = 0.000103
```



```
with(employees, shapiro.test(annual_comp[job_title == 'Research
  →  Scientist']))
```



```
## 
##  Shapiro-Wilk normality test
##
## data: annual_comp[job_title == "Research Scientist"]
## W = 0.96002, p-value = 3.427e-07
```

Based on these tests, distributions of annual compensation for Managers and Research Scientists are non-normal ($p < .05$).

While we should not proceed with performing Welch's *t*-test due to unequal variances, let's do so merely to illustrate how the test is implemented in R. To perform Welch's *t*-test in R, we can simply pass into the `t.test()` function a numeric vector for each of the two groups.

```
# Create compensation vectors for two jobs
comp_mgr <- unlist(subset(employees, job_title == 'Manager',
  →  select = annual_comp))
comp_rsci <- unlist(subset(employees, job_title == 'Research
  →  Scientist', select = annual_comp))

# Run Welch's t-test
t.test(comp_mgr, comp_rsci)
```

```
##
```

```
## Welch Two Sample t-test
##
## data: comp_mgr and comp_rsci
## t = 0.22623, df = 159.55, p-value = 0.8213
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -8860.685 11153.244
## sample estimates:
## mean of x mean of y
## 139900.8 138754.5
```

If the data adhered to the assumptions of Welch's t -test, we would conclude that the mean difference between annual compensation for Managers ($\bar{x} = 139,901$) and Research Scientists ($\bar{x} = 138,755$) is not significant ($t(159.55) = .23$, $p = .82$).

Note that we can access specific metrics from this output by storing results to an object and then referencing specific elements by name or index:

```
# This assigns each element of results from Welch's t-test to an
→ indexed position in the object
t_rslts <- t.test(comp_mgr, comp_rsci)

t_rslts$statistic # t-statistic

##          t
## 0.2262262

t_rslts$parameter # df

##          df
## 159.5544

t_rslts$p.value # p-value

## [1] 0.8213151

t_rslts$method # type of t-test

## [1] "Welch Two Sample t-test"
```

When object elements are referenced by index, the element name is displayed in the output to clarify what the metric represents:

```
t_rslts[1] # t-statistic
```

```
## $statistic
##      t
## 0.2262262
```

```
t_rslts[2] # df
```

```
## $parameter
##      df
## 159.5544
```

```
t_rslts[3] # p-value
```

```
## $p.value
## [1] 0.8213151
```

```
t_rslts[9] # type of t-test
```

```
## $method
## [1] "Welch Two Sample t-test"
```

Given $df = 159.55$, you may be wondering how df is calculated for Welch's t -test given that thus far, we have only discussed the basic df calculation outlined in Chapter @ref(inf-stats); namely, $df = n - 1$. Welch's t -test uses the **Welch-Satterthwaite equation** for df (Satterthwaite, 1946; Welch, 1947). This equation approximates df for a linear combination of independent sample variances; which means that if samples are not independent, this approximation may not be valid. The Welch-Satterthwaite equation is defined by:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1-1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2-1}\left(\frac{s_2^2}{n_2}\right)^2}$$

Cohen's d is a standardized measure of the difference between two means that helps us understand the size (or *practical* significance) of observed mean differences. Cohen's d is defined by:

$$d = \frac{\bar{x}_1 - \bar{x}_2}{s_p},$$

where s_p represents the pooled standard deviation defined by:

$$s_p = \sqrt{\frac{s_1^2 + s_2^2}{2}}$$

Cohen's d can be produced using the `cohen.d()` function from the `effsize` package in R. The following thresholds can be referenced as a *general* rule of thumb for interpreting effect size:

- **Small** = 0.2
- **Medium** = 0.5
- **Large** = 0.8

```
# Load library
library(effsize)

# Perform Cohen's d
effsize::cohen.d(comp_mgr, comp_rsci)

## 
## Cohen's d
##
## d estimate: 0.0273669 (negligible)
## 95 percent confidence interval:
##       lower      upper
## -0.2004390  0.2551728
```

Not only are the differences statistically insignificant, Cohen's $d = .03$ indicates a negligible difference. Therefore, there is nothing of interest based on these statistical and practical significance tests.

Mann-Whitney U Test

A popular nonparametric (distribution-free) alternative to Welch's t -test is the **Mann-Whitney U Test**, also referred to as the **Wilcoxon Rank-Sum Test**. Rather than comparing the mean between two groups, like the Student's t -test or Welch's t -test, the Mann-Whitney U test considers the entire distribution by evaluating the extent to which the *ranks* are consistent between groups (i.e., similarity in the proportion of records with each value). When distributions are similar, the medians of the two groups are compared.

The `wilcox.test()` function is used to run this test in R. Let's illustrate by examining whether engagement (an ordinal variable in our dataset) is significantly different between those who have been promoted in the past year and those who have not:

```
# Create dummy-coded promotion variable
employees$promo <- ifelse(employees$last_promo == 1, 1, 0)

# Create numeric engagement vectors for promo groups
no_promo <- unlist(subset(employees, promo == 0, select =
  ~ engagement))
promo <- unlist(subset(employees, promo == 1, select =
  ~ engagement))

# Perform the Mann-Whitney U (aka Wilcoxon rank-sum) test
wilcox.test(no_promo, promo)
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data: no_promo and promo
## W = 196056, p-value = 0.6707
## alternative hypothesis: true location shift is not equal to 0
```

Based on these results, we fail to reject the null hypothesis, which states that there is no difference in engagement between those with and without promotions ($W = 196,056$, $p = .67$). Note the reference to continuity correction in the output. **Continuity correction** is applied when using a continuous distribution to approximate a discrete distribution. The Mann-Whitney U test we performed approximated a continuous distribution for testing differences between our ordinal (discrete) engagement data by applying this continuity correction.

Just as Cohen's d is used to measure the magnitude of difference between a pair of means, **Cliff's delta** can be leveraged to evaluate the size of differences between ordinal variables. Cliff's delta measures how often a value in one distribution is higher than values in another, and this is appropriate in situations in which a nonparametric test of differences is used. This statistic can be produced using the `cliff.delta()` function from the `effsize` package in R.

```
# Calculate Cliff's Delta
effsize::cliff.delta(no_promo, promo)
```

```
##
## Cliff's Delta
##
## delta estimate: -0.0131625 (negligible)
## 95 percent confidence interval:
##       lower      upper
## -0.07329491  0.04706528
```

Some (e.g., Vargha & Delaney, 2000) have endeavored to categorize the Cliff's delta statistic, which ranges from -1 to 1, into effect size buckets. However, such categorizations are far more controversial than thresholds attributed to Cohen's d . Nevertheless, the near-zero delta estimate of -.01 indicates a negligible difference.

Paired Samples t-Test

A **Paired Samples t-Test** is used to compare means between pairs of measurements. This test is known by many other names, such as a **dependent samples t-test**, **paired-difference t-test**, **matched pairs t-test**, and **repeated-samples t-test**.

The assumption of normality in the context of a paired samples t -test relates to normally distributed paired differences. This is important, as the p -value for the test statistic will not be valid if this assumption is violated.

To illustrate, let's design an experiment. Let's assume morale has declined for employees who travel frequently, and several actions have been proposed by a task force to help address this. The task force has decided to pilot a new flexible work benefit over a six-month period to determine if it has a meaningful effect on morale. This new benefit is piloted to a random sample of frequent travelers, and our task is to test whether the outcomes warrant a broader rollout to frequent travelers.

Our DV (happiness) will be measured using a composite index derived from individual engagement, environment satisfaction, job satisfaction, and relationship satisfaction scores. Our objective is to determine if there is a significant improvement in this happiness index for the treatment group (those who are part of the flexible work pilot) relative to the pre/post difference for the control group (those not selected for the flexible work pilot).

While we could simply look at the pre/post differences for the treatment group, we understand from Chapter @ref(research) that this would be a weak design that may lead to inaccurate conclusions. There could be alternative explanations for any observed increases in happiness that are unrelated to the intervention itself. For example, between time 1 and time 2, travel frequency may have decreased for everyone, which may contribute to overall happiness. By comparing pre/post differences between the treatment and control groups, we gain more confidence in isolating the effect of the flexible work benefit on happiness since alternative explanations should be reflected in any pre/post changes observed for the control group.

Let's prepare the data for this experiment. Since `employees` is a cross-sectional dataset (single point-in-time), we will generate simulated data for repeated measures (i.e., post-intervention scores).

```
# Set seed for reproducible results
set.seed(1234)
```

```

# Derive happiness index from survey variables
employees$happiness_ind <- (employees$engagement +
  ↵ employees$env_sat + employees$job_sat + employees$rel_sat) /
  ↵ 4

# Sample size of frequent travelers
n = nrow(subset(employees, business_travel ==
  ↵ 'Travel_Frequently', select = employee_id))

# Randomly assign half of frequent travelers to treatment and
control groups
treat_ids <- sample(unlist(subset(employees, business_travel ==
  ↵ 'Travel_Frequently', select = employee_id)), floor(n * .5))
ctrl_ids <- unlist(subset(employees, business_travel ==
  ↵ 'Travel_Frequently' & !employee_id %in% treat_ids, select =
  ↵ employee_id))

# Initialize dfs for pre/post metrics
treat_metrics = data.frame(pre_ind = length(treat_ids),
                           rand_num = rnorm(length(treat_ids),
  ↵ mean = 15, sd = 5) * .001,
                           post_ind = length(treat_ids),
                           diff = length(treat_ids))

ctrl_metrics = data.frame(pre_ind = length(ctrl_ids),
                           rand_num = rnorm(length(ctrl_ids), mean
  ↵ = 0, sd = 1) * .001,
                           post_ind = length(ctrl_ids),
                           diff = length(ctrl_ids))

# Store happiness indices for treatment and control groups
treat_metrics$pre_ind <- unlist(subset(employees, employee_id
  ↵ %in% treat_ids, select = happiness_ind))
ctrl_metrics$pre_ind <- unlist(subset(employees, employee_id %in%
  ↵ ctrl_ids, select = happiness_ind))

# Create vectors with artificially inflated post-intervention
happiness indices
treat_metrics$post_ind <- treat_metrics$pre_ind +
  ↵ treat_metrics$rand_num
ctrl_metrics$post_ind <- ctrl_metrics$pre_ind +
  ↵ ctrl_metrics$rand_num

```

It's important to remember that a paired samples *t*-test requires that each of

the paired measurements be obtained from the same subject. Therefore, if an employee terms between time 1 and time 2, or does not provide the survey responses needed to calculate the happiness index at both time 1 and time 2, the employee should be removed from the data since paired measurements will not be available.

The variance is not assumed to be equal for a paired test; therefore, the homogeneity of variance assumption is not applicable in this context.

Next, we will evaluate whether paired differences are normally distributed using the Shapiro Wilk test. While individual survey items are measured on an ordinal scale, our derived happiness index is the average of multiple ordinal items and can be considered an *approximately* continuous variable. There are $2^p - p - 1$ combinations of scores, where p is the number of variables. For our happiness index, there are $2^4 - 4 - 1 = 11$ combinations.

```
# Load library
library(ggpubr)

# Calculate pre/post differences
treat_metrics$diff <- treat_metrics$post_ind -
  → treat_metrics$pre_ind
ctrl_metrics$diff <- ctrl_metrics$post_ind - ctrl_metrics$pre_ind

# Histogram for distribution of pre/post treatment group
  → differences
p_treat <- ggplot2::ggplot() +
  ggplot2::aes(treat_metrics$diff) +
  ggplot2::labs(title = "Treatment Group", x =
  → "Happiness Index Differences", y = "Frequency") +
  ggplot2::geom_histogram(fill = "#414141") +
  ggplot2::theme_bw() +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))

# Histogram for distribution of pre/post control group
  → differences
p_ctrl <- ggplot2::ggplot() +
  ggplot2::aes(ctrl_metrics$diff) +
  ggplot2::labs(title = "Control Group", x = "Happiness
  → Index Differences", y = "Frequency") +
  ggplot2::geom_histogram(fill = "#414141") +
  ggplot2::theme_bw() +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))

# Display histograms side-by-side
ggpubr::ggarrange(p_treat, p_ctrl, ncol = 2, nrow = 1)
```

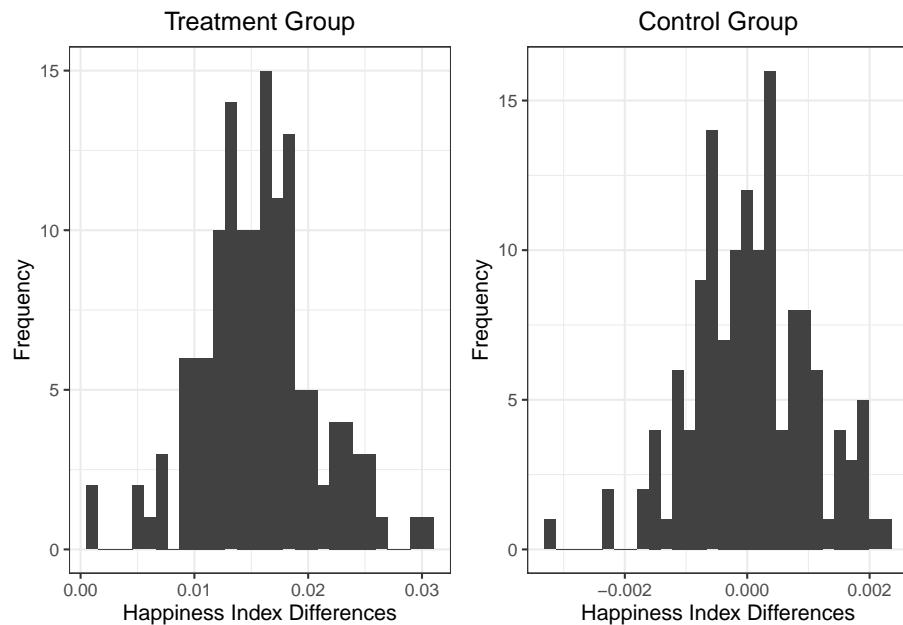


Figure 45: Pre/post differences for treatment and control groups

Based on a visual inspection, the distributions of differences appear to be roughly normal. This should not be surprising given random values were sampled from normal distributions to derive artificial post-intervention happiness indices.

Let's test for normality by performing the Shapiro-Wilk test on vectors of differences:

```
# Compute Shapiro-Wilk test of normality
shapiro.test(treat_metrics$diff)
```

```
##
##  Shapiro-Wilk normality test
##
## data: treat_metrics$diff
## W = 0.98936, p-value = 0.3738
```

```
shapiro.test(ctrl_metrics$diff)
```

```
##
##  Shapiro-Wilk normality test
```

```
##  
## data: ctrl_metrics$diff  
## W = 0.99096, p-value = 0.5134
```

Since $p \geq .05$ for both tests, the assumption of normally distributed differences is met. Given the data generative process implemented for this example, differences would become increasingly normal as the sample size increases. We now have the greenlight to perform the paired samples t -test.

We can run a paired samples t -test in R by passing `paired = TRUE` as an argument to the same `t.test()` function used for the independent samples t -test. Since we are investigating whether the average post-intervention happiness index is significantly *greater* than the average pre-intervention happiness index, we also need the `alternative = "greater"` argument since the default two-tailed test only evaluates whether the average indices are significantly different (regardless of whether the pre- or post-intervention index is larger).

```
# Perform one-tailed paired samples t-test for treatment group  
t.test(treat_metrics$post_ind, treat_metrics$pre_ind, paired =  
  TRUE, alternative = "greater")
```

```
##  
## Paired t-test  
##  
## data: treat_metrics$post_ind and treat_metrics$pre_ind  
## t = 35.906, df = 137, p-value < 2.2e-16  
## alternative hypothesis: true mean difference is greater than 0  
## 95 percent confidence interval:  
##  0.01490482      Inf  
## sample estimates:  
## mean difference  
##               0.01562551
```

These results indicate that the post-intervention happiness index is significantly larger than the pre-intervention happiness index. This is encouraging with respect to the potential efficacy of the flexible work pilot, but the question about whether the control group experienced a commensurate improvement over the observation period remains unanswered.

Let's run the same paired samples t -test using the control group indices:

```
# Perform one-tailed paired samples t-test for control group  
t.test(ctrl_metrics$post_ind, ctrl_metrics$pre_ind, paired =  
  TRUE, alternative = "greater")
```

```

## Paired t-test
##
## data: ctrl_metrics$post_ind and ctrl_metrics$pre_ind
## t = 0.59995, df = 138, p-value = 0.2748
## alternative hypothesis: true mean difference is greater than 0
## 95 percent confidence interval:
## -8.719262e-05           Inf
## sample estimates:
## mean difference
## 4.953658e-05

```

Since $p \geq .05$, we can conclude that there was not a significant increase in happiness indices for the control group, which provides additional – but not conclusive – support for the effectiveness of the flexible work benefit.

Chapter @ref(lm) will introduce linear regression, which is a powerful modeling tool for people analytics that helps control for multiple alternative explanations of associations with the DV in order to isolate the unique effects of each IV.

Difference-in-differences (DiD) estimation is an alternative quasi-experimental approach that originated from econometrics for evaluating the effects of interventions like these, but it is beyond the scope of this book. Angrist and Pischke (2009) is an excellent resource for learning about these methods.

We can evaluate the magnitude of mean differences for these paired samples by passing the `paired = TRUE` argument to the same `cohen.d()` function used for independent samples:

```

# Perform Cohen's d
effsize::cohen.d(treat_metrics$post_ind, treat_metrics$pre_ind,
                  paired = TRUE)

##
## Cohen's d
##
## d estimate: 0.03132117 (negligible)
## 95 percent confidence interval:
##      lower      upper
## 0.02960345 0.03303890

```

Though pre/post indices are statistically significant for the treatment group, the size of the difference is negligible ($d = .03$).

Wilcoxon Signed-Rank Test

The **Wilcoxon Signed-Rank Test** is the nonparametric alternative to the paired samples *t*-test. This distribution-free test does not require normally distributed differences.

The matched Wilcoxon Signed-Rank test is performed in R using the same `wilcox.test()` function used to perform the unmatched Wilcoxon Rank-Sum test. Though we can use a paired samples *t*-test to test differences for our flexible work benefit study since the assumption of normally distributed differences is met, let's run a Wilcoxon Signed-Rank test for demonstrative purposes:

```
# Perform Wilcoxon Signed-Rank test
wilcox.test(treat_metrics$post_ind, treat_metrics$pre_ind, paired
↪ = TRUE)

## 
## Wilcoxon signed rank test with continuity correction
##
## data: treat_metrics$post_ind and treat_metrics$pre_ind
## V = 9591, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0

# Perform Wilcoxon Signed-Rank test
wilcox.test(ctrl_metrics$post_ind, ctrl_metrics$pre_ind, paired =
↪ TRUE)

## 
## Wilcoxon signed rank test with continuity correction
##
## data: ctrl_metrics$post_ind and ctrl_metrics$pre_ind
## V = 5166, p-value = 0.5275
## alternative hypothesis: true location shift is not equal to 0
```

Consistent with results from the paired samples *t*-tests, significantly higher post-intervention happiness indices were observed for the treatment group but not for the control group.

We can evaluate the magnitude of differences for these paired samples by passing the `paired = TRUE` argument to the same `cliff.delta()` function used for independent samples:

```
# Run Cliff's Delta
effsize::cliff.delta(treat_metrics$post_ind,
↪ treat_metrics$pre_ind, paired = TRUE)
```

```
##  
## Cliff's Delta  
##  
## delta estimate: 0.1544844 (small)  
## 95 percent confidence interval:  
##      lower      upper  
## 0.01652557 0.28667094
```

With Cliff's delta, we observe a small difference between pre/post indices for the treatment group (delta estimate = .15).

Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) is used to determine whether the means of scale-level DVs are equal across nominal-level variables with three or more independent categories.

It is important to understand that H_0 in ANOVA not only requires all group means to be equal but their complex contrasts as well. For example, if we have four groups named A, B, C, and D, H_0 requires that $\mu_A = \mu_B = \mu_C = \mu_D$ is true as well as the various complex contrasts such as $\mu_{A,B} = \mu_{C,D}$ and $\mu_A = \mu_{B,C,D}$ and $\mu_D = \mu_{B,C}$. Therefore, a difference between one or more of these contrasts results in a decision to reject H_0 in ANOVA. As a result, we may find a significant F -statistic but no significant differences between pairwise means.

It is also possible to find a significant pairwise mean difference but a non-significant result from ANOVA. As you may recall from Chapter @ref(inf-stats), multiple comparisons reduce the power of statistical tests. Since multiple tests of mean differences are performed with ANOVA, the family-wise error rate is used to adjust for the increased probability of a Type I error across the set of analyses. Since the power of a single pairwise test is greater relative to the power of familywise comparisons, we may find a significant result for the former but not the latter.

ANOVA requires IVs to be categorical (nominal or ordinal) and the DV to be continuous (interval or ratio). A **one-way ANOVA** is used to determine how one categorical IV influences a continuous DV. A **two-way ANOVA** is used to determine how two categorical IVs influence a continuous DV. A **three-way ANOVA** is used to evaluate how three categorical IVs influence a continuous DV. An ANOVA that uses two or more categorical IVs is often referred to as a **factorial ANOVA**. As discussed in Chapter @ref(getting-started), it is important to remain grounded in specific hypotheses, as a significant ANOVA may not actually test what is being hypothesized.

ANOVA is not a test, per se, but a F -test underpins it. The mathematical procedure behind the F -test is relatively straightforward:

1. Compute the **within-group variance**, which is also known as residual variance. Simply put, this tells us how different each member of the group is from the average.
2. Compute the **between-group variance**. This represents how different the group means are from one another.
3. Produce the *F*-statistic, which is the ratio of *within-group variance* to *between-group variance*.

More formally, the *F*-statistic is defined by:

$$F = \frac{MS_{between}}{MS_{within}},$$

where:

$$MS_{between} = \frac{SS_{between}}{df_{between}},$$

$$MS_{within} = \frac{SS_{within}}{df_{within}},$$

$$SS_{between} = \sum_{j=1}^p n_j (\bar{x}_j - \bar{x})^2,$$

$$SS_{within} = \sum_{j=1}^p \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$$

One-Way ANOVA

To illustrate how to perform a one-way ANOVA, we will test the hypothesis that mean annual compensation is equal across job satisfaction levels.

Each observation in `employees` represents a unique employee, and a given employee can only have one job satisfaction score and one annual compensation value. The assumption of independence is met since each record exists independent of one another and each job satisfaction group is comprised of different employees.

Levene's test (Levene, 1960) can be used to test the homogeneity of variance assumption – even with non-normal distributions. This can be performed in R using the `leveneTest()` function from the `car` package:

```
# Perform Levene's test for homogeneity of variance
car:::leveneTest(annual_comp ~ as.factor(job_sat), data =
  employees)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value Pr(>F)
## group     3  0.3293 0.8042
##           1466
```

The test statistic associated with Levene's test relates to the null hypothesis that there are no significant differences in variances across the job satisfaction levels. Since $p \geq .05$, we fail to reject this null hypothesis and can assume equal variances.

Next, let's test the assumption of normality. It is important to note that the assumption of normality *does not* apply to the distribution of the DV but to the distribution of residuals for each group of the IV. Residuals in the context of ANOVA represent the difference between the actual values of the continuous DV relative to its mean value for each level of the categorical IV (e.g., $y - \bar{y}_A$, $y - \bar{y}_B$, $y - \bar{y}_C$). In ANOVA, we expect the residuals to be normally distributed around a mean of 0 (the balance point) when the data are normally distributed within each IV category; the more skewed the data, the larger the average distance of each DV value from the mean.

```
# Create function to visualize distribution
dist.viz <- function(data, x) {

  viz <- ggplot2::ggplot() +
    ggplot2::aes(data) +
    ggplot2::labs(title = paste("Job Sat = ", x), x = "Annual
← Compensation", y = "Frequency") +
    ggplot2::geom_histogram(fill = "#414141") +
    ggplot2::theme_bw() +
    ggplot2::theme(plot.title = element_text(hjust = 0.5))

  return(viz)
}

# Produce annual compensation vectors for each job satisfaction
# level
# Unlist() is needed to convert the default object from subset()
# into a numeric vector
group_1 <- unlist(subset(employees, job_sat == 1, select =
  annual_comp))
group_2 <- unlist(subset(employees, job_sat == 2, select =
  annual_comp))
group_3 <- unlist(subset(employees, job_sat == 3, select =
  annual_comp))
group_4 <- unlist(subset(employees, job_sat == 4, select =
  annual_comp))
```

```

# Call UDF to build annual comp histogram for each job
→ satisfaction level
viz_1 <- dist.viz(data = group_1, x = 1)
viz_2 <- dist.viz(data = group_2, x = 2)
viz_3 <- dist.viz(data = group_3, x = 3)
viz_4 <- dist.viz(data = group_4, x = 4)

# Display distribution visualizations
ggpubr::ggarrange(viz_1, viz_2, viz_3, viz_4,
                  ncol = 2, nrow = 2)

```

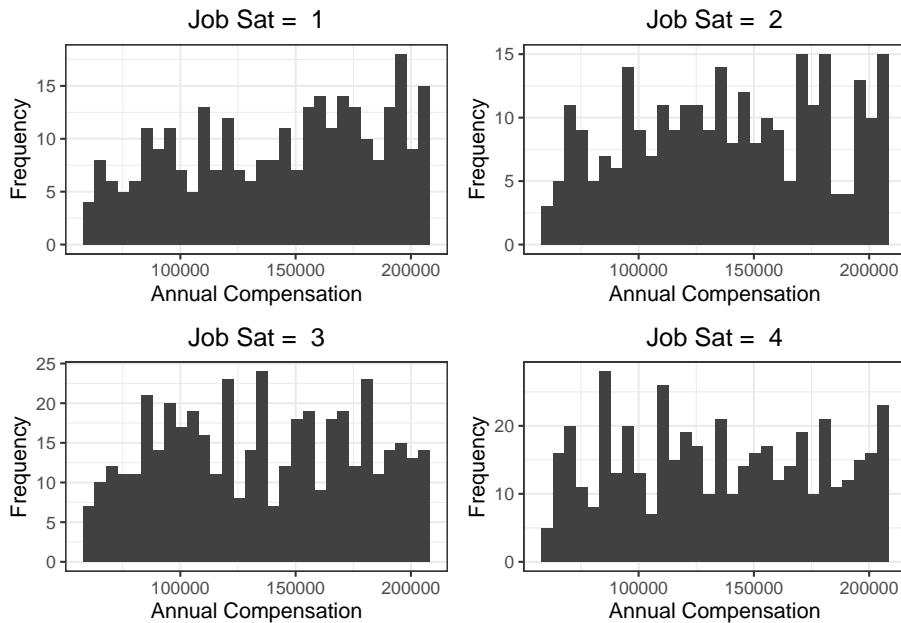


Figure 46: Annual compensation distribution by job satisfaction level

As we can see, annual compensation data are not normally distributed within job satisfaction groups. Therefore, we would not expect the distribution of residuals to be normally distributed within these groups either.

To test whether the assumption of normality is met, we will first produce and review a **quantile-quantile (Q-Q) plot**. A Q-Q plot compares two probability distributions by plotting their quantiles (data partitioned into equal-sized groups) against each other. After partitioning annual compensation into groups differentiated by job satisfaction level, we can use the `ggqqplot()` function from the `ggpubr` library to build a Q-Q plot and evaluate the distribution of residuals.

```
# Generate residuals for each group
residuals <- c(group_1 - mean(group_1), group_2 - mean(group_2),
  ~ group_3 - mean(group_3), group_4 - mean(group_4))

# Create a Q-Q plot of residuals
ggpubr::ggqqplot(residuals)
```

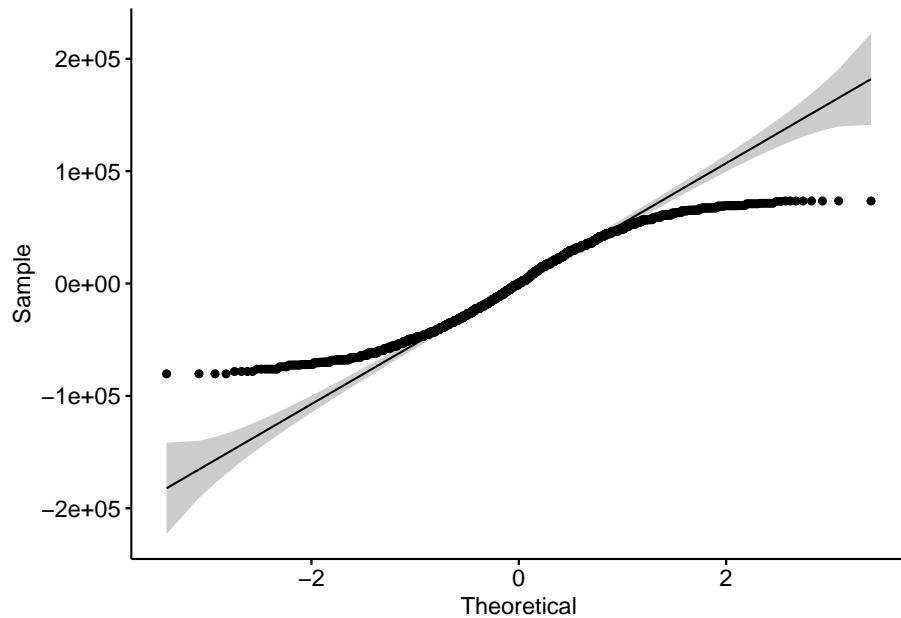


Figure 47: Q-Q plot of annual compensation residuals

To satisfy the assumption of normality, residuals must lie along the linear line. Based on the Q-Q plot in Figure @ref(fig:qq-plot), there is a clear departure from normality at both ends of the theoretical range.

Let's test for normality using the Shapiro-Wilk test:

```
# Compute Shapiro-Wilk test of normality
shapiro.test(residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.95874, p-value < 2.2e-16
```

Since $p < .05$, we reject the null hypothesis of normally distributed data, which indicates that the assumption of normality is violated. This should not be surprising based on the deviation from normality we observed in Figure @ref(fig:qq-plot).

Because the assumption of normality is violated, we have two options. First, we can attempt to transform the data so that the residuals using the transformed values are normally distributed. If the data are resistant to transformation, we can leverage a nonparametric alternative to ANOVA.

Let's first try several common data transformations and then examine the resulting Q-Q plots:

```
# Build a linear model using the natural logarithm of annual comp
ln.model <- lm(log(annual_comp) ~ job_sat, data = employees)

# Build a linear model using the log base 10 of annual comp
log10.model <- lm(log10(annual_comp) ~ job_sat, data = employees)

# Build a linear model using the square root of annual comp
sqrt.model <- lm(sqrt(annual_comp) ~ job_sat, data = employees)

# Store Q-Q plots to viz objects
ln.viz <- ggpibr::ggqqplot(residuals(ln.model)) +
  ggtitle("Natural Log")
log10.viz <- ggpibr::ggqqplot(residuals(log10.model)) +
  ggtitle("Log Base 10")
sqrt.viz <- ggpibr::ggqqplot(residuals(sqrt.model)) +
  ggtitle("Square Root")

# Display Q-Q plots of residuals
ggpubr::ggarrange(ln.viz, log10.viz, sqrt.viz,
  ncol = 3, nrow = 1)
```

Even with these transformations, there is still a clear S-shaped curve about the residuals. Though we cannot proceed with ANOVA due to violated assumptions, let's demonstrate the implementation steps for ANOVA. Performing ANOVA involves pairing the `aov()` function with the `summary()` function to display model output:

```
# One-way ANOVA investigating mean differences in annual comp by
# job satisfaction
one.way <- aov(annual_comp ~ job_sat, data = employees)

summary(one.way)
```

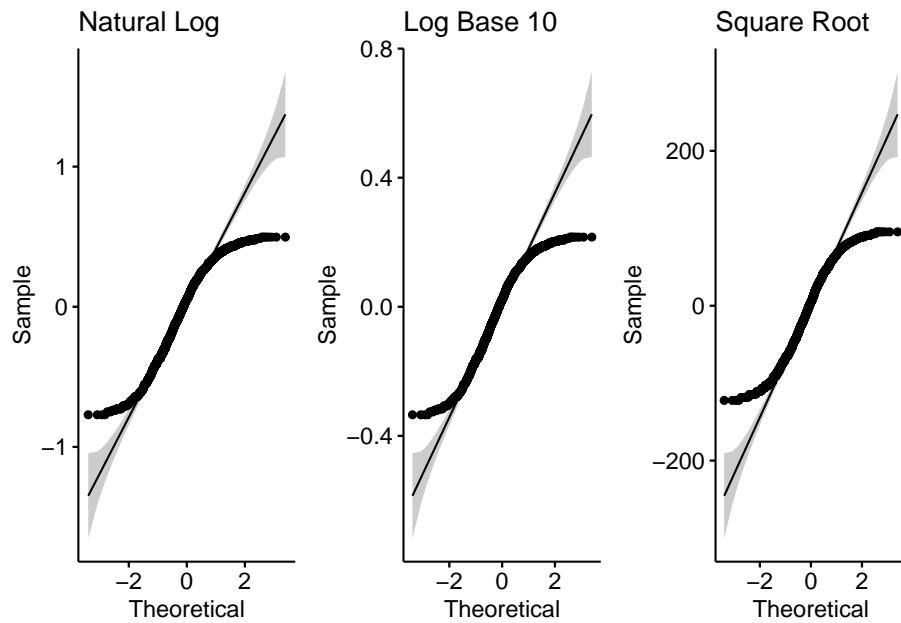


Figure 48: Q-Q plots of transformed annual compensation residuals

```
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## job_sat      1 1.337e+10 1.337e+10   7.508 0.00622 **
## Residuals  1468 2.613e+12 1.780e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The **Kruskal Wallis H Test** is the nonparametric alternative to a one-way ANOVA (Daniel, 1990) and an appropriate alternative for investigating median differences in annual comp by job satisfaction in our data. This test can be performed using the `kruskal.test()` function in R:

```
# Nonparametric Kruskal one-way ANOVA
kruskal.test(annual_comp ~ job_sat, data = employees)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: annual_comp by job_sat
## Kruskal-Wallis chi-squared = 8.3242, df = 3, p-value = 0.03977
```

Since $p < .05$, we can conclude that there are significant differences in median compensation across the groups. However, this test does not indicate which

groups are different. We can utilize the `pairwise.wilcox.test()` function to compute pairwise Wilcoxon rank-sum tests to identify where differences exist:

```
pairwise.wilcox.test(employees$annual_comp, employees$job_sat,
  → p.adjust.method = "BH")

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: employees$annual_comp and employees$job_sat
##
##   1     2     3
## 2 0.298 -    -
## 3 0.041 0.298 -
## 4 0.041 0.298 0.879
##
## P value adjustment method: BH
```

Based on the results, there are significant pairwise differences in median annual compensation for job satisfaction levels 3 and 4 relative to level 1.

Factorial ANOVA

Factorial ANOVA is any ANOVA which uses two or more categorical IVs, such as a two-way or three-way ANOVA. The following output reflects the cross-tabulation of average annual compensation for each combination of two factors – job satisfaction and stock option level.

```
# Calculate mean for each IV pair
combos <- aggregate(annual_comp ~ job_sat + stock_opt_lvl,
  → employees, mean)
combos
```

	job_sat	stock_opt_lvl	annual_comp
## 1	1	0	141254.0
## 2	2	0	138753.3
## 3	3	0	132159.2
## 4	4	0	132227.0
## 5	1	1	141763.9
## 6	2	1	135494.5
## 7	3	1	135235.7
## 8	4	1	135569.0
## 9	1	2	146240.0
## 10	2	2	146432.0
## 11	3	2	145080.0

```

## 12      4      2      143019.3
## 13      1      3      154844.4
## 14      2      3      144254.1
## 15      3      3      135672.7
## 16      4      3      127102.9

```

As we have already discussed, a difference between one or more of these contrasts may result in a decision to reject H_0 in ANOVA. We may also find a significant pairwise difference but a non-significant result from ANOVA since the family-wise error rate adjustment is applied in the context of multiple comparisons which reduces statistical power.

Factorial ANOVA can be performed by adding variables with + within the same `aov()` function used for one-way ANOVA:

```

# Factorial ANOVA investigating mean differences in annual comp
# by job satisfaction and stock option level
factorial <- aov(annual_comp ~ job_sat + stock_opt_lvl, data =
# employees)

summary(factorial)

```

```

##          Df Sum Sq Mean Sq F value Pr(>F)
## job_sat     1 1.337e+10 1.337e+10    7.523 0.00617 **
## stock_opt_lvl 1 6.840e+09 6.840e+09    3.850 0.04995 *
## Residuals 1467 2.606e+12 1.777e+09
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

While mean annual compensation is significantly different by both job satisfaction and stock option level factors, this output alone is not too helpful in understanding the nature of the differences. These statistical significance markers indicate that there are meaningful differences that warrant a deeper understanding. Relationships of job satisfaction and stock option level with annual compensation are illustrated more effectively in Figure @ref(fig:two-way-factorial).

As we can see, there is a strong negative relationship between job satisfaction and average annual compensation among employees with the highest stock option level (3). The relationship between job satisfaction and average annual compensation appears to be negative for employees with other stock option levels as well, albeit much weaker.

These relationships may initially seem counterintuitive, as one might expect higher levels of job satisfaction to contribute to higher performance and consequently, higher compensation. There may be other variables that happen to be correlated with job satisfaction and/or stock option level that are the actual

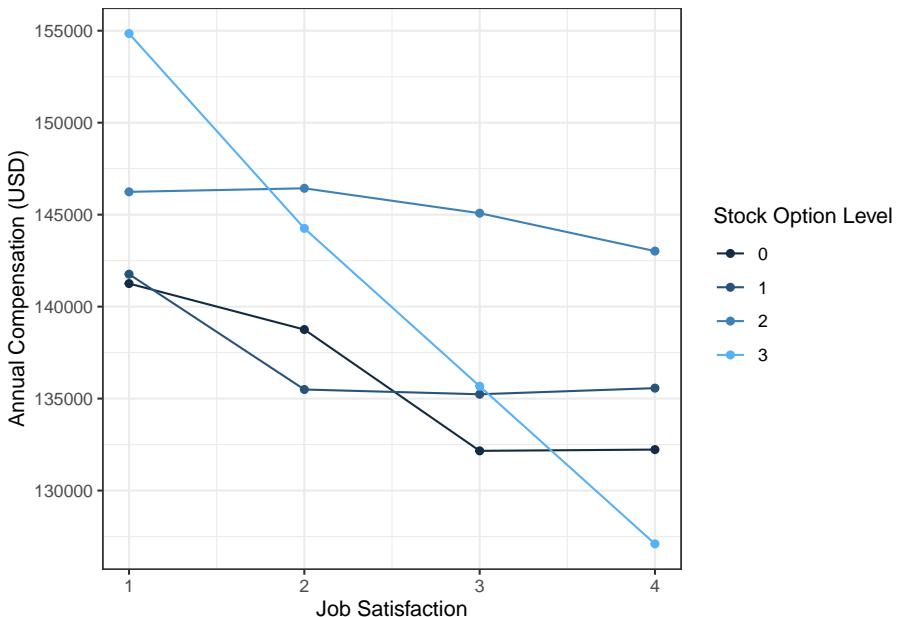


Figure 49: Relationships of job satisfaction and stock option level with annual compensation

determinants of annual compensation. For example, there may be a relationship between jobs that each feature a stock option level of 3 but for which there are markedly different average job satisfaction scores and annual compensation among workers in these jobs. Without accounting for additional variables that may explain why employees vary in the amount of annual compensation they earn, the limited set of relationships shown in Figure @ref(fig:two-way-factorial) may lead to a flawed understanding and inaccurate conclusions.

Three-way factorials (and beyond) become difficult to visualize and understand in the way one-way ANOVA and two-way factorials have been explained in this chapter. In Chapter @ref(lm), we will discuss how to create linear combinations of many IVs and parse the output to understand how they independently and jointly help explain variation in the DV.

Review Questions

1. What are the main differences between a Chi-square test and Fisher's exact test?
2. Why is it problematic to test for significant differences using the χ^2 statistic with extremely small samples (e.g., $n < 5$)?

3. What are the general assumptions of parametric tests?
4. What is a benefit of Welch's t -test over the Student's t -test?
5. How does a paired-samples t -test differ from an independent samples t -test?
6. In what ways does the Wilcoxon signed-rank test differ from the paired samples t -test?
7. How can the magnitude of differences (i.e., practical significance) be quantified when working with data measured on a continuous scale?
8. How can the magnitude of differences (i.e., practical significance) be quantified when working with data measured on an ordinal scale?
9. What null hypothesis does ANOVA test?
10. What are some ways to better understand the nature of statistical differences indicated in the output of ANOVA?

Linear Regression

Regression is perhaps the most important statistical learning technique for people analytics. If you have taken a statistics course at the undergraduate or graduate levels, you have surely already encountered it. Let's first develop an intuitive understanding of the mechanics of regression.

Imagine we are sitting at a large public park in New York City on a nice fall afternoon. If asked to estimate the annual compensation of the next person to walk by, how would you estimate this in the absence of any additional information? Most would likely estimate the *average* annual compensation of everyone capable of walking by. Since this would include both residents and visitors, this would be a very large population of people! The obvious limitation with this approach is that among the large group of people capable of walking by, there is likely a significant range of annual compensation values. Many walking by may be children, unemployed, or retirees who earn no annual compensation, while others may be highly compensated senior executives at the pinnacle of their careers. Since the range of annual compensation could be zero to millions of dollars, estimating the average of such a large population is likely going to be highly inaccurate without more information.

Let's consider that we are sitting outside on a weekday afternoon. Should this influence our annual compensation estimate? It is likely that we can eliminate a large segment of those likely to walk by, as we would expect most children to be in school on a typical fall weekday afternoon. It's also less likely that those who are employed and not on vacation will walk by on a fall weekday afternoon. Therefore, factoring in that it is a weekday should limit the size of the population which in turn may reduce the range of annual compensation values for our population of passerbys.

Let's now consider that the park is open only to invited guests for a symposium on people analytics. Though it may be difficult to believe, a relatively small subset of the population is likely interested in attending such a symposium, so this information will likely be quite helpful in reducing the size of the population who could walk by. This should further reduce the range of annual compensation since we probably have a good idea of the profile of those most likely to attend. This probably also lessens (or altogether eliminates) the importance of

the weekday factor in explaining why people vary in the amount of compensation they earn each year. That an important variable may become unimportant in the presence of another variable is a key feature of regression.

In addition, let's consider that only those who reside in NYC and Boise, Idaho were invited, and that the next person to walk by resides in Boise. Most companies apply a significant cost of living multiplier to the compensation for those in an expensive region such as NYC, resulting in a significant difference in compensation relative to those residing in a much less expensive city like Boise – all else being equal. Therefore, if we can partition attendees into two groups based on their geography, this should significantly limit the range of annual compensation *within each* – likely making the average compensation in each group a more nuanced and reasonable estimate.

What if we also learn the specific zip code in which the next passerby from Boise resides? The important information is likely captured at the larger city level (NYC vs. Boise), as the compensation for the specific zip codes within each city are unlikely to vary to a meaningful degree. Assuming this is true, it probably would not make sense to consider both the city name and zip code since they are effectively redundant pieces of information with regard to explaining the variance in annual compensation.

What if we learn that the next person to walk by will be wearing a blue shirt? Does this influence your estimate? Unless there is research to suggest shirt color and earnings are related, this information will likely not contribute any significant information to our understanding of why people vary in the amount of compensation they earn and should, therefore, not be considered.

You can probably think of many relevant variables that would help further narrow the range of annual compensation. These may include job, level, years of experience, education, among other factors. The main thing to understand is that for each group of observations *with the same characteristics* – such as senior analysts with a graduate degree who reside in NYC – there is a distribution of annual compensation. This distribution reflects *unexplained* variance. That is, we do not have information to explain why the compensation for each and every person is not the same and in social science contexts, it simply is not practical to explain 100 percent of the variance in outcomes. For example, two people may be similar on dozens of factors (experience, education, skills) but one was a more effective negotiator when offered the same role and commanded a higher salary. It's likely we do not have data on salary negotiation ability so this information would leave us with unexplained variance in compensation. The goal is to identify the variables that provide the most information in helping us tighten the distribution so that estimating the expected (average) value will generally be an accurate estimate for those in the larger population with the same characteristics.

While we can generally improve our estimates with more relevant information (not shirt color or residential zip code in this case), it is important to understand

that samples which are too small ($n < 30$) lend to anomalies; modeling noise in sparse data can result in models that are unlikely to generalize beyond the sample data. For example, if the only people from Boise to attend the people analytics symposium happen to be two ultra wealthy tech entrepreneurs who earn millions each year, it would not be appropriate to use this as the basis for our estimates of all future attendees from Boise.

This is the essence of linear regression modeling: find a limited number of variables which independently and/or jointly provide significant information that helps explain (by reducing) variance around the average value. As illustrated in this example, adding additional variables (information) can impact the importance of other variables or may offer no incremental information at all. In this chapter, we will cover how to identify which variables are important and how to quantify the effect they have on an outcome.

Assumptions & Diagnostics

As we learned in the context of power analysis in Chapter @ref(inf-stats), the sample size needs to be large enough to model and detect significant associations of one or more predictors with the response variable. In practice, people analytics practitioners are often constrained by the data at hand, which is to say that one generally has little control over the amount of data that can be collected. For example, despite the most earnest participation campaigns, only a subset of invited employees are likely to complete a survey, so collecting additional data to achieve a larger sample is likely not a viable option. It is important to establish a minimum – and realistic – n -count threshold during the planning stage of a project based on the research objectives and variables that need to be factored into the analysis.

Consistent with the assumptions of parametric tests covered in Chapter @ref(aod), there are several assumptions that need to be validated to determine if a linear model is appropriate for understanding relationships in the data. These assumptions largely relate to the residuals ($\hat{y} - y$):

1. **Independence:** Residuals are independent of each other; consecutive residuals in time series data are unrelated
2. **Homoscedasticity:** Variance of residuals is constant across values of X
3. **Normality:** Residuals must be normally distributed (with mean of 0) across values of X
4. **Linearity:** Relationship between X and Y is linear

Beyond these core assumptions for linear models, additional diagnostics are important to incorporate into the early data screening stage:

1. **High-Leverage Observations:** Influential data that significantly changes the model fit
2. **Collinearity:** Independent variables that are highly correlated (these should be *independent*)

Sample Size

While a general rule-of-thumb for regression analysis is a minimum of a 20:1 ratio of observations to IV, Chapter @ref(inf-stats) covered a more rigorous approach for calculating the sample size needed to observe significant effects.

For linear regression, power analysis involves a comparison of model fit between a model with a full set of predictors relative to one with only a subset of the full model's predictors. The function from the `pwr` library to call is `pwr.f2.test(u = , v = , f2 = , sig.level = , power =)`, where `u` and `v` are the numerator and denominator degrees of freedom, respectively, and `f2` is defined as:

$$f^2 = \frac{R_{AB}^2 - R_A^2}{1 - R_{AB}^2},$$

where R_{AB}^2 represents the variance accounted for by a full model with all predictors, and R_A^2 represents the variance accounted for by a model containing only a subset of the full model's predictors. Power analysis can be leveraged in determining the sample size needed for detecting the incremental main effects for a set of predictors beyond the variance accounted for by a set of controls.

Simple Linear Regression

Simple linear regression is a simple technique for estimating the value of a quantitative DV, denoted as Y , on the basis of a single IV, denoted as X . It is assumed that there is an approximately linear relationship between X and Y . Often, this relationship is expressed as *regressing* Y onto X and is defined mathematically as:

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

where β_0 is the expected value of Y when $X = 0$ (the *intercept*), and β_1 represents the average change in Y for a one-unit increase in X (the *slope*). β_0 and β_1 are unknown *parameters* or *coefficients*. The error term, ϵ , acknowledges that there is variation in Y not accounted for by this simple linear model – *unexplained* variance. In other words, it is highly unlikely that there is a perfectly linear relationship between X and Y , as additional variables not included in the model are likely to also influence Y .

Once we estimate the unknown model coefficients, β_0 and β_1 , we can estimate Y for a particular value of X by calculating:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i,$$

where \hat{y} represents an estimate of Y for the i -th value of $X = x$. The hat () symbol is used to denote an estimated value of an unknown coefficient, parameter, or outcome.

The earliest form of linear regression is the *least squares* method, which was developed at the beginning of the nineteenth century and applied to astronomy problems (James, Witten, Hastie, & Tibshirani, 2013). While there are several approaches to fitting a linear regression model, **ordinary least squares (OLS)** is the most common. OLS selects coefficients for $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize the residual sum of squares (RSS) defined by:

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

For each value of X , OLS fits a model for which the squared difference between the predicted ($\hat{\beta}_0 + \hat{\beta}_1 x_i$) and actual (y_i) values are as small as possible. Figure @ref(fig:lm-residuals) illustrates the result of minimizing RSS using OLS:

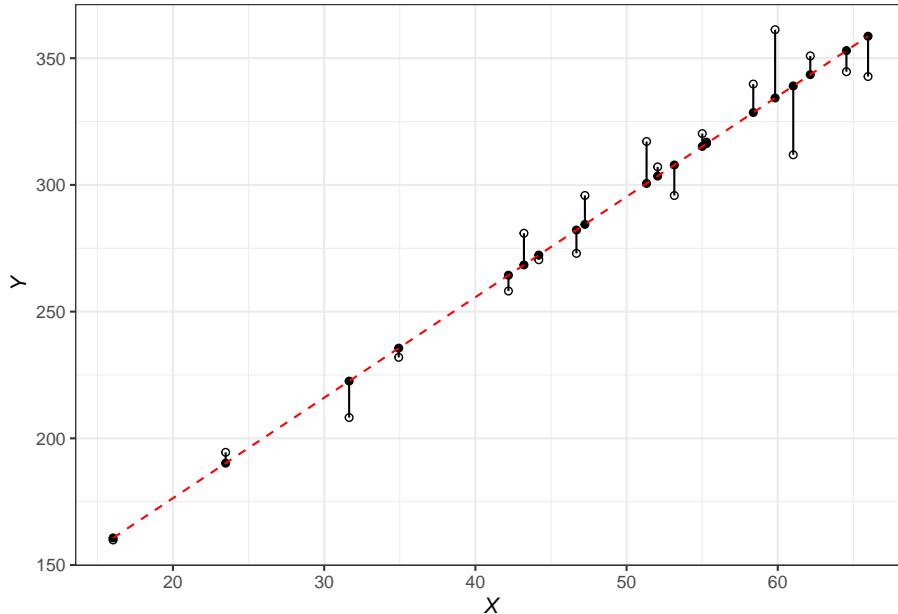


Figure 50: Minimizing RSS with Ordinary Least Squares (OLS) fit

The minimizers for the least squares coefficient estimates are defined by:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where \bar{x} and \bar{y} are sample means.

It is important to understand the role sample size plays in achieving accurate estimates of Y . Figure @ref(fig:lm-fit-compare) illustrates the impact of fitting a model to too few observations. With $n = 2$, it would be easy to fit a perfect model to the data; that is, one representing a line that connects the two data points. However, it is highly unlikely that these data points represent the best model for a larger sample, as there would likely be some distribution of Y for each value of X .

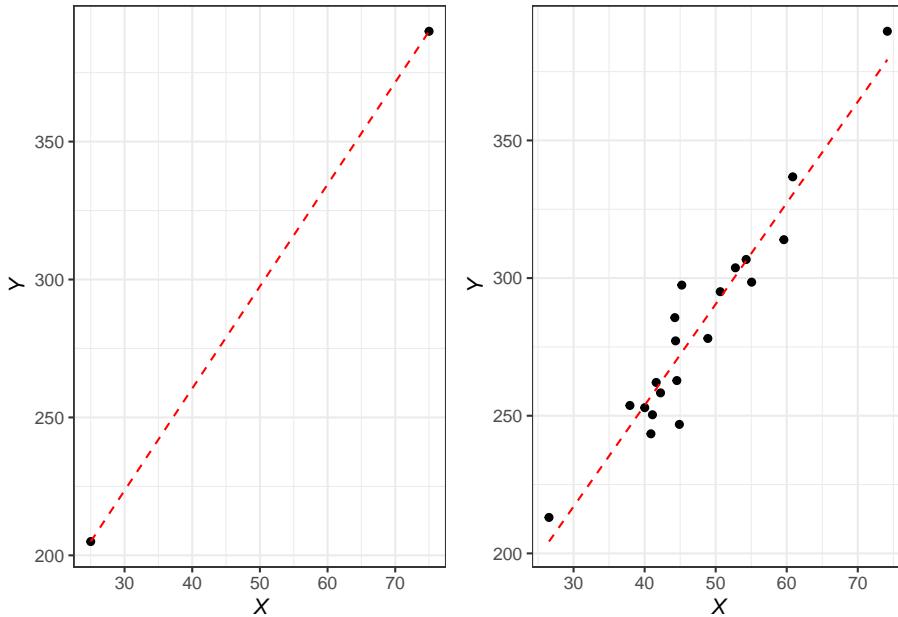


Figure 51: Left: Least squares regression model fit to $n = 2$ observations. Right: Least squares regression model fit to $n = 20$ observations.

In R, we can build (or *fit*) a simple linear regression model using the `lm()` function. The syntax is `lm(Y ~ X, dataset)`:

```

# Load library
library(peopleanalytics)

# Load data
data("employees")

# Subset employees data frame; leads are only applicable for
# those in sales positions
data <- subset(employees, job_title %in% c('Sales Executive',
# 'Sales Representative'))

# Regress YTD leads on engagement
slm.fit <- lm(ytd_leads ~ engagement, data)

```

In practice, linear assumptions are rarely – if ever – perfectly met, but there must be evidence that the assumptions are *generally* satisfied.

Before performing model diagnostics, it is important to note the following:

1. Collinearity diagnostics are only applicable in the context of multiple regression, as simple linear models have only one IV (this will be covered later in the chapter).
2. Outliers are not always an issue, as we discussed in Chapter @ref(data-prep). Figure @ref(fig:high-leverage) illustrates differences between an outlier that does not influence the model fit (left) relative to one which has significant leverage on the model fit (right).

We can conveniently perform linear model diagnostics using the `plot()` function in conjunction with the object holding linear model results (`slm.fit`). This produces the following standard plots shown in Figure @ref(fig:slm-diagnostics):

- **Residuals vs Fitted:** Shows how residuals (y -axis) vary across the range of fitted values (x -axis)
- **Normal Q-Q:** Compares two probability distributions by plotting their quantiles (data partitioned into equal-sized groups) against each other
- **Scale-Location:** Shows how *standardized* residuals (y -axis) vary across the range of fitted values (x -axis)
- **Residuals vs Leverage:** Shows the leverage of each data point (x -axis) against their standardized residuals (y -axis)

The *Residuals vs Fitted* and *Scale-Location* plots help evaluate assumptions of homoscedasticity, linearity, and normality – which are intricately linked. Data are heteroscedastic if there is flarring or funnel patterning about the residuals across the range of fitted values. That is, there must be constant variance with

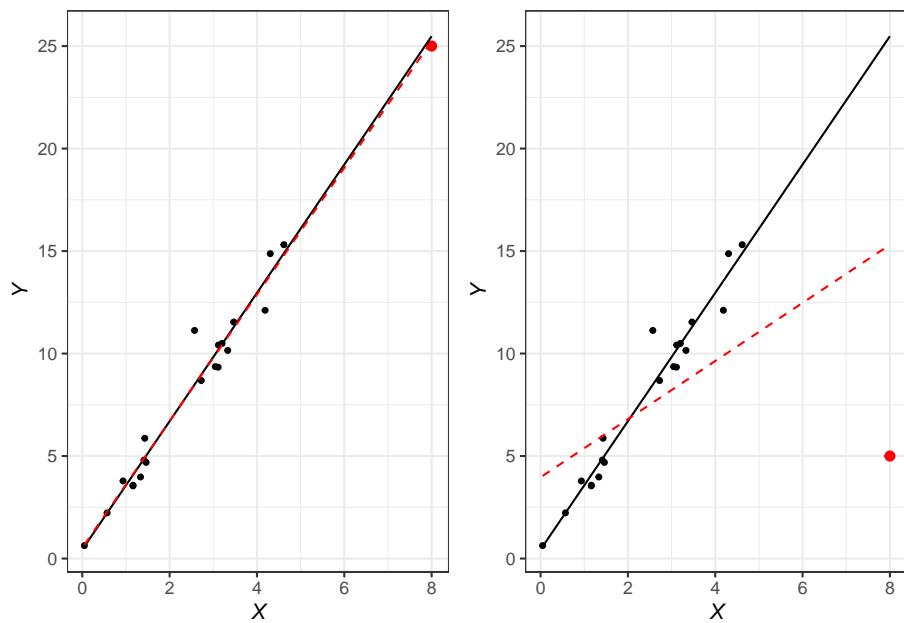


Figure 52: Left: Model fit with non-influential outlier. Right: Model fit with high leverage outlier. Outlier shown in red. Black solid line represents model fit without outliers. Red dashed line represents model fit with outliers.

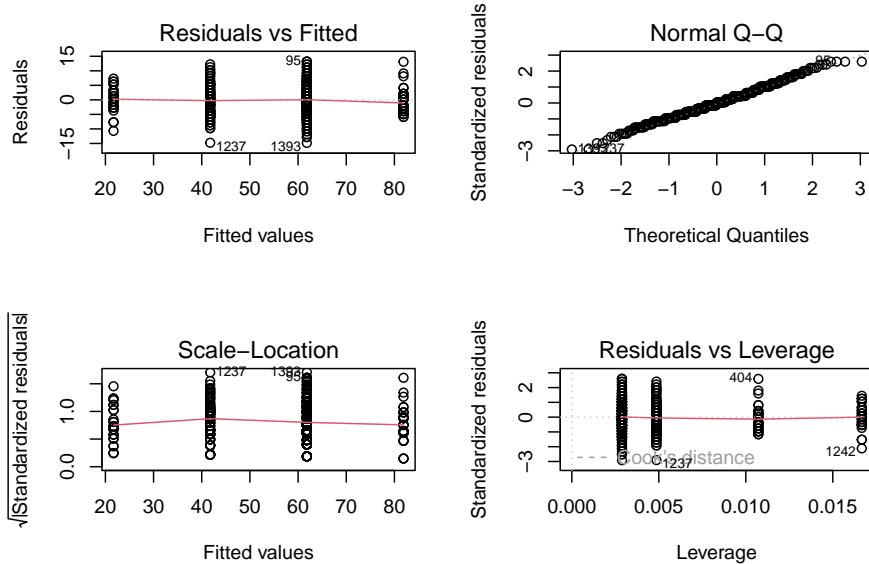


Figure 53: Simple linear regression model diagnostics.

respect to the residual errors in order for the assumption of homoscedasticity to be met. This occurs when there is a linear relationship between X and Y , in which case residuals will be normally distributed around a mean of 0. While the spread of residuals is greater for larger fitted values in this model, resulting in the lower standard error for smaller fitted values indicated in the *Scale-Location* plot, the slope of the line in the *Residuals vs Fitted* plot is effectively flat which indicates that the model does not perform significantly better for certain fitted values relative to others.

Cook's distance, shown in the *Residuals vs Leverage* plot, provides a measure of how much our model estimates for all observations change if high leverage observations are removed from the data. Higher numbers indicate stronger influence. R conveniently labels the three observations with the highest leverage, though the degree of leverage is only problematic for observations beyond the dashed Cook's distance line. In this case, there are no observations with enough leverage for the dashed Cook's distance line to show on the plot, so no action is warranted.

In addition to the visual inspection, we can perform the **Breusch-Pagan test** using the `bptest()` function from the `lmtest` library to test the null hypothesis that the data are homoscedastic. If $p < .05$ for the test statistic, we reject the null hypothesis and conclude that there is evidence of heteroscedasticity in the data.

```
# Run the Breusch-Pagan test for evaluate homoscedasticity
lmtest::bptest(slm.fit)

##
## studentized Breusch-Pagan test
##
## data: slm.fit
## BP = 0.07603, df = 1, p-value = 0.7828
```

Since $p \geq .05$, we fail to reject the null hypothesis of homoscedasticity; therefore, this assumption is satisfied. If this was not the case, a common approach to addressing heteroscedasticity is transforming the response variable by taking the natural logarithm (`log()`) or square root (`sqrt()`) of Y . While transformations may correct for violations of linear model assumptions, they also result in a less intuitive interpretation of model output relative to the raw untransformed data.

Let's illustrate how to transform the response variable:

```
# Square root transformation of YTD leads
slm.fit.trans <- lm(sqrt(ytd_leads) ~ engagement, data)

# Natural logarithmic transformation of YTD leads
slm.fit.trans <- lm(log(ytd_leads) ~ engagement, data)
```

The *Normal Q-Q Plot* in Figure @ref(fig:slm-diagnostics) is used to test the assumption of normally distributed model residuals. A perfectly normal distribution of residuals will result in data lying along the line situated at 45-degrees from the x -axis. Based on a visual inspection, our residuals appear to be normally distributed, as there are only a small number of minor departures in the upper and lower ends of the quantile range.

We can also visualize the distribution of model residuals using a histogram. In the majority of cases, the residual should be 0; this indicates the model correctly estimates YTD leads, resulting in no difference between estimated and observed values ($\hat{y} - y = 0$).

```
# Produce histogram to visualize distribution of model residuals
ggplot2::ggplot() +
  ggplot2::aes(slm.fit$residuals) +
  ggplot2::labs(x = "YTD Leads Residuals", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..), fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
  ggplot2::theme_bw()
```

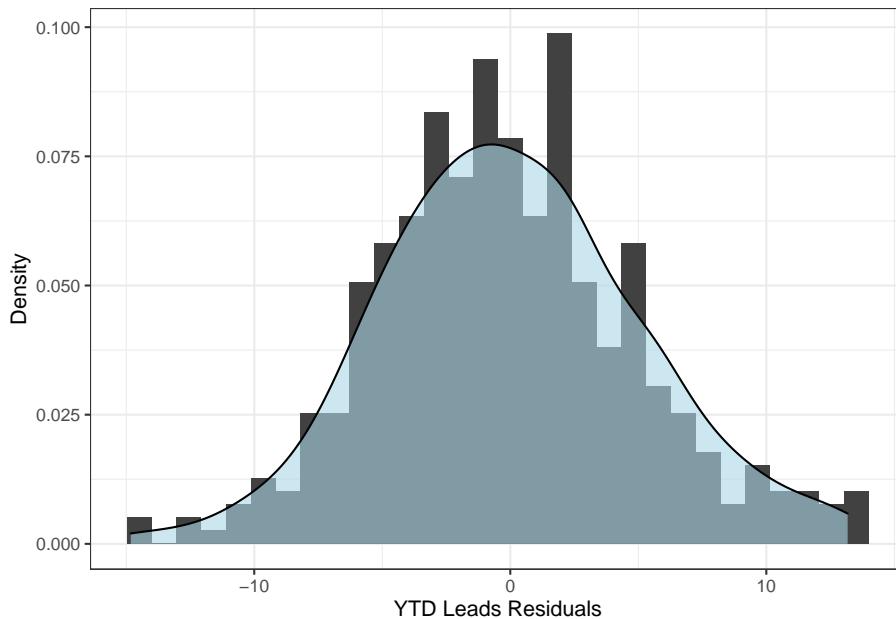


Figure 54: Distribution of model residuals.

Based on both the Normal Q-Q Plot and histogram, the residuals conform to the assumption of normality. We can confirm using the Shapiro-Wilk test, in which a non-significant test statistic ($p \geq .05$) is sufficient evidence of normality:

```
# Compute Shapiro-Wilk test of normality
shapiro.test(slm.fit$residuals)

##
##  Shapiro-Wilk normality test
##
## data: slm.fit$residuals
## W = 0.99339, p-value = 0.07029
```

Next, let's display our simple linear model results:

```
# Produce model summary
summary(slm.fit)
```

```
##
## Call:
```

```

## lm(formula = ytd_leads ~ engagement, data = data)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -14.8236 -3.7591  0.1118  3.1764 13.1764
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.6301    0.9970   1.635   0.103
## engagement  20.0645    0.3571  56.193  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.095 on 407 degrees of freedom
## Multiple R-squared:  0.8858, Adjusted R-squared:  0.8855
## F-statistic:  3158 on 1 and 407 DF,  p-value: < 2.2e-16

```

There are several important pieces of information in this output:

- **Estimate:** *Unstandardized* Beta coefficient associated with the predictor
- **Std. Error:** Average distance between the observed and estimated values per the fitted regression line
- **t value:** Test statistic calculated by `Estimate / Standard Error`. Larger values provide more evidence for a non-zero coefficient (relationship) for the respective predictor in the population.
- **Pr(>|t|):** *p*-value for evaluating whether there is sufficient evidence in the sample that the coefficient (relationship) between the respective predictor and response variable is not 0 in the population (i.e., *x* has a relationship with *y*).
- **Intercept:** Mean value of the response variable when all predictors are equal to 0. Note that the interpretation of the intercept is often nonsensical since many predictors cannot have 0 values (e.g., age, day, month, quarter, year).
- **Signif. codes:** Symbols to quickly ascertain whether predictors are significant at key levels, such as $p < .001$ (***) $, p < .01$ (**), or $p < .05$ (*).
- **Residual standard error:** Measure of model fit which reflects the standard deviation of the residuals ($\sqrt{\sum(y - \hat{y})^2/df}$)
- **Degrees of freedom:** $n - p$, where n is the number of observations and p is the number of predictors
- **Multiple R-squared:** Percent of variance in *y* (when multiplied by 100) explained by the predictors in the model. This is also known as the **Coefficient of Determination**. For simple linear regression, this is simply the squared value of Pearson's *r* for the bivariate relationship between the predictor and response (execute `cor(data$engagement, data$ytd_leads)^2` to validate).

- **Adjusted R-squared:** Modified version of R^2 that adjusts the estimate for non-significant predictors. A large delta between R^2 and Adjusted R^2 coefficients generally indicates a model containing a larger number of non-significant predictors.
- **F-statistic:** Statistic used in conjunction with the p-value for testing differences between the specified model and an intercept-only model (a model with no predictors). This test helps us evaluate whether our predictors are helpful in explaining variance in y .

The output of this simple linear regression model indicates that for each one-unit increase in engagement, the average increase in YTD leads is about 20 ($\beta = 20.1$, $t(407) = 56.2$, $p < .001$). Had we transformed the response variable from its original unit of measurement, the interpretation would be expressed in the transformed units (e.g., β is the square root or natural log of the average change in leads for a one-level increase in engagement).

With these normally distributed *residuals*, we can draw upon the properties of the CLT and conclude that the true relationship between engagement and YTD leads in the population is statistically unlikely to be 0 since the 95% CI ($\beta \pm 2SE$) does not include 0.

While it may be tempting to conclude that employee engagement has a significant influence on leads based on the model output, we know that bivariate relationships may be spurious; that is, engagement may be correlated with another variable that is actually influencing leads. In practice, a simple linear model is rarely sufficient for explaining a meaningful percent of variance in a response variable – especially in a social science context. Additional predictors are usually needed to capture the complex and nuanced relationships characteristic of people analytics problems.

The R^2 value indicates that 8.9% of the variance in leads can be explained by the variation in engagement levels. Put differently, this simple model does not account for 91.1% of variation in leads. Since a large portion of the variance in leads is unexplained, we need signal from additional predictors to understand the other dimensions along which leads vary.

Figure @ref(fig:slm-fit) illustrates how the regression equation for this simple linear model ($y = 20.1x + 1.6$) fits the data points for sales employees. The distribution of leads at each engagement level indicates that there are other factors that explain variance in leads that need to be accounted for in the regression equation to achieve more accurate estimates. The reduction in the spread of leads for a combination of significant predictor values increases R^2 (explained variance in YTD leads).

We can use the `extract_eq()` function from the `equatiomatic` library to easily build a properly formatted LaTex regression equation from the model object:

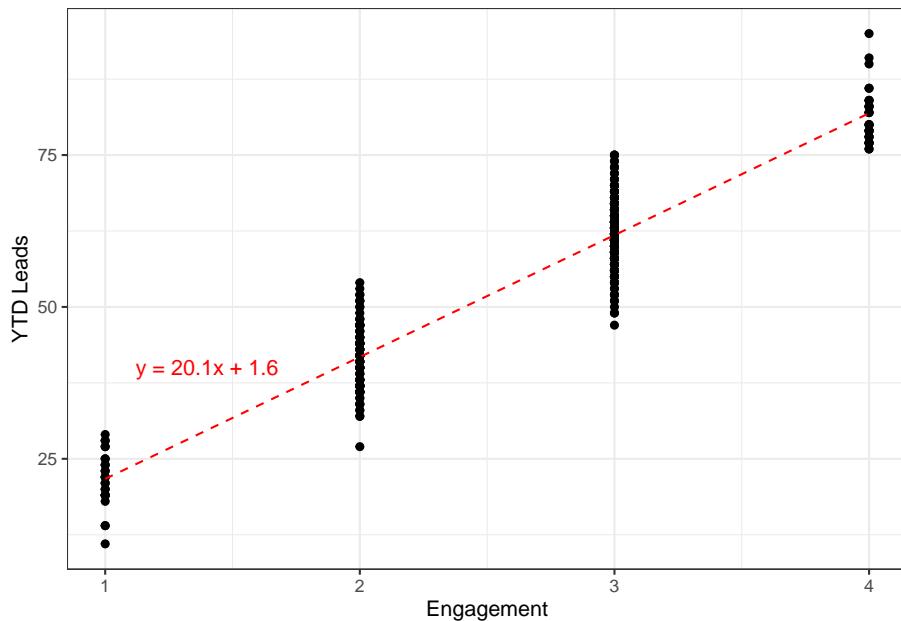


Figure 55: Simple linear model fit line for $y = 20.1x + 1.6$

```
# Load library
library(equatiomatic)

# Convert model to LaTeX regression equation
equatiomatic::extract_eq(slm.fit)
```

$$\text{ytd_leads} = \alpha + \beta_1(\text{engagement}) + \epsilon \quad (1)$$

Multiple Linear Regression

Multiple linear regression extends the simple linear model to one with two or more predictor variables. Assuming the multiple predictors add meaningful information, multiple regression models generally explain more variance in the response relative to simple linear models, and is defined by:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon$$

Once we estimate the unknown model coefficients, β_0 through β_p , we can estimate Y for a particular combination of values for X_1 through X_p by calculating:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \dots + \hat{\beta}_p x_{ip}$$

Collinearity Diagnostics

In addition to the assumptions we tested in the context of the simple linear model, multiple linear regression warrants collinearity diagnostics. **Collinearity** refers to situations in which predictors that are related to the response variable also have strong associations with one another. In practice, there is usually some level of collinearity between variables, so the goal of collinearity diagnostics is to identify and address *problematic* levels of collinearity.

Models should be built with predictors that have a strong association with the outcome but not with one another. If predictors are highly correlated with each other, it indicates that they are redundant and do not provide unique information. A large amount of collinearity can cause serious issues with the underlying calculus of a regression model, which can manifest in the form of effects of significant predictors being masked or suppressed or a negative sign/effect showing in the output when a positive association between the predictor and the response actually exists (or vice versa). As a result, it would be premature to fit a linear model before running collinearity diagnostics, as there may be false negatives – predictors that appear unimportant but are actually statistical drivers of the response. If problematic collinearity is not addressed, false conclusions may be drawn from the model output which may lead to bad business decisions.

Kuhn and Johnson (2013) recommend the simple procedure outlined below to identify and address problematic collinearity:

1. Determine the two predictors associated with the largest absolute pairwise correlation (whether they are positively or negatively related does not matter) – call them predictors A and B.
2. Determine the average absolute correlation between predictor A and the other variables. Do the same for predictor B.
3. If predictor A has a larger average absolute correlation, remove it; otherwise, remove predictor B. The exception to this rule is when predictors A and B have similar average absolute correlations with all other predictors but the predictor with the slightly higher correlation is a key variable that, if dropped, will prevent you from addressing one or more stated objectives or hypotheses.
4. Repeat steps 1-3 until $|r| < .7$ for each pair of predictors.

Let's demonstrate the procedures and mechanics for multiple linear regression by estimating YTD sales using multiple predictor variables. While not appropriate in practice, we will select a subset of the available predictors from `data` to simplify this example. In Chapter @ref(pred-mod), we will discuss the use of machine learning (ML) models for more efficient and comprehensive variable selection.

We can leverage the `ggpairs()` function from the `GGally` library introduced in Chapter @ref(desc-stats) to efficiently compute bivariate correlations and visualize relationships:

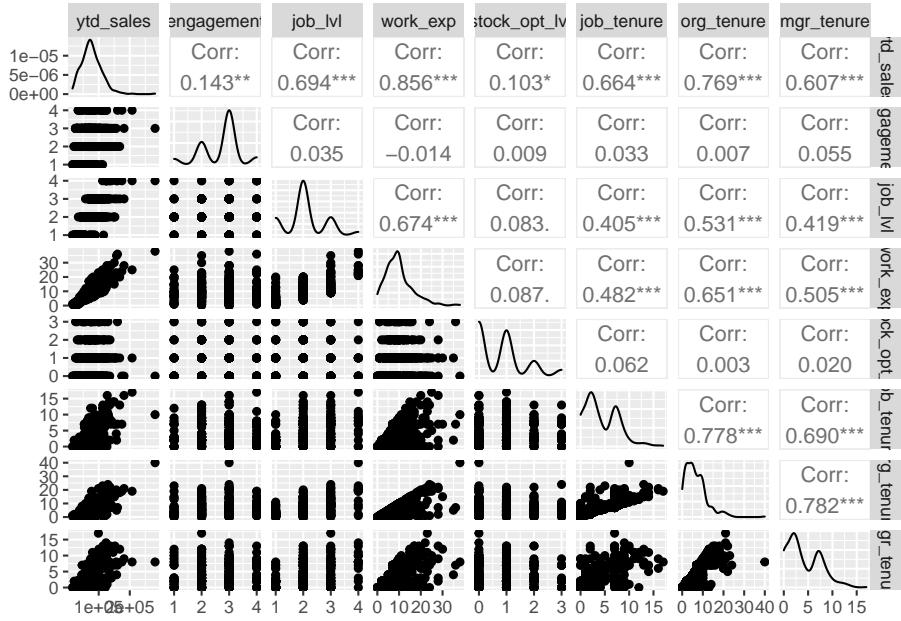


Figure 56: GGpairs bivariate correlations and data distributions

Based on the correlations, `org_tenure` is highly correlated with `job_tenure`, `mgr_tenure` and `work_exp`. These relationships indicate that job and manager changes for those in sales roles have been infrequent since joining the organization, and that a large portion of their work experience has been with this organization. Since `org_tenure` has the strongest relationship with our response, `ytd_sales`, we will drop `job_tenure`, `mgr_tenure`, and `work_exp`.

Since $|r| < .7$ for all pairwise relationships, let's fit the more parsimonious multiple regression model using the resulting subset of predictors. We can chain together multiple predictors in the model using the `+` symbol in the `lm()` function:

```
# Regress YTD sales on a combination of predictors
mllm.fit <- lm(ytd_sales ~ engagement + job_lvl + stock_opt_lv +
~ org_tenure, data)
```

While Kuhn and Johnson's procedure is a good first step, this may not eliminate what is known as **multicollinearity**, which is collinearity among three or more predictors. It is possible for collinearity to exist between three or more

variables, even in the absence of a strong correlation for a pair of variables. We can evaluate the **Variance Inflation Factor (VIF)** for the predictors that remain following the bivariate correlation review to ensure multicollinearity is not present. *VIF* is defined by:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R_{X_j|X_{-j}}^2},$$

where the denominator, $R_{X_j|X_{-j}}^2$, is the R^2 from regressing X_j onto all other predictors. The smallest value of *VIF* is 1, which indicates a complete absence of collinearity. Problematic collinearity exists if *VIF* for any variable exceeds 5.

We can produce *VIF* for each variable using the `vif()` function from the `car` library:

```
# Load library
library(car)

# Produce VIF for each predictor
car::vif(mlm.fit)

##      engagement      job_lvl stock_opt_lvl      org_tenure
##      1.001459       1.407192      1.009435       1.395797
```

Based on the output, $VIF < 5$ for each predictor, which indicates that multicollinearity is not an issue.

Next, let's evaluate the linear model assumptions to validate that fitting a linear model to these data is appropriate:

Based on these visuals, there are obvious violations of linear model assumptions that need to first be addressed.

First, given the long right tail for the `ytd_sales` distribution shown in Figure @ref(fig:pre-mlm-diagnostics), let's apply a square root transformation to the response variable:

```
# Regress YTD sales on a combination of predictors
mlm.fit <- lm(sqrt(ytd_sales) ~ engagement + job_lvl +
  stock_opt_lvl + org_tenure, data)
```

Additionally, the diagnostic plots indicate that there are data points with high leverage on the fit. Let's address using Cook's distance as the criterion:

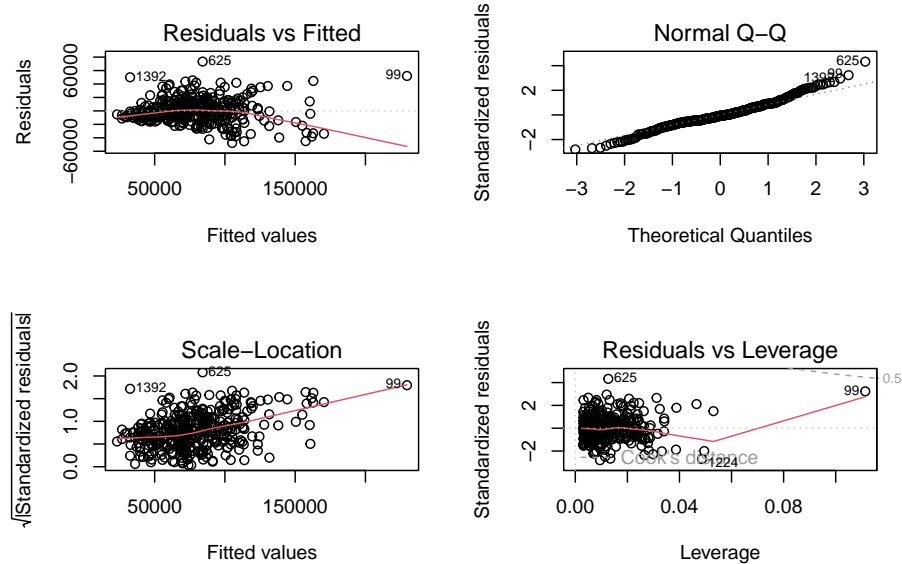


Figure 57: Multiple linear regression model diagnostics

```
# Remove high leverage observations per Cook's distance
w <- abs(rstudent(mlm.fit)) < 3 & abs(cooks.distance(mlm.fit)) <
  4/nrow(mlm.fit$model)
mlm.fit <- update(mlm.fit, weights = as.numeric(w))
```

Now we can produce a refreshed set of diagnostic plots to evaluate the impact of transforming the response variable and removing high leverage observations:

There is a clear improvement towards satisfying linear model assumptions.

Let's perform the Breusch-Pagan test to validate that the assumption of homoscedasticity is met:

```
# Run the Breusch-Pagan test for evaluate homoscedasticity
lmtest::bptest(mlm.fit)
```

```
##
## studentized Breusch-Pagan test
##
## data: mlm.fit
## BP = 2.2111, df = 4, p-value = 0.697
```

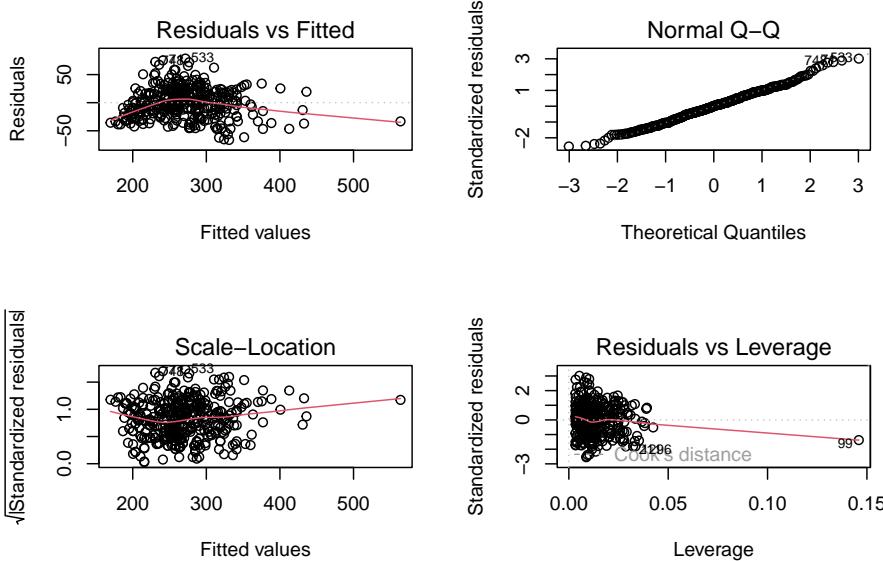


Figure 58: Multiple linear regression model diagnostics (post-transformation)

Next, let's ensure residuals are normally distributed around 0:

```
# Produce histogram to visualize distribution of model residuals
ggplot2::ggplot() +
  ggplot2::aes(mlm.fit$residuals) +
  ggplot2::labs(x = "YTD Leads Residuals", y = "Density") +
  ggplot2::geom_histogram(aes(y = ..density..), fill = "#414141") +
  ggplot2::geom_density(fill = "#ADD8E6", alpha = 0.6) +
  ggplot2::theme_bw()

# Compute Shapiro-Wilk test of normality
shapiro.test(mlm.fit$residuals)

## 
## Shapiro-Wilk normality test
## 
## data: mlm.fit$residuals
## W = 0.99342, p-value = 0.07202
```

Based on the diagnostic plots and statistical tests, our data satisfy the requirements for building a multiple linear regression model.

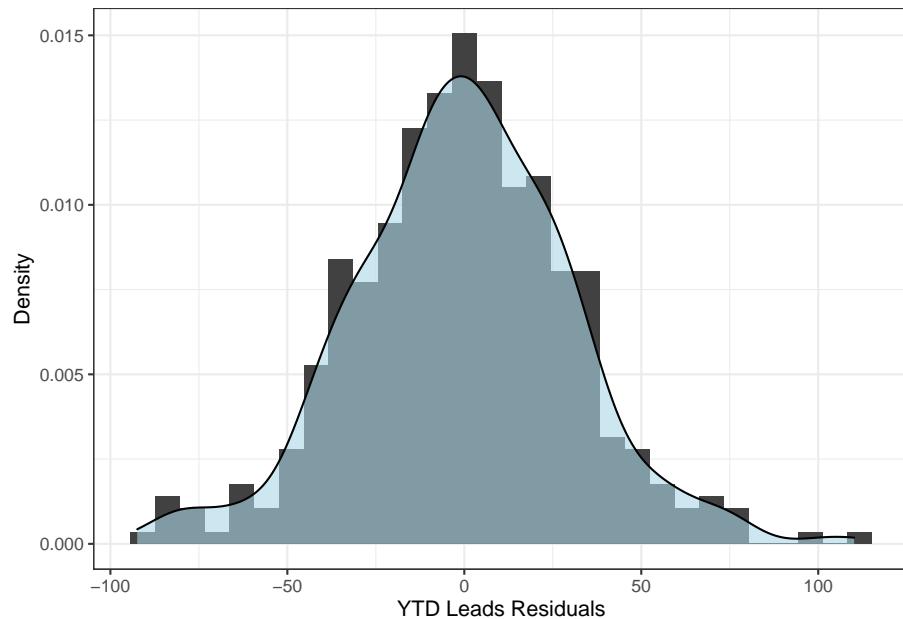


Figure 59: Distribution of model residuals

Variable Selection

Next, we need to reduce our model to the subset of predictors with statistically significant relationships with the response variable. **Backward Stepwise Selection** is a common and simple variable selection procedure, and the steps are outlined below:

1. Remove the predictor with the highest p -value greater than the critical value ($\alpha = .05$).
2. Refit the model, and repeat step 1.
3. Stop when all p -values are less than the critical value.

Each predictor in our model has a statistically significant relationship with `ytd_sales` – indicating that the slope of the relationships with the response is unlikely 0 in the population – so further variable reduction is not required.

```
##  
## Call:  
## lm(formula = sqrt(ytd_sales) ~ engagement + job_lvl + stock_opt_lvl +  
##      org_tenure, data = data, weights = as.numeric(w))  
##  
## Weighted Residuals:
```

```

##      Min    1Q Median    3Q   Max
## -66.31 -16.01   0.00  16.72  78.50
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept) 125.9807     6.9032 18.250 < 2e-16 ***
## engagement   10.1046     1.9742  5.118 4.93e-07 ***
## job_lvl      33.7679     2.4254 13.922 < 2e-16 ***
## stock_opt_lvl 5.1662     1.6211  3.187  0.00156 **
## org_tenure    6.8118     0.3414 19.952 < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.15 on 375 degrees of freedom
## Multiple R-squared:  0.7783, Adjusted R-squared:  0.7759
## F-statistic: 329.1 on 4 and 375 DF, p-value: < 2.2e-16

```

We can view the multiple regression equation for estimating `sqrt(ytd_sales)` with these four predictors using the `extract_eq` function:

```
# Convert model to LaTeX regression equation
equatiomatic::extract_eq(mlm.fit)
```

$$\text{sqrt(ytd_sales)} = \alpha + \beta_1(\text{engagement}) + \beta_2(\text{job_lvl}) + \beta_3(\text{stock_opt_lvl}) + \beta_4(\text{org_tenure}) + \epsilon \quad (2)$$

Based on the model output, the combination of predictors explains about 78% of the variance in YTD sales ($R^2 = .778$). In people analytics settings, it is rare to explain three-quarters of the variance for an outcome given people data are especially noisy.

By default, the coefficients on the predictors are unstandardized; that is, they represent the average change in the square root transformed response for each one-unit increase for the respective predictor. Since the predictors have different units of measurement, such as `stock_opt_lvl` ranging from 0 to 3 and `org_tenure` ranging from 0 to 40, the unstandardized coefficients cannot be compared to determine which variable has the largest effect on YTD sales. We must standardize these coefficients and adjust for differences in the units of measurement for an apples-to-apples comparison.

We can scale variables by subtracting the variable's mean from x and dividing the difference into the variable's standard deviation:

$$x_{\text{scaled}} = \frac{x_i - \bar{x}}{s}$$

We can leverage the `scale()` function to standardize the predictors' units of measurement and determine which has the largest effect on `ytd_sales`:

```
## 
## Call:
## lm(formula = ytd_sales ~ engagement + job_lvl + stock_opt_lvl +
##     org_tenure, data = data_std)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.47125 -0.28770 -0.04054  0.30025  2.26587 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -3.313e-18  2.606e-02   0.000  1.00000  
## engagement   1.247e-01  2.611e-02   4.777 2.49e-06 *** 
## job_lvl      3.845e-01  3.095e-02  12.423 < 2e-16 *** 
## stock_opt_lvl 6.788e-02  2.622e-02   2.589  0.00997 ** 
## org_tenure    5.637e-01  3.083e-02  18.285 < 2e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 0.527 on 404 degrees of freedom
## Multiple R-squared:  0.7249, Adjusted R-squared:  0.7222 
## F-statistic: 266.2 on 4 and 404 DF,  p-value: < 2.2e-16
```

Based on the standardized coefficients in Figure @ref(fig:mlm-scaled), `org_tenure` has the largest effect ($\beta = .56$, $t(404) = 18.29$, $p < .001$) and `job_lvl` has the second largest effect ($\beta = .39$, $t(404) = 12.42$, $p < .001$).

Moderation

As discussed in Chapter @ref(measure-sampl), a moderating variable is a third variable which amplifies (strengthens) or attenuates (weakens) the relationship between an IV and the response. Accounting for a moderating variable in a linear model requires an **interaction term**, which is the product of the two variables ($X_1 * X_2$).

Let's examine whether `org_tenure` influences the strength of the relationship between `job_lvl` and `sqrt(ytd_sales)`. We would generally expect sales to increase as the job level of salespeople increases, and longer tenure may amplify the strength of this association. Step one is testing whether the interaction term is statistically significant, and step two is determining the nature of any observed statistical interaction. Including the interaction term in the model (`job_lvl * org_tenure`) will add the predictors independently *and* jointly:

```

## 
## Call:
## lm(formula = sqrt(ytd_sales) ~ job_lvl * org_tenure, data = data)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -89.090 -16.918 -0.589 17.490 107.461 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 123.8354   6.2497 19.815 < 2e-16 ***
## job_lvl      50.5727   3.0119 16.791 < 2e-16 ***  
## org_tenure    12.4813   0.8653 14.425 < 2e-16 ***  
## job_lvl:org_tenure -2.5142   0.2996 -8.393 8.01e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 29.4 on 405 degrees of freedom
## Multiple R-squared:  0.7412, Adjusted R-squared:  0.7393 
## F-statistic: 386.6 on 3 and 405 DF,  p-value: < 2.2e-16

```

The results show that both the main and interaction effects are statistically significant ($p < .001$).

Since interaction terms can be highly correlated with independent predictors, it is good to check for collinearity:

```
# Produce VIF for each model term
car::vif(mlm.fit.int)
```

##	job_lvl	org_tenure	job_lvl:org_tenure
	2.189565	9.246489	12.308142

VIF is greater than 5 for both `org_tenure` and the interaction term; therefore, there is a problematic level of collinearity between these variables.

A common method of addressing collinearity in the context of interaction testing is **variable centering**, in which each value of the predictor is subtracted from its mean ($x - \bar{x}$). Unlike other transformations we have explored, centering does not impact the interpretation of model coefficients. Coefficients continue to represent the average change in the response for a one-unit change in a predictor, as the range of values for centered variables is consistent with the range for untransformed variables. However, the coefficients on centered variables may be considerably different relative to the model with untransformed variables due to the effects of high collinearity.

```

## Call:
## lm(formula = sqrt(ytd_sales) ~ job_lvl_cntrd * org_tenure_cntrd,
##      data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -89.090 -16.918 -0.589 17.490 107.461
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                276.5606   1.5653 176.680 < 2e-16 ***
## job_lvl_cntrd              34.0427   2.4065 14.146 < 2e-16 ***
## org_tenure_cntrd            7.2623   0.3789 19.165 < 2e-16 ***
## job_lvl_cntrd:org_tenure_cntrd -2.5142   0.2996 -8.393 8.01e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.4 on 405 degrees of freedom
## Multiple R-squared:  0.7412, Adjusted R-squared:  0.7393
## F-statistic: 386.6 on 3 and 405 DF,  p-value: < 2.2e-16

# Produce VIF for centered variables
car::vif(mlm.fit.int)

```

```

##                  job_lvl_cntrd          org_tenure_cntrd
##                         1.397803                      1.773279
## job_lvl_cntrd:org_tenure_cntrd
##                         1.337702

```

After centering the variables, *VIF* is well beneath the threshold of 5.

Comparing Figure @ref(fig:mlm-int) to Figure @ref(fig:mlm-int-cntrd), we can observe that the main effects for `job_lvl` and `org_tenure` are inflated – and population parameter estimates less precise (larger *SE*) – when high collinearity is present.

To better understand the nature of the interaction effect ($\beta = -2.51$, $t(405) = -8.39$, $p < .001$), two equations can be built to evaluate changes in the slope of the relationship with high ($\bar{x} + 1s$) and low ($\bar{x} - 1s$) organization tenure:

- *High organization tenure*: $Y = -2.51(6.57 + 5.12)X + 276.56$
- *Low organization tenure*: $Y = -2.51(6.57 - 5.12)X + 276.56$,

where 6.57 is `mean(data$org_tenure)`, 5.12 is `sd(data$org_tenure)`, X is a vector of values for `job_lvl`, and 276.56 is the *y*-intercept.



Figure 60: Regression of square root transformed YTD sales onto job level x organization tenure interaction term. High organization tenure (red line): $Y = -2.51 (6.57 + 5.12)X + 276.56$. Low organization tenure (blue line): $Y = -2.51 (6.57 - 5.12)X + 276.56$.

As shown in Figure @ref(fig:int-effects), the slope of both regression lines is negative. However, the drop in sales is much more significant as job level increases for those with high ($\bar{x}+1s$) organization tenure relative to those with low ($\bar{x}-1s$) organization tenure. Perhaps those with longer tenure in the organization gain additional responsibilities beyond selling (e.g., mentoring junior salespeople) as they are promoted into higher job levels.

Mediation

As discussed in Chapter @ref(measure-sampl), mediating variables may fully or partially mediate the relationship between a predictor and response. Full mediation indicates that the mediator fully explains the effect; in other words, without the mediator in the model, there is no relationship between an IV and DV. Partial mediation indicates that the mediator partially explains the effect; that is, there is still a relationship between an IV and DV without the mediator in the model.

Baron and Kenny's (1986) four-step approach involves several regression analyses to examine paths a, b, and c shown in Figure @ref(fig:med-paths).

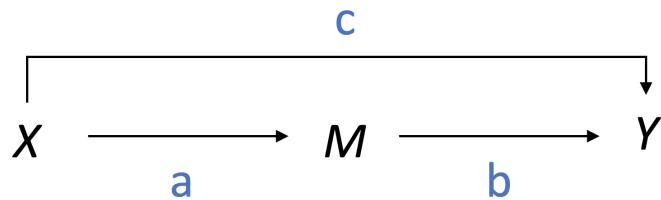


Figure 61: Paths for mediation analysis

- **Step 1:** Fit a simple linear regression model with X predicting Y (path c), $Y = \beta_0 + \beta_1 X + \epsilon$.
- **Step 2:** Fit a simple linear regression model with X predicting M (path a), $M = \beta_0 + \beta_1 X + \epsilon$.
- **Step 3:** Fit a simple linear regression model with M predicting Y (path b), $Y = \beta_0 + \beta_1 M + \epsilon$.
- **Step 4:** Fit a multiple linear regression model with X and M predicting Y (paths b and c), $Y = \beta_0 + \beta_1 X + \beta_2 M + \epsilon$.

The purpose of Steps 1-3 is to determine if zero-order relationships exist. If one or more of these relationships is nonsignificant, mediation is unlikely – though not impossible. Mediation exists if the relationship between M and Y (path b) remains significant after controlling for X in Step 4. If X is no longer significant

in Step 4, support for full mediation exists; if X remains significant, support for partial mediation exists.

Let's illustrate the implementation of this approach in R by testing the following hypothesis: Job level mediates the relationship between education level and YTD sales. Stated differently, the relationship between job level and YTD sales exists because those with more education tend to have higher job levels, and those in higher job levels tend to have stronger sales performance.

```
##  
## Call:  
## lm(formula = sqrt(ytd_sales) ~ ed_lvl, data = data)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -138.34  -36.55    0.50   34.72  249.05  
##  
## Coefficients:  
##                 Estimate Std. Error t value Pr(>|t|)  
## (Intercept)  245.229     8.472  28.945 < 2e-16 ***  
## ed_lvl        9.072     2.740   3.311  0.00101 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 56.89 on 407 degrees of freedom  
## Multiple R-squared:  0.02623,   Adjusted R-squared:  0.02384  
## F-statistic: 10.96 on 1 and 407 DF,  p-value: 0.001012  
  
##  
## Call:  
## lm(formula = job_lvl ~ ed_lvl, data = data)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.19782 -0.19782 -0.08516  0.14016  2.14016  
##  
## Coefficients:  
##                 Estimate Std. Error t value Pr(>|t|)  
## (Intercept)  1.74717    0.10522 16.605 < 2e-16 ***  
## ed_lvl       0.11266    0.03403   3.311  0.00101 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.7065 on 407 degrees of freedom  
## Multiple R-squared:  0.02623,   Adjusted R-squared:  0.02384  
## F-statistic: 10.96 on 1 and 407 DF,  p-value: 0.001012
```

```

## 
## Call:
## lm(formula = sqrt(ytd_sales) ~ job_lvl, data = data)
## 
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -85.572 -28.593  -2.794  25.059 148.629 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 152.761     6.156   24.81 <2e-16 ***
## job_lvl      57.294     2.804   20.43 <2e-16 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 40.51 on 407 degrees of freedom
## Multiple R-squared:  0.5063, Adjusted R-squared:  0.5051 
## F-statistic: 417.4 on 1 and 407 DF,  p-value: < 2.2e-16 

## 
## Call:
## lm(formula = sqrt(ytd_sales) ~ ed_lvl + job_lvl, data = data)
## 
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -87.177 -29.481  -3.048  23.932 146.922 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 146.220     7.805   18.734 <2e-16 *** 
## ed_lvl       2.688     1.975   1.361   0.174    
## job_lvl      56.668     2.839   19.961 <2e-16 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 40.47 on 406 degrees of freedom
## Multiple R-squared:  0.5085, Adjusted R-squared:  0.5061 
## F-statistic: 210 on 2 and 406 DF,  p-value: < 2.2e-16

```

Output from these models show significant paths for Steps 1-3 but when adding both `ed_lvl` and `job_lvl` in the multiple regression model in Step 4, `ed_lvl` is no longer significant. Therefore, support for full mediation exists.

Review Questions

1. What is Ordinary Least Squares (OLS) regression, and how does it work?
2. What assumptions must be satisfied to fit a linear model?
3. What does a statistically significant result for the Breusch-Pagan test indicate about linear model assumptions?
4. What does a statistically significant result for the Shapiro-Wilk test indicate about linear model assumptions?
5. In what ways can high collinearity among predictors impact the quality of model results?
6. When are outliers problematic for fitting a regression model?
7. How is unstandardized β interpreted in the output of a linear model?
8. How does the delta between R^2 and Adjusted R^2 change as additional non-significant variables are included in a model?
9. How does the backward stepwise variable selection procedure work?
10. What is the purpose of interaction effects in a regression model?

Linear Model Extensions

This chapter covers several techniques for expanding the linear regression framework covered in Chapter @ref(lm) to test hypotheses with more nuance and complexity.

Model Comparisons

Assuming it is warranted by the research objective, it is sometimes helpful to subset data and compare coefficients between models to determine how the strength of associations between predictors and the response compares between cohorts. This is a common approach in pay equity studies, as it clearly highlights differences in how a particular factor such as job level, job profile, or geography impacts compensation for male vs. female employees or across ethnic groups.

To illustrate, let's fit a multiple regression model to understand drivers of YTD sales for salespeople with overtime relative to those without overtime.

```
# Subset employees data frame; leads are only applicable for
#   those in sales positions
data <- subset(employees, job_title %in% c('Sales Executive',
#   'Sales Representative'))

# Partition data into overtime and non-overtime groups
data_ot <- subset(data, overtime == 'Yes')
data_nonot <- subset(data, overtime == 'No')

# Regress transformed YTD sales on a combination of predictors
#   for overtime and non-overtime groups
mlm.fit.ot <- lm(sqrt(ytd_sales) ~ engagement + job_lvl +
#   stock_opt_lvl + org_tenure, data_ot)
mlm.fit.nonot <- lm(sqrt(ytd_sales) ~ engagement + job_lvl +
#   stock_opt_lvl + org_tenure, data_nonot)
```

```

## 
## Call:
## lm(formula = sqrt(ytd_sales) ~ engagement + job_lvl + stock_opt_lvl +
##      org_tenure, data = data_ot)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -80.927 -22.171 -1.383 19.740 106.769
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 121.815    14.767   8.249 3.27e-13 ***
## engagement   13.171     4.569   2.883  0.00472 **
## job_lvl      35.983     4.754   7.570 1.10e-11 ***
## stock_opt_lvl  7.139     3.342   2.136  0.03481 *
## org_tenure     5.369     0.722   7.437 2.15e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.78 on 113 degrees of freedom
## Multiple R-squared:  0.688, Adjusted R-squared:  0.6769
## F-statistic: 62.29 on 4 and 113 DF, p-value: < 2.2e-16


## 
## Call:
## lm(formula = sqrt(ytd_sales) ~ engagement + job_lvl + stock_opt_lvl +
##      org_tenure, data = data_nonot)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -81.952 -19.422  0.136 20.813 96.003
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 132.3391    8.6695 15.265 < 2e-16 ***
## engagement   9.8523    2.4721  3.985 8.56e-05 ***
## job_lvl      33.1396    3.0014 11.042 < 2e-16 ***
## stock_opt_lvl  4.6377    2.1587  2.148  0.0325 *
## org_tenure     6.0435    0.4039 14.964 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.98 on 286 degrees of freedom
## Multiple R-squared:  0.7332, Adjusted R-squared:  0.7295
## F-statistic: 196.5 on 4 and 286 DF, p-value: < 2.2e-16

```

Since we are comparing two models, we need not scale the variables since comparing a specific predictor's relationship with the response in the overtime model can be juxtaposed against the same predictor in the non-overtime model using the original units of measurement.

Based on the output shown in Figures @ref(fig:mlm-ot) and @ref(fig:mlm-nonot), the model for salespeople who worked overtime explains more variance in square root transformed `ytd_sales` ($R^2 = .73$) relative to the model for salespeople without overtime ($R^2 = .69$).

We can see that `engagement` has a larger effect on the transformed response among salespeople who worked overtime ($\beta = 13.17$, $t(113) = 2.88$, $p < .01$) relative to those who worked no overtime ($\beta = 9.85$, $t(286) = 3.99$, $p < .001$). In addition, `job_lvl` has a stronger association with the response in the overtime group ($\beta = 35.98$, $t(113) = 7.57$, $p < .01$) relative to the non-overtime group ($\beta = 33.14$, $t(286) = 11.04$, $p < .001$). Given that the intercept (average square root of `ytd_sales` when the values of all predictors are set to 0) is higher for the non-overtime group ($\beta = 132.34$, $t(286) = 15.27$, $p < .001$) than for the overtime group ($\beta = 121.82$, $t(113) = 8.25$, $p < .001$), differences in the coefficients on `job_lvl` may indicate that one's job level is a proxy for skill and capacity to sell more in fewer hours.

Hierarchical Regression

A multiple model approach can also be useful for understanding the incremental value a given variable – or set of variables – provides above and beyond a set of control variables. **Hierarchical regression** is a method by which variables are added to the model in steps, and changes in model statistics are evaluated after each step. Let's use hierarchical regression to test the hypothesis below.

H1: Among salespeople who work overtime, engagement has a significant positive relationship with YTD sales after controlling for job level, stock option level, and organization tenure.

```
##  
## Call:  
## lm(formula = sqrt(ytd_sales) ~ job_lvl + stock_opt_lvl + org_tenure,  
##      data = data_ot)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -73.279 -23.803  -0.339  23.017  96.742  
##  
## Coefficients:  
##                               Estimate Std. Error t value Pr(>|t|)
```

```

## (Intercept) 154.6969    9.6759 15.988 < 2e-16 ***
## job_lvl      37.5715    4.8707  7.714 5.02e-12 ***
## stock_opt_lvl 5.2397    3.3794  1.550   0.124
## org_tenure    5.4935    0.7434  7.389 2.64e-11 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.81 on 114 degrees of freedom
## Multiple R-squared: 0.665, Adjusted R-squared: 0.6562
## F-statistic: 75.44 on 3 and 114 DF, p-value: < 2.2e-16

##
## Call:
## lm(formula = sqrt(ytd_sales) ~ engagement + job_lvl + stock_opt_lvl +
##     org_tenure, data = data_ot)
##
## Residuals:
##    Min      1Q  Median      3Q      Max
## -80.927 -22.171 -1.383 19.740 106.769
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 121.815    14.767   8.249 3.27e-13 ***
## engagement   13.171     4.569   2.883  0.00472 **
## job_lvl      35.983     4.754   7.570 1.10e-11 ***
## stock_opt_lvl 7.139     3.342   2.136  0.03481 *
## org_tenure    5.369     0.722   7.437 2.15e-11 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.78 on 113 degrees of freedom
## Multiple R-squared: 0.688, Adjusted R-squared: 0.6769
## F-statistic: 62.29 on 4 and 113 DF, p-value: < 2.2e-16

```

Comparing Figure @ref(fig:hier-ctrl) to Figure @ref(fig:hier-main), we can determine that the addition of `engagement` to the control set explains an additional 2% of the variance in YTD sales ($\Delta R^2 = .69 - .67 = .02$).

In addition, Figure @ref(fig:hier-ctrl) shows that without `engagement` in the model, `stock_opt_lvl` is not significant. This is a good reminder that regression does not examine bivariate relationships of each predictor with the response *independent of other variables*; rather, the relationships among all variables in the model impact which predictors emerge as having a statistical association with the response.

Multilevel Models

The models covered thus far have focused only on observation-level effects. That is, there has been an inherent assumption that predictor variables have *fixed* effects on the outcome and these effects do not vary based on group(s) to which the observations belong. These models are sometimes referred to as **fixed effects** models.

It is often the case, however, that the strength and nature of predictors' effects on an outcome vary across categorical dimensions. For example, estimating the number of requisitions that can be filled by a Talent Acquisition team over a certain period may require inputs such as the number of recruiters and position backfill expectations based on attrition assumptions. However, the model should probably account for how these factors impact recruiter productivity at the intersections of group-level factors such as geography, job family, and job level as well. Estimates for recruiters who are focused on filling executive-level positions in geographies with a limited talent pool or fiercely competitive labor market will look quite different relative to recruiters focused on entry-level, low-skilled positions that are location agnostic. Failure to incorporate these group-level effects may result in inaccurate estimates or incorrectly concluding that variables are not significant in explaining why recruiters vary in the number of requisitions they can fill.

You may wonder how this concept is different from simply including dummy-coded variables in the model to reflect the groups to which individual observations belong. The difference is that the average value of Y when all predictors are set to 0 – namely the Y -intercept β_0 – does not vary by group with dummy-coded categorical variables. In a multilevel model, the intercept is *random* rather than *fixed* for each group. Group-level effects can also be modeled for select (or all) X variables in addition to varying β_0 for each group.

Consider a linear model constructed to test hypothesized relationships of every X variable with an outcome Y . This is the equivalent of building G independent models, where G is the number of groups, using data subsetted for the respective group:

$$Y_G = \beta_{G0} + \beta_{G1}X_1 + \beta_{G2}X_2 + \dots + \beta_{Gp}X_p + \epsilon$$

In this case, it is easy to consider wrapping the `lm()` function within a loop that iterates through each G group, filtering to each of the respective group's data in turn. However, if we hypothesize that the effects of only *certain* variables depend on the G group, we need to estimate both group-level *and* observation-level effects within the same model. A multilevel model featuring this mixture of fixed and random effects is known as a **mixed effects** model. This is also known as **Hierarchical Linear Modeling (HLM)**, which is materially different from Hierarchical Regression covered in the prior section, which compared nested regression models.

A model in which group-level effects are hypothesized for β_0 and X_1 and observation-level effects are hypothesized for all other predictors is expressed as:

$$Y_G = \beta_{G0} + \beta_{G1}X_1 + \beta_2X_2 + \dots + \beta_pX_p + \epsilon$$

To fit a linear mixed effects model in R, we can leverage the `lmer()` function from the `lmerTest` package. Let's demonstrate how to fit a model to understand the random effects of `stock_opt_lvl` and fixed effects of `engagement`, `job_lvl`, and `org_tenure` on `sqrt(ytd_sales)`:

```
# Load library
library(lmerTest)

# Fit linear mixed model
lme.fit <- lmerTest::lmer(sqrt(ytd_sales) ~ engagement + job_lvl
                         + (1 | stock_opt_lvl) + org_tenure, data_ot)

# Summarize model results
summary(lme.fit)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [ 
## lmerModLmerTest]
## Formula: sqrt(ytd_sales) ~ engagement + job_lvl + (1 | stock_opt_lvl) +
##          org_tenure
## Data: data_ot
##
## REML criterion at convergence: 1141.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.52388 -0.63661  0.00411  0.61215  3.13684
##
## Random effects:
## Groups      Name        Variance Std.Dev.
## stock_opt_lvl (Intercept) 51.16    7.152
## Residual           1069.27  32.700
## Number of obs: 118, groups: stock_opt_lvl, 4
##
## Fixed effects:
##             Estimate Std. Error     df t value Pr(>|t|)    
## (Intercept) 133.6129   14.5677  88.8995  9.172 1.67e-14 ***
## engagement   12.0038    4.5140 113.9662  2.659  0.00896 **  
## job_lvl      35.8950    4.7470 112.4054  7.562 1.17e-11 ***
```

```

## org_tenure    5.2542      0.7265 113.5918    7.232 5.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) enggnm jb_lvl
## engagement -0.727
## job_lvl     -0.445 -0.110
## org_tenure   0.046 -0.054 -0.463

```

The results of `lmer()` contain sections for both fixed and random effects. Consistent with the interpretation of linear regression model output, we can see that the fixed effects of each predictor are statistically significant. The key difference here is that the variance shown for the intercept of the random effects model is large. This indicates that there are meaningful differences in the relationships between predictors and `sqrt(ytd_sales)` across the levels of `stock_opt_lvl` that would be missed without a mixed model that accounts for these group-level effects.

For a more comprehensive treatment on multilevel models, see Gelman and Hill (2006).

Polynomial Regression

Linear regression is a powerful approach to understanding the relative strength of predictors' associations with a response variable. While linear models have advantages in interpretation, inference, and implementation simplicity, the linearity assumption often limits predictive power since this assumption is often a poor approximation of actual relationships in the data. In this section, we will discuss how to extend the linear regression framework and relax linear model assumptions to handle non-linear relationships.

In a people analytics context, many data sets are cross-sectional and time-invariant, meaning they represent data collected at a single point in time. However, data collected across multiple points in time (time series data) are needed for forecasting future values of a dependent variable (e.g., a workforce planning model that estimates hires by month).

There is often a seasonality element inherent in the relationship between time and the outcome that is being estimated, which requires accounting for time-variant features (e.g., monthly attrition rate assumptions). **Seasonality** is the variation that occurs at regular intervals within a year. For example, companies with an annual bonus often experience a seasonal spike in voluntary attrition following bonus payouts (beginning in March for many organizations). Accounting for seasonality in models helps reduce error, but it requires estimating a more complex set of model coefficients relative to a more naive linear projection.

The simple linear regression equation, $Y = \beta_0 + \beta_1 X + \epsilon$, can be easily extended to include higher order polynomial terms and achieve a more flexible fit. This is known as **polynomial regression**.

- Quadratic (2nd Order Polynomial) Regression Equation: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
- Cubic (3rd Order Polynomial) Regression Equation: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

Figure @ref(fig:poly-fun) illustrates how higher-order polynomial functions can fit more curvilinear trends relative to a simple linear projection.

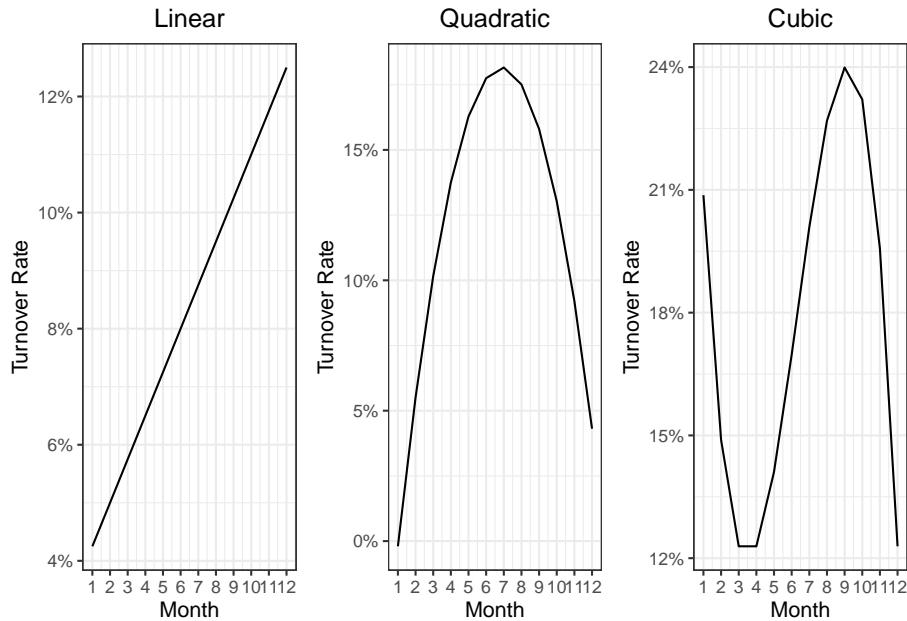


Figure 62: Left: Linear turnover trend for $y = .75x + 3.5$. Middle: Quadratic turnover trend for $y = 7.3x - .53x^2 - 6.97$. Right: Cubic turnover trend for $y = -12.48x + 2.47x^2 - .13x^3 + 31.01$.

It is important to note that adding higher order terms to the regression equation usually increases R^2 due to a more flexible fit to the data, but the additional coefficients are not necessarily significant. R^2 will approach 1 as the power of x approaches $n - 1$ since the fit line will connect every data point. However, a model that results in a perfect – or near perfect – fit is likely too flexible to generalize well to other data. This problem is known as overfitting and will be covered in Chapter @ref(pred-mod). As a general rule, it is best not to add polynomial terms beyond the second or third orders to protect against overfitting the model.

Comparing the Adjusted R^2 for models with higher-order terms to one with only linear terms will help in determining whether higher-order polynomials add value to the model in explaining incremental variance in the response. Evaluating whether the coefficients on higher-order polynomials are statistically significant is important in determining *which variables* are contributing to any observed increases in Adjusted R^2 .

Let's demonstrate how to fit a regression model with polynomial terms in R using the `turnover_trends` dataset. First, we will subset this data frame to level 4 People Scientists who work remotely, based on the notion that turnover varies by `level` and `remote`, and then visualize the turnover trend to understand month-over-month variation across years.

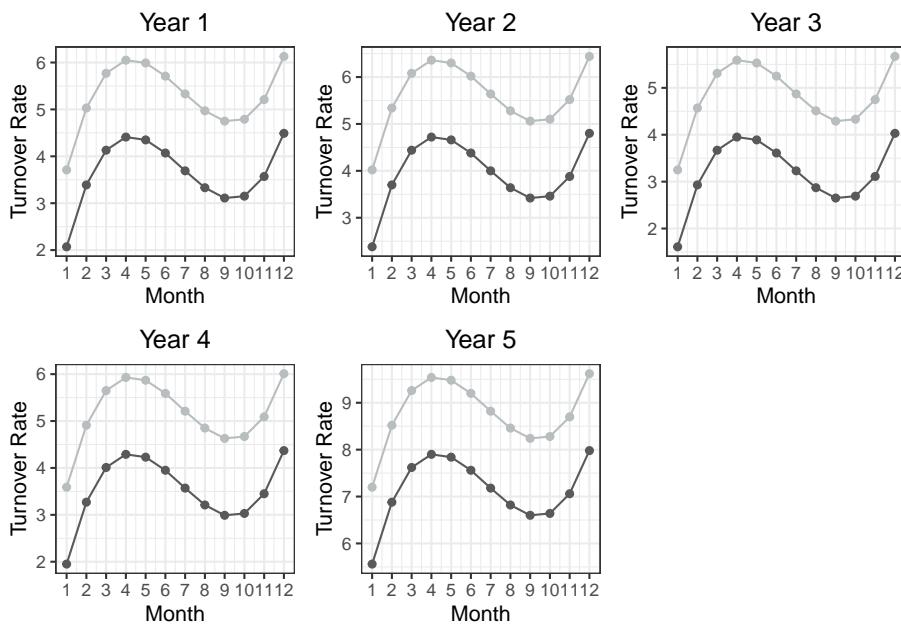


Figure 63: Year 1-5 turnover trends for level 4 People Scientists, stratified by remote (dark grey line) vs. non-remote (light grey line)

As we can see in Figure @ref(fig:ps-turnover-trends), the relationship between month and turnover rate is non-linear, and level 4 People Scientists who work remotely leave at lower rates relative to those who do not work remotely. There is a clear seasonal pattern that is consistent across all five years as well as remote vs. non-remote groups; namely, there is a spike in turnover between March and June as well as later in the year (November/December). Fitting a model to these data will require non-linear terms.

Adding polynomial terms requires an indicator variable `I()` in which the value of x is raised to the desired order (e.g., $x^2 = I(x^2)$). Let's start by fitting linear,

quadratic, and cubic regression models (to compare performance) using only `month` as a predictor. Notice that the shape of the trends resemble the cubic function shown in Figure @ref(fig:poly-fun) in that there are two discernible inflection points at which the trend reverses directions.

```
# Fit linear, quadratic, and cubic models to ps_turnover data
ps.lin.fit <- lm(turnover_rate ~ month, data = ps_turnover)
ps.quad.fit <- lm(turnover_rate ~ month + I(month^2), data =
  ~ ps_turnover)
ps.cube.fit <- lm(turnover_rate ~ month + I(month^2) +
  ~ I(month^3), data = ps_turnover)

##
## Call:
## lm(formula = turnover_rate ~ month, data = ps_turnover)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -3.2807 -1.3007 -0.3407  0.9218  4.5293
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.85067   0.35047 13.84   <2e-16 ***
## month       0.04000   0.04762   0.84    0.403
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.801 on 118 degrees of freedom
## Multiple R-squared:  0.005944, Adjusted R-squared:  -0.00248
## F-statistic: 0.7056 on 1 and 118 DF, p-value: 0.4026

##
## Call:
## lm(formula = turnover_rate ~ month + I(month^2), data = ps_turnover)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -2.9140 -1.2790 -0.3990  0.9535  4.6560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.24400   0.58692  7.231 5.35e-11 ***
## month       0.30000   0.20758   1.445   0.151
## I(month^2) -0.02000   0.01554 -1.287   0.201
## ---
```

```

## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.796 on 117 degrees of freedom
## Multiple R-squared: 0.01981, Adjusted R-squared: 0.003058
## F-statistic: 1.182 on 2 and 117 DF, p-value: 0.3101

##
## Call:
## lm(formula = turnover_rate ~ month + I(month^2) + I(month^3),
##      data = ps_turnover)
##
## Residuals:
##    Min      1Q Median      3Q     Max
## -1.924 -1.464 -0.114  0.486  3.666
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.514000  0.873921  1.732   0.0859 .
## month       2.410000  0.558831  4.313 3.41e-05 ***
## I(month^2) -0.410000  0.097879 -4.189 5.49e-05 ***
## I(month^3)  0.020000  0.004963  4.030  0.0001 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.689 on 116 degrees of freedom
## Multiple R-squared: 0.1402, Adjusted R-squared: 0.1179
## F-statistic: 6.304 on 3 and 116 DF, p-value: 0.0005334

```

The linear ($F(1, 118) = .71, p = .40$) and quadratic ($F(2, 117) = 1.18, p = .31$) models are not significant. However, as expected based on the shape of the turnover trend, the cubic model is significant ($F(3, 116) = 6.30, p < .001$) and the linear (`month`), quadratic (`I(month^2)`), and cubic (`I(month^3)`) terms all provide significant information in estimating turnover rates ($p < .001$).

While the cubic model achieved statistical significance at the $p < .001$ level, 86% of the variance in monthly turnover rates remains unexplained ($1 - R^2 = .86$). To improve the performance of the model, our model needs to reflect the fact that turnover varies as a function of `year` and `remote` in addition to `month`.

As shown in Figure @ref(fig:turnover-pred), the multidimensional data vary widely around estimates produced by the two-dimensional models (i.e., `turnover_rate` predicted on the basis of `month`). While the cubic regression model reflects the seasonality in month-over-month turnover, there are notable differences between remote and non-remote turnover rates as well as differences across years.

Let's add `remote` to the cubic regression model to see how performance changes.

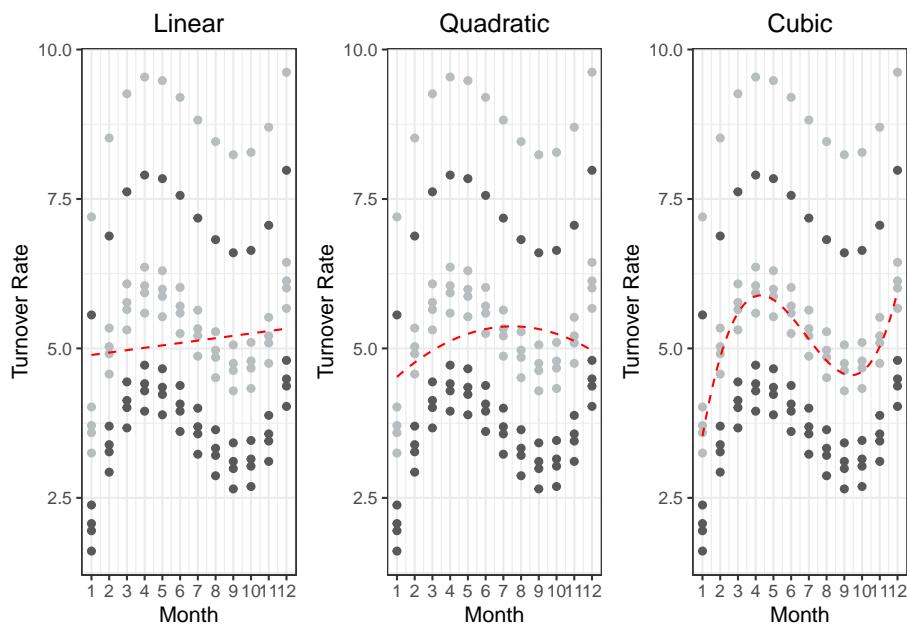


Figure 64: Linear, quadratic, and cubic models fitted to turnover data (red dashed lines). Remote workers are represented in dark grey points, and non-remote workers in light grey points.

```

# Fit linear, quadratic, and cubic models to ps_turnover df
ps.cube.fit <- lm(turnover_rate ~ month + I(month^2) + I(month^3)
  ↪ + remote, data = ps_turnover)

# Produce model summary
summary(ps.cube.fit)

## 
## Call:
## lm(formula = turnover_rate ~ month + I(month^2) + I(month^3) +
##     remote, data = ps_turnover)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -1.104 -0.764 -0.644 -0.334  2.846
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.334000  0.775051  3.011  0.0032 ***
## month       2.410000  0.488069  4.938 2.70e-06 ***
## I(month^2)  -0.410000  0.085485 -4.796 4.89e-06 ***
## I(month^3)  0.020000  0.004335  4.614 1.03e-05 ***
## remoteYes  -1.640000  0.269344 -6.089 1.54e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.475 on 115 degrees of freedom
## Multiple R-squared:  0.3498, Adjusted R-squared:  0.3272
## F-statistic: 15.47 on 4 and 115 DF,  p-value: 3.758e-10

```

As shown in Figure @ref(fig:ps-cube-mnthem-output), accounting for remote status increases explained variance by 21% ($\Delta R^2 = .35 - .14$). In addition to the increase in explained variance, the coefficient on `remote` is statistically significant ($\beta = -1.64$, $t(115) = -6.09$, $p < .001$). On average, the turnover rate for remote People Scientists is 1.64% lower than the turnover rate for non-remote People Scientists.

Next, let's include `year` as a linear term in the model since turnover rates also vary along this dimension.

```

# Fit linear, quadratic, and cubic models to ps_turnover df
ps.cube.fit <- lm(turnover_rate ~ year + month + I(month^2) +
  ↪ I(month^3) + remote, data = ps_turnover)

```

```
# Produce model summary
summary(ps.cube.fit)

##
## Call:
## lm(formula = turnover_rate ~ year + month + I(month^2) + I(month^3) +
##     remote, data = ps_turnover)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -1.419 -1.104  0.321  0.666  1.536
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.36900   0.63650  0.580   0.563
## year        0.65500   0.07338  8.926 8.71e-15 ***
## month       2.41000   0.37609  6.408 3.43e-09 ***
## I(month^2) -0.41000   0.06587 -6.224 8.28e-09 ***
## I(month^3)  0.02000   0.00334  5.988 2.53e-08 ***
## remoteYes -1.64000   0.20755 -7.902 1.90e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 114 degrees of freedom
## Multiple R-squared:  0.6173, Adjusted R-squared:  0.6005
## F-statistic: 36.77 on 5 and 114 DF,  p-value: < 2.2e-16
```

Explained variance increases to 62% by adding `year` to the model. While the coefficient on `year` is statistically significant ($\beta = .66$, $t(114) = 8.93$, $p < .001$), the change in attrition by year is not linear. Visualizing the distribution of turnover rates by year will provide evidence that a linear year-over-year growth factor will result in some large residuals since it will not capture the more complex trend present in these data.

Given the cubic nature of the change in turnover year-over-year, let's add quadratic and cubic terms for `year` to examine changes in model performance:

```
##
## Call:
## lm(formula = turnover_rate ~ year + I(year^2) + I(year^3) + month +
##     I(month^2) + I(month^3) + remote, data = ps_turnover)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -0.0025714 -0.0004286 -0.0004286  0.0017143  0.0017143
```

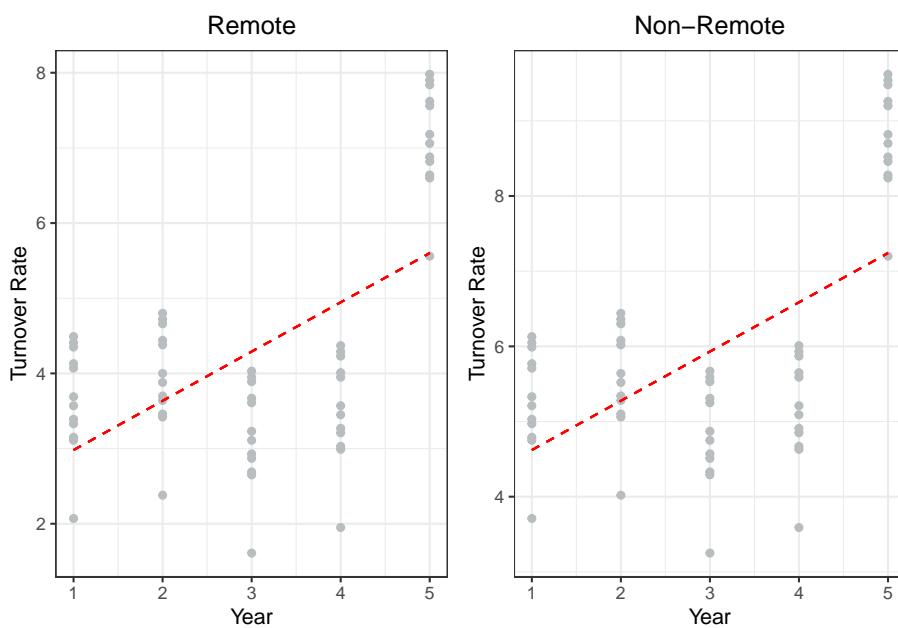


Figure 65: Turnover rate distribution by year for remote (left) and non-remote (right) groups. Red dashed line reflects linear relationship between year and turnover rate, with y -intercept lowered 1.64 percent for remote group.

```

## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.866e+00 1.875e-03 -995.2 <2e-16 ***
## year        5.906e+00 2.179e-03  2710.7 <2e-16 ***
## I(year^2)   -2.712e+00 8.087e-04 -3353.4 <2e-16 ***
## I(year^3)    3.625e-01 8.929e-05  4060.0 <2e-16 ***
## month       2.410e+00 5.491e-04  4388.7 <2e-16 ***
## I(month^2)  -4.100e-01 9.618e-05 -4262.8 <2e-16 ***
## I(month^3)   2.000e-02 4.877e-06  4100.8 <2e-16 ***
## remoteYes   -1.640e+00 3.030e-04 -5411.7 <2e-16 ***
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.00166 on 112 degrees of freedom
## Multiple R-squared:      1, Adjusted R-squared:      1
## F-statistic: 1.996e+07 on 7 and 112 DF,  p-value: < 2.2e-16

```

The inclusion of higher-order polynomials on `year` result in a perfect fit to these data ($R^2 = 1$). Albeit a statistical improbability in practice, this indicates that the slope of the relationship between `month` and `turnover_rate` is perfectly consistent across years within remote and non-remote groups.

Our resulting equation for estimating `turnover_rate` on the basis of a combination of linear and non-linear values of `year`, `month`, and `remote` is defined by:

$$\hat{y} = -1.87 + 5.91\text{year} - 2.71\text{year}^2 + .36\text{year}^3 + 2.41\text{month} - .41\text{month}^2 + .02\text{month}^3 - 1.64\text{remote} + \epsilon$$

The performance of this model may initially seem like a cause for celebration, but the probability is low that this model would estimate future turnover with such a high degree of accuracy. While these data were generated with a goal to simplify illustrations and facilitate a working knowledge of polynomial regression mechanics, data which conform to such a constant pattern of seasonality across multiple years is a highly improbable situation in practice. As stated earlier in this chapter, a model that results in a perfect fit is likely too flexible to generalize well to other data, and methods of evaluating how well models are likely to perform on future data will be covered in Chapter @ref(pred-mod).

Review Questions

1. What are some people analytics applications for comparing output from several regression models?

2. What modeling technique is appropriate for understanding an independent variable's contribution to a model's R^2 beyond a set of control variables?
3. In the context of Hierarchical Regression, what is the indicator that ΔR^2 is statistically significant when evaluating whether a particular independent variable provides meaningful information beyond a set of controls?
4. What are some examples of hypotheses that would warrant a linear mixed effects model over a general linear model?
5. What are the differences between Hierarchical Linear Modeling (HLM), which is also referred to as multilevel or mixed effects modeling, and Hierarchical Regression?
6. In what ways does polynomial regression differ from linear regression?
7. Why is it important to evaluate the nature of relationships at various levels of a categorical or time variable?
8. What shape characterizes a quadratic function?
9. If the coefficient on the cubic term is not statistically significant ($p >= .05$) in a cubic regression model, but the linear and quadratic terms are statistically significant ($p < .05$), what does this indicate about the model's fit to the data?
10. Why might adding higher-order polynomial terms to a model be problematic, even though the additional terms increase the model's R^2 ?

Logistic Regression

Logistic regression is a type of **generalized linear model**, which is a family of models for which key linear assumptions are relaxed. Logistic regression is an excellent tool for modeling relationships with outcomes that are not measured on a continuous scale (a key requirement for linear regression). Logistic regression is often leveraged to model the probability of observations belonging to different classes of a categorical outcome, and this type of modeling is known as **classification**. The context for classification can be binomial for two classes (e.g., active/inactive, promoted/not promoted), multinomial for multiple unordered classes (e.g., job family, location), or ordinal for multiple ordered classes (e.g., survey items measured on a Likert scale, performance level, education level). Regardless of the outcome variable's classes, logistic regression is in fact a type of *regression* analysis, which by definition returns a numeric outcome – and probabilities are numeric! Logistic regression accomplishes this by using a link function to generalize the linear model for non-continuous outcomes.

You may be wondering why linear regression cannot be implemented when the categorical outcome is dummy coded as outlined in Chapter @ref(data-prep). In a binary case, in which the categorical response has been coded as 1/0, least squares regression would produce an estimate for $\hat{\beta}X$ that represents the estimated probability of the outcome coded as 1 given X . For example, if attrition is the binary outcome and $Y = 1$ for employees who left and $Y = 0$ for employees who stayed, $\hat{Y} > .5$ could lend to a termination prediction assuming this is an appropriate probability threshold. Linear regression may produce estimates lower than 0 and higher than 1, however, which complicates the interpretation of estimates as probabilities.

This issue is not limited to binary categorical outcomes. Response variables with more than 2 categories cannot naturally be converted into quantitative values that are appropriate for linear regression. Instead of modeling the response directly as in linear regression, logistic regression models the probability of an outcome's class given values for one or more predictors. For example, we can leverage our `employees` dataset to model the probability of `active` given a value for `interview_rating`, which would be written as $Pr(\text{active} = \text{Yes} | \text{interview_rating})$ or simply, $p(\text{interview_rating})$. A probability of `active` = Yes will be estimated for a given value of `interview_rating`, and the prob-

ability threshold for determining the predicted class needs to be defined based on the business context. If we want to minimize false positives (i.e., incorrectly flagging at-risk employees who do not actually leave), we may set the threshold to something north of .5 (e.g., .7) to gain more confidence that those classified into the termination class are highly likely to exit.

Binomial Logistic Regression

Since estimating a binary outcome using linear regression can result in $p(X) < 0$ for some values of X and $p(X) > 1$ for others, we need a function that constrains the output to a $[0,1]$ interval. For logistic regression, the *logistic* function is used. This function converts the linear model, $p(X) = \beta_0 + \beta_1 X$, to the following form:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Irrespective of the value of X , the logistic function will always produce a sigmoidal (*S-shaped*) curve.

Taking the ratio of $\frac{p(X)}{1-p(X)}$ will give the odds of the outcome, which ranges between 0 (very low) and ∞ (very high). The logarithm of this ratio, $\log(\frac{p(X)}{1-p(X)})$, is known as the *log odds* or *logit*, and is fundamental to logistic regression. Log odds is a *monotonic transformation*, meaning the greater the odds, the greater the log of odds (and vice versa).

Recall that in linear regression, the coefficient β on a predictor is interpreted as the average change in Y for a one-unit increase in the respective predictor's value. In logistic regression, the interpretation is similar but rather than β representing the average change in Y , it represents the average unit change in the *log of the odds* for a one-unit increase in the predictor's value.

In R, the `glm()` function is used in conjunction with the `family = binomial` argument to fit a logistic regression model. As we covered in Chapter @ref(inf-stats), discrete probability distributions can be leveraged to model different types of nominal variables, and the binomial distribution is appropriate for a sequence of independent observations with only two outcomes – such as our `active` variable featuring only *yes* and *no* values. Therefore, we need to pass the `family = binomial` argument into the `glm()` function. The formula passed into the function is structured consistent with the `lm()` function used for linear regression: `glm(y ~ x, data)`.

```
##  
## Call:  
## glm(formula = active ~ interview_rating, family = "binomial",
```

```

##      data = employees)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.03045  0.00000  0.00002  0.00531  2.06562
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -74.486     10.333  -7.208 5.67e-13 ***
## interview_rating     21.963      2.997   7.329 2.32e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1298.58 on 1469 degrees of freedom
## Residual deviance: 103.27 on 1468 degrees of freedom
## AIC: 107.27
##
## Number of Fisher Scoring iterations: 11

```

The output shown in Figure @ref(fig:glm-rating) has some differences relative to the output of a linear regression model.

- **Estimate:** Average change in the log of odds for each one-unit increase in the value of the predictor
- **z value:** Ratio of Estimate / Standard Error. Assuming $\alpha = .05$, a $|z\text{-value}| \geq 2$ (2s of the mean) is a good rule of thumb for achieving statistical significance per the properties of the normal distribution.
- **Null deviance:** Measure of how well the response can be predicted by a model with only an intercept term; the lower the number, the better the fit.
- **Residual deviance:** Measure of how well the response can be predicted by a model with p predictors; the lower the number, the better the fit. The larger the delta between residual and null deviance, the better the model with p predictors relative to the intercept-only model.

Given the positive coefficient on `interview_rating`, we can interpret this to mean that for each one-unit increase in the average interviewer rating during the onsite stage of the employee's recruiting process, the log of the odds of the employee staying with the organization (`active` status of `Yes` coded as 1) increases by 21.96.

To illustrate why the logistic function is necessary, let's demonstrate differences in applying linear and logistic regression models by regressing a binary outcome `active` onto `interview_rating`.

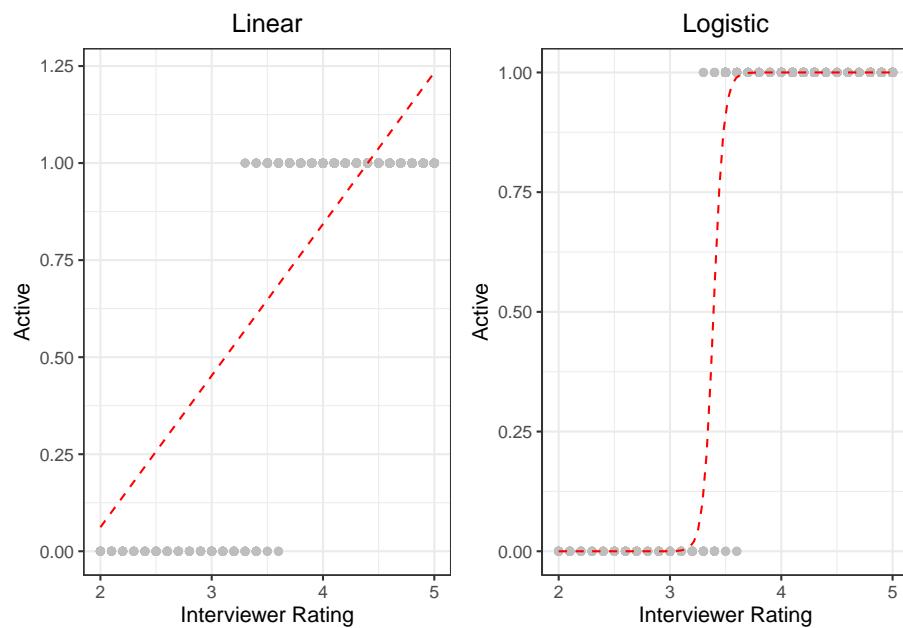


Figure 66: Linear (left) and logistic (right) functions applied to models regressing active status (1/0) onto median interviewer rating

Figure @ref(fig:lm-glm-compare) illustrates that for high values of `interview_rating`, a linear model would estimate probabilities for `active` that are greater than 1. Since probabilities range from 0 (impossible) to 1 (certain), anything outside the [0,1] interval does not make sense. On the other hand, the logistic function produces the *S-shaped* curve described previously. Using the logistic function, the probabilities are constrained to the [0,1] interval, and the visual reflects the fact that `active` can be perfectly predicted for low and high values of `interview_rating` but it is a mixed bag for values in the middle of the range (3.3 - 3.6).

It is often helpful to explain the relationship between a predictor and binary outcome in terms of a percentage increase or decrease. When $\beta = 1$, this indicates that the likelihood of the outcome is identical between the two groups of the predictor. If we exponentiate the coefficients, we can convert the log odds into odds ratios to facilitate a more intuitive interpretation. Therefore, $(\exp(\beta) - 1) * 100$ will provide the percentage increase or decrease in the odds of the *included* group relative to the *omitted* group.

For example, Figure @ref(fig:glm-ot-lvl) provides the log odds for `active` regressed onto two binary predictors: `overtime` and `job_lvl2plus`.

```
# Create dummy-coded variable for job level 2+
employees$job_lvl2plus <- ifelse(employees$job_lvl > 1, 1, 0)

# Fit a logistic regression model
glm.fit <- glm(active ~ overtime + job_lvl2plus, data =
  employees, family = 'binomial')

# Produce model summary
summary(glm.fit)

## 
## Call:
## glm(formula = active ~ overtime + job_lvl2plus, family = "binomial",
##      data = employees)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.3688    0.3532    0.3532    0.6300    1.1270
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.5163    0.1192 12.718 < 2e-16 ***
## overtimeYes -1.3965    0.1522 -9.176 < 2e-16 ***
## job_lvl2plus  1.2270    0.1523  8.055 7.93e-16 ***
## ---
```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1298.6  on 1469  degrees of freedom
## Residual deviance: 1149.8  on 1467  degrees of freedom
## AIC: 1155.8
##
## Number of Fisher Scoring iterations: 5

```

We can convert these coefficients into odds ratios by exponentiating the coefficients:

```

# Return exponentiated coefficients
exp(coef(glm.fit))

```

```

## (Intercept) overtimeYes job_lvl2plus
##      4.5551147    0.2474668    3.4108303

```

The exponentiated coefficient on `overtime` is $\exp(\beta) = .25$, so there is a $(1 - .25) * 100 = 75$ percent *decrease* in the odds of being active for employees who work overtime (since `overtime = Yes` is the included group) relative to those who do not work overtime. The exponentiated coefficient on `job_lvl2plus` is $\exp(\beta) = 3.41$, so there is a $(3.41 - 1) * 100 = 241$ percent *increase* in the odds of being active for those with a job level of 2 or greater relative to those with a job level of 1 (i.e., attrition is a larger concern for level 1 employees).

Generalized linear mixed models can be fitted using the `glmer()` function from the `lme4` library. The syntax is identical to the illustration of `lmer()` for multi-level linear models in Chapter @ref(lm) with the exception of needing to define `family` as an additional argument.

A juxtaposition of `glm()` and `glmer()` fits for logistic regression is shown in the following block of code. The mixed model example features an additional random (group-level) effect on `business_travel` via `1 | business_travel`, and fixed (observation-level) effects on remaining predictors which are consistent with the standard logistic regression model:

```

# Load library
library(lme4)

# Logistic model
glm(active ~ overtime + job_lvl2plus, data = employees, family =
  'binomial')

```

```

## Call: glm(formula = active ~ overtime + job_lvl2plus, family = "binomial",
##           data = employees)
##
## Coefficients:
## (Intercept)  overtimeYes  job_lvl2plus
##           1.516        -1.396         1.227
##
## Degrees of Freedom: 1469 Total (i.e. Null); 1467 Residual
## Null Deviance: 1299
## Residual Deviance: 1150 AIC: 1156

# Logistic mixed model
lme4::glmer(active ~ overtime + job_lvl2plus + (1 |
  ~ business_travel), data = employees, family = 'binomial')

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: active ~ overtime + job_lvl2plus + (1 | business_travel)
## Data: employees
##      AIC      BIC      logLik deviance df.resid
## 1148.0596 1169.2317 -570.0298 1140.0596     1466
## Random effects:
## Groups      Name      Std.Dev.
## business_travel (Intercept) 0.4351
## Number of obs: 1470, groups: business_travel, 3
## Fixed Effects:
## (Intercept)  overtimeYes  job_lvl2plus
##           1.546        -1.382         1.223

```

Multinomial Logistic Regression

Multinomial logistic regression is used to estimate the probability of an unordered categorical response with $K > 2$ classes. With an understanding of binomial logistic regression, extending the binomial model to a multinomial logistic regression model should be relatively intuitive.

To extend the binomial model to a multinomial context, we need to first identify a reference level. This decision may be arbitrary or guided by the research question or hypothesis, but the decision is nonetheless important as it impacts how the model coefficients are interpreted – always relative to the reference level. With the reference level defined, we can then express the multinomial logistic regression model as:

$$Pr(Y = k|X = x) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{1 + \sum_{j=1}^{K-1} e^{\beta_{j0} + \beta_{j1}x_1 + \dots + \beta_{jp}x_p}},$$

where K is the reference class, j is a $K - 1$ non-reference level, and k is the specified class for which the probability is being estimated on the basis of values for one or more X predictors.

Consider `dept` from our `employees` data, which has values of `Research & Development`, `Sales`, and `Human Resources`. This is a nominal variable because differences in these levels are not ordered in the same way job levels ranging from 1 to 10 or Likert scales ranging from 1 to 5 are. Employees in the Sales department may be greater in number relative to employees in the Human Resources department, for example, but it would not be appropriate to assign to the Sales department a numeric value that indicates it is *higher* or *better* relative to the Human Resources department.

Multinomial models are essentially a collection of binomial models which compare the log-odds of each non-reference category to the specified reference category. If the Human Resources department is identified as the reference category K , then β_{k0} for $k = Sales$ can be interpreted as the log odds of Sales department membership relative to Human Resources department membership in the following equation:

$$\log\left(\frac{Pr(Y = k|X = x)}{Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{xp}x_p$$

Depending on the research objective, it may be appropriate to compute the odds of one category relative to all other categories. This can actually be accomplished using binomial regression if the category of interest is coded as 1 and all other categories are coded as 0. In this case, the reference for the binomial model is a *collection* of $K - 1$ categories. If understanding the odds of *each category* relative to a reference category is more appropriate based on the research objective, multinomial logistic regression is the proper model.

Let's illustrate how to implement multinomial logistic regression by determining how variables in the `employees` dataset help in classifying employees into departments. To build this model in R, we will use the `multinom` function from the `nnet` package. It is important that the nominal response variable is defined as a factor before implementing multinomial logistic regression, so we will first convert the data type of `dept` from its native character type to a factor. We also need to identify the reference department against which the probability of each of the other departments will be evaluated; we will define this using the `ref` argument within the `relevel()` function:

```

# Load library
library(nnet)

# Convert dept to factor
employees$dept <- factor(employees$dept)

# Specify reference level
employees$dept <- relevel(employees$dept, ref = "Human
  ↪ Resources")

# Fit multinomial logistic regression model
# An omitted group for categorical variables is defined by
  ↪ default
multinom.fit <- nnet::multinom(dept ~ overtime + ed_field, data =
  ↪ employees)

## # weights: 24 (14 variable)
## initial value 1614.960064
## iter 10 value 887.448722
## iter 20 value 833.883230
## iter 30 value 833.513790
## final value 833.513639
## converged

# Summarize results from model object
summary(multinom.fit)

## Call:
## nnet::multinom(formula = dept ~ overtime + ed_field, data = employees)
##
## Coefficients:
##                               (Intercept) overtimeYes ed_fieldLife Sciences
## Research & Development -18.92857   0.1674815      22.19842
## Sales                  -19.40948   0.1602262      21.60505
##                               ed_fieldMarketing ed_fieldMedical ed_fieldOther
## Research & Development    25.72879     22.21217     21.93577
## Sales                   37.42515     21.27842     20.96823
##                               ed_fieldTechnical Degree
## Research & Development          22.0455
## Sales                      21.5113
##
## Std. Errors:
##                               (Intercept) overtimeYes ed_fieldLife Sciences
## Research & Development    1.808496   0.3917603      1.817514

```

```

## Sales          1.810390  0.4071887      1.819982
##             ed_fieldMarketing ed_fieldMedical ed_fieldOther
## Research & Development 8.991934   1.820007   1.864206
## Sales          8.996707   1.823207   1.874536
##             ed_fieldTechnical Degree
## Research & Development           1.84963
## Sales          1.85451
##
## Residual Deviance: 1667.027
## AIC: 1695.027

```

Notice the output from the `multinom()` function is quite limited relative to `glm()` and `lm()`. Coefficients and standard errors are provided, but *p*-values are not available in the output so we will need to calculate them separately.

A new statistical measure is included in the output of this model, **Akaike Information Criterion (AIC)**, which is a score that is helpful for model selection. AIC is calculated by:

$$AIC = -2\frac{\ell}{n} + 2\frac{k}{n},$$

where n is the number of observations, k is the number of parameters (predictors + intercept), and ℓ is the log likelihood function where:

$$\ell = -\frac{n}{2}(1 + \ln(2\pi) + \ln(\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2))$$

Just as we compared R^2 across linear regression models in Chapter @ref(lm), AIC can be compared across models to determine which one is a better fit to the data; lower AIC values indicate better fit. Consistent with how the Adjusted R^2 statistic adjusts for variables that do not provide information, AIC penalizes models that use more parameters. Therefore, if two models explain the same amount of variance in the response, the model with fewer parameters will achieve a lower AIC score.

To determine whether the coefficients are statistically significant, we need to perform an additional step to compute *p*-values using *z* scores:

```

# Calculate z-scores
z_scores <- summary(multinom.fit)$coefficients /
  ~ summary(multinom.fit)$standard.errors

# Produce p-values
p_values <- (1 - pnorm(abs(z_scores))) * 2

```

```
# Transpose and display rounded p-values
data.frame(t(round(p_values, 3)))
```

```
##                                     Research...Development Sales
## (Intercept)                               0.000 0.000
## overtimeYes                                0.669 0.694
## ed_fieldLife Sciences                      0.000 0.000
## ed_fieldMarketing                         0.004 0.000
## ed_fieldMedical                           0.000 0.000
## ed_fieldOther                             0.000 0.000
## ed_fieldTechnical Degree                  0.000 0.000
```

These *p*-values indicate that those who work overtime are not significantly more likely to work in either the Research & Development or Sales departments relative to the Human Resources department. In other words, a considerable portion of employees in all three departments work overtime, so this variable is not helpful in classifying employees into their correct departments. This is evident in Figure @ref(fig:ot-dept).

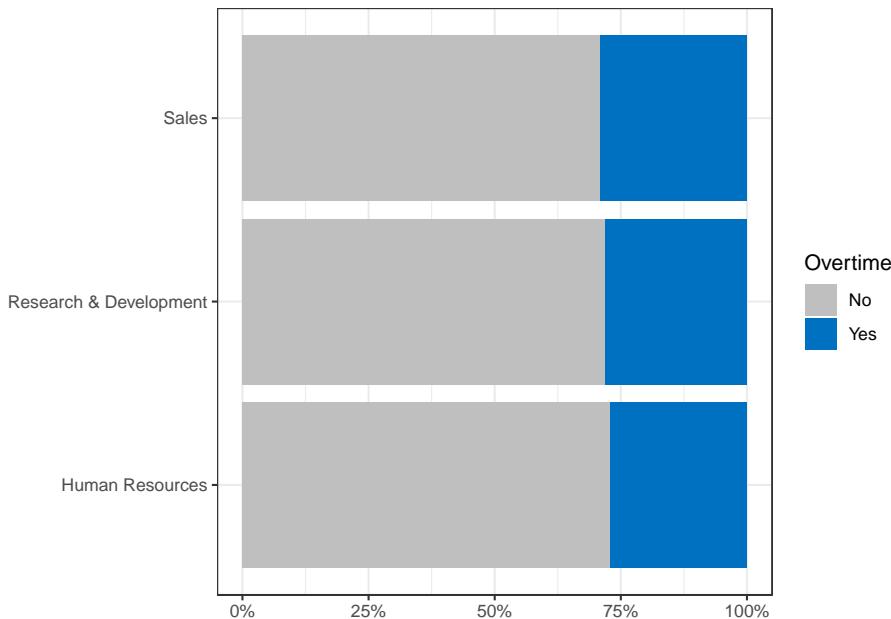


Figure 67: Overtime by department

The *p*-values indicate that educational background is a strong predictor of department. This is evidenced in Figure @ref(fig:ed-field-dept), which shows that

each educational field is generally dominated by a single department. This means that we can achieve strong departmental *purity* on the basis of `ed_field` alone. All employees who studied Marketing work in Sales and all employees who studied HR work in the HR department. However, those who work in R&D have a variety of educational backgrounds, so the signal is not as clear for these cases and additional variables would be needed to more accurately assign these employees to the correct departments.

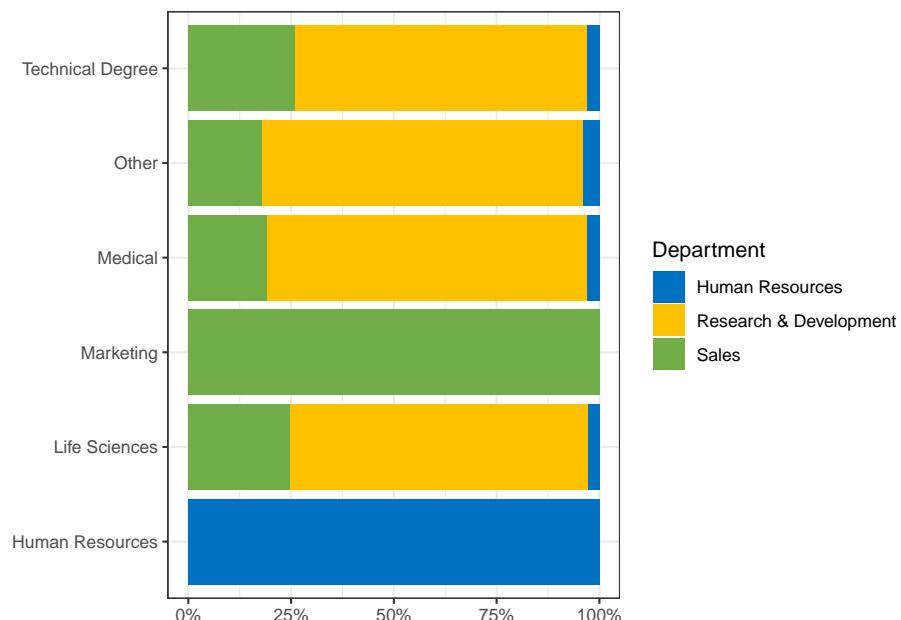


Figure 68: Department distribution by educational field

As we did for binomial logistic regression, we can compute the exponential of model coefficients for the multinomial logistic regression model to convert the log-odds to more intuitive odds ratios:

```
# Return exponentiated coefficients from model object
# Transpose rows to columns for improved readability
data.frame(t(exp(coef(multinom.fit))))
```

	Research...	Development	Sales
## (Intercept)	6.017654e-09	3.720234e-09	
## overtimeYes	1.182323e+00	1.173776e+00	
## ed_fieldLife Sciences	4.371726e+09	2.415207e+09	
## ed_fieldMarketing	1.492349e+11	1.792819e+16	
## ed_fieldMedical	4.432226e+09	1.742213e+09	

```
## ed_fieldOther           3.361886e+09 1.277572e+09
## ed_fieldTechnical Degree 3.751786e+09 2.199075e+09
```

Consistent with our approach for binomial logistic regression, we can interpret these exponentiated coefficients in terms of having greater or lesser odds. Importantly, in the multinomial context the odds are *relative to the reference category* – the Human Resources department in this case.

For example, the odds ratio associated with a Medical educational field is $\exp(\beta) = 4.43$ for Research & Development and $\exp(\beta) = 1.74$ for Sales. Therefore, those with a Medical education have $(4.43 - 1) * 100 = 343$ greater odds of being in the Research & Development department and $(1.74 - 1) * 100 = 74$ greater odds of being in the Sales department relative to the Human Resources department. We can ignore the odds ratios associated with `overtime` since this variable does not provide significant information.

Ordinal Logistic Regression

Many projects in people analytics involve understanding how variables influence ordinal outcomes, such as performance ratings or survey items measured on a Likert scale. Stepwise changes in the levels of ordinal outcomes may or may not be consistent. For example, it may be easy for one to be promoted from job level 1 to 2 but relatively difficult to progress from 5 to 6. Linear regression should not be used in these settings, as linear assumptions are designed for data measured on a continuous scale and will not hold for ordinal data. This section will cover **ordinal logistic regression**, which is a modeling technique designed for understanding how variables influence stepwise changes in a multi-class ordinal outcome.

Beyond the universal data screening procedures we have covered, such as ensuring problematic collinearity is not present, ordinal logistic regression features a unique **proportional odds assumption** that must be satisfied. This assumption, also known as the **parallel regression assumption**, requires that each independent variable has an equal effect at each level of the ordinal outcome. If the effect varies across levels of the outcome, separate models are needed to accurately reflect the associations with each pair of levels.

Though other approaches exist for modeling ordinal outcomes, the proportional-odds model based on cumulative distribution probabilities is the most common. Its intercepts are dependent on the j levels but slopes are equal as defined by:

$$\log\left(\frac{Pr(Y \leq j)}{Pr(Y > j)}\right) = \beta_{j0} - \sum_{i=1}^n \beta_{1i}x_{1i} + \dots + \beta_{pi}x_{pi}$$

The implementation of ordinal logistic regression will be demonstrated by evaluating the statistical drivers of engagement. First, we need to define `engagement` as an ordered factor:

```
# Define ordered factor
employees$engagement <- ordered(employees$engagement, levels =
  c(1, 2, 3, 4, 5))

# Verify structure of engagement variable
str(employees$engagement)

## Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<...: 3 2 2 3 3 3 4 3 2 3 ...
```

Next, the proportional odds assumption will be checked using the **Brant test**. The Brant test is a set of comparisons of the separate binary logistic models underlying the overall model (Brant, 1990). This test evaluates whether β_{j1} through β_{jp} are consistent across each of the j levels. This is done via a χ^2 test to compare coefficients and determine whether observed differences are larger departures from what we would expect by chance. The null hypothesis states that coefficients are not statistically different across the j levels; therefore, $p < .05$ indicates that significant differences in effects are present across the j levels and, thus, the proportional odds assumption is violated.

We can leverage the `brant` package in R for this, which is compatible with the `polr()` function from the `MASS` package that will be used to perform ordinal logistic regression. Since we will be evaluating model statistics, rather than merely using the model for prediction (the subject of Chapter @ref(pred-mod)), we need to specify the `Hess = TRUE` argument in the `polr()` function to include the *Hessian matrix* (observed information matrix) in the output.

```
# Load libraries
library(MASS)
library(brant)

# Fit a ordinal logistic regression model
ord.fit <- MASS:::polr(engagement ~ org_tenure, data = employees,
  Hess = TRUE)

# Test proportional odds assumption using the Brant test
brant::brant(ord.fit)

## -----
## Test for X2 df probability
## -----
```

```

## Omnibus      0.39    2   0.82
## org_tenure  0.39    2   0.82
## -----
## 
## H0: Parallel Regression Assumption holds

```

Notice the line in the output for the omnibus test, which shows an identical χ^2 , df , and p -value to the line associated with `org_tenure`. **Omnibus tests** are statistical tests which test for the significance of several parameters in a model at once. For example, a one-way ANOVA evaluating differences in mean commute time across three locations is an omnibus test since it has more than two parameters. As we covered in Chapter @ref(aod), the null hypothesis is rejected in the context of ANOVA if there is at least *one* difference in complex contrasts – even if the mean commute time is not significantly different between *all* groups. Test statistics are identical for this ordinal logistic regression model because there is a single predictor, but Brant's omnibus test investigates equality of coefficients for all predictors jointly in the case of more than two parameters (Martin, 2022). The null and alternative hypotheses for Brant's omnibus test are:

- H_0 : The odds are proportional for all predictors in the model.
- H_A : The odds are non-proportional for at least one predictor.

The results of Brant's test indicate that we fail to reject H_0 since $p = .82$ for the omnibus test, so the proportional odds assumption holds for these data. If the model featured more than one predictor, we could also evaluate the statistics on individual predictors – but only to determine for which predictor(s) the proportional odds assumption is violated *if* $p < .05$ for the omnibus test. As the number of variables and tests increases, so too does our risk of incorrectly rejecting the proportional odds assumption; therefore, decisions regarding the proportional odds assumption should not be based on statistics for individual predictors alone. Even if the odds are truly proportional for each predictor independently, with 20 predictors we would expect to find one by chance for which $p < .05$ since our tolerance for a Type I error is 1 in 20 with $\alpha = .05$.

Since the proportional odds assumption holds, let's review the model output:

```

# Summarize ordinal logistic regression model
summary(ord.fit)

## Call:
## MASS::polr(formula = engagement ~ org_tenure, data = employees,
##           Hess = TRUE)
##
## Coefficients:

```

```

##           Value Std. Error t value
## org_tenure -0.006965  0.008032 -0.8671
##
## Intercepts:
##           Value Std. Error t value
## 1|2      -2.8661   0.1271  -22.5423
## 2|3      -0.8419   0.0804  -10.4703
## 3|4      2.1711   0.1036   20.9479
## 4|5     4122.5567  0.1036 39776.9095
##
## Residual Deviance: 3084.592
## AIC: 3094.592

```

The output provides the average effect of a one-unit increase in `org_tenure` (Coefficients section) as well as intercepts on each pair of levels for `engagement` (Intercepts section). The effect of a one-unit increase in `org_tenure` in moving `engagement` from one ordinal level to the next is quite small ($\beta = -.007$). The intercepts are often referred to as cutpoints and can be roughly translated as thresholds. While the intercepts for each cutpoint vary, note that the single coefficient on `org_tenure` is only possible because our proportional odds assumption holds and the effect is consistent (proportional) across levels of our ordered factor, `engagement`.

Consistent with the default output from the `multinom()` function used for multinomial logistic regression, *p*-values are not provided in the standard output from the `polr()` function. However, we can calculate them for reasonably large samples by comparing the *t*-values against the standard normal distribution:

```

# Store coefficients to df
coef_df <- coef(summary(ord.fit))

# Produce p-values
p <- pnorm(abs(coef_df[, "t value"]), lower.tail = FALSE) * 2

# Combine p values with coefficients df
coef_df <- cbind(coef_df, "p value" = p)

# Display df contents
coef_df

##           Value Std. Error     t value     p value
## org_tenure -0.00696485 0.008032275 -0.867108 3.858828e-01
## 1|2        -2.86612408 0.127144280 -22.542297 1.598105e-112
## 2|3        -0.84185679 0.080404351 -10.470289 1.182814e-25
## 3|4        2.17107818 0.103641956   20.947870 1.962145e-97
## 4|5     4122.55669611 0.103641956 39776.909511 0.000000e+00

```

We can now estimate the likelihood of a particular observation having a specified level of Y , such as $Y \leq 3$, as follows:

$$\log\left(\frac{Pr(Y \leq 3)}{Pr(Y > 3)}\right) = 2.17 - .007x_{orgtenure}$$

Review Questions

1. Why can't linear regression be used when outcome variables are measured on a non-continuous scale?
2. What are some examples of hypotheses for which logistic regression would be an appropriate model?
3. Why is it helpful to calculate the exponential of log-odds in a logistic regression model?
4. What does an odds ratio of 1.25 indicate in a binomial context?
5. What does an odds ratio of 0.75 indicate in a multinomial context?
6. How does Akaike Information Criterion (AIC) compare to R^2 with respect to its purpose and function?
7. In what type of R object do ordinal data need to be stored in order to implement ordinal logistic regression?
8. Why can't linear regression be used to understand associations of predictors with an ordinal outcome?
9. What does the proportional odds (or parallel regression) assumption assume about model coefficients?
10. Why is it important to evaluate Brant's omnibus test – over the test statistics on independent predictors alone – when determining whether the proportional odds assumption for ordinal logistic regression holds?

Predictive Modeling

In people analytics, inferential models like those covered in Chapters @ref(lm), @ref(lme), and @ref(log) are generally warranted by the research objectives. However, there are times when we need to go beyond interpreting coefficients to understand relative influences of predictors on an outcome and leverage the models to estimate or *predict* the most likely future values. This type of modeling is often referred to as **predictive analytics** and is the subject of this chapter.

A branch of **Artificial Intelligence (AI)** known as **Machine Learning (ML)** is often associated with predictive modeling. ML is a set of methods which aim to improve performance on a set of tasks by learning from data (Mitchell, 1997). ML applications can be found in medical diagnostics, autonomous vehicles, speech recognition, automated securities trading, lending decisions, marketing, and many other domains. The difference between statistics and ML is largely philosophical. Logistic regression, for example, is covered in both statistics and ML textbooks. While statistics focuses more on modeling, and ML is more algorithmic, both can be used for prediction. The broader field of **data science** often further confounds distinctions between these disciplines, though data science represents the entire end-to-end process – from data extraction and engineering to modeling and analysis.

A good use case for a predictive model in people analytics is data restatement to adjust for reorganizations over time. In this case, accuracy may be more important than the explainability of the model. A predictive model may be used to predict and assign a current functional executive to historical records to support leader-wise trending analyses. For example, consider a scenario in which the current VP of Product Marketing was hired six months ago to replace the former VP of Product Marketing who was in the role for the prior five-year period. If the new VP wants to see monthly termination counts for their organization over the past three years, term records prior to the VP's start date need to be associated with the *current* – rather than *former* – VP of Product Marketing to accomplish this. A model can be trained to learn from patterns in the combinations of current workers' department, leader, and job attributes that can be used to assign current executives to past data (e.g., historical termination events, month-end worker snapshots).

It is important to note that while there are AI/ML applications for people analytics, feeding data to black box models without an understanding of how the underlying algorithms work is generally a bad idea. Despite the glamour predictive analytics has seen in recent years due to the allure of a magical elixir that can portend the future, more times than not inferential statistical approaches are more appropriate in a people analytics setting. Unfortunately, the hype has given rise to AI snake oil to justify a premium price point for modern HR tech solutions, which are often little more than the descriptive statistics covered in Chapter @ref(desc-stats).

It is both a blessing and a curse that a predictive model can be built with a single line of code in R, and it is dangerous to blindly make recommendations on the basis of the output. A simpler model that you can understand and explain is generally a better option than a more complex one that you cannot. There is a high probability that stakeholders will ask deeper questions about *why* a particular segment of the workforce is predicted to exit at higher rates, for example, and answering these questions requires a deeper understanding of the factors that led to them being classified as such.

People data are messy, and understanding why people vary in attitudes, perceptions, and behaviors is an inherently difficult endeavor. Spoiler alert: There is no crystal ball that will soften this reality.

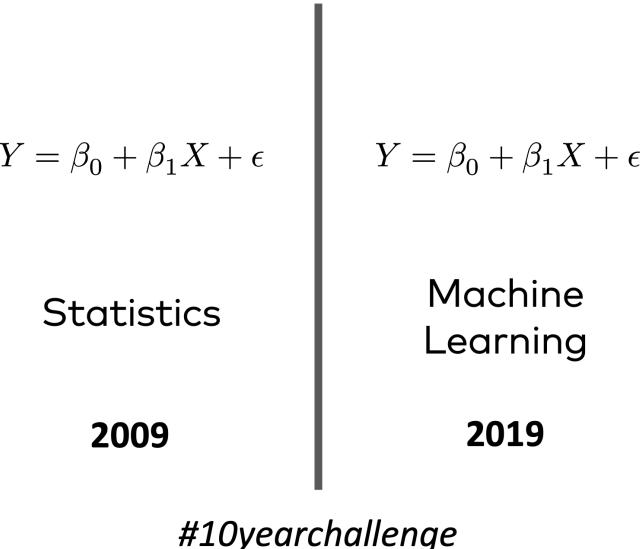


Figure 69: Satirical illustration based on the viral meme of 2019 which depicts the enduring popularity and utility of linear regression, even in light of the billions companies invest in ML each year.

Cross-Validation

Predictive modeling involves training a model on a data set referred to as a **training set** and using the model to make predictions for observations on a separate data set known as the **test set** or **validation set** to evaluate model performance.

A model is blind to data in the test set, since only the data in the training set are used to *train* the model. Therefore, the test set provides a convenient way to compare the actual known values to the predicted values and estimate how well the model will generalize to other data. Evaluating model performance on the basis of training data would be akin to having students take an exam after providing them the answers. Strong performance on the training data is almost a certainty, so the value of a model rests on its performance on data not used to build it.

This partitioning procedure is known as **cross-validation (CV)**. While there are many methods of splitting data into training and test sets, a common feature among all is that the partitioning strategy is random. Without randomization, the model may learn patterns characteristic of the training set that result in inaccurate predictions for other data which do not feature consistent patterning.

This section will explore a few of the most common CV methods.

Validation Set Approach

The **validation set approach** is the most basic form of CV. This approach involves defining proportions by which to partition data into training and test sets – usually 2/3 and 1/3, respectively. The model is trained on the training set and then differences between actual y and predicted \hat{y} values are calculated on the test set to evaluate model performance.

Leave-One-Out

Leave-one out CV fits a model using $n - 1$ observations n times. The test error is then evaluated by calculating differences between actual y and predicted \hat{y} values for all omitted observations.

k-Fold

k-fold CV randomly partitions observations into k groups, or *folds*, that are approximately equal in size. The first fold is treated as the test set, and the model is trained on the remaining $k - 1$ folds. This procedure is repeated k times, each with a different set of observations (*fold*) as the test set. The test error is then evaluated by calculating differences between actual y and predicted \hat{y} values across all test sets.

Model Performance

There are several methods of quantifying how well models perform on test data in order to assess the extent to which the model will generalize. Predictive modeling applications will be categorized as either *classification* or *forecasting* to reflect the families of use cases germane to people analytics.

Classification

In a classification setting, predictions are either right or wrong. Therefore, calculating the overall error rate across test data is straightforward:

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i),$$

where I is an indicator variable equal to 1 if $y_i \neq \hat{y}_i$ and 0 if $y_i = \hat{y}_i$.

A **confusion matrix** is often used in classification to parse the overall model accuracy rate into component parts and understand whether the model performs at a level appropriate to a defined tolerance level per the research objective. In an attrition project, it may be more important to correctly predict high performers who leave than to correctly predict those who stay, as prediction errors for the former are likely far more costly. Several performance metrics are provided by the confusion matrix to aid in a more granular understanding of model performance, which are represented in Figure @ref(fig:confusion-mtx):

- **True Positive:** Number of correct true predictions
- **True Negative:** Number of correct false predictions
- **False Positive:** Number of incorrect true predictions (type 1 error)
- **False Negative:** Number of incorrect false predictions (type 2 error)
- **Accuracy:** Rate of correct predictions overall
- **Sensitivity:** Rate of actual true cases predicted correctly (also known as **Recall**)
- **Specificity:** Rate of actual false cases predicted correctly
- **Precision:** Rate of correct predictions among all cases predicted true
- **Negative Predictive Value:** Rate of correct predictions among all cases predicted false

Forecasting

While predictions are either right or wrong in a classification context, evaluating model performance in a forecasting context involves assessing the *magnitude* of differences between actual y and predicted \hat{y} values – usually across time periods. There are many methods for assessing forecasting model performance, and we will focus on some of the most common.

		Predicted		
		TRUE	FALSE	
Actual	TRUE	True Positive (TP)	False Negative (FN) TYPE II ERROR	Sensitivity $TP / (TP + FN)$
	FALSE	False Positive (FP) TYPE I ERROR	True Negative (TN)	Specificity $TN / (TN + FP)$
		Precision $TP / (TP + FP)$	Negative Predictive Value $TN / (TN + FN)$	Accuracy $(TP + TN) / n$

Figure 70: Confusion matrix

- Mean absolute deviation (MAD): Average absolute difference between actual y and predicted \hat{y} values

$$MAD = \frac{\sum |y_i - \hat{y}_i|}{n}$$

- Mean square error (MSE): Average squared difference between actual y and predicted \hat{y} values

$$MSE = \frac{\sum (y_i - \hat{y}_i)^2}{n}$$

Squaring differences accomplishes two key objectives: (1) converts negative differences to positive (consistent with the MAD approach), and (2) imposes a greater penalty on larger differences, which causes error rates to increase at an exponential rather than linear rate (e.g., $2^2 = 4$, $3^2 = 9$, $4^2 = 16$). MSE is perhaps the most pervasive model performance measure in predictive modeling.

- Mean absolute percentage error (MAPE): Average absolute difference expressed as a percentage

$$MAPE = \left(\frac{100}{n} \right) \sum \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Bias-Variance Tradeoff

Bias-variance tradeoff refers to the important endeavor of minimizing two sources of error that prevent models from generalizing beyond their training data: *bias* and *variance*.

- **Bias:** Error from erroneous assumptions in the model. High bias results from models that are too simplistic to accurately capture the relationships between predictors and the outcome; this is known as **underfitting**.
- **Variance:** Error from sensitivity to small fluctuations in training data. High variance results from models that capture random noise rather than the significant patterns in the training data; this is known as **overfitting**.

As a general rule, the more flexible the model, the more variance and less bias. As shown in Figure @ref(fig:bias-var-tradeoff), minimizing test error by achieving a model with optimal fit to the data requires limiting both bias and variance.

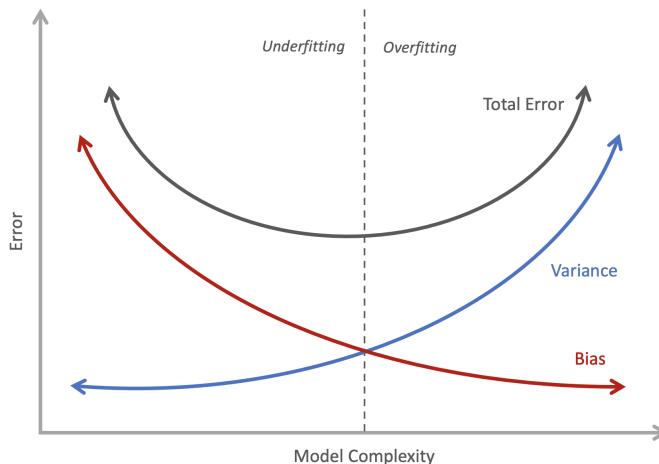


Figure 71: Bias-variance tradeoff. Dashed line represents optimal model performance.

Tree-Based Algorithms

While there are many flexible ML algorithms, such as **Extreme Gradient Boosting (XGBoost)**, **Artificial Neural Networks (ANN)**, and **Support Vector Machines (SVM)**, that tend to perform well across a range of prediction problems, these will not be covered as we will focus on more *interpretable* tree-based algorithms that have more applications to people analytics.

Decision Trees

In addition to the inferential models covered in previous chapters, **decision trees** are also excellent tools that lend to simple and effective narratives about factors influencing outcomes in either a regression or classification setting. As illustrated in Figure @ref(fig:decision-tree), decision trees resemble a tree that depicts a set of decisions as well as consequences of those decisions. The top-level **Department** variable is known as a root node, and the remaining **Tenure**, **Performance Rating**, and **Remote** nodes are known as interior nodes. Decisions represented in **Active** and **Inactive** boxes are referred to as leaf, terminal, or end nodes.

As evidenced by the inactive status prediction in the leaf node, this decision tree shows that employees in the Engineering department are unlikely to stick around for two or more years. In addition, employees in other departments terminate if they are low performers or if they are high performers who do not work remotely. It's important to note that in practice, it is rare to achieve complete purity in leaf nodes, as there is usually a mix of results in a given node – though a more frequent class or range of values is expected in the presence of meaningful variables. If leaf nodes for a classification problem are comprised of a single class, it may be evidence of overfitting, especially if the *n*-count is small.

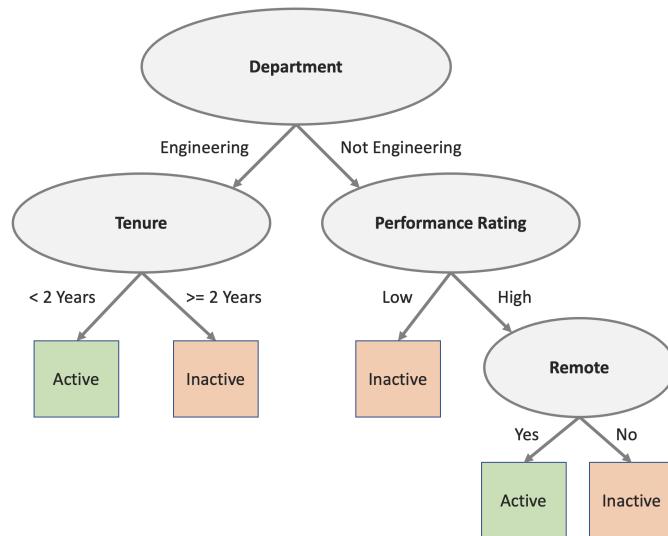


Figure 72: Conceptual decision tree for employee attrition prediction

Predictions based on a deep tree with an excessive number of partitions and few observations will likely be highly inaccurate beyond the training data. The goal of decision trees is to arrive at a set of decisions that best delineate one class or range of values from others by identifying patterns and natural cutpoints among

a reasonably large subset of the data at each level. There must be signal in the features used to partition the data such that predictions are an improvement over random guessing (e.g., 50/50 chance in a binary classification setting).

Random Forests

A **random forest (RF)** is a natural extension of the decision tree. As the name implies, a random forest is a large number (forest) of individual decision trees that operate as an ensemble. The process of fitting multiple models on different subsets of training data and then combining predictions across all models is referred to as **bagging**. This is a case of *wisdom of the crowd* decision-making in which a large number of uncorrelated trees (models) functioning as a committee should outperform individual trees.

To understand the mechanics of a random forest, consider an investment strategy in which you diversify an investment portfolio by spreading investments across different assets. By investing in assets that are uncorrelated, there is a lower likelihood that the portfolio's value will be negatively impacted by a negative event impacting a single holding. In the same way, a random forest constructs an ensemble of trees that are each based on different randomized subsets of data and combinations of features to amalgamate the information and arrive at more accurate predictions. The potential for poor performance from a single tree is mitigated by the many trees working in concert with one another.

Though random forests combine information from many decision trees, there are still intuitive ways of understanding which features are most important in segmenting employees to understand drivers of various outcomes.

Predictive Modeling

We will now integrate these concepts into attrition classification and forecasting examples. The high-level prediction workflow will follow four steps:

- **Step 1:** Partition data into training and test sets for cross-validation.
- **Step 2:** Build models using training data.
- **Step 3:** Use models to make predictions on test data.
- **Step 4:** Evaluate model performance.

Classification

To demonstrate the prediction workflow steps for classification, we will leverage our `employees` data set:

```
# Load library
library(peopleanalytics)

# Load data
data("employees")

# Load employee data
prediction_dat <- employees

# One-hot encode active outcome variable, setting inactives to 1
# and actives to 0
prediction_dat$active <- ifelse(prediction_dat$active == 'No', 1,
                                 0)
```

Step 1: Partition data into training and test sets for cross-validation.

For this example, we will implement the validation set approach for CV.

```
# Load library
library(dplyr)

# Set seed for reproducible training and test sets
set.seed(9876)

# Randomly select 2/3 of employees for the training set
training_ids <- sample(prediction_dat$employee_id, size =
                           nrow(prediction_dat) * 2/3, replace = FALSE)

# Create training data
training_dat <- prediction_dat |> dplyr::filter(employee_id %in%
                           training_ids)

# Create test data using remaining 1/3 of observations
test_dat <- prediction_dat |> dplyr::filter(!employee_id %in%
                           training_ids)

# Return Boolean to validate that all observations are accounted
# for
nrow(training_dat) + nrow(test_dat) == nrow(prediction_dat)

## [1] TRUE
```

Step 2: Build models using training data.

Machine learning (ML) algorithms are sensitive to imbalanced classes. That is, when there is not an equal representation of each class we wish to predict in the data (e.g., employees who left and employees who did not), it can adversely impact our results. In the case of our `employees` data set, there are 1,233 observations for active employees but only 237 observations for inactive employees. Therefore, we will introduce a popular technique to address this known as **Synthetic Minority Oversampling Technique (SMOTE)**, which will be important for the RF model we will train. This technique takes random samples with replacement (i.e., each observation may be chosen more than once) from the minority class to augment the class's representation in the dataset and achieve balanced classes.

While functions exist for implementing SMOTE, such as the `SMOTE()` function from the `DMwR` library, in the spirit of demystifying black box ML approaches we will step through this procedure without the use of an available function:

```
# Calculate class representation delta in training data
training_class_delta <- nrow(training_dat) |> dplyr::filter(active == 0)) - nrow(training_dat) |> dplyr::filter(active == 1))

# Copy training data to separate data frame for oversampling
training_dat_os <- training_dat

# Set seed for reproducible results
set.seed(9876)

# Oversample the underrepresented inactive class by
#   training_class_delta to align observation counts with active
#   class
# Note: A loop is not the most efficient -- especially with large
#   data sets -- but it is leveraged here to simplify instruction
#   on SMOTE mechanics
for (i in 1:training_class_delta){

  # Sample employee id from underrepresented class
  oversampled_id <- sample(training_dat_os[training_dat_os$active == 1, 'employee_id'], size = 1, replace = TRUE)

  # Store observation for sampled employee id
  new_obs <- unique(training_dat_os |> dplyr::filter(employee_id == oversampled_id))

  # Append sampled observation to training data frame
  training_dat_os <- rbind(training_dat_os, new_obs)
}
```

```
# Return Boolean to validate that classes are equal in the
→ training data
nrow(training_dat_os |> dplyr::filter(active == 0)) ==
→ nrow(training_dat_os |> dplyr::filter(active == 1))
```

```
## [1] TRUE
```

Next, we will fit a binomial logistic regression model using a subset of predictors from the oversampled training data. For comparison, let's summarize a model using original and oversampled data.

```
##
## Call:
## glm(formula = active ~ overtime + job_lvl + engagement + interview_rating,
##      family = binomial, data = training_dat)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.78408 -0.00157  0.00000  0.00000  2.54959
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 85.7594   16.4594   5.210 1.88e-07 ***
## overtimeYes  2.5156    0.8766   2.870  0.00411 **
## job_lvl     -0.1372    0.4805  -0.285  0.77528
## engagement   -0.6845    0.6304  -1.086  0.27755
## interview_rating -24.9181   4.7673  -5.227 1.72e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 869.013  on 979  degrees of freedom
## Residual deviance: 44.466  on 975  degrees of freedom
## AIC: 54.466
##
## Number of Fisher Scoring iterations: 11
##
## Call:
## glm(formula = active ~ overtime + job_lvl + engagement + interview_rating,
##      family = binomial, data = training_dat_os)
##
## Deviance Residuals:
```

```

##      Min      1Q Median      3Q      Max
## -3.801   0.000   0.000   0.000   2.704
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           131.12905  19.56650  6.702 2.06e-11 ***
## overtimeYes          2.27109   0.67614  3.359 0.000782 ***
## job_lvl              0.04793   0.36730  0.130 0.896172
## engagement          -0.41182   0.42738 -0.964 0.335252
## interview_rating    -37.87539   5.64636 -6.708 1.97e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2276.295  on 1641  degrees of freedom
## Residual deviance:  88.655  on 1637  degrees of freedom
## AIC: 98.655
##
## Number of Fisher Scoring iterations: 12

```

As we can see, results of the two binomial logistic regression models are consistent in the sense that only `overtime` and `interview_rating` emerge as significant in classifying employees into active and inactive classes. Since the oversampled data has a larger n -count, there is greater power to detect effects, and we see this reflected in the larger coefficients and lower standard errors.

Using a similar syntax, we can fit a RF using the `randomForest()` function from the package by the same name:

```

# Load library
library(randomForest)

# Train RF model using original data
rf.fit <- randomForest::randomForest(active ~ overtime + job_lvl
                                     + engagement + interview_rating, data = training_dat)

# Train RF model using oversampled data
rf.os.fit <- randomForest::randomForest(active ~ overtime +
                                         + job_lvl + engagement + interview_rating, data =
                                         + training_dat_os)

```

While RFs generally offer a significant lift in performance beyond a single decision tree, we could also apply *tuning wrappers* around the `randomForest()` function to tune the model's hyperparameters. Experimenting with a range of

values for parameters, such as `mtry` for the number of variables to randomly sample and `ntree` for the number of trees to grow, may further improve model performance. Hyperparameter tuning is beyond the scope of this book, but Kuhn and Johnson (2013) is an excellent resource for a more exhaustive treatment on ML models.

Step 3: Use models to make predictions on test data.

While there are packages in R which provide performance metrics for predictive models, we will create a function for greater visibility into how each metric is calculated:

```
# Develop function that returns a data frame of classification
→ model performance statistics
classifier.perf <- function(actual, predicted){

  # Check for missing values; metrics will be computed on
  → non-missing values only
  predicted <- predicted[!is.na(actual)]
  actual <- actual[!is.na(actual)]
  actual <- actual[!is.na(predicted)]

  # Produce counts for model performance metrics
  TP <- sum(actual == 1 & predicted == 1) # true positives
  TN <- sum(actual == 0 & predicted == 0) # true negatives
  FP <- sum(actual == 0 & predicted == 1) # false positives
  FN <- sum(actual == 1 & predicted == 0) # false negatives
  P <- TP + FN # total positives
  N <- FP + TN # total negatives

  # Store rates to variables
  accuracy <- signif(100 * (sum(actual == predicted) /
  → length(actual)), 3)
  sensitivity <- signif(100 * (TP / (TP + FN)), 3)
  specificity <- signif(100 * (TN / (TN + FP)), 3)
  precision <- signif(100 * (TP / (TP + FP)), 3)
  neg_pred_val <- signif(100 * (TN / (TN + FN)), 3)

  # Format output
  stat_nm <- c("accuracy", "sensitivity", "specificity",
  → "precision", "neg_pred_val")
  stat_vl <- c(accuracy, sensitivity, specificity, precision,
  → neg_pred_val)

  # Return model performance statistics in a data frame
  return(data.frame(stat_nm, stat_vl))
}
```

```
}
```

We can use the `predict()` function in conjunction with the object holding the trained model to predict class values for our test data. For classification, we need to define a probability threshold for classifying observations into classes. This is an important consideration since we want to avoid investing in retention strategies for employees who are not actually going to leave (minimizing false positives), while ensuring employees who are truly at risk are flagged as such (maximizing true positives).

We will predict the class using both binomial logistic regression and RF models, trained on both balanced (SMOTE) and imbalanced class data, and store performance metrics in a single data frame for easy comparison:

```
# Initialize empty data frame for model performance stats
class.perf.metrics <- NULL

# Set probability threshold for classification
prob_threshold <- .7

# Predict with logistic regression model
class.perf.metrics <- rbind(class.perf.metrics, cbind.data.frame(
  model = rep("GLM", nrow(test_dat)),
  classifier.perf(
    actual = test_dat$active,
    predicted = ifelse(predict(glm.fit,
      → test_dat, type = "response") >=
      → prob_threshold, 1, 0)))))

# Predict with logistic regression model (SMOTE)
class.perf.metrics <- rbind(class.perf.metrics, cbind.data.frame(
  model = rep("GLM (SMOTE)", nrow(test_dat)),
  classifier.perf(
    actual = test_dat$active,
    predicted = ifelse(predict(glm.os.fit,
      → test_dat, type = "response") >=
      → prob_threshold, 1, 0)))))

# Predict with RF model
class.perf.metrics <- rbind(class.perf.metrics, cbind.data.frame(
  model = rep("RF", nrow(test_dat)),
  classifier.perf(
    actual = test_dat$active,
    predicted = ifelse(predict(rf.fit,
      → test_dat, type = "response") >=
      → prob_threshold, 1, 0))))
```

```
# Predict with RF model (SMOTE)
class.perf.metrics <- rbind(class.perf.metrics, cbind.data.frame(
  model = rep("RF (SMOTE)", nrow(test_dat)),
  classifier.perf(
    actual = test_dat$active,
    predicted = ifelse(predict(rf.os.fit,
      ~ test_dat, type = "response") >=
      prob_threshold, 1, 0))))
```

Step 4: Evaluate model performance.

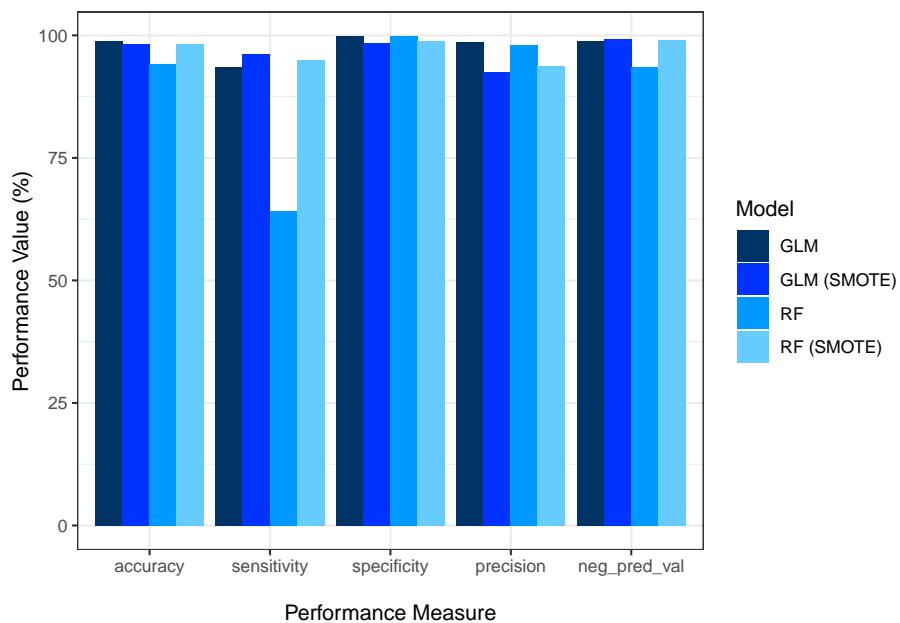


Figure 73: Classification performance for binomial logistic regression and RF models using balanced (SMOTE) and imbalanced classes

As we can see in Figure @ref(fig:class-mdl-perf), with our stringent probability threshold set at .7 for class delineation, all models had perfect specificity (correctly predicting those who stay) and precision (all employees who were predicted to attrit did). However, we see notable differences in sensitivity across the model types, and this is generally a very important performance measure in a predictive attrition project since the cost of not flagging employees who attrit can be costly. Sensitivity for the RF model trained on balanced classes (SMOTE) performed much better than its imbalanced RF counterpart (95.6%

vs. 71.6%), reinforcing that ML models are sensitive (no pun intended) to imbalanced classes.

The results also show that there is likely no benefit to compromising model interpretability by using a flexible ML model like RF since our trusty binomial logistic regression model performs just as well on these data. Nevertheless, we can construct what is known as a **Variable Importance Plot** on RF output using the `varImpPlot()` function from the `randomForest` library to understand the relative importance of each predictor in the model.

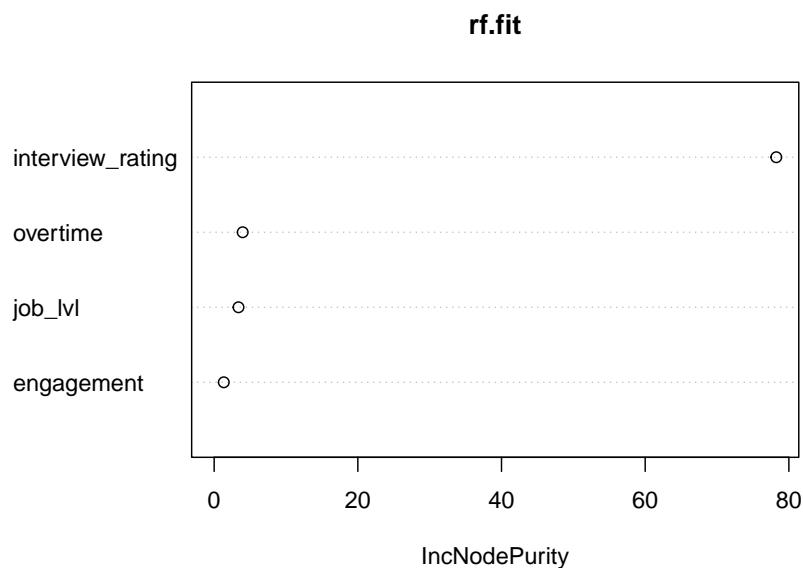


Figure 74: Variable importance plot for Random Forest model

Variable importance plots are based on the mean decrease in **Gini importance**. Gini importance measures the average gain of purity by splits of a given variable. In other words, if a variable is useful it tends to split mixed labeled nodes (nodes with both employees who separated and employees who stayed) into pure single class nodes. The most important variables are at the top of the variable importance plot, indicating that without these variables nodes will not be as pure and as a result, classification performance will not be as strong.

Figure @ref(fig:var-imp-plot) shows that `interview_rating` is far more important than the second most important predictor, `overtime`. This is consistent with what we observed in the results of the binomial logistic regression models too.

Let's compare the information from the RF's Variable Importance Plot to a single decision tree built on training data with balanced classes. We can build

and visualize a decision tree in R using the `rpart()` and `rpart.plot()` functions from libraries by the same names. `rpart` is an acronym for *recursive partitioning and regression trees*:

```
# Load libraries
library(rpart)
library(rpart.plot)

# Construct decision tree on balanced training data
tree <- rpart::rpart(active ~ overtime + job_lvl + engagement +
  interview_rating, data = training_dat_os, method = "class")

# Visualize decision tree
rpart.plot::rpart.plot(tree)
```

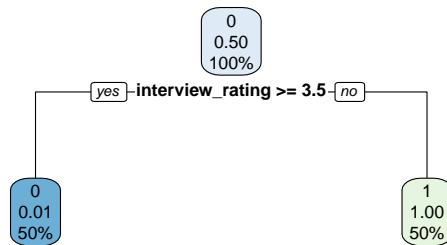


Figure 75: Decision tree for employee attrition prediction using training data with balanced classes (SMOTE)

As shown in Figure @ref(fig:rpart-tree), the most important predictor for splitting the data is `interview_rating` when using a single decision tree. At the root node, there is a 50 percent chance of leaving and staying, which is expected given we balanced the classes using SMOTE. A prediction that is no better than a fair coin toss is of course not helpful. Walking down to the leaf nodes, we can see that for the 50% of employees with `interview_rating >= 3.5`, the algorithm predicts they will stay (`status = 0`); for the 50% of employees with `interview_rating < 3.5`, the algorithm predicts they will leave (`status = 1`). Given what we observed in the results of the binomial logistic regression output, additional partitioning by `overtime` may further increase node purity on the test data since classes are mixed for those with interview ratings

between 3.3 and 3.6. However, given the strong performance splitting data only on `interview_rating`, further partitioning will likely result in modeling noise and overfitting.

Forecasting

To demonstrate the prediction workflow steps for forecasting, we will leverage our `turnover_trends` data set:

```
# Store forecasting data
forecasting_dat <- turnover_trends
```

Step 1: Partition data into training and test sets for cross-validation.

Since we have 60 months of data for each combination of values for the `job`, `level`, and `remote` variables, we will select a combination for which `turnover_rate` can be projected for future months. To simplify, we will train a model using data for the first 48 months and test using the final 12 months.

```
# Create training data
train_dat <- forecasting_dat |> dplyr::filter(job == 'People'
  ~ 'Scientist' & level == 1 & year %in% 1:4)

# Create test data
test_dat <- forecasting_dat |> dplyr::filter(job == 'People'
  ~ 'Scientist' & level == 1 & remote == 'Yes' & year == 5)
```

Step 2: Build models using training data.

Given the significant quadratic and cubic terms identified in Chapter @ref(lme), we will fit a cubic regression model on the training data.

```
# Fit cubic model
train.cube.fit <- lm(turnover_rate ~ year + month + I(month^2) +
  ~ I(month^3) + remote, data = train_dat)

# Produce model summary
summary(train.cube.fit)

## 
## Call:
## lm(formula = turnover_rate ~ year + month + I(month^2) + I(month^3) +
##     remote, data = train_dat)
##
```

```

## Residuals:
##      Min     1Q Median     3Q    Max
## -0.3360 -0.1605  0.0075  0.1680  0.3210
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.7650000 0.1594281 17.343 < 2e-16 ***
## year        -0.1130000 0.0230955 -4.893 4.33e-06 ***
## month       2.4100000 0.0935806 25.753 < 2e-16 ***
## I(month^2)  -0.4100000 0.0163907 -25.014 < 2e-16 ***
## I(month^3)  0.0200000 0.0008311 24.064 < 2e-16 ***
## remoteYes   -1.6400000 0.0516430 -31.756 < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.253 on 90 degrees of freedom
## Multiple R-squared: 0.9499, Adjusted R-squared: 0.9471
## F-statistic: 341.4 on 5 and 90 DF, p-value: < 2.2e-16

```

All linear, quadratic, and cubic terms on `month` are statistically significant at the $p < .001$ level.

Step 3: Use models to make predictions on test data.

We will again build a function to evaluate the performance of the fitted models applied to test data rather than using delivered functions. The following function will return *mean absolute deviation (MAD)*, *mean squared error (MSE)*, and *mean absolute percentage error (MAPE)*:

```

# Develop function that returns a data frame of forecasting model
# performance statistics
forecast.perf <- function(actual, predicted){

  # Check for missing values; metrics will be computed on
  # non-missing values only
  predicted <- predicted[!is.na(actual)]
  actual <- actual[!is.na(actual)]
  actual <- actual[!is.na(predicted)]

  # Store rates to variables
  mad <- round(mean(abs(actual - predicted)), 2)
  mse <- round(mean((actual - predicted)^2), 2)
  mape <- round(mean(abs((actual - predicted) / actual)) * 100,
  # 2)

  # Return model performance statistics in a data frame
}
```

```

    return(data.frame(mad, mse, mape))
}

```

For this forecast, we will also produce a prediction interval. A **prediction interval** is a range of values that is likely to contain the outcome value for a single new observation given a set of predictors. For example, a 95% prediction interval of [10 15] indicates that we can be 95% confident that the observation for which a prediction is being made will have an actual outcome value between 10 and 15. Note that this is very different from a confidence interval in inferential statistics, which is a range of values that likely contains the value of an unknown population parameter.

We can produce a prediction interval by passing an additional `interval = 'predict'` argument into the `predict()` function:

```

# Initialize empty data frames for model predictions and
# → performance stats
forecast.metrics <- NULL
forecast.err.rates <- NULL

# Predict on test_dat
forecast.metrics <- rbind(forecast.metrics, cbind.data.frame(
  month = test_dat$month,
  actual = test_dat$turnover_rate,
  predicted = predict(train.cube.fit,
    ← test_dat, type = "response"),
  lwr_bound =
    ← as.data.frame(predict(train.cube.fit,
    ← test_dat, type = "response",
    ← interval = "predict"))$lwr,
  upr_bound =
    ← as.data.frame(predict(train.cube.fit,
    ← test_dat, type = "response",
    ← interval = "predict"))$upr))

```

Step 4: Evaluate model performance.

Next, we can pass a vector of actual and corresponding predicted values into the `forecast.perf()` function to return MAD, MSE, and MAPE performance metrics for the fitted model applied to year 5 data.

```

# Calculate error rates for year 5 forecast
forecast.perf(actual = forecast.metrics$actual, predicted =
  ← forecast.metrics$predicted)

```

```
##     mad    mse   mape
## 1 3.84 14.75 48.39
```

Given 95% of the variance in turnover rates was explained by the cubic regression model fitted to our training data ($R^2 = .95$), these error rates are surprisingly high for the test data.

Evaluating the average turnover rate by year will help in reconciling the high R^2 on the training data with the high error rates on the test data:

```
# Calculate year-wise turnover rate mean
yr1_avg <- train_dat |> dplyr::filter(remote == 'Yes' & year ==
  ~ 1) |> dplyr::summarize(Mean = mean(turnover_rate))
yr2_avg <- train_dat |> dplyr::filter(remote == 'Yes' & year ==
  ~ 2) |> dplyr::summarize(Mean = mean(turnover_rate))
yr3_avg <- train_dat |> dplyr::filter(remote == 'Yes' & year ==
  ~ 3) |> dplyr::summarize(Mean = mean(turnover_rate))
yr4_avg <- train_dat |> dplyr::filter(remote == 'Yes' & year ==
  ~ 4) |> dplyr::summarize(Mean = mean(turnover_rate))
yr5_avg <- test_dat |> dplyr::summarize(Mean =
  ~ mean(turnover_rate))

# Display year-wise turnover rate mean
print(c(yr1_avg, yr2_avg, yr3_avg, yr4_avg, yr5_avg))
```

```
## $Mean
## [1] 4.506667
##
## $Mean
## [1] 4.816667
##
## $Mean
## [1] 4.046667
##
## $Mean
## [1] 4.386667
##
## $Mean
## [1] 7.996667
```

There is clearly a significant difference in average turnover for year 5 (test data) relative to years 1-4 (training data). Since the fitted model had no visibility into year 5 data, it did not account for the spike in turnover beyond year 4.

Differences are further evidenced in Figure @ref(fig:mdl-yr4-5), in which actual values for year 5 are far and away outside the 95% prediction interval.

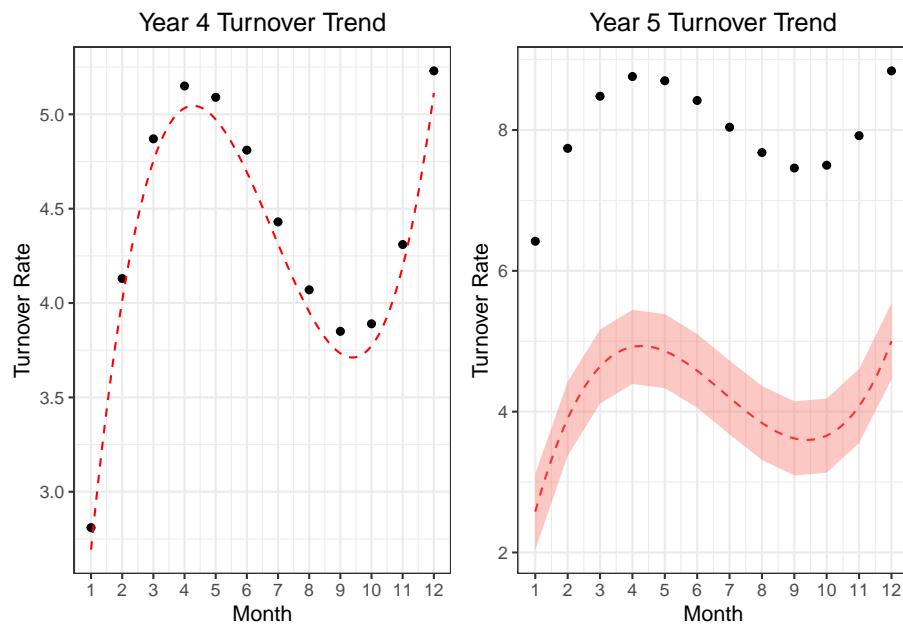


Figure 76: Left: Fitted model (red dashed line) with good fit to year 4 training data (black dots). Right: Fitted model (red dashed line) with poor fit to year 5 test data (black dots). 95 percent prediction interval is represented by the red shaded area around the fit line.

This is an important lesson that highlights the centrality of cross-validation in evaluating whether predictive models will generalize beyond the available data. We can easily fit a model that performs well on training data and claim that the model has exceptional accuracy. However, what matters in predictive modeling is how well the model performs on data it has not seen as part of the training process.

Review Questions

1. What factors influence the balance between model interpretability and flexibility?
2. How does cross-validation (CV) help improve the performance of predictive models?
3. What is bias-variance tradeoff?
4. In a classification setting, what performance metrics are available in a confusion matrix?
5. What are some measures used to evaluate the performance of a forecast?
6. What is Synthetic Minority Oversampling Technique (SMOTE), and how does it help improve the performance of machine learning (ML) models?
7. How is the stack ranking of predictors in a variable importance plot determined?
8. How does a prediction interval differ from a confidence interval?
9. How can a prediction interval be calculated in R?
10. What does high R^2 on training data and high MSE on test data indicate about the utility of a predictive model?

Unsupervised Learning

The inferential and predictive models covered thus far can be categorized as **supervised learning** models. For each observation x_i in the data, there is an associated response y_i in a supervised learning setting, and the goal is to fit a model that relates y to one or more predictors to understand relationships or predict future values on the basis of the identified associations. However, in an **unsupervised learning** setting, no response y_i is associated with x_i . As a result, we cannot *supervise* the analysis and are limited to understanding how observations cluster or group together based on patterns across the available p attributes.

People analytics often involves the unique challenge of analyzing high-dimensional data with a large number of p attributes but relatively few n observations – a phenomenon often referred to as the *curse of dimensionality*. Given the sample size requirements covered in previous chapters, we ideally want n to be an order of magnitude larger than p to support statistical power and increase our chances of detecting meaningful patterns and population effects in sample data. Since people data sets are often wide and short, **dimension reduction** is important for reducing the dimensions to a limited subset that captures the majority of the information and optimizes the $n : p$ ratio.

Consider a case in which a colleague uses verbose rhetoric to convey a simple message that could be effectively communicated with fewer words. The superfluous language is unnecessary and does not provide additional information or value. This is analogous to dimension reduction in that we are interested in identifying a limited set of meaningful attributes and discarding redundant and unimportant information that does not contribute to the analysis objectives.

Dimensionality reduction techniques project data onto a lower-dimensional subspace that retains the majority of the variance in the data points. If we take a picture of a group of colleagues during a team outing, for example, we would lose some 3D information by encoding the information into a 2D image. This 2D representation is a *subspace* of the 3D coordinates. While we would not know how far one person is from another in the 2D representation, we would see that people in the back appear smaller than people in the front. Therefore, the per-

spective in the 2D image would still capture *some* information about distance. The limited information loss in moving from three to two dimensions is likely acceptable, assuming the objective is to capture the memory of the team in a photograph.

Dimension reduction is particularly important in survey research because longer surveys are costly and may result in lower response rates due to the increased completion time requirements. Survey instrumentation with strong psychometric properties features highly correlated survey items for constructs that are relatively uncorrelated with survey items used to measure other independent constructs. Intuitively, we know that highly correlated variables do not capture unique information, as one is a sufficient proxy to capture the available signal in the larger number of features. As we have covered, models with highly correlated variables can create problems due to multicollinearity, and dimension reduction is an alternative approach to variable selection techniques such as the backward stepwise procedure covered in Chapter @ref(lm).

This chapter will cover dimension reduction fundamentals as well as technical implementations.

Factor Analysis

The development of survey instrumentation, whether a single item or a larger multidimensional scale, begins with a good theory. The theory provides conceptual support for the construct – the particular dimensions that characterize the construct, the antecedent variables which theoretically influence it, and the outcomes it will likely drive. With a strong theoretical framework, the researcher can begin proposing ways of *operationalizing* the conceptual scheme into a measurement approach.

There are clear measurement approaches for business metrics such as leads generated, new business growth, cNPS, and net revenue but in the social sciences, we often need indicators of **latent constructs** that are difficult – or impossible – to directly measure. If we want to understand the extent to which employees are engaged in their work, we need a comprehensive measure that captures facets of the theoretical frame. For example, vigor, absorption, and dedication are dimensions of Schaufeli, Bakker, and Salanova's (2006) conception of work engagement which were operationalized in the Utrecht Work Engagement Scale (UWES).

Quantifying the energy levels one brings to work (vigor), the extent to which one feels time passes quickly while working (absorption), and the level of one's commitment to seeing tasks through to completion (dedication) is challenging since we cannot leverage transactional data or digital exhaust to directly quantify this as we can with operational business metrics. We need a comprehensive

– yet parsimonious – set of survey items that function as indicators of the dimensions of the latent work engagement construct. Constructing a larger aggregate measure from the individual indicators, such as the average or sum of all survey items, enables us to reduce the number of variables and optimize the $n : p$ ratio in supervised learning settings.

Exploratory Factor Analysis (EFA)

Exploratory factor analysis (EFA) is a variable reduction technique by which factors are extracted from data – usually as part of the development process for new survey instruments.

A researcher may work with a panel of experts in a particular domain to develop an inventory of items that tap various aspects of the construct per the theoretical framework that underpins it. Based on how the items cluster together, the empirical data will be reconciled against the theoretical conception to define dimensions of the measure. Within a cluster of highly correlated items for a particular dimension, the researcher needs to decide which items are essential and which are redundant and eligible for removal. Aside from the principal clusters (or factors), remaining items also need to be evaluated for their relevance and support for the underlying theory. If items are believed to be members of the theoretical dimensions but do not cluster together with other similar items, it may be indicative of poorly written items that have different interpretations among survey takers. EFA is the empirical process that supports these objectives.

To illustrate the steps for EFA, we will leverage the `survey_responses` data.

```
# Load library
library(peopleanalytics)

# Load data
data("survey_responses")

# Store data in df with curtailed name
survey_dat <- survey_responses

# Show dimensions of survey data
dim(survey_dat)

## [1] 400 12
```

EFA is implemented via a three-step procedure:

1. Assess the factorability of the data.

2. Extract the factors.
3. Rotate and interpret the factors.

Step 1: Factorability Assessment

With respect to factorability, there needs to be some correlation among variables in order for a dimension reduction technique to collapse variables into linear combinations that capture a large portion of the variance in the data. The data feature sufficient factorability if we achieve a **Kaiser-Meyer-Olkin (KMO)** statistic of at least .60 (Kaiser, 1974) and **Bartlett's Test of Sphericity** reaches statistical significance (Bartlett, 1954). The KMO statistic estimates the proportion of variance that may be common variance; the lower the proportion, the greater the factorability. Bartlett's test essentially measures the degree of redundancy in the data, where the null hypothesis states that the variables are orthogonal (uncorrelated); rejecting this null hypothesis indicates that there is sufficient correlation for dimension reduction.

The `KMO()` and `cortest.bartlett()` functions from the `psych` library can be used for the KMO statistic and Bartlett's test, respectively:

```
# Load library
library(psych)

# Kaiser-Meyer-Olkin (KMO) statistic
psych::KMO(survey_dat)

## Kaiser-Meyer-Olkin factor adequacy
## Call: psych::KMO(r = survey_dat)
## Overall MSA =  0.9
## MSA for each item =
##   belong   effort     incl   eng_1   eng_2   eng_3    happ psafety   ret_1   ret_2
##   0.94     0.86     0.86     0.86     0.89     0.89     0.92     0.90     0.91     0.89
##   ret_3   ldrshp
##   0.90     0.93

# Bartlett's Test of Sphericity
psych::cortest.bartlett(cor(survey_dat), nrow(survey_dat))

## $chisq
## [1] 2933.161
##
## $p.value
## [1] 0
##
## $df
## [1] 66
```

Data satisfy the factorability requirements since $KMO = .90$ (Overall MSA) and Bartlett's test is significant at the $p < .001$ level.

Step 2: Factor Extraction

For the second step, we will visually inspect a **scree plot** and determine how many factors are necessary to explain most of the variance in the data. A scree plot is a line plot that helps visualize the portion of the total variance explained by each factor using **eigenvalues**. While the linear algebraic underpinnings are out of scope for this book, it is important to understand that **eigenvectors** are vectors of a linear transformation which have corresponding eigenvalues λ that represent factors by which the vectors are scaled. As a general rule, factors with $\lambda \geq 1$ are extracted when running a factor analysis.

The `scree()` function from the `psych` library can be used to generate a scree plot:

```
# Produce scree plot
psych::scree(survey_dat, pc = FALSE)
```

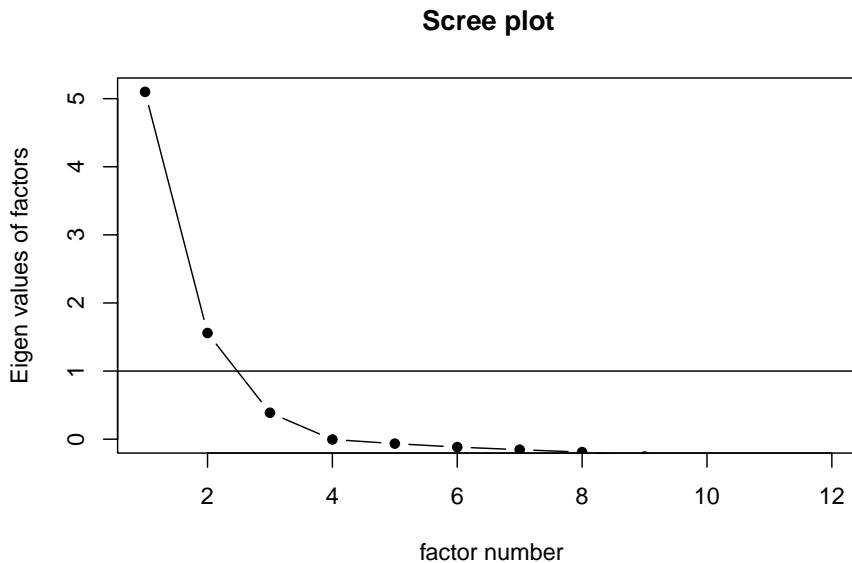


Figure 77: Scree plot showing eigenvalues by factor relative to the extraction threshold (horizontal line)

Based on Figure @ref(fig:scree-plot), factors 1 and 2 appear to provide relatively outsized information gain as $\lambda > 1$ for both.

You may notice the `pc = FALSE` argument in the `scree()` function call. This relates to **principal components analysis (PCA)**, which is an alternative method of dimension reduction. Principal components are new independent variables that represent linear transformations of scaled $((x - \bar{x})/s)$ versions of the observed variables. While we will focus on factor analysis and PCA in this section, which are most common in the social sciences, there are additional dimension reduction techniques one could explore (e.g., **parallel analysis**).

While there are similarities between factor analysis and PCA, the mathematics are fundamentally different. PCA approaches dimension reduction by creating one or more index variables (linear combinations of original variables) from a larger set of measured variables; these new index variables are referred to as components. On the other hand, factor analysis can be viewed as a set of regression equations with weighted relationships that represent the measurement of a latent variable. Most variables are latent in social psychology contexts since we cannot directly measure constructs like psychological safety or belonging.

To illustrate how to extract principal components, we can use base R's `prcomp()` function:

```
# Load library
library(ggplot2)

# Perform PCA
pca <- prcomp(survey_dat, scale = TRUE)

# Calculate explained variance for each principal component
pca_var = (pca$sdev^2 / sum(pca$sdev^2)) * 100

# Create scree plot
ggplot2::qplot(1:length(pca_var), pca_var) +
  ggplot2::geom_line() +
  ggplot2::scale_x_continuous(breaks = 1:length(pca_var)) +
  ggplot2::labs(x = "Principal Component", y = "Variance Explained
  ~ (%)" ) +
  ggplot2::theme_bw()
```

Note that while we can theoretically have as many factors as we have variables ($p = 12$), this defeats the purpose of dimension reduction – whether PCA or factor analysis. The objective of dimension reduction is to *reduce* the number of factors (or components) to a subset that captures the majority of the information in the data.

As shown in Figure @ref(fig:pca-var-plot), principal components are sorted in descending order according to the percent of total variance they explain. The first principal component alone explains nearly half of the total variance in the data. We are looking for the *elbow* to ascertain the inflection point at which

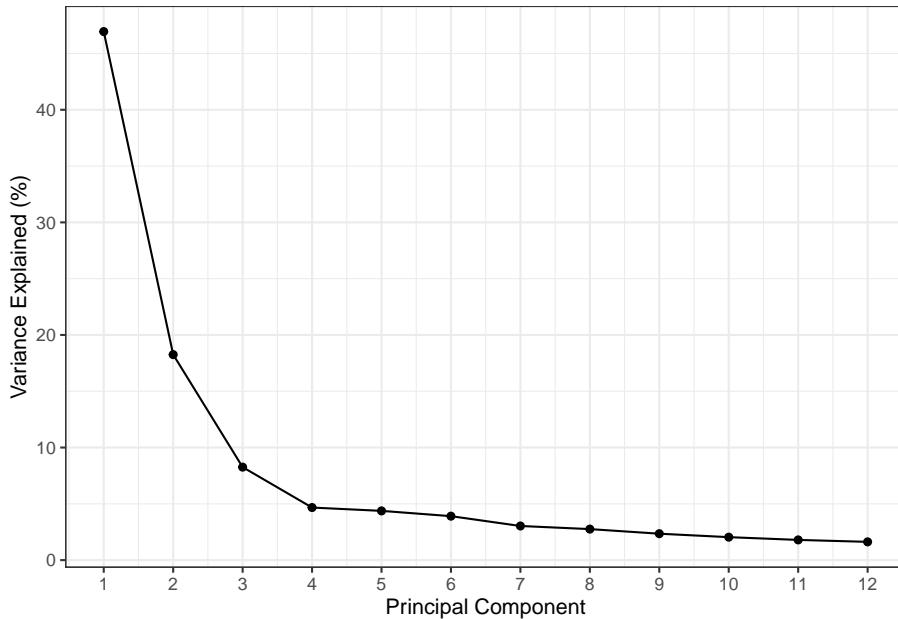


Figure 78: Plot showing the percent of total variance explained by 12 principal components

explained variance plateaus. It is clear that the slope of the line begins to flatten beyond the third principal component, indicating that components 4-12 provide relatively little information. Put differently, we could extract only the first three components without sacrificing much information and gain the benefit of fewer, more meaningful variables.

As an aside, in a supervised learning context we could insert these principal components as predictors in a regression model in lieu of a larger number of original variables. This is known as **principal components regression (PCR)**. However, given the importance of explaining models in a people analytics setting, PCR will not be covered since inserting index variables as predictors in the model compromises interpretability.

Step 3: Factor Rotation & Interpretation

For the third step, we will use an oblimin method to rotate the factor matrix. The oblimin rotation is an oblique – rather than orthogonal – rotation and is selected here since it is best suited when underlying dimensions are assumed to be correlated (Hair et al., 2006).

The `fa()` (factor analysis) function from the `psych` package can be used for the implementation in R. Based on the scree plot, we will specify three factors for this analysis. Note that the oblimin rotation is the default for factor analy-

sis, while a varimax (orthogonal) rotation is the default for PCA. Many other rotations can be implemented based on the nature of data and n -count.

```
# Principal axis factoring using 3 factors and oblimin rotation
efa.fit <- psych::fa(survey_dat, nfactors = 3, rotate =
  ~ 'oblimin')

# Display factor loadings
efa.fit$loadings

## 
## Loadings:
##          MR1     MR2     MR3
## belong    0.283      0.456
## effort   -0.114    0.869
## incl        0.747
## eng_1       0.886
## eng_2       0.172    0.782
## eng_3       0.799
## happ        0.558      0.355
## psafety      0.609
## ret_1        0.791
## ret_2        0.922     -0.111
## ret_3        0.822
## ldrshp       0.556      0.276
##
##          MR1     MR2     MR3
## SS loadings  2.906  2.817  1.363
## Proportion Var 0.242  0.235  0.114
## Cumulative Var 0.242  0.477  0.590
```

The sum of squared loadings (**SS loadings**) are the eigenvalues for each factor. It is also helpful to review the percent of total variance explained by each factor (**Proportion Var**) along with the cumulative percent of total variance (**Cumulative Var**). We can see that the three factors have $\lambda \geq 1$, which together explain 59 percent of the total variance in the data.

By reviewing the factor loadings, we gain an understanding of which variables are part of each factor (i.e., highly correlated variables which cluster together). Factor loadings represent the correlation of each item with the respective factor. While there is not a consensus on thresholds, a general rule of thumb is that *absolute* factor loadings should be at least .5. Items with lower factor loadings should be removed from the measurement model.

For the first factor **MR1**, the three retention items cluster together with happiness and leadership. This indicates that happier employees who have more favorable perceptions of leadership are less likely to leave the organization.

Loadings for the second factor MR2 indicate that the three engagement items cluster together with discretionary effort. This makes intuitive sense, as we would expect highly engaged employees to contribute higher levels of effort towards their work.

Loadings for the third factor MR3 show that belonging, inclusion, and psychological safety cluster together. In other words, employees who feel a stronger sense of belonging and perceive the environment to be more inclusive tend to experience a more favorable climate with respect to psychological safety.

We can visualize this information using the `fa.diagram()` function from the `psych` library:

```
psych::fa.diagram(efa.fit)
```

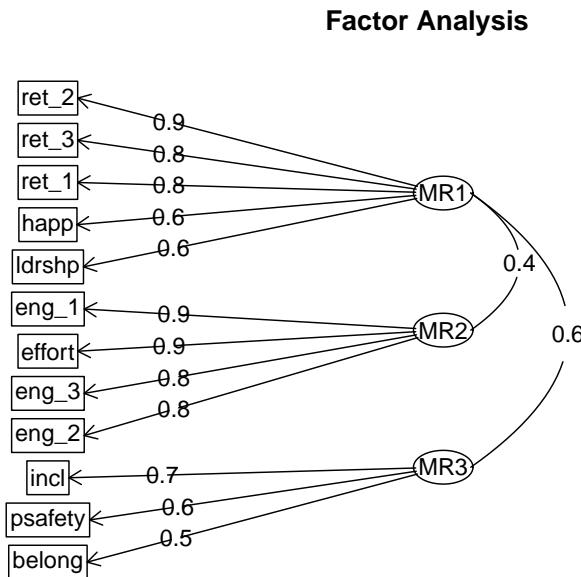


Figure 79: Diagram showing factor loadings (correlations) for each item with the respective factor

Confirmatory Factor Analysis (CFA)

Confirmatory factor analysis (CFA) is used to test how well data aligns with a theoretical factor structure.

We expect items associated with a given construct to be highly correlated with one another but relatively uncorrelated with items associated with independent

constructs. Consider engagement and retention, which are two independent – yet likely correlated – constructs. If multiple items are needed to measure the theoretical dimensions of both engagement and retention, we would expect the engagement items to be more highly correlated with one another than with the retention items. Theory may suggest that retention likelihood increases as engagement increases, but there are many other factors which also influence one's decision to leave an organization beyond engagement, so we would not expect changes in engagement levels to *always* be associated with a commensurate change in retention.

We can illustrate using our `survey_responses` data, which contains three items for both engagement and retention. Let's first evaluate pairwise relationships between the items:

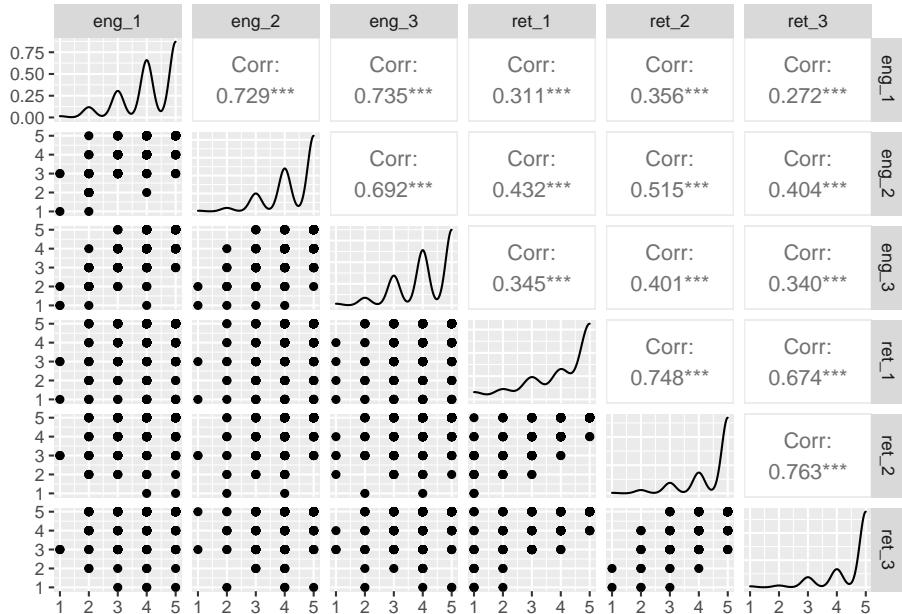


Figure 80: Bivariate correlations and relationship visualizations for engagement and retention survey items

As expected, `eng_1`, `eng_2`, and `eng_3` have stronger correlations with each other ($r \geq .70$) than with `ret_1`, `ret_2`, or `ret_3` ($r \leq .52$).

CFA enables us to move beyond inter-item correlations to quantify the extent to which latent variables in our data fit an expected theoretical model. We can leverage the `lavaan` package in R to perform CFA, which is implemented via a three-step procedure:

1. Define the model.

2. Fit the model.
3. Interpret the output.

Step 1: Define the model.

Step one is defining the model within a string per the syntax required by lavaan:

```
# Load library
library(lavaan)

# Model specification; each line represents a separate latent
# factor
model <- paste('engagement =~ eng_1 + eng_2 + eng_3
                 retention =~ ret_1 + ret_2 + ret_3')
```

Step 2: Fit the model.

Step two is fitting the model to the data using the `cfa()` function from the lavaan package:

```
# Fit the model
cfa.fit <- lavaan::cfa(model, data = survey_dat)
```

We can also create what is known as a **path diagram** to assist with understanding the CFA model. A path diagram is a symbolic visualization of the measurement model, with circles depicting latent variables (factors), rectangles representing observed indicators (survey items), and arrows indicating paths (relationships) between variables. The measurement model (CFA) together with the structural (path) model is known as **structural equation modeling (SEM)**; CFA is a subset of the SEM umbrella.

The `lavaanPlot()` package can be used to create and visualize path diagrams in R:

```
# Load library
library(lavaanPlot)

# Visualize path diagram
lavaanPlot::lavaanPlot(model = cfa.fit, coefs = TRUE, stand =
  TRUE)
```

Path diagram showing survey items as indicators of latent engagement and retention factors

Factor loadings are shown for each indicator of the latent variable in Figure @ref(fig:path-diagram). All are well above the *absolute* threshold of .5.

Step 3: Interpret the model.

Step three is interpreting the output of the fitted model:

```
cfa.fit <- lavaan::cfa(model, data = survey_dat)

# Summarize the model
summary(cfa.fit, fit.measures = TRUE)

## lavaan 0.6-12 ended normally after 25 iterations
##
##             ML
## Estimator          ML
## Optimization method NLMINB
## Number of model parameters      13
##
## Number of observations           400
##
## Model Test User Model:
## 
##             35.477
## Test statistic
## Degrees of freedom              8
## P-value (Chi-square)            0.000
##
## Model Test Baseline Model:
## 
##             1495.174
## Test statistic
## Degrees of freedom              15
## P-value                          0.000
##
## User Model versus Baseline Model:
## 
##             0.981
## Comparative Fit Index (CFI)
## Tucker-Lewis Index (TLI)         0.965
##
## Loglikelihood and Information Criteria:
## 
##             -2456.223
## Loglikelihood user model (H0)
## Loglikelihood unrestricted model (H1) -2438.484
##
##             4938.446
## Akaike (AIC)
## Bayesian (BIC)                  4990.335
## Sample-size adjusted Bayesian (BIC) 4949.085
##
## Root Mean Square Error of Approximation:
## 
##             0.093
## RMSEA
```

```

##    90 Percent confidence interval - lower          0.063
##    90 Percent confidence interval - upper          0.125
##    P-value RMSEA <= 0.05                      0.011
##
## Standardized Root Mean Square Residual:
##
##    SRMR                               0.040
##
## Parameter Estimates:
##
##    Standard errors                   Standard
##    Information                       Expected
##    Information saturated (h1) model   Structured
##
## Latent Variables:
##              Estimate Std.Err z-value P(>|z|)
## engagement =~
##   eng_1           1.000
##   eng_2           0.880   0.044  19.827  0.000
##   eng_3           0.963   0.050  19.355  0.000
## retention =~
##   ret_1           1.000
##   ret_2           0.799   0.039  20.384  0.000
##   ret_3           0.699   0.038  18.562  0.000
##
## Covariances:
##              Estimate Std.Err z-value P(>|z|)
## engagement ~~
##   retention       0.406   0.051  7.932  0.000
##
## Variances:
##              Estimate Std.Err z-value P(>|z|)
##   .eng_1          0.235   0.027  8.839  0.000
##   .eng_2          0.188   0.021  9.013  0.000
##   .eng_3          0.265   0.027  9.820  0.000
##   .ret_1          0.482   0.044 10.952  0.000
##   .ret_2          0.095   0.018  5.145  0.000
##   .ret_3          0.215   0.020 10.571  0.000
##   engagement     0.649   0.064 10.188  0.000
##   retention      0.917   0.097  9.458  0.000

```

The `lavaan` package provides many fit measures, but we will focus only on the most common for evaluating how well the data fit the measurement model.

- Model Chi-Square (χ^2): Tests whether the covariance matrix derived from

the model represents the population covariance (Test Statistic under the Model Test User Model section of the `lavaan` output)

- Comparative Fit Index (CFI): Values range from 0 to 1, with $CFI > .95$ indicating good fit
- Tucker Lewis Index (TLI): Values range from 0 to 1, with $TLI > .95$ indicating good fit
- Root Mean Square Error of Approximation (RMSEA): Values of .01, .05, and .08 indicate excellent, good, and mediocre fit, respectively (though some texts suggest .10 is an adequate threshold for mediocre fit)
- Standardized Root Mean Square Residual (SRMR): Square root of the difference between residuals of the sample covariance matrix and the hypothesized model, with $SRMR < .08$ indicating good fit

Given data sets are often small in people analytics, it is important to note that RMSEA often exceeds thresholds with small df and n – even when the model is correctly specified (Kenny, Kaniskan, & McCoach, 2015). Therefore, it is important to index more on fit indices such as *CFI* and *LTI* in determining how well the data fit the measurement model.

The χ^2 statistic is sometimes referred to as a ‘badness of fit’ measure since rejecting the null hypothesis ($p < .05$) indicates a lack of fit. Though χ^2 is significant ($p < .001$), both CFI (.98) and TLI (.97) are above the .95 threshold for good fit. In addition, $SRMR = .04$ is beneath the threshold of .08 and $RMSEA = .09$ is between the mediocre fit threshold range of .08 and .10. Therefore, the indicators (survey items) in these data adequately fit the two latent constructs defined by this measurement model.

For more extensive coverage of SEM, Kline (2005) is an excellent resource.

Clustering

Clustering is a ML technique that groups observations into clusters which have similar characteristics but different characteristics relative to the observations in other clusters. Clustering is similar to factor analysis in that it is also unsupervised since there is not a response variable, but it differs in that it does not seek to find a low-dimensional representation of observations that capture a large portion of variance in the data; clustering aims to find homogeneous subgroups among observations.

Clustering is common in marketing in which it is implemented to create customer segments with shared characteristics. By grouping customers based on attributes such as income, household size, occupation, and geography, companies can tailor marketing campaigns to each segment based on what is most likely to appeal to the unique needs of each.

In people analytics, clustering has important applications as well. For example, clustering can be implemented to define personas based on unique talent

development needs (e.g., early tech career, newly promoted people leaders) or attrition risk (e.g., rising stars with hot skills, low performers in high churn roles, high performers in specialized roles). Data privacy regulations often prevent assigning a predictive score to individuals; therefore, grouping employees based on relative attrition risk levels can support action planning at the segment level.

This section will cover two popular clustering techniques: *k-means* and *hierarchical* clustering.

K-Means Clustering

K-means clustering is a simple approach to grouping observations into K distinct clusters. *K*-means clustering is implemented via a four-step process:

1. Define K .
2. Randomly assign observations to one of K clusters.
3. For each of the K clusters, compute the cluster centroid.
4. Assign each observation to the cluster with the closest centroid (middle).

To assign observations to the cluster with the nearest centroid, a distance metric needs to be selected in order to measure the distance between each observation and cluster centroids. While calculating the distance between observations in two dimensions is simple, distance in higher dimensional space is more challenging. We will focus on the most common distance metric, **Euclidean distance**, though there are many others (e.g., Manhattan, Jaccard, Minkowski, Cosine). The Euclidean distance between two data points is the straight line distance based on the observations' coordinates using the **Pythagorean theorem**:

$$a^2 + b^2 = c^2,$$

where a and b are sides of a triangle that intersect to form a right angle, and c is the hypotenuse (the side opposite the right angle).

Let's implement *K*-means clustering using numeric variables in the `employees` data. Since the scale of variables matters when comparing distances between observations and cluster centers, we will first scale the variables to have $\bar{x} = 0$ and $s = 1$ in support of a consistent, apples-to-apples comparison:

```
# Filter employee data to numeric variables
idx <- which(sapply(employees, is.numeric)) # store indices of
# numeric variables
employees <- employees[, idx] # filter df using indices

# Drop unimportant and sparsely populated sales variables
employees <- subset(employees, select = -c(employee_id,
# standard_hrs, ytd_leads, ytd_sales))
```

```
# Center and scale data
employees_trans <- scale(employees, center = TRUE, scale = TRUE)
```

Next, we need to define K . One way to determine the optimal number of clusters is to leverage the `fviz_nbclust()` function from the `factoextra` library to visualize the sum of squared differences between observations and cluster centers against the range of clusters:

```
# Load library
library(factoextra)

# Determine optimal number of clusters
factoextra::fviz_nbclust(employees_trans, kmeans, method = "wss")
```

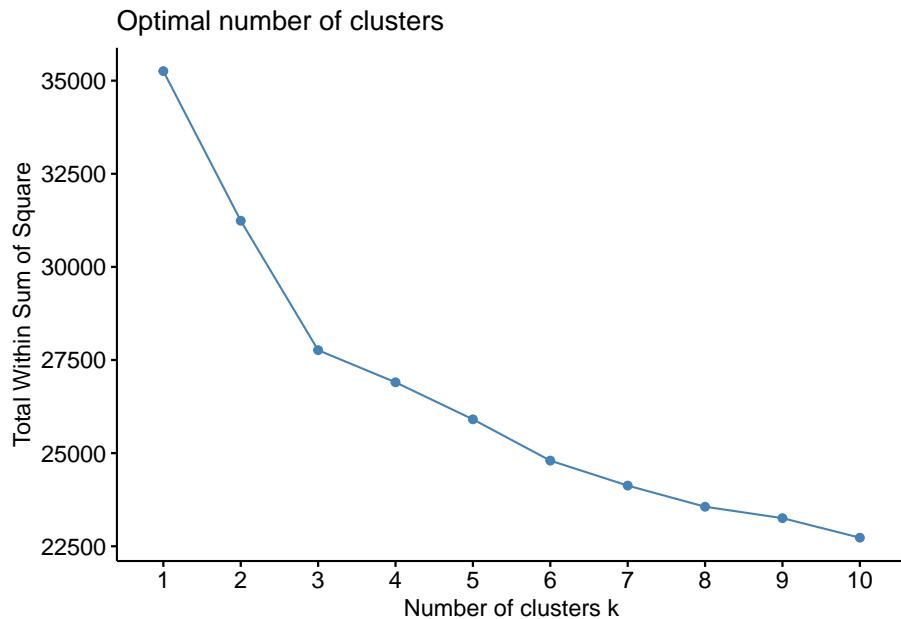


Figure 81: Total within sum of squares (WSS) across cluster count

When interpreting Figure @ref(fig:kmeans-elbow-plot), we are looking for the *elbow* which marks the inflection point at which the sum of squares begins to level off. The goal is to achieve the fewest number of clusters, optimizing for subgroups that are distinctly different *between* and highly similar *within*. The elbow indicates the optimal number of clusters, as additional clusters beyond

the elbow do not offer a meaningful improvement in achieving homogeneous subgroups of the observations.

There is a discernible elbow at three clusters in Figure @ref(fig:kmeans-elbow-plot). Intuitively, fewer clusters promotes action taking in people analytics since clusters need to be defined, and this becomes increasingly challenging as the number of clusters increases. With a large number of clusters, it may be difficult to meaningfully tailor career development or retention strategies, for example, to the unique needs of employees assigned to each cluster as the distinction between each subgroup becomes more opaque.

We can now implement K -means clustering with $K = 3$ using the `kmeans()` function in base R:

```
# Perform K-means clustering
km <- kmeans(employees_trans, centers = 3)

# Return n-count of clusters
km$size
```

```
## [1] 592 603 275
```

Of the 1,470 employees in our `employees` data, the n distribution across the $K = 3$ clusters is 592, 603, and 275.

We can calculate the mean (and other descriptives) for each variable by cluster to better understand cluster distinctions:

```
# Calculate mean of each cluster using original data
aggregate(employees, by = list(cluster = km$cluster), mean)

##   cluster stock_opt_lvl trainings      age commute_dist    ed_lvl engagement
## 1       1     0.7787162  2.805743 35.31757    8.907095 2.880068  2.685811
## 2       2     0.7860697  2.781095 35.38474    9.646766 2.854063  2.792703
## 3       3     0.8436364  2.825455 43.75636    8.810909 3.112727  2.687273
##   job_lvl hourly_rate daily_comp monthly_comp annual_comp salary_hike_pct
## 1 1.834459    47.09291   376.7432     8162.77    97953.24    15.26520
## 2 1.724710    83.51078   668.0862    14475.20   173702.42    15.24876
## 3 3.301818    67.72364   541.7891    11738.76   140865.16    15.00364
##   perf_rating prior_emplr_cnt env_sat job_sat rel_sat wl_balance work_exp
## 1      3.148649    2.668919 2.746622 2.859797 2.673986   1.824324 9.315878
## 2      3.159204    2.645108 2.666667 2.665008 2.706468   1.852405 8.684909
## 3      3.152727    2.850909 2.789091 2.585455 2.807273   1.854545 21.196364
##   org_tenure job_tenure last_promo mgr_tenure interview_rating
## 1      5.146959    3.452703 1.467905  3.300676        3.979730
```

```
## 2    4.552239   2.827529   1.077944   2.878939       3.952570
## 3   16.527273   8.974545   6.170909   8.621818       4.090545
```

We can see that relative to the first two clusters, the third cluster has – on average – an older demographic with more education and a higher job level. In addition, employees in the first cluster earn significantly lower compensation, on average, which may be correlated with categorical variables that were initially dropped such as `dept` or `job_title`.

We can also add a new column in the `employees` data frame with the cluster assignment from K -means to facilitate further analysis:

```
# Add cluster assignment to df
employees <- cbind(employees, km$cluster)
```

While K -means clustering is a simple and efficient algorithm (even for large datasets), an a priori specification of K is not always ideal. K -means clustering will create K clusters – even if they are nonsensical – so caution must be exercised. Plotting WSS against cluster count as shown in Figure @ref(fig:kmeans-elbow-plot) can be helpful in defining K , but alternative clustering algorithms exist that do not require K to be predefined.

Hierarchical Clustering

Like K -means clustering, **hierarchical clustering** seeks to group observations into clusters which have similar characteristics but different characteristics relative to the observations in other clusters. However, unlike K -means, the number of clusters is not specified prior to implementing the algorithm with hierarchical clustering. The optimal number of clusters is determined using a **dendrogram**, which is a tree diagram visualizing the hierarchical relationships in data.

One key difference between K -means and hierarchical clustering is that hierarchical clustering involves linkage methods to measure cluster similarity. There is not a one-size-fits-all option for linkage, as the performance of a given linkage technique can vary based on the structure of the data. Outlined below are the five most common types of linkage in hierarchical clustering:

1. Complete Linkage: distance between two clusters is defined as the maximum distance between any individual data point in cluster A and any individual data point in cluster B
2. Single Linkage: distance between two clusters is defined as the minimum distance between any individual data point in cluster A and any individual data point in cluster B
3. Average Linkage: distance between two clusters is defined as the average distance between data points in cluster A and data points in cluster B

4. Centroid Method: distance between two clusters is defined as the distance between the centroid of cluster A and the centroid of cluster B
5. Ward's Method: ANOVA-based approach in which the distance between clusters A and B is based on how the sum of squared distances increases when the clusters are merged

To implement hierarchical clustering, we will leverage the same centered and scaled data used for K -means clustering in the prior section. Note that the `km_cluster` column was only added to the original `employees` data; if this column was present in `employees_trans`, we would need to drop it so that the hierarchical clustering algorithm is not influenced by results of another clustering technique (K -means).

Since we do not know what linkage method will work best for these data, we will also develop a function that enables us to try a range of techniques and select the one that performs best. The `agnes()` function from the `cluster` library is used to implement hierarchical clustering:

```
# Load library
library(cluster)

# Define linkage methods
# Note: centroid is not available for agnes() function
methods <- c("complete", "single", "average", "ward")
names(methods) <- c("complete", "single", "average", "ward")

# Create function to compute agglomerative coefficient
agg_coeff <- function(x) {

  cluster::agnes(employees_trans, method = x)$ac
}

# Compute agglomerative coefficient for each linkage method
sapply(methods, agg_coeff)

## complete    single    average      ward
## 0.7990373 0.6248688 0.7582732 0.9571736
```

Agglomerative coefficients closer to 1 indicate stronger clustering performance. Therefore, Ward's distance measure performs best on these data, and we will implement hierarchical clustering using this linkage option.

```
# Perform hierarchical clustering using Ward's linkage method
hclust <- cluster::agnes(employees_trans, method = "ward")
```

To produce a dendrogram, the `pltree()` function from the `cluster` library can be used in conjunction with the `hclust` object holding the clustering results:

```
cluster::pltree(hclust, main = "Dendrogram")
```

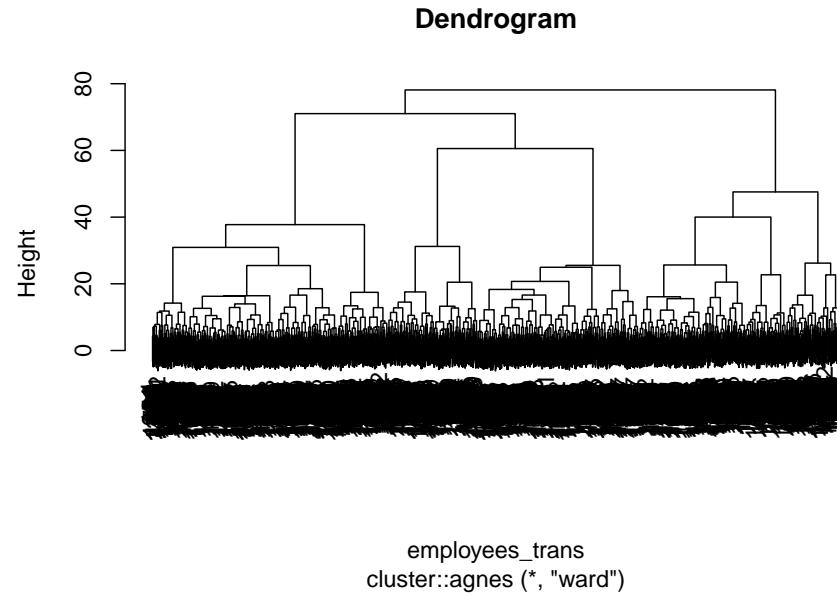


Figure 82: Dendrogram for hierarchical clustering of employees using Ward linkage

At the bottom of the denogram shown in Figure @ref(fig:hclust-dendrogram), each leaf of the tree represents an individual observation. Since $n = 1,470$, the bottom of the tree is too congested to interpret. As we move up the tree, individual observations are fused together based on the degree of similarity as defined by Ward's linkage method.

To aid in determining the optimal number of clusters, a **gap statistic** can be calculated, which compares the within-cluster variation for different K values to reference values for a random uniform distribution with no clustering. We will use the `clusGap()` function from the `cluster` library to calculate the gap statistic, and then visualize using the `fviz_gap_stat()` function from the `factoextra` library:

```
# Calculate gap statistic across 1-10 clusters
gap_stat <- cluster::clusGap(employees_trans, FUN = hcut, nstart
                           = 25, K.max = 10, B = 50)
```

```
# Generate plot of gap statistic against cluster count
factoextra::fviz_gap_stat(gap_stat)
```

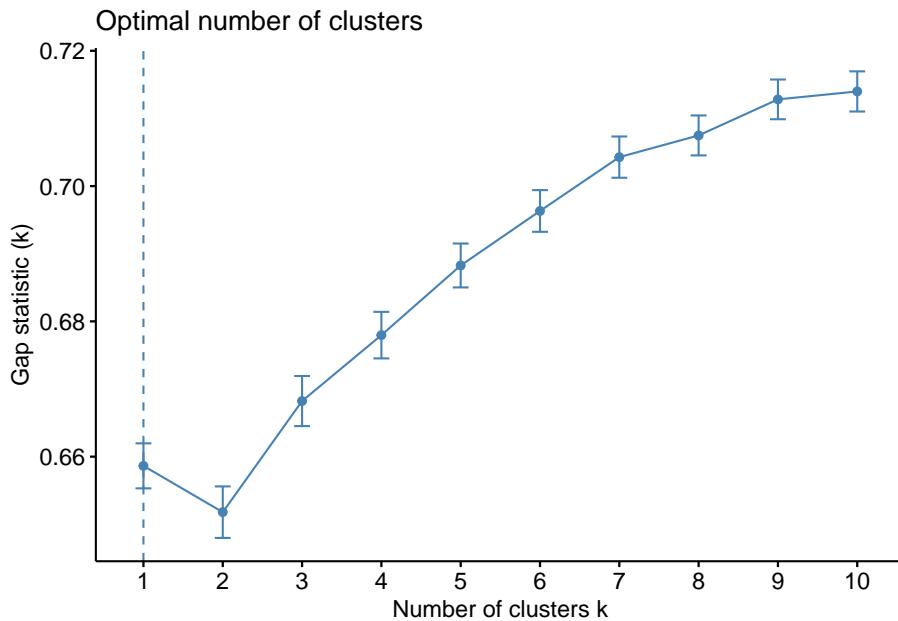


Figure 83: Plot of gap statistic against cluster count for hierarchical clustering

We ideally want to select the value of K that maximizes the gap statistic. In practice, however, balancing cluster parsimony with maximization of the gap statistic is not always straightforward. Figure @ref(fig:hclust-gap-stat) indicates that the gap statistic increase is fairly constant across the range of $K = 1$ to $K = 10$ clusters. In this case, we may look to select a value of K based on an inflection point at which the trajectory of increase in the gap statistic begins to slow. Based on this approach, we may select $K = 7$.

We can now cut the dendrogram into 7 clusters using the `cutree()` function, and then append the cluster to each observation in our original `employees` data:

```
# Compute distance matrix
d_matrix <- dist(employees_trans, method = "euclidean")

# Perform hierarchical clustering using Ward's method
hclust_final <- hclust(d_matrix, method = "ward.D2" )

# Cut the dendrogram into 7 clusters
```

```
groups <- cutree(hclust_final, k = 7)

# Append cluster labels to original data
employees <- cbind(employees, hier_cluster = groups)
```

Review Questions

1. How can high dimensional data create problems in analytics, and how do dimension reduction techniques remediate these issues?
2. What is the difference between Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA)?
3. What is the difference between Exploratory Factor Analysis (EFA) and Principal Components Analysis (PCA)?
4. What is Structural Equation Modeling (SEM), and what are some use cases for it in people analytics?
5. How can we test whether data satisfy the eligibility criteria for factor analysis?
6. How are factor loadings interpreted to ascertain which variables are members of each factor?
7. What is a data-informed approach to selecting the optimal value of K in K -means clustering?
8. What is Euclidean distance, and what is its function in clustering?
9. How is a dendrogram interpreted in the context of hierarchical clustering?
10. When optimizing for both cluster parsimony and gap statistic maximization is not feasible, how can the optimal value of K be determined in hierarchical clustering?

Data Visualization

Unlike content in the preceding chapters, data visualization is more an *art* than a *science*. Data visualization is heavily dependent on the type of data being presented, key messages that need to be conveyed, the unique characteristics of the audience to whom information needs to be communicated, and various contextual features germane to a proper interpretation and understanding of material.

This chapter will provide general best practices as well as strategies specific to various types of data that will promote success when communicating data to technical and non-technical stakeholders alike. Since there is a considerable amount of code required to construct the plethora of graphics in this chapter, only the data prep code will be included. You can reference Chapter @ref(appendix) for a curated set of fully reproducible code for each data visualization provided in this chapter.

Best Practices

While the type of visual depends on the nature of data being presented, there are general best practices for effectively communicating information that are applicable for all data types, audiences, and contexts.

Color Palette

There are several important considerations when choosing colors for data visualization.

Color has meaning.

The meaning of color is not consistent across the world. For example, in Western cultures red generally has unfavorable connotations (e.g., off track, danger) while green signals a favorable status (e.g., on track, cash flow positive). However, in Asian cultures red indicates success.

It is important to understand what color indicates in the contexts in which information is shared to ensure color choices are consistent with the messages that need to be conveyed.

Respect color blindness.

Most estimates indicate that 1 in 12 men (8%) and 1 in 200 women (0.5%) are colorblind (~4.5% of the world population). Males are more likely to be colorblind because the X chromosome contains the genes responsible for the most common forms of color blindness. Trouble distinguishing between red and green is the most common manifestation of colorblindness (Gordon, 1998).

This has important implications for data visualization, and it is generally best to avoid the use of red and green. One alternative is to leverage orange to indicate unfavorable data points and blue for favorable. This strategy will be implemented throughout this chapter.

Adhere to brand colors.

In most organizations, the marketing team defines a color palette consistent with the branding used for consumer products and services. In this case, analysts may be constrained to the use of colors within the branding palette.

Be sure to consult with marketing colleagues and adhere to any color palette requirements.

Use color consistently.

To support a correct interpretation of information, it is important to use a consistent color scheme across the various visuals in dashboards, slides, and documents. For example, if blue is assigned to highlight how the Engineering department compares to other departments in a particular view, blue should be assigned to the Engineering department in every view in which department is a grouping dimension.

The consistent use of color requires the audience to wield less effort to understand information, which in turn reduces the risk of incorrect interpretation.

Chart Borders

The goal in data visualization is to enable the most important data to take center stage. Formatting should support – not detract – from this objective. Chart borders are a usual suspect and prime example of formatting that can divert focus away from the data.

Figure @ref(fig:chart-borders) illustrates the use of heavy and light borders for a tabular presentation of data. Formatting takes center stage in the first table due to heavy borders, while formatting is not the focus in the second table given the minimal light grey border.

Dimension	Metric 1	Metric 2	Metric 3
Dimension A	x_1	y_1	z_1
Dimension B	x_2	y_2	z_2
Dimension C	x_3	y_3	z_3

Dimension	Metric 1	Metric 2	Metric 3
Dimension A	x_1	y_1	z_1
Dimension B	x_2	y_2	z_2
Dimension C	x_3	y_3	z_3

Figure 84: Comparison of data tables with heavy (left) and light (right) borders

Zero Baseline

The use of a non-zero baseline can exaggerate differences in metrics, resulting in misleading conclusions.

Figure @ref(fig:baseline-compare) illustrates how minor differences are exaggerated when a non-zero baseline is applied to the y -axis. Given that the average hires across these four years is 1,000, there is actually little variation in hire volume YoY, and this is accurately reflected in the bar chart with a zero baseline.

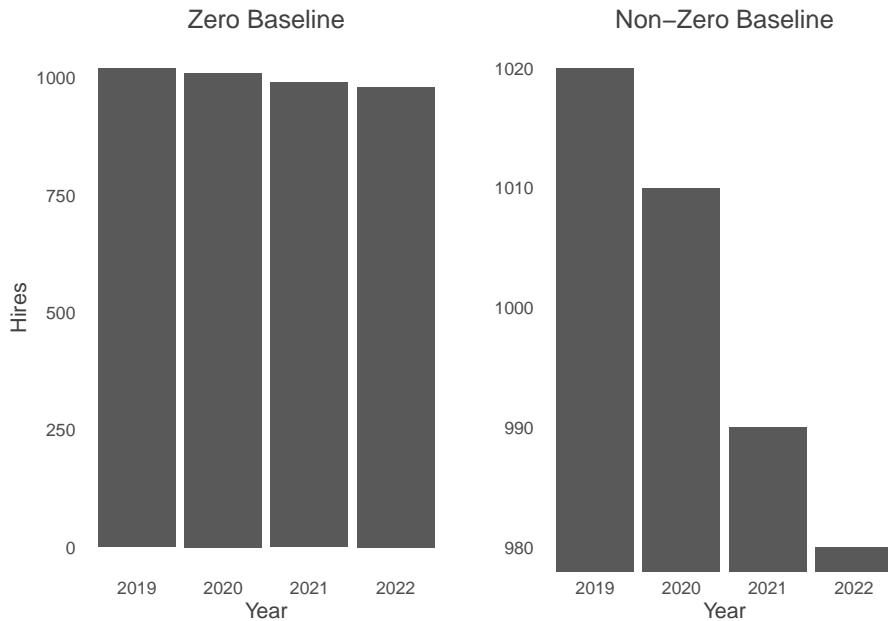


Figure 85: Hires by year with zero baseline (left) and non-zero baseline (right)

Intuitive Layout

When visualizing data, it is best not to deviate from the way in which the audience will naturally interpret the data. Outlined below are a few important considerations:

- Numbers lower on a y -axis will naturally be interpreted to be smaller than numbers higher on the y -axis.
- When there are both positive and negative numbers along an x -axis, negative metrics are best placed on the left and positive on the right since people naturally consume content from left to right.
- Rotated axis labels require more time to read and interpret. If labels must be rotated to fit along an axis, a 45-degree angle is generally preferred over a more extreme 90-degree rotation.

Figure @ref(fig:flipped-axis-viz) shows murders committed using firearms in Florida across time. This visual has an inverted y -axis, which is highly misleading since it suggests spikes in murders are dips. Irrespective of whether this was politically motivated or simply a failed attempt at creativity, this is a deceptive data visualization.

Gun deaths in Florida

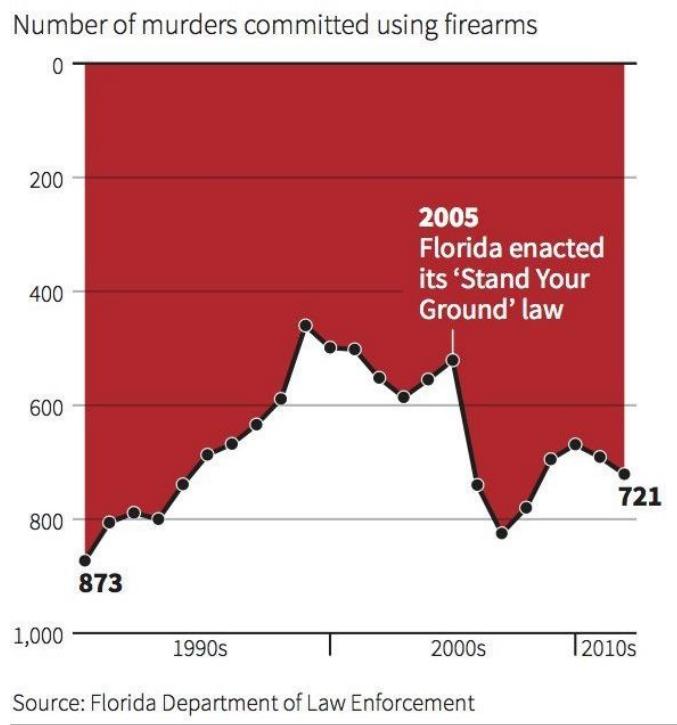


Figure 86: Misleading inverted y-axis

Preattentive Attributes

Preattentive attributes signal where to look and help our audience focus on content we wish to emphasize (Knaflic, 2015).

It is important to remember that data do not always need to be visualized with a chart. Sometimes communicating with text is a sufficient and effective way to convey information. In the case of text, preattentive attributes are often implemented using bold text and/or contrasting colors for keywords and phrases that need to take center stage. **Bolded** text is generally more effective than *italizing* or underlining, as it highlights the chosen elements without unnecessary noise and compromised legibility.

Grey is an important color in the implementation of preattentive attributes as it facilitates a move of less important data and design elements to the background to make room for more predominate colors to highlight the most important information. Figure @ref(fig:pre-attr-1) illustrates the application of preattentive attributes to emphasize the key message in a sentence. This is accomplished by pushing the less important content to the background using grey text and using bold orange text to highlight the labor cost increase.

**Our operating costs have increased
consistently over the past several years,
with **labor costs increasing 30 percent**
YoY due to stiff competition for top
talent and increased hiring in high-cost
locations.**

Figure 87: The application of preattentive attributes to highlight labor costs as a key driver of increased operating expenses

Preattentive attributes can also be applied to data visualizations to focus the audience's attention. Figure @ref(fig:pre-attr-2) illustrates the use of preattentive attributes by assigning a different color (orange) to the labor cost for the current year and muting the labor costs associated with prior years using a less predominate grey.

Simply put, if something is **really important**, make sure it is different from other content on the page, slide, or section.

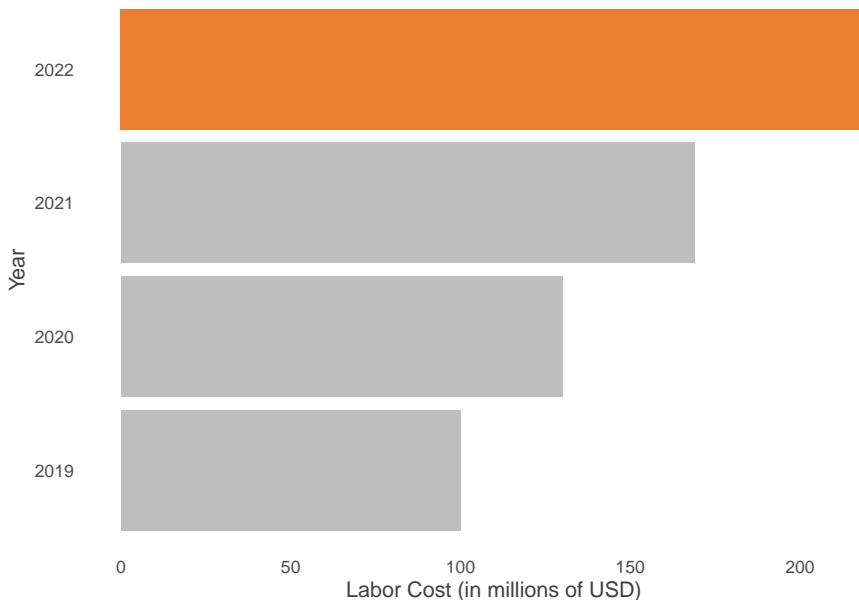


Figure 88: The application of preattentive attributes to highlight labor costs for the current year relative to prior years

Step-by-Step Visual Upgrade

Software knows neither what we wish to highlight nor the audience to whom we intend to communicate the information. Therefore, regardless of the software, design defaults are rarely best.

This section will implement data visualization best practices step-by-step to improve upon the design defaults for the `ggplot2` library.

Step 1: Build Bar Chart with Defaults

We will begin by building a bar chart that shows the distribution of active employees' educational backgrounds. To simplify the data structure for data visualization, let's ingest our `employees` data and create an aggregated cube with counts by `ed_field` for active employees:

```
# Load libraries
library(peopleanalytics)
library(dplyr)
```

```
# Load data
data("employees")

# Create data cube for active employees
smmry_ed_field <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::count(ed_field)
```

A bar chart can be created using the `geom_bar()` function, and we will start with the `ggplot2` library's design defaults:

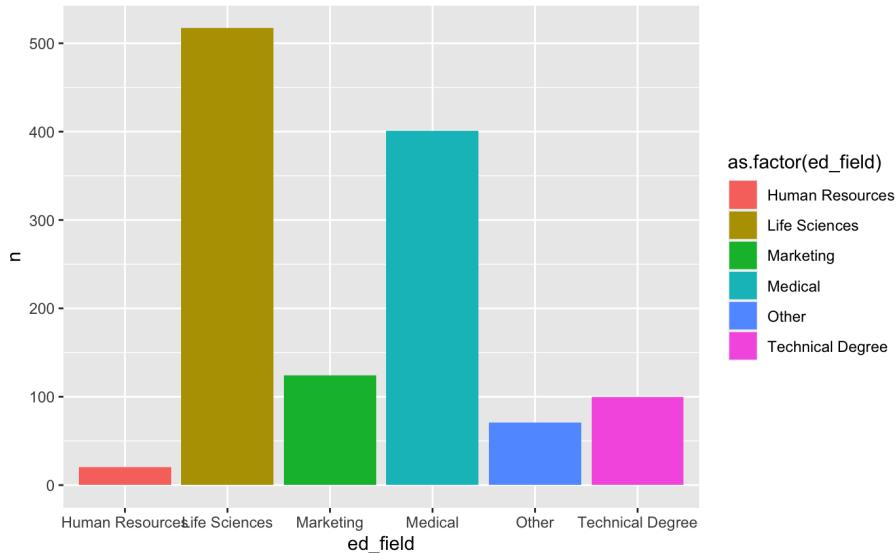


Figure 89: Step 1: Bar chart with defaults

Step 2: Remove Legend

Since the legend is not necessary for interpreting the education field to which each bar corresponds, we will remove it using `theme(legend.position = "none")`. Eliminating the legend also provides more real estate for the chart, which helps make the x-axis values more legible.

Step 3: Assign Colors Strategically

There is a lot competing for attention due to this vibrant color palette. Let's assume that the objective is to highlight the education field pursued by the

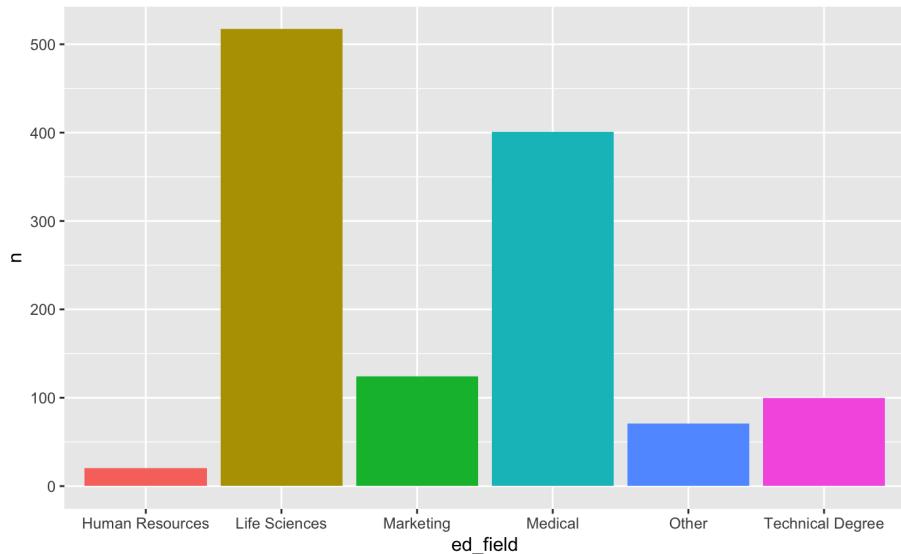


Figure 90: Step 2: Remove legend

largest number of employees. Preattentive attributes lends well to this, and we can assign specific hex color codes to the education categories.

Since Life Sciences was the field pursued by the most employees, let's highlight the corresponding bar in blue and move the remaining bars to the background through the assignment of light grey. One method of accomplishing this is via the following explicit category assignments:

```
scale_fill_manual(values = c("Human Resources" = "#BFBFBF",
"Life Sciences" = "#0070C0", "Marketing" = "#BFBFBF", "Medical" =
"#BFBFBF", "Other" = "#BFBFBF", Technical Degree" = "#BFBFBF"))
```

Step 4: Add Axis Titles and Margins

Axis titles are the column names by default, which is usually not the most user-friendly option. We can assign new mixed case axis titles by chaining `labs(x = 'Education Field', y = 'Headcount')` to the visualization code.

Spacing can also be added between the axis titles and labels to improve upon the default formatting and reduce text congestion. The `ggplot2::element_text(margin = margin(t = 0, r = 0, b = 0, l = 0))` parameter can be defined for the `x` and `y` axes via `axis.title.x` and `axis.title.y`, respectively, where:

- `t` = space on the top of axis title

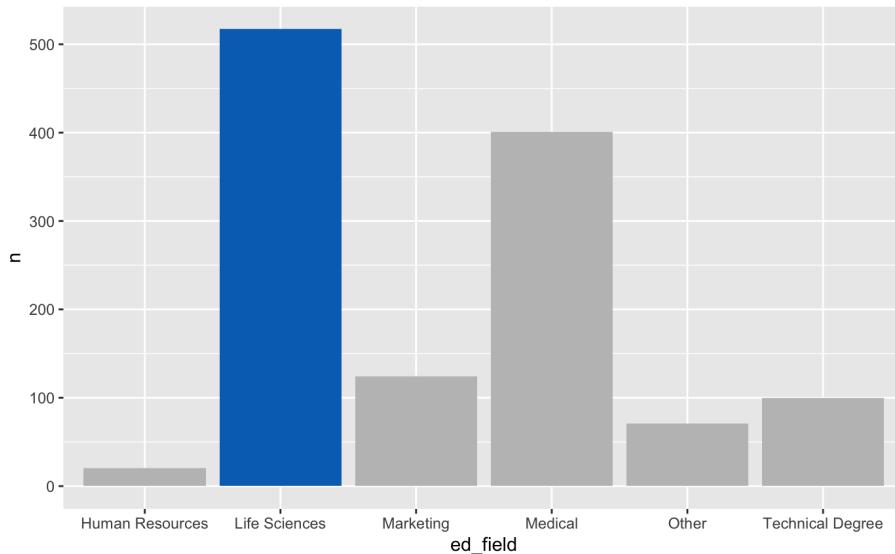


Figure 91: Step 3: Assign colors strategically

- **r** = space on the right of axis title
- **b** = space on the bottom of axis title
- **l** = space on the left of axis title

Let's create a margin of white space above the *x*-axis title and to the right of the *y*-axis title by defining the following parameters:

- `axis.title.x = ggplot2::element_text(margin = margin(t = 10, r = 0, b = 0, l = 0))`
- `axis.title.y = ggplot2::element_text(margin = margin(t = 0, r = 10, b = 0, l = 0))`

Step 5: Add Left-Justified Title

We can add a chart title via `labs(title = 'This is a chart title.')`. This title can be centered via `theme(plot.title = ggplot2::element_text(hjust = 0.5))`, left justified via `theme(plot.title = ggplot2::element_text(hjust = 0))`, or right justified via `theme(plot.title = ggplot2::element_text(hjust = 1))`.

It is a best practice to left justify titles since readers consume information beginning with the left side of the page (like reading a book), and left justifying

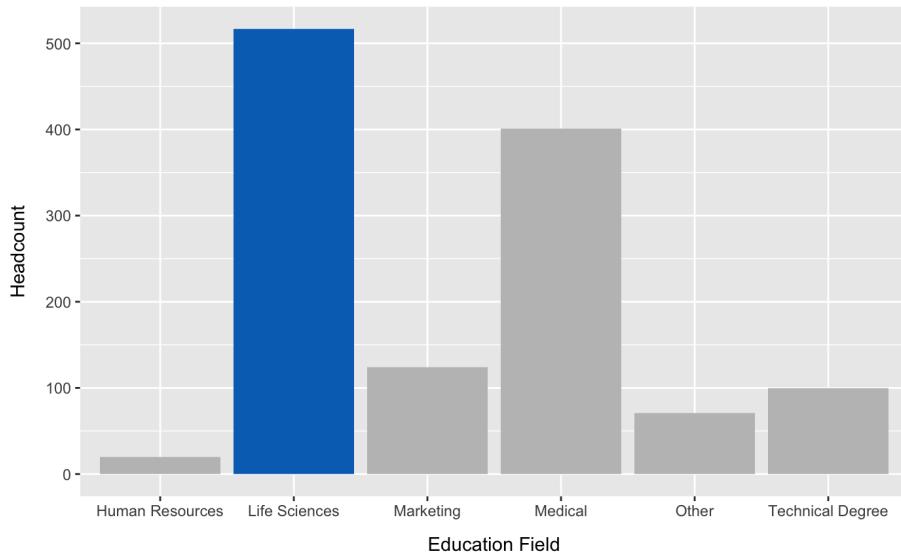


Figure 92: Step 4: Add mixed case axis titles

the title increases the probability that the audience will read the title and understand its purpose before engaging with the visual. Left justification is the default for `ggplot2` titles.

It is also a best practice to assign a descriptive title to charts to highlight the key message(s) we want to convey to the audience through the visual. For longer titles, the new line character `\n` can be used to break titles into multiple lines.

Step 6: Remove Background

The default grey background is a distraction from the data we need to take center stage. We can remove this background using `panel.background = element_blank()` to achieve a cleaner aesthetic and allow the bars in this chart to become more pronounced.

Step 7: Remove Axis Ticks

Axis ticks add noise to this visual and are not necessary to ascertain to which values along the axes the data align. Axis ticks can be removed for each axis independently with `axis.ticks.x = element_blank()` for the x-axis and `axis.ticks.y = element_blank()` for the y-axis.

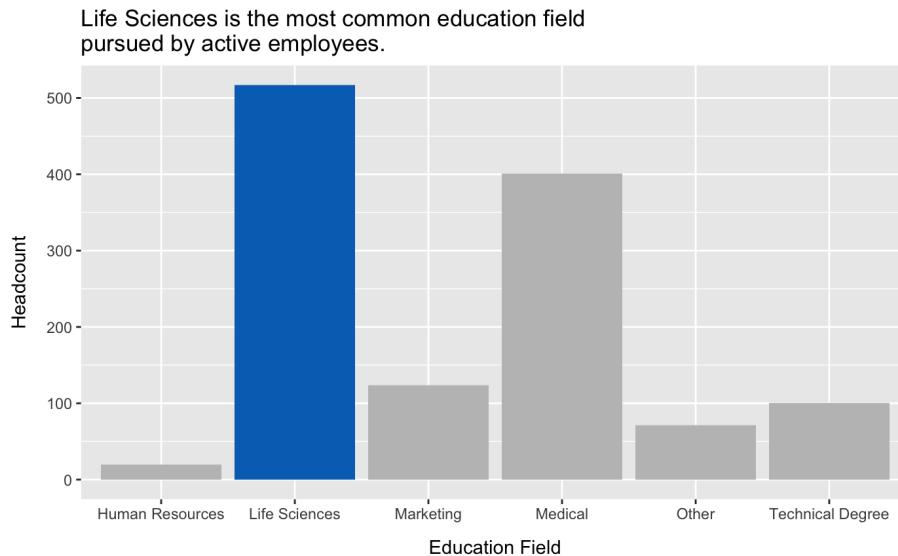


Figure 93: Step 5: Add title

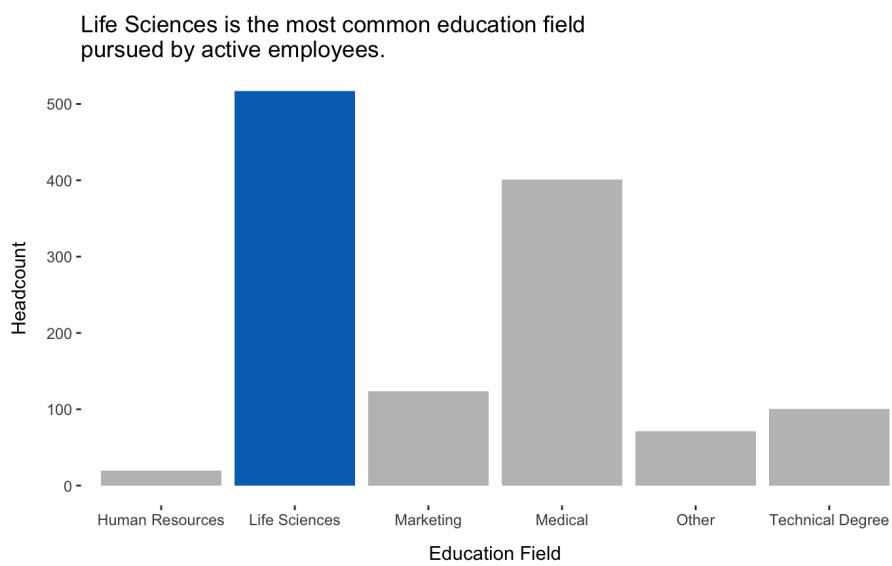


Figure 94: Step 6: Remove background

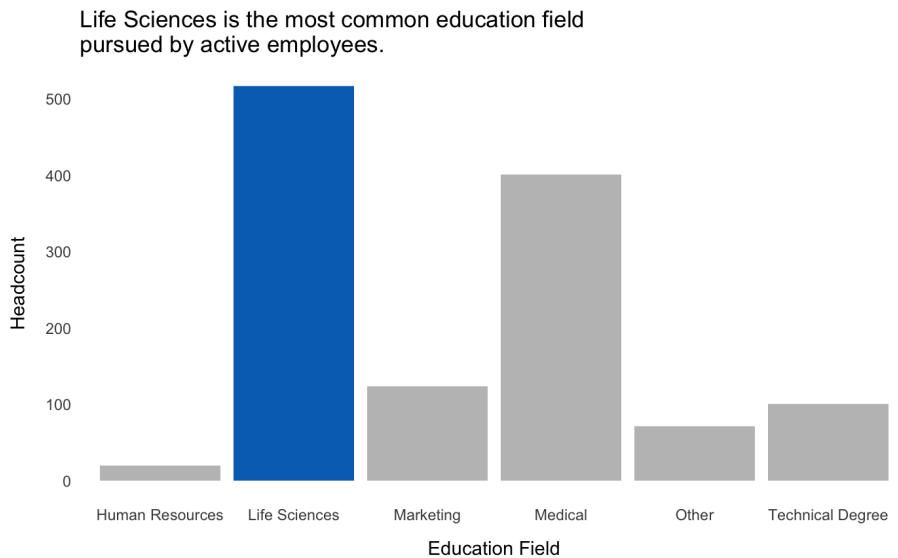


Figure 95: Step 7: Remove axis ticks

Step 8: Mute Titles

While the chart and axis titles are important for clarifying what information is represented in the visual, these should not be the focus. Just as we pushed the less important education categories to the background using grey text, we can mute the chart and axis titles with grey text to help draw attention to the data.

We can change the color of the chart title to light grey with `plot.title = ggplot2::element_text(colour = "#404040")`. The axis titles can be changed to the same color using `axis.title.x = ggplot2::element_text(colour = "#404040")` for the x-axis and `axis.title.y = ggplot2::element_text(colour = "#404040")` for the y-axis.

Step 9: Flip Axes

Flipping the coordinates of axes to convert the default vertical bar chart into a horizontal bar chart allows the audience to more easily scan down the right side of the visual to quickly identify and understand the relative frequencies of education categories.

Coordinates can be flipped using `coord_flip()`:

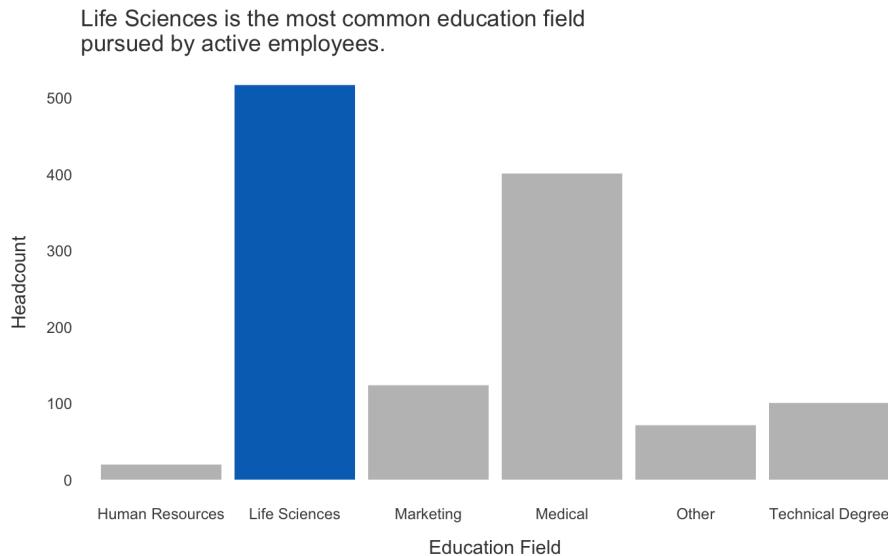


Figure 96: Step 8: Mute titles

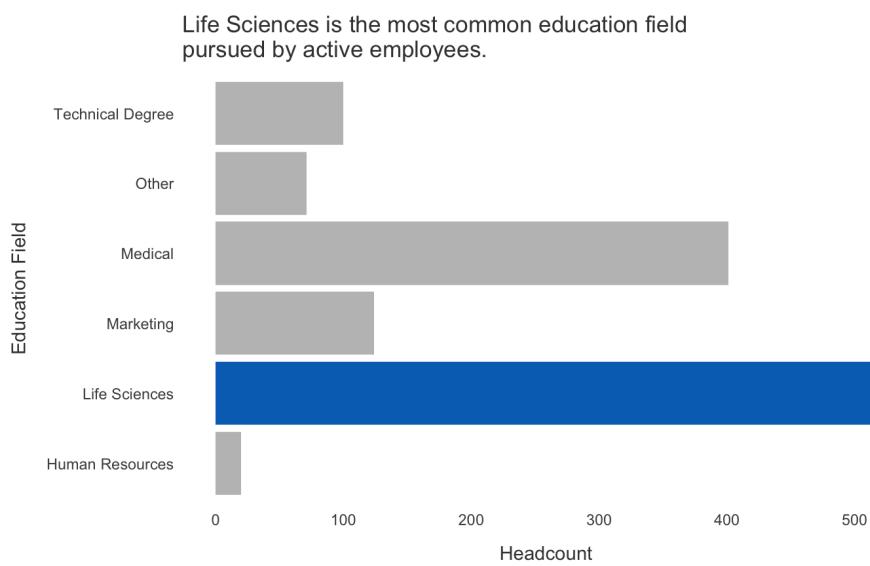


Figure 97: Step 9: Flip axis

Step 10: Sort Data

This visual can be further simplified by sorting the bars from highest to lowest value. With sorted bars, the audience can more easily ascertain the relative ranking of each education field.

We can pass `reorder(ed_field, n)` into the `aes()` function to sort the education field bars from highest to lowest *n*-count. If we needed to sort in the opposite direction, `reorder(ed_field, -n)` will reverse the sort direction.

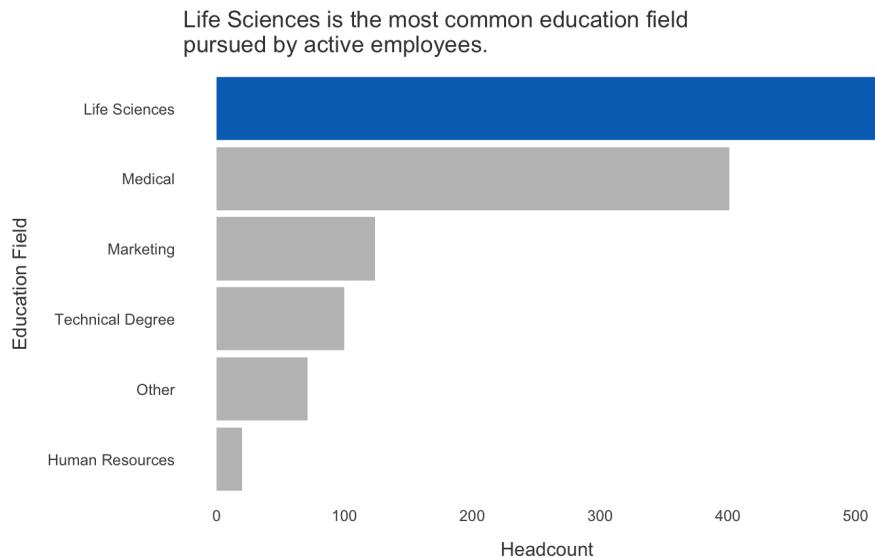


Figure 98: Step 10: Sort data

The final visual is a marked improvement over the design defaults in step 1.

Visualization Types

There are some highly advanced and interactive data visualizations that can be built using JavaScript (JS) libraries such as D3. With some exceptions, JS libraries are generally out of scope for this book. `ggplot2` is capable of building very elegant data visualizations, and this will be the tool used to implement most of the data visualizations in this chapter.

All visuals will have a set of common parameters for design aesthetics, consistent with the themes used to produce the enhanced chart design shown in Figure @ref(fig:pre-post-declutter). Therefore, each section will highlight the

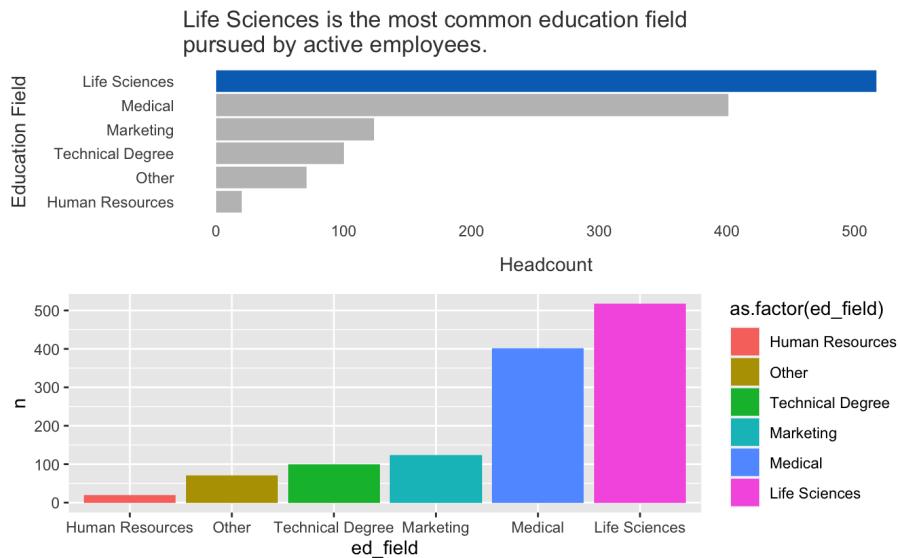


Figure 99: Enhanced design (top). Design defaults (bottom).

differences needed to achieve the respective data visualization. For production applications with many visualizations, it may be helpful to wrap common `ggplot2` design elements in a function to simplify chart building.

Tables

Tables are the most basic way to organize data. Since tables generally contain many metrics, they are usually better situated as reference material in the Appendix of a doc/deck or within a metric drill-through in dashboards rather than occupying prime real estate that should be leveraged strategically to focus the audience's attention on key messages.

A simple cube containing employee counts by department and tenure band will be constructed for demonstrating how to display tabular output.

```
# Append new tenure band column
employees$tenure_band <- dplyr::case_when(
  employees$org_tenure < 1 ~ "Under 1 Year",
  employees$org_tenure < 2.5 ~ "1-2 Years",
  employees$org_tenure < 5.5 ~ "3-5 Years",
  employees$org_tenure <= 10 ~ "6-10 Years",
  TRUE ~ "Over 10 Years"
)
```

```
# Store aggregate measures to cube
dept_tenure <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::group_by(dept, tenure_band) |>
  dplyr::summarise(cnt = dplyr::n())

# Specify ordered factor
dept_tenure$tenure_band <- ordered(dept_tenure$tenure_band,
  levels = c("Under 1 Year", "1-2 Years", "3-5 Years", "6-10
  Years", "Over 10 Years"))
```

The default display of a tibble (data frame produced by `dplyr`) is very basic:

```
# Display output using default settings
dept_tenure
```

```
## # A tibble: 14 x 3
## # Groups:   dept [3]
##   dept          tenure_band     cnt
##   <chr>        <ord>      <int>
## 1 Human Resources 1-2 Years     7
## 2 Human Resources 3-5 Years    19
## 3 Human Resources 6-10 Years    17
## 4 Human Resources Over 10 Years  8
## 5 Research & Development 1-2 Years 148
## 6 Research & Development 3-5 Years 262
## 7 Research & Development 6-10 Years 252
## 8 Research & Development Over 10 Years 148
## 9 Research & Development Under 1 Year 18
## 10 Sales           1-2 Years    57
## 11 Sales           3-5 Years    93
## 12 Sales           6-10 Years   124
## 13 Sales           Over 10 Years 70
## 14 Sales           Under 1 Year 10
```

While these data could easily be copied and pasted into presentation software for formatting, additional libraries exist in R for formatting tables. For example, R Markdown scripts can leverage the `DT` package to provide filtering, pagination, sorting, search, and other interactive features for HTML output. Field names can also be changed to proper case via the `dplyr::rename()` function.

```
# Assign proper case field names
dept_tenure_proper <- dept_tenure |>
```

```
rename('Department' = dept,
      'Tenure' = tenure_band,
      'Count' = cnt)
```

Department	Tenure	Count
Human Resources	1-2 Years	7
Human Resources	3-5 Years	19
Human Resources	6-10 Years	17
Human Resources	Over 10 Years	8
Research & Development	1-2 Years	148
Research & Development	3-5 Years	262
Research & Development	6-10 Years	252
Research & Development	Over 10 Years	148
Research & Development	Under 1 Year	18
Sales	1-2 Years	57

Showing 1 to 10 of 14 entries

Previous 1 2 Next

Figure 100: Data table

Heatmaps

Heatmaps use a shading scheme to highlight the relative magnitude of numbers in a tabular format.

The `geom_tile()` function can be used to build a heatmap with `ggplot2`. The range of colors can be specified via the `scale_fill_continuous(low = 'minimum value color', high = 'maximum value color')` function since the fill variable defined by `aes(fill = cnt)` is measured on a continuous scale:

This heatmap is excellent for focusing attention on the department + tenure segments with highest and lowest employee counts. However, if the specific employee counts are required, complimenting this heatmap with a basic table of metrics is a good option to avoid cluttering the heatmap with $3 * 5 = 15$ additional numbers.

Scatterplots

Scatterplots are useful for visualizing relationships between numeric variables.

Let's build a scatterplot to visualize the relationship between `work_exp` and `ytd_sales`, with a goal of focusing the audience's attention on salespeople whose

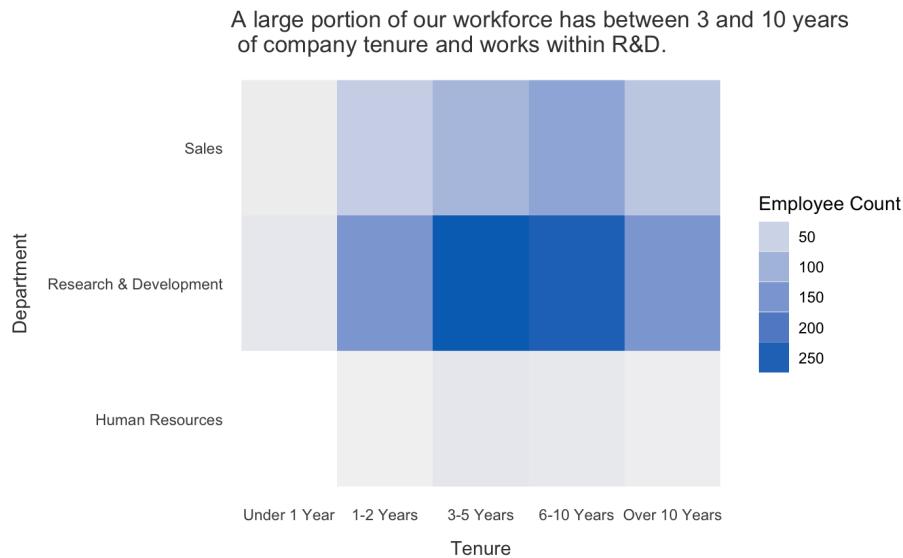


Figure 101: Heatmap showing the concentration of employees within departments and tenure bands

`ytd_sales` meet or exceed the full year sales quota of 150,000 USD. Let's first subset `employees` data to active salespeople and append a flag to indicate sales quota attainment for use in shading data points in the scatterplot.

```
# Subset df to active sales employees
sales <- subset(employees, dept == 'Sales' & active == 'Yes')

# Set quote attainment flag for data viz coloring
sales$quota_flg <- ifelse(sales$ytd_sales >= 150000, 1, 0)
```

The `geom_point()` function is used to create a scatterplot. The `scale_y_continuous()` function can be used in conjunction with the `scales` library to override the default scale for the y-axis (scientific notation) with more intuitive values.

While the basic scatterplot in Figure @ref(fig:scatterplot-1) is effective in visualizing the relationship between `work_exp` and `ytd_sales`, additional design elements are needed to highlight the data points that meet or exceed the full year sales quota of 150,000 USD.

The `color` argument for the `geom_point()` function is used to apply preattentive attributes to this visual by specifying the field used for conditional data point shading. We can define the color for each of the field's values using the `scale_color_manual()` function. Additionally, the `geom_hline()` function is used to add a dotted horizontal line at a specified position on the y-axis (at

The relationship between work experience and YTD sales is positive.

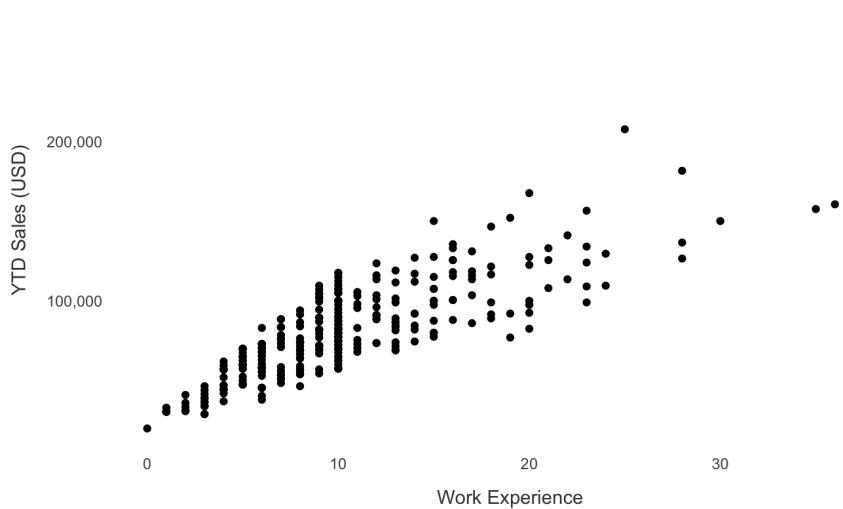


Figure 102: Scatterplot showing the relationship between work experience and YTD sales for active salespeople

150,000 USD), and annotation is added at a specified pair of x and y coordinates using the `annotate()` function.

Line Graphs

Line graphs are used for visualizing continuous data across time.

When visualizing trended data, it is important to avoid cumulative trends because they suggest an upward trajectory (positive slope) even when the corresponding non-cumulative metrics indicate a declining trend.

Consider the following data frame with decreasing hire counts by year:

```
# Print hires df
print(hires_dat[order(hires_dat$year, decreasing = FALSE), ])

##   year hires cum_hires
## 4 2019  1020      980
## 3 2020  1010     1970
## 2 2021   990     2980
## 1 2022   980     4000
```

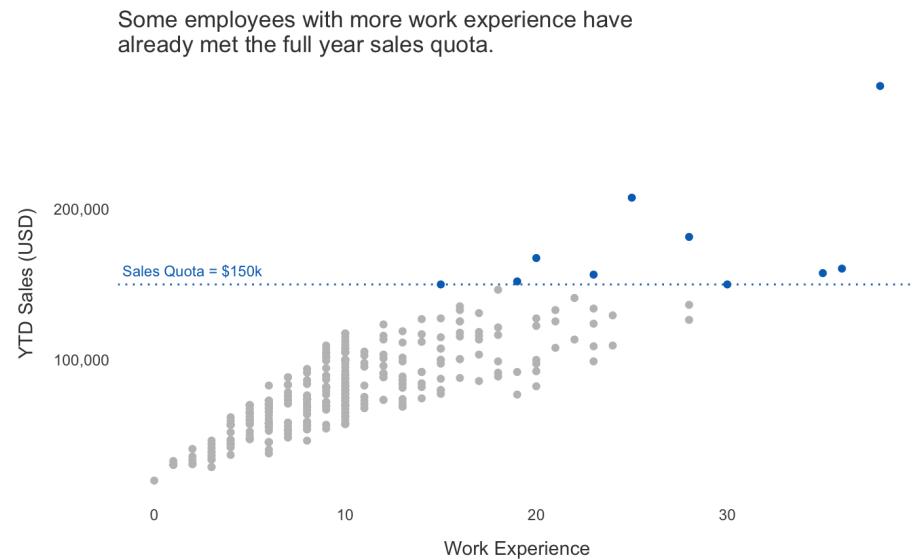


Figure 103: Scatterplot with data points colored relative to the full year sales quota of USD 150k

Figure @ref(fig:cum-hires-trend) juxtaposes a trended line chart with hires by year (left) against a cumulative version of the same (right). Since hires at each year are additive in the cumulative line chart, the slope is positive and misleading.

Figure @ref(fig:cum-iphone-sales) shows a deceptive cumulative graph of iPhone sales. Superimposing a bar chart with non-cumulative quarterly sales atop an overall cumulative trend that occupies far more real estate and dominates the screen can lend to the audience drawing invalid conclusions from the data.

Single Series

The most basic type of line graph is a **single series line graph**, which reflects a trend for a single group.

Let's generate some attrition data for illustrating various types of line graphs:

```
# Set seed for reproducibility
set.seed(1234)

# Create data
months = 1:24
eng_rt = 5 - runif(1, 2.7, 2.9) + 2.41*months - .41*months^2 +
  -.02*months^3
fin_rt = runif(1, 5, 8) - 6.97 + 15*months - .53*months^2
```

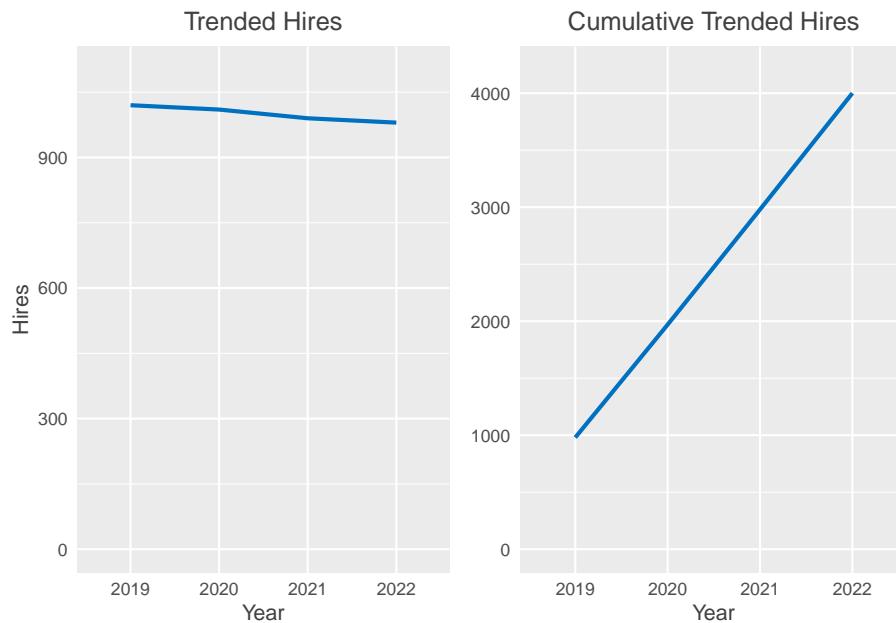


Figure 104: Hires by year (left) and cumulative hires by year (right)

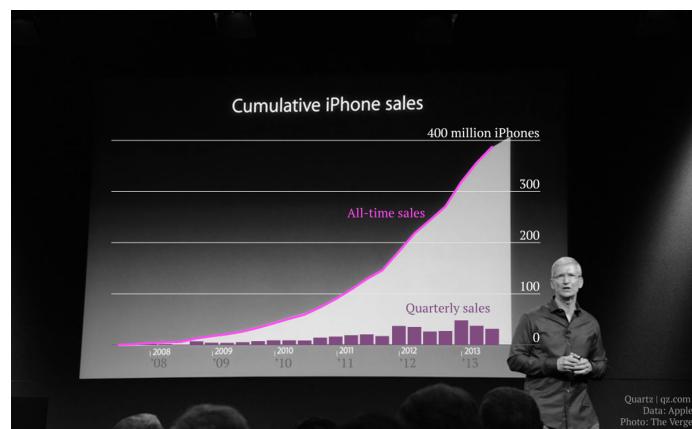


Figure 105: Deceptive cumulative sales trend

```

ppl_rt = 3 - runif(1, 5, 8) - 6.97 + 12*months - .4*months^2
prd_rt = runif(1, 5, 8) - 6.97 + 13*months - .53*months^2

# Combine dimensions and metrics within df
attrition_dat <- data.frame(month = rep(months, 4),
                               dept = c(rep('Engineering',
                                          → length(months)),
                                         rep('Finance',
                                         → length(months)),
                                         rep('People',
                                         → length(months)),
                                         rep('Product',
                                         → length(months))),
                               rate = c(eng_rt,
                                       fin_rt,
                                       ppl_rt,
                                       prd_rt))

```

Line graphs can be constructed using the `geom_line()` function in `ggplot2`. Figure @ref(fig:single-line-graph) shows a single series line graph built in `ggplot2`.

Engineering attrition has increased over the last 24 months, largely due to an exponential spike in the last year.

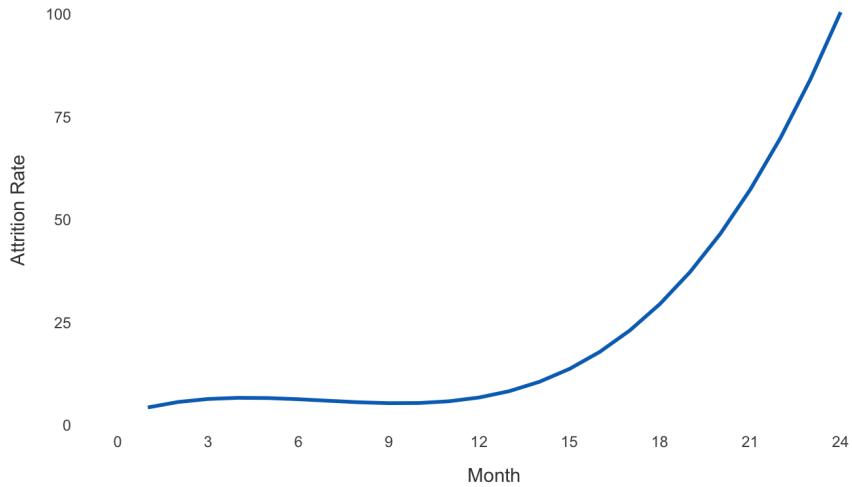


Figure 106: Single series line graph

Two Series

A **two series line graph** reflects trends for two groups, as shown in Figure

@ref(fig:dbl-line-graph).

The `aes(color = group)` parameter defines the group by which lines are stratified.

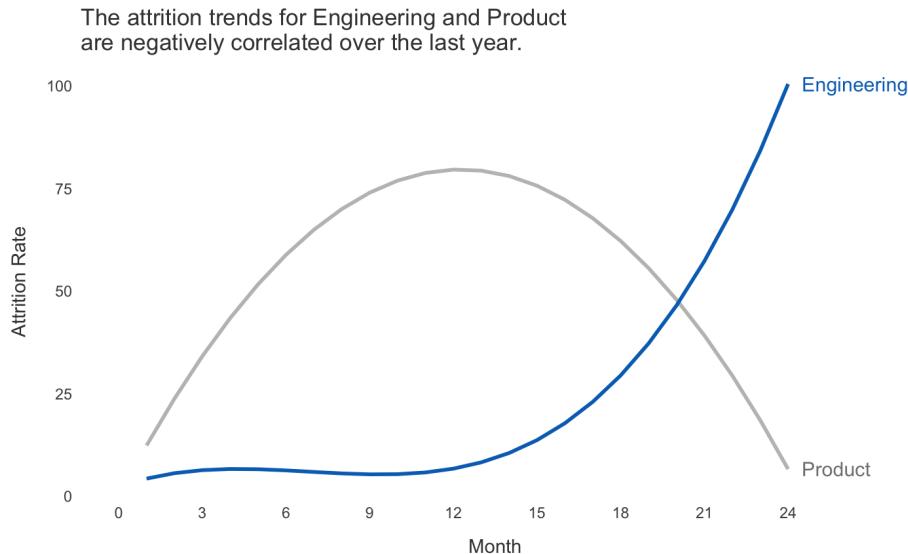


Figure 107: Two series line graph

Multiple Series

A **multiple series line graph** reflects trends for three or more groups.

Note the preattentive attributes applied in Figure @ref(fig:multi-line-graph). If the focus is on how a single group (Engineering in this case) compares to two or more other groups, there is no need to differentiate the other groups with respect to color – only label.

Slopegraphs

Slopegraphs are helpful in illustrating relative changes between two points in time.

Common applications for slopegraphs in people analytics include survey variable score changes between two time periods, pre/post changes to outcome variables in an experimental context, and various metrics (e.g., headcount, TTM attrition) for which changes need to be evaluated MoM, QoQ, or YoY when data points between the start and end points are unimportant.

The data structure needed to support a slopegraph is consistent with a line graph. To illustrate how to construct a slopegraph in R, a data frame will

The attrition trend for Engineering is quite different relative to attrition patterns for other departments.

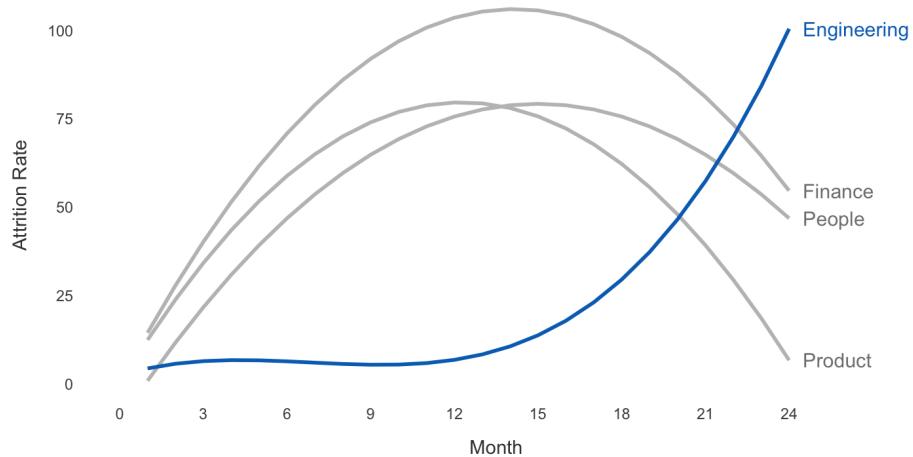


Figure 108: Multiple series line graph

be constructed that holds engagement scores for two points in time for both a treatment and control group.

```
# Build data frame with YoY headcount metrics by department
prepost_scores <- data.frame(date = c(rep('Time 1', 2), rep('Time
  ~ 2', 2)),
                                group = rep(c('Treatment',
  ~ 'Control'), 2),
                                score = c(50, 53, 75, 56))
```

In `ggplot2`, a slopegraph is simply a line graph with only two values on the x -axis and some additional formatting including annotations and y -axis removal. Figure @ref(fig:slopegraph) is a slopegraph comparing the engagement score changes for treatment and control groups over an observation period, with preattentive attributes applied to focus attention on the treatment group.

Bar Charts

Bar charts are used to display categorical data. Four common types of bar charts are *vertical*, *horizontal*, *stacked*, and *bidirectional*.

Vertical

A more favorable change in engagement was observed for the treatment group relative to the control group.

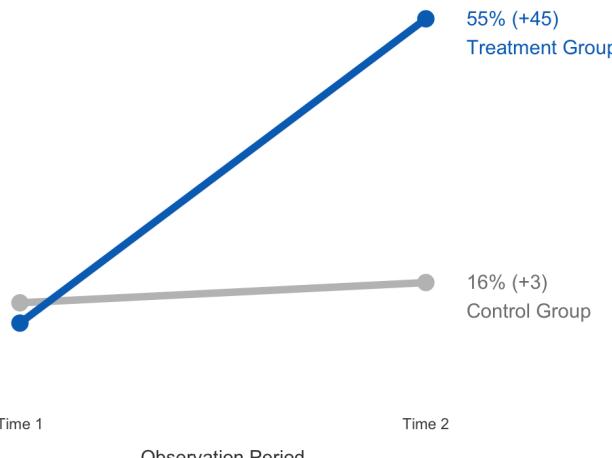


Figure 109: Slopegraph

A **vertical bar chart** is the most basic and pervasive method of visualizing categorical data. Like line charts, bar charts can be single series, two series, or multiple series based on the data that need to be displayed.

To demonstrate how to build bar charts, departmental engagement data will be simulated with some rank variables to support preattentive attributes.

```
### Data Prep ###

# Set seed for reproducibility
set.seed(1234)

# Generate favorability distributions
fav_pct <- round(runif(7, 20, 45), 0)
neu_pct <- round(runif(7, 20, 45), 0)
unfav_pct <- 100 - fav_pct - neu_pct
scores <- c(fav_pct, neu_pct, unfav_pct)

# Average top box (favorable) score
topbox_avg <- round(mean(fav_pct), 0)

# Build data frame with YoY headcount metrics by department
engagement_scores <- data.frame(dept = rep(c('Engineering',
  'Finance', 'Legal', 'Marketing', 'People', 'Product',
  'Sales'), 3),
```

```

    favorability = c(rep('Favorable',
    ↳ 7), rep('Neutral', 7),
    ↳ rep('Unfavorable', 7)),
    pct = scores)

# Rank departments by top box score to support sorting
dept_rank <- engagement_scores |>
  dplyr::filter(favorability == 'Favorable') |>
  dplyr::arrange(desc(pct)) |>
  dplyr::mutate(rank = dense_rank(desc(pct))) |>
  dplyr::select(dept, rank)

# Flag top department records in df to support conditional
← coloring
engagement_scores$top_score = ifelse(engagement_scores$dept ==
← dept_rank[1, 'dept'], 1, 0)

# Add department rank based on top box scores
engagement_scores <- left_join(engagement_scores, dept_rank, by =
← "dept")

```

The `geom_bar()` function can be leveraged to construct a bar chart using `ggplot2`. The `geom_hline()` and `annotate()` functions can be added to include a reference line with the average top box score. Preattentive attributes can also be applied by setting the derived `top_score` variable as the fill parameter in the `aes()` function.

Horizontal

The **horizontal bar chart** is a horizontal version of the vertical bar chart, and it tends to be easier to read.

The vertical bar chart can be converted to a horizontal bar chart by adding `coord_flip()` to swap the axes.

Stacked

The **stacked bar chart** is useful for illustrating the relative contribution of subcomponents to a whole. In a people analytics setting, a 100% stacked bar chart is an excellent tool for visualizing the favorability distribution across survey items and various categorical dimensions (e.g., departments, locations, job profiles).

The only adjustment needed to build a stacked area chart is to specify the variable containing the favorability categories as an ordered factor for the `fill` parameter. Colors can be specified for each favorability category via the `scale_fill_manual()` parameter.

The People team is leading in department-level engagement.

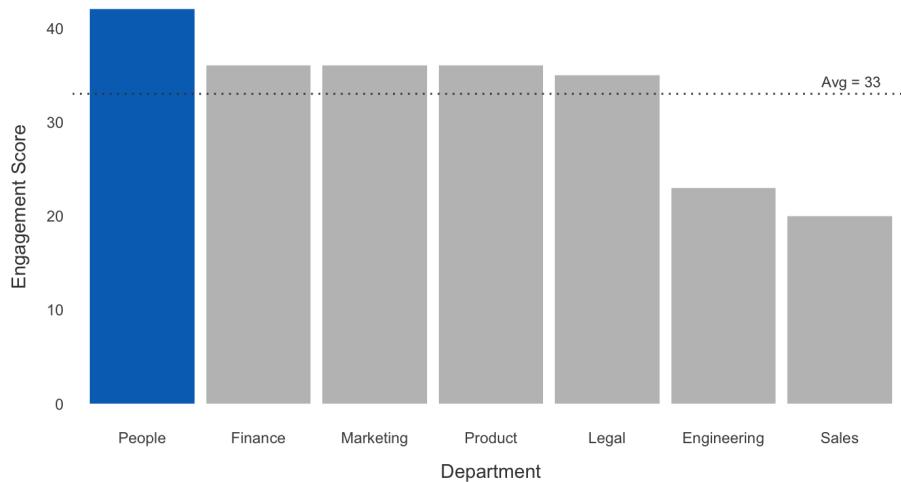


Figure 110: Vertical bar chart

The People team is leading in department-level engagement.

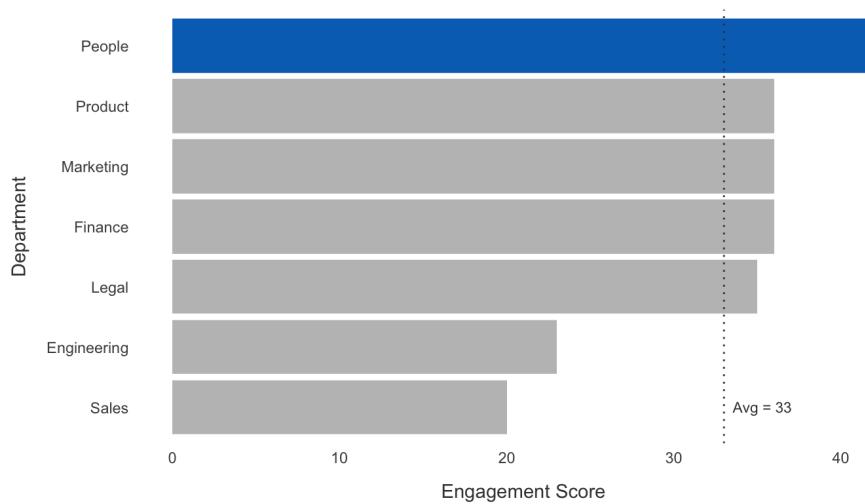


Figure 111: Horizontal bar chart

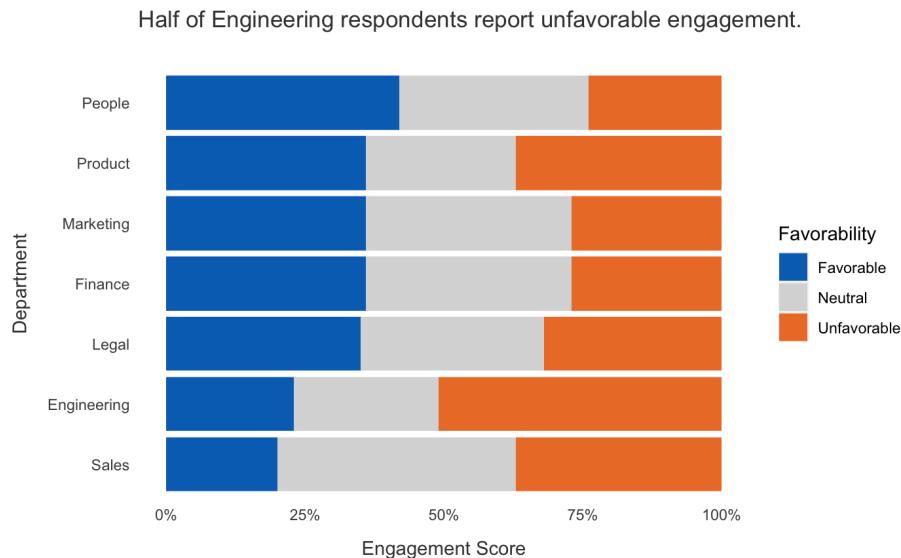


Figure 112: 100 percent stacked bar chart

Bidirectional

The **bidirectional bar chart** is an effective visual for comparing two metrics side-by-side across values of a categorical variable. The bidirectional bar chart is sometimes referred to as a **divergent bar chart**, **back-to-back bar chart**, or **mirror bar chart**.

Let's illustrate two levers of departmental headcount change – hires and terminations – using a bidirectional bar chart. While the visualization code is consistent with that of horizontal bar charts, the data need to be transformed such that losses (terms) are negative numbers and gains (hires) are positive numbers.

```
# Set seed for reproducibility
set.seed(1234)

# Build data frame with hire and term metrics by department
hires_terms <- data.frame(dept = c('Engineering', 'Finance',
  ~ 'Legal', 'Marketing', 'People', 'Product', 'Sales'),
  ~ metric = rep(c('Hires', 'Terms'), 7),
  ~ cnt = round(runif(14, 5, 150), 0))

# Append transformed count column to support bidirectional bar
# charts
hires_terms$cnt_trans <- ifelse(hires_terms$metric == 'Terms', 0
  ~ - hires_terms$cnt, hires_terms$cnt)
```

Using the `cnt_trans` field containing both negative and positive integers, we can visualize net changes in departmental headcount.

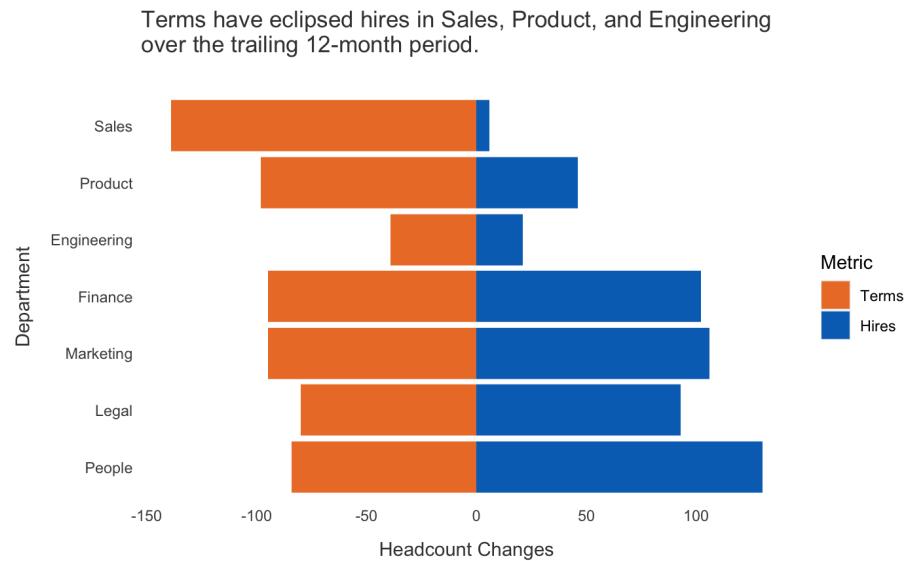


Figure 113: Bidirectional bar chart

Combination Charts

Combination charts display data using different types of visualizations within the same chart.

In a people analytics context, we may wish to highlight differences between regrettable (bad) and non-regrettable (good) turnover trends relative to total voluntary turnover rates. It is usually more difficult to compare the magnitude of regrettable and non-regrettable rates across time using a stacked bar chart, and a combination chart is often a more intuitive method of presenting this information.

As illustrated in Figure @ref(fig:combo-chart), we can leverage a two-series line chart for monthly regrettable and non-regrettable rates relative to the total voluntary turnover rate visualized with a light grey vertical bar chart in the background.

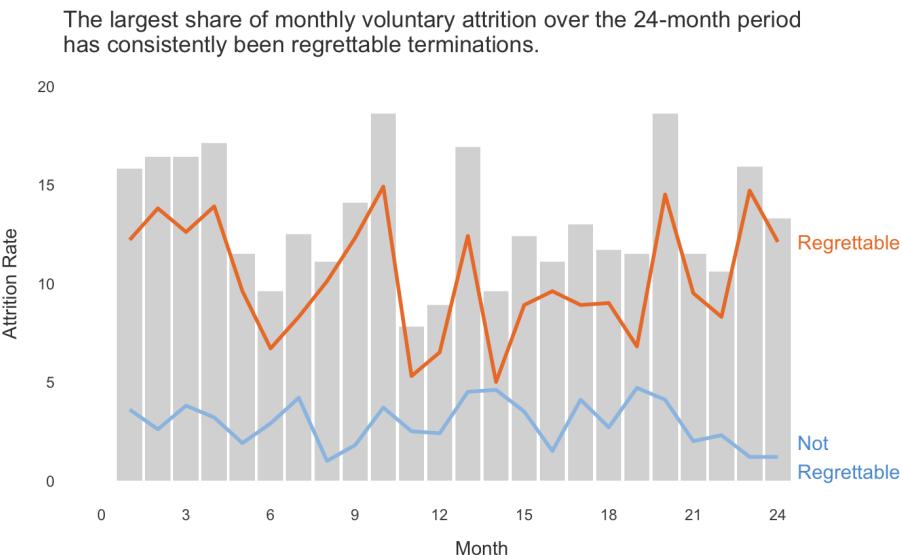


Figure 114: Combination chart

Waterfall Charts

Waterfall charts are alternatives to stacked bar charts that aid in understanding events between two points in time that explain a change in a starting and ending period value. Explaining drivers of headcount changes over time is a common use case for waterfall charts in people analytics.

Building a waterfall chart requires a bit more data prep relative to the data structure requirements for a stacked bar chart. Unlike a bar chart, a waterfall chart needs a start and end value for each event type so that each bar beyond the first begins where the previous bar ends.

```
# Generate headcount data with id field to define bar order
hc_dat <- data.frame(id = 1:6,
  event = c("Starting HC", "Hires", "Transfers
    ↪ In", "Transfers Out", "Exits", "Ending
    ↪ HC"),
  type = c("Headcount", "Growth", "Growth",
    ↪ "Loss", "Loss", "Headcount"),
  count = c(100, 50, 10, -10, -20, 130),
  start = NA,
  end = NA)

# Define start and end values to support waterfall chart
hc_dat$end <- cumsum(hc_dat$count)
```

```

hc_dat$end <- c(head(hc_dat$end, -1), 0)
hc_dat$start <- c(0, head(hc_dat$end, -1))

# Swap start/end values for last record (Ending HC)
hc_dat[nrow(hc_dat), "end"] <- hc_dat[nrow(hc_dat), "start"]
hc_dat[nrow(hc_dat), "start"] <- 0

```

With data properly structured, we can use the `geom_rect()` function to build the waterfall chart. Strategies such as labeling only the Beginning HC and Ending HC reduces clutter and lends to a cleaner design aesthetic.

Hires and inbound transfers have outpaced outbound transfers and exits in Engineering, leading to a net YTD increase in headcount.

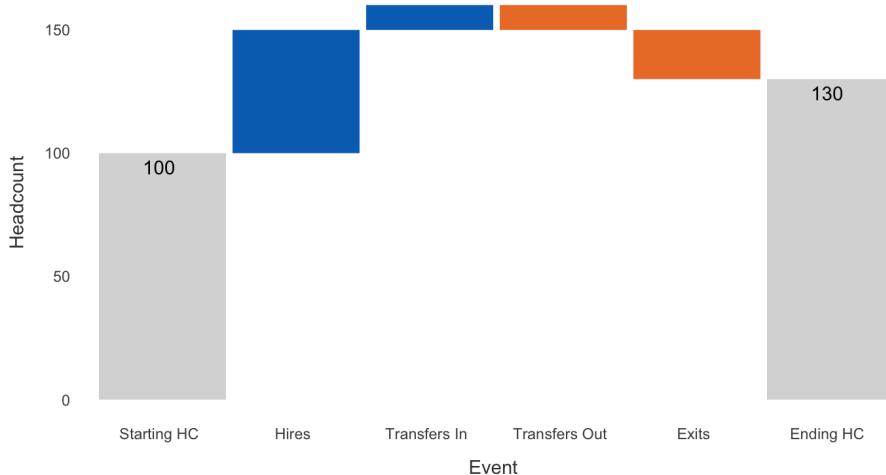


Figure 115: Waterfall chart

Waffle Charts

Waffle charts, also known as **square area charts**, are well-suited for illustrating parts of a whole.

A common application in people analytics is illustrating candidate movement through the recruiting funnel. It is often helpful to visualize a normalized applicant pool (e.g., per 100 applicants) to identify where bottlenecks exist across funnel stages and how pass-through rates compare across business areas.

Waffle charts require some data prep to visualize:

```

# Create df with TA funnel metrics
ta_dat <- data.frame(stage = c("Apply", "Phone Screen",
  ↪ "Interview", "Offer Extend", "Offer Accept"),
  ↪ cnt = c(60, 20, 10, 6, 4))

# Set depth of waffle chart (# of y-axis rows)
depth <- 10

# Each observation needs an x and y coordinate, and y needs to be
→ specified first for a waffle chart with horizontal
→ accumulation
waffle_dat <- expand.grid(y = 1:depth,
  ↪ x = seq_len(ceiling(sum(ta_dat$cnt) /
  ↪ depth)))

# Expand the applicant counts into a vector of stage labels
stages <- rep(ta_dat$stage, ta_dat$cnt)

# Integrate stages and fill any extra tiles with NA
waffle_dat$stage <- c(stages, rep(NA, nrow(waffle_dat) -
  ↪ length(stages)))

```

With this data structure, a waffle chart can be produced using the `geom_tile()` function from `ggplot2`.

The waffle chart in Figure @ref(fig:waffle-chart) illustrates how candidates for a set of filled requisitions move from application through the phone screen, interview, offer extend, and offer acceptance stages of the recruiting lifecycle. Based on this chart, 60% of applicants are rejected without a conversation, 40% receive a phone screen, 20% land an interview, 10% receive an offer, and 4% accept an offer.

Sankey Diagrams

Sankey diagrams, or alternatives such as **chord diagrams** and **alluvial plots**, are flow diagrams that are particularly effective in depicting a many-to-many relationship between two domains.

Sankey diagrams have many applications in people analytics. For example, sankeys may be used to understand internal transfers over a period of time (i.e., inflows and outflows among departments) or recruiting sources by which employees in various departments have been hired. To demonstrate how to implement a sankey diagram, let's use the `sankeyNetwork()` function from the `networkD3` library to visualize employee transfers between departments. D3 is an advanced JavaScript framework for creating interactive visualizations, and the `networkD3`

We select 10 in every 100 applicants.
Our offer acceptance rate (OAR) is 40%.

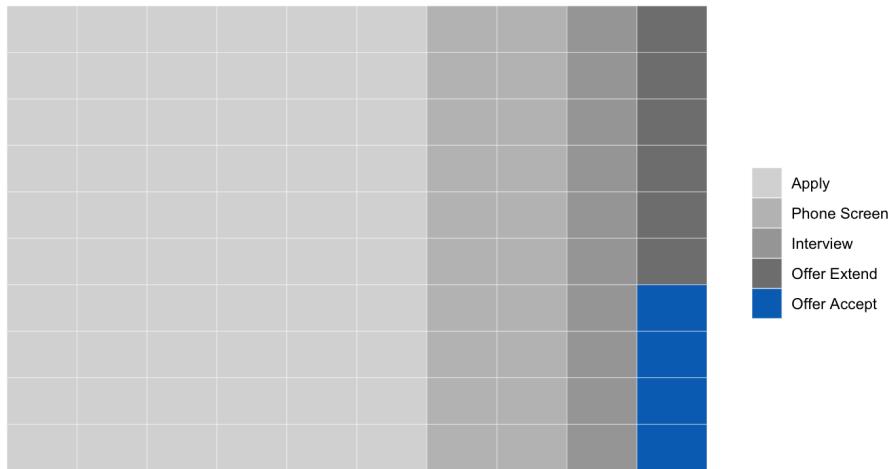


Figure 116: Waffle chart visualizing pass-through rates across recruitment stages per 100 job applicants

library provides an easy interface for constructing sankey diagrams with interactive components. Building a sankey diagram using `networkD3` requires data to be structured within two data frames:

- **nodes**: Defines the source and destination node names (i.e., the departments employees transfer into and out of)
- **links**: Connections between pairs of source and destination nodes using an index beginning at 0 to represent the corresponding node from the nodes data frame

```
# Set seed for reproducibility
set.seed(1234)

# Create nodes df
nodes <- data.frame(name = c('Engineering', 'Finance', 'Legal',
                           'Marketing', 'People', 'Product', 'Sales', # source
                           'Engineering', 'Finance', 'Legal',
                           'Marketing', 'People',
                           'Product', 'Sales')) # destination

# Create links df
links <- expand.grid(source = 0:6, target = 7:13)
```

```
# Append employee transfer counts per department pair to links df
links$value <- ifelse(links$source == links$target-7,
  round(rnorm(1000, 150, 50), 0), round(rnorm(1000, 30, 5), 0))
```

With data properly structured within the `nodes` and `links` data frames, these data can be passed into the `sankeyNetwork()` function to construct the sankey diagram. As illustrated in Figure @ref(fig:sankey-1), hovering over a connection displays the count of employee transfers between the corresponding departments ($n = 222$ moves within Product).

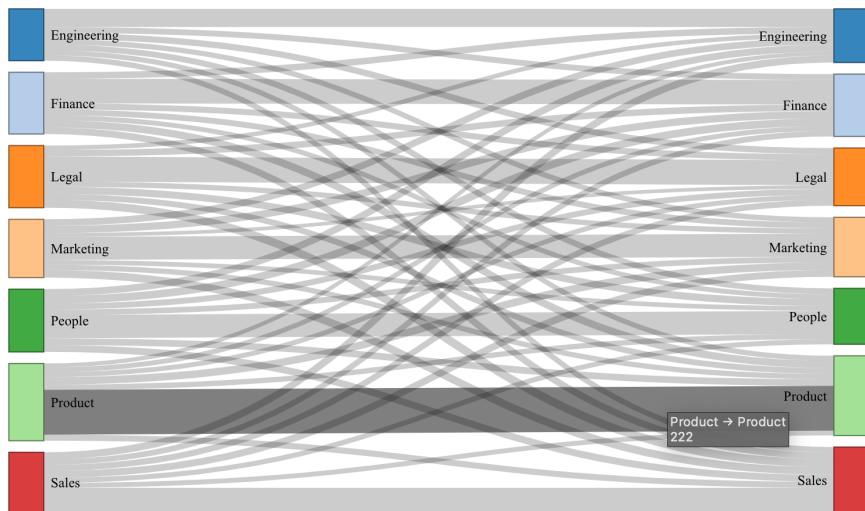


Figure 117: Sankey diagram showing employee transfers between departments with default colors

Though it can quickly become noisy as the number of nodes increases, coloring connections based on the source departments from which each group flows can be helpful in tracing the employee flow to the various destination departments.

A department variable needs to be defined and added to the `links` data frame in order to color connections based on the source department (transfers out). The `NodeGroup` and `LinkGroup` parameters in the call to the `sankeyNetwork()` function define the column in the `nodes` and `links` data frames, respectively, that specify the group values by which the nodes and links should be colored.

```
# Append source department variable for colored connections
links$dept <- dplyr::case_when(
  links$source == 0 ~ "Engineering",
```

```

links$source == 1 ~ "Finance",
links$source == 2 ~ "Legal",
links$source == 3 ~ "Marketing",
links$source == 4 ~ "People",
links$source == 5 ~ "Product",
links$source == 6 ~ "Sales",
TRUE ~ "NA"
)

```

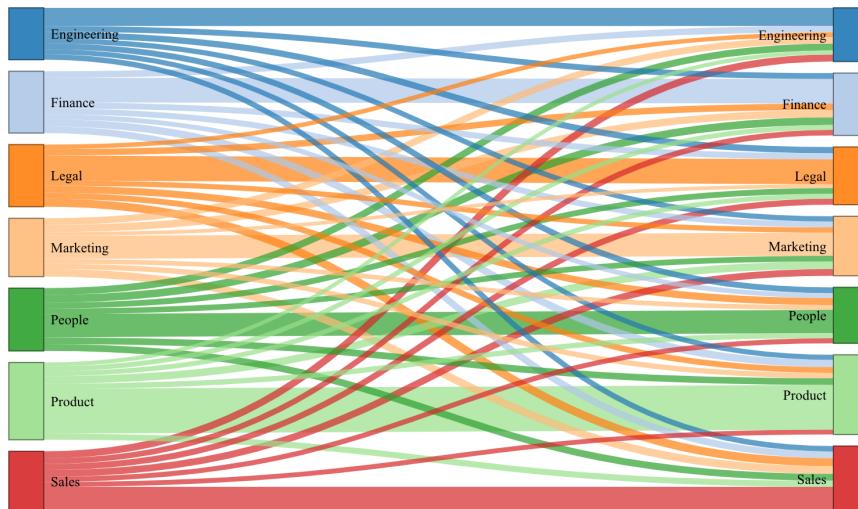


Figure 118: Sankey diagram showing employee transfers between departments with connections shaded based on a source department color scheme

We can add another grouping variable to support coloring intradepartmental (within) moves differently from interdepartmental (outside) moves.

```

# Append within/outside department transfer variable for colored
# connections
links$wiout_dept <- dplyr::case_when(
  (links$source == 0 & links$target == 7) | (links$source == 1 &
  links$target == 8) | (links$source == 2 & links$target == 9) |
  (links$source == 3 & links$target == 10) | (links$source ==
  4 & links$target == 11) | (links$source == 5 & links$target ==
  12) | (links$source == 6 & links$target == 13) ~ "Within",
  TRUE ~ "Outside"
)

```

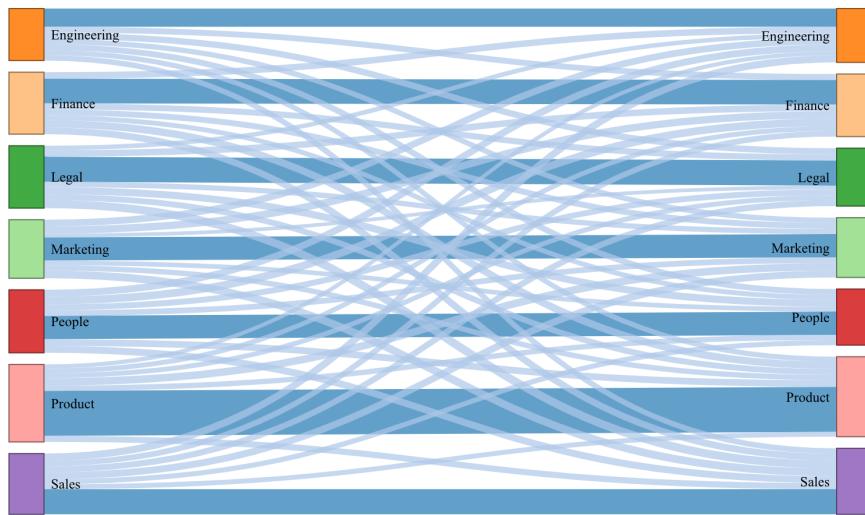


Figure 119: Sankey diagram showing employee transfers between departments with connections shaded based on an intradepartmental vs. interdepartmental color scheme

If the focus is on a single department's transfers, preattentive attributes can be applied to help focus the audience's attention and move the less important information to the background to reduce clutter.

To facilitate this, a grouping variable can be defined to differentiate a single department, such as Product, from remaining departments. The color of each group can then be specified using valid D3 code: `d3.scaleOrdinal(["group_1_color", "group_2_color"])`:

```
# Append dichotomous product/other indicator to links and nodes
→ data frames
links$prod <- ifelse(links$dept == 'Product', 'Product', 'Other')
nodes$prod <- ifelse(nodes$name == 'Product', 'Product', 'Other')
```

Pie Charts

Pie charts are rarely an effective way to visualize data, and this book does not generally endorse them. Pie charts can be appropriate in cases where there are two or three mutually exclusive and collectively exhaustive groups and we need to visualize the relative contribution of each to the whole. However, it quickly becomes difficult to ascertain relative size beyond a few groups with a pie chart,

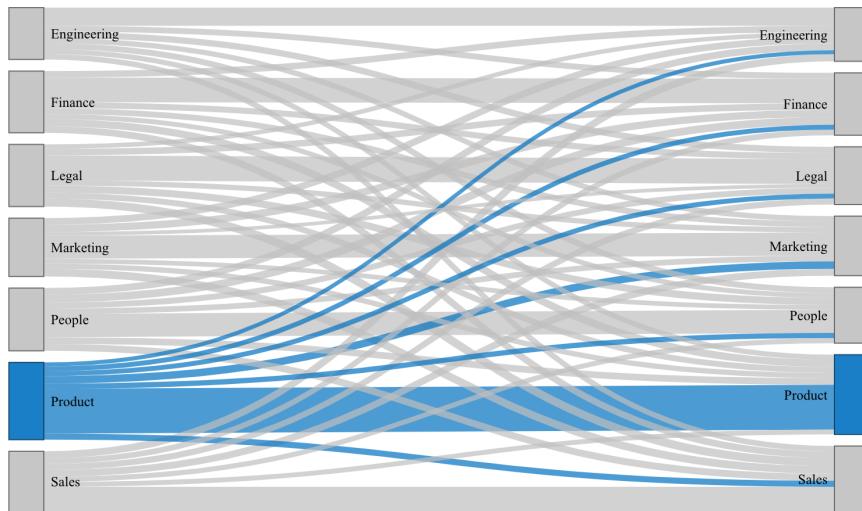


Figure 120: Sankey diagram with preattentive attributes to highlight employee transfers out of the Product department

and labeling many slices adds a lot of clutter and noise. Proportions should always sum to 1 so if data are not mutually exclusive parts of a whole, a pie chart is not appropriate.

Figure @ref(fig:bad-donut-chart) shows a donut chart – which is simply a pie chart with the center cut out – that is intended to show Kane Williamson’s outsized contribution to New Zealand’s cricket runs. However, top scorers from other countries are also included in this visual, so the metrics do not sum to 100%. Pie and donut charts are not appropriate for these data.

There is no specific `geom()` function for building pie charts using `ggplot2`. Therefore, we need to create a bar chart and make it circular by adding a `coord_polar("y", start = 0)` transformation. The `coord_polar()` transformation complicates the positioning of labels, but we can add a `position = position_stack(vjust = 0.5)` argument within the `geom_text()` function to easily achieve metric centering within the respective category. We can also easily improve the aesthetics with the `theme_void()` function, which removes the default background, grid, and labels.

The data need to be structured consistent with the requirements for a bar chart. A basic data cube with descriptive statistics by gender category will simplify the construction of the pie chart in `ggplot2`:

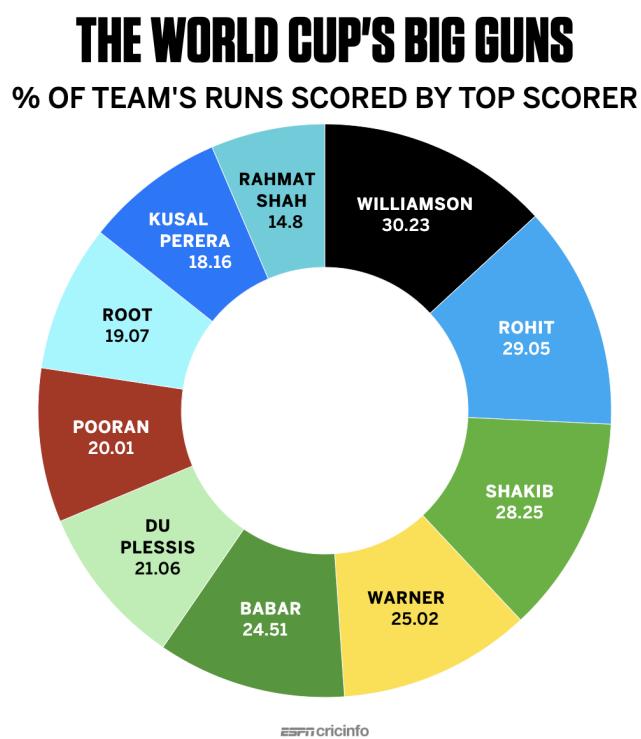


Figure 121: Improper use of a donut chart

```
# Create data cube for gender representation among active
→ employees
smmry_gender <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::group_by(gender) |>
  dplyr::summarise(cnt = dplyr::n()) |>
  dplyr::mutate(pct = round(cnt / sum(cnt) * 100,
← 1)) |>
  dplyr::arrange(desc(pct))
```

Nearly 60 percent of active employees identify as male.

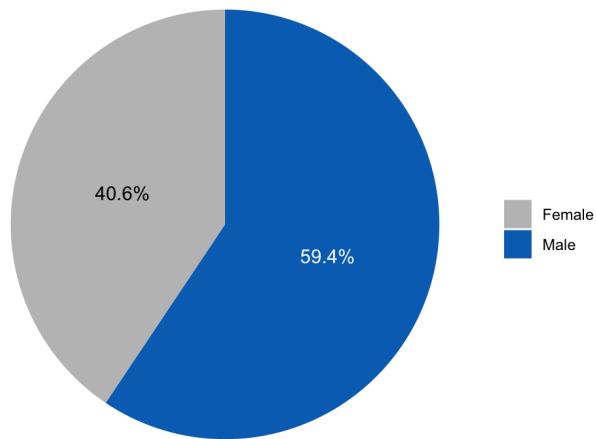


Figure 122: Pie chart showing gender representation for active employees

3D Visuals

An element of uncertainty is inherent in analytics, which is why this book avoids the use of categorical terms such as *always* and *never*. However, an exception will be made for this section. 3D visualizations are *never* appropriate. 3D visuals often misrepresent the data they are intended to visualize and are unnecessarily difficult to interpret.

Figure @ref(fig:3d-pie-chart) shows the distribution of headcount across locations using a 3D pie chart. Based on this visual, it may not be apparent that Los Angeles (25%) is considerably larger than New York (19%). The misleading perspective of this tilted visual causes locations on the back side of the pie chart to appear smaller relative to locations on the front side.

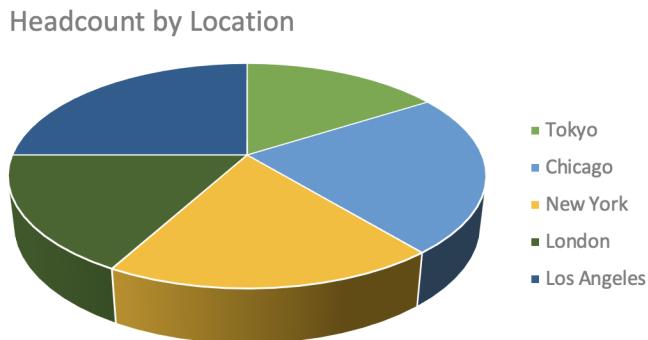


Figure 123: Headcount by location misrepresented with a 3D pie chart

A 3D bar chart is not an improvement over a 3D pie chart. As illustrated in Figure @ref(fig:3d-bar-chart), bars do not appear to align with the actual corresponding values (e.g., Los Angeles = 25%). This is because software determines the height of bars in a 3D bar chart using an invisible tangent plane to intersect the bars at the correct y-axis location.



Figure 124: Headcount by location visualized with a misleading 3D bar chart

As shown in Figure @ref(fig:3d-bar-chart_conv), by leveraging a horizontal bar chart with labeled bars and preattentive attributes, it becomes easy to understand relative headcount across locations.

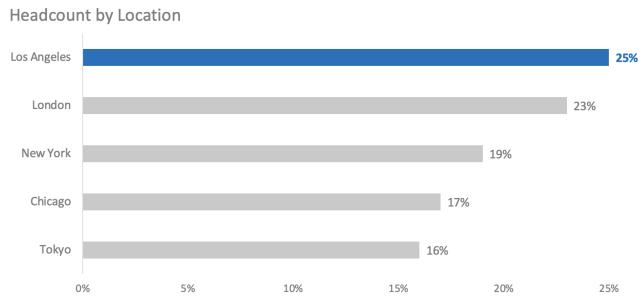


Figure 125: Headcount by location properly visualized with a horizontal bar chart

Elegant Data Visualization

This chapter has covered the fundamentals of effective data visualization. Applying these best practices will elevate impact when presenting results of analyses to stakeholders.

Figure @ref(fig:good-viz) is an example of an excellent data visualization that brings together many of the design elements promoted in this chapter. This visual shows seasonally adjusted jobless claims over a period of two decades to highlight the significance of the Coronavirus's impact on the labor market using the following design elements:

- Larger and darker text for the headline
- Smaller and lighter text for the commentary
- Grey axis labels and chart annotations to avoid competing with *current* statistics
- Low-profile dotted horizontal lines to help connect bars across the long *x*-axis to their corresponding *y*-axis values
- Orange coloring to represent jobless claims (an **unfavorable** event)
- Creative use of a bar chart resembling an area chart to represent the weekly interval for jobless claims over the 20-year period
- Two helpful reference points to provide perspective: (1) 20-year weekly average; and (2) cumulative jobless claims during the last significant U.S. recession in 2008

Review Questions

1. What is the function of preattentive attributes in data visualization?

More Than 3 Million Americans Lost Their Jobs Last Week. See Your State.

Official statistics have revealed how severely coronavirus has hurt the job market. But it may take several months before we know whether this economic disaster will resemble a storm or a long winter.

BY QUOCTRUNG BUI and JUSTIN WOLFERS

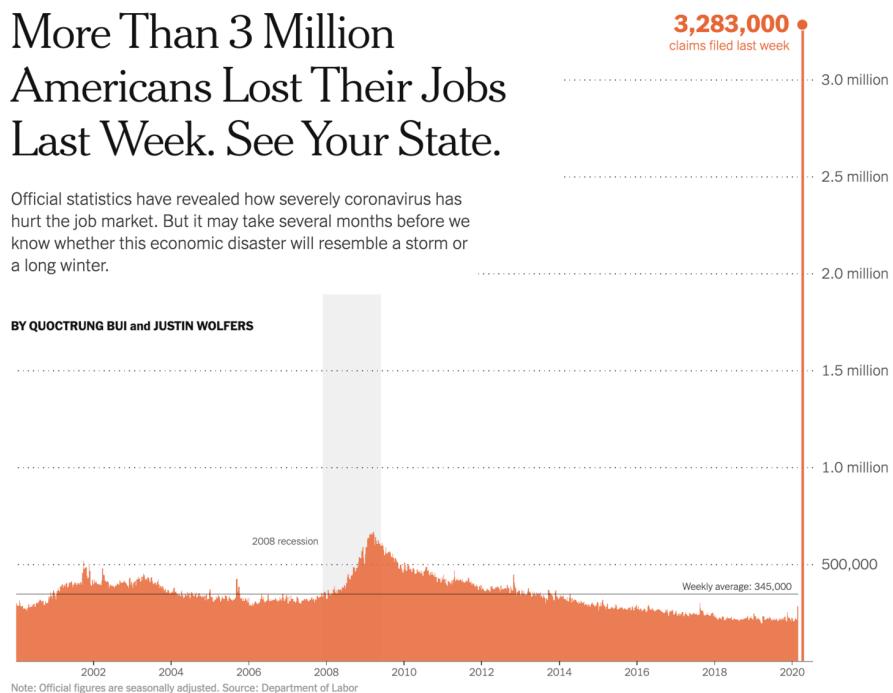


Figure 126: An archetype of data visualization elegance

2. Why is the use of red and green in data visualization potentially problematic?
3. Why are horizontal bar charts easier to interpret than pie charts?
4. What types of data are visualized in scatterplots?
5. What advantages do heatmaps provide over tables?
6. What are some people analytics use cases for slopegraphs?
7. What are some people analytics use cases for sankey diagrams?
8. Why is it important to use a zero baseline for y-axes?
9. Why is it a best practice to avoid rotating axis labels?
10. What is the primary purpose of applying a consistent color palette across visuals?

Data Storytelling

Like Chapter @ref(data-viz), this chapter leans more *art* than *science* in the coverage of strategies to translate results of analyses into compelling stories that have a high probability of landing effectively with target audiences. One may build an impressive set of complex models and perform robust analyses but unless the results and recommendations are clear to stakeholders, this is an exercise in futility. We need stories to distill analytical output down to a non-technical and logical organization of unified ideas that arouse the energy and attention of the audience.

Know Your Audience

First, we must remember that we are communicating *for the audience*, not for ourselves. To do this effectively, we need to know the audience – what is important to them (e.g., making money, reducing expenses, gaining market share, innovation) and how they define and measure success. Knowing what matters to those we seek to influence with information enables us to tailor content in a way that appeals to the unique needs of those who can ultimately authorize action and drive change.

A unique challenge for people analytics practitioners is that the audience is often non-technical folks in the People organization. The strategies needed to communicate effectively to this audience are quite different from those required to present an analysis to Finance, for example. Using technical jargon is a sure way to quickly lose a non-technical audience and sacrifice what could otherwise be a poignant and impactful call to action. Therefore, it is critically important to identify and understand the target audience before packaging and delivering results.

In general, it is best not to assume the audience knows what you know because chances are good that no one else has been as deeply immersed in the problem as you.

Several contextual questions are important to consider:

1. What essential background information does the audience already have, and what additional context needs to be shared?
2. Does the audience already have access to any relevant data that would strengthen or conflict with the messages we need to convey?
3. Does the audience or decision maker have a known perspective on the topic?
4. How should the content be messaged to proactively appeal to potential risks or biases the audience may possess that could create resistance to the findings and/or recommendations?
5. What is the audience's preferred format for content sharing (e.g., document vs. deck)?

Production Status

Prior to documents or decks being circulated to the target audience, there are generally several revs to refine the content and storyline in partnership with internal analytics colleagues and other stakeholders. If working docs are shared prematurely, it can result in the analytics team needing to address unnecessary questions that may compromise the project scope and timeline. Stakeholders will naturally have questions and ideas for further exploration, and we should always value their engagement and enthusiasm for more insights. However, unless there is a critical business need to venture down another path, the analytics team must protect the project scope and cite in the presentation the need to prioritize the additional analyses as a next step.

It is important to define a production status taxonomy to clarify content readiness. The framework below is one example:

1. **Draft:** Working document/deck being prepared by the analytics team (do not share!)
2. **Initial Review:** Document/deck suitable for initial input from stakeholders
3. **Final Review:** Document/deck reflects feedback from initial review and ready for final review and sign off
4. **Complete:** Document/deck complete and ready for delivery to target audience

Structural Elements

Entrepreneurs pitching to VCs craft a concise *elevator pitch* that highlights the key features of their product or service complemented by data to support the unique value proposition and attractive upside of the business (e.g., market size and penetration, revenue growth trajectory). In the same way, a good early

step in storytelling with data is drafting a short story and iterating on it until only the essential content remains and it requires no more than a few minutes to read.

When preparing a presentation, a general rule is to never begin with slides. The aesthetics and formatting are of secondary importance to the organization and sequencing of content. Begin by outlining the structure and organizing content in a way that supports and punctuates the key messages you wish to convey to the audience. The outline can then be translated into a presentation-ready deck or document, which will support a logical and effective flow of information.

While the structure and format may vary company to company, and across stakeholder groups within a particular company, there are some essential elements that will support the effectiveness of an analysis presentation.

TL;DR

A **TL;DR (too long; don't read)** is an abstract intended to provide a concise synopsis of the research objective, high-level approach, key findings, and recommendations and next steps. The TL;DR is popular among tech companies, though the general idea is embraced by organizations in other industries under various names (e.g., Executive Summary).

Here is an example of a TL;DR for an analysis intended to support a solution to an organization's location capacity issue:

Based on our strategic workforce plan, we will exceed the capacity at our HQ facility within three years. Labor market analyses indicate that by expanding our remote workforce, we can source from a larger pool of talent with the skills we need to maintain our competitive position in the market, decrease our average time to fill positions, lower our average cost per employee, and solve for the impending capacity breach at our HQ facility. A shift to remote-first hiring is recommended over the next six months to proactively address both our strategic workforce needs and location capacity constraints.

This short TL;DR led with the problem statement, highlighted a solution and associated benefits based on the analysis, and finished with a recommended next step and timeline. While this short summary does not provide the intricate details of the analysis, in most cases it should be sufficient to communicate the high-level story to those with limited time. This short summary may also function to pique the interest of stakeholders who are initially unsure if the analysis warrants their time and attention.

Purpose

Studies show that our audience determines in 3-8 seconds whether or not to give us their attention based on what we have put in front of them (Knaflc, 2015).

Therefore, it is crucial to let the audience know upfront what we are going to share with them and why it matters.

There is perhaps no more important initial question to answer than: “What does a successful outcome look like?” To answer this question, we must be able to clearly articulate the problem statement. That is, why is it important to perform this analysis over the many other analytics opportunities aging on the team’s backlog?

If we cannot clearly articulate the problem statement and desired success outcome(s), the audience will likely be unclear about what we are asking them to do and why they should make it a priority. All content and messages should support what we want our audience to know or do; therefore, defining what success looks like is a prerequisite for outlining and structuring the presentation.

While the problem statement need not be dramatized, this is an excellent opportunity to arouse an emotional response that hooks the audience. Consider the following juxtaposition of two very different approaches to articulating the same problem statement:

- **Option 1:** Our offer acceptance rate (OAR) has declined from 80% to 40% YoY in Engineering.
- **Option 2:** To deliver on our strategic workforce plan and support our ambitious product roadmap, we need to attract the best and brightest Engineering talent. However, an increasing number of Engineering candidates are declining our offers (OAR has declined 50 percent YoY), and this presents a material risk to the guidance issued to shareholders during our most recent earnings call.

The severity of the situation is far more palpable with the framing in Option 2 relative to the terse and factual approach taken in Option 1. As a result, it stands to reason that Option 2 features a higher likelihood of the audience taking seriously subsequent content which supports a solution to this problem.

Methodology

It is usually appropriate to provide a *high-level* overview of the analysis approach. In doing this, it is important to know what level of information is appropriate for the target audience.

It is easy to provide too much information in this section. For example, does the audience need to know that we built a multilevel model with quadratic terms on time variables? If the People team is the target audience, there is a good chance that this level of information is *not* appropriate. Sharing that an analysis method was used that is well-suited for understanding drivers of the outcome being studied is likely sufficient, as the content and presentation should

index more on the findings and recommended next steps than on methodological details.

Presenting results of analyses to a non-technical audience using technical nomenclature will at best be seen as hubris and create confusion; at worst, the important information you need to convey will be completely lost on the audience and the efforts to improve the employee experience will be a deplorable miss. In general, if information is included in the presentation, we should ask ourselves whether the audience *really* needs to know it. If the key points are likely to land without the information, we should generally eliminate it. To paraphrase a quote attributed to Einstein: “Everything should be made as simple as possible, but not simpler.”

Results

Just as the methodology should not be presented to stakeholders on the People team using technical nomenclature, results of statistical analyses also need an appropriate translation for the audience. If you are operating in a tech company, it is possible that terms like *modeling* and *n*-count are broadly understood. This is again a case in which it is critical to know your audience to ensure the key messages resonate based on their knowledge, vernacular, and delivery preferences.

Tables containing the output of regression models, for example, should not be included in the main body of the document or deck. As we covered in Chapter @ref(lm), regression lends to a highly intuitive interpretation – assuming relationships between predictors and the outcome are not modeled using complex transformations. Therefore, we can speak to the results using language such as:

“When controlling for alternative influences such as $X_2 \dots X_p$, the average change in Y for each one-unit increase in X_1 is β_1 . The association between X_1 and Y is statistically significant.”

By replacing $X_2 \dots X_p$ with the names of control variables, X_1 with the name of the main predictor variable, Y with the name of the outcome variable, and β_1 with the corresponding numerical for the average change, we do not need statistical symbols or technical jargon to communicate high-level results of regression analyses.

If data are visualized, this does not require them to be included in the ultimate doc or deck presented to stakeholders. Myriad visuals are often generated during the EDA phase of analytics projects to understand distributions, relationships, differences, density, and trends, but relatively few – and sometimes none – need to be shared beyond the project team.

Limitations

This is the section in which the limits of the study should be made known to the audience. There are limits to every study – even the most rigorous – and the failure of a researcher to report them does not change this fact.

Outlined below are some questions to consider in identifying limitations:

1. Does the research design support causal effects?
2. Were there confounding variables that may have compromised the internal validity of the research?
3. Were there data quality issues that may have biased results?
4. Was low participation a factor, and could feedback from nonrespondents materially change the results?
5. Is the observation window in which data were collected recent enough to reflect current sentiments and behaviors?
6. Were mitigating factors implemented to control for social desirability?
7. Were strategies deployed to mitigate the risk of common method variance for self-reported data collection?
8. Were employees incentivized to participate, and could this have influenced their responses or performance?
9. Were data dimensions standardized and consistent across time, or were data imputation and/or mapping strategies required to support longitudinal analyses?
10. Did time constraints require curtailing the project scope?

Next Steps

Presentations should conclude with recommended next steps that are likely to generate a productive and action-oriented dialogue among the stakeholders. Assuming there was a firm commitment to action at the outset of the project (see Chapter @ref(getting-started)), there is a good chance the audience does not understand what was shared or is not sure what to do next if their only response following the presentation is, “That’s interesting!” Leaving the audience with clear answers to the following questions will help drive accountability for action:

- What are we asking the audience to do based on the findings?
- What are we – the analytics team – committed to doing as a next step (e.g., working with operations teams to improve data quality for future analyses, pre/post study on the efficacy of proposed interventions)?
- When is an appropriate time to follow up to help remove any barriers to action taking?

Appendix

Given the significant time required to wrangle, model, and analyze data, it is often tempting to include in a presentation details associated with every aspect of the analytics project of which we are so proud. However, nonessential details will likely detract from the key messages we wish to convey.

Including in the Appendix of a document or deck information that is superfluous to the main points ensures the details are available should they be needed in addressing the audience's questions whilst ensuring the audience is focused only on the material elements of the presentation. This may be tables containing metrics across various dimensions or results from supplemental analyses adjacent to the primary research objective(s).

Q&A

It is important to reserve time and be prepared for Q&A with the audience when delivering a live presentation. Despite our most earnest planning efforts, the likelihood is low that even the most effective presentations will address *all* the questions our audience has. Addressing questions or concerns in the moment, or by way of follow up in the case of an asynchronous review of a doc or deck, will increase the probability of action being taken on the recommendations.

Review Questions

1. Based on research, how long do we have to convince an audience that what we plan to present is worth their time and attention?
2. Why is there no one-size-fits-all approach to storytelling with data?
3. What are some early contextual considerations as it relates to understanding the target audience?
4. What is the utility of preattentive attributes in storytelling?
5. What are some strategies for organizing and structuring content to support compelling narratives?
6. What is a TL;DR, and what is its purpose?
7. Why is it important to clearly define success?
8. What are some features of an effective conclusion that supports accountability for action?

9. What are some common limitations in people analytics projects, and why is it important to note them when presenting results of analyses?
10. What are some examples of content that should be moved to the Appendix?

Bibliography

- Aguinis, H., Gottfredson, R., & Joo, H. (2013). Best-Practice Recommendations for Defining, Identifying, and Handling Outliers. *Organizational Research Methods*, 16(2), 270-301.
- Albright, S. C., & Winston, W. L. (2016). *Business Analytics: Data Analysis & Decision Making* (6th ed.). Boston, MA: Cengage.
- Angrist, J. D., & Pischke, J.-S. (2009). *Mostly harmless econometrics: An empiricist's companion*. Princeton, NJ: Princeton University Press.
- Baron, R. M., & Kenny, D. A. (1986). The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology*, 51(6), 1173–1182.
- Bartlett, M. S. (1954). A note on the multiplying factors for various chi square approximations. *Journal of the Royal Statistical Society: Series B*, 16(2), 296-298.
- Bartlett, R. (2013). *A Practitioner's Guide to Business Analytics: Using Data Analysis Tools to Improve Your Organization's Decision Making and Strategy*. New York, NY: McGraw-Hill.
- Bickel, P. J., Hammel, E. A., & O'Connell, J. W. (1975). Sex Bias in Graduate Admissions: Data from Berkeley. *Science*, 187(4175), 398–404. doi:10.1126/science.187.4175.398
- Brant, R. (1990). Assessing Proportionality in the Proportional Odds Model for Ordinal Logistic Regression. *Biometrics*, 46(4), 1171–1178. <https://doi.org/10.2307/2532457>
- Chakravarti, I. M., Laha, R. G., & Roy, J. (1967). *Handbook of methods of applied statistics*. New York: Wiley.
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.
- Creswell, J. W., & Creswell, J. D. (2018). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (5th ed.). Los Angeles: Sage.

- Cronbach, L. J., & Meehl, P. E. (1955). Construct validity in psychological tests. *Psychological Bulletin*, 52(4), 281–302. <https://doi.org/10.1037/h0040957>
- Daniel, W. W. (1990). Kruskal-Wallis one-way analysis of variance by ranks. *Applied Nonparametric Statistics* (2nd ed.). Boston: PWS-Kent. pp. 226–234.
- Delacre, M., Lakens, D., & Leys, C. (2017). Why Psychologists Should by Default Use Welch's t-test Instead of Student's t-test. *International Review of Social Psychology*, 30(1), 92–101. DOI: <http://doi.org/10.5334/irsp.82>
- DeVellis, R. F. (2012). Scale development: Theory and applications. Thousand Oaks, Calif: Sage.
- Edmondson, A. (1999). Psychological safety and learning behavior in work teams. *Administrative Science Quarterly*, 44(2), 350-383.
- Gelman, A., & Hill, J. (2006). Analytical methods for social research: Data analysis using regression and multilevel/hierarchical models. Cambridge University Press.
- Gordon, N. (1998). Colour blindness. *Public Health*, 112(2), 81–84. doi:10.1038/sj.ph.1900446.
- Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate data analysis* (6th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. New York: Springer.
- Kahneman, D. (2011). *Thinking, fast and slow*. New York: Farrar, Straus and Giroux.
- Kaiser, H. (1974). An index of factorial simplicity. *Psychometrika*, 39(1), 31-36.
- Kenny, D. A., Kaniskan, B., & McCoach, D. B. (2015). The performance of RMSEA in models with small degrees of freedom. *Sociological Methods & Research*, 44(3), 486–507. <https://doi.org/10.1177/0049124114543236>
- Kerlinger, F., & Lee, H. (2000). *Foundations of behavioral research* (4th ed.). Melbourne: Wadsworth.
- Kimball, R. (1996). *The Data Warehouse Toolkit*. New York: Wiley.
- Kline, R. B. (2005). *Principles and practice of structural equation modeling* (2nd ed.). New York, NY: Guilford Press.
- Knaflic, C. N. (2015). *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Hoboken, NJ: Wiley.
- Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. New York: Springer.
- Langkjær-Bain, R. (2017), The murky tale of Flint's deceptive water data. *Significance*, 14: 16-21. <https://doi.org/10.1111/j.1740-9713.2017.01016.x>

- Levene, H. (1960). Robust Tests for Equality of Variances. In: Olkin, I., Ed., Contributions to Probability and Statistics. Palo Alto: Stanford University Press. pp. 278-292.
- Martin, P. (2022). Regression Models for Categorical and Count Data (1st ed.). SAGE Publications. Retrieved from <https://www.perlego.com/book/3277492/regression-models-for-categorical-and-count-data-pdf> (Original work published 2022)
- Mitchell, Tom (1997). Machine Learning. New York: McGraw Hill.
- Perneger, T. V. (1998). What's wrong with Bonferroni adjustments. *BMJ*, 316(7139), 1236-1238.
- Pishro-Nik, H. (2014). Introduction to Probability: Statistics and Random Processes. Blue Bell, PA: Kappa Research, LLC.
- Roethlisberger, F. J., & Dickson, W. J. (1939). Management and the worker. Harvard Univ. Press.
- Satterthwaite, F. E. (1946). An Approximate Distribution of Estimates of Variance Components. *Biometrics Bulletin*, 2(6), 110–114. <https://doi.org/10.2307/3002019>
- Schaufeli, W. B., Bakker, A. B., & Salanova, M. (2006). The measurement of work engagement with a short questionnaire: A cross-national study. *Educational and Psychological Measurement*, 66(4), 701-716. <https://doi.org/10.1177/0013164405282471>
- Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4), 591–611. <https://doi.org/10.1093/biomet/52.3-4.591>
- Snedecor, G. W., & Cochran, W. G. (1980). Statistical methods (7th ed.). Ames, Iowa: Iowa State University Press.
- Spector, P. E. (2006). Method Variance in Organizational Research: Truth or Urban Legend? *Organizational Research Methods*, 9(2), 221-232.
- Starbuck, C. R. (2016). Managing insidious barriers to upward communication in organizations: An empirical investigation of relationships among implicit voice theories, power distance orientation, self-efficacy, and leader-directed voice (Doctoral dissertation, Regent University).
- Vargha, A. & Delaney, H. D. (2000). A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25(2), 101–132.
- Wald, Abraham. (1943). A Method of Estimating Plane Vulnerability Based on Damage of Survivors. Statistical Research Group, Columbia University. CRC 432 — reprint from July 1980 Archived 2019-07-13 at the Wayback Machine. Center for Naval Analyses.

Welch, B. L. (1947). The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika*, 34(1/2), 28–35. <https://doi.org/10.2307/2332510>

Wheelan, C. (2013). Naked Statistics: Stripping the Dread from the Data. New York: W.W. Norton.

Appendix

4D Framework

This section provides a detailed set of considerations and framework for structuring analytics projects.

Discover

During the Discover phase, it is important to remain in the problem zone; seek to understand your clients' needs through active listening and questions. This is not the time for solutioning or committing to any specific deliverables or timelines. If the client's needs are ambiguous, proceeding without clarity is unlikely to result in a favorable outcome.

It is generally helpful to think about analytics solutions – whether a dashboard with basic metrics and trends or an advanced analysis – like a Product Owner thinks about the initial and subsequent releases of a commercial product. A **Minimum Viable Product (MVP)** is a version of the solution with the minimum number of features to be useful to early customers who can provide feedback for future enhancements. It is important to clarify that the MVP version of solutions often has both a limited number of users and features, which protects against the tendency to boil the ocean by striving to address every question for every stakeholder. Breaking down large projects into small sets of features that are easier to communicate and adopt provides space for agility and real-time adjustments to the product roadmap per user feedback.

There are some initial questions and considerations that will help frame an analytics project and support its success.

Client

Who is the client? A client can be a person or organization contracting you for consulting services, or an internal stakeholder within your organization who has need. What is important to them?

Primary Objective

- What is the client ultimately hoping to accomplish?
- Is the request merely to satisfy one's curiosity, or are there actions that can realistically be taken to materially influence the stated objective?

Problem Statement

- One of the most important early steps is clearly defining the problem statement. If your understanding of the problem – after translating from the business anecdotes in which it was likely initially expressed – is misaligned with the client's needs, none of the subsequent steps matter.

Guiding Theories

- What theoretical explanations can the client offer as potential rationalizations for the phenomena of interest?
- Are there existing theories in the organizational literature that should guide how the problem is tackled (e.g., findings from similar research implemented in other contexts)?

Research Questions

To respect the nuances of the problem statement, it is important to unpack it and frame as a set of overarching questions to guide the research.

- Q1: ...
- Q2: ...
- Q3: ...

To ensure it is possible (and appropriate) to take action in response to these questions, consider the following after drafting each question: "What would I do if I knew the answer?"

Research Hypotheses

Once research questions are developed, what do you expect to find based on anecdotal stories or empirical findings? As a next step, these expectations should be expressed in the form of research hypotheses. Note that these research hypotheses are different from statistical hypotheses; both will be covered in detail in later chapters.

- H1: ...
- H2: ...
- H3: ...

To ensure the hypotheses lend to actionable analyses, it is important to consider the following: “What does success look like?” In other words, once the project is complete, against which success measures will the project’s success be determined? Curiosity is not a business reason and hope is not a reasonable strategy. The following questions may prove helpful in the promotion of actionable – over merely interesting – outcomes:

- What will be done if the hypotheses are supported?
- What will be done if the hypotheses are *not* supported?

Assumptions

At this point, it’s helpful to consider what assumptions may be embedded in this discovery work. Are the questions and hypotheses rooted in what the client has theorized, or are these the product of an ambiguous understanding of the client’s needs?

Cadence

- Is this analysis a one-off, or could there be a need to refresh this analysis on a regular cadence?
- Are there dates associated with programs or actions this analysis is intended to support?

Aggregation

Is there a need for individual-level detail supporting the analysis? Aggregate data should generally be the default unless a compelling justification exists and exception granted from legal and privacy partners. One important role of analysts is to help keep the audience focused on the bigger picture and findings. Access to individual-level detail can not only introduce unnecessary legal and compliance risk but can also lead to questions and probing that can delay taking necessary actions based on the results.

Deliverable

What is the preferred method of communicating the results of the analysis (e.g., interactive dashboard, static slide deck, document)? It is important to determine this early so that subsequent efforts can be structured to support the preferred deliverable. For example, if an interactive dashboard is preferred, does your Engineering department need to prioritize dependent tasks such as data pipelines, row-level security, BI development, and migrations to production servers?

Filters & Dimensions

How does your client prefer to segment the workforce? Some common grouping dimensions are business unit, division, team, job family, location, tenure, and management level, but the client may have custom segmentation requirements that will be important to identify and define early in the project.

Design

Perhaps the most important initial question to answer in the design phase is: “Does anything already exist that addresses part, or all, of the client’s objectives?” If an existing solution will suffice, or a previous analysis can be easily refreshed with recent data, it may be possible to allocate time and resources elsewhere. If related or complimentary analyses have already been performed, they may accelerate new analyses.

The end-user experience is of paramount importance during the Design phase, as solutions should have a consistent look and feel regardless of who developed them. Defining and implementing design guidelines, such as those outlined in Chapter @ref(data-viz), will ensure consistency across analytics projects as well as within large projects in which multiple analysts are collaborating on various elements of the solution.

Data Privacy

Are there potential concerns with the study’s objective, planned actions, and/or requested data elements from an employee privacy or legal perspective? A cross-functional data governance committee can help with efficient and consistent decisioning on requests for people data and analytics.

In cases where sensitive attributes such as gender, ethnicity, age, sexual orientation, and disability status are requested, it is best to consult with legal and privacy partners to ensure they green light the intended use of the data. The decision on whether or not to include these sensitive data elements is often less about what the audience can view (e.g., People Partners may already have access to the information at the individual level in the source system) and more anchored in what they plan to do with the information.

Data Sources & Elements

Is the required data already accessible in a data warehouse or other analytics environment? If not, does it need to be? What is required to achieve this?

- What data sources are required?
- What data elements are required?

Data Quality

It is important to understand the data generative process and never make assumptions about how anomalies or missing data should be interpreted. After identifying what data sources will be required for a particular analysis, it is important to meet with source system owners and data stewards to deeply understand the business processes by which data are generated in the system(s). An ideal should always be *unimpeachable* data quality at the analysis presentation stage, and this begins with an early understanding and investigation of data quality in the source systems. Below are some helpful questions to consider:

- Is there a better data source, such as data generated further upstream?
- Are the data actively validated with automated data quality checks and regular exceptions reports?
- What is the SLA for the data refreshes, and how often is this SLA met?
- Is there a clear (and accessible) data source owner/steward?
- Is there comprehensive documentation and field definitions?
- Do the tables drive business critical deliverables?
- Who are the other consumers of the data?

Variables

How will the constructs be measured (e.g., survey instrument, derived attribute, calculated field)?

Analysis Method

What are the appropriate analysis methods based on the research hypotheses? If modeling is required, is it more important to index on accuracy or interpretability?

Dependencies

Are other teams required to develop this solution? What is the nature of the work each dependent team will perform? Are there required system configuration changes? Do these teams have capacity to support?

Change Management

Will this solution impact current processes or solutions? If so, what is the change management plan to facilitate a seamless transition and user experience?

Sign-Off

Generally, it is best for the client to signoff on the problem statement, analysis approach, and wire frame for the deliverable (if applicable) before providing an ETA and proceeding to the development phase. This ensures alignment on the client's needs and the perceived utility of the solution in addressing those needs.

Develop

While development patterns can vary widely across analytics teams, establishing a set of standards can pay dividends in the form of greater efficiency and reliability over time. Pattern-based development ensures analysts who were not involved in a particular project can access the code and easily and quickly understand each step of the analysis.

Development Patterns

- Are there development patterns that should guide the development approach to support consistency?

- Are there existing calculated fields that can/should be leveraged for derived data?
- Are there best practices that should be employed to optimize performance (e.g., load time for dashboards, executing complex queries during non-peak times)?
- Are there standard color palettes that should be applied?

Productionizable Code

- How do models and data science pipelines need to be developed to facilitate a seamless migration from lower to upper environments? For example, initial EDA may be performed using curated data in flat files for the purpose of identifying meaningful trends, relationships, and differences, but where will this data need to be sourced in production to automate the refresh of models at a regular interval? If the data were provided from multiple source systems, what joins are required to integrate the data? What transformation logic or business rules need to be applied to reproduce the curated data?

Unit Testing

- What test cases will ensure the veracity of data?
- Who will perform the testing?

User Acceptance Testing (UAT)

- In the spirit of agility and constant contact with the client to prevent surprises, it is generally a good idea to have the client take the solution for a test run within the UAT environment and then provide sign-off before migrating to production. If the deliverable is a deck or doc with results from a model, UAT may surface clarifying questions that can be addressed before releasing to the broader audience.

Deliver

The Deliver phase can take many forms depending on the solution being released. If the solution is designed for a large user base, a series of recorded trainings may be in order so that there is a helpful reference for those unable to attend the live sessions or new joiners in the future. It is important to monitor success measures, which could be insights aligned to research hypotheses, dashboard utilization metrics, progress following data-informed interventions, or any number of others defined within the Discover phase.

Data Visualization

This section provides the code needed to reproduce visuals in Chapter @ref(data-viz).

Step-by-Step Visual Upgrade

```
# Load libraries
library(dplyr)
library(ggplot2)
library(ggpubr)
library(peopleanalytics)

# Load data
data("employees")

# Create data cube for active employees
smmry_ed_field <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::count(ed_field)

# Step 1: Build Bar Chart with Defaults
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity")

# Step 2: Remove Legend
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::theme(legend.position = "none")

# Step 3: Assign Colors Strategically
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::scale_fill_manual(values = c("Human Resources" =
  "#BFBFBF",
  "Life Sciences" =
  "#0070C0",
  "Marketing" = "#BFBFBF",
  "Medical" = "#BFBFBF",
  "Other" = "#BFBFBF",
```

```

    "Technical Degree" =
    ↪ "#BFBFBF")) +
ggplot2::theme(legend.position = "none")

# Step 4: Add Axis Titles and Margins
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
    ↪ as.factor(ed_field))) +
ggplot2::geom_bar(stat = "identity") +
ggplot2::labs(x = 'Education Field', y = 'Headcount') +
ggplot2::scale_fill_manual(values = c("Human Resources" =
    ↪ "#BFBFBF",
    "Life Sciences" =
    ↪ "#0070C0",
    "Marketing" = "#BFBFBF",
    "Medical" = "#BFBFBF",
    "Other" = "#BFBFBF",
    "Technical Degree" =
    ↪ "#BFBFBF")) +
ggplot2::theme(legend.position = "none",
    axis.title.x = ggplot2::element_text(margin =
        ↪ ggplot2::margin(t = 10, r = 0, b = 0, l = 0)),
    axis.title.y = ggplot2::element_text(margin =
        ↪ ggplot2::margin(t = 0, r = 10, b = 0, l = 0)))

# Step 5: Add Left-Justified Title
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
    ↪ as.factor(ed_field))) +
ggplot2::geom_bar(stat = "identity") +
ggplot2::labs(title = 'Life Sciences is the most common education
    ↪ field \npursued by active employees.', x = 'Education Field',
    ↪ y = 'Headcount') +
ggplot2::scale_fill_manual(values = c("Human Resources" =
    ↪ "#BFBFBF",
    "Life Sciences" =
    ↪ "#0070C0",
    "Marketing" = "#BFBFBF",
    "Medical" = "#BFBFBF",
    "Other" = "#BFBFBF",
    "Technical Degree" =
    ↪ "#BFBFBF")) +
ggplot2::theme(legend.position = "none",
    plot.title = ggplot2::element_text(hjust = 0),
    axis.title.x = ggplot2::element_text(margin =
        ↪ ggplot2::margin(t = 10, r = 0, b = 0, l = 0)),
    axis.title.y = ggplot2::element_text(margin =
        ↪ ggplot2::margin(t = 0, r = 10, b = 0, l = 0)))

```

```
# Step 6: Remove Background
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::labs(title = 'Life Sciences is the most common education
  \nfield npursued by active employees.', x = 'Education Field',
  y = 'Headcount') +
  ggplot2::scale_fill_manual(values = c("Human Resources" =
  "#BFBFBF",
  "Life Sciences" =
  "#0070C0",
  "Marketing" = "#BFBFBF",
  "Medical" = "#BFBFBF",
  "Other" = "#BFBFBF",
  "Technical Degree" =
  "#BFBFBF")) +
  ggplot2::theme(legend.position = "none",
  panel.background = ggplot2::element_blank(),
  axis.title.x = ggplot2::element_text(margin =
  ggplot2::margin(t = 10, r = 0, b = 0, l = 0)),
  axis.title.y = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 10, b = 0, l = 0)))

# Step 7: Remove Axis Ticks
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::labs(title = 'Life Sciences is the most common education
  \nfield npursued by active employees.', x = 'Education Field',
  y = 'Headcount') +
  ggplot2::scale_fill_manual(values = c("Human Resources" =
  "#BFBFBF",
  "Life Sciences" =
  "#0070C0",
  "Marketing" = "#BFBFBF",
  "Medical" = "#BFBFBF",
  "Other" = "#BFBFBF",
  "Technical Degree" =
  "#BFBFBF")) +
  ggplot2::theme(legend.position = "none",
  panel.background = ggplot2::element_blank(),
  axis.title.x = ggplot2::element_text(margin =
  ggplot2::margin(t = 10, r = 0, b = 0, l = 0)),
  axis.title.y = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 10, b = 0, l = 0)),
```

```

    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank())

# Step 8: Mute Titles
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  ↪ as.factor(ed_field))) +
ggplot2::geom_bar(stat = "identity") +
ggplot2::labs(title = 'Life Sciences is the most common education
  ↪ field \npursued by active employees.', x = 'Education Field',
  ↪ y = 'Headcount') +
ggplot2::scale_fill_manual(values = c("Human Resources" =
  ↪ "#BFBFBF",
  "Life Sciences" =
  ↪ "#0070C0",
  "Marketing" = "#BFBFBF",
  "Medical" = "#BFBFBF",
  "Other" = "#BFBFBF",
  "Technical Degree" =
  ↪ "#BFBFBF")) +
ggplot2::theme(legend.position = "none",
  panel.background = ggplot2::element_blank(),
  axis.title.x = ggplot2::element_text(margin =
  ↪ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
  ↪ colour = "#404040"),
  axis.title.y = ggplot2::element_text(margin =
  ↪ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  ↪ colour = "#404040"),
  plot.title = ggplot2::element_text(hjust = 0,
  ↪ colour = "#404040"),
  axis.ticks.x = ggplot2::element_blank(),
  axis.ticks.y = ggplot2::element_blank())

# Step 9: Flip Axes
ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y = n, fill =
  ↪ as.factor(ed_field))) +
ggplot2::geom_bar(stat = "identity") +
ggplot2::coord_flip() +
ggplot2::labs(title = 'Life Sciences is the most common education
  ↪ field \npursued by active employees.', x = 'Education Field',
  ↪ y = 'Headcount') +
ggplot2::scale_fill_manual(values = c("Human Resources" =
  ↪ "#BFBFBF",
  "Life Sciences" =
  ↪ "#0070C0",
  "Marketing" = "#BFBFBF",
  "Medical" = "#BFBFBF",
  "Other" = "#BFBFBF",
  "Technical Degree" =
  ↪ "#BFBFBF"))

```

```

    "Medical" = "#BFBFBF",
    "Other" = "#BFBFBF",
    "Technical Degree" =
      ↳ "#BFBFBF")) +
ggplot2::theme(legend.position = "none",
  panel.background = ggplot2::element_blank(),
  axis.title.x = ggplot2::element_text(margin =
  ↳ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
  ↳ colour = "#404040"),
  axis.title.y = ggplot2::element_text(margin =
  ↳ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  ↳ colour = "#404040"),
  plot.title = ggplot2::element_text(hjust = 0,
  ↳ colour = "#404040"),
  axis.ticks.x = ggplot2::element_blank(),
  axis.ticks.y = ggplot2::element_blank())

# Step 10: Sort Data
ggplot2::ggplot(smmry_ed_field, aes(x = reorder(ed_field, n), y =
  ↳ n, fill = as.factor(ed_field))) +
ggplot2::geom_bar(stat = "identity") +
ggplot2::coord_flip() +
ggplot2::labs(title = 'Life Sciences is the most common education
  ↳ field \npursued by active employees.', x = 'Education Field',
  ↳ y = 'Headcount') +
ggplot2::scale_fill_manual(values = c("Human Resources" =
  ↳ "#BFBFBF",
    "Life Sciences" =
      ↳ "#0070C0",
    "Marketing" = "#BFBFBF",
    "Medical" = "#BFBFBF",
    "Other" = "#BFBFBF",
    "Technical Degree" =
      ↳ "#BFBFBF")) +
ggplot2::theme(legend.position = "none",
  panel.background = ggplot2::element_blank(),
  axis.title.x = ggplot2::element_text(margin =
  ↳ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
  ↳ colour = "#404040"),
  axis.title.y = ggplot2::element_text(margin =
  ↳ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  ↳ colour = "#404040"),
  plot.title = ggplot2::element_text(hjust = 0,
  ↳ colour = "#404040"),
  axis.ticks.x = ggplot2::element_blank(),
  axis.ticks.y = ggplot2::element_blank())

```

```

axis.ticks.y = ggplot2::element_blank()

# Pre/post viz comparison
viz_pre <- ggplot2::ggplot(smmry_ed_field, aes(x = ed_field, y =
  n, fill = as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity")

viz_post <- ggplot2::ggplot(smmry_ed_field, aes(x =
  reorder(ed_field, n), y = n, fill = as.factor(ed_field))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::coord_flip() +
  ggplot2::labs(title = 'Life Sciences is the most
  common education field \npursued by active employees.', x =
  'Education Field', y = 'Headcount') +
  ggplot2::scale_fill_manual(values = c("Human
  Resources" = "#BFBFBF",
                                         "Life Sciences"
                                         =
                                         "#0070C0",
                                         "Marketing" =
                                         "#BFBFBF",
                                         "Medical" =
                                         "#BFBFBF",
                                         "Other" =
                                         "#BFBFBF",
                                         "Technical
                                         Degree" =
                                         "#BFBFBF"))
  +
  ggplot2::theme(legend.position = "none",
  panel.background =
    ggplot2::element_blank(),
  axis.title.x =
    ggplot2::element_text(margin =
      ggplot2::margin(t = 10, r = 0, b
      = 0, l = 0), colour =
      "#404040"),
  axis.title.y =
    ggplot2::element_text(margin =
      ggplot2::margin(t = 0, r = 10, b
      = 0, l = 0), colour =
      "#404040"),
  plot.title =
    ggplot2::element_text(hjust = 0,
    colour = "#404040"),
  
```

```

axis.ticks.x =
  ↵ ggplot2::element_blank(),
axis.ticks.y =
  ↵ ggplot2::element_blank()

# Display pre/post visualizations side-by-side
ggpubr::ggarrange(viz_post, viz_pre, ncol = 1, nrow = 2)

```

Tables

```

### Data Prep ###

# Load libraries
library(dplyr)
library(peopleanalytics)

# Load data
data("employees")

# Append new tenure band column
employees$tenure_band <- dplyr::case_when(
  employees$org_tenure < 1 ~ "Under 1 Year",
  employees$org_tenure < 2.5 ~ "1-2 Years",
  employees$org_tenure < 5.5 ~ "3-5 Years",
  employees$org_tenure <= 10 ~ "6-10 Years",
  TRUE ~ "Over 10 Years"
)

# Store aggregate measures to cube
dept_tenure <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::group_by(dept, tenure_band) |>
  dplyr::summarise(cnt = dplyr::n())

# Specify ordered factor
dept_tenure$tenure_band <- ordered(dept_tenure$tenure_band,
  ↵ levels = c("Under 1 Year", "1-2 Years", "3-5 Years", "6-10
  ↵ Years", "Over 10 Years"))

# Assign proper case field names
dept_tenure_proper <- dept_tenure |>
  rename('Department' = dept,

```

```
'Tenure' = tenure_band,
'Count' = cnt)
```

```
### Data Table ###

# Load library
library(DT)

# Format output with datatable
DT::datatable(dept_tenure_proper, rownames = FALSE)
```

Heatmaps

```
### Data Prep ###

# Load library
library(dplyr)

# Append new tenure band column
employees$tenure_band <- dplyr::case_when(
  employees$org_tenure < 1 ~ "Under 1 Year",
  employees$org_tenure < 2.5 ~ "1-2 Years",
  employees$org_tenure < 5.5 ~ "3-5 Years",
  employees$org_tenure <= 10 ~ "6-10 Years",
  TRUE ~ "Over 10 Years"
)

# Store aggregate measures to cube
dept_tenure <- employees |>
  dplyr::filter(active == 'Yes') |>
  dplyr::group_by(dept, tenure_band) |>
  dplyr::summarise(cnt = dplyr::n())

# Specify ordered factor
dept_tenure$tenure_band <- ordered(dept_tenure$tenure_band,
  levels = c("Under 1 Year", "1-2 Years", "3-5 Years", "6-10
  Years", "Over 10 Years"))
```

```
### Heatmap ###

# Load library
```

```

library(ggplot2)

ggplot2::ggplot(dept_tenure, aes(x = tenure_band, y = dept, fill
→   = cnt)) +
ggplot2::geom_tile() +
ggplot2::labs(title = 'A large portion of our workforce has
→   between 3 and 10 years\n of company tenure and works within
→   R&D.', x = 'Tenure', y = 'Department') +
ggplot2::guides(fill = guide_legend(title = "Employee Count")) +
ggplot2::scale_fill_continuous(low = "#F2F2F2", high = "#0070C0")
→   +
ggplot2::theme(panel.background = ggplot2::element_blank(),
               axis.ticks.x = ggplot2::element_blank(),
               axis.ticks.y = ggplot2::element_blank(),
               plot.title = ggplot2::element_text(hjust = 0,
→   colour = "#404040"),
               axis.title.x = ggplot2::element_text(margin =
→   ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
→   colour = "#404040"),
               axis.title.y = ggplot2::element_text(margin =
→   ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
→   colour = "#404040"))

```

Scatterplots

```

### Data Prep ###

# Load library
library(peopleanalytics)

# Load data
data("employees")

# Subset df to active sales employees
sales <- subset(employees, dept == 'Sales' & active == 'Yes')

# Set quote attainment flag for data viz coloring
sales$quota_flg <- ifelse(sales$ytd_sales >= 150000, 1, 0)

### Scatterplots ###

# Load libraries

```

```

library(ggplot2)
require(scales)

# Basic scatterplot
ggplot2::ggplot(sales, aes(x = work_exp, y = ytd_sales, group =
  ~ quota_flg)) +
  ggplot2::geom_point() +
  ggplot2::labs(title = 'The relationship between work experience
  ~ and YTD sales is positive.', x = 'Work Experience', y = 'YTD
  ~ Sales (USD)') +
  ggplot2::scale_y_continuous(labels = scales::comma) +
  ggplot2::theme(panel.background = ggplot2::element_blank(),
    legend.position = "none",
    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank(),
    plot.title = ggplot2::element_text(hjust = 0,
      ~ colour = "#404040"),
    axis.title.x = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      ~ colour = "#404040"),
    axis.title.y = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
      ~ colour = "#404040"))

# Scatterplot with preattentive attributes
ggplot2::ggplot(sales, aes(x = work_exp, y = ytd_sales, group =
  ~ quota_flg)) +
  ggplot2::geom_point(aes(color = as.factor(quota_flg))) +
  ggplot2::geom_hline(yintercept = 150000, linetype = 'dotted',
  ~ color = "#0070C0") +
  ggplot2::annotate(geom = "text", x = 2, y = 159000, label =
  ~ "Sales Quota = $150k", color = "#0070C0", size = 3) +
  ggplot2::labs(title = 'Some employees with more work experience
  ~ have already met the full year sales quota.', x = 'Work
  ~ Experience', y = 'YTD Sales (USD)') +
  ggplot2::scale_color_manual(values = c("0" = "#BFBFBF",
  ~ "1" = "#0070C0")) +
  ggplot2::scale_y_continuous(labels = scales::comma) +
  ggplot2::theme(panel.background = ggplot2::element_blank(),
    legend.position = "none",
    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank(),
    plot.title = ggplot2::element_text(hjust = 0,
      ~ colour = "#404040"),
    axis.title.x = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      ~ colour = "#404040"),
    axis.title.y = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
      ~ colour = "#404040"))

```

```
axis.title.y = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  colour = "#404040"))
```

Line Charts

```
### Data Prep ###

# Set seed for reproducibility
set.seed(1234)

# Create data
months = 1:24
eng_rt = 5 - runif(1, 2.7, 2.9) + 2.41*months - .41*months^2 +
  .02*months^3
fin_rt = runif(1, 5, 8) - 6.97 + 15*months - .53*months^2
ppl_rt = 3 - runif(1, 5, 8) - 6.97 + 12*months - .4*months^2
prd_rt = runif(1, 5, 8) - 6.97 + 13*months - .53*months^2

# Combine dimensions and metrics within df
attrition_dat <- data.frame(month = rep(months, 4),
  dept = c(rep('Engineering',
    length(months)),
    rep('Finance',
    length(months)),
    rep('People',
    length(months)),
    rep('Product',
    length(months))),
    rate = c(eng_rt,
    fin_rt,
    ppl_rt,
    prd_rt))

# Ordered factor for subsequent layering of lines in graph (last
# value takes the top position)
attrition_dat$dept <- factor(attrition_dat$dept, c("Product",
  "Finance", "People", "Engineering"))

### Single Series Line Graphs ###

# Load library
```

```

library(ggplot2)

ggplot2::ggplot(subset(attrition_dat, dept == 'Engineering'),
  ~ aes(x = month, y = rate, group = dept)) +
  ggplot2::labs(title = 'Engineering attrition has increased over
  ~ the last 24 months, \n largely due to an exponential spike in
  ~ the last year.', x = 'Month', y = 'Attrition Rate') +
  ggplot2::geom_line(color = '#0070C0', size = 1) +
  ggplot2::scale_x_continuous(limits = c(0, 25), breaks = seq(0,
  ~ 24, by = 3)) +
  ggplot2::theme(legend.position = "none",
    panel.background = ggplot2::element_blank(),
    axis.title.x = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      ~ colour = "#404040"),
    axis.title.y = ggplot2::element_text(margin =
      ~ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
      ~ colour = "#404040"),
    plot.title = ggplot2::element_text(hjust = 0,
      ~ colour = "#404040"),
    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank())
  
```

Two Series Line Graphs

```

# Load library
library(ggplot2)

ggplot2::ggplot(subset(attrition_dat, dept %in% c('Engineering',
  ~ 'Product')), aes(x = month, y = rate, color = dept)) +
  ggplot2::labs(title = 'The attrition trends for Engineering and
  ~ Product \n are negatively correlated over the last year.',
    x = 'Month',
    y = 'Attrition Rate',
    col = 'Department') +
  ggplot2::geom_line(size = 1) +
  ggplot2::scale_x_continuous(limits = c(0, 27), breaks = seq(0,
  ~ 24, by = 3)) +
  ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
    ~ .5, y = attrition_dat[attrition_dat$dept == 'Engineering' &
    ~ attrition_dat$month == max(attrition_dat$month), 'rate'],
    ~ label = 'Engineering', color = "#0070C0", size = 4, hjust =
    ~ 0) +
  ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
    ~ .5, y = attrition_dat[attrition_dat$dept == 'Product' &
    ~ attrition_dat$month == max(attrition_dat$month), 'rate'],
    ~ label = 'Product', color = "#808080", size = 4, hjust = 0) +
  
```

```
ggplot2::scale_color_manual(values = c("Engineering" = "#0070C0",
                                      "Product" = "#BFBFBF")) +
ggplot2::theme(panel.background = ggplot2::element_blank(),
               legend.position = "none",
               axis.title.x = ggplot2::element_text(margin =
                                         ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
                                         colour = "#404040"),
               axis.title.y = ggplot2::element_text(margin =
                                         ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
                                         colour = "#404040"),
               plot.title = ggplot2::element_text(hjust = 0,
                                         colour = "#404040"),
               axis.ticks.x = ggplot2::element_blank(),
               axis.ticks.y = ggplot2::element_blank())
```

Multiple Series Line Graphs

```
# Load library
library(ggplot2)

ggplot2::ggplot(attrition_dat, aes(x = month, y = rate, color =
  dept)) +
ggplot2::labs(title = 'The attrition trend for Engineering is
  quite different \nrelative to attrition patterns for other
  departments.',
  x = 'Month',
  y = 'Attrition Rate',
  col = 'Department') +
ggplot2::geom_line(size = 1) +
ggplot2::scale_x_continuous(limits = c(0, 27), breaks = seq(0,
  24, by = 3)) +
ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
  .5, y = attrition_dat[attrition_dat$dept == 'Engineering' &
  attrition_dat$month == max(attrition_dat$month), 'rate'],
  label = 'Engineering', color = "#0070C0", size = 4, hjust =
  0) +
ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
  .5, y = attrition_dat[attrition_dat$dept == 'Product' &
  attrition_dat$month == max(attrition_dat$month), 'rate'],
  label = 'Product', color = "#808080", size = 4, hjust = 0) +
ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
  .5, y = attrition_dat[attrition_dat$dept == 'Finance' &
  attrition_dat$month == max(attrition_dat$month), 'rate'],
  label = 'Finance', color = "#808080", size = 4, hjust = 0) +
```

```

ggplot2::annotate(geom = "text", x = max(attrition_dat$month) +
  .5, y = attrition_dat[attrition_dat$dept == 'People' &
  attrition_dat$month == max(attrition_dat$month), 'rate'],
  label = 'People', color = "#808080", size = 4, hjust = 0) +
ggplot2::scale_color_manual(values = c("Engineering" = "#0070C0",
                                      "Product" = "#BFBFBF",
                                      "Finance" = "#BFBFBF",
                                      "People" = "#BFBFBF")) +
ggplot2::theme(panel.background = ggplot2::element_blank(),
               legend.position = "none",
               axis.title.x = ggplot2::element_text(margin =
  ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
  colour = "#404040"),
               axis.title.y = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  colour = "#404040"),
               plot.title = ggplot2::element_text(hjust = 0,
  margin = ggplot2::margin(t = 0, r = 0, b = 20,
  l = 0), colour = "#404040"),
               axis.ticks.x = ggplot2::element_blank(),
               axis.ticks.y = ggplot2::element_blank())

```

Slopegraphs

```

### Data Prep ###

# Build data frame with YoY headcount metrics by department
prepost_scores <- data.frame(date = c(rep('Time 1', 2), rep('Time
  2', 2)),
                                group = rep(c('Treatment',
  'Control'), 2),
                                score = c(10, 13, 55, 16))

```

```

### Slopegraphs ###

# Load library
library(ggplot2)

# Calculate treatment and control group changes for inclusion in
  annotation
treatment_chg <- subset(prepost_scores, group == 'Treatment' &
  date == 'Time 2', select = score) - subset(prepost_scores,
  group == 'Treatment' & date == 'Time 1', select = score)

```

```

control_chg <- subset(prepot_scores, group == 'Control' & date
←   == 'Time 2', select = score) - subset(prepot_scores, group
←   == 'Control' & date == 'Time 1', select = score)

ggplot2::ggplot(prepot_scores, aes(x = date, y = score, group =
←   group)) +
ggplot2::geom_line(aes(color = group), size = 2) +
ggplot2::geom_point(aes(color = group), size = 4) +
ggplot2::expand_limits(y = 0) +
ggplot2::annotate(geom = "text", x = 2.1, y =
←   prepot_scores[prepot_scores$group == 'Treatment' &
←   prepot_scores$date == 'Time 2', 'score'] - 2, label =
←   paste0(prepot_scores[prepot_scores$group == 'Treatment' &
←   prepot_scores$date == 'Time 2', 'score'], "% (",
←   ifelse(treatment_chg >= 0, '+', ''), treatment_chg, ")",
←   "\nTreatment Group"), color = "#0070C0", size = 4, hjust = 0)
←   +
ggplot2::annotate(geom = "text", x = 2.1, y =
←   prepot_scores[prepot_scores$group == 'Control' &
←   prepot_scores$date == 'Time 2', 'score'] - 2, label =
←   paste0(prepot_scores[prepot_scores$group == 'Control' &
←   prepot_scores$date == 'Time 2', 'score'], "% (",
←   ifelse(control_chg >= 0, '+', ''), control_chg, ")",
←   "\nControl Group"), color = "#808080", size = 4, hjust = 0) +
ggplot2::labs(title = 'A more favorable change in
←   engagement was observed \n      for the treatment group
←   relative to \n      the control group.', x = 'Observation
←   Period') +
ggplot2::scale_color_manual(values = c("Control" = "#BFBFBF",
←   "Treatment" = "#0070C0")))
←   +
ggplot2::theme(panel.background = ggplot2::element_blank(),
              legend.position = "none",
              axis.ticks.x = ggplot2::element_blank(),
              axis.ticks.y = ggplot2::element_blank(),
              axis.text.y = ggplot2::element_blank(),
              plot.title = ggplot2::element_text(colour =
←   "#404040"),
              axis.title.x = ggplot2::element_text(margin =
←   ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
←   colour = "#404040"),
              axis.title.y = ggplot2::element_blank())

```

Bar Charts

```

#### Data Prep ####

# Load library
library(dplyr)

# Set seed for reproducibility
set.seed(1234)

# Generate favorability distributions
fav_pct <- round(runif(7, 20, 45), 0)
neu_pct <- round(runif(7, 20, 45), 0)
unfav_pct <- 100 - fav_pct - neu_pct
scores <- c(fav_pct, neu_pct, unfav_pct)

# Average top box (favorable) score
topbox_avg <- round(mean(fav_pct), 0)

# Build data frame with YoY headcount metrics by department
engagement_scores <- data.frame(dept = rep(c('Engineering',
  'Finance', 'Legal', 'Marketing', 'People', 'Product',
  'Sales'), 3),
  favorability = c(rep('Favorable',
    7), rep('Neutral', 7),
    rep('Unfavorable', 7)),
  pct = scores)

# Rank departments by top box score to support sorting
dept_rank <- engagement_scores |>
  dplyr::filter(favorability == 'Favorable') |>
  dplyr::arrange(desc(pct)) |>
  dplyr::mutate(rank = dense_rank(desc(pct))) |>
  dplyr::select(dept, rank)

# Flag top department records in df to support conditional
# coloring
engagement_scores$top_score = ifelse(engagement_scores$dept ==
  dept_rank[1, 'dept'], 1, 0)

# Add department rank based on top box scores
engagement_scores <- left_join(engagement_scores, dept_rank, by =
  "dept")

```

```
### Vertical Bar Charts ###

# Load library
library(ggplot2)

ggplot2::ggplot(subset(engagement_scores, favorability ==
  'Favorable'), aes(x = reorder(dept, -pct), y = pct, fill =
  as.factor(top_score))) +
  ggplot2::labs(title = 'The People team is leading in
  department-level engagement.', x = 'Department', y =
  'Engagement Score') +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::geom_hline(yintercept = topbox_avg, linetype = 'dotted',
  color = "#404040") +
  ggplot2::annotate(geom = "text", x = 7.1, y = topbox_avg + 1.3,
  label = paste0("Avg = ", topbox_avg), color = "#404040", size
  = 3) +
  ggplot2::scale_fill_manual(values = c("1" = "#0070C0",
  "0" = "#BFBFBF")) +
  ggplot2::theme(legend.position = "none",
    panel.background = ggplot2::element_blank(),
    axis.title.x = ggplot2::element_text(margin =
      ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      colour = "#404040"),
    axis.title.y = ggplot2::element_text(margin =
      ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
      colour = "#404040"),
    plot.title = ggplot2::element_text(margin =
      ggplot2::margin(t = 0, r = 0, b = 20, l = 0),
      colour = "#404040"),
    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank())
```

Horizontal Bar Charts

```
# Load library
library(ggplot2)

ggplot2::ggplot(subset(engagement_scores, favorability ==
  'Favorable'), aes(x = reorder(dept, pct), y = pct, fill =
  as.factor(top_score))) +
  ggplot2::labs(title = 'The People team is leading in
  department-level engagement.', x = 'Department', y =
  'Engagement Score') +
```

```

ggplot2::geom_bar(stat = "identity") +
  ggplot2::coord_flip() +
  ggplot2::geom_hline(yintercept = topbox_avg, linetype = 'dotted',
  ←   color = "#404040") +
  ggplot2::annotate(geom = "text", x = 1, y = topbox_avg + 2.3,
  ←   label = paste0("Avg = ", topbox_avg), color = "#404040", size
  ←   = 3) +
  ggplot2::scale_fill_manual(values = c("1" = "#0070C0",
  ←   "0" = "#BFBFBF")) +
  ggplot2::theme(legend.position = "none",
    panel.background = ggplot2::element_blank(),
    axis.title.x = ggplot2::element_text(margin =
      →   ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      →   colour = "#404040"),
    axis.title.y = ggplot2::element_text(margin =
      →   ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
      →   colour = "#404040"),
    plot.title = ggplot2::element_text(margin =
      →   ggplot2::margin(t = 0, r = 0, b = 20, l = 0),
      →   colour = "#404040"),
    axis.ticks.x = ggplot2::element_blank(),
    axis.ticks.y = ggplot2::element_blank())
  
```

Stacked Bar Charts

```

# Load library
library(ggplot2)

ggplot2::ggplot(engagement_scores, aes(x = reorder(dept, -rank),
  ←   y = pct, fill = factor(favorability, levels =
  ←   c("Unfavorable", "Neutral", "Favorable")))) +
  ggplot2::labs(title = 'Half of Engineering respondents report
  ←   unfavorable engagement.', x = 'Department', y = 'Engagement
  ←   Score') +
  ggplot2::guides(fill = guide_legend(title = "Favorability")) +
  ggplot2::geom_bar(position = "fill", stat = "identity") +
  ggplot2::coord_flip() +
  ggplot2::scale_fill_manual(values = c("Favorable" = "#0070C0",
  ←   "Neutral" = "#D9D9D9",
  ←   "Unfavorable" = "#ED7D31"))
  ←   +
  ggplot2::scale_y_continuous(labels = scales::percent) +
  ggplot2::theme(panel.background = ggplot2::element_blank(),
    axis.title.x = ggplot2::element_text(margin =
      →   ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      →   colour = "#404040"),
    
```

```

axis.title.y = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  colour = "#404040"),
plot.title = ggplot2::element_text(margin =
  ggplot2::margin(t = 0, r = 0, b = 20, l = 0),
  colour = "#404040"),
axis.ticks.x = ggplot2::element_blank(),
axis.ticks.y = ggplot2::element_blank())

### Bidirectional Bar Chart Data Prep ###

# Set seed for reproducibility
set.seed(1234)

# Build data frame with hire and term metrics by department
hires_terms <- data.frame(dept = c('Engineering', 'Finance',
  'Legal', 'Marketing', 'People', 'Product', 'Sales'),
  metric = rep(c('Hires', 'Terms'), 7),
  cnt = round(runif(14, 5, 150), 0))

# Append transformed count column to support bidirectional bar
#   charts
hires_terms$cnt_trans <- ifelse(hires_terms$metric == 'Terms', 0
  - hires_terms$cnt, hires_terms$cnt)

### Bidirectional Bar Charts ###

# Load library
library(ggplot2)

ggplot2::ggplot(hires_terms, aes(x = reorder(dept, -cnt_trans), y
  = cnt_trans, fill = as.factor(metric))) +
  ggplot2::labs(title = 'Terms have eclipsed hires in Sales,
    Product, and Engineering \nover the trailing 12-month
    period.', x = 'Department', y = 'Headcount Changes') +
  ggplot2::guides(fill = guide_legend(title = "Metric")) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::coord_flip() +
  ggplot2::scale_fill_manual(values = c("Terms" = "#ED7D31",
    "Hires" = "#0070C0")) +
  ggplot2::theme(panel.background = ggplot2::element_blank(),
    axis.title.x = ggplot2::element_text(margin =
      ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
      colour = "#404040"),

```

```

axis.title.y = ggplot2::element_text(margin =
  ↪ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
  ↪ colour = "#404040"),
plot.title = ggplot2::element_text(margin =
  ↪ ggplot2::margin(t = 0, r = 0, b = 20, l = 0),
  ↪ colour = "#404040"),
axis.ticks.x = ggplot2::element_blank(),
axis.ticks.y = ggplot2::element_blank())

```

Combination Charts

```

### Data Prep ###

# Set seed for reproducibility
set.seed(12345)

# Turnover rates
reg_rt <- round(runif(24, 5, 15), 1)
nonreg_rt <- round(runif(24, 1, 5), 1)
vol_rt <- reg_rt + nonreg_rt
turnover_dat <- data.frame(month = 1:24,
                            reg_rt = reg_rt,
                            nonreg_rt = nonreg_rt,
                            vol_rt = vol_rt)

### Combination Charts ###

# Load library
library(ggplot2)

ggplot2::ggplot(turnover_dat) +
  ggplot2::labs(title = 'The largest share of monthly voluntary
  ↪ attrition over the 24-month period \nhas consistently been
  ↪ regrettable terminations.',
    x = 'Month',
    y = 'Attrition Rate') +
  ggplot2::geom_bar(aes(x = month, y = vol_rt), fill = '#E7E6E6',
  ↪ stat = "identity") +
  ggplot2::geom_line(aes(x = month, y = reg_rt), size = 1, color =
  ↪ "#ED7D31", group = 1) +
  ggplot2::geom_line(aes(x = month, y = nonreg_rt), size = 1, color
  ↪ = "#9BC2E6", group = 1) +

```

```

ggplot2::scale_x_continuous(limits = c(0, 27), breaks = seq(0,
→ 24, by = 3)) +
ggplot2::scale_y_continuous(limits = c(0, 20), breaks = seq(0,
→ 20, by = 5)) +
ggplot2::annotate(geom = "text", x = max(turnover_dat$month) +
→ .7, y = turnover_dat[turnover_dat$month ==
→ max(turnover_dat$month), 'reg_rt'], label = 'Regrettable',
→ color = "#ED7D31", size = 4, hjust = 0) +
ggplot2::annotate(geom = "text", x = max(turnover_dat$month) +
→ .7, y = turnover_dat[turnover_dat$month ==
→ max(turnover_dat$month), 'nonreg_rt'], label =
→ 'Not\nRegrettable', color = "#5B9BD5", size = 4, hjust = 0) +
ggplot2::theme(panel.background = ggplot2::element_blank(),
               legend.position = "none",
               axis.title.x = ggplot2::element_text(margin =
→ ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
→ colour = "#404040"),
               axis.title.y = ggplot2::element_text(margin =
→ ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
→ colour = "#404040"),
               plot.title = ggplot2::element_text(hjust = 0,
→ colour = "#404040"),
               axis.ticks.x = ggplot2::element_blank(),
               axis.ticks.y = ggplot2::element_blank())

```

Waterfall Charts

```

### Data Prep ###

# Generate headcount data with id field to define bar order
hc_dat <- data.frame(id = 1:6,
                       event = c("Starting HC", "Hires", "Transfers
→ In", "Transfers Out", "Exits", "Ending
→ HC"),
                       type = c("Headcount", "Growth", "Growth",
→ "Loss", "Loss", "Headcount"),
                       count = c(100, 50, 10, -10, -20, 130),
                       start = NA,
                       end = NA)

# Define start and end values to support waterfall chart
hc_dat$end <- cumsum(hc_dat$count)
hc_dat$end <- c(head(hc_dat$end, -1), 0)

```

```

hc_dat$start <- c(0, head(hc_dat$end, -1))

# Swap start/end values for last record (Ending HC)
hc_dat[nrow(hc_dat), "end"] <- hc_dat[nrow(hc_dat), "start"]
hc_dat[nrow(hc_dat), "start"] <- 0

```

```

### Waterfall Chart ###

# Load library
library(ggplot2)

ggplot2::ggplot(hc_dat, aes(x = event, fill = type)) +
  ggplot2::geom_rect(aes(x = reorder(event, id),
                         xmin = id - 0.45,
                         xmax = id + 0.50,
                         ymin = end,
                         ymax = start)) +
  geom_text(aes(id, end, label = ifelse(id %in% c(1,6), end, '')),
            # Only label start and end HC
            vjust = 1.5) +
  ggplot2::labs(title = 'Hires and inbound transfers have outpaced
                → outbound transfers and exits \n\nin Engineering, leading to a
                → net YTD increase in headcount.',
                x = 'Event',
                y = 'Headcount') +
  ggplot2::scale_fill_manual(values = c("Headcount" = "#D9D9D9",
                                       "Growth" = "#0070C0",
                                       "Loss" = "#ED7D31")) +
  ggplot2::theme(legend.position = "none",
                panel.background = ggplot2::element_blank(),
                axis.title.x = ggplot2::element_text(margin =
                  ggplot2::margin(t = 10, r = 0, b = 0, l = 0),
                  colour = "#404040"),
                axis.title.y = ggplot2::element_text(margin =
                  ggplot2::margin(t = 0, r = 10, b = 0, l = 0),
                  colour = "#404040"),
                plot.title = ggplot2::element_text(hjust = 0,
                  colour = "#404040"),
                axis.ticks.x = ggplot2::element_blank(),
                axis.ticks.y = ggplot2::element_blank())

```

Waffle Charts

```

#### Data Prep ####

# Create df with TA funnel metrics
ta_dat <- data.frame(stage = c("Apply", "Phone Screen",
                                "Interview", "Offer Extend", "Offer Accept"),
                      cnt = c(60, 20, 10, 6, 4))

# Set depth of waffle chart (# of y-axis rows)
depth <- 10

# Each observation needs an x and y coordinate, and y needs to be
# specified first for a waffle chart with horizontal
# accumulation
waffle_dat <- expand.grid(y = 1:depth,
                           x = seq_len(ceiling(sum(ta_dat$cnt) /
                                depth)))

# Expand the applicant counts into a vector of stage labels
stages <- rep(ta_dat$stage, ta_dat$cnt)

# Integrate stages and fill any extra tiles with NA
waffle_dat$stage <- c(stages, rep(NA, nrow(waffle_dat) -
                                length(stages)))

```



```

#### Waffle Chart ####

# Load library
library(ggplot2)

ggplot2::ggplot(waffle_dat, aes(x = x, y = y, fill = stage)) +
  ggplot2::labs(title = "We select 10 in every 100 applicants.
                \nOur offer acceptance rate (OAR) is 40%.") +
  ggplot2::geom_tile(color = "white") +
  ggplot2::scale_fill_manual(values = c("Apply" = "#D9D9D9",
                                       "Phone Screen" = "#BFBFBF",
                                       "Interview" = "#A6A6A6",
                                       "Offer Extend" = "#808080",
                                       "Offer Accept" =
                                         "#0070C0")) +
  ggplot2::scale_y_reverse() +
  ggplot2::theme_void()

```

```
ggplot2::theme(legend.title = ggplot2::element_blank(),
  plot.title = ggplot2::element_text(hjust = 0,
    colour = "#404040"))
```

Sankey Diagrams

```
### Data Prep ###

# Load library
library(dplyr)

# Set seed for reproducibility
set.seed(1234)

# Create nodes df
nodes <- data.frame(name = c('Engineering', 'Finance', 'Legal',
  'Marketing', 'People', 'Product', 'Sales', # source
  'Engineering', 'Finance', 'Legal',
  'Marketing', 'People',
  'Product', 'Sales')) # destination

# Create links df
links <- expand.grid(source = 0:6, target = 7:13)

# Append employee transfer counts per department pair to links df
links$value <- ifelse(links$source == links$target-7,
  round(rnorm(1000, 150, 50), 0), round(rnorm(1000, 30, 5), 0))

# Append source department variable for colored connections
links$dept <- dplyr::case_when(
  links$source == 0 ~ "Engineering",
  links$source == 1 ~ "Finance",
  links$source == 2 ~ "Legal",
  links$source == 3 ~ "Marketing",
  links$source == 4 ~ "People",
  links$source == 5 ~ "Product",
  links$source == 6 ~ "Sales",
  TRUE ~ "NA"
)

# Append within/outside department transfer variable for colored
# connections
```

```

links$wiout_dept <- dplyr::case_when(
  (links$source == 0 & links$target == 7) | (links$source == 1 &
  ~ links$target == 8) | (links$source == 2 & links$target == 9)
  ~ | (links$source == 3 & links$target == 10) | (links$source ==
  ~ 4 & links$target == 11) | (links$source == 5 & links$target
  ~ == 12) | (links$source == 6 & links$target == 13) ~ "Within",
  TRUE ~ "Outside"
)

# Append dichotomous product/other indicator to links and nodes
# data frames
links$prod <- ifelse(links$dept == 'Product', 'Product', 'Other')
nodes$prod <- ifelse(nodes$name == 'Product', 'Product', 'Other')

```

Sankey Diagrams

```

# Load library
library(networkD3)

# Sankey diagram 1
networkD3::sankeyNetwork(Links = links,
                          Nodes = nodes,
                          Source = "source",
                          Target = "target",
                          Value = "value",
                          NodeID = "name",
                          fontSize = 12,
                          nodeWidth = 30)

# Sankey diagram 2
networkD3::sankeyNetwork(Links = links,
                          Nodes = nodes,
                          Source = "source",
                          Target = "target",
                          Value = "value",
                          NodeID = "name",
                          fontSize = 12,
                          nodeWidth = 30,
                          LinkGroup = "dept")

# Sankey diagram 3
networkD3::sankeyNetwork(Links = links,
                          Nodes = nodes,
                          Source = "source",

```

```

Target = "target",
Value = "value",
NodeID = "name",
fontSize = 12,
nodeWidth = 30,
LinkGroup = "wiout_dept")

# Sankey diagram 4
# Define color palette
colorJS <- paste('d3.scaleOrdinal(["#BFBFBF", "#0070C0"])')

networkD3::sankeyNetwork(Links = links,
                         Nodes = nodes,
                         Source = "source",
                         Target = "target",
                         Value = "value",
                         NodeID = "name",
                         fontSize = 12,
                         nodeWidth = 30,
                         NodeGroup = "prod",
                         LinkGroup = "prod",
                         colourScale = colorJS)

```

Pie Charts

```

### Data Prep ###

# Create data cube for gender representation among active
→ employees
smmry_gender <- data.frame(gender = c('Male', 'Female'),
                             cnt = c(732, 501),
                             pct = c(59.4, 40.6))

```

```

### Pie Chart ###

# Load library
library(ggplot2)

ggplot2::ggplot(smmry_gender, aes(x = "", y = pct, fill =
  → as.factor(gender))) +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::coord_polar("y", start = 0) +

```

```
ggplot2::geom_text(aes(label = paste0(pct, "%")), color =
  ↪ ifelse(smmry_gender$gender == 'Male', 'white', 'black'),
  ↪ position = position_stack(vjust = 0.5)) +
ggplot2::labs(title = 'Nearly 60 percent of active employees
  ↪ identify as male.') +
ggplot2::scale_fill_manual(values = c("Female" = "#BFBFBF",
  "Male" = "#0070C0")) +
ggplot2::theme_void() +
ggplot2::theme(legend.title = ggplot2::element_blank(),
  panel.background = ggplot2::element_blank(),
  axis.ticks.x = ggplot2::element_blank(),
  axis.ticks.y = ggplot2::element_blank(),
  plot.title = ggplot2::element_text(colour =
  ↪ "#404040"))
```

